1 Introduction

1.1 Purpose of this document

This is a system for oriental weavers (not official) that covers and facilitate financials, transactions, Bills, Accounts, Carpets, Orders and CRM processes that happen in this organization. To achieve more income and reduce time by recording all the details of the customer that occurs daily within the organization and by making a well-managed statistics about the customer (the most wanted carpet type, color, etc..) also by recording every single transaction occur in the organization, which will guarantee less time and more efficiency in the work. This document addresses the head of the administrative system and the one responsible for making decisions in the company.

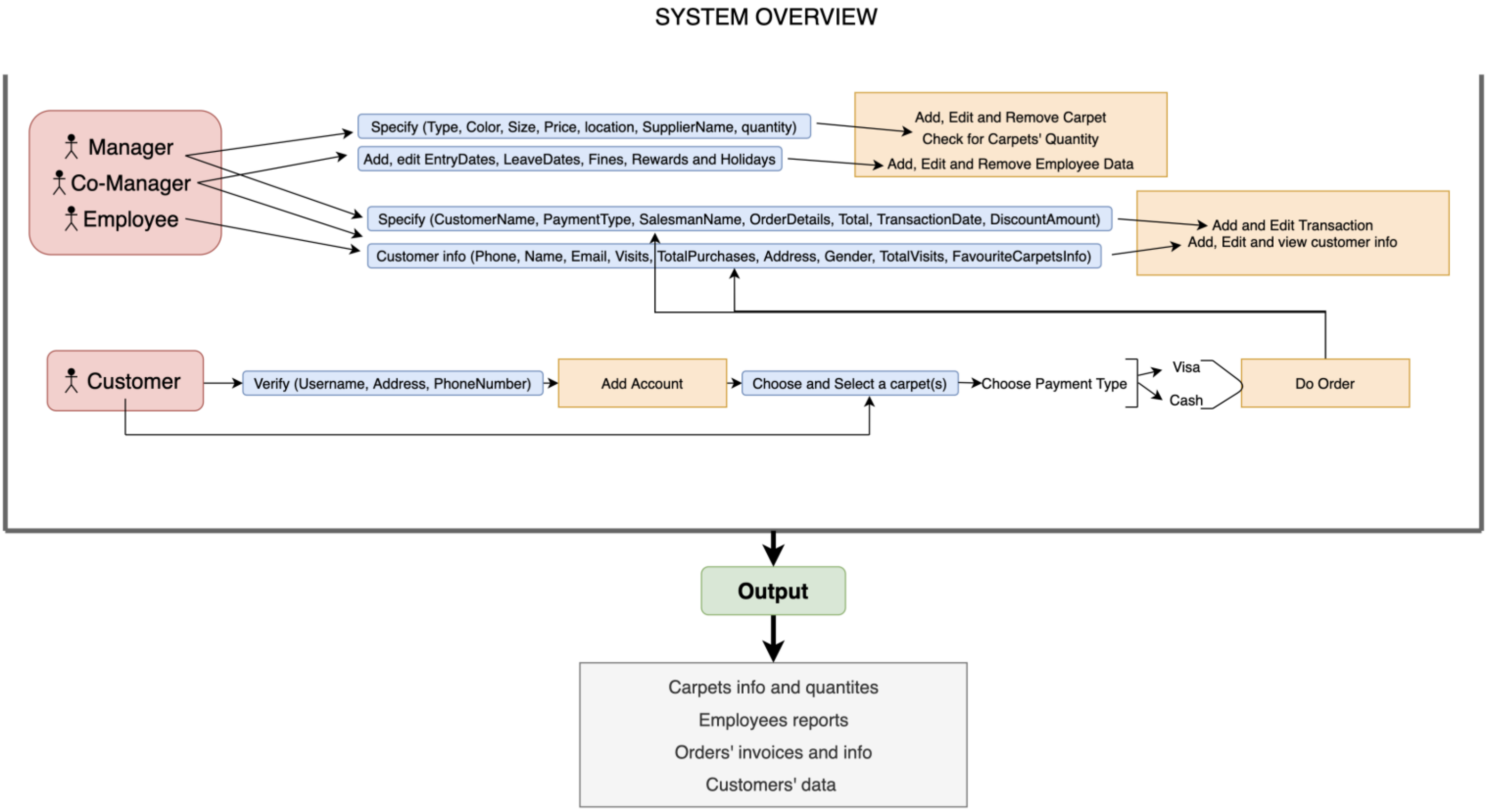
1.2 Scope of this document

The system will focus on the financial transactions, carpets, accounts, orders and CRM processes, because it is considered the main important parts of any commercial system. As for the team we are four members in the team, every two focus on a specific function, in order to be more specific, the roles are divided as follows:

- Project has a delivery date between: **2 - 4 months**

1.3 Overview

As it is shown in the next **Figure (1)**


SYSTEM OVERVIEW
Manager
Co-Manager
Employee
Specify (Type, Color, Size, Price, location, SupplierName, quantity)
Add, edit EntryDates, LeaveDates, Fines, Rewards and Holidays
Add, Edit and Remove Carpet
Check for Carpets' Quantity
Add, Edit and Remove Employee Data
Specify (CustomerName, PaymentType, SalesmanName, OrderDetails, Total, TransactionDate, DiscountAmount)
Customer info (Phone, Name, Email, Visits, TotalPurchases, Address, Gender, TotalVisits, FavouriteCarpetsInfo)
Add and Edit Transaction
Add, Edit and view customer info
Customer
Verify (Username, Address, PhoneNumber)
Add Account
Choose and Select a carpet(s)
Choose Payment Type
Visa
Cash
Do Order
Output
Carpets info and quantites
Employees reports
Orders' invoices and info
Customers' data

1.4 Business Context

The main target was to provide to the customer less time consumption. To be clearer, more profit with less effort. In order to achieve all of this, the idea was to create a system which records every single transaction that occurs within the organization for example: if the customer ordered a carpet, the system will record the type of the carpet, the color, the size, to create a specific report about the purchases, so after that, the manager, co-manager and employees could do a study for the report in order to see what is the **best-selling** item in the branch and what is the item that causes him losses so he can increase the production for the best-selling item and decrease the production for the item that cause losses. Also, the reports that we will collect from the customers' data will statistically help identifying the most area that customers from, and this is important in making offers and sending messages to customers rather randomizing. The system will also record all the details about the employees themselves such as his behavior (entry date and leave date) , also number of holidays he took or if he would be fined for some reason , so these things will be saved in order to find for the specific customer the perfect employee and that will also save time.

2 General Description

2.1 Product Functions

The system mainly can be classified into

Administrative system of the branch:

1- Transactions: in order to record every transaction in the system in order to prevent any human order to happen.

2- Accounts: the system provide to both the employees and the client accounts in order to make them do their functions easier.

3- CRM: It will help the company in order to create a well-managed report about the profits and the loss ,Also know what kind is the best-selling carpet.

Regulatory system of the branch:

1- Manage carpets: every single detail about the types, numbers, the size and the color of the carpet will be recorded in the system, also check if it is available or not

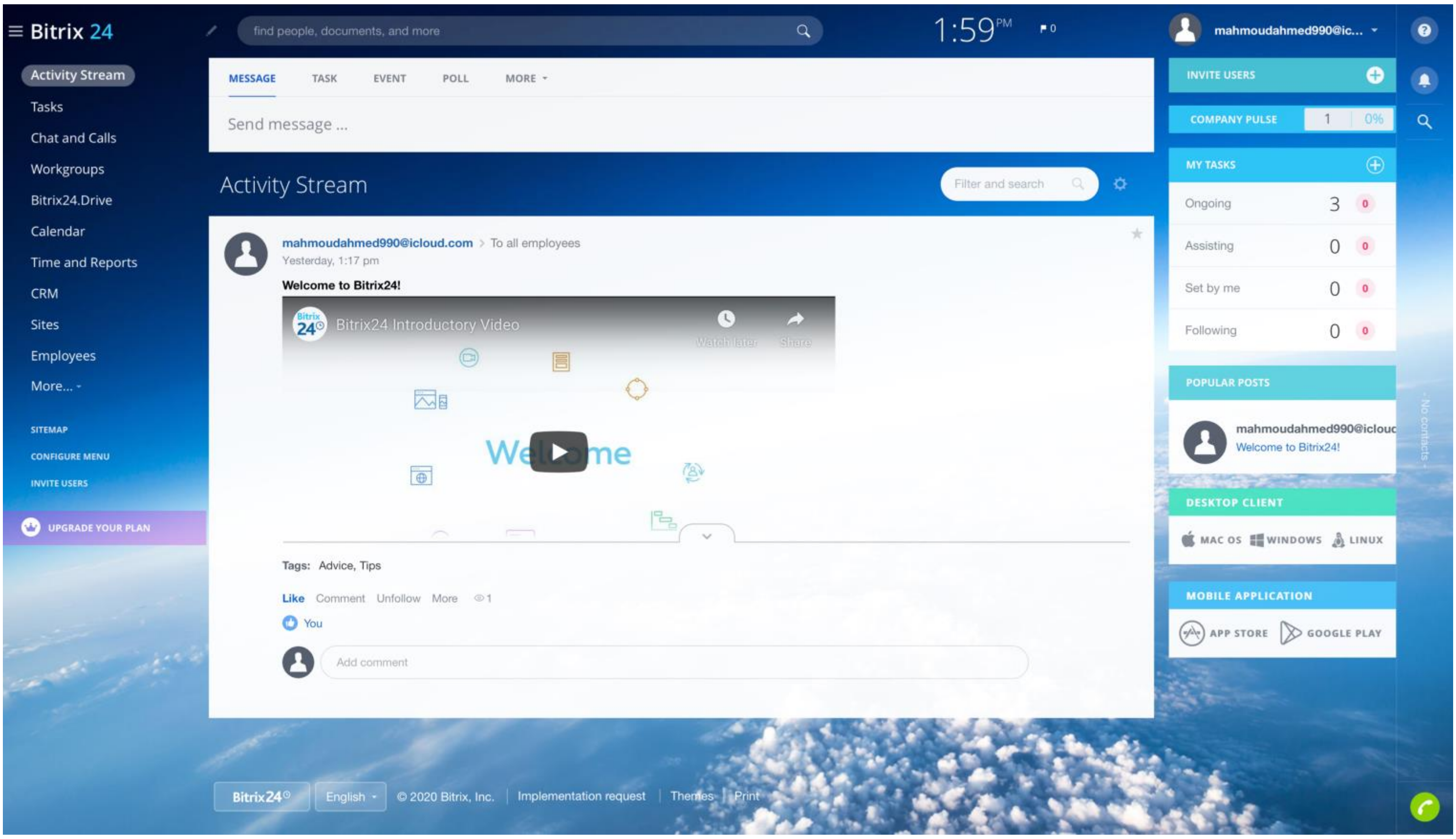
2- Orders: This will facilitate the purchase process where customers will have the ability to buy any carpet from the website and chose the method of payment or even to purchase from the branch offline.

2.2 Similar System Information

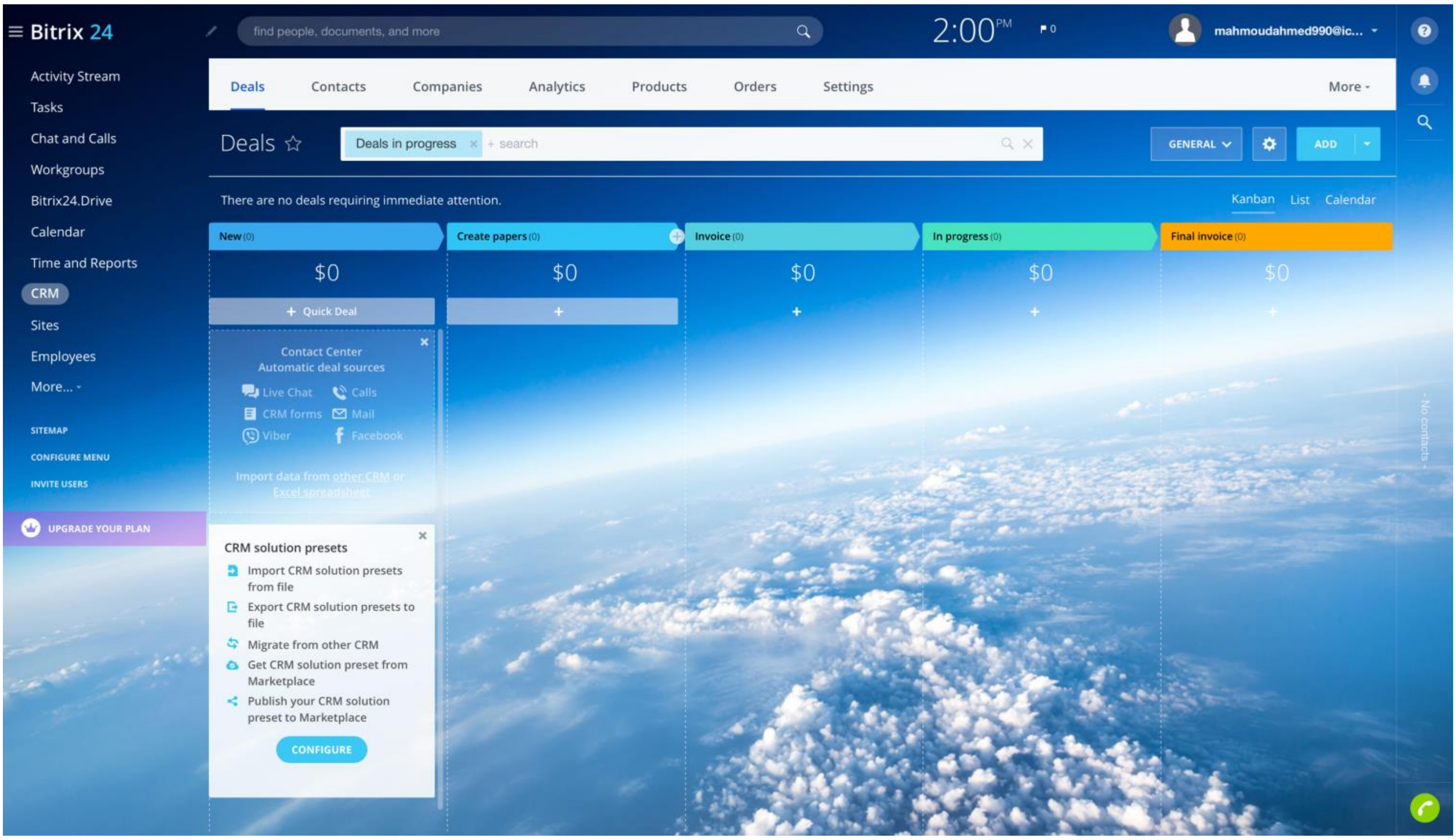
N.B: There is no Carpets or Weavers Management Systems found, so we have searched for Sales Management System instead.

1- Bitrix24

- This system has the capability to create papers, chats, calls and create tasks that facilitate the organization functions, also, a customer relationship management marketing that can manage all the relations with customers, the only problem they had that they don't record all the transactions occur in the system.



mahmoudahmed990@icloud.com > To all employees
Yesterday, 1:17 pm
Welcome to Bitrix24!



There are no deals requiring immediate attention.

2- LevelEleven

- This system is focusing on the design and the reports for each KPI metrics for their team members; also, they have opportunities, accounts, dashboard, contacts functions. All of these with a sample design and amazing colors. The only gap they have in the system is they don't have a CRM system.

- Our system will be different of these systems because it provides to the company a full well-managed customer relationship management system wish is not provided in Level eleven system.

- Also, our system will record the daily transaction that occur within the company wish is not provided in Bitrix24 system.

Systems	Time	Reports	Transactions	CRM
Bitrix24	No	Yes	No	Yes
LevelEleven	No	Yes	Yes	No
Our System	Yes	Yes	Yes	Yes

2.3 User Characteristics

The user community can be classified mainly into 4 categories:

Normal Employee: This type of users has the ability to handle simple operations in the company so mainly he will do only the following functions: add or remove transaction, check the type of carpet the user prefers. Therefore, it does not need experience in dealing with the system and does not need a specific gender

Manager: The Manager can handle large operations in the company such as: add, edit and remove carpets, add and edit transactions, add and edit employee info, also, add, edit and view customers characteristics and history with the company and his purchases. Therefore, it need experience in dealing with the system.

Co-Manager: Which can handle the same operations of the Manager but he is responsible to verify the bills.

Customer: This is a normal type of user who deals with the routine procedures of any company such as: register, login, do an order, select carpet and pay. Only need to know how to deal with methods of verifying identity and submitting info, and also electronic purchases methods.

2.4 User Problem Statement

The problems will be divided according to the user classification in the previous point:

Normal Employee:_ The biggest problem was working with a large amount of paper in order to record a transaction with a customer in addition to the difficulty of knowing what the customer wants because of the lack of knowledge of the previous purchases of the client.

Manager:_The Manager's problems is how to handle and search in a paper if the amount is large as much, also there is a lack of knowledge about their customers.

CO-Manager : The Co-Manager also has the same problem of the normal employee wish was working with a large amount of paper in order to record a transaction with a customer or within the organization but this time the problem is bigger because the co manger is also the supervisor of the normal employee so the amount of data is bigger.

Customer: The customer problem differs than the previous problem, the problem was that the customer finds it difficult to find his product, unless he goes to the branch himself.

2.5 User Objectives

The user objectives can be classified into two groups:

Client objective: The customer will be able to order the product that suits him easily without going to the branch , and in case of going to the branch, the system will provide the purchase history of the customer and the product he prefers, which will save the time of the customer and the employee who deal with him.

Company objective: The system will do reduce much time and costs related to this organization. For example: This system will help reduce human error and reduce the amount of paperwork, and also will help in setting and determining each customer and the type of carpet he prefers, and thus, the ease of sending marketing messages via mobile to specific customers, depending on the type of carpet they prefer, and also, to a specific location depending on the customer's address they submitted.

2.6 General Constraints

The system does not require much requirements to run it, but there are technical problems that the user may face ,we will talk about it from two perspectives:

The client: The customer may face some problems while logging in or paying for the purchases as a result of pressure on the system or problems in the network.

The company staff: The staff may face some difficulties during the registration, recording of daily transactions, in the case of adding a new customer or adding a new type of carpet causing a pressure on the system or weakness in the network.

3 Functional Requirements

Function name (3.1)	AddnewTransaction(Bill B)
Description	1- Check with an if statement if the employee has an account on the system and then return true 2- Save the transaction on the system by inserting it in the database 3- After doing the previous step check if the records are saved in the database or not by displaying a message Box in the insertion to make sure.
critically	This function will help in order to create a well-managed report that that company might use.
Input	Bill info (Order obj, Bill Date, Net Amount, discounts)
Output	-Stored transactions in the database -Bills
Technical issues	- Error during submitting the transaction fields, like leave an empty field. - Error during recording transactions, due to Pressure on the system at the same time. - Database connection normal problems.
Risks	- System Failures: The result of pressure on the system due to the large number of daily transactions for the company, the company must then have a fast backup system for the system to work until the original system returns to work.
Cost and schedule	In order that the oriental weavers will use the system, they will need a large server to hold these many transactions, also, submitting a transaction does not take more than 5 minutes.
Dependencies with other requirements	This function can't happen without that the employee has an account (Function 3.2)
Pre-Condition	-The database is connected truly
Post-Condition	-The company transaction is recorded in both the system and the database.

Function name (3.2)	CreateAccount()
Description	Take the info of the user that is registering and insert it in the database with respect to the mandatory and optional fields
Input	- Client information's (first name, last name, username, password, address, phone, gender, email, last visit, total paid, salesman name, favorite carpets) - Employee information's (name, address, phone, job Type, password)
output	- Account on the system in order to do any function
critically	This function is the most important one, without it the user can't access to the system.
Technical issues	- Error during insertion - Repeated username, phone number, email
Risks	- Disconnection from the system during registrations
Cost and schedule	- The create account function doesn't require any cost, it would take only minutes.
Dependencies with other requirements	It does not depend on other function
Pre-Condition	-The database is empty -The user can't use the system
Post-Condition	The worker user will be able to manage and use the functions of the system.

Function name (3.3)	SearchCarpet(Carpets C)
Description	- loop over a number of carpets and then search by an if condition in the variables inside this C object, for example: if(C.getcarpetcolor()==”blue”) {return C;} - If the quantity of any carpet == 0, then change the Boolean variable isExist from true to false and then display messageBox to the user and worker that this carpet is not exist;
Input	- Carpet characteristic
output	- Carpet info, quantity, characteristic(s)
critically	This function will make the employee always updated with the carpet information and can find a specific one easily if a customer asked for.
Technical issues	- Database connection problem
Risks	Improper Use of Data and inserting
Cost and schedule	This function doesn’t need a cost and doesn’t need much time to proceed, depending on the quantity of carpets.
Dependencies with other requirements	This function can’t happen without that the employee has an account (Function 3.2)
Pre-Condition	-The warehouse was empty or fully loaded that caused a fatal error
Post-Condition	Carpet has been added to the warehouse and the quantities has increased

Function name (3.4)	DoOrder(Order O)
Description	- Start new carpet object and save its declarations - Selection of payment methods will be saved as string - Add this transaction by its details to the database - Display the bill
Input	- Credit card Number
output	- Bill - The selected carpet.
critically	This will facilitate the purchase process where customers will have the ability to buy any carpet from the website directly.
Technical issues	- Error during the login process - Error during the purchase process
Risks	In case of online purchase, the system may face hacking attempt.
Cost and schedule	No cost for this function, only the cost will be paid from the customer when he buys.
Dependencies with other requirements	The client can't order online without having an account on the system (Function 3.2) and the system must check the quantity of the carpet first and this carpet is exist or not (Function 3.3)
Pre-Condition	The shopping cart was empty
Post-Condition	The item has been added to the shopping cart

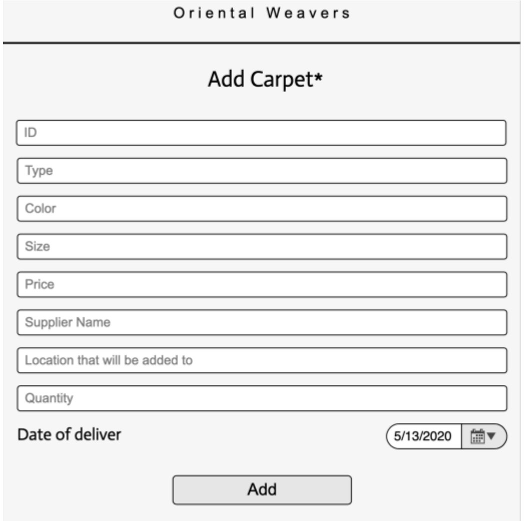
Function name (3.5)	AddnewCustomer(Customer C)
Description	- Pop up a form that contains text fields to be filled with the customers' info that the workers know about it. - Give a chance of editing or searching for these existing customers' info by inserting the phone number and start searching in the database for this client if he is already exist and then display editable or viewing area.
Input	- Client Info - Worker Login
output	- Customer information
critically	This function will help in order to have a well-managed statistical report about customers and their relation with carpets
Technical issues	- Database connection problem - Phone Number, Email are repeated - Mandatory fields didn't filled
Risks	Improper Use of Data
Cost and schedule	No cost or schedule needed
Dependencies with other requirements	This Function depends on (Function 3.2) as to add, edit or view customers' info, workers have to register, also, in some cases, a transaction info needed to add it to customers' info if they bought anything (Function 3.1)
Pre-Condition	The company wasn't having any statistics about the customers and the Best-selling carpet in the company for example.
Post-Condition	The company has details of the best-selling carpet that uses that information and uses it to make a bigger profit and can use the customers' info in a large scale

Function name (3.6)	AddnewEmployee(Employee E)
Description	Take the Employee object as a parameter and use its variables to declare the employee's info by importing these info through a text fields and then save it to the database related to the employees.
Input	- Employee info (for saving)
output	- Employees info (for checking)
critically	This function will help keep employees' data managed and saved to be easily checked if needed.
Technical issues	No Technical issues may be faced either if there was a lack of database connection so the data wouldn't be added successfully.
Risks	If the manager account got hacked from any employee, he can edit his data.
Cost and schedule	No cost or schedule needed.
Dependencies with other requirements	This function depends on Function (3.2) Because whoever adding or editing employee account need to have an account too.
Pre-Condition	There wasn't a data holder that keeps employee info and save it for a future use if needed in any case
Post-Condition	Employees' data will be saved and can be checked anytime if needed, rather than depending on human memory.

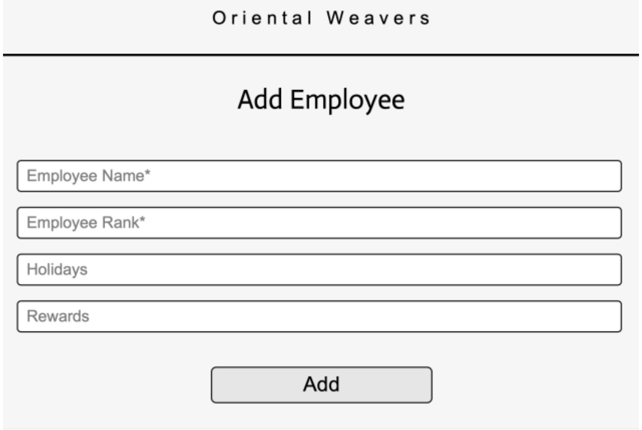
4 Interface Requirements

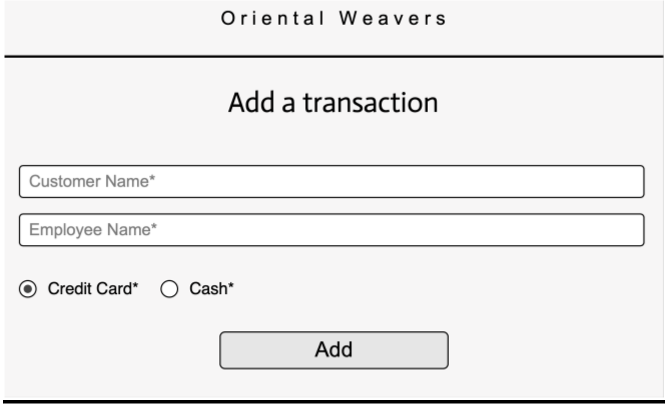
4.1 User Interfaces

4.1.1 GUI

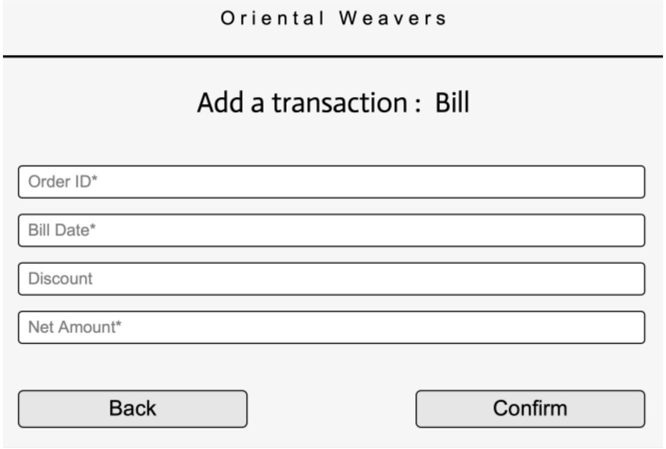


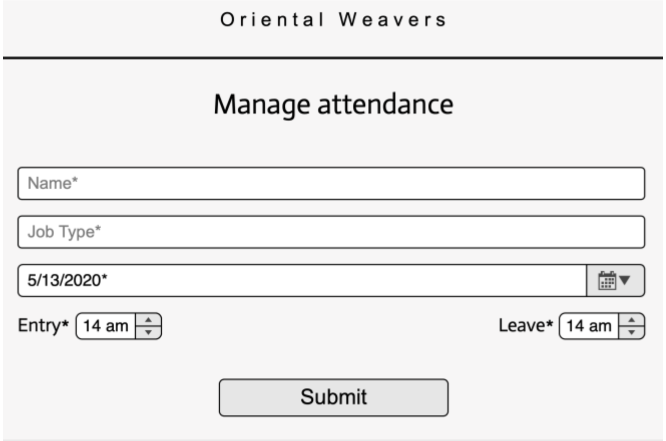
Date of deliver: 5/13/2020





● Credit Card* ○ Cash*

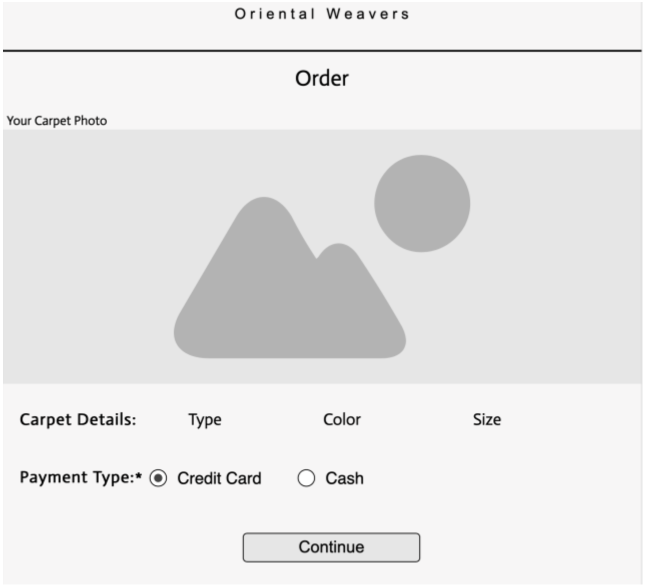




5/13/2020*

Entry* 14 am

Leave* 14 am


Oriental Weavers

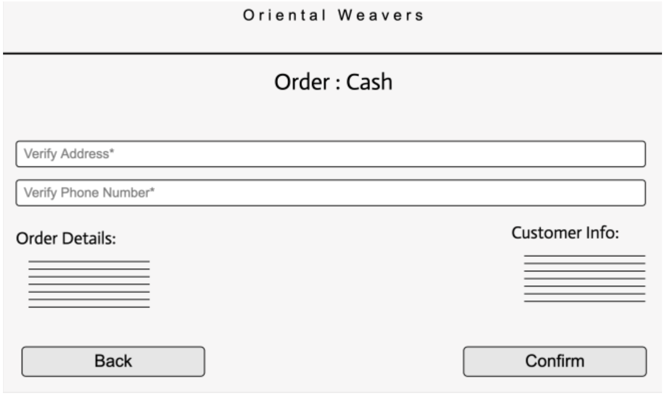
Order

Your Carpet Photo

Carpet Details: Type Color Size

Payment Type:* ● Credit Card ○ Cash

Continue


Oriental Weavers

Order : Cash

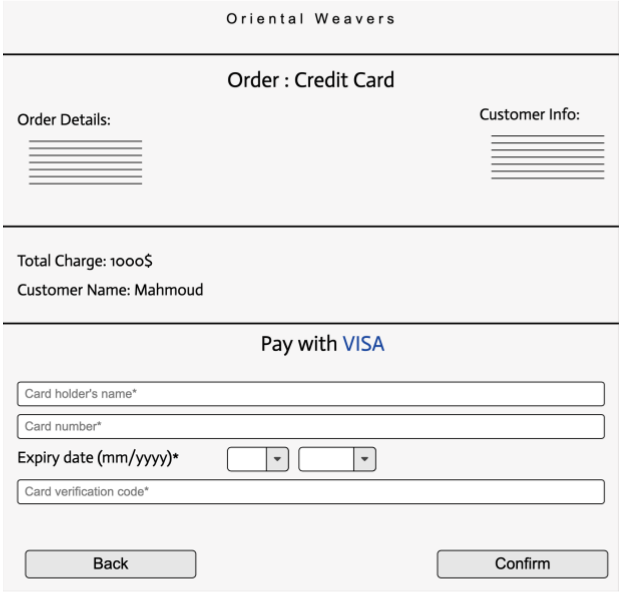
Verify Address*

Verify Phone Number*

Order Details:

Customer Info:

Back Confirm


Oriental Weavers

Order : Credit Card

Order Details:

Customer Info:

Total Charge: 1000$

Customer Name: Mahmoud

Pay with VISA

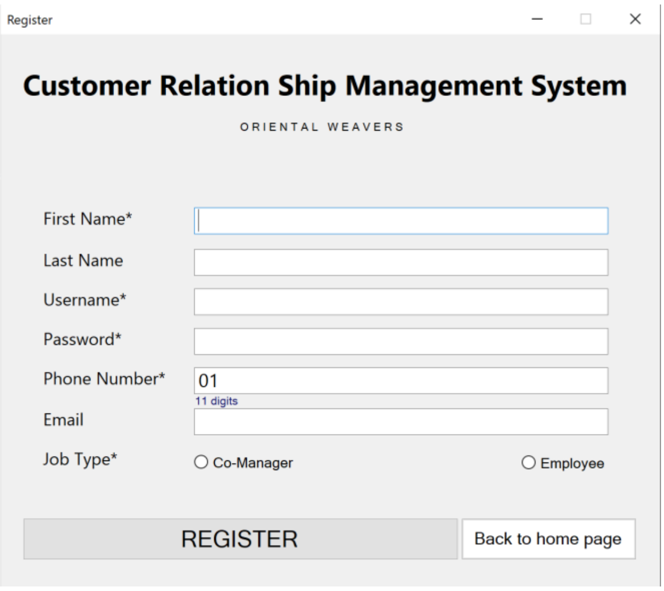
Card holder's name*

Card number*

Expiry date (mm/yyyy)*

Card verification code*

Back Confirm


Register

Customer Relation Ship Management System

ORIENTAL WEAVERS

First Name*

Last Name

Username*

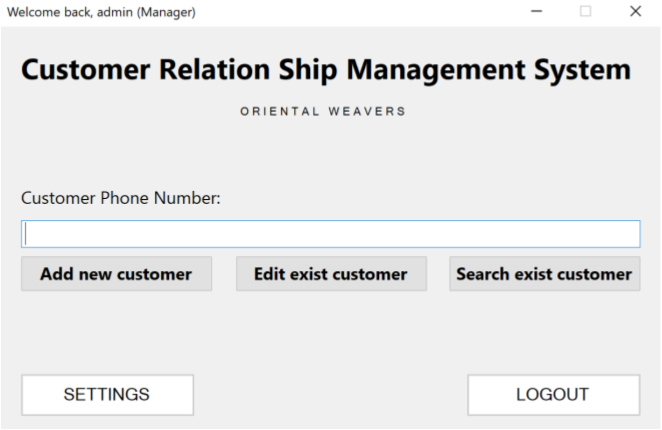
Password*

Phone Number* 01
11 digits

Email

Job Type* ○ Co-Manager ○ Employee

REGISTER Back to home page


Welcome back, admin (Manager)

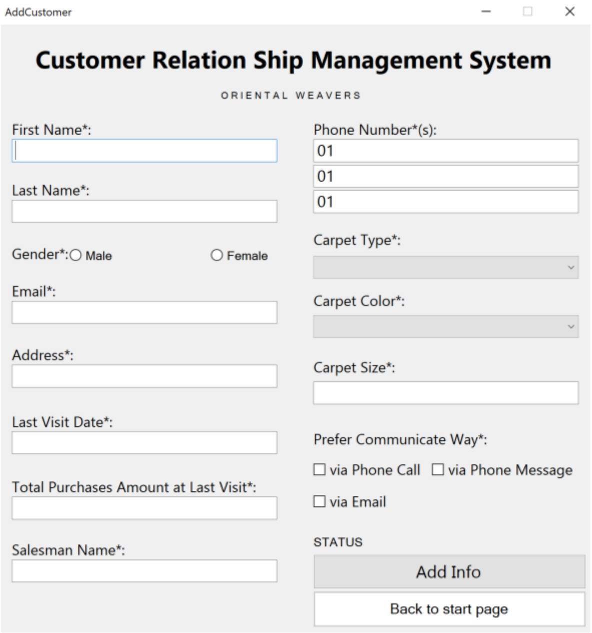
Customer Relation Ship Management System

ORIENTAL WEAVERS

Customer Phone Number:

Add new customer Edit exist customer Search exist customer

SETTINGS LOGOUT


AddCustomer

Customer Relation Ship Management System

ORIENTAL WEAVERS

First Name*:

Last Name*:

Gender*: ○ Male ○ Female

Email*:

Address*:

Last Visit Date*:

Total Purchases Amount at Last Visit*:

Salesman Name*:

Phone Number*(s):
01
01
01

Carpet Type*:

Carpet Color*:

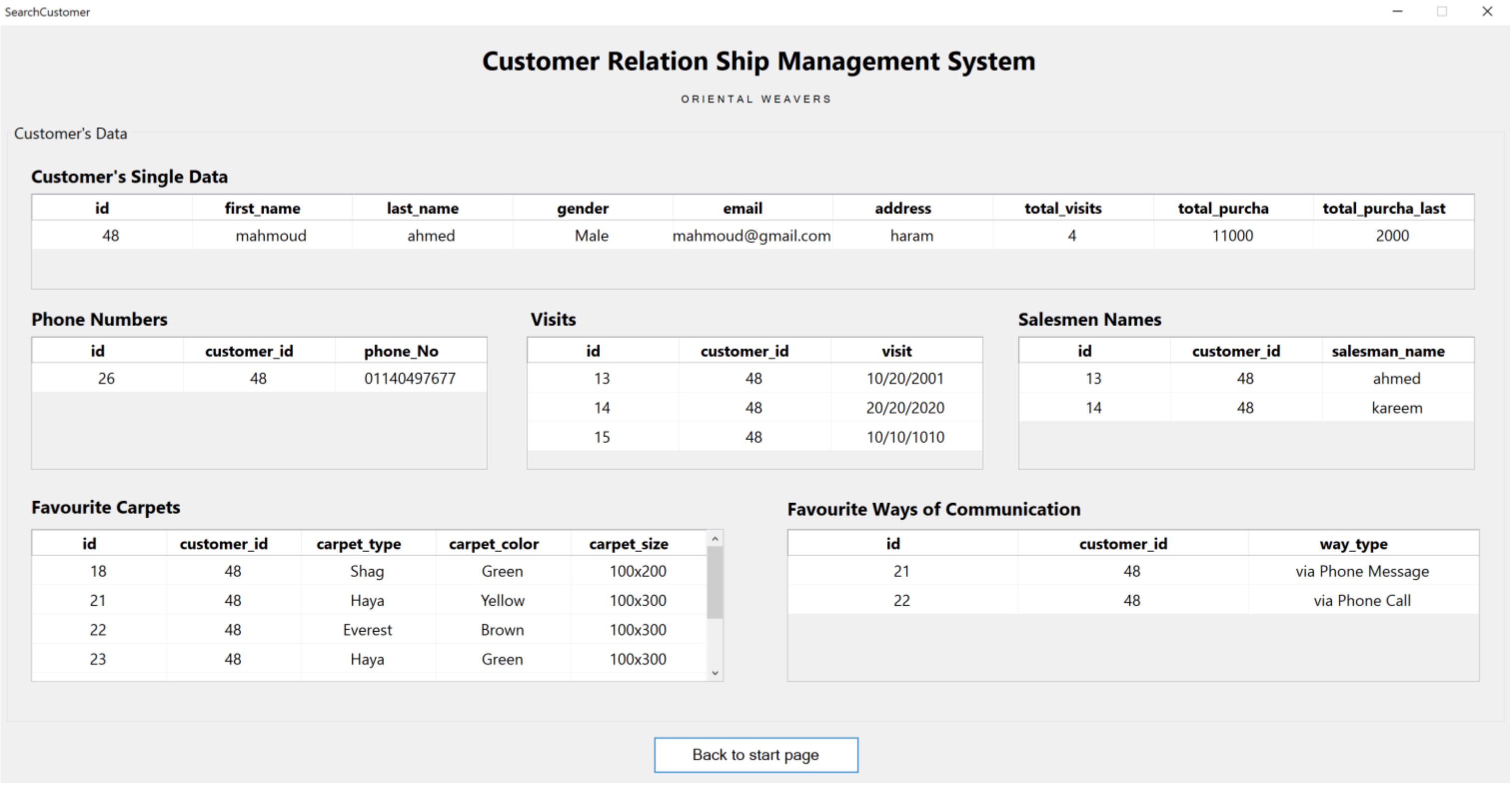
Carpet Size*:

Prefer Communicate Way*:
□ via Phone Call □ via Phone Message
□ via Email

STATUS

Add Info

Back to start page



Customer Relation Ship Management System

ORIENTAL WEAVERS

Customer's Data

Customer's Single Data

id	first_name	last_name	gender	email	address	total_visits	total_purcha	total_purcha_last
48	mahmoud	ahmed	Male	mahmoud@gmail.com	haram	4	11000	2000

Phone Numbers

id	customer_id	phone_No
26	48	01140497677

Visits

id	customer_id	visit
13	48	10/20/2001
14	48	20/20/2020
15	48	10/10/1010

Salesmen Names

id	customer_id	salesman_name
13	48	ahmed
14	48	kareem

Favourite Carpets

id	customer_id	carpet_type	carpet_color	carpet_size
18	48	Shag	Green	100x200
21	48	Haya	Yellow	100x300
22	48	Everest	Brown	100x300
23	48	Haya	Green	100x300

Favourite Ways of Communication

id	customer_id	way_type
21	48	via Phone Message
22	48	via Phone Call

4.1.2 CLI

4.1.3 API

4.1.4 Diagnostics or ROM

We will return the exception code with try and catch method, so, we will return the exception code, and then we will know what to diagnose and debug it.

4.2 Hardware Interfaces

4.3 Communications Interfaces

We will use our local host to start the system, using local host requires to install xampp or wampp application which will open these ports to us (80, 443, 3306).

4.4 Software Interfaces

C#, it was designed by Anders Hejlsberg, and its development team is currently led by Mad Jorgensen. The most recent version is 8.0, which was released in 2019 with Visual Studio 2019 “that we use” version

5 Performance Requirements

Long memory = GC.GetTotalMemory (true);

The line code written above shows how much we use memory from our program, which is equals to 187312 at our program load.

We will use any CPU (64bit and 32bit) as we will use C# by Visual Studio Compiler which allows us to determine whether 64bit or 32bit our program will depend on or both.

Also, we should do some tips to improve our performance, such as using for loop instead of foreach because it was tested and shown that for loop is much faster than foreach.

6 Design Constraints

6.1 Standards Compliance

6.2 Hardware Limitations

There are no specific hardware limitations for using our system program, because it is written by C# which is mostly executed in the most PC's

And also, we can target the framework versions which will allows which PC's can run this program

(A lower version implies less language features but more compatibility usually).

6.3 others as appropriate

7 Other non-functional attributes

7.1 Security

Our system will be secured with many methods as example:

First: Every worker (Manager or Co-Manager or Employee or User) will have a password and unique username, and in another side, all of them have their specific functionality on the system.

In addition, passwords will be protected by **md5**
(Message-Digest algorithm 5) 128-bit hash technique.
EasyEncryption.MD5.ComputeMD5Hash (string password);
Which is a readymade code and easy to use and easy to encrypt.

Second: If any one hack the system and his goal to delete or damage database system, there are a backup that store the database. We will back up the data by using Redundant Array of Inexpensive Disks (RAID) as data storage technology; there are many types of RAID but we will focus on RAID 5 that if a drive fails, we still have access to all data, even while the failed drive is being replaced and the storage controller rebuilds the data on the new drive. Additionally, we may use RAID 6 in the future if our data is expanded to the same extent. Because RAID 6 covers asynchronous disk failure, but it costs and performs less like RAID 5, so, In RAID 6 we discussed that if two drives fail, you still have access to all data, even while the failed drives are being replaced. So, RAID 6 is more secure than RAID 5 but with high price and less performance.

7.2 Binary Compatibility

We will use any CPU (64bit and 32bit) as we will use C# by Visual Studio Compiler which allows us to determine whether 64bit, 32bit or both our program will depend on.

7.3 Reliability

Not Applicable.

7.4 Maintainability

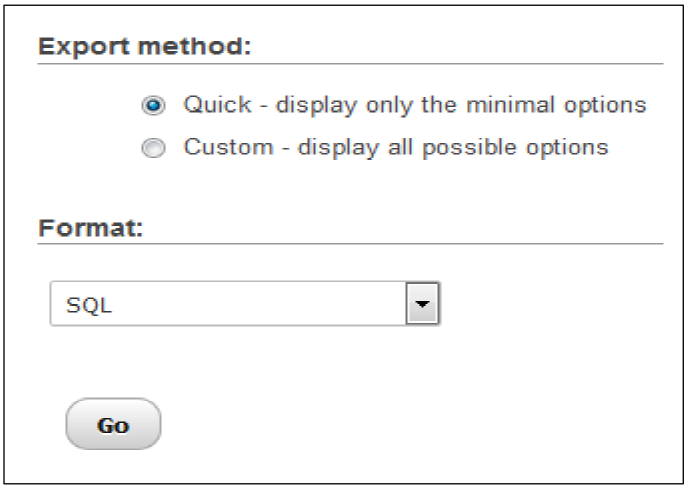
Not Applicable.

7.5 Portability

The whole system should be able to work in any PC with any operating system as it is related to the internet, and it will be coded by HTML, CSS, JS, PHP connected to the localhost, while the CRM is coded by C# WinForms, so it only works on Windows 7, 8, 8.1 or 10, so this need to work with C# and use visual studio as a compiler is recommended.

7.6 Extensibility

In this part we will manage the database to avoid damage of the system if the storage is going to full, in this section, we are going to back up our data that is implemented in phpMyAdmin. The database is exposed to be damaged or deleted by any future reason, so we need to export the the database usually by the export method in phpMyAdmin which allows to save these database offline in a hard disk. There are two methods for exporting, which are quick and Custom, Quick are the default and Custom recovers all the possible options that will be available as shown in the figure. Also, when we save the database in our disk, it will be also backup using RAID technique as said in the "security".



Export method:

● Quick - display only the minimal options

○ Custom - display all possible options

Format:

SQL

7.7 Re-usability

The system is divided into components, which will be re-used to obtain information to benefit data from it and re-prioritize it to the client, and also to obtain this data for advertising.

Such as: The most carpet that was sold, that means that this carpet was liked by the customers then the system will automatically will prefer this carpet to the clients to increase the sales, the most customers that bought carpets, we can make for them an offers to increase their quantity and that will increase the sales, the calculation of the day salaries are help us to calculate the month inventory and to broke down the target number and to calculate also the profits of the year.

7.8 Application Affinity/Compatibility

Not Applicable.

7.9 Resource Utilization

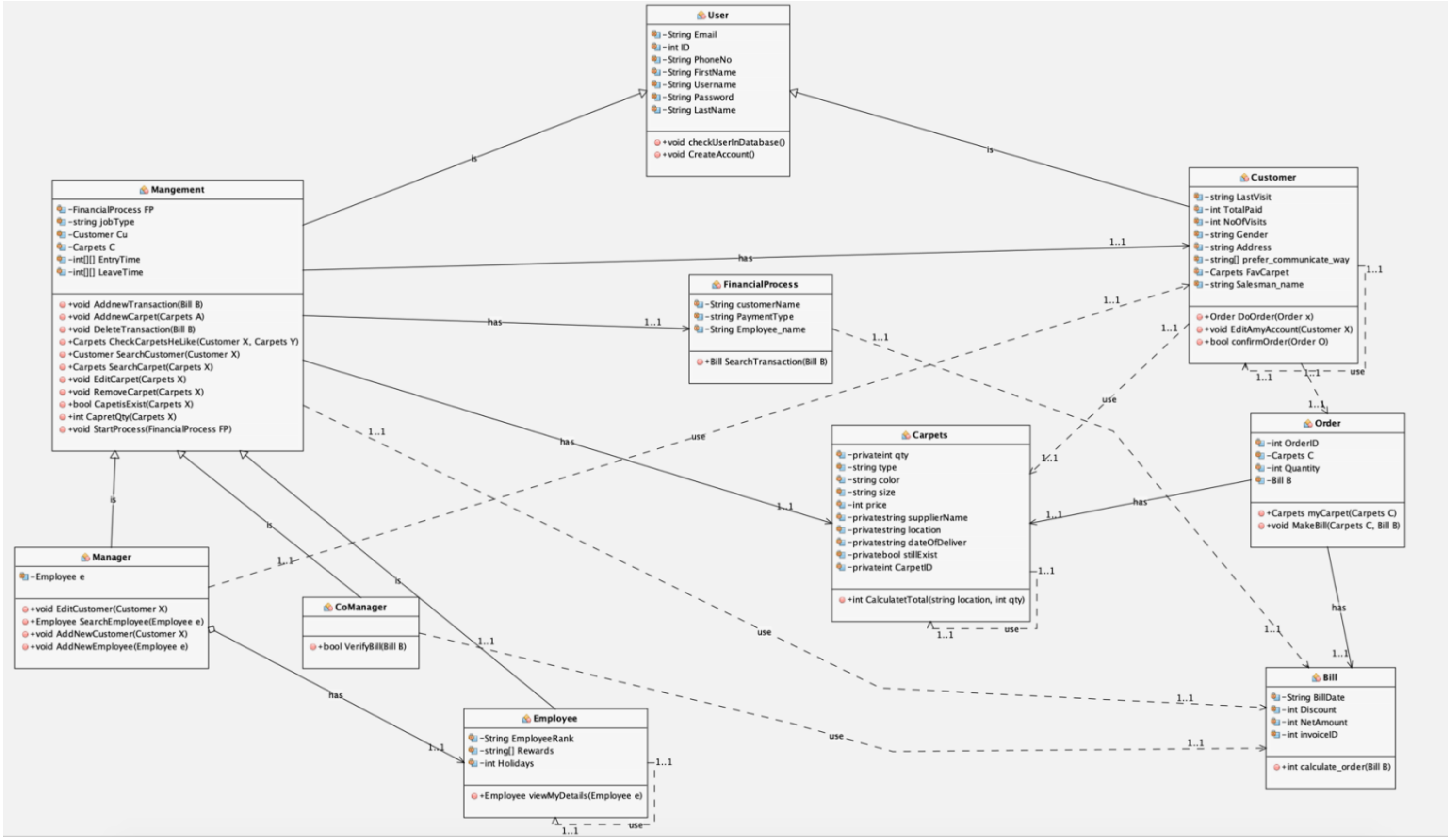
Not Applicable.

7.10 Serviceability

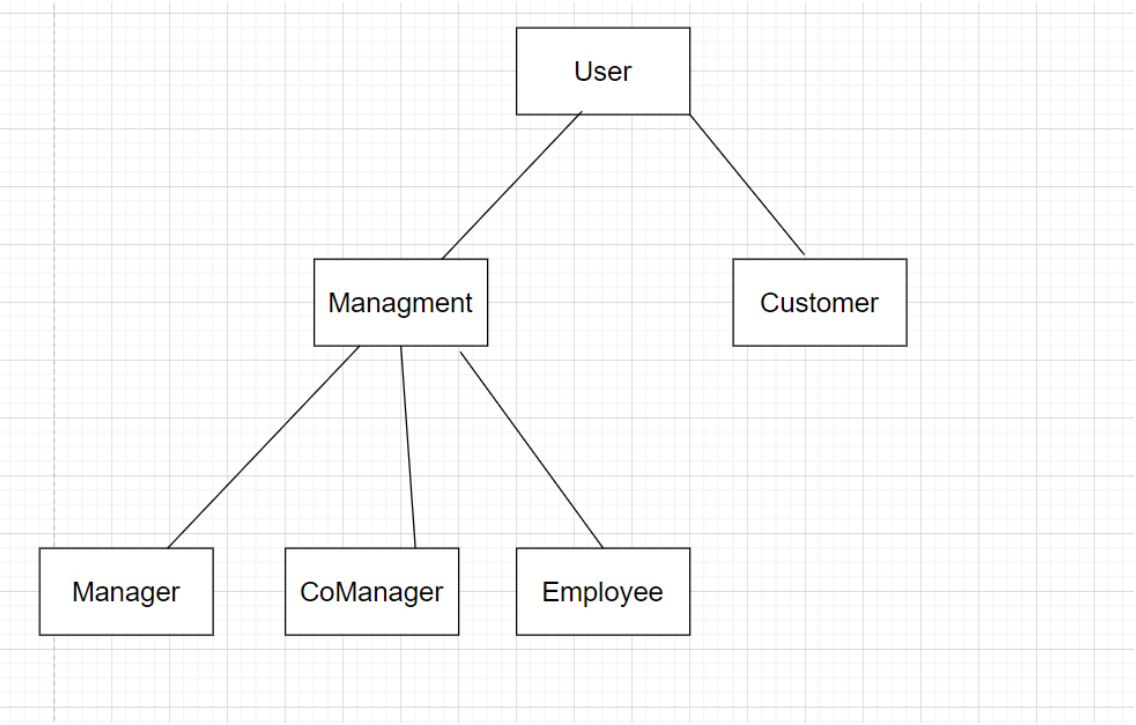
Not Applicable.

7.11 others as appropriate

8 Preliminary Object-Oriented Domain Analysis


User
-String Email
-int ID
-String PhoneNo
-String FirstName
-String Username
-String Password
-String LastName
+void checkUserInDatabase()
+void CreateAccount()
Mangement
-FinancialProcess FP
-string jobType
-Customer Cu
-Carpets C
-int[][] EntryTime
-int[][] LeaveTime
+void AddnewTransaction(Bill B)
+void AddnewCarpet(Carpets A)
+void DeleteTransaction(Bill B)
+Carpets CheckCarpetsHeLike(Customer X, Carpets Y)
+Customer SearchCustomer(Customer X)
+Carpets SearchCarpet(Carpets X)
+void EditCarpet(Carpets X)
+void RemoveCarpet(Carpets X)
+bool CapetisExist(Carpets X)
+int CapretQty(Carpets X)
+void StartProcess(FinancialProcess FP)
FinancialProcess
-String customerName
-string PaymentType
-String Employee_name
+Bill SearchTransaction(Bill B)
Customer
-string LastVisit
-int TotalPaid
-int NoOfVisits
-string Gender
-string Address
-string[] prefer_communicate_way
-Carpets FavCarpet
-string Salesman_name
+Order DoOrder(Order x)
+void EditAmyAccount(Customer X)
+bool confirmOrder(Order O)
Carpets
-privateint qty
-string type
-string color
-string size
-int price
-privatestring supplierName
-privatestring location
-privatestring dateOfDeliver
-privatebool stillExist
-privateint CarpetID
+int CalculatetTotal(string location, int qty)
Order
-int OrderID
-Carpets C
-int Quantity
-Bill B
+Carpets myCarpet(Carpets C)
+void MakeBill(Carpets C, Bill B)
Manager
-Employee e
+void EditCustomer(Customer X)
+Employee SearchEmployee(Employee e)
+void AddNewCustomer(Customer X)
+void AddNewEmployee(Employee e)
CoManager
+bool VerifyBill(Bill B)
Employee
-String EmployeeRank
-string[] Rewards
-int Holidays
+Employee viewMyDetails(Employee e)
Bill
-String BillDate
-int Discount
-int NetAmount
-int invoiceID
+int calculate_order(Bill B)
is
has
use
1..1

8.1 Inheritance Relationships



8.2 Class descriptions

List of Super Classes	List of Subclasses
User	Customer
Management	Manager
	CoManager
	Employee

8.2.(1,2,3,4,5,6) CRC Cards

Name: User	**ID: 1**	**Type: Concrete**
Description: The Parent class of the system which all sub classes will be inherited from this class.		
Responsibilities	**Collaborators**	
1- It creates a new user **2- Return a verify if the user is in the database**	**None**	
Attributes: Email (**string**), ID (**intgear**), PhoneNo (**string**), Firstname (**string**), Username (**string**), Password (**string**), Lastname (**string**)		
Relationships: A parent to **2** Classes, **Management** and **Customer**		

<table>
<tr><td><b>Name:
Management</b></td><td><b>ID: 2</b></td><td><b>Type: Concrete</b></td></tr>
<tr><td colspan="3"><b>Description: Class that contains the main functionalities of the financial processes and customer relation management processes and is the parent for Manager, Co-Manager and Employee classes.</b></td></tr>
<tr><td colspan="2"><b>Responsibilities</b></td><td><b>Collaborators</b></td></tr>
<tr><td colspan="2"><b>- Add new Transaction
- Add new carpet
- Delete Transaction
- Check what carpet the customer like
- Search for customer information
- Check Carpets Quantity
- Check if a carpet exists
- Edit Carpet
- Remove carpet
- Search for a Carpet
- Start the Financial Process</b></td><td><b>1- Customer</b></td></tr>
<tr><td colspan="3"><b>Attributes:</b>
FinancialProcess (<b>object</b>), Carpets (<b>object</b>), jobType (<b>string</b>), Customer (<b>Object</b>),
EntryTime (<b>2D array of integer</b>), LeaveTime (<b>2D array of integer</b>)</td></tr>
<tr><td colspan="3"><b>Relationships:</b>
Generalization(a-kind-of):<b>User</b>
Aggregation(Part-of):<b>FinacialProcess,Carpets,Customer</b>
other Association: <b>Bill</b></td></tr>
</table>

<table>
<tr><td><b>Name: Manager</b></td><td colspan="2"><b>ID: 3</b></td><td><b>Type: Concrete</b></td></tr>
<tr><td colspan="4"><b>Description: An individual that has the ability to add,edit and remove customer accounts and manage employee data .</b></td></tr>
<tr><td colspan="2"><b>Responsibilities</b></td><td colspan="2"><b>Collaborators</b></td></tr>
<tr><td colspan="2">- <b>Edit customer</b>
- <b>Search employees</b>
- <b>Add new customer</b>
- <b>Add new employee</b>
- <b>Other functions from its parent (Management)</b></td><td colspan="2"><b>- Customer</b></td></tr>
<tr><td colspan="4"><b>Attributes:</b>
Employee (<b>Object</b>)</td></tr>
<tr><td colspan="4"><b>Relationships:</b>
Generalization(a-kind-of):<b>Management</b>
Aggregation(Part-of):<b>Employee</b>
other Association: <b>Customer</b></td></tr>
</table>

<table>
<tr><td><b>Name: CoManager</b></td><td colspan="2"><b>ID: 4</b></td><td><b>Type: Concrete</b></td></tr>
<tr><td colspan="4"><b>Description: An individual that authenticate the bills</b></td></tr>
<tr><td colspan="2"><b>Responsibilities</b></td><td colspan="2"><b>Collaborators</b></td></tr>
<tr><td colspan="2"><b>1- Verify the bills and other functions that is his parent (Management) do</b></td><td colspan="2"><b>- Bill</b></td></tr>
<tr><td colspan="4"><b>Attributes:</b>
No Attributes</td></tr>
<tr><td colspan="4"><b>Relationships:</b>
Generalization(a-kind-of):<b>Management</b>
other Association: <b>Bill</b></td></tr>
</table>

<table>
<tr><td>Name: Employee</td><td colspan="2">ID: 5</td><td>Type: Concrete</td></tr>
<tr><td colspan="4">Description: An individual can add bills and use the customer relation management functionalities</td></tr>
<tr><td colspan="2">Responsibilities</td><td colspan="2">Collaborators</td></tr>
<tr><td colspan="2">1- Can do his Parent’s Functions (Management)
2- Check his details himself</td><td colspan="2">Employee (itself)</td></tr>
<tr><td colspan="4">Attributes:
EmployeeRank (**string**), Rewards (**array of string**), Holidays (**integer**)</td></tr>
<tr><td colspan="4">Relationships:
Generalization(a-kind-of):**Managment**
other Association: **Employee** (**itself**)</td></tr>
</table>

Name: FinancialProcess	**ID: 6**	**Type: Concrete**
Description: This classes manages the transactions done by the Customers and controlled by the Workers		
Responsibilities	**Collaborators**	
1- Return the whole transaction info	**- Bill**	
Attributes: CustomerName (**string**), PayementType (**string**), Employee_name (**string**)		
Relationships: other Association: **Bill**		

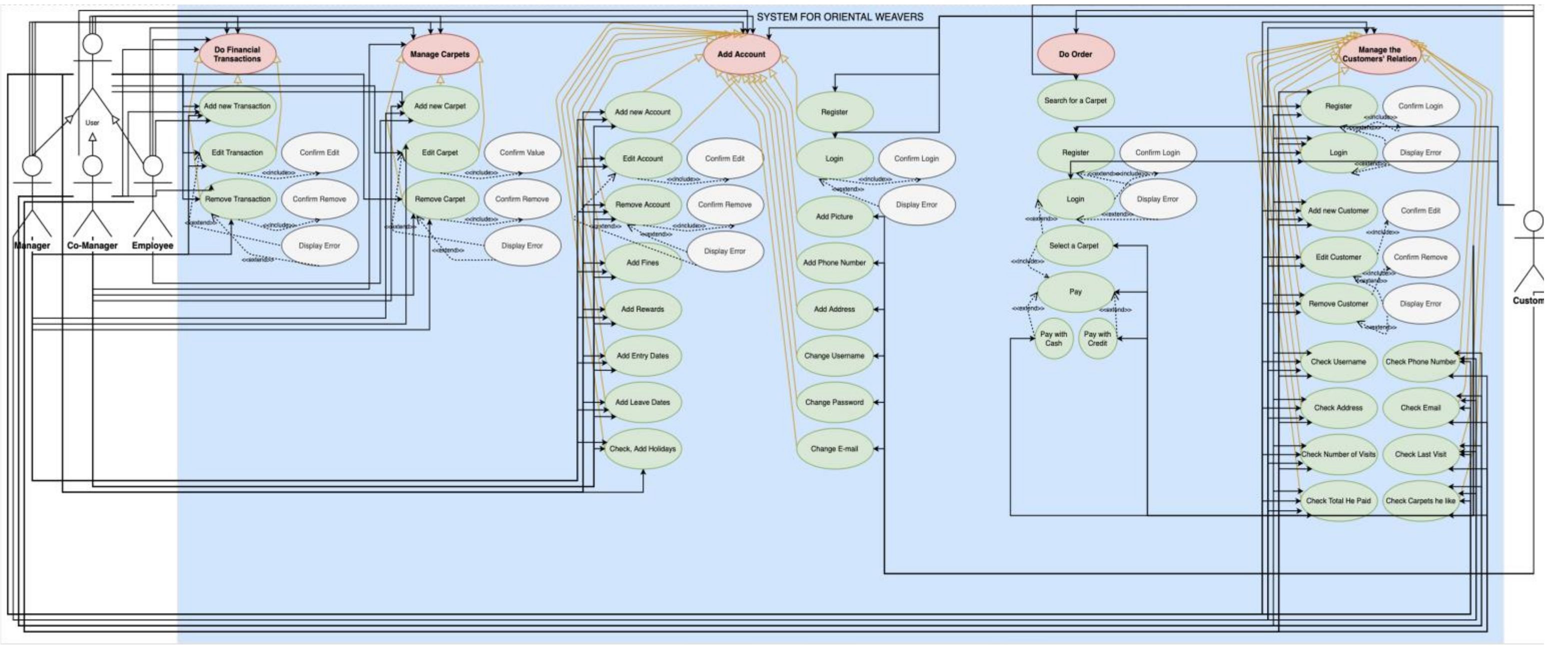
<table>
<tr><td><b>Name: Carpets</b></td><td colspan="2"><b>ID: 7</b></td><td><b>Type: Concrete</b></td></tr>
<tr><td colspan="4"><b>Description: This class contains the carpets characteristics and all its info</b></td></tr>
<tr><td colspan="2"><b>Responsibilities</b></td><td colspan="2"><b>Collaborators</b></td></tr>
<tr><td colspan="2"><b>1- Return the total of the carpets’ quantity in which location (either factory or the branch)</b></td><td colspan="2"><b>Carpets (itself)</b></td></tr>
<tr><td colspan="4"><b>Attributes:</b>
type (<b>string</b>), color (<b>string</b>), size (<b>string</b>), price (<b>integer</b>),
supplierName (<b>string</b>), Location (<b>string</b>), DateOfdeliver (<b>string</b>),
StillExist (<b>boolen</b>), ID (<b>integer</b>)</td></tr>
<tr><td colspan="4"><b>Relationships:</b>
other Association: <b>Carepts</b></td></tr>
</table>

<table>
<tr><td><b>Name: Customer</b></td><td colspan="2"><b>ID: 8</b></td><td><b>Type: Concrete</b></td></tr>
<tr><td colspan="4"><b>Description: An individual that has the ability to order a carpet and also his information is stored to be shown by the workers.</b></td></tr>
<tr><td colspan="2"><b>Responsibilities</b></td><td colspan="2"><b>Collaborators</b></td></tr>
<tr><td colspan="2"><b>1- Do an order</b>
<b>2- Edit his account</b>
<b>3- Has the ability to verify the order or not</b></td><td colspan="2"><b>- Order</b>
<b>- Customer (itself)</b>
<b>- Carpets</b></td></tr>
<tr><td colspan="4"><b>Attributes:</b>
LastVisit (<b>string</b>), TotalPaid (<b>integer</b>), NoOfVisits (<b>integer</b>),
Gender (<b>string</b>), Address (<b>string</b>), prefer_communicate_way(<b>array of string</b>),
Carpets(<b>Object</b>), Salesman_name(<b>string</b>)</td></tr>
<tr><td colspan="4"><b>Relationships:</b>
Generalization(a-kind-of):<b>User</b>
other Association: <b>Order, Customer(itself), Carpets</b></td></tr>
</table>

<table>
<tr><td><b>Name: Order</b></td><td colspan="2"><b>ID: 9</b></td><td><b>Type: Concrete</b></td></tr>
<tr><td colspan="4"><b>Description: This class is made to order carpets</b></td></tr>
<tr><td colspan="2"><b>Responsibilities</b></td><td colspan="2"><b>Collaborators</b></td></tr>
<tr><td colspan="2"><b>1- Select a carpet</b>
<b>2- Create the bill related to the customer’s order</b></td><td colspan="2"><b>None</b></td></tr>
<tr><td colspan="4"><b>Attributes:</b>
OrderID (<b>integer</b>), Carpets (<b>object</b>), Quantity (<b>integer</b>), Bill (<b>Object</b>)</td></tr>
<tr><td colspan="4"><b>Relationships:</b>
Aggregation(Part-of): <b>Carpets, Bill</b></td></tr>
</table>

<table>
<tr><td><b>Name: Bill</b></td><td colspan="2"><b>ID: 10</b></td><td><b>Type: Concrete</b></td></tr>
<tr><td colspan="4"><b>Description: This class is responsible to create the bill after an order is made by the Workers.</b></td></tr>
<tr><td colspan="2"><b>Responsibilities</b></td><td colspan="2"><b>Collaborators</b></td></tr>
<tr><td colspan="2"><b>1- Calculate the Order price</b></td><td colspan="2"><b>None</b></td></tr>
<tr><td colspan="4"><b>Attributes:</b>
BillDate (<b>string</b>), Discount (<b>integer</b>), NetAmount (<b>integer</b>), Billinvoice (<b>integer</b>)</td></tr>
<tr><td colspan="4"><b>Relationships:</b>
None</td></tr>
</table>

8.2.7 Operations



8.2.8 Constraints:

9 Operational Scenarios

This section should describe a set of scenarios that illustrate, from the user's

Initial assumption: A customer tries to make an online order he logs in into his account through the system and choose the carpets that he like and make the order.

Normal: The Customer searches for a specific color that he wants for the carpets and then view the types of carpet based on the color he searched for, after that he select the carpet size that he wants and adds it to the cart. Finally, when he finishes shopping, he goes to the order menu and select online payment and then he types his information and click do order after that a message pop up that transaction is done successfully.

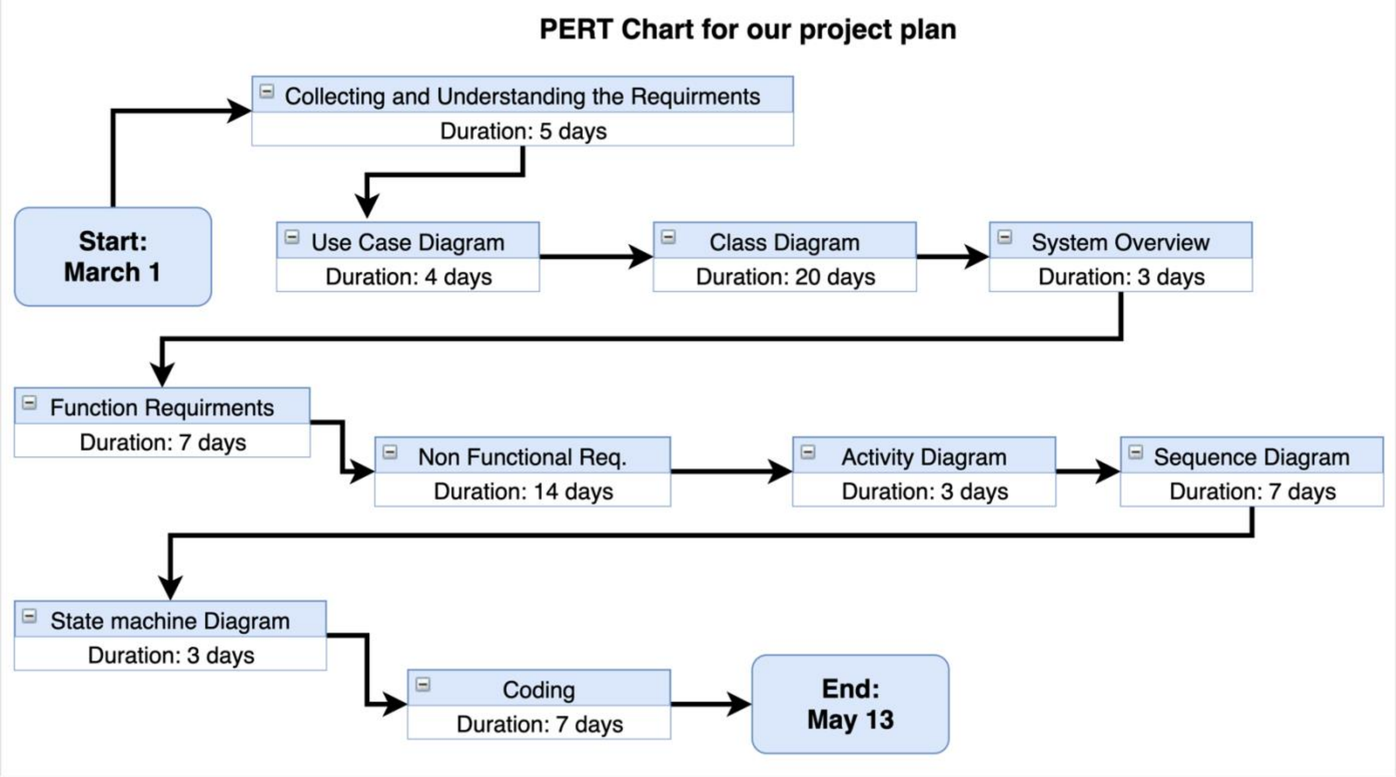
What can go wrong: The transaction was made successfully but the order was not recorded to the system. Customer will not receive his order. Manger or Co-Manager or Employee should check the transactions date and time and then ask the customer what did he order to make a new bill.

Other activities: Order maybe made but not saved in the database due to lack of connection.

System state on completion: Customer logged in. The transaction is made successfully but the order is not recorded on the system. a record of transaction is added to the log showing when was it done.

10 Preliminary Schedule Adjusted

The initial version of the project plan was taking about 2 – 4 months, and yes, we started 2 months ago and today 13 May 2020 we finished. We start with the Use-Case-Diagram then Class-Diagram then system overview then NonFunctional Requirements then Functional Requirements then Sequence, State machine, Activity Diagrams followed with the implemented code. Also, Class Diagram was edited with every new move in any other task to be fully completed. Our program doesn't need much hardware requirements as it is code either in WEB or C# which is mostly easily coded and executed in most of the PCs' hardware. We used Visual Studio Community as our software, but if we talked about the whole system, we use Visual Studio Code for Web Coding.



11 Preliminary Budget Adjusted

The initial budget for this project was 11$ one time and 20$ per month and it remains the same, also, the implementation part was totally free which C# coded and it also still free as it was.

12 Appendices

12.1 Definitions, Acronyms, Abbreviations

(**CRM**): Customer relationship management

(**KPI**): Key Performance Indicator

(**NBM**): Novell Border Manager

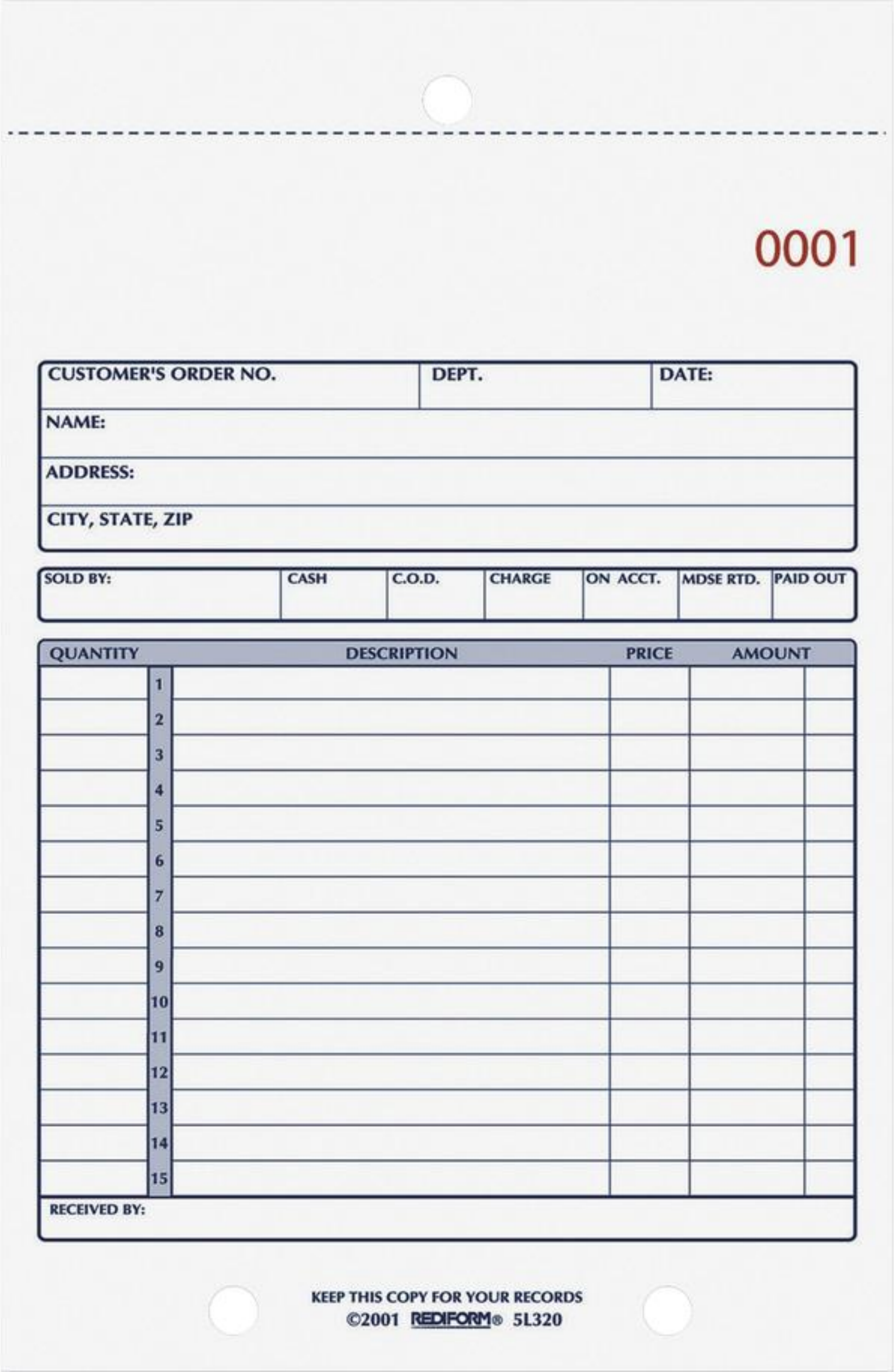
(**API**): Application Programming Interface

(**GUI**): Graphical User Interface

(**RAID**): Redundant Array of Inexpensive disks

(**MD5**): Message Digest Algorithm 5

12.2 Collected material



0001

CUSTOMER'S ORDER NO.	DEPT.	DATE:
NAME:		
ADDRESS:		
CITY, STATE, ZIP		

SOLD BY:	CASH	C.O.D.	CHARGE	ON ACCT.	MDSE RTD.	PAID OUT

QUANTITY		DESCRIPTION	PRICE	AMOUNT	
	1				
	2				
	3				
	4				
	5				
	6				
	7				
	8				
	9				
	10				
	11				
	12				
	13				
	14				
	15				

RECEIVED BY:

KEEP THIS COPY FOR YOUR RECORDS
©2001 REDIFORM® 5L320

Balance Sheet Support Schedule
Accounts Payable
& Sales Tax Payable

This form covers the month of___________

Accounts Payable

	This Month This Year	This Month Last Year
Balance per Balance Sheet:	$ ________________	$ ________________

Balance Per Detailed Accounts Payable System

	This Month This Year	This Month Last Year
Beginning Balance:	$ ________________	________________
Purchases:	________________	________________
Disbursements:	________________	________________
Adjustments:	________________	________________
Ending Balance:	$ ________________	________________
Additional Amounts:	________________	________________
Total Accounts Receivable Balance:	$ ________________	________________

Detail totals must agree with Accounts Payable on Balance Sheet.

Sales Tax Payable

	This Month This Year	This Month Last Year
Balance per Balance Sheet:	$ ________________	$ ________________

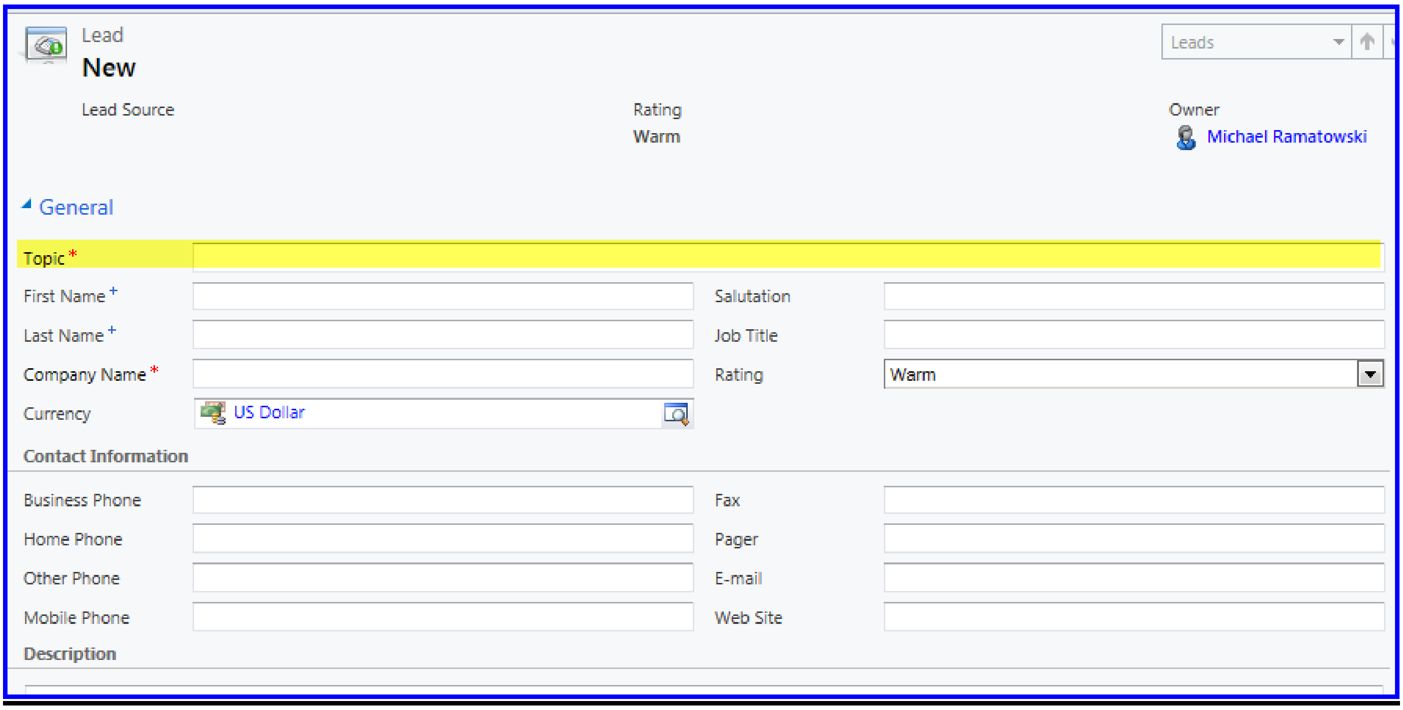
Balance Per Detailed Monthly Sales Tax Schedules

	This Month This Year	This Month Last Year
For Month 1:	$ ________________	________________
For Month 2:	________________	________________
For Month 3:	________________	________________
Total Sales Tax Payable for Quarter:	$ ________________	________________
Additional Amounts:	________________	________________
Total Sales Tax Payable:	$ ________________	________________

Detail totals must agree with Sales Tax Payable on Balance Sheet.

Prepared By ____ Date ____________________

Reviewed By ____ Date ____________________



Lead

New

Lead Source	Rating Warm	Owner Michael Ramatowski

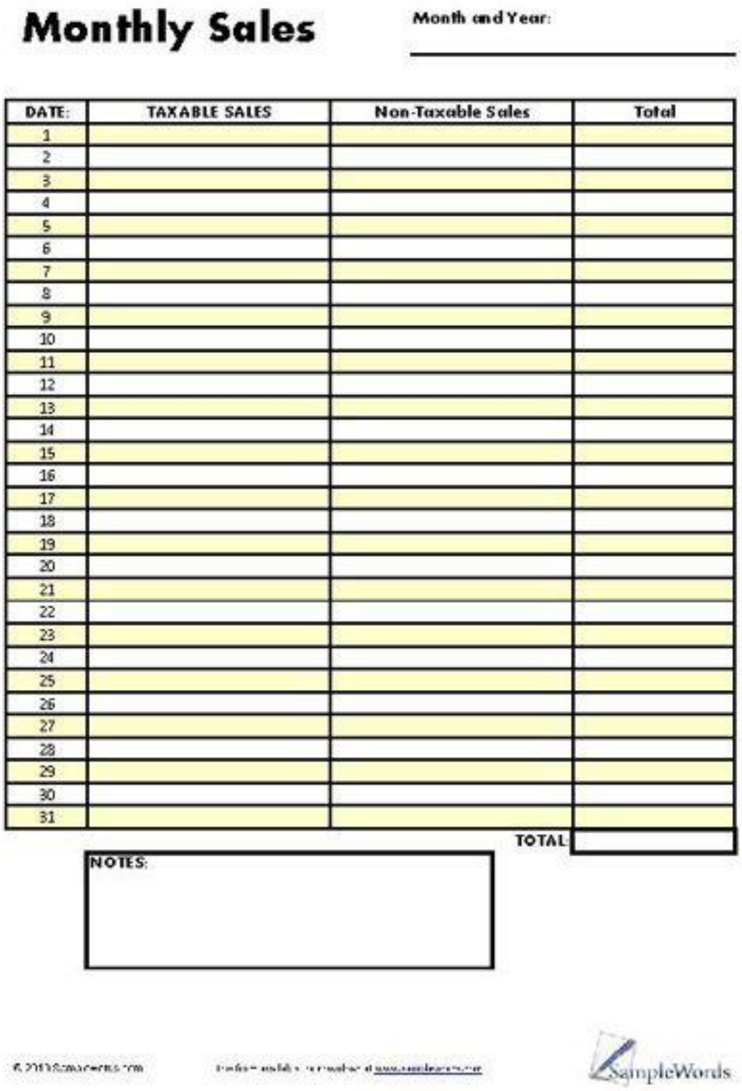
General

Topic *			
First Name +		Salutation	
Last Name +		Job Title	
Company Name *		Rating	Warm
Currency	US Dollar		

Contact Information

Business Phone		Fax	
Home Phone		Pager	
Other Phone		E-mail	
Mobile Phone		Web Site	

Description



Monthly Sales

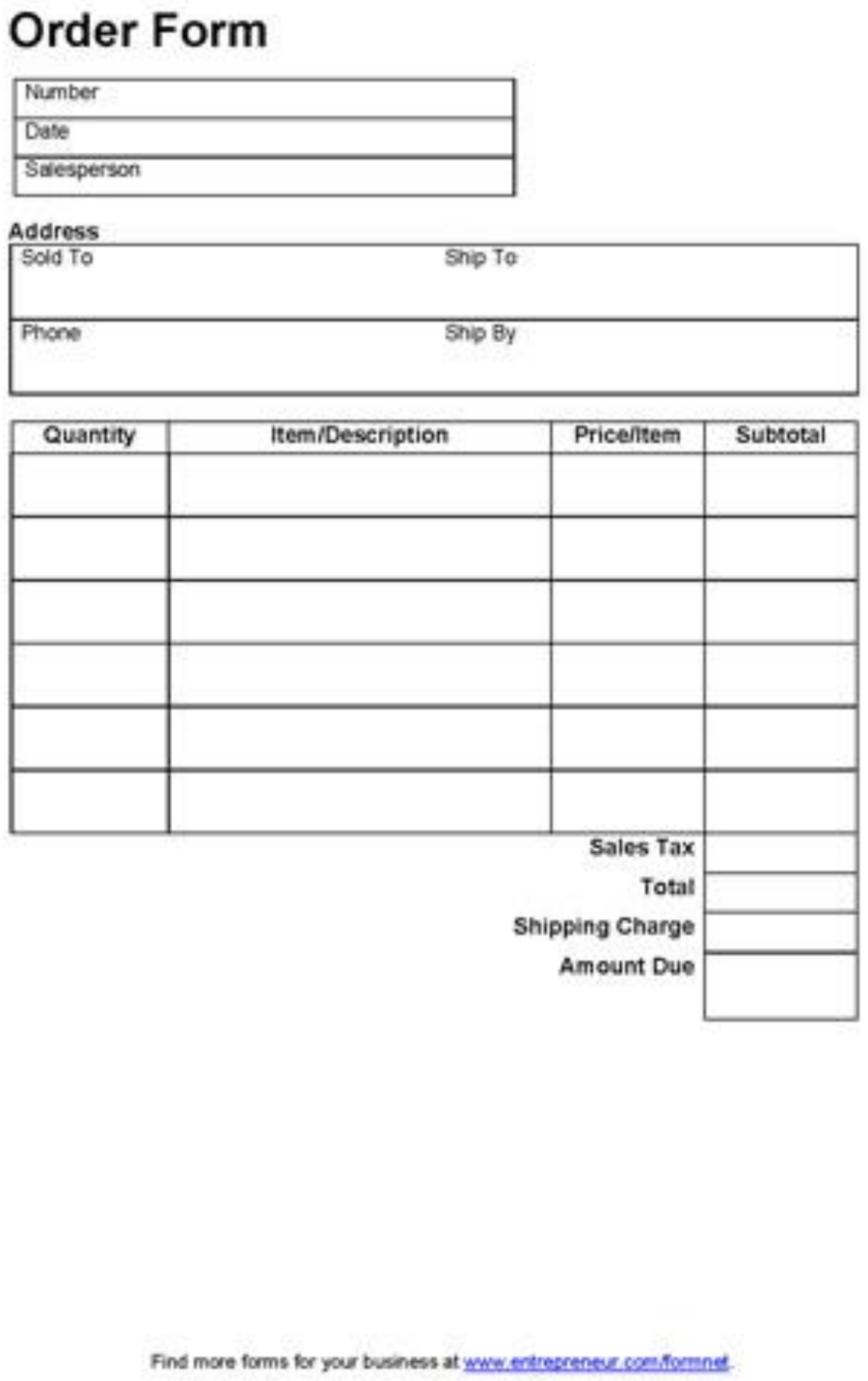
Month and Year: ____________________

DATE:	TAXABLE SALES	Non-Taxable Sales	Total
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			
		TOTAL:	

NOTES:

[illegible] [illegible]

SampleWords



Order Form

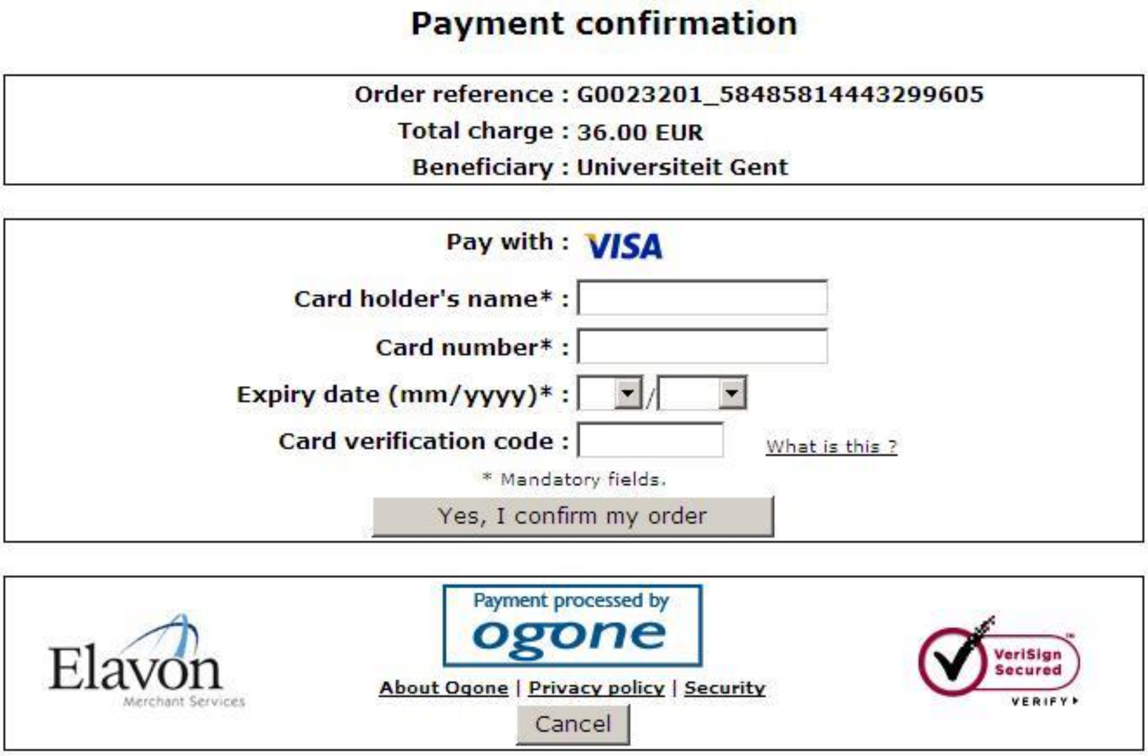
Number
Date
Salesperson

Address

Sold To	Ship To
Phone	Ship By

Quantity	Item/Description	Price/Item	Subtotal
		Sales Tax	
		Total	
		Shipping Charge	
		Amount Due	

Find more forms for your business at www.entrepreneur.com/formnet.



Payment confirmation

Order reference : G0023201_58485814443299605

Total charge : 36.00 EUR

Beneficiary : Universiteit Gent

Pay with : VISA

Card holder's name* :

Card number* :

Expiry date (mm/yyyy)* : /

Card verification code :

* Mandatory fields.

13 References

References

https://help.dreamhost.com/hc/en-us/articles/214395738-phpMyAdmin-How-to-backup-or-export-a-database-or-table

https://support.microsoft.com/en-us/help/323434/how-to-establish-a-striped-volume-with-parity-raid-5-in-windows-server

https://stackoverflow.com/questions/14032515/how-to-get-the-amount-of-memory-used-by-an-application

https://www.c-sharpcorner.com/UploadFile/dacca2/5-tips-to-improve-performance-of-C-Sharp-code/

https://stackoverflow.com/questions/33456472/requirements-for-running-compiled-c-sharp-programs-exe

https://www.tecmint.com/secure-phpmyadmin-centos-ubuntu/