Software Requirement Specification Document

Ahmed Emad, Mohamed Abdelhamed , Fady Bassel, Mark Refaat

May 10, 2020

1 Introduction

Version 2	Update SRS with latest information in GUI pictures, Functional Requirements tables, Class Descriptions, Class Diagram, and Database Schema.
Version 1	Create SRS with the initial requirements.
GitHub Link	github.com/markRefaat/tourism-company-project/tree/develop

1.1 Purpose of this document

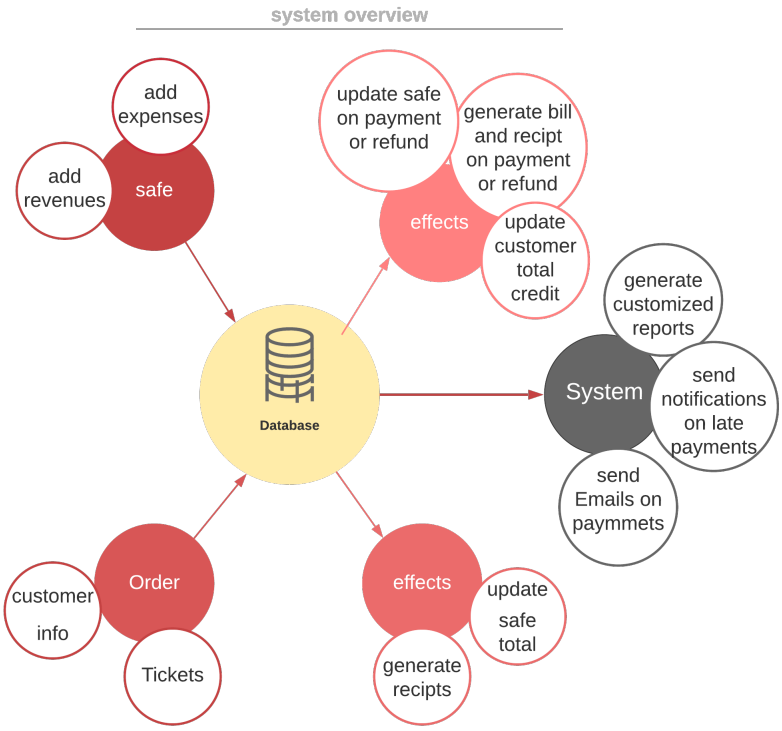
The main purpose of this Software Requirements Specification document is to illustrate and outline the requirements for our project (local system for a tourism company) that are mainly tracking customers data, booking flight tickets and managing the company's finances. Our aim is to facilitate the work 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.

1.2 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.

1.3 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 its customers into two categories regular and ordinary customers, 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/revenues and see detailed general and custom reports generated by the system, also tickets information should be added automatically in the revenues upon payment.



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

Customer Model:

Function	orders
ID	F00
Description	This Function return orders of specific customer
Input	-
Action	It returns array of orders for selected customer from the database
Output	Array of orders
Precondition	user must select customer first
Post-condition	Return to orders page
Dependencies	-

Function	receipts
ID	F01
Description	This Function return receipts of specific customer
Input	-
Action	It returns array of receipts for selected customer from the database
Output	Array of receipts
Precondition	user must select customer first
Post-condition	Return to receipts page
Dependencies	-

Destination Model:

Function	receipts
ID	F02
Description	This Function return receipts of specific destination
Input	-
Action	It returns array of receipts for selected destination from the database
Output	Array of receipts
Precondition	user must select destination first
Post-condition	Return to receipts page
Dependencies	-

Order Model:

Function	tickets
ID	F03
Description	This Function return tickets of specific order
Input	-
Action	It returns array of tickets for selected order from the database
Output	Array of tickets
Precondition	user must select order first
Post-condition	Return to tickets page
Dependencies	-

Function	customer
ID	F04
Description	This Function return the customer of specific order
Input	-
Action	It returns customer data for selected order from the database
Output	Customer object
Precondition	user must select destination first
Post-condition	Return to customer page
Dependencies	-

Function	ticketsAmount
ID	F05
Description	This Function return the order data and total price and payed amount of specific order
Input	-
Action	It returns array of 3 (order data, total amount, payed amount) for selected order from the database
Output	Array
Precondition	user must select order first
Post-condition	-
Dependencies	-

Receipt Model:

Function	tickets
ID	F06
Description	This Function return the tickets belong to specific order
Input	-
Action	It returns array of tickets for selected receipt from the database
Output	Array of tickets
Precondition	user must select receipt first
Post-condition	Return to tickets page
Dependencies	-

Function	safe
ID	F07
Description	This Function return the safe belong to specific order
Input	-
Action	It returns safe object that this receipt belongs to from the database
Output	Safe Object
Precondition	user must select receipt first
Post-condition	Return to safe page
Dependencies	-

Role Model:

Function	users
ID	F08
Description	This Function return the all users that assigned to specific role
Input	-
Action	It returns all users object for role from the database
Output	Array of users
Precondition	user must login first
Post-condition	-
Dependencies	-

Safe Model:

Function	receipts
ID	F09
Description	This Function return array of receipts of specific safe
Input	-
Action	It returns array of receipts for selected safe from the database
Output	Array of receipts
Precondition	user must select safe first
Post-condition	Return to receipts page
Dependencies	-

Ticket Model:

Function	receipts
ID	F10
Description	This Function return array of receipts that this ticket belongs to
Input	-
Action	It returns array of receipts for selected ticket from the database
Output	Array of receipts
Precondition	user must select ticket first
Post-condition	Return to receipts page
Dependencies	-

Function	orders
ID	F11
Description	This Function return order of specific ticket
Input	-
Action	It returns order object for selected ticket from the database
Output	Order Object
Precondition	user must select ticket first
Post-condition	Return to orders page
Dependencies	-

Function	calculate
ID	F12
Description	This Function return the status of the ticket(full payed or not and the payed amount and the left amount
Input	-
Action	It returns array of (status , payed , left) for selected ticket from the database
Output	Array
Precondition	user must select ticket first
Post-condition	Return to tickets page
Dependencies	-

User Model:

Function	role
ID	F13
Description	This Function return the role of the logged in user
Input	-
Action	It returns role object for the auth user in the system form the database
Output	Role Object
Precondition	user must login in first
Post-condition	-
Dependencies	-

Customer Controller:

Function	index
ID	F14
Description	This Function return the required data for all customers page
Input	-
Action	It returns array of all customers and counts of all customers, ongoingpayments and latepayments from the database
Output	Array of all customer and Array of counts
Precondition	-
Post-condition	Return to all customers page
Dependencies	-

Function	ongoingpayments
ID	F15
Description	This Function return the required data for ongoingpayments page
Input	-
Action	It returns array of ongoing orders and counts of all customers, ongoingpayments and latepayments from the database
Output	Array of ongoing orders and Array of counts
Precondition	-
Post-condition	Return to ongoingpayments page
Dependencies	-

Function	latepayments
ID	F16
Description	This Function return the required data for latepayments page
Input	-
Action	It returns array of late orders and counts of all customers, ongoingpayments and latepayments from the database
Output	Array of late orders and Array of counts
Precondition	-
Post-condition	Return to latepayments page
Dependencies	-

Function	store
ID	F17
Description	This Function to insert new customer
Input	-
Action	It validate new customer data and insert it in the database
Output	-
Precondition	-
Post-condition	Return to all customers page
Dependencies	-

Function	show
ID	F18
Description	This Function to show customer data
Input	CustomerID
Action	It finds the customer in the database referring to his id and return his profile data and all his orders and his receipts (expenses and revenues)
Output	Customer Object and his orders, expenses and revenues
Precondition	-
Post-condition	Return to customer profile page
Dependencies	-

Function	updatenote
ID	F19
Description	This Function to update notes of customer
Input	CustomerID
Action	It finds the customer in the database referring to his id and return update his note with the new note
Output	-
Precondition	-
Post-condition	Return to customer profile page
Dependencies	-

Function	edit
ID	F20
Description	This Function to open edit customer page
Input	CustomerID
Action	It returns customer data to edit customer profile page
Output	Customer object
Precondition	-
Post-condition	Return to edit customer profile page
Dependencies	-

Function	update
ID	F21
Description	This Function to update customer data
Input	CustomerID, Request Object
Action	It finds the customer in the database referring to his id and return update his data with the new data
Output	-
Precondition	-
Post-condition	Return to all customers page
Dependencies	-

Home Controller:

Function	index
ID	F22
Description	This Function return the required data for home page
Input	-
Action	It returns array of counts of customers, tickets, refundedTickets and latepayments from the database
Output	Array of counts
Precondition	-
Post-condition	Return to home page
Dependencies	-

Mail Controller:

Function	Mail Admin
ID	F23
Description	This functions send mail to the superadmin/admin privileged users saved in database
Input	Mail subject and mail content
Action	Check if the mail is sent to the player else error message will appear
Output	HTML mail is sent
Precondition	Receiver must be existed in the database
Post-condition	Mail is created and sent
Dependencies	-

Function	Mail Helpdesk
ID	F24
Description	This function send mail to the helpdesk privileged users saved in database
Input	Mail subject and mail content
Action	Check if the mail is sent to the player else error message will appear
Output	HTML mail is sent
Precondition	Receiver must be existed in the database
Post-condition	Mail is created and sent
Dependencies	-

Function	Mail
ID	F25
Description	This function send mail to the users saved in database for test purposes
Input	None
Action	Check if the mail is sent to the player else error message will appear
Output	HTML mail is sent
Precondition	Receiver must be existed in the database
Post-condition	Mail is created and sent
Dependencies	-

Notifications Controller:

Function	Notify
ID	F26
Description	This Function is Laravel built in, for the system to send notifications to the user
Input	Notification object
Action	Check if the notification is sent to the customer else error message will appear
Output	String notification is sent
Precondition	User must be existed in the database
Post-condition	Notification is created and inserted in the database.
Dependencies	-

Function	Mark All As Read
ID	F27
Description	This Function marks all user notifications as read
Input	-
Action	Marking all the notification in the database as read for the user
Output	-
Precondition	check if the notifications exist and unread
Post-condition	-
Dependencies	-

Function	Unread
ID	F28
Description	This Function is for listing all the unread notifications in the database for the user
Input	-
Action	Retrieving all the unread notification in the database for the user
Output	Array of notifications
Precondition	check if the notifications exist
Post-condition	-
Dependencies	-

Function	View All
ID	F29
Description	This Function is for listing all the notifications in the database for the user
Input	-
Action	Retrieving all the notification either read or unread in the database for the user
Output	Array of notifications
Precondition	check if the notifications exist
Post-condition	-
Dependencies	-

Order Controller:

Function	index
ID	F30
Description	This Function return the required data for all orders page
Input	-
Action	It returns array of all orders from the database or array of orders in progress and today's date
Output	Array of orders and today's date
Precondition	-
Post-condition	Return to all orders page
Dependencies	-

Function	create
ID	F31
Description	This Function to go to create order page
Input	-
Action	It returns array of all customers to create orders page
Output	Array of customers
Precondition	-
Post-condition	Return to create order page
Dependencies	-

Function	store
ID	F32
Description	This Function to insert new order
Input	Request Object
Action	It validate new order data and check if the user is cash or later to put the customer id and insert it in the database
Output	-
Precondition	-
Post-condition	Return to ticket create page
Dependencies	-

Function	show
ID	F33
Description	This Function to show order data
Input	Order Object
Action	It finds the order in the database referring to its id and return its data, tickets, total price, payed amount(ticketsAmount). it deletes the order if empty
Output	order object, order tickets, total price and payed amount
Precondition	-
Post-condition	Return to order show page or all orders page if the order is empty
Dependencies	F05

Function	confirmpayment
ID	F34
Description	This Function to confirm order payment
Input	Order Object, String total
Action	It finds the customer in the database referring to his id and decrease his total credit with the total of the order
Output	-
Precondition	-
Post-condition	Return to order show page
Dependencies	-

Function	confirmview
ID	F35
Description	return view for order edit before confirmation
Input	Order Object
Action	It returns order confirmation view
Output	-
Precondition	Order placed and status is still unpaid
Post-condition	-
Dependencies	F05

Function	destroy
ID	F36
Description	This Function to delete order
Input	orderID
Action	It finds the order in the database referring to its id and delete it if the user is admin or superadmin
Output	-
Precondition	-
Post-condition	Return to all orders page
Dependencies	-

Function	print
ID	F37
Description	This Function to download and print invoice for order
Input	Order Object
Action	It create invoice.pdf with all order data and download it
Output	PDF
Precondition	-
Post-condition	-
Dependencies	-

Function	payAll
ID	F38
Description	This Function to return confirmation of pay all order
Input	Order Object
Action	get all unpaid tickets of order,It is shortcut to pay all tickets of order without need to select everyone
Output	view of order payment confirmation
Precondition	order with unfinished payments.
Post-condition	-
Dependencies	F05

Receipt Controller:

Function	store
ID	F39
Description	This Function to insert new receipt
Input	Order Object, String Total
Action	It creates new receipt and add fill its data from auth user, order object and save it in database and download invoice.pdf with details
Output	PDF
Precondition	-
Post-condition	-
Dependencies	-

Function	storeAllOrder
ID	F40
Description	This Function to insert new receipt
Input	Order Object, String Total
Action	It is shortcut to create receipt for paying all order and save it in database and download invoice.pdf with details
Output	PDF
Precondition	-
Post-condition	-
Dependencies	-

Function	refundTickets
ID	F41
Description	This Function to insert new receipt
Input	Request Object
Action	It finds the tickets id in request object and refund them and update order total price
Output	-
Precondition	-
Post-condition	Return to order page
Dependencies	-

Function	print
ID	F42
Description	This Function to download and print invoice for receipt
Input	Receipt Object
Action	It create invoice.pdf with all order data and download it
Output	PDF
Precondition	-
Post-condition	-
Dependencies	-

Report Controller:

Function	getEndDate
ID	F43
Description	This Function to for report
Input	Date startingdate
Action	It returns ending report date after n days (report period in settings) from starting date
Output	Date endingdate
Precondition	user must select startdate in report page
Post-condition	-
Dependencies	-

Function	tickets
ID	F44
Description	This Function to get all tickets and their orders
Input	-
Action	It returns every ticket with order details and get report period from settings
Output	tickets, period
Precondition	-
Post-condition	Return to tickets report page
Dependencies	-

Function	printTickets
ID	F45
Description	This Function to print tickets report
Input	Date startingdate
Action	It generate report data according to start and end date
Output	PDF
Precondition	-
Post-condition	Return to print tickets report page
Dependencies	F39

Function	printReceipts
ID	F46
Description	This Function to print receipts report
Input	Date startingdate
Action	It generate report data according to start and end date
Output	PDF
Precondition	-
Post-condition	Return to print receipts report page
Dependencies	F39

Function	receipts
ID	F47
Description	This Function to get all receipts
Input	-
Action	It returns every receipt with customer or destination details and get report period from settings
Output	tickets, period
Precondition	-
Post-condition	Return to receipts report page
Dependencies	-

Function	excelTickets
ID	F48
Description	This Function export tickets data to excel sheet
Input	Date startingdate
Action	It downloads excel sheet with tickets data in range of startdate and enddate
Output	PDF
Precondition	-
Post-condition	-
Dependencies	-

Function	excelReceipts
ID	F49
Description	This Function export tickets data to excel sheet
Input	Date startingdate
Action	It downloads excel sheet with receipts data in range of startdate and enddate
Output	PDF
Precondition	-
Post-condition	-
Dependencies	F39

Safe Controller:

Function	index
ID	F50
Description	This Function return the required data for safe page
Input	-
Action	It returns array of all receipts from the database and array of all destinations
Output	Array of receipts and destinations
Precondition	-
Post-condition	Return to safe page
Dependencies	-

Function	store
ID	F51
Description	This Function to insert new receipt
Input	order Object,total of receipt
Action	It creates new receipt and add fill its data from auth user, order object and save it in database and download invoice.pdf with details
Output	PDF
Precondition	choose tickets to pay.
Post-condition	payment of selected tickets and receipt generation in safe
Dependencies	-

Setting Controller:

Function	index
ID	F52
Description	This Function return the required data for setting page
Input	-
Action	It returns array of settings from the database
Output	Array of setting
Precondition	-
Post-condition	Return to setting page
Dependencies	-

Function	update
ID	F53
Description	This Function update settings
Input	Request Object
Action	validate and update with new settings
Output	-
Precondition	-
Post-condition	Return to setting page
Dependencies	-

Ticket Controller:

Function	index
ID	F54
Description	This Function return the required data for all tickets page
Input	-
Action	It returns array of all tickets from the database
Output	Array of tickets
Precondition	-
Post-condition	Return to all tickets page
Dependencies	-

Function	create
ID	F55
Description	This Function return create page
Input	-
Action	It redirect to create page
Output	-
Precondition	-
Post-condition	Return to create ticket page
Dependencies	-

Function	orderticket
ID	F56
Description	This Function return create ticket page for order
Input	Order Object, Integer Status
Action	It redirect to create ticket page for order
Output	-
Precondition	-
Post-condition	Return to create ticket page
Dependencies	-

Function	store
ID	F57
Description	This Function return create ticket page for order
Input	StoreTicket Object
Action	It add ticket data to session tickets which save all order tickets
Output	-
Precondition	-
Post-condition	Return to create ticket page
Dependencies	-

Function	refundedTicketsShow
ID	F58
Description	This Function return all refunded tickets page
Input	-
Action	It select all tickets from database with type refunded
Output	-
Precondition	-
Post-condition	Return to all refunded tickets page
Dependencies	-

Function	edit
ID	F59
Description	This Function return edit page
Input	Ticket Object
Action	It returns to edit page with tickets data to edit
Output	-
Precondition	-
Post-condition	Return to edit ticket page
Dependencies	-

Function	update
ID	F60
Description	This Function update ticket data
Input	StoreTicket Object, Ticket Object
Action	It updates ticket data in the session before save it to database
Output	-
Precondition	-
Post-condition	Return to confirm order page
Dependencies	-

Function	destroy
ID	F61
Description	This Function delete ticket
Input	Ticket Object
Action	It deletes the ticket data from the order
Output	-
Precondition	-
Post-condition	Return to order confirm page
Dependencies	F31

Function	checkprice
ID	F62
Description	fills session data with tickets to be paid and and undo payments before confirm by ajax.
Input	Request Object
Action	validate amount to be payed by ticket and add it to session or remove
Output	return jso itn response to ajax call
Precondition	ajax request
Post-condition	Return to edit ticket page
Dependencies	-

Function	confirmReceipt
ID	F63
Description	This Function return payments to confirmation view
Input	Ticket Object
Action	It returns to confirm page with tickets to be payed
Output	-
Precondition	session set of payment
Post-condition	-
Dependencies	-

4 Interface Requirements

4.1 User Interfaces

4.1.1 GUI

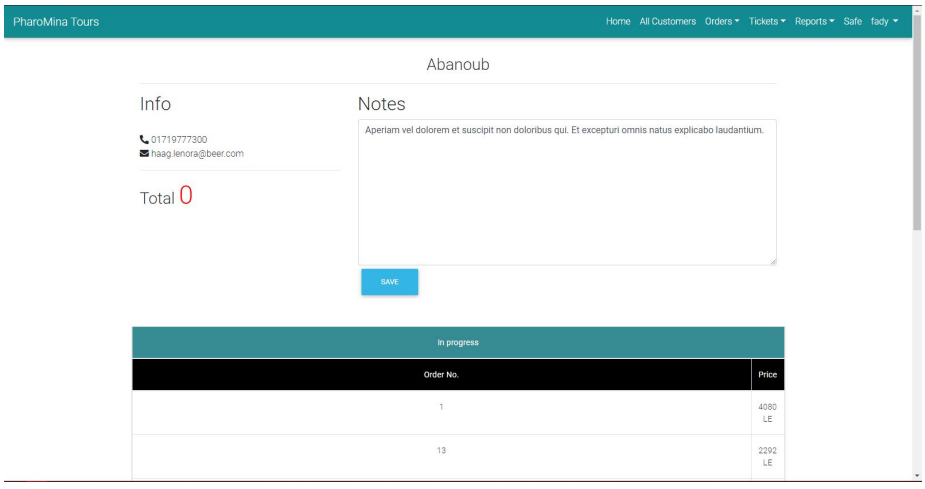


Figure 1: Customer Profile

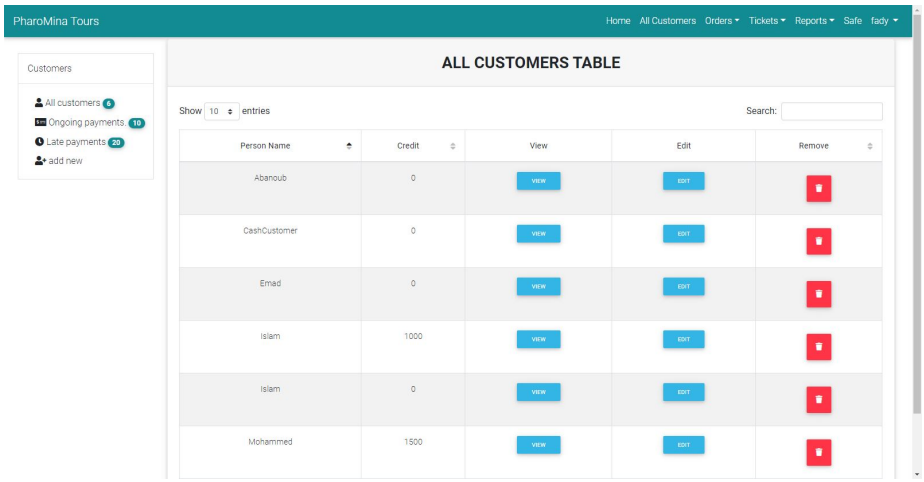


Figure 2: All Customers

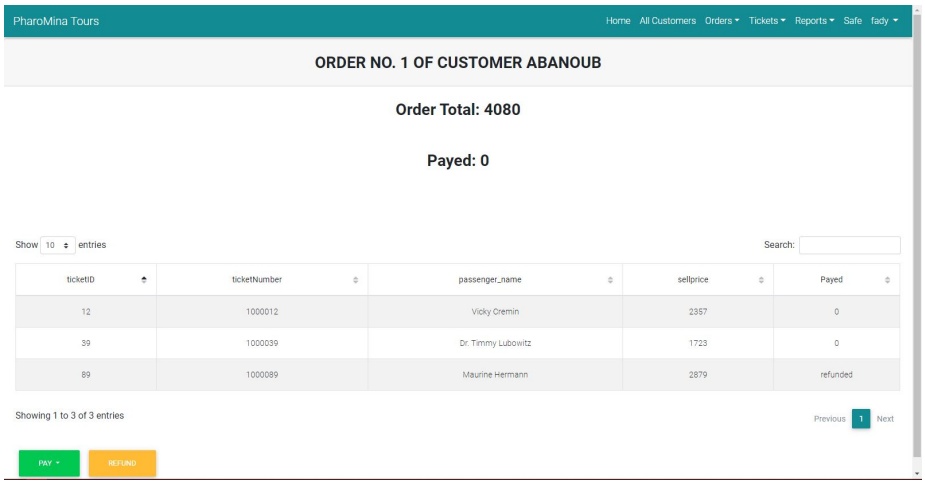


Figure 3: Order Tickets

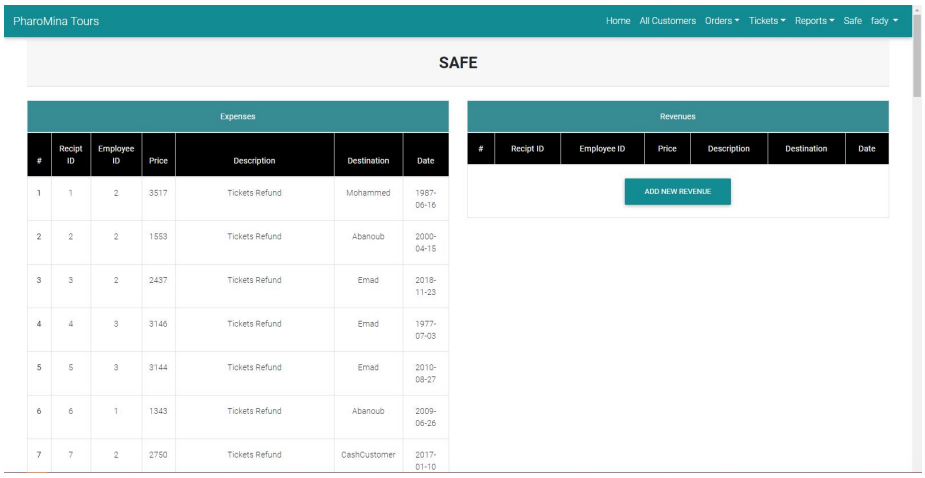


Figure 4: Safe

The initial prototype helped us better understand the requirements by showing it to the client and gave us the following feedback:

(a) He wanted more details about each ticket in tickets of receipt page.

(b) He wanted us to merge the 3 tables in profile page into one table.

(c) He wanted us to merge the 2 tables into one 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

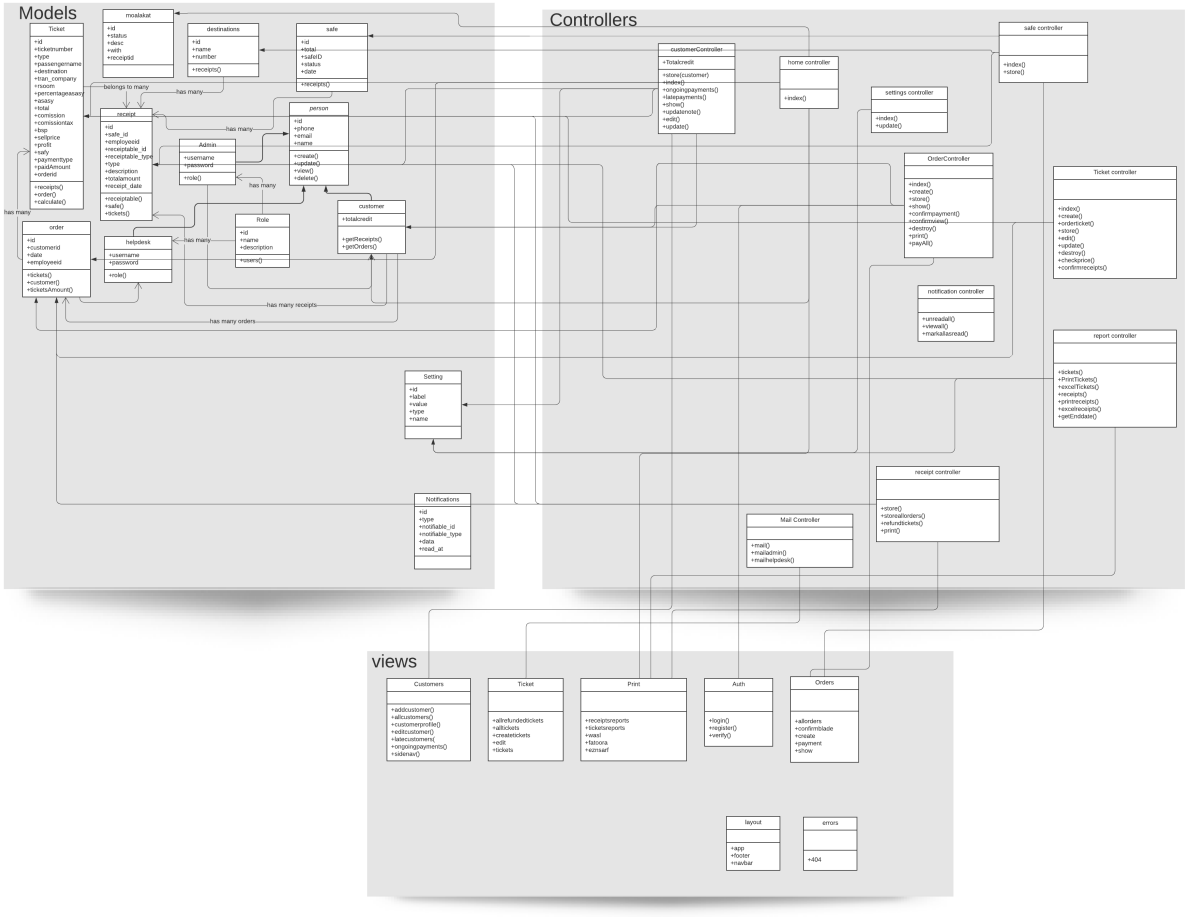


Figure 5: Class Diagram

7.2 Database Schema

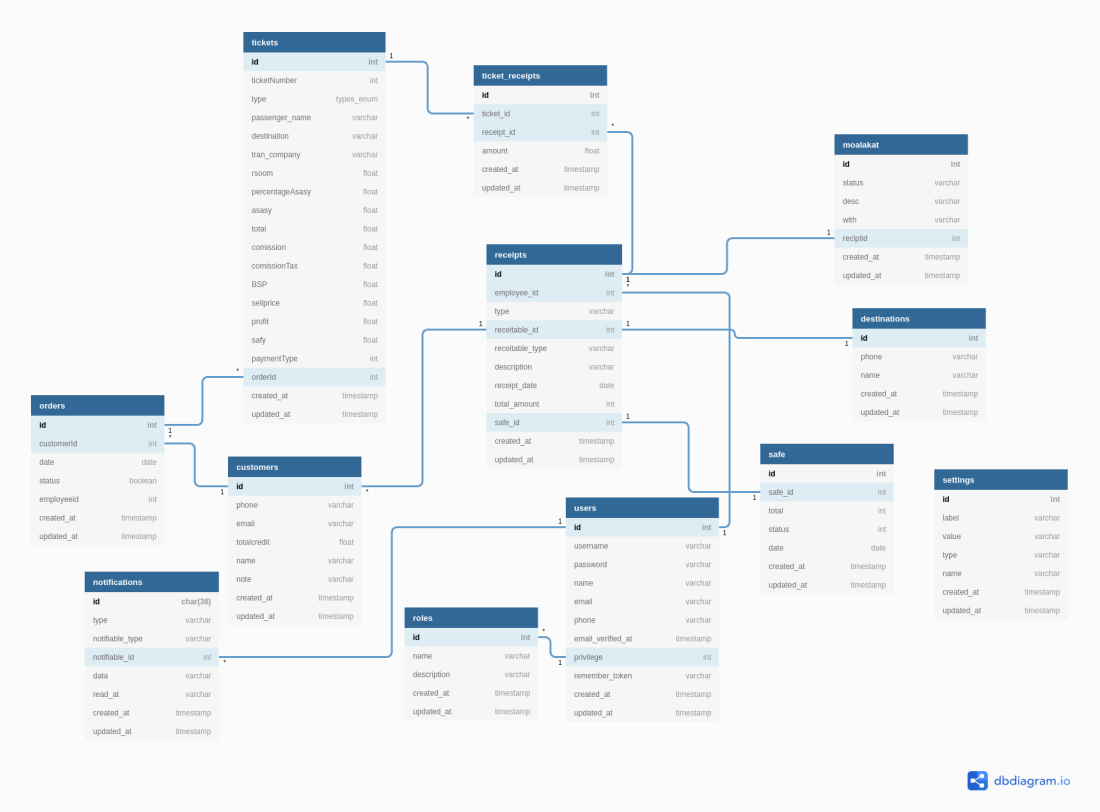


Figure 6: Database Schema

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: 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 5: 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 6: 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 7: 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 8: user 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 9: 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 10: 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 11: 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 12: 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 13: 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 14: 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 15: 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 16: 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 17: 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 18: 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 19: 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	-

Table 20: Mails Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	Send mails.
Collaborations	-
Attributes	-
Operations	SendMail.
Constraints	-

Table 21: Notifications Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	View/Create Notifications.
Collaborations	-
Attributes	-
Operations	viewNotifications, SendNotifications, MarkAsRead.
Constraints	-

Table 22: Destination Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	View/Create Destination.
Collaborations	-
Attributes	-
Operations	viewDestinations.
Constraints	-

Table 23: Role Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	View/Create Destination.
Collaborations	-
Attributes	-
Operations	viewDestinations.
Constraints	-

Table 24: Setting Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	View Settings.
Collaborations	-
Attributes	-
Operations	viewSettings.
Constraints	-

Table 25: SettingController Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	Control settings data.
Collaborations	Settings Class, to be able to retrieve the data and write over it.
Attributes	-
Operations	index, update.
Constraints	-

Table 26: ReportController Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	Control reports data.
Collaborations	TicketsExport And ReceiptsExport Class, to be able to retrieve the data and download reports.
Attributes	-
Operations	index, update.
Constraints	-

Table 27: TicketsExport Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	Retrieve tickets for download report.
Collaborations	ReportController class, send data to download it‘
Attributes	-
Operations	map, headings, collection.
Constraints	-

Table 28: ReceiptsExport Class

Abstract or Concrete:	Concrete.
List of Superclasses	-
List of Subclasses	-
Purpose	Retrieve receipts for download report .
Collaborations	ReportController class, send data to download it‘
Attributes	-
Operations	map, headings, collection.
Constraints	-

8 Operational Scenarios

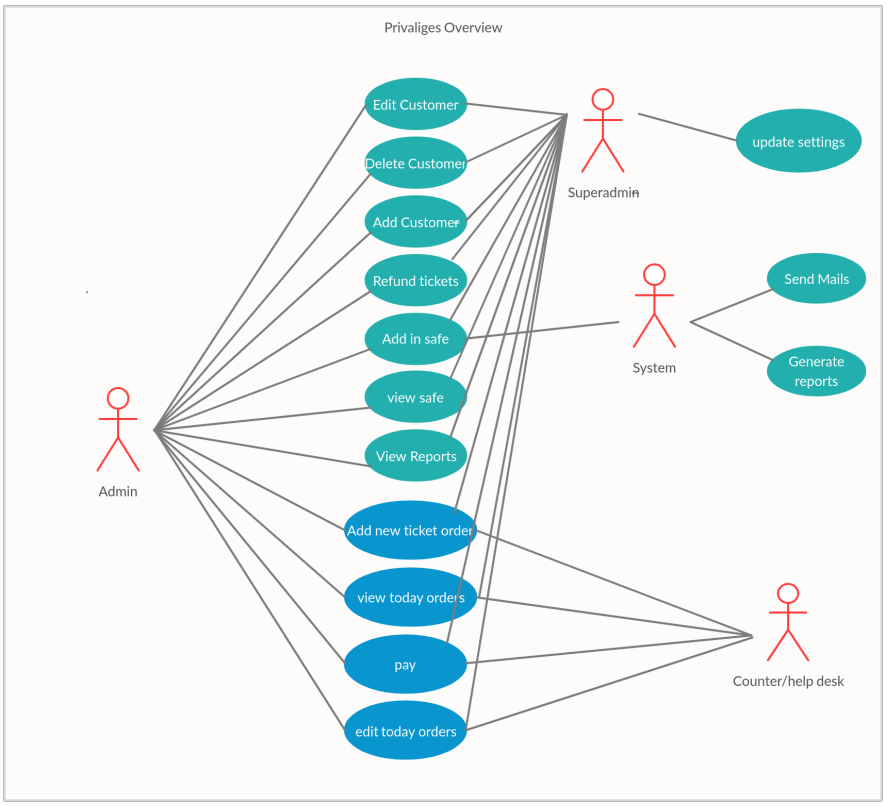


Figure 7: use case diagram

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.

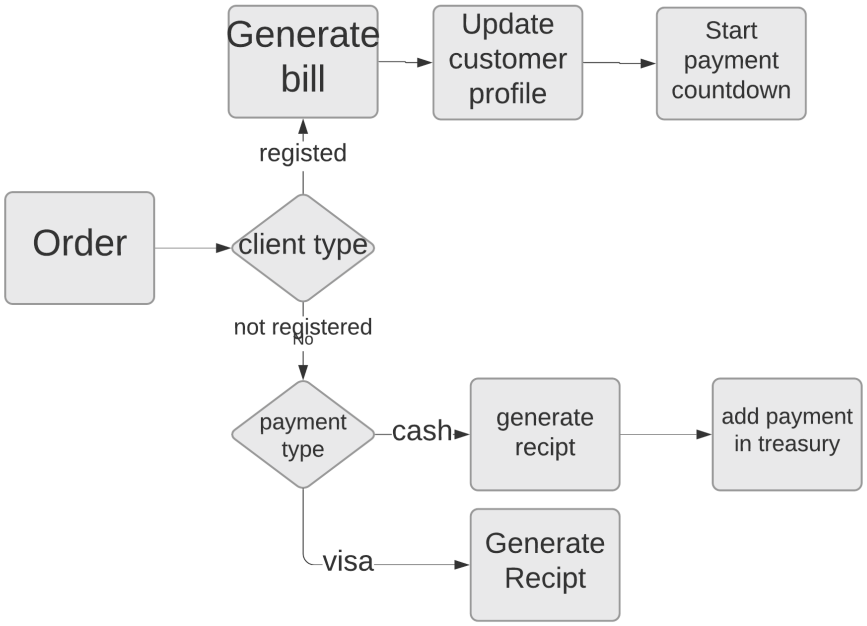


Figure 8: order scenario

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.

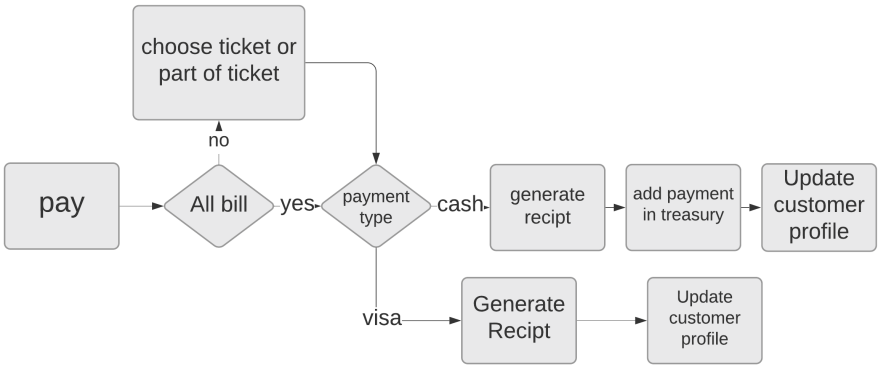


Figure 9: Payment scenario

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.

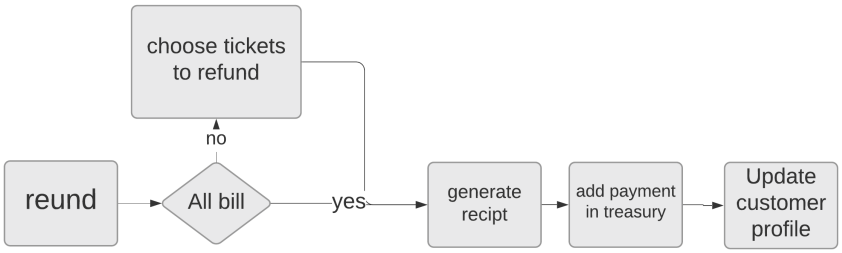


Figure 10: Refund scenario

9 Project Plan

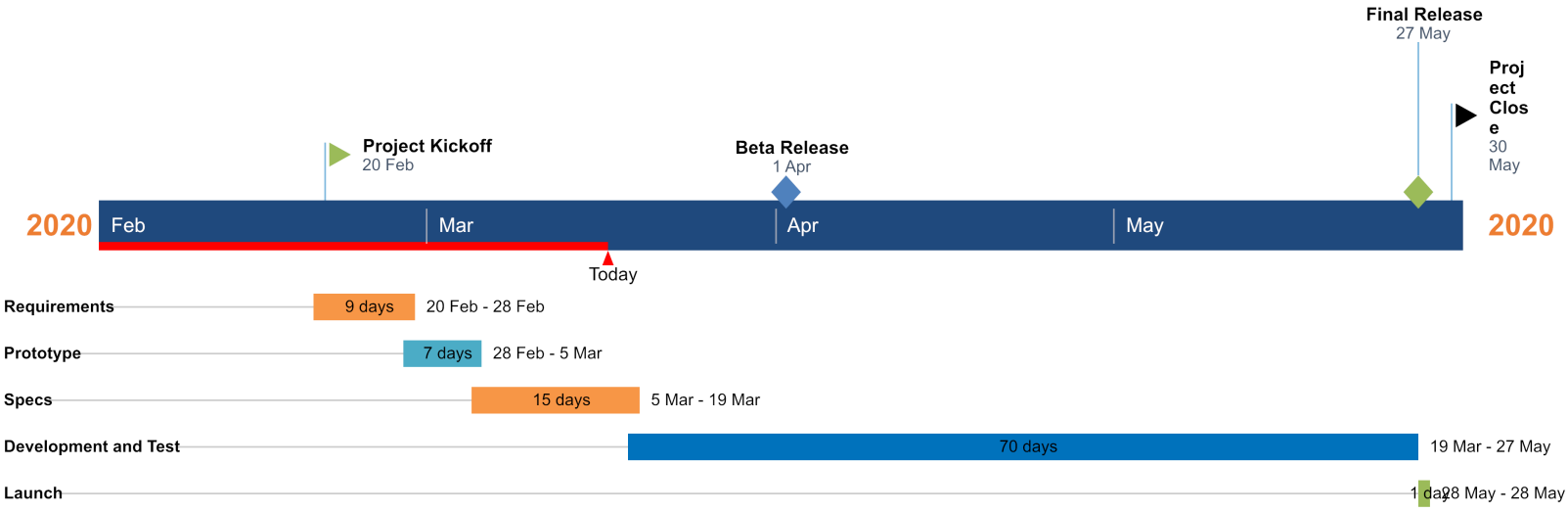


Figure 11: project plan

10 Appendices

10.1 Collected material

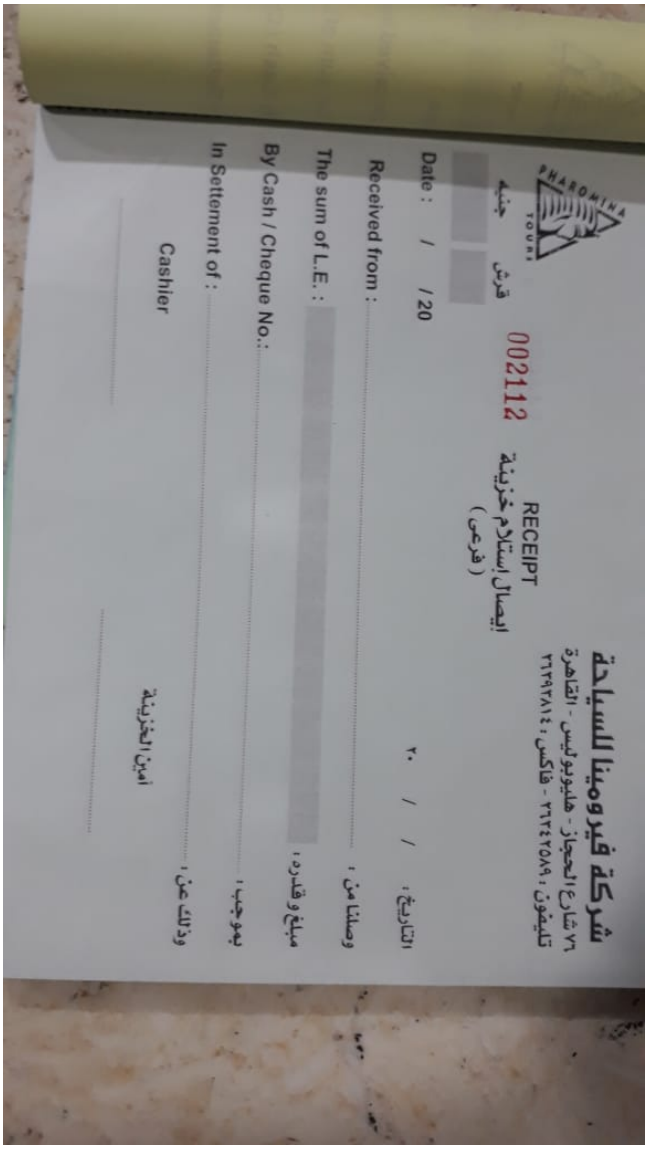


Figure 12: Current used receipt

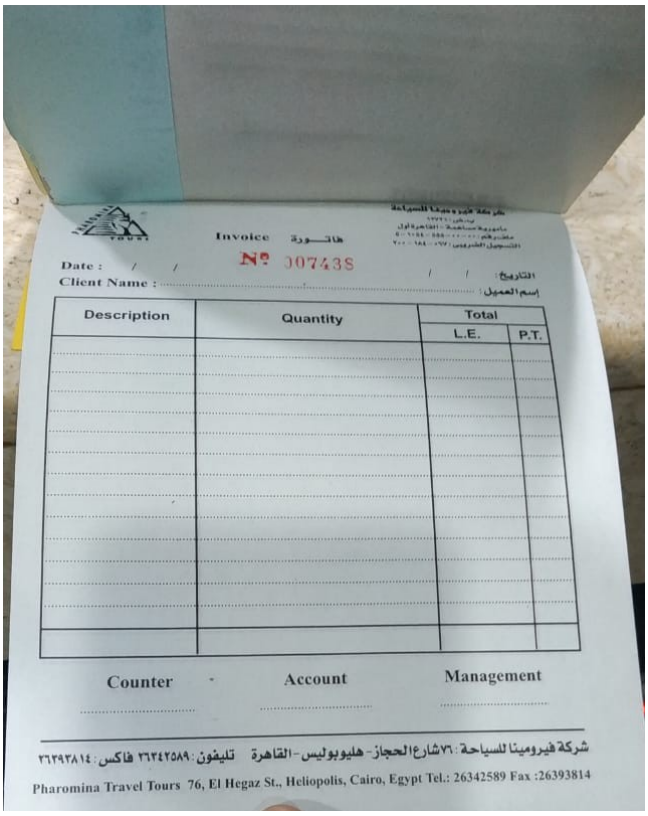


Figure 13: Current Used invoice

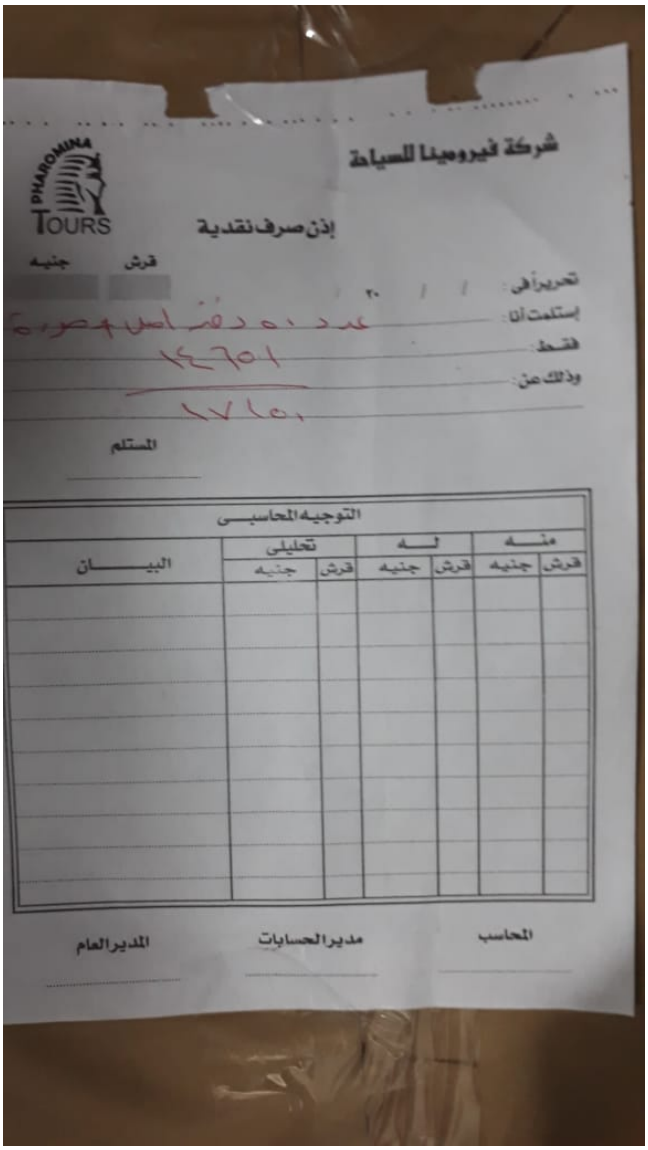


Figure 14: Current used permission form to get money from safe

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