Software Requirement Specification Document

Ahmed Emad, Mohamed Abdelhamed , Fady Bassel, Mark Refaat

March 27, 2020

1 Introduction

1.1 Document versions

Version 1	Create SRS with the initial requirements

1.2 Purpose of this document

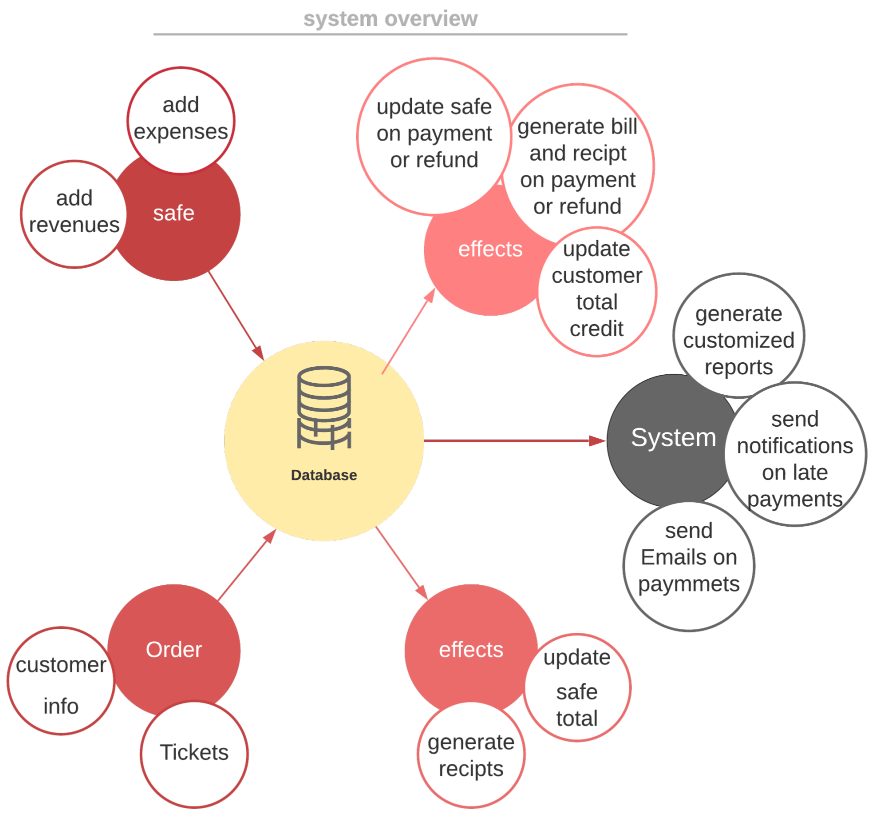
The main purpose of this Software Requirements Specification document is to illustrate and outline the requirements for our project (local system for a tourism company) that are mainly tracking customers data, booking flight tickets and managing the company's finances. Our aim is to facilitate the work flow of our client by converting the working environment from papers and excel sheets to a reliable and easy to use software [4]. This will be done by working on four main parts of the system mentioned above as well as generating reports and invoices for tracking sales progress and help take better business decisions. This document will provide a fulfilled and detailed description about each process in the system. And stating down system constraints, what difficulties have we faced during development and how should we interact with it.

1.3 Scope of this document

Our Local travel system is aimed to help a travelling company throughout it's processes at work. These processes involve tracking customers flight tickets payments and manage their finances.This aims to reduce costs and time for the system users,as well as helping them analyze and take decisions based on detailed and accurate reports and statistics.

1.4 Overview

The system aims to help a tourism company track customers flight tickets payments , manage their finances and generate automated bills and invoices as well as generating detailed reports at times specified by the user. The system is divided into three main parts customers data, booking flight tickets and safe management. The company divides it's customers into two categories regular and ordinary customer, the regular customer has a 15 days span to deliver the ticket(s) payment (15 days upon client request but could be changeable) the system should provide basic information of the customer and if he has any unpaid tickets as well as his previous purchased tickets and notify the company if a customer passed the 15 days allowed for paying the system should also deal with refunding tickets. Booking tickets is done through a form containing basic ticket information (date,destination,price,passenger,etc..) and the result of submitting is a printed bill alongside adding the ticket in the customers account and commencing the 15 day payment period. The system also should deal with the safe , accountants should be able to add expenses and revenues and see detailed general and custom reports generated by the system,also tickets information should be added automatically in the revenues upon payment.


system overview
add expenses
add revenues
safe
update safe on payment or refund
generate bill and recipt on payment or refund
effects
update customer total credit
Database
generate customized reports
System
send notifications on late payments
send Emails on paymmets
Order
customer info
Tickets
effects
update safe total
generate recipts

2 General Description

2.1 Product Functions

- The system shall enable the company to add registered customers.
- The system shall enable the company to track refunds and payments.
- Users shall be able to track customers payments and profiles.
- The system shall enable customers to pay part of the ticket amount or refund certain tickets only.
- The system shall notify the company with late payments if exceeded 15 days (based on the client request and could be changeable)
- The system shall enable the company to manage it's finances through adding revenues and expenses.
- All tickets payments and revenues shall automatically update in the company's finances.
- The system shall notify the manager upon customer payments by email.
- The system shall generate printed invoices upon payment and refunds.
- The system shall generate reports every specific time period selected by the user with detailed information about payments, customers, company's revenues and expenses.

2.2 Similar System Information

Tavelworks solution [2]: is an web application that manage money transactions inside the company, accounts payable, accounts receivable, general ledger, accounting operations (month end, year end, bank reconciliations...) and reporting. They also have real time sales performance. They also offer automatic reminders for deadlines and outstanding payments.

2.3 User Characteristics

In this document, we proposed a system that deals with the common issues of using paperwork, therefore, the system is user friendly enough that any user is able to use it after a brief tutorial (around 1 hour) on how to deal and work with the system and it shall be used by different employees on relatively low computer specifications so a web based system is the suitable choice.

2.4 User Objectives

Customer Module: the user shall add and view registered customers with all their previous payments and view all pending payments and notify him if a payment deadline has passed.
Tickets Module: tickets are added to an order specified to a customer and a bill is generated with the amount and the tickets bought, the amount automatically affects the customers current balance and the deadline span begins, upon paying part or all of the bill amount a receipt is generated and and the balance is updated in the customer profile .

Safe Module: the user shall add all the revenues and expenses of the company and see total credit and view detailed reports by date or by dedicated organization also all customer payments reflect in the revenues and refunds in the expenses automatically.

2.5 User Problem Statement

The tourism company uses hard copied bills and excel sheets to document customers data and payments which slows down and burdens the management and tracking process of the whole process.

3 Functional Requirements

Function	Check Username Validation
ID	F00
Description	This Function is to check if the username is valid before checking it in the database
Input	Username
Action	Checks if the username entered in a real exist username and if it username is valid in format. If so it returns true means that the username is valid to be checked if exist in the database
Output	Boolean true or false
Precondition	user entered username into the required EditText
Post-condition	Return to login function (F02)
Dependencies	-

Function	Check Password Validation
ID	F01
Description	This Function is to check if the password is valid before checking it in the database
Input	Password
Action	Checks if the password entered in a real exist password and if it password is valid in format. If so it returns true means that the password is valid to be checked if exist in the database
Output	Boolean true or false
Precondition	user entered password into the required EditText
Post-condition	return to login function (F02)
Dependencies	-

Function	Login
ID	F02
Description	This Function is for the user to login into the system using his/her account
Input	Username and Password
Action	Check if all data are filled and compare data that was entered to the record in the database, if so function returns true otherwise returns false
Output	Boolean true or false
Precondition	the user needs to input his/her username and password into the EditTexts and that the inputs are verified (FR00, FR01)
Post-condition	Redirect to the home page
Dependencies	F00, F01

Function	Check phone number validation
ID	F03
Description	This Function is to check if the phone number is valid before adding it in the database
Input	Customer's phone number
Action	Checks if the phone number is valid in format. If so it returns true means that the phone number is valid to be added in the database
Output	Boolean true or false
Precondition	User entered the phone number into the EditText
Post-condition	Return to create customer function (F05)
Dependencies	-

Function	Check Email Validation
ID	FR04
Description	This Function is to check if the email is valid before adding it in the database
Input	Customer's email
Action	Checks if the email entered in a real exist email and if it email is valid in format. If so it returns true means that the email is valid to be added in the database
Output	Boolean true or false
Precondition	User entered email into the EditText
Post-condition	Return to create customer function (F05)
Dependencies	-

Function	Add Customer
ID	F05
Description	This Function is to add new customer in the database
Input	Customer object
Action	Checks if the user's email and phone number are validated, if so it enters the user record in the database and returns true else it returns false to show error message
Output	Boolean true or false
Precondition	user entered email, phone no into the required EditTexts
Post-condition	Redirect to Customers page
Dependencies	F03, F04

Function	Delete Customer
ID	F06
Description	The function fire up a query to remove the selected customer row from the customers table
Input	ID of the customer selected to be removed
Action	Check if the customers id exists in the database to remove it
Output	Boolean true or false
Precondition	the user needed to be selected from the list
Post-condition	Return to Customers page
Dependencies	-

Function	Update Customer
ID	F07
Description	This Function fires up a query to customers table for updating the customer data according to the Customer ID
Input	Customer ID, customer object
Action	Check if the customers data is updated and return true else it returns false to show error message
Output	Boolean true or false
Precondition	Check if the customers exists
Post-condition	Return to Customers page
Dependencies	-

Function	Get All Customers
ID	F08
Description	the function fire up a query lists all the customers from customers table that is available in the system database.
Input	-
Action	Retrieving all information about customers
Output	Array of customers
Precondition	-
Post-condition	Return to Customers page
Dependencies	-

Function	Get safe Data
ID	F09
Description	the function fire up a query lists all safe data from safe table that is available in the system database
Input	-
Action	Retrieving all information about safe
Output	Array of data
Precondition	-
Post-condition	Return to safe page
Dependencies	-

Function	View Orders
ID	F10
Description	the function fire up a query lists all orders data for specific customer from orders table that is available in the system database
Input	customer ID
Action	Retrieving all orders data for specific customer
Output	Array of orders
Precondition	-
Post-condition	Return to orders page
Dependencies	-

Function	Add New Order
ID	F11
Description	This Function fires up a query to orders table for creating row with the order data
Input	order object
Action	Check if the order data is inserted and return true else it returns false to show error message
Output	Boolean true or false
Precondition	-
Post-condition	Return to orders page
Dependencies	-

Function	Edit Order
ID	F12
Description	This Function fires up a query to orders table for updating the order data according to the Customer ID and Order ID
Input	order ID , order object
Action	Check if the user data is updated and return true else it returns false to show error message
Output	Boolean true or false
Precondition	Check if the order exists
Post-condition	Return to orders page
Dependencies	-

Function	Calculate Total Credit
ID	F13
Description	This Function fires up a query to get sum orders total price - total paid amount for specific customer
Input	customer ID
Action	Calculate total credit for customer
Output	Integer total credit
Precondition	-
Post-condition	Return to customer page
Dependencies	-

Function	Create ticket
ID	F14
Description	Create the ticket that the customer wants and uploading its data to the database
Input	Ticket object
Action	Check if the ticket data is inserted and return true else it returns false to show error message
Output	Boolean true or false
Precondition	-
Post-condition	Return to tickets page
Dependencies	-

Function	Delete ticket
ID	F15
Description	Delete the ticket that the customer wants and work on the refund process
Input	Ticket id
Action	Check if the ticket id exists in the database to remove it
Output	Boolean true or false
Precondition	the ticket is needed to be selected from the list
Post-condition	Return to customer page
Dependencies	-

Function	View tickets
ID	F16
Description	the function fire up a query lists all tickets data for specific customer from tickets table that is available in the system database
Input	customer ID
Action	Retrieving all orders data for specific customer
Output	Array of tickets if found any
Precondition	-
Post-condition	Return to customer's ticket page
Dependencies	-

Function	Delete order
ID	F17
Description	The function fire up a query to remove the selected order row from the orders table
Input	order id
Action	Check if the order id exists in the database to remove it
Output	Boolean true or false
Precondition	-
Post-condition	Return to customer's order page
Dependencies	-

Function	Update ticket
ID	F18
Description	This Function fires up a query to tickets table for updating the ticket data according to the ticket ID
Input	ticket id
Action	Check if the ticket data is updated and return true else it returns false to show error message
Output	Boolean true or false
Precondition	Check if the ticket exists
Post-condition	Return to customer's tickets page
Dependencies	-

Function	Create safe
ID	F19
Description	This Function fires up a query to safe table for creating safe row with the safe data
Input	safe object
Action	Check if the data is inserted and return true else it returns false to show error message
Output	Boolean true or false
Precondition	-
Post-condition	Return to safe page
Dependencies	-

Function	Update safe
ID	F20
Description	This Function fires up a query to safe table for updating the data according to the safe id
Input	safe id
Action	Check if the safe data is updated and return true else it returns false to show error message
Output	Boolean true or false
Precondition	Check if the safe exists
Post-condition	Return to safe page
Dependencies	-

Function	Delete safe
ID	F21
Description	The function fire up a query to remove the selected safe row from the safe table
Input	safe id
Action	Check if the safe id exists in the database to remove it
Output	Boolean true or false
Precondition	Check if the safe exists
Post-condition	Return to safe page
Dependencies	-

Function	Delete receipt
ID	F22
Description	The function fire up a query to remove the selected receipt row from the receipts table
Input	receipt id
Action	Check if the receipt id exists in the database to remove it
Output	Boolean true or false
Precondition	-
Post-condition	Return to customer page
Dependencies	-

Function	Update receipt
ID	F23
Description	This Function fires up a query to receipt table for updating the receipt data according to the receipt id
Input	receipt id
Action	Check if the receipt data is updated and return true else it returns false to show error message
Output	Boolean true or false
Precondition	Check if the receipt exists
Post-condition	Return to receipt page
Dependencies	-

Function	Create receipt
ID	F24
Description	This Function fires up a query to receipt table for receipt row with the receipt data
Input	receipt object
Action	Check if the receipt data is inserted and return true else it returns false to show error message
Output	Boolean true or false
Precondition	-
Post-condition	Return to receipt page
Dependencies	-

Function	View receipt
ID	F25
Description	the function fire up a query lists all receipt data for specific customer from receipt table that is available in the system database
Input	customer ID
Action	Retrieving all receipt data for specific customer
Output	Array of receipts if found any
Precondition	-
Post-condition	Return to customer's receipt page
Dependencies	-

Function	Update refunded ticket
ID	F26
Description	This Function fires up a query to refunded ticket table for updating the refunded ticket data according to the refunded ticket id
Input	refunded ticket id, updated data
Action	Check if the refunded ticket data is updated and return true else it returns false to show error message
Output	Boolean true or false
Precondition	Check if the refunded ticket exists
Post-condition	Return to refunded tickets page
Dependencies	-

Function	Update moalakat
ID	F27
Description	This Function fires up a query to moalakat table for updating the moalakat data according to the moalakat id
Input	moalakat id, updated data
Action	Check if the moalakat data is updated and return true else it returns false to show error message
Output	Boolean true or false
Precondition	Check if the moalakat exists
Post-condition	Return to moalakat page
Dependencies	-

Function	View refunded ticket
ID	F28
Description	the function fire up a query lists all refunded tickets data for specific customer from refunded ticket table that is available in the system database
Input	customer ID , refunded ticket ID
Action	Retrieving all refunded tickets data for specific customer
Output	Array of refunded tickets if found any
Precondition	-
Post-condition	Return to customer’s tickets page
Dependencies	-

Function	View moalakat
ID	F29
Description	the function fire up a query lists all moalakat data for specific customer from moalakat table that is available in the system database
Input	customer ID , moalakat ID
Action	Retrieving all moalakat data for specific customer
Output	Array of moalakat if found any
Precondition	-
Post-condition	Return to customer's moalakat page
Dependencies	-

Function	Create moalakat
ID	F30
Description	This Function fires up a query to moalakat table for moalakat row with the moalakat data
Input	moalakat data , moalakat ID
Action	Check if the moalakat data is inserted and return true else it returns false to show error message
Output	Boolean true or false
Precondition	-
Post-condition	Return to customer's moalakat page
Dependencies	-

Function	Create refunded ticket
ID	F31
Description	This Function fires up a query to refunded tickets table for refunded ticket row with the refunded ticket data
Input	refunded ticket data , refunded ticket ID
Action	Check if the refunded ticket data is inserted and return true else it returns false to show error message
Output	Boolean true or false
Precondition	-
Post-condition	Return to customer's tickets page
Dependencies	-

Function	Delete refunded ticket
ID	F32
Description	The function fire up a query to remove the selected refunded ticket row from the refunded ticket table
Input	refunded ticket ID
Action	Check if the refunded ticket id exists in the database to remove it
Output	Boolean true or false
Precondition	-
Post-condition	Return to customer's tickets page
Dependencies	-

Function	Delete molakat
ID	F33
Description	The function fire up a query to remove the selected molakat row from the molakat table
Input	molakat ID
Action	Check if the molakat id exists in the database to remove it
Output	Boolean true or false
Precondition	-
Post-condition	Return to molakat page
Dependencies	-

Function	Create Report
ID	F34
Description	This function is for the user to create a full reports about the system and money
Input	period of time needed for the report
Action	If the inputs are not empty the report will be generated
Output	the report
Precondition	The user choose the time period to create report on
Post-condition	Report will be shown on the screen for the user
Dependencies	-

Function	Send Mail
ID	F35
Description	the functions send mail to the users saved in database
Input	Mail object
Action	Check if the mail is sent to the player else error message will appear
Output	String mail is sent or error message
Precondition	Receiver must be existed in the database
Post-condition	Mail is created and sent
Dependencies	-

Function	Send Notifications
ID	F36
Description	This Function is for the user to send notifications to the customers
Input	Notification object
Action	Check if the notification is sent to the customer else error message will appear
Output	String notification is sent or error message
Precondition	Customer must be existed in the database
Post-condition	Notification is created and inserted in the database.
Dependencies	-

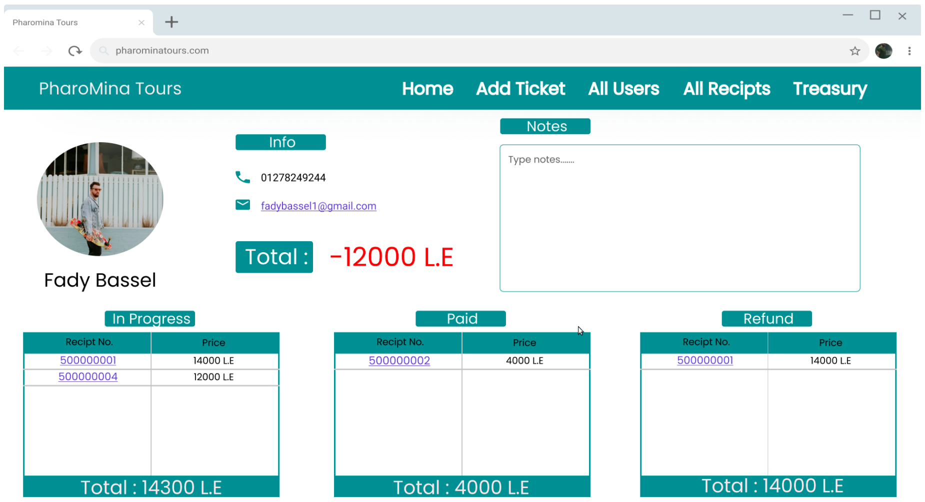
Function	View Notifications
ID	F37
Description	This Function is for listing all the notifications in the database
Input	-
Action	Retrieving all the notification in the database
Output	Array of notifications
Precondition	check if the notifications exist
Post-condition	-
Dependencies	-

Function	Delete Notifications
ID	F38
Description	This Function is for the user if he wants to delete notification from the database
Input	Notification object
Action	Check if the notification exists in the database and delete it
Output	Acceptance message if the record of notification is deleted, else false to show error message
Precondition	select the notification to be deleted
Post-condition	The record of notification in database is deleted
Dependencies	-

4 Interface Requirements

4.1 User Interfaces

4.1.1 GUI



01278249244

fadybassel1@gmail.com

Total : -12000 L.E

In Progress

Recipt No.	Price
500000001	14000 L.E
500000004	12000 L.E

Total : 14300 L.E

Paid

Recipt No.	Price
500000002	4000 L.E

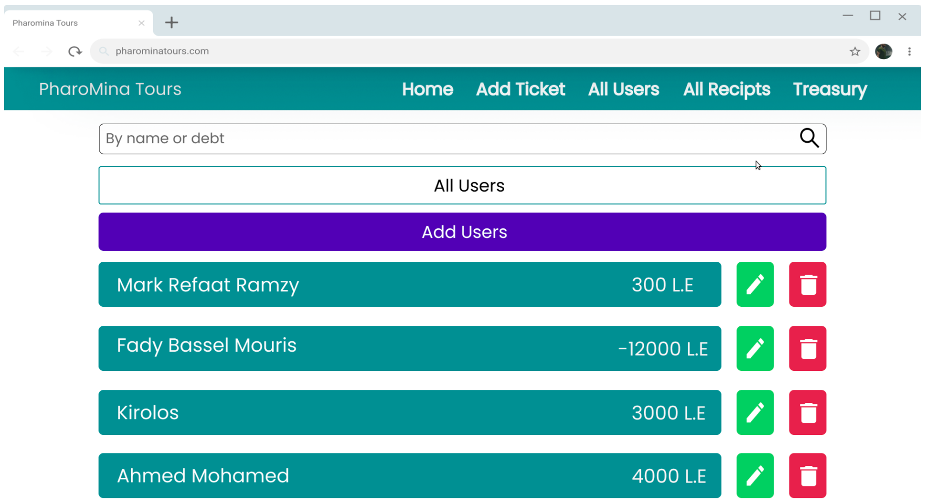
Total : 4000 L.E

Refund

Recipt No.	Price
500000001	14000 L.E

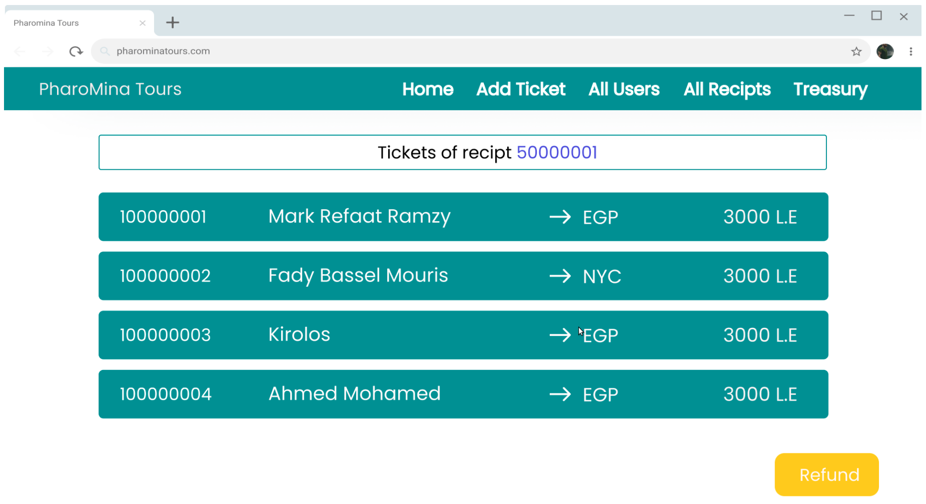
Total : 14000 L.E

Figure 1: Customer profile



Mark Refaat Ramzy	300 L.E
Fady Bassel Mouris	-12000 L.E
Kirolos	3000 L.E
Ahmed Mohamed	4000 L.E

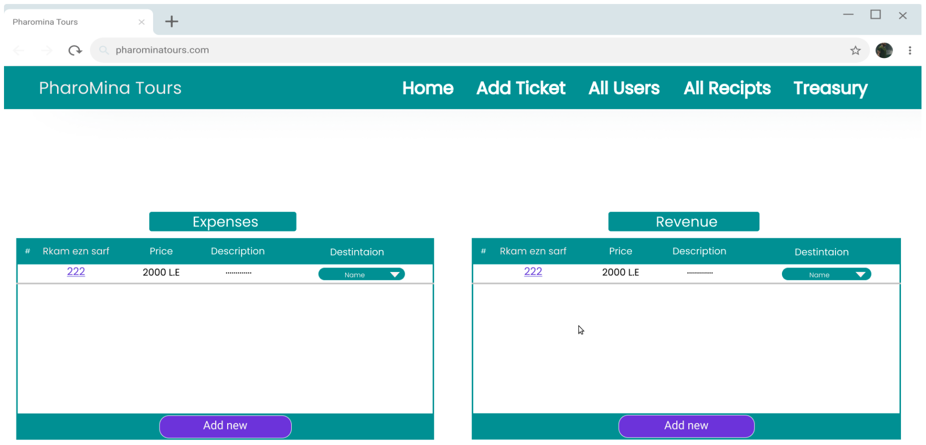
Figure 2: All customers



Tickets of recipt 50000001

100000001	Mark Refaat Ramzy	→ EGP	3000 L.E
100000002	Fady Bassel Mouris	→ NYC	3000 L.E
100000003	Kirolos	→ EGP	3000 L.E
100000004	Ahmed Mohamed	→ EGP	3000 L.E

Figure 3: Tickets of receipt



Expenses

#	Rkam ezn sarf	Price	Description	Destintaion
	222	2000 L.E	....	Name

Revenue

#	Rkam ezn sarf	Price	Description	Destintaion
	222	2000 L.E	....	Name

Figure 4: safe

This prototype helped us better understand the requriments by showing it to the client and gave us the following feedback:

(a) He want more details about each tickets in tickets of receipt page

(b) he want to merge the 3 tables in profile page in one table

(c) he want to merge the 2 tables in on table in safe page

4.1.2 CLI

git

(a) to pull git pull

(b) to add files git add . or git add *

(c) to make a commit git commit -m ”what sort of commit”

(d) to push to server git push -u origin master

(e) to get status git status

5 Design Constraints

Any device that has browser and must has connection with the internet

6 Other non-functional attributes

6.1 Security

The username and password should be encrypted and the data transmitted to database should be saved securely.

6.2 Reliability

- Speed is an important feature in the system as it should provide the client with real-time notifications to notify them on any actions taken and confirm their actions.

- The user should be able to trust the system as it aims for high accuracy to get the best and accurate results for reports.

6.3 Maintainability

The system ensures ease of maintainability through the implementation of MVC using Laravel Framework [3] . It should be easy to maintain to minimize the amount of changes that would be done to the code.

6.4 Portability

This feature is applied by implementing a responsive website using mdbootstrap [1] that allows any user to use the system on any web browser from any device.

6.5 Usability

- Learnability: Proportion of functionalities or tasks mastered doesn't need time to be learned
- Memorability: This system is easy to be memorized due to the small number of tasks the user will do

7 Preliminary Object-Oriented Domain Analysis

7.1 Class Diagram

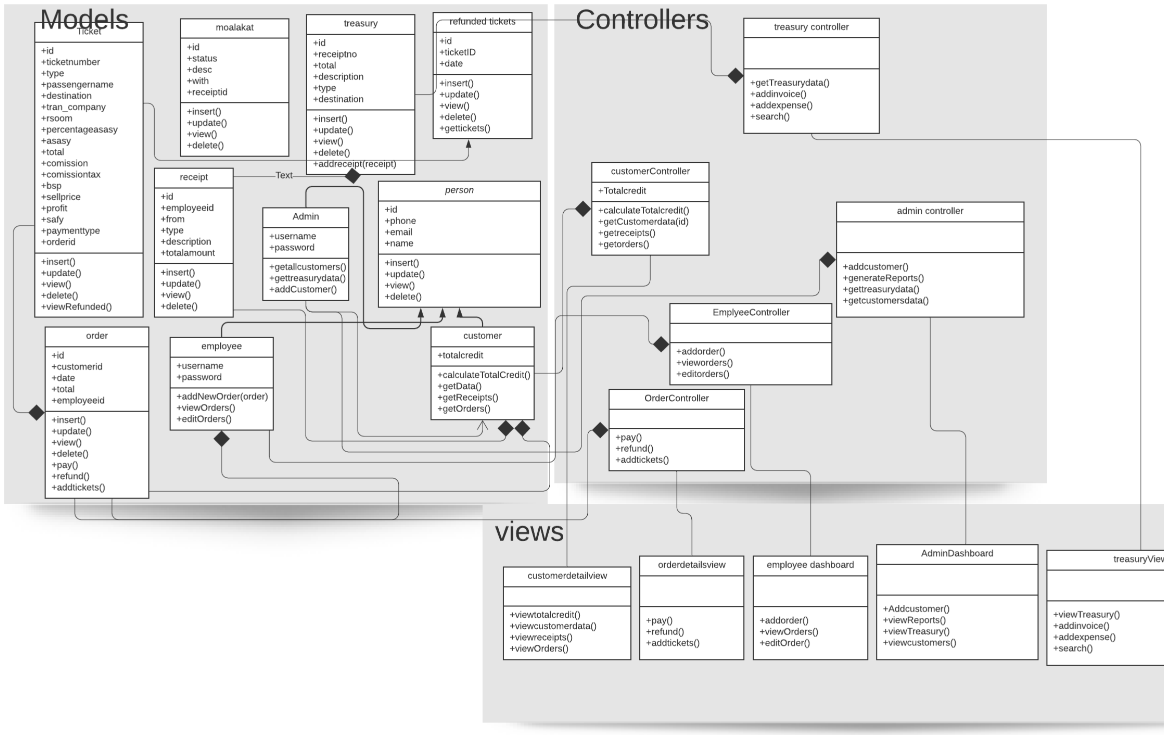


Figure 5: Class Diagram

7.2 Database Schema

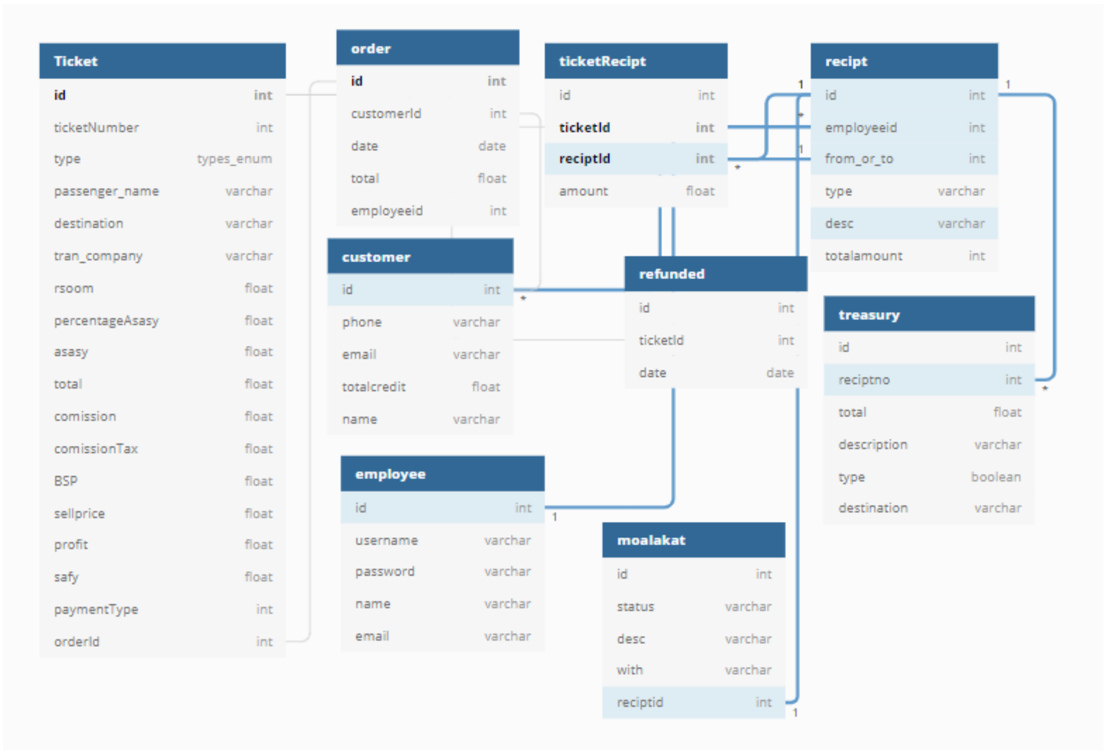


Figure 6: Database Schema

7.3 Class descriptions

Table 1: Ticket Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	Generating and retrieving ticket info.
Collaborations	-
Attributes	id, ticketnumber, type, passengername, destination, trancompany, rsoom, percentageasasy, asasy, total, comission, comissiontax, bsp, sellprice, profit, safy, paymenttype, orderid.
Operations	viewRefunded
Constraints	-

Table 2: safe Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	Retrieving receipts data.
Collaborations	receipt class to add its data to the total safe data.
Attributes	id, receiptno, total, description, type, destination.
Operations	addreceipt(receipt)
Constraints	-

Table 3: moalakat Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	Generating and retrieving moalakat data.
Collaborations	Names each class with which this class must interact in order to accomplish its purpose, and how.
Attributes	id, status, desc, with, receiptid.
Operations	-
Constraints	-

Table 4: refunded tickets Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	Retrieving refunded tickets data.
Collaborations	-
Attributes	id, ticketID, date.
Operations	-
Constraints	-

Table 5: person Class

Abstract or Concrete:	Abstract.
List of Superclasses	-
List of Subclasses	Admin, employee, customer.
Purpose	Generating and retrieving person data.
Collaborations	-
Attributes	id, phone, email, name.
Operations	-
Constraints	-

Table 6: order Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	Generating and retrieving order data.
Collaborations	Ticket class, to have many tickets in the order.
Attributes	id, customerid, date, total, employeeid.
Operations	pay, refund, addtickets.
Constraints	-

Table 7: receipt Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	Generating and retrieving receipt data.
Collaborations	-
Attributes	id, employeeid, from, type, description, totalamount.
Operations	-
Constraints	-

Table 8: Admin Class

Abstract or Concrete:	Concrete.
List of Superclasses	Person.
List of Subclasses	-
Purpose	Generating and retrieving admin data.
Collaborations	-
Attributes	username, password.
Operations	getallcustomers, gettrasurydata, addCustomer.
Constraints	Person class should be abstract.

Table 9: employee Class

Abstract or Concrete:	Concrete.
List of Superclasses	Person.
List of Subclasses	-
Purpose	Generating and retrieving employee data.
Collaborations	order class, to be able to view and edit the orders.
Attributes	username, password.
Operations	addneworder, viewOrders, editorders.
Constraints	Person class should be abstract.

Table 10: customer Class

Abstract or Concrete:	Concrete.
List of Superclasses	Person.
List of Subclasses	-
Purpose	Generating and retrieving customer data.
Collaborations	receipt class, to create and retrieve receipts. Order class, to retrieve orders history.
Attributes	totalcredit.
Operations	calculatetotalcredit, getdata, getreceipts, getorders.
Constraints	Person class should be abstract.

Table 11: safe controller Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	Control safe data.
Collaborations	safe Class, to be able to retrieve the data and write over it.
Attributes	-
Operations	getsafedata, addinvoice, addexpense, search.
Constraints	-

Table 12: customerController Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	Control customer data.
Collaborations	Customer Class, to be able to retrieve the data and write over it.
Attributes	Totalcredit.
Operations	calculateTotalcredit, getCustomerdata, getreceipts, getorders.
Constraints	-

Table 13: admin controller Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	Control admin data.
Collaborations	Admin Class, to be able to retrieve the data and write over it.
Attributes	-
Operations	addcustomer, generateReports, getsafedata, getcustomersdata.
Constraints	-

Table 14: EmployeeController Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	Control employee data.
Collaborations	Employee Class, to be able to retrieve the data and write over it.
Attributes	-
Operations	addorder, vieworders, editorders.
Constraints	-

Table 15: OrderController Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	Control order data.
Collaborations	Order Class, to be able to retrieve the data and write over it.
Attributes	-
Operations	pay, refund, addtickets.
Constraints	-

Table 16: customerdetailview Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	View/retrieve customer data.
Collaborations	-
Attributes	-
Operations	viewtotalcredit, viewcustomerdata, viewreceipts, viewOrders.
Constraints	-

Table 17: orderdetailsview Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	View/retrieve order data.
Collaborations	-
Attributes	-
Operations	pay, refund, addtickets.
Constraints	-

Table 18: employee dashboard Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	View/retrieve employee data.
Collaborations	-
Attributes	-
Operations	addorder, viewOrders, editOrder.
Constraints	-

Table 19: AdminDashboard Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	View/retrieve admin data.
Collaborations	-
Attributes	-
Operations	Addcustomer, viewReports, viewsafe, viewcustomers.
Constraints	-

Table 20: safeView Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	View/retrieve safe data.
Collaborations	-
Attributes	-
Operations	viewsafe, addinvoice, addexpense, search.
Constraints	-

8 Operational Scenarios

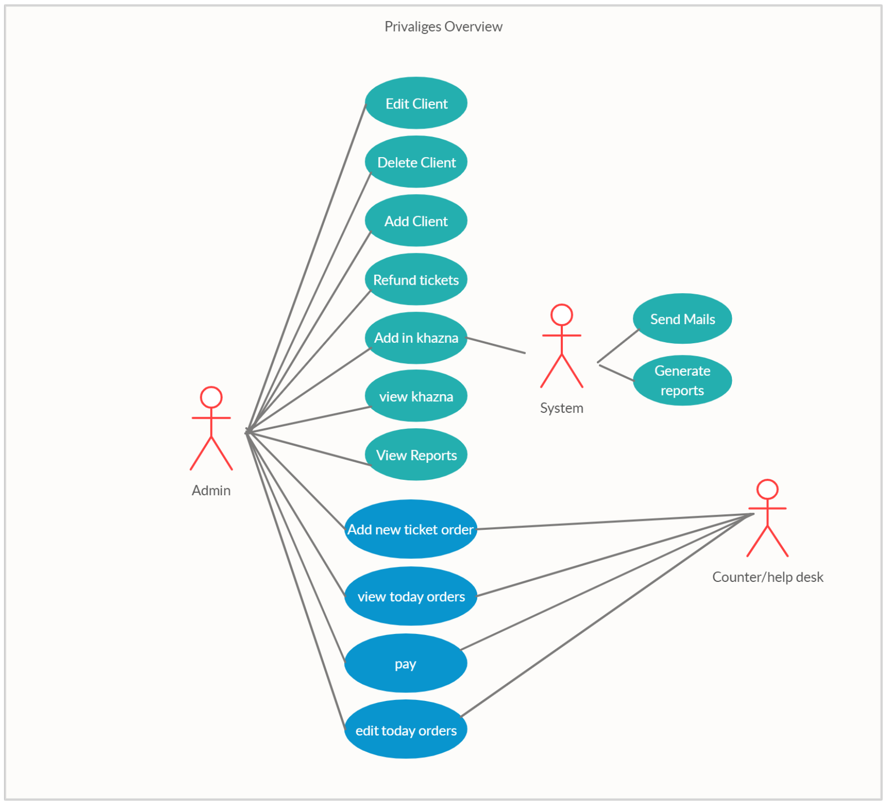


Figure 7: use case diagram

8.1 order scenario:

on making a new tickets order first thing we check if the bill is for a registered user or not if yes then the tickets are created and the bill is generated and the customer profile is updated and the countdown deadline payment begins.But if the user is not a registered customer then a receipt is generated and if payment is cash it is automatically submitted in the treasury.

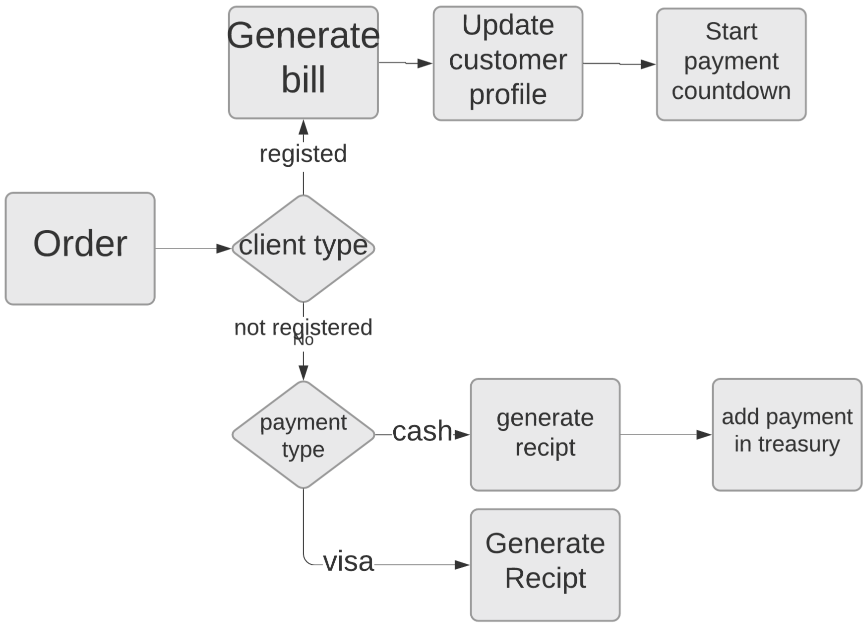


Figure 8: order scenario

8.2 payment scenario:

when a registered user comes to pay a bill he chooses to pay for the whole bill or a part and this amount is reflected in the customers profile and if the payment is cash it reflects in the treasury as a revenue.

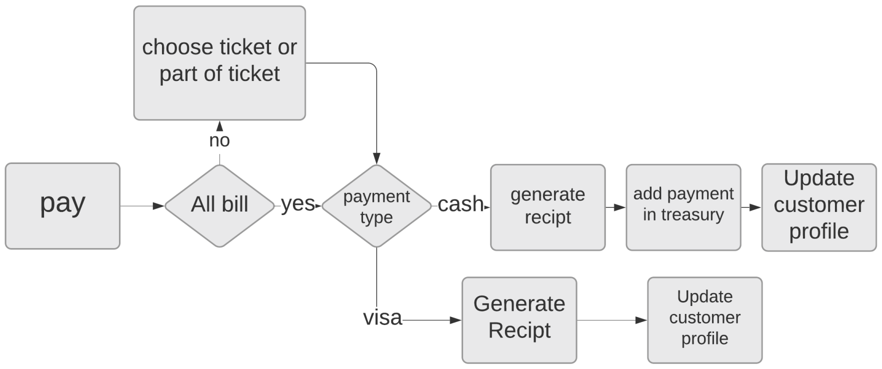


Figure 9: Payment scenario

8.3 Refund scenario:

the refund could be for the whole bill or for selected tickets the refund updates in the treasury as a expense and reflects in the customers profile if he is registered.

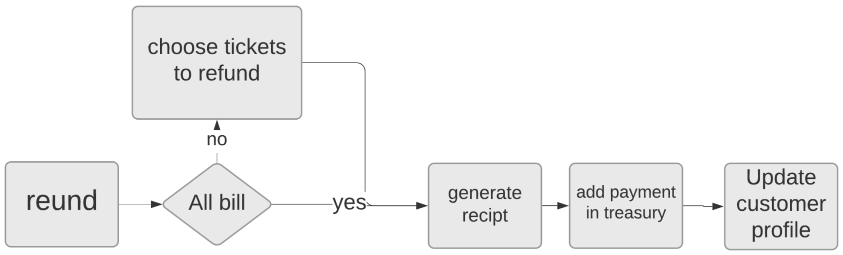


Figure 10: Refund scenario

9 Project Plan

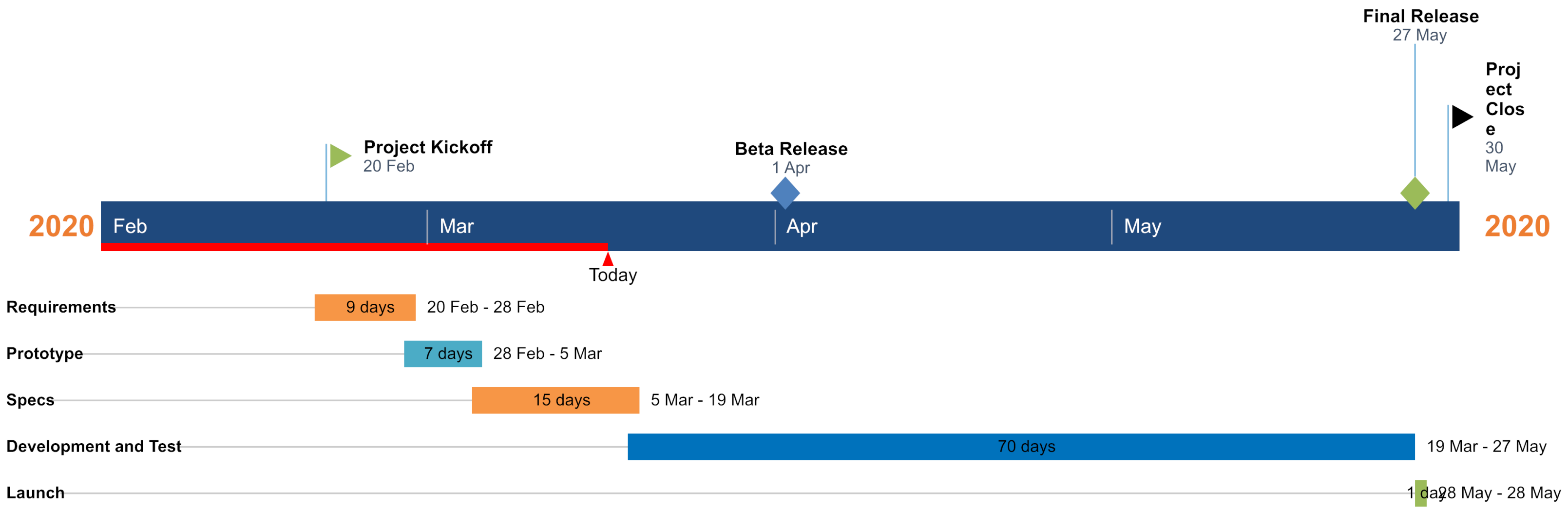
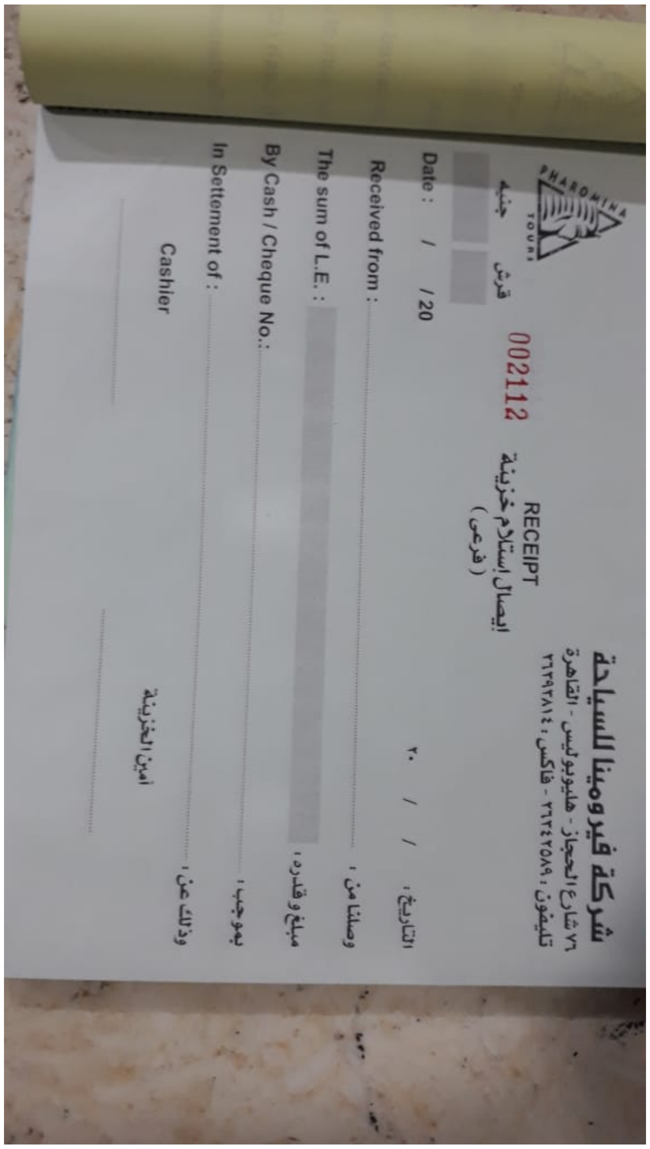


Figure 11: project plan

10 Appendices

10.1 Collected material



شركة فيرومينا للسياحة

٢٦ شارع الحجاز - هليوبوليس - القاهرة

تليفون : ٢٦٢٤٢٥٨٩ - فاكس : ٢٦٢٩٢٨١٤

PHAROMINA TOURS

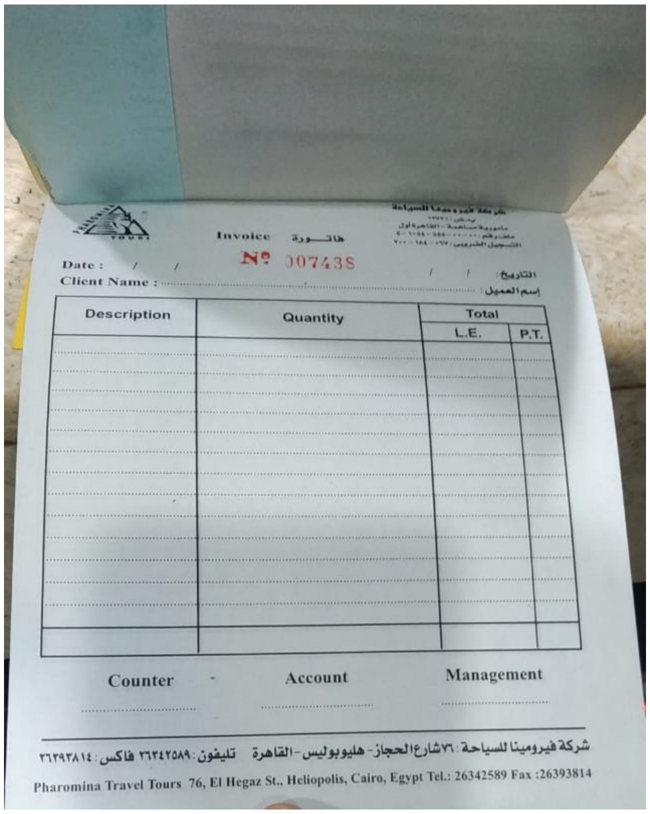
RECEIPT

إيصال إستلام خزينة
(فرعى)

002112

جنيه	قرش	
Date : / / 20		التاريخ : / / ٢٠
Received from :		وصلنا من :
The sum of L.E. :		مبلغ وقدره :
By Cash / Cheque No.:		بموجب :
In Settement of :		وذلك عن :
Cashier		أمين الخزينة

Figure 12: Current used receipt



شركة فيرومينا للسياحة
[illegible]

Invoice فاتــــورة

N° 00743S

Date : / / التاريخ: / /

Client Name : .... :إسم العميل

Description	Quantity	Total	
		L.E.	P.T.

Counter ....

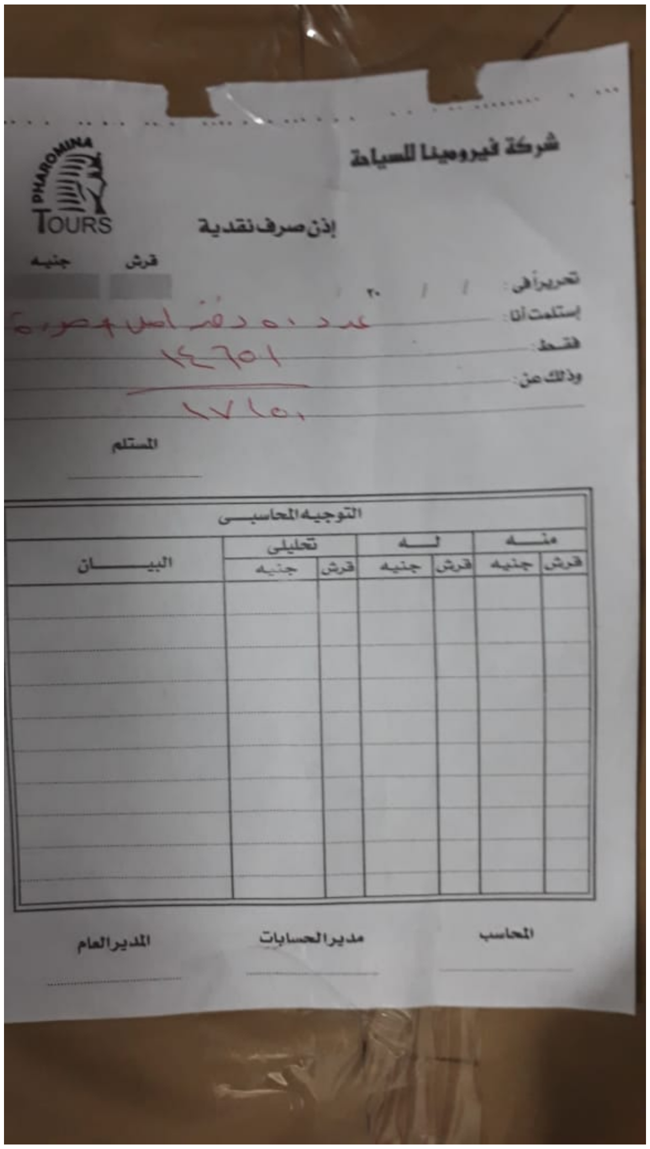
Account ....

Management ....

شركة فيرومينا للسياحة : ٧٦شارع الحجاز- هليوبوليس -القاهرة   تليفون : ٢٦٣٤٢٥٨٩ فاكس : ٢٦٣٩٣٨١٤

Pharomina Travel Tours 76, El Hegaz St., Heliopolis, Cairo, Egypt Tel.: 26342589 Fax :26393814

Figure 13: Current Used invoice



شركة فيرومينا للسياحة

PHAROMINA TOURS

إذن صرف نقدية

قرش	جنيه

تحريراً في : / / ٢٠

إستلمت أنا : عدد ١٠ دفاتر اصل وصورة

فقط : ٢٤٦٥١

وذلك عن :

١٧١٥٠

المستلم

التوجيه المحاسبى						
منه		لـه		تحليلى		البيان
قرش	جنيه	قرش	جنيه	قرش	جنيه	

المحاسب

مدير الحسابات

المدير العام

Figure 14: currently used permission form to get money from safe

References

[1] "Material design for bootstrap 4 - the most popular free ui kit." [Online]. Available: https://mdbootstrap.com/

[2] "Online travel aency software - back office solutions." [Online]. Available: http://travelworkssolution.com/en/

[3] “The php framework for web artisans.” [Online]. Available: https://laravel.com/

[4] J. Lee and J. Lee, “Paper in a digital world: Time to eliminate the inefficiency and waste,” Dec 2016. [Online]. Available: https://www.cio.com/article/3149529/paper-in-a-digital-world-time-to-eliminate-the-inefficiency-and-waste.html