



**CCS6344 T2410 Assignment 2 Submission**  
**Group 11**

NUR INQSYIRA BINTI ZAMRI	1211103098
DANIEL DANISH BIN SUHAIMI	1201102370
MOHAMED ARIQUE BIN MOHAMED AZIYEN	1211306413

**Task 3**

Presentation - <https://youtu.be/G4hJKZ2Aem8?si=87hIsjRSUQINKgmU>

GitHub -

# Task 1 Implementation Design

To address the client's requirement for migrating their existing application to a cloud environment, our team recommends leveraging Amazon Web Services (AWS) modules tailored to ensure scalability, flexibility, and reliability. We propose using Relational Database Service (RDS) for this migration.

Amazon Elastic Compute Cloud (Amazon EC2) is an AWS service that provide scalable virtual services with flexible configurations and security measure features for dynamic application hosting, while RDS (Relational Database Service) offers automated maintenance, high availability and support multiple engines such as MySQL, PostgreSQL, SQL Server, and Oracle, ensuring efficient, secure and reliable cloud infrastructure.

We begin with the creation of Virtual Private Cloud (VPC) that will provide a secure and isolated network environment. We configured one public subnet and one private subnet within a single availability zone to ensure a clear separation between web and database servers

To secure our infrastructure, we established two security groups. The first, named AgentHub Web Security Group, allows HTTP access to the web servers from any IP address, ensuring the application is accessible to users. The second, named AgentHub DB Security Group, restricts database access to only the web servers within the VPC, enhancing database security. Additionally, we created extra public and private subnets in a second availability zone as this will ensure high availability and fault tolerance by distributing resources across multiple zones.

For database management, we configured an RDS instances it provide a cost-effective solution for varying workloads. The instance includes 20 GiB of General-Purpose SSD storage for balanced performance and cost-efficiency. We also created a DB subnet group named agenthub-db-subnet-group, spanning both availability zones to enhance availability and resilience. The RDS instance is publicly accessible and configured with security groups to ensure secure connections from the web servers.

Finally, to validate the setup, we tested the AWS server database connection using MySQL Workbench and the SQLTools extension in VS Code. This process involved copying the RDS endpoint value and configuring new connections in both tools to ensure proper accessibility and functionality of the RDS instance.

By leveraging AWS services such as VPC, subnets, security groups, and RDS, this implementation design ensures a secure, scalable, and highly available infrastructure. This setup aligns with our client's business requirements, providing them with the necessary tools to adapt to changing market conditions, enhance security, and achieve their business objectives in the cloud.

# Task 2 Implementation Using AWS Academy Cloud Foundations

## 2.1 ERD Diagram

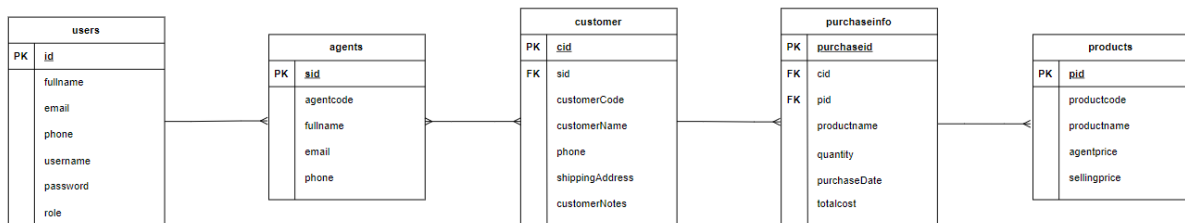


Figure 1: ERD Diagram

- Relationship between the users and agent is one-to-many.
- Relationship between the agents and customer is many-to-many.
- Relationship between the customer and purchaseinfo is one-to-many.
- Relationship between the purchaseinfo and products is one-to-many.

## 2.2 Step-By-Step Description

### 2.2.1 Access AWS Academy and Sandbox Environment

- Log in to AWS Academy and access the sandbox environment provided.
- Ensure you have the necessary permissions to create resources like RDS databases and EC2 instances.

### 2.2.2 Configure AWS RDS (Relational Database Service)

- Navigate to AWS RDS service.
- Create a new RDS instance using MySQL as the database engine.
- Configure the instance with appropriate settings such as instance type, storage, and security groups.
- Note down the endpoint value once the RDS instance is created.

### 2.2.3 Set Up MySQL Workbench

- Download and install MySQL Workbench version 8.0.25 on your local computer if not already installed.
- Launch MySQL Workbench and create a new connection:
- Hostname: Use the endpoint value of the RDS instance.
- Username: Set the username configured during RDS instance creation.
- Password: Set the password configured during RDS instance creation.
- Test the connection to ensure it's successful.

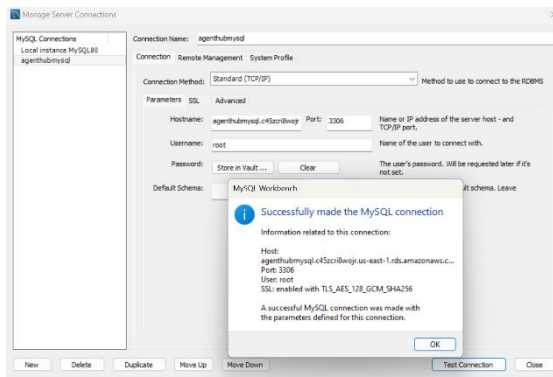


Figure 2: Connection is successful

#### 2.2.4 Update Application Code with JDBC Driver

- Download the MySQL Connector/J JDBC driver (mysql-connector-java-8.0.25.jar).
- Update your application code to include the JDBC driver and configure the database connection details:

```
src > g11 > agenthub > db_connect > DbConnection.java > DbConnection
9 public class DbConnection implements AutoCloseable {
16     public DbConnection() {
17         this.dbUrl = "jdbc:mysql://agenthubmysql.c45zcri8wojr.us-east-1.rds.amazonaws.com:3306/agenthub";
18         this.dbUser = "root";
19         this.dbPass = "Pa$$w0rd";
20         this.dbDriver = "com.mysql.cj.jdbc.Driver";
21     }
```

Figure 3: Database connection details with respect to RDS instance configuration

#### 2.2.5 Set Up VS Code with SQLTools Extension

- Install the SQLTools extension in Visual Studio Code (VS Code).
- Configure a new connection to your RDS instance in SQLTools:
- Server: Amazon AWS server (use the endpoint DNS name).
- Port: Use the port defined in your RDS security group (default is 3306 for MySQL).
- Driver: Select MySQL.
- Username: Set it to root or the username you configured.
- Password: Enable the "ask on connection" option or provide the password directly (ensure security practices).

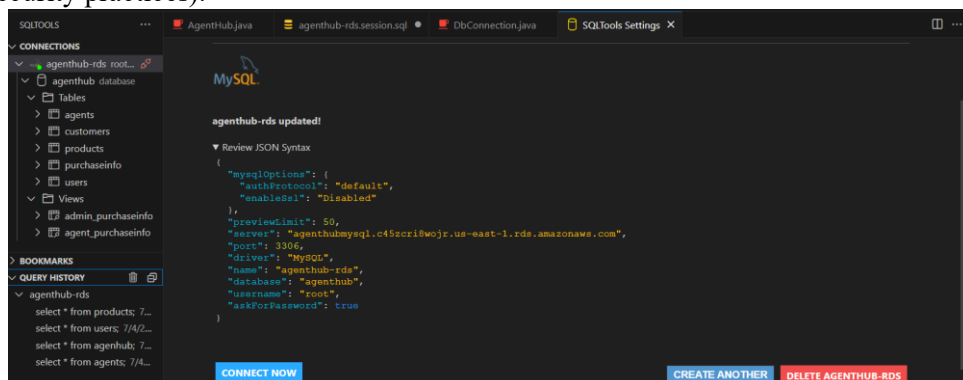
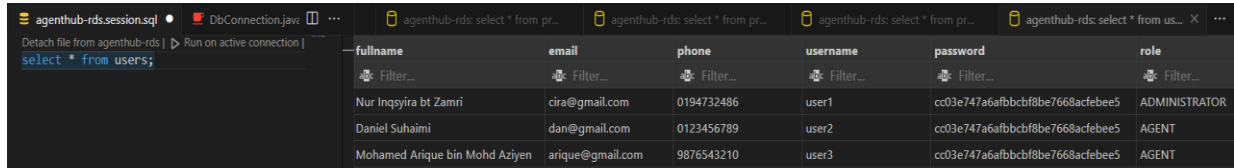


Figure 4: New connection named agenthub-rds is successfully made.

#### 2.2.6 Verify and Test

- Save the configurations and test the connection from SQLTools to ensure it connects successfully to your RDS instance.
- Test your application locally to ensure it can connect to the RDS database using the updated JDBC connection settings.



The screenshot shows the SQLTools application interface. On the left, a code editor displays the SQL query `select * from users;`. The top panel shows several tabs, including `agenthub-rds.session.sql` and `DbConnection.java`. The right panel displays the results of the query as a table with 6 columns: `fullname`, `email`, `phone`, `username`, `password`, and `role`. The table contains 3 rows of data.

fullname	email	phone	username	password	role
Nur Inqsyira bt Zamri	cira@gmail.com	0194732486	user1	cc03e747a6afbbcbf8be7668acfebee5	ADMINISTRATOR
Daniel Suhaimi	dan@gmail.com	0123456789	user2	cc03e747a6afbbcbf8be7668acfebee5	AGENT
Mohamed Arique bin Mohd Aziyen	arique@gmail.com	9876543210	user3	cc03e747a6afbbcbf8be7668acfebee5	AGENT

Figure 5: Perform simple SQL queries to test the connection.

## 2.3 Security Measures AWS

Security measures that were taken on AWS is implementing RDS at the application. Firstly, VPC configuration are done to provides a secure and isolated environment, it is to reduce the risk of unauthorised access. Next, subnets are configured into public and private to prevent from disclosure of database and sensitive components private. Other than that, Internet Gateway helps to facilitate external communication, it allows web server to serve content globally. Additionally, security groups enhance the control over traffic flow, ensuring the web server’s reliability, scalability, and security within the AWS infrastructure. Here are some detailed steps for the implementation:

### 2.3.1 VPC Configuration

#### 2.3.1.1 Create VPC

Create the VPC with specified configuration, first choose the “**VPC and more**” and provide name such as ‘**agenthub**’. Keep the IPv4 CIDR block as ‘**10.0.0.0/16**’ and tenancy as “**default**”. Set the number of **Availability Zones to one**, with one public subnet ‘**10.0.1.0/16**’ and one private subnets ‘**10.0.2.0/16**’. The NAT gateways is set to “**In 1 AZ**” and VPC endpoints are set to “**None**”. Lastly, DNS options both are **enable**. Figure 6 shows the created VPC.

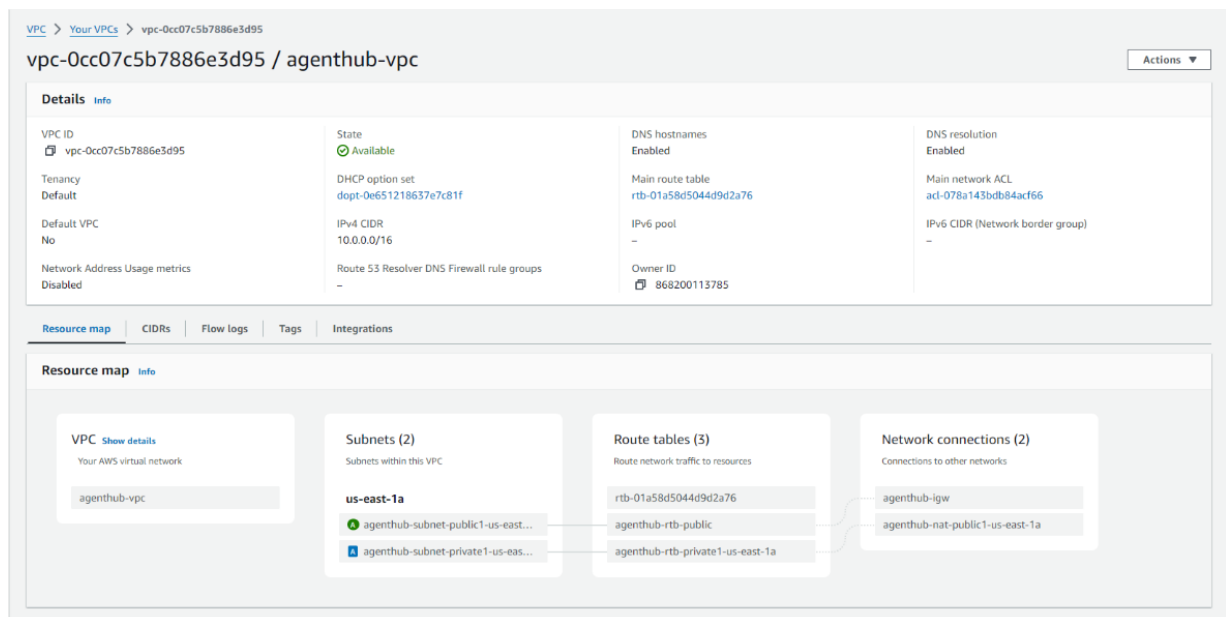


Figure 6: VPC Created

### 2.3.1.2 Configure New Subnets

Create two new subnets to ‘**agenthub-vpc**’, which is one public subnet “**10.0.3.0/24**” and one private subnet “**10.0.4.0/24**”. Go to route table and edit subnet association by adding the new public subnet into the public route table, while the private subnet into the private route table. The updated resource map will be as Figure 7.

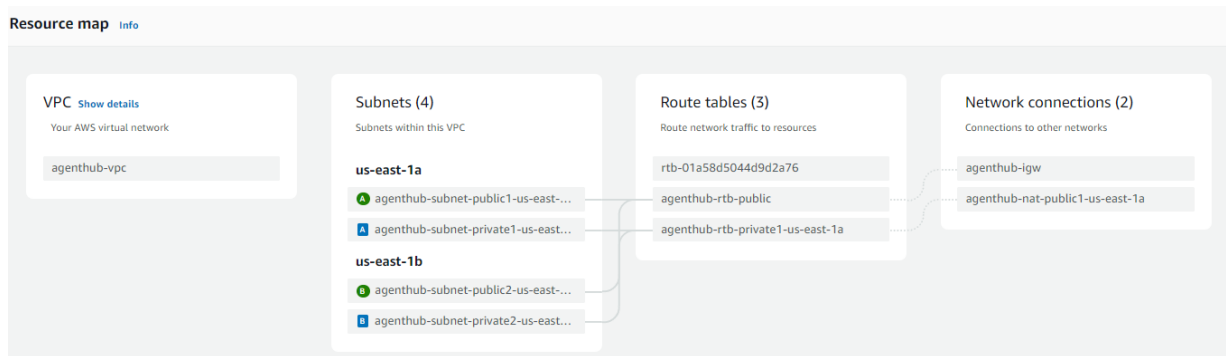


Figure 7: Updated Resource Map

### 2.3.2 Configure VPC Security Group

There are two security groups that need to be create.

#### 2.3.2.1 HTTP

Provide name such as ‘**AgentHub Web Security Group**’ and give description as **enabling HTTP access**. Set as the “**agenthub-vpc**”. Edit the inbound rules, set type as “**HTTP**”. Source will be “**Anywhere-IPv4**”, and it will be **permitting web requests**.

sg-0f377fb13df162bbe - AgentHub Web Security Group

Actions

**Details**

Security group name AgentHub Web Security Group	Security group ID sg-0f377fb13df162bbe	Description Enable HTTP access	VPC ID vpc-0cc07c5b7886e3d95
Owner 868200113785	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

Inbound rules | Outbound rules | Tags

**Inbound rules (1)**

Search

	Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
<input type="checkbox"/>	-	sg-0880859b259e29...	IPv4	HTTP	TCP	80	0.0.0.0/0	Permit web request

Manage tags Edit inbound rules

Figure 8: AgentHub Web Security Group

#### 2.3.2.2 RDS DB Instance

Provide name such as ‘**AgentHub DB Security Group**’ and give description as **permit access from Web Security Group**. Set as the “**agenthub-vpc**”. Edit the inbound rules, set type as “**MYSQL/Aurora**” and source will be **custom** “**AgentHub Web Security Group**”.

**Create security group** [info](#)

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

**Basic details**

Security group name [info](#)  
 AgentHub DB Security Group  
 Name cannot be edited after creation.

Description [info](#)  
 Permit access from Web Security Group

VPC [info](#)  
 vpc-0cc07c5b7886e3d95 (agenthub-vpc)

**Inbound rules** [info](#)

Type	Protocol	Port range	Source	Description - optional
MYSQL/Aurora	TCP	3306	Custom	sg-0f377fb13df162bbe

[Add rule](#)

**Outbound rules** [info](#)

Figure 9: Creation of AgentHub DB Security Group

## 2.3.3 Configuring Amazon RDS

### 2.3.3.1 Create DB Subnet Group

Create a DB Subnet Group named ‘**agenthub-db-subnet-group**’ and choose the ‘**agenthub-vpc**’ as VPC. Availability Zones that were chosen is “us-east-1a” and “us-east-1b”. The subnet chosen are the public subnets from each zone which is “10.0.1.0/24” and “10.0.3.0/24”.

**agenthub-db-subnet-group**

**Subnet group details**

VPC ID  
 vpc-0cc07c5b7886e3d95 [link](#)

ARN  
 arn:aws:rds:us-east-1:868200113785:subgrp:agenthub-db-subnet-group

Supported network types  
 IPv4

Description  
 AgentHub DB Subnet Group

**Subnets (2)**

Availability zone	Subnet ID	CIDR block
us-east-1a	subnet-0eb22bd4b3b6c059f <a href="#">link</a>	10.0.1.0/24
us-east-1b	subnet-08da157a76ea3c5aa <a href="#">link</a>	10.0.3.0/24

Figure 10: DB Subnet Group

### 2.3.3.1 Create Database



Create the DB by selecting “**standard create**” and proceed with selecting the engine type, edition, and version, in this case is “**MySQL**”, “**MySQL Community**”, and “**MySQL 8.0.35**”. Pick the “**Free Tier**” for template and continue with fill in settings.

DB instance is ‘**agenthub-db-01**’, username is ‘**root**’ and **self manage** password is ‘**Pa\$\$w0rd**’. The instance configuration selected is “**db.t3.micro**”. Storage type is “**gp2**” with allocation storage of ‘**20**’ GiB, the autoscaling are **disable**.

For the connectivity, compute resource are set to “**Don’t connect to an EC2 compute resource**”. The “**agenthub-vpc**” are selected as VPC and “**agenthub-db-subnet group**” are used. It is a **public access** and the VPC security group are the existing “**AgentHub DB Security Group**” and “**AgentHub Web Security Group**”. **No preference** for the availability zone.

Additional configuration also needs to be setup. Such as name is “**agenthub**” or it will not create a database. All the log exports are selected, “**Audit log**”, “**Error log**”, “**General log**” and “**Slow query log**”. The created database will be like Figure 1.

RDS > Databases > agenthub-db-01

agenthub-db-01

Summary

DB identifier  
agenthub-db-01

CPU  
4.07%

Status  
Available

Class  
db.t3.micro

Role  
Instance

Current activity  
0 Connections

Engine  
MySQL Community

Region & AZ  
us-east-1b

Recommendations

Connectivity & security

Monitoring

Logs & events

Configuration

Zero-ETL integrations

Maintenance & backups

Tags

Recommendations

Connectivity & security

Endpoint & port

Endpoint  
agenthub-db-01.c68gdcultjd2.us-east-1.rds.amazonaws.com

Port  
3306

Networking

Availability Zone  
us-east-1b

VPC  
agenthub-vpc (vpc-0cc07c5b7886e3d95)

Subnet group  
agenthub-db-subnet-group

Subnets  
subnet-0eb22bd4b3b6c059f  
subnet-06da157a76ea3c5aa

Network type  
IPv4

Security

VPC security groups  
AgentHub DB Security Group (sg-0161a1eec0fc2566b)  
Active  
AgentHub Web Security Group (sg-0f377fb13df162bbe)  
Active

Publicly accessible  
Yes

Certificate authority  
rds-ca-rsa2048-g1  
Certificate authority date  
May 26, 2061, 07:34 (UTC+08:00)  
DB instance certificate expiration date  
July 04, 2025, 22:27 (UTC+08:00)

Figure 11: Database Created

## 2.4 Application Testing

To ensure the functionality and integration of the AgentHub application with AWS RDS, testing is conducted using a local setup with VS Code and MySQLTools extension. This process involves applying add/delete functions within the AgentHub application locally via VS Code and verifying results using MySQLTools to query the AWS RDS database.

Here is the snapshot of the original data for your convenient:

pid	productcode	productname	agentprice	sellingprice
Filter...	Filter...	Filter...	Filter...	Filter...
1	prod1	The City Works Tokyo Notebook	79	95
2	prod2	The City Works Melbourne Notebook	89	109
3	prod3	The City Works Malaysia Notebook	99	119
4	prod4	Lico Notebook in Terracotta	95	115
5	prod5	Lico Notebook in Brown	105	125
9	prod6	Lico Notebook in White	115	135

Figure 12: Product table before performing the application testing

### 2.4.1 Insert New Entry

The screenshot shows the 'Product Management' application window. It features a search bar at the top with the text 'Search by Product Name / Product Code' and a 'Search' button. Below the search bar is a table with columns: productcode, productname, agentprice, and sellingprice. The table lists six products: prod1 (The City Works Tokyo Notebook, 79.0, 95.0), prod2 (The City Works Melbourne Notebook, 89.0, 109.0), prod3 (The City Works Malaysia Notebook, 99.0, 119.0), prod4 (Lico Notebook in Terracotta, 95.0, 115.0), prod5 (Lico Notebook in Brown, 105.0, 125.0), and prod6 (Lico Notebook in White, 115.0, 135.0). To the right of the table are input fields for 'Product Name:', 'Agent Price:', and 'Selling Price:', each with a corresponding button ('Add Product', 'Edit Product', 'Delete Product'). At the bottom right are 'Clear' and 'Refresh' buttons. A 'Success' dialog box is open in the center, displaying a blue checkmark icon and the text 'Product Added Successfully' with an 'OK' button.

Figure 13: Add a product named "Before Breakfast Collection" .

pid	productcode	productname	agentprice	sellingprice
Filter...	Filter...	Filter...	Filter...	Filter...
1	prod1	The City Works Tokyo Notebook	79	95
2	prod2	The City Works Melbourne Notebook	89	109
3	prod3	The City Works Malaysia Notebook	99	119
4	prod4	Lico Notebook in Terracotta	95	115
5	prod5	Lico Notebook in Brown	105	125
9	prod6	Lico Notebook in White	115	135
11	prod7	Before Breakfast Collection	45	63

Figure 14: Verify the product is successfully added in the database.

## 2.4.2 Delete Old Entries

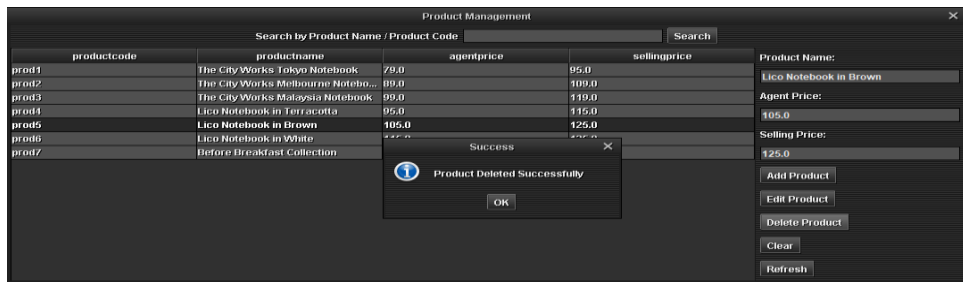


Figure 15: Delete a product with the productcode of "prod5".

pid	productcode	productname	agentprice	sellingprice
Filter...	Filter...	Filter...	Filter...	Filter...
1	prod1	The City Works Tokyo Notebook	79	95
2	prod2	The City Works Melbourne Notebook	89	109
3	prod3	The City Works Malaysia Notebook	99	119
4	prod4	Lico Notebook in Terracotta	95	115
9	prod6	Lico Notebook in White	115	135
11	prod7	Before Breakfast Collection	45	63

Figure 16: Verify the product is successfully deleted in the database.

## 2.4.3 Insert Another New Entry

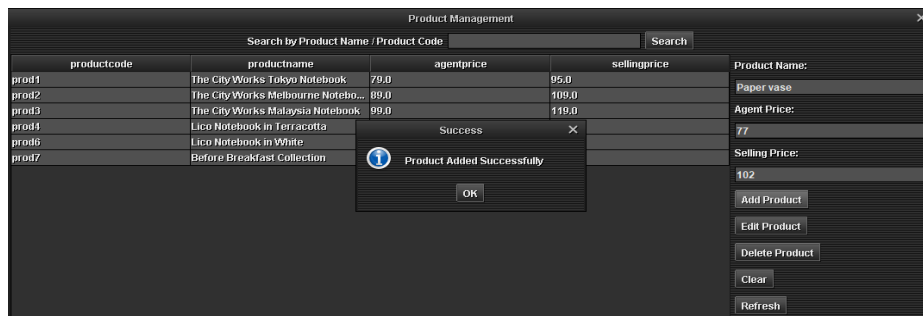


Figure 17: Add another new product named "Paper vase".

pid	productcode	productname	agentprice	sellingprice
Filter...	Filter...	Filter...	Filter...	Filter...
1	prod1	The City Works Tokyo Notebook	79	95
2	prod2	The City Works Melbourne Notebook	89	109
3	prod3	The City Works Malaysia Notebook	99	119
4	prod4	Lico Notebook in Terracotta	95	115
9	prod6	Lico Notebook in White	115	135
11	prod7	Before Breakfast Collection	45	63
12	prod8	Paper vase	77	102

Figure 18: Verify the product is successfully added in the database.