VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**SOFTWARE ENGINEERING FINAL PROJECT**

**GOODS DISTRIBUTION MANAGEMENT SYSTEM**

*Instructor*: **Mr. Phạm Thái Kỳ Trung**

*Students*: **PHAN HỒ TUẤN KIỆT – 519H0025**

**NGUYỄN VÕ HOÀNG – 519H0106**

**HO CHI MINH CITY, JANUARY 2022**

**PROJECT COMPLETED AT TON DUC THANG UNIVERSITY**

I hereby declare that this is my own research work and is under the scientific guidance of Mr. Pham Thai Ky Trung; The research contents and results in this topic are honest and have not been published in any form before. The data in the tables for analysis, comments and evaluation are collected by the author himself from different sources, clearly stated in the reference section.

In addition, the thesis also uses some comments, assessments as well as data of other authors, other agencies and organizations, all with citations and source annotations.

**If I find any fraud, I will take full responsibility for the content of my thesis.** Ton Duc Thang University is not related to copyright and copyright violations caused by me during the implementation process (if any).

*Ho Chi Minh City, 5th January, 2022*

*Author*

*Phan Hồ Tuấn Kiệt*

*Nguyễn Võ Hoàng*

**TABLE OF CONTENTS**

[**I.** **INTRODUCTION** 1](#_Toc92319153)

[**1.1.** **Purpose and scope:** 1](#_Toc92319154)

[**1.2.** **Product Overview:** 1](#_Toc92319155)

[**1.3.** **Structure of the Document:** 1](#_Toc92319156)

[**1.4.** **Terms, Acronyms and Abbreviations:** 2](#_Toc92319157)

[**II.** **PROJECT MANAGEMENT PLAN:** 2](#_Toc92319158)

[**2.1.** **Project Organization:** 2](#_Toc92319159)

[**2.2.** **Lifecycle Model Used:** 3](#_Toc92319160)

[**2.3.** **Risk Analysis:** 4](#_Toc92319161)

[**2.4.** **Hardware and Software Resource Requirements:** 5](#_Toc92319162)

[**2.5.** **Deliverables and Schedule:** 5](#_Toc92319163)

[**2.6.** **Professional Standards:** 6](#_Toc92319164)

[**2.7.** **Impact of the project on individuals and organizations:** 6](#_Toc92319165)

[**III.** **REQUIREMENT SPECIFICATIONS:** 7](#_Toc92319166)

[**3.1.** **Stakeholders for the system:** 7](#_Toc92319167)

[**3.2.** **Use case model :** 7](#_Toc92319168)

[***3.2.1.*** ***Graphical use case model:*** 7](#_Toc92319169)

[***3.2.2.*** ***Textual Description for each use case:*** 7](#_Toc92319170)

[**3.3.** **Functional requirements:** 12](#_Toc92319171)

[**3.4.** **Non-functional requirements:** 12](#_Toc92319172)

[**IV.** **ARCHITECTURE:** 13](#_Toc92319173)

[**4.1.** **Architectural style used:** 13](#_Toc92319174)

[**4.2.** **Architectural model:** 13](#_Toc92319175)

[**4.3.** **Technology, software and hardware used:** 14](#_Toc92319176)

[**4.4.** **Rationale for your architectural style and model:** 15](#_Toc92319177)

[**V.** **DESIGN** 16](#_Toc92319178)

[**5.1.** **Database design** 16](#_Toc92319179)

[**5.2.** **Static model – class diagrams:** 16](#_Toc92319180)

[**5.3.** **Dynamic model – sequence diagrams:** 17](#_Toc92319181)

[**5.3.1.** **Sequence diagram: Add product** 17](#_Toc92319182)

[**5.3.2.** **Sequence diagram: Delete product** 18](#_Toc92319183)

[**5.3.3.** **Sequence diagram: Add Goods Received** 18](#_Toc92319184)

[**5.3.4.** **Sequence diagram: Delete Goods Received** 19](#_Toc92319185)

[**5.3.5.** **Sequence diagram: Create Goods Delivery Note** 19](#_Toc92319186)

[**5.3.6.** **Sequence diagram: Delete Goods Delivery Note** 20](#_Toc92319187)

[**5.3.7.** **Sequence diagram: Update Goods Delivery Note** 20](#_Toc92319188)

[**5.3.8.** **Sequence diagram: Print Delivery Slip** 21](#_Toc92319189)

[**5.3.9.** **Sequence diagram: Order Payment** 21](#_Toc92319190)

[**5.4.** **GUI design:** 22](#_Toc92319191)

[***5.4.1.*** ***GUI: Products Management:*** 22](#_Toc92319192)

[**5.4.2.** ***GUI: Revenue Report*:** 22](#_Toc92319193)

[***5.4.3.*** ***GUI: Stock Report:*** 23](#_Toc92319194)

[***5.4.4.*** ***GUI: Goods Received:*** 23](#_Toc92319195)

[***5.4.5.*** ***GUI: Goods Delivery Note:*** 24](#_Toc92319196)

[**VI.** **TEST PLAN** 24](#_Toc92319197)

[**6.1.** **Requirements/specifications-based system level test cases** 24](#_Toc92319198)

[**6.2.** **Traceability of test cases to use cases** 24](#_Toc92319199)

[***6.2.1.*** ***Test case: Create Goods Received*** 24](#_Toc92319200)

[***6.2.2.*** ***Test case: Create Goods Delivery Note*** 25](#_Toc92319201)

[***6.2.3.*** ***Test case: View Stock Report*** 25](#_Toc92319202)

[**6.3.** **Techniques used for test generation** 25](#_Toc92319203)

[**6.4.** **Assessment of the goodness of your testsuite** 26](#_Toc92319204)

[**VII.** **DEMO:** 26](#_Toc92319205)

[**7.1.** **Database:** 26](#_Toc92319206)

[**7.2.** **Source code:** 27](#_Toc92319207)

[**REFERENCES** 29](#_Toc92319208)

1. **INTRODUCTION**
   1. **Purpose and scope:**

In any business agent, information storage and management is essential. However, the difficulty of paperwork is the requirement of manpower, time-consuming and risky. So there is a need of a computer application to solve this problem, or at least, to make it more convenient. Due to this issue, we have planned and implemented a project to realize this idea.

* 1. **Product Overview:**

This product was created with functions to simplify the goods statistic, as well as import and export data storage. It provides an interface allowing accountant to create and print out import/export receipts. Besides, accountant can also check again the stored data and the revenue of each month. For agents, we also made them a web platform allowing them to order goods, as well as choosing payment method for it.

* 1. **Structure of the Document:**

|  |  |
| --- | --- |
| **I. Introduction** | 1.1. Purpose and Scope  1.2. Product Overview  1.3. Structure of the Document  1.4. Terms, Acronyms, and Abbreviations |
| **II. Project Management Plan** | 2.1. Project Organization  2.2. Lifecycle Model Used  2.3. Risk Analysis  2.4. Hardware and Software Resource Requirements  2.5. Deliverables and Schedule  2.6. Professional Standards  2.7. Impact of the project on individuals and organizations |
| **III. Requirement Specifications** | 3.1. Stakeholders for the system  3.2. Use case model  3.2.1. Graphical use case model  3.2.2. Textual Description for each use case  3.3. Functional requirements  3.4. Non-functional requirement |
| **IV. Architecture** | 4.1. Architectural style(s) used  4.2. Architectural model  4.3. Technology, software, and hardware used  4.4. Rationale for your architectural style and mode |
| **V. Design** | 5.1. Database design  5.2. Static model – class diagrams  5.3. Dynamic model – sequence diagrams  5.4. GUI Design |
| **VI. Test Plan** | 6.1. Requirements/specifications-based system level test cases  6.2. Traceability of test cases to use cases  6.3. Techniques used for test generation  6.4. Assessment of the goodness of your testsuite |
| **VII. Demo** | 7.1. Database  7.2. Source code  7.3. Testing |

* 1. **Terms, Acronyms and Abbreviations:**

MVC: Model – View – Controller

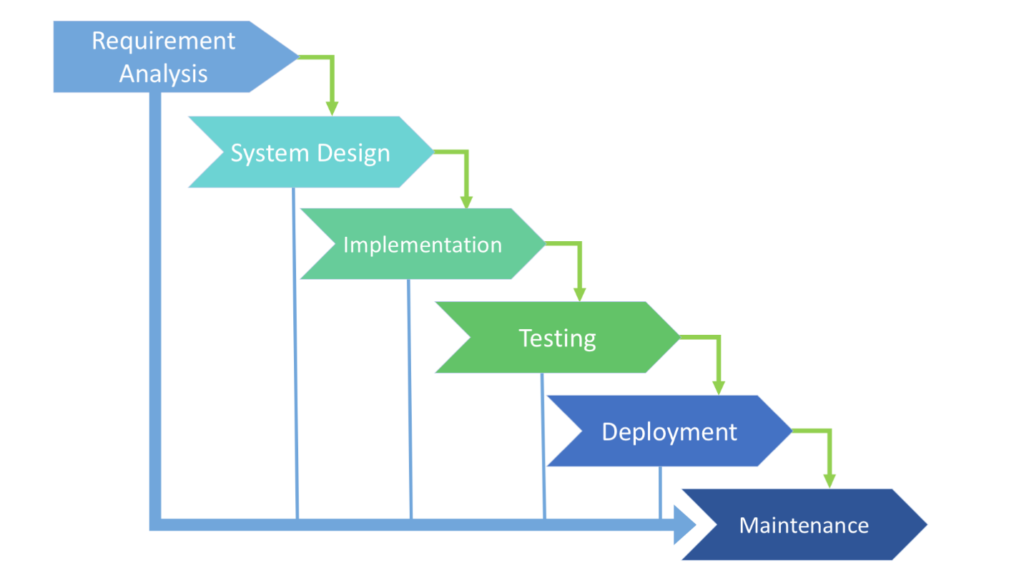
1. **PROJECT MANAGEMENT PLAN:**
   1. **Project Organization:**

Our team includes 2 members:

* 519H0025 – Phan Hồ Tuấn Kiệt
* 519H0106 – Nguyễn Võ Hoàng

We are studying Software Engineering in the same class, so we decided to make a team to cooperate in this final project.

* 1. **Lifecycle Model Used:**



There are separate identified phases in the waterfall model:

- Requirements analysis and definition

- System and software design

- Implementation and unit testing

- Integration and system testing

- Operation and maintenance

The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.

Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements. Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.

However, in the context of a final project, there would be no customer requirements. Also, we have planned every steps carefully, there might be no change during the process. Finally, we decided to choose this Lifecycle Model.

* 1. **Risk Analysis:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Risk** | **Likelihood of the risk occuring** | **Impact if the risk occurs** | **Proposed reduction strategies** |
| 1 | Project purpose and need is not well-defined. | Medium | High | Planning carefully and ensure purpose is well-defined |
| 2 | Project schedule is not clearly defined or understood | Low | Medium | Explain with the team so they can understand the plan and likelihood of missed tasks is reduced. |
| 3 | Lack of communication | Medium | Medium | Regularly check the project progress with partner, hold a meeting if needed |
| 4 | Inefficient division of work | Low | Medium | Write down all tasks and the time needed to complete the task, then assign to members equally. |
| 5 | The project occurs errors | High | High | Propose as many test cases as possible to ensure there are no bugs/errors. |
| 6 | The project cannot be done on time | Medium | High | Try to complete any possible tasks, then pass the minor ones. |

* 1. **Hardware and Software Resource Requirements:**

**Software:**

* .NET Framework 4.5.2 or above
* Internet Explorer 11 or Microsoft Edge for internet-related scenarios

**Hardware:**

* 1.8 GHz or faster processor. Quad-core or better recommended
* 2 GB of RAM; 8 GB of RAM recommended (2.5 GB minimum if running on a virtual machine)
* Hard disk space: Minimum of 800MB up to 210 GB of available space, depending on features installed; typical installations require 20-50 GB of free space.
* Video card that supports a minimum display resolution of 720p (1280 by 720);
  1. **Deliverables and Schedule:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Activity** | **Time** | **Description** | **Dependenciy** | **Allocation** |
| Planning | 1 day | - Conduct preliminary analysis  - Discover the aims and issues  - Propose a set of solutions | Needed for requirement analysis | Teamwork |
| Requirements analysis | 2 days | - Analyze the requirements  - Find related documents | Needed for design | Tuấn Kiệt |
| Design | 2 days | - Choose system architecture  - Design database  - Design UML diagrams | Needed for implementation | Võ Hoàng |
| Implementation | 5 days | - Write code for the product | The most important step, must be done for testing | Teamwork |
| Testing | 1 day | - Test functions to meet the requirements  - Check bugs in code | Needed to make sure the system works | Tuấn Kiệt |
| Write a report | 2 days |  | Required | Tuấn Kiệt |
| Demo | 1 day | - Record a demo | Required | Võ Hoàng |

* 1. **Professional Standards:**

We made up a good team and exhibit cooperative behavior. There is no disrespectful or dishonest behavior throughout the project making process.

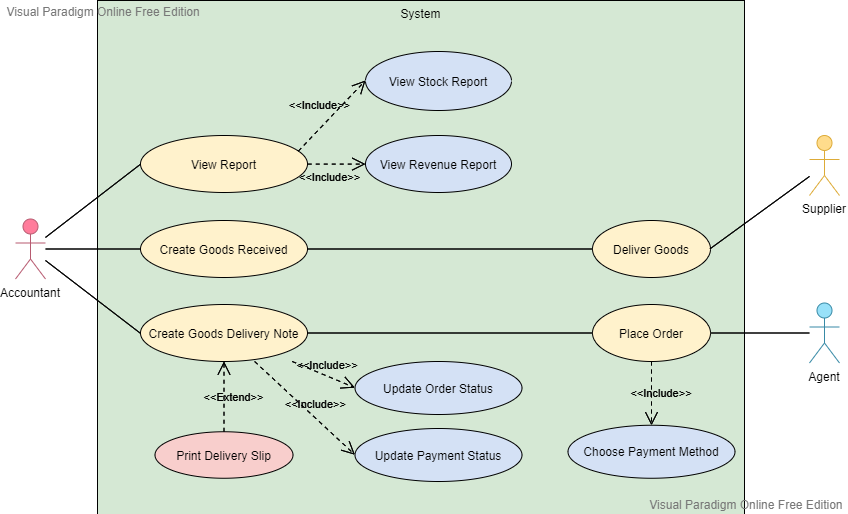
* 1. **Impact of the project on individuals and organizations:**

**On individuals:** The accountants will take fewer steps and the data will be stored in the database instead of papers, which is an improvement over the usual paperwork. So, the accountants can save their effort and time, as well as reduce the risk of making mistakes.

**On organizations:** This project will help companies to simplify the management and storage of the import and export processes for agents. Thereby, it saves companies a cost of paying for human resources and avoids data loss. Agents can now order from companies and make payment in a faster and simpler way. Via our web platform, they can view the status of available goods, select items and make payment by choosing several methods of payment.

1. **REQUIREMENT SPECIFICATIONS:**
   1. **Stakeholders for the system:**

* Agent
* Supplier
* Company’s accountant
* Project development team
  1. **Use case model :**
     1. ***Graphical use case model:***

****

* + 1. ***Textual Description for each use case:***

#### i. Use case: Place Order fully description:

|  |  |  |
| --- | --- | --- |
| **Use case name** | Place Order | |
| **Scenario** | Agent needs a new shipment | |
| **Triggering event** | Agent proceeds to place an order on website | |
| **Brief description** | Agent selects items and adds to cart, chooses payment method, then makes payment. | |
| **Actors** | Agent | |
| **Related use case** | Create Goods Delivery Note | |
| **Stakeholders** | Accountant | |
| **Preconditions** | The payment method must be available  The items must be in stocks | |
| **Postconditions** | A goods delivery note must be created  Payment status must be updated  Order status must be updated | |
| **Flow of activities** | **Actor** | **System** |
| 1. Agent selects to view item.  2. Agent adds item to cart.  3. Agent finalizes the order.  4. Agent fills in the form, selects method of payment then confirms the order. | 1.1. Display information of item  2.1. Add item to cart  3.1. Display client information form  3.2. Display payment methods  4.1. Save the order into database. |
| **Exception Conditions** | 1.1. Item is not available  4.1. Cannot connect to database | |

#### ii. Use case: Create Goods Received fully description:

|  |  |  |
| --- | --- | --- |
| **Use case name** | Create Goods Received | |
| **Scenario** | Accountant needs to create a good received for company | |
| **Triggering event** | The company imports goods | |
| **Brief description** | When the company imports a new shipment, the acccountant creates a good received | |
| **Actors** | Accountant | |
| **Related use case** |  | |
| **Stakeholders** | Accountant, Supplier, Company | |
| **Preconditions** | Accountant must be authorized to access the system  The supplier has contacted the company  The goods must be intact  The goods need to be checked before entering the warehouse | |
| **Postconditions** | A Goods Received is created and added into database  The goods enter the warehouse safely | |
| **Flow of activities** | **Actor** | **System** |
| 1. Accountant opens Goods Received form  2. Accountant fills out information of the shipment, then saves. | 1.1. Display Goods Received form  2.1. Save the shipment information into database |
| **Exception Conditions** | 2.1. Cannot connect to database | |

#### iii. Use case: Create Good Delivery Note fully description:

|  |  |  |
| --- | --- | --- |
| **Use case name** | Create Good Delivery Note | |
| **Scenario** | Accountant needs to create a Good Delivery Note for agent | |
| **Triggering event** | Agent places a new order | |
| **Brief description** | After the agent places an order, the accountant create a Good Delivery Note to deliver goods to them. | |
| **Actors** | Accountant | |
| **Related use case** | Place Order | |
| **Stakeholders** | Accountant, Agent, Company | |
| **Preconditions** | Accountant must be authorized to access the system  Agent successfully places an order  Ordered goods must be on stock | |
| **Postconditions** | A Good Delivery Note is created and added into database  Payment status must be updated  Order status must be updated as being transferred | |
| **Flow of activities** | **Actor** | **System** |
| 1. Accountant checks the agent’s order.  2. Accountant approves the order after confirming goods status in warehouse.  3. Accountant accepts to print the slip | 1.1. Display the order information  2.1. Update order status as “being transferred”  2.2. Update payment status  2.3. Ask to print the delivery slip  3.1. Print out the delivery slip |
| **Exception Conditions** | 3.1. The printer does not work. | |

#### iv. Use case: View Reports fully description:

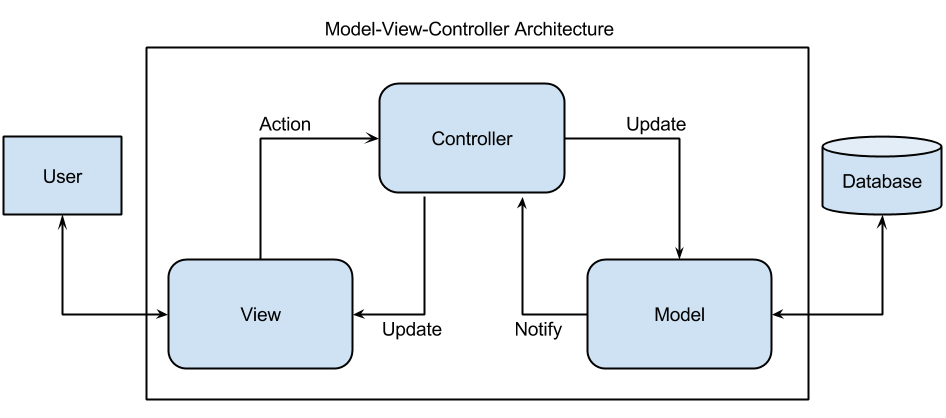
|  |  |  |
| --- | --- | --- |
| **Use case name** | View Reports | |
| **Scenario** | The company requires the accountant to do statistics of incoming/outgoing stock and revenue monthly | |
| **Triggering event** | Accountant wants to view reports | |
| **Brief description** | Accountant chooses the type of report to view | |
| **Actors** | Accountant | |
| **Related use case** |  | |
| **Stakeholders** | Accountant, Company | |
| **Preconditions** | Accountant must be authorized to access the system  The data must be stored in the database  The database must be connected with the system | |
| **Postconditions** | Accountant gets the the reports | |
| **Flow of activities** | **Actor** | **System** |
| 1. Accountant chooses to view incoming/outgoing stock report  2. Accountant chooses to view revenue monthly report | 1.1. Display the report of incoming/outgoing stock  2.1. Display the report of revenue monthly |
| **Exception Conditions** |  | |

* 1. **Functional requirements:**
* The system allows accountants to create new Goods Received
* The system allows accountants to create, as well as print out Goods Delivery Note
* The system must update orders status and payment status when Goods Delivery Note is created.
* The system allows accountants to view reports, including stock report and revenue monthly report
* The system allows agents to place orders
* The system allows agents to choose payment method
  1. **Non-functional requirements:**
* The system interface should be friendly and easy to approach.
* The system functions should work properly.

1. **ARCHITECTURE:**
   1. **Architectural style used:** MVC (Model – View – Controller)

Introducing 3 levels of MVC model:

* ***Model:***This level is very important as it represents the data to the user. This level defines where the application’s data objects are stored. The model doesn’t know anything about views and controllers. So, whenever there are changes done in the model it will automatically notify observers that the changes are made. The model may be a single object or a structure of objects.
* ***Views:***A view is a visual representation of the MVC model. This level creates an interface to show the actual output to the user. However, a view will not display anything itself. It is the controller or model that tells view of what to display to the user. It also handles requests from the user and informs the controller. A view is connected to its model and gets the data necessary for the presentation by asking certain questions. Sometimes, it also updates the model by sending appropriate messages. All these questions and messages are sent back to the model in such an easy terminology that can easily understand the information sent by a model or a controller.
* ***Controller:***The controller is a level that acts as the brain of the entire MVC system. A controller also acts as a link between a user and the system. It provides the user with the input by providing appropriate views to present it appropriately on the screen. The controller understands user output, converts it into the appropriate messages and passes the same to views.
  1. **Architectural model:**



* 1. **Technology, software and hardware used:**

**Technology:**

* Client framework: C#
* Database: MySQL

**Software:**

* Visual Studio 2019
* Visual Studio Code 2019
* Microsoft SQL Server Management Studio 17
* Microsoft Edge (Web browser)
* Figma (Design GUI)
* Paradigm Online (Design diagrams)

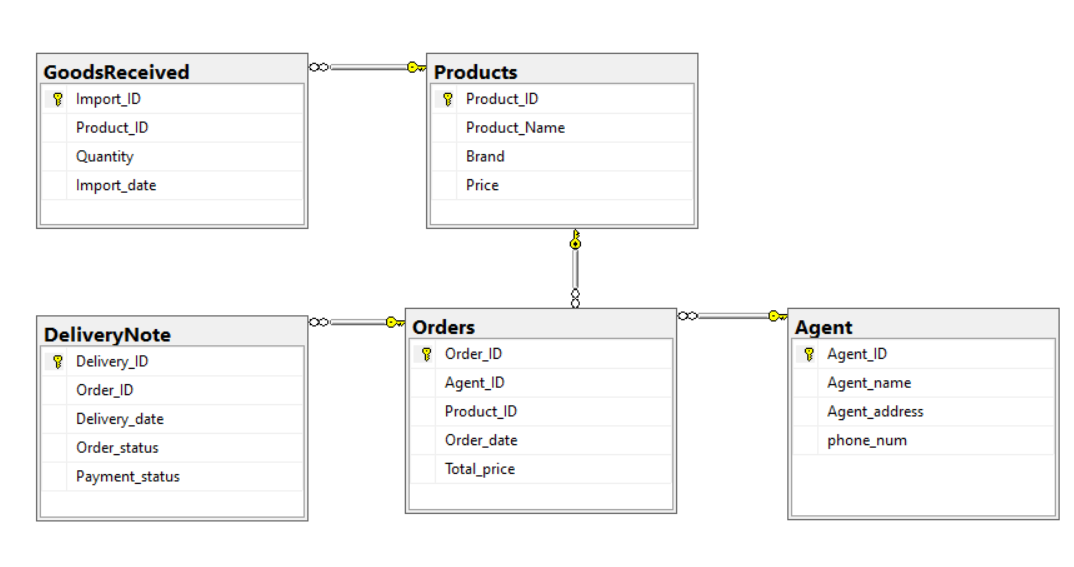
**Hardware:**

* CPU 1:
  + Processor: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21 GHz
  + RAM: 8 GB
* CPU 2:
  + Processor: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.20 GHz
  + RAM: 16 GB
  1. **Rationale for your architectural style and model:**

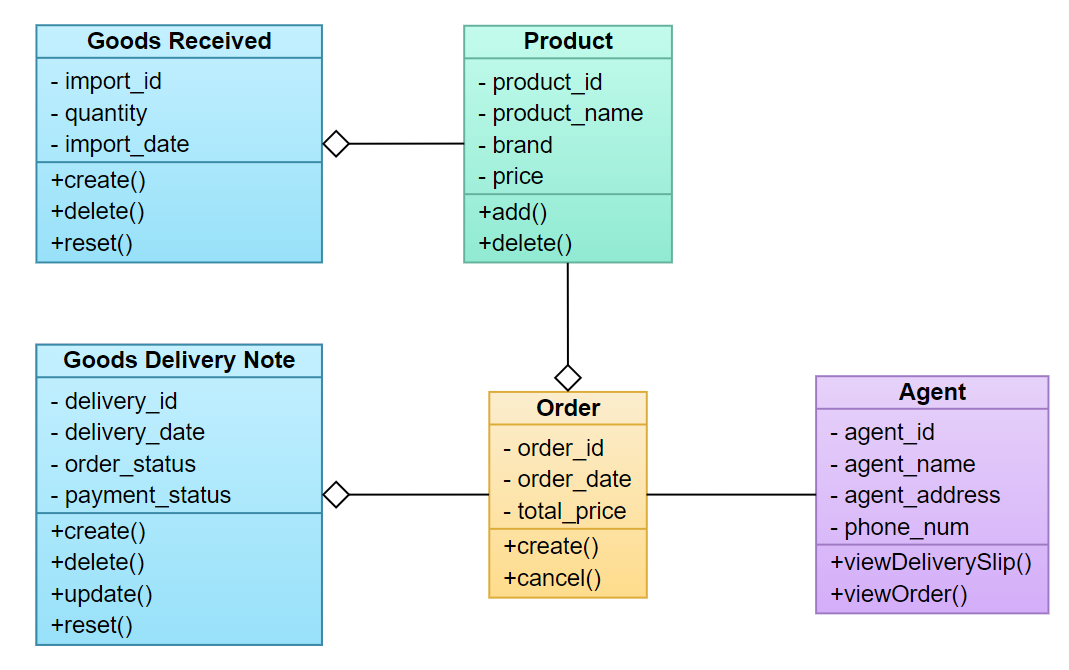
We use this architectural due to the advantages of using MVC:

* **Faster Development Process:** MVC supports rapid and parallel development. If an MVC model is used to [develop any particular web application](https://www.brainvire.com/asp.net-core-development) then it is possible that one programmer can work on the view while the other can work on the controller to [create the business logic of the web application](https://www.brainvire.com/microsoft-enterprise-services/). Hence this way, the application developed using the MVC model can be completed three times faster than applications that are developed using other development patterns.
* **Ability To Provide Multiple Views:** In the MVC Model, you can create multiple views for a model. Today, there is an increasing demand for new ways to access your application and for that MVC development is certainly a great solution. Moreover, in this method, Code duplication is very limited because it separates data and business logic from the display.
* **Support For Asynchronous Technique:** The MVC architecture can also integrate with the JavaScript Framework. This means that MVC applications can be made to work even with PDF files, site-specific browsers, and also with desktop widgets. MVC also supports an asynchronous technique, which helps developers to develop an application that loads very fast.
* **The Modification Does Not Affect The Entire Model:** For any web application, the user interface tends to change more frequently than even the business rules of the [.net development company](https://www.brainvire.com/asp.net-development). It is obvious that you make frequent changes in your web application like changing colors, fonts, screen layouts, and adding new device support for mobile phones or tablets. Moreover, Adding a new type of view are very easy in the MVC pattern because the Model part does not depend on the views part. Therefore, any changes in the Model will not affect the entire architecture.
* **MVC Model Returns The Data Without Formatting:** MVC pattern returns data without applying any formatting. Hence, the same components can be used and called for use with any interface. For example, any kind of data can be formatted with HTML, but it could also be formatted with Macromedia Flash or Dream viewer.

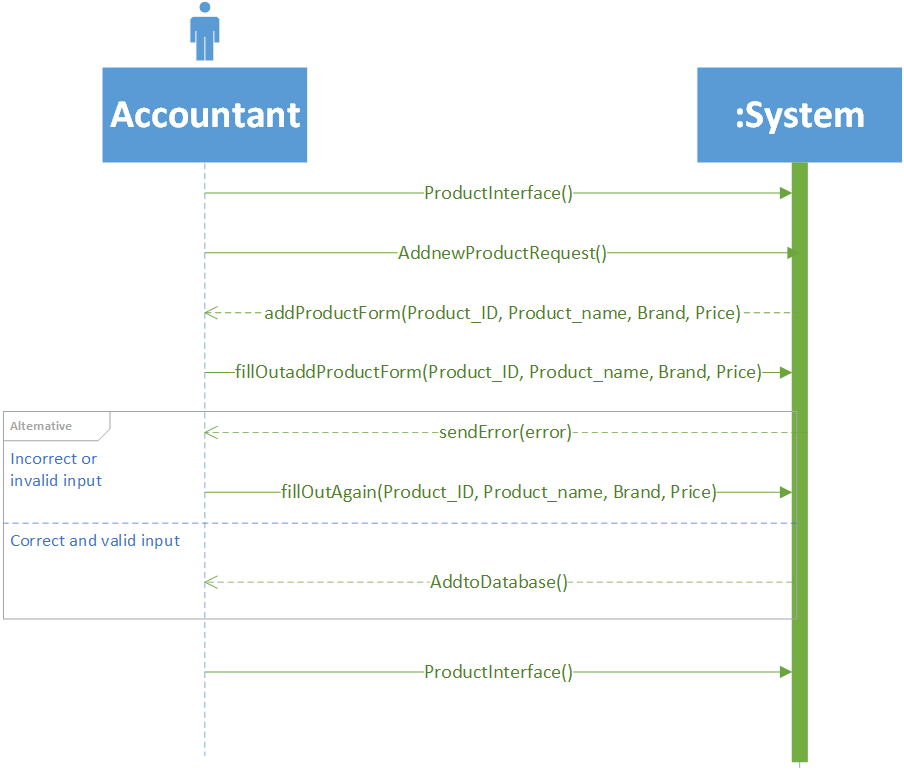
1. **DESIGN**
   1. **Database design**

****

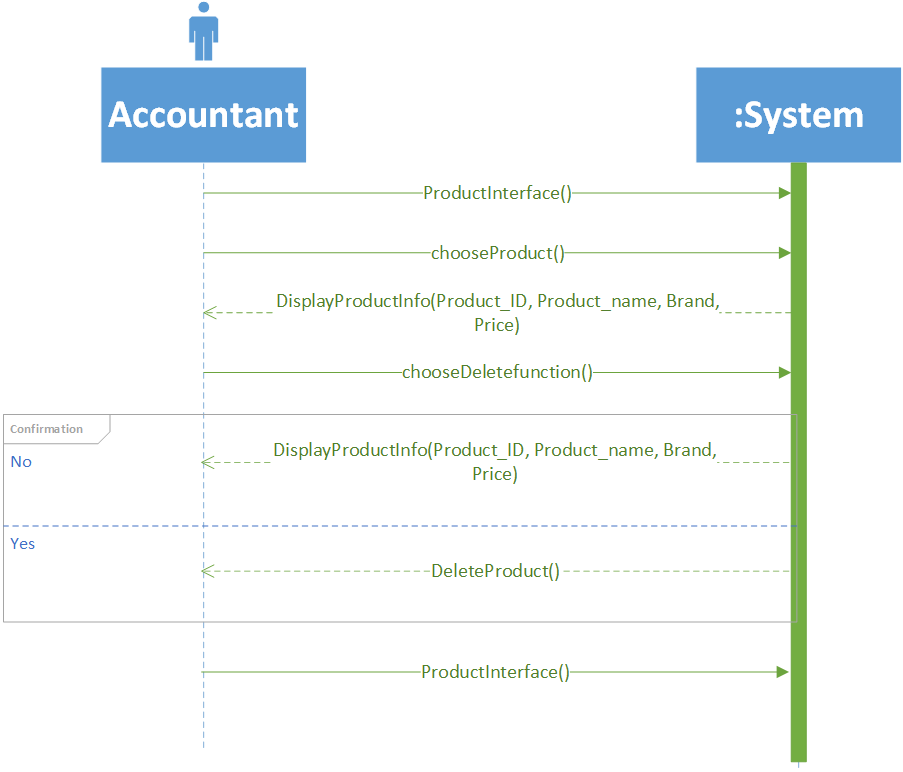
* 1. **Static model – class diagrams:**

****

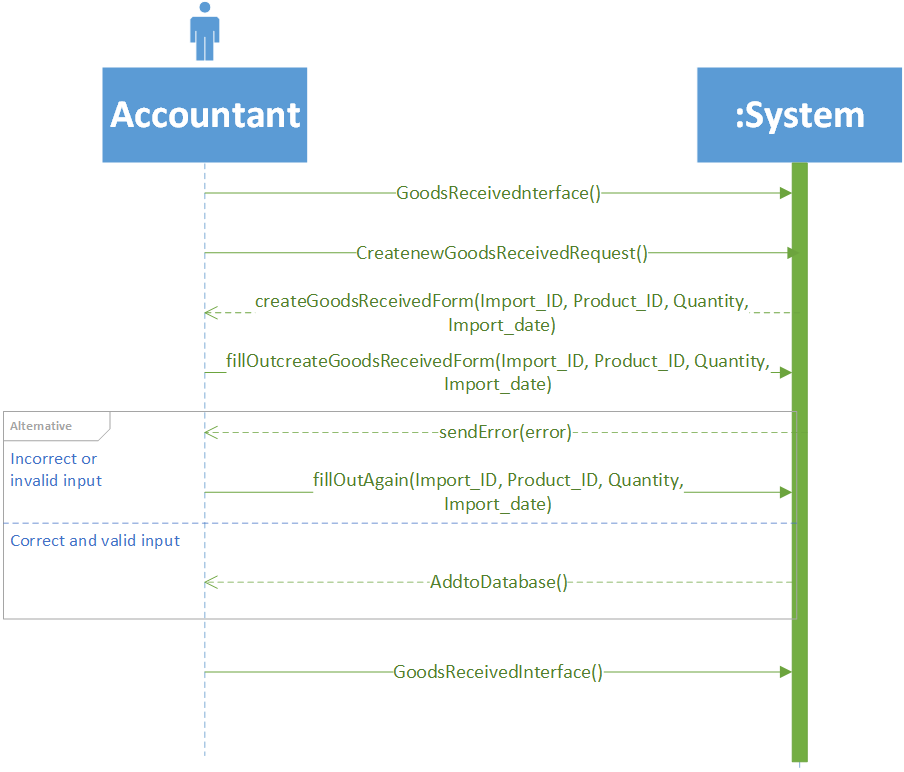
* 1. **Dynamic model – sequence diagrams:**
     1. **Sequence diagram: Add product**

****

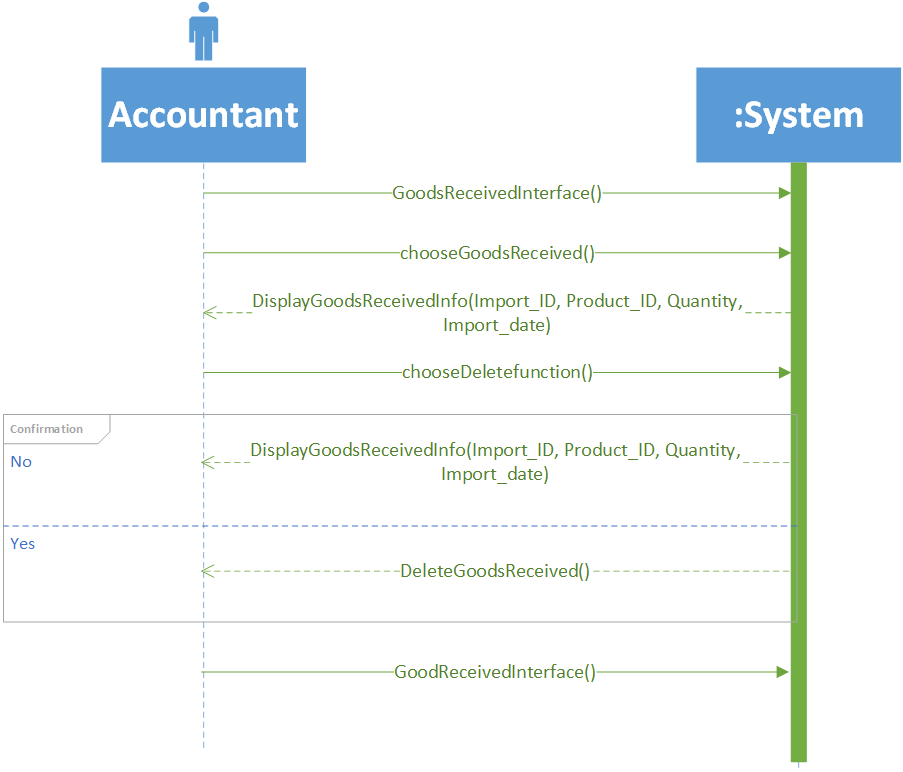
* + 1. **Sequence diagram: Delete product**

****

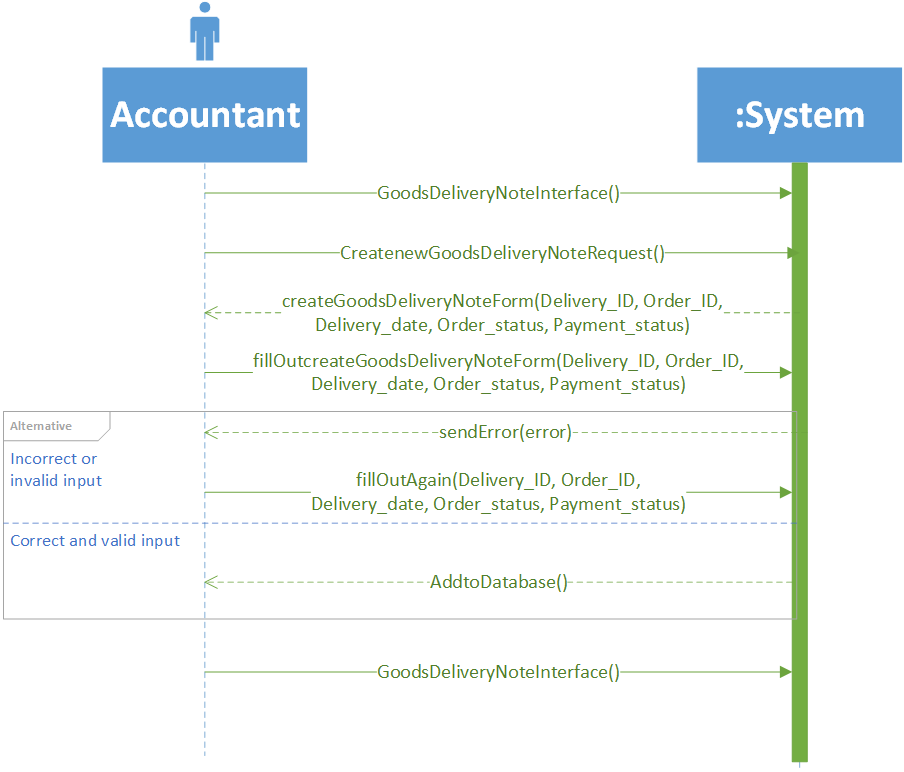
* + 1. **Sequence diagram: Add Goods Received**

****

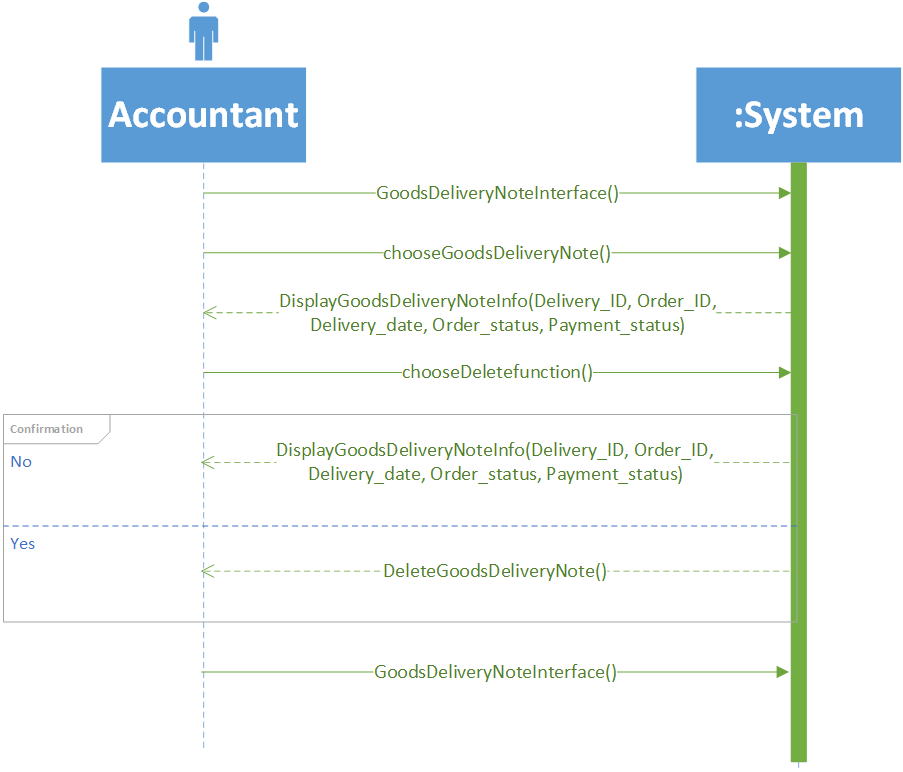
* + 1. **Sequence diagram: Delete Goods Received**

****

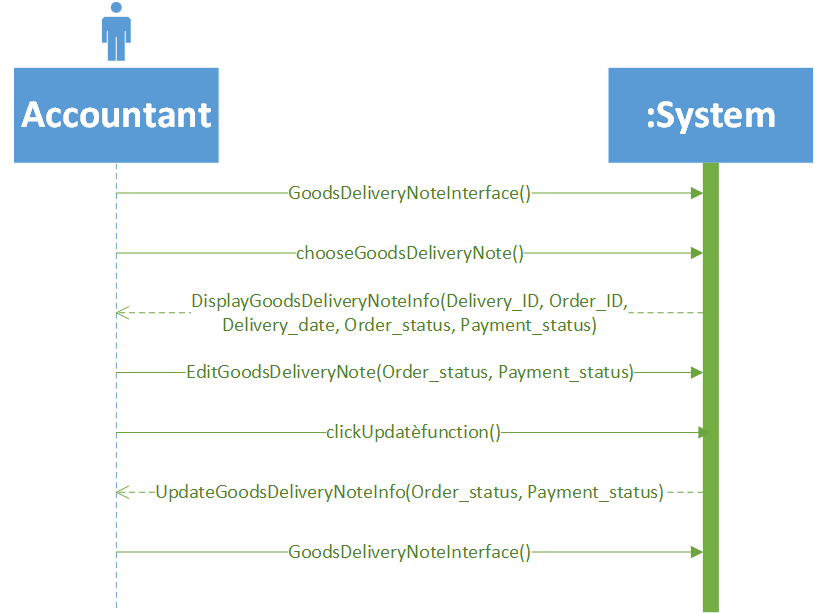
* + 1. **Sequence diagram: Create Goods Delivery Note**

****

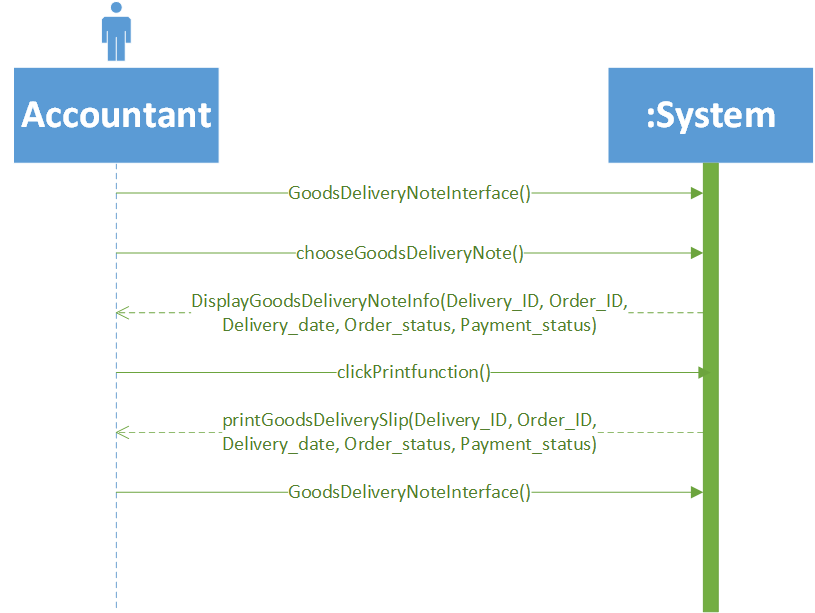
* + 1. **Sequence diagram: Delete Goods Delivery Note**

****

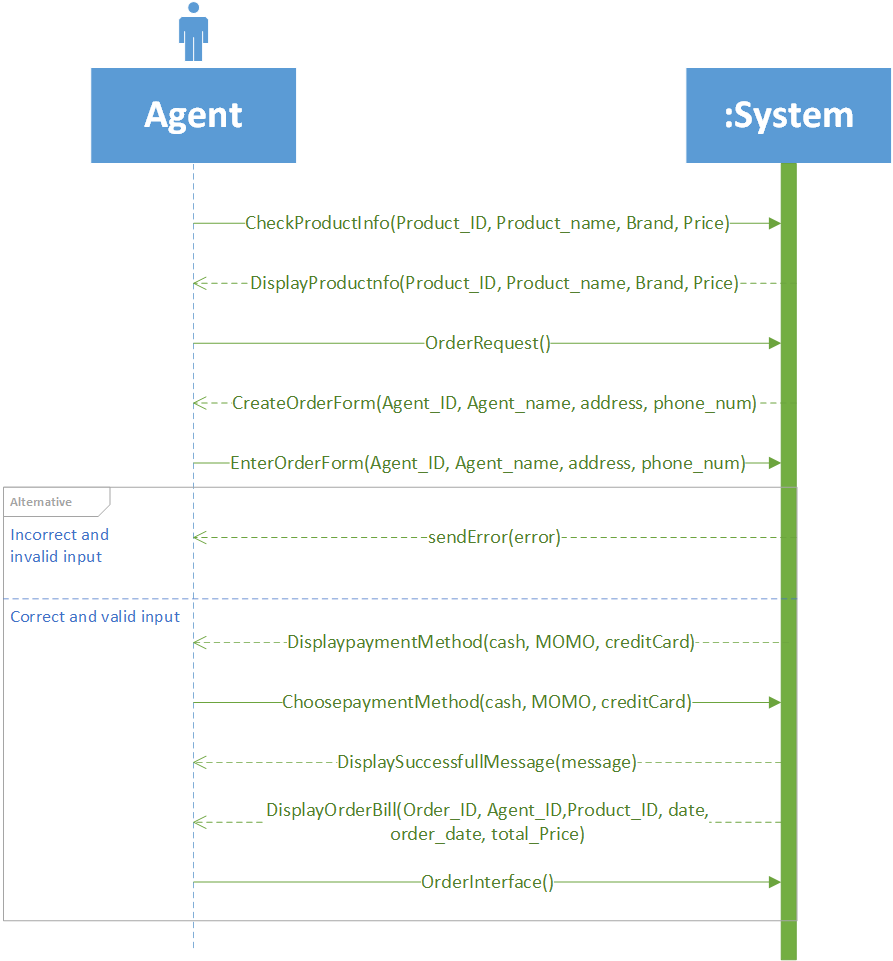
* + 1. **Sequence diagram: Update Goods Delivery Note**

****

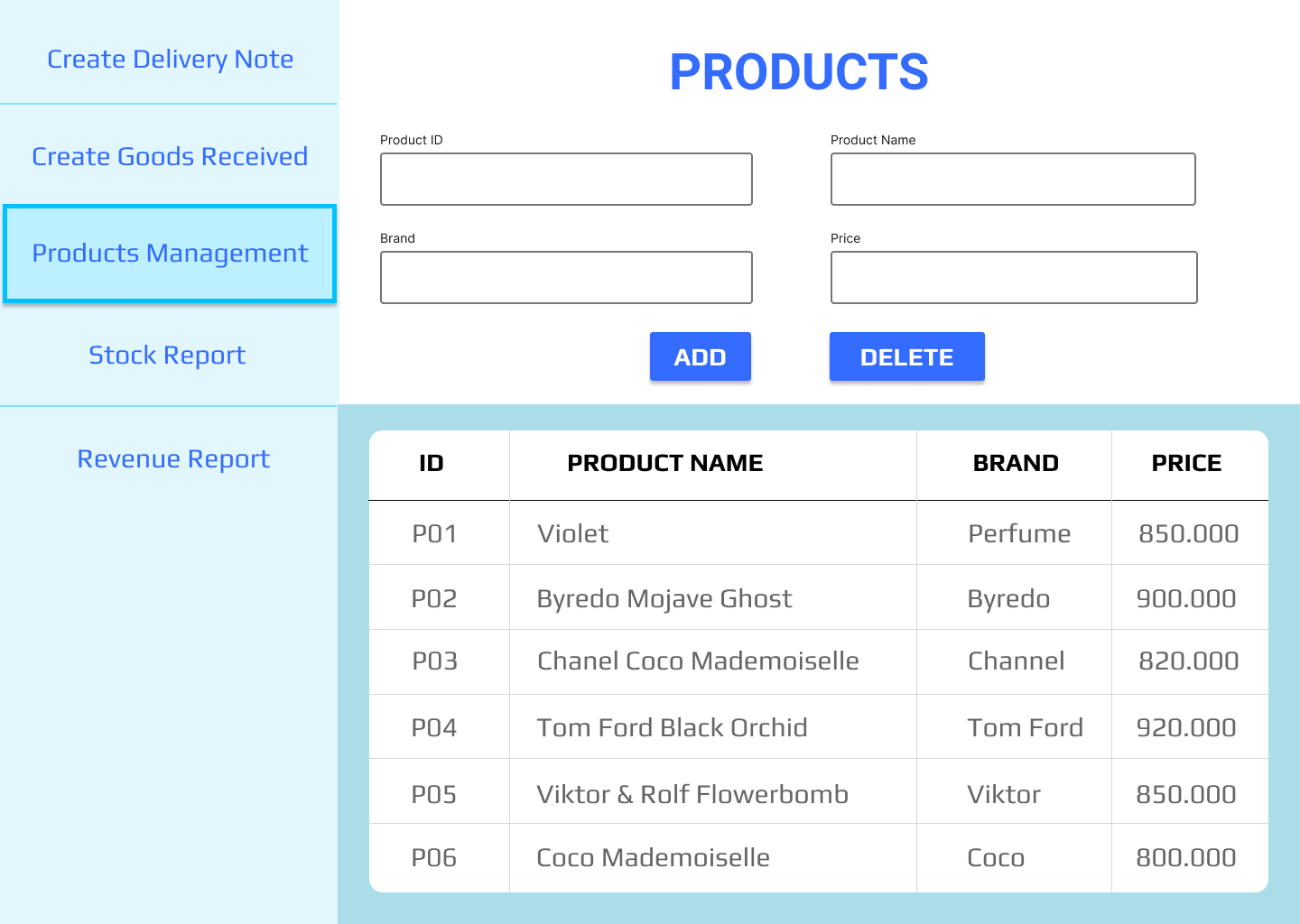
* + 1. **Sequence diagram: Print Delivery Slip**

****

* + 1. **Sequence diagram: Order Payment**

****

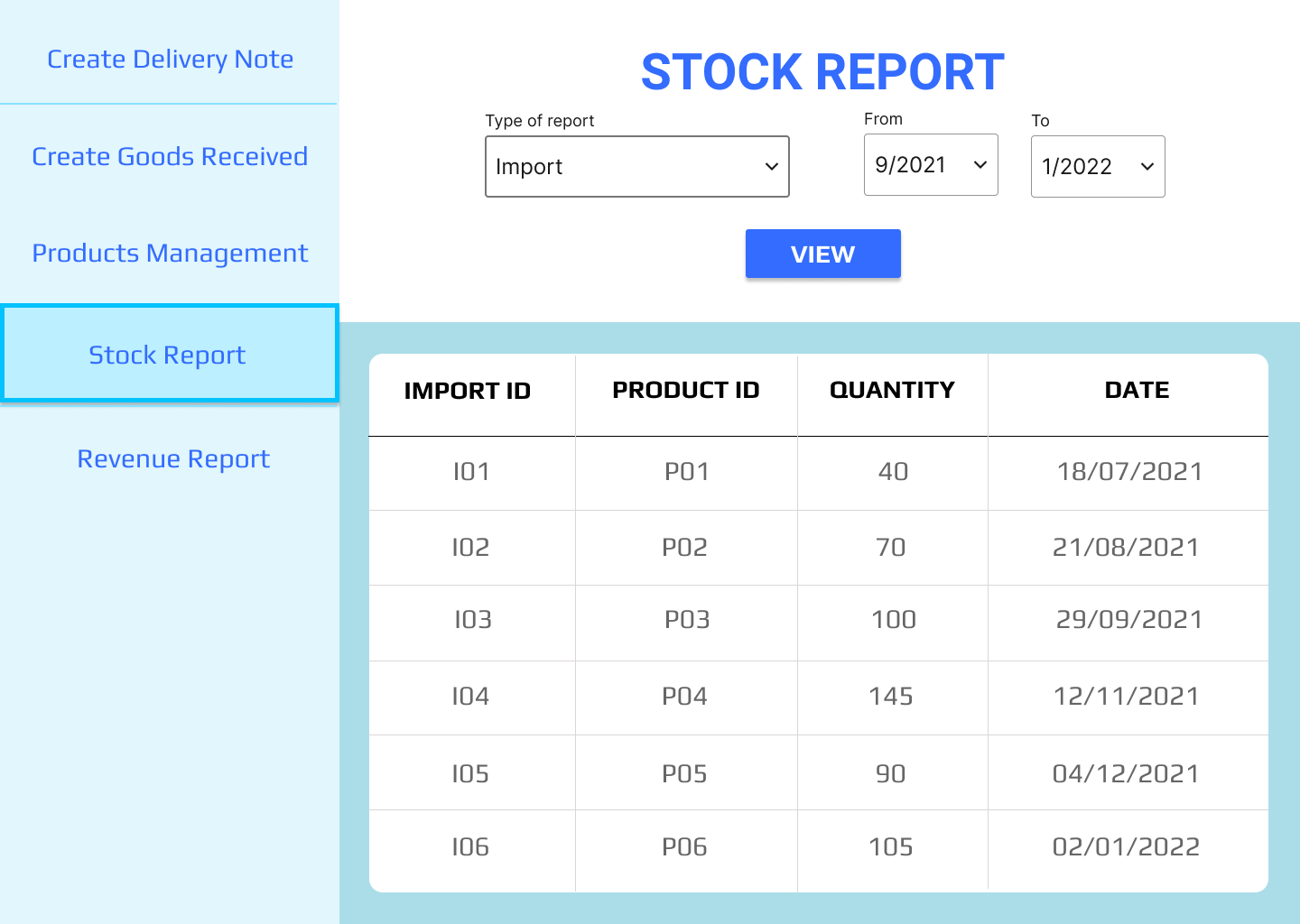
* 1. **GUI design:**
     1. ***GUI: Products Management:***

****

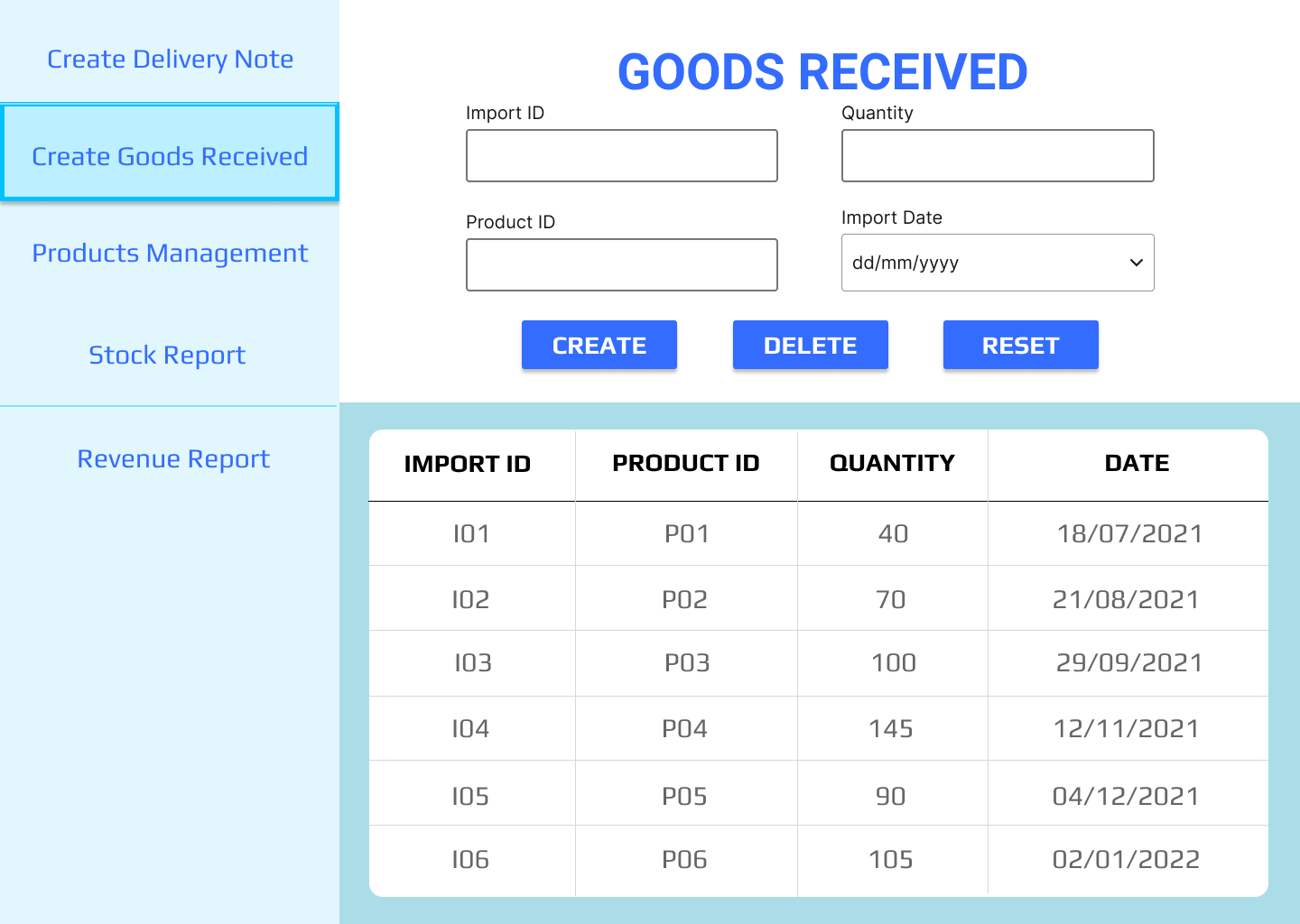
* + 1. ***GUI: Revenue Report*:**



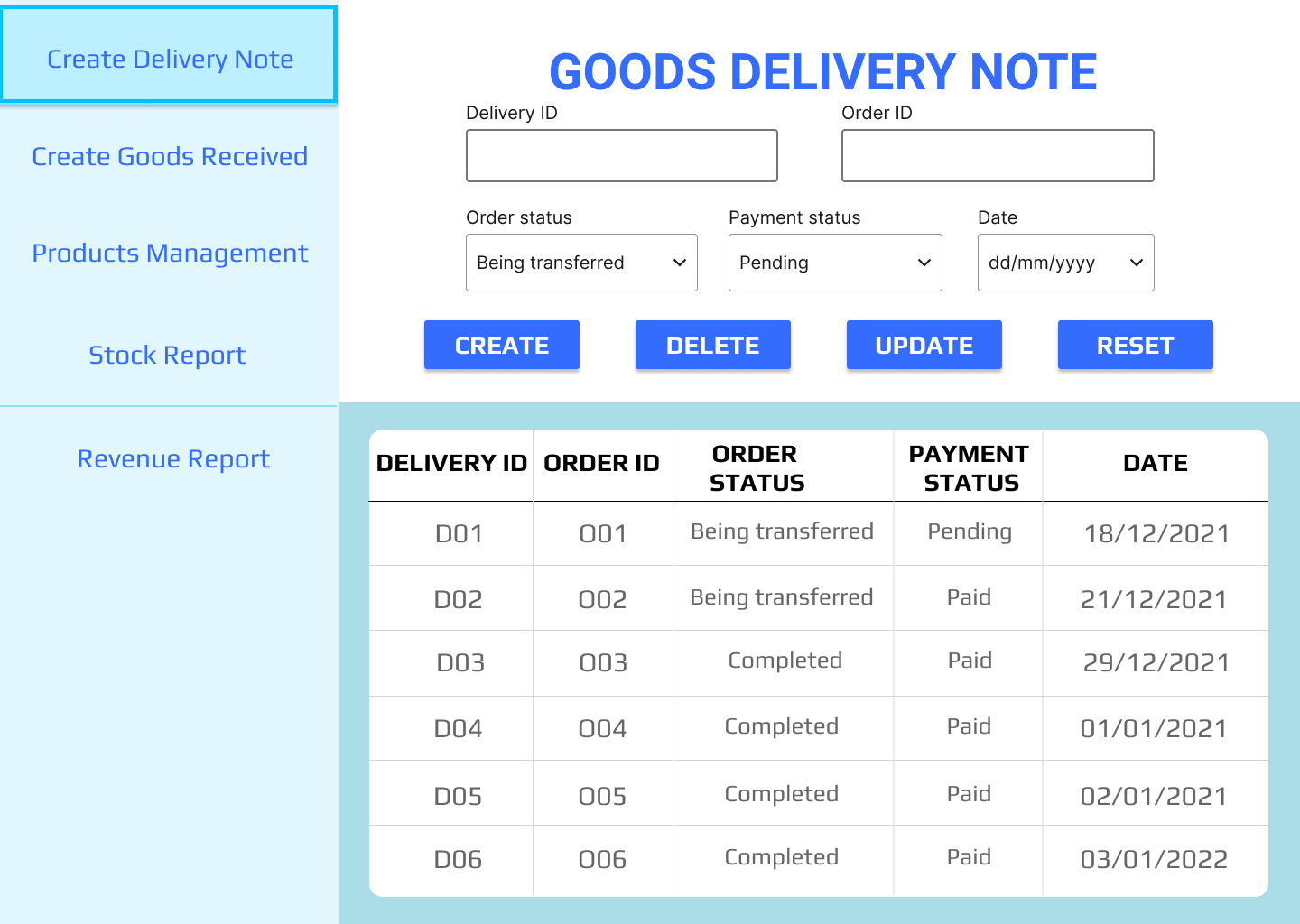
* + 1. ***GUI: Stock Report:***



* + 1. ***GUI: Goods Received:***

****

* + 1. ***GUI: Goods Delivery Note:***

****

1. **TEST PLAN**
   1. **Requirements/specifications-based system level test cases**

**-** The functions should work properly and meet the system requirements.

**-** The data in the database should be fully displayed.

* 1. **Traceability of test cases to use cases** 
     1. ***Test case: Create Goods Received***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Import ID** | **Product ID** | **Quantity** | **Date** |
| **Valid input** | I01  I3000 | P69  P100 | 300  100000000 | 04/01/2022 |
| **Invalid input** | empty  ABCDE123456 | empty  ABCDE123456 | empty  ABC  -100 | empty |

* + 1. ***Test case: Create Goods Delivery Note***

|  |  |  |
| --- | --- | --- |
|  | **Order ID** | **Delivery ID** |
| **Valid input** | O01  O3000 | D01  D999 |
| **Invalid input** | empty  ABCDE123456 | empty  ABCDE123456 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Order status** | **Payment status** | **Date** |
| **Valid input** | Being transferred  Completed | Pending  Paid | 04/01/2022 |
| **Invalid input** | empty | empty | empty |

* + 1. ***Test case: View Stock Report***

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Type of report** | **From date** | **To date** |
| Valid input | Import  Export | 02/01/2022 | 04/01/2022 |
| Invalid input | empty | empty | empty |

* 1. **Techniques used for test generation**

**- Black box testing** is a type of software testing, which checks for the functionality of a software or an application without knowing the design, internal components, or structure of an application to be tested. It is also referred to as Specifications-based testing.

- The **black box testing method** is mainly used to find missing functions, performance errors, initialization errors, and errors while accessing the external database.

- In Black box technique, we decided to use **Equivalence Partitioning** techniques, which means the input data of an application to be tested into equal partitions. This technique ensures to cover each partition at least once.

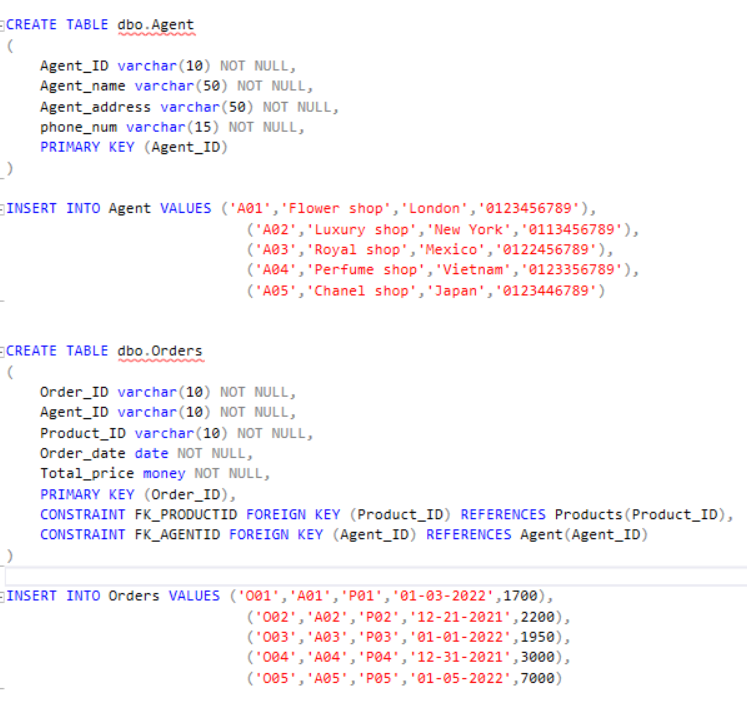
* 1. **Assessment of the goodness of your testsuite**

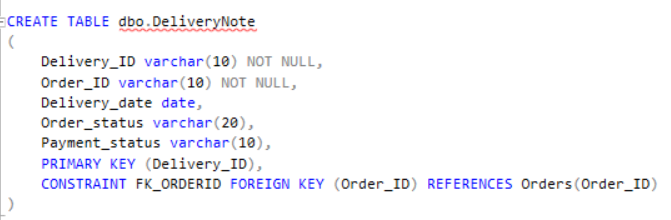
The testsuite is good and complete enough to ensure the system works properly

1. **DEMO:**

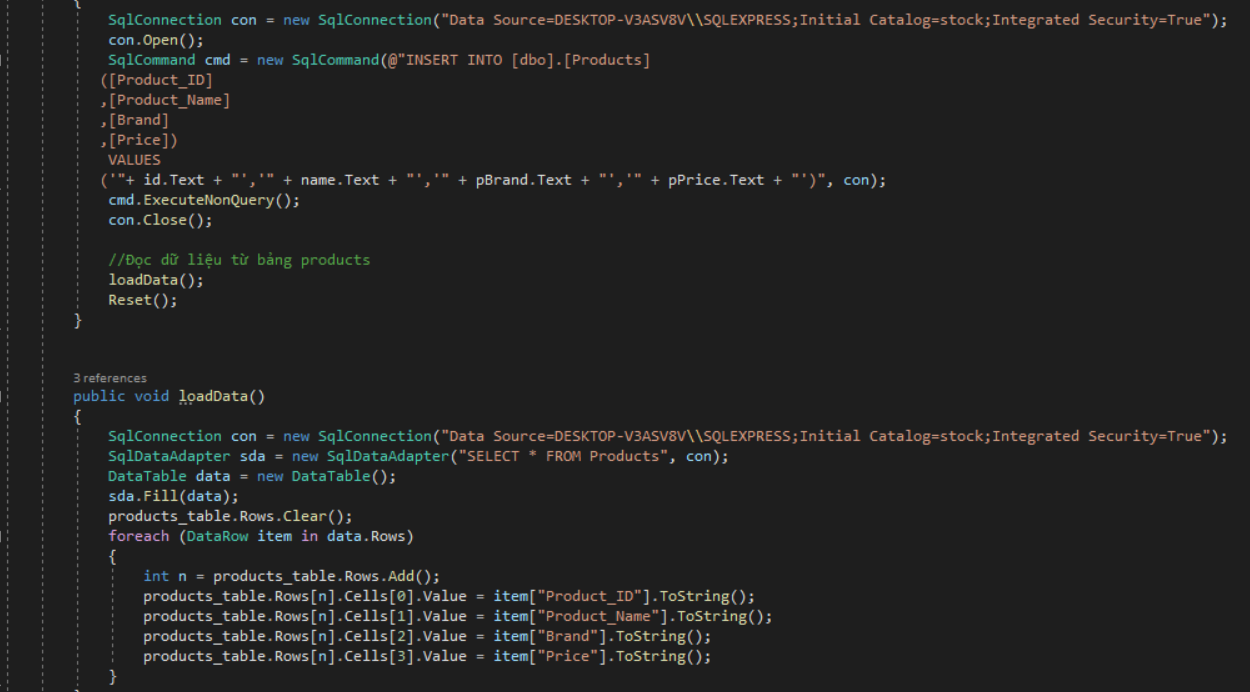
**Link demo:** [demo.mp4 - Google Drive](https://drive.google.com/file/d/1hUeOYp0pUNw7ApWC8zJ3-FNGfG3jv5qH/view?fbclid=IwAR1UJUvo_O3RuXa_WCWQUNZgFEX78T0saFeyiUXmJ295-n4UReiNvMfchOo)

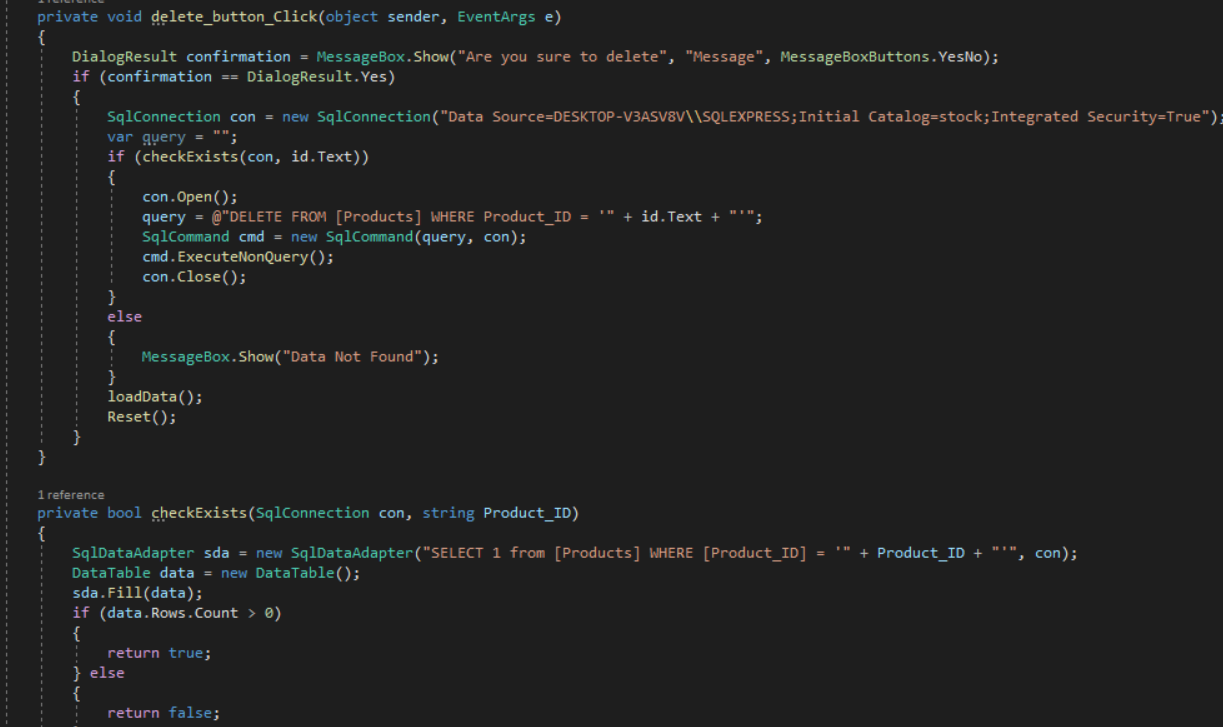
* 1. **Database:**





* 1. **Source code:**



****

****

# REFERENCES

1. Six Benefits Of Using MVC Model For Effective Web Application Development, from [6 Advantages of using MVC model for effective web application development (brainvire.com)](https://www.brainvire.com/six-benefits-of-using-mvc-model-for-effective-web-application-development/)
2. What are Testing Techniques : Types, Advantages & Disadvantages, from [Testing Techniques : Types, Advantages and Disadvantages (elprocus.com)](https://www.elprocus.com/what-are-testing-techniques-types-advantages-disadvantages/)