

Project Report - SEP2

Group 9:

Rokas Barasa – 285047

Raimonds Vacietis – 285048

Mohammed Marwan Summakieh – 285805

Hugh Ramseth McIntyre – 286129

Supervisors:

Ib Havn (IBA)

Troels Mortensen (TRMO)

Software Engineering IT-1-A19

Semester 2

Character Count: 23560

Version: 20 December 2019

Declaration

I hereby declare that my project group as listed above did prepare this Project Report and that all sources of information have been duly acknowledged.



Rokas Barasa



Raimonds Vacietis



Mohammed Marwan Summakieh



Hugh Ramseth McIntyre

ABSTRACT

The project task is to create a logistics system implementing a client-server solution with multiple users. The result is a skeleton system was analysis of the problem with little value to address the problem.

T

TABLE OF CONTENTS

1	Introduction	1
2	Requirements.....	2
2.1	Functional Requirements.....	3
2.2	Non-Functional Requirements.....	4
3	Analysis	5
3.1	User Access	6
4	Design.....	7
4.1	Architecture	9
4.2	User Access	9
4.3	Database Diagram.....	10
5	Implementation	13
6	Test.....	16
7	Results and Discussion	16
8	Project Future	17
9	Conclusions	18
10	Sources of Information	9
	Appendices.....	i

1 INTRODUCTION

A small drug company were offering their products in a small area and wanted to expand their business due to their success. However, their methods were simple as mentioned in the Background description in the project description.

The company's current system required a lot of administrative work, and it can not handle more workload. It had already caused some errors regarding dates of expiration, not to mention the poor communication methods between the different parts of the company regarding orders.

The goal is to make it simpler for the staff to track orders, keep the data on product storage movement and storage up-to-date and accurate, and view data relating to the performance of each area of the relevant company logistics as it was defined in the project description.

2 REQUIREMENTS

Below are the user stories to form the basis of which features are most important and grouped among each user. These stories provide generic actions for which functionality can be built and tested in the system and those which are required yet cannot be tested due to the nature of the request.

The following words will specifically apply the meanings written thereafter for the sake of clarifying potentially ambiguous words in these requirements.

SA	System Administrator
FM	Factory Manager
DM	Distribution Centre Manager
PM	Pharmacy Manager
User	All the above
Building	Generic term for factories, distribution centres, or pharmacies represented in the system
Manager	System user with access restrictions based on their assigned unique building and its building type
Goods	Any non-specific physical items that PharmaBank would produce and sell
Products	Identities and relevant information attached to the specific types of goods (i.e. ibuprofen as a product can be produced many times as individual goods)
Stock	Aggregate number of products stored in a distribution centre
View	The ability of a user to see the data. Does not include further interaction or any alteration of given data where not explicitly stated
Orders	Digital documents detailing the request for a movement of goods from one place to another
Delivery	Recorded movement of goods from one location to another. Goods in a delivery are no longer at either the start or end points of the transit until the order has been confirmed or cancelled
Confirm Order	The details of an order have been acknowledged by the recipient manager and the process to send the requested goods has begun in the system queue
Confirm Delivery	The goods contained in the delivery have been received at the intended destination and acknowledged by the local manager
Recall	An order for goods to be returned to closest distribution centres based on product and batch ID's. Different from orders as they can be sent to pharmacies, do not have a minimum or maximum quota, and are not concluded at any point.
Sister-centre	Separate distribution centre from the subject distribution centre where both are within the administration of PharmaBank
Report	A generated document or file containing circumstantially specified information. Details to be elaborated upon later in documentation

2.1 Functional Requirements

1. As a **SA**, I want to **create buildings**, so that I can add new locations to the system.
2. As a **SA**, I want to **create managers** for buildings, so I can grant limited access and interaction to a building's data in the system to a person or persons.
3. As a **SA**, I want to **edit buildings**, so I can alter the location information without having to remove the building and start again.
4. As a **SA**, I want to **edit managers**, so I can change their given information.
5. As a **SA**, I want to **remove buildings**, so I can prevent any further interaction with the location in the system.
6. As a **SA**, I want to **remove managers**, so I can revoke access and interaction to the relevant building's data.
7. As a **FM**, I want to be able to **create products** in the system, so I can specify the information of the goods factory manufactures and allow centres to order those products from my factory.
8. As a **DM**, I want to **view the stock** currently in my centre, so I know what is stored in my centre and can plan accordingly.
9. As a **DM**, I want to **view orders** sent to my centre, so I know what the sender needs from my centre.
10. As a **DM**, I want to **confirm an order** sent to me, so I can allocate the required products from my stock for a delivery.
11. As a **DM**, I want to be able to **create a delivery**, so the ordered goods can be moved from my centre to that of the requestee.
12. As a **DM**, I want to be able to **cancel a delivery**, so I can return the products to my stock and end the sending process.
13. As a **DM**, I want to **view the storage space** my centre has for dry and cold goods, so my stock does not exceed the amount that my building can handle.
14. As a **DM**, I want to **create orders to factories**, so I can request fresh products to be manufactured for me.
15. As a **DM**, I want to **confirm a delivery** of my order to me, so I can close the order.
16. As a **DM**, I want to view my **sister-centres' stock**, so I can see if I can reduce my deficit of a product from another centre's surplus of the same product.
17. As a **FM**, I want to **view orders** received to plan production for that order.
18. As a **FM**, I want to **confirm orders**, so the DC knows I have begun production and the expected date of completion.
19. As a **DM**, I want to **create orders to sister-centres**, so I can request existing goods from other centres to improve efficiency of delivery to pharmacies and prevent excessive manufacturing of a product in PharmaBank.
20. As a **User**, I want to **log in** so I can use the system.
21. As a **User**, I want to change my **password**, so I can control access to my account.

- 22. As a **PM**, I want to **create orders**, so I can resupply my pharmacy.
- 23. As a **PM**, I want to **view my orders**, so I can see the history of the orders I have sent.
- 24. As a **PM**, I want to view the **status of my orders**, so I can track their progress and plan accordingly.
- 25. As a **PM**, I want to **confirm a delivery** of my order to me, so I can close the order.
- 26. As a **PM**, I want to view a list of my **order history**, so I have a log of orders sent.
- 27. As a **PM**, I want to view which products are **in stock** at the distribution centre, so I can see what I can order from the centre's current stock.
- 28. As a **PM**, I want to see when an **order is estimated** to arrive before I make an order, so I can tell my customer how long to expect to wait.
- 29. As a **FM**, I want to **discontinue products** so they cannot be ordered anymore from my factory.
- 30. As a **PM**, I want to **cancel orders**, so I can retract my request in coordination with our distribution centre.
- 31. As a **SA**, I want to view all the **data** that each manager sees, so I can investigate issues.
- 32. As a **DM**, I want to get a **report** which gives a summary of all the **centres' stock**, so I can use the data for analysis regarding what goods PharmaBank has in its centres.
- 33. As a **DM**, I want to get a **report** which gives me a summary of all the **orders**, so I can use the data for analysis regarding the supply, demand, and movement of goods within and from PharmaBank.
- 34. As a **DM**, I want to **create a recall** of a product, so I can notify all locations the product was sent to that, where possible, the products must be returned immediately.
- 35. As a **PM**, I want to **view recall notifications**, so I can contact my affected customers and return as many of the products as reasonably possible.
- 36. As a **PM**, I want to create a **recall-delivery**, so my distribution centre knows the details of a recalled product being returned.
- 37. As a **DM**, I want to **discontinue a recall** of a product, so I can remove the relevant notification.

2.2 Non-Functional Requirements

- 38. The system must use a connected client and server.
- 39. The system must allow for use by different users.
- 40. The client must be built using Java and JavaFx.
- 41. The server must implement PostgreSQL.

3 ANALYSIS

All users access the system using a login name and password. No functionality for any user is available if the login is not successfully entered and accepted, apart from the login itself.

The SA adds Managers to the system, which have different access and fulfil different roles in the system. Those Managers are assigned to Buildings which in turn defines what type of Manager they are from the three seen in Figure 1. The SA can then also view the current data of the products, orders, deliveries, recalls, and stock of each of the Buildings.

The FM manages the products' information that is created at their factory. They can view and update the orders sent to them but cannot create new orders or edit the information within. The FM can send deliveries and manage their own but cannot receive deliveries regarding the goods in this system.

The PM creates and sends orders and can manage the information within. It does not update the

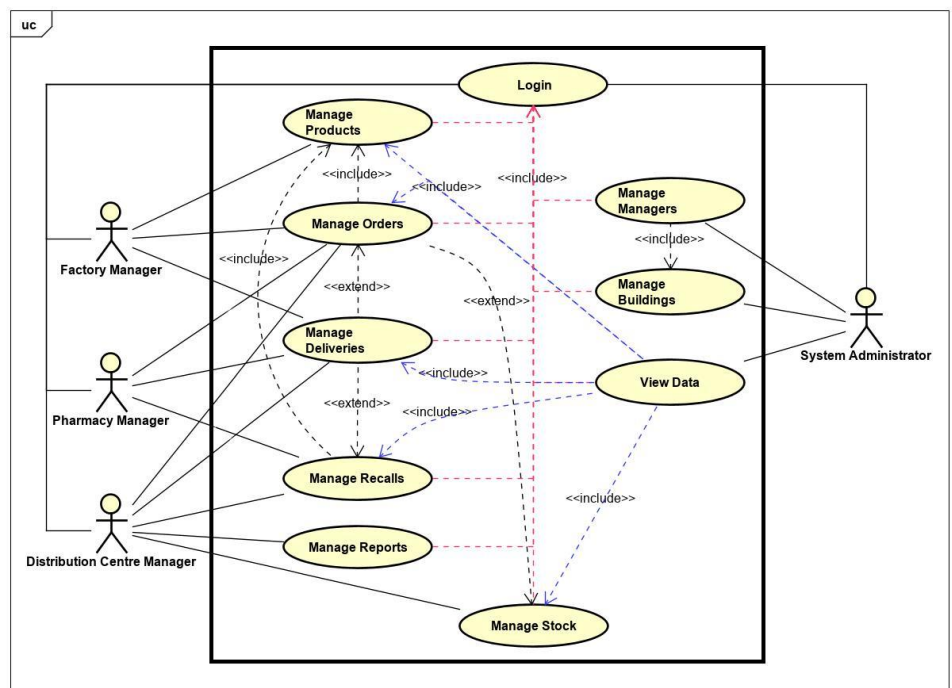


Figure 1 Use Case Diagram (see Appendix B.i)

status except when it confirms receipt of a delivery. A PM cannot send or create deliveries. If there is an error in an order it is cancelled the location that sent it and the aggregate goods are returns to the stock count at that same source. The PM can also view Recall information where it becomes relevant to the location. Finally, a PM can see the stock available at its assigned distribution centre.

The DM is the most important user on the system and as a result has the most abilities when compared to the other Manager types. A DM can send and receive both orders and deliveries. It manages the stock it has in its own building and can see the stock held in its sister-centres, so it can maintain what it deems to be proper levels of its inventory. It can also see the capacity of all distribution centres, most importantly its own, for prevention of overflow. It creates recall notices which are sent to all the final destinations of the identified products that should be returned. Lastly it is the DM that can generate a data report for use by PharmaBank's analysts.

3.1 Buildings

The original concept was to add Buildings to act as both users and fulfil the function of an object. However, such an approach means that there could only be a single operator in a Building, which hinders future flexibility regarding adding or removing users later. The result is quite simple in that a Manager is assigned to a Building which can still exist and keep its data, even when more users are added, or none exist at all. The Building type defines what the Manager can do and see and forms the central point of access for its information. Alternatively, a new Manager could then run the operations of the Building while the other Manager's access is revoked.

It should be noted there is no explicit point that a building should only have one Manager attached to it at any time. To that end there would likely be no set limit on the number of Manager assigned to a single Building. On the other hand, Managers would likely only be assigned to a single Building for the time being to mitigate confusion between individual Buildings of the same type.

Some buildings have additional attributes. Distribution centres have a maximum capacity for both cold and dry storage. This storage capacity status is reflected in the amount of stock present at the location or returning from a cancelled delivery.

3.2 Multiple Users

The System Administrator provided the solution to a logic issue for launching the system upon delivery to the customer: if different users must be added and removed, then there must be a user that will be able to perform these actions yet unable to remove itself or be added. It would therefore be the only user included in the initial implementation. As it is not required by implication or explicit request, the SA were not be able to add other SA's.

The Managers would be the primary actors in using the system, however there is no relation between the user type's name and the position of the person accessing the system in PharmaBank or one of its customer locations. Further, a Manager must be assigned to a Building when added to the system as they otherwise would serve no purpose. The intention here is

3.3 User Access

Different individuals require access to their own data and likely not data that is irrelevant to their role in PharmaBank. In order achieve this while delivering a client-server system as requested; an access point would need to guide the user to their relevant interface. A measure of security would also be necessary; given the sensitivity of company information that may be stored on the system, as well as the fact PharmaBank customers will have access to the system. So long as it fulfils this purpose: such a feature would not likely need to be complex.

As it would likely be the first action taken by a user, it would be reasonable to place the access function at the top of the priority of requirements. However, taken at its individual value as a function; the need for controlled access would only become strictly necessary immediately before access would be granted to the pharmacy manager who does not operate within the PharmaBank company structure. The requirements before it provide greater value than the ability to differentiate users as they would be a part of the PharmaBank staff.

3.4 Products

The creation of a product will provide a single point for any relevant meta-data concerning a product's expected lifetime and any other part that may be found to be automated.

3.5 Orders

Orders and deliveries form a defined difference between two things that could be easily confused with one another. In this project: an order is simply the request for goods, whereas a delivery is the fulfilment of that request.

The order contains information on the Manager and Building that sent it as well as the goods requested. While creating the order, the Manager will be able to see how much stock is available to request of the good they are ordering from the specified supplier. For simplicity's sake, only one product type can be ordered at a time.

3.6 Deliveries

As stated above, these represent the physical movement of goods from one location to another. When it is created/sent, the quantity of the goods is removed from the origin's stock. They are only added back to the system if the delivery is cancelled. If it is accepted by a distribution centre: the quantity is added to the destination's stock. If accepted by a pharmacy: the quantity will no longer be counted among the goods in the possession of PharmaBank as those pharmacies will not be sharing their storage information or stock levels.

3.7 Error Investigation

The inclusion of this feature has proven to be the lynch pin to much of the complications in conceptualising the problem. It can be identified in Figure 1 by the blue lines leading to each of the core elements of the system. The benefit of such a feature would be the readily accessible ability for PharmaBank to investigate their own mistakes or confusion in what can be seen by any Manager without being able to instigate any changes to what is there and thereby increase any potential damage done.

The downside is that to create such a feature would mean that it would need to be updated and tested each time another feature was changed that affected any of the SA's investigation views. A solution may become apparent during the design or implementation phase. For now, it will remain a lower priority and deemed to be a potentially costly risk to productivity.

3.8 Recalls and Reports

These two features are deemed to be the least significant features compared to the others given the expected complexity of their requirements and comparative infrequency of daily or even monthly use.

Recalls are to act as an open-ended order to be automatically sent to any destination that received the specified goods. The good would be identified by a unique assigned code or possibly its product name alongside its source factory and production or expiry date. Once sent to a pharmacy, there will be no way to determine how much of the goods will be returned for numerous potential reasons. Due to this, the order would remain open until closed by its source distribution centre and likely without a minimum quota to be achieved.

Reports will generate non-specific data required by PharmaBank's data analysts. The detail provided is expected to evolve over time and new report algorithms may also be conceived. Until the changing nature of the requests can be address for a more maintenance free solution: it will remain to be one of the final features to be made.

4 DESIGN

The system will be developed with Java 11 and JavaFX for user interface.

4.1 Architecture

The chosen pattern for the code structure is Model View ViewModel (MVVM) based architecture. In which the system is broken down into parts. The parts are:

- Model – Manipulates data and validates it.
The logic part of the system
- View – Takes user inputs and presents GUI to user.
- ViewModel – Handles user input: validates, formats and changes GUI state.

Following MVVM architecture we decided to break the system further. We will make layers of inner system functionality making it easy to navigate and added interfaces classes for easy coupling and

decoupling for changing parts of the system without affecting other parts. The layers are:

- GUI – Handles user input. Represents View and ViewModel part of MVVM.
- Client logic – Represents Model part of MVVM.
- Networking – Holds clients and server connection.
- Business logic - Represents logic part of server, like client logic.
- Persistence – Holds server connection to database and adapters for functions related to database.

The combination of MVVM, interfaces, and the layer approach will allow development of the system to be made in small pieces which are easily changeable and don't depend on each other and letting implementation of multiple parts of the system be done at the same time.

4.2 User Access

The system will use a login to differentiate what can be accessed between users. A user will enter details which get sent to server to be compared to the user information in the database. If the details match a user, then the user class

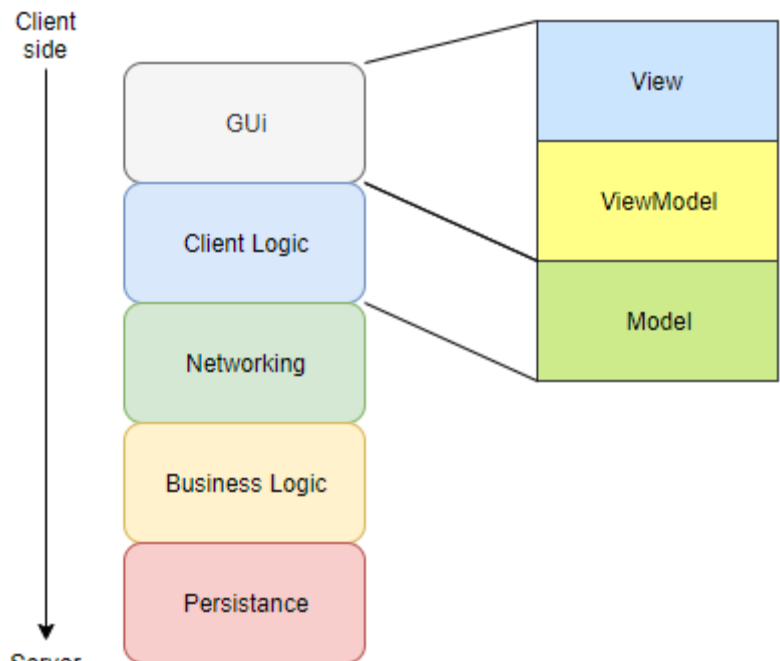


Figure 2

is sent back as a key which lets the user perform certain program functionality depending on the access level and building which the user is set to. See sequence diagram in appendix section.

4.3 Admin Menu

When the admin logs in a menu will open with default tab of buildings. He will have a table that shows the current buildings in the system and their information. The admin will be able to add a new building by pressing an add button in the building tab after which a new window will pop up asking to enter building details specifying the building name, address, and building number. Depending on the type of building the user will select a primary letter to the building number, for example F- for factory, D – for distribution centre and p – for pharmacy. After the admin presses save the window would close and the information about the building would get sent to the server to be stored in the database and the building would be added to the list of all building the admin sees.

The admin will also have a button to delete a building, which will work only if a building from a table is selected. Pressing the button will make a small window pop up asking the admin if he is sure about this action so that he doesn't delete an important object in the system accidentally. If admin presses cancel the window will close and nothing will change, otherwise if he presses yes, the system will send a message to the database to delete selected building and all things that depend on it and then the system will update the admins view of all buildings without the deleted building.

4.4 Database

The systems communication to the database will be done through data access objects (DAO) which will act as adapters that based on what the server requests from them will turn that request into sql code that will be injected into the database to receive its output. The database will store all information that the system will use. When a user is created in the system it will be assigned to a building, the building can have multiple users or none at all but the user will only have one building or none if the user is a admin. The building will be the most important entity in the database if it is deleted all entities related to it are also deleted. The building entity will be inherited by the Distribution centre(DC), Pharmacy and Factory which do not hold any extra value but are actors to which the functionality is attached too. DC will have 2 Storage entities assigned that will have different types of storage: dry or cold. Storage will contain product that the distribution centre stores. The DC will also have orders and recalls assigned to it using the buildingNo that it inherits. The Pharmacy will use recalls and orders in the same way as DC. When factory will create products it will assign its buildingNo to the product making a connection between them, in addition to that factory will also have orders assigned to it so it can receive orders. All of this is shown in the ER diagram below.

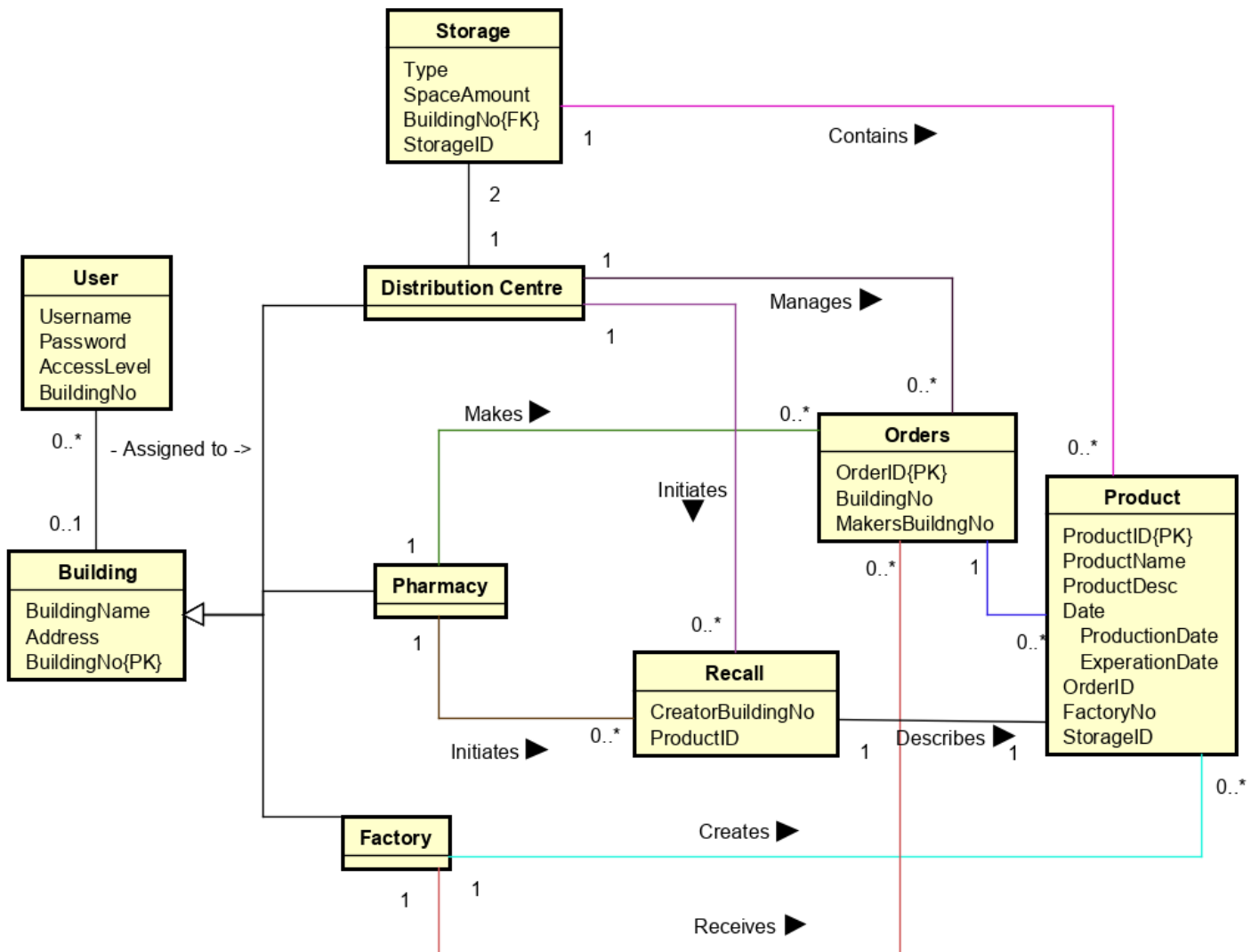
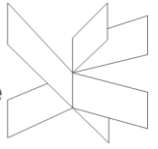



Figure 3

4.5 Graphical User Interface

All the UI elements will be made in SceneBuilder due to ease of use. When starting the system every user will see the login screen first and then depending on which user logs in will get replacement window that shows the menu of chosen user. All major functionality in the user menus will be divide into tabs. The UI designs are shown below



Window Name: Login



PIIARMA BANK A/S

GROUP 9 IT-A19-SEP2

Username:

Password:

Login

Figure 4

Buildings Managers

Building	Address	Building number

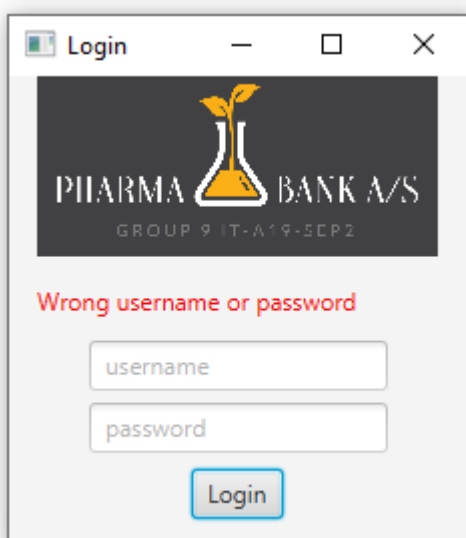
Create Building

Figure 5

Figure 6

5 IMPLEMENTATION

5.1 Login:



The login in the project is a simple transfer of credentials through the MVVM architecture all the way to the database to get the correct data access for the user. The classes involved in this transition are LoginController, LoginVm, LoginModelImp, AdminClientImp, LoginClientImp, ClientSocketHandler, ServerSocketHandler, ModelImp, UserDaoImp and finally DataBaseConnection. When attempting to login the entire flow starts from the LoginController class. It is where the credentials get recorded from the interface and passed on to the LoginVM which is the ViewModel of the MVVM in that class checks if the boxes are empty or not. It can be seen in the code below.

```
public void login() {
    if(username.get() != null && !username.get().isEmpty() && password.get() != null && !password.get().isEmpty())
    {
        loginReponse.setValue(null);
        model.login(username.get(), password.get());
    }
    else{
        loginReponse.setValue("Must enter both, username and password");
        username.setValue(null);
        password.setValue(null);
    }
}
```


then passes through the classes mentioned above until it reaches the ServerSocketHandler it is where the class establishes connection between the client and the server using threads and sockets.

After the client-server connection is established UserDaoImp class comes in play as it is the gate that will check if the credentials are correct and they exist as a registered user in the database. It can be seen I code below.

```
User userFromServer = null;
System.out.println("before database");
try {
    database.setStatement(database.getConnection().createStatement());
    ResultSet rs = database.getStatement().executeQuery( "SELECT *FROM \"PharmaSystem\".SystemUser " +
        "WHERE username = '" + username + "' AND password = '" + password + "';" );
    while ( rs.next() ) {
        String usernameDB = rs.getString("username");
        String passwordDB = rs.getString("password");
        String accessLevelDB = rs.getString("accessLevel");
        String buildingNoDB = rs.getString("buildingno");
        String userNoDB = rs.getString("userno");

        System.out.println(usernameDB);
        System.out.println(passwordDB);
        System.out.println(accessLevelDB);
        System.out.println(buildingNoDB);
        System.out.println(userNoDB);
        userFromServer = new User(usernameDB, passwordDB, accessLevelDB, buildingNoDB, userNoDB);
        rs.close();
        database.getStatement().close();
        database.getConnection().close();
        return userFromServer;
    }
} catch ( Exception e ) {
    System.err.println( e.getClass().getName()+" : "+ e.getMessage() );
    System.exit(0);
}
```

If user with such credentials in username and password is found in database a class is created and sent back. If it doesn't find the user with the same credentials It returns null. The ServerSocketHandler gets the User class and sends it back to ClientSocketHandler.

Below is what the previous code section calls in the database. The code sends out a SQL query requesting a user with the username and password equal to what it received from the client. If found the users details will be returned to the system.

```
set search_path = 'PharmaSystem';
CREATE TABLE SystemUser
(
    username varchar(25) NOT NULL,
    password varchar(25) UNIQUE NOT NULL,
    accessLevel varchar(10),
    buildingNo varchar(4),
    userNo varchar(10)
);
Insert into SystemUser(username, password, accessLevel, buildingNo, userNo)
```

```
VALUES ('admin', 'admin', 'admin', 'null', 'U001');

create domain d_buildingNo as char(5);/*Could be something else, but right now don't know what it initials*/
create domain d_buildingName as varchar(20);
create domain d_address as varchar(40);

CREATE TABLE Building
(
    BuildingNo d_buildingNo not null PRIMARY KEY,
    Address d_address not null,
    BuildingName d_buildingName not null
);
CREATE TABLE Factory
() inherits (Building);

CREATE TABLE DistributionCentre
() inherits (Building);

CREATE TABLE Pharmacy
() inherits (Building);
```

When the ClientSocketHandler receives the User, it has to check if the user is null or not. If it is null, the user will send a response back to the login client and say that the details entered are wrong. If it receives a user class, it gets the buildingNo from it and divides it into sections to get the first section of the divided number which is a letter determining the building type. Before sending the user to the separate building menus it checks the access level of the user, if it is "admin" the user is sent to admin menu. If user access level is not admin it is sent back to a building with the matching letter.

```
try {
    while (true){
        Object fromServer = inFromServer.readObject();
        // these 2 check if user is null or actual user object and assigns it to a client like
        admin factory...
        if(fromServer instanceof User && fromServer != null){
            User fromServerUser = (User) fromServer;
            String[] facility = fromServerUser.getBuildingNo().split("(?!^)");
            System.out.println();
            if(fromServerUser.getAccessLevel().equals("admin")){
                client.getAdmin().getAccess(fromServerUser);
                System.out.println("got admin");
                login = true;
            }else if(facility[0].equals("F")){
                client.getAdmin().getAccess(fromServerUser);
                login = true;
            }
            else if(facility[0].equals("D")){
                //client.getFactory().getAccess(fromServerUser);
                login = true;
            }
            else if(facility[0].equals("P")){
                //client.getPharmacy().getAccess(fromServerUser);
                login = true;
            }
        }else if(!login && fromServer == null) {
```

```

        System.out.println("Wrong username or password");
        client.getLogin().loginResponse("Wrong username or password");
    }
}
} catch (IOException | ClassNotFoundException e){
    e.printStackTrace();
}
}

```

This is when the appropriate user menu is open. Adding and removing buildings was not implemented therefore we do not have a menu to open

6 TEST

Every new requirement was tested upon completion of implementation to see if it was successful. The following table shows implemented code and its working status according to what is required. Requirements that are not listed have not been implemented.

Requirement	Status	Notes
As a User , I want to log in so I can use the system.	Working	
As a SA , I want to create buildings , so that I can add new locations to the system.	Not working	
As a SA , I want to remove buildings , so I can prevent any further interaction with the location in the system.	Not working	

7 RESULTS AND DISCUSSION.

The skeleton of the system is made, and the system lets the user login if the user is an admin or any other user if he is stored in the database. No other functionality was fully implemented.

The outcome of the system would be deemed a failure. Only one of 37 requirements that were needed have been implemented. The foundation piece of the system, the login, works as intended and does not produce any problems but it does not have any independent value for the customer. The login was a priority in deployment because all the users need to login before they do any work.

8 PROJECT FUTURE

The future of the project depends very much on what steps would be taken next. The decision of whether to continue development or cease further work will likely reside with the customer and will depend on the terms of completion, the respective intellectual property rights of the customer and the production team, and the residual relationship of the two from this project's end. This section will assume sufficiently harmonious continuation of development by the team for the customer.

The system was intended to carry the bulk of the work required by a much larger company than PharmaBank A/S was on the date this report was published. Further development is still quite possible while the company scales upward. Further, the nuances of what is necessary and possible have been prepared for by an already familiar team to a still present problem facing the customer.

What has been implemented does provide a skeleton of a usable system that could be utilised as a first step for developing many different systems. However, these possibilities will be limited to the original problem provided by the customer:

- a) **Solution completion** – the requirements could be completed now that the core infrastructure is in place. Likely what is required may be simplified further to track better progress. The features could then also be shared with the customer in separate iterations to distribute the work required in converting the data from PharmaBank's current documents and system data.
- b) **Multiple buildings at a location** – sites belonging to the customer could contain different buildings to allow multiple factory production lines or both production and storage facilities at a single geographical location.
- c) **Multiple users** – much of the day to day work required of Managers could be delegated to administrative staff. This would allow for continuous smooth operations and limit functionality that carry greater risks of mistakes or misuse.
- d) **Add/Remove user types or building types** – a system administrator could directly control what a user can access based on a grid with a cross section of all user functions and views, or alternatively create new user types that could perform a bespoke list of functions that others could not.
- e) **Manager control of administrative users** – grant Managers the ability to create basic users for their assigned building(s) to reduce the workload on a system administrator. It would not be advised that they be given the ability to create new Managers or assign anyone to buildings outside of that Manager's control.
- f) **Elaborated Details** – information which were planned for use in the reports could be expanded or included for use by the users in the system itself. For example: the volume, weight, and stability of products could be added as additional attributes. This could affect how the goods are transported or stored to better calculate the space taken while in transit or storage, reduce risk of damage by what is placed upon them or how much

of the goods could be stored on shelves with a maximum weight, and how the weight may affect transit times or fuel costs.

- g) **User auditing** – provide a log that records the actions taken by every user and the time they took place. The benefit of this would be to track the source of mistakes or misuse of the system for investigations. One possibility could be a system bug in making an order which seems to be caused by human error. A log would show the individual had completed the action correctly and the real source of the problem could be better identified. The reverse could be human error excused by a bug that cannot be found in the system.
- h) **Vanilla System** – While it has been designed with a pharmaceutical company in mind, it is not limited to that niche market. Cold and dry storage are used by meat packers and florists, among many other professions. The problem here regards intellectual property rights when and if the system were to be presented and/or sold to other potential customers. Nonetheless, remedies or accommodations could be explored with the customer to achieve mutual satisfaction in the matter. From there the core completed system could be sold as a core product and altered to more closely suit the needs of the buyer as an optional service. One possible incentive would be their core system could be continuously maintained and updated at a lower cost; the work would remain the same regardless, but the cost would be effectively shared by each customer and provide the production team the capital to give the software more time and attention.

9 CONCLUSIONS

The goal was to make it simpler for staff to track orders, keep the data on product storage movement and storage up-to-date and accurate, and view data relating to the performance of each area of the relevant company logistics. The analysis was done and the requirements for the system were created along with the use cases and domain model of the system. These helped to figure out the problems the client was having and not get lost in the depth of the system.

The design stage and the implementation stage were done closely together to maintain consistency between them. The functionality that is implemented is tested and working but there is a lot of requirements missing from the system. The system currently has one of thirty-seven requirements completed but the base is built making way forward easier if the customer decides to continue the project development. With that said the main purpose of the project was not achieved.

APPENDICES

- A. Project Description
- B. Diagrams
 - i. Use Case Diagram
 - ii. Domain model
 - iii. ER diagram
 - iv. Class diagram
 - v. Use Case Description Add or remove building
 - vi. Login Use Case description
 - vii. Login sequence Diagram
 - viii. Login activity diagram (white box)
 - ix. Add building sequence diagram (Black box)
 - x. Delete building sequence diagram (Black box)
- C. Java Doc
- D. Source code