جامعة الأخوين

ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⵓⵅⴰⵡⵉⵏ

AL AKHAWAYN UNIVERSITY

Fall 2021

Friday, October 29, 2021

CSC 332 – Database systems

« WeHelpYou »

Soufiane Tounsi

Salma Louhaychi

Ziad El Ismaili

Hamza Zaher

Supervised by:

Pr. Lamiae Bouanane

Table of Content

I. Introduction:

a) Project description:

Many small businesses such as shops, retails stores and hardware stores lack databases in their line of work; in fact, owners usually operate by hand, doing calculations rather than scanning items in a specific device, which is clearly not practical since implementing a database for instance, would clearly help owners by organizing information about the shop, making it easier to access, manage and update the items. It could also help with transactions and product inventory, as well as improving the overall customer satisfaction since fast and easy service will be provided.

However, one of the reasons owners would not implement a database in their commerce might be the lack of interest and knowledge about it, but most importantly, they seem to have a small future vision of their business.

Therefore, today we decided in our project to design an application that will help a small hardware store business. The goal would be facilitating storing and accessing items tasks. The owner could later buy and use advanced devices and gadgets at the checkout, so that items will be scanned and automatically updated on the hardware store database.

b) Client's information:

Client Name: Mr. Abdesselam Rachda.

Hardware Store name: "Aaouinati"

Address: Saada Lotissement 69, Hay Caharf, 40100.

City: Marrakech.

Contact number: (+212) 6 35 73 88 94

c) Project title and team members:

For this project, our team decided to choose the following title: ***WeHelpYou***, and you can find in the table below our team information:

Full Name	Major	Academic Status	E-mail
Ziad El Ismaili	Computer Science	Junior	Z.Elismaili@aui.ma
Soufiane Tounsi	Computer Science	Junior	So.Tounsi@aui.ma
Salma Louhaychi	Computer Science	Junior	S.Louhaychi@aui.ma
Hamza Zaher	Computer Science	Junior	H.Zaher@aui.ma

II. Requirements Gathering:

a) Client's Hardware Store Current System:

For the moment, small business such as shops, retail stores and especially hardware stores, are used to operating on paper using physical file systems to manage their inventory, keep track of sales and doing the necessary calculations to assess their financial growth. In addition, there is no platform that allows customers to keep up with the price, location, or even name of these small shops online. Small business in Morocco, in particular hardware stores, are in desperate need of digitalization of the data they work with.

Therefore, in our case, our client's store is on desperate need to a digitalized platform and a database instead of a physical file, in order to manage well its inventory and make remarkable profit.

b) Importance of a digitalized hardware store:

Today, many independent electricians, plumbers or builders in Morocco pay for their own equipment during a job, which they include in the client bill at the end. Since there is a little number of big surface stores such as "Bricoma" and Ikea that offer off the shelf equipment, the workers are obliged to go to different hardware stores, where they may or may not find the specific parts they are looking for.

Thus, part of our project will focus on providing an online platform to check the availability of products in the hardware stores near you.

C) Objectives and functionalities:

Because of the current problems mentioned above that small non-digitalized hardware stores face, we decided in our project to help our client overcome them, and that is mainly by achieving the following objectives below:

- Create a database that includes information about the hardware store items, their description and price along with any other data appearing in the physical file.
- Use an appropriate DBMS to facilitate operations such as data retrieval and calculations using aggregate functions.
- Data availability, meaning the system should guarantee easy access the data (items), in a meaningful format.
- Provide an online platform in order to check the availability of products.
- Avoid data redundancy and respect all the database criteria. (Data integrity, security, independence…)
- Minimize data inconsistency.

In fact, implementing the objectives above would certainly:

- Overcome almost all the physical file system problems (current system), which will save the client time and resources.
- Facilitate access to our client's hardware store, whose locations are mostly known to the residents of that specific area.
- Centralize all information regarding the location, availability, and prices of products that the hardware store has to offer.

III. Requirements Specification:

a) Functional Requirements:

Concerning the functional requirements part of our project and after talking with our client, we agreed and decided first that functions are divided into 2 categories: ***Admin requirements*** and ***User requirements.***

Admin requirements: Only the manager (which is our client) can access and use the admin functions, such as Manage Items or Update Items…

User requirements*:* functions that could be used by a cashier at the checkout for instance, to compute the total, generate ticket…

N.B: Please note that an admin could also access user functions and requirements; therefore, the manager can access any requirement or functions he wishes.

- ***Manage Items: (As an admin)***

  → *Consisting of SCRUD functions that represent sub-processes such as add_item, remove_item, search_item…*

- ***Update Items: (As an admin)***

  → *The client / manager could easily update the price, quantity, or description of the items, through sub-functions such as edit_item…*

- ***View Items: (As a user & admin)***

  → *Display all items or some of them in tables depending on a given criteria / condition.*

In a detailed view of our functional requirements, we would design our database requirements as the following:

1. Manage Items

1.1 add_item,

Input: item_code, item_name, description, quantity, and all other attributes.

Output: item along with all its attribute added to the database.

Logic used: inserting a row in the table of items existing in the database using SQL

1.2 remove_item,

Input: item_name or item_code (PK) inputed.

Output: item removed to the database.

Logic used: removing a row in the table of items existing in the database, using SQL.

2. Update Items

2.1 update_name,

Input: new item_name entred.

Output: item_name updated in the item table.

Logic used: updating the item_name in the item table using SQL.

2.2 update_quantity,

Input: new quantity entred manually.

Output: item_quantity updated in the item table.

Logic used: updating the item_quantity in the item table using SQL.

2.3 update_description,

Input: new description entered.

Output: description updated in the item table.

Logic used: updating the description in the item table using SQL.

N.B: The item_code cannot not be updated since it is a primary key in the item table and every item has a unique code.

3. View Items

1.1 display_all_items

Input: no input.

Output: table of items.

Logic used: displaying all rows in the table of items existing in the database, using SQL.

1.2 search_item,

Input: item name or code (PK) inputted, or any other attribute the admin would want, under a certain condition.

Output: item / items displayed to the user in a format of tables

Logic used: displaying specific rows in the table of items existing in the database using SQL.

1.3 check_availability,

Input: item name or code (PK) inputted, or any other attribute the admin would want, under a certain condition.

Output: item_quantity displayed in one row intersecting with a column.

Logic used: displaying specific item_quantity of a chosen item database using SQL.

b) Non-Functional Requirements:

For this part, after a long discussion with of our team members we can agreed that certain non-functional requirements would be:

- The Client insist on having a **Reliable Database System.**

- Use an easy **English Graphical Interface** for the users and the client to work with. (After a working English graphical interface, our team could easily translate it to a French / Arabic one).
- Managing time and **respect all the deadlines** provided by the instructor, or even the client.
- **Meet virtually with the client** whenever necessitated and communicate to him the progress of our work.

Moreover, we could think of other ethical and legal non-functional requirements such as:

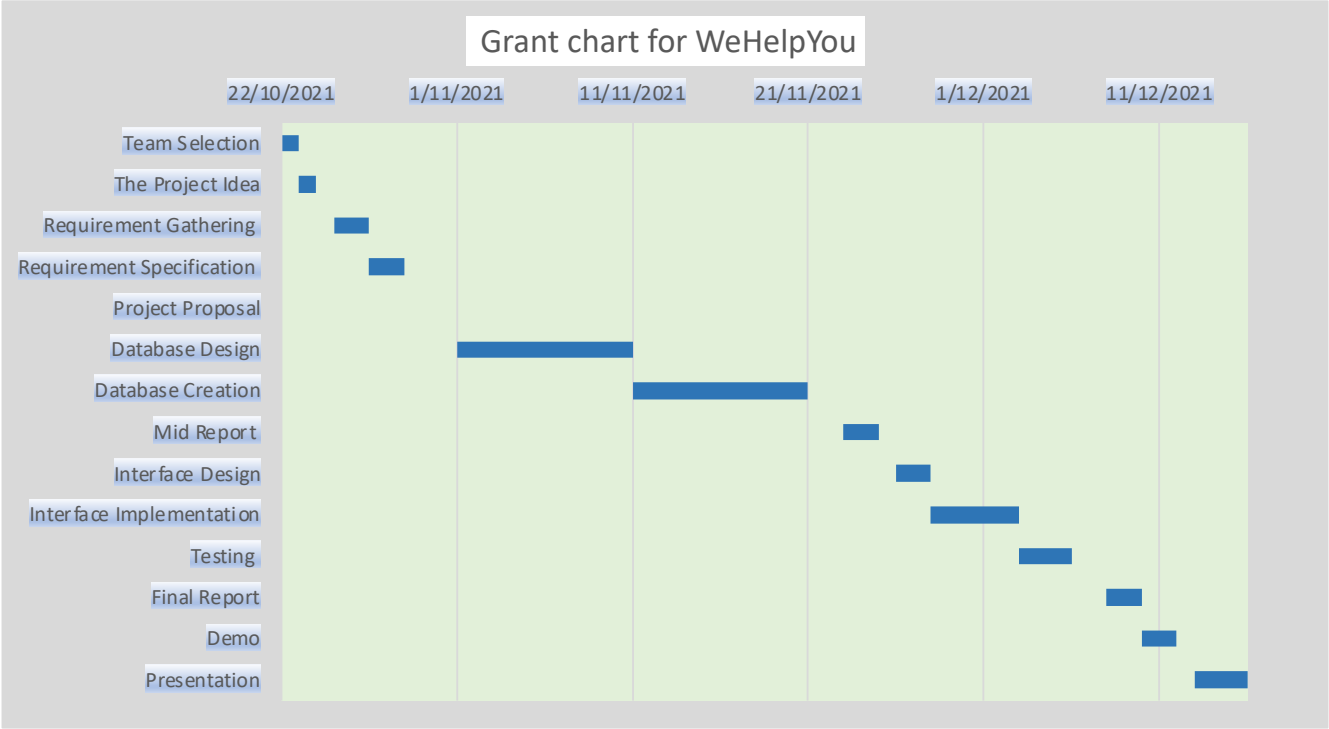
- The system should keep track of the contents entered, making sure that no misleading, unethical or immoral contents is inserted in the database system. Such behavior should not be tolerated.
- Information and data should be secure, protected and preserved within the system. The digitalized system should not share any restricted information unless authorized.

IV. Project Management Plan:

a) Time Management:

To efficiently accomplish this project, a follow up tasks must be attained over time. Those steps are demonstrated each in the table and the Gantt Chart below:

No.Task	Start Date	End Date	Task	Duration in d	Priority	Status	Done By:
1	22/10/2021	23/10/2021	Team Selection	1	High	Done	Z
2	23/10/2021	24/10/2021	The Project Idea	1	High	Done	H
3	25/10/2021	27/10/2021	Requirement Gathering	2	Medium	Done	Z- H- Sa- So
4	27/10/2021	29/10/2021	Requirement Specification	2	Medium	Done	Z- H- Sa- So
5	29/10/2021	29/10/2021	Project Proposal	0	High	Done	Z
6	1/11/2021	11/11/2021	Database Design	10	High	Done	Z-So
7	11/11/2021	21/11/2021	Database Creation	10	Medium	Done	So-H-Sa
8	23/11/2021	30/11/2021	Mid Report	7	Low	Done	Z- H- Sa- So
9	1/12/2021	1/12/2021	Interface Design	0	High	In Progress	Z-H
10	1/12/2021	5/12/2021	Interface Implementation	4	High	To Do	To be determined
11	5/12/2021	9/12/2021	Testing	4	High	To Do	To be determined
12	9/12/2021	11/12/2021	Final Report	2	Medium	To Do	To be determined
13	11/12/2021	13/12/2021	Demo	2	High	To Do	To be determined
14	14/1/2022	16/12/2021	Presentation	(29)	Medium	In Progress	Sa-So
H: Hamza Zaher - Z: Ziad El Ismaili - So: Soufiane Tounsi - Sa: Salma Louhaychi							



b) Task distribution:

Concerning the tasks, we agreed that we will try as much as possible to get each member interacting with other's tasks, in order to improve the quality of our final project. The tasks are divided between us as shown in table above..

V. Entity Relationship Diagram:

a) Discussing and defining entities:

Before creating our project's ER model, we need first to highlight and discuss the entities, which some were previously mentioned in the requirement specification part. In short, the entities are the following:

- ***user**, is **manager** or **cashier***
- ***log_register***
- ***item***
- ***supplier***

- ***aouinati_store***
- ***customer***

User entity should be divided to 2 entities, meaning in a database design perspective, we can have the user entity as a superclass, including only shared attributes between **admin** and **cashier**, that are the 2 possible sub-classes of it. **User** entity should contain the following attributes:

USER entity attributes:

- user_ID *int, PK*
- user_password *varchar (20)*
- user_fname *varchar (20)*
- user_lname *varchar (20)*
- user_email *varchar (40)*
- *user_phone varchar (15)*

Consequently, we need to assign different attribute for both user entity sub-classes:

MANAGER entity attributes:

- user_ID *int, PK*

CASHIER entity attributes:

- user_ID *int, PK*
- salary *int*

Moreover, the other main difference between the 2 sub-classes might be the functions that each entity could perform, for instance, admin can update_item(), while the cashier has only few limited functions.

Also, we need a 1-to-1 relationship between **cashier** entity and **log_register** entity, where it contains the different attribute necessary to keep track of the cashier working day. Please keep

in mind that since it is a 1-to-1 relationship, we had the choice to simply add the **log_register** attributes to the cashier entity; however, this is not optimal because the **log_register** attributes (working hours, in time, out time…) are subject to change every day, and it would be better if we had a different entity that handles the work of a cashier.

LOG_REGISTER entity attributes:

- user_ID *int, PK*
- *in_time varchar (40)*
- *out_time varchar (40)*
- working_hours *int*
- _date *date*

One of the important entities is certainly the **item** entity, where it contains all information concerning the products available in the store, as well as products **manager** wishes to add to the store.

ITEM entity attributes:

- item_code *int, PK*
- description varchar (40)
- in_date *date*
- qoh *int*
- price *decimal (6.2)*
- discount_rate *decimal (3.2)*

Now, according to the client, each item has a **supplier**, and some might not have one (i.e few ones made by aouinati store). Therefore, we need a **supplier** entity linked to the **item** entity, where we can find information concerning the supplier of any product in the store. (Foreign key supplier_ID is assigned and displayed in the ERD in the next part..)

SUPPLIER entity attributes:

- supplier_ID *int, PK*

- supplier_name *varchar (40)*
- supplier_phone *varchar (15)*
- supplier_address *varchar (40)*
- supplier_country *varchar (10)*
- supplier_city v*archar (10)*

Furthermore, the manager / client insisted on having a discount for some **customer**, through a loyalty card that contains information about each **customer** such as name, address and more importantly the discount.

CUSTOMER entity attributes:

- cus_ID *int, PK*
- cus_fname *varchar (40)*
- cus_lname *varchar (40)*
- cus_address *varchar (40)*
- *cus_city varchar (40)*
- cus_discount *decimal (3,2)*

Item is found in **aouinati_store, customer** is registred in aouinati_store… In fact, another highly important entity is **aouinati_store,** where items are stored. The entity contains also information about the store, since one of the main objectives of this project is to automate the business of the hardware store and make it available to customer on the internet. Therefore, the **aouinati_store** entity should include public information such as address, city, country, phone and a name as a primary key.

AOUINATI_STORE entity attributes:

- store_name *varchar (40), PK*
- store_address *varchar (40)*
- store_country *varchar (15)*
- store_city *varchar (15)*
- *store_phone varchar (15)*

Lastly, in order for a customer to buy items, an invoice should be generated, with the following attributes:

INVOICE entity attributes:

- invoice_num *int, PK*
- invoice_date *date*

N.B: Foreign keys, relationships, constraints, and other features will be displayed in the next part when designing the ER model.

b) Business rules and relationships:

Some of the business rules were already stated above when describing the use of entities; however, in this part we will highlight and focus more on business rules that will be converted to relationships, following the client demands:

Relationships:

- ***user is manager:*** "a user can be the manager".
- ***user is cashier:*** "a user can be a cashier".
- ***cashier has log_register:*** "one cashier has only many log registers, one for each day but a log_register contains information about only one cashier".

  → ***cashier 1:M log_register***

- ***cashier works in aouinati_store:*** "one cashier works in aouinati_store but aouinati_store can have multiple cashiers work inside as cashiers".

  → ***aounati_store*** *1:M* ***cashier***

- ***supplier supplies item:*** "a supplier supplies many items, but each item is supplied by only one supplier"

→ ***supplier*** *1:M* ***item***

- ***aouinati_store sells item:*** "the store sells many items, but each item is only sold in only one store which is the AOUINATI store"

  → ***aouinati_store*** *1:M* ***item***

N.B: You might find difficulties with understanding the 2 previous relationships, thinking that the relationship is many-to-many instead of one-to-many. However, keep in mind that the AOUINATI_STORE is one single entity and not many stores. We decided to create an entity called AOUINATI_STORE only because when interviewing the client, he was interested in posting store information on the internet. Therefore, creating and populating a store entity might be a great idea.

- ***customer is_registred in aouinati_store:*** "the store has information about many customers, but each customer is registered in only one store which is AOUINATI store"

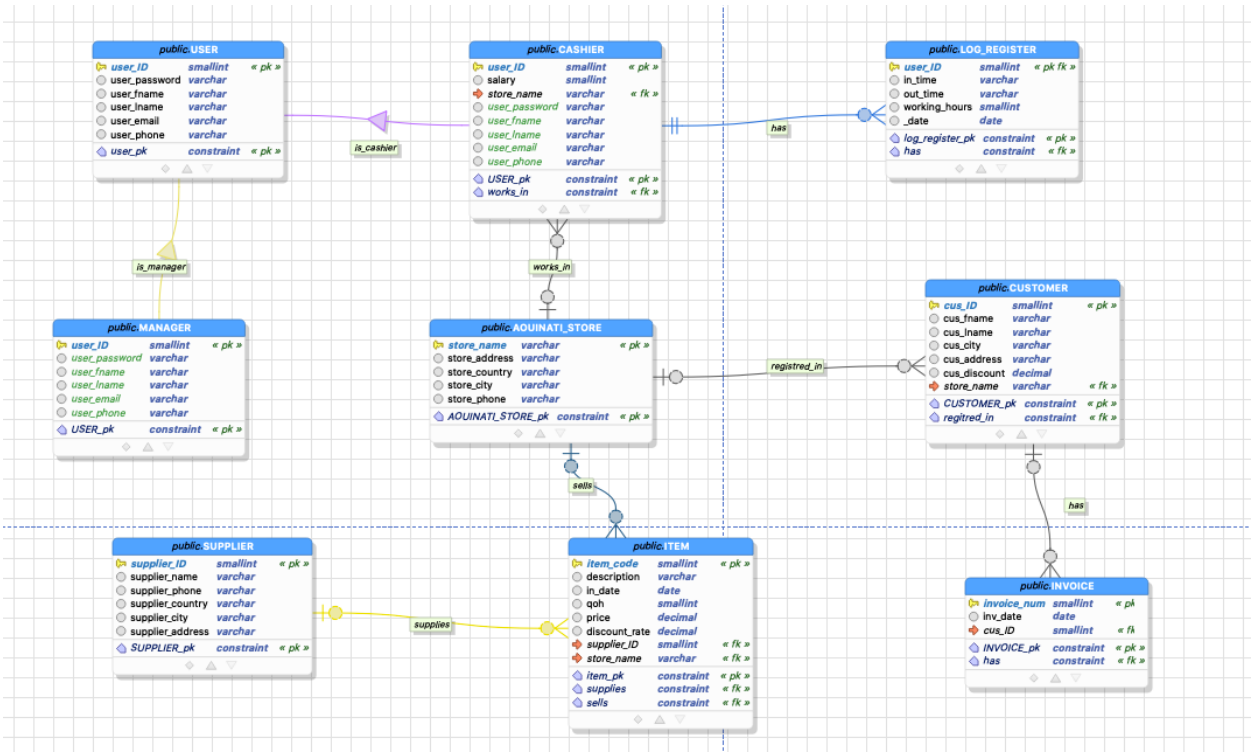
  → ***aounati_store*** *1:M* ***customer***

- ***customer has many invoices:*** "a customer could have many invoices while a single invoice is issued by only one customer"

  → ***cutomer 1:M invoice***

C) Entity-Relationship Diagram:

According to the business rules and all of the previous above relationships, attributes and constraints, this is what we consider our ER diagram:


public.USER
user_ID smallint « pk »
user_password varchar
user_fname varchar
user_lname varchar
user_email varchar
user_phone varchar
user_pk constraint « pk »
public.CASHIER
user_ID smallint « pk »
salary smallint
store_name varchar « fk »
user_password varchar
user_fname varchar
user_lname varchar
user_email varchar
user_phone varchar
USER_pk constraint « pk »
works_in constraint « fk »
public.LOG_REGISTER
user_ID smallint « pk fk »
in_time varchar
out_time varchar
working_hours smallint
_date date
log_register_pk constraint « pk »
has constraint « fk »
is_cashier
has
is_manager
works_in
public.MANAGER
user_ID smallint « pk »
user_password varchar
user_fname varchar
user_lname varchar
user_email varchar
user_phone varchar
USER_pk constraint « pk »
public.AQUINATI_STORE
store_name varchar « pk »
store_address varchar
store_country varchar
store_city varchar
store_phone varchar
AQUINATI_STORE_pk constraint « pk »
registred_in
public.CUSTOMER
cus_ID smallint « pk »
cus_fname varchar
cus_lname varchar
cus_city varchar
cus_address varchar
cus_discount decimal
store_name varchar « fk »
CUSTOMER_pk constraint « pk »
regitred_in constraint « fk »
sells
has
public.SUPPLIER
supplier_ID smallint « pk »
supplier_name varchar
supplier_phone varchar
supplier_country varchar
supplier_city varchar
supplier_address varchar
SUPPLIER_pk constraint « pk »
supplies
public.ITEM
item_code smallint « pk »
description varchar
in_date date
qoh smallint
price decimal
discount_rate decimal
supplier_ID smallint « fk »
store_name varchar « fk »
item_pk constraint « pk »
supplies constraint « fk »
sells constraint « fk »
public.INVOICE
invoice_num smallint « pk
inv_date date
cus_ID smallint « fk
INVOICE_pk constraint « pk »
has constraint « fk »