

Table of Contents

1. Introduction. 4

1.1 Purpose. 4

1.2 Scope. 4

1.3 Definitions, Acronyms, and Abbreviations. 4

1.4 References. 4

2. Data store. 5

*2.1* *Data* definition. 5

2.2 Size calculation. 5

2.3 Entity-Relationship Diagram.5

3. Use Cases. 6

3.1 Actors. 6

3.1.1 Actors diagram.. 6

3.1.2 Actor descriptions. 6

3.2 Use Case Descriptions. 6

3.2.1 <*Name of Use Case 1*>. 6

4. Domain Model Class Diagram.. 8

5. Activities Plan. 9

5.1 Development Phases. 9

5.2 Activity Plan. 9

5.2.1 Milestones. 9

5.2.2 Deliverables. 9

5.2.3 Time plan. 9

6. Appendices. 10

Revision Chart

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Primary Author(s) | Description of Version | Date Completed |
| Draft | All members | Initial draft created for distribution and review comments | October 09, 2019 |
| Preliminary | All members | Second draft incorporating initial review comments, distributed for final review | October 21 , 2019 |

**1.** **Introduction**

## **1.1** **Purpose**

The purpose is the same with the report of the requirements.

## **1.2** **Scope**

The scope is the same with the report of the requirements.

## **1.3** **Definitions, Acronyms, and Abbreviations**

This section is the same with the report of the requirements.

## **1.4** **References**

No References

**2.** **Data store**

## ***2.1*** ***Data* definition**

ELEMENT DATA TYPE

Employer

Name: Varchar(20)

Surname: Varchar(20)

ID: Smallint

Date of Birth: smallDatetime

Email: Varchar(30)

Position: Varchar(20)

SSN: Smallint

Personal phone: Smallint

Residents phone: Smallint

Address: Varchar(50)

Salary: smallmoney

IsAdmin: bit

(Employer has 1 authentication)

Authentication

Username: Varchar(30)

Password: Varchar(30)

Client

Name: Varchar(20)

Surname: Varchar(20)

ID: Smallint

Date of Birth: smallDatetime

Email: Varchar(30)

Personal phone: Smallint

Residents phone: Smallint

Address: Varchar(50)

(Client has N Pallets)

Transactions

Date: smallDate

Price: smallmoney

(Transaction has one Client)

(Transaction has N Pallets)

Warehouse

ID: Smallint

Location: Varchar(50)

Capacity: Smallint

Warehouse has (0 to N pallets)

2 + 52 + 2

Pallet

Position: Smallint

ID: Smallint

Import Date: SmallDate

Export Date: SmallDate

IsFood: Bit

Expiration Date: SmallDate

Cost\_of\_Service: Smallmoney

Additional \_cost: Smallmoney

(Pallet has one Client)

LogFile

Text: Text

Date: SmallDate

Created\_by: Varchar(50)

50 + 300004

## **2.2** **Size calculation**

First let’s calculate each element separated.

Employer = 167 bytes

Authentication = 64 bytes

Client = 138 bytes

Transactions = 8 bytes

Warehouse = 56 bytes

Pallet = 25 bytes

LogFile = 300054 bytes

Every Employer has authentication so the data for every employer is 231 bytes

The company has around 15 employers.

The warehouse maximum capacity is 792 pallets. So for warehouse we have 56 + 792 \* 25 = 19856 bytes

In the worst case we can have different clients for each pallet so we can have up to 792 clients at a time. So 792 \* (138 + 10 \* 8(Transaction)) = 172656 bytes

Size = NoEmployers \* EmployerData + WarehouseData + AllClientsData + Logfile

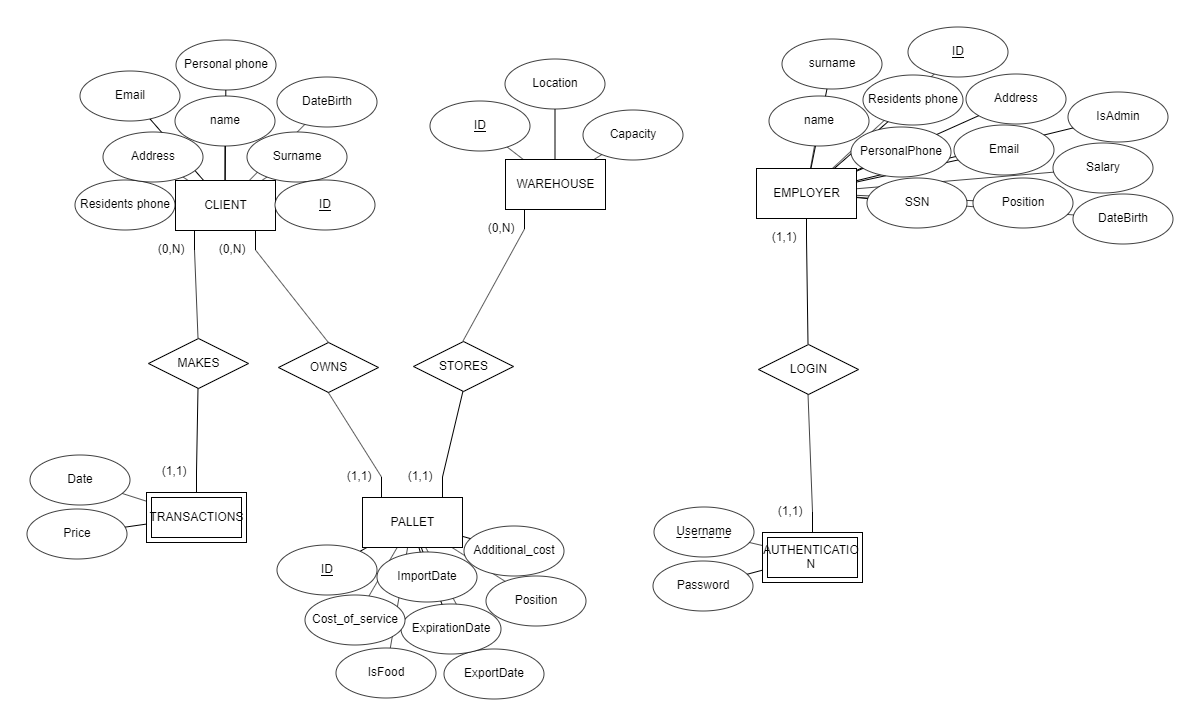
= 15 \* 231 + 19856 + 172656 + 300054

= 496031 bytes

= 496.031 Kbyte

So our Database size is **500 Kbyte**.

**2.3** **Entity-Relationship Diagram**

****

# 

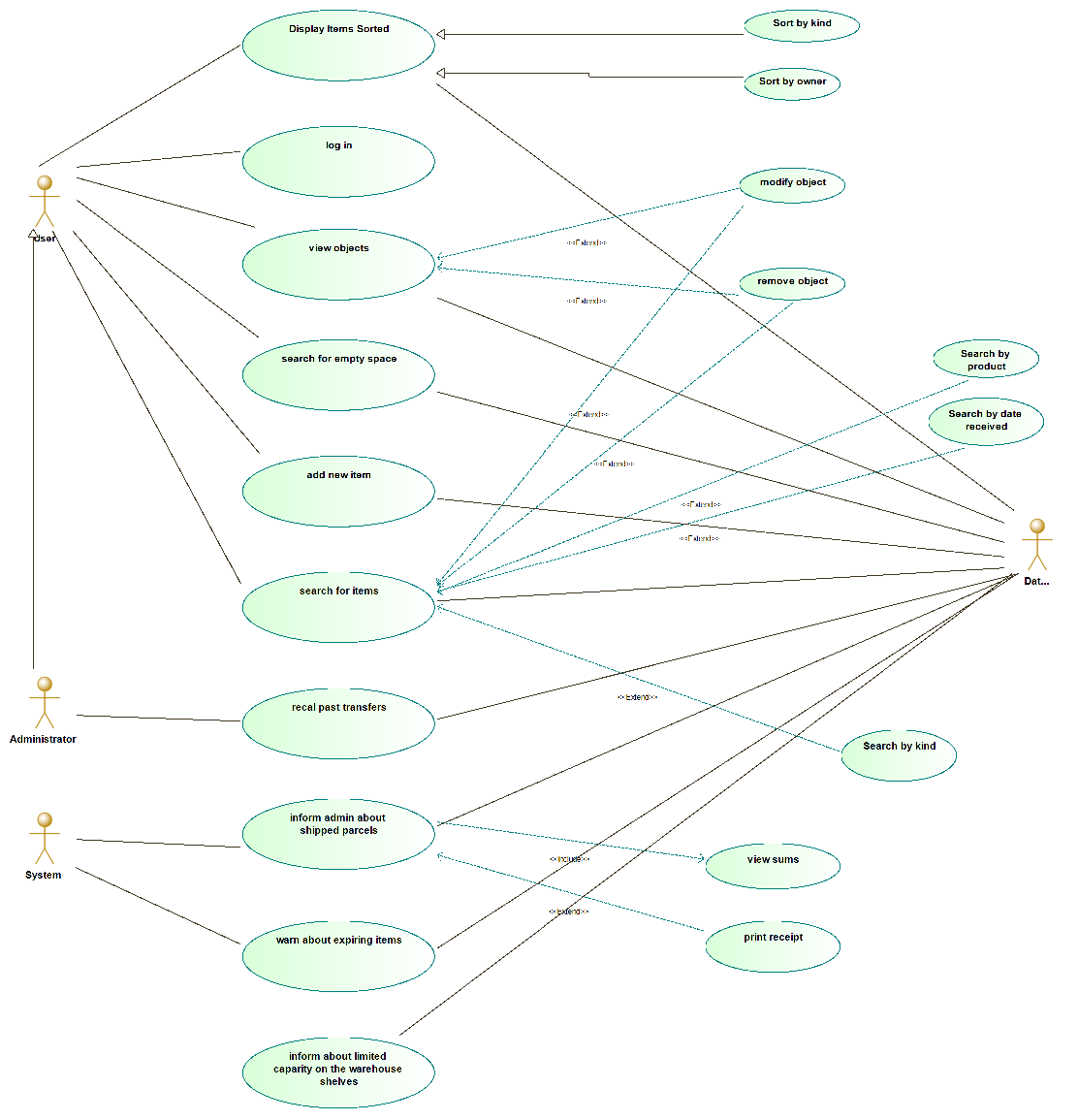
# 

# 

# 

# 

# **3.** **Use Cases**



## 

## 

## **3.1** **Actors**

*Actor number 1 : User → Main employee of the company . Will use the DB using queries and various other functions provided by the software itself .*

*Actor number 2 : Administrator → Can do exactly the same as the user and also has a couple of extra permissions .*

*Actor number 3 : System automatizations → We can consider the system as an actor , as it does procedures that would normally require a normal actor to be completed , however the client has requested an automated form of what the user for example could do / has to do .*

### 

### **3.1.1** **Actors diagram**

*In the company’s environment , we have incoming and outgoing products to / from the warehouse . Arriving from different sources , exported to different destinations , the main point is to store and then forward the shipment to its destination .*

*The main user of the software is the actor number 1 -> User . The user is responsible for most of the operations occurring in the warehouse and the database . The majority of operations is done by the user that is responsible for maintaining the correct state of the warehouse . Whenever a product is brought in the warehouse , the database is informed . The user can see the state of the warehouse and or modify it . Browsing it in any way he wants and viewing information of its contents . The administrator , as requested by the client , has all of the user’s permissions as well as some extra allowances as to what he is able to see . The system itself also has a few tasks to do . These tasks were asked to be done automatically .*

### 

### 

### **3.1.2** **Actor descriptions**

|  |  |
| --- | --- |
| **Description**  **USER** | *The main user of the software . He can view and edit the contents of the database . His job is to enter a new record when a product arrives , remove one when it’s exported or edit one if any change has to be done ( new export date for example ) . He can also see the state of the warehouse and modify it ,(eg move a product in order to create extra space )* |
| **Aliases** | *User*  *Employee* |
| **Inherits** | *None* |
| **Actor Type** | *Actor is an active and the most important in the usage of the software . He is a person and can number from 1..N* |

|  |  |
| --- | --- |
| **Description**  **ADMINISTRATOR** | *The administrator is basically the supervisor of the User . He can do exactly what a normal user and has some extra permissions . Therefore he can be considered as an extended form of the user with the extra benefit of being able to recall past transfers and print invoices/receipts .* |
| **Aliases** | *None* |
| **Inherits** | *Ancestor : User because he has the exact same attributes* |
| **Actor Type** | *Active . Person .* |

|  |  |
| --- | --- |
| **Description**  **System** | *By system , we refer to 3 main functions that are performed periodically and automatically by the system . It performs a lookup on the database and decides whether it needs to take an action or not ( mainly send inform messages ) .* |
| **Aliases** | *None* |
| **Inherits** | *None* |
| **Actor Type** | *Active/External System.* |

## **3.2** **Use Case Descriptions**

To help ease and to prevent unnecessary writing , we consider that any use case that involves the user also involves the admin because the admin can do the exact same thing .

### **3.2.1** **<*Logging in into the system* >**

**Description:**

The user and the administrator need to enter their credentials for the system to acknowledge them and allow them access to the database .

**Actors :**

User

### **3.2.2** **<*View object* >**

**Description :**

The user can see every object in the database . He has a clear view of the information for every single object . When he views any object , he can either modify its information or completely remove it . In either occasion he is asked to confirm his changes when he proceeds to apply them .

**Actors :**

User

### **3.2.3** **<*Search object* >**

**Description :**

The user is allowed to search for the desired objects in the database . He can use criteria for his search such as : search by product , search by date received , search by kind . When he searches for a product , he can view it ( in the way mentioned above ) ( view info / modify / remove ) .

**Actors :**

User

### **3.2.4** **<*Display items sorted* >**

**Description :**

The user can also see the products in the database sorted . Based on the situation and the demands , he can choose whether he wants a sorted representation of the products in the warehouse . The two ways of viewing them are : 1 ) sort by kind

2 ) sort by owner

**Actors :**

User

### **3.2.5** **<*Add new item* >**

**Description :**

As soon as the user is informed about a new product that has entered the warehouse , he is responsible for updating the database . He must add the new product as a new entry and also fill the information about it .

**Actors :**

User

### **3.2.6** **<*Search for empty space* >**

**Description :**

The user can search and find which tills / floors are empty in the warehouse . This way , he can know the exact state of the warehouse , and enter new products in valid places.

**Actors :**

User

### **3.2.7** **<*Recall past transfer* >**

**Description:**

The administrator ( and only him ) is allowed to see the past transfer history . Any transfer that involves the company is stored in the system so that it can be recalled / viewed / examined by the administrator . This way , he can see all of the transactions for a specific customer . Moreover , he has the ability to print a receipt / invoice or view those as well .

**Actors :**

Admin

### **3.2.8** **<*Inform admin about shipped parcels* >**

**Description :**

This use case is related to the system . Whenever a user adds a new record to the database , the administrator is instantly informed ( via email or SMS ) .

**Actors :**

User

Admin

System

### **3.2.9** **<*Warning messages* >**

**Description :**

The user sends warning messages to the administrator and the user in two cases :

1 ) There is an item of foodstock / medicine that is closing in in the expiration date , therefore it needs out as soon as possible .

2) The warehouse’s capacity is nearing a critical percentage . It needs to inform the actors that if no action is taken , ( export , remove , replace , move ) the warehouse might not be able to store any more objects.

Informs via text message and email .

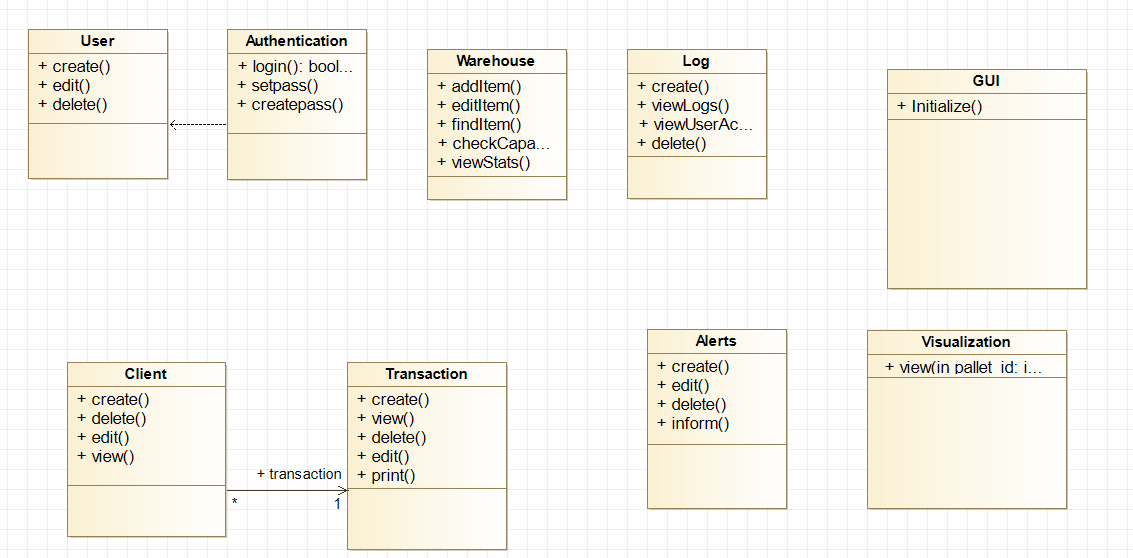
**Actors :**

User

Admin

System

# **4.** **Domain Model Class Diagram**

**

Explanation:

Almost all of the classes do nοt have attributes and the functions do nοt have input parameters.

This is because in java we do not save any data or process any data. All the data processing will happen using the SQL server/Database. The main role of the functions is to bring up a GUI , where the user will input the data , the data will go to the SQL server and the result will be shown on the screen. SQL is much faster at processing data than Java.

# **5.** **Activities Plan**

## **5.1** **Development Phases**

**PHASE 1: PLANNING AND REQUIREMENTS:**

Phase 1 was dedicated in recording all the requirements that the desired system must meet . Before meeting with our client, we sat down and thought about design ideas that would be useful to export the essential requirements. In addition, we gathered all the information we needed for our first delivery that would include the system requirements that we would try to set up in the coming phases of our project. We are also coordinated with the help of GitHub which helps everyone to keep up the pace, and of course the kanban flow which we strictly follow to be sure when and who completed each section of our project so we don’t repeat the same actions or need to communicate further to be able to continue.

We weren’t able to shoot a video at the first meeting with the client because we were in a working environment with many employees, so we recorded the meeting in a 45 minutes audio. At the end of the phase we delivered the first template with a small section of 5 minutes from the whole recording.

**PHASE 2: SYSTEM ANALYSIS AND REQUIREMENTS**

Phase 2 was focused on the extraction of system specifications from the requirements collected from the previous development phase. Then we created our MS Project plan and timetable, so that all members know what tasks are assigned to them, how much time we have to complete them and what are the deadlines for each task. Next, we created our Modelio Usage Cases, Database Charts, our Domain Model Class Diagrams.

**PHASE 3: SYSTEM DESIGN**

Phase 3 is dedicated to system design and creating the architectural side of the software. Here, we will create a more detailed and specified new UML diagram that will describe with detail the functionality of the system. We may also expand our database charts and use cases as we come closer to the completion of the system design. The end of this phase comes with the delivery of the System Design Report.

**PHASE 4: SYSTEM DEVELOPMENT**

The first prototype that we will present to our client will arise after the development of the software and it will be discussed later. Phase 4 will include the coding of the software and the SQL database. In this phase we will also present our prototype and get feedback from our client that we will use to optimise our software. This will lead to a second prototype that we will finally create.

**PHASE 5: TESTING**

In this phase we will concentrate on testing and debugging any problems that we may find in our software after implementing all the changes. Here we must optimise our software and be ready for implementation.

**PHASE 6: IMPLEMENTATION AND MAINTENANCE**

In this phase we will implement our software and finalize it, then we will give it to our client to use it. Maintenance will be a long term phase,this means that we will be available for the client if there are any problems with the software in the near future. Also, we will support our client any time needed with the software. At this point we will reach the end of the development phase.

**5.2** **Activity Plan**

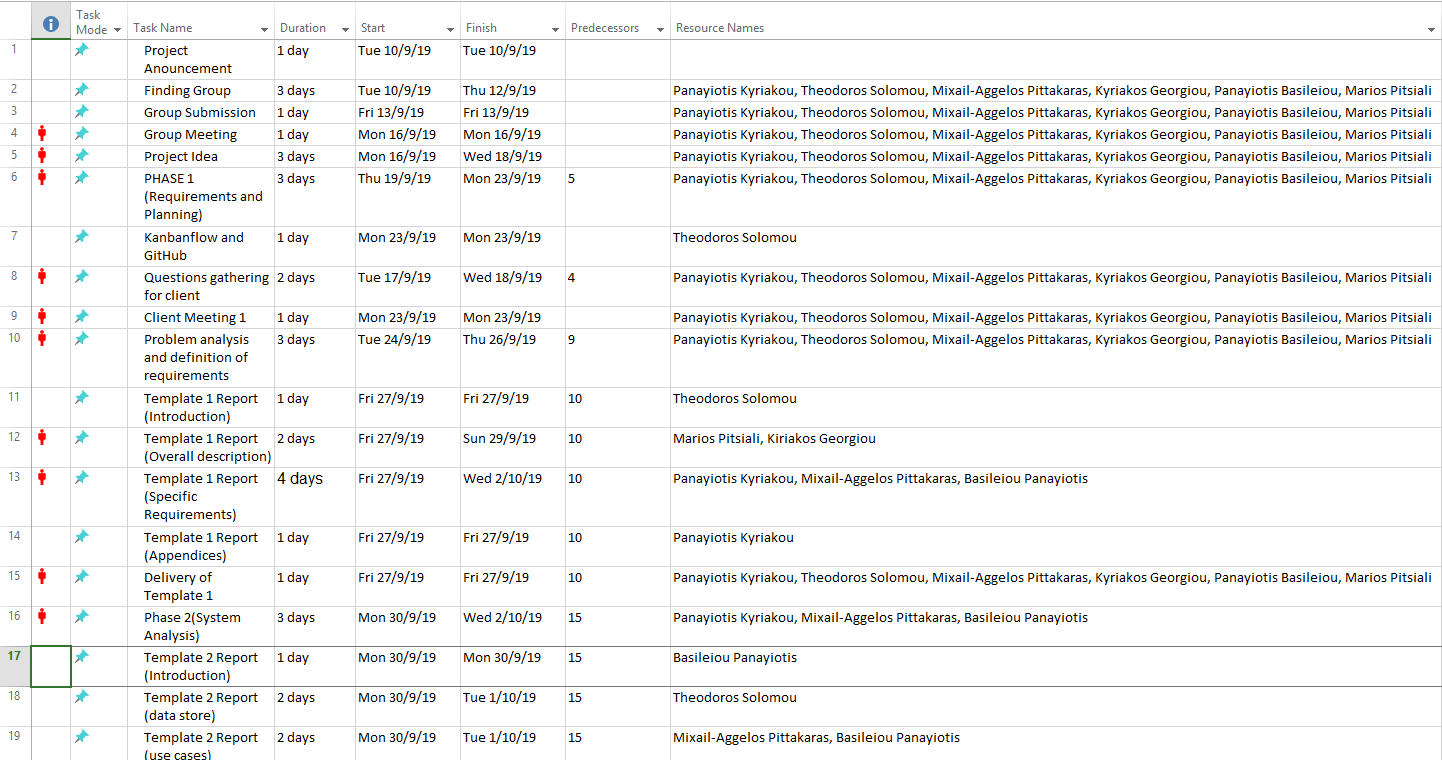
### **5.2.1** **Milestones**

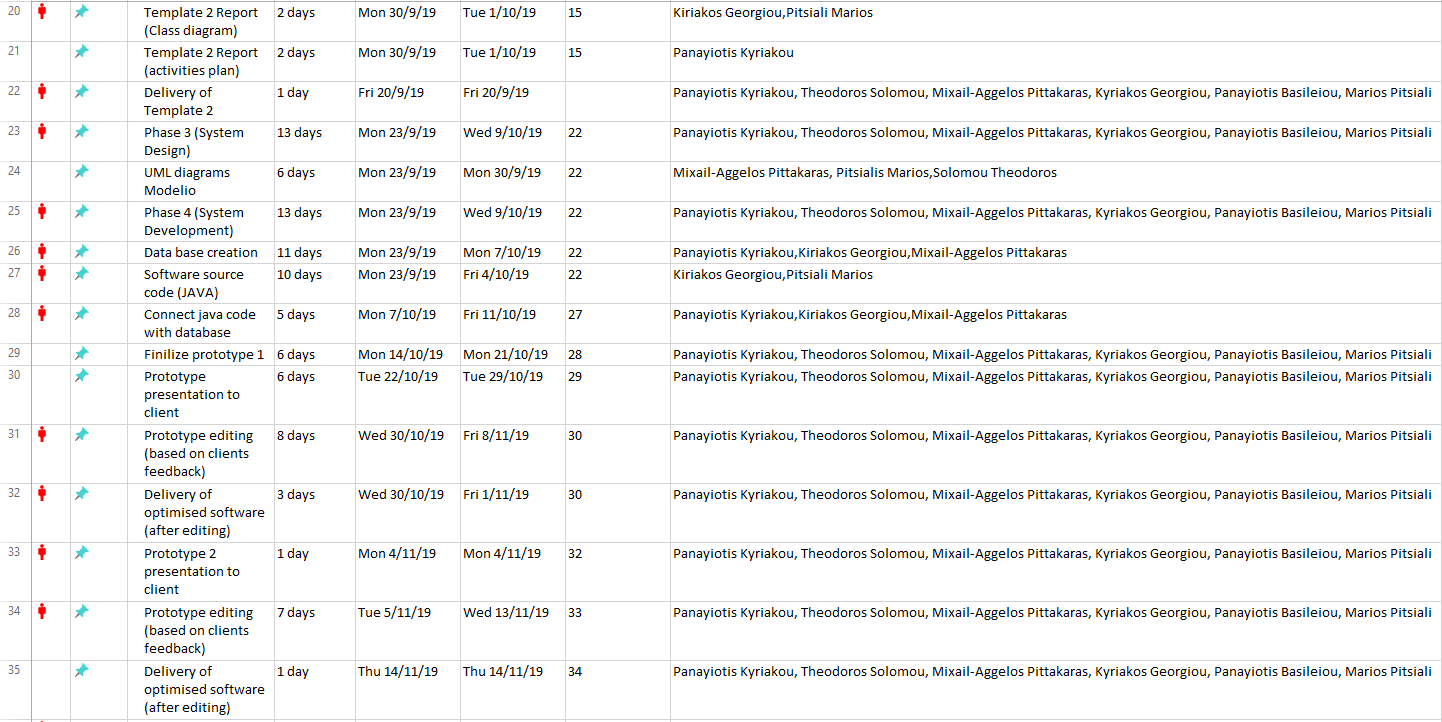
* Customer Requirements Analyzation and Interview
* Delivery of System Requirements Report
* Delivery of System Analysis and Requirements Report
* Delivery of System Design Report
* Delivery of WMS Source Code
* Delivery of SQL script
* Delivery of software

### **5.2.2** **Deliverables**

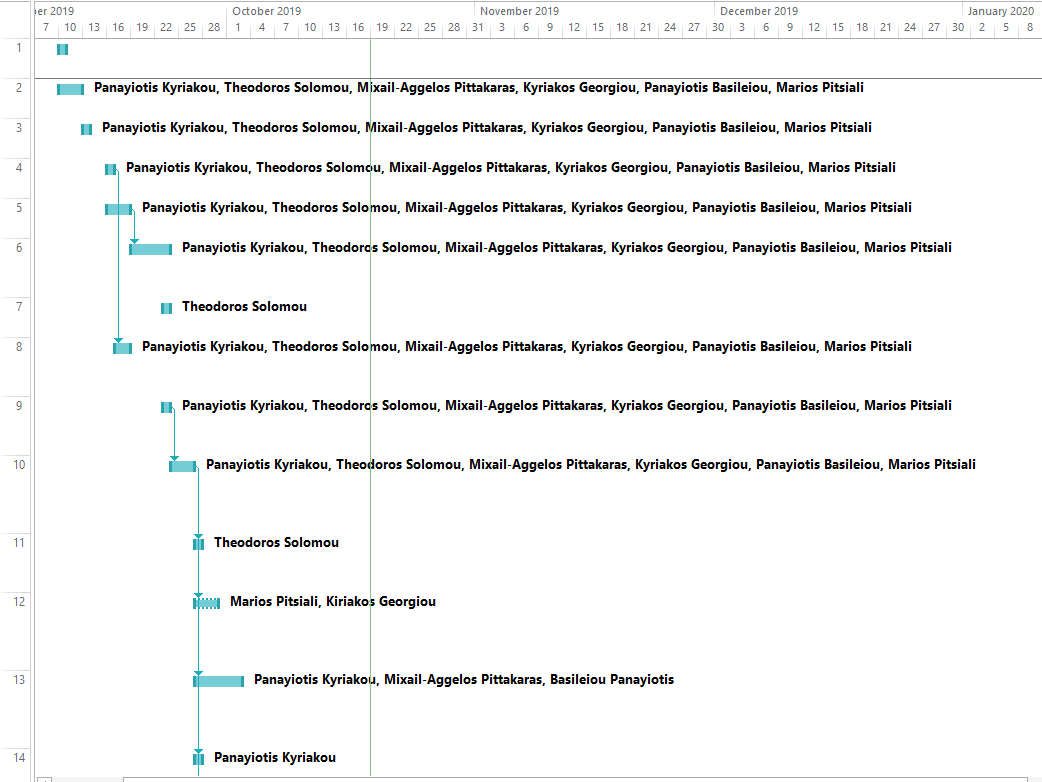
* System Requirements Report
* System Analysis and Requirements Report
* System Design Report
* Software Source Code and SQL Script

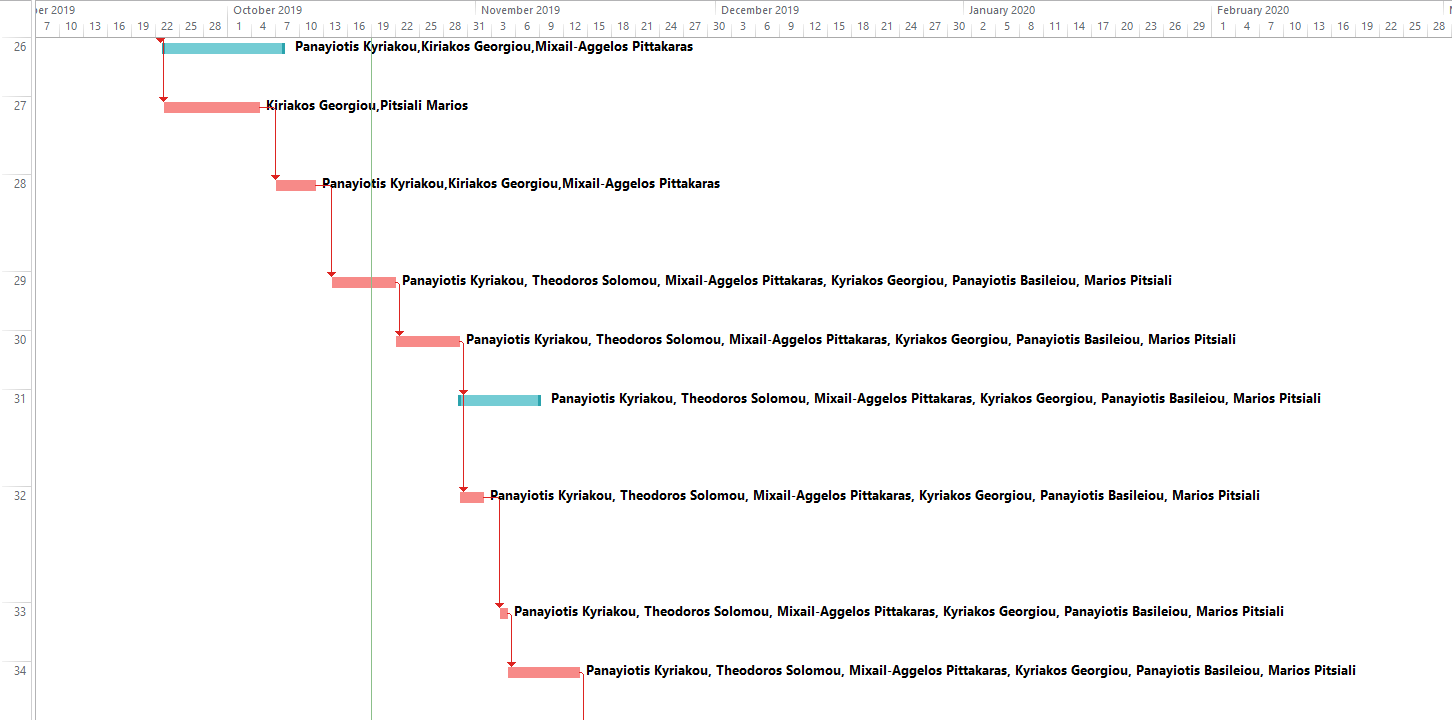
### **5.2.3** **Time plan**

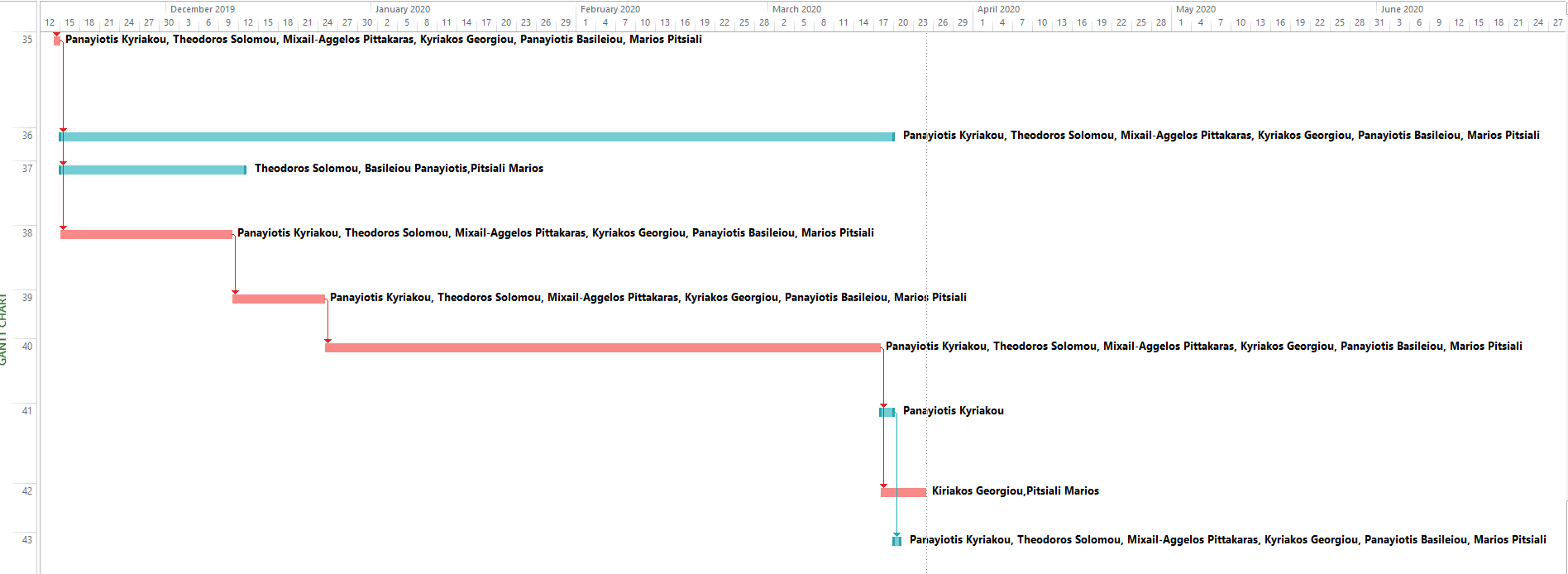












# **6.** **Appendices**