VANIER
CÉGEP/COLLEGE

Faculty of Science and Technology

420-436-VA | System Development

DELIVERABLE #6

Due Date:

Tuesday, November 19th 2024

Red Team

Amir-Georges

Douyon

Raeeba

Grechelle

Client: Georges AMO

Contact Name: Georges

SIGNATURES

We certify that this assignment is our own work.

I, **Amir-Georges**, certify that I have contributed to this deliverable, A

I, **Douyon**, certify that I have contributed to this deliverable, D

I, **Grechelle**, certify that I have contributed to this deliverable, G

I, **Raeeba**, certify that I have contributed to this deliverable, R

TABLE OF CONTENTS

EXECUTIVE OVERVIEW

In this deliverable, we worked on creating a relational database system that fit the needs of our client. To create the database design, we began by consulting our initial UML class diagram created during the third deliverable. However, we quickly realised that the latter could be improved. Therefore, we also updated our UML class diagram while working on the database design.

The database design is represented by an Entity Relationship Diagram (ERD) which visually represents the tables in our database, the relationships between them, as well as their attributes. Additionally, a narrative description of the database design, paired with a data dictionary, are also included to better explain the functioning of the system. We also took into account how to optimise queries, and the access speed as we expect the user of our application to access the database very frequently.

Alongside our modified UML class diagram and following the updated user stories and story map from the last deliverable, we also updated the other diagrams from the fourth deliverable (i.e., flowchart system, sequence diagrams, use case diagram).

BUSINESS PROBLEM

Here is our business problem, it has not changed since the last deliverable :

The client is dissatisfied with his current inventory management system, which relies on an inefficient process built in Excel. His current inventory system consists of multiple spreadsheets filled with a large volume of supply data, but lacks proper filtering and categorization of materials, making it difficult to manage and track inventory effectively. Consequently, this reduces the client's productivity and leads to frustration as the client will have to look through numerous rows of data to locate necessary materials. As the business expands, managing inventory will become even

more challenging.

We aim to address these challenges through our web application, which will optimise the business's inventory management. Additionally, since the Excel spreadsheets contain essential formulas and perform necessary calculations for construction, the web application will feature an integrated calculator to handle these calculations, enhancing both convenience and efficiency.

NARRATIVE DESCRIPTION OF THE DATABASE DESIGN

Our database has 14 tables. Specific tables exist to define users and their rights, while the others are for products.

USERS TABLES

Our application has two different types of users: Super Admin and Admin. Each user can perform a specific set of actions. Super Admins can perform all available CRUD actions, whereas Admins are limited to specific ones, notably viewing and calculating. We used the database to implement a user rights system which will be used to check which users are Super Admins and which are Admins during login.

The different types of users are defined in the ***groups*** table. Since the application only has two types of users, the ***groups*** table only has two entries : Super Admin and Admin.

The group(s) a user belongs to is defined in the ***usergroup*** tables. If a user is a Super Admin, they will also be added to the Admin group, since Super Admins have access to the same basic features as Admins.

The ***rights*** table contains all actions as well as the controller (inventory, employee, calculator) that action belongs to. For example, the 'calculate' action can only be done in the 'Calculator' page. This table, in conjunction with the groups table,

is referenced by the ***groupactions*** table. The ***groupactions*** table contains the actions users belonging to a specific group can do. For example, both Admins and Super Admins can calculate using the Calculator, but only Super Admins can delete a product from the Inventory. Admins can view products in the inventory and use the calculator, however only Super Admins can add, delete or modify products in the inventory or employees in the system.

User information (i.e., email, name, birthday) is defined in the ***userinfo*** table whose primary key is *email*.

The ***userlogin*** table references the ***userinfo*** table using the foreign key *email*. This table also has a password field. The ***userlogin*** table contains all users' login information. Their personal information is stored elsewhere, in the ***userinfo*** table. This is so if the application is ever compromised, the user's information will not immediately be taken.

PRODUCT TABLES

Products are separated into tables depending on their category: ***building***, ***glue***, ***isolant*** (insulation), and ***miscellaneous***.

All categories are defined in the ***categories*** table.

Each item in every product category table (***building***, ***glue***, ***isolant***, and ***miscellaneous***) also has a *family*. The family refers to the form the product comes in (i.e., plank, lumber, liquid, tape, spray, and physical). Each product category table references the field *family_name* from the families table. This is so each item in the table may have a defined family.

All families are defined in the ***families*** table. The ***families*** table references the foreign key *category_id* from the ***categories*** table. This is because certain categories already have specific families, or forms they can come in. For example, an adhesive cannot come in the form of a plank.

The ***suppliers*** table defines all suppliers of products (i.e., Home Depot, Rona, etc.)

Products from different categories are assembled into the ***products*** table using the foreign key *product_id* in their respective tables. The ***products*** table also references the ***categories*** table, ***families*** table, and ***suppliers*** table using the foreign keys *category_id*, *family_id*, and *supplier_id* respectively.

Super Admins can add new products to the inventory. The user must provide a name for the product in French, as well as the initial product stock, a low stock value, the unit of measurement, a product category and family, and the supplier which stocks the product when adding a product or modifying a product's information. An English name for the product can also be provided, however this is optional.

Depending on which type of product the user is adding, they have to provide additional information such as a cure time and strength for glues, and an insulation strength value for insulation. Only one product may be added at a time.

Upon adding a product, the product will be added to the respective category table it belongs to, as well as the ***products*** table. It's the same for modifying a product's information or deleting a product, both the respective category table and ***products*** table will be changed. Only one product may be modified at a time, but multiple products can be removed at a time.

APPENDIX 1 - DATA DICTIONARY

BUILDING TABLE

Field Name	*Data Type*	*Data Format*	*Field Size*	*Description*	*Example*
building_id	Integer	11111111111	11	Unique number for building products	10
product_id	Integer	11111111111	11	Unique number referenced by the product table that identifies a product	1
name	varchar		255	Name of the building product	2-inch x 4-inch x 8-ft SPF Select 2Btr Grade Lumber
namefr	varchar		255	Name of the building product in French	Planche de pin sélectionné/clair de 1 pouce x 2 pouces x 10 pieds
family	varchar		100	The form in which the product comes in, references the family table	planks
unit	varchar		50	The unit in which the product is measured	Unit(s)

CATEGORIES TABLE

Field Name	*Data Type*	*Data Format*	*Field Size*	*Description*	*Example*
category_id	Integer	11111111111	11	Unique number to	10

				identify categories	
category_name	varchar		100	Name given to the category	building

FAMILIES TABLE

Field Name	*Data Type*	*Data Format*	*Field Size*	*Description*	*Example*
family_id	Integer	11111111111	11	Unique number to identify the families	10
category_id	Integer	11111111111	11	Unique number to identify categories, referenced from the categories table	10
family_name	varchar		100	Family name	lumber

SUPPLIERS TABLE

Field Name	*Data Type*	*Data Format*	*Field Size*	*Description*	*Example*
supplier_id	Integer	11111111111	11	Unique number to identify suppliers	10
supplier_name	varchar		255	Unique name to identify the supplier	Home Depot
contact_info	varchar		255	Unique name to identify the supplier website	inventory

GLUE TABLE

Field Name	*Data Type*	*Data Format*	*Field Size*	*Description*	*Example*
glue_id	Integer	11111111111	11	Unique number for glue products	10
product_id	Integer	11111111111	11	Unique number referenced by the product table that identifies a product	1
name	varchar		255	Name of the glue product	Loctite PL Premium Max Construction Adhesive
namefr	varchar		255	Name of the glue in French	Loctite PL Construction Max Adhésive Premium
family	varchar		100	The form in which the product comes in, references the family table	liquid
unit	varchar		50	The unit in which it is measured	Unit(s)
cure_time	varchar		50	How long it takes to dry	instant
strength	varchar		50	How durable it is	Strong
glue_type	varchar		50	The type of glue	Urethane

ISOLANT TABLE

Field Name	*Data Type*	*Data Format*	*Field Size*	*Description*	*Example*
isolant_id	Integer	11111111111	11	Unique number for insulation materials	10
product_id	Integer	11111111111	11	Unique number referenced by the product table that identifies a product	1
name	varchar		255	Name of the insulation material	12 oz. Gaps and Cracks Insulating Spray Foam Sealant
namefr	varchar		255	Name of the insulation in French	12 onces. Mousse d'étanchéité isolante pour interstices et fissures
family	varchar		100	The form in which the product comes in, references the family table	Spray
unit	varchar		50	The unit in which it is measured	Bags
isolant_strenght	var		10	The strength of the insulation	Medium

MISCELLANEOUS TABLE

Field Name	*Data Type*	*Data Format*	*Field Size*	*Description*	*Example*
misc_id	Integer	11111111111	11	Unique number for building materials	10
product_id	Integer	11111111111	11	Unique number referenced by the product table that identifies a product	1
name	varchar		255	Name of the building material	6 in. x 2-Gauge 60-penny Bright Steel Smooth Shank Common Nails
namefr	varchar		255	Name of the building material in French	Clous communs à tige lisse en acier brillant de 6 po x calibre 2, 60 penny, boîte de 50 lb
family	varchar		100	The form in which the product comes in, references the family table	Nails
unit	varchar		50	The unit in which it is measured	Box(es)

PRODUCTS TABLE

Field Name	*Data Type*	*Data Format*	*Field Size*	*Description*	*Example*
product_id	Integer	11111111111	11	Unique number for products	1
category_id	Integer	11111111111	11	Unique number for different category referencing the category table	1
family_id	integer	11111111111	11	Unique number for each family referencing the family table	2
supplier_id	integer	11111111111	11	Unique number for each supplier, referencing the suppliers table	3
lowstock	integer	11111111111	11	The number of items in the inventory that are running low	1
stock	integer	11111111111	11	The number of items in the inventory	25

RIGHTS TABLE

Field Name	*Data Type*	*Data Format*	*Field Size*	*Description*	*Example*
id	Integer	11111111111	11	Unique number to identify	10

				rights/actions	
action	varchar		100	Unique name to identify the action	list
controller	varchar		100	Unique name to identify the controller	inventory

GROUPSACTION TABLE

Field Name	*Data Type*	*Data Format*	*Field Size*	*Description*	*Example*
id	Integer	11111111111	11	Unique number to identify groups action	10
group_id	Integer	11111111111	11	Unique number referencing the id of groups	1
action_id	Integer	11111111111	11	Unique number referencing the id of action	1

GROUPS TABLE

Field Name	*Data Type*	*Data Format*	*Field Size*	*Description*	*Example*
id	Integer	11111111111	11	Unique number to identify groups	10
name	Integer	11111111111	11	Unique name describing the groups	Admin

USERGROUP TABLE

Field Name	*Data Type*	*Data Format*	*Field Size*	*Description*	*Example*
id	Integer	11111111111	11	Unique number to identify suppliers	10
email	varchar		50	Unique name to identify the email referenced in the userlogin	amirgeorges.123@icloud.com
group_id	varchar		11	Unique number that identifies and references the group table	inventory

USERINFO TABLE

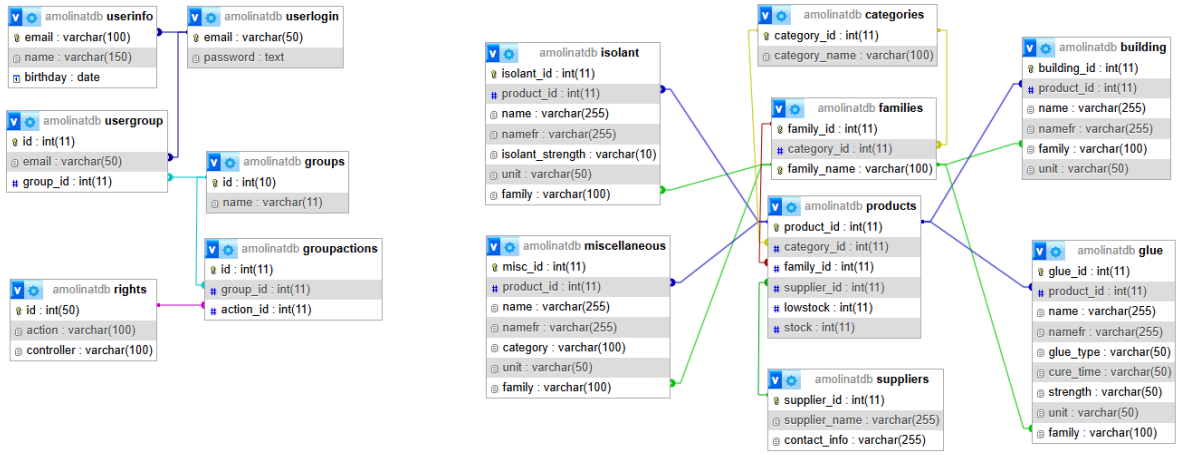
Field Name	*Data Type*	*Data Format*	*Field Size*	*Description*	*Example*
email	varchar		50	Unique name to identify the email referenced in the userlogin	amirgeorges.123@icloud.com
name	varchar		50	Unique name to identify the person	Amir-Georges
birthday	date	YYYY/MM/DD		Unique date to represent the birthday	2005-06-28

USERLOGIN TABLE

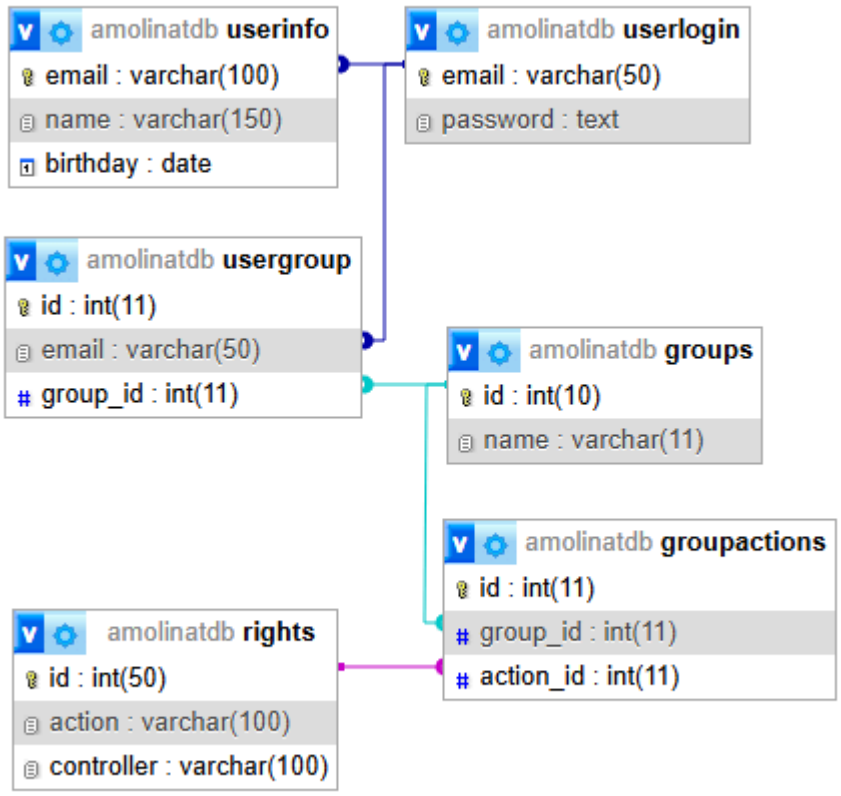
Field Name	*Data Type*	*Data Format*	*Field Size*	*Description*	*Example*
email	varchar		50	Unique name	amirgeorges.123

				to identify the email	@icloud.com
password	Text			password to identify the person	34db527779e3829fe6a4f17afd6a086ee70fd005

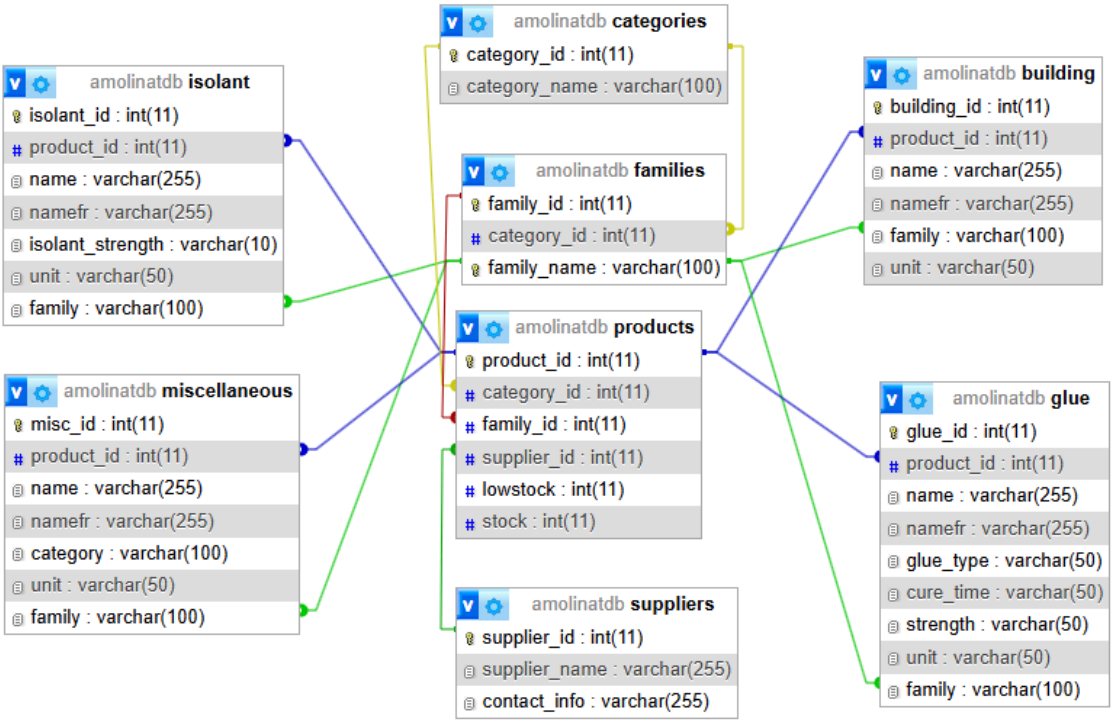
APPENDIX 2 - DIAGRAM



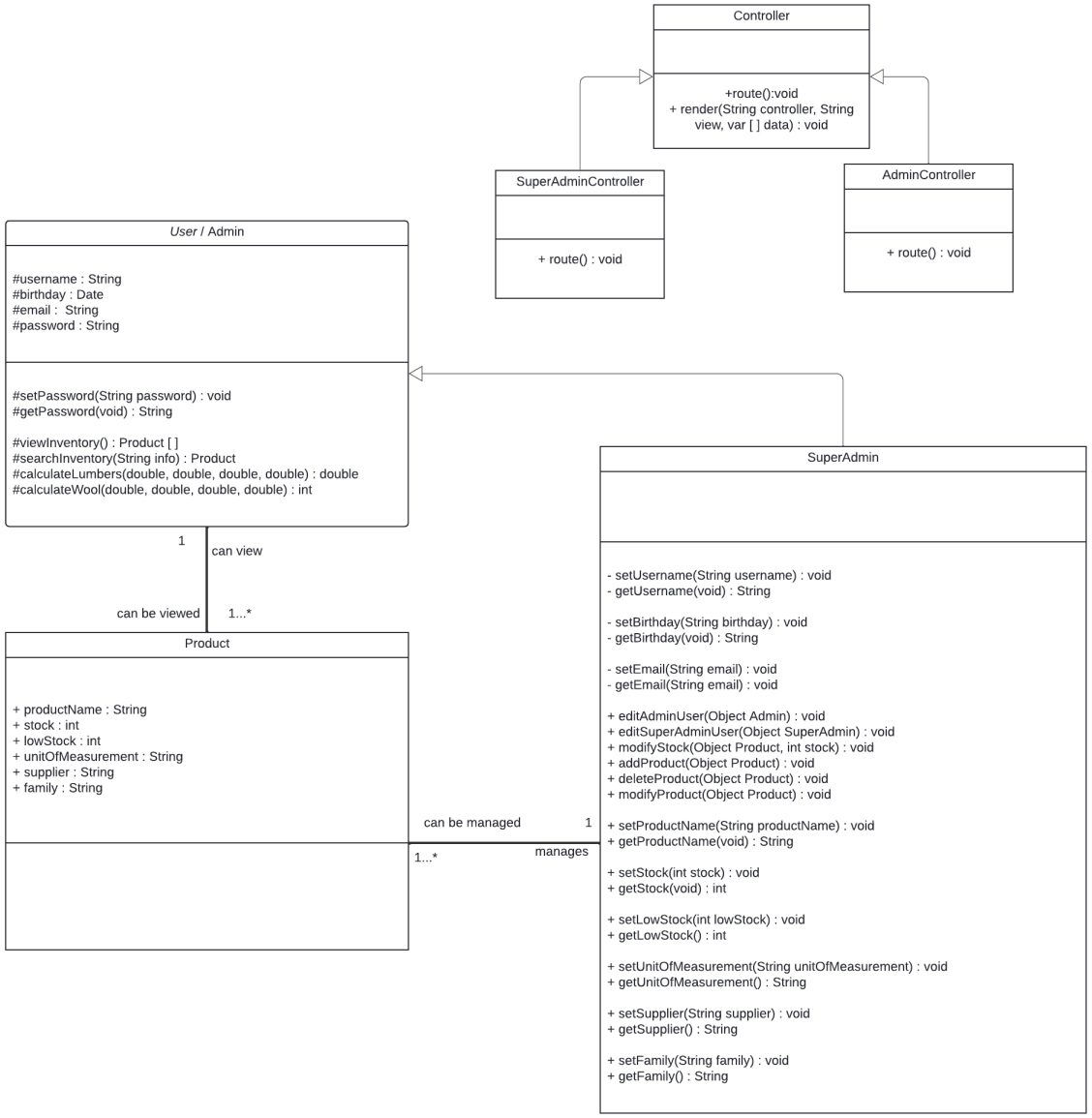
Entity Relationship Diagram ERD


amolinatdb userinfo
email : varchar(100)
name : varchar(150)
birthday : date
amolinatdb userlogin
email : varchar(50)
password : text
amolinatdb usergroup
id : int(11)
email : varchar(50)
group_id : int(11)
amolinatdb groups
id : int(10)
name : varchar(11)
amolinatdb groupactions
id : int(11)
group_id : int(11)
action_id : int(11)
amolinatdb rights
id : int(50)
action : varchar(100)
controller : varchar(100)

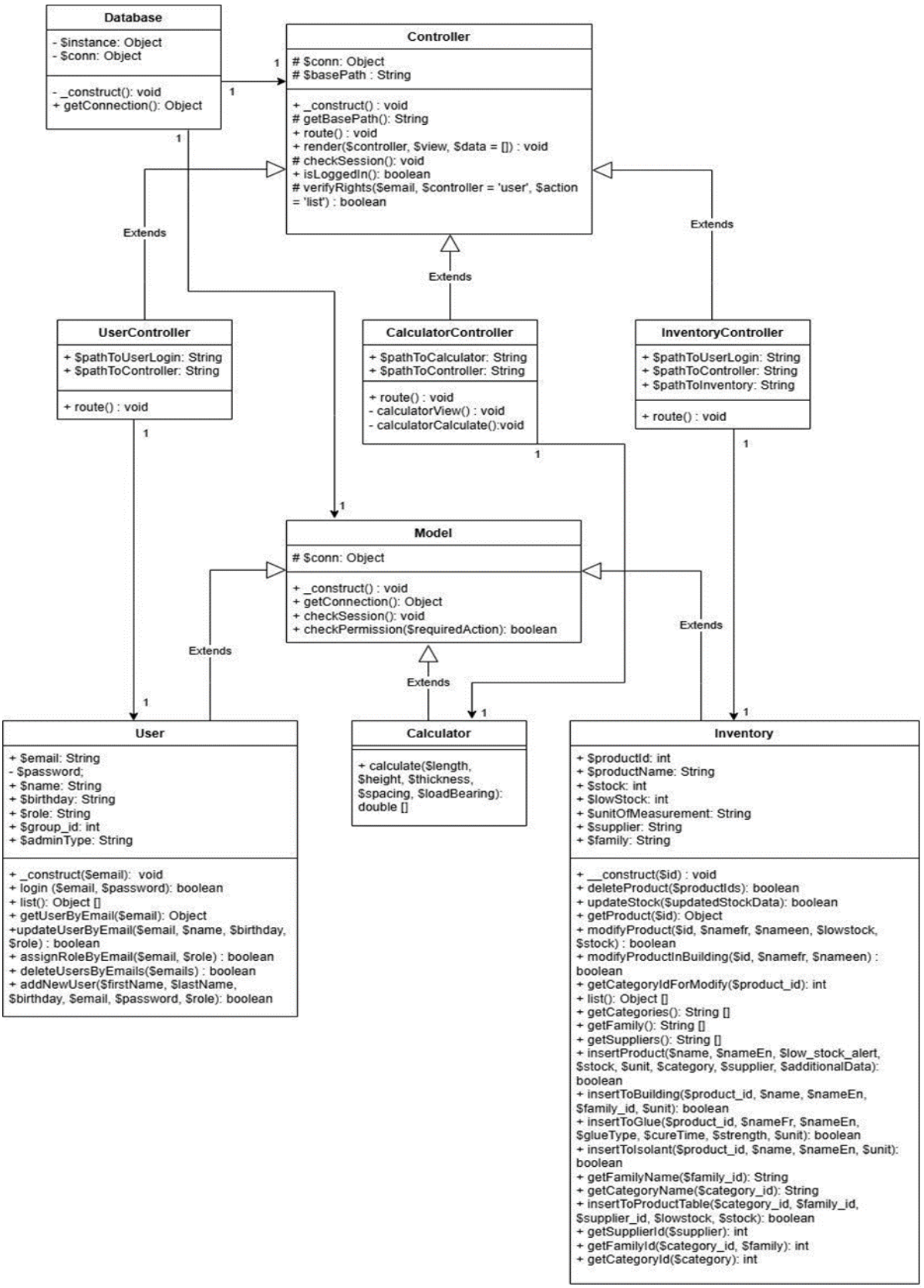
User tables



Products tables



Class diagram of Deliverable 4



Updated class diagram

We made a few changes to the initial UML class diagram in Deliverable 4. While both diagrams follow an MVC structure, the modified class diagram is more complete, as the Controller classes have additional useful methods and attributes, The $basePath attribute, for example, is used to obtain the path of the Controller class file in order for the application to be able to redirect to different pages of the application dynamically.

We added a Calculator Controller and Model class, and renamed the “Product” class to be “Inventory”. We added a Controller class for the Inventory, which we were missing in the initial class diagram.

In our initial class diagram, we created two classes for users, as our application only has two types of users: Super Admin and Admin. This was done as each user has a different level of access to the features of the application, Super Admin having access to all features while Admin can only view and calculate. However, while working on the database, we were told by our teacher that we could instead implement user rights into our database. This removes the need for each type of user to have their own class and simplifies the process of adding other types of users in the case where we decide to expand the application in the future.

UPDATED DIAGRAMS

In addition to updating the class diagram, we also updated the other diagrams from Deliverable #4. You will find them under **Appendix 5 - Updated Diagrams**.

APPENDIX 3 - QUERY OPTIMIZATION

There are going to be many queries in our application, however, most of them are quite simple. We do need to optimise the more complex queries for the sake of efficiency, notably for the following processes: the query to retrieve all the products from the database and display them on the application. In order to not slow down the application, we must optimise the queries.

In order to list the products in the inventory, we used the COALESCE() SQL function, to retrieve the first non-null value out of the following values: building name, glue name, and isolant name. We used the COALESCE() functions as only one of these values is expected to be non-null. Since each product belongs to only one of the following tables: building, glue, isolant, and miscellaneous, the name fields in the other tables would be null. We used this function in order to be able to display the name of the product in the ‘Product Name’ column.
Following this same logic, COALESCE() is also used to retrieve the first non-null value out of the following values: building unit, glue unit, and isolant unit.

We also used the SQL LEFT JOIN keyword to get the details of all products in the products table from their corresponding category tables (building, glue, isolant, miscellaneous). This lets us display the details of all the products in the database in one table on the application.

To maximise efficiency, we also used prepared statements which provide better security. Prepared statements can also be used repeatedly, which makes them more efficient.

Only the columns that were to be displayed in the table were selected in the SQL queries. For example, for the table with all products listed, the additional data fields in the glue, isolant, and miscellaneous tables are not supposed to be

displayed, thus they were not selected.

Our database is already normalised, as it follows the first three normal forms (1NF, 2NF, and 3NF).

APPENDIX 4 - ACCESS SPEED

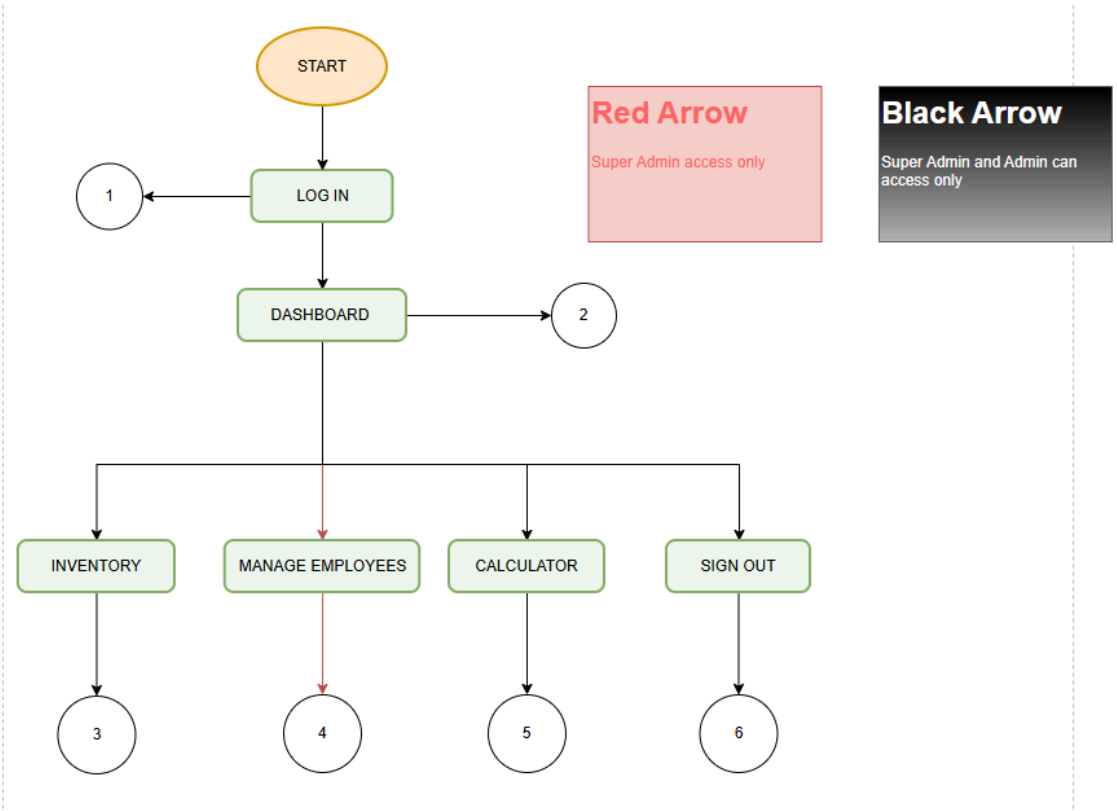
The required access speed must be fast, as the data must be accessed, updated, and removed quickly. The optimization of complex queries and normalisation of the database will permit this. We have also used Views to fetch and display the table to the dashboard to improve efficiency and readability. The database will be accessed constantly by the application since it needs to be up-to-date whenever a Super Admin user updates the inventory or employees. The response time needs to be quick to maintain seamless application performance and user experience.

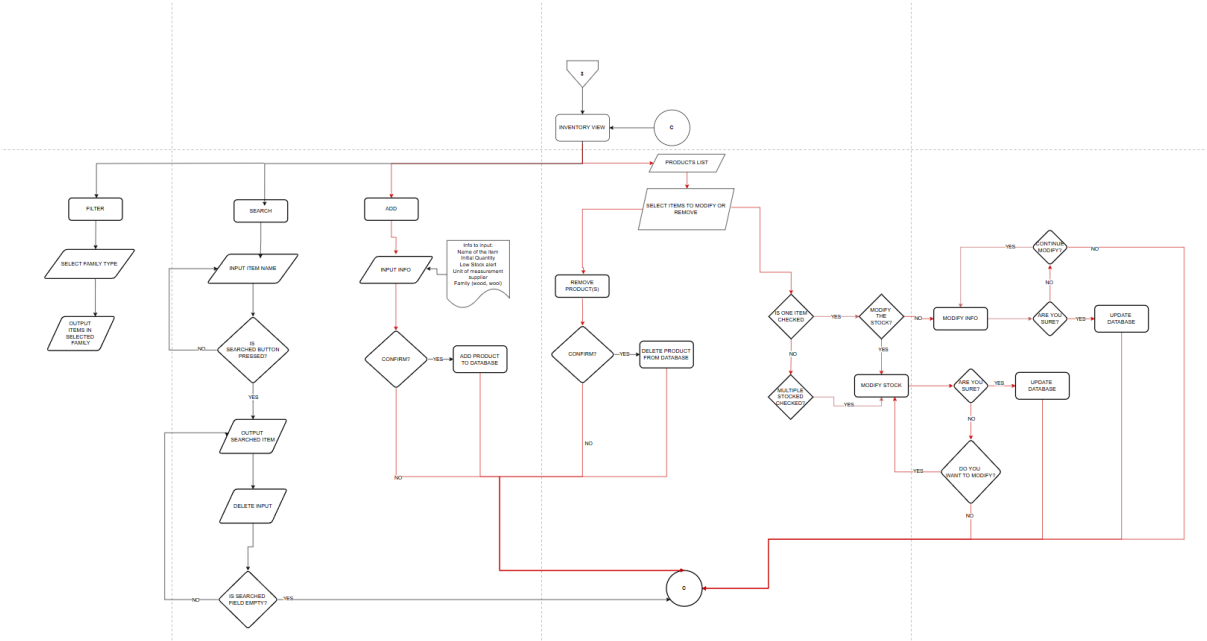
APPENDIX 5 - UPDATED DIAGRAMS

Here you will find all our updated diagrams from Deliverable #4 following the updated user stories and story map from Deliverable #5.

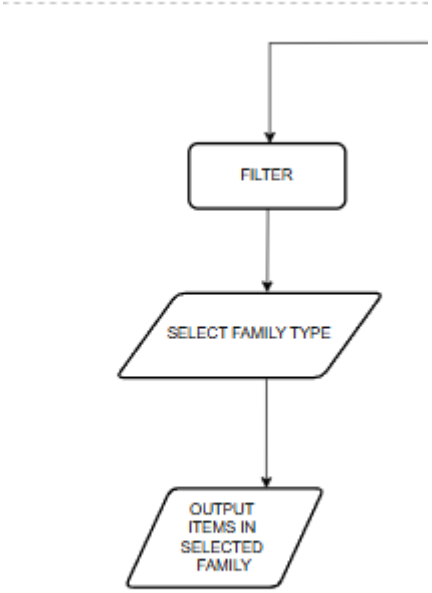
FLOWCHART

The Inventory tab was updated to better represent how the Modify Product and Remove Product features work. The "Add/Remove/Modify Employees” tab was renamed “Manage Employees”. In addition the Filter feature was added to the “Inventory” tab, and the Search feature was added to the “Manage Employees” tab.

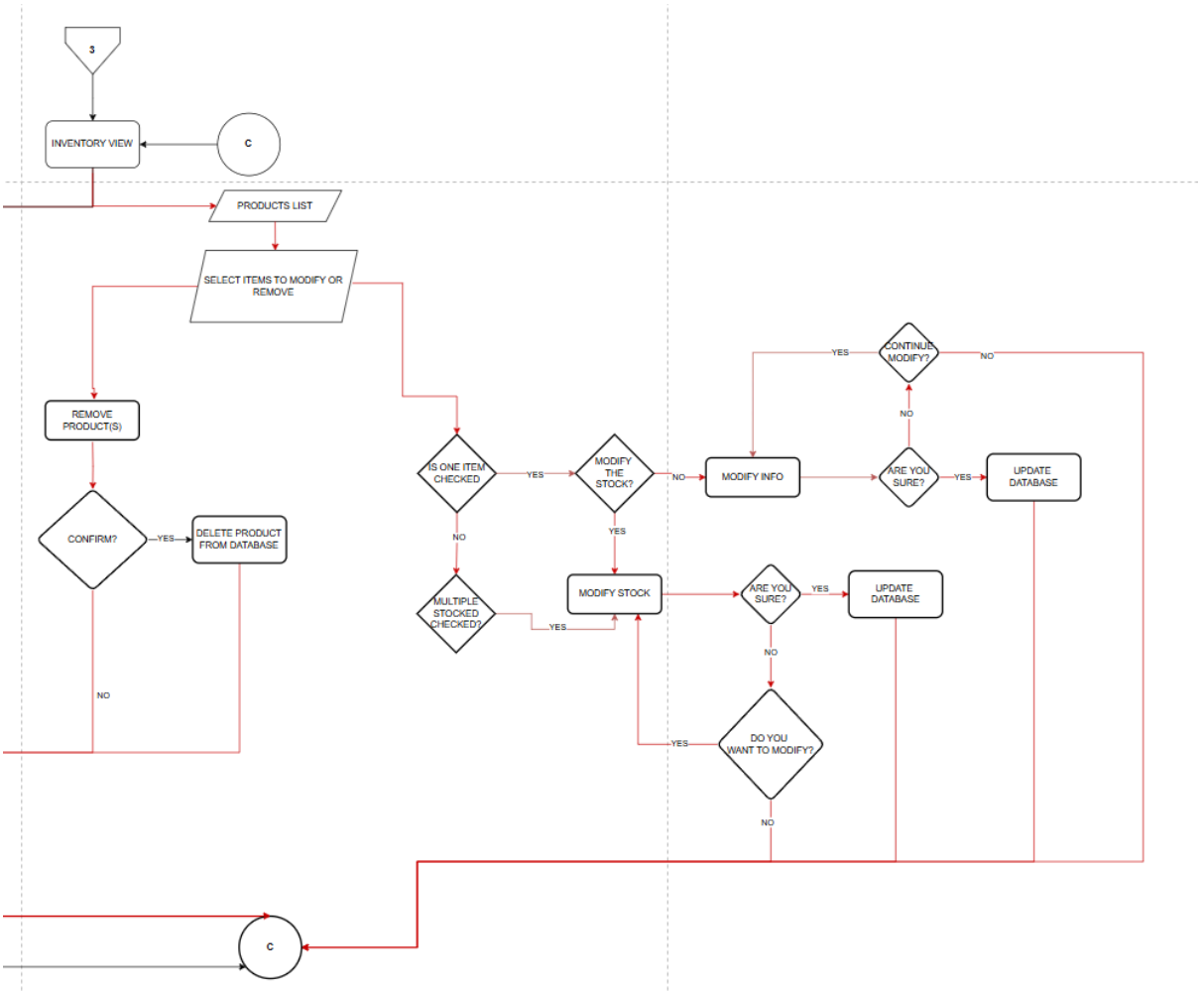

START
LOG IN
1
DASHBOARD
2
INVENTORY
MANAGE EMPLOYEES
CALCULATOR
SIGN OUT
3
4
5
6
Red Arrow
Super Admin access only
Black Arrow
Super Admin and Admin can access only



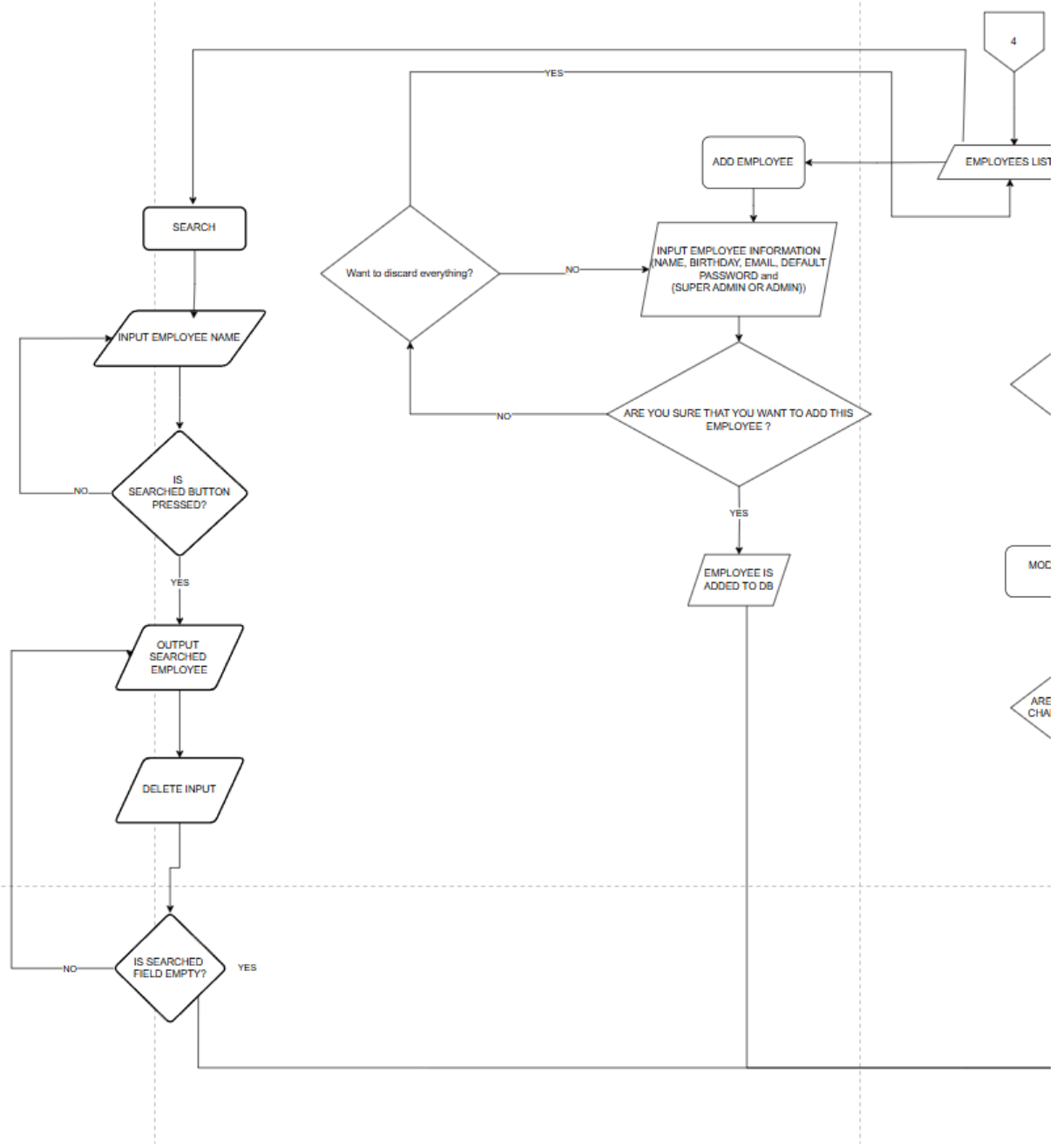
Inventory



Filter



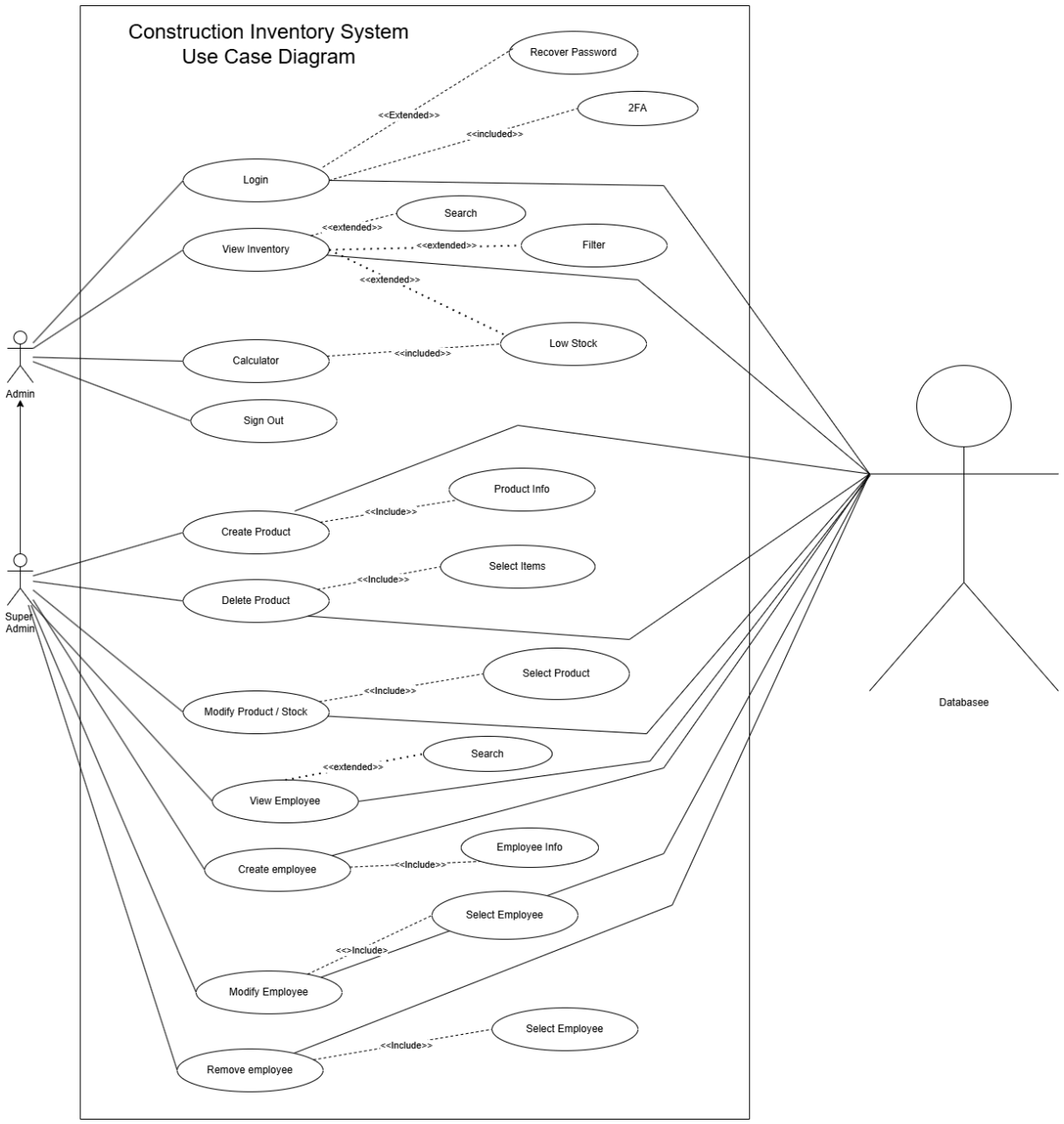
Remove and Modify Products



Search for Employees

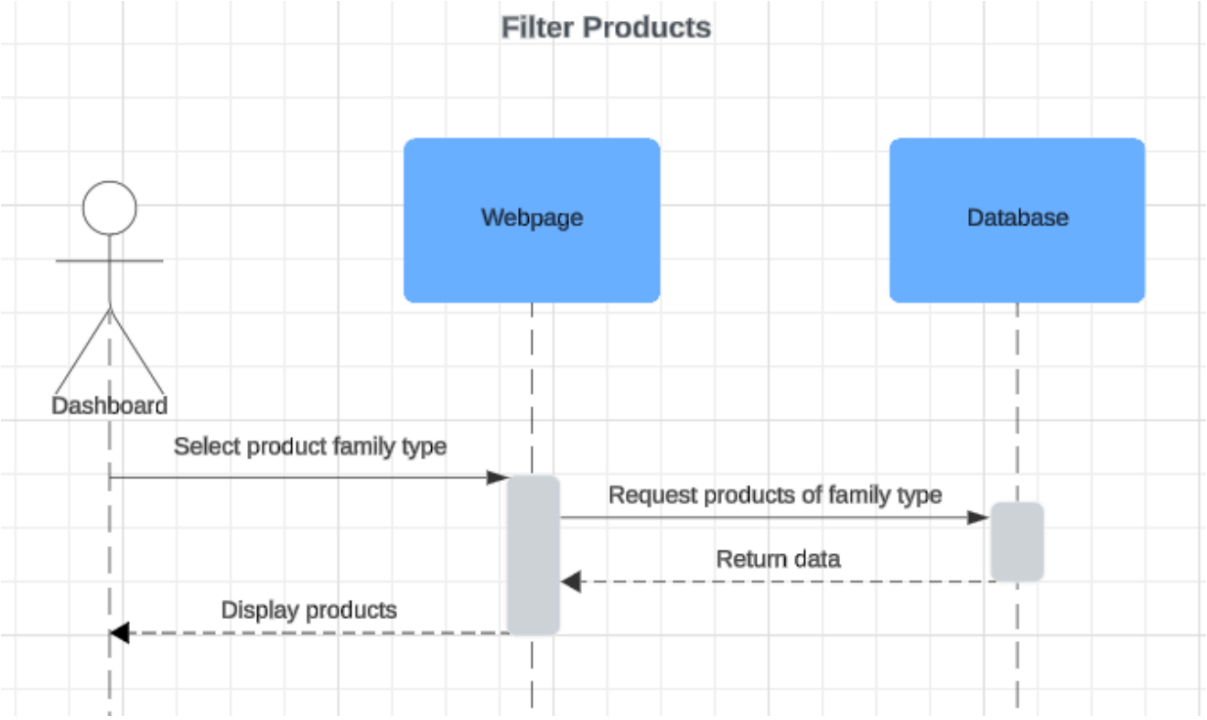
USE CASE

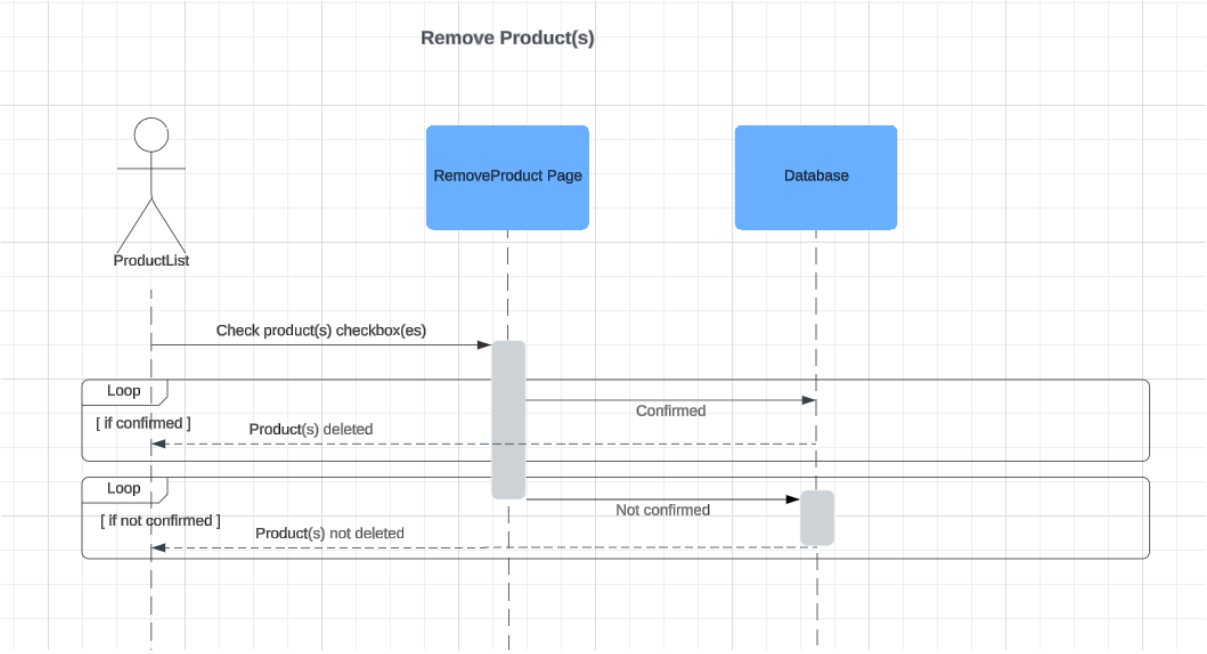
Similarly, Filter was added to View Inventory, and Search was added to View Employee.

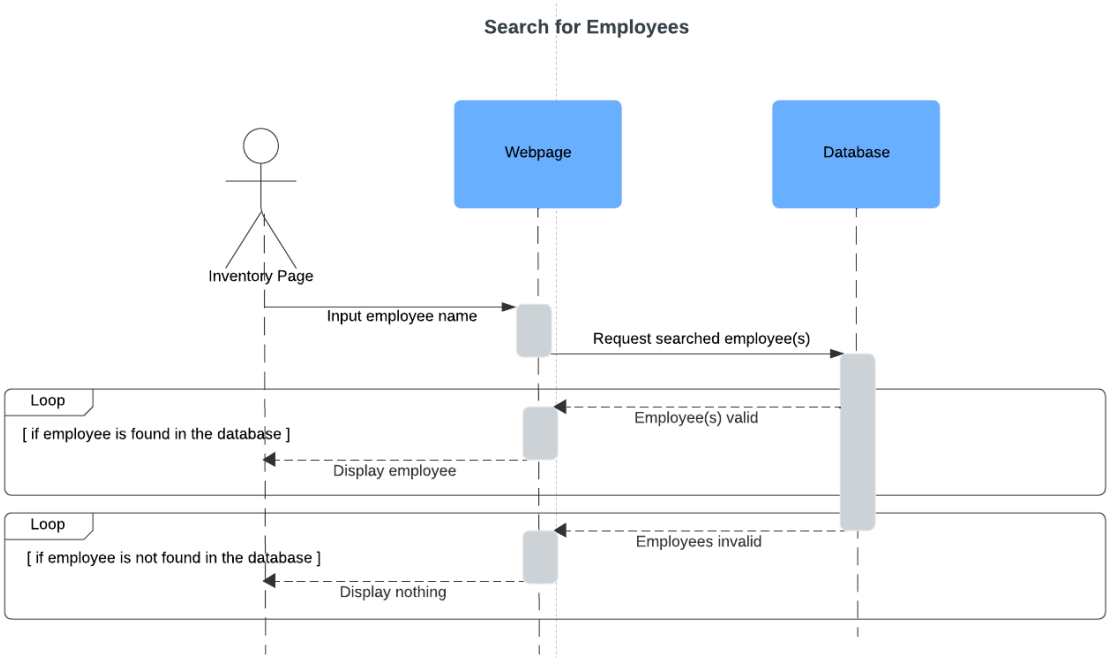


SEQUENCE DIAGRAMS

Filter Products was added, as well as Remove products and Search for Employees.


Filter Products
Webpage
Database
Dashboard
Select product family type
Request products of family type
Return data
Display products


Remove Product(s)
RemoveProduct Page
Database
ProductList
Check product(s) checkbox(es)
Loop
[if confirmed]
Confirmed
Product(s) deleted
Loop
[If not confirmed]
Not confirmed
Product(s) not deleted


Search for Employees
Webpage
Database
Inventory Page
Input employee name
Request searched employee(s)
Loop
[if employee is found in the database]
Employee(s) valid
Display employee
Loop
[if employee is not found in the database]
Employees invalid
Display nothing