CCS6344 DATABASE AND CLOUD SECURITY

Term 2430

Assignment 2 Submission

Group Name: Group 47

Name	Student ID	Contribution
Nur Farah Nabila Binti Ramzairi	1221301140	50%
Muhammad Aniq Fahmi Bin Azhar	1211101533	50%

GitHub Link	https://github.com/aniqfahmi03/CCS6344-Assignment-2
Slide Link	https://www.canva.com/design/DAGerRok0_s/CFA6ADebNFzkXv6Ygt0weg/edit?utm_content=DAGerRok0_s&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton
Youtube Link	https://youtu.be/pGzWbxV-8vw?si=LAzgkGW22aL02Qkz

Task 1 Implementation Design

To meet the client's requirements for migrating their existing application to the cloud, our team recommends leveraging Amazon Web Services (AWS) to ensure scalability, flexibility, and reliability. Specifically, we propose utilizing **Amazon Relational Database Service (RDS)** to facilitate a seamless transition while maintaining performance and security.

For dynamic application hosting, **Amazon Elastic Compute Cloud (EC2)** serves as a robust solution, providing scalable virtual servers with customizable configurations and built-in security features. Meanwhile, **RDS** supports multiple database engines—including MySQL, PostgreSQL, SQL Server, and Oracle—offering automated maintenance, high availability, and enhanced security for efficient cloud infrastructure management.

Our migration strategy begins with setting up a **Virtual Private Cloud (VPC)** to establish a secure, isolated network environment. Within this VPC, we implemented a **public subnet** for web-facing components and a **private subnet** dedicated to database resources, ensuring a clear separation between the application and data layers.

To enhance security, we configured two distinct **security groups**:

- **AgentHub Web Security Group** – Permits HTTP access to the web servers from any IP address, ensuring application accessibility.
- **AgentHub DB Security Group** – Restricts database access exclusively to web servers within the VPC, preventing unauthorized access and strengthening security.

To achieve **high availability and fault tolerance**, we extended our architecture to a second availability zone, introducing additional **public and private subnets**. This approach enables **resource distribution across multiple zones**, reducing downtime and ensuring continuous operation in case of failures.

For **database management**, we provisioned an **Amazon RDS instance**, offering a cost-effective solution adaptable to varying workloads. The instance is configured with **20 GiB of General-Purpose SSD storage**, striking a balance between performance and affordability. Additionally, we created a **DB subnet group (agenthub-db-subnet-group)** spanning both availability zones, further improving resilience and availability. The RDS instance is publicly accessible but securely configured to allow only authorized connections from the designated web servers.

To validate our deployment, we conducted connectivity tests using **MySQL Workbench** and the **SQLTools extension in VS Code**. By retrieving the RDS endpoint and establishing new connections within both tools, we ensured that the database is accessible and fully functional.

By integrating AWS services such as **VPC, subnets, security groups, and RDS**, this infrastructure design offers a **secure, scalable, and highly available cloud environment**. The solution aligns with the client's business needs, enabling them to adapt to evolving demands, fortify security, and optimize operations for sustained growth in the cloud.

Task 2 Implementation Using AWS Academy Cloud Foundations

2.1 ERD Diagram

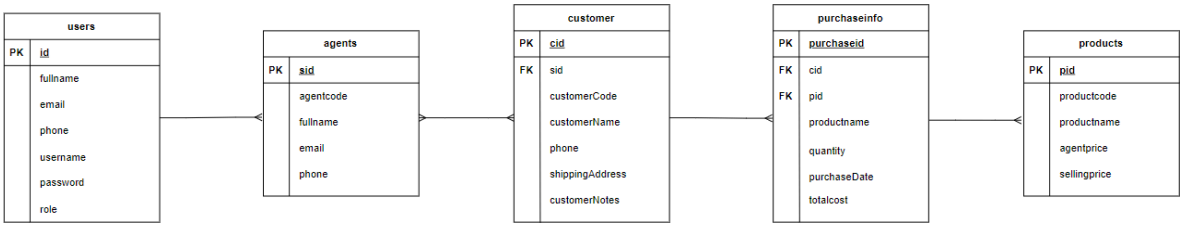

Figure 1: ERD Diagram

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

2.2 Step-by-Step Guide: Setting Up AWS RDS and Configuring Database connectivity

Step 1: Access AWS Academy and Sandbox Environment

- Sign in to AWS Academy and navigate to the sandbox environment provided for cloud-based learning.
- Verify that your account has the required permissions to create and manage AWS resources such as RDS databases and EC2 instances.

Step 2: Configure Amazon RDS (Relational Database Service)

- Open the AWS Management Console and go to the RDS service.
- Click on "Create Database", then select MySQL as the preferred database engine.
- Adjust configuration settings, including:
    - Instance type based on processing and memory requirements.
    - Storage allocation (recommended: 20 GiB General-Purpose SSD for efficiency).
    - Security groups to allow authorized database connections.
- Once the instance is successfully deployed, copy and store the RDS endpoint, as it will be needed for future configurations.

Step 3: Set Up MySQL Workbench

- Download and install MySQL Workbench 8.0.25 on your system (if not already installed).
- Launch MySQL Workbench and create a new connection:
    - Hostname: Enter the RDS endpoint retrieved earlier.
    - Username: Use the database username specified during RDS setup.
    - Password: Enter the corresponding password configured at setup.
- Click Test Connection to confirm successful database access.

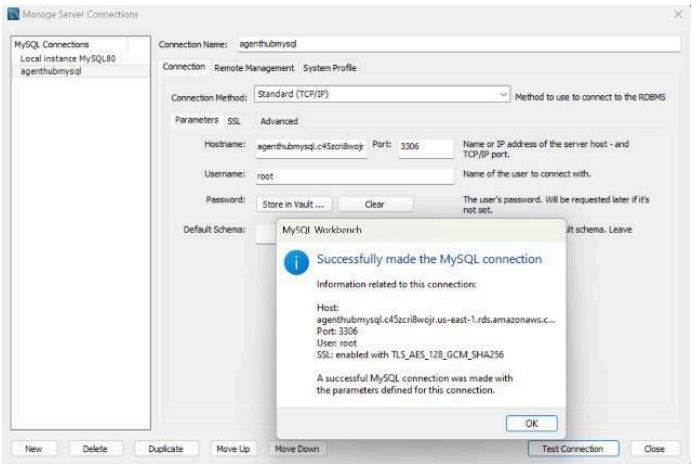

Figure 2: Connection is successful

Step 4: Integrate Application Code with JDBC Driver

- Obtain the **MySQL Connector/J JDBC driver** (mysql-connector-java-8.0.25.jar) and include it in your project dependencies.
- Modify your **application's database connection settings**:
    - Define the **JDBC connection URL** using the **RDS endpoint**.
    - Set the **username** and **password** for authentication.
    - Implement **error handling** to ensure stable connectivity.

```
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

Step 5: Configure SQLTools Extension in VS Code

- Open **VS Code** and install the **SQLTools extension** from the extensions marketplace.
- Set up a **new connection** for the AWS RDS instance:
    - **Server**: Enter the **AWS RDS endpoint (DNS name)**.
    - **Port**: Use **3306**, the default for MySQL.
    - **Driver**: Select **MySQL**.
    - **Username**: Use the **configured database user**.
    - **Password**: Choose **"ask on connection"** for security or enter it directly.
- Save the configuration.

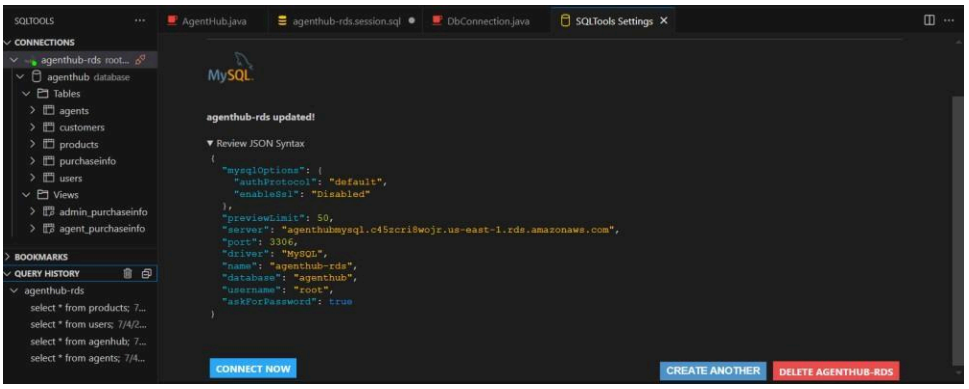

Figure 4: New connection named agenthub-rds is successfully made.

Step 6: Verify and Validate Connectivity

- Run a **test connection** in SQLTools to ensure proper communication with the database.
- Launch your **application locally** and validate that it successfully connects to the **AWS RDS instance** using the **JDBC configuration**

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

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.

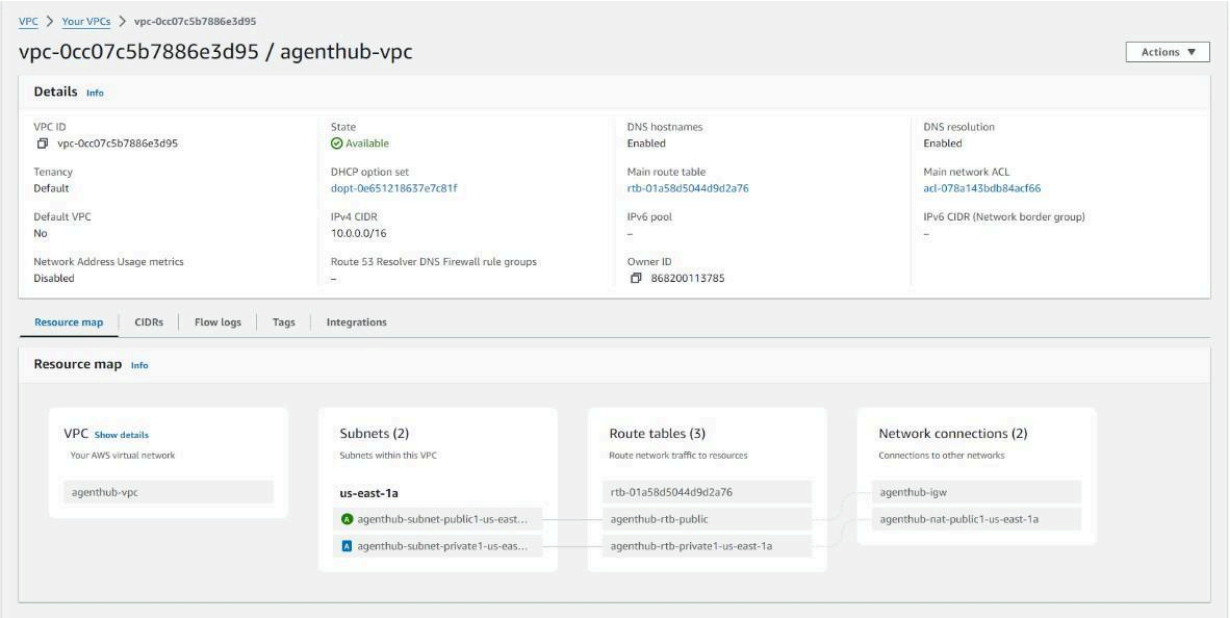

Figure 5: VPC Created

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 6: Updated Resource Map

2.3.2 Configure VPC Security Group

There are two security groups that need to be create.

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**.

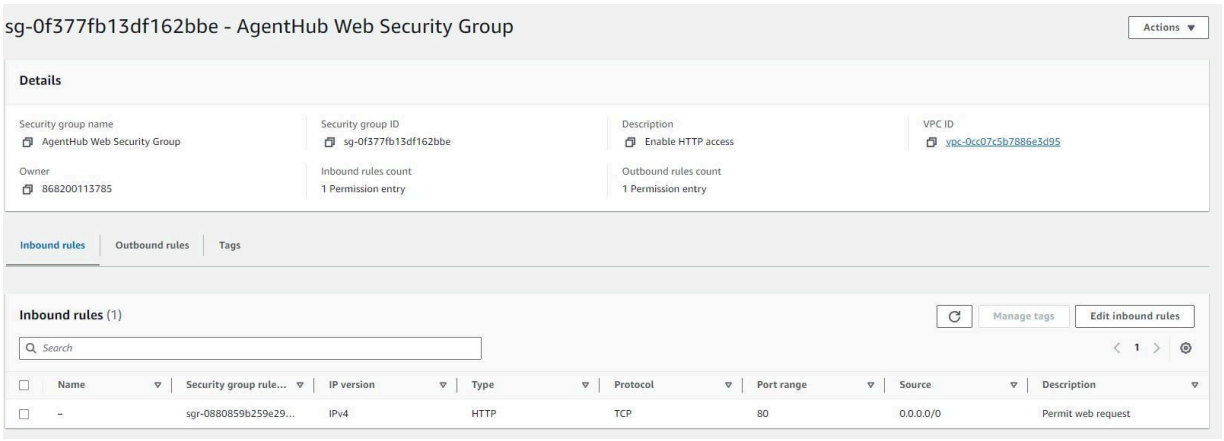

Figure 7: AgentHub Web Security Group

2. **RDS DB Instance**

Provide a 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**".

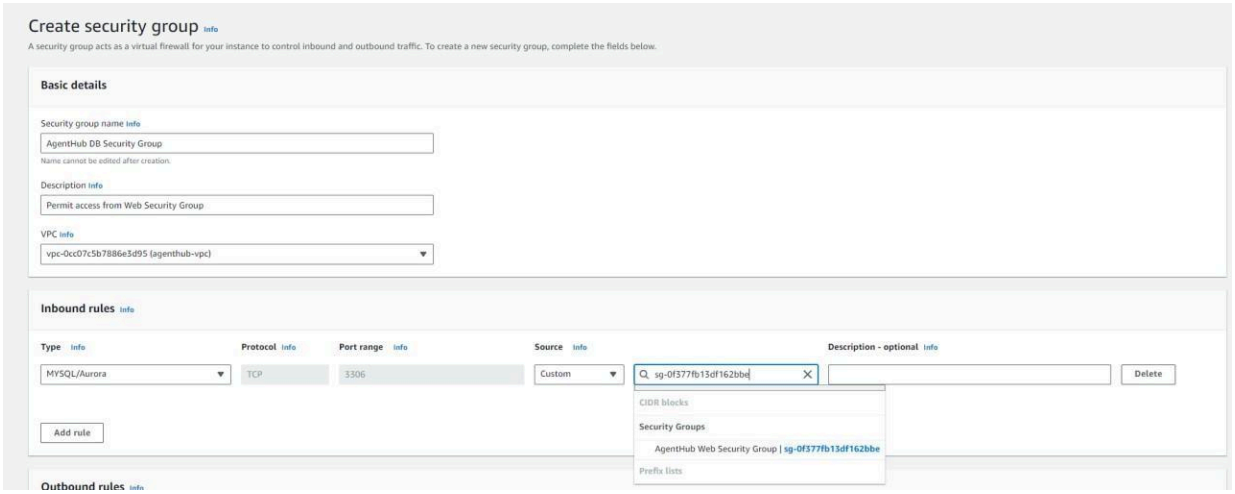

Figure 8: Creation of AgentHub DB Security Group

2.3.3 Configuring Amazon RDS

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".

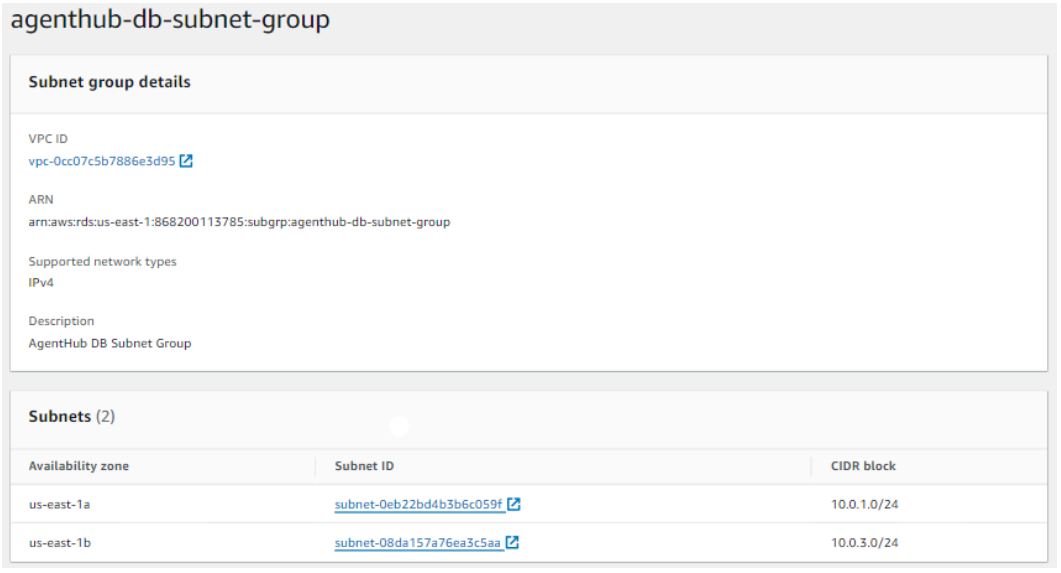

Figure 9: DB Subnet Group

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.

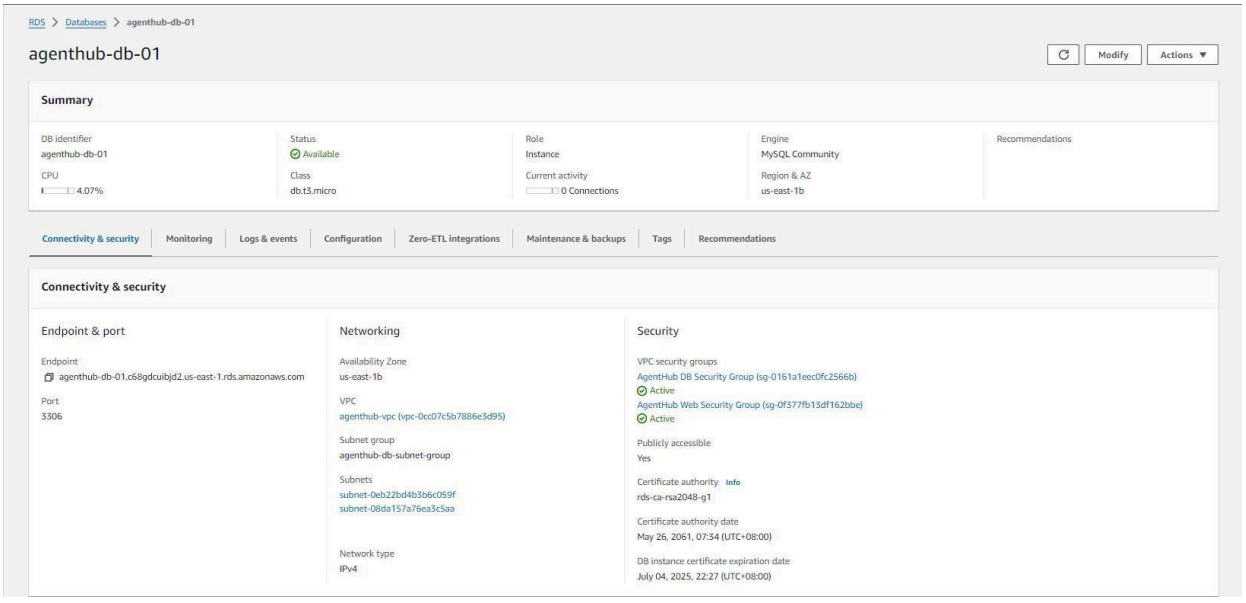

Figure 10: Database Created

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 11: Product table before performing the application testing

Insert New Entry

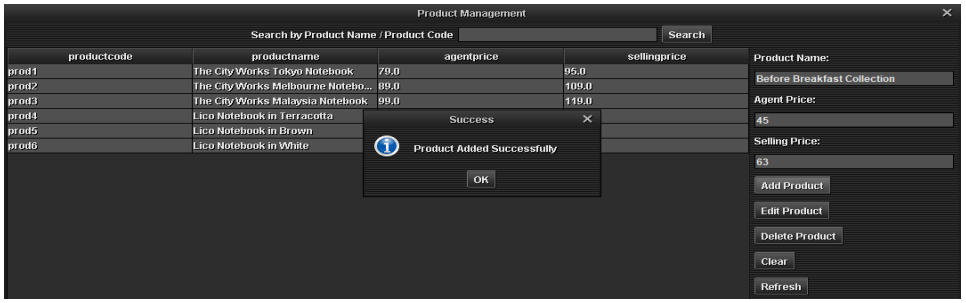

Figure 12: Add a product named ”Before Breakfast Collection” .

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 13: Verify the product is successfully added in the database.

Delete Old Entries

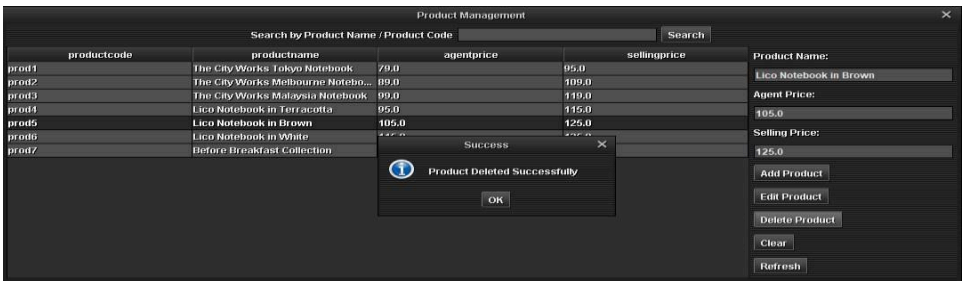

Figure 14: Delete a product with the productcode of ”prod5”.

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 15: Verify the product is successfully deleted in the database.

Insert Another New Entry

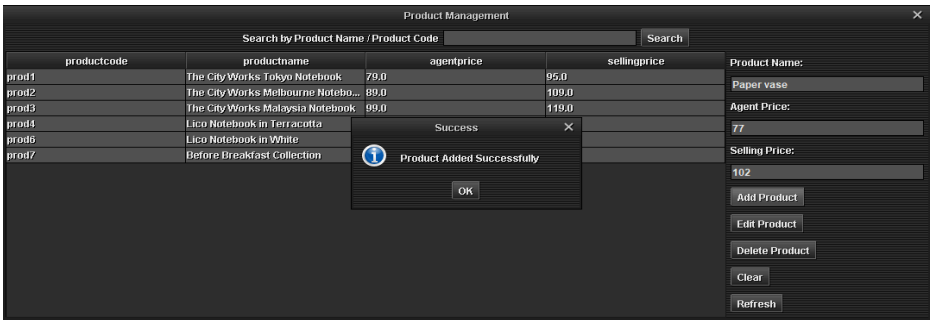

Figure 16: Add another new product named ”Paper vase” .

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 17: Verify the product is successfully added in the database