Faculty of Economics and Administrative Sciences

Business Informatics Department

Software Engineering Project

“Beta Business”

CEN – 302

EPOKA UNIVERSITY

2016-2017

Faculty of Economics and Administrative Sciences

Business Informatics Department

Software Engineering Project

"Beta Business"

CEN – 302

Prepared by:

Albi Trashani

Igli Tola

Ernaldo Gjomakaj

Arnold Valteri

Ornela Torra

Prepared for:

Igli Hakrama

1. Abstract

"Software is a great combination between artistry and engineering." **- Bill Gates**

Due to hard work and team contribution, we present to you "Beta Business", a computer based software, which will help any business deal with their daily finances, costumers, the whole business, in a very effective and efficient way, and more importantly helping its creators (Beta Business team members) developing effective written, oral, thinking & programming skills into making such a project as this one.

2. Acknowledgment

In performing our Software Engineering project, we had to take the help and guideline of a respected person, who deserves our greatest gratitude. The completion of this project gives us much pleasure. We would like to show our gratitude to **Mr. Igli Hakrama**, who has helped us not only during this project, but being an important influence in our programming skills during these last two years.

3. Contents

This Page is intentionally left blank

4. Executive Summary

4.1 Project Description

Albania is a developing country which has started to emerge in the region after 50 years of communism, in a formerly centrally planned economy. With the help of international aid, and some other strategic assistance, the country has made a lot of progress, growing from the poorest nation in Europe in the 1990's to middle income status in 2008. The economy is mainly bolstered by agriculture and it employs 47.8% of the population and it has the largest contribution to the GDP. The second largest factor of the economy in Albania is services which employs 46.8% of the population. Many foreign investments have been done in Albania, while many Albanian citizens have been trying to start their own businesses. Many of the local businesses have been successful, and some others have gone bankrupt. There are numerous reasons why these businesses have gone bankrupt. In some cases the demand for the products the businesses were offering was low, so they went bankrupt, in some other cases, because of the bad management of the finances the business had a breakdown. Why did this happen? This was caused mainly because of the way the managers were managing the business, not having a clear vision of their inventory, their sales, reports and debts, and miscalculations. So this brings out a very important question. What can be done in order to have a clear vision of a business finances? This project is called "Beta Business" and it's an innovation in Albanian businesses. What is "Beta Business"? It is a software that helps company to manage their finances, costumers and whole business. It will offer easy data exchange between different departments of the company. Our program is suitable for all types of businesses, no matter how small or big they are. The purpose of this project is to make a program which will help managers use their company data in a more efficient and effective way. The program will have some key features like recording daily documents and transactions in a quick and easy way, inventory will be easily manageable, it will monitor transactions made by company, will keep your business performance boosted.

4.2 Purpose and Scope of this specification

Being a developing country, Albania has done a lot of improvements in the service sector, being the second after the agriculture sector. Some businesses need a huge staff, while some others don't need help to run their business. But what all businesses need, is a user-friendly software, that can help them in their daily working hours. Beta Business is the solution to all those businesses who are struggling in managing their finances by hand-writing their data, which includes transactions, inventory, reports, which often leads to business- losses, chaos and crisis. Our project will help in trying to solve these problems by using our program which will be accessible by any working computer. The intended audience for this software are the managers of the business who is using it, and also to the employees, which will have some limited privileges in which we will talk later on the project. The staff is thought to use the program on a daily basis, to record new data, and to edit and update existing data. The software "Beta Business" will cope with a huge amount of data that will be uploaded daily by its users. It will have some main parts such as: inventory, transactions which will be daily, monthly, yearly. Other parts include debts, suppliers and reports.

The key to this successful project will be the security of the program, overall performance and data consistency.

5. Product/Service Description

5.1 Product Context

This product is related to other financial software's, because it helps in managing the finances of the whole business. It is independent, it doesn't need to interface with a variety of related systems as all of the components will be genuine.

5.2 User Characteristics

This product is designed to be used by many people who work in the company, and this program will require users to have knowledge on using this application and a two week course will be made in order to learn how to use this program. Employees with no knowledge will be strictly prohibited to use it.

So who is going to use this program?
Let's start with the higher hierarchy: The top managers: CEO, CFO will have a unique access to the program, different from the others since they need to know absolutely everything going on in their company (stuff like total revenues, cash flows, reports, transactions, debts etc.).
Besides them, the employees, or let's call them sales agents will also have access to the program, but unlike managers, sales agents will have limited access. It is not their responsibility, and definitely not their business to know every detail in the company they work for. So for now they will only have access to daily transactions, reports, bills, and also partially at the debts page.

5.3 Assumptions

The company should have the following resources:

At least a medium PC, internet connection also a software to make possible connection with a database. (E.g. XAMPP, WAMPSERVER)

5.4 Constraints

For the product to work perfectly well, it needs to be installed at the proper machines. It will run smoothly on a high-spec PC, and okay in a medium PC, but it definitely will crash if you install it on an old, low-spec PC. Having all those databases and data will need a lot of free space in your PC.

5.5 Dependencies

Beta Business program doesn’t have any specific dependency. The only one that can be mentioned is that the program is dependent only by the user who is using the program.

6. Requirements

6.1 Functional Requirements

Req#	Requirement	Comments	Priority	Date Rvwd	SME Reviewed / Approved
BR_01	The program should include every product that the company has in inventory.	Business Process = "Adding products"	1	03/04/17	Albi T, ErnaldoGj, Arnold V, Igli T, Ornela T
BR_02	The program should calculate how much is left, cost of each product, how much will they sell it.	Business Process = "Calculations" A missing product alert will help in informing our suppliers	2	03/04/17	Albi T, ErnaldoGj, Arnold V, Igli T, Ornela T
BR_06	The program should keep a record of all of the clients of the company.	Business Process = "Adding clients"	2	03/04/17	Albi T, ErnaldoGj, Arnold V, Igli T, Ornela T
BR_09	The program should be able to specify the clients, to add them in the debts page if they have any debts in the company.	Business Process="Add client debts" When clients pay their debts, they will no longer be in the debts page, and their status will be "No debts"	2	03/04/17	Albi T, ErnaldoGj, Arnold V, Igli T, Ornela T

Req#	Requirement	Comments	Priority	Date Rvwd	SME Reviewed / Approved
BR_13	The program should be able to see reports from daily sales of the company.	Business Process=" Reports" A calendar will be visible to make it easier for users to navigate through the days/months to see their balances and numbers	2	03/04/17	Albi T, Ernaldo Gj, Arnold V, Igli T, Ornela T
BR_14	The program should be able to identify users	Business Process="Credentials" A sales agent doesn't have same access as managers so the login page will identify users.	1	03/04/17	Albi T, Ernaldo Gj, Arnold V, Igli T, Ornela T

6.2 User Interface Requirements

There will be a menu in the main page of the program, with the buttons showing different modules of the program, to make it easier for the users (admins and employees).

It will be user friendly and intuitive. If the user makes a mistake while using the program, error alerts will appear with specific messages shown what you have done wrong while operating in it. The modules will be shown in 2 columns and 3 rows, making it a simple and users will have it easier to find their modules.

6.3 Usability

Accessibility – The top managers: CEO, CFO will have a unique access to the program, different from the others since they need to know absolutely everything going on in their company (stuff like total revenues, cash flows, reports, transactions, debts etc.).
Besides them, the employees, or let's call them sales agents will also have access to the program, but unlike managers, sales agents will have limited access.

Learnability – The system will be easy to use, everything will be explained inside the program on mouse hover. Although users have to learn this program 2 weeks before they start using it.

6.4 Performance

We will make sure our program performs at its highest speed. When the user tries to find a product from our database, the program searches it by name and it will be able to find it in 1-2 seconds, depending on the search key the user has provided. When looking at transactions made during the day, the program will firstly show the most recent ones, then if the user wants to see more, he will have a button to show all of the transactions made during that day (or other days).

6.4.1 Capacity

The program will be flexible for all types of businesses, which means that no matter how big the business is, the program will still be able to process all the data in no big deal, it will have a big capacity. Users will be able to get as many reports as they want, anytime they want them.

6.4.2 Availability

Hours of operation: The program will be required to be online by users at least 8 hours (since the average working hours per day for a worker is 8 hours).

Level of availability: The program will be available 24/7 for the users.

By the geographic means, the program will be available from everywhere, the only constraint may be the slow internet connection that users may face from where they are using the program.

6.4.3 Latency

The response time for the program to process some data will be very efficient and very fast, not including conditions where users have installed this program on old and slow PC's.

6.5 Manageability/Maintainability

6.5.1 Monitoring

The program as we have previously mentioned will be user friendly, and very easy to use. Users will have no issues in using our program, since everything will be explained on mouse hover on everything they see on the program. Also a "?" button will be available on top corner of the program's window so the user can check for solutions in case they have some problems with the program. If the information that they find instructed on the program is not enough for them, and they are facing problems that they cannot solve, we will provide users with some contact details from the programmers of the Beta Business which will assist its clients in the best and fastest way possible.

6.5.2 Maintenance

The simple and efficient design of Beta Business will help users do their job without programming skills. It is easily understandable. Later on updates will be available to download for the program.

6.5.3 Operations

Users can store new products when the program doesn't have that information.

The user can access the system whenever they need information.

The user can add, delete clients.

The user can add, delete, and update debts.

The user can store new transactions.

The user can see daily, weekly, monthly, yearly reports.

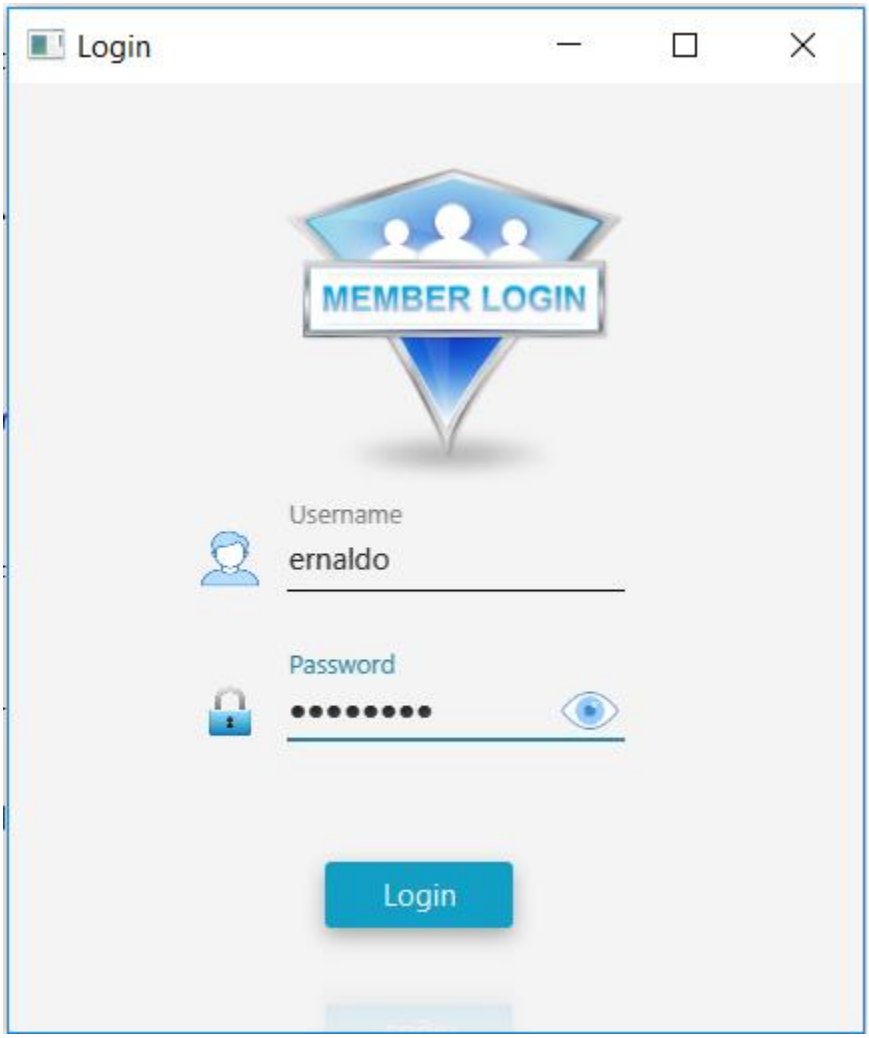
The user can search for specific person.

The user can search for specific transaction.

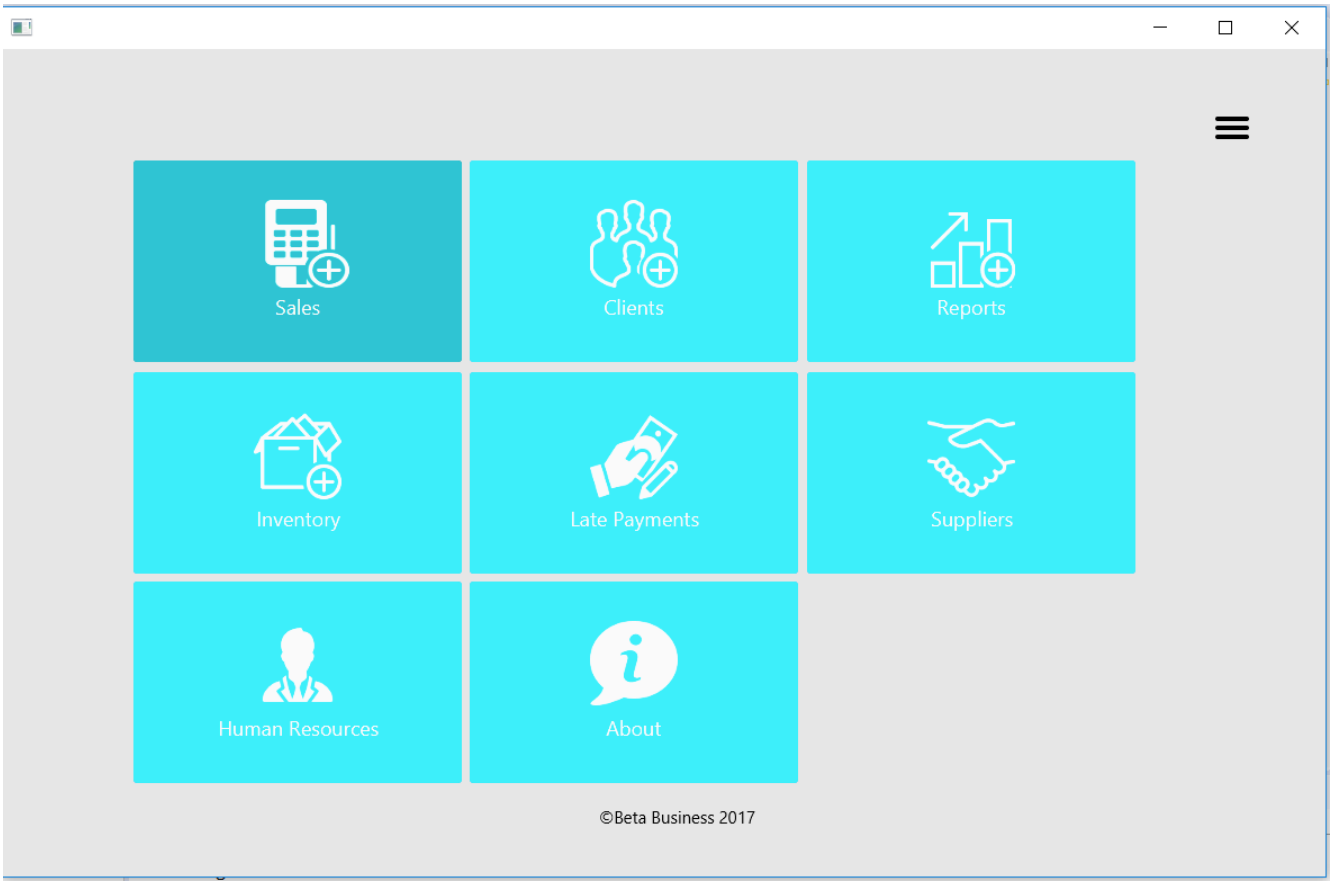
The user can search for specific debt.

The user can check financial situations for specific periods.

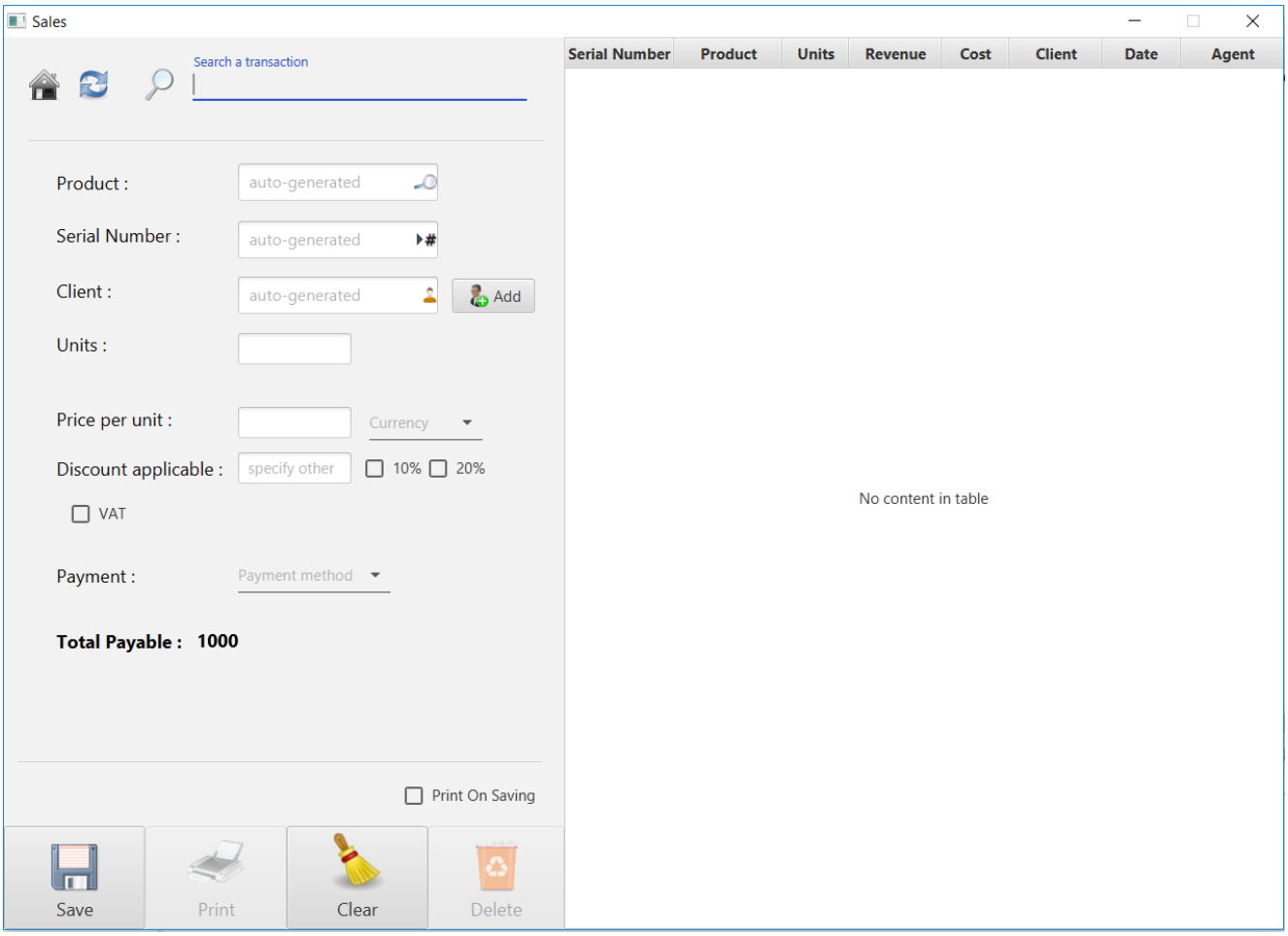
6.6 System Interface/Integration



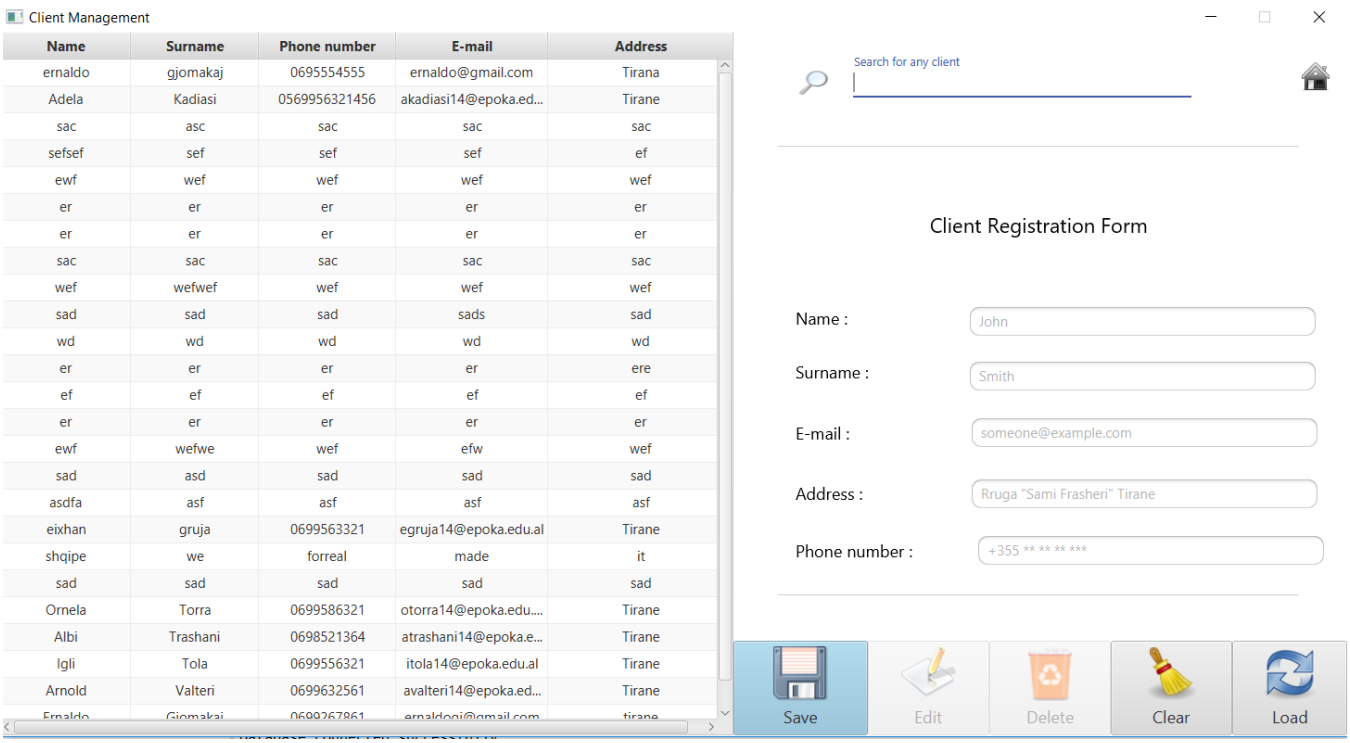
Login Page



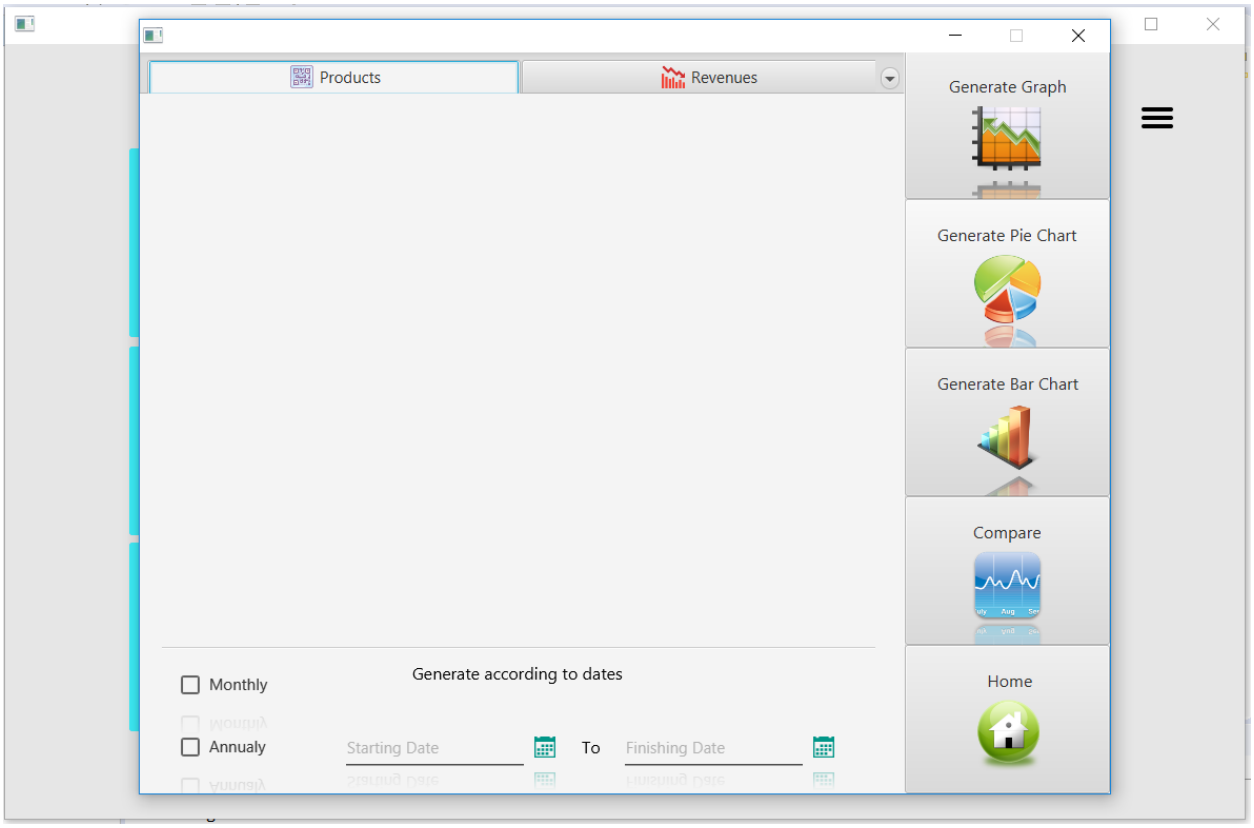
Main Menu Page



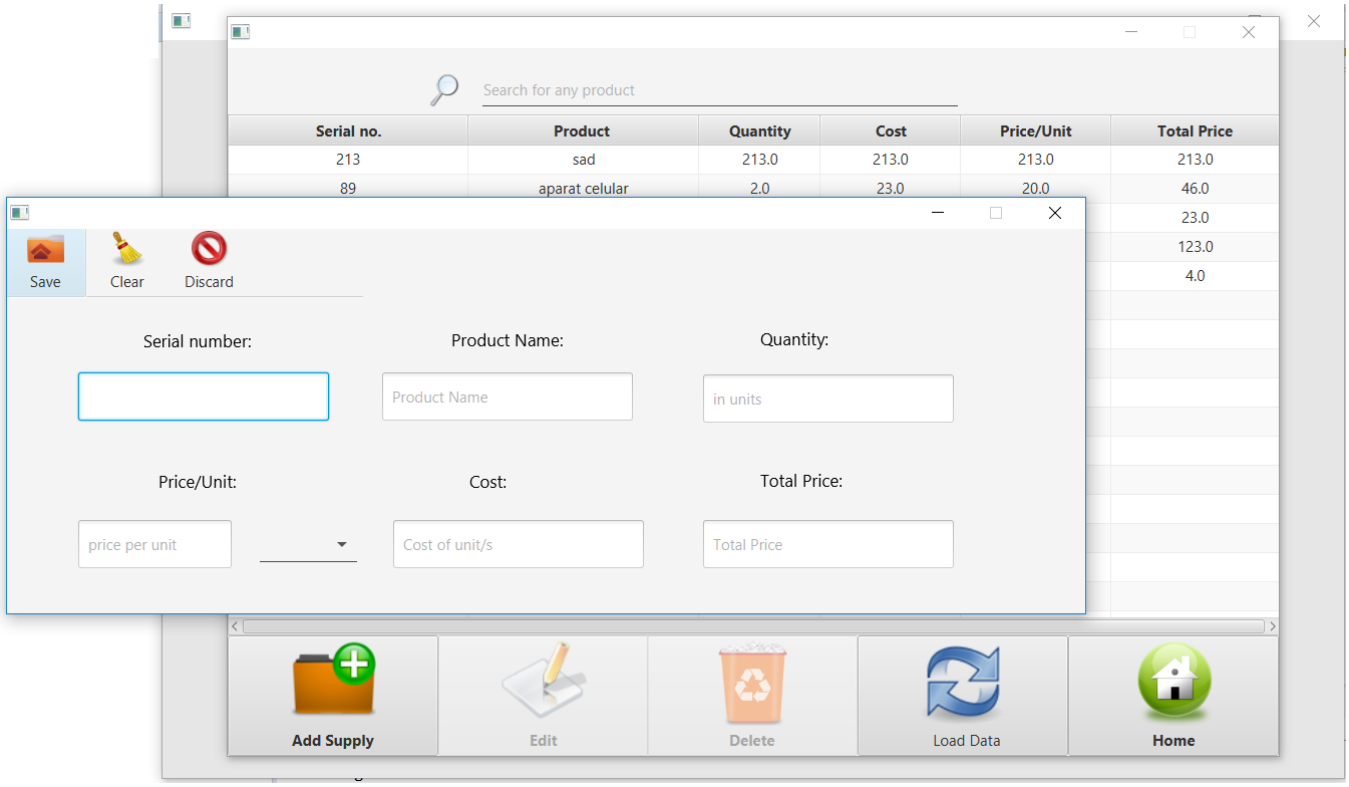
Transactions Page



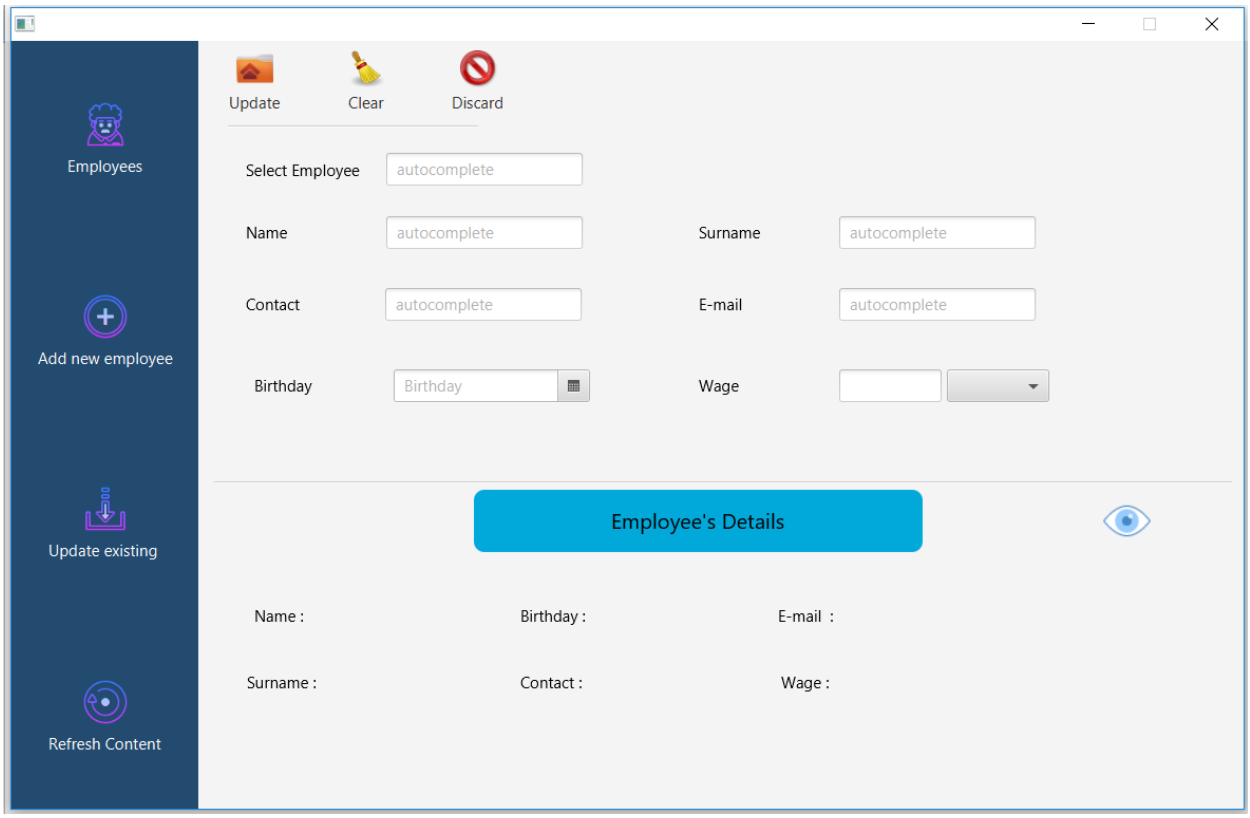
Client Management Page



Reports Page



Inventory Page



Employees Page

6.6.1 Network and Hardware Interfaces

Beta Business is a software that serves as a connecting interface between users and a database, which will contain everything it needs in order to run a business. The program can be accessible from every computer that has Beta Business installed, with the only condition not to be very low spec PC, for a maximal performance and efficiency.

6.7 Security

6.7.1 Protection

The data that will be stored in the "Beta Business" program will be very secure and private known only for the users that will use the program (and the government but that's another story). The external persons will have no access to the program. We will protect the system from malicious or accidental access, modification, disclosure, destruction, or misuse. The program will have MD5 secure encryption and if the user will be able to log in only if he has the right combination of the username and password.

6.7.2 Authorization and Authentication

The program will be able to know exactly who is accessing their information or site. Authentication and Authorization are used together. The staff of the business are required to authenticate before accessing the Beta Business. The authentication they provide determines what data they are authorized to see. The authorization step prevents employees from seeing data of managers. Employees will have different privileges. They will be able to access only the transactions page, client page and inventory page. The authentication doesn’t require internet service.

6.8 Standards Compliance

Standards Compliance means writing code according to a given set of specifications. We as the team members will try and program the code according to our set of specifications that we have written above. Beta Business will function also as an audit tracer as it will have every report and transaction that will be done in the daily basis. The system will do the tracing by searching the needed information according to the given specification. As mentioned above, the data will be secured and sensitive only to the users of the program, and also will serve for government institutions that will maybe want to have a look at the business records to see if there is something wrong with the business, or in case of a tax invasion.

6.9 Portability.

Domain requirements reflect the environment in which the system operates so, when we talk about our program domain we mean environments such as transactions, reports etc.

The domain requirements are fundamental for our project because a lot of calculations shall be done. If these requirements are not satisfied, it may be impossible to make the system work satisfactorily. There will be multiple multiplications such as:

Total products price= price x quantity.

Also there will be a lot of additions such as: sum of different products.

This Page is intentionally left blank

7. User Stories, Scenarios and Use Cases

7.1 User stories

Nr.	**User Story Name**	**Description**
1.	Log in	Provides the users to log in, by using their user name and password, in order to access the program. Employee's login is different from Admin's login.
2.	Add Products	Both types of users have access to this part of the program, and they have to add new products.
3	Remove Products	Both types of users have access to this part of the program, and they have to remove any products.
4.	Add Clients	Both types of users have access to this part of the program, and they have to add new clients.
5.	Add Debts	Both types of users have access to this part of the program, and they have to add new debts.
6.	Delete Debts	Both types of users have access to this part of the program, and they have to delete debts.

7.	Reports	This module is accessed only by the Admin. They can have a detailed information about business activity.
8.	Search name	The user search for a person that he want to his activity in company by typing his name. After the name is typed there it will be shown all information for this person.
9.	Search date	The user can search for all transactions made in company during a period of time by entering beginning and ending date.
10	Add transaction	Both types of users have access to this part of the program, and they have to add new transaction.
11	Add supplier	Both types of users have access to this part of the program, and they have to add new supplier.
12	Update supplier	Both types of users have access to this part of the program, and they have to update supplier.
13	Delete supplier	Both types of users have access to this part of the program, and they have to delete supplier.

14.	Log out	Provides the user to end his accessibility in the program.

7.2 Scenarios

Scenario 1

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.
5. Admin opens inventory module.
6. Admin views the information in inventory.
7. Admin after completing the task logs out.

Scenario 2

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.
5. Admin opens inventory module.
6. Admin add new product by entering needed information.
7. The information is entered correctly and database is updated.
8. Admin after completing the task logs out.

Scenario 3

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.

5. Admin opens inventory module.
6. Admin add new product by entering needed information.
7. An error message pops up informing that information is not correct
8. Admin corrects information and database is updated.
9. Admin after completing the task logs out.

Scenario 4

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.
5. Admin opens inventory module.
6. Admin add new product by entering needed information.
7. An error message pops up informing that there is unfilled information
8. Admin fix error and database is updated.
9. Admin after completing the task logs out.

Scenario 5

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.
5. Admin opens inventory module.
6. Admin clicks remove button and removes product.
7. After clicking save database is updated.
8. Admin after completing the task logs out.

Scenario 6

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.
5. Admin opens inventory module.
6. Admin clicks the update button and update information.
7. After clicking save database is updated.
8. Admin after completing the task logs out.

Scenario 7

1. User enter username and password. (as user correctly)
2. System checks username and password.
3. User is logged in.
4. User panel is showed.
5. User opens inventory module.
6. User views the information in inventory.
7. User after completing the task logs out.

Scenario 8

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. User is logged in.
4. User panel is showed.
5. User opens client module.
6. User views the information in client's module.
7. User after completing the task logs out.

Scenario 9

1. User enter username and password. (correctly)
2. System checks username and password.
3. User is logged in.
4. User panel is showed.
5. User opens client module.
6. User views the information in client's module.
7. User after completing the task logs out.

Scenario 10

1. User enter username and password. (correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.
5. Admin opens client module.
6. Admin register new client by entering needed information.
7. The information is entered correctly and database is updated.
8. Admin after completing the task logs out.

Scenario 11

1. User enter username and password. (correctly)
2. System checks username and password.
3. User is logged in.
4. User panel is showed.
5. User opens client module.
6. User register new client by entering needed information.
7. An error message pops up informing that information is not correct
8. User corrects information and database is updated.
9. User after completing the task logs out.

Scenario 12

1. User enter username and password. (correctly)

2. System checks username and password.
3. User is logged in.
4. User panel is showed.
5. User opens client module.
6. User register new client by entering needed information.
7. An error message pops up informing that there is unfilled information
8. User fix error and database is updated.
9. User after completing the task logs out.

Scenario 13

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.
5. Admin opens debts module.
6. Admin views information.
7. Admin after completing the task logs out.

Scenario 14

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.
5. Admin opens debts module.
6. Admin updates(delete) depts.
7. Admin after completing the task logs out.

Scenario 15

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.

5. Admin opens debts module.
6. Admin views information.
7. Admin print needed information.
8. User after completing the task logs out.

Scenario 16

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.
5. Admin opens reports module.
6. Admin generate monthly reports selected by date.
7. Admin after completing the task logs out.

Scenario 17

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.
5. Admin opens reports module.
6. Admin generate annual reports selected by date.
7. Admin after completing the task logs out.

Scenario 18

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.
5. Admin opens reports module.
6. Admin generate annual reports selected by date.

7. An error message pops up informing that there is incomplete information.
8. Admin fix the error.
9. Admin after completing the task logs out.

Scenario 19

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.
5. Admin opens reports module.
6. Admin generate annual reports selected by date.
7. Admin generate desired graphical chart.
8. Admin after completing the task logs out.

Scenario 20

1. User enter username and password. (correctly)
2. System checks username and password.
3. User is logged in.
4. User panel is showed.
5. User opens transactions module.
6. User add new transaction.
7. User enter information for products he is about to sell.
8. User saves information.
9. Uses after completing the task logs out.

Scenario 21

1. User enter username and password. (correctly)
2. System checks username and password.
3. User is logged in.
4. User panel is showed.

5. User opens transactions module.
6. User add new transaction.
7. User enter information for products he is about to sell.
8. User print information.
9. Uses after completing the task logs out.

Scenario 22

1. User enter username and password. (correctly)
2. System checks username and password.
3. User is logged in.
4. User panel is showed.
5. User opens transactions module.
6. User add new transaction.
7. User enter information for products he is about to sell.
8. An error pops up that product information is incorrect.
9. Uses after completing the task logs out.

Scenario 23

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.
5. Admin opens supplier's module.
6. Admin views information.
7. Admin after completing the task logs out.

Scenario 24

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.
5. Admin opens supplier's module.
6. Admin add new supplier information.
7. Admin after completing the task logs out.

Scenario 25

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.
5. Admin opens supplier's module.
6. Admin update supplier information.
7. Admin after completing the task logs out.

Scenario 26

1. User enter username and password. (as admin correctly)
2. System checks username and password.
3. Admin is logged in.
4. Admin panel is showed.
5. Admin opens supplier's module.
6. Admin delete supplier.
7. Admin after completing the task logs out.

7.3 Use Cases

Use case name: Log in

Overview: Provides the users to log in, by using their user name and password, in order to access the program. Employee's login is different from Admin's login.

Actor(s): Users (Admin), Users (Employee)

Precondition(s): Valid username and password.

Scenario flow

Main (success) flow:

1. Beta Business displays the log in to the users

2. User enters the username and password.

3. Beta Business checks the User’s username and password.

4. Beta Business displays the home page.

Alternate Flows-

1: Wrong username or password.

Beta Business displays an error message. The system displays the log in form again.

Post condition(s) User is logged in. They have access according to their position in the company.

Use case name: Add Products

Overview: Both types of users have access to this part of the program, and they have to add new products.

Actor(s): Users (Admin), Users (Employee)

Precondition(s): User must be logged in.

Scenario flow

Main (success) flow:

1. User logs in to Beta Business.

2. User goes to Inventory, and goes to add new tab.

3. User enters the new product's info

4. User clicks add new product.

Alternate Flows-

1: No valid data were put inside the boxes.

2. Some of the boxes were left blank

Beta Business displays an error message. The system displays the form to complete it again.

Post condition(s) Data is updated. Now user can access other functionalities.

Use case name: Add Clients

Overview: Both types of users have access to this part of the program, and they have to add new clients.

Actor(s): Users (Admin), Users (Employee)

Precondition(s): User must be logged in.

Scenario flow

Main (success) flow:

1. User logs in to Beta Business.

2. User goes to Clients, and goes to add new tab.

3. User enters the new Client's info

4. User clicks add new client.

Alternate Flows-

1: No valid data were put inside the boxes.

2. Some of the boxes were left blank

Beta Business displays an error message. The system displays the form to complete it again.

Post condition(s) Data is updated. Now user can access other functionalities.

Use case name: Add Debts

Overview: Both types of users have access to this part of the program, and they have to add new debts.

Actor(s): Users (Admin), Users (Employee)

Precondition(s): User must be logged in.

Scenario flow

Main (success) flow:

1. User logs in to Beta Business.

2. User goes to Debts, and goes to add new debt tab.

3. User enters the new Debt's info

4. User clicks add new debt.

Alternate Flows-

1: No valid data were put inside the boxes.

2. Some of the boxes were left blank

Beta Business displays an error message. The system displays the form to complete it again.

Post condition(s) Data is updated. Now user can access other functionalities.

Use case name: Delete Debts.

Overview: Both types of users have access to this part of the program, and they have to delete debts.

Actor(s): Users (Admin), Users (Employee)

Precondition(s): User must be logged in.

Scenario flow

Main (success) flow:

1. User logs in to Beta Business.

2. User goes to Clients.

3. User searches for the Client's info.

4. User check's if the client is a debitor in the company.

5. User deletes the debt he has if he has paid the required amount.

Alternate Flows-

1. No debts are in the company.

2. Client hasn't paid the required amount

Beta Business displays an error message.

Post condition(s) Data is updated. Now user can access other functionalities.

Use case name: Reports

Overview: This module is accessed only by the Admin. They can have a detailed information about business activity.

Actor(s): Users (Admin),

Precondition(s): User must be logged in.

Scenario flow

Main (success) flow:

1. User logs in to Beta Business.

2. User goes to Reports.

3. User chooses the date he want to check.

Alternate Flows-

1: No valid data were put inside the boxes.

2. Some of the boxes were left blank.

3. This date isn't registered yet.

Beta Business displays an error message. The system displays the form to complete it again.

Post condition(s) Data is updated. There will be shown the requested reports.

Use case name: Search Name

Overview: Both types of users have access to this part of the program, and they have to add new products.

Actor(s): Users (Admin), Users (Employee)

Precondition(s): User must be logged in.

Scenario flow

Main (success) flow:

1. User logs in to Beta Business.

2. User goes to Clients, and goes to search tab.

3. User enters the name.

4. User clicks search.

Alternate Flows-

1: No valid data were put inside the boxes.

2. Some of the boxes were left blank

Beta Business displays an error message. The system displays the form to complete it again.

Post condition(s) Data is uploaded. Now user can see information for this person.

Use case name: Search Date

Overview: Both types of users have access to this part of the program, and they have to add new products.

Actor(s): Users (Admin), Users (Employee)

Precondition(s): User must be logged in.

Scenario flow

Main (success) flow:

1. User logs in to Beta Business.

2. User goes to Transaction, and goes to date tab.

3. User enters the Date that he wants information.

4. User clicks search.

Alternate Flows-

1: No valid data were put inside the boxes.

2. Some of the boxes were left blank

3. Date may be invalid.

Beta Business displays an error message. The system displays the form to complete it again.

Post condition(s) Data is showed. Now user can see all transactions of that day.

Use case name: Add Transaction

Overview: Both types of users have access to this part of the program, and they have to add new products.

Actor(s): Users (Admin), Users (Employee)

Precondition(s): User must be logged in.

Scenario flow

Main (success) flow:

1. User logs in to Beta Business.

2. User goes to transaction, and goes to add new tab.

3. User enters the new transaction's info

4. User clicks add new transaction.

Alternate Flows-

1: No valid data were put inside the boxes.

2. Some of the boxes were left blank

Beta Business displays an error message. The system displays the form to complete it again.

Post condition(s) Data is updated. Now user can access other functionalities

Use case name: Add Supplier

Overview: Both types of users have access to this part of the program, and they have to add new products.

Actor(s): Users (Admin), Users (Employee)

Precondition(s): User must be logged in.

Scenario flow

Main (success) flow:

1. User logs in to Beta Business.

2. User goes to suppliers, and goes to add new tab.

3. User enters the new supplier's info

4. User clicks add new suplier.

Alternate Flows-

1: No valid data were put inside the boxes.

2. Some of the boxes were left blank

Beta Business displays an error message. The system displays the form to complete it again.

Post condition(s) Data is updated. Now user can access other functionalities

Use case name: Update Supplier

Overview: Both types of users have access to this part of the program, and they have to add new products.

Actor(s): Users (Admin), Users (Employee)

Precondition(s): User must be logged in.

Scenario flow

Main (success) flow:

1. User logs in to Beta Business.

2. User goes to supplier, and goes to update tab.

3. User enters the new suppliers info

4. User clicks update information.

Alternate Flows-

1: No valid data were put inside the boxes.

2. Some of the boxes were left blank

Beta Business displays an error message. The system displays the form to complete it again.

Post condition(s) Data is updated. Now user can access other functionalities

Use case name: Delete Supplier

Overview: Both types of users have access to this part of the program, and they have to add new products.

Actor(s): Users (Admin), Users (Employee)

Precondition(s): User must be logged in.

Scenario flow

Main (success) flow:

1. User logs in to Beta Business.

2. User goes to supplier, and goes to delete tab.

3. User clicks delete supplier.

Alternate Flows-

1: No valid data were put inside the boxes.

2. Some of the boxes were left blank

Beta Business displays an error message. The system displays the form to complete it again.

Post condition(s) Data is updated. Now user can access other functionalities

Use case name: Log Out

Overview: Both types of users have access to this part of the program, and they have to add new products.

Actor(s): Users (Admin), Users (Employee)

Precondition(s): User must be logged in.

Scenario flow

Main (success) flow:

1. User logs in to Beta Business.

2. User clicks log out.

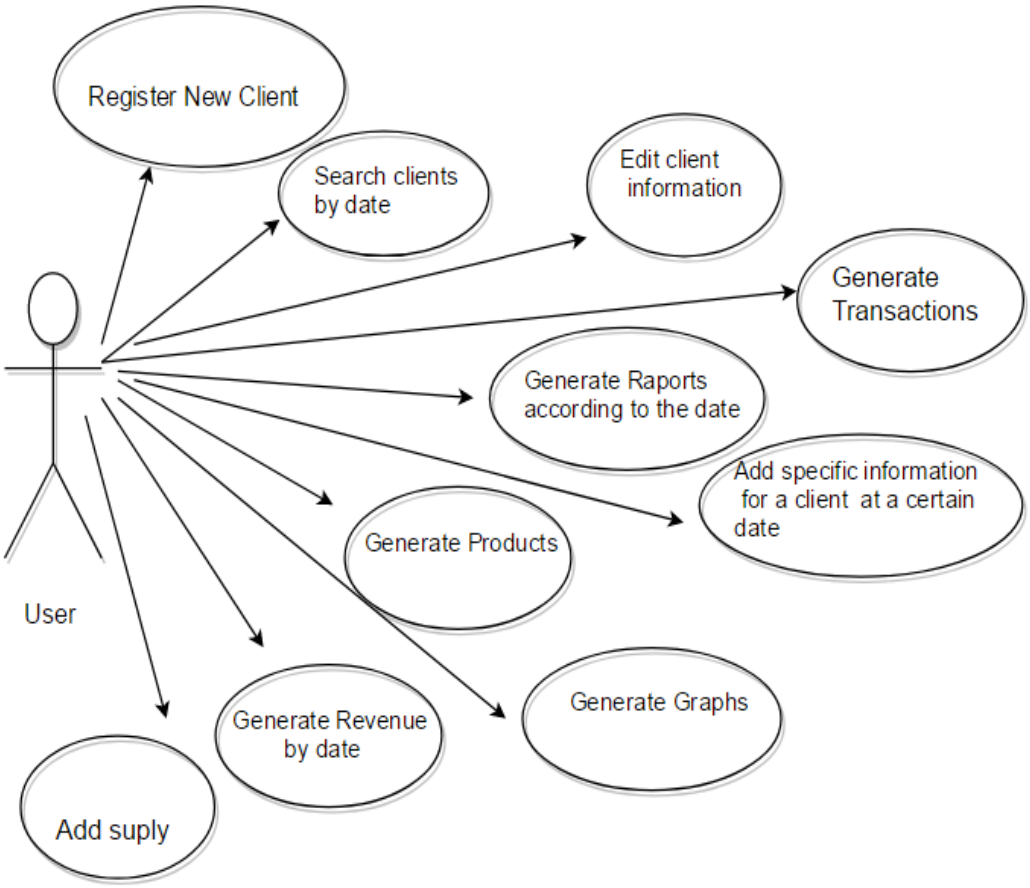
Alternate Flows-

No alternative Flow

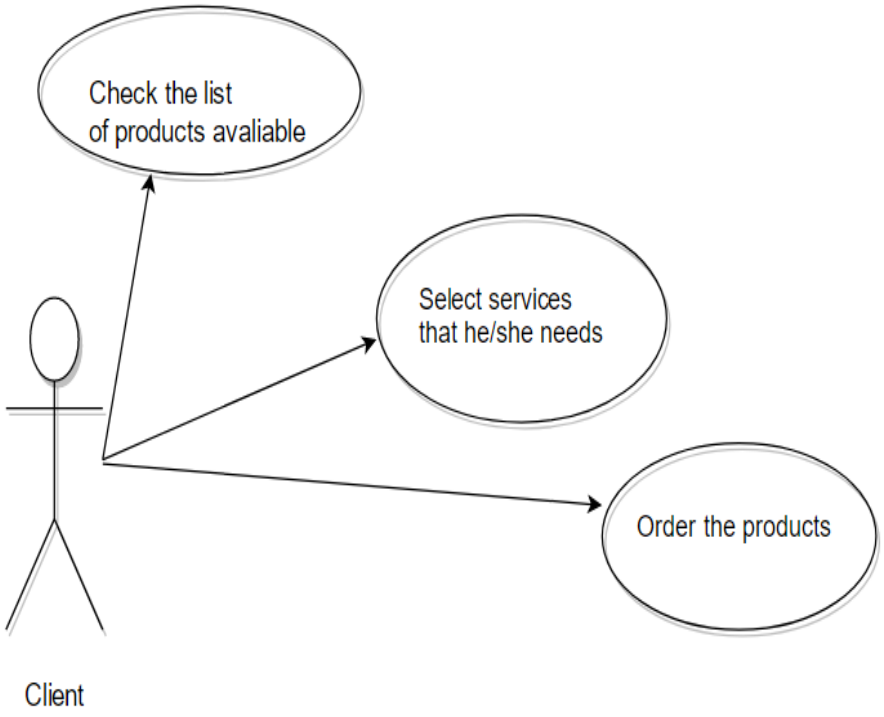
Post condition(s) User is Logged out. Database is updated and saved.

8. Diagrams

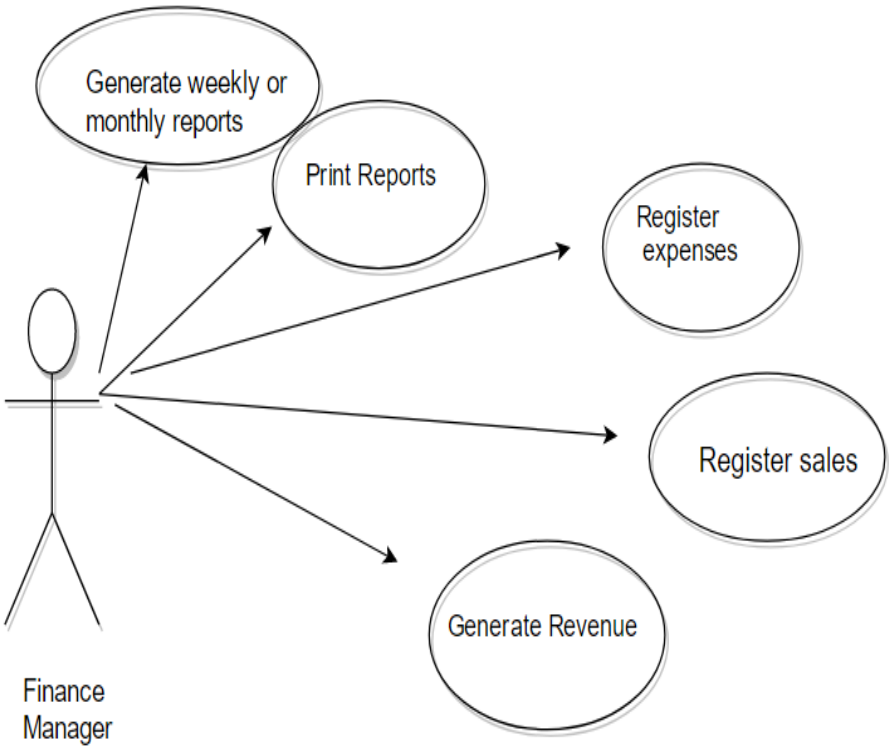
8.1 Use Case Diagrams



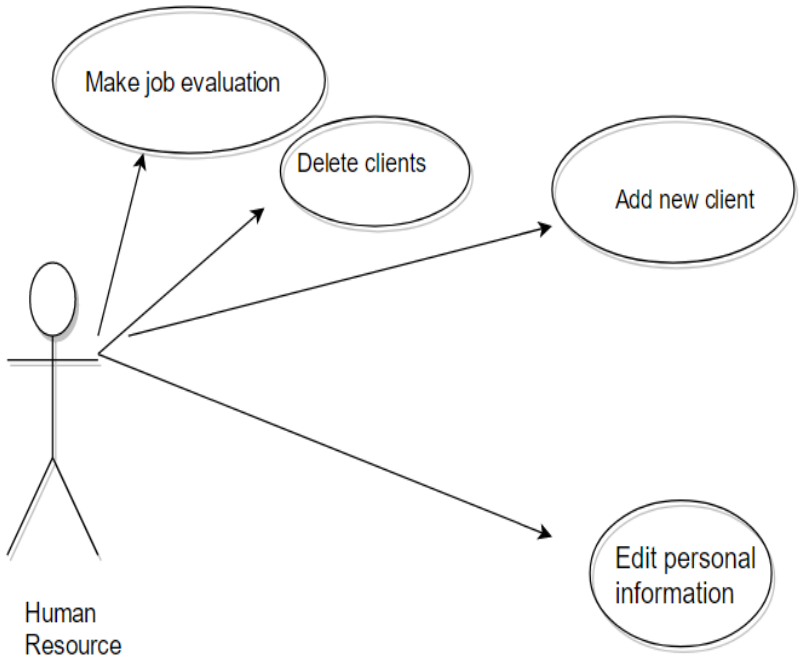
Admin Use Case



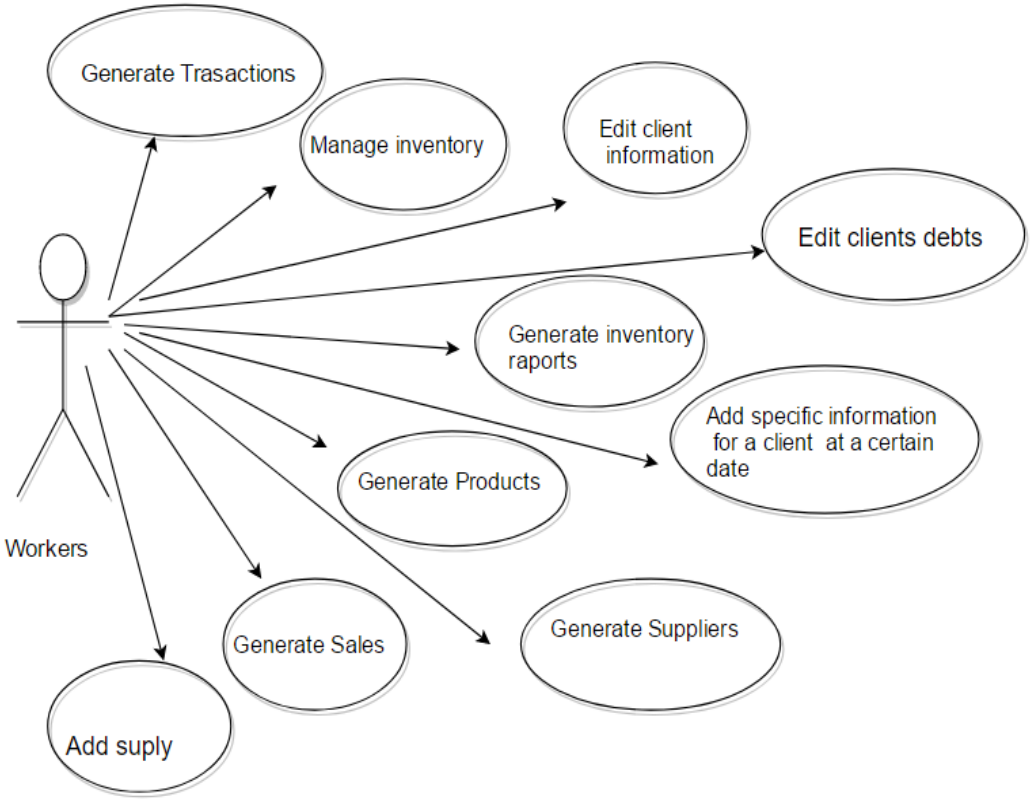
Client Use Case



Finance Manager Use Case

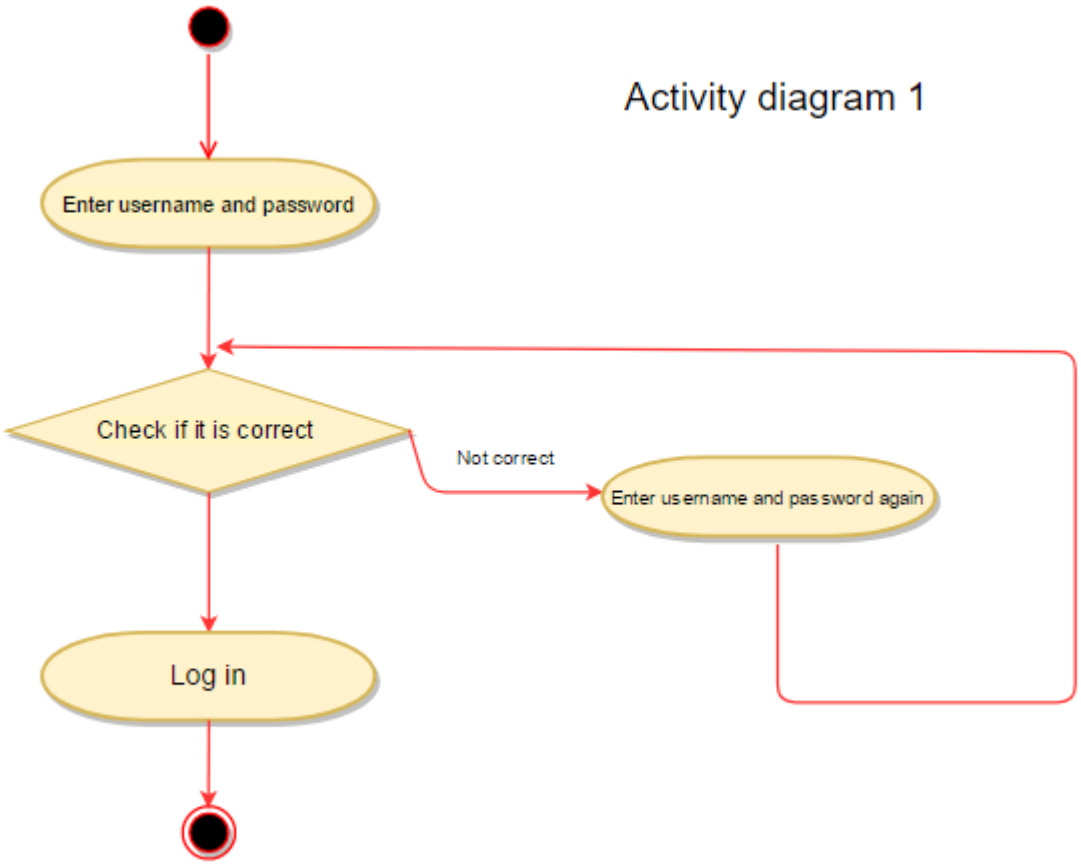


User Use Case

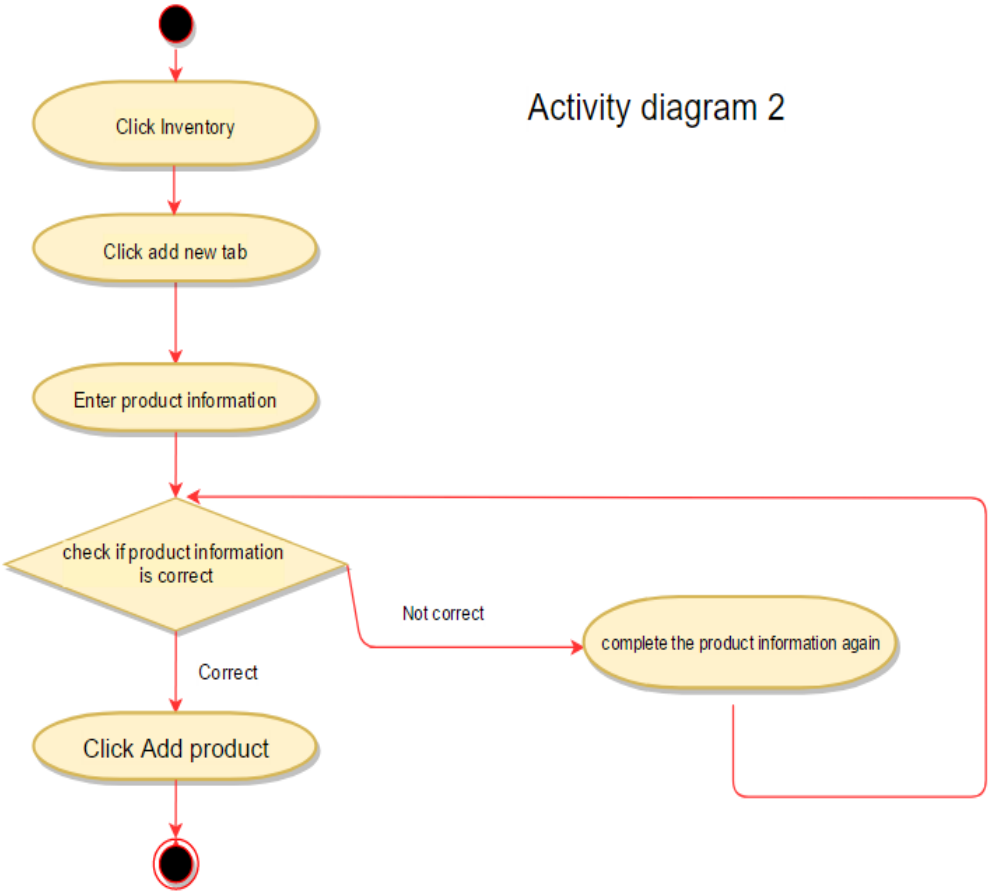


Worker Use Case

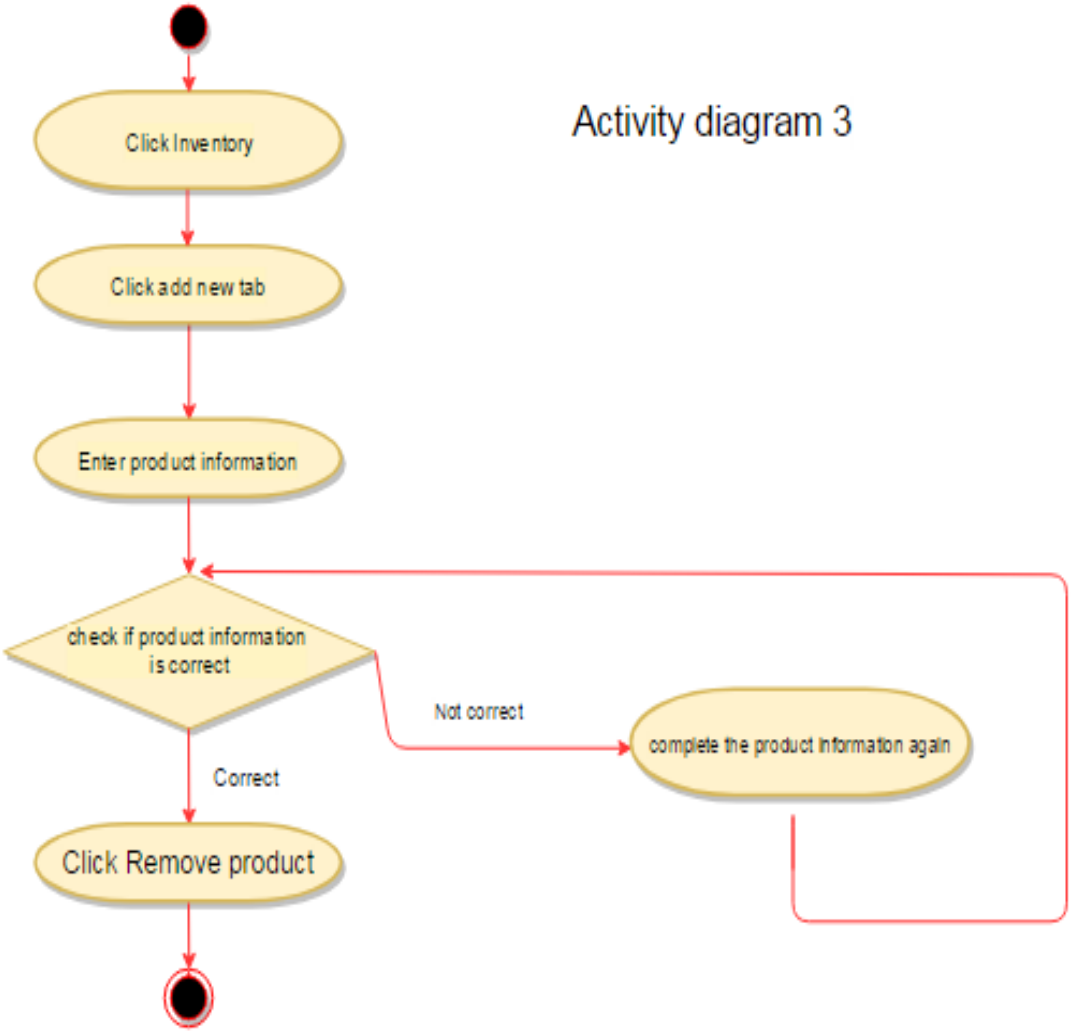
8.2 Activity Diagrams



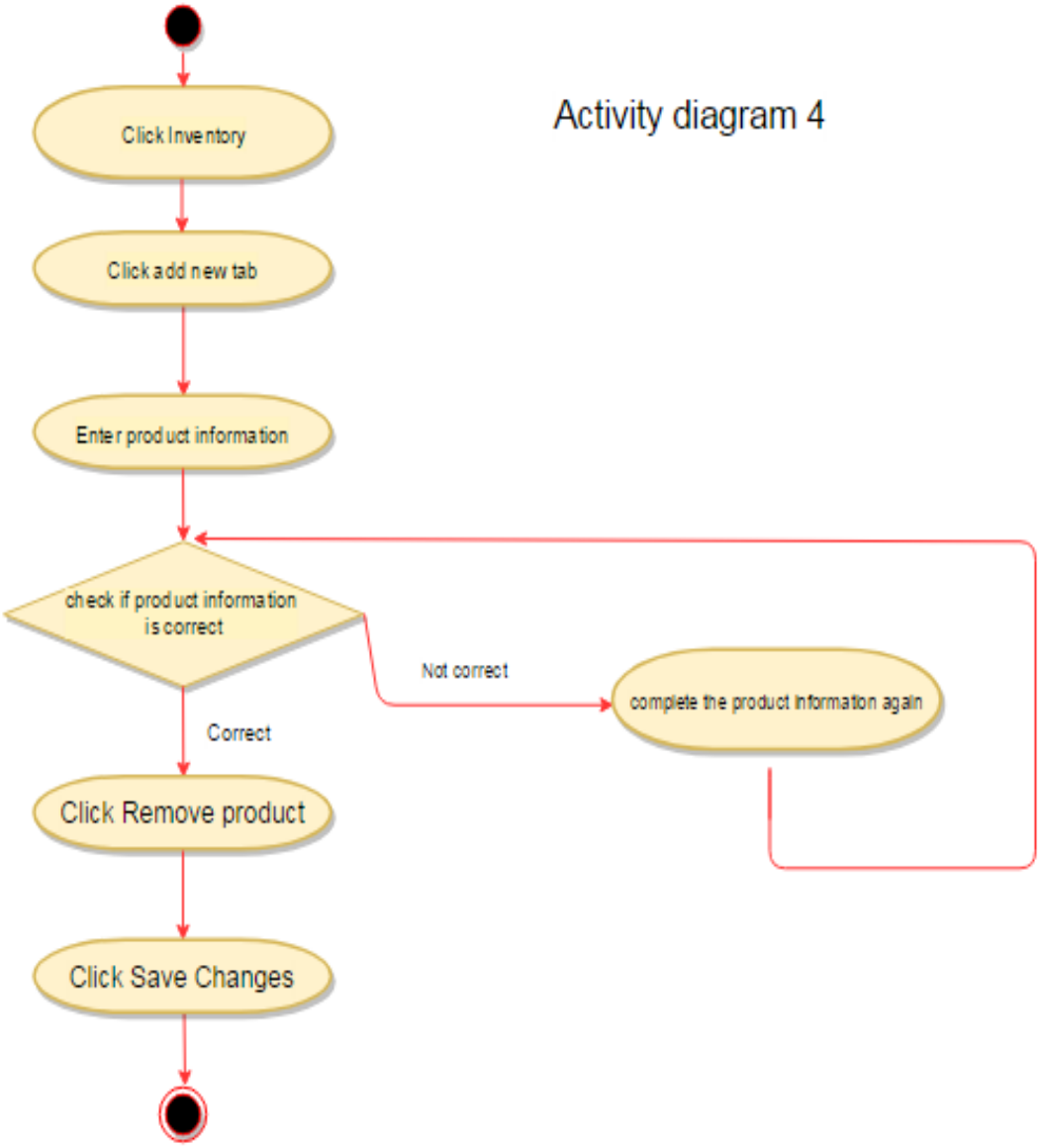
(Use Stories 1)



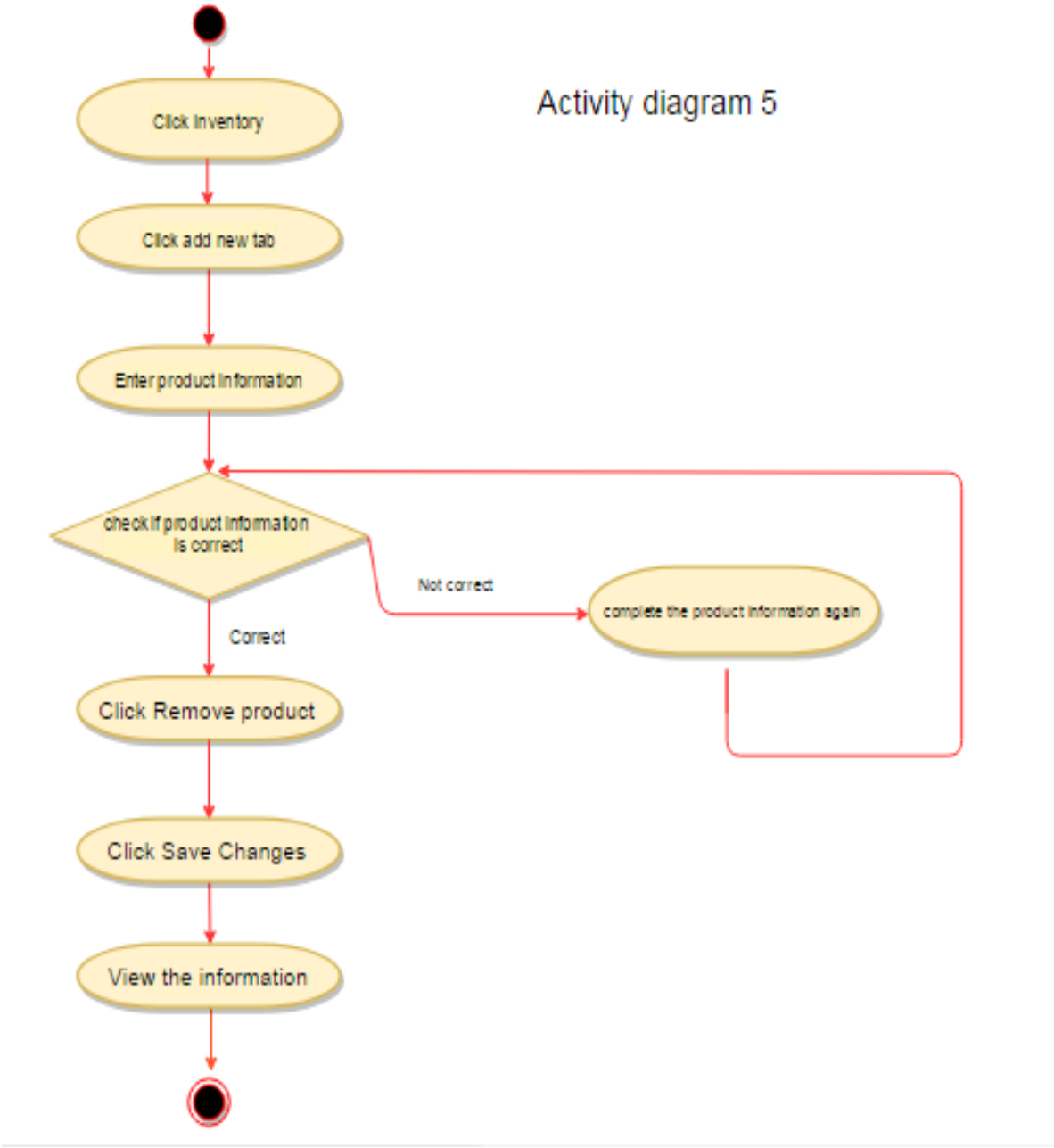
(Use Story 2, Scenario 1, 2, 3, 4)



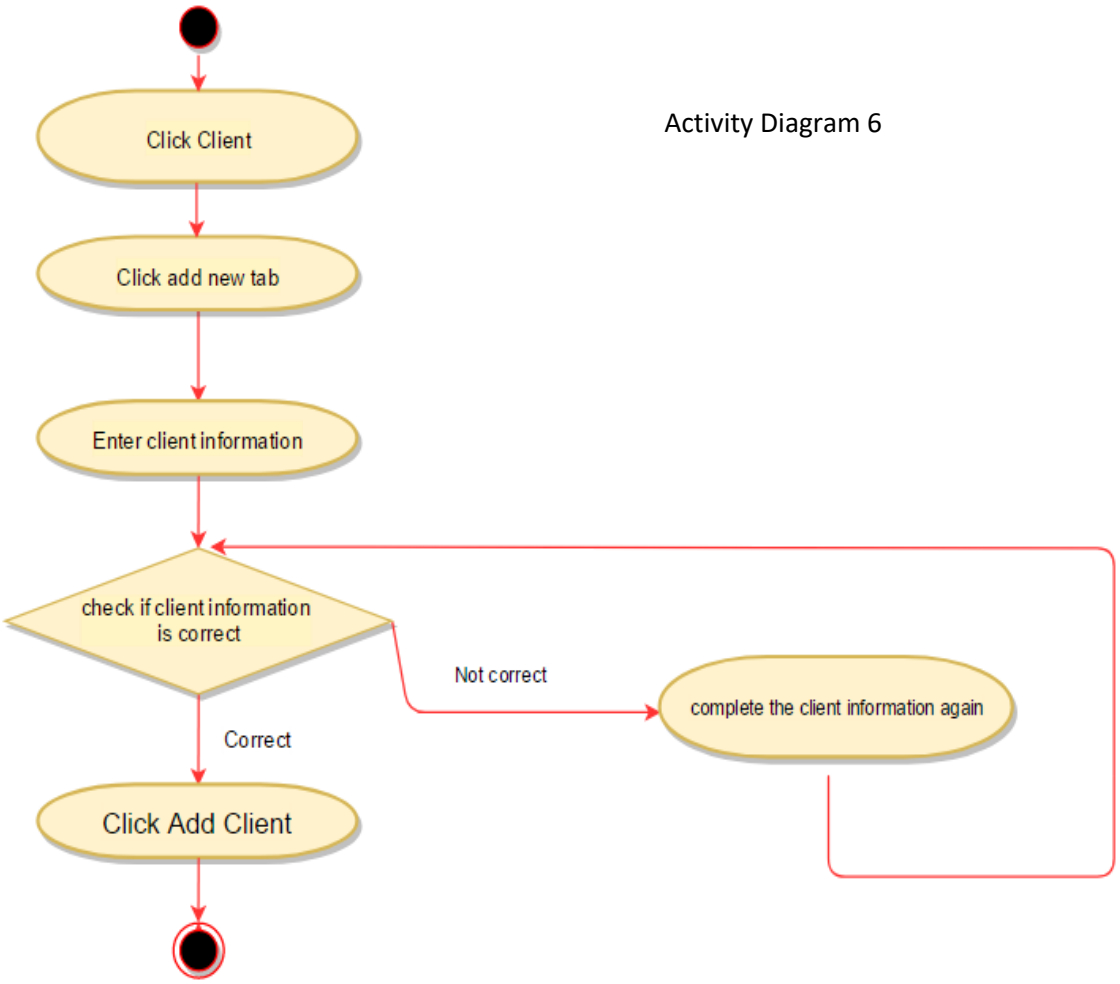
(Use Story 3, Scenario 5)



(Use Story 3, Scenario 6)

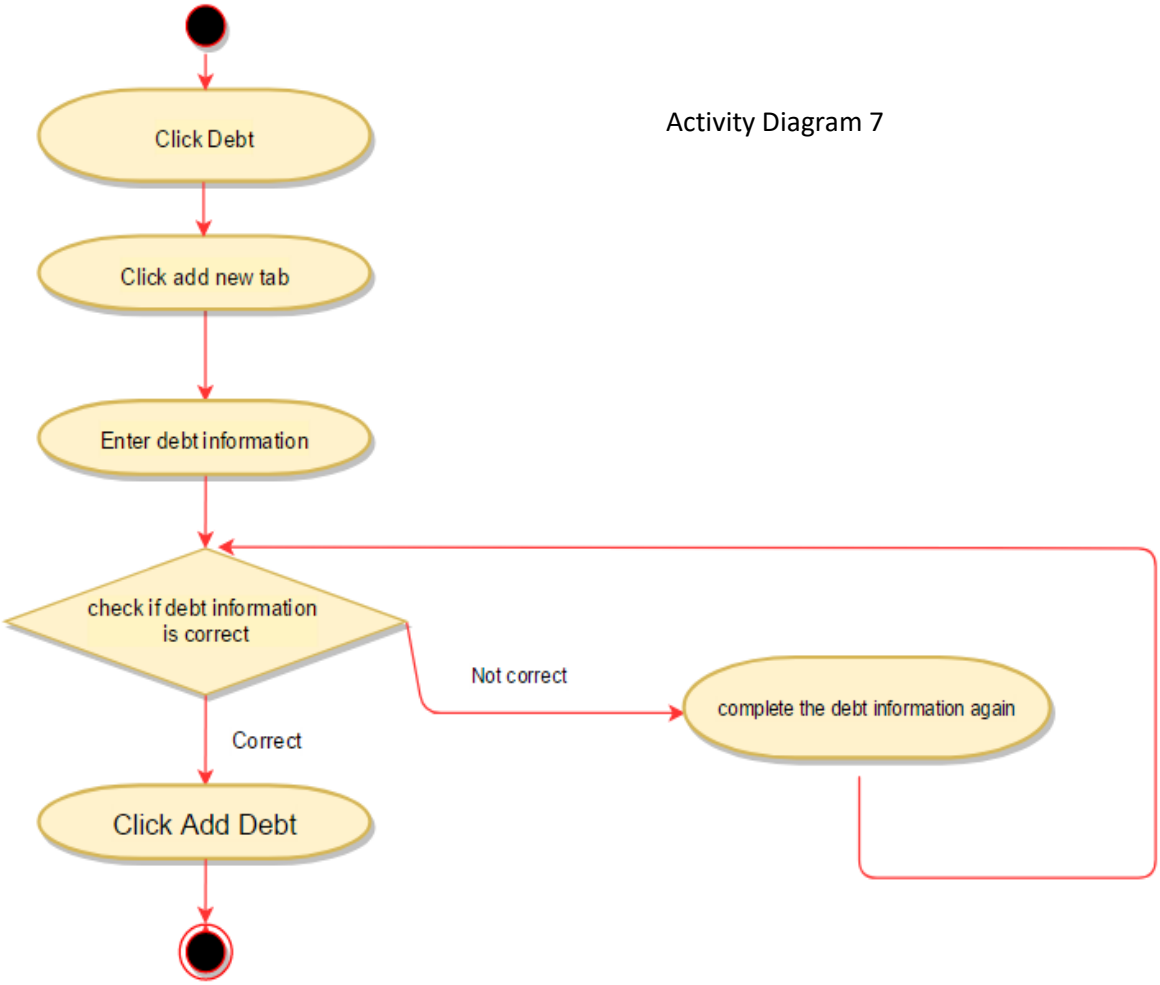


(Use Story 3, Scenario 7)



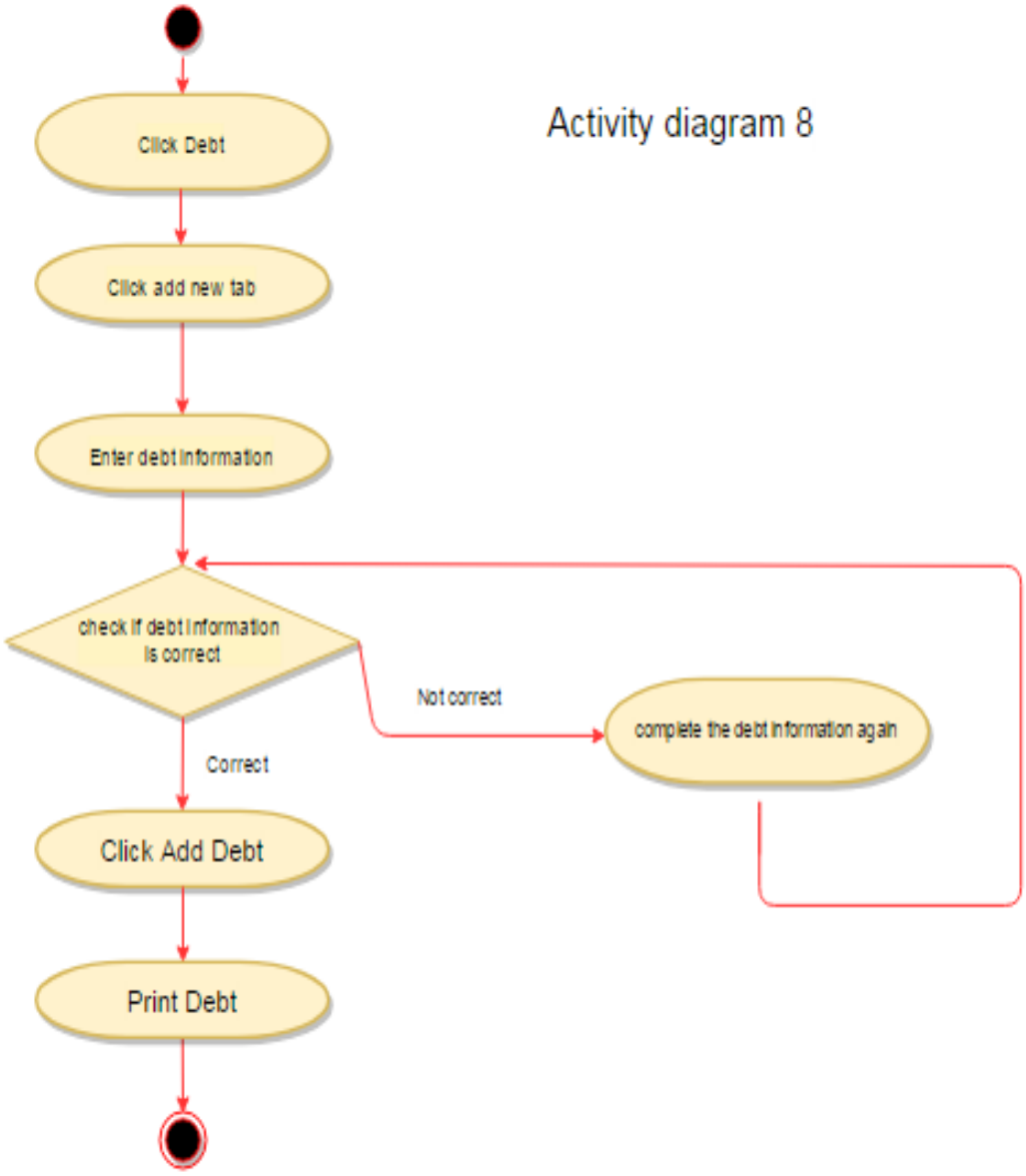
Activity Diagram 6

(Use Story 4, Scenario 8, 9, 10, 11, 12)

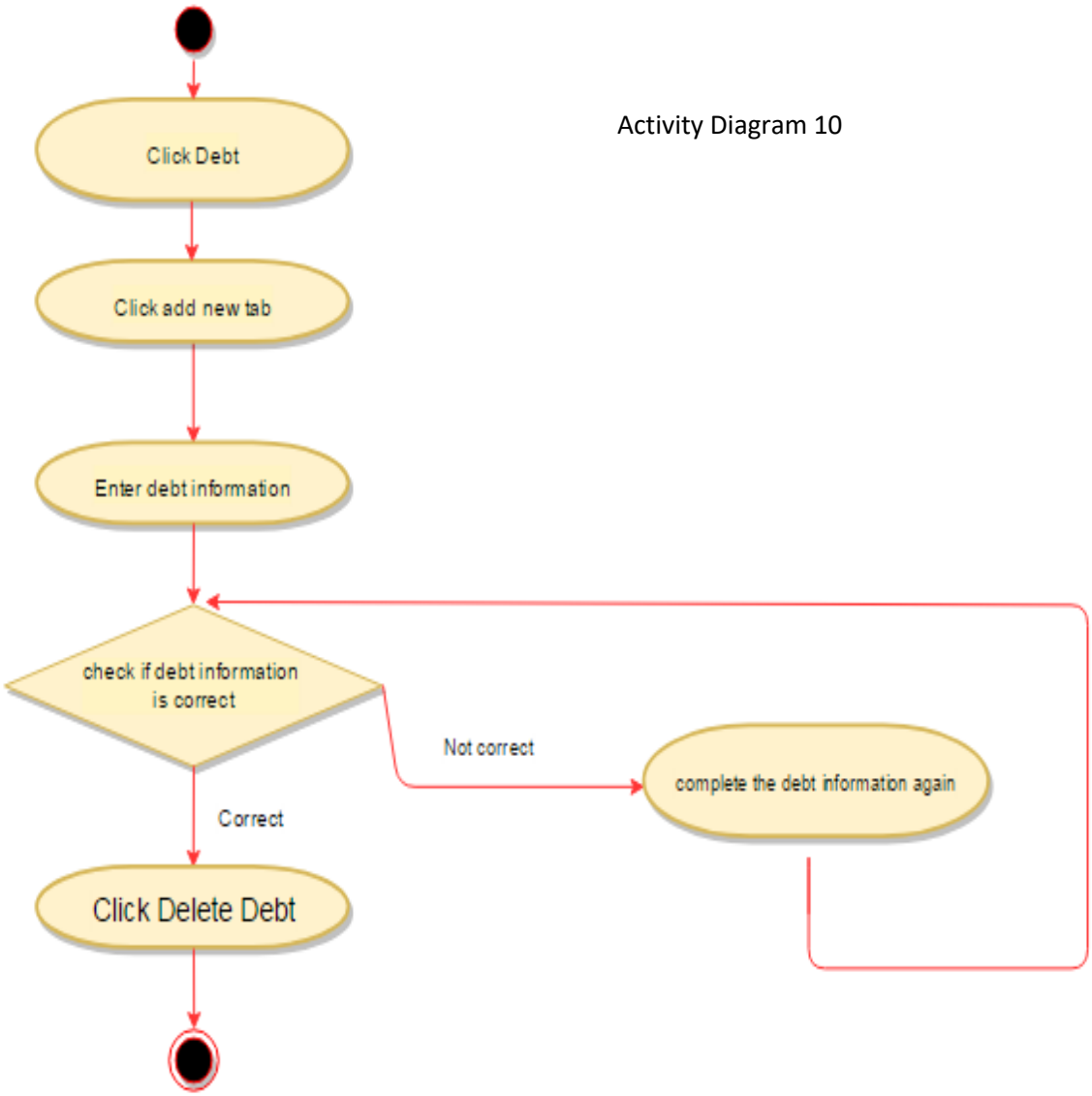


Activity Diagram 7

(Use Story 5, Scenario 13)

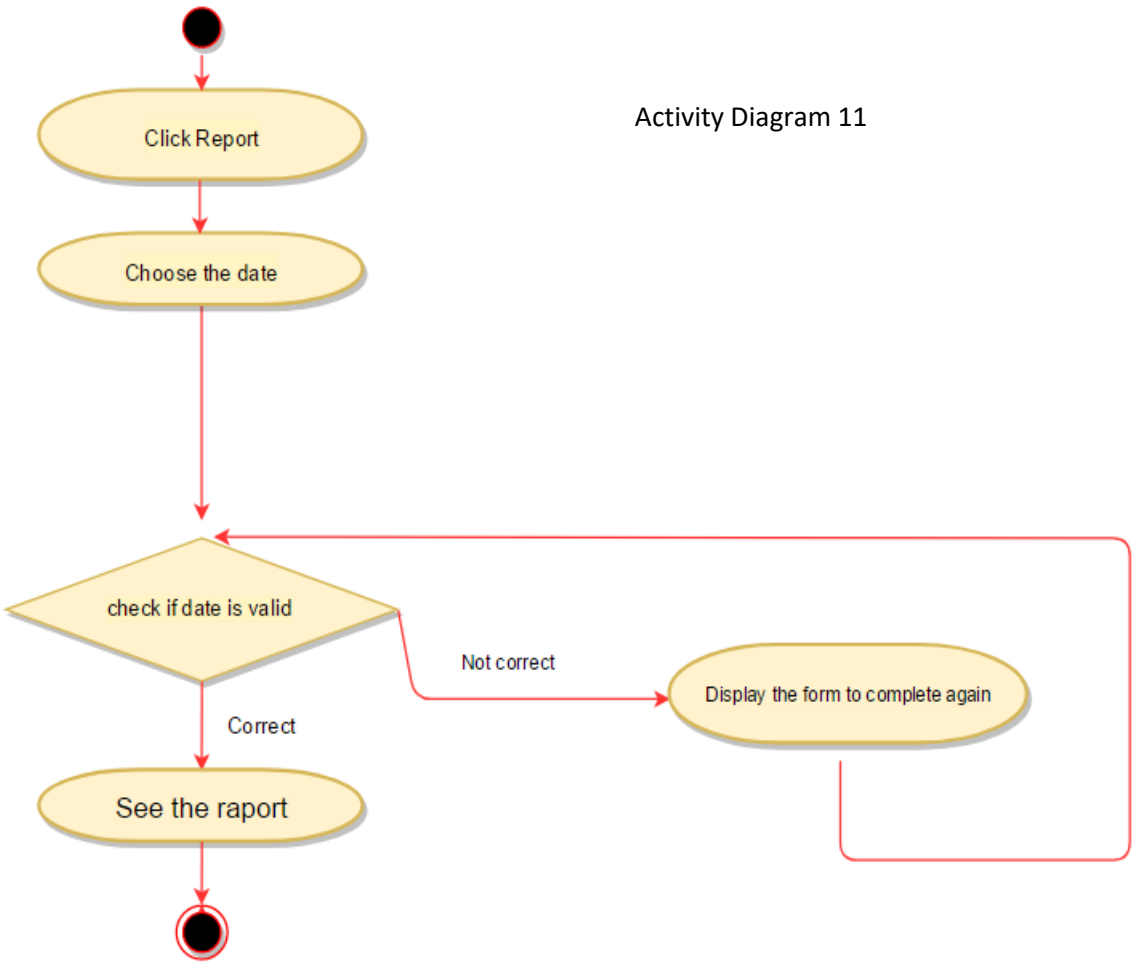


(Use Story 5, Scenario 15)



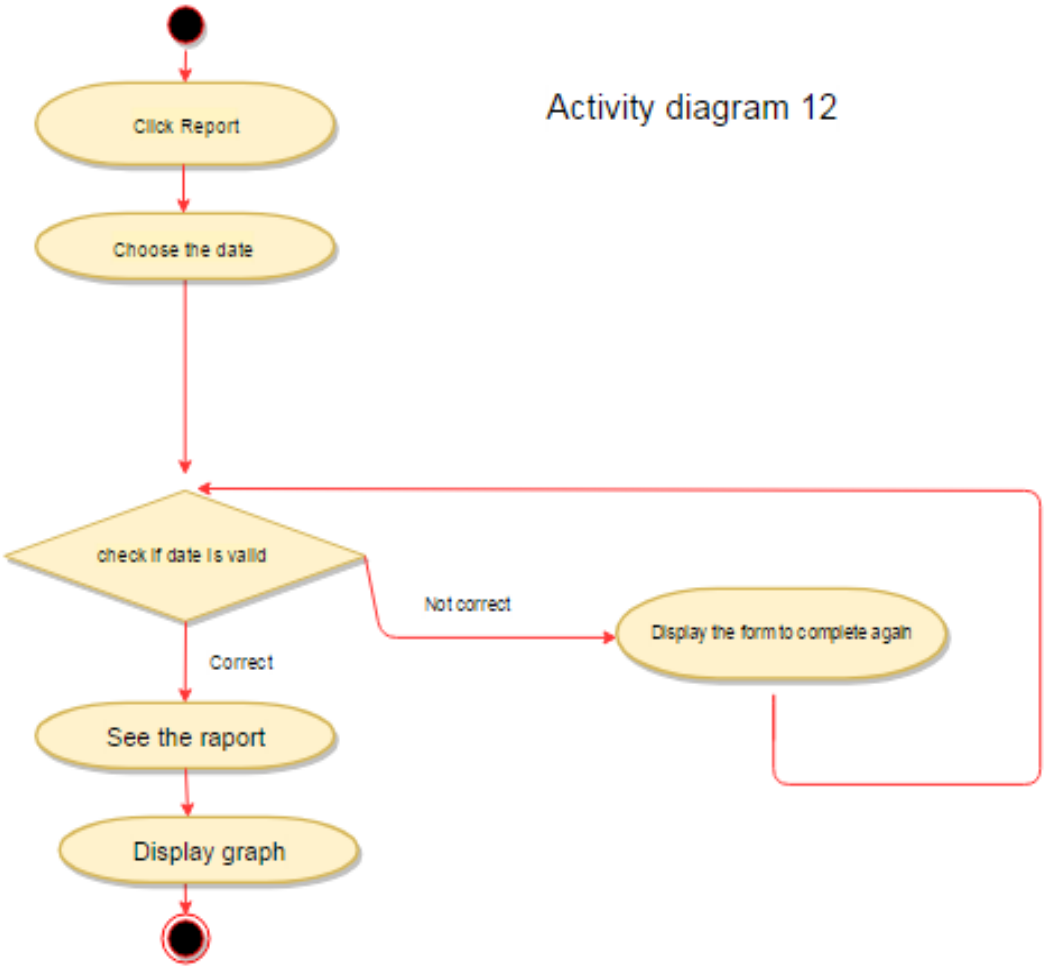
Activity Diagram 10

(Use Story 6, Scenario 14)

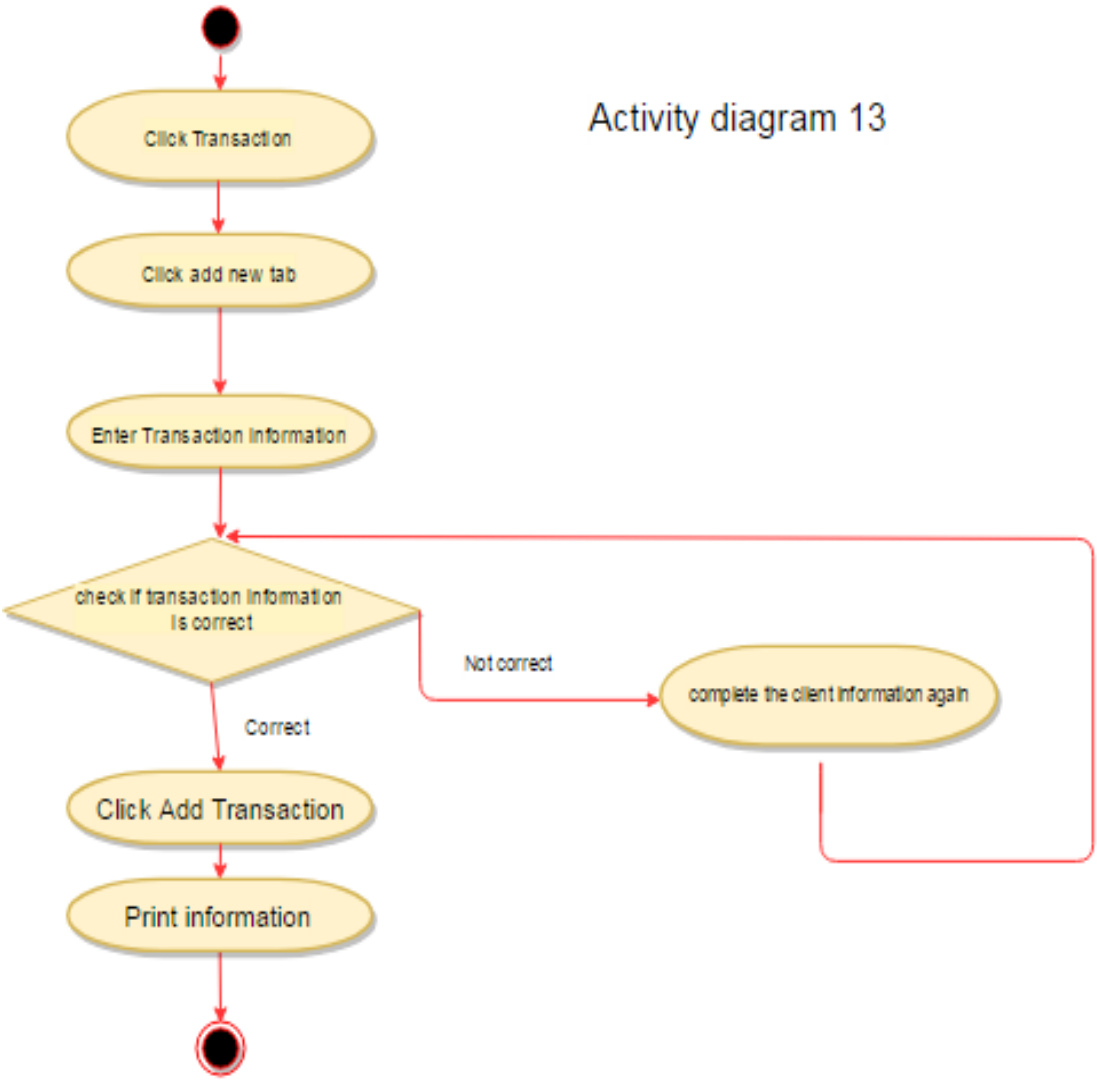


Activity Diagram 11

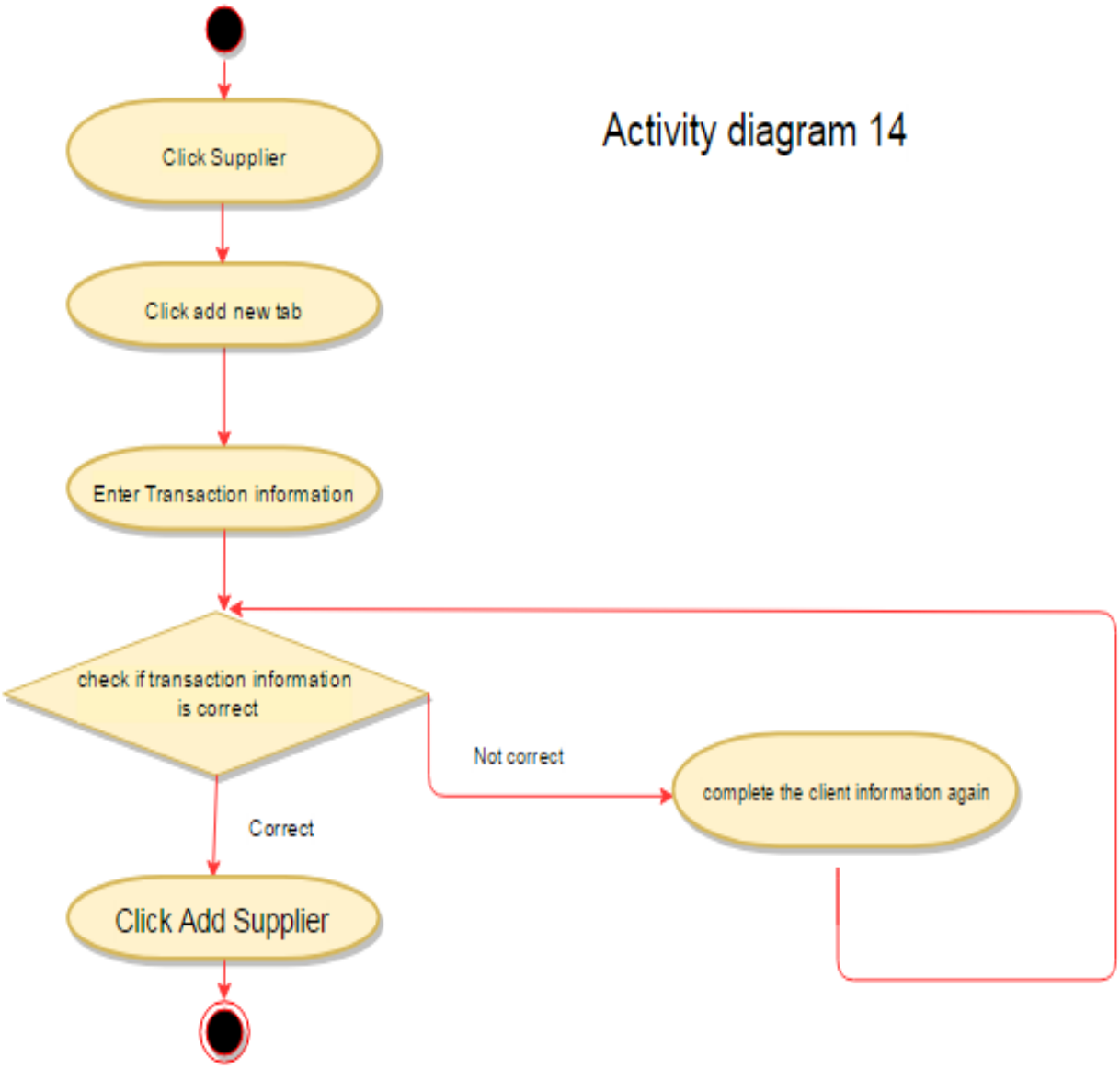
(Use Story 7, 8, 9 Scenario 16, 17, 18)



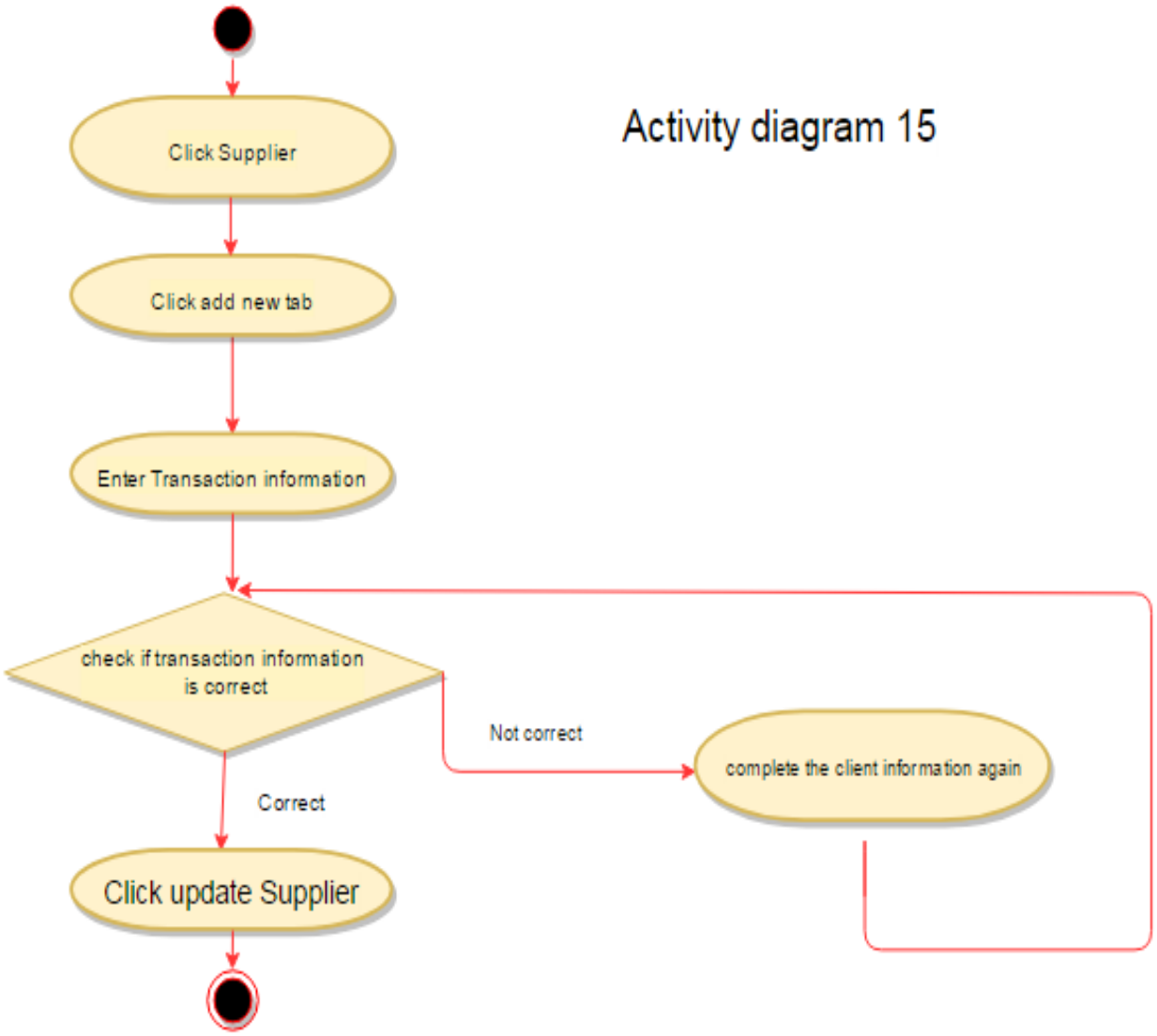
(Use Story 7, 8, 9 Scenario 19)



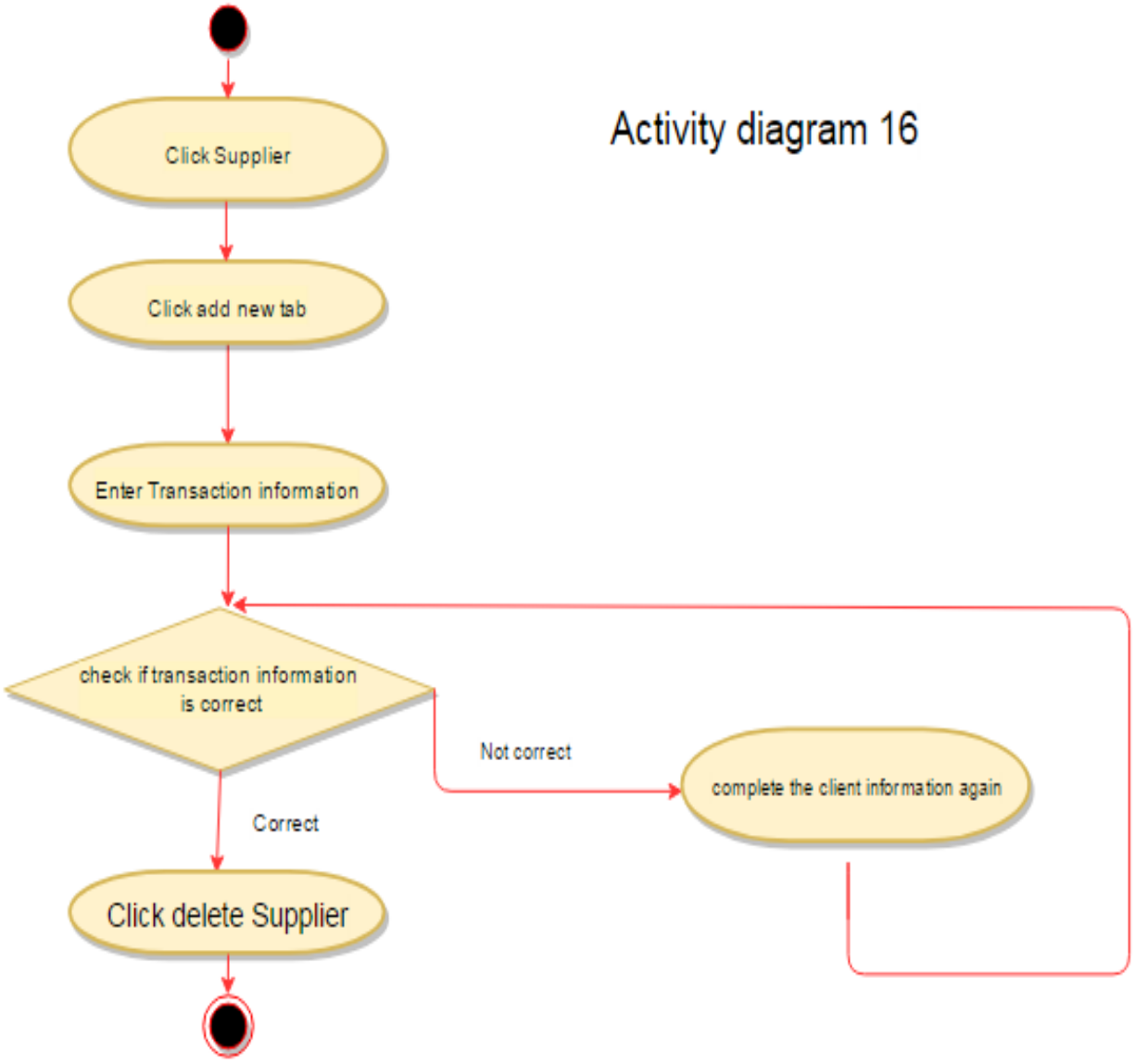
(Use Story 10, Scenario 20, 21, 22)



(Use Story 11, Scenario 23, 24)

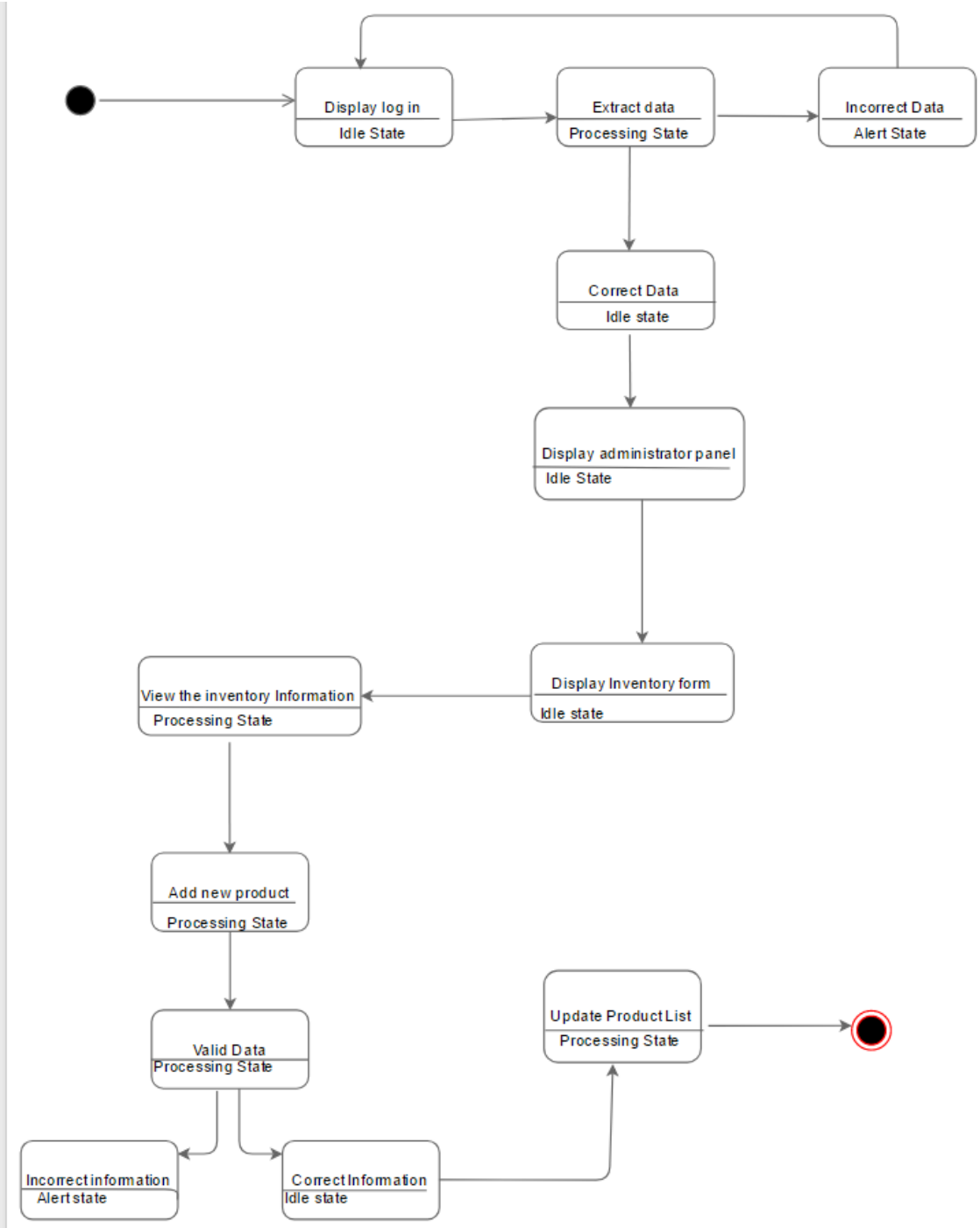


(Use Story 12, Scenario 25)

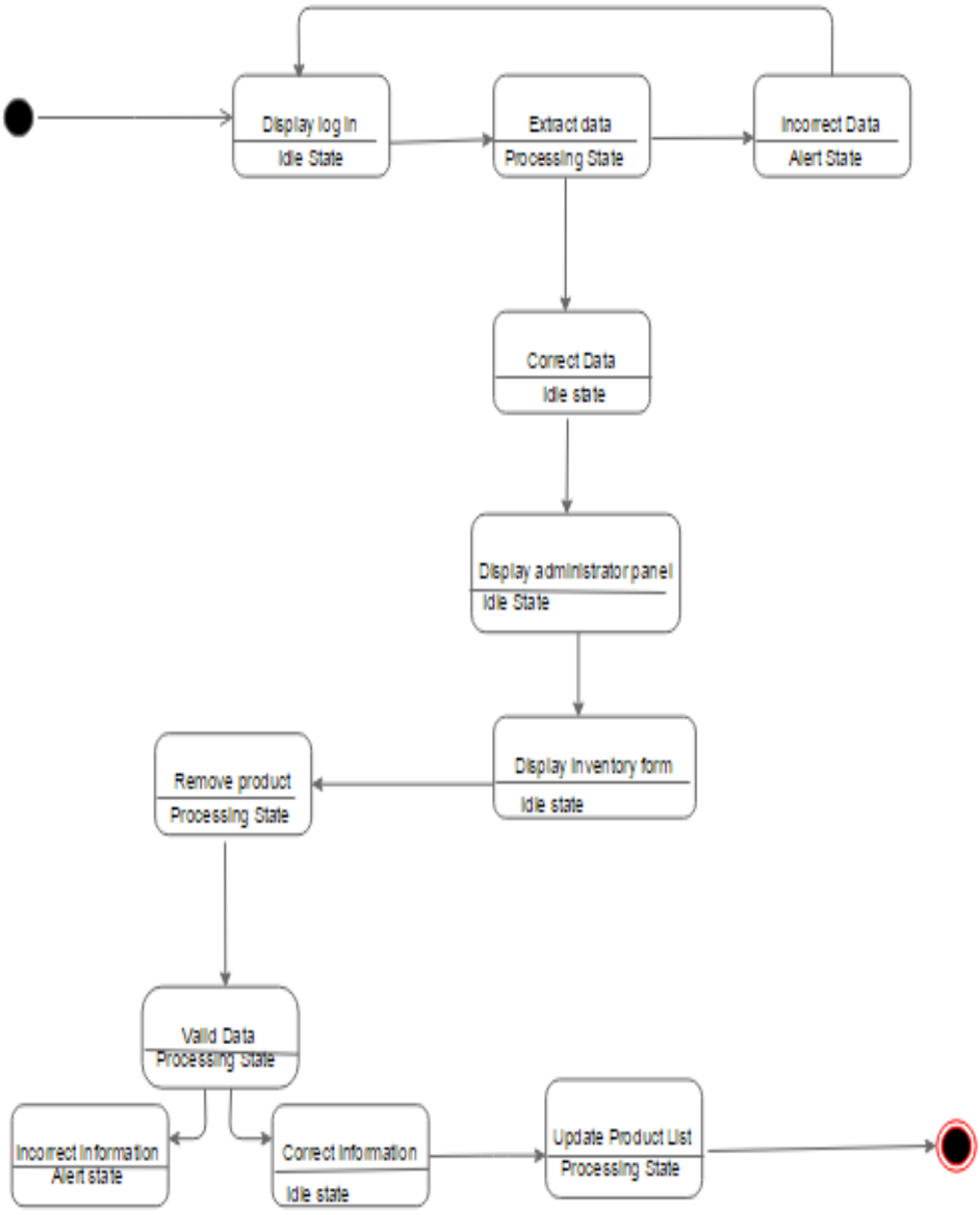


(Use Story 13, Scenario 26)

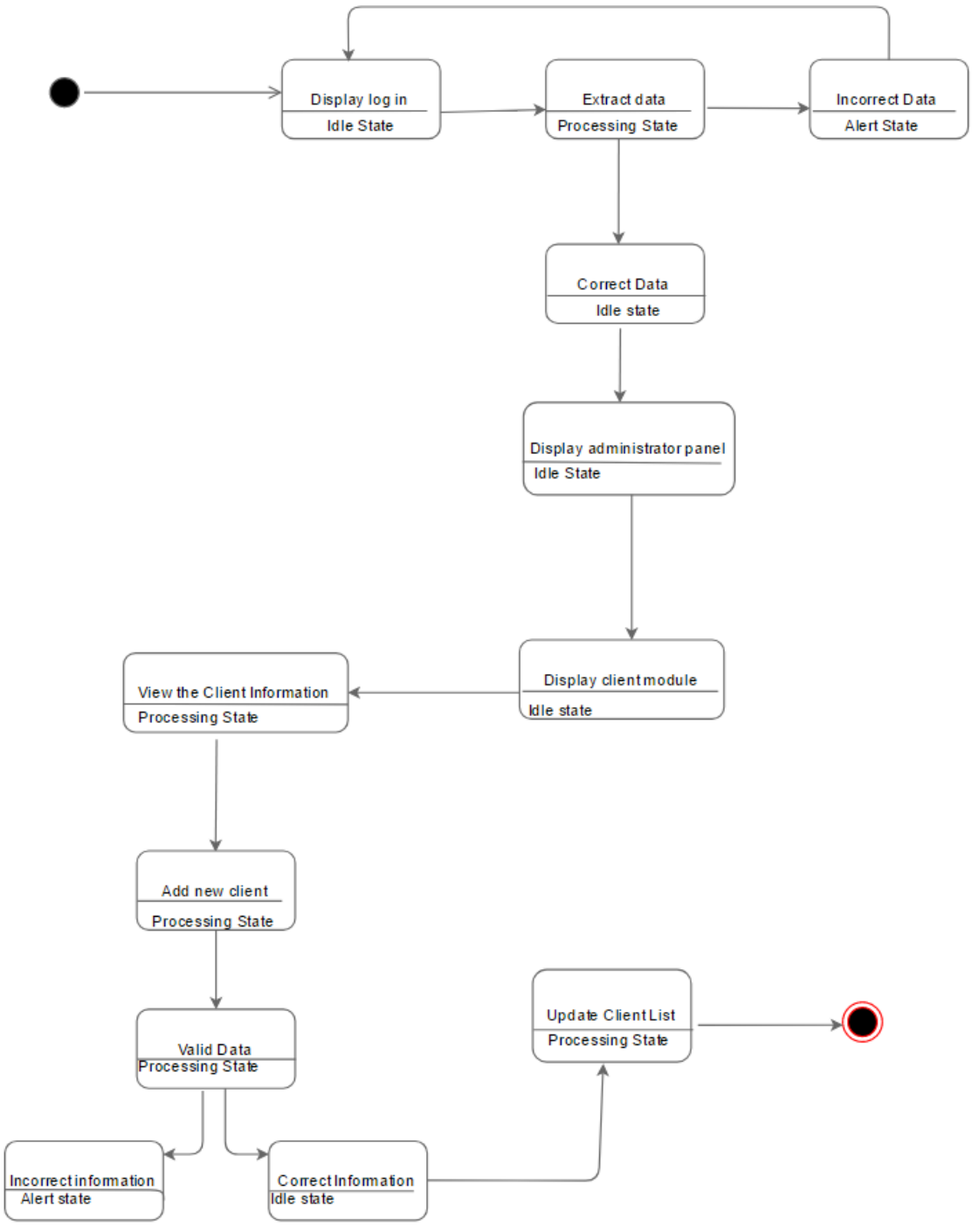
8.3 State Diagrams



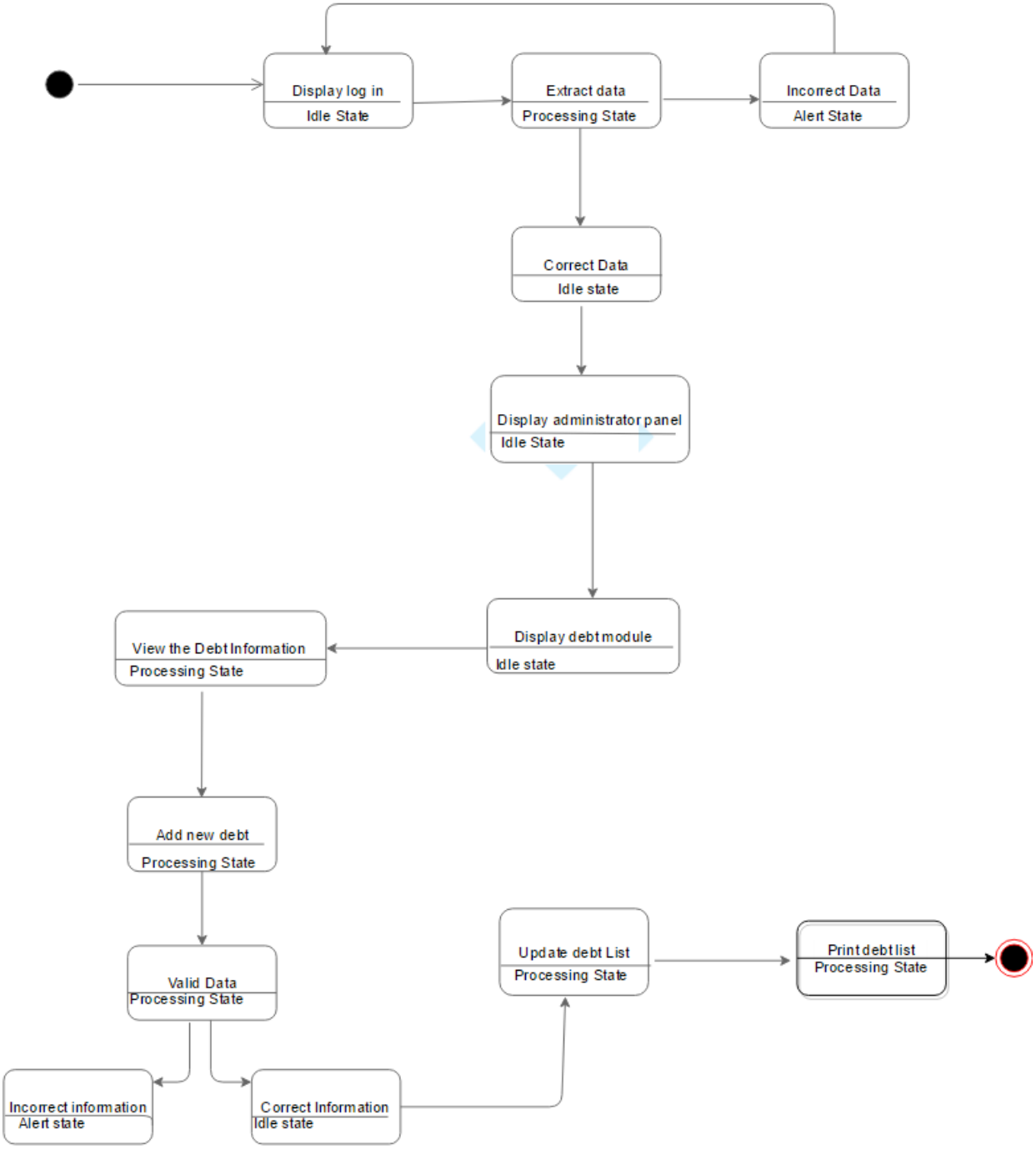
State Diagram 1(Use Story 1, 2, Scenario 1, 2, 3, 4 , 7)



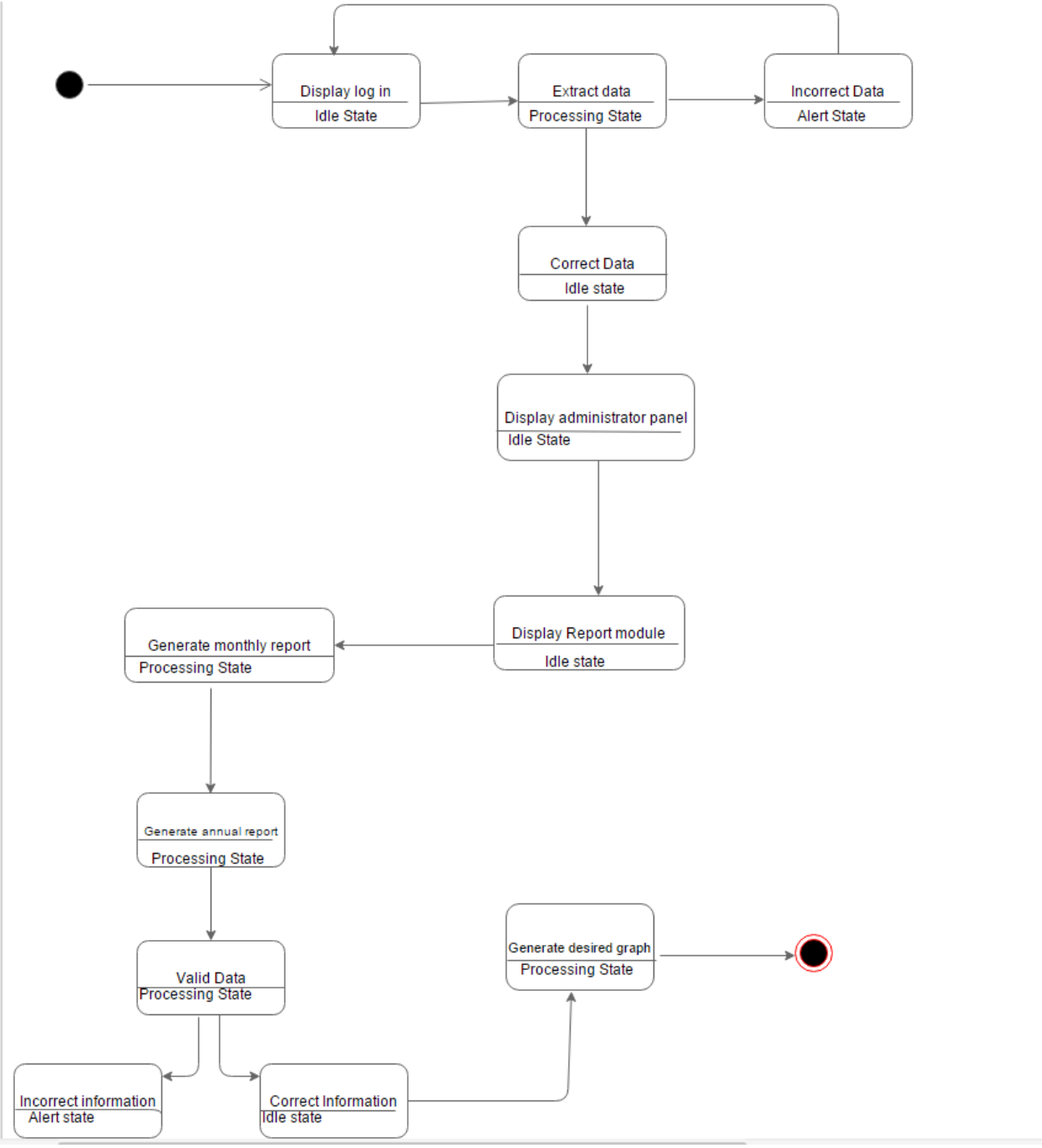
State Diagram 2(Use Story 3, Scenario 5, 6)



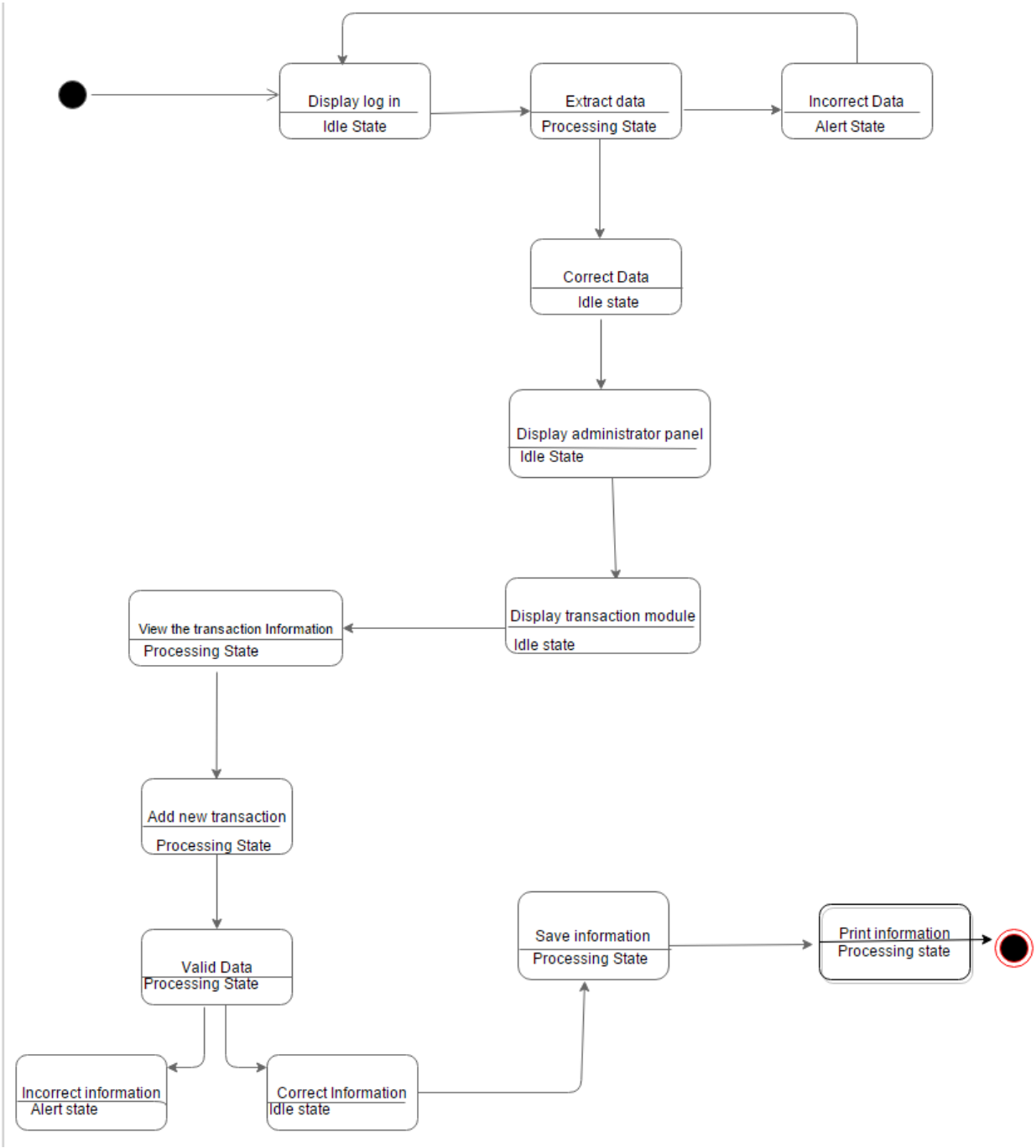
State Diagram 3(Use Story 3, Scenario 8, 9, 10, 11, 12)



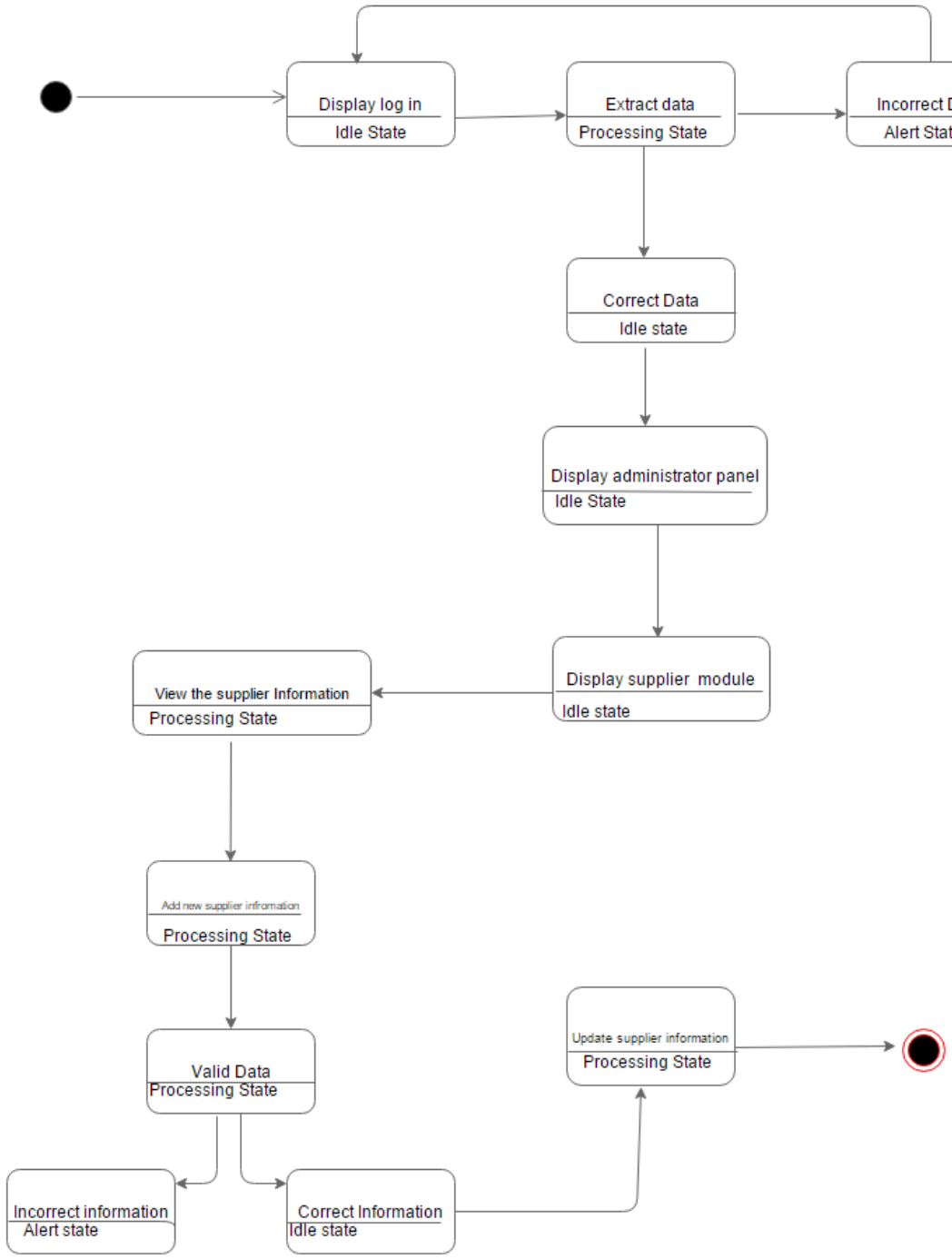
State Diagram 4(Use Story 5, Scenario 13, 14, 15)



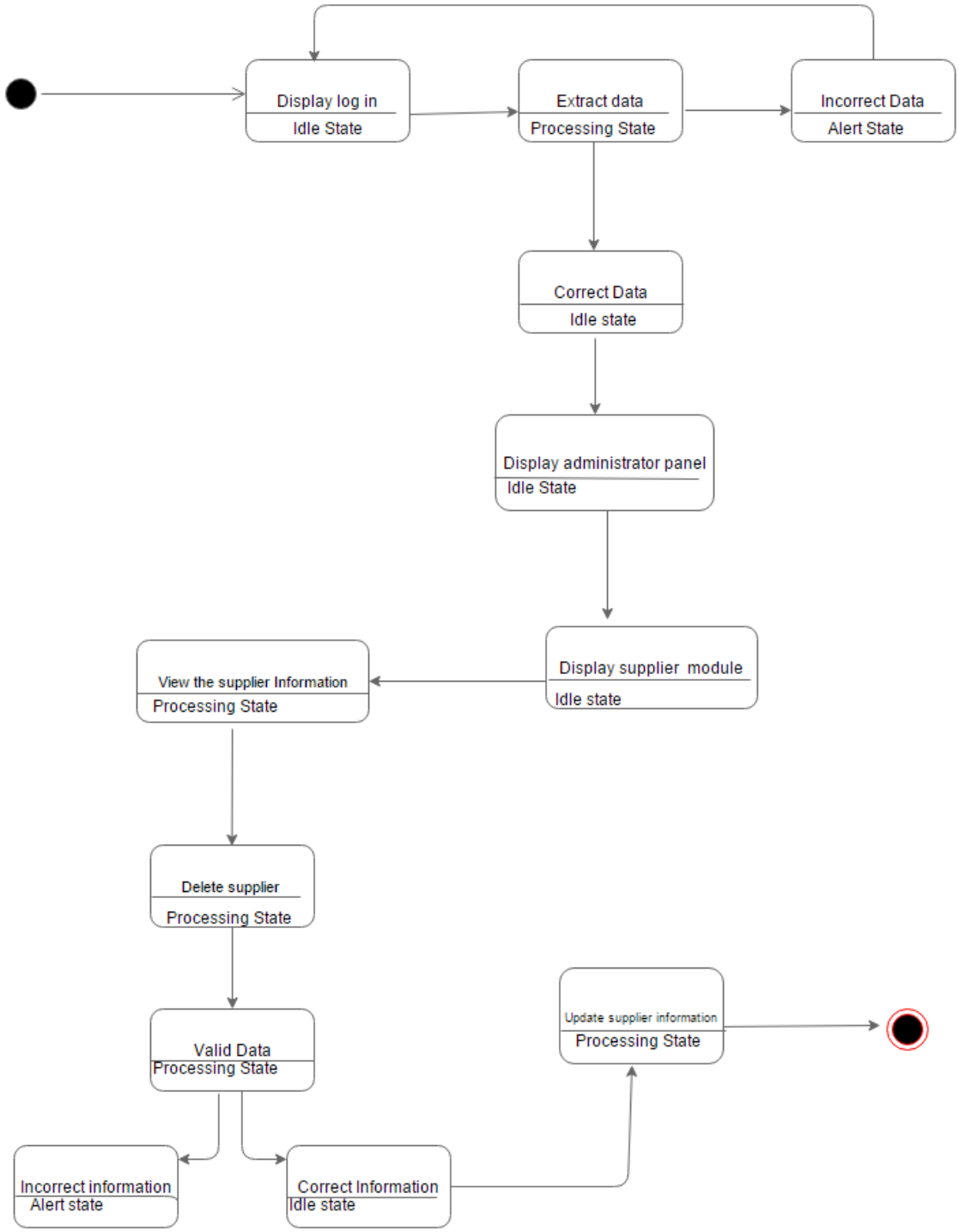
State Diagram 5(Use Story 7, 8, 9 Scenario 16, 17, 18.19)



State Diagram 6(Use Story 10, Scenario 20, 21, 22)

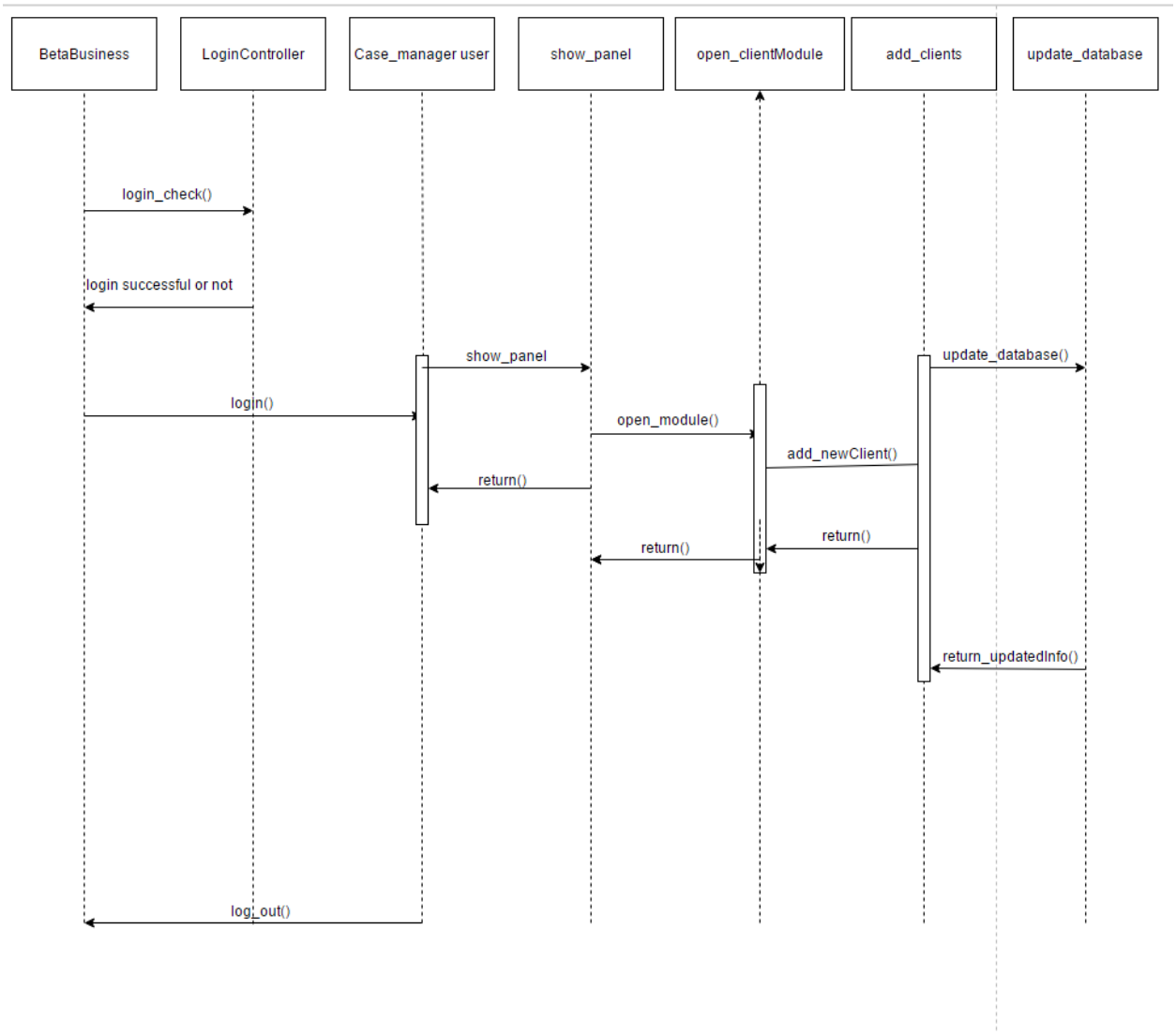


State Diagram 7(Use Story 11, 12, Scenario 23, 24, 25)

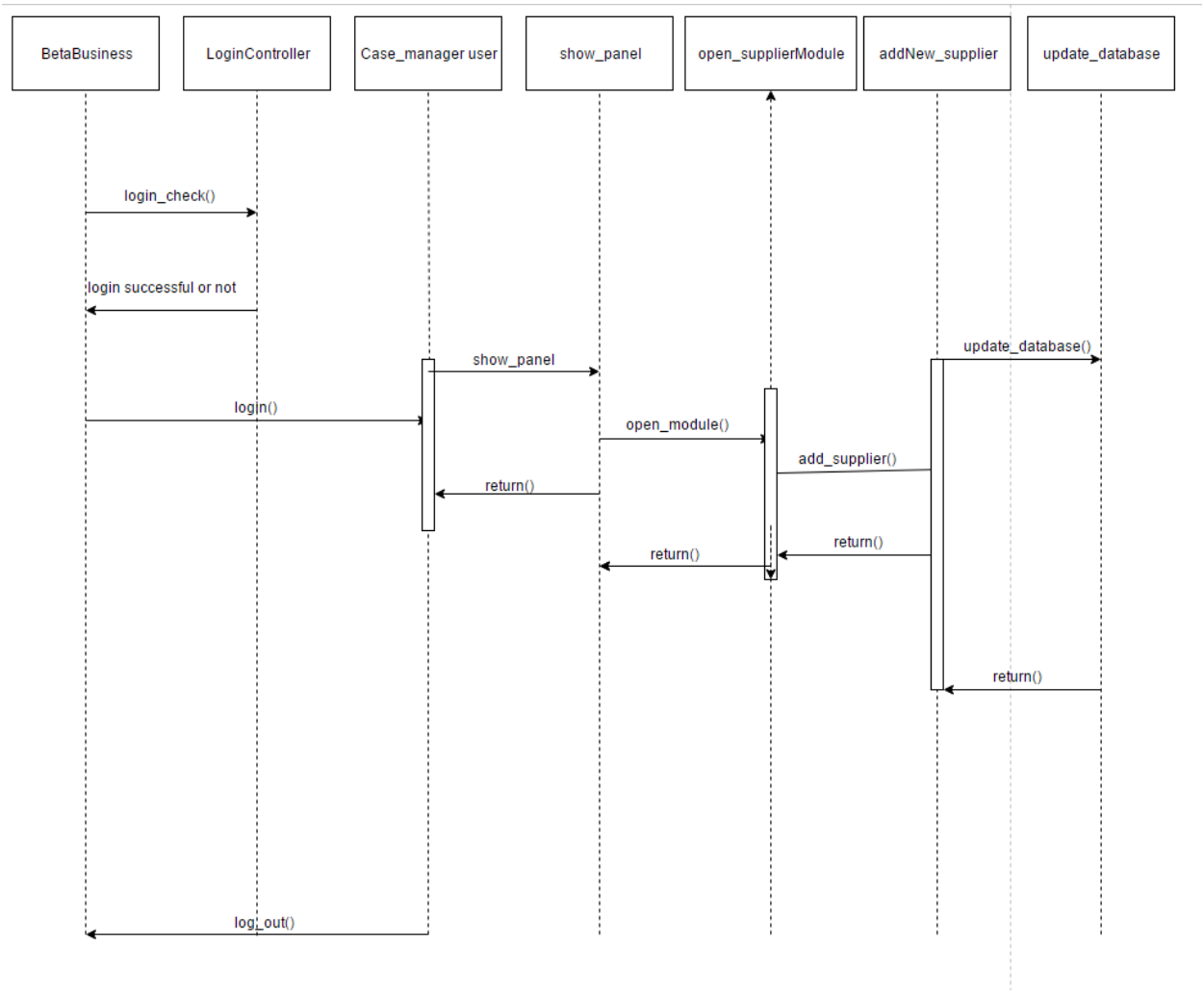


State Diagram 8(Use Story 13, Scenario 26)

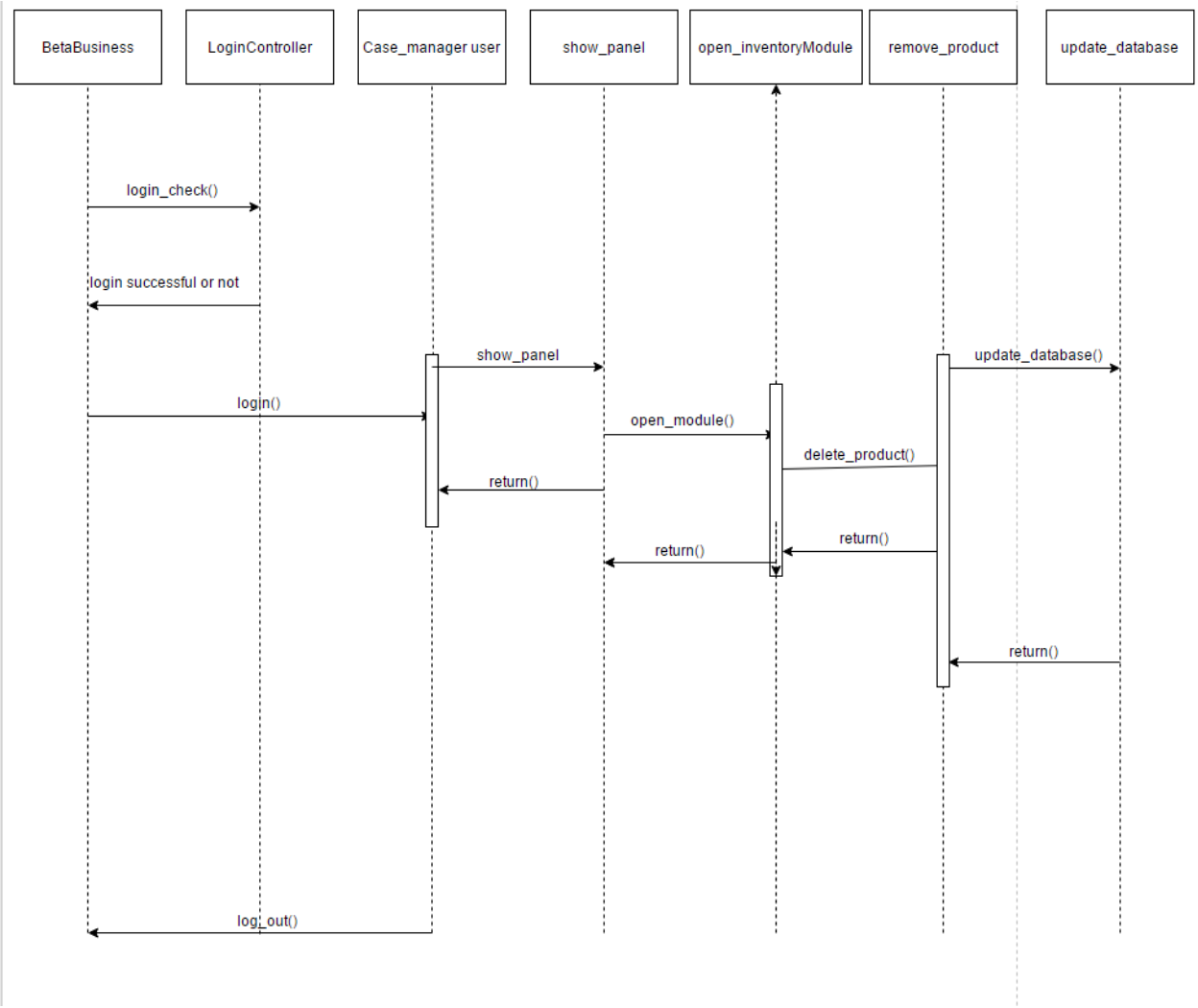
8.4 Sequence Diagrams



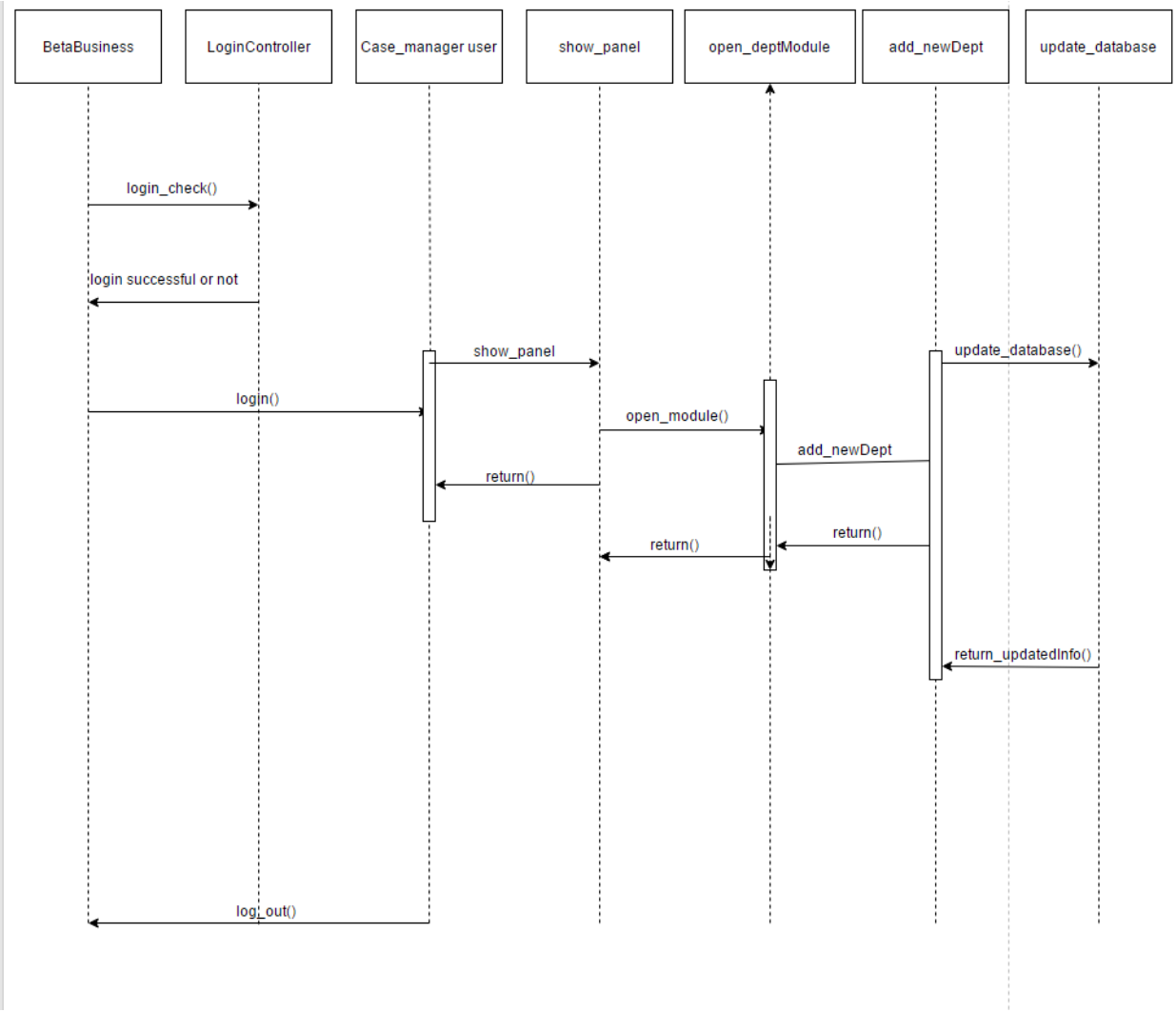
Sequence Diagram 1



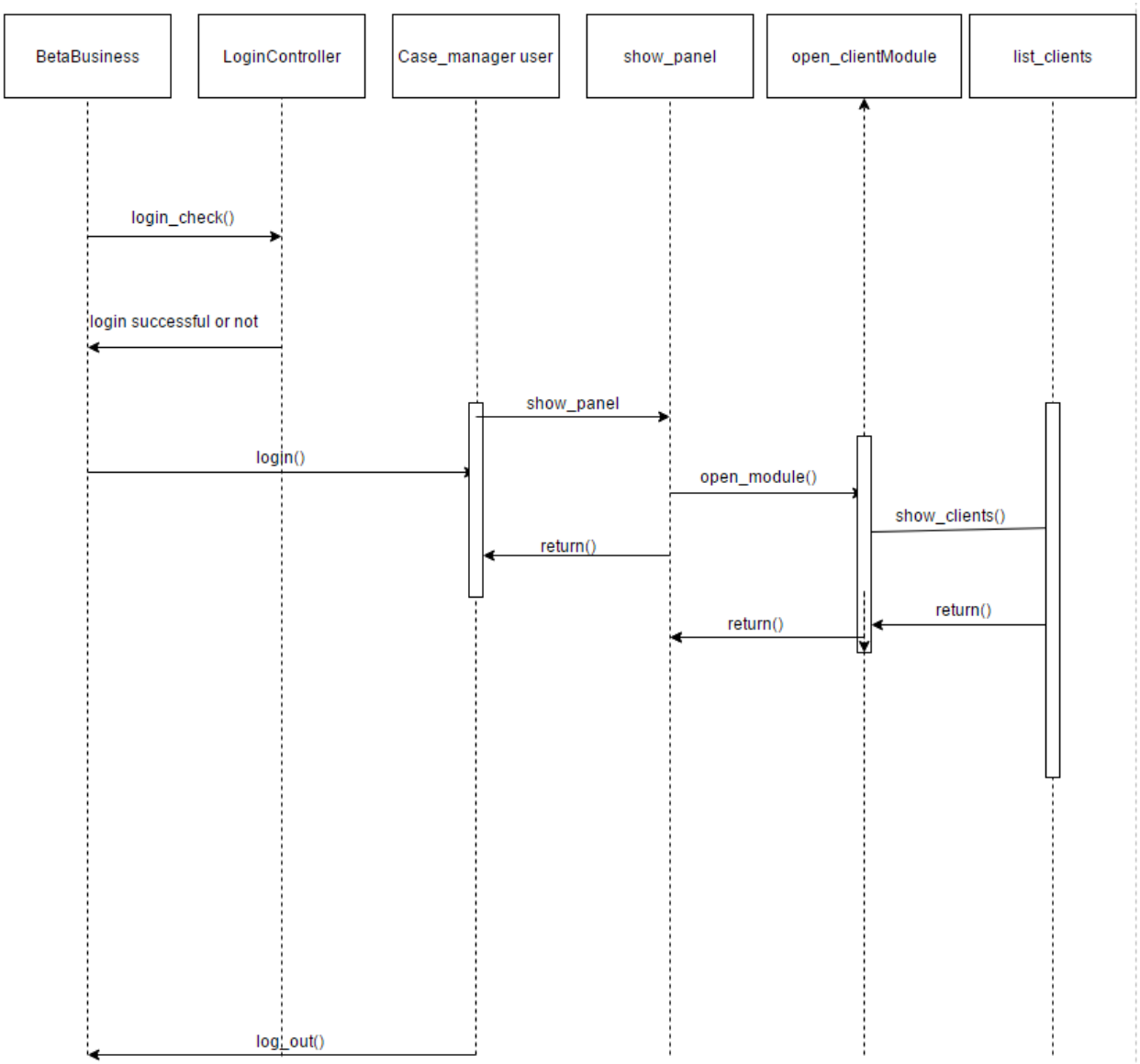
Sequence Diagram 2



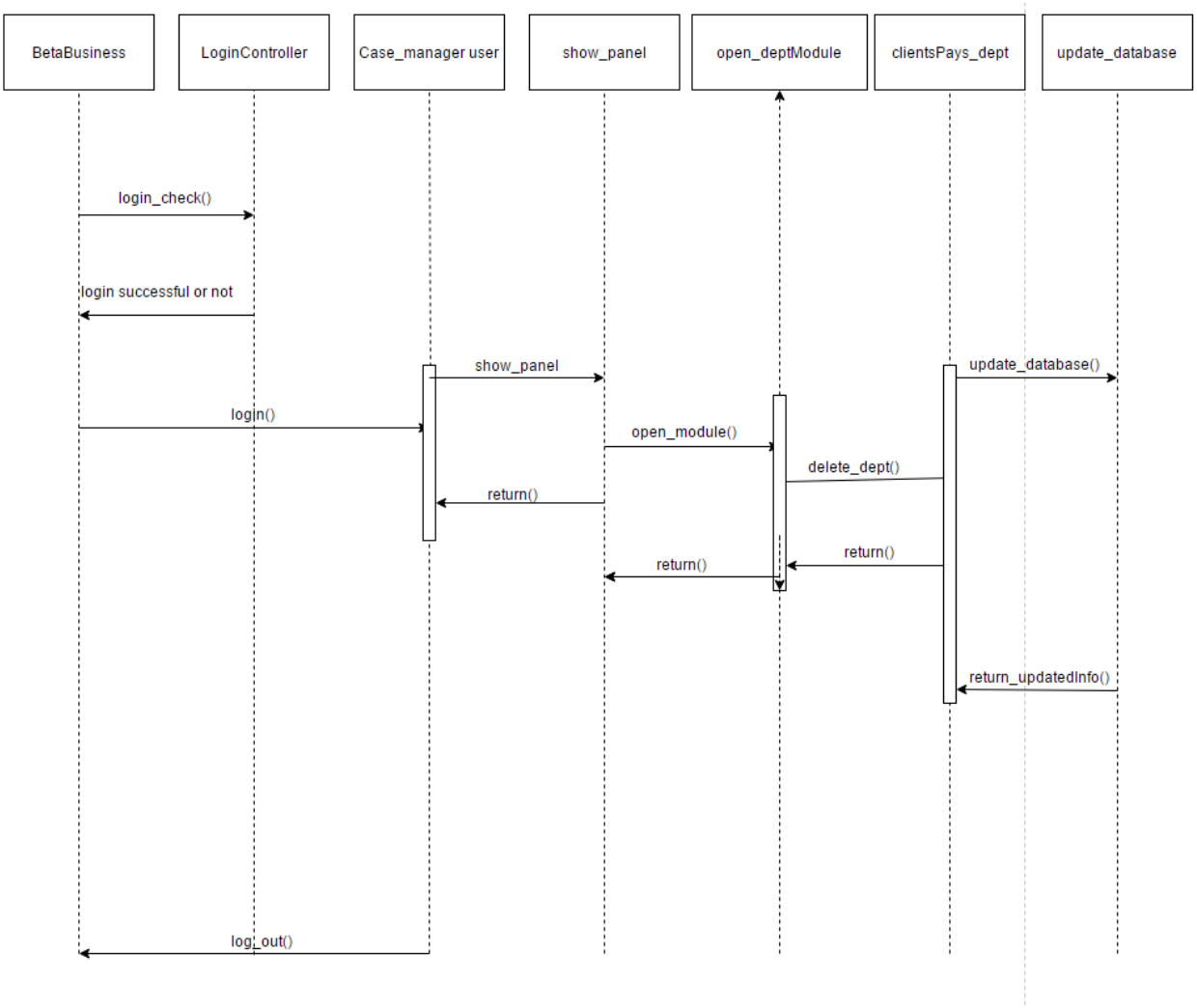
Sequence Diagram 3



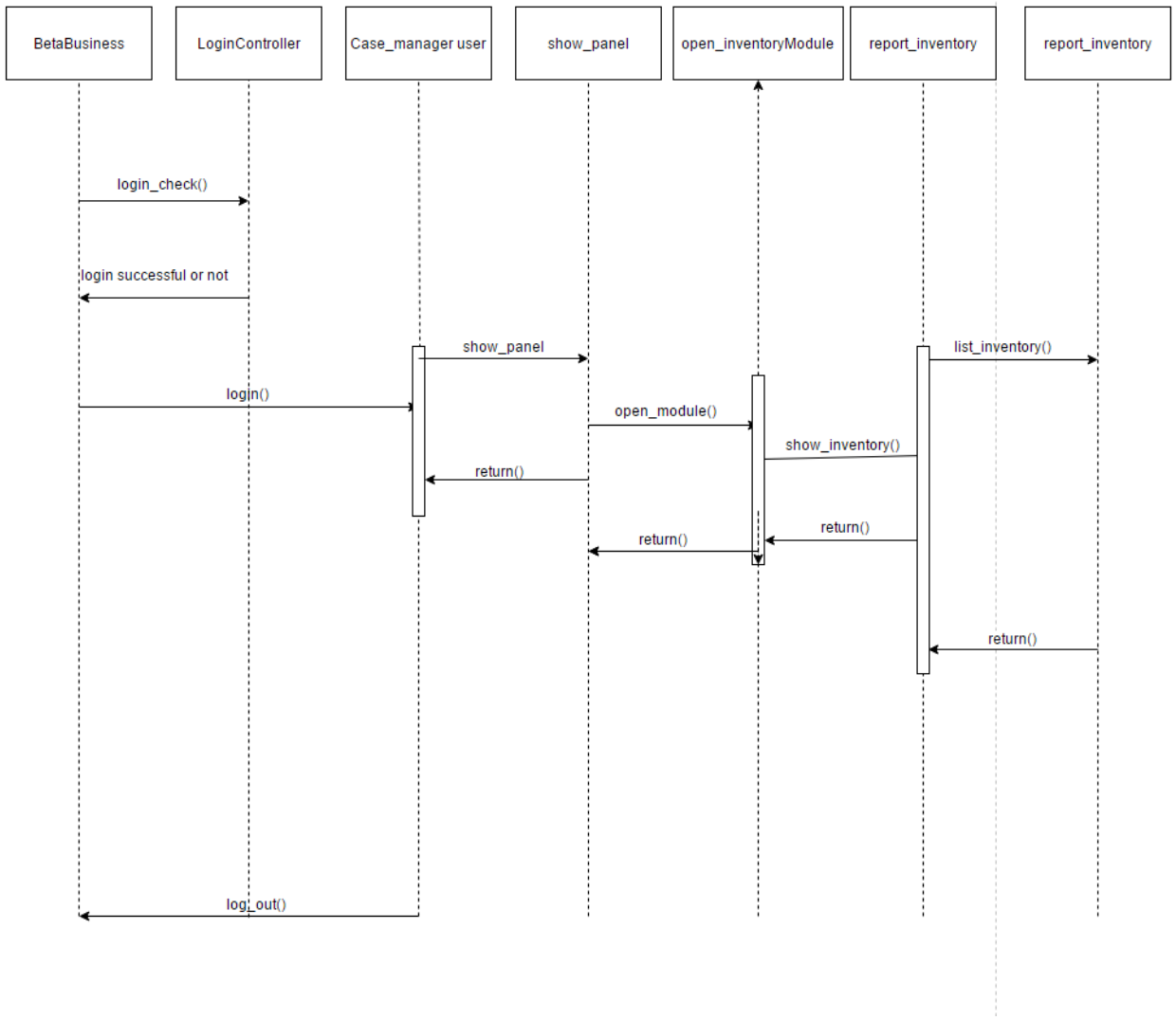
Sequence Diagram 4



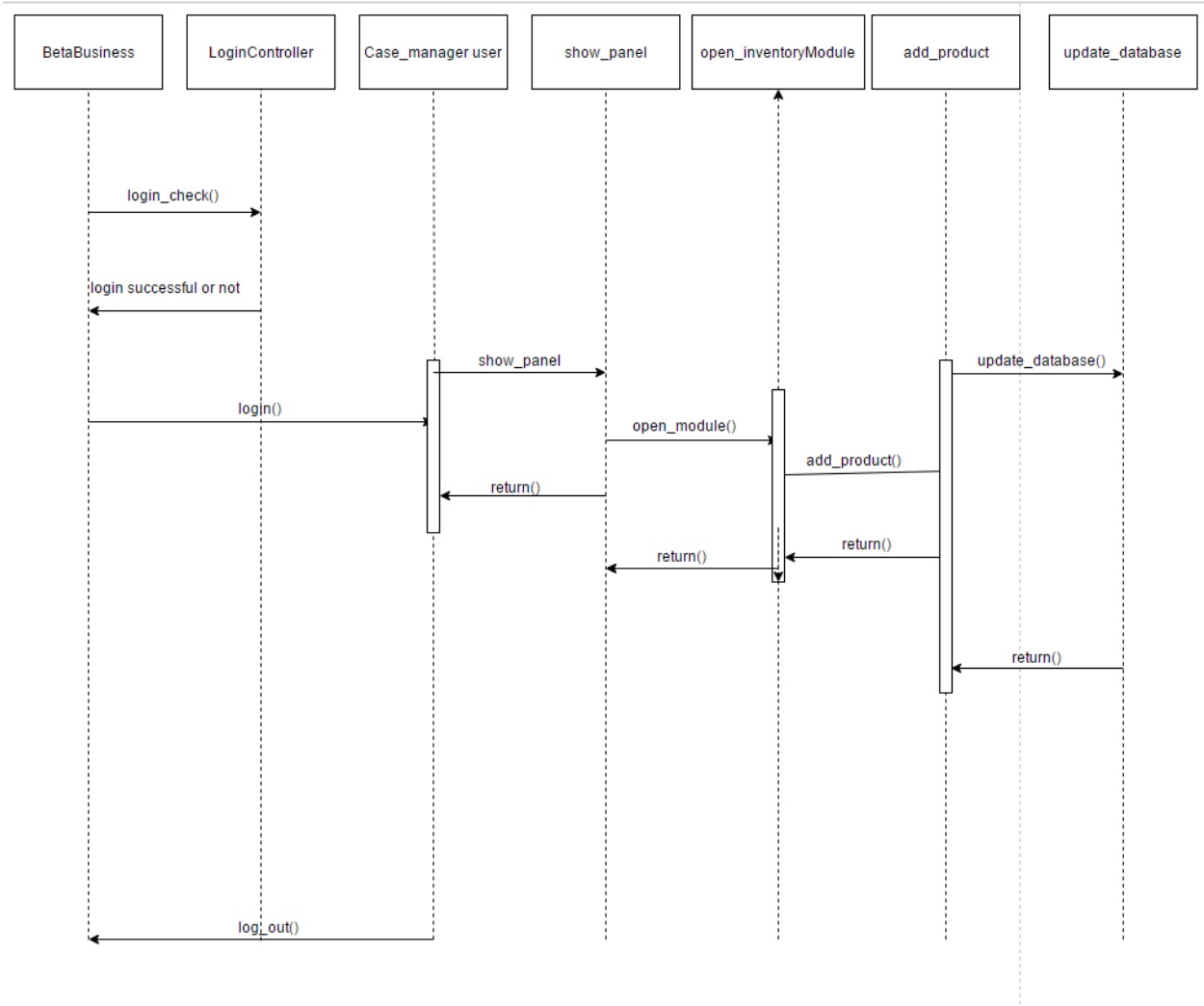
Sequence Diagram 5



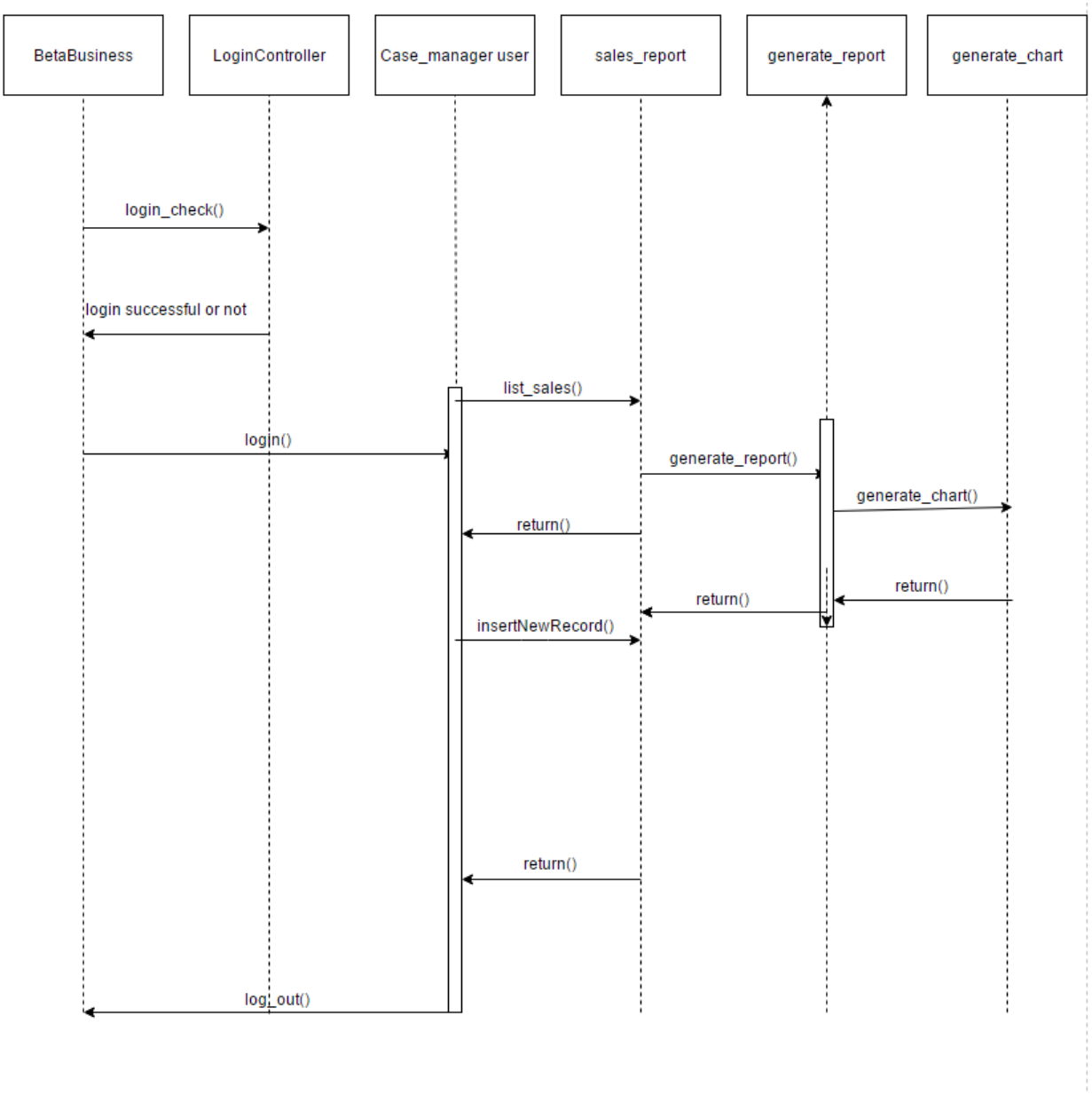
Sequence Diagram 6



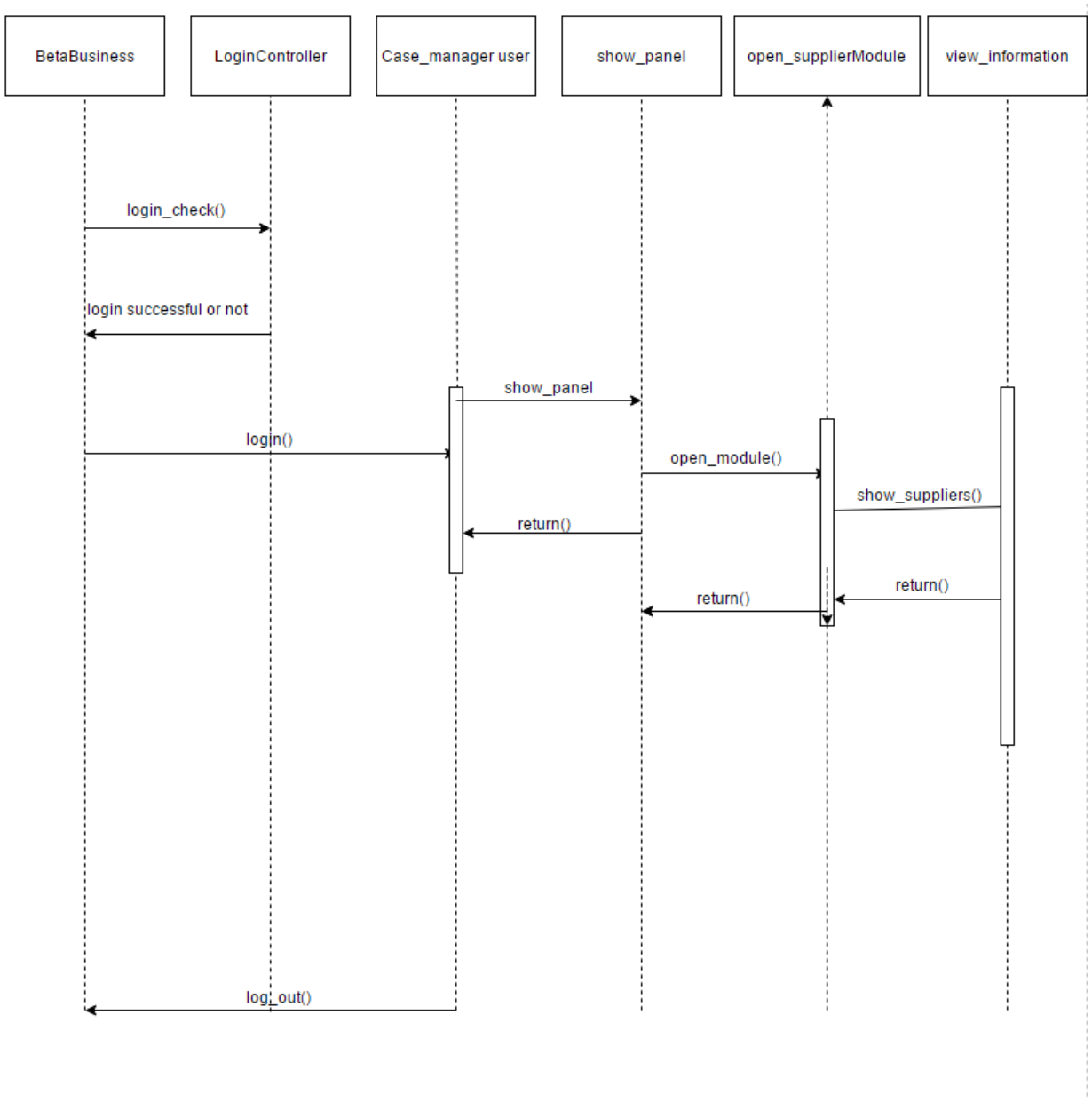
Sequence Diagram 7



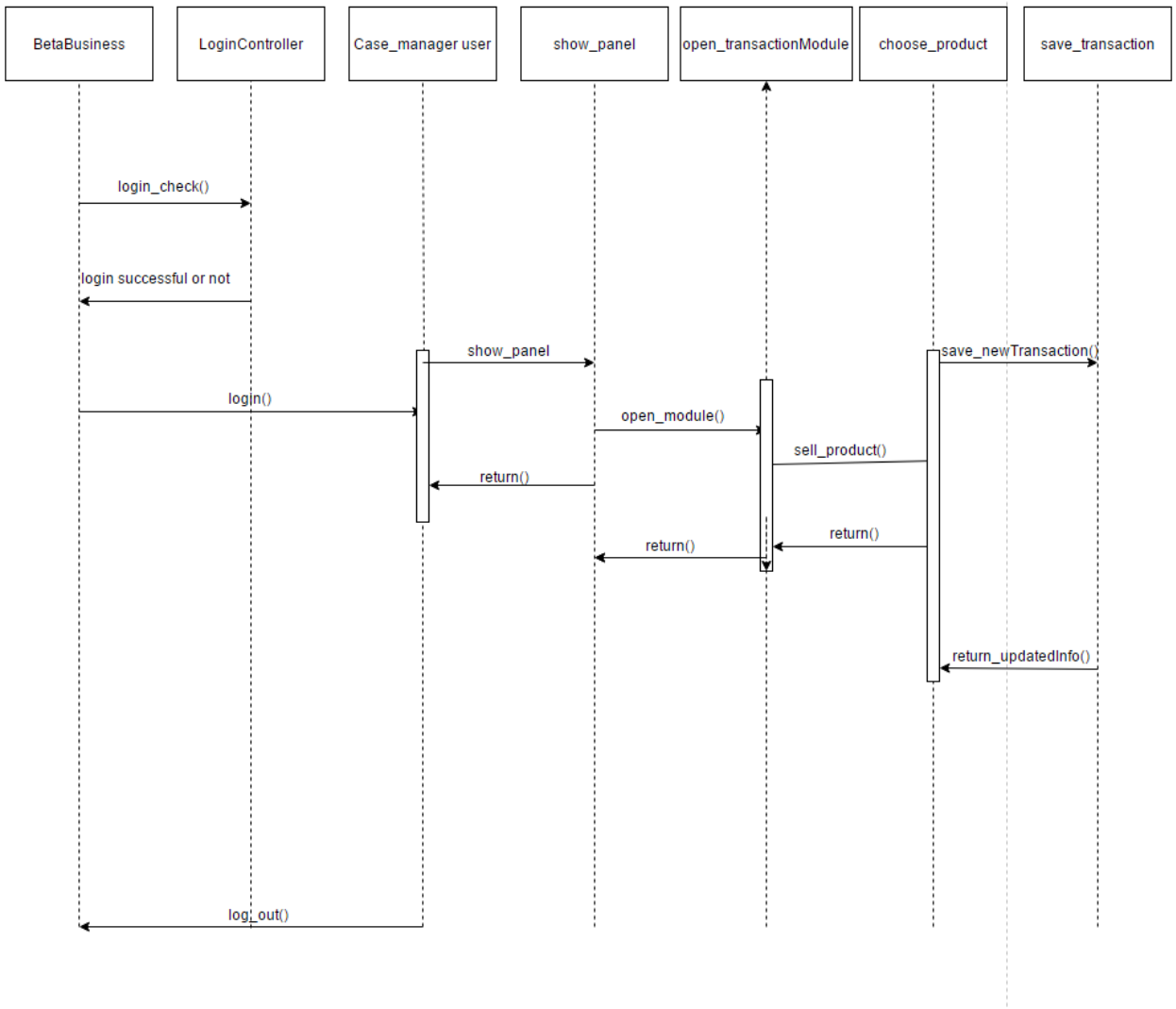
Sequence Diagram 8



Sequence Diagram 9

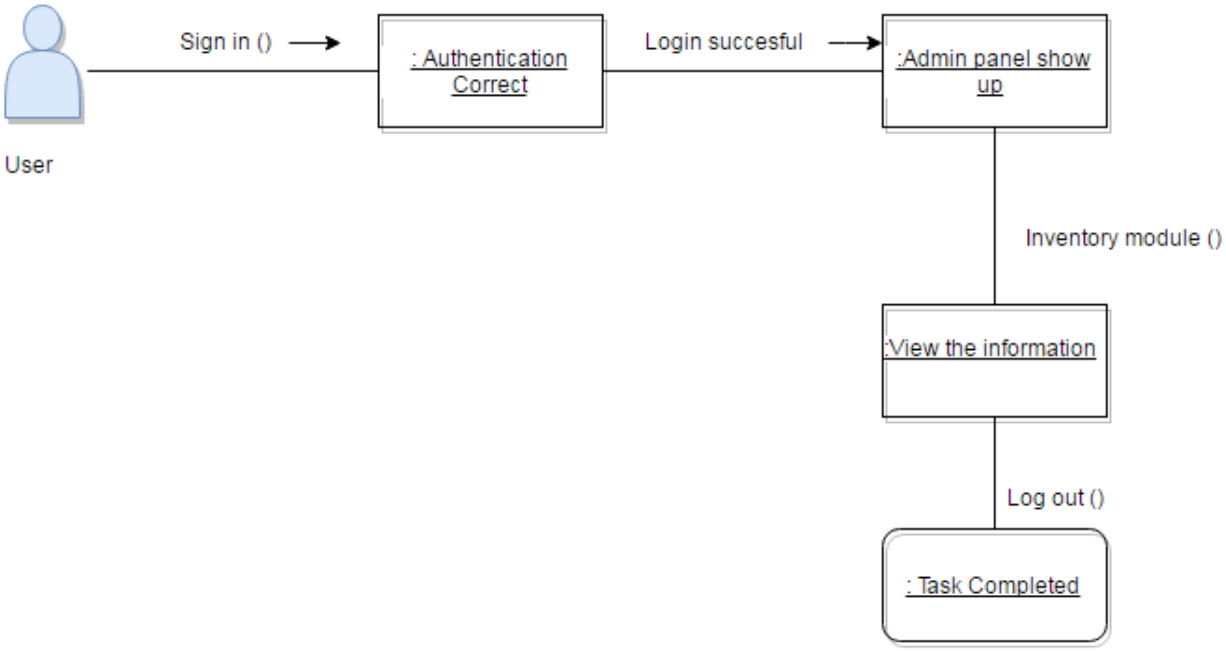


Sequence Diagram 10

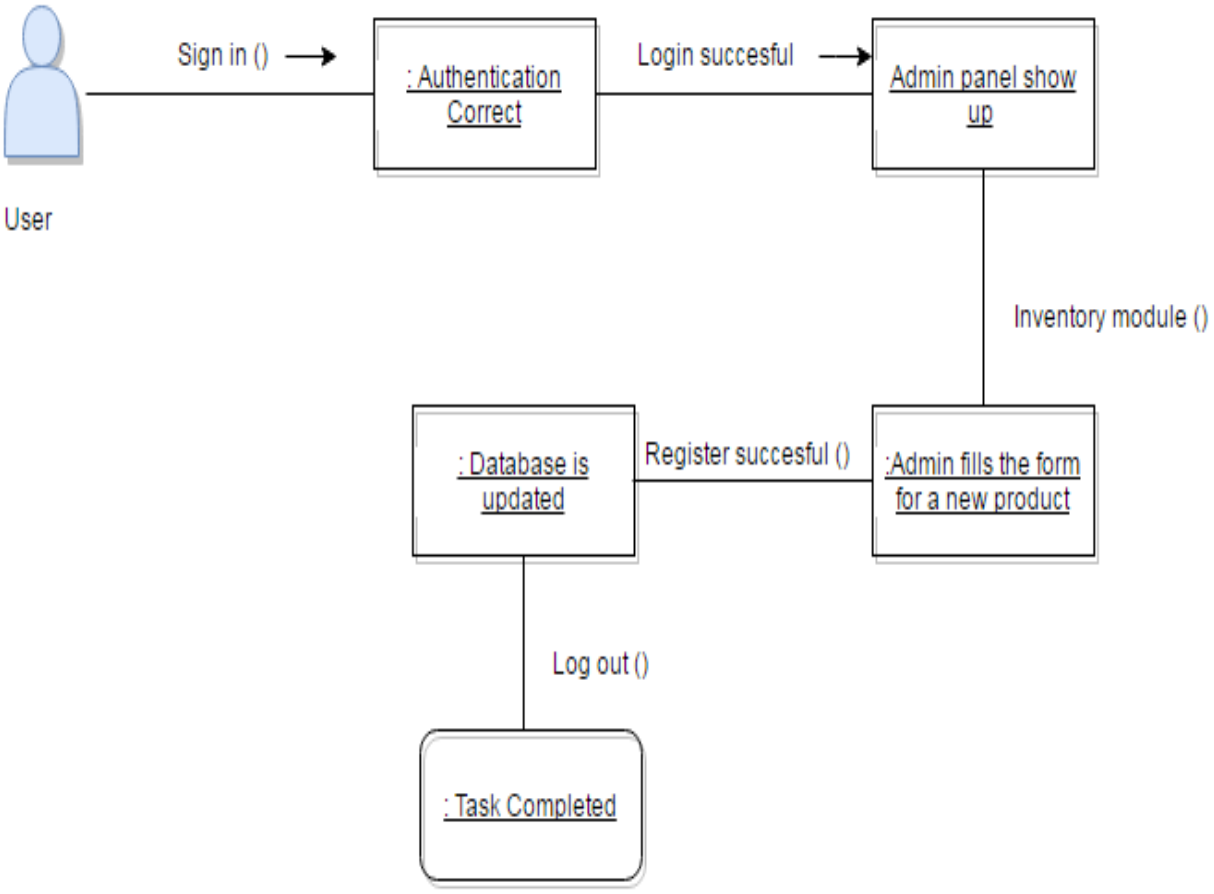


Sequence Diagram 11

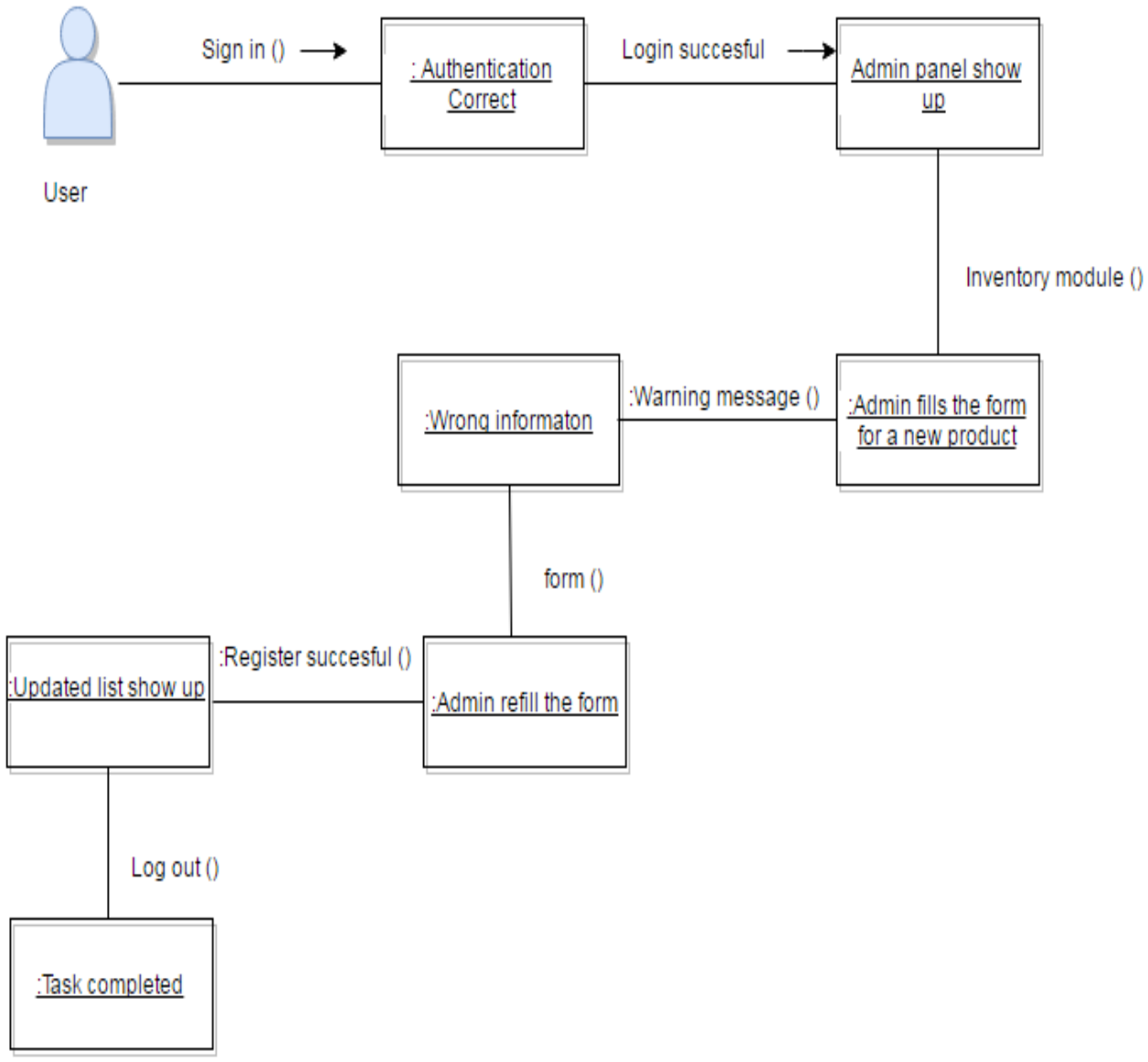
8.5 Collaboration Diagrams



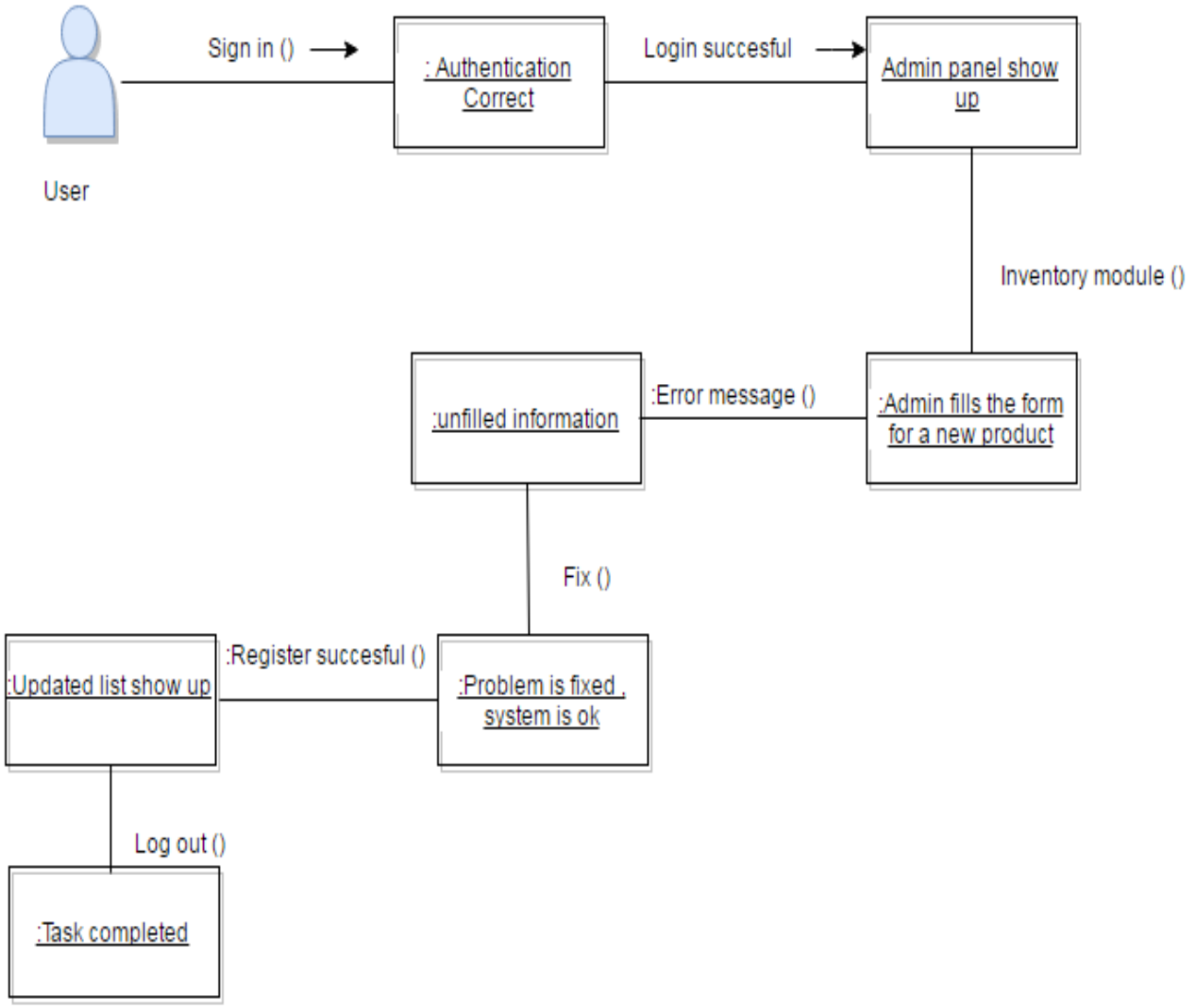
Collaboration Diagram 1 (Scenario 1)



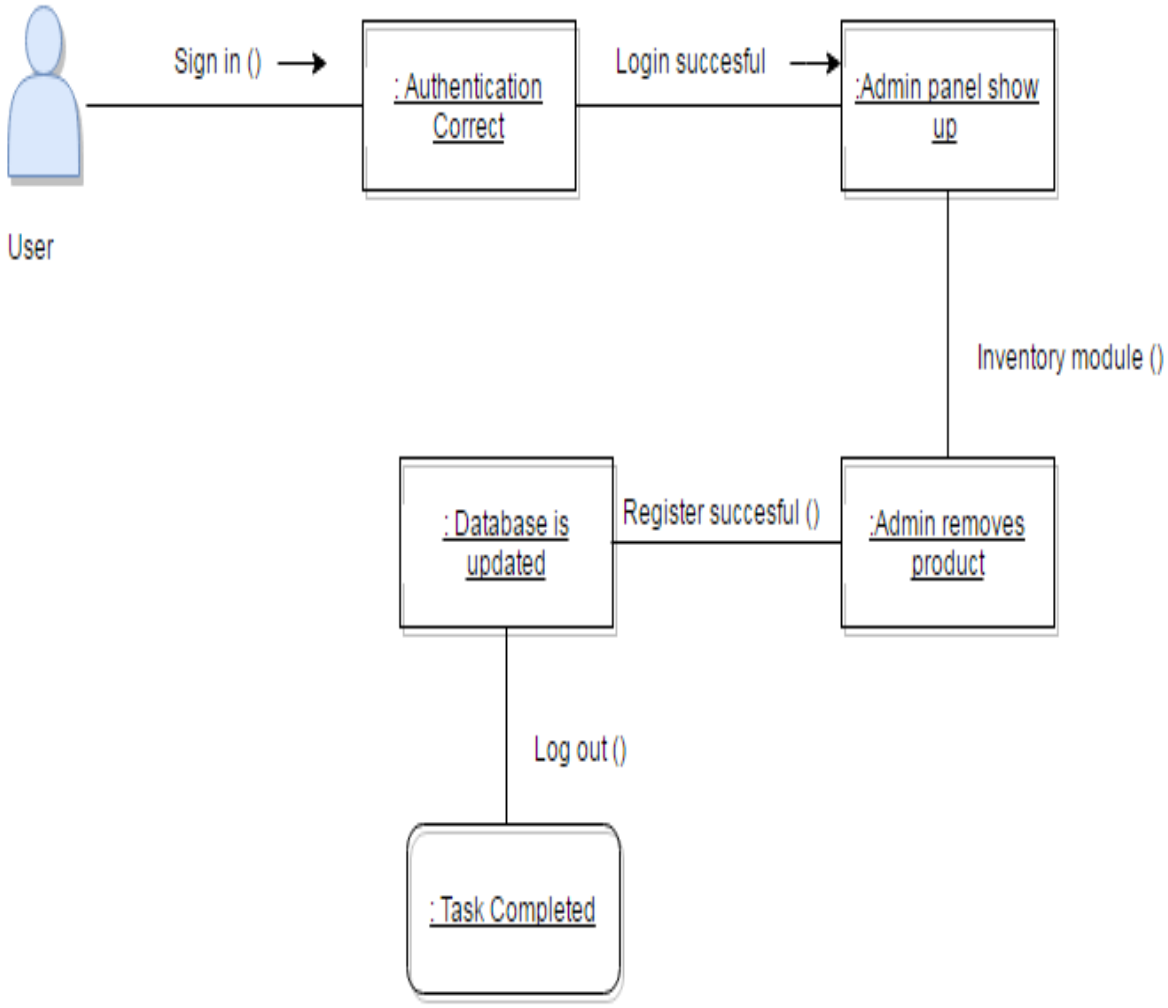
Collaboration Diagram 2 (Scenario 2)



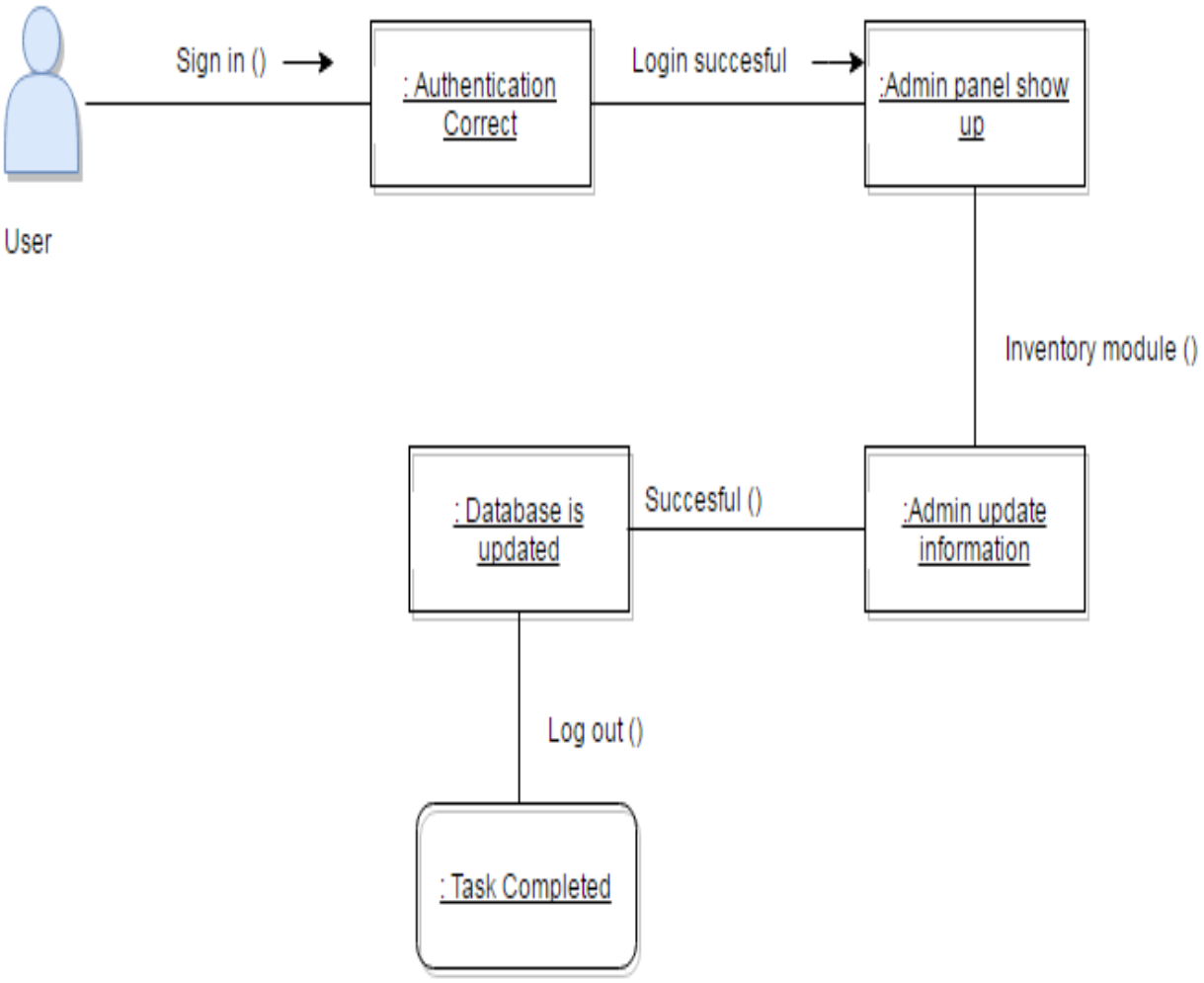
Collaboration Diagram 3 (Scenario 3)



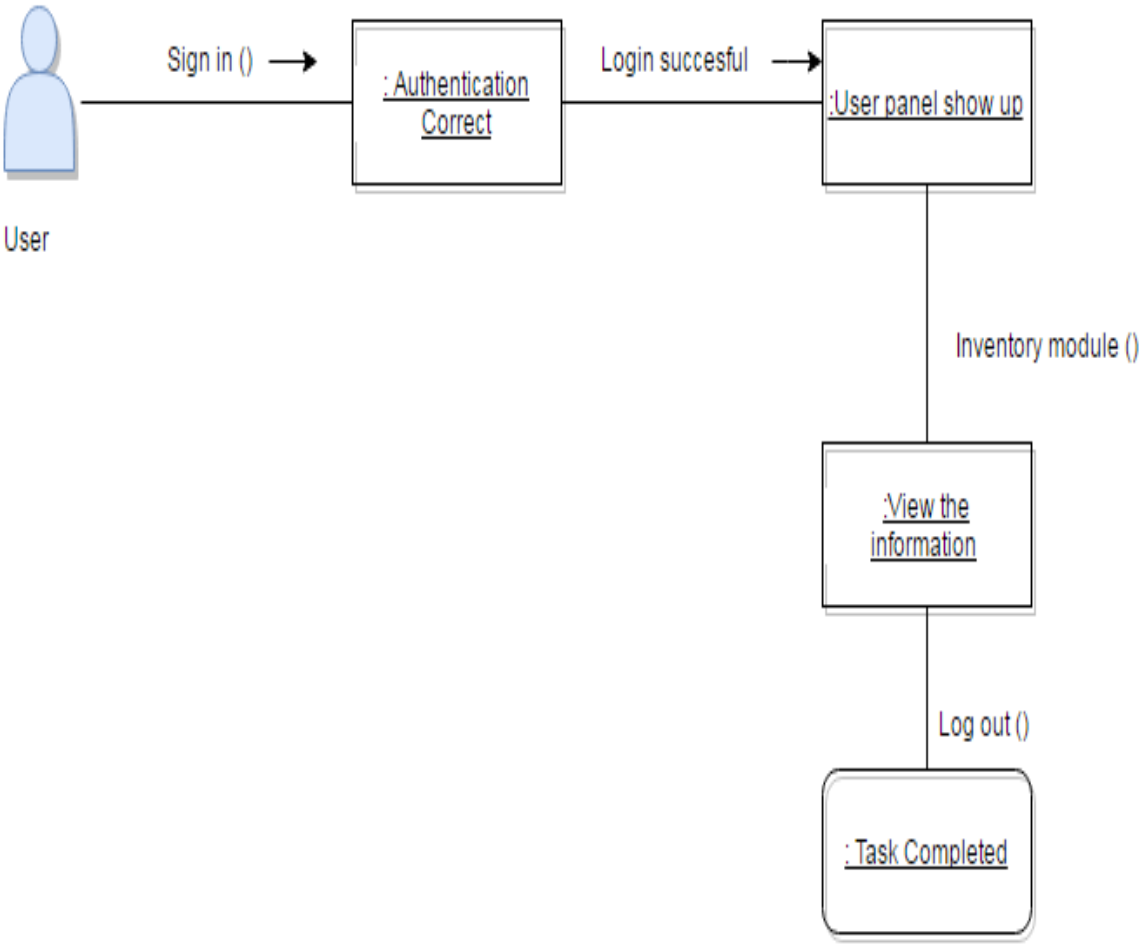
Collaboration Diagram 4 (Scenario 4)



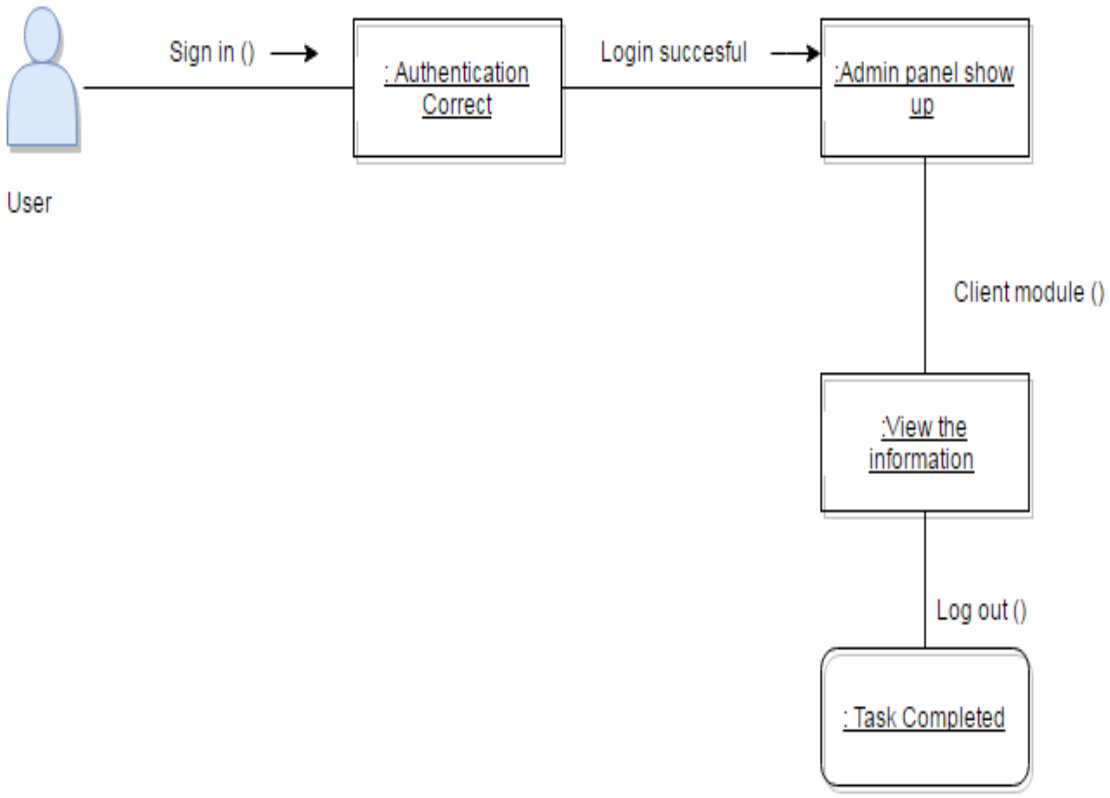
Collaboration Diagram 5 (Scenario 5)



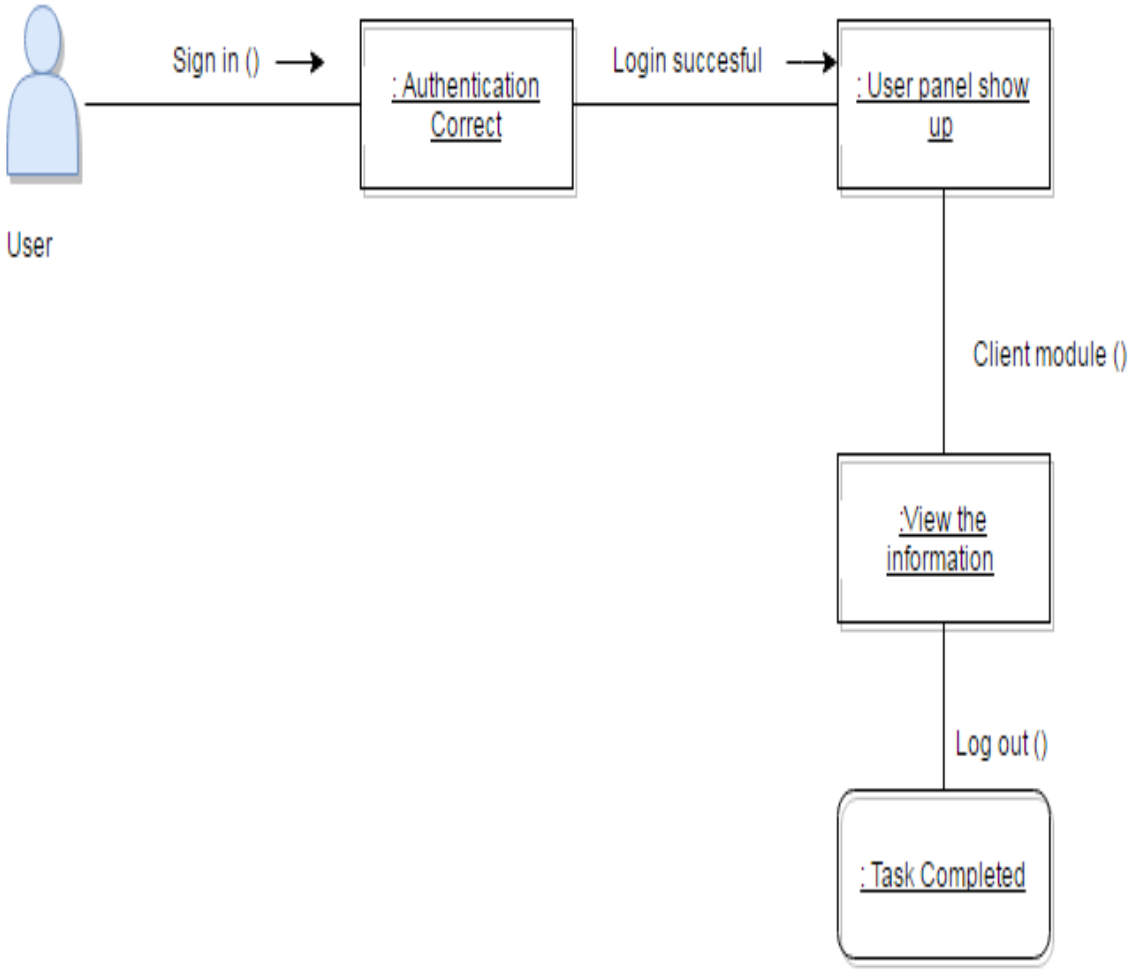
Collaboration Diagram 6 (Scenario 6)



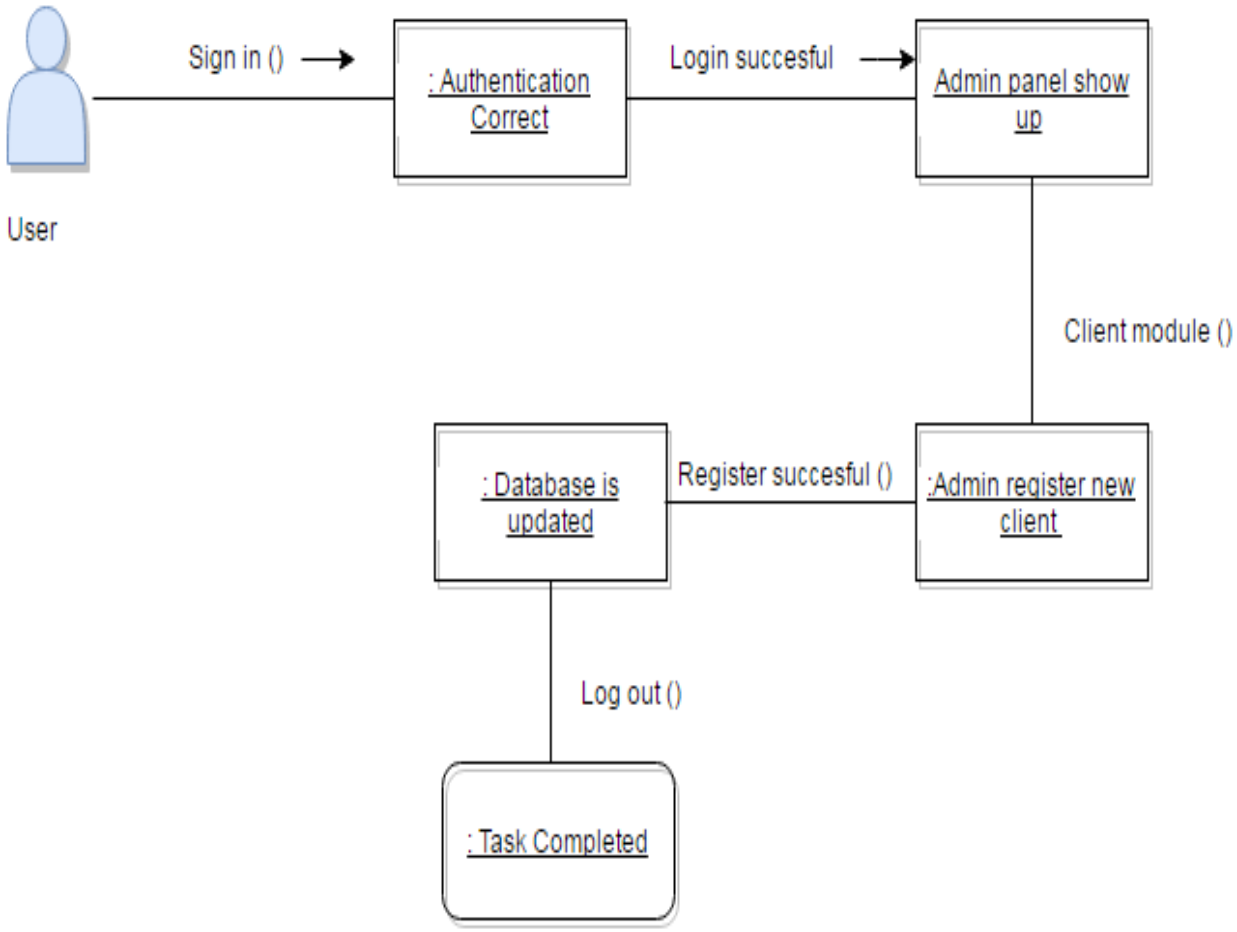
Collaboration Diagram 7 (Scenario 7)



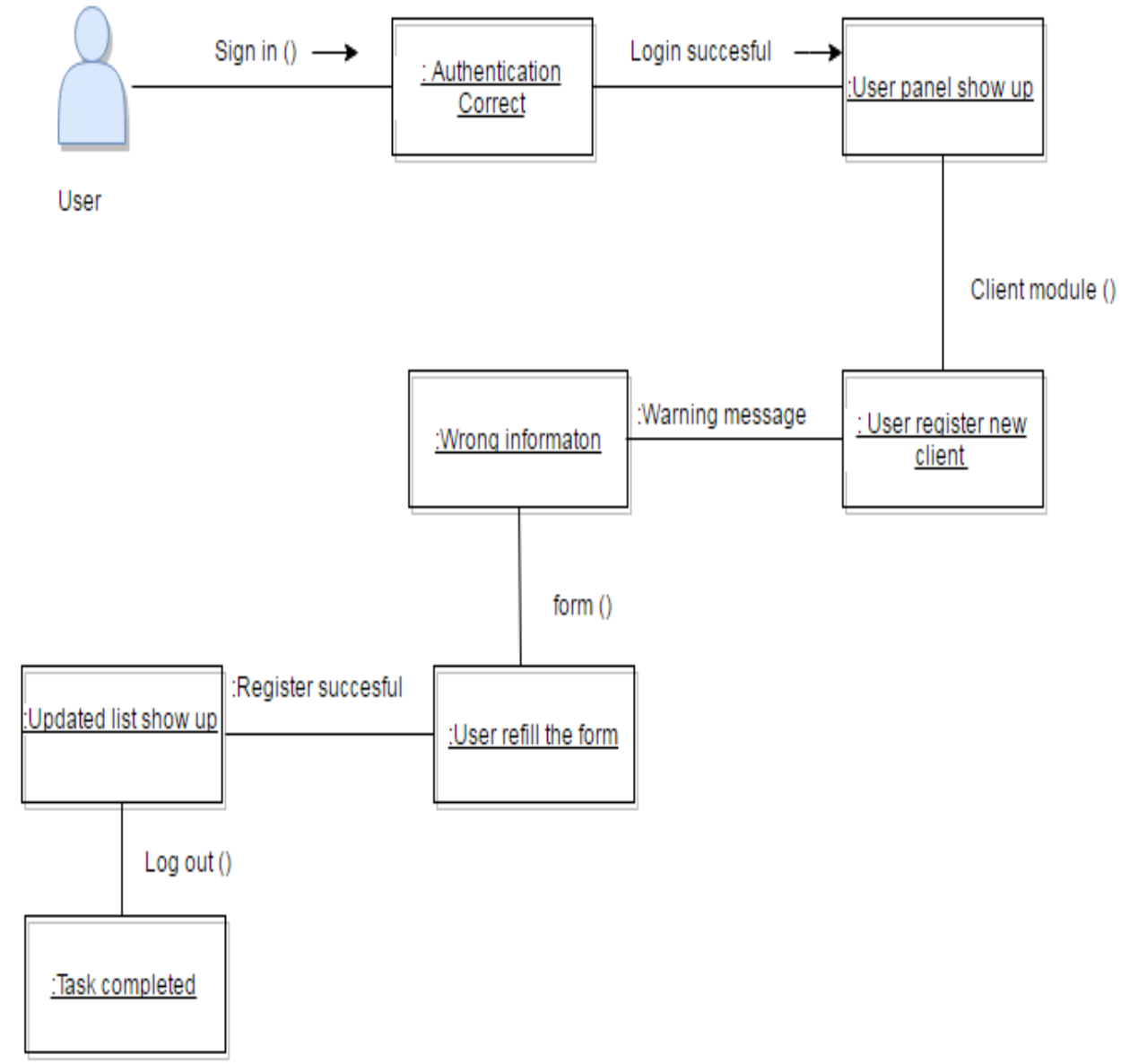
Collaboration Diagram 8 (Scenario 8)



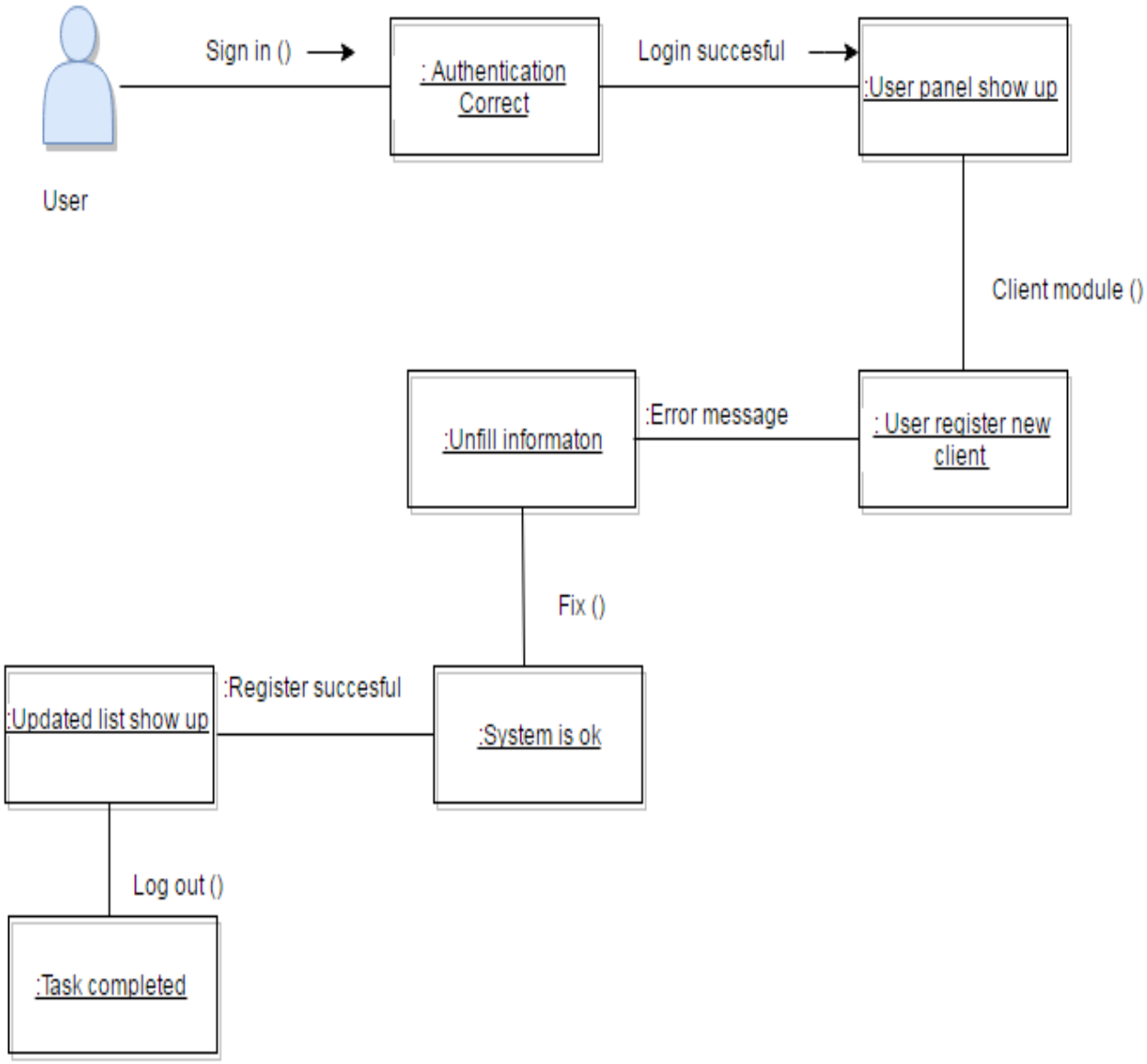
Collaboration Diagram 9 (Scenario 9)



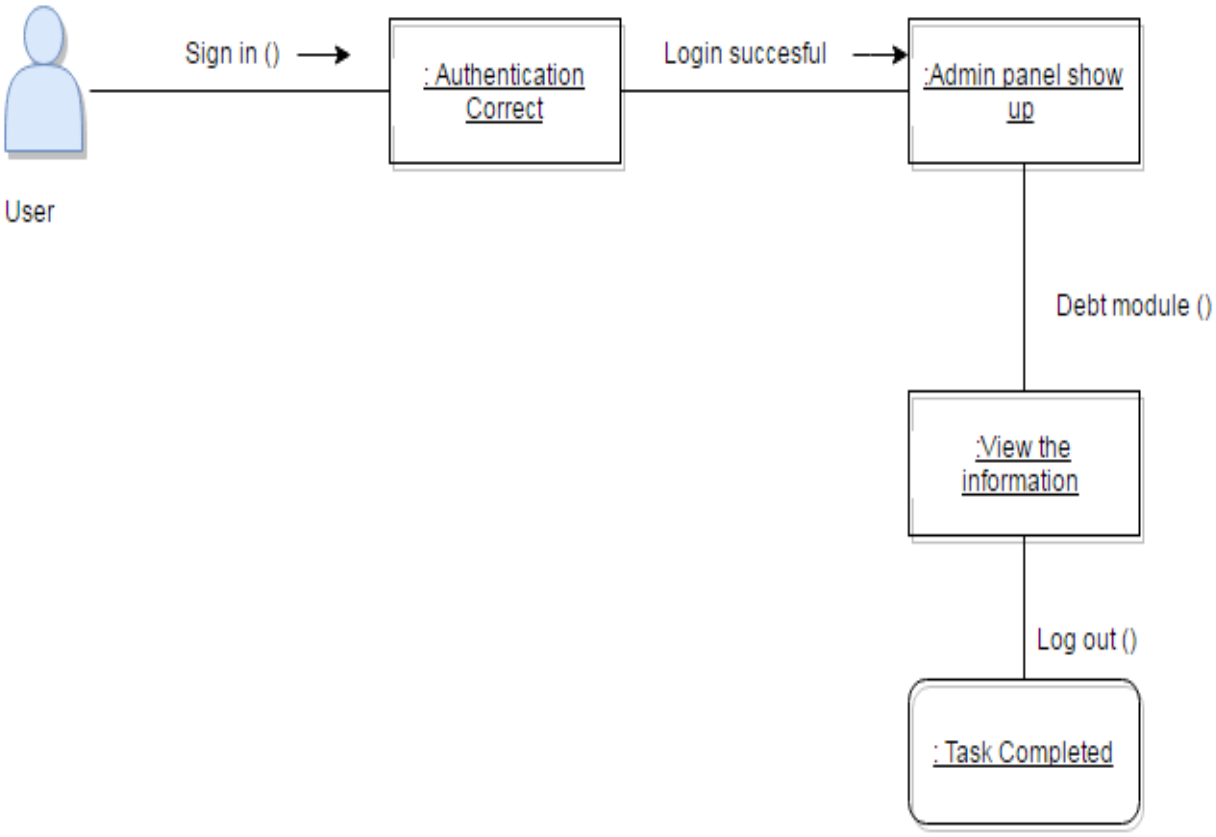
Collaboration Diagram 10 (Scenario 10)



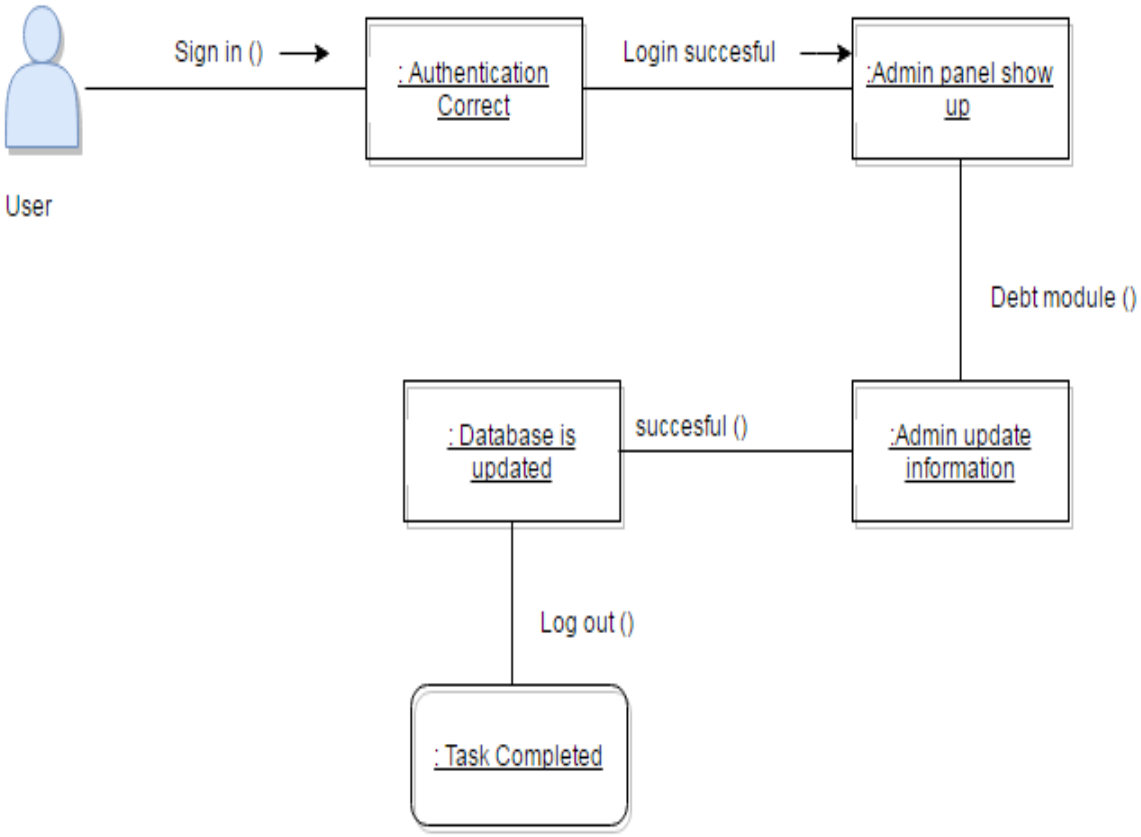
Collaboration Diagram 11 (Scenario 11)



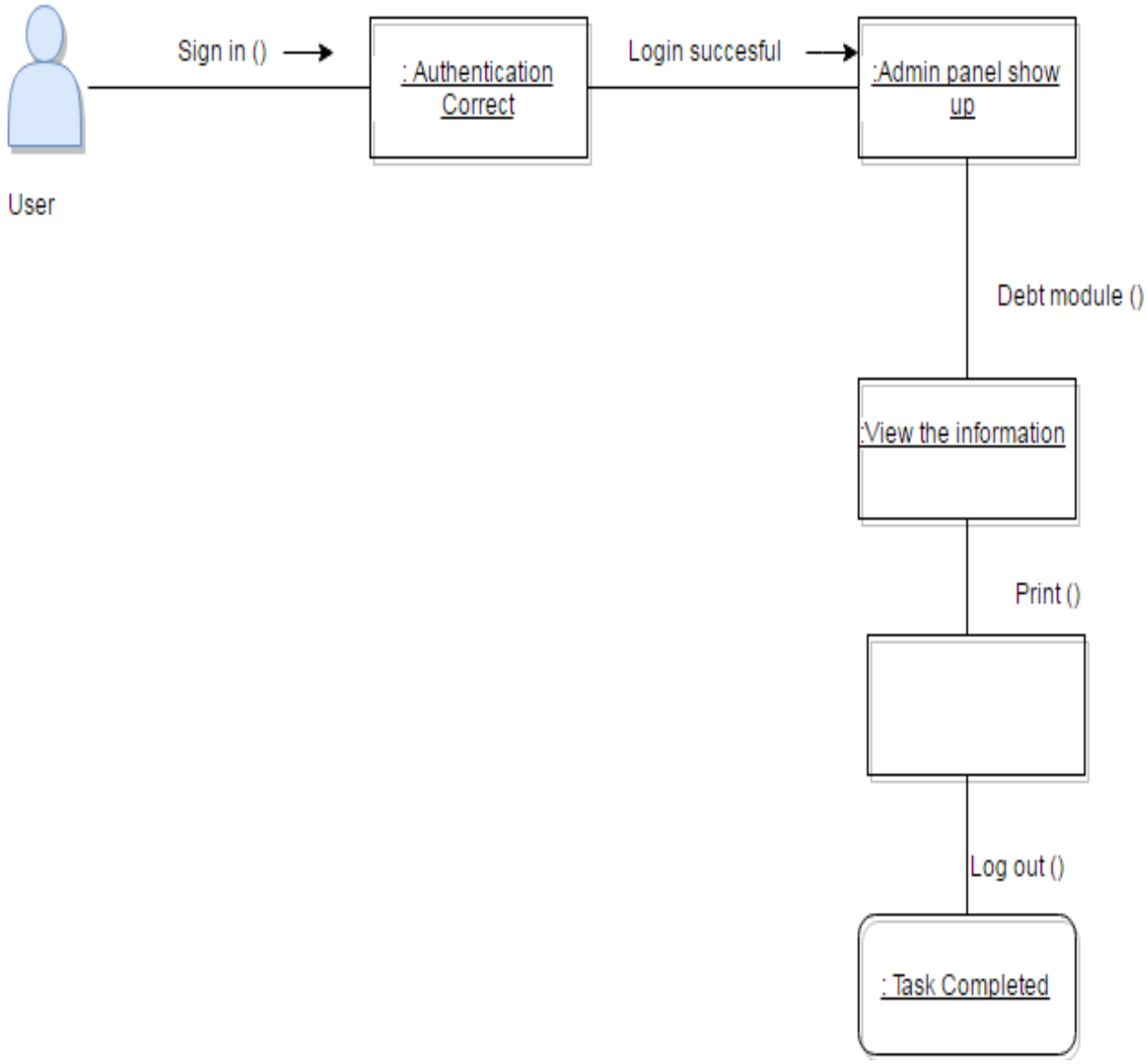
Collaboration Diagram 12 (Scenario 12)



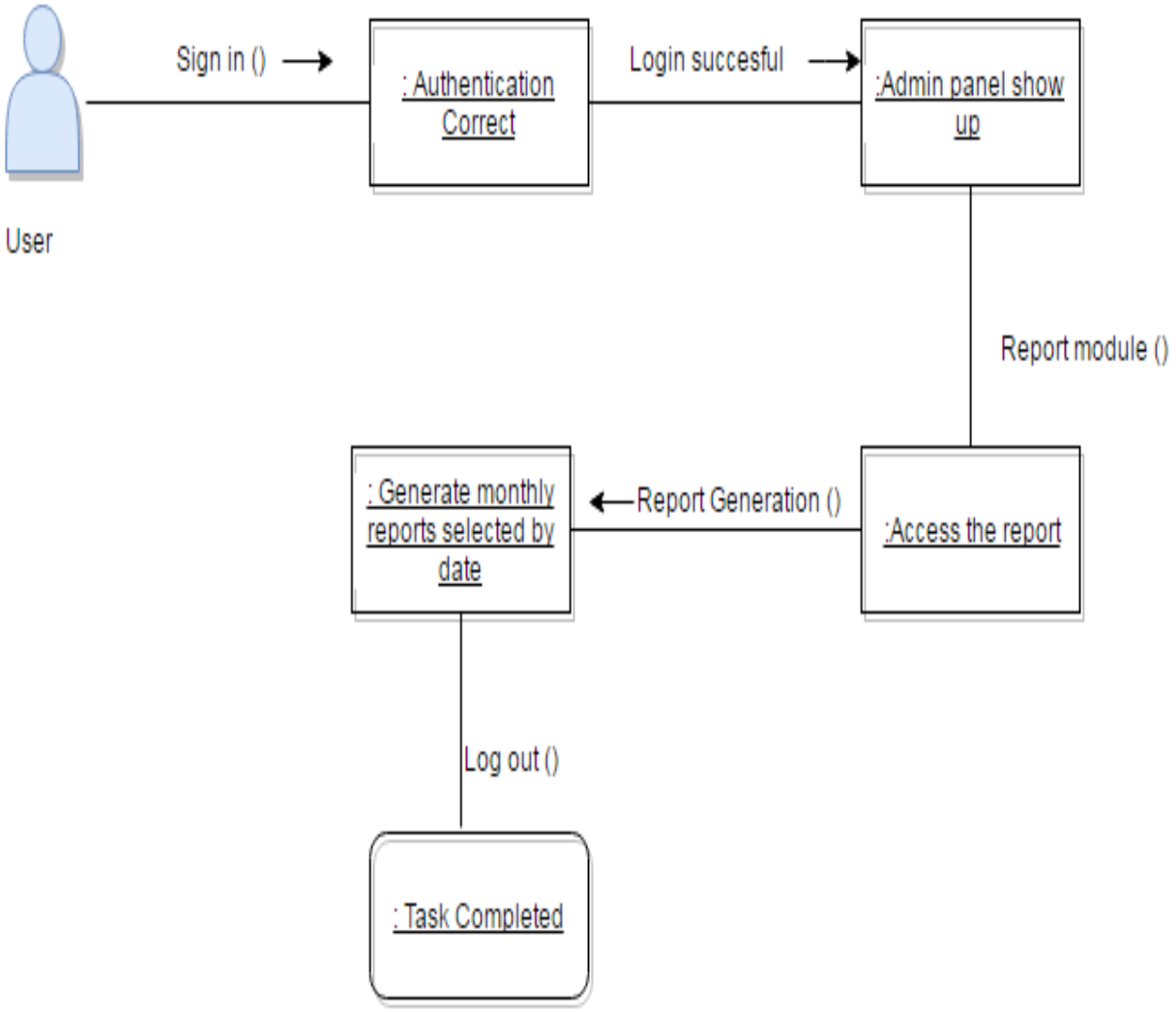
Collaboration Diagram 13 (Scenario 13)



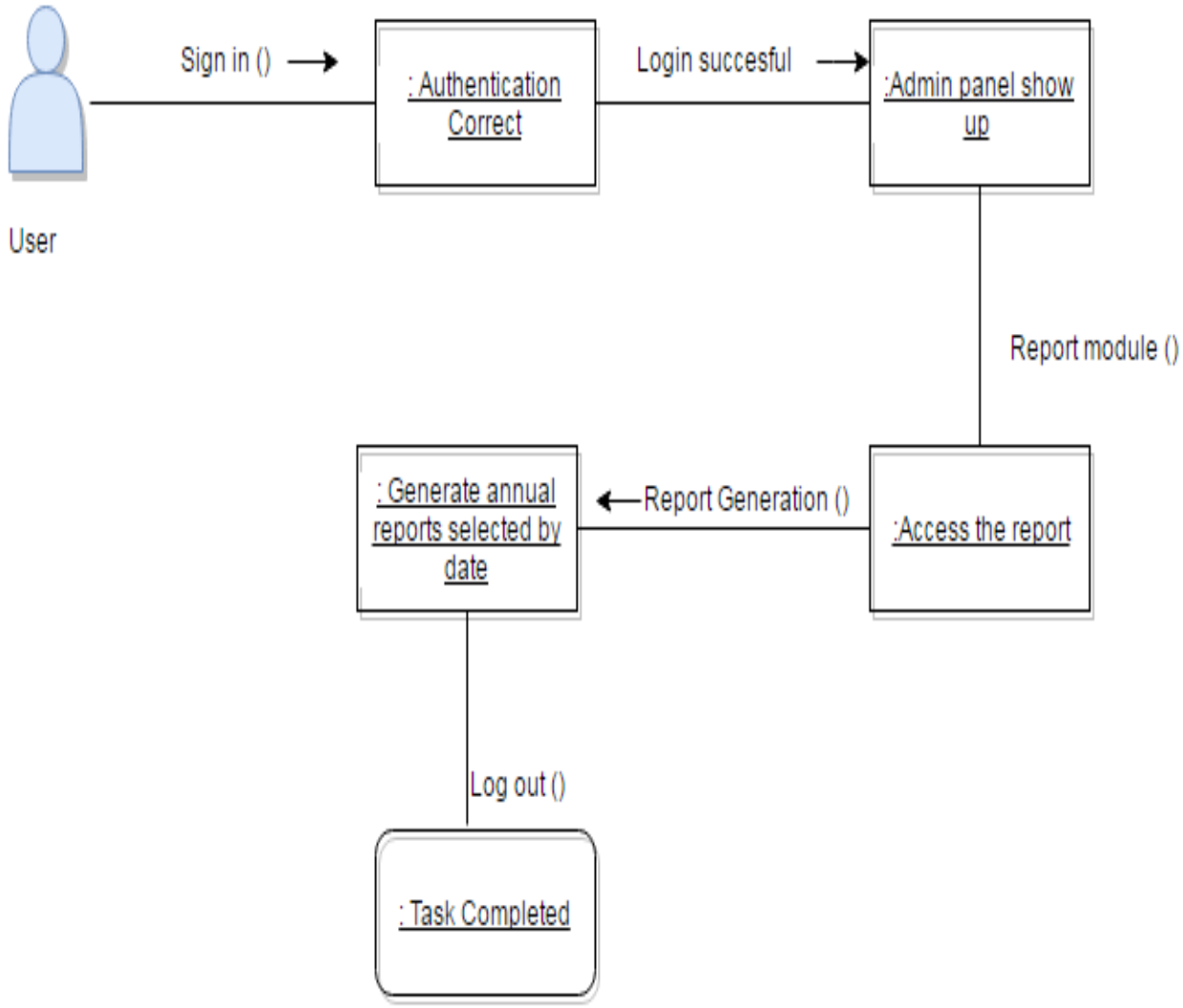
Collaboration Diagram 14 (Scenario 14)



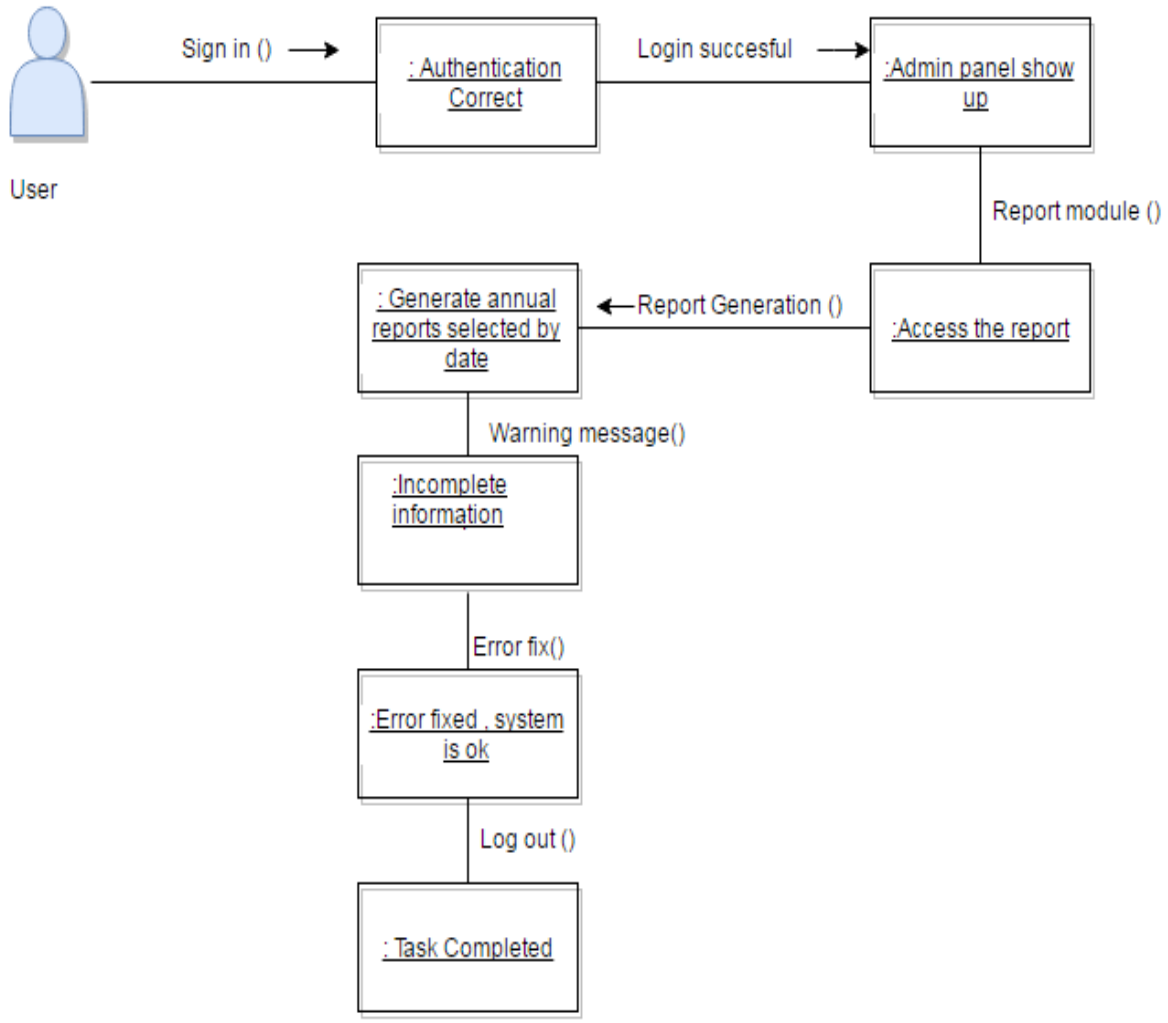
Collaboration Diagram 15 (Scenario 15)



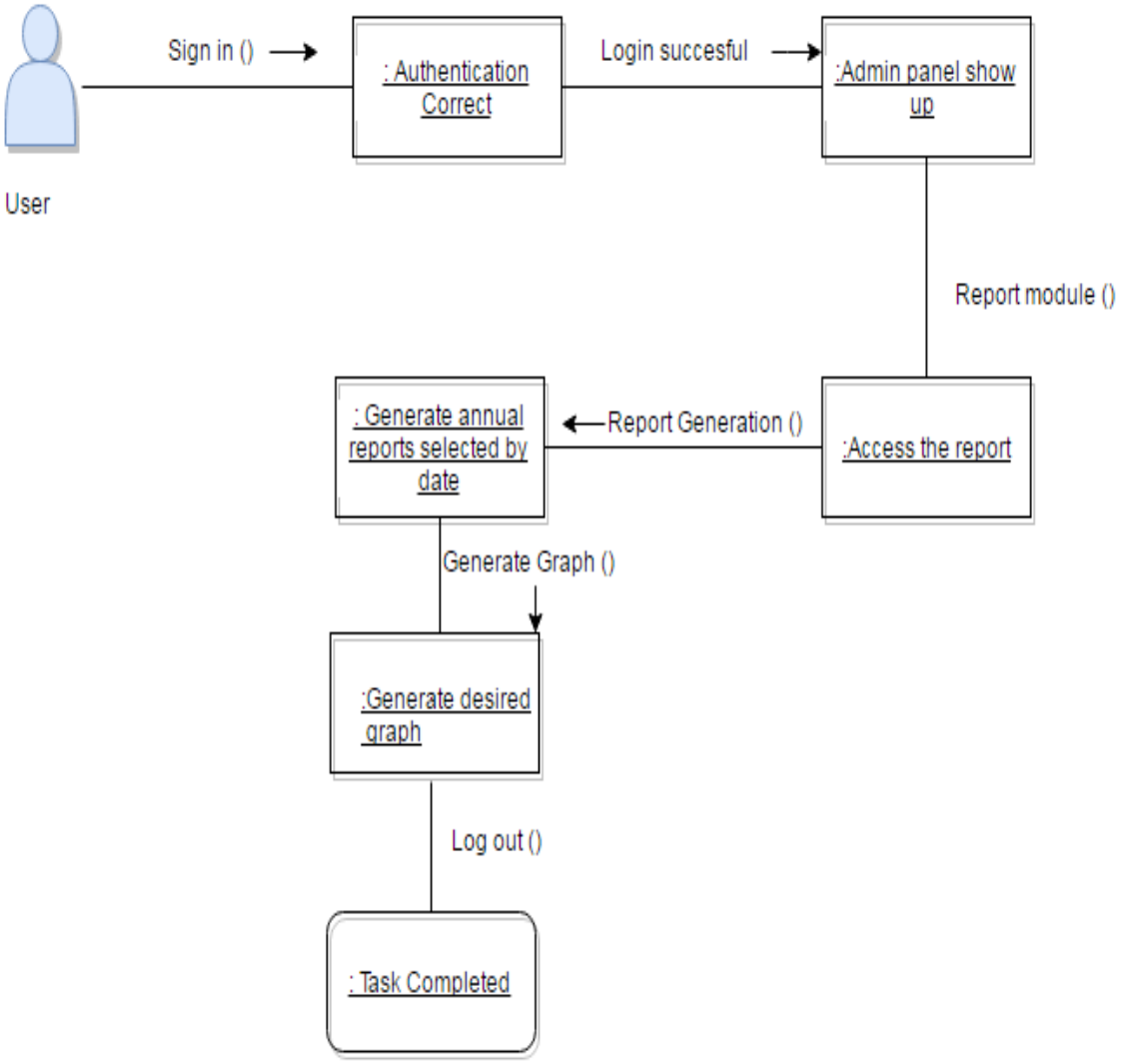
Collaboration Diagram 16 (Scenario 16)



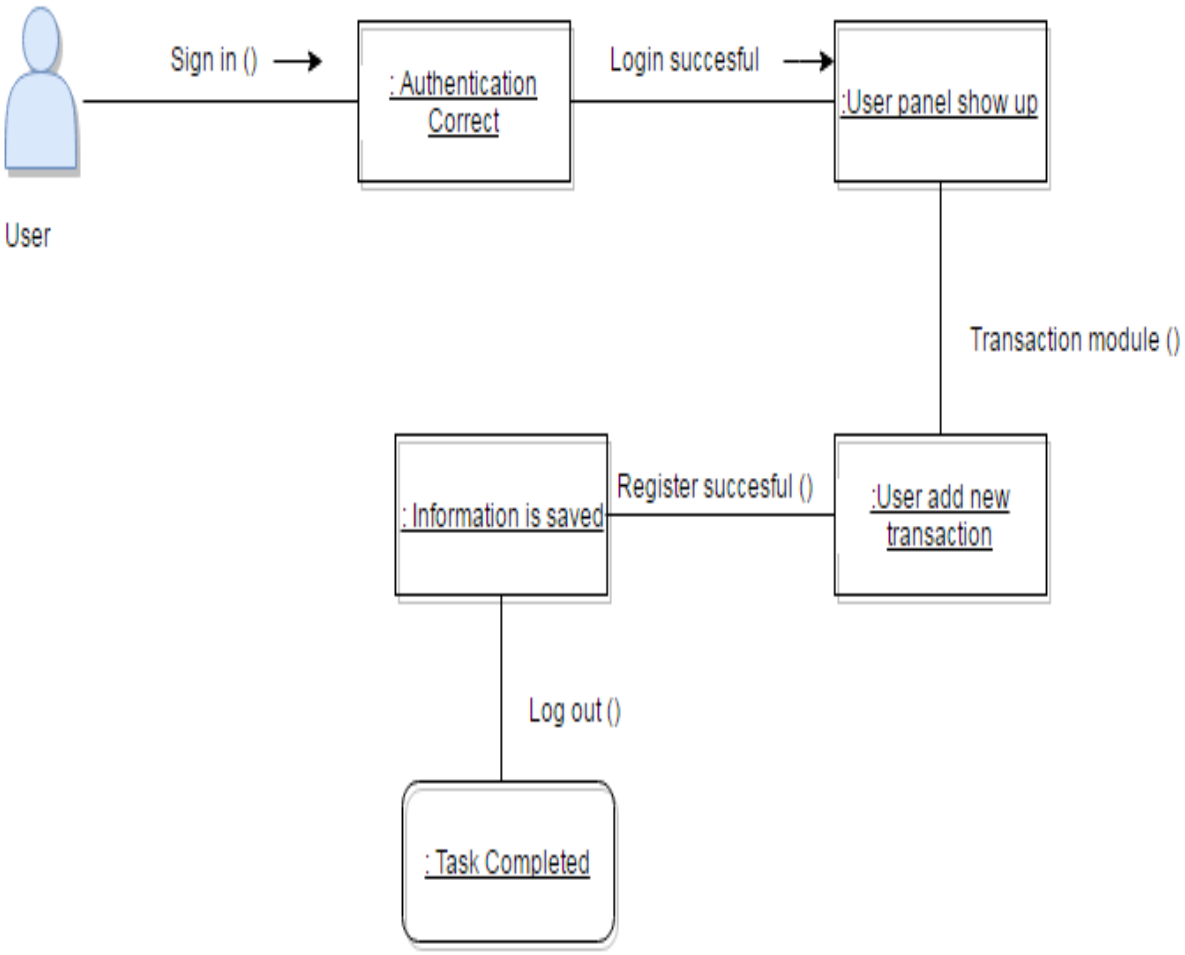
Collaboration Diagram 17 (Scenario 17)



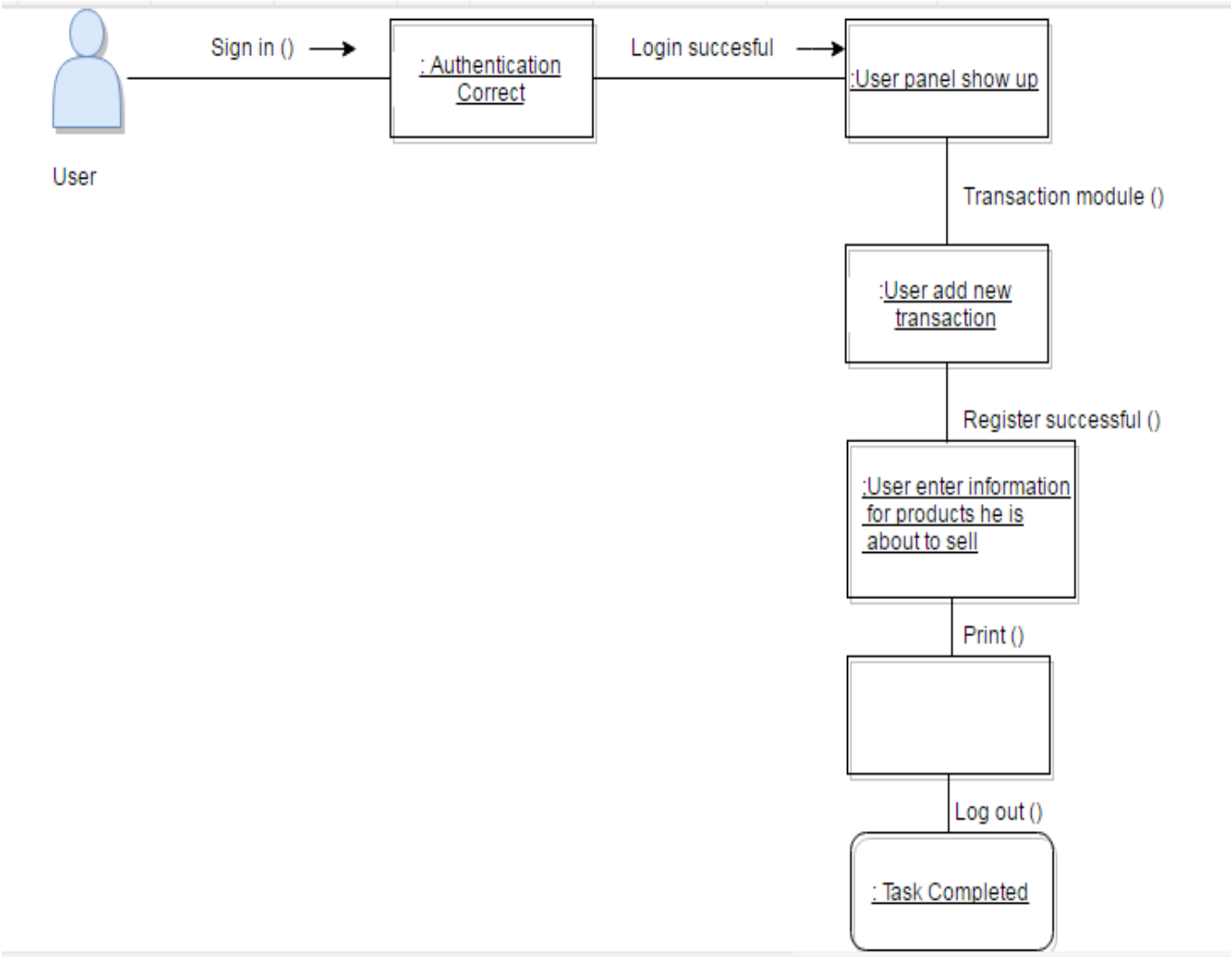
Collaboration Diagram 18 (Scenario 18)



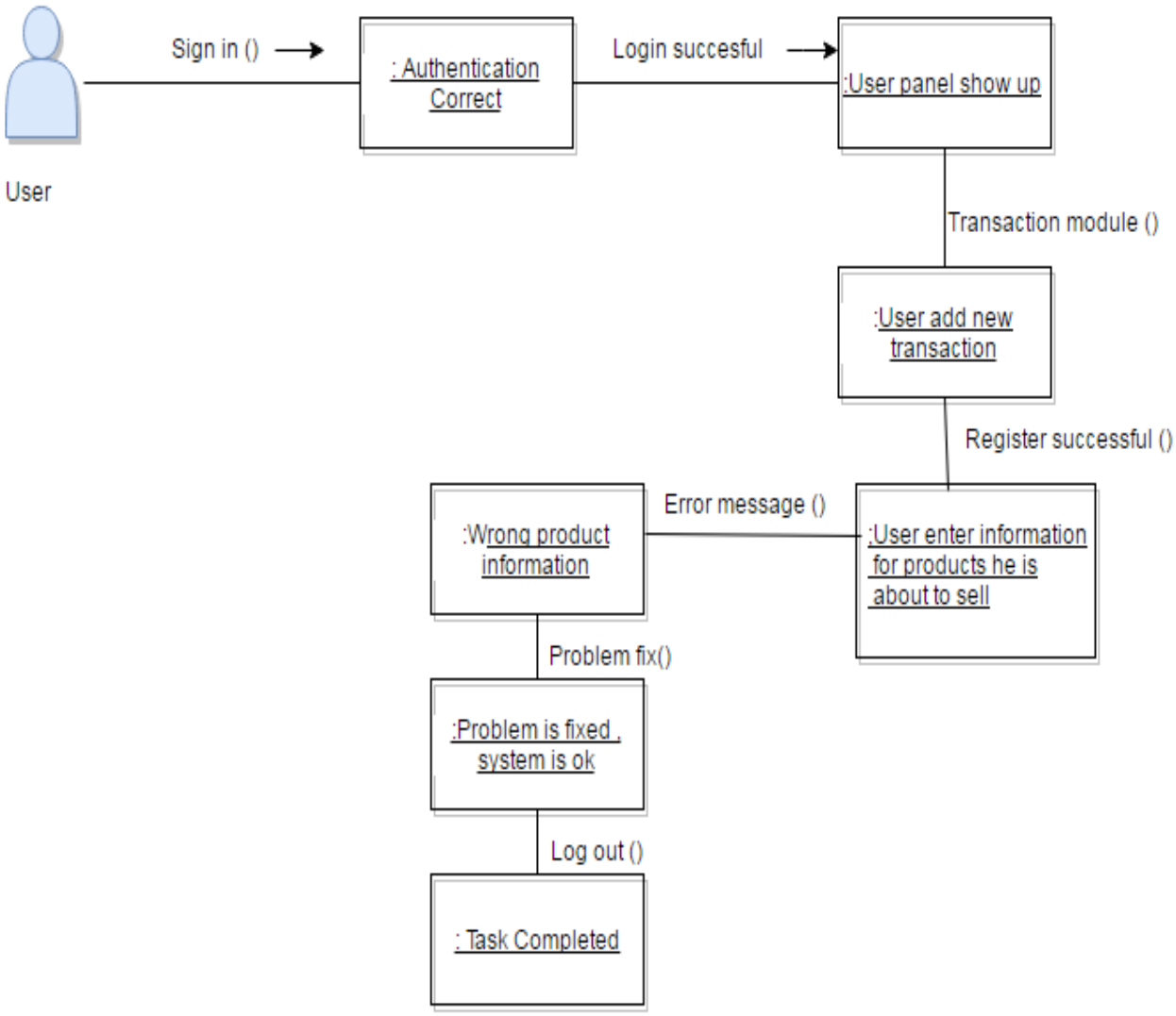
Collaboration Diagram 19 (Scenario 19)



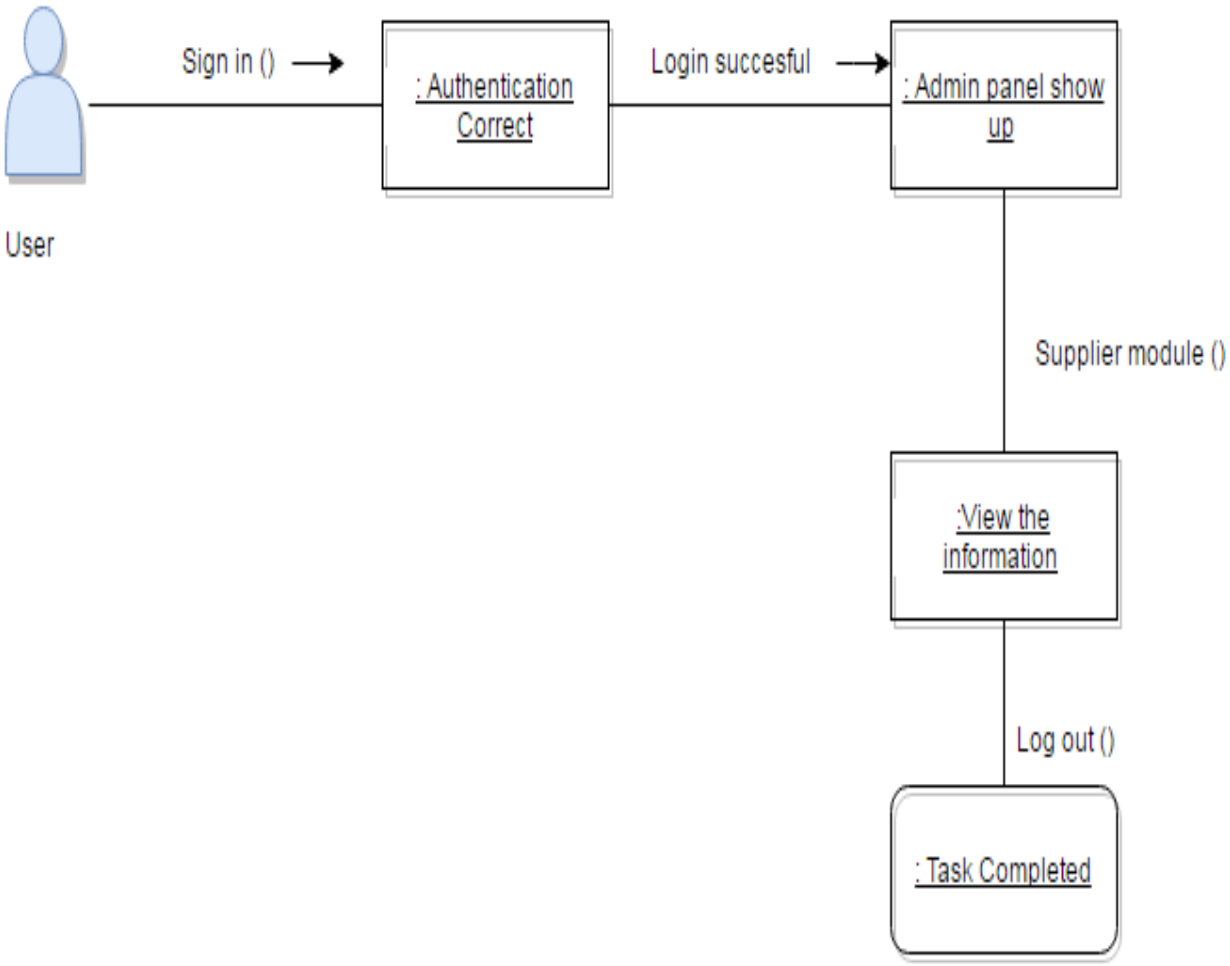
Collaboration Diagram 20 (Scenario 20)



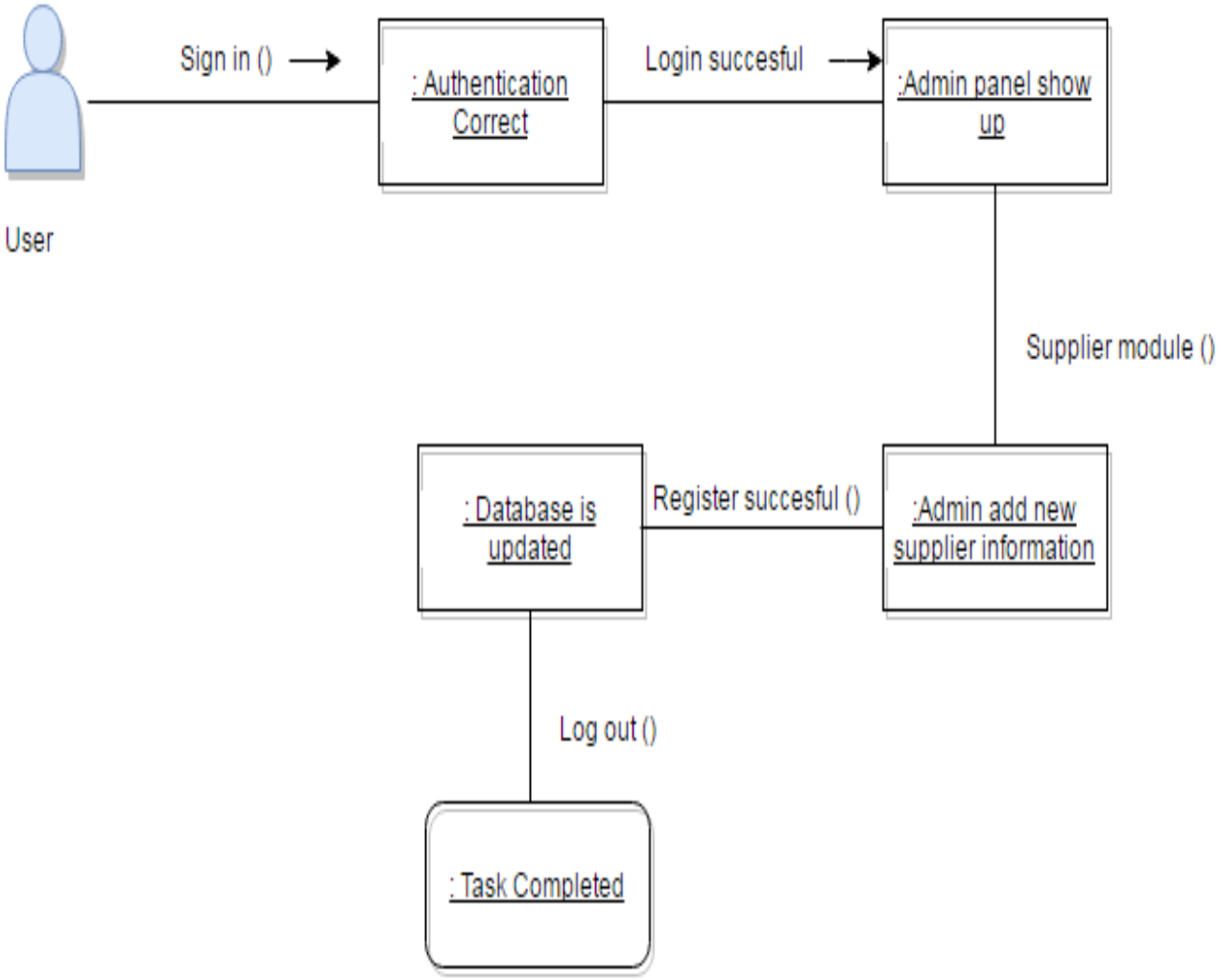
Collaboration Diagram 21 (Scenario 21)



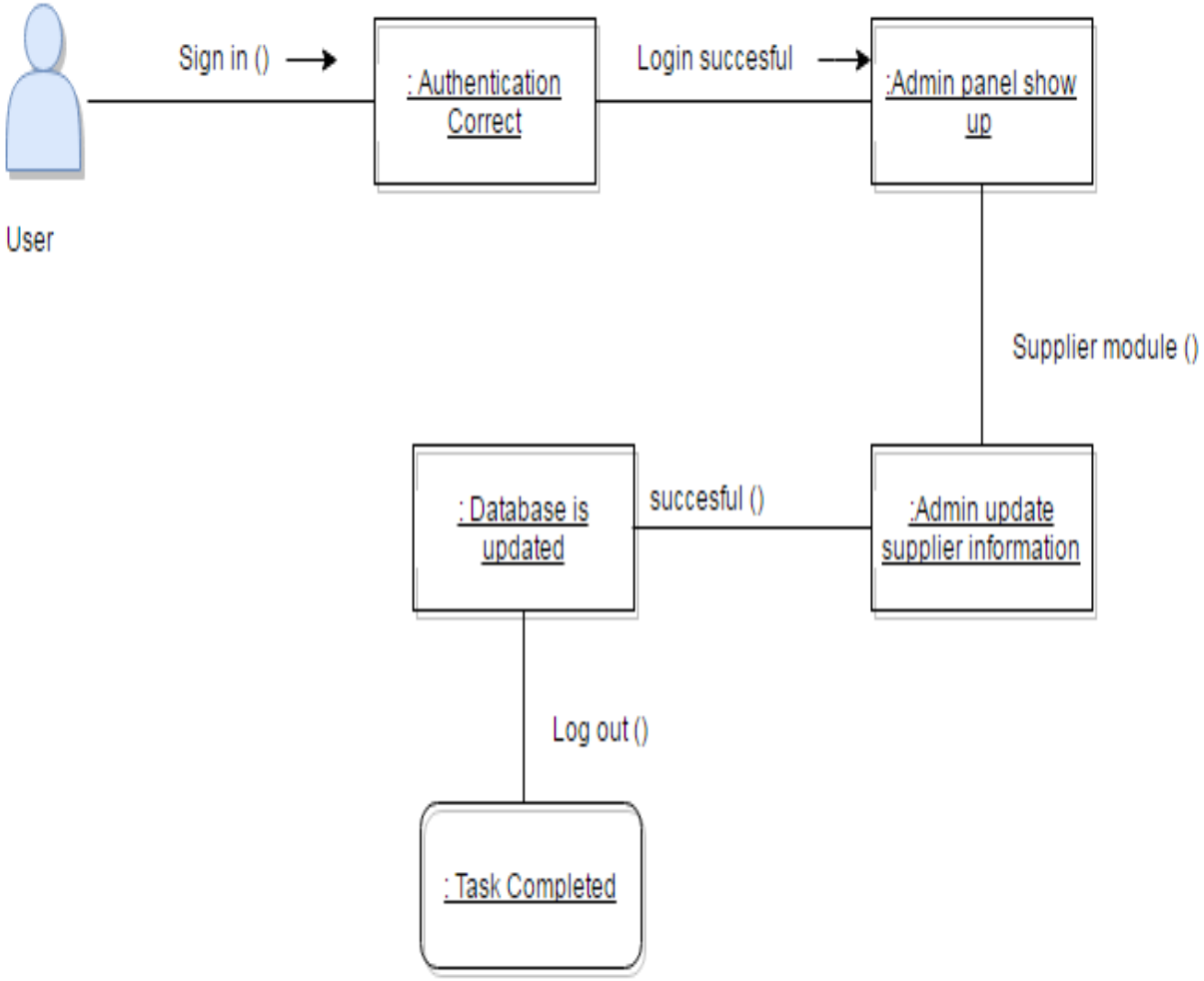
Collaboration Diagram 22 (Scenario 22)



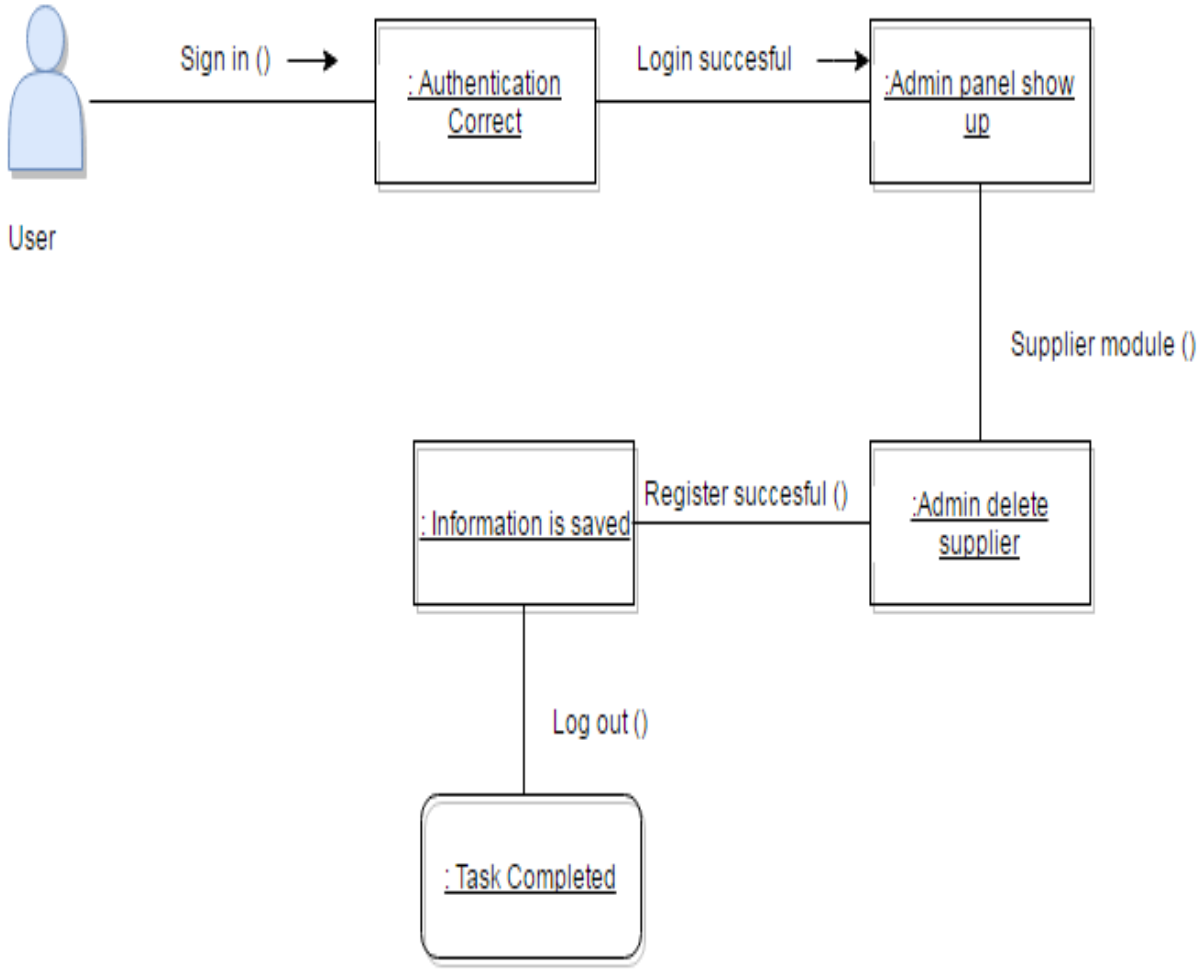
Collaboration Diagram 23 (Scenario 23)



Collaboration Diagram 24 (Scenario 24)

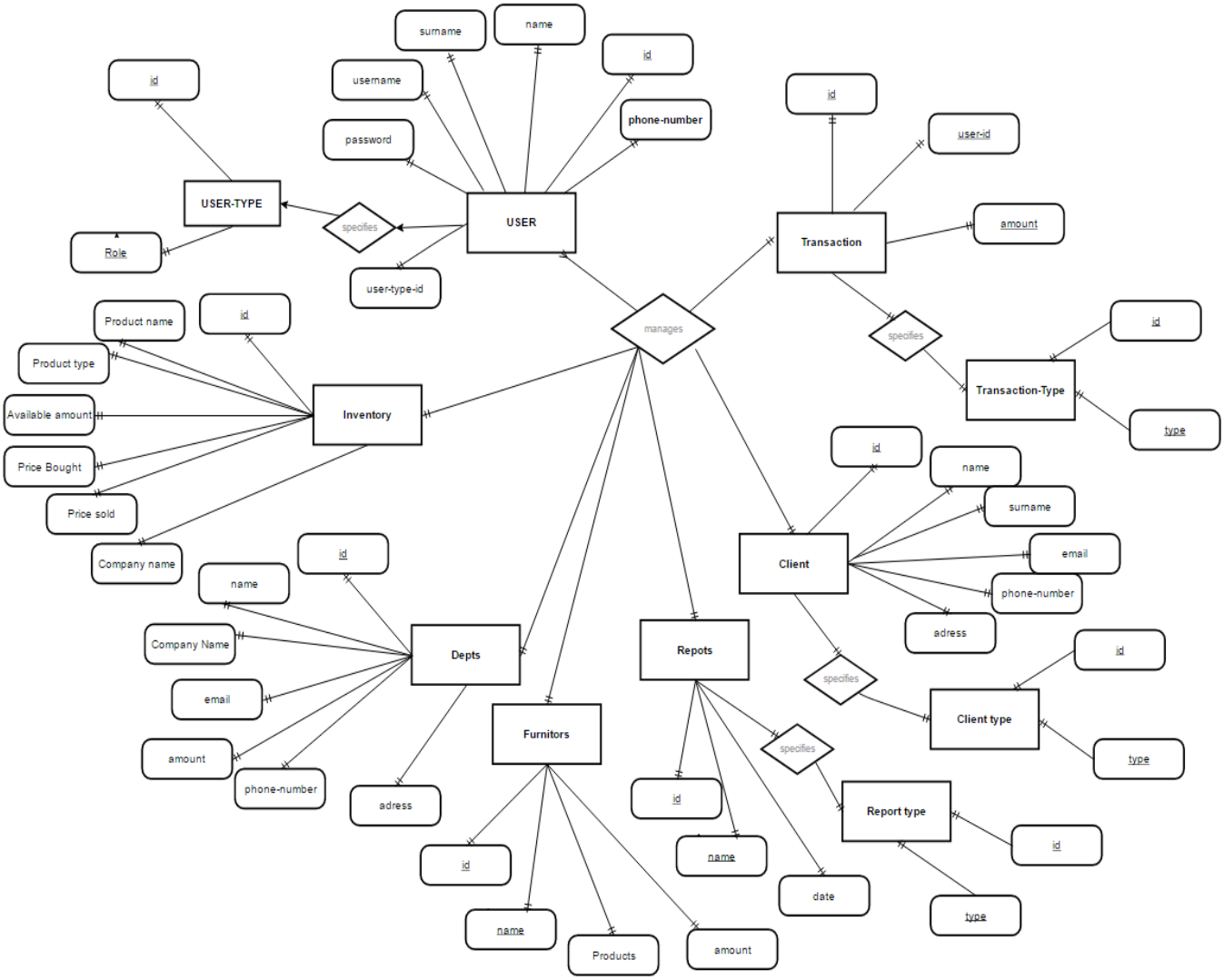


Collaboration Diagram 25 (Scenario 25)



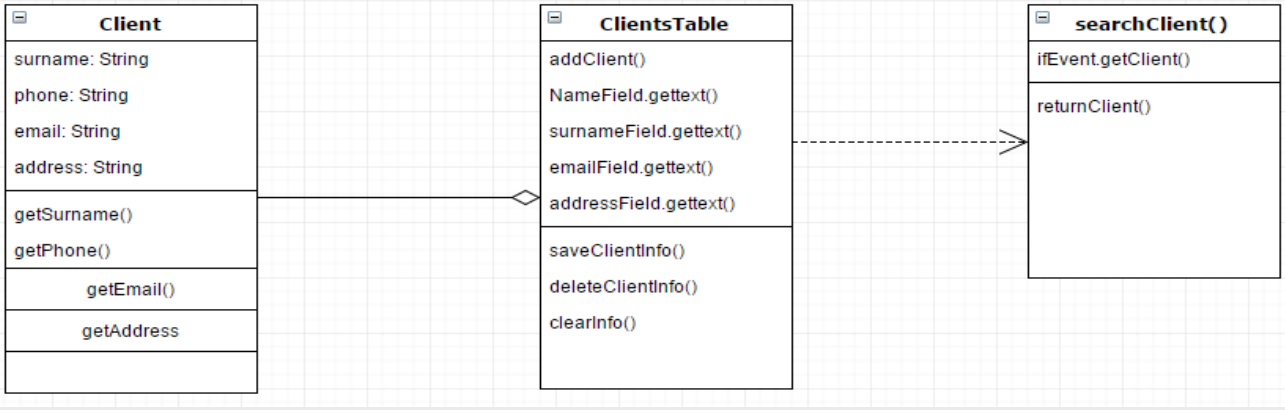
Collaboration Diagram 26 (Scenario 26)

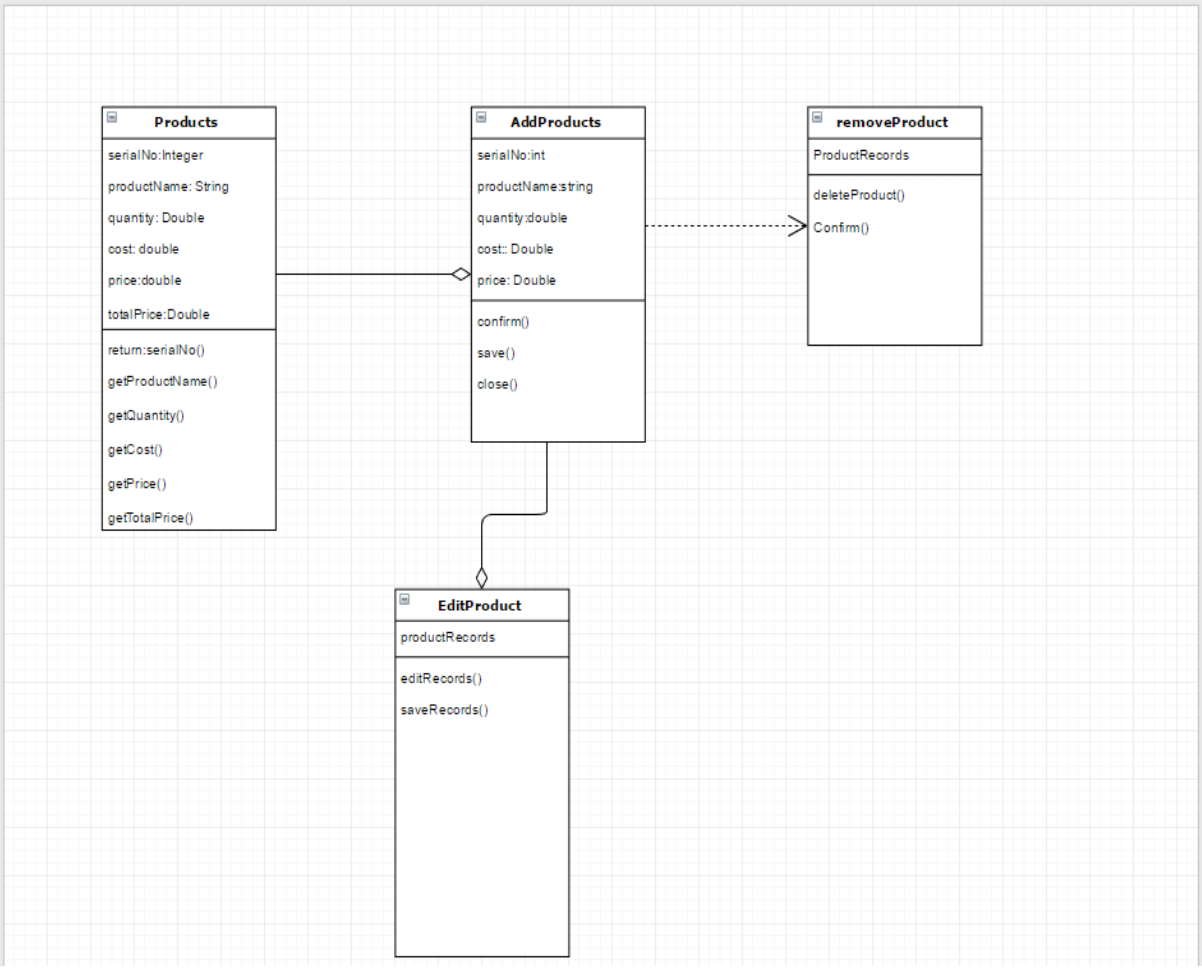
8.6 Entity Relationship Diagram



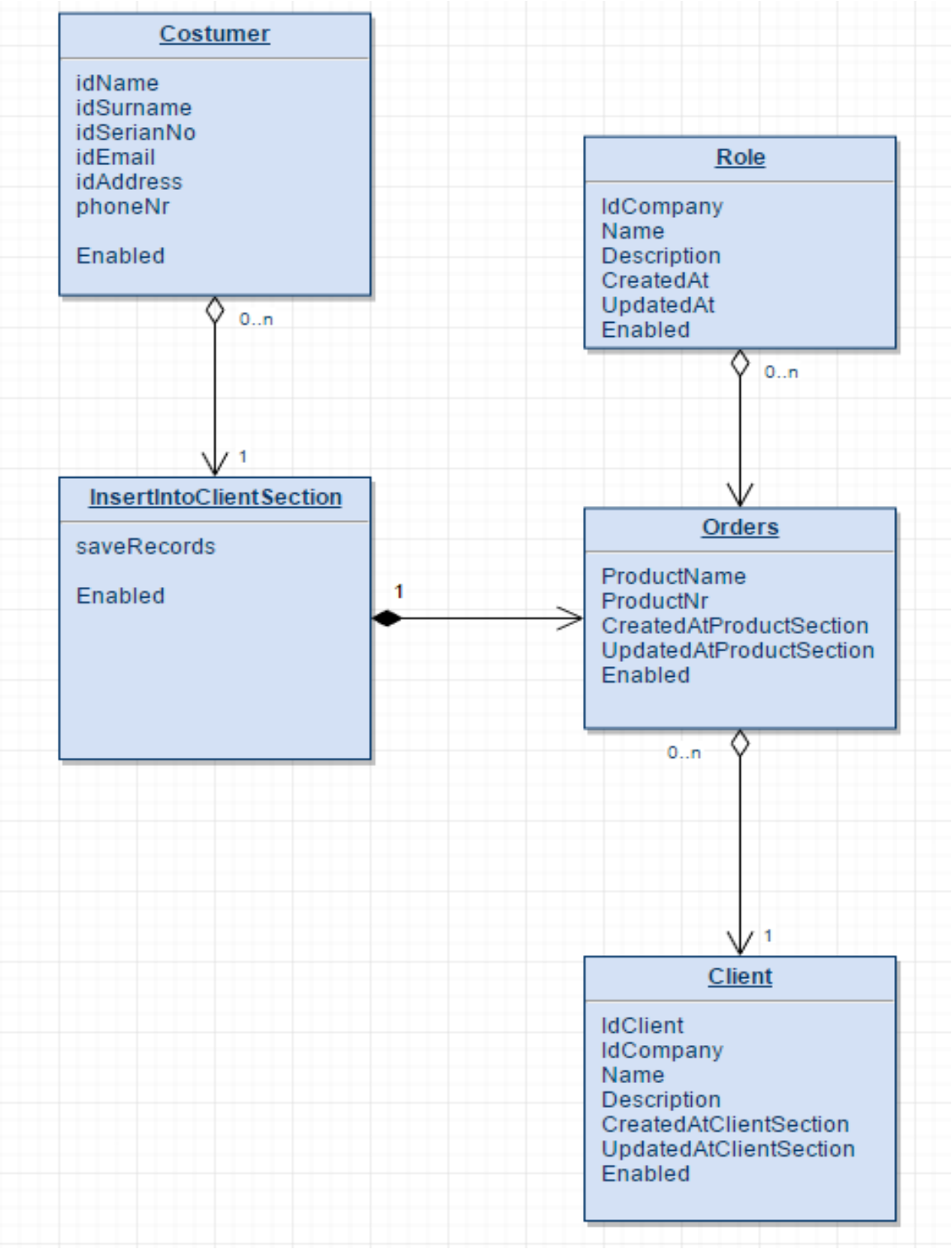
8.7 Database Schema

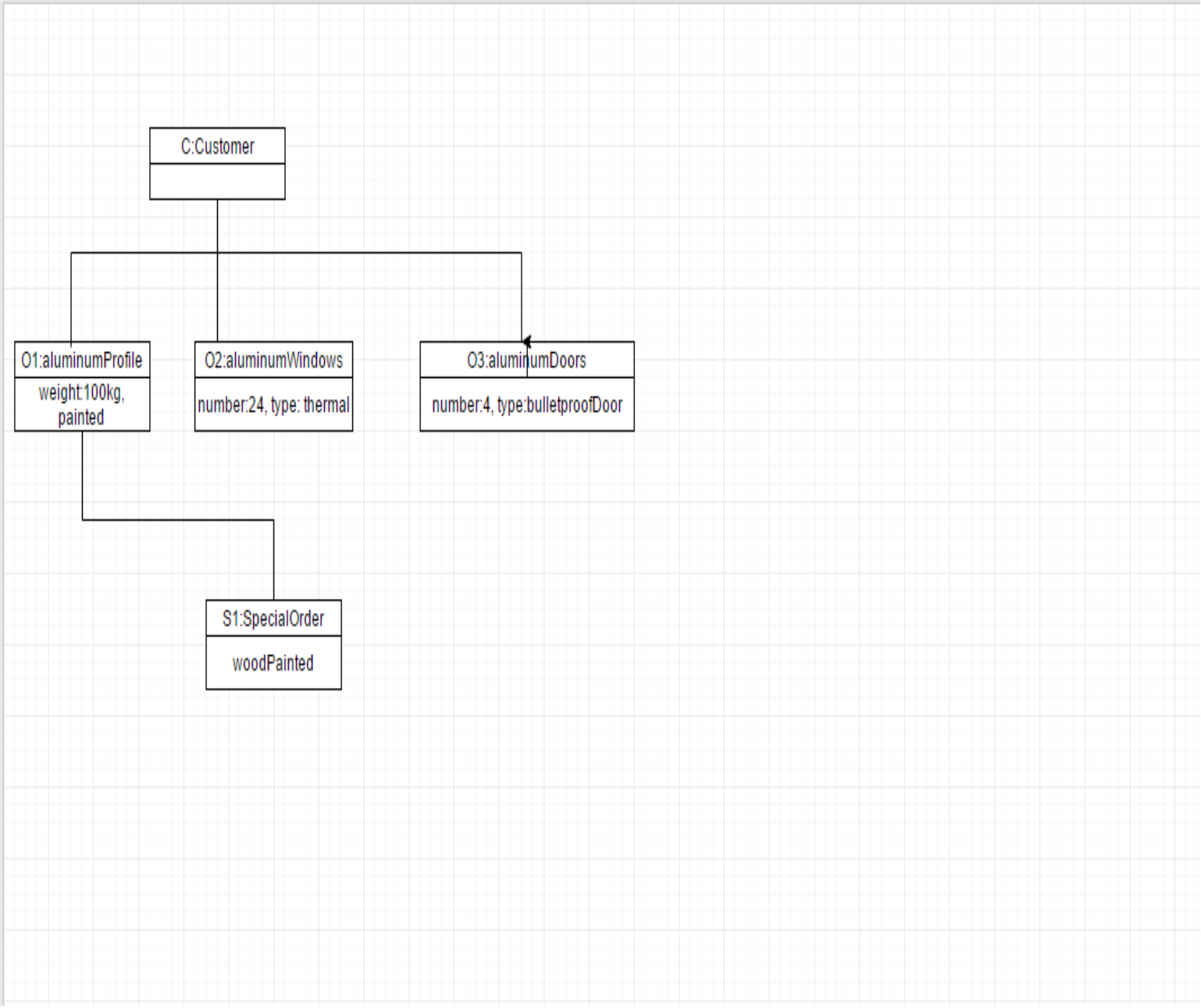
8.8 Class Diagram



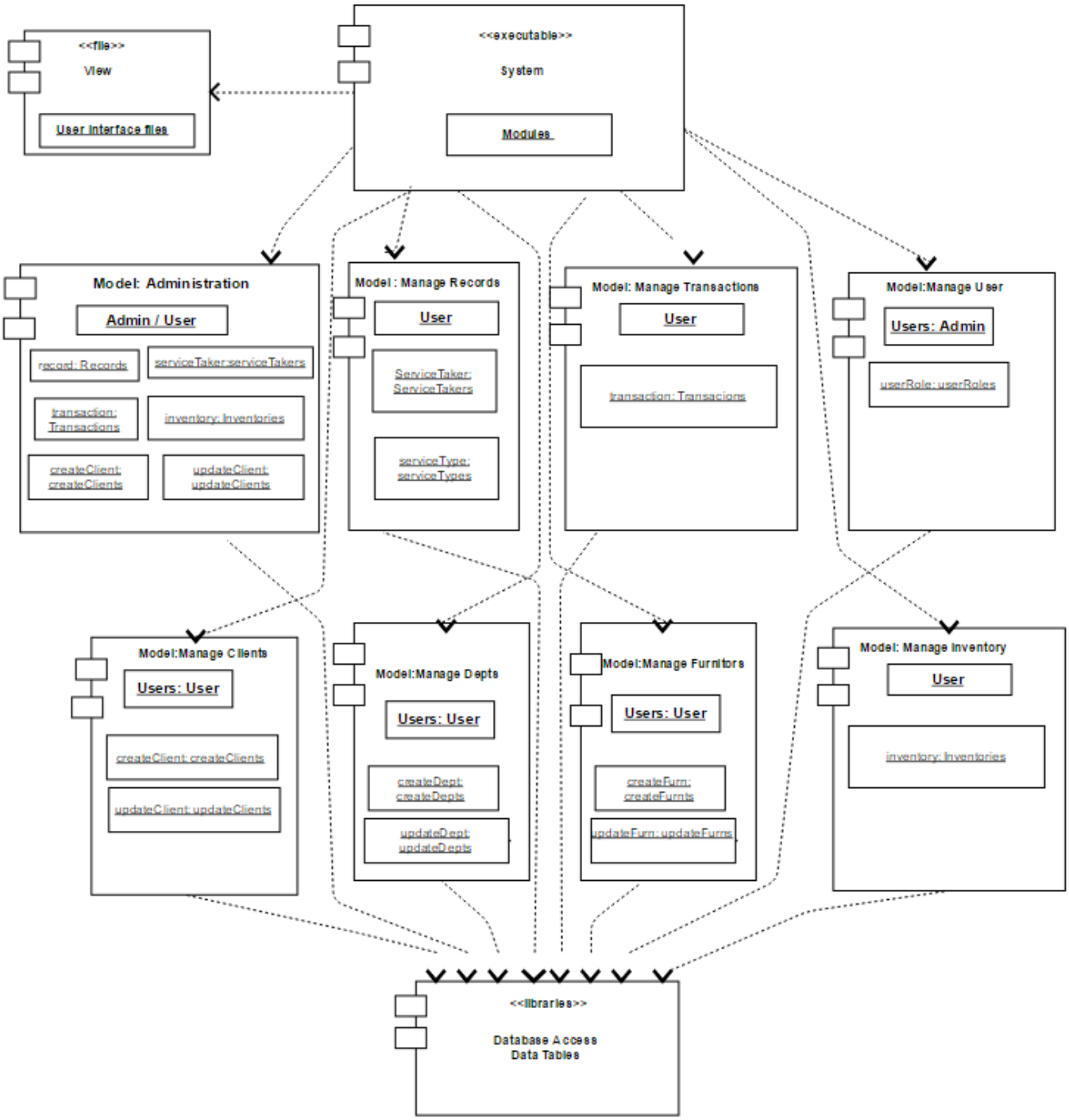


8.9 Object Diagram

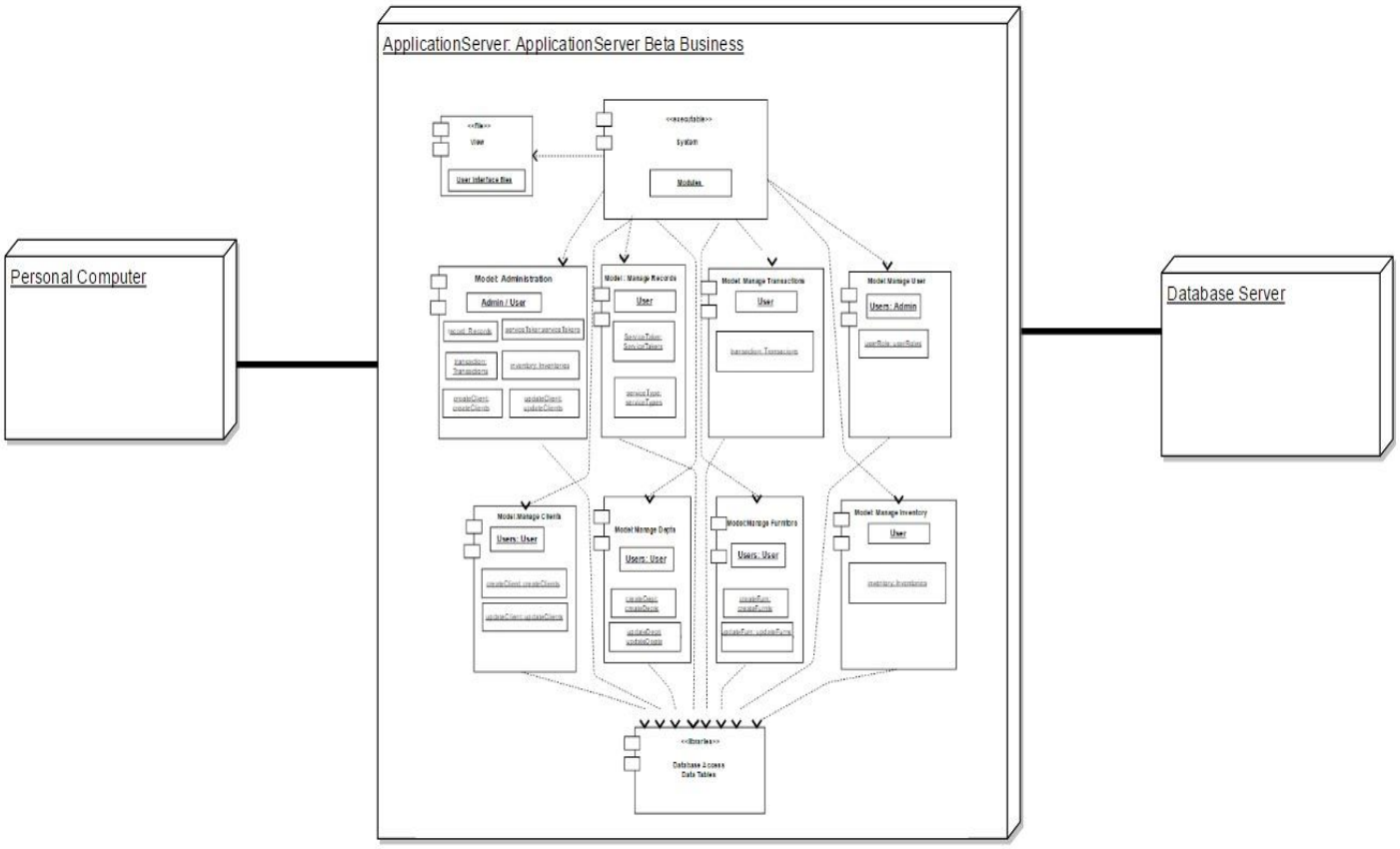



C:Customer
O1:aluminumProfile
weight:100kg, painted
O2:aluminumWindows
number:24, type: thermal
O3:aluminumDoors
number:4, type:bulletproofDoor
S1:SpecialOrder
woodPainted

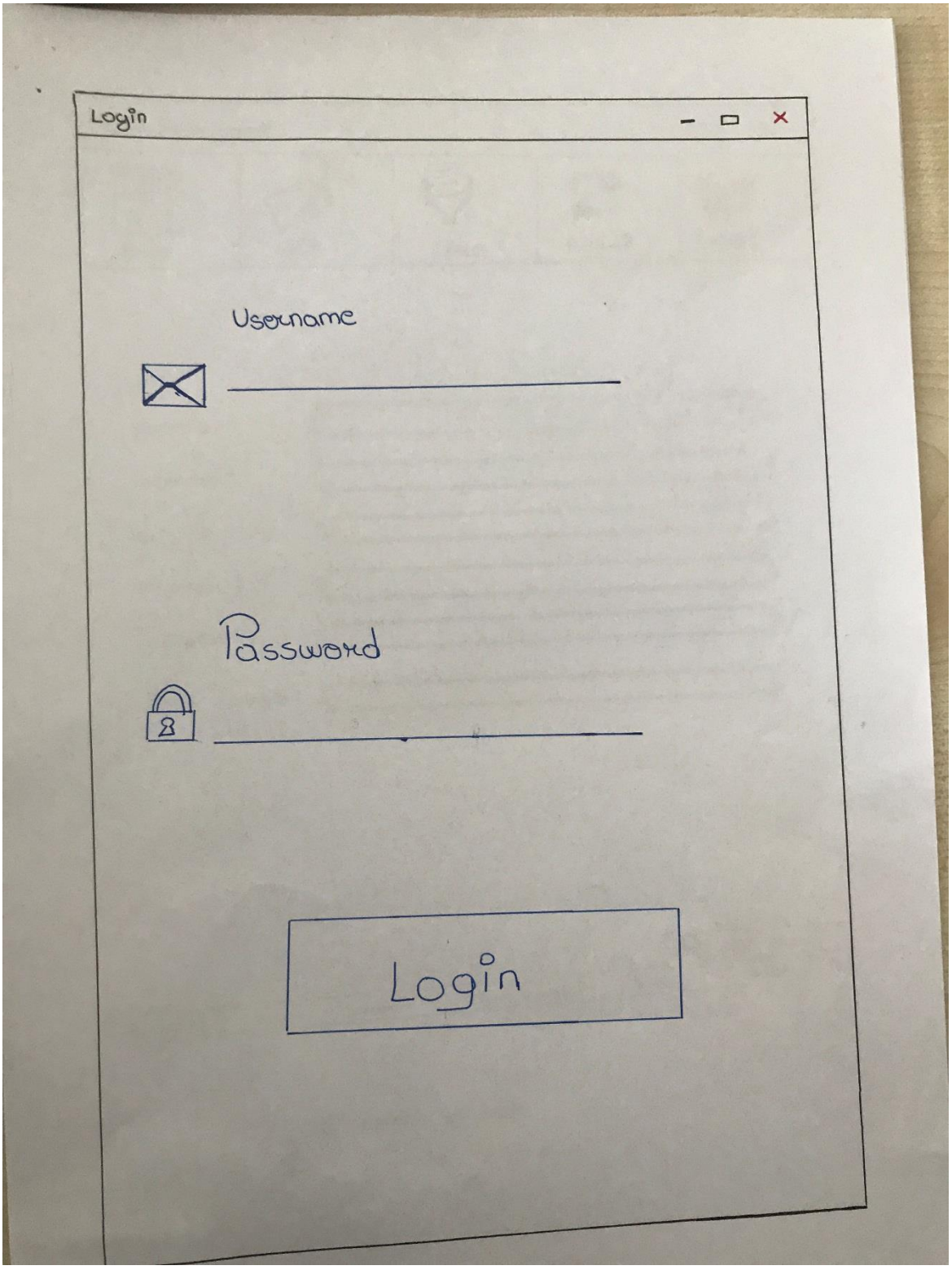
8.10 Component Diagram

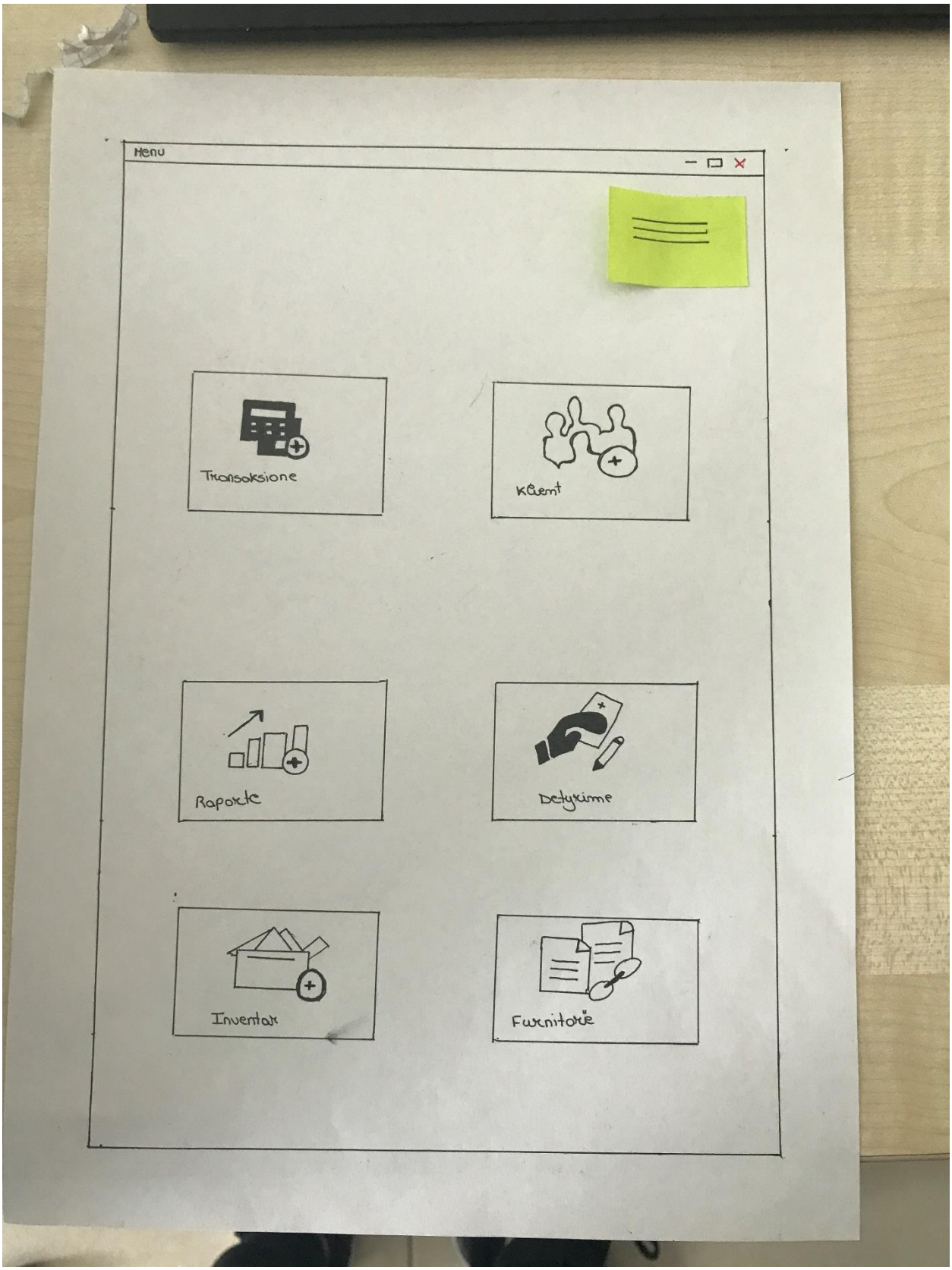


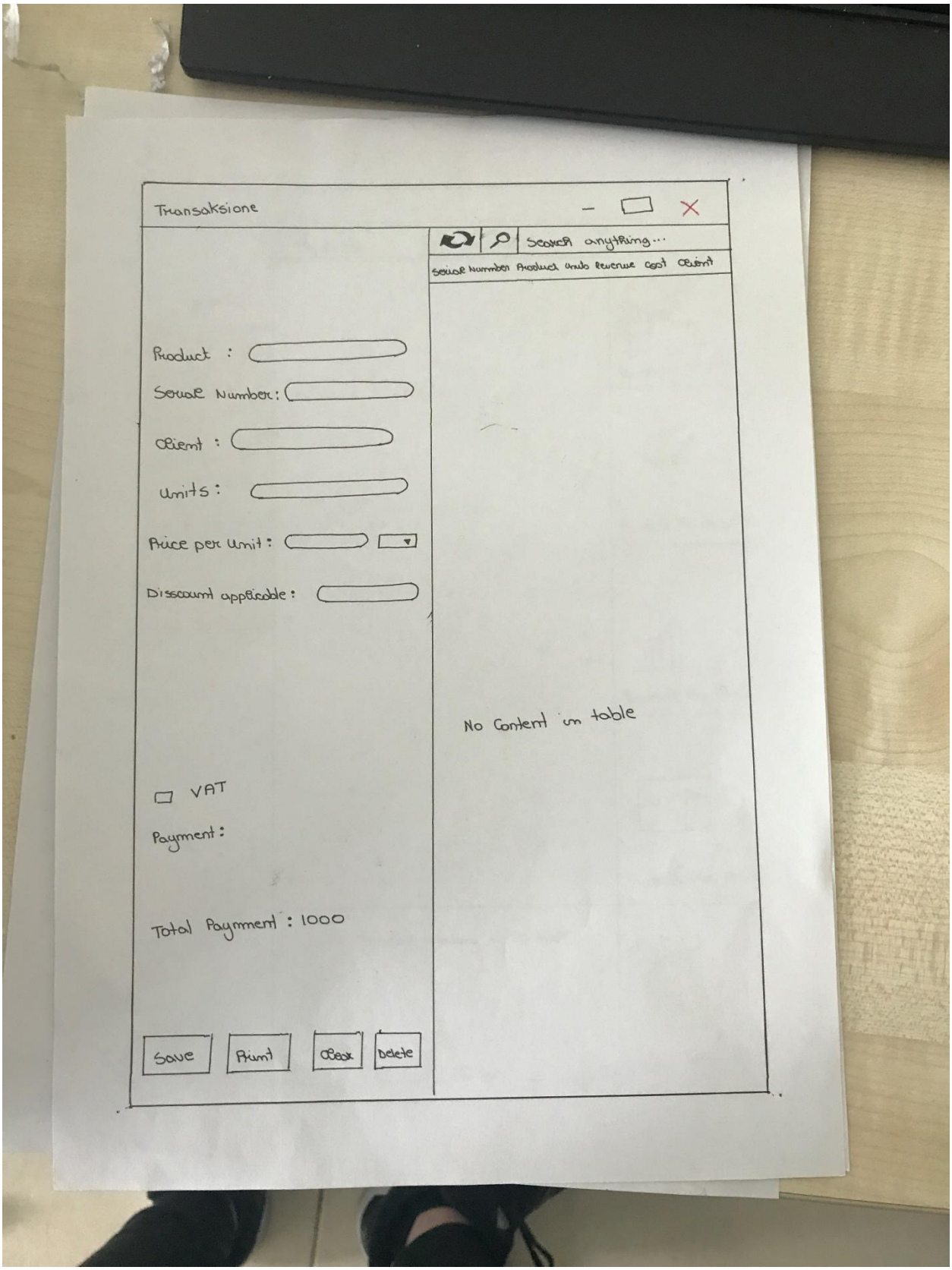
8.11 Deployment Diagram

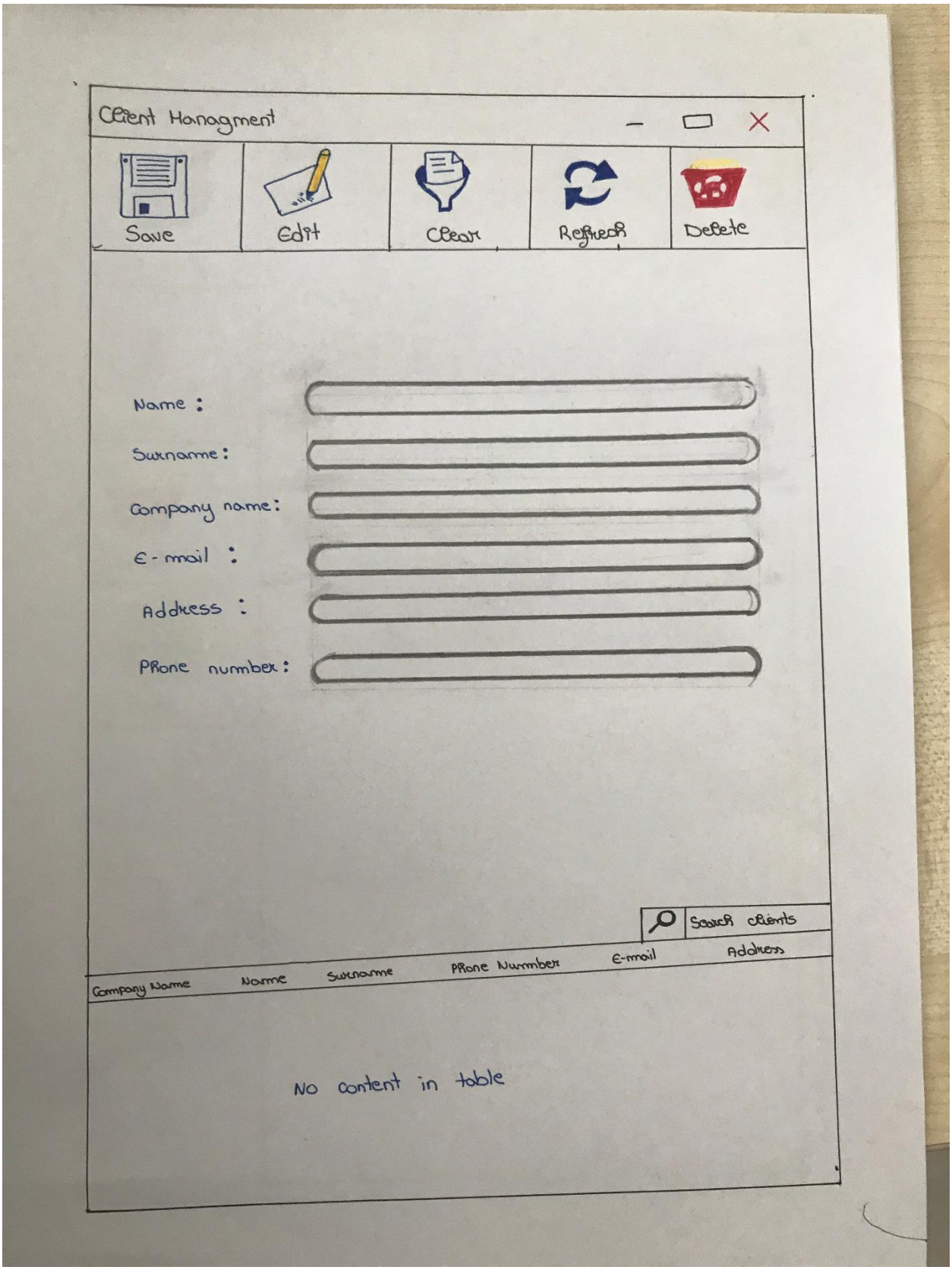


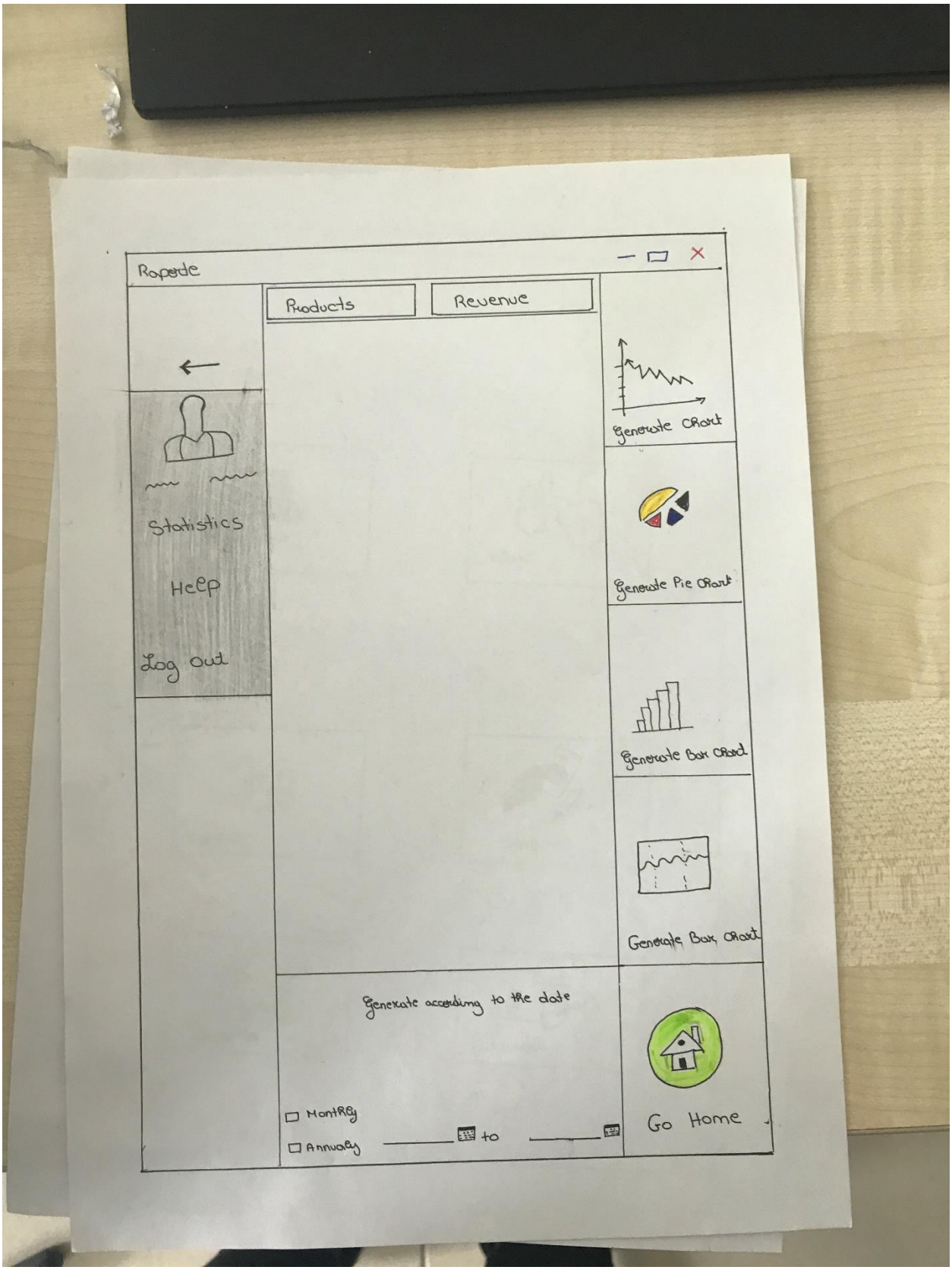
9. Sketches

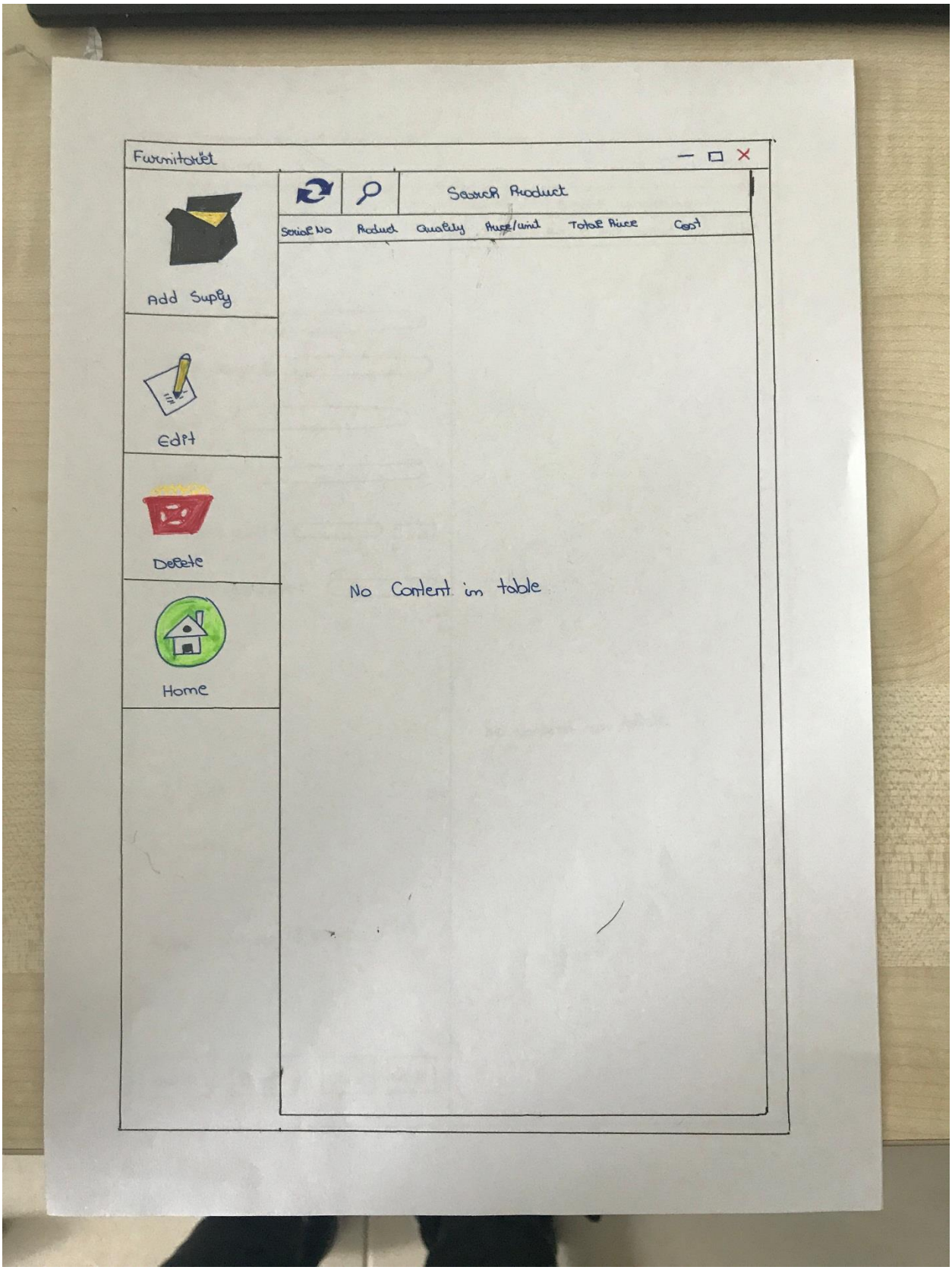



Menu
Transaksione
Klient
Raporte
Detyrime
Inventar
Furnitore


Transaksione
Search anything...
Serial Number Product Units Revenue Cost Client
Product :
Serial Number:
Client :
Units :
Price per Unit :
Discount applicable :
No Content in table
VAT
Payment :
Total Payment : 1000
Save
Print
Clear
Delete


Client Hanagment
Save
Edit
Clear
Refresh
Delete
Name :
Surname :
Company name :
E-mail :
Address :
Phone number :
Search clients
Company Name
Name
Surname
Phone Nummber
E-mail
Address
No content in table


Raporde
Products
Revenue
Statistics
Help
Log out
Generate Chart
Generate Pie Chart
Generate Bar Chart
Generate Bar Chart
Generate according to the date
Monthly
Annualy
to
Go Home


Furnitaret
Search Product
Serial No
Product
Quality
Price/unit
Total Price
Cost
Add Supply
Edit
Delete
Home
No Content in table

10. Software Testing

The framework that is used to build in this software is JavaFX. Using this framework was more efficient and reliable because we could test our code at any time, and with the scene builder we could design the program easily.

General Testing is made below:

Testing the Client:

```
package helloworld;

public class Client {

      public String fname;
 public String surname;
 public String phone;
 public String email;
 public String address;

 public Client(String fname,String surname,String phone, String email,String
address ) {

            this.fname=fname;
            this.surname=surname;
            this.phone=phone;
            this.email=email;
            this.address=address;
      }



 public String getFname(){
            return fname;
      }

 public String getSurname(){
            return surname;
      }

      public String getPhone(){
            return phone;
      }

      public String getEmail(){
            return email;
      }

      public String getAddress(){
            return address;
```

```
        }
}
package helloworld;

import java.awt.Color;
import java.net.URL;
import java.util.ResourceBundle;

import javax.management.Notification;

import org.controlsfx.control.Notifications;

import com.jfoenix.controls.JFXButton;
import com.jfoenix.controls.JFXTextField;

import eu.hansolo.enzo.notification.Notification.Notifier;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.geometry.Pos;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.MouseEvent;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Text;
import javafx.util.Duration;
public class ClientsTable implements Initializable{



        @FXML
        private TextField NameField;

        @FXML
        private TextField SurnameField;

        @FXML
        private TextField PhoneField;

        @FXML
        private TextField EmailField;

        @FXML
        private TextField CountryField;
```

```
        @FXML
        private TextField AddressField;

       @FXML
       private Text ClientSearchFieldClear;


        @FXML
        private TableView<Client> ClientTable;

       // private
ObservableList<Client>data=FXCollections.observableArrayList();


        @FXML
        private TableColumn<?, ?> nameColumn;

        @FXML
        private TableColumn<?, ?> surnameColumn;;

        @FXML
        private TableColumn<?, ?>  phoneColumn;

        @FXML
        private TableColumn<?, ?>  emailColumn;

        @FXML
        private TableColumn<?, ?>   addressColumn;


        @FXML
        private Button ClearButton;

        @FXML
        private Button RefreshButton;

        @FXML
        private Text Validator;

        @FXML
        private Button SaveButton ;

        @FXML
        private ImageView NotificationImage;

        // adding a new client to database through client menu along with
validations

            public void AddClient(ActionEvent event){
                  if(event.getSource()==SaveButton) {
                   boolean success = false;
```

```
                if(   NameField.getText().isEmpty()
                      || SurnameField.getText().isEmpty()
                      || EmailField.getText().isEmpty()
                      ||PhoneField.getText().isEmpty()
                  ){
            //    Validator.setVisible(true);
                }
                else  {
                      //Validator.setVisible(false);
                 success = Database.AddClientDb(

                            NameField.getText(),
                            SurnameField.getText(),
                            PhoneField.getText(),
                            EmailField.getText(),
                            AddressField.getText()
                            );

                // Create a custom Notification without icon
                 //  Image img=new Image("Ok-96.png");
                  // ImageView view=new ImageView(img);


                  Notifier.INSTANCE.notifySuccess("Success", "All detailes
were saved successfully!");

                // Notifications notification=Notifications.create()



                      //  .title("Success")
                      //  .text("All detailes were saved successfully!")

                      //  .hideAfter(Duration.seconds(5))
                //      .position(Pos.BOTTOM_RIGHT);

                      // notification.darkStyle();
                      //  notification.showError();

                // Alert alert = new Alert(AlertType.CONFIRMATION);
                //    alert.setTitle("Success");
                //    alert.setHeaderText("Details were saved
successfully!");
                      //alert.show();

                      NameField.clear();
                   SurnameField.clear();
                 EmailField.clear();
                PhoneField.clear();
                  AddressField.clear();
                  }

                  }

            }
```

```
    @FXML
       private void Clear(ActionEvent event) {
         if(event.getSource()==ClearButton) {
         NameField.clear();
         SurnameField.clear();

         EmailField.clear();
         PhoneField.clear();
         AddressField.clear();


         }
       }


      private void updateClientTable(String search) throws
NullPointerException{


            nameColumn.setCellValueFactory(new
PropertyValueFactory<>("fname"));
            surnameColumn.setCellValueFactory(new
PropertyValueFactory<>("surname"));
            phoneColumn.setCellValueFactory(new
PropertyValueFactory<>("phone"));
            emailColumn.setCellValueFactory(new
PropertyValueFactory<>("email"));
            addressColumn.setCellValueFactory(new
PropertyValueFactory<>("address"));


            nameColumn.setStyle( "-fx-alignment: CENTER;");
            surnameColumn.setStyle( "-fx-alignment: CENTER;");
            phoneColumn.setStyle( "-fx-alignment: CENTER;");
            emailColumn.setStyle( "-fx-alignment: CENTER;");
            addressColumn.setStyle( "-fx-alignment: CENTER;");

            ClientTable.getItems().clear();
//(companyColumn,nameColumn,surnameColumn,phoneColumn,emailColumn,addressColu
mn);
            ClientTable.setItems(Database.getClientTable(search) );
      }

 @FXML
 private JFXTextField ClientSearchField;

 @FXML
 private Button ClientSearchButton;

  // @FXML
      //private void SearchClient(ActionEvent event){
      //    if(event.getSource()==ClientSearchButton){
      //    String search = ClientSearchField.getText();
      //    System.out.println(search);
```

```
	//	if(search.isEmpty()) return;
		//else updateClientTable(search);
	//}
	//}

 @FXML
 private void LoadDataToTable(ActionEvent event){
		try {
			if(event.getSource()==ClientSearchButton){
				String search = ClientSearchField.getText();
				System.out.println("Shqipe we made it");
				updateClientTable(search);
			}
		}catch(Exception e){
			e.printStackTrace();
		}
}


@FXML
 private void RefreshDataToTable(ActionEvent event){
		try {
			if(event.getSource()==RefreshButton){
//				String search = ClientSearchField.getText();
				System.out.println("Shqipe we made it");
				updateClientTable(null);
			}
		}catch(Exception e){
			e.printStackTrace();
		}
}

@FXML
private void ClientSearchClear(KeyEvent event){
	if(ClientSearchField.getText().isEmpty()) {
		ClientSearchFieldClear.setVisible(false);
	}else if (  !ClientSearchField.getText().isEmpty()) {
		ClientSearchFieldClear.setVisible(true);
	}

	if (ClientSearchField.isFocused()){
		ClientSearchButton.setDefaultButton(true);
	} else {
		SaveButton.setDefaultButton(true);
	}
}


@FXML
private void ClearClientSearch(MouseEvent event) {
	ClientSearchField.clear();
	ClientSearchFieldClear.setVisible(false);
}
```

```
	@Override
	public void initialize(URL arg0, ResourceBundle arg1) {
		// TODO Auto-generated method stub
		updateClientTable(null);
	}









}
```

Controller:

```
package helloworld;


import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

import org.controlsfx.control.Notifications;

import com.jfoenix.controls.*;
import com.jfoenix.transitions.hamburger.HamburgerBackArrowBasicTransition;

import eu.hansolo.enzo.notification.Notification.Notifier;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Pos;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.MouseEvent;
import javafx.scene.text.Text;
import javafx.scene.control.Button;
import javafx.scene.image.ImageView;
import javafx.stage.Stage;
import javafx.util.Duration;

import java.lang.String;
```

```
public class Controller  implements Initializable {



      public static int user=0;


      @FXML
      private Button konfirmo;

      @FXML
      private JFXTextField username;

      @FXML
      private JFXPasswordField password;

      @FXML
      private Text error;

      @FXML
      private Text error1;


           @FXML
            private JFXHamburger hamburger;

            @FXML
            private JFXDrawer drawer;



       @FXML
       private JFXButton TransactionMenu;

       @FXML
       private JFXButton ReportMenu;

       @FXML
       private JFXButton DebtMenu;

       @FXML
       private JFXButton InventoryMenu;

       @FXML
       private JFXButton SupplyMenu;

       @FXML
       private JFXButton HRMenu;

       @FXML
       private JFXButton ClientMenu;

       @FXML
```

```
        private JFXTextField PasswordTextField;

        @FXML
        private JFXCheckBox PasswordCheckBox;

        @FXML
        private ImageView UnmaskPasswordButton;









        @FXML
        private  void HamEffect(MouseEvent event) {

             //    boolean clicked=false;
                  //HamburgerBackArrowBasicTransition transition = new
HamburgerBackArrowBasicTransition(hamburger);
//transition.setRate(-1);
      //           if (hamburger.isPressed()){
             //    clicked=true;
                  //transition.setRate(transition.getRate()* -1);
                  //transition.play();

                  if (drawer.isShown()) {
                        drawer.close();
                  } else {
                        drawer.open();

                  }

             }



        //open transactionmenu
        @FXML
        private void OpenTransaction(ActionEvent event) throws IOException {
              try {
              if(event.getSource()==TransactionMenu){
                    Stage stage=new Stage();
                    Parent
root=FXMLLoader.load(getClass().getResource("sale.fxml"));
                         Scene scene=new Scene(root,1255,871);

                         stage.setScene(scene);
                         stage.setTitle("Sales");
                         stage.setResizable(false);
                  stage.show();
              }
              }catch(Exception e){
```

```
                        System.out.println("Error : " + e.getMessage());
                              System.out.println("Caused By : " +
e.getCause());
                }

        }



      //open client menu
      @FXML
      private void OpenClient(ActionEvent e)  {
            try {
            if(e.getSource()==ClientMenu) {
                  Stage stage=new Stage();
                   Parent
root=FXMLLoader.load(getClass().getResource("client.fxml"));
                        Scene scene=new Scene(root,1438,748);

                        stage.setScene(scene);
                        stage.setTitle("Client Management");
                        stage.setResizable(false);
                  stage.show();
            }   }  catch(Exception error) {

                  System.out.println("Error : " + error.getMessage());
                  System.out.println("Caused by : " + error.getCause());
            }


            }

      //open report menu
      @FXML
      private void OpenReports(ActionEvent event) {

            try {
                  if(event.getSource()==ReportMenu) {
                        Stage stage=new Stage();
                         Parent
root=FXMLLoader.load(getClass().getResource("report.fxml"));
                              Scene scene=new Scene(root,910,691);

                              stage.setScene(scene);
                              stage.setResizable(false);
                        stage.show();
                  }   }  catch(Exception error) {

                        System.out.println("Error : " + error.getMessage());
                        System.out.println("Caused by : " +
error.getCause());
                  }


                  }
```

```
	//open inventory menu
	@FXML
	private void OpenInventory(ActionEvent event){
		try {
			if(event.getSource()==InventoryMenu) {


				Stage stage=new Stage();
				 Parent
root=FXMLLoader.load(getClass().getResource("Inventory.fxml"));
					Scene scene=new Scene(root,1036,693);

					stage.setScene(scene);
					stage.setResizable(false);
				stage.show();


			}   }  catch(Exception error) {

				System.out.println("Error : " + error.getMessage());
				System.out.println("Caused by : " +
error.getCause());
			}



		}


	@FXML
	private void OpenHRMenu(ActionEvent event){
		try {
			if(event.getSource()==HRMenu) {
				Stage stage=new Stage();
				 Parent
root=FXMLLoader.load(getClass().getResource("hr.fxml"));
					Scene scene=new Scene(root,1155,715);

					stage.setScene(scene);
					stage.setResizable(false);
				stage.show();
			}   }  catch(Exception error) {

				System.out.println("Error : " + error.getMessage());
				System.out.println("Caused by : " +
error.getCause());
			}



		}
```

```
      @FXML
      private void Login(ActionEvent event)  {
            try {

            if( validation(username.getText(), 1) &&
validation(password.getText(), 2)
                        && Database.logini(username.getText(),
password.getText())  ){
            //if(event.getSource()==konfirmo) {
                  //if(username.getText().equals("ernaldo") &&
password.getText().equals("gjomakaj")){

                  user=Database.user;

                  //mbyll stage e loginit
                  Stage a = (Stage)
((Node)(event.getSource())).getScene().getWindow();
                  a.close();
                  //nderto dhe loado skenen e mainit

                  Stage stage=new Stage();
               Parent
root=FXMLLoader.load(getClass().getResource("demoMain.fxml"));
                   Scene scene=new Scene(root,1178,735);

                   stage.setScene(scene);
            stage.show();

            }
            else if(username.getText().isEmpty() ||
password.getText().isEmpty()){
                  Notifications notification=Notifications.create()
                          .title("Warning")
                          .text("All field are required!")
                         //.graphic(view)
                          .hideAfter(Duration.seconds(2))
                          .position(Pos.CENTER);

                         notification.darkStyle();
                          notification.showWarning();



            }
             else {
                        System.out.println("Wrong username or password!");
                        Notifications notification=Notifications.create()
                                .title("Error!")
                                .text("Wrong username or password!")
                               //.graphic(view)
```

```
                                .hideAfter(Duration.seconds(2))
                                .position(Pos.CENTER);

                               notification.darkStyle();
                                notification.showError();

                          }

            }catch(Exception e){
            //    System.out.println("Error : " + e.getMessage());
            //    System.out.println("Caused By : " + e.getCause());
            }
      }



      //validim me regex
 public  boolean validation(String input,int type){

       boolean validationSuccessful=false;

       String regex = null;
            if(type==1){
                  regex = "^[a-zA-Z]{0,}$"; // words
                  //regex="[A-Za-z0-9]"; // only letters and numbers
(username)
            }if(type==2){
                   // words
                  regex = "^[a-zA-Z0-9]+$"; //numbers
                  //regex="/^(?=.*[a-z0-9])[a-z0-9!@#$%&*.]{7,}$/i";
//letters and numbers w/special characters(password)
            }

            if(input.matches(regex)){
                  validationSuccessful = true;
            }
            return validationSuccessful;
      }



 public void validimi(KeyEvent event) throws IllegalArgumentException {

       konfirmo.setDefaultButton(true);
            if(event.getSource()==username){

  if(validation(username.getText(),1)) {

        error.setVisible(false);
  } else {
        error.setVisible(true);
```

```
  }
            }
  if(event.getSource()==password){
  if (validation(password.getText(),2) || password.getText().isEmpty()){
        error1.setVisible(false);
  } else {

        error1.setVisible(true);

  }
 }
 }



      // rregjistro produktet te inventory



      @FXML
      private JFXTextField TransactionSearchField;
      @FXML
      private Text TransactionSearchFieldClear;

                @FXML
                  private void Field(KeyEvent event){
                        if(TransactionSearchField.getText().isEmpty()) {
                              TransactionSearchFieldClear.setVisible(false);
                        }else if (
!TransactionSearchField.getText().isEmpty()) {
                              TransactionSearchFieldClear.setVisible(true);
                        }
                  }

                @FXML
                private void ClearSearch(MouseEvent event) {
                    TransactionSearchField.clear();
                    TransactionSearchFieldClear.setVisible(false);
                }


                @FXML private JFXComboBox<String> payment,currency;
                public ObservableList<String> paymentMethods =
FXCollections.observableArrayList("Cash ","Mirebesim","me keste");
                public ObservableList<String> currencies =
FXCollections.observableArrayList("ALL ","$ Dollar","€ Euro");



                @FXML
                private void Payment(){
                    payment.setItems(paymentMethods);

                }
```

```
                @FXML
                private void Currencies(){
                    currency.setItems(currencies);

                }



                //unmask the password at the login when the checkbox is
checked ..

                @FXML
                private void UnmaskPassword(){

            //  if(UnmaskPasswordButton.isPressed()){
              //  PasswordTextField.setVisible(false);

                  //
PasswordTextField.managedProperty().bind(PasswordCheckBox.selectedProperty())
;
                  //
PasswordTextField.visibleProperty().bind(PasswordCheckBox.selectedProperty())
;


PasswordTextField.visibleProperty().bind(UnmaskPasswordButton.pressedProperty
());

                    //
password.managedProperty().bind(PasswordCheckBox.selectedProperty().not());

password.visibleProperty().bind(UnmaskPasswordButton.pressedProperty().not())
;

                      // Bind the textField and passwordField text values
bidirectionally.

PasswordTextField.textProperty().bindBidirectional(password.textProperty());

                }




                @FXML
                private void passwordFieldOnAction(KeyEvent event){

                    if(password.getText().isEmpty()){
                          UnmaskPasswordButton.setVisible(false);
                    }else {
                          UnmaskPasswordButton.setVisible(true);

                    }
```

```
            }


      @Override
      public void initialize(URL location, ResourceBundle resources) {



}

}
```

Database:

```
package helloworld;
import java.sql.*;
import java.util.HashMap;
import java.util.Map;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

public class Database {

      private  static Connection conn= getConnection();
      public static int user = 0;


      public static void main(String[] args){

      }


      //metode e gjeneruar per te lidhur databasen me eclipsin

      public static Connection getConnection() {
            try{
                  Class.forName("com.mysql.jdbc.Driver");
                  //1.Get a connection to database
                  conn =
DriverManager.getConnection("jdbc:mysql://localhost/ernaldo","ernaldo","ernal
do5.");
                  System.out.println("Database connected successfully");
            }catch(Exception e){
                  System.out.println("Error : " + e.getCause());
                  e.printStackTrace();
            }
            return conn;
      }


      //logini i lidhur me databasen
```

```
	public static boolean logini(String u, String p){
		try{
			Statement stm=conn.createStatement();
			ResultSet rs=stm.executeQuery("select * from user");

			//4.Process the result set
			while(rs.next()){
				if( rs.getString("username").equals(u) &&
rs.getString("password").equals(p) ){
					user = rs.getInt("id");
					System.out.println("login true");
					return true;
				}

			}
		}catch(Exception e){
			System.out.println("Error : " + e.getCause());
			e.printStackTrace();
		}
		System.out.println("login false");
		return false;
	}



	//inserts a new client to database
	public static boolean AddClientDb( String name, String surname, String
phone,String email,String address){
		try{
			PreparedStatement pstmt = conn.prepareStatement("INSERT
INTO client( name, surname, phone, email,address) VALUES (?,?,?,?,?) ");

			pstmt.setString(1, name);
			pstmt.setString(2, surname);
			pstmt.setString(3, phone);
			pstmt.setString(4, email);
			pstmt.setString(5, address);
			//pstmt.setInt(5, user);
			if(pstmt.executeUpdate() > 0) return true;
		}catch(Exception e){
			System.out.println("Error : " + e.getCause());
			e.printStackTrace();
		}
		return false;
	}



	//loads the table and searches for clients
	public static ObservableList<Client> getClientTable(String search){
		ObservableList<Client> data =
FXCollections.observableArrayList();
		try{
			String sql = "SELECT  * FROM client ";
```

```
                  if(search!=null){
                        sql += " WHERE  name LIKE ? OR surname LIKE ? OR
phone LIKE ? OR email LIKE? OR address LIKE ? ";
                  }
                  sql += " ORDER BY timestamp DESC ";
                  PreparedStatement pstmt = conn.prepareStatement(sql);
                  if(search!=null){
                      pstmt.setString(1, "%" + search + "%");
                        pstmt.setString(2, "%" + search + "%");
                        pstmt.setString(3, "%" + search + "%");
                        pstmt.setString(4, "%" + search + "%");
                        pstmt.setString(5, "%" + search + "%");



                  }
                  ResultSet rs = pstmt.executeQuery();
                  while(rs.next()){


                        String name = rs.getString("name");
                        String surname = rs.getString("surname");
                        String phone = rs.getString("phone");
                        String email = rs.getString("email");
                        String address = rs.getString("address");
                        data.add(new Client(name, surname,
phone,email,address));
                    }
                        return data;
            }catch(Exception e){
                        System.out.println("Error : " + e.getCause());
                        e.printStackTrace();
            }
            return data;
      }


//ads a new product to database (table products)
      public static boolean AddProductDb( Integer serialnumber, String
productname, Double quantity,Double cost,Double price,Double totalprice){
            try{
                        PreparedStatement pstmt = conn.prepareStatement(
            "INSERT INTO inventory( serialnumber, productname, quantity,
cost,price,totalprice) VALUES (?,?,?,?,?,?) ");


                  pstmt.setInt(1, serialnumber);
                  pstmt.setString(2, productname);
                  pstmt.setDouble(3, quantity);
                  pstmt.setDouble(4, cost);
                  pstmt.setDouble(5, price);
                  pstmt.setDouble(6, totalprice);
                  //pstmt.setString(7, currency);
                  //pstmt.setInt(5, user);
                  if(pstmt.executeUpdate() > 0) return true;
```

```
            }catch(Exception e){
                  System.out.println("Error : " + e.getCause());
                  e.printStackTrace();
            }
            return false;
      }


      //serves to extract data from database to products table in inventory
section
      public static ObservableList<Products> getProductsTable(String search){
            ObservableList<Products> data =
FXCollections.observableArrayList();
            try{
                  String sql = "SELECT  * FROM inventory ";
                  if(search!=null){
                        sql += " WHERE   productname LIKE ? ";
                  }
                  //sql += " ORDER BY timestamp DESC ";
                  PreparedStatement pstmt = conn.prepareStatement(sql);
                  if(search!=null){
                      pstmt.setString(1, "%" + search + "%");
                  //    pstmt.setString(2, "%" + search + "%");
                        //pstmt.setString(3, "%" + search + "%");
                        //pstmt.setString(4, "%" + search + "%");
                        //pstmt.setString(5, "%" + search + "%");
                        //pstmt.setString(6, "%" + search + "%");



                  }
                  ResultSet rs = pstmt.executeQuery();
                  while(rs.next()){

                        Integer serial = rs.getInt("serialnumber");
                        serial.toString();
                        String product = rs.getString("productname");
                        Double quantity =rs.getDouble("quantity");
                        quantity.toString();
                        Double cost = rs.getDouble("cost");
                        cost.toString();
                        Double price = rs.getDouble("price");
                        price.toString();
                        Double totalprice =rs.getDouble("totalprice");
                        totalprice.toString();
                        data.add(new Products(serial, product,
quantity,cost,price,totalprice));
                  }
                  return data;
            }catch(Exception e){
                  System.out.println("Error : " + e.getCause());
                  e.printStackTrace();
            }
            return data
```

HR Controller:

```
package helloworld;

import java.net.URL;
import java.util.ResourceBundle;

import com.jfoenix.controls.JFXButton;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.layout.Pane;

public class HRController implements Initializable{

	@FXML
	private JFXButton EmployeeButton,AddNewButton,UpdateButton;

	@FXML
	private Pane HREmployee,HRAddNew,HRUpdate;


	@FXML
	private void EmployeeAction(ActionEvent event){
		if(event.getSource()==EmployeeButton){
			HREmployee.setVisible(true);
			HRAddNew.setVisible(false);
			HRUpdate.setVisible(false);
			EmployeeButton.requestFocus();
		}
	}

	@FXML
	private void AddNewAction(ActionEvent event){
		if(event.getSource()==AddNewButton){
			HREmployee.setVisible(false);
			HRAddNew.setVisible(true);
			HRUpdate.setVisible(false);
			AddNewButton.requestFocus();
		}
	}



	@FXML
	private void updateAction(ActionEvent event){
		if(event.getSource()==UpdateButton){
			HREmployee.setVisible(false);
			HRAddNew.setVisible(false);
			HRUpdate.setVisible(true);
			UpdateButton.requestFocus();
		}
	}
```

```
	@Override
	public void initialize(URL location, ResourceBundle resources) {
		// TODO Auto-generated method stub

	}

}
```

Products Code:

```
package helloworld;

public class Products {

	public Integer serialno;
	public String productname;
	public Double quantity;
	public Double cost;
	public Double price;
	public Double totalprice;

	public Products(Integer serial, String productname, Double quantity,
Double cost, Double price,
			Double totalprice) {
		super();
		this.serialno = serial;
		this.productname = productname;
		this.quantity = quantity;
		this.cost = cost;
		this.price = price;
		this.totalprice = totalprice;
	}

	public Integer getSerialno() {
		return serialno;
	}

	public String getProductname() {
		return productname;
	}

	public Double getQuantity() {
		return quantity;
	}

	public Double getCost() {
		return cost;
	}

	public Double getPrice() {
		return price;
	}
```

```
	public Double getTotalprice() {
		return totalprice;
	}



}
```

Products Class:

```
package helloworld;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

import org.controlsfx.control.Notifications;

import com.jfoenix.controls.JFXButton;
import com.jfoenix.controls.JFXComboBox;
import com.jfoenix.controls.JFXTextField;

import eu.hansolo.enzo.notification.Notification.Notifier;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Pos;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.Stage;
import javafx.util.Duration;

public class ProductsClass implements Initializable {


	private static final ActionEvent Mou = null;



	@FXML
	private Button AddSupply;
```

```
	@FXML
	private TableColumn<?, ?>
SerialNumberColumn,ProductColumn,QuantityColumn,CostColumn,PriceColumn,TotalP
riceColumn;

	@FXML
	private TableView<Products> ProductsTable;



	@FXML
	private TextField
SerialField,ProductField,QuantityField,PriceField,CostField,TotalPriceField;

	@FXML
	private JFXComboBox<?> PriceComboBox,CostComboBox;

	@FXML
	private JFXButton
SaveOnAddSupplyButton,ClearOnAddButton,DiscardOnAddButton;


	@FXML
	private void DiscardFunction(ActionEvent event) {

		Stage a = (Stage)
((Node)(event.getSource())).getScene().getWindow();
		a.close();

	}


	@FXML
	private void ClearOnAddSupply(ActionEvent event) {


		SerialField.clear();
		ProductField.clear();
	  QuantityField.clear();
	   PriceField.clear();
	  CostField.clear();
	  TotalPriceField.clear();
	PriceComboBox.getSelectionModel().clearSelection();
    //  CostComboBox.getSelectionModel().clearSelection();

	}

	@FXML
	private void AddProducts (ActionEvent event ) {
		boolean success=false;

		if
(SerialField.getText().isEmpty()||ProductField.getText().isEmpty() ||
```

```
                        QuantityField.getText().isEmpty() ||
PriceField.getText().isEmpty() ||

                        CostField.getText().isEmpty() ||
TotalPriceField.getText().isEmpty() ) {

                 Notifications notification=Notifications.create()
                         .title("Warning")
                         .text("All field are required!")
                        //.graphic(view)
                         .hideAfter(Duration.seconds(2))
                         .position(Pos.CENTER);

                        notification.darkStyle();
                         notification.showWarning();
            } else    {    success=Database.AddProductDb(


                        Integer.parseInt(SerialField.getText()),
                        ProductField.getText(),
                        Double.parseDouble(QuantityField.getText()),
                        Double.parseDouble(PriceField.getText()),
                        Double.parseDouble(CostField.getText()),
                        Double.parseDouble(TotalPriceField.getText())
                         //PriceComboBox.getSelectionModel().toString()

                    );



             Notifier.INSTANCE.notifySuccess("Success", "All detailes were
saved successfully!");
                //   Notifications notification=Notifications.create()
                 //  .title("Success")
                 //  .text("Inventory was updated successfully")
                  //.graphic(view)
                  // .hideAfter(Duration.seconds(5))
                  // .position(Pos.BOTTOM_RIGHT);

                 // notification.darkStyle();
                 //  notification.showConfirm();



                      SerialField.clear();
            ProductField.clear();
            QuantityField.clear();
             PriceField.clear();
            CostField.clear();
            TotalPriceField.clear();
            Stage a = (Stage)
((Node)(event.getSource())).getScene().getWindow();
                 a.close();
```

```
	}

	}

public void updateProductsTable(String search) throws NullPointerException{


	SerialNumberColumn.setCellValueFactory(new
PropertyValueFactory<>("serialno"));
		ProductColumn.setCellValueFactory(new
PropertyValueFactory<>("productname"));
		QuantityColumn.setCellValueFactory(new
PropertyValueFactory<>("quantity"));
		CostColumn.setCellValueFactory(new
PropertyValueFactory<>("cost"));
		PriceColumn.setCellValueFactory(new
PropertyValueFactory<>("price"));
		TotalPriceColumn.setCellValueFactory(new
PropertyValueFactory<>("totalprice"));


		SerialNumberColumn.setStyle( "-fx-alignment: CENTER;");
		ProductColumn.setStyle( "-fx-alignment: CENTER;");
		QuantityColumn.setStyle( "-fx-alignment: CENTER;");
		CostColumn.setStyle( "-fx-alignment: CENTER;");
		PriceColumn.setStyle( "-fx-alignment: CENTER;");
		TotalPriceColumn.setStyle( "-fx-alignment: CENTER;");

		ProductsTable.getItems().clear();
//(companyColumn,nameColumn,surnameColumn,phoneColumn,emailColumn,addressColu
mn);
		ProductsTable.setItems(Database.getProductsTable(search) );
	}

	@FXML
	private Button InventorySearchButton;

	@FXML
	private JFXTextField InventorySearchField;


@FXML
private void LoadDataToTable(ActionEvent event){
		try {
			if(event.getSource()==InventorySearchButton){
				String search = InventorySearchField.getText();
				System.out.println("Shqipe we made it");
				updateProductsTable(search);
			}
		}catch(Exception e){
			e.printStackTrace();
		}
}
```

```
@FXML
private Button ProductTableLoadButton;


@FXML
private void RefreshDataToTable(ActionEvent event){
            try {
                  if(event.getSource()==ProductTableLoadButton){
//                      String search = ClientSearchField.getText();
                        System.out.println("Shqipe we made it");
                        updateProductsTable(null);
                 }
            }catch(Exception e){
                  e.printStackTrace();
            }
}


@FXML
private void OnAddSupply(ActionEvent event) throws IOException{
      if(event.getSource()==AddSupply){
            Stage stage=new Stage();
             Parent
root=FXMLLoader.load(getClass().getResource("AddProduct.fxml"));
                  Scene scene=new Scene(root,1063,371);
                 stage.setResizable(false);
                  stage.setScene(scene);
            stage.show();


      }

}


      @Override
      public void initialize(URL location, ResourceBundle resources) {




      }









}
```

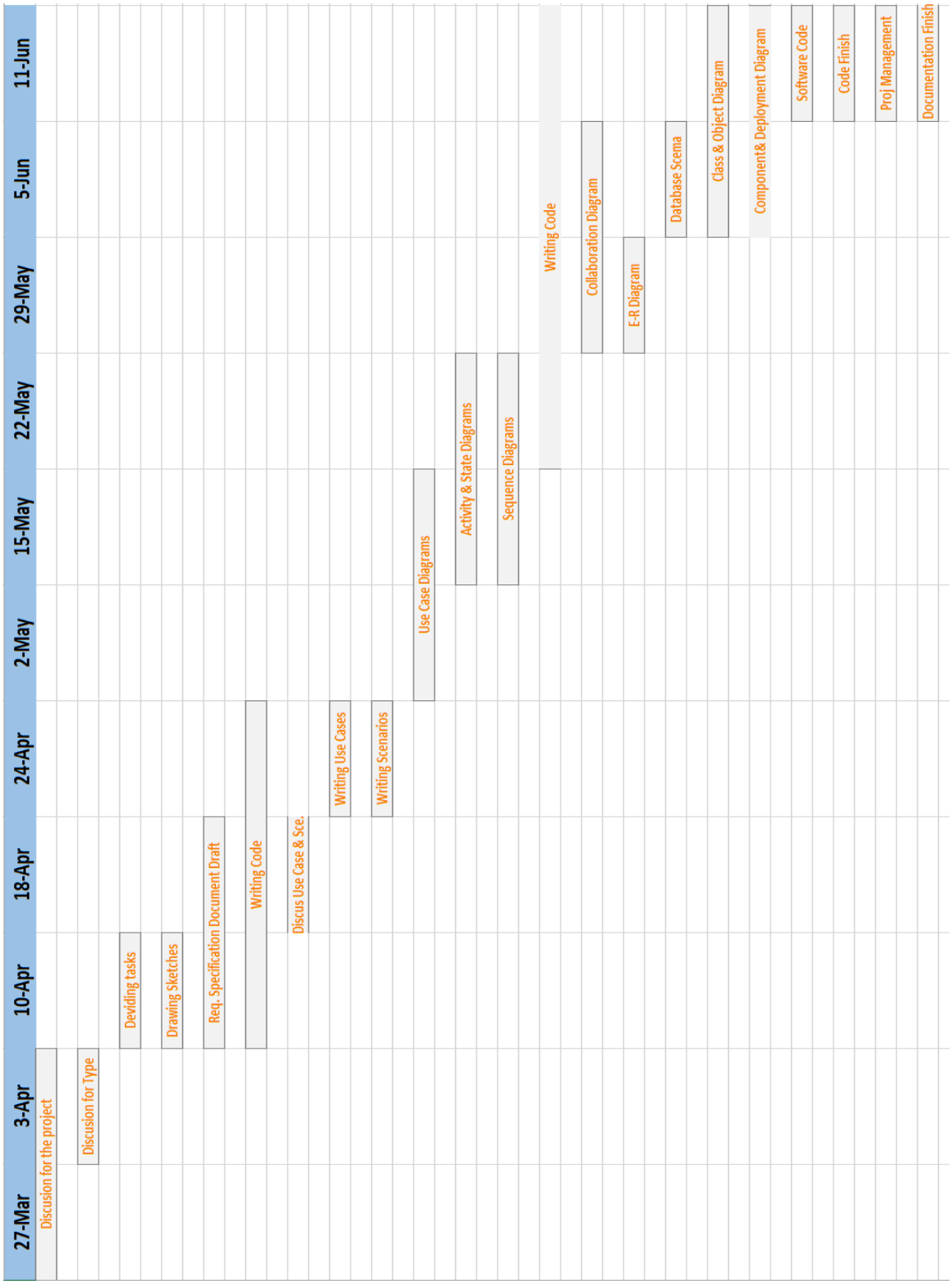
10.1 Installation Manual

The program is very easy to install and it can be used only into one computer, since it is **not** a web- application. The software comes with a cd which will be mounted into the PC.

To be able to use the “Beta Business” application in your personal computer you should:

a) Install XAMPP (or Wampserver, notice that if you do not install these kind of control panels the application won’t work).
b) Open the bb.zip file in the cd drive.
c) Search for the htdocs folder located into the bb.zip file.
d) Open XAMPP and press the start button next to Apache and MySQL
e) Install the executable file.
f) Run the program
g) Use it.

11. PROJECT MANAGEMENT



12. Requirements Confirmation/Stakeholder sign-off

Meeting Date	Attendees (name and role)	Comments
03/20/17	Albi Trashani (Group Leader) Igli Tola (Group Member) Arnold Valteri (Group Member) Ernaldo Gjomakaj (Group Member) Ornela Torra (Group Member)	Team gathered to decide what the application will be about.
03/27/17	Igli Hakrama (Group Advisor) Albi Trashani (Group Leader) Igli Tola (Group Member) Arnold Valteri (Group Member) Ernaldo Gjomakaj (Group Member) Ornela Torra (Group Member)	Team decided the project concept and did research in businesses to see what is needed for the application
03/28/17	Albi Trashani (Group Leader) Igli Tola (Group Member) Arnold Valteri (Group Member) Ernaldo Gjomakaj (Group Member) Ornela Torra (Group Member)	Tasks were assigned for each member of the group.
04/03/17	Albi Trashani (Group Leader) Igli Tola (Group Member) Arnold Valteri (Group Member) Ernaldo Gjomakaj (Group Member) Ornela Torra (Group Member)	Team discussed and decided the programming language for the code, which will be in JAVA.

04/05/17	Igli Hakrama (Group Advisor) Albi Trashani (Group Leader) Igli Tola (Group Member) Arnold Valteri (Group Member) Ernaldo Gjomakaj (Group Member) Ornela Torra (Group Member)	Team started the project, by using the templates that our Advisor gave to us.
04/09/17	Albi Trashani (Group Leader) Igli Tola (Group Member) Arnold Valteri (Group Member) Ernaldo Gjomakaj (Group Member) Ornela Torra (Group Member)	The coding of the application was started, sticking to the specifications that the group members started.
04/22/17	Albi Trashani (Group Leader) Igli Tola (Group Member) Arnold Valteri (Group Member) Ernaldo Gjomakaj (Group Member) Ornela Torra (Group Member)	Team members started writing the scenarios.
04/27/17	Igli Hakrama (Group Advisor) Albi Trashani (Group Leader) Igli Tola (Group Member) Arnold Valteri (Group Member) Ernaldo Gjomakaj (Group Member) Ornela Torra (Group Member)	Project review until now by the advisor, critics and improvements to the project.
05/02/17	Albi Trashani (Group Leader) Igli Tola (Group Member) Arnold Valteri (Group Member) Ernaldo Gjomakaj (Group Member) Ornela Torra (Group Member)	After completing the scenarios started off with the diagrams.

05/18/17	Albi Trashani (Group Leader) Igli Tola (Group Member) Arnold Valteri (Group Member) Ernaldo Gjomakaj (Group Member) Ornela Torra (Group Member)	Finishing Coding Setting Goals for next meeting.
05/29/17	Albi Trashani (Group Leader) Igli Tola (Group Member) Arnold Valteri (Group Member) Ernaldo Gjomakaj (Group Member) Ornela Torra (Group Member)	Finishing the last diagrams.
06/02/17	Albi Trashani (Group Leader) Igli Tola (Group Member) Arnold Valteri (Group Member) Ernaldo Gjomakaj (Group Member) Ornela Torra (Group Member)	Writing Software Testing and User Manual.
06/10/17	Albi Trashani (Group Leader) Igli Tola (Group Member) Arnold Valteri (Group Member) Ernaldo Gjomakaj (Group Member) Ornela Torra (Group Member)	Reviewing the project for any uncompleted task & improvements.