

Abstract

In nearly all sorts of organizations and even simple business requires a machine-managed environment that diminishes human blunder these days. Thus a computerized environment that resides in the heart of an organization usually requires well-developed database systems that benefit the objectives and goals of that organization or business. In our project a supermarket database application manages the information about goods, products, and logistic records of the supermarket because of that it needs a well-organized system to make a profit from time which is crucial for these types of large establishments.

A Summary of the Application and Requirements

The first thing users come across is a login page. There are pre-registered accounts on the database. Note that there is no account creation because it is assumed that users of this application are not regular people but employees of the supermarket thus account is assumed to be created by the establishment that utilizes this application on supermarket warehouses. For further operations, the user must enter his/her account correctly in order to access operations that are provided by the application. These pre-registered accounts are as follows :

Username : Hamza

Password : 123

Username: Yigit

Password : 123

Username: ABC

Password: 123

Note that these username and password fields are case sensitive thus there is a difference between username 'Hamza' and 'hamza'.

After a user enters his/her account then further operations can be accessible on the web interface.

Program comes with several python script files, at the first run it is important to follow these steps

- Run the **connect.py** file to connect to the database. This script includes configuration for connecting to specific users on MySQL database. To appropriately run connect.py **username, password and host names must be configured according to users' MySQL credentials present in the environment.** After the connect.py file successfully executes, connection should be established between the MySQL system and the Flask program.
- After executing the connect.py file, the user must run the **config.py** file in order to create a database, tables and data that populate related tables. It should be enough to run it once.

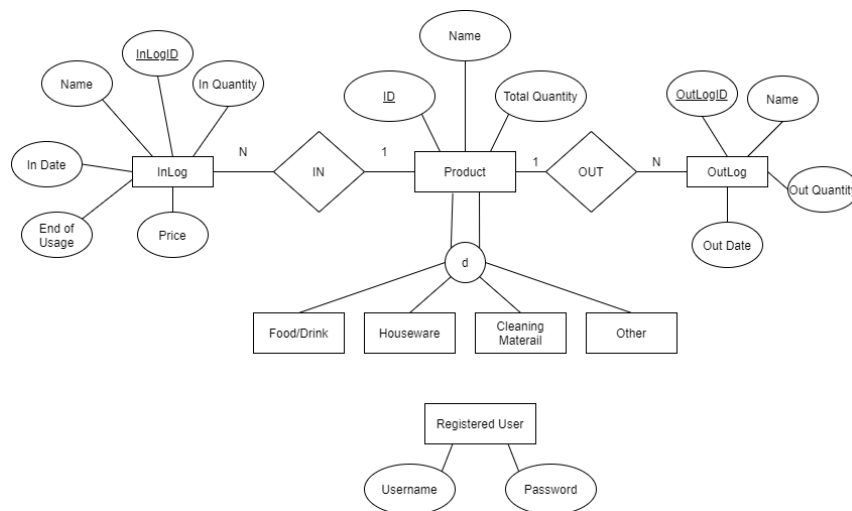
- After config.py file execution, the user can run the app.py file which starts the Flask app that starts on localhost.

After successfully done all the steps and entered credentials correctly user can access several pages which is listed below :

- Main page
- Cleaning material page
- Food drink page
- Houseware page
- Other material page
- Incoming goods page
- Outgoing goods page

All of the operations done on the mainpage which has a form that collects data from the user then sends it to the program for further processing. These operations affect tables present on the database.

Entity-Relationship Design



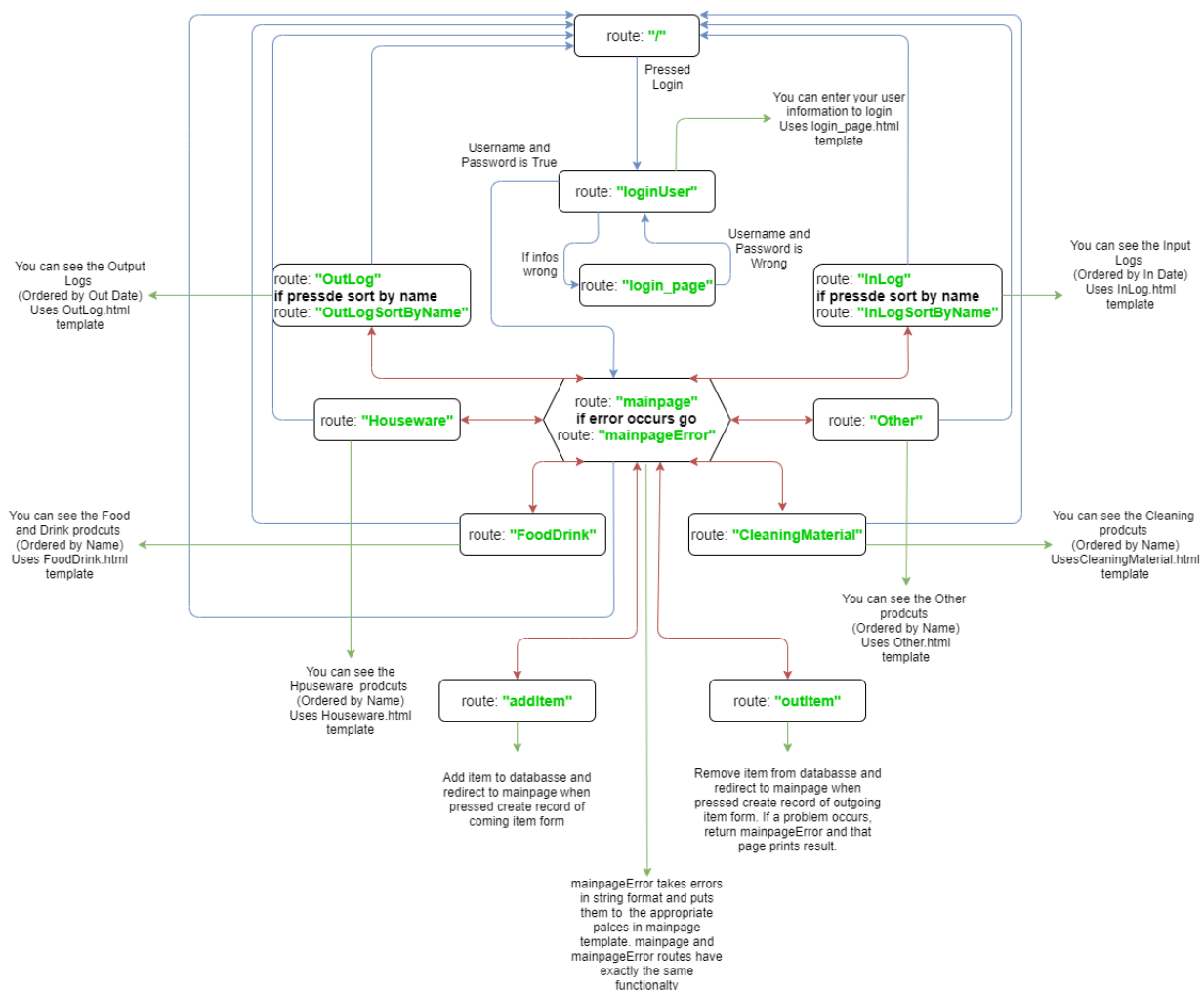
Data Requirements

- * Product is the main source for the other subclasses (Food/Drink, Cleaning, Houseware, Other). Every product must belong to one of the four subclasses.
- * Product and its subclasses store a unique id for each product, name of the product and the total quantity of the same product for each entry.
- * Name is a candidate key in product table and its subclasses (Means name is unique for each product) but we chose ID as primary key.
- * The "InLog" contains information about the arrived deliveries. Each delivery has name and id of the product, quantity delivered, delivery date, unit price of the product, a unique id and end of usage time (nullable because housewares or others have no end of usage date)

* The “Out Log” contains information about gone products (Products goes to the market). Each entry of “Out Log” contains an unique “out id”, date of out day, quantity of out product, name and id of the product.

Our thinking process while designing the ERD and RM, multiple subclasses inherits from PRODUCT entity which has PK as ID, the assumption here was multiple entities inheriting from superclass these entities have total participation. The PRODUCT entity has a total quantity attribute that increases or decreases according to arrival or outgoing of the goods to the supermarket. For example, if any goods arrive at the supermarket depot it is added as another row to the IN LOG with the quantity of newly arrived goods and if it does not exist in the PRODUCT entity then a new row is added for that product to the table but if it exists in the table it updates the total quantity attribute. For the OUT LOG entity, the IN LOG mechanism is valid. Each ID and Name is unique for the PRODUCT. Also, OUT LOG ID and IN LOG ID are unique for each instance.

The General Hierarchical Structure For the Application



You can see the flow of the application in the above chart.

The program starts with the initial route “/”. That route prints the login page template. When the login form is submitted to the server it changes route to the “loginUser” route. Here it checks the entered information, whether is it a valid account or not. The user information is located on the server. If the given information is not valid then the application changes the route to the “login_page”. Here it prints some error message and clears the login form. Users can try to log in again or leave the application. If the given account is a valid one then it changes the route to the “mainpage”. On the main page, the user can do several things.

1) Users can look into the four types of item tables, in log table and out log table. The product tables are ordered by name. The in log and out log tables are ordered by the date attribute but the user can change the order criteria by name for these two tables. There is no other functionality on these pages. When the buttons are activated, the application gets proper routes. The routes below:

* Food/Drink table	=>	route: /FoodDrink
* Houseware	=>	route: /Houseware
* Cleaning Material	=>	route: /CleaningMaterial
* Other	=>	route: /Other
* In Log (sorted by Date)	=>	route: /InLog
* Out Log (sorted by Date)	=>	route: /OutLog
* In Log (sorted by Name)	=>	route: /InLogSortByName
* Out Log (sorted by Name)	=>	route: / OutLogSortByName

2) Users can look into the four types of item tables, in log table and out log table. The product tables are ordered by name. The in log and out log tables are ordered by the date attribute but the user can change the order criteria by name for these two tables. There is no other functionality on these pages. When the buttons are activated, the application gets proper routes. The routes below:

3) Users can create records for outgoing items. When the form is filled and sent the application changes the route to the "outItem". When the form is filled in the wrong way (wrong, id quantity, type) application changes the route "/mainpageError". The only difference between "mainpage" and this page is the error message for the outgoing form. If the entries are correct then it creates the proper Log data and updates the database.

Additional Files

The config.py file is executed in the order mentioned in the beginning of the report. After this file executed all the tables, instances and relations created with the specified queries in the file. After these queries executed database becomes populated with instances that obeys the rules that are specified with the model. You can view all of the queries that are used to create database, tables and instances in the config.py file. Some of the queries that is included in this file are below:

Below sql is for creating in_log table which stores the in_log records:

```
CREATE TABLE in_log( ID INTEGER, Name VARCHAR(100), In_Quantity INTEGER, In_Date
Date, EUD Date, In_Log_ID INTEGER, Price FLOAT, Foreign Key(ID) references product(ID),
PRIMARY KEY(In_Log_ID) );
```

Below query is for creating instance for the in_log table:

```
INSERT INTO In_log (ID, Name, In_Date, EUD, In_Log_ID, In_Quantity, Price) VALUES (756443, 'Sponges', '2021-4-28', NULL, 986214281, 9000, 1100);
```

Below query is for creating registered_user table which stores the user credentials:

```
CREATE TABLE Registered_User( Username VARBINARY(100), Password VARBINARY(100), PRIMARY KEY(Username) );
```

Below query is for creating instances for the registered_user table:

```
INSERT INTO Registered_User (Username, Password) VALUES ('Yigit', '123');
```

```
INSERT INTO Registered_User (Username, Password) VALUES ('Hamza', '123');
```

Below query is for creating product table which have several connections to related tables that are explained deeply in the report:

```
CREATE TABLE product( ID INTEGER, Name VARCHAR(100), Quantity INTEGER, PRIMARY KEY(ID) );
```

Populating product table with the below query:

```
INSERT INTO product (ID, Name, Quantity) VALUES (756443, 'Sponges', 10000);
```

The **Utils.py** file contains the helper functions. You can see the name of the functions and an explanation for them.

- **generate_product_id()** => Returns an unique id for the product.
- **generate_in_log_id()** => Returns an unique id for an inLog record.
- **generate_in_out_id()** => Returns an unique id for an outLog record.
- **is_unique(name)** => Checks whether given product name is exist in the database or not. If exists, returns product's id and quantity. If not, returns -1,-1
- **check_user(username, password)** => Checks the given username and password is valid. If valid returns true. If not, returns false.
- **get_item_byID(ID,type)** => Finds the product by using id and type. If finds return quantity and name of the product. If not, returns -1,-1.
- **sortTuples(elementList, x = "ID")** => Sorts the given sequence by using the metric sorting metric "x".

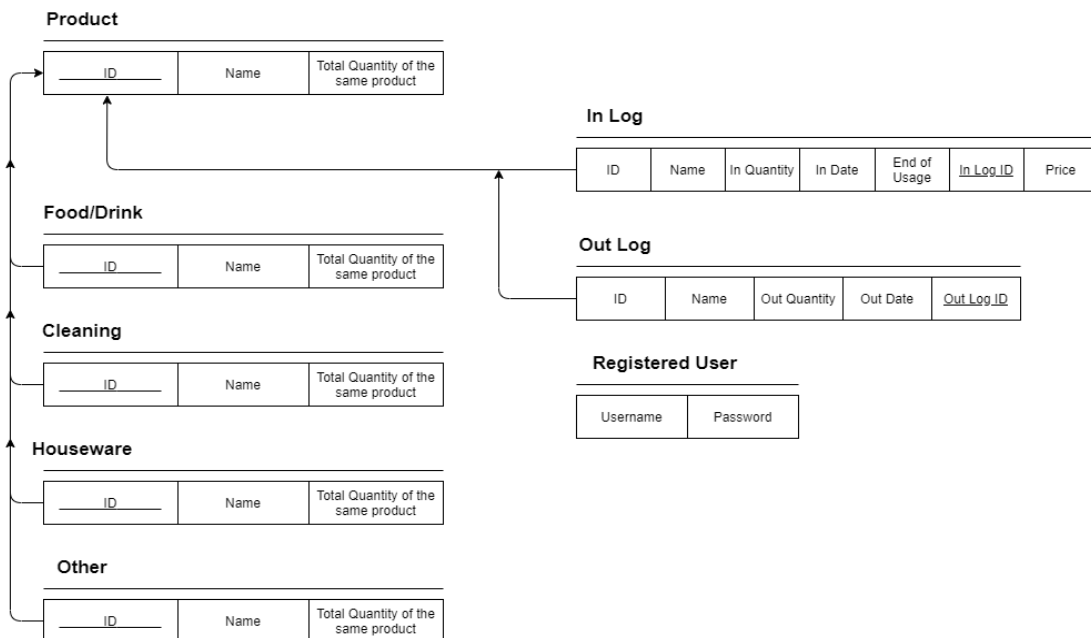
The **app.py** function also includes some functions to edit and get information from database.

- **addNewItem(item_id, new_item)** => Add new item to the databse. Item_id is the new unique id for the item and new_item is a list that contains the proporites of the new item and creates the inLog record for the newly arrived item.

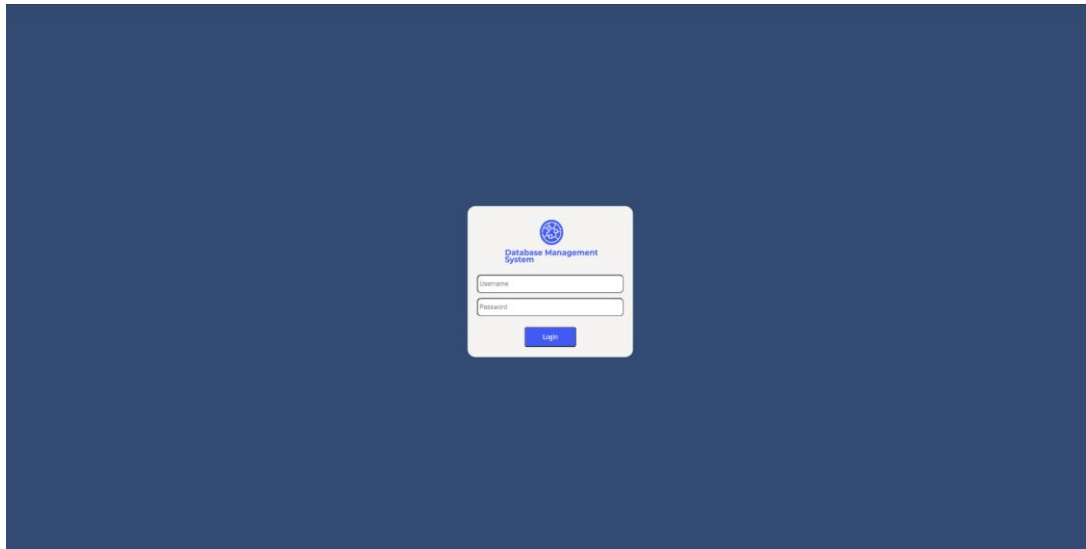
- **updateInItem(isID, new_item)** => Updates quantity of an existing item and creates the inLog record for the newly arrived item. isID contains the id and the quantity of the item. New_item contains the information about the coming item.
- **updateOutItem(new_item, item)** => Updates the outgoing item's quantity. new_item contains the given form information and the "item" is the existing item in the database.
- **getCountsAndQuantities()** => Returns number of unique items and the total number of items for each of the item table.

The Logical Database Design

Updated version of relational model shown below. Registered User table added to store username and password credentials of the user.



User Manual



After 127.0.0.1:5000 run on the browser login page will be loaded on the browser. To access the application, user must enter his/her credentials into the “Username”, “Password” field on the login form then press the Login button in order to redirect to the mainpage. If credentials are typed wrong to the text field then a warning message will show up saying “Username or Password incorrect”. And the program does not accept this field to be empty.

After logged in to the application main page will be loaded on the browser

	Houseware	Cleaning Material	Food/Drink	Other
Number of Types	2	6	5	3
Quantity	6000	28000	12000	13000

Record Coming Items
Name
Quantity
Cost
End of Usage Date:
Type of Item:
Create Record

Record Outgoing Items
ID
Quantity
Type of Item:
Create Record

In this page, the product table shows a brief summary of overall products in the supermarket and there are two forms “Record Coming Items” and “Record Outgoing Items”.

“RecordComing Items” collects inputs for incoming deliveries to the supermarket. Fields on the form cannot be empty. These fields are Name, Quantity, Cost, End of Usage Date, Type of Item. Name of the goods arriving at the supermarket must be written to the Name field on the form. Total quantity arrived at the supermarket must be written to the Quantity field. If the arrival good has an end of usage date then it should be recorded with the date.

May 2021

Su	Mo	Tu	We	Th	Fr	Sa
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

Today

mm/dd/yyyy

Type of Item

Food/Drink

Create Record

End of usage date
should be chosen from
the calendar

Record Coming Items

Name

Quantity

Cost

End of Usage Date:

mm/dd/yyyy

Type of Item

Food/Drink

Create Record

End of usage date should
be leave as it is. Highlight
with red line

There are four choices on the form for the type of the good “Food/Drink”, “Cleaning”, “Houseware”, “Other”. Checking the type of the goods is not possible using only the program so it must be selected correctly by the user, if not then goods will be inserted into the wrong table. For example, if “Food/Drink” types of goods arrived at the supermarket and on the “Record Coming Items” form if type selected as “Other” than good will be inserted into the “Other” table.

“Record Outgoing Items” have, ID, Quantity, Type of Item fields, these fields cannot be empty. Outgoing items should be recorded by entering the ID of the item.

ID must be written correctly in order to successfully create the record. ID(s) of the outgoing good(s) can be viewed from the related tables. If a wrong ID is written that exists on the supermarket then it will make the changes on that item or if an ID that does not exist on the database is written it will not make the changes and will show a brief warning message.

Quantity of the outgoing items should be written to the Quantity field on the form. If wrong amount of quantity is entered to the field than program will not accepts the changes and show an

Record Outgoing Items

ID

Quantity

Type of Item

Food/Drink

Wrong ID, Type or Quantity

Create Record

error message, such as, if written quantity on the form exceeds the total quantity of that item currently stored in the supermarket then system do not make the changes and show a brief warning:

Type of the good must be selected correctly in order to update the item or good. If the wrong type of good is selected then changes will not be reflected to the system. For example if “Food/Drink” type is outgoing from the supermarket and “Other” types selected then changes will not be reflected to the system instead the program will show a brief warning message.

A demonstration of the creating record for outgoing item explained with figures as well :

Cleaning Products

ID Number	Name	Quantity
645335	Brush	1000
876534	Bucket	1000
234234	Microfibre cloths	1000
865235	Protective Gloves	5000
756443	Sponges	10000
123432	Toilet Paper	10000

Go Back

Copy ID of the Brush and paste it to “Record Outgoing Items” form ID field

Record Outgoing Items

ID

645335

Quantity

100

Type of Item

Cleaning

Create Record

Notice that outgoing quantity does not exceed the total quantity (Brush quantity is 1000 which can be viewed on the above highlighted table) and type is selected correctly which Cleaning after “Create Record” button pressed results will look like below.

Quantity of the brush dropped down to 900 as expected.

Cleaning Products

ID Number	Name	Quantity
645335	Brush	900
876534	Bucket	1000
234234	Microfibre cloths	1000
865235	Protective Gloves	5000
756443	Sponges	10000
123432	Toilet Paper	10000

Go Back

Each item will be displayed according to their types on the related pages; these pages can be accessed through the buttons on the main page highlighted above. These buttons redirect you to the several pages listed below.

Database Management System Logout

Product Summary

	Houseware	Cleaning Material	Food/Drink	Other
Number of Types	2	6	5	3
Quantity	6000	28000	12000	13000

Food/Drink Cleaning Material Houseware Other In Log Out Log

Record Coming Items

Name

Quantity

Cost

End of Usage Date:

Type of Item:

Create Record

Record Outgoing Items

ID

Quantity

Type of Item:

Create Record

After each incoming delivery, if a new product arrives it is added as a new row, if the product already exists quantity attribute updated, quantity is decreased for the goods that are outgoing from the supermarket.


- Cleaning Material page : This page lists available cleaning products stored in the supermarket.
- Food / Drink page : This page displays food / drink products stored in the supermarket
- Houseware page : This page displays goods present on the supermarket
- Other page : This page displays the goods that are different from cleaning, food/drink, houseware type of goods. Such as a mobile phone is listed on the others page.
- In Log page : Displays the record of the goods that arrived at the supermarket. This page shows more detailed information about the goods such as delivery date and End of Usage date which has a value for specific type of goods. For goods that do not have end of usage attribute the table shows "None".
- Out Log page : This page shows outgoing goods from the supermarket.

Note that above listed pages do not allow any other user input besides the main page, these are just for viewing the product information.

With go back buttons on each page users can go back to the main page. Each page has a logout button which allows the user to logout from the system.

Product tables sorted by name but inlog and outlog tables sorted by record date from newest to oldest, if the user wants to list inlog and outlog tables than using “sort by name” button records can be sorted by name.

Inlog page


Logout

Input Logs

Input ID	Item ID	Name	In Quantity	In Date	Cost	End of Usage
256211104	845634	Battery	5000	2021-01-10	1000.0	None
235211101	512676	Blanket	3000	2021-01-10	2000.0	None
432212209	645335	Brush	1000	2021-03-20	4000.0	None
642212202	876534	Bucket	1000	2021-02-20	5000.0	None
325212203	543455	Cake	2000	2021-02-20	10000.0	2021-03-01
865212205	745652	Coke	2000	2021-02-20	9000.0	2022-01-01
235211102	121346	Earphone	3000	2021-01-10	2200.0	None

Go BackSort by Name

Outlog page

Logout

Out Logs

Input ID	Item ID	Name	Out Quantity	Out Date
525214291	756443	Sponges	100	2021-04-29

Go BackSort by Name