CAP

Car accessories & spare parts system

Graduation Project, Part-II (SWE 497)
Software Engineering Department
CCIS, KSU

Project Advisor:
Dr. Fazale Amin

Submitted by
Abdullah Alshaya 439101442
Nasser Alfaraj 439103161
Abdulmajeed Alhaqbani 439102008
Hamoud Alkharji 439101000
Nawaf Alhamidi 439102199

Date submitted

21/11/2022

ABSTRACT

This document describes the process of creating a web application that focuses on saving people time with the short time people have these days, when you can order a car accessory and spare parts and have it delivered to you and installed if you want, and to act as a “middleman” between the user and the seller, making the user browse through different items from different companies that each seller displayed, with the ability to have it delivered and installed with a captain.

Table of Contents

List of Figures

List of Tables

1. Introduction

In these days when everything happened so fast and people having short time, sometime fixing small things on the car might take a lot of time many people don't have, we considered that not every person know how to acquire car parts and searching for the right part with the good company and price might take much time.
With all that together and after we notices that there is big difference between the prices in the local stores and the online stores for the spare parts, we came up with "CAP".

We propose an online ordering system for car accessories and spare parts to make life easier, the main advantage of the system is that it simplifies the ordering process for users and car companies.

In this project we plan to develop a software system that allows each user to browse the spare parts and the companies provide it with different quality and prices for every class of society, we provide an installation service for the spare parts.

There is seller account which can upload the items and all the information for it, and a captain account that can deliver and install the items.

2. Terms definition

User: who will view the items and may purchase.
Seller: who will offer the items for sale.
Captain: who will deliver the items and install it those who purchase.
Admin: who will manage the system and manage the registration of sellers and captains
Items (products): spare parts products.
Assigned order: orders which the seller changes its status from order to ready and has captain who responsible for it.
User request: Request to deliver and install the ordered items.

3. Domain Analysis

This project is concerned to apply the e-commerce [4] concept, which is emerged recently, and it exists heavily nowadays, the concept is to buy and sell goods and services over the internet that can be conducted by the people throw their computers, tablets, smartphone and other devices, and today a lot of products and services can be purchase through the internet.

This will give the users more convenience in their life since they can shop and get a look for the product prices and images and other information.

In our project we will focus on the cars accessories and spare parts selling in which the seller will offer his products in the system and the system will accept different sellers, and there will be a feature in which the user can order to deliver and install these spare parts and accessories for his requests after he purchase an accessory or spare parts by a specialist captain, and those captains have the ability to register to the system to provide these services (deliver the item and install it).

There are some websites and mobile applications have approximately the same idea of our system locally such as Speero [1], it provides the user the ability to purchase the spare parts and the user can go to some car repair shop which are the website have a contract with them to install user's items, there are also other website its name is Afyal [2] , it is also concerned with spare parts selling, after the user purchase an item it will be delivered by other shipping company, and then the user will look for installing his parts out of this system context.

TM[3] application is mobile application offers services providers for many purposes, the user can view the list of service providers and every one will show his contact info so the user contact with the service provider for his purpose and discuss with him about the intended work and the pricing , the user can contact with car mechanic who can install the spare parts, but this mobile application don't offers products for selling, this system, this system sort the sellers and service provides list depend on the nearest distance to the user.

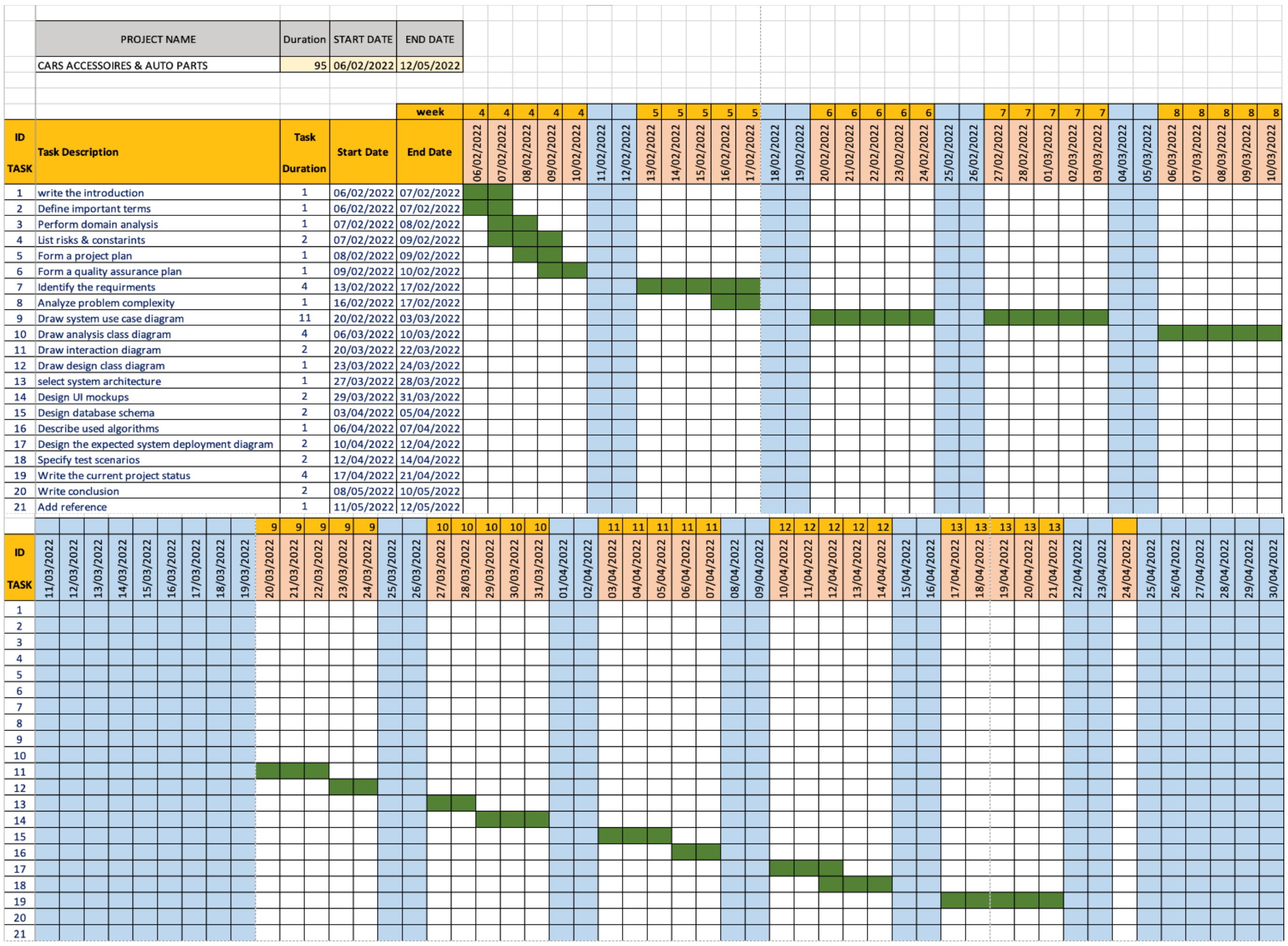
In the table below we show the features provided by our system and other similar systems.

Feature	Speero [1]	Afyal [2]	TM [3]	Our system
Payment for the item	✓	✓	✕	✓
Seller location	✓	✕	✓	✓
User location	✕	✕	✓	✓
install the spare parts or provide the service by a captain or car mechanic	✕	✕	✓	✓
View the list of items	✓	✓	✕	✓
Sellers can offer their products	✓	✓	✕	✓
User can contact with the item seller or service provider	✓	✕	✓	✓

Table 3.1 Domain analysis

4. Project Plan

4.1 Part-1 plan



PROJECT NAME	Duration	START DATE	END DATE
CARS ACCESSOIRES & AUTO PARTS	95	06/02/2022	12/05/2022

ID TASK	Task Description	Task Duration	Start Date	End Date
1	write the introduction	1	06/02/2022	07/02/2022
2	Define important terms	1	06/02/2022	07/02/2022
3	Perform domain analysis	1	07/02/2022	08/02/2022
4	List risks & constarints	2	07/02/2022	09/02/2022
5	Form a project plan	1	08/02/2022	09/02/2022
6	Form a quality assurance plan	1	09/02/2022	10/02/2022
7	Identify the requirments	4	13/02/2022	17/02/2022
8	Analyze problem complexity	1	16/02/2022	17/02/2022
9	Draw system use case diagram	11	20/02/2022	03/03/2022
10	Draw analysis class diagram	4	06/03/2022	10/03/2022
11	Draw interaction diagram	2	20/03/2022	22/03/2022
12	Draw design class diagram	1	23/03/2022	24/03/2022
13	select system architecture	1	27/03/2022	28/03/2022
14	Design UI mockups	2	29/03/2022	31/03/2022
15	Design database schema	2	03/04/2022	05/04/2022
16	Describe used algorithms	1	06/04/2022	07/04/2022
17	Design the expected system deployment diagram	2	10/04/2022	12/04/2022
18	Specify test scenarios	2	12/04/2022	14/04/2022
19	Write the current project status	4	17/04/2022	21/04/2022
20	Write conclusion	2	08/05/2022	10/05/2022
21	Add reference	1	11/05/2022	12/05/2022

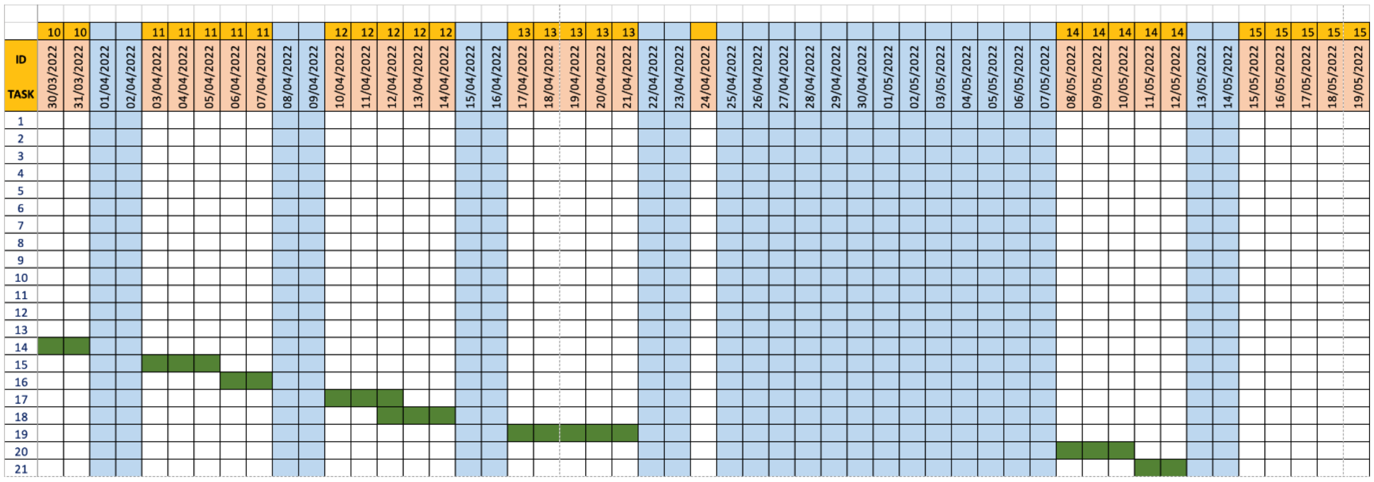


Figure 4.1.1 part-1 plan

4.2 Part-2 plan

Semester's week	Activity/implemented requirement	Date
Week-1	1- Preparation and work distribution 2- Integrate mail API and google map API to CAP system	28/8/2022 – 3/9/2022
Week-2		4/9/2022 – 10/9/2022
Week-3	SR1-SR19	11/9/2022 – 17/9/2022
Week-4	Integrate payment API	18/9/2022 – 24/9/2022
Week-5	SR20, SR22,SR23 SR24-SR28	25/9/2022 – 1/10/2022
Week-6		2/10/2022 – 8/10/2022
Week-7	SR29-SR38 SR39-SR47 SR50, SR54	9/10/2022 – 15/10/2022
Week-8	SR31-SR38 SR21,SR48 SR49 SR55-SR58	16/10/2022 – 22/10/2022
Week-9	1- Testing 2- Modify founded bugs 3- complete documentation	23/10/2022 – 29/10/2022
Week-10		30/10/2022 – 5/11/2022
Week-11		6/11/2022 – 12/11/2022

Table 4.2.1 part-2 plan

5. Quality Assurance Plan

5.1 Inspection:
After each member has completed the part of the work which assigned to him, we will review the work that each team member do before meeting with the advisor.

5.2 Formal review:
After completing the inspection session, we will have some meetings with the advisor to review the work we have accomplished during the semester.

5.3 Verification:
In the verification, we will test the unit that implement a specific function to check if that unit is bug-free, and check if it is implemented as specified.

5.4 Validation:
In our meeting with the advisor, we will run a tour of the system in front of the advisor to validate that all functional requirements are met.

6. Requirements

6.1 Functional Requirements

6.1.1 Register

SR1. The user shall be able to register to the system.
SR2. The captain shall be able to register to the system.
SR3. The seller shall be able to register to the system.

6.1.2 Log in

SR4. The user shall be able to log in to the system.
SR5. The captain shall be able to log in to the system.
SR6. The admin shall be to log in to the system.
SR7. The seller shall be able to log in to the system.

6.1.3 Forget password
SR8. The user shall be able to send a request to change his password.
SR9. The seller shall be able to send a request to change his password.
SR10. The captain shall be able to send a request to change his password.
SR11. The admin shall be able to send a request to change his password.

6.1.4 Log out
SR12. The user shall be able to log out from the system.
SR13. The captain shall be able to log out from the system.
SR14. The admin shall be to log out from the system.
SR15. The seller shall be able to log out from the system.

6.1.5 Manage account
SR16. The user shall be able to manage his account.
SR17. The seller shall be able to manage his account.
SR18. The captain shall be able to manage his account.
SR19. The admin shall be able to manage his account.

6.1.6 View item
SR20. The user shall be able to view the offered items.

6.1.7 View delivery info
SR21. The captain shall be able to view order information.

6.1.8 Search for item
SR22. The user shall be able to search for an item.
SR23. The user shall be able to sort the items.

6.1.9 Purchase an item
SR24. The user shall be able to add an item in the cart.
SR25. The user shall be able to delete an item in the cart.
SR26. The user shall be able to edit the items quantity in the cart.
SR27. The user shall be able to choose the mode of delivery.
SR28. The user shall be able to complete the payment.
SR29. The user shall be able to get the order number.
SR30. The user shall be able to get the confirmation code for the order.

6.1.10 View the list of captains / sellers
SR31. The admin shall be able to view the list of captains.
SR32. The admin shall be able to view the list of sellers.

SR33. The admin shall be able to search for a seller.
SR34. The admin shall be able to search for a captain.
SR35. The admin shall be able to remove a seller.
SR36. The admin shall be able to remove a captain.

6.1.11 View the list of new sellers & captains
SR37. The admin shall be able to review the list of new sellers.
SR38. The admin shall be able to review the list of new captains.

6.1.12 Manage item
SR39. The seller shall be able to add an item.
SR40. The seller shall be able to delete an item.
SR41. The seller shall be able to edit an item.
SR42. The seller shall be able to add a discount for an item.
SR43. The admin shall be able to search for an item.
SR44. The admin shall be able to delete an item.

6.1.13 Get user request
SR45. All available captains shall be able to get notification for the ready user orders.

6.1.14 Manage user request
SR46. The captain shall be able to accept or reject the user order.

6.1.15 View the assigned task
SR47. The captain shall be able to view a list of assigned orders.

6.1.16 Change the status of order
SR48. The seller shall be able to change the status to be ready for captain to pick up.
SR49. The captain shall be able to provide the confirmation code from the user to change the status of order (or include provide status code).

6.1.17 Report an item
SR50. The user shall be able to report the item for a problem.

6.1.18 Review reports
SR51. The admin shall be able to review the report.

6.1.19 Wishes list
SR52. The user shall be able to add an item to the wishes list.
SR53. The user shall be able to view his wishes list.
SR54. The user shall be able to delete an item from the wishes list.

6.1.20 View seller order
SR55. The seller shall be able to view the list of new orders to prepare the order.
SR56. The seller shall be able to view the previous order.

6.1.21 View order
SR57. The user shall be able to view list of orders.
SR58. The user shall be able to view the status of the order.

6.2 Non-Functional Requirements
6.2.1 Performance
SR59. The system shall accommodate 100 orders per minute.
SR60. The system's load time should not be more than three seconds for each page.
SR61. Average repair time should be less than one hour.

6.2.2 Usability
SR62. Non-technical users shall be able to learn how to use the system in less than one hours.
SR63. Eighty percent of the users shall be able to make an order in 10 minutes without requiring any assistance.

6.2.3 Reliability
SR64. The system shall have no more than 1 hour downtime per month.
SR65. Mean time between failures shall be at least 30 days.

6.2.4 Availability
SR66. The system shall be available 99.86% of the time.

6.2.5 Confidentiality
SR67. The system shall not retain user's credit or debit card information entered during the Checkout payment processing.
SR68.The system shall allow only authorized people to access the data.
SR69. The system shall encrypt the passwords in the database.

6.2.6 Interoperability
SR70. The system shall support google maps API.

6.2.7 Integrity
SR71. User's orders shall be backed up at least once per month to prevent data loss.

6.3 Design constraints

There are no specific design constraints for this project.

7. Problem Complexity

- Many sub-problems

In our system we have 58 functional requirements with different actors which is not an easy number, so there should be more efforts to deal with task distribution among the development team and a strict management to conduct the changes in the documents and the actual work.

- Diverse groups of stakeholders

In our system we have four primary actors who are users, sellers, captains, and the admin, and each one have functionalities effect to the other actors, so more coordination efforts should be taken place between them, and also we have three secondary actors which map API, email API, and payment API, and they have functionalities serves the primary actors functionalities so these should factors should be taken into account for the development team.

8. System Use-Cases

8.1 Use case diagram

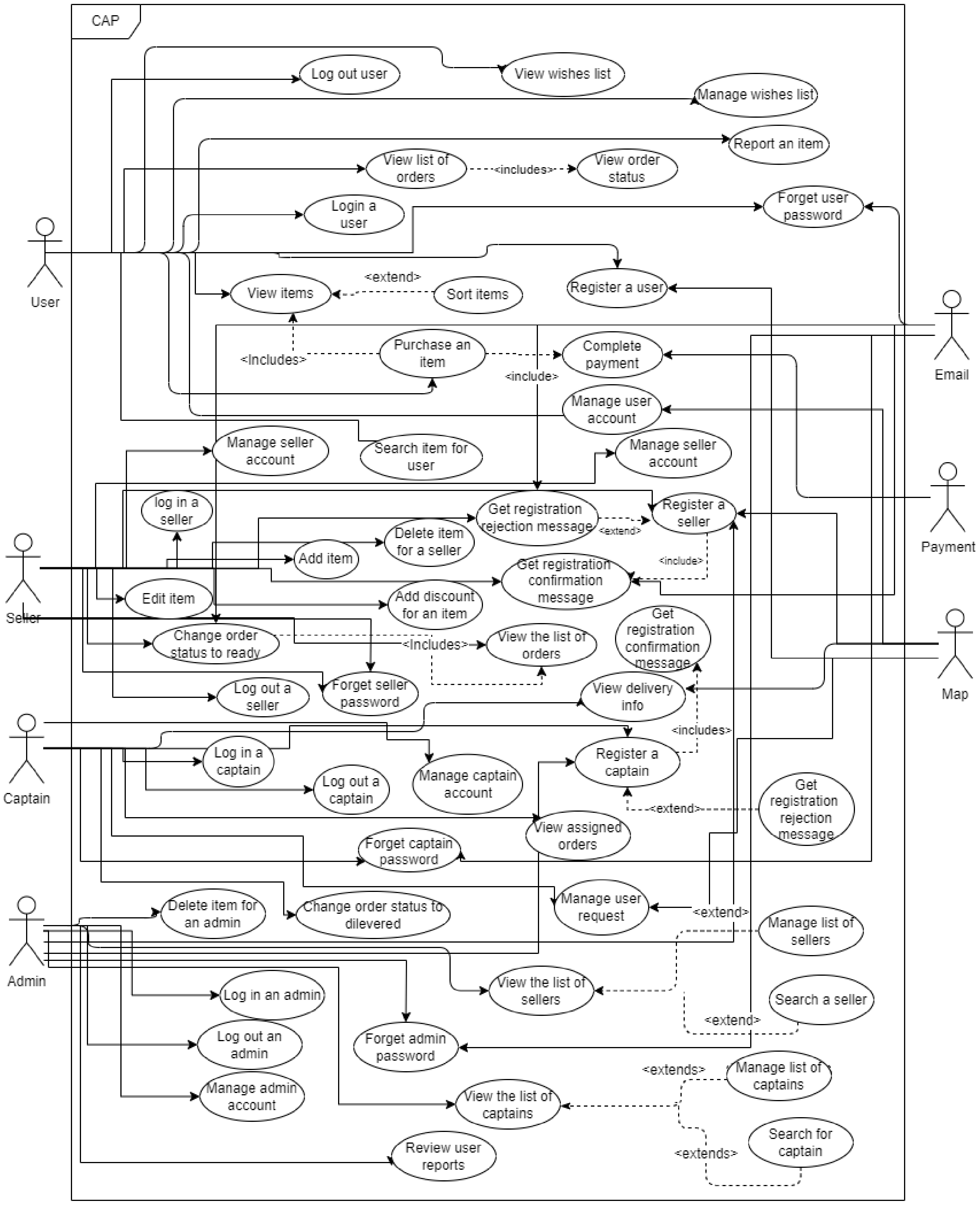


Figure 8.1 Use case diagram

8.2 Use case description

Table 8.2.1 Register a user

Use Case Description		
Use Case name: Register a user		
Primary actor: User	**Other actors:** Map	
Description: This use case clarifies how the user registers his info to have a new account in the system.		
Relationships ▪**Includes:** NA ▪**Extends:** NA		
Pre-conditions: NA		
Inputs: name, email, phone number, password, and location		
Steps:		
User	**System**	**Map**
1- The user register a new account in the system.		
	2- The system asks the user to give his name, email, phone number and password.	
3- The user gives his name, email, phone number and password.		
	4- The system validates the user inputs. 5- The system asks the map API to provide the map.	
		6- The map API gives the map to the system.

8- The user set his location.	7- The system shows the map. 9- The system validates the user location. 10- The system registers the user with his provided information and location.	

Alternative and exceptional flows: 4a- The user's email already exists 4a.1 The system shows a message to indicate that error 4a.2 The system returns to step two. 4b- The user's name, password, phone number or email is invalid. 4b.1 The system shows a message to indicate that error. 4b.2 The system returns to step two. 9a- The user set a location out of the system domain 8a.1 The system shows an error message. 8a.2 The system returns to step five.
Post-conditions: The user has a new account and can use it to log in to the system.

Table 8.2.2 Log in a user

Use Case Description	
Use Case name: Log in a user	
Primary actor: user	**Other actors:** NA
Description: This use case describes how the user log in to the system.	
Relationships ▪**Includes:** NA ▪**Extends:** NA	
Input: Email, password	
Pre-conditions: The user must be registered in the system.	
Steps:	
User	**System**

1- The user login to the system. 3- The user provides his email and password.	2- The system asks the user to provide his email and password. 4- The system validates the provided information. 5- The system logs in the user.
Alternative and exceptional flows: 3a- The information entered is invalid. 3a1- The system displays an error message. 3a2- The system returns to step 2.	
Post-conditions: The user logs into the system.	

Table 8.2.3 Log out a user

Use Case Description	
Use Case name: Log out a user	
Primary actor: user	**Other actors:** NA
Description: this use case describes how the user logs out from the system.	
Relationships ▪**Includes:** NA ▪**Extends:** NA	
Pre-conditions: The user must be logged in.	
Steps:	
User	**System**
1- The user logout for the system. 3- The user confirms the action.	2- The system asks for confirmation. 4- The system logs out the user.
Alternative and exceptional flows: 3a- The user cancels the action. 3a1- Exit the use case.	
Post-conditions: The user logs out of the system.	

Table 8.2.4 Register a seller

Use Case Description			
Use Case name: Register a seller			
Primary actor: Seller		**Other actors:** Map, Admin	
Description: This use case clarify how the Seller registers his info to have a new account in the system.			
Relationships ▪**Includes:** Get registration confirmation message ▪**Extends:** Get registration rejection message			
Pre-conditions: NA			
Input: name, email, phone number, password, location, and CV			
Steps:			
Seller	System	Map	Admin
1- The seller register a new account in the system.			
	2- The system asks the seller to give his name, email, phone number, password and CV.		
3- The user gives his name, email, phone number, password and CV.			
	4- The system validates the seller inputs.		
	5- The system asks the map API to provide the map.		
		6- The map API gives the map to the system.	
	7- The system shows the map.		

8- The seller set his location.	9- The system validates the seller's location. 10- The system gives the seller information to the admin to be able to review it 13- The system executes Get registration confirmation message use case. 14- The system registers the seller with his provided information and location.		11- The admin reviews the seller and decides to accept or reject the seller. 12- The admin accepts the seller
Alternative and exceptional flows: 4a- The seller’s email already exists 4a.1 The system shows a message to indicate that error 4a.2 The system returns to step two. 4b- The seller’s name, password, phone number or email is invalid. 4b.1 The system shows a message to indicate that error. 4b.2 The system returns to step two. 9a- The seller set a location out of the system domain 8a.1 The system shows an error message. 8a.2 The system returns to step five. 11a- The admin rejects the seller. 11a.1 The Get registration rejection message use case will be executed.			

Post-conditions: The seller has a new account and can use it to log in to the system.

Table 8.2.5 Register a captain

Use Case Description		
Use Case name: Register a captain		
Primary actor: captain	**Other actors:** Admin	
Description: This use case clarifies how the captain registers his info to have a new account in the system.		
Relationships ▪**Includes:** Get registration confirmation message ▪**Extends:** Get registration rejection message		
Pre-conditions: none		
Input: name, email, phone number, password and CV.		
Steps:		
Captain	System	Admin
1- The captain register a new account in the system.		
	2- The system asks the captain to give his name, email, phone number, password and CV.	
3- The captain gives his name, email, phone number, password and CV.		
	4- The system validates the captain inputs.	
	5- The system gives the captain information to the admin to be able to review it	
		6- The admin reviews the captain and decides whether to accept or reject the captain.
		7- The admin accepts the captain

	8- The system will execute Get registration confirmation message use case. 9- The system registers the captain with his provided information and location.	
Alternative and exceptional flows: 4a- The Captain's email already exists 4a.1 The system shows a message to indicate that error 4a.2 The system returns to step two. 4b- The Captain's name, password, phone number or email is invalid. 4b.1 The system shows a message to indicate that error. 4b.2 The system returns to step two. 6a- The admin rejects the captain. 6a.1 The Get registration rejection message use case will be executed.		
Post-conditions: The Captain has a new account and can use it to log in to the system.		

Table 8.2.6 Review user reports

Use Case Description	
Use Case name: Review user reports	
Primary actor: Admin	**Other actors:** NA
Description: This use case shows how the admin reviews the users reports related to a specific item.	
Relationships ▪**Includes:** NA ▪**Extends:** NA	
Pre-conditions: The admin must be logged in.	
Steps:	
Admin	**System**
1- The admin requests to view the reports done by users.	2- The system checks the reports that is already registered. 3- The system shows the list of all reports that are registered in the system.

Alternative and exceptional flows: 2a- The system doesn’t find any report from the user 2a.1 The system shows a message that signifies that there are no reports to show.
Post-conditions: The users reports are displayed to the admin

Table 8.2.7 View list of orders

Use Case Description	
Use Case name: View list of orders	
Primary actor: Seller	**Other actors:** NA
Description: This use case describes how the seller views his ordered items.	
Relationships ▪**Includes:** NA ▪**Extends:** NA	
Pre-conditions: The seller must be logged in.	
Steps:	
seller	**System**
1- The seller asks the system to show his ordered items by the users	2- The system validates items that are ordered by a user. 3- The system shows the seller items that are ordered by a user.
Alternative and exceptional flows: 2a- The system doesn't find any items ordered for the seller. 2a.1 The system shows a message that signifies that there are no orders to show.	
Post-conditions: The seller views his ordered items	

Table 8.2.8 Report an item

Use Case Description	
Use Case name: Report an item.	
Primary actor: User	**Other actors:** NA
Description: This use case describes how a user writes a report about an item in the system.	
Relationships ▪**Includes:** NA ▪**Extends:** NA	
Pre-conditions: The user must be logged in.	
Steps:	
User	**System**
1- The user writes a report about an item.	
	2- The system provides a space for writing a report.
3- The user writes a report and sends it.	
	4- The system displays a success message.
Alternative and exceptional flows: 3a- The user sends empty report 3a.1- The system displays a message signifies that the report body is empty. 3a.2- The system returns to step 2.	
Post-conditions: A new report about an item is created.	

Table 8.2.9 Forget password

Use Case Description		
Use Case name: Forget password.		
Primary actor: User	**Other actors:** Email	
Description: This use case describes how a user reset his password.		
Relationships ▪**Includes:** none ▪**Extends:** Log in		
Input: Email, new password.		
Pre-conditions: The user must be registered.		
Steps:		
User	**System**	**Email**
1- The user reset his password.		
	2- The system asks for their email.	
3- The user provides the email.		
	4- The system validates the email existence.	
	5-The system asks the email Actor (API) to send a new password to the user.	
		6- Email delivers an email message to the provided email with a new password.
	7- The system displays a success message.	
	8- The system redirects the user to the login page.	
Alternative and exceptional flows: 4a- The email entered does not exist. 4a1- The system displays an error message. 4a2- The system returns to step 2.		

Post-conditions: The user password is modified.

Table 8.2.10 Login a seller

Use case name: login a seller	
Primary actor: Seller	**Other actors:** NA
Description: this case describes the way the seller logs in to the system	
Relationships: **1. Includes:** NA **2. Extends:** NA	
Input: Email, Password	
Pre-condition: the Seller must be registered in the System	
Steps:	
Seller	**System**
1- The Seller login to the system. 3- The Seller provides his email and password	2- The system displays a login form. 4-The system validates the provided information. 5- The system logs in the Seller. 6- The system redirects the Seller to the homepage.
Alternative and exceptional flows: 3a- The information entered is invalid or the user doesn't exist. 3a1- The system displays an error message. 3a2- The system returns to step 2.	
Post-condition: The Seller is logged in in the system.	

Table 8.2.11 Add item

Use case name: Add item	
Primary actor: Seller	**Other actors:** NA
Description: this case describes the way the seller adds a new item to the system	
Relationships: **1. Includes:** NA **2. Extends:** NA	
Input: item name, picture, description, quantity, and price.	
Pre-condition: The Seller must be logged in to the System	
Steps:	
Seller	**System**
1. The Seller views the home page 2. The Seller adds a new item to the system 4. The Seller provides information needed for the item	3- The system display adds item page 5- The system adds the item
Alternative and exceptional flows: 4a The Seller didn't enter all the information 4a1 The system displays an error message 4a2 The seller returns to step 3	
Post-condition: The item added successfully	

Table 8.2.12 Edit item

Use case name: Edit item	
Primary actor: Seller	**Other actors:** NA
Description: this case describes the way the seller edits items that have been added	
Relationships: 1. **Includes:** NA 2. **Extends:** NA	
Input: item name, picture, description, price.	
Pre-condition: the item must be added to the System	
Steps:	
Seller	**System**
1- The Seller views the item.	
2- The Seller edit the item.	
	3- The system displays the edit item page.
4- The Seller edits the selected item's info.	
	5- The system saves the edit operation
Alternative and exceptional flows: 4a- the seller provides empty input to the edited info 4a.1- the system displays an error message 4a.2- the seller returns to step number 3	
Post-condition: item's info is edited	

Table 8.2.13 change order status to ready

Use case name: change order status to ready		
Primary actor: Seller	**Other actors:** Email	
Description: this use case describes the way the order changes to ready to pick up		
Relationships: **1. Includes:** View the list of orders **2. Extends:** NA		
Input: item status		
Pre-condition: The item must have been ordered by a user		
Steps:		
Seller	System	Email
1- the seller changes the order status to ready	2- The system accepts the new order status 3- The system asks the Email API to send a message to the assigned captain about the new order status	4- Email API sends a message to the captain about the new order status
Alternative and exceptional flows: NA		
Post-condition: The order has the new status entered		

Table 8.2.14. Manage account of user

Use Case Description		
Use Case name: Manage account of user		
Primary actor: user		**Other actors:** Map
Description: Allow users to change name, email, phone number, password, location.		
Relationships ▪**Includes:** NA ▪**Extends:** NA		
Input: name, email, phone number, password, location		
Pre-conditions: user shall have an account.		
Steps:		
User	**System**	**Map**
1- The user manages his account in the system.	2- The system asks the user which field he wants to change it.	
3- The user chooses the fields {name, email, phone number, password, location} need to change it.	4- The system unblocks the fields the user chooses to change it.	
	5- The system asks map API to provide map.	6- the map API gives the map to the system
	7- The system shows the map.	
8- The user can change name, email, phone number, password and location in unblock fields.		

	9- The system validates the user changes. 10- The system accepts user changes.	
Alternative and exceptional flows: 9a- The user email already exists. 9a.1 The system shows a message to indicate that error. 9a.2 The system returns the user to step 8. 9b- The username, password, phone number or email is invalid. 9b.1 The system shows a message to indicate that error. 9b.2 The system returns the user to step 8.		
Post-conditions: NA		

Table 8.2.15. Manage account of seller

Use Case Description		
Use Case name: Manage account of seller		
Primary actor: Seller		**Other actors:** Map
Description: Allow seller to change name, email, phone number, password, location.		
Relationships ▪**Includes:** NA ▪**Extends:** NA		
Input: name, email, phone number, password, location		
Pre-conditions: Seller shall have an account.		
Steps:		
Seller	**System**	**MAP**
1- The seller manages his account in the system.		
	2- The system asks sellers which field want to change it.	
3- The seller chooses the fields {name, email, phone number, password, location} need to change it.		
	4- The system unblocks the fields the seller chooses it to change it.	
	5- The system asks map API to provide map.	
		6- the map API gives the map to the system
	7- The system shows the map.	
8- The seller can change name, email, phone number, password and location in unblock fields.		
	9- The system validates the seller changes.	
	10- The system accepts seller changes.	

Alternative and exceptional flows: 9a- The seller email already exists. 9a.1 The system shows a message to indicate that error. 9a.2 The system returns the seller to step 8. 9b- The seller's name, password, phone number or email is invalid. 9b.1 The system shows a message to indicate that error. 9b.2 The system returns the seller to step 8.
Post-conditions: NA

Table 8.2.16 Manage account of Captain.

Use Case Description	
Use Case name: Manage account of captain.	
Primary actor: Captain	**Other actors:** NA
Description: Allow Captain to change name, email, phone number, and password.	
Relationships ▪**Includes:** NA ▪**Extends:** NA	
Input: name, email, phone number, password.	
Pre-conditions: Captain shall have an account.	
Steps:	
Captain	**System**
1- The captain manages his account in the system.	
	2- The system asks Captain which fields he wants to change it.
3- The Captain chooses the fields {name, email, phone number, and password} need to change it.	
	4- The system unblocks the fields the captain chooses it to change it.
5- The Captain can change name, email, phone number and password in unblock fields.	
	6- The system validates the captain changes.
	7- The system accepts Captain changes.
Alternative and exceptional flows: 6a- The Captain email already exists. 6a.1 The system shows a message to indicate that error. 6a.2 The system returns to step 5. 6b- The Captain name, password, phone number or email is invalid. 6b.1 The system shows a message to indicate that error. 6b.2 The system returns to step 5.	

Post-conditions: NA

Table 8.2.17 Manage account of Admin.

Use Case Description	
Use Case name: Manage account of Admin.	
Primary actor: Admin	**Other actors:** NA
Description: Allow Admin to change email, password.	
Relationships ▪**Includes:** NA ▪**Extends:** NA	
Input: email, password.	
Pre-conditions: admin shall have an account.	
Steps:	
Admin	**System**
1- The Admin manages his account in the system.	
	2- The system asks Admin which field he wants to change it.
3- The Admin choose the fields {email, password} need to change it.	
	4- The system unblocks the fields the admin chooses it to change it.
5- The Admin able to change email and password in unblock fields.	
	6- The system validates the admin changes.
	7- The system accepts Admin changes.
Alternative and exceptional flows: 6a- The Admin email already exists. 6a.1 The system shows a message to indicate that error. 6a.2 The system returns to step 5. 6b- The Admin password or email is invalid. 6b.1 The system shows a message to indicate that error. 6b.2 The system returns to step 5.	
Post-conditions: NA	

Table 8.2.18 View Assigned Order.

Use Case Description	
Use Case name: View Assigned Order.	
Primary actor: Captain	**Other actors:** NA
Description: Allow Captain to view the orders which he assigned it.	
Relationships ▪**Includes:** NA ▪**Extends:** NA	
Input: NA	
Pre-conditions: Captain shall have an account.	
Steps:	
Captain	**System**
1- The captain view his assigned orders.	2- The System displays the assigned order to the captain.
Alternative and exceptional flows: 2a. The captain doesn’t have assigned order yet. 2a.1 The system shows messages to indicate that there are no orders to show.	
Post-conditions: NA	

Table 8.2.19 Manage user Request

Use Case Description		
Use Case name: Manage user Request.		
Primary actor: Captain		**Other actors:** Map
Description: Allow captain to accept the new order.		
Relationships ▪**Includes:** NA ▪**Extends:** NA		
Pre-conditions: Captain shall have an account.		
Steps:		
Captain	**System**	**MAP**
1- The captain gets a notification for a new user order.		
	2- The system asks map API to provide user location who create the order.	
		3- The map API gives the location to the system
	4- The system displays order information with the location to the captain.	
5- The captain gets the order's info.		
	6- The system asks the captain to accept or reject the order.	
7- The captain accepts the order.		
	8- The system removes order's info from the other captains who didn't give a response for that order.	

Alternative and exceptional flows: 7a-The captain rejects the order. 7a.1- The system waits for the other captain’s response. 7b- All captains in the system reject the order. 7b.1 The system asks the email API to send a message to the user that his order is cancelled.
Post-conditions: The order will be accepted or rejected by the captain.

Table 8.2.20 Change order status to delivered

Use Case Description	
Use Case name: Change order status to delivered	
Primary actor: Captain	**Other actors:** NA
Description: Allow Captain to change status by using user confirmation code.	
Relationships ▪**Includes:** NA ▪**Extends:** NA	
Input: NA	
Pre-conditions: Captain shall have an order.	
Steps:	
Captain	**System**
1- The captain changes the status of the order to delivered. 3- The Captain gives the system confirmation code.	2- The System asks the captain to enter confirmation code. 4- The system validates the code to change the status of the order to delivered.
Alternative and exceptional flows: 4a. The captain enters invalid code. 4a.1 The system shows a message to indicate that error. 4a.2 The system returns to step 2.	
Post-conditions: the status of order will be delivered.	

Table 8.2.21 Purchase an item

Use Case Description	
Use Case name: Purchase an item	
Primary actor: User	**Other actors:** NA
Description: This use case shows how the user purchase an item through the system	
Relationships ▪**Includes:** 1-View item 2- complete purchase ▪**Extends:** NA	
Pre-conditions: The user must be logged into the system	
Steps:	
User	**System**
1-The user selects the item.	
	2- The user displays the item information.
3- The user sets the quantity of the item.	
4- The user add the item to the cart.	
	5- The system displays the item in the cart.
6-The user selects the cart.	
	7- The system displays the cart.
	8-The system asks the user to select the payment method and mode of delivery.
9- The user selects the payment method and mode of delivery.	
10- The user asks to purchase the item.	
	11- The system asks the user to complete the payment.
12- The user completes the payment.	
	13- The system validates the purchase process.

	14- The system gives the order number. 15- The system gives the confirmation code for the order.
Alternative and exceptional flows: 3a- the requested quantity exceeds the available 3a1 – the system displays message that the quantity is exceeding the available	
Post-conditions: message displayed that the order completed, and the order is display in order lists	

Table 8.2.22. View items

Use Case Description	
Use Case name: View items	
Primary actor: User	**Other actors:** NA
Description: This use case shows how the user will views items in the system	
Relationships ▪**Includes:** NA ▪**Extends:** NA	
Pre-conditions: The user must be logged into the system	
Steps:	
User	**System**
1-The user enters the homepage 2-The user views the items	3-the system displays the list of items
Alternative and exceptional flows: NA	
Post-conditions: The items in the system is shown	

Table 8.2.23 Sort items

Use Case Description	
Use Case name: Sort items	
Primary actor: User	**Other actors:** NA
Description: This use case shows how to sort the items in the system	
Relationships ▪**Includes:** NA ▪**Extends:** View items	
Input: name of the way of sorting	
Pre-conditions: The user must be logged into the system	
Steps:	
User	**System**
1-The user selects the list for sort items. 3- The user selects the way of sorting.	2-The system asks the user to select the way of sorting. 4- The system sorts the items.
Alternative and exceptional flows: NA	
Post-conditions: the items sorted in the way the user has selected	

Table 8.2.24 Search items

Use Case Description	
Use Case name: Search items	
Primary actor: User	**Other actors:** NA
Description: This use case shows how to search for items in the system	
Relationships ▪**Includes:** NA ▪**Extends:** NA	
Input: name of the item	
Pre-conditions: The user must be logged into the system	
Steps:	
Actor	**System**
1-The user selects the search field	
	2-The system asks the user to enter
3- The user enters the name of the item	
	4- The system validates the data entered
	5- The system displays the result
Alternative and exceptional flows: 4a- The data not found 4a1- The system displays a message say that the item is not found	
Post-conditions: displaying the result of search	

Table 8.2.25 Complete purchase

Use Case Description		
Use Case name: Complete purchase		
Primary actor: User	**Other actors:** Payment	
Description: This use case shows how the user completes the payment for an order		
Relationships ▪**Includes:** ▪**Extends:**		
Input: Credit card details (card number, card expiration date, email, CVV)		
Pre-conditions: The user must be logged into the system		
Steps:		
User	**System**	**payment**
1-The user completes purchase for the order		
	2-The system asks the user to enter credit card details (card number, card expiration date, email, CVV)	
3- The user enters the credit card details		
	4- The system sends the credit card details to payment API	
		5- The payment API verifies credit card details
		6-The payment asks confirmation from user
7- The user confirms the payment		
		8- The payment API completes the transaction
	9- The system completes the purchase	

Alternative and exceptional flows:

5a- the credit card details are incorrect

5a1- the system displays message that the credit card details incorrect please try again

5a2- return to step 3

5b- the credit card does not have sufficient funds to make the payment

5b1- the system displays "payment failed"

5b2- return to step 3

Post-conditions: message displays thank you for purchase and the order displays in the order lists.

9. Analysis Class

In this section we show the analysis class diagram for eight use cases, which is done by Entity-Control-Boundary Pattern [5]:

9.1 Purchase Item

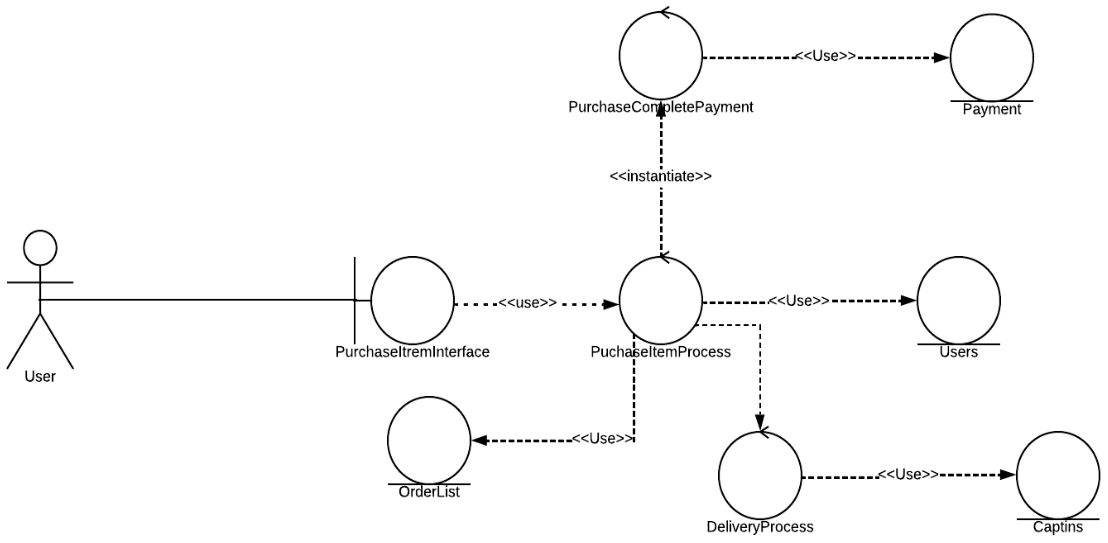


Figure 9.1.1 Purchase item AC

9.2 View item

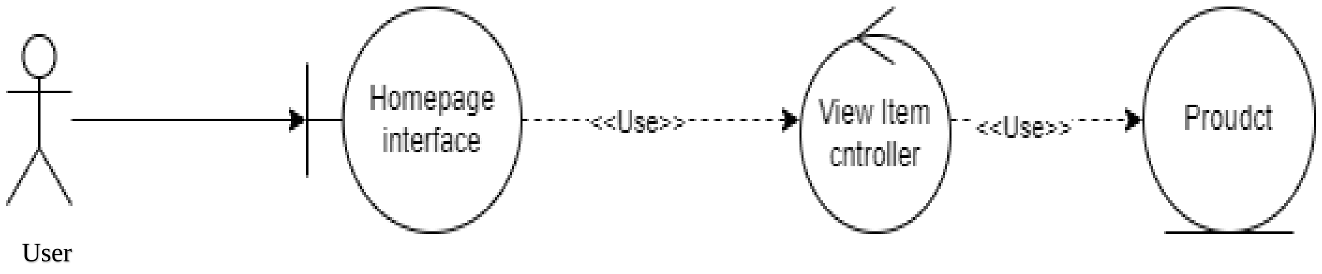


Figure 9.2.1 View item AC

9.3 Change order status to delivered

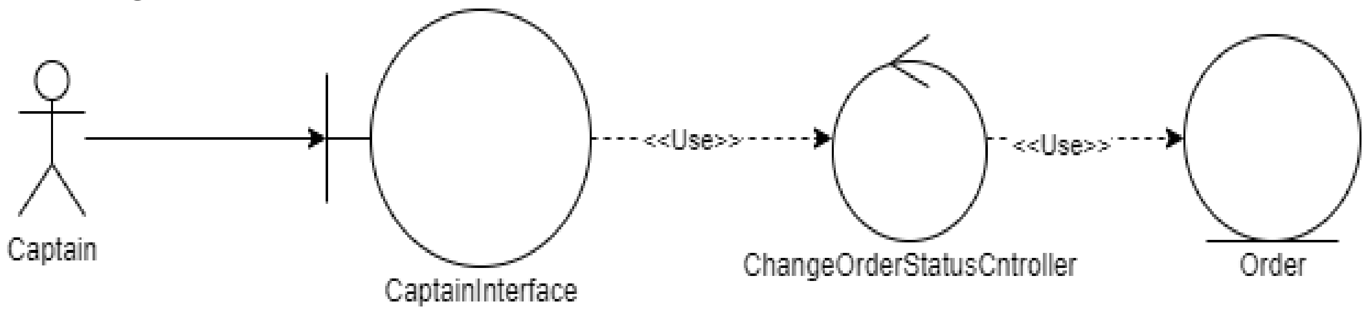


Figure 9.3.1 Change order status to delivered AC

9.4 Manage user request

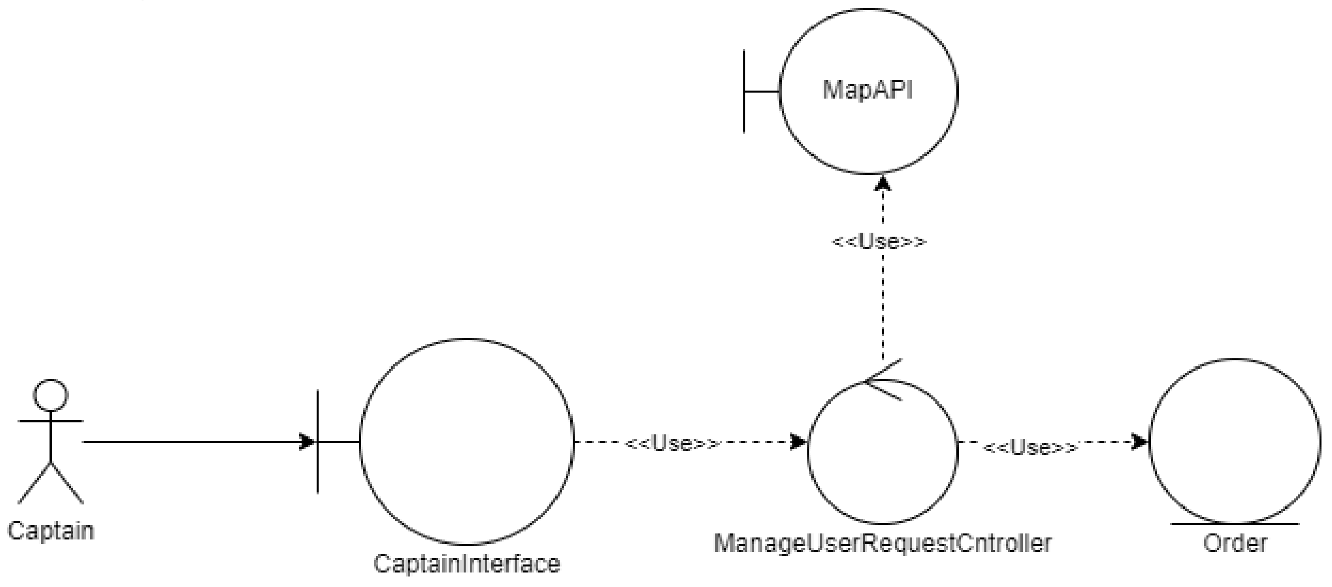


Figure 9.4.1 Manage user request AC

9.5 Change order status to Ready

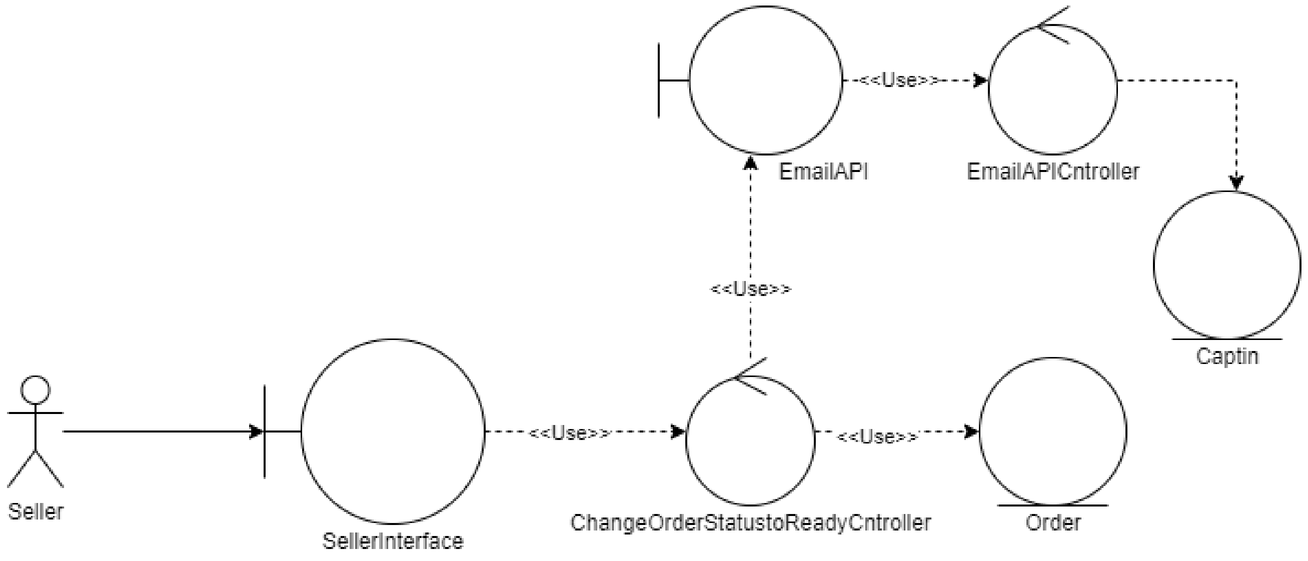


Figure 9.5.1 Change order status to ready AC

9.6 Register Seller

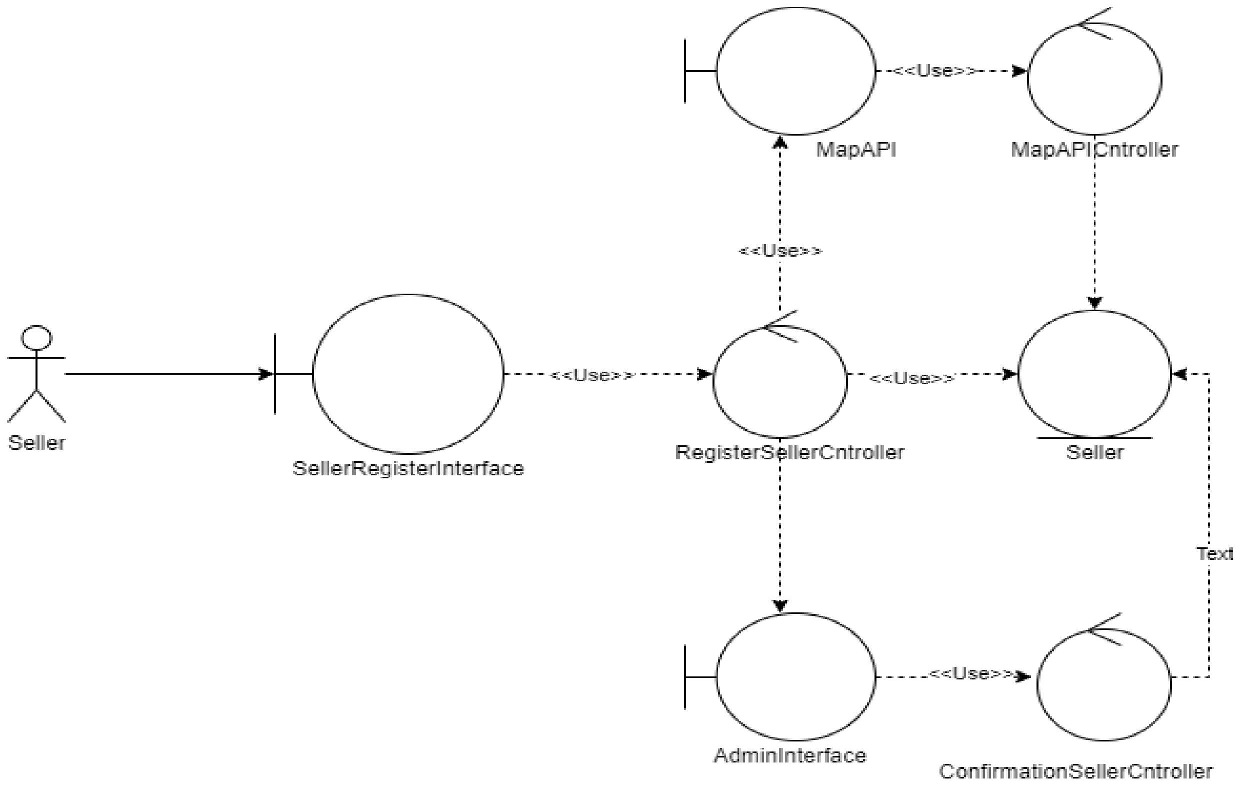


Figure 9.6.1 Register a seller AC

9.7 View delivery info

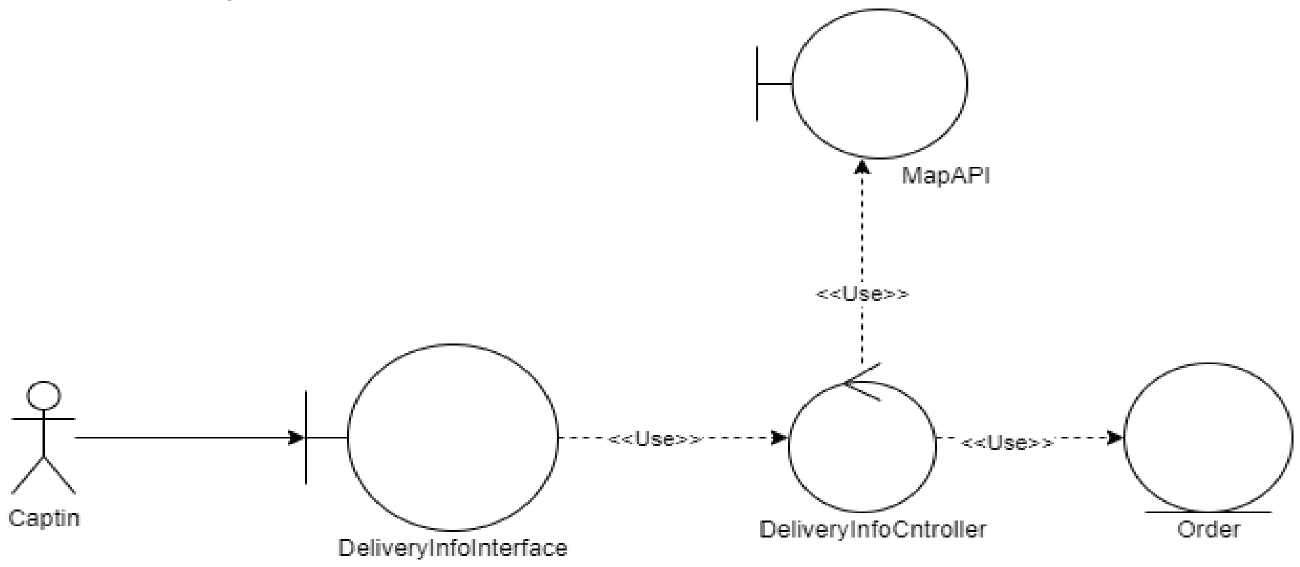


Figure 9.7.1 View delivery info AC

9.8 Report item

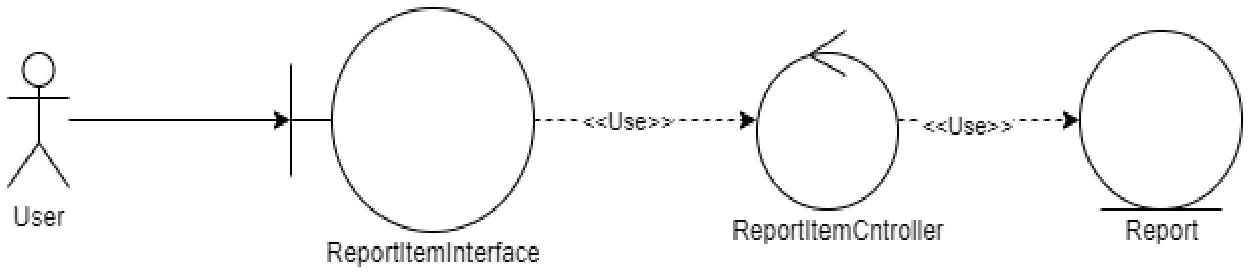


Figure 8.8.1 Report item AC

10. Interaction Diagram

In this section we show the sequence diagram for eight use cases:

10.1 Purchase item

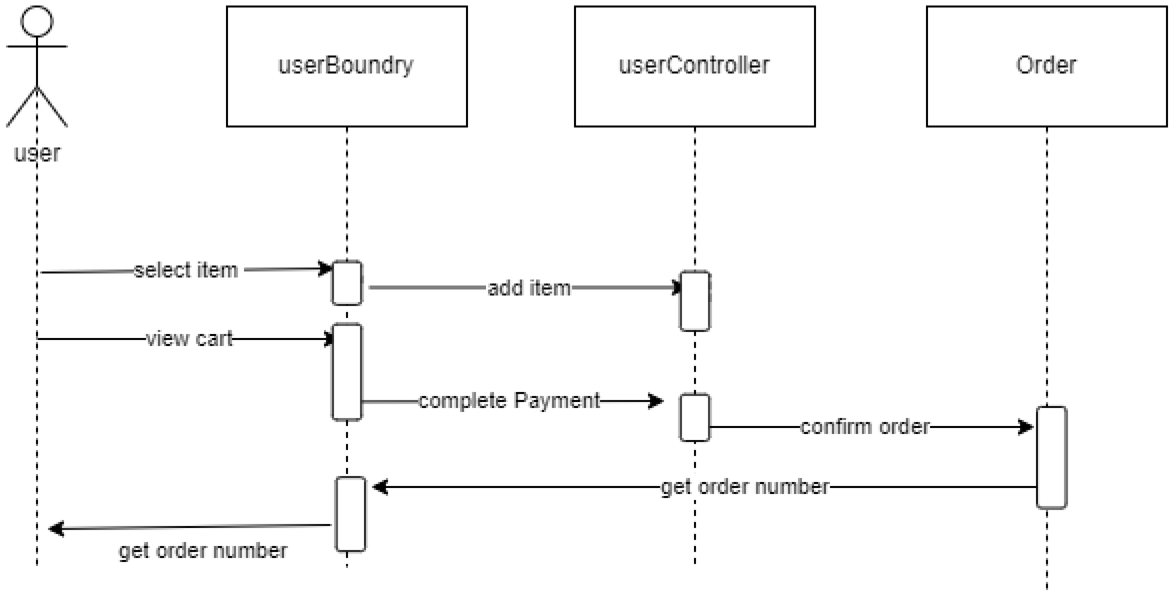


Figure 10.1.1 Purchase item SD

10.2 View item

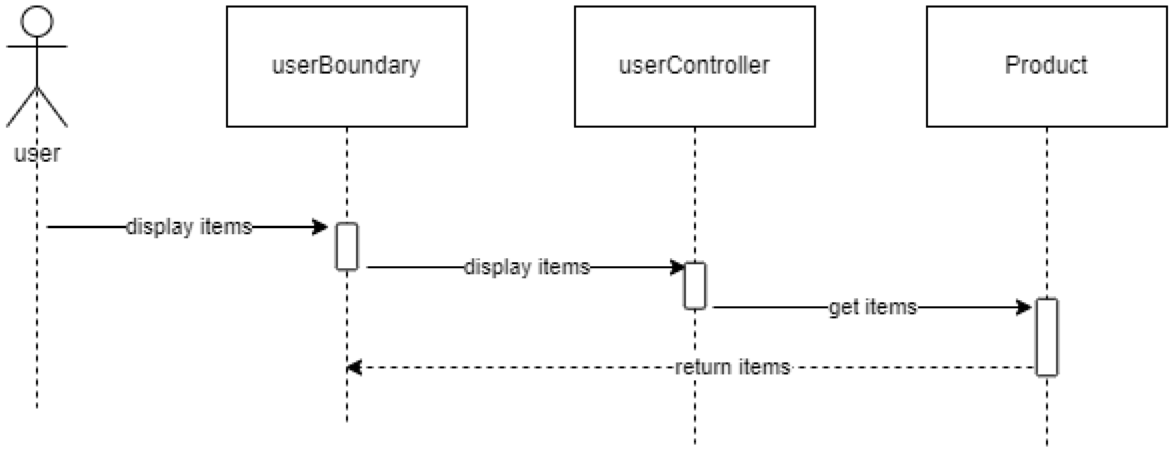


Figure 10.2.1 View item SD

10.3 Manage user request

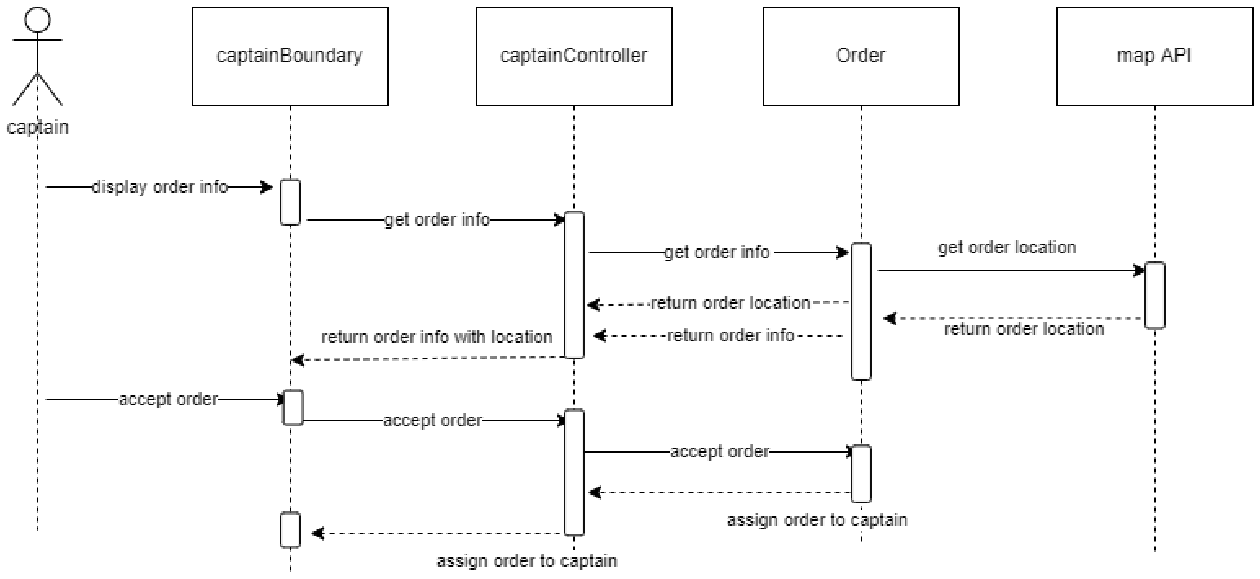


Figure 10.3.1 Manage user request SD

10.4 Change order status to delivered

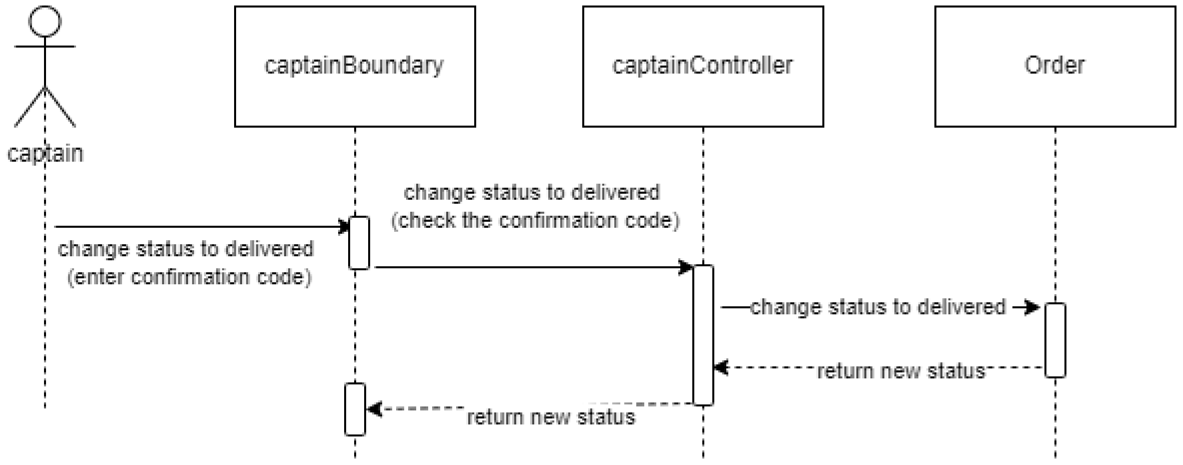

captain
captainBoundary
captainController
Order
change status to delivered
(enter confirmation code)
change status to delivered
(check the confirmation code)
change status to delivered
return new status
return new status

Figure 10.4.1 Change order status to delivered SD

10.5 Change order status to ready

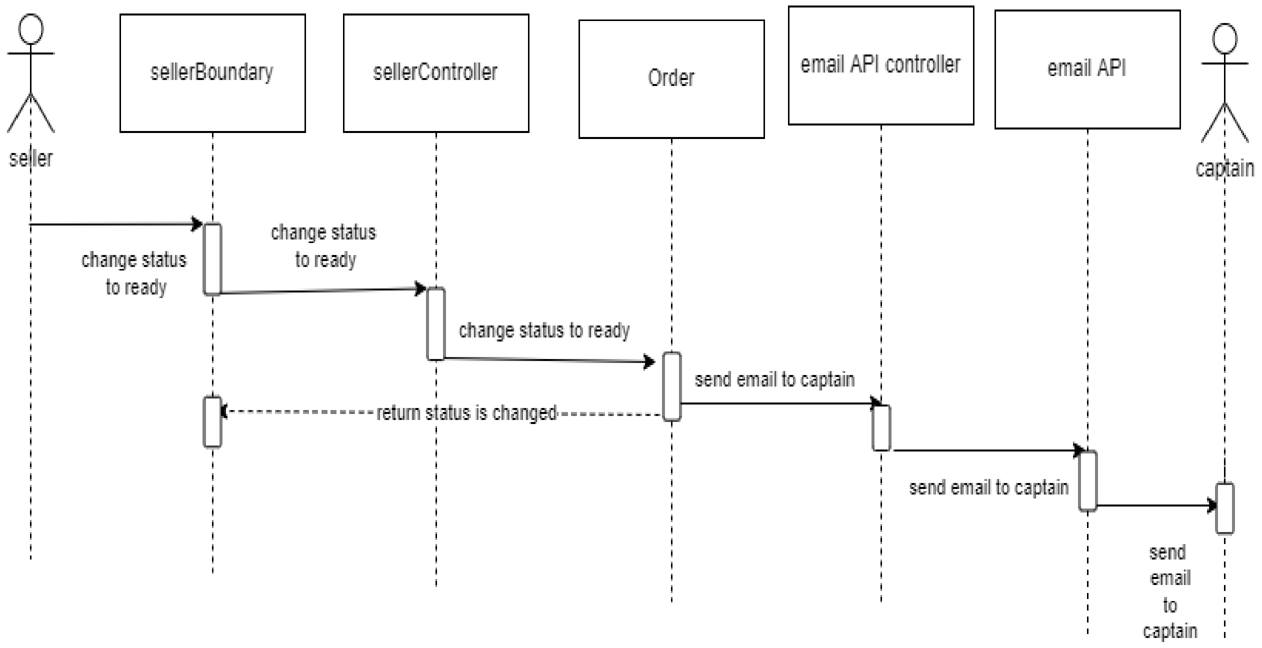


Figure 10.5.1 Change order status to ready SD

10.6 Register a seller

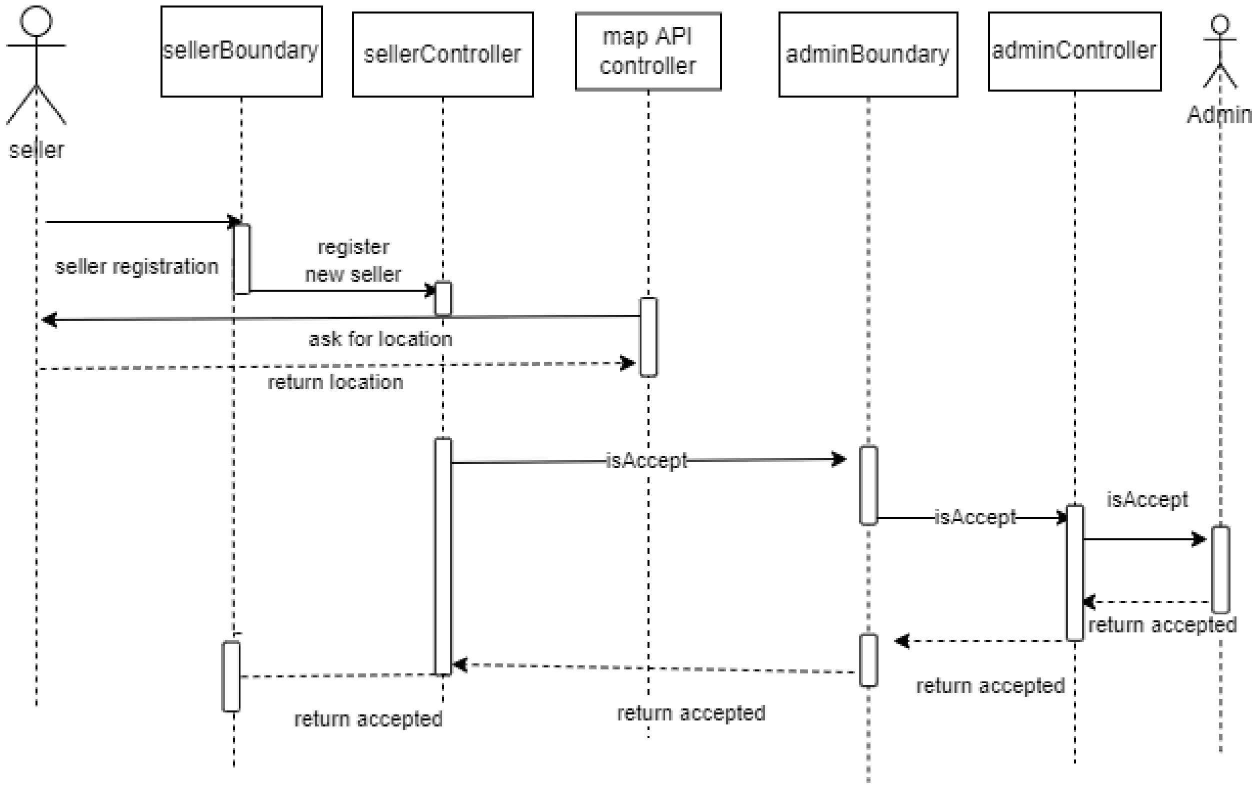


Figure 10.6.1 Register a seller SD

10.7 View delivery info

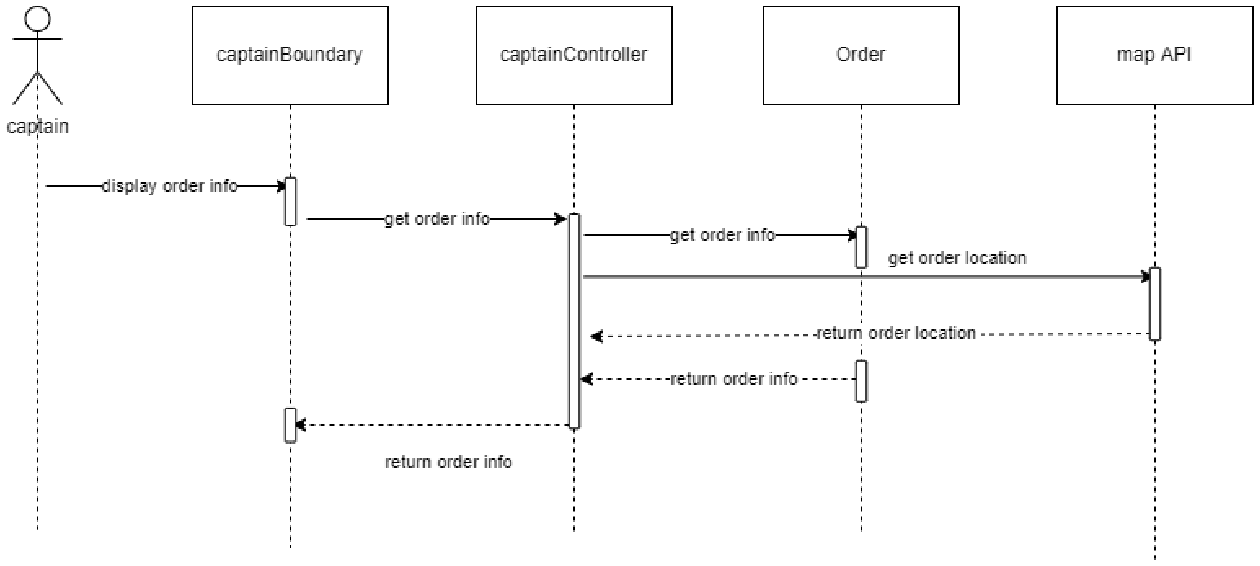


Figure 10.7.1 View delivery info SD

10.8 Report item

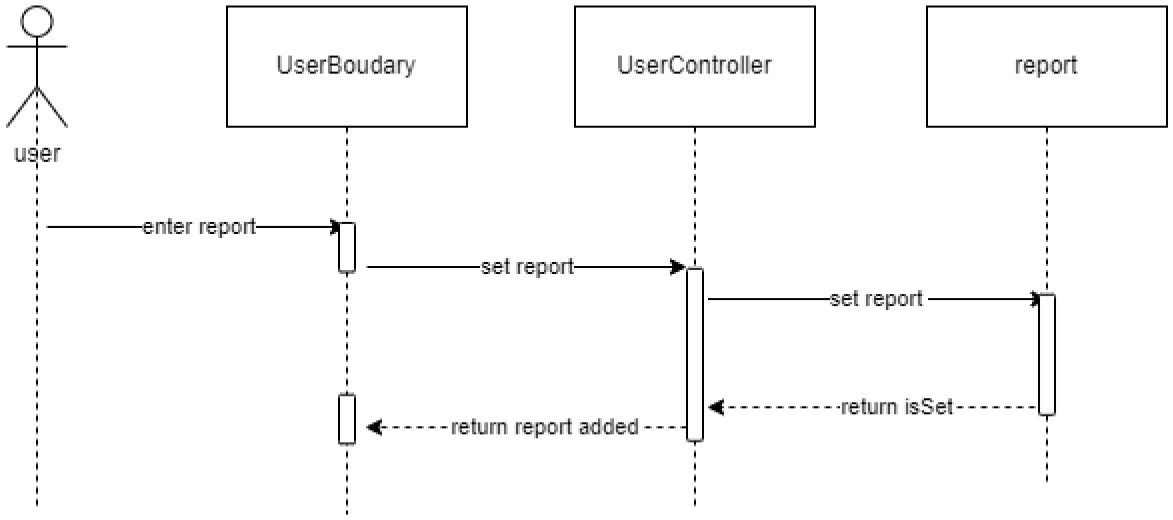


Figure 10.8.1 Report item SD

11. System Architecture

We will implement our design by using client-server architecture, it is the most common architecture used currently, and many web and mobile applications use that architecture, it separates the responsibilities between many agents:
Client: it is responsible for user interface and the communication or interaction between the system and the user
Application server: it is responsible for implementing the business logic of the system, and it is located inside the web server.
Database server: it is responsible for the related data to the system, and it is located inside with the web server.

12. Prototype Description

In this section we shows some details of software implementation.

12.1 Implementation platform

We implement our web application by using the following languages and frameworks:

1- Front end development
HTML: a mark-up language which specify the structure of the page which then will be displayed in the web browser.
CSS: a styling language used to describe how the HTML elements will displayed.
W3.CSS: a modern framework used to simplify the styling of HTML elements.
JavaScript: a dynamic programming language used to enhance the functionality of web pages and allows the client-side to interact with the user.
JQuery: a JavaScript library used to simplify the code of JavaScript (i.e: write less, do

more).

2- Back end development
PHP: server-side language that makes a dynamic and interactive web pages.
MYSQL: database management system used to retrieve and manipulate the stored data in the database, it is the most popular database system used with PHP.

12.2 Algorithms

12.2.1 addItem(String Name, String Description, String Picture, Double Price, String Status)

Algorithm:

addItem takes information about an item to add new item to CAP database.

Pseudo code:

```
function addItem(String Name, String Description, String Picture, Double price)
begin function:
if DB connected
    begin if:
    conn.query('INSERT INTO Items VALUES(Name, Description, Picture, Price;)')
    end if
end function
```

12.2.2 reportItem(String Report)

Algorithm:

reportItem takes user report and adds it to an item.

Pseudo code:

```
 function reportItem(String Report)
 begin function:
 if DB connected
    begin if:
    conn.query('INSERT INTO Report VALUES(Report) WHERE name = item.name')
    end if
end function
```

12.2.3 editItem(String n, String d, String Pic, Double Price)

Algorithm:

editItem takes information about an item and update the current information of the item.

Pseudo code:

```
    function editItem(String n, String d, String Pic, Double Price)
    begin function:
    if DB connected
        begin if:
        conn.query('UPDATE Items SET name = n, description = d, picture = pic,
price = pric,')
        end if
   end function
```

12.3 Mapping between requirements and implemented functions

The following table shows the files which impalement each requirements

Functional requirement	Files which implement this requirement
SR1	register(U).php
SR2	register(C).php
SR3	register(S).php
SR4	index.php
SR5	
SR6	
SR7	
SR8	forgetPassword.php changePassword.php
SR9	
SR10	
SR11	
SR12	signout.php
SR13	
SR14	
SR15	
SR16	Account.php
SR17	Account(S).php
SR18	Account(C).php
SR19	Account(A).php
SR20	homepage.php
SR21	orderInfo(C).php
SR22	homepage.php
SR23	homepage.php
SR24	addTocart.php Cart.php
SR25	Cart.php
SR26	Cart.php
SR27	Cart.php
SR28	Cart.php PaymentAPI.php
SR29	checkoutHandle.php mail.php

SR30	checkoutHandle.php
SR31	Clist.php
SR32	Slist.php
SR33	Slist.php
SR34	Clist.php
SR35	Slist.php
SR36	Clist.php
SR37	NewSlist.php
SR38	NewClist.php
SR39	addItem.php
SR40	homepage(S).php
SR41	editItem.php
SR42	editItem.php
SR43	Plist.php
SR44	Plist.php
SR45	order(S).php mail.php
SR46	order(C).php
SR47	AssignedOrders.php
SR48	order(S).php
SR49	SubmitOrder(C).php
SR50	writeReport.php
SR51	viewReport.php
SR52	Wisheslistprocess.php
SR53	Wisheslistpage.php
SR54	Removefavorite.php
SR55	order(S).php
SR56	previousOrders(S).php
SR57	order(U).php
SR58	order(U).php

Table 12.3.1 Mapping between requirements and implemented functions

12.4 Implementation details

In this section will show the coding details of some use cases.

12.4.1 Register a user

- Form which take the user information and location

```
<form action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>"  method="POST"
class="w3-container w3-animate-right w3-round-large w3-monospace" style=
"background-color:LightGray;
  margin-top: 10px;
  margin-bottom: 1px;
  margin-right: 400px;
  margin-left: 400px;
  border-style: groove;
  border-color: rgb(136, 0, 68);
  border-width : 10px">
    <h3 class = "w3-cursive"> Register </h3>
    <div class="inputbox">
  <label for="name"><b>Name</b></label> <br>
  <input class="w3-input w3-border w3-round" type="text" placeholder="Enter name"
name="name" required>
   <br>
   <br>
</div>
    <div class="inputbox">
  <label for="email"><b>Email</b></label> <br>
  <input class="w3-input w3-border w3-round" type="email" placeholder="Enter
Email" name="email" required>
  <br>
  <br>
</div>
<div class="inputbox">
  <label for="psw"><b>Password</b></label> <br>
  <input class="w3-input w3-border w3-round" type="password" placeholder="Enter
Password" name="psw" required>
   <br>
   <br>
</div>
<div class="inputbox">
  <label for="phone"><b>Phone number</b></label> <br>
  <input class="w3-input w3-border w3-round" type="text" placeholder="Enter Phone
number"
   name="phone" required>
   <br>
   <br>
</div>

<div>
     <!-----google-maps----->
  <label for="location"><b>Location</b></label>
  <br>
  <br>
<input type="hidden" id="latitude" name="lat">
<input type="hidden" id="longitude" name="lng">

<div id="map" style="height: 500px"></div>
```

```
<br>
</div>

<!-----google-maps----->
  <input type="submit" class="w3-button w3-indigo w3-round w3-border w3-border-
black" name = "reg"
   value="Register"
  style = " border : 2px">
  <br> <br>
  </form>
  <br> <br>
</div>


  <input type="submit" class="w3-button w3-indigo w3-round w3-border w3-border-
black" name = "reg"
   value="Register"
  style = " border : 2px">
  <br> <br>
  </form>
```

– Validate the user input and register the user information and location in the database

```
if(isset($_POST['reg']))

 {

  $name = $_POST['name'];
  $email = $_POST['email'];
  $psw = $_POST['psw'];
  $phn = $_POST['phone'];
  $lat = $_POST['lat'];
  $lng = $_POST['lng'];
 if (strlen($psw) < 4 ) {
   echo "<script> alert(\"number of alphanumeric for password must be more than
4\"); </script>";
 }
 else if ($lat > 24.971320 || $lat < 24.526383 || $lng > 46.843450 || $lng <
46.564269)  // if the location not inside riyadh
{
  echo "<script> alert(\"The location is out of CAP boundaries\"); </script>";
 }
 else {
  $psw = sha1($psw); //encrypt the user password
 $sql = "insert into user (user_name, Email,  Password,
phone_number,latitude,longitude)
 VALUES ( '$name', '$email', '$psw', '$phn','$lat','$lng') ";

 $result = mysqli_query($con,$sql);
 if ( !$result ) {
   echo "<script>alert(\"email is already used\");</script>";
 }
 else {
  echo "<script>alert(\"You registered successfully\");</script>";
  echo "<script> window.location.replace('index.php'); </script>";
// back to log in page
 }

 }
 }
```

12.4.2 Log in a user

- Form to get email and password of the user, seller, captain or admin

```
<form method = "POST" action="<?php echo htmlspecialchars($_SERVER['PHP_SELF']);
?>"
class="w3-container w3-animate-right w3-round-large w3-monospace" style=
"background-color:LightGray;
  margin-top: 10px;
  margin-bottom: 1px;
  margin-right: 400px;
  margin-left: 400px;
  border-style: groove;
  border-color: rgb(136, 0, 68);
  border-width : 10px">

  <h3 class = "w3-cursive">Login to CAP </h3>
  <div class="inputbox">
  <label for="email"><b>Email</b></label> <br>
  <input class="w3-input w3-border w3-round" type="text" placeholder="Enter
Email" name="email" required>
  <br>
  <br>
</div>
<div class="inputbox">
  <label for="psw"><b>Password</b></label> <br>
  <input class="w3-input w3-border w3-round" type="password" placeholder="Enter
Password" name="psw" required>

  <br>

</div>

  <input type="submit" class="w3-button w3-indigo w3-round w3-border w3-border-
black" name = "log" value="Login"
  style = "border : 2px">
  <br>
  <a href="selectRole.php" class="w3-hover-blue"> Create an account</a>  <br>
  <a href="forgetPassword.php" class="w3-hover-blue"> Forget your password</a>
  <br> <br>
  </form>
```

- validate input and log in the user, seller, captain, or admin in the database

```
if ( $_SERVER["REQUEST_METHOD"] == "POST" ) {
  $email = $_POST["email"];
  $psw = $_POST["psw"];


  $sql = "select * from admin where email =  '$email' AND  password = '$psw' ";
  $result = mysqli_query($con, $sql);
  $row = mysqli_fetch_array($result);
  $count = mysqli_num_rows($result);
  if ($count == 1 ) { // if there is one matched account
    $_SESSION["useremailA"] = $email;
    header("Location: homepage(A).php");
    exit();
  }

  $psw = sha1($psw); // Decrypt the password
  $sql = "select * from user where Email =  '$email' AND  Password = '$psw' ";
  $result = mysqli_query($con, $sql);
  $count = mysqli_num_rows($result);
  $row = mysqli_fetch_assoc($result);


  if ($count == 1 ) {  // if there is one matched account
    $_SESSION["useremail"] = $email;
    $_SESSION["userid"] = $row['UserID'];
    $sql = "DELETE FROM cart";
  $result = mysqli_query($con, $sql);
    header("Location: homepage.php");
    exit();
  }
  $sql = "select * from seller where Email =  '$email' AND  password = '$psw' ";
  $result = mysqli_query($con, $sql);
  $count = mysqli_num_rows($result);
  $row = mysqli_fetch_array($result);

  if ($count == 1 && $row['Accepted_registration'] == 1)  // if there is one
matched account and the seller is accepted
  {
    $_SESSION["useremailS"] = $email;
    $_SESSION["id"] = $row['SellerID'];
    header("Location: homepage(S).php");
    exit();
  }

    $sql = "select * from captain where email =  '$email' AND  password = '$psw'
";
  $result = mysqli_query($con, $sql);
  $count = mysqli_num_rows($result);
  $row = mysqli_fetch_array($result);

  if ($count == 1 && $row['Accepted_registration'] == 1) {  // if there is one
matched account and the captain is accepted
    $_SESSION["CaptainID"] = $row['CaptainID'];
    $_SESSION["useremailC"] = $email;
    header("Location: homepage(C).php");
    exit();
  }
    else {
    echo " <script> alert(\"Wrong email or password\"); </script>";
    }

}
```

12.4.3 Add to cart

-The user chooses a specific product to be added to the cart

```
$pid = $_POST['pid']; // Product id choosen by the user
$price = $_POST['price']; // product price
$uid = $_SESSION["userid"];  // user id

$sql2 = "select * from cart where ProductID = '$pid' ";
$result2 = mysqli_query($con, $sql2);
$count = mysqli_num_rows($result2);
if ($count == 1) { //if the product already exist in the cart
    echo " <script> alert(\"The Product is already exist in the cart\");
</script>"; // message
    echo "<script> window.location.replace('homepage.php'); </script>"; // back
to home page
}
else { // the product is not exist in the cart, so add it.
$sql = "insert into cart (ProductID, UserID, Quantity, Total_Price)
 VALUES ('$pid', '$uid', 1, '$price') ";
    $result = mysqli_query($con,$sql);
    if ($result) {
    echo " <script> alert(\"The Product is added to Cart\"); </script>"; //
message
    echo "<script> window.location.replace('homepage.php'); </script>";  // back
to home page
    }
}
```

12.4.4 increase items quantity

- The user clicks increase button to increase the quantity of product items by one item more

```
if (isset($_POST['increase'])) { // user clicks increase
  $id = $_POST['ID']; // product id
  $sql = "SELECT * FROM cart, product where ID = ProductID AND ProductID =
'$id'";
  $result = mysqli_query($con,$sql);
  $row = mysqli_fetch_array($result);
  $inputQuantity = $row['Quantity']; // quantity of items of the products that
added to the cart
  $actualQuantity = $row['Item_Quantity']; // avalilable product quantity
registered in the system
```

```
  if ($inputQuantity < $actualQuantity ){ // purchased quantity should not
exceeds the actual quantity of the product in the stock

  $query = "UPDATE cart SET Quantity = Quantity + 1 WHERE ProductID='$id'"; //
Update the quantity of the product in the cart
  $query_run = mysqli_query ($con, $query); // run the query

  $sql1 = "SELECT * FROM product where ID = '$id'";
  $result1 = mysqli_query($con,$sql1);
  $row1 = mysqli_fetch_array($result1);
  $itemPrice = $row1['Price'];
  $query1 = "UPDATE cart SET Total_Price = ('$itemPrice' * Quantity) WHERE
ProductID='$id'"; // Update the total price of a specific product
  $query_run1 = mysqli_query ($con, $query1);
  }
  else {
    echo " <script> alert(\"You exceeds the maximum quantity of this product\");
</script>";
  }


}
```

12.4.5 Decrease items quantity

- The user clicks decrease button to decrease the quantity of product items by one item less

```
if (isset($_POST['decrease'])) { // user clicks decrease
  $id = $_POST['ID']; // product id
  $sql = "SELECT * FROM cart where ProductID = '$id'";
  $result = mysqli_query($con,$sql);
  $row = mysqli_fetch_array($result);
  $quantity = $row['Quantity'];
  if ($quantity > 1){
  $query = "UPDATE cart SET Quantity = Quantity - 1 WHERE ProductID='$id'";
  $query_run = mysqli_query ($con, $query);

  $sql1 = "SELECT * FROM product where ID = '$id'";
  $result1 = mysqli_query($con,$sql1);
  $row1 = mysqli_fetch_array($result1);
  $itemPrice = $row1['Price'];
  $query1 = "UPDATE cart SET Total_Price = ('$itemPrice' * Quantity) WHERE
ProductID='$id'";
  $query_run1 = mysqli_query ($con, $query1);
  }
  else { // if the qunatity of items is 1, the product will be removed from cart.
    $query = "DELETE FROM cart WHERE ProductID='$id'";
    $query_run = mysqli_query ($con, $query);

  }


}
```

12.4.6 Handle the order after payment

```
 $sql = "SELECT * FROM cart";
    $result = mysqli_query($con,$sql);
    $row = mysqli_fetch_array($result);
    $confirmation_code= rand(10000,99999); // generate random code with five
digits
    $userID = $row['UserID']; // the user who complete the order
    $date = date("Y/m/d"); // get today's date to make it the date of order
    $mode = "";

    //determine the delivary mode
    if($_SESSION['install'] == 0) {
        $mode = 'delivary';
    }
    else {
        $mode = 'delivary and install';
    }


   $sql1 = "Select SUM(Total_Price) from cart";
    $result1 = mysqli_query($con,$sql1);
    $row1 = mysqli_fetch_array($result1);
    $cost = $row1['SUM(Total_Price)'] + 30 + $_SESSION['install']; // calculate
the total cost of the order


    $sql2 = "Select SUM(Quantity) from cart";
    $result2 = mysqli_query($con,$sql2);
    $row2 = mysqli_fetch_array($result2);
    $quantity = $row2['SUM(Quantity)']; // get the total number of all products
quantity

    $sql9 = "Select COUNT(ProductID) from cart";
    $result9 = mysqli_query($con,$sql9);
    $row9 = mysqli_fetch_array($result9);
    $num_product = $row9['COUNT(ProductID)']; // determine the number of ordered
product (not qunatity of items)

    $sql = "insert into orders (Confirmation_code, order_status, Date, userID,
delivary_mode,
    cost, order_quantity, number_of_product)
 VALUES ('$confirmation_code', 'ordered', '$date', '$userID', ' $mode', '$cost',
'$quantity',
  $num_product) "; // add the order information in the database

 $result = mysqli_query($con,$sql);


 $sql6 = "SELECT order_ID FROM orders where Confirmation_code =
'$confirmation_code' ";
 // as the order number is auto increament, so we get here the order number by
the confirmation code
 $result6 = mysqli_query($con,$sql6);
$row6 = mysqli_fetch_array($result6);
$orderID = $row6['order_ID'];
 $sql4 = "SELECT * FROM cart";
 $result4 = mysqli_query($con,$sql4);


 while($row4 = mysqli_fetch_array($result4)) { // add the ordered product in the
order_product table in the database
    $productID = $row4['ProductID'];
    $OPquantity = $row4['Quantity'];
    $sql5 = "insert into ordered_product (OP_OrderId, OP_ProductID, OP_quantity)
 VALUES ('$orderID','$productID', '$OPquantity') ";
 $result5 = mysqli_query($con,$sql5);
}

$email = $_SESSION["useremail"]; // get the email of user who do the order
$msg = 'Thank you for choosing CAP <br>
        Order number:' . $orderID . '<br>
        Confirmation code: '. $confirmation_code . '<br>
        The Captain will contact you when the order becomes ready to be delivered
and installed' ;

            sendmail($email, 'Order info' , $msg);
 $sql3 = "DELETE FROM cart"; // delete the items in the cart
 $result3 = mysqli_query($con,$sql3);
 echo " <script> alert(\"The order is completed\"); </script>"; // message to
show the current system status
 echo "<script> window.location.replace('homepage.php'); </script>"; // back to
home page
```

12.4.7 Change order status to ready

```
if (isset($_POST['confirm'])) { // if seller clicks ready for a product
  $pid1 = $_POST['pid3']; // get the confirmed product by the seller
  $orderID = $_POST['orderId']; // get the order id of confiremed prodcut
  $quantity = $_POST['quantity']; // get the quantity of ordered product
  $sql = "insert into ready_product (RI_ProductID, RI_OrderID)
 VALUES ('$pid1','$orderID') "; // add the product to ready products table
 $result = mysqli_query($con,$sql);


$sql1 = "SELECT number_of_product FROM orders WHERE order_ID = '$orderID'  ";
 $result1 = mysqli_query($con,$sql1);
 $row1 = mysqli_fetch_array($result1);
$n1 = $row1['number_of_product']; // the number of ordered product of this order

$sql2 = "SELECT COUNT(RI_OrderID) FROM ready_product WHERE RI_OrderID =
'$orderID'  ";
 $result2 = mysqli_query($con,$sql2);
 $row2 = mysqli_fetch_array($result2);
$n2 = $row2['COUNT(RI_OrderID)'];// the number of ready product of this order

 if ($n1 <= $n2) { // if number of ready product equals the number of ordered
products, the order status will changed to ready, otherwise it will remains
ordered
  $sql3 = "UPDATE orders SET order_status = 'ready' WHERE order_ID = '$orderID'";
  $result3 = mysqli_query($con,$sql3);

  $sql4 = "SELECT Email FROM captain";
  $result4 = mysqli_query($con,$sql4);

  while ($row4 = mysqli_fetch_array($result4)) { // send to all captains about
the new ready order
    $email = $row4['Email'];
    $msg = 'Please check the new orders list in your account if you available<br>
    There is unassigned order<br>
    Order number: ' . $orderID . '<br> Thank you' ;
    sendmail($email, 'New order' , $msg);
  }
 }

  $sql6 = "UPDATE product SET Item_Quantity = (Item_Quantity - '$quantity') WHERE
ID = '$pid1' "; // decrease the number of products in the database for the
ordered ready products
   $result6 = mysqli_query($con,$sql6);
}
```

12.4.8 Change order status to delivered

```
if (isset($_POST['confirm'])) { // captain provide the confirmation code
    $orderID = $_POST['orderID1']; // get targeted order ID
    $conCode = $_POST['conCode'];  // get the provided confirmation code
    $sql = "SELECT Confirmation_code FROM orders WHERE order_ID = '$orderID'";
    $result = mysqli_query($con,$sql);
    $row = mysqli_fetch_array($result);
    $orderConCode = $row['Confirmation_code']; // get the actual confirmation
code of the order
    if ($conCode === $orderConCode) { // if the provided confirmation code is
correct
    $sql = "UPDATE orders SET order_status = 'delivered' WHERE order_ID =
'$orderID'";
    $result = mysqli_query($con,$sql);
    $sql = "DELETE FROM assigned_order WHERE AO_OrderId = '$orderID'";
    $result = mysqli_query($con,$sql);

    echo " <script> alert(\"The order is submitted successfully\"); </script>";
    echo "<script> window.location.replace('homepage(C).php'); </script>";
    }
    else { // Wrong provided confirmation code
        echo " <script> alert(\"Wrong confirmation code\"); </script>";
       echo "<script> window.location.replace('AssignedOrders.php'); </script>";
    }
 }
```

12.4.9 add new product

```
if(isset($_POST['add'])) // the seller click add
{
// product info
 $name = $_POST['name'];
 $price = $_POST['price'];
 $description = $_POST['description'];
 $quantity = $_POST['quantity'];
 $sid =  $_SESSION['id']; // seller ID
if($price <= 0 || $quantity <= 0) {
  echo " <script> alert(\"Wrong inputs\"); </script>";
  echo "<script> window.location.replace('addItem.php'); </script>";
  exit();
}
 // handle the input image
 $image = $_FILES['image']['name'];
 $imageTmp = $_FILES['image']['tmp_name'];
 $imageError = $_FILES['image']['error'];
 $imageSize = $_FILES['image']['size'];
 $imageExt = explode('.', $image);
$imageActualExt = strtolower(end($imageExt));
$allowed = array('jpg', 'jpeg', 'png');

if(in_array($imageActualExt,$allowed) && $imageError === 0 && $imageSize <
1000000) {
  $imageNewName =  uniqid('',true) . "." . $imageActualExt;
  $imageDestenation = 'images/' . $imageNewName;
  move_uploaded_file($imageTmp, $imageDestenation);

$sql = "insert into product (Name, price, description, Item_Quantity, image,
SellerID)
VALUES ( '$name', '$price', '$description','$quantity',
'images/$imageNewName',$sid) ";
$result = mysqli_query($con,$sql);
if ($result) {
echo "<script> alert(\" The item is added successfully \")</script>";
echo "<script> window.location.replace('homepage(S).php'); </script>";
}
else {
  echo "<script> alert(\" Item failed to be added \")</script>";
}
} else {
  echo "<script> alert(\"Wrong inputs\")</script>";
}

}
```

12.4.10 Delete product by seller

```
if (isset($_POST['delete'])) {  // if user clicks delete for a specific product
  $id = $_POST['ID']; // get the ID of targeted product
  $query = "DELETE FROM product WHERE ID='$id'";
  $query_run = mysqli_query ($con, $query);
  if($query_run){
  echo "<script> alert(\"The item is deleted successfuly\"); </script>";
```

```
    }
    else {
      echo "<script> alert(\"Failed to be deleted\"); </script>";
    }
  }
```

12.5 Database Schema

Here we show the database schema for CAP system:

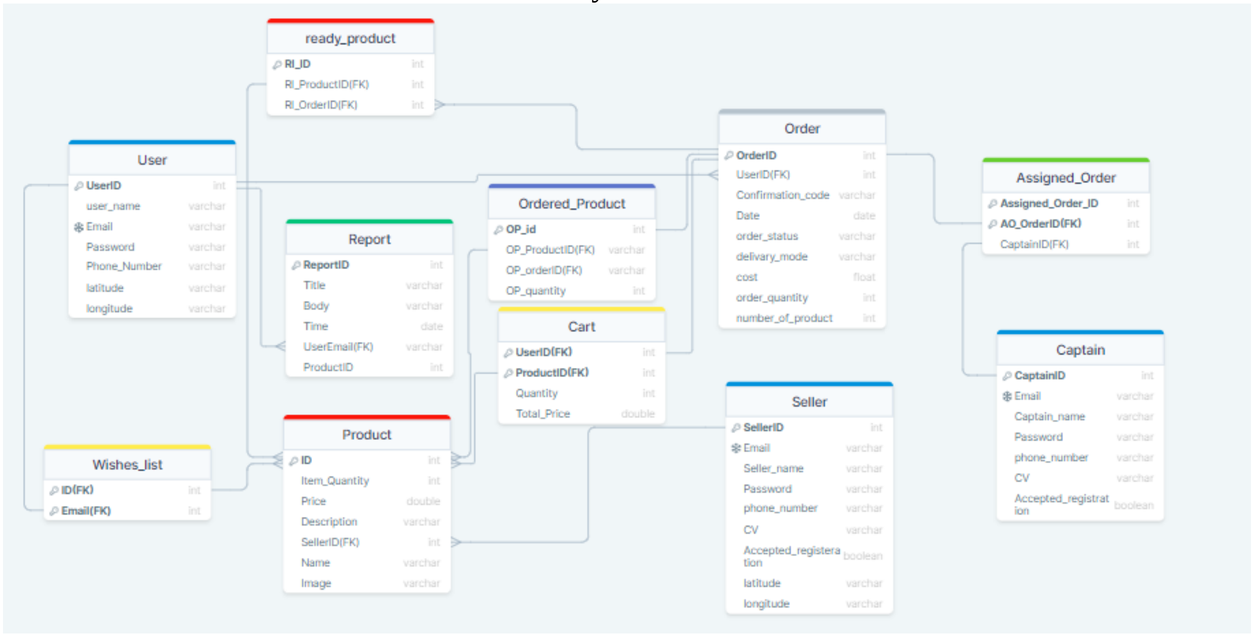


Figure 12.5.1 Updated Database schema

12.6 User interface

In this section we will show the interface of some use cases.

12.6.1 Log in a user

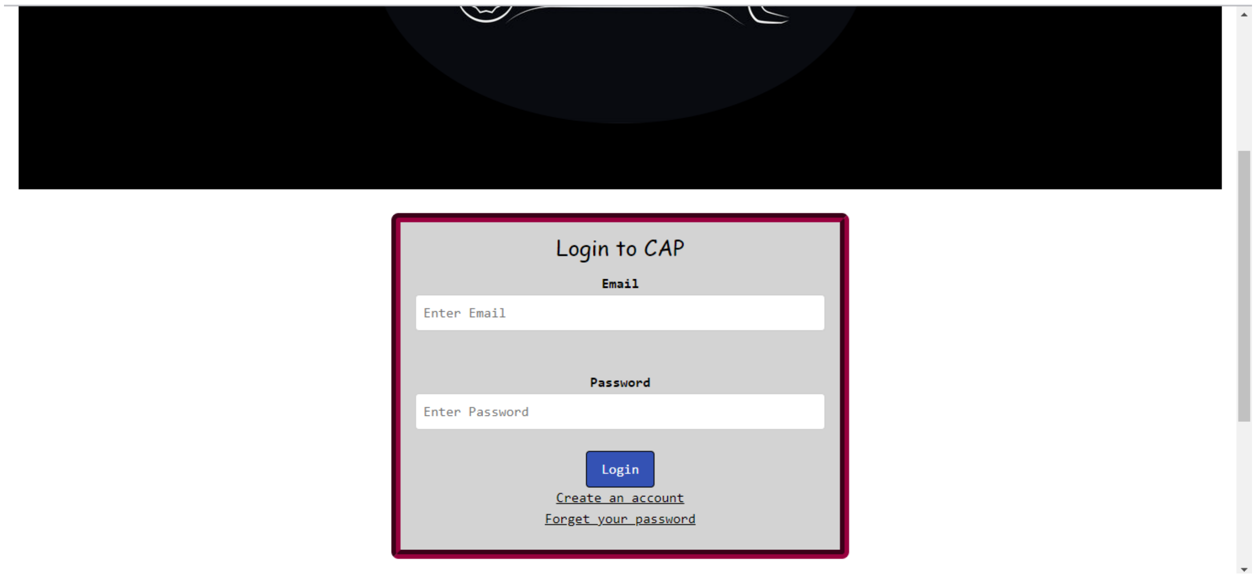
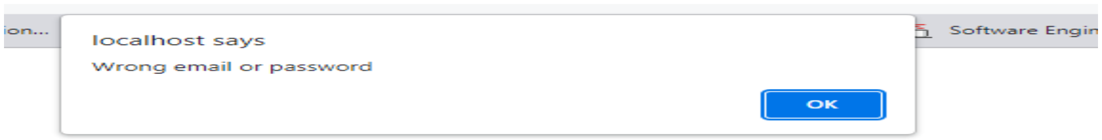


Figure 12.6.1.1 log in form



localhost says

Wrong email or password

Figure 12.6.1.2 Wrong inputs of log in form

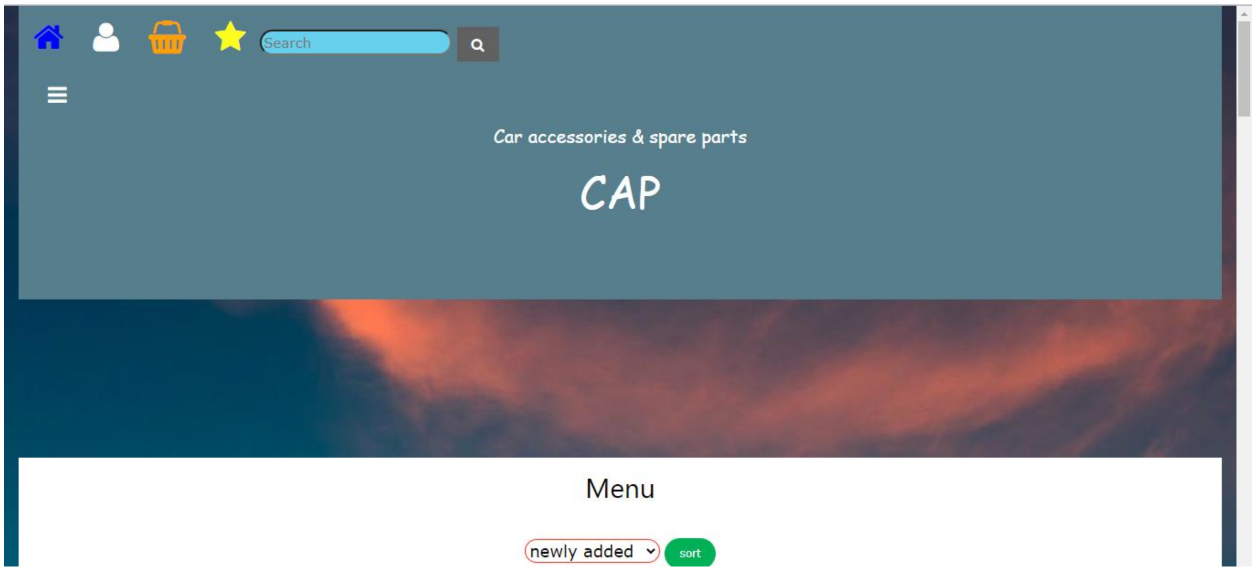


Figure 12.6.1.3 Valid log in input (Home page will be opened)

12.6.2 Register a user

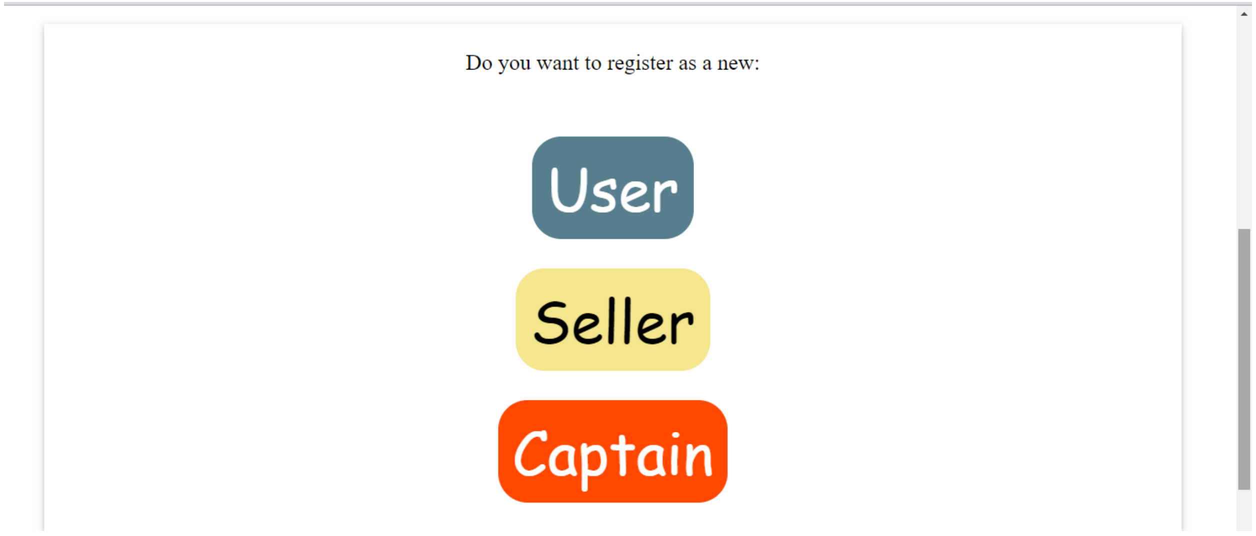


Figure 12.6.2.1 Choose role for registration

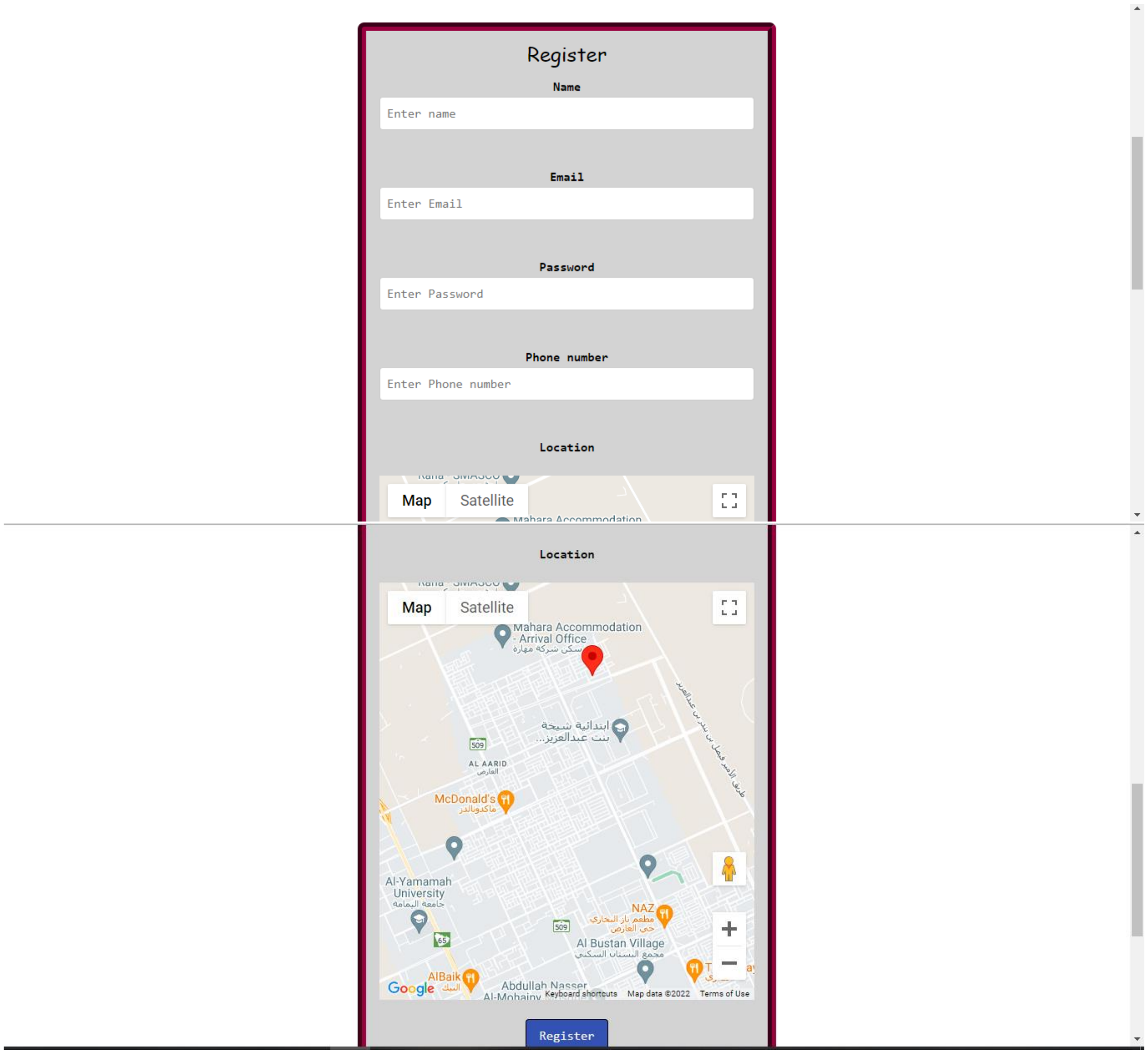
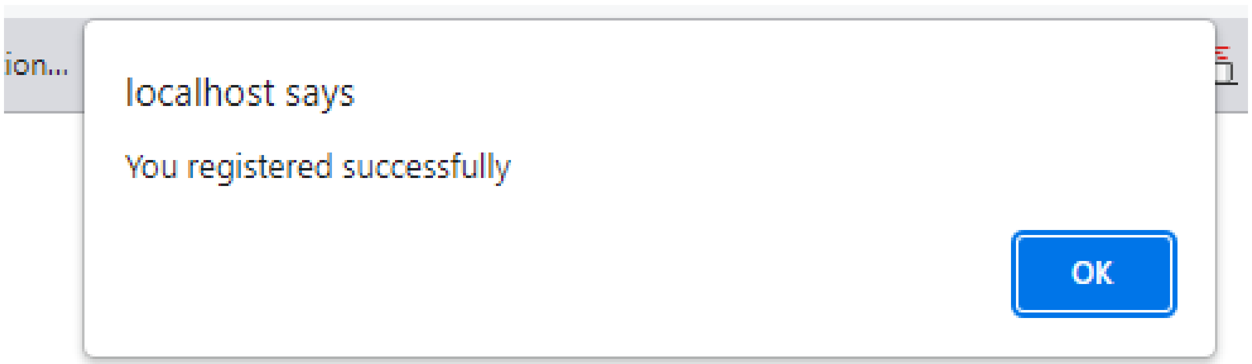


Figure 12.6.2.2 Registration form



localhost says

You registered successfully

Figure 12.6.2.3 Status message if the registration information is valid

12.6.3 Purchase items

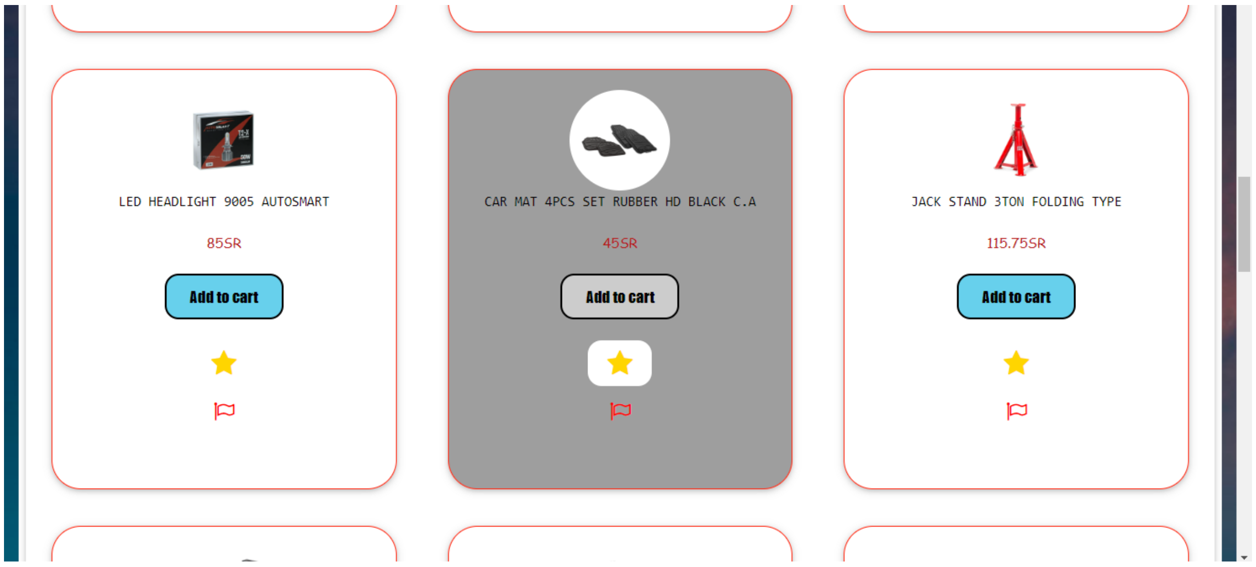

LED HEADLIGHT 9005 AUTOSMART
85SR
Add to cart
CAR MAT 4PCS SET RUBBER HD BLACK C.A
45SR
Add to cart
JACK STAND 3TON FOLDING TYPE
115.75SR
Add to cart

Figure 12.6.3.1 View products and add to cart

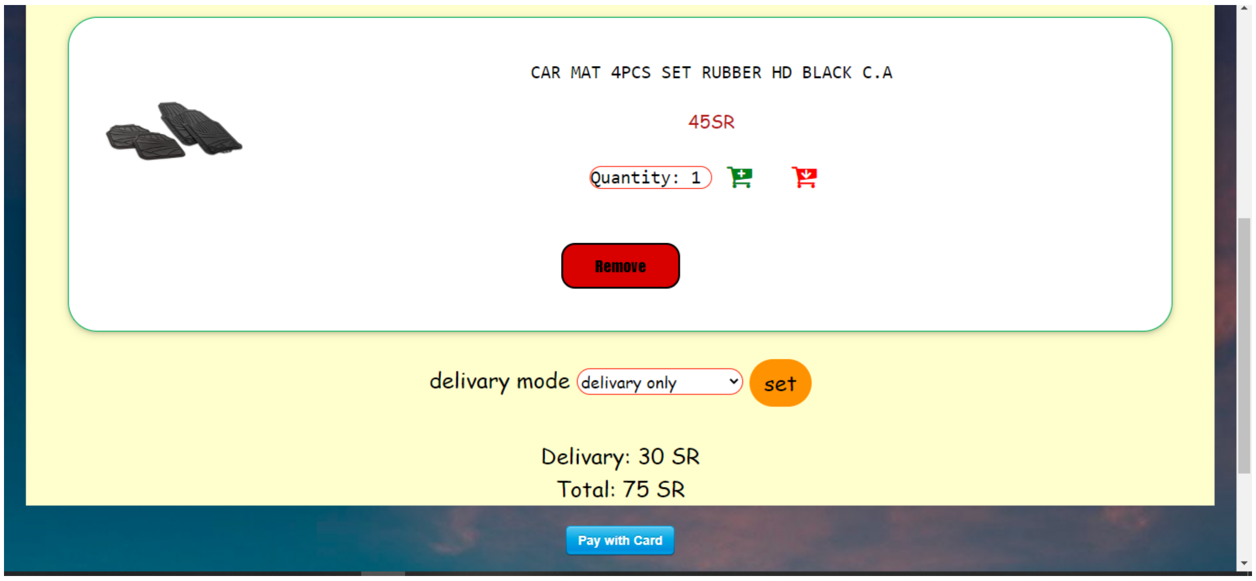
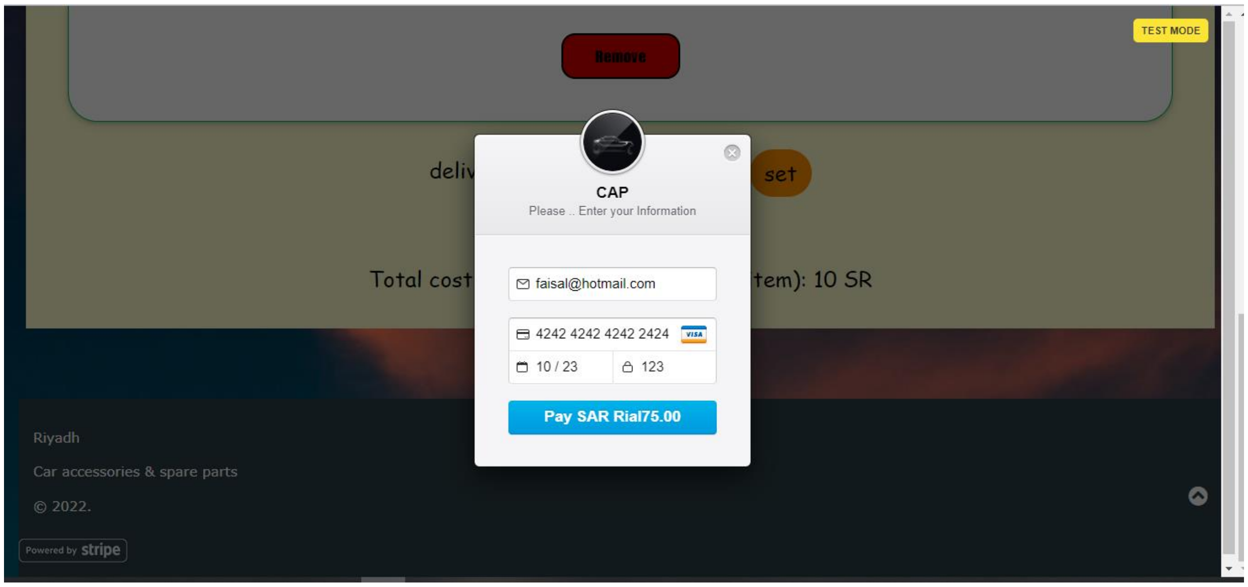


Figure 12.6.3.2 View items in the cart with the cart operations
(I.e. choose delivery mode, change quantities, remove items from cart, view order pricing info, and click pay).


TEST MODE

Remove

CAP

Please .. Enter your Information

faisal@hotmail.com

4242 4242 4242 2424

10 / 23

123

Pay SAR Rial75.00

deliv

set

Total cost

tem): 10 SR

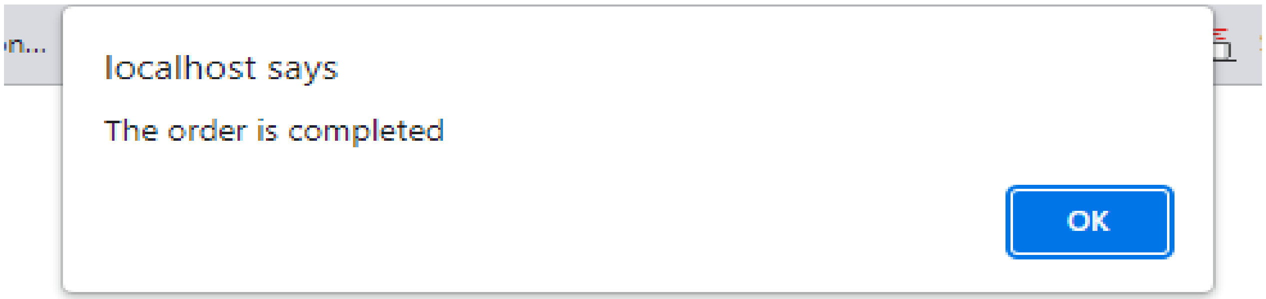
Riyadh

Car accessories & spare parts

© 2022.

Powered by stripe

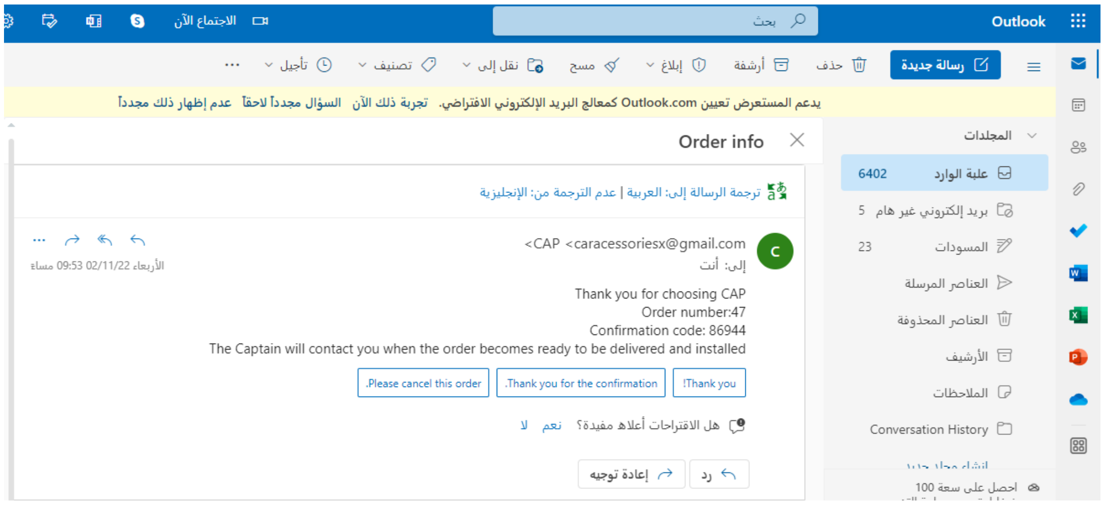
Figure 12.6.3.3 Provide payment info


localhost says

The order is completed

OK

Figure 12.6.3.4 message for successfully completed order

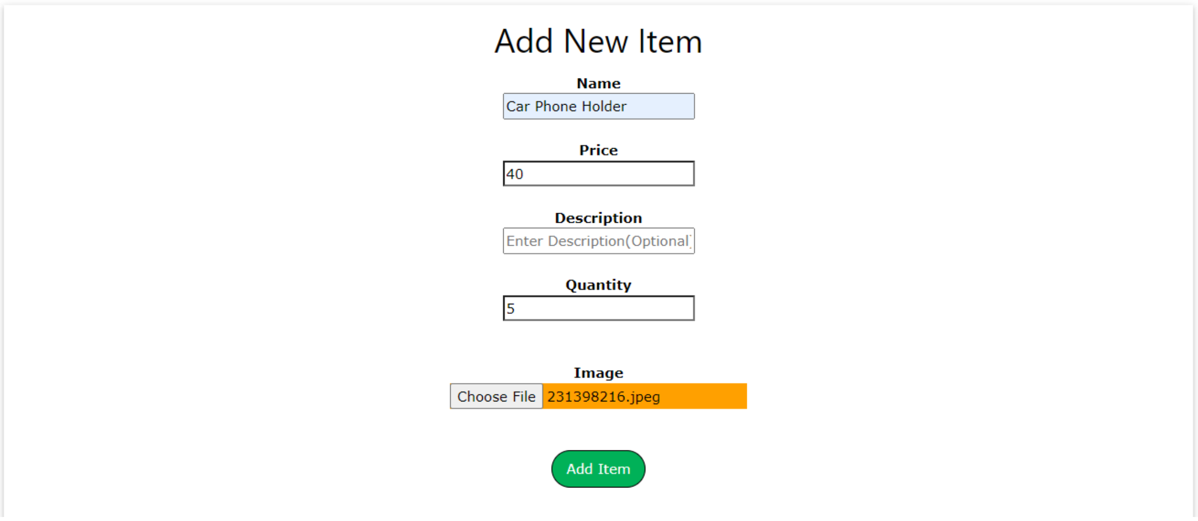


<CAP <caracessoriesx@gmail.com
إلى: أنت
الأربعاء 02/11/22 09:53 مساء

Thank you for choosing CAP
Order number:47
Confirmation code: 86944
The Captain will contact you when the order becomes ready to be delivered and installed

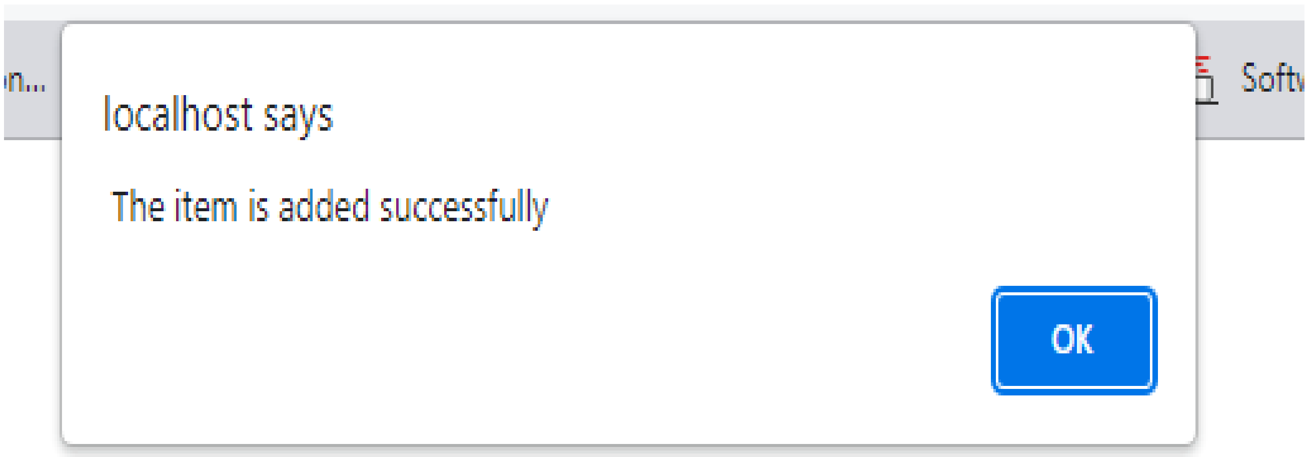
Figure 12.6.3.5 Email message to the user about the order

12.6.4 Add new item


Add New Item

Name	Car Phone Holder
Price	40
Description	Enter Description(Optional
Quantity	5
Image	Choose File 231398216.jpeg

Figure 12.6.4.1 new item details


localhost says

The item is added successfully

Figure 12.6.4.2 Confirmation message for item addition

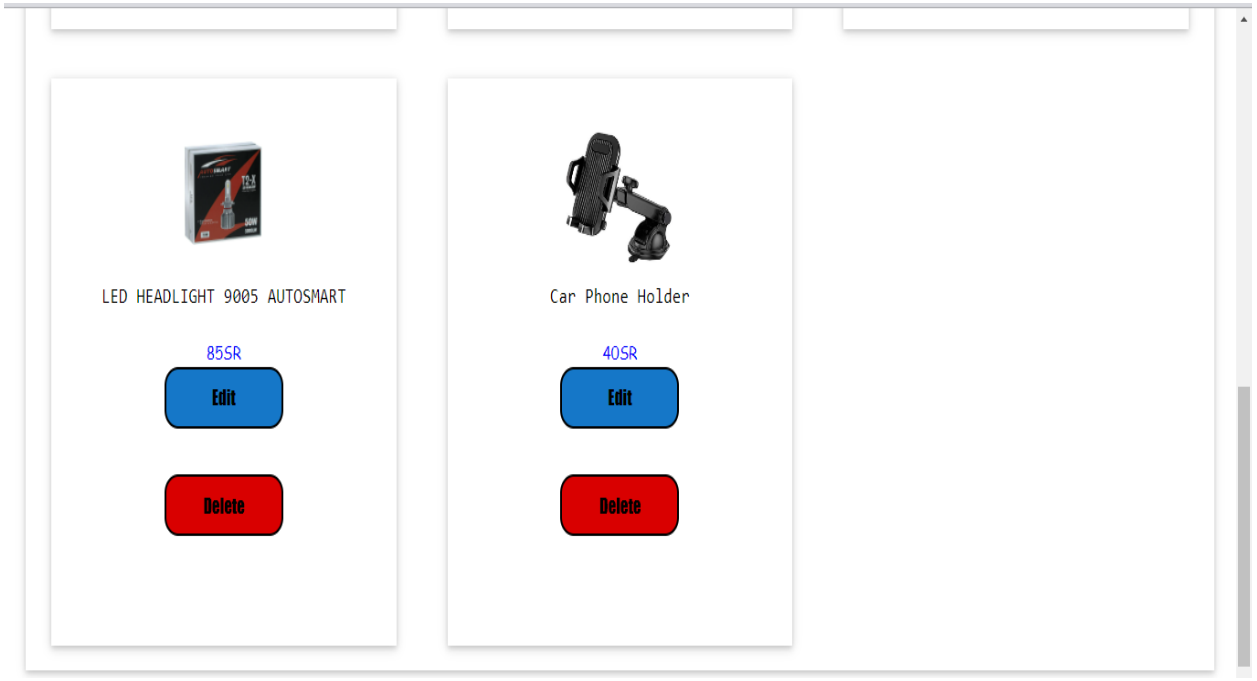


Figure 12.6.4.3 the new item appear in the seller products list

13. Testing

In this section we shows how we document the test cases which used to check if the system is bug free and complied with the requirement, and give the system to some participant to give their opinions about the usability of the system.

13.1 Test Scenario

In this section we show the test scenario for some units in our system.

Module name:	Register user								
Test case ID	Test Scenario	Test case	Pre-condition	Test Steps	Test Data	Expected result	Post condition	Actual result	Status (Pass /Fail)
TC_REGISTER_001	Register new account to the website with valid inputs	Enter Valid name, valid email, valid password , valid phone number, and valid location	none	1-enter name 2-enter email 3-enter password 4-enter phone number 5- select location 6- click register	1-mohammed 2-imj3600i@gmail.com 3- 123456 4- 0555555554 5-Lat:24.548058897989648 Long:46.68185191491699	registered successfully	Massage appears Says "You registered successfully"	registered successfully	pass
TC_REGISTER_002	Register new account to the website with invalid password	Enter Valid name, valid email, invalid password , valid phone number, and valid location	none	1-enter name 2-enter email 3-enter password 4-enter phone number 5- select location 6- click register	1-mohammed 2-imj36000i@gmail.com 3- 123 4- 0555555554 5- location	Register rejected and message appears "number of alphanumeric for password must be more than 4"	Message appears "number of alphanumeric for password must be more than 4"	Register rejected and message appears "number of alphanumeric for password must be more than 4"	pass

Table 13.1.1 Register a user test cases

Module name:	Login								
Test case ID	Test Scenario	Test case	Pre-condition	Test Steps	Test Data	Expected result	Post condition	Actual result	Status (Pass/Fail)
TC_Login_001	Verify login to the website	Enter valid email and valid password	Require already registered account	1-enter email 2-enter password 3- click login	1- imj3600i@gmail.com 2- 123456	Login successfully to website	Homepage Is shown	Login successfully to website	pass
TC_Login_002	Verify login to the website	Enter invalid email and valid password	Require already registered account	1-enter email 2-enter password 3- click login	1- imj36i@gmail.com 2- 123456	A message " wrong email or password" shown	Message appears " wrong email or password"	A message " wrong email or password " shown	pass
TC_Login_003	Verify login to the website	Enter valid email and invalid password	Require already registered account	1-enter email 2-enter password 3- click login	1- imj3600i@gmail.com 2- 1234567	A message " wrong email or password" shown	Message appears " wrong email or password"	A message " wrong email or password " shown	pass

Table 13.1.2 Log in a user test cases

Module name:	Forget password								
Test case ID	Test Scenario	Test case	Pre-condition	Test Steps	Test Data	Expected result	Post condition	Actual result	Status (Pass/Fail)
TC_Forget password_001	Verify forget password function with valid inputs	Enter email and select account type	Account already existed in the system	1-enter email 2- select account type 3- click change	1- imj3600i@gmail.com 2- select user	Message sent to the user email with new password	Message is shown " The message is sent"	Message sent to the user email with new password	pass
TC_Forget password_002	Verify forget password function with invalid email	Enter email and select account type	Account already existed in the system	1-enter email 2- select account type 3- click change	1- imj36000i@gmail.com 2- select user	Message is shown "This email is not registered"	Message is shown "This email is not registered"	Message is shown "This email is not registered"	pass
TC_Forget password_003	Verify forget password function with invalid account type	Enter email and select account type	Account already existed in the system	1-enter email 2- select account type 3- click change	1- imj3600i@gmail.com 2- select seller	Message is shown "This email is not registered"	Message is shown "This email is not registered"	Message is shown "This email is not registered"	pass

Table 13.1.3 Forget password test cases

Module name:	Purchase an item								
Test case ID	Test Scenario	Test case	Pre-condition	Test Steps	Test Data	Expected result	Post condition	Actual result	Status (Pass/Fail)
TC_ Purchase an item _001	Verify purchase an item function with valid inputs	Add item to cart, go to cart enter quantity and select delivery mode	Logged in to user account	1- add to cart 2- go to cart 3-enter quantity 4- select delivery mode 5- click set 6- click pay with card 7- enter email 8-enter card number 9-enter date 10- enter cvc 11- click pay	1-add: Car Cup holder Double-black 3- Quantity: 3 4- delivery only 7- imj3600i@gmail.com 8- 4242 4242 4242 4242 9- 12/34 10- 123	The order is completed, message sent to user's email with order information and message is shown " The order is completed"	message is shown " The order is completed"	The order is completed, message sent to user's email with order information and message is shown " The order is completed"	pass
TC_ Purchase an item _002	Verify purchase an item function with invalid quantity	Click add to cart, go to cart enter quantity and select delivery mode	Logged in to user account	1- add to cart 2- go to cart 3-enter quantity	1-add: Custom Car Sunshade Silver 2-pieces 3- Quantity: 13	It will not exceed the maximum and message is shown "You exceeds the maximum quantity of this product"	message is shown "You exceeds the maximum quantity of this product"	It will not exceed the maximum and message is shown "You exceeds the maximum quantity of this product"	pass

Table 13.1.4 Purchase items test cases

Module name:	Add item								
Test case ID	Test Scenario	Test case	Pre-condition	Test Steps	Test Data	Expected result	Post condition	Actual result	Status (Pass/Fail)
TC_Add item_001	Verify add item with valid inputs	Enter name, price, quantity and image	Logged in to seller account	1-enter name 2-enter price 3- enter quantity 4- add image 5- click add item	1- Sandberg Car Charger 2USB 1A+2.1A SAVER 2- 40 3- 7 4- car charger.jpg	Item is added successfully And message is shown " The item is added successfully"	message is shown " The item is added successfully"	Item is added successfully And message is shown " The item is added successfully"	pass
TC_ Add item _002	Verify add item with invalid price	Enter name, price, quantity and image	Logged in to seller account	1-enter name 2-enter price 3- enter quantity 4- add image 5- click add item	1- HUAWEI Cp37 Type C Car Charger with Super Charge, Grey 2- 0 3- 7 4- Huawei car charger.jpg	Unable to add item And message is shown "Wrong inputs"	message is shown "Wrong inputs"	Unable to add item and message is shown "Wrong inputs"	pass

Table 13.1.5 Add item test cases

Module name:	Change order status to delivered								
Test case ID	Test Scenario	Test case	Pre-condition	Test Steps	Test Data	Expected result	Post condition	Actual result	Status (Pass/ Fail)
TC_ Change order status to delivered _001	Verify Change order status to delivered with valid input	Accept new order then go to assigned order submit enter confirmation code	Logged in to captain's account	1- accept new order 2- go to assigned order 3-click on submit 4- enter confirmation code 5-click on confirm	4-23004	Submit the order and change status to delivered and message is shown " The order is submitted successfully"	message is shown " The order is submitted successfully"	Submit the order and change status to delivered and message is shown " The order is submitted successfully"	pass
TC_ Change order status to delivered _002	Verify Change order status to delivered with invalid confirmation code	Accept new order then go to assigned order submit enter confirmation code	Logged in to captain's account	1- accept new order 2- go to assigned order 3-click on submit 4- enter confirmation code 5-click on confirm	4-25232	Unable to submit the order and message is shown "Wrong confirmation code"	message is shown "Wrong confirmation code"	Unable to submit the order and message is shown "Wrong confirmation code"	pass

Table 13.1.6 Change order status to delivered test cases

Module name:	Search item								
Test case ID	Test Scenario	Test case	Pre-condition	Test Steps	Test Data	Expected result	Post condition	Actual result	Status (Pass/Fail)
TC_Search_001	Verify search function for an item on the system	Enter item name in search bar	Logged in to user account	1-click on search bar 2-enter item name	2- car charger	Item is shown	Homepage is shown with the searched item only	Item is shown	pass
TC_Search_002	Verify search function for an item not on the system	Enter item name in search bar	Logged in to user account	1-click on search bar 2-enter item name	2- car battery	item is not shown and message is shown "No matching items"	message is shown "No matching items"	item is not shown and message is shown "No matching items"	pass

Table 13.1.7 Search item test cases

13.2 Functional test

In the previous section we show the test scenario for the critical units in our system, so the flow between different units is clear and the functional test is completed for the critical functions like purchasing and complete an order.

13.3 Usability Test

For the usability test, we make a questionnaire with some questions which asks about the interface components and the usability for the user to interact with the system, and then we give it to some participants after they use the system, we get a responses from six participants.

The form questions and results:

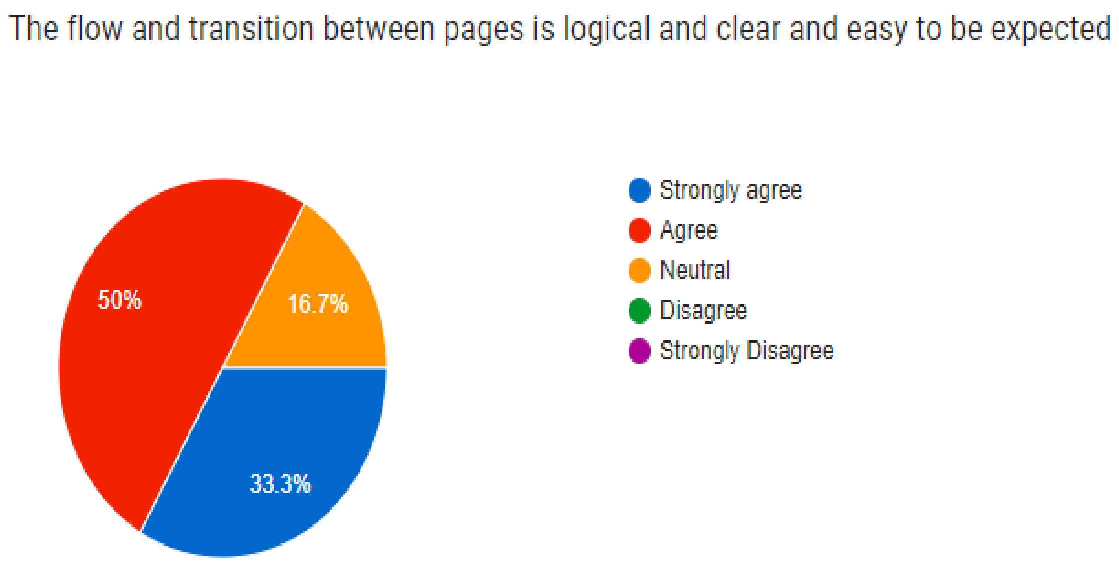


Figure 13.3.1 usability test question 1

Strongly agree: 2
Agree: 3
Neutral: 1
Disagree: 0
Strongly disagree: 0

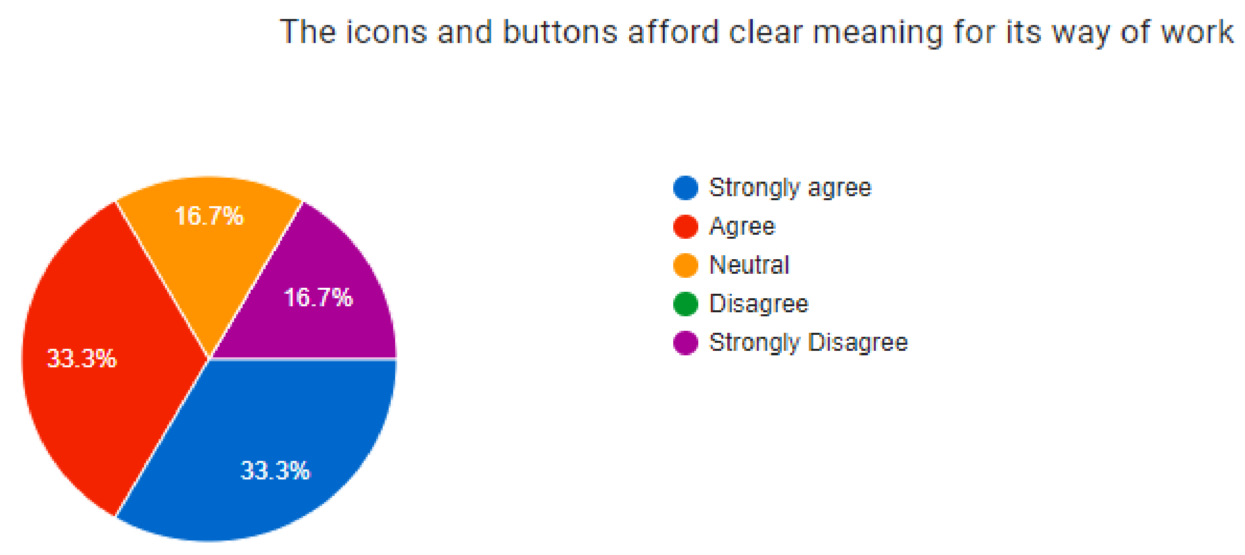


Figure 13.3.2 usability test question 2

Strongly agree: 2
Agree: 2
Neutral: 1
Disagree: 0
Strongly disagree: 1

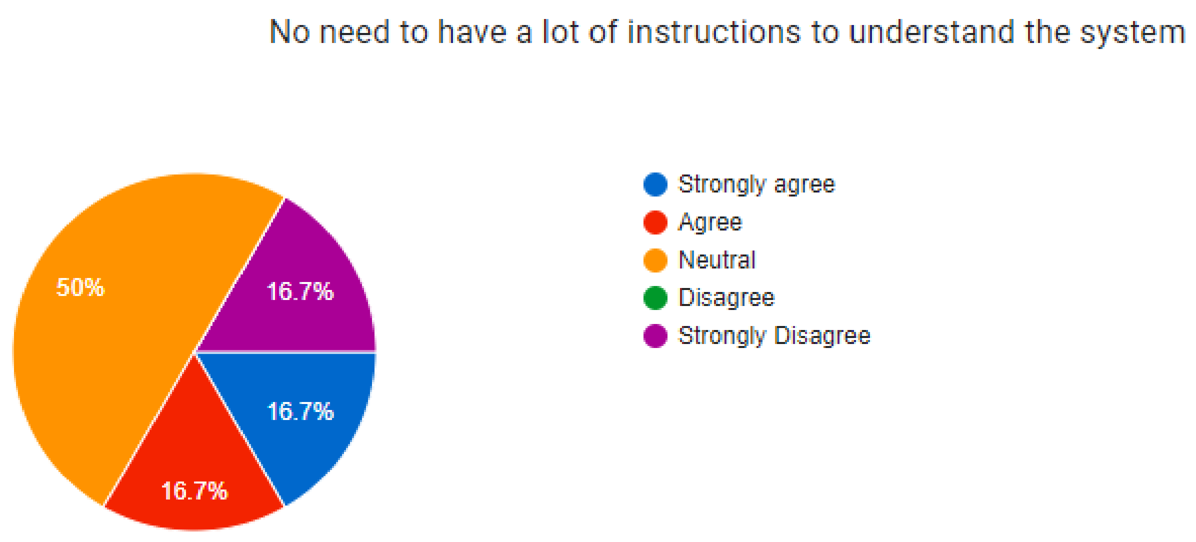


Figure 13.3.3 usability test question 3

Strongly agree: 1
Agree: 1
Neutral: 3
Disagree: 0
Strongly disagree: 1

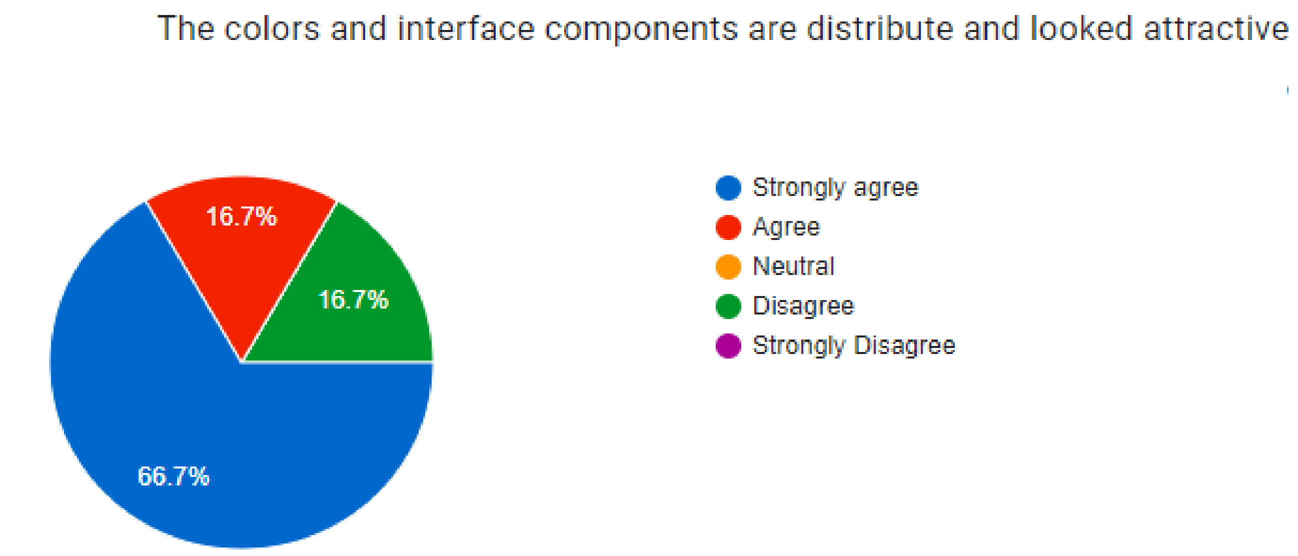


Figure 13.3.4 usability test question 4

Strongly agree: 4
Agree: 1
Neutral: 0
Disagree: 1
Strongly disagree: 0

14. Deployment of the system

Here we describe the actual deployment of the system.

1. Browser: The place where the user and other actors will use the system and communicate with it.
2. Internet: Connect between the user and other actors with web server.
3. Web Server: Used to serve pages to the user through https requests, it contains application server and database server.
4. Application Server: Where the system components are responsible for generating the pages to the user, it is located inside the web server.
5. Database Server: A server that will communicate with the database, it will be used to send or retrieve user related data to/from the system, it is located inside the web server.

6. Google web services: A set of web services provided by Google.

7. Google maps API: A web service that is used to access google maps, retrieve locations, and other services.

8. Gmail API: A web service that is used to send and receive emails, manage drafts and attachments, create labels and filters, and manage settings, we use PHPMailer library for sending emails which uses Gmail SMTP server.

9. Payment services: A set of online payment services.

10. Payment API: An online service that manages the transfer of funds from a user to the merchant of an e-commerce website, we use Stripe API in test mode to operate the order functions.

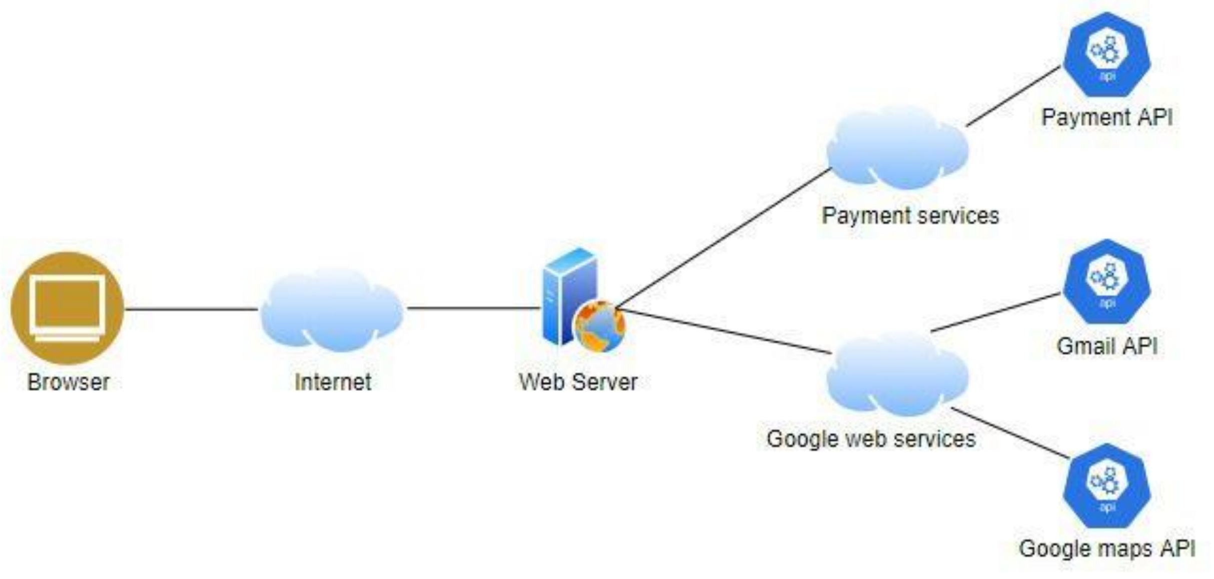


Figure 14.1 System deployment diagram

15. Limitation of the System

All of the listed requirement are implemented, but we found some problems to host our system in a server, and there is other forced limitations due to the limited domain of the system which is:

- The system usage is limited with Riyadh city, so the system should not accept the locations which outside of Riyadh.

16. Conclusion and Future Work

In conclusion, we showed the process of creating a web application that concern to apply the

concept of e-commerce used with cars accessories and spare parts, which is a predominant concept that have a lot of applications implement it, and a lot of users use it in the world.
We analyze that new system to identify the needed requirements and then we organize these requirements and put it in a form of use cases which is shows the interaction between the different actors with system.
Then we show some of the system components through analysis class diagrams and design class diagram, and then we show the interaction between these components through sequence diagrams, and we show how the related data in the system will be stored in the database schema, and also show some test cases which is conducted for some functionalities after the implementation to ensure they work properly and as it is required, and then we implement the all of document requirements in a web application.
In the future we may try to find a sponsor to support this website, and when it delivered to more users, they may give some suggestions to add some features, or modify the system functionality.

17. Reference

1- Speero
https://speero.net/
2- Afyal
https://afyal.com/
3- TM application
https://play.google.com/store/apps/details?id=app.gettm.mobileapp&hl=ar&gl=US
4- e-commerce
https://www.investopedia.com/terms/e/ecommerce.asp
5- The Entity-Control-Boundary Pattern
https://www.cs.sjsu.edu/~pearce/modules/patterns/enterprise/ecb/ecb.htm

Appendix A. Risks/ Constraints Management History

A.1 Risk/Constraints

No.	Risk	How we managed it
1	Lack of communication, causing lack of clarity and confusion.	Discuss the different solutions between the team member for these problems, after we rich to a solution

		we discuss it with the our advisor to validate that solution
2	The lack of time because of exams and assignments	We organized our time and we tried to finish the work before the deadline
3	the lack of clarity of the requirement could waste time and effort	We reviewed the requirements together and clarify the ambiguous requirements
No.	Project constraint	How we managed it
1	The project must be implemented and tested within a semester	We start from the beginning of the semester in the preparation for the project and the work and distribute the work early in which each team member organize his schedule and fix the conflicts with other courses

Table A.1 Risk/Constraint

Appendix B. Functional Requirement Updates

We didn’t change anything in functional requirements

Appendix C. Non-Functional Requirement Updates

We didn’t change anything in Non-functional requirements

Appendix D. Use Case Updates

We didn’t change anything in Use case diagram
But we change the inputs for the use case description complete purchase since the API require different inputs from what we have put earlier.
And we remove the input “status” from add item because we didn’t need it for item, we need only for the order.

Appendix E. Software Architecture Updates

We make the system architecture two tiers architecture instead of three tiers.

Appendix F. Design Updates

We didn’t change anything in Design

Appendix G. Algorithm Updates

As mentioned before we remove the status from the items.

Appendix H. Database Schema Updates

- We add the ready products table to be able to show to the seller in accept order page only the products which is not ready (in ordered status).
- We add the item price and its quantity columns in the cart table to be able to show the total price of the order in the cart page.
- We remove the location columns in user and seller tables, instead of that we put longitude and magnitude to facilitate the inputs formats from google map and store it in the database in an easy to be retrieved when the location needed.