KSU Delivery Service

Graduation Project, Part-II (SWE 496)
Software Engineering Department
CCIS, KSU

Project Advisor:
L.Nouf Almobarak

Submitted by:

Student Name	Student ID
Fay Almutairi	439200810
Lama Alsanie	439202875
Aljoury Binosseil	439201286
Alanoud Alsuwailem	439201625
Reem Aldosari	439200943

07/05/2022

ABSTRACT

King Saud University campus has a large area, and sometimes getting what is needed takes plenty of time and effort from the members, which reflects badly on the work. In addition, getting late to classes or not having the time to do the work is handled, which makes it difficult to accomplish their commitments. KSU's delivery system aims to help members by providing an application that will make life & service in the campus more efficient by delivering food, books, and anything they need from anywhere in the campus.

Table of Contents

Table of figures

Table of tables

1. Introduction

While living in a dynamic world, sometimes people find it difficult to manage simple tasks. Fortunately, consumers can now solve these tasks with a few taps on their mobile phones. Smartphones have become their tool to obtain everything they want at their doorstep because of on-demand services. Indeed, digital technology is reshaping the delivery market [1].

It has always been hard to get what Student and faculty members and university staff needs, when having a full schedule, or when the mileage is distant. King Saud University area is enormous, and getting needs has always been a challenge to the members of the university or to the students. There is a delivery application used that helps users need outside the college region, why not have a delivery application made especially for King Saud University.

KSUDS is an on-demand delivery app that will help users to order anything from anywhere in KSU campus by providing a wide range of couriers around KSU campus throughout a flexible system. It's like having a personal assistant who is ready to serve your needs and save your time when you are around campus. For couriers, KSUDS is like a side job; It allows couriers to use their free time to earn more money in an easily accessible way.

However, the main point of this project is to build an application that can provide King Saud University members and students services, where they can order what they need from the anywhere in university at any time. Other features that KSUDS provides: Users can track their order, contact the couriers and so many extra features will be discussed later. Also give a chance to gain work experience and for those who have free time to be couriers. Our targeted user: Staff, students, and faculty who will benefit from KSUDS Application.

The application is developed using the IOS platform and utilizing Flutter as the main programming language used to develop such as iOS Apps, iPad Apps, and mac Apps. In this project we will use three APIs to provide all the needs of the users: API Payment, API to send emails, API to make calls.

API Payment First and foremost, APIs enable you to significantly decrease costs and time, to comply with users' needs, a tech API payment platform should be able to account for flexibility, speed, and cost [2]. In addition, the application supports calling between users. It will solve the communication problem. We will use google maps. It has vector graphics which allow our application to use less data than Google Maps [3][4].

2. Domain Analysis

There will be applications within this section that share similar concepts to our own, and some features that will be incorporated into our application. It will help us to build our application with the most essential and unique features.

2.1 Mrsool

Mrsool is one among the biggest delivery platforms within the region. It allows people to position orders for any product from anywhere within the city since it covers all restaurants and stores, like pharmacies, restaurants, stores and far more. Users can join Mrsool and begin delivering orders. Additionally, users can order from multiple places within the same order.[5]

Figure 1 mrsool application

2.2 ToYou

ToYou app offers a growing selection of stores and restaurants, making it possible for you to own your needs delivered wherever you wish. As an extra feature, it provides users with social communication channels when shopping. while user can book a ride to anywhere. [6]

Figure 2 toYou application

2.3 Wssel

Wssel provides users a superb experience by enabling users to order from a spread of stores that include restaurants, sweets, drinks, bakeries, ice cream, coffee and even groceries, with no minimum order. Wssel allows users to trace your order go on the map and users can even contact our customer care and delivery specialists relaxed.[7]

Figure 3 wssel application

2.4 Nana

Nana is a Saudi mobile application that supports iOS and android. Nana provides you with a full and varied list of groceries and residential essentials. Users can prepare your e-cart now and schedule the delivery whenever it fits you.[8]

Figure 4 Nana application

2.5 Jahez

Jahez may be a Saudi Platform that makes a specialty of delivering restaurant orders online. It provides new features like selecting the delivery time, users can track the order, and can be notified once the driving force is near the address. Today, Jahez is understood for easy order, delivery speed, payment options, and customer service support.[9]

Figure 5 Jahez application

2.6 PIK

PIK application may be a mobile app for all merchants within the local market of Riyadh serving defined geographical location toward targets. PIK provides major brands of various commodities to mass market customers, PIK enables delivery services between customers and courier, PIK delivers within the fastest way and is out there 24/7.[10]

Figure 6 PIK application

Table (1) lists the common features of the applications mentioned above of our application "KSU Campus Delivery Service" are tracking the order, calling the courier, adding tips to the courier, users sending things between each other, receiving notification about order's status, Users can rate the order, gain points that offer discounts, view previous orders.

Features/ Applications	Mrsool	ToYou	Wssel	Nana	Jahez	PIK	KSU Campus Delivery Service
Users shall be able to track the order.	✔	✔	✔	✔	✔	✔	✔
Users shall be able to call the courier.	✔	✔	✔		✔	✔	✔
Users shall be able to add tips to the courier.				✔			✔
Users shall be able to send	✔	✔	✔				✔

things between each other.							
Users shall be able to receive notification about order's status.	✔	✔	✔	✔	✔	✔	✔
Users shall be able to rate the order.	✔	✔	✔	✔	✔	✔	✔
Users shall be able to have a variety of places to order from.	✔	✔					✔
Users shall be able to gain points that offer discounts.	✔			✔			✔
Users shall be able to view previous orders.				✔	✔		✔

Table 1 Domain Analysis

3. Risk/Constraints

The following table demonstrates the risks that could impact the project during its development lifecycle (DLC). Listed below are some risks that may arise during the development of our project and some risk management strategies to avoid them.

#	Risk	Type	Severity	Likelihood	Risk Management Strategy
1	Schedule management, when project tasks and activities will take longer to complete than estimated.	Project	High	Medium	Finish tasks and deliverables on time by putting deadlines before the submission.
2	Limited knowledge of new tools and technologies, especially in Flutter language that we never used before.	Technical	High	High	Searching for resources of the new tool/technology and taking online courses to increase our knowledge about it.
3	Link our application with API's that we need	Technical	High	High	Searching for resources of the way and taking online courses to increase our knowledge about it.
4	The status of the order should be updated continuously.	Product	Low	High	Explore other applications that care about the tracking status feature.
5	Competition with similar applications	Business	Low	High	Exploring the weakness points in similar applications and try to make it a strength in our application.
6	Keep the data of our customer safe especially credit card information	Technical	Medium	High	Searching for strong strategy to cipher the data from attacking
7	Interface suitable for both the customer and the courier	Product	Medium	High	Explore other applications that targets the same users and learn from their usability techniques

Table 2 Risk and constraints

4. Project Plan:

4.1 Project Methodology:

Using the waterfall methodology, we will create a system that is sequential and linear, following a linear approach. This model is called waterfall because it develops in a downward sequence from one phase to another.

Waterfall model is divided into different phases and the output of one phase is used as the input of the next phase, each phase must be completed before the next begins, and phases cannot overlap.

So, it will provide us with some benefits such as the start and end points for each phase are set, which makes it easy to measure our progress. As well as providing a Detailed documentation of each development stage, waterfall has 5 phases: requirement, analysis, design, implementation, and testing.

In the requirement phase we delivered the use case diagram with its description, For the analysis phase we collected the requirements through a survey to capture the needs of our users, We considered using MMVP (Model View View Model) as our system architecture in the design phase, In the implementation phase we will use Flutter to code our app, In the testing phase we used (reviews, templates, walkthroughs, and configuration management), In Conclusion waterfall is suitable for small projects, which is similar to our project [11].

4.2 Project Work Plan:

The project will be achieved by using waterfall methodology, from the 2nd of September until the 07 of May. Through completing major tasks, each member will be assigned for certain tasks and activities.

The project work plan will be shown by using Gantt Chart to illustrate all the phases, tasks within each phase and who will work on them. The below project plan figure is planned to use [Team Gannt online management software][12].

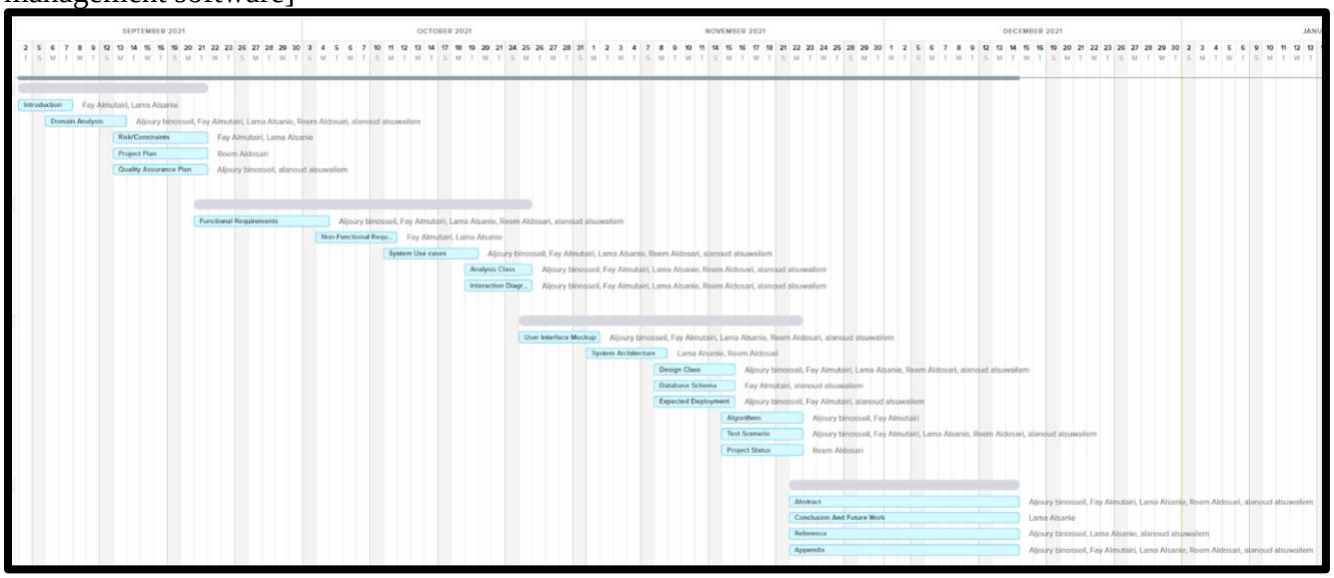


Figure 7 project plan for the first semester

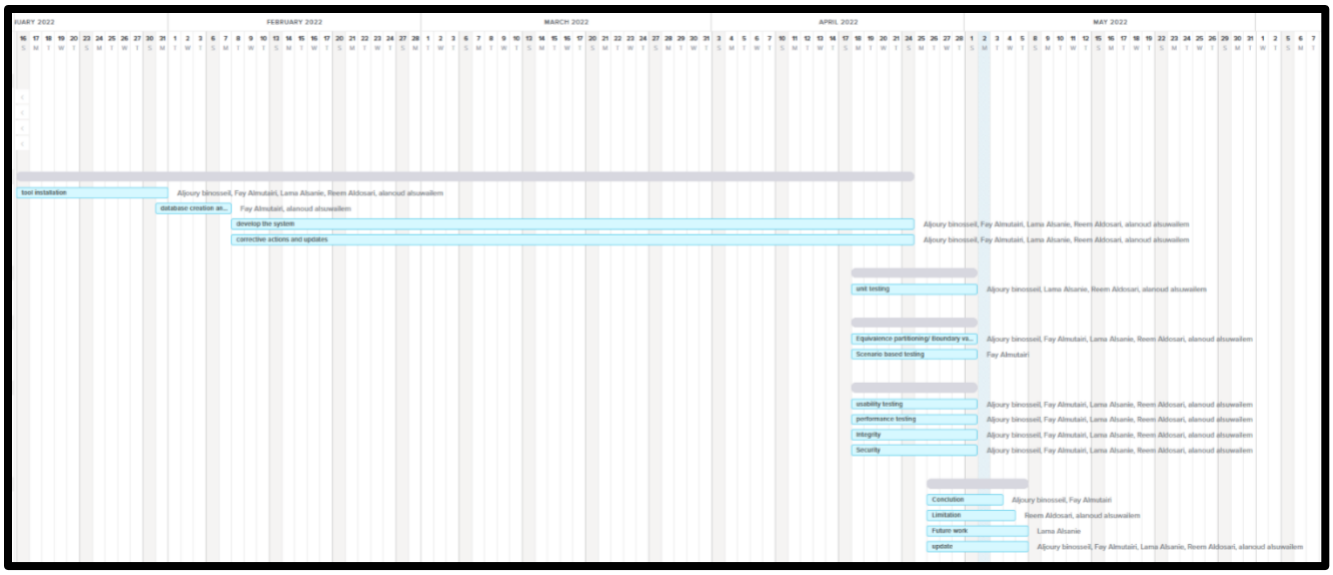


Figure 8 Project plan for the second semester

The Gantt chart in figure 7, and 8 shows the activities and tasks of each phase, it shows when, the phase begins and when it ends. And it shows the assigned tasks and responsibilities for each member.

5. Quality Assurance Plan

This section will contain a list of activities that will be performed and followed to ensure the final product is of the utmost quality. Quality assurance plans will help to monitor and control the level of quality of the project, which will further indicate the level of adherence to the targets.

5.1 Checklist:

During the phases of the Graduation Project. We will follow the checklist that was provided by the SWE department. To make sure consistency and completeness in carrying out the task and reduce failure.

5.2 Templates:

Templates will help us to save the time required to define the structure of the document and contribute to the completeness of the document. In this project we are using the Graduation Project template provided by the Graduation Project Committee.

5.3 Walkthroughs:

Walkthrough is a form of peer review. It is an informal review performed by peers after each phase of the project. A piece of work done by a team member is given to another team member to review it. Then, each team member will provide suggestions on each phase to increase the quality of the final system [See appendix 1].

5.4 Configuration management:

Configuration Management is a system engineering method for ensuring that a product's performance, functional, and physical qualities are consistent with its requirements, design, and operational information throughout its life cycle. It facilitates the orderly management of system information and system changes, which will be achieved by using GitHub and Trello.[13]

5.5 Training Team Members:

To guarantee that all the team members have enough skills to build the project, each member will join an online course in Udemy platform. Udemy is an online learning and teaching marketplace [14] Udemy will help us learn iOS development as we will watch online tutorials related to Real-Time Databases i.e., Firebase as it is needed in this project.

6. Requirements

SDLC (software development life cycle) begins with requirements gathering, which is the first and most crucial phase. Therefore, an effort was made to gather functional and non-functional requirements.

This section is dedicated to the requirements of our application, where we elicited and identified the functional and non-functional requirements. We used four methods to gather requirements (questionnaires, interview, brainstorming).

6.1 Requirement elicitation methods

In this section, we will describe the requirement elicitation methods and techniques. We used to gather requirements for the KSUDS System:

Questionnaires:

We used a questionnaire which is an efficient way to collect important requirements to obtain information suitable for our application and know the value of delivery services to gather requirements to build an efficient application, so we sent the questionnaires to the members of king Saud university, and it was filled by a diverse selection of members, so we assembled one hundred sixty-three responses to ten questions and here are the results. Our Questionnaire contains the needs that our application needs to take into consideration such as having a delivery application for the university helps with the long distance between buildings and the result showed that 96.7% agrees and 4.3% disagrees, and another result of our Questionnaire is that 49.1% of the members would like to fill their free time to work as a courier and 50.9% of the members would not prefer working as a courier. [See appendix 2]

Question	answer			
هل يعتبر وجود تطبيق توصيل داخل الحرم الجامعي حل فعال لحل ازمة تباعد المباني؟	نعم (١٥٦ صوت) ٩٦.٧٪		لا (٧صوت) ٤.٣٪	
هل تودين أن يتوفر من يقوم عنك ببعض المهام كشراء الكتب من كلية الدراسات الإسلامية او قهوة الصباح او طباعة بعض الملفات؟	نعم (١٥٦ صوت) ٩٠.١٪		لا (١٦صوت) ٩.٩٪	
ما هي الأسباب التي تدفعك إلى استخدام تطبيقات التوصيل؟	توفير الوقت والجهد (صوت ١٤٦) ٩٠.١٪	الخصومات (صوت ٦٦) ٤٠.٧٪	عدم وجود خدمة التنقل (صوت ٧٩) ٤٨.٤٪	تعتبر العملية أسهل لديك عند الطلب بخدمة التوصيل (٩٢ صوت) ٥٦.٨٪
هل يعد التنقل بين الكليات الأخرى من مشاكلك اليومية؟	نعم (١١١ صوت) ٦٨.١٪		لا (٥٢ صوت) ٣١.٩٪	
يعد الوقت عامل مهم في الحياة الاكاديمية والحاجات المتوفرة تكاد أن تكون محدودة في بعض المباني هل رغبتي بتوفر شخص يقوم عنك ببعض المهام كنقل بعض الاشياء أو جلبها لك؟	نعم (١٤٣ صوت) ٨٧.٧٪		لا (٢٠ صوت) ١٢.٣٪	
كيف تفضلين التواصل مع مندوبة التوصيل؟	عبر الرسائل النصية (١٥٧ صوت) ٩٦.٩٪		عن طريق الاتصال (٥٦ صوت) ٣٤.٦٪	
ما هي أفضل طريقة تفضلينها للدفع؟	الدفع عند الاستلام (٨٧ صوت) ٥٣.٧٪	Apple pay (١٣٧ صوت) ٨٤.٦٪	STC pay (٦٧ صوت) ٤١.٤٪	البطاقة الائتمانية (٧٣ صوت) ٤٥.١٪
كعضو في جامعة الملك سعود هل تودين أن تملئي وقت فراغك بتوصيل طلبات داخل المدينة الجامعية بمقابل مادي؟	نعم (٧٩ صوت) ٤٩.١٪		لا (٨٢ صوت) ٥٠.٩٪	
هل يعد التقييم بعد الطلب أمر مهم بالنسبة لك؟	نعم، واحرص على معرفة تقييم الطرف الاخر (١١٣صوت) ٧١.١٪	لا، لا يهمني لأنني في حال عدم الحصول على الطلب بالشكل الذي أرغب فيه سأشتكي (٤٥صوت) ٢٨.٣٪	كلاهما، ارغب في معرفة التقييم قبل الاختيار، والابلاغ عن مشكلة بعد حصوله (١صوت) ٠.٦٪	
هل تعد الخصومات عامل جذب لك لاستخدام التطبيق؟	نعم (١٥٤صوت) ٩٥.٧٪		لا (٧صوت) ٤.٣	

Table 3 Questionnaires

Brainstorming Sessions:

We conducted a brainstorming session as team members to elicit more requirements and features for the system. During the sessions, we discussed several topics and ideas. That could enhance the system, then filter them and write them down. In the session, we were able to come up with several features that can help our system have a better performance and useability.

6.2 System boundary

The following figure outlines the proposed system environment. This figure shows what interacts with the proposed system. Customer, courier are the primary stakeholders who will interact directly with the system. The API that will be called enables users to communicate with each other, Google map can locate the customer location and the location of the order, Pay Pal use for making the payment online and easy, Finally, the mail server sends a notification.

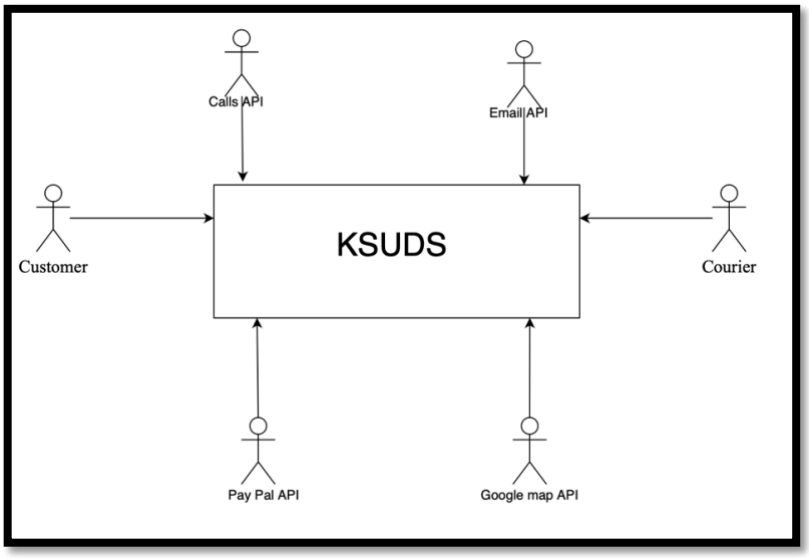


Figure 8 System boundary of KSUDS

6.3 Updated Functional Requirements

Functional Requirements of Customer:

6.3.1 The customer shall be able to register by entering username, email, phone number, and password.

6.3.2 The customer shall be able to log in by entering username and password.

6.3.3 The customer shall be able to log out from the application.

6.3.4 The customer shall be able to view his/her information profile.

6.3.5 The customer shall be able to edit his/her account information (First name, Last name, Picture, Email, Phone number).

6.3.6 The customer shall be able to view the courier profile (Name, Email, Picture, Phone number, Ratings, total orders).

6.3.7 The customer shall be able to make an order by writing a description of his/her order.

6.3.8 The customer shall be able to locate his/her location.

6.3.9 The customer shall be able to track his/her order by receiving the status of the order (Going to pick up, picked food, arrived at drop location, delivered).

6.3.10 The customer shall be able to rate the courier.

6.3.11 The customer shall be able to pay his/her order.

6.3.12 The customer shall be able to view all deliver offers received.

6.3.13 The customer shall be able to choose a suitable delivery offer.

6.3.14 The customer shall be able to contact the courier via call.

6.3.15 The customer shall be able to contact the courier via chat.

6.3.16 The customer shall be able to view his/her active order.

6.3.17 The customer shall be able to view his/her orders history.

6.3.17.1 The customer shall be able to view his/her orders history details (status, courier name, order id, total bill, order details, date, time, payment status).

6.3.18 The customer shall be able to report an issue in the order.

6.3.19 The customer shall be able to browse through all available the shops on campus.

6.3.20 The customer shall be able to viewpoints earned from ordering.

6.3.21The customer shall be able to cancel an order delivery.

Functional Requirements of courier:

6.3.22 The courier shall be able to register by entering username, email, number phone, and password.

6.3.23 The courier shall be able to log in by entering username and password.

6.3.24 The courier shall be able to log out from the application.

6.3.25The courier shall be able to view active orders.

6.3.26 The courier shall be able to view his/her account information.

6.3.27 The courier shall be able to edit his/her account information (First name, Last name, Picture, Email ,Phone number).

6.3.28 The courier shall be able to view orders requests.

6.3.29 The courier shall be able to select an order.

6.3.29.1 The courier shall be able to view the order's information.

6.3.29.2 The courier shall be able to send the offer.

6.3.30 The courier shall be able to contact the customer via call.

6.3.31 The courier shall be able to contact the customer via chat.

6.3.32 The courier shall be able to view the customer profile (Name, Email, Phone number, Ratings, Total orders).

6.3.33 The courier shall be able to report an issue in the order.

6.3.34 The courier shall be able to rate the customer.

6.3.35 The courier shall be able to view his/her orders history.

6.3.35.1 The courier shall be able to view his/her orders history details (date, time, order description, name of the customer, location, total bill, order id, payment status, chat, call).

6.3.36 The courier shall be able to view his/her rating.

6.3.37 The courier shall be able to view his/her earnings from the orders he/she deliver.

6.3.38 The courier shall be able to view the location of the place he/she will pick the order from.

6.3.39 The courier shall be able to view the location of the place he/she will deliver the order to.

6.3.40 The courier shall be able to change order status (Going for pickup, Picked Food, arrived at drop location, Delivered).

6.3.41 The courier shall be able to view the ID for the order.

6.3.42 The courier shall be able to export the bill by the chat when he/she pick-up the order.

6.3.43 The courier shall be able to access the camera to upload the bill picture.

Functional Requirements of System:

6.3.44 The system shall be able to calculate the average rating for the customer or the courier.

6.3.45 The system shall be able to calculate the total price including the delivery of the order.

6.3.46 The system shall be able to generate ID for the order.

6.3.47 The system shall be able to show a confirmation payment message to the customer after paying.

6.3.48 The system shall be able to validate the payment information before accepting it.

6.3.49 The system shall be able to notify the receiver when the sender sends a message.

6.3.50 The system shall be able to calculate points earned after each order to get a discount.

6.3.51 The system shall be able to notify the customer when order arrives.

6.3.52 The system shall be able to notify the customer/courier when the customer/courier canceled the order.

6.4 Updated Non-Functional Requirements

Reliability:

6.4.1 The system shall be backed up every 24 hours.

Usability:

6.4.2 The customer shall be able to learn how to use the system in less than 20 minutes.

Performance:

6.4.3 The system response time shall be less than 3 seconds.

Availability:

6.4.4 The system shall be available 90% 24 hour.

Integrity:

6.4.5 The system shall ensure that the customers' data can be accessed only by authorized customers.

Maintainability:

6.4.6 Installation of a new version shall leave all database contents and all personal settings unchanged.

Security:

6.4.7 The authentication in the application shall be through email and password.

Design Constraints:

6.4.8 The system shall be developed on the IOS platform.

6.4.9 The ability to adapt to different screen sizes and orientations.

6.4.10 The system shall interface with existing bank systems via an electronic transaction.

6.4.11 The system shall integrate with Maps, bank, and emails systems.

7. Project complexity

In this section we will discuss the issues and problem complexity we faced during the development of Kind Saud University delivery service , and how we created a useful solution and understood the problem domain.

7.1 Having two types of access

How to develop an application with two types of access may not be easy in one single application, since carrier and admin will have different features which makes the application huge.

7.2 Users with No Technical Background

Since the main users of the system are king Saud University members with no background on the delivery systems and how they work , the system must be designed to be self-learning and developed like most famous frequently used application , in order to allow KSU members with low background to benefit from its purpose.

7.3 Management complexity

Our project members are five and we must manage time in order to finish implementing ,testing and documenting to be done through one term.

7.4 Integration of googles map API

Google map API gives the application access to the map, since our application domain is delivering, the most important point was using Google map API, but we faced a problem since the destination of one

building to the other is a close, so we had to discover a way to allow it to use short destination. By assigning the buildings and the restaurants of King Saud University manually and give the customer the ability to choose between the names of the buildings.

7.5 Integration API of PayPal

Gaining access and performing real sensitive transactions is out of scope for this project therefore we used PayPal API to have transactions that works such as the real transaction service.

8. Updated System Use-Cases

Methodology utilized in system analysis to spot, clarify and organize system requirements. The employment case is formed from a collection of possible sequences of interactions between systems and users in a very particular environment and associated with a specific goal [15].

8.1 Use Case Diagram:

A use case diagram is a graphical depiction of a user's possible interactions with a system [16]. This section will list all use-cases that the system provides and everyone it's possible actors of KSUDS system together with their relationships with one another. To do so, clarify software was wont to model the project's diagrams.

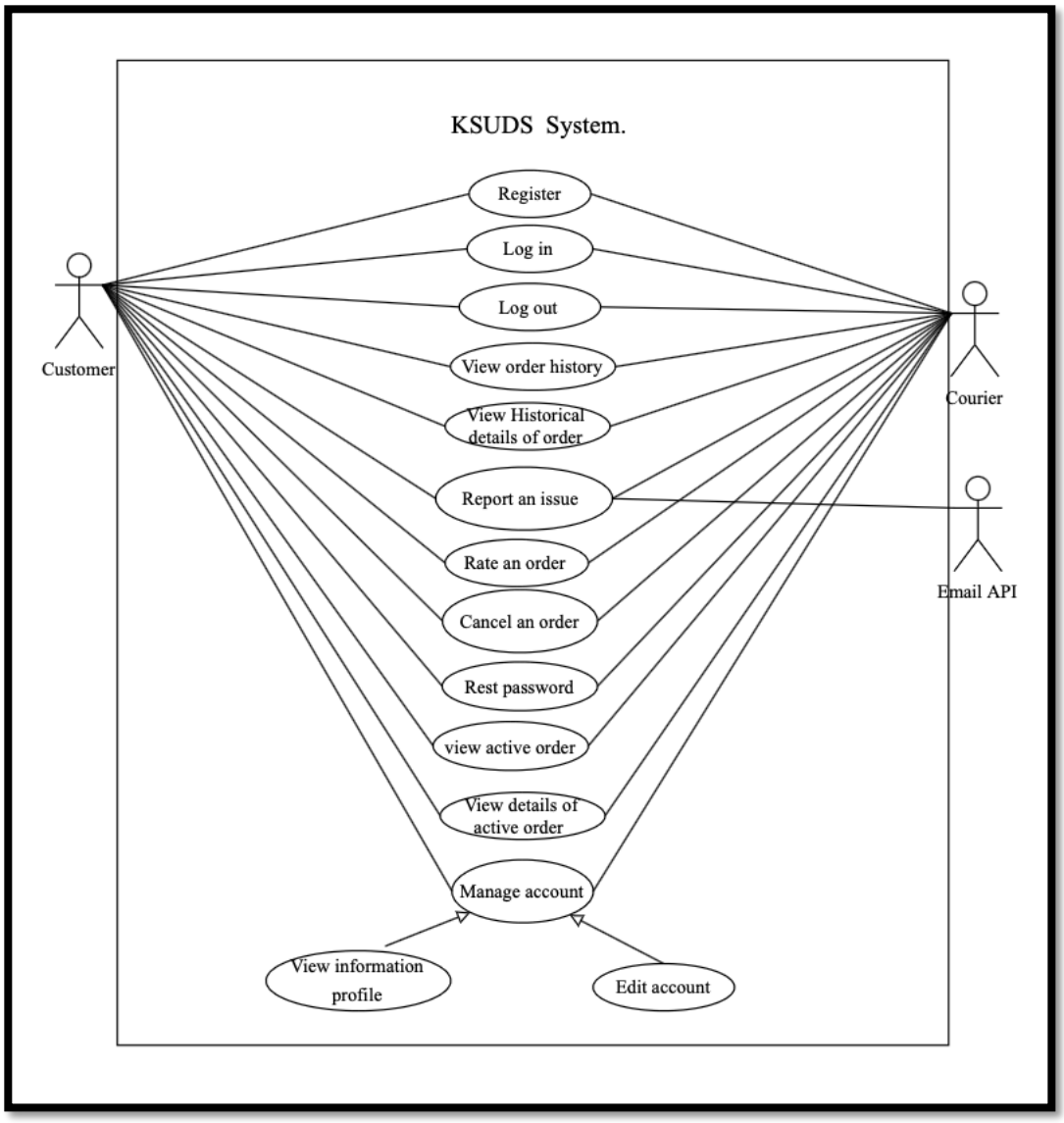


Figure 9 Use case diagram part 1

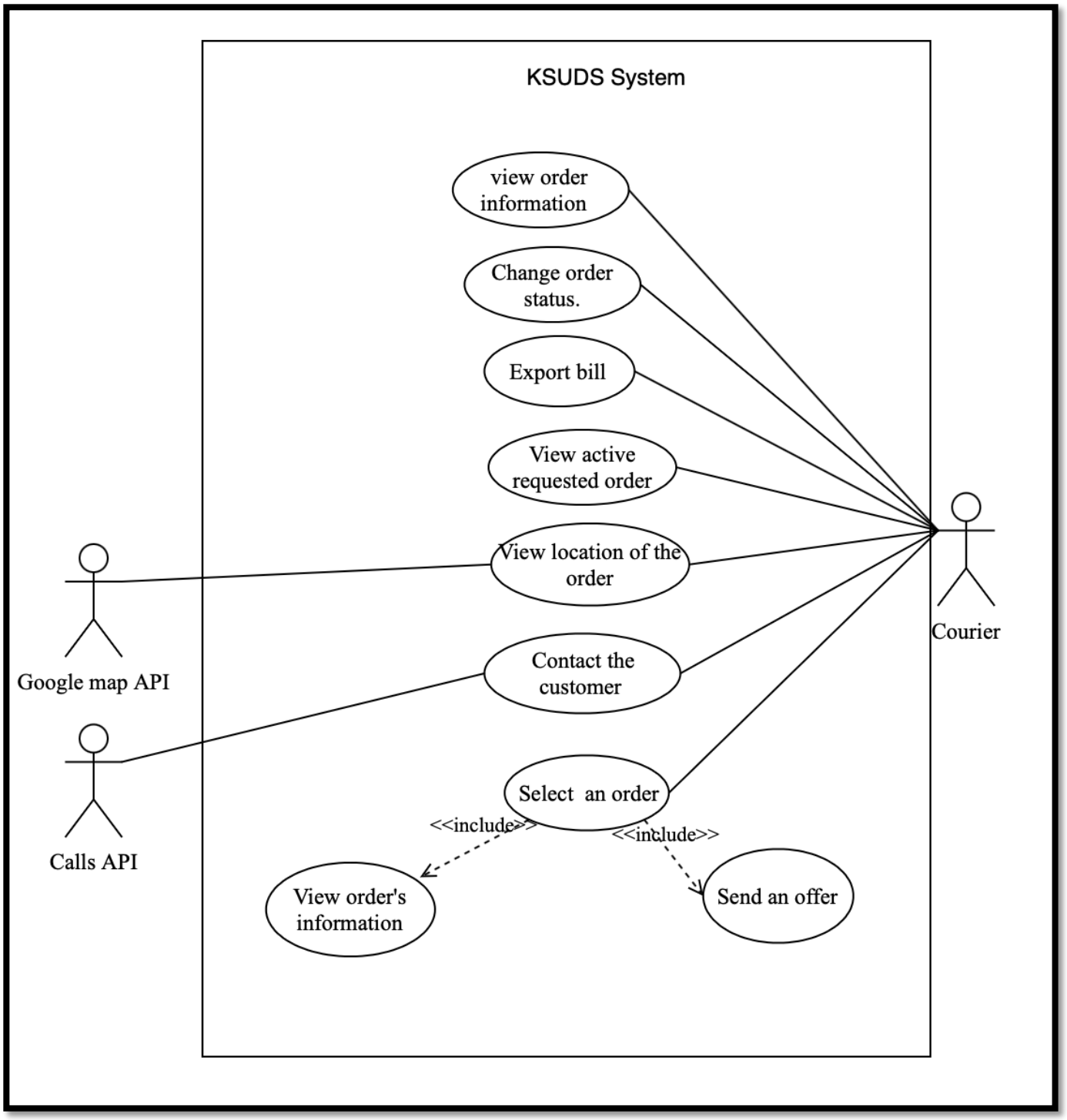


Figure 10 Use Case Diagram part 2

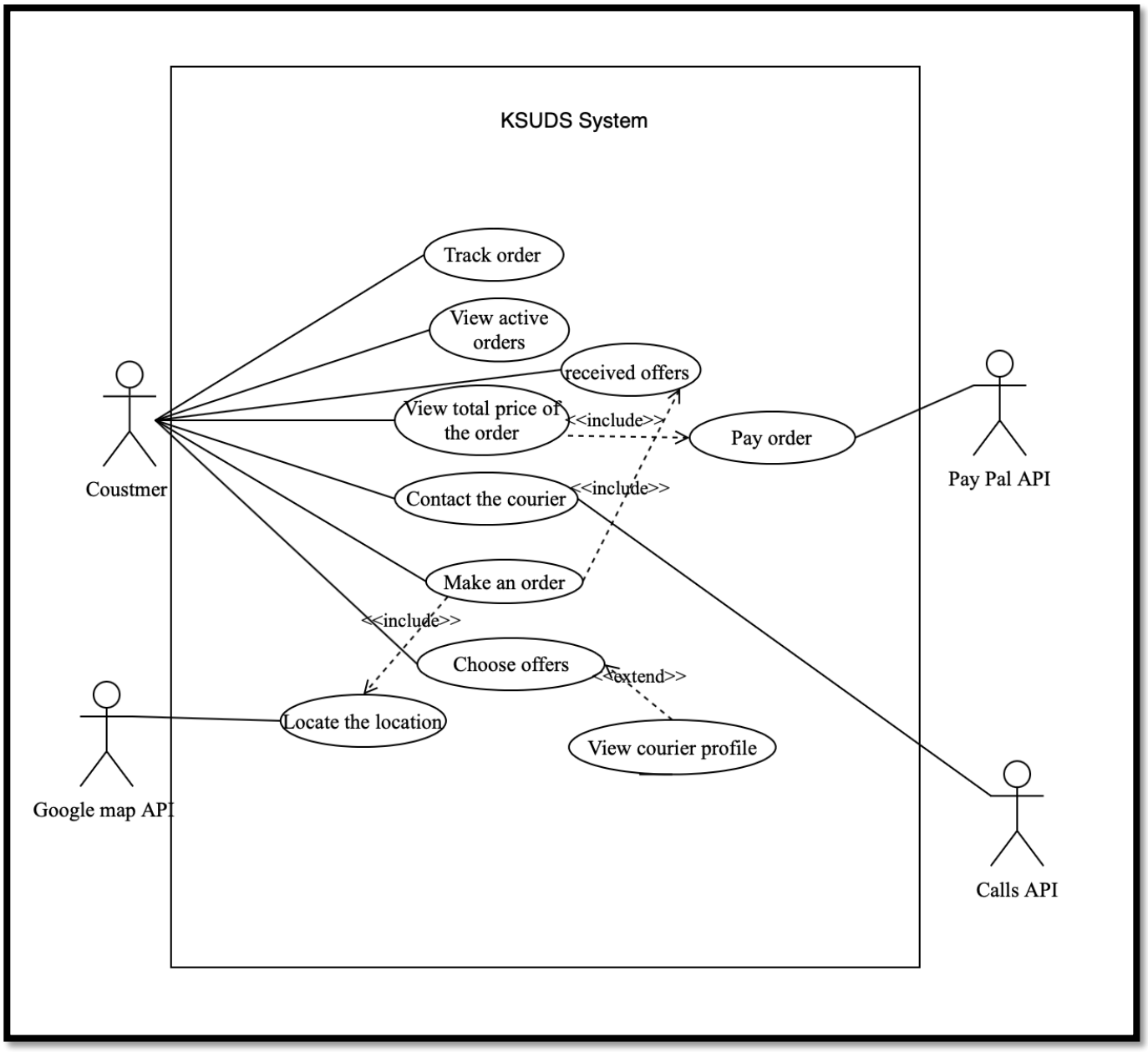


Figure 11 Use case diagram part 3

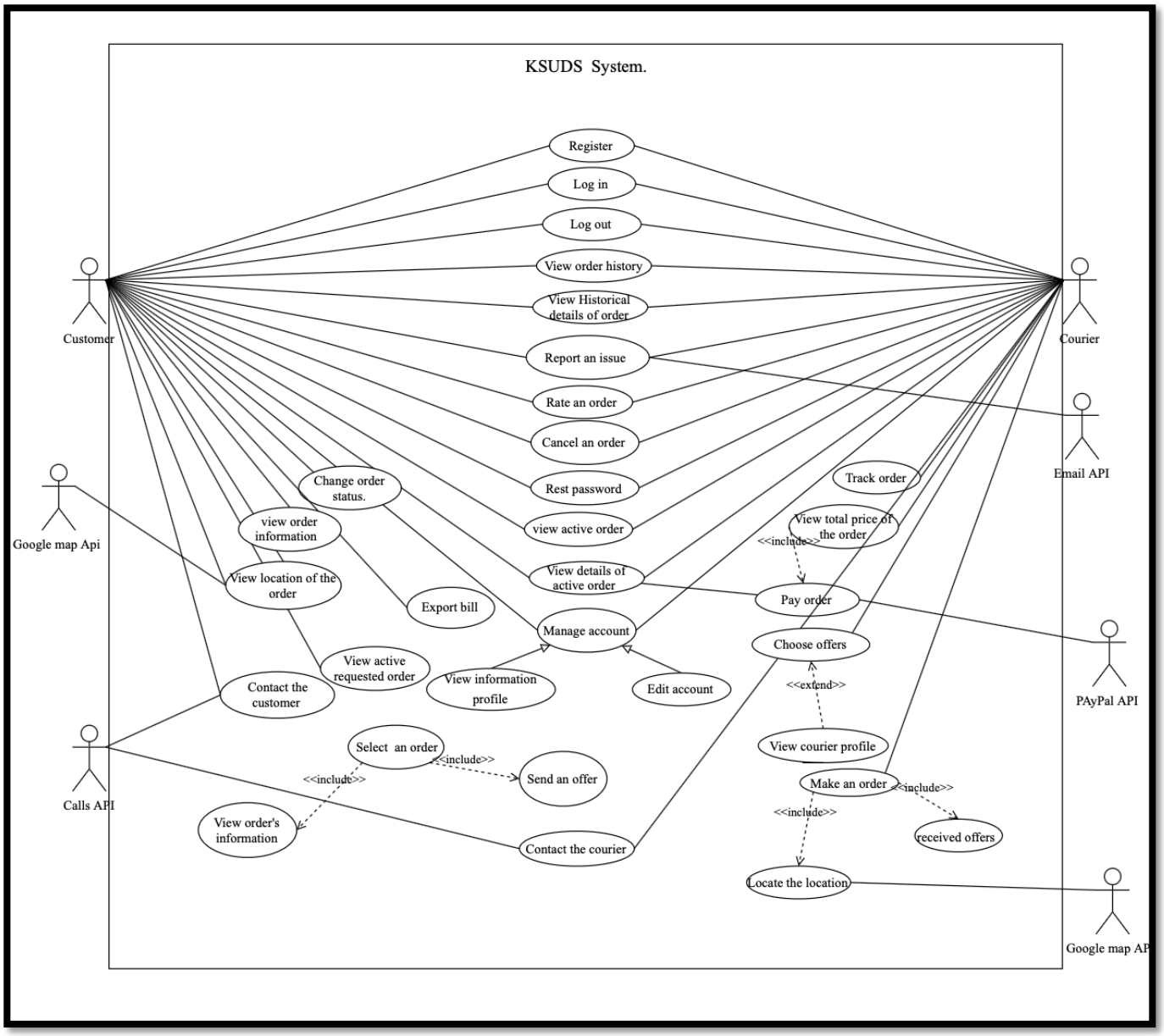


Figure 12 use case diagram part 4

8.2 Updated Use Case Description:

This section contains the use case descriptions for the critical use cases of our system.

8.2.1 Use Case 1

Use Case Description		
System: KSU Delivery Service		
Use Case name: Make an order		
Primary actor: Customer.	**Secondary actor(s):** None.	
Description: This use case describes how a customer makes a new order.		
Relationships ▪ **Includes:** Locate the location. ▪ **Extends:** None.		
Pre-conditions: Customer logged-in successfully.		
Steps:		
Primary Actor (Customer)	**System**	**Secondary Actor(s) (if applicable)**
Basic Flow: 1. This use case begins when the customer selects the restaurant and moves to the" type order" page.	2. System directs the customer to the "type order" page.	
3. The customer types descriptions of the order.		
4.Customer selects the "locate" option.	5.System directs the customer to the "Locates location" page.	
6. Customer confirms the location.		

8. Customer confirms the order.	7.System redirect customer to the "type order" page. 9. System displays all offers for order.	
Alternative and exceptional flows: **1. Choosing gift option:** In step 8: 8.1 After choosing time, customer can select option "Gift" 8.2 Steps complete. **2. Location outside campus:** In step 5: 5.1 Customer opens map and points the location outside the campus. 5.2 System display error message "location invalid". 5.3 Step 4 is resumed. **3. Customer quits:** In step 2: 2.1 Customer clicks "cancel" 2.2 The use case ends with failure conditions.		
Post-conditions: **Successful conditions**: The System successfully adds a new order on the system. **Failure Condition:** The System fails to add a new order.		

Table 4 Description for "Make an order" use case

8.2.2 Use Case 2

Use Case Description		
System: KSU Delivery Service		
Use Case name: Pay order		
Primary actor: Customer	**Secondary actor(s):** Payment system.	
Description: This use case describes how the customer pays for an order.		
Relationships ▪ **Includes:** None. ▪ **Extends:** None.		
Pre-conditions: Customer logged-in successfully, make his/her order and view exported invoice successfully.		
Steps:		
Primary Actor (Customer)	**System**	**Secondary Actor(s) (if applicable)**
Basic Flow: 1. This use case begins when the customer selects the "Pay now" option.	2. System directs the customer to the "payment method page" (Pay Pal). 3. System accepts the payment method selected. 4. The System displays a confirmation message.	

Alternative and exceptional flows: **Select another payment method** In step 3 the customer selects Pay Pal as payment method: 3.1 The System displays page to pay. 3.2 Customer pays with Pay Pal. 3.3 Step 4 is resumed.
Post-conditions: **Successful conditions:** The System successfully accepts payment method. **Failure Condition:** Payment method is declined.

Table 5 Description for "Pay order" use case

8.2.3 Use Case 3

Use Case Description		
System: KSU Delivery Service		
Use Case name: View history orders.		
Primary actor: Courier.	**Secondary actor(s):** None.	
Description: This use case describes how the courier can view his/her history delivery orders.		
Relationships ▪ **Includes:** None. ▪ **Extends:** None.		
Pre-conditions: Courier logged-in successfully.		
Steps:		
Primary Actor (Courier)	**System**	**Secondary Actor(s) (if applicable)**
Basic Flow: 1. This use case begins when the courier selects the "history order" tap. 3. The courier selects the order he/she wants.	2. The system displays all past orders. 4.The system displays the details of the desire order (order id, name of the customer, location, payment status, distance, duration, rate).	

Alternative and exceptional flows: None.
Post-conditions: **Successful conditions:** The courier views the order history successfully. **Failure Condition:** The courier do not view any order.

Table 6 Description for "View history order" use case

8.2.4 Use Case 4

Use Case Description		
System: KSU Campus Delivery Service		
Use Case name: View active requested order.		
Primary actor: Courier	**Secondary actor(s):** None.	
Description: This use case describes how the courier can view active requested orders.		
Relationships ▪ **Includes:** None. ▪ **Extends:** None.		
Pre-conditions: Courier logged-in successfully, Courier accepted the order successfully.		
Steps:		
Primary Actor (Courier)	**System**	**Secondary Actor(s) (if applicable)**
Basic Flow: 1. This use case begins when the customer selects the "Active orders" tap. 3. The courier selects the order he/she wants to view.	2. The system displays all the active orders as chats. 4. The system displays the chats with the customer including all the	

	information of the order (status, location, order id, location, name of the palace, and rating).	
Alternative and exceptional flows: None.		
Post-conditions: **Successful conditions:** The courier view actively requested orders successfully. **Failure Condition:** The courier does not view any order.		

Table 7 Description for "View active requested order" use case

8.2.5 Use Case 5

Use Case Description		
System: KSU Delivery Service		
Use Case name: Report an issue.		
Primary actor: Customer	**Secondary actor(s):** Email API.	
Description: This use case describes how the customer reports an issue faced.		
Relationships · **Includes:** None. · **Extends:** None.		
Pre-conditions: Customer logged-in successfully, Customer type his/her order and submit it successfully.		
Steps:		
Primary Actor (Customer)	**System**	**Secondary Actor(s) (if applicable)**

Basic Flow: 1.This use case begins when the customer enters the “chat” page. 3.Customer selects the “help icon”. 5.Customer fills the form with the following information (name of issue, and description). 6.Customer submits the form.	2. System displays the chat. 4.System displays the “report issue” form. 8. System displays a message that the issue has been submitted successfully.	7. system sends a confirmation email.
Alternative and exceptional flows: **Missing field:** In step 7: 7.1 The customer misses a required field. 7.2 The system displays an error message indicating the required field 7.3 Step 7 is resumed. **Not receiving email** in step 8: 8.1 The customer doesn't receive a confirmation email. 8.2. System displays an error message that the issue has not been submitted. 8.3 step8 is resumed.		

Post-conditions: **Successful condition:** The system successfully submits the report issue. **Failure condition:** The report issue will not be submitted for the customer successfully.

Table 8 Description for "Report an issue" use case

8.2.6 Use Case 6

Use Case Description		
System: KSU Campus Delivery Service.		
Use Case name: Rating an order.		
Primary actor: Customer.	**Secondary actor(s):** None.	
Description: This use case describes that the customer sends his/her rating service after the order.		
Relationships · **Includes:** None. · **Extends:** None.		
Pre-conditions: Customer logged-in successfully, type his/her order and submit it successfully.		
Steps:		
Primary Actor (Customer)	**System**	**Secondary Actor(s) (if applicable)**

Basic Flow: 1.This use case begins after the customer has made a payment. 3. Customer reviews the service with five-star rating system. 4. Customer submit the rate form.	2. System displays a screen. 5. The system displays "feedback submitted successfully".	
Alternative and exceptional flows: **Missing field:** In step 3: 3.1 The customer leaves the five-star rating empty. 3.2 The system displays an error message indicating the required field. 3.3 Step 3 is resumed. **service provider quits:** In step 3: The customer closes the application; the use case ends with failure condition.		
Post-conditions: **Successful condition:** The system successfully submits the feedback. **Failure condition:** The feedback will be considered a zero out of five rating.		

Table 9 Description for "Rating an order" use case

8.2.7 Use Case 7

Use Case Description
System: KSU Delivery Service
Use Case name: Choose an offer.

Primary actor: Customer	**Secondary actor(s):** None.
Description: This use case describes how the customer chooses an offer from the couriers offers.	
Relationships · **Includes:** None. · **Extends:** View courier profile Contact the courier	
Pre-conditions: Customer logged-in successfully, type his/her order and submit it successfully.	

Steps:		
Primary Actor (Customer)	**System**	**Secondary Actor(s) (if applicable)**
basic flow:		
	1.This use case begins when the system displays the “offers” page (courier name, price and rate).	
2. the customer selects one of the listed couriers offers containing (courier name, price and rate).		
	3.System displays a pop-up verification screen containing a summary of the courier's information (courier name, price, rate, phone number).	
4.Customer select “agree”.		
	5.System displays the chat page of the courier chosen.	

Alternative and exceptional flows: **Service provider quit:** In step 4: The customer selects ‘cancel;’ the use case ends with failure condition.
Post-conditions: **Successful condition:** The customer successfully selects an offer. **Failure condition:** The customer will not be able to select an offer.

Table 10 Description for “Choose an offer” use case

8.2.8 Use Case 8

Use Case Description		
System: KSU Campus Delivery Service		
Use Case name: Select an order.		
Primary actor: Courier.	**Secondary actor(s):** Customer.	
Description: This use case describes how the courier picks order requests.		
Relationships · **Includes:** Send an offer, View order's information. · **Extends:** None.		
Pre-conditions: Courier logged in successfully.		
Steps:		
Primary Actor (Courier)	**System**	**Secondary Actor(s) (if applicable)**
Basic Flow:	1.This use case begins when the System displays all the requested	

	orders containing (order id, location, duration, if there any comments from the customer).	
2. Courier selects the desired order request.		
	3.System displays a pop-up enter the price.	
4. Courier selects “send an offer”.		
	5.The system displays the “chat” page after customer accepted.	
Alternative and exceptional flows: **Customer select unwanted selection:** In step 2: 2.1 The courier selects an unwanted order request by accident. 2.2 The courier Cancels the order. 2.3 Step 2 is resumed.		
Post-conditions: **Successful condition:** The system successfully Picked an order request. **Failure condition:** The system will not Successfully pick order request.		

Table 11 Description for “Select an order” use case

8.2.9 Use Case 9

Use Case Description	
System: KSU Delivery Service	
Use Case name: Register	
Primary actor: Customer	**Secondary actor(s):** None.
Description: This use case describes how the Customer registers.	

Relationships

- **Includes:** None.
- **Extends:** None.

Pre-conditions: None.

Steps:

Primary Actor (Customer)	System	Secondary Actor(s) (if applicable)
Basic Flow: 1. This use case begins when the Customer selects the "register" option. 3. Customer fills his/her information which includes (name, email, phone number, password). 4. Customer clicks the "register" button and submits	2. System directs the Customer to the "register" page. **5**. System validate the customer inputs including (email, phone, password) **6.** System displays a successful message	

Alternative and exceptional flows:

1.Missing fields:

If in step 3:

1.1 The Customer misses a required field in 'register Customer.

1.2 The system displays an error message indicating the required fields.

1.3 Step 3 'register Customer' flow is resumed.

2.Customer quits: If in any step before step 4 flow for register: 2.1The Customer selects ‘return button’, the use case ends with a failure condition. **3.Wrong format:** If in step 3 the customer enters the wrong format field: 3.1 Error message will be displayed. 3.2 The customer resumes step 3 with the correct format.
Post-conditions: **Successful condition:** The system successfully registered a customer. **Failure condition:** The system will not Successfully register a customer.

Table 12 Description for “Register” use case

8.2.10 Use Case 10

Use Case Description		
System: KSU Campus Delivery Service		
Use Case name: Manage account.		
Primary actor: Customer.	**Secondary actor(s):** None.	
Description: This use case describes that the Customer can manage his/her information by viewing and updating his/her information from the system.		
Relationships · **Includes:** None. · **Extends:** None.		
Pre-conditions: Customer logged-in successfully.		
Steps:		
Primary Actor (customer)	**System**	**Secondary Actor(s) (if applicable)**

Basic Flow: **1.** This use case begins when the Customer selects “profile”. **3.** the Customer view his/her information which includes (Name, Email address, Phone number, reward points, total order, picture). **5.** User specify the function he/she would like to perform · If the Customer selects **Save** the Update information sub flow 1 is executed. **Sub Flows:** **1. Update Customer information** **1.1.** The customer fills his/her information that he/she wants to update including (Name, Phone number). **1.3.** The Customer selects the “Update” option.	**2.** System directs the Customer to the” profile” page. **4.** the user can select the filed (name, phone number) he/she wants to change. **1.2** System validate customer input including (Name, Phone number). **1.4.** System displays a successful message.	

Alternative and exceptional flows:

1.Missing fields:

In step 2:

1.1. The Customer misses a required field in ‘update Customer information’.

1.2. The system displays an error message indicating the required field.

1.3. Step 2 ‘update Customer information’ sub flow is resumed.

2. Customer quits:

In any step before step 3 sub flow for update and step 3 for delete Customer information:

2.1. The Customer selects ‘quits, the use case ends with failure condition.

Post-conditions:

Successful condition:

· If View Customer information flow-performed: The system successfully viewed User information.

·If Update Customer information sub-flow performed: The system successfully updated User information.

Failure condition:

·If view Customer information flow: Customer information is not viewed.

· If Update Customer information sub-flow performed: The Customer didn’t update its information.

Table 13 Description for “Manage account” use case

9.Interaction Diagram

Interaction diagram accustomed show the system, the actor, and the way they impart with one another. This section contains sequence diagrams for the critical use cases. Sequence diagram is an interaction diagram accustomed show a sequence of messages exchanged between the actors and also the objects to perform a particular task.[17] [18]

9.1 Make an order Sequence

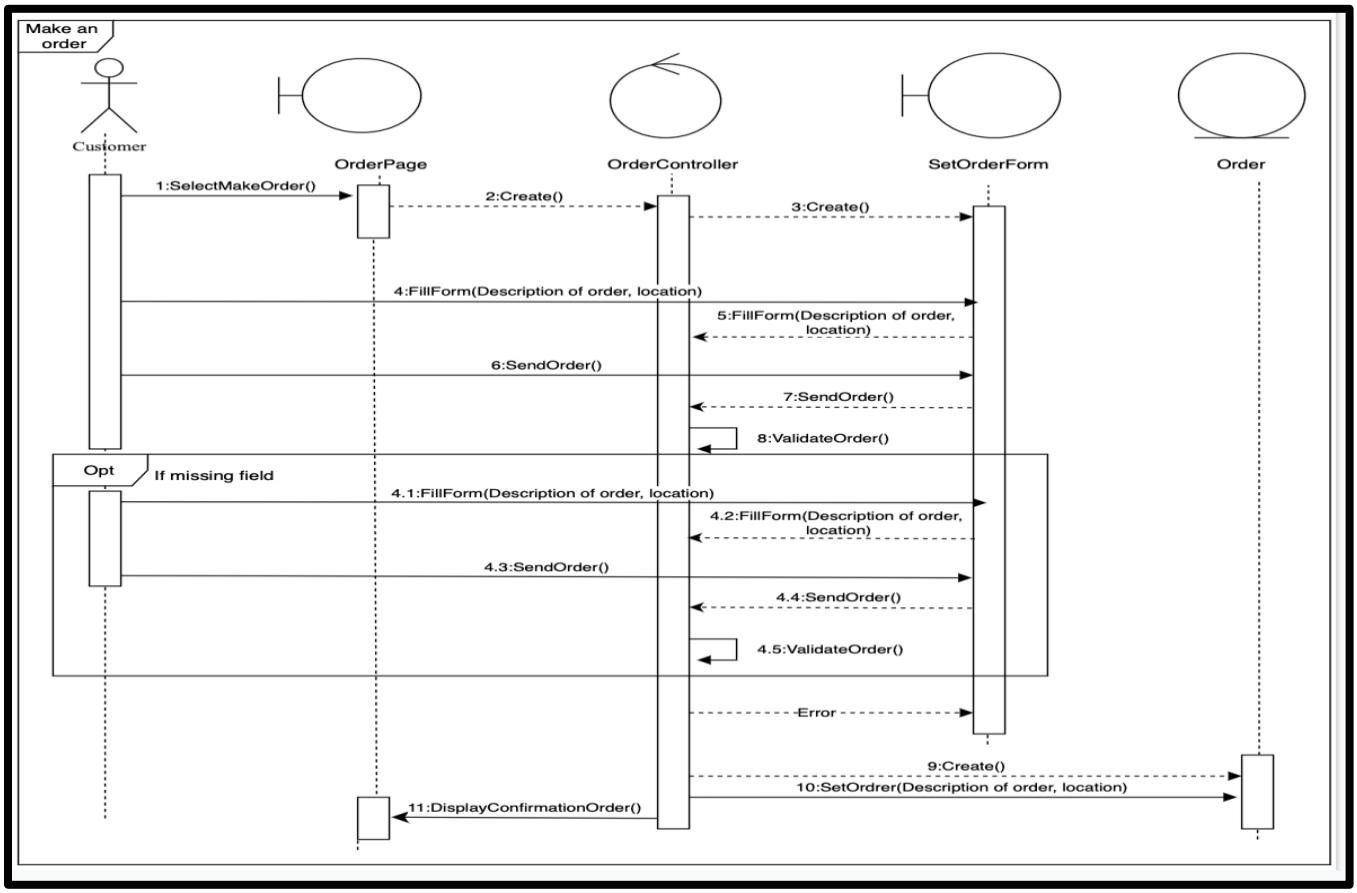


Figure 13 Make an order Sequence diagram

9.2 Pay an order Sequence

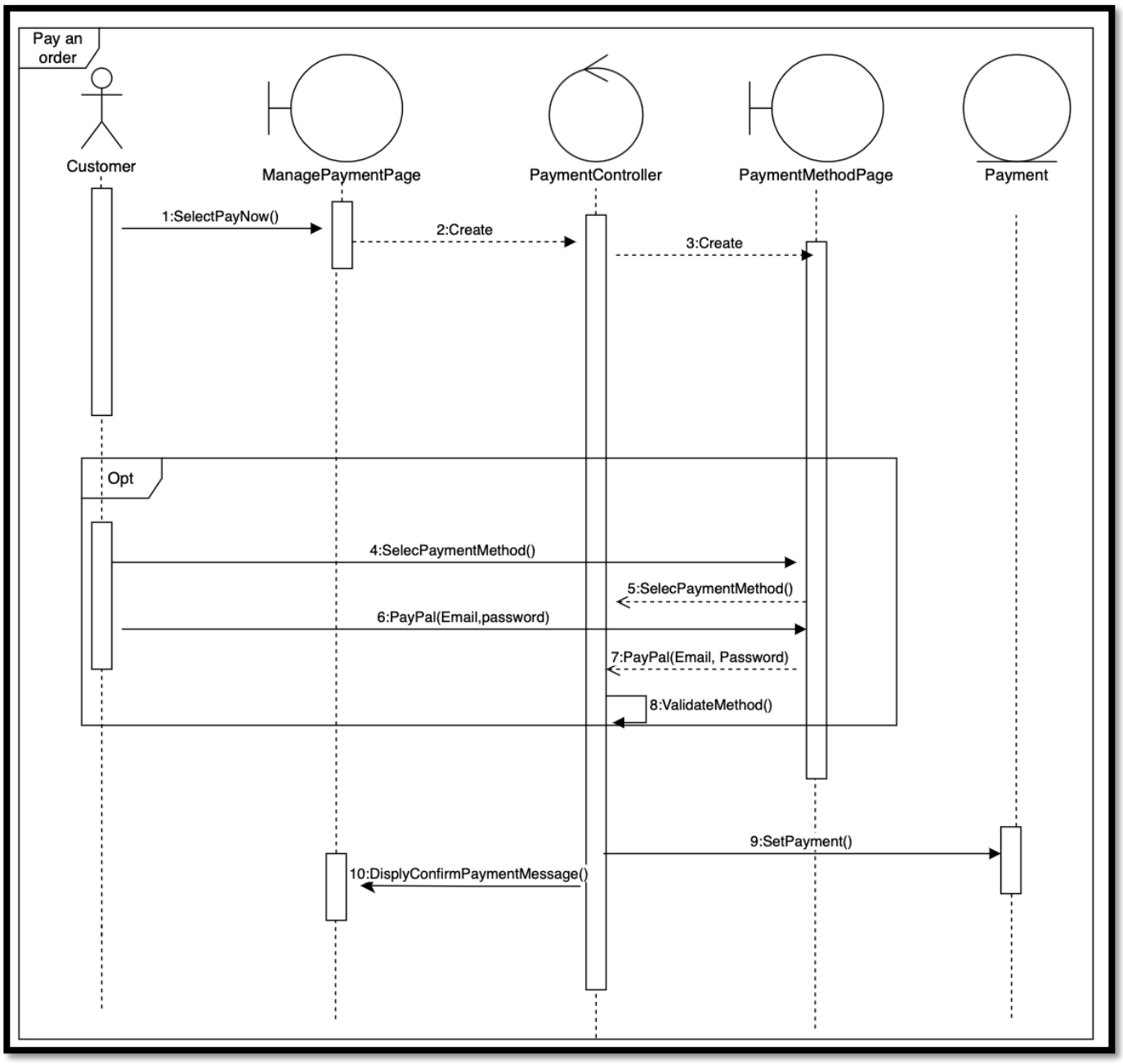


Figure 14 Pay an order Sequence diagram

9.3 View active requested order Sequence

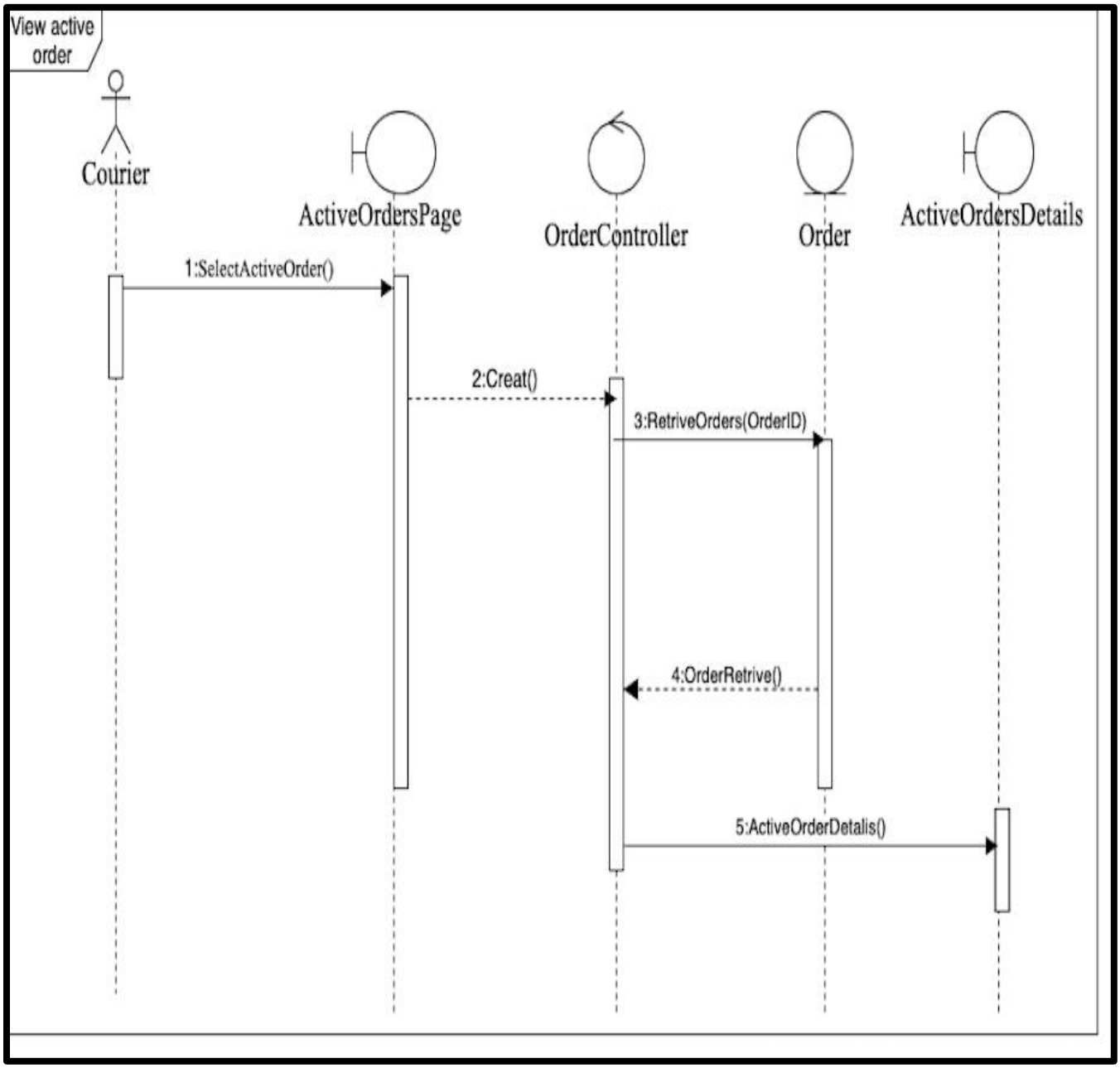


Figure 15 View active order Sequence diagram

9.4 View history orders Sequence

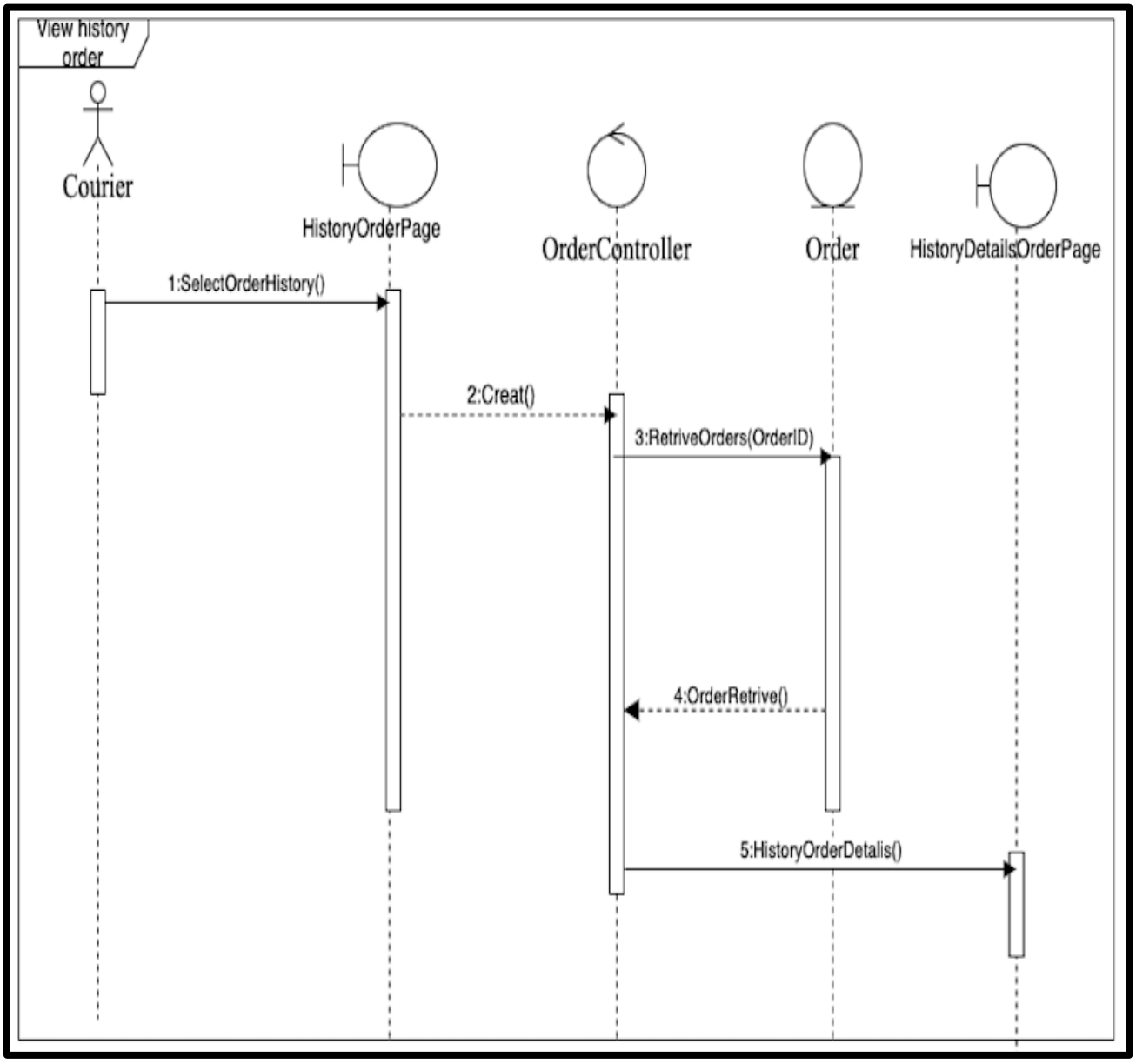


Figure 16 View history orders Sequence diagram

9.5 Report an issue Sequence

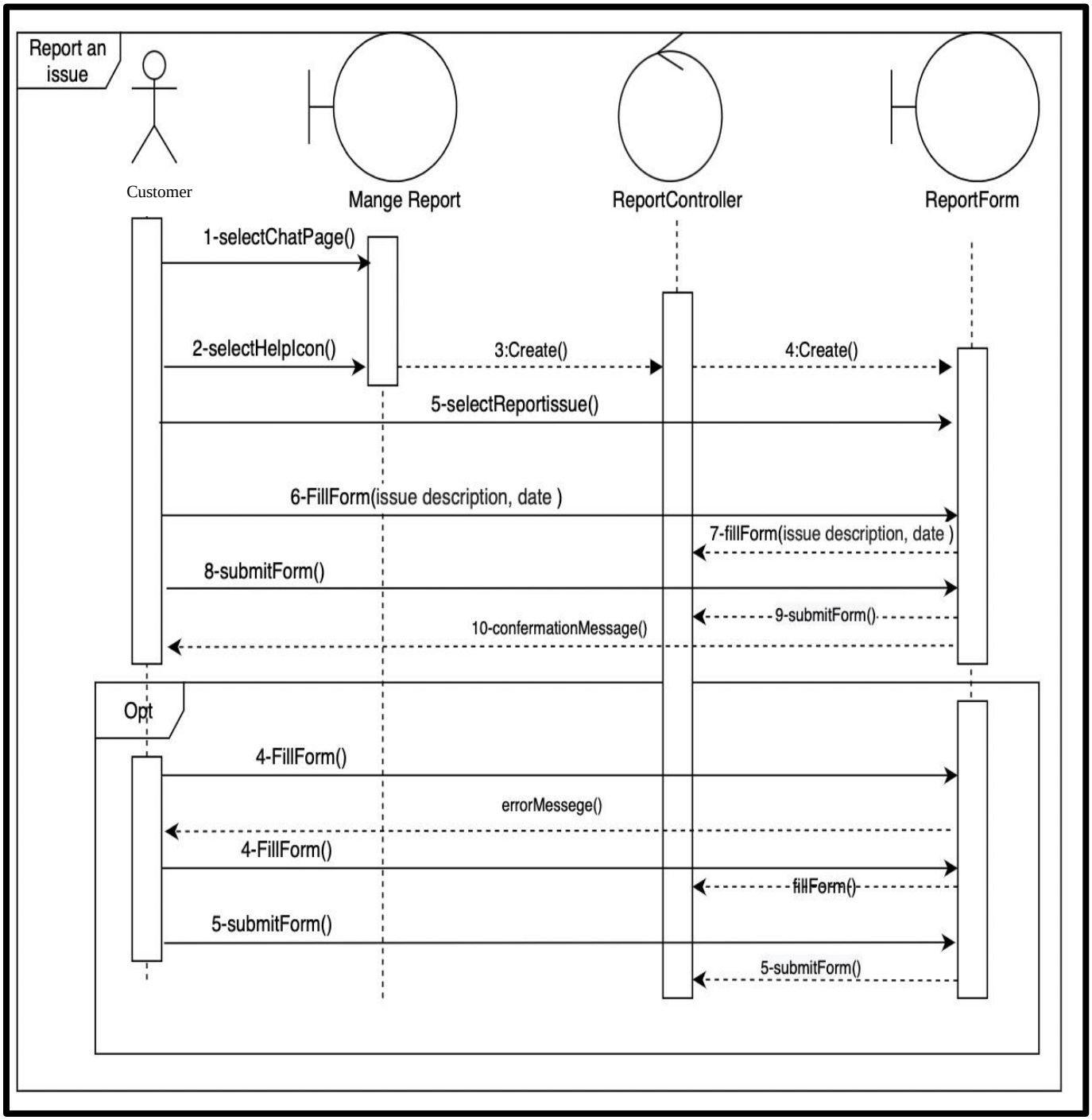


Figure 17 Report an issue Sequence diagram

9.6 Rate an order Sequence

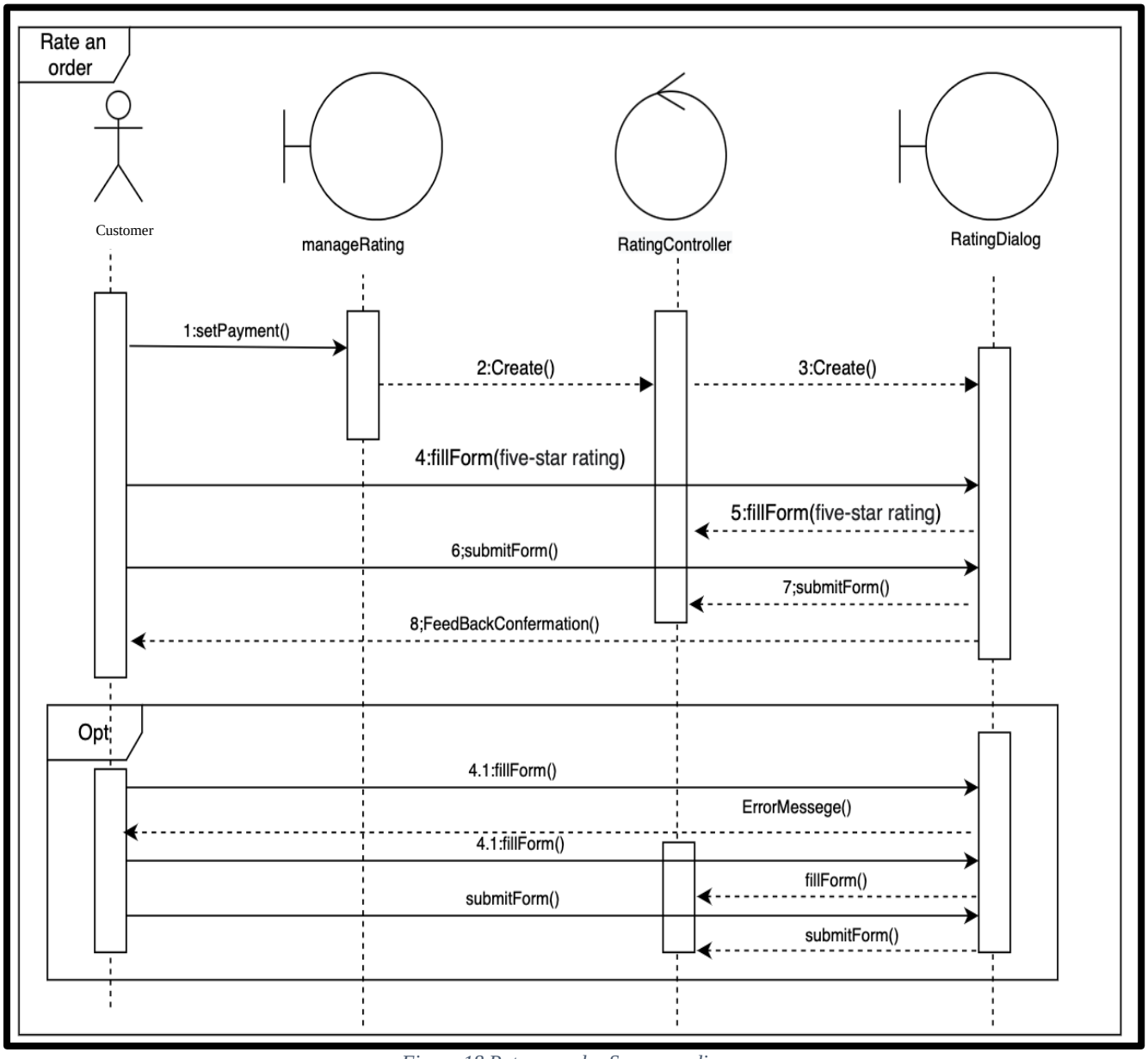


Figure 18 Rate an order Sequence diagram

9.7 Choose an offer Sequence

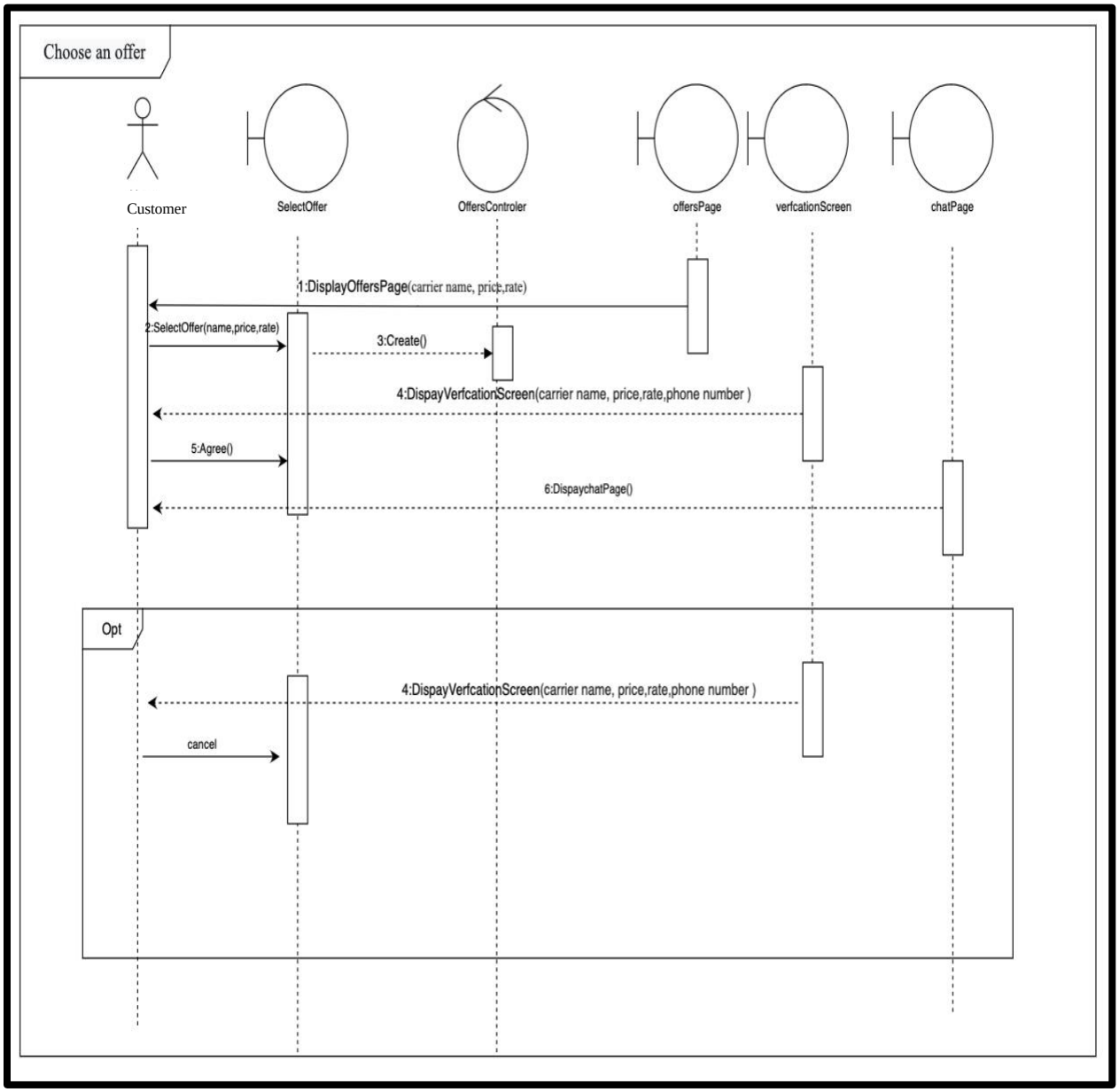


Figure 19 Choose an offer Sequence diagram

9.8 Select an order Sequence

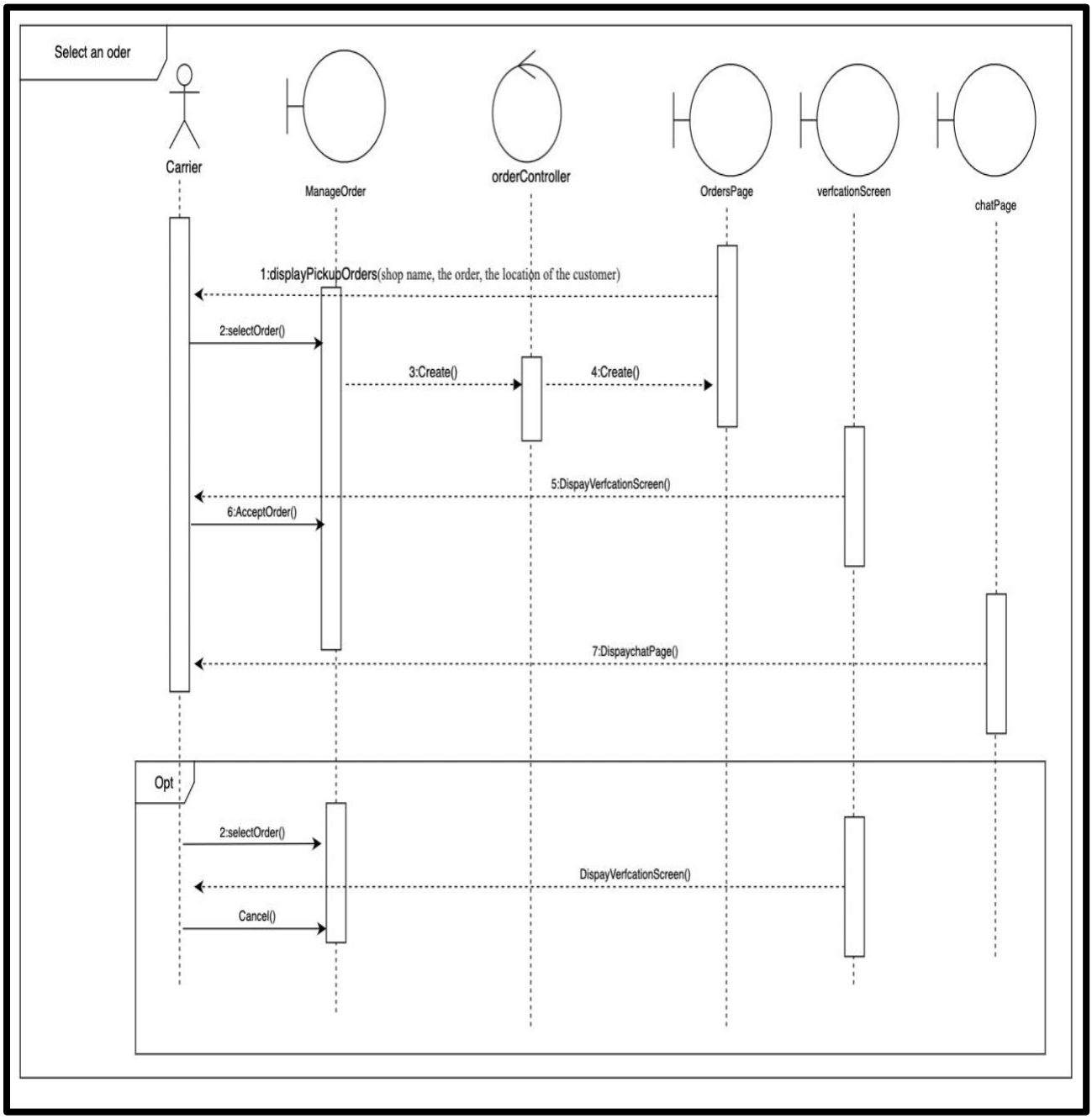


Figure 20 Select an order Sequence diagram

9.9 Register Sequence

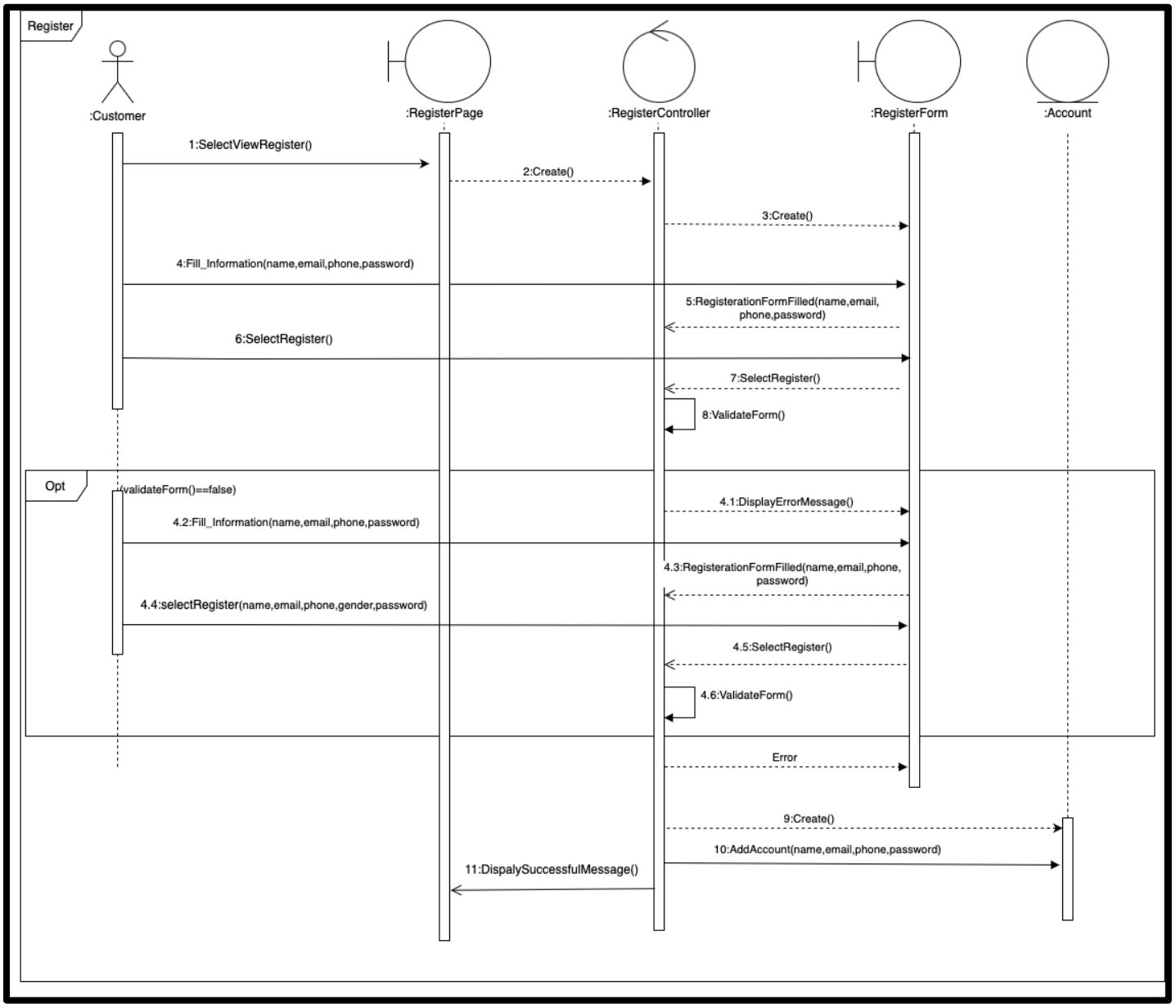


Figure 21 Register Sequence diagram

9.10 Manage account Sequence

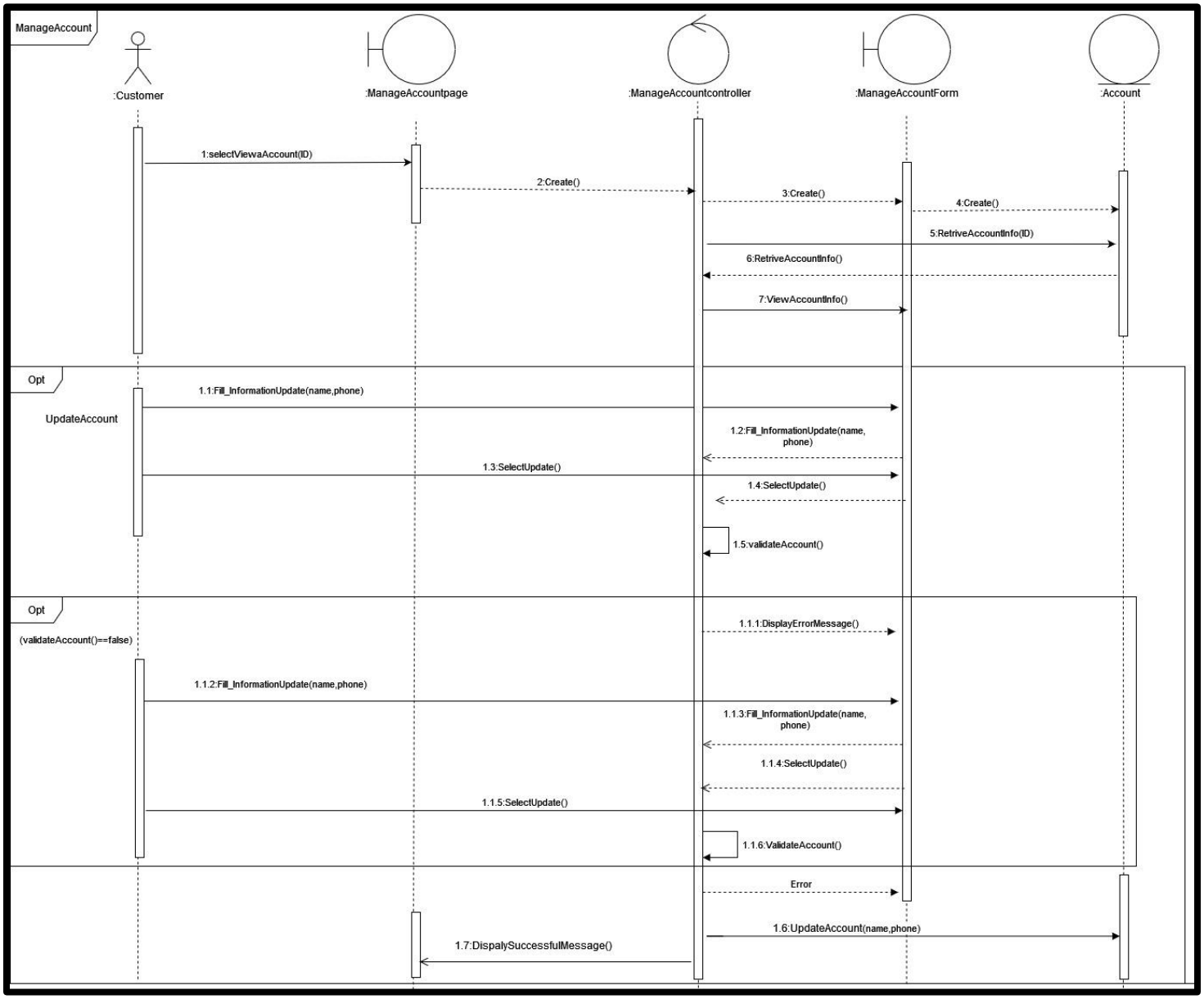


Figure 22 Manage account Sequence diagram

10.Analysis Class

VOPC is a class diagram that contains all classes that are used in a sequence or collaboration diagram, or it will draw a diagram that contains all classes that participate in a collaboration or that implement a use case. [19] This section presents a VOPC diagram of important use cases.

10.1 Make an order VOPC

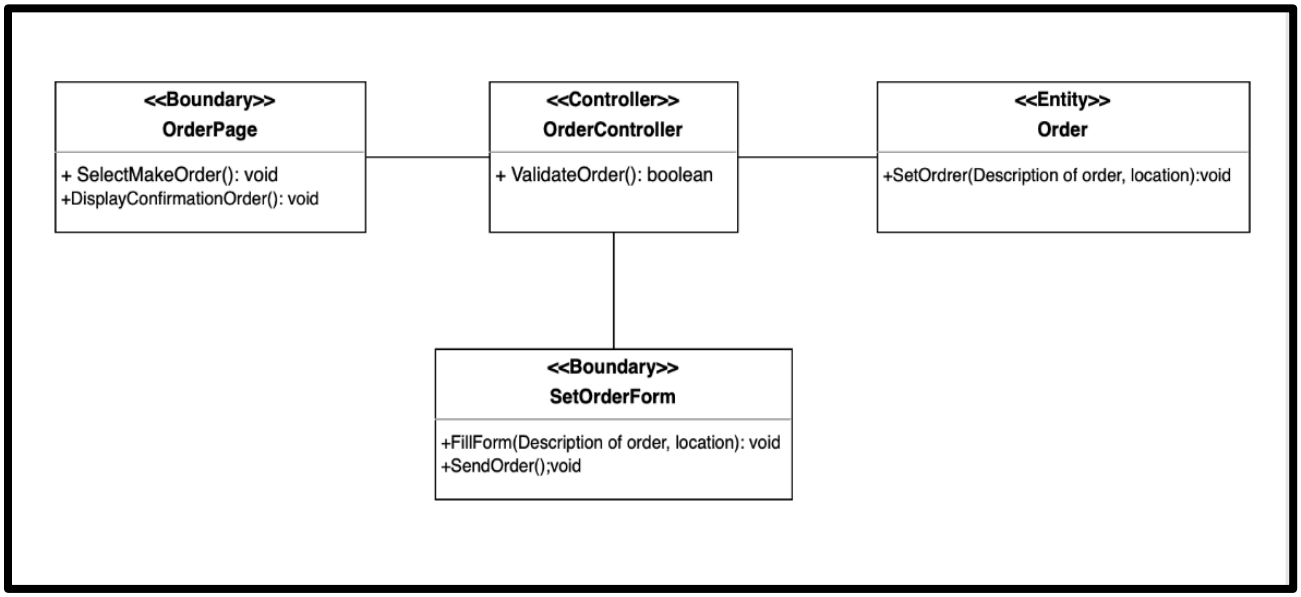


Figure 23 Make an order VOPC

10.2 Pay an order VOPC

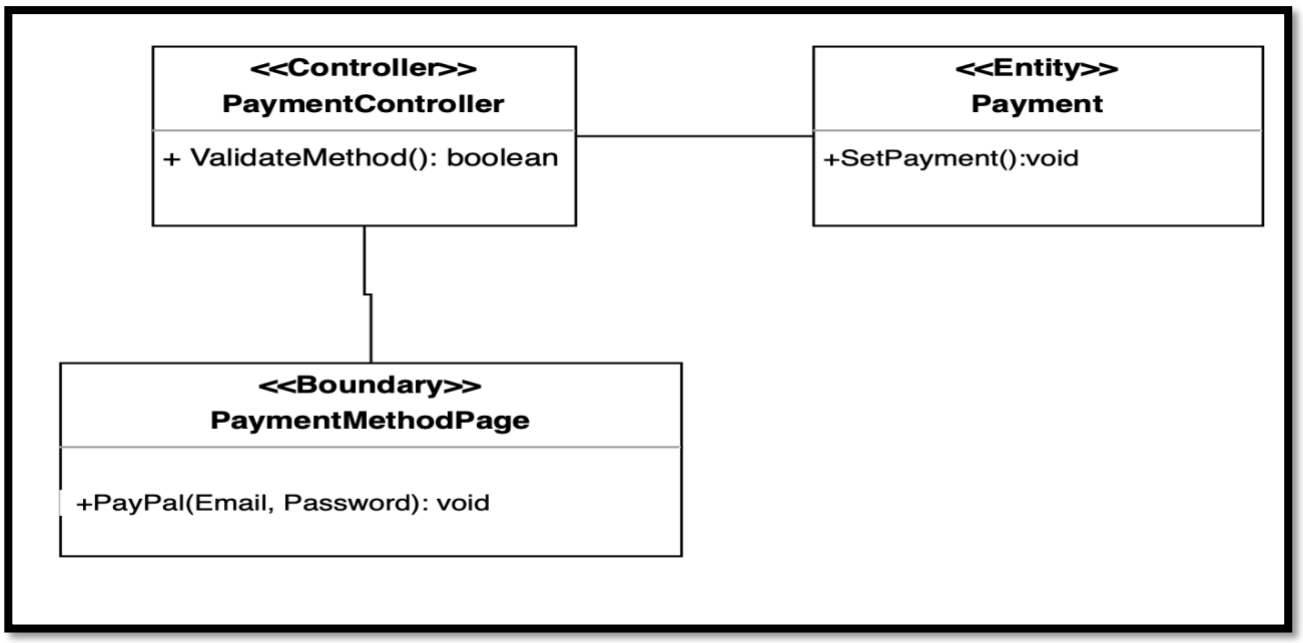


Figure 24 Pay an order VOPC

10.3 View active requested order VOPC

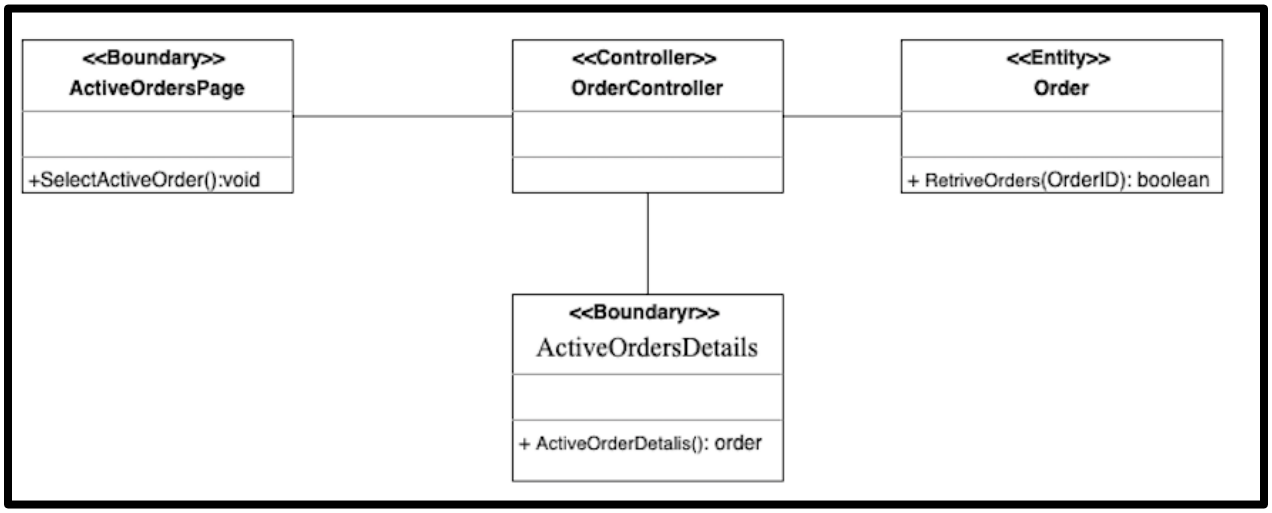


Figure 25 View active requested order VOPC

10.4 View history orders VOPC

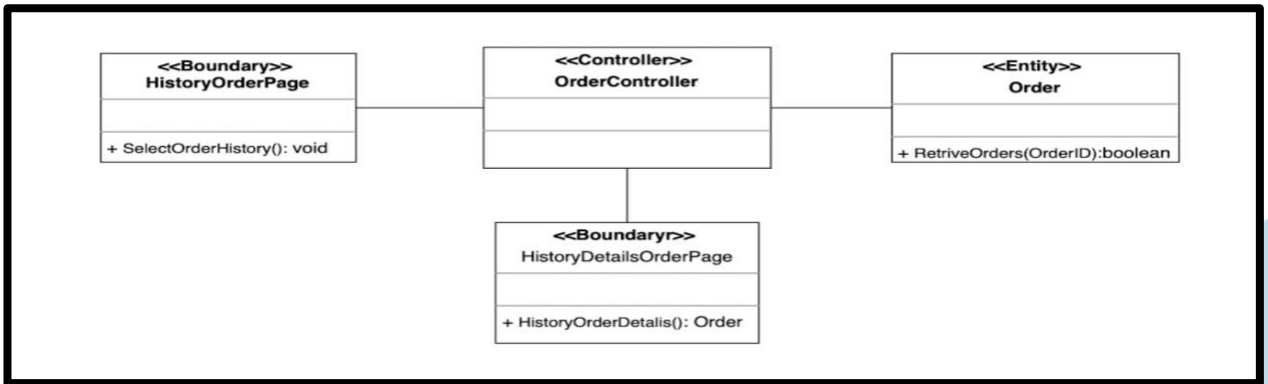


Figure 26 View history order VOPC

10.5 Report an issue VOPC

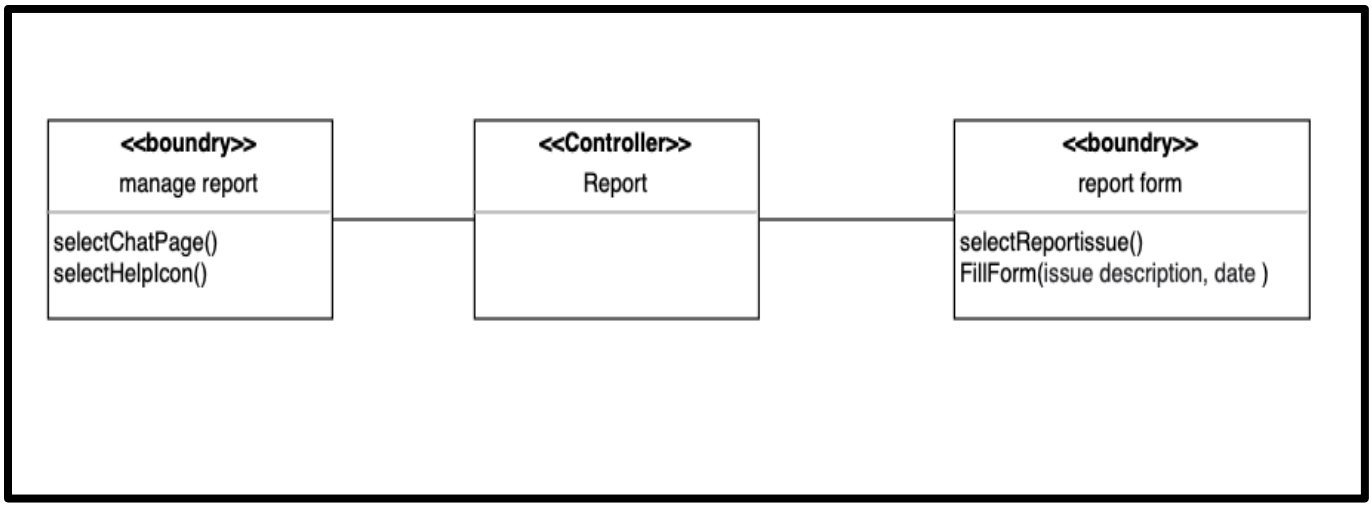


Figure 27 Report an issue VOPC

10.6 Rate an order VOPC

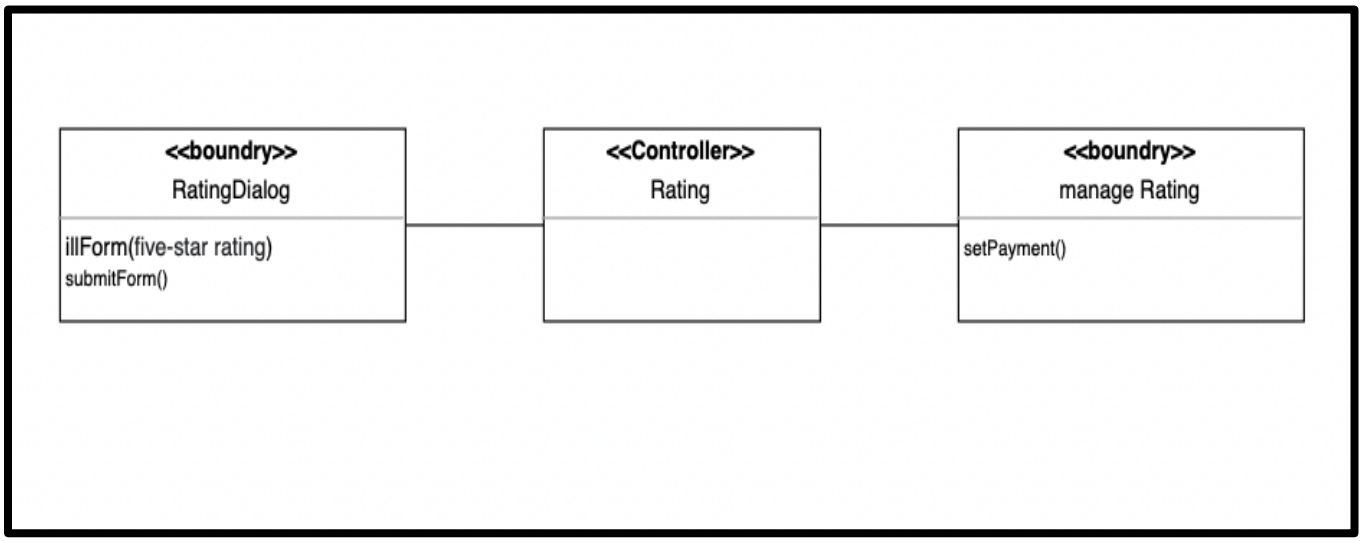


Figure 28 Rate an order VOPC

10.7 Choose an offer VOPC

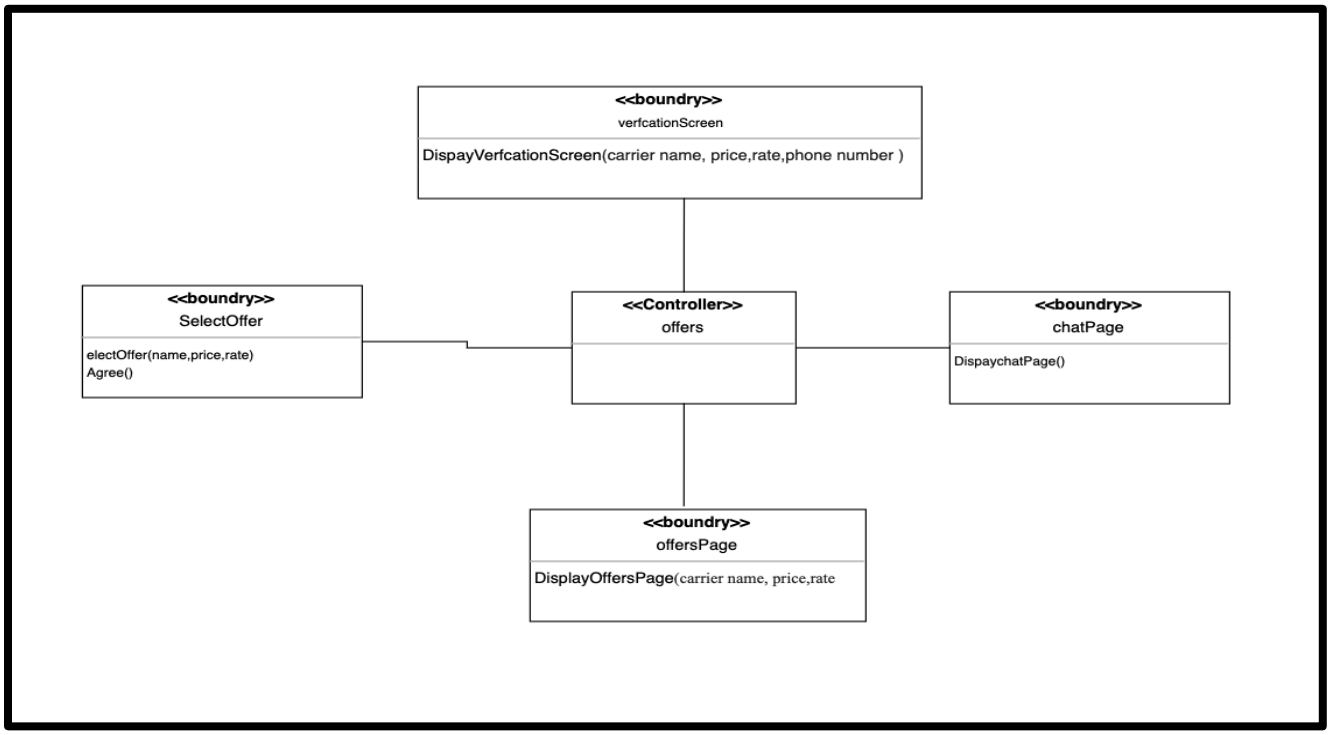


Figure 29 Choose an offer VOPC

10.8 Select an Oder VOPC

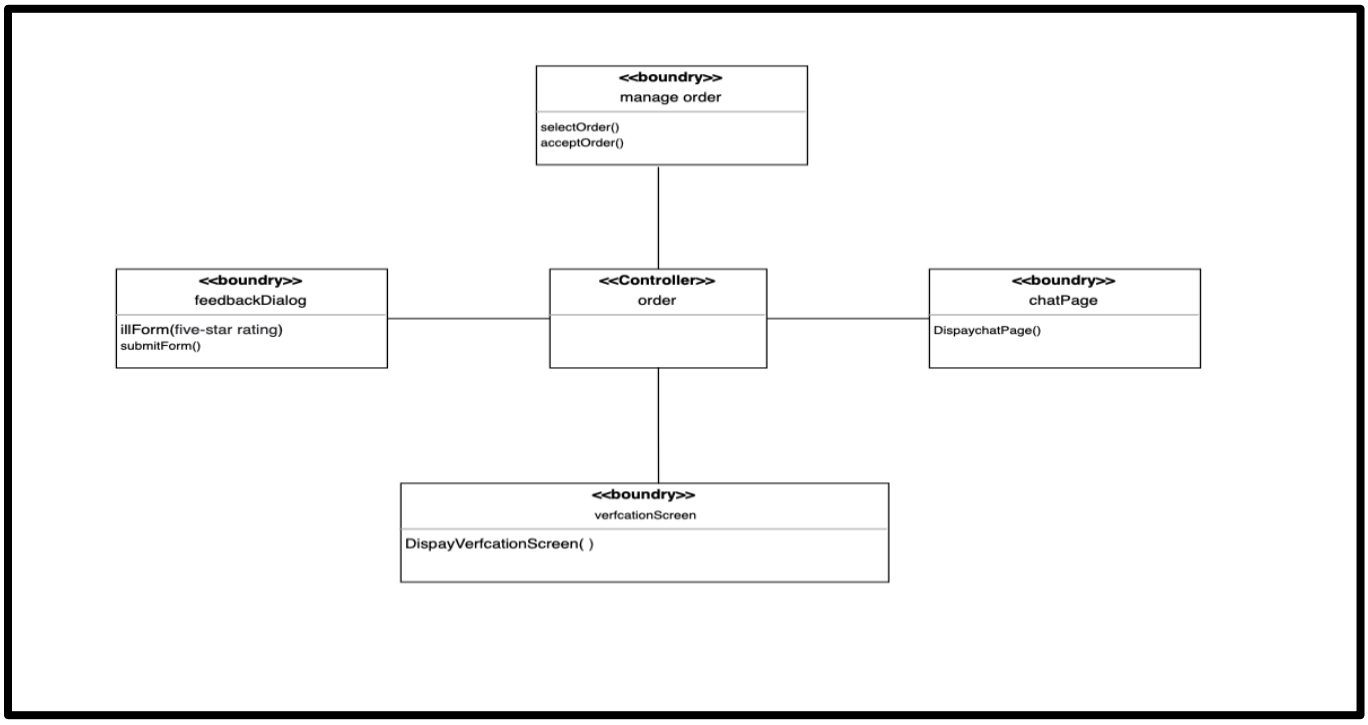


Figure 30 Select an order VOPC

10.9 Register VOPC

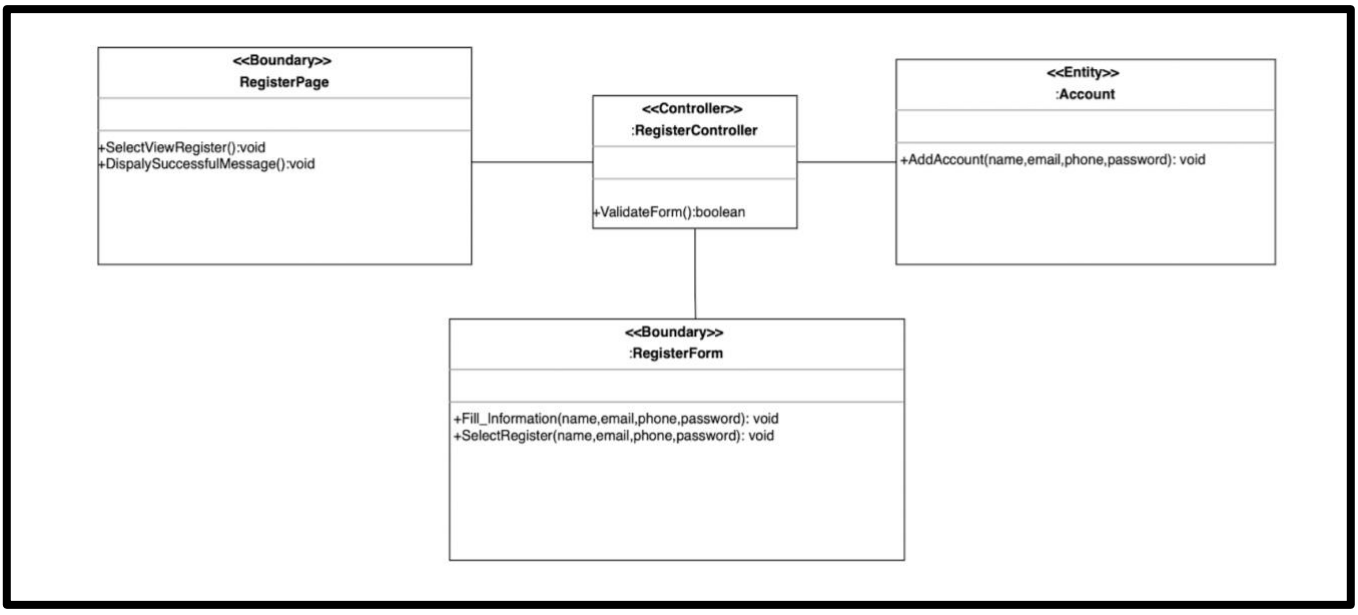


Figure 31 Register VOPC

10.10 Manage account VOPC

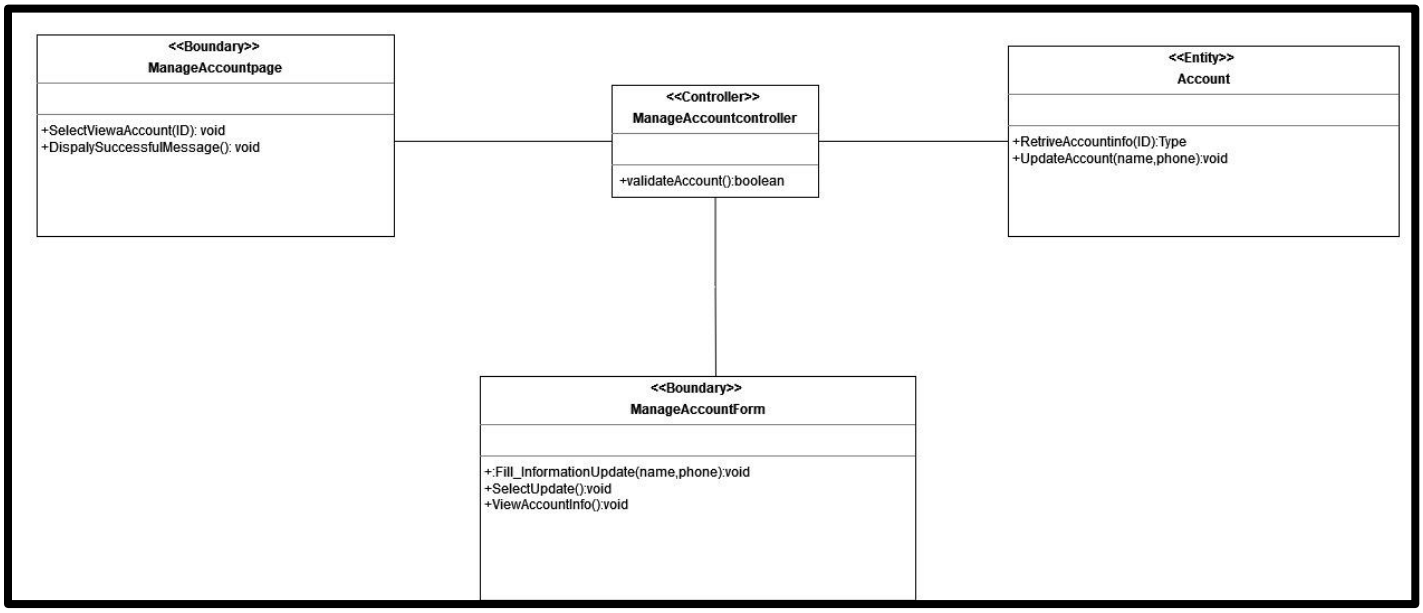


Figure 32 manage account VOPC

10.11 Unified VOPC

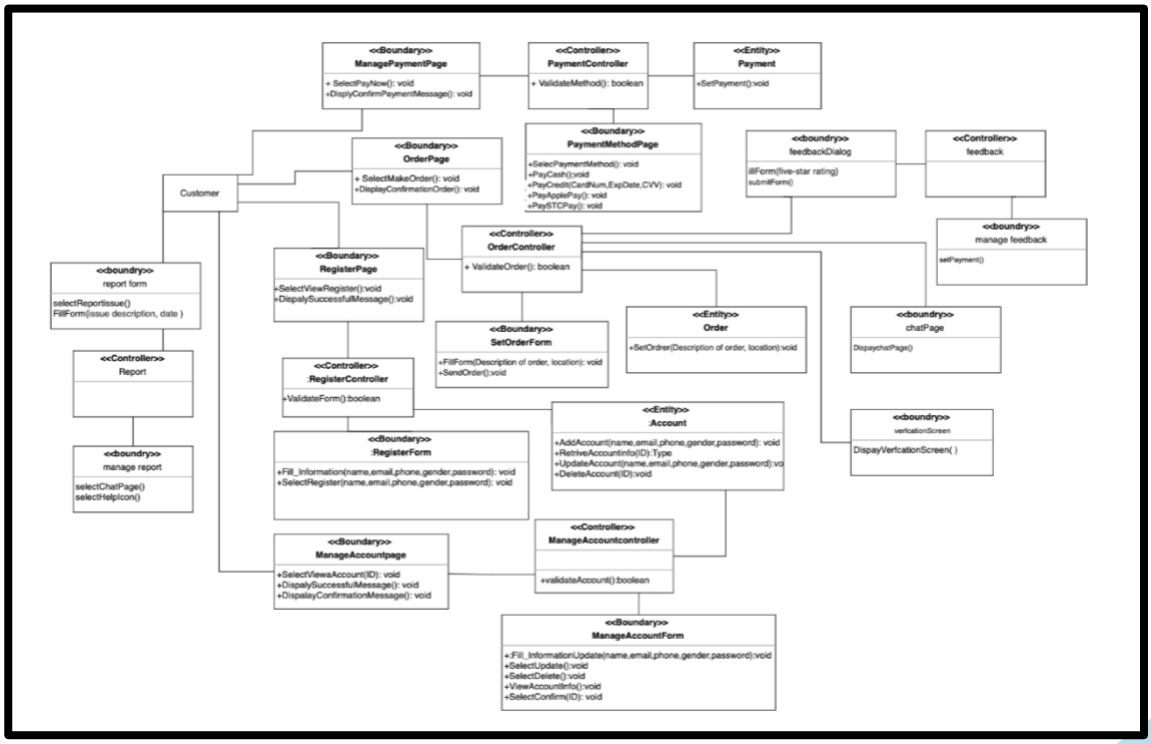


Figure 33 unified part1 VOPC

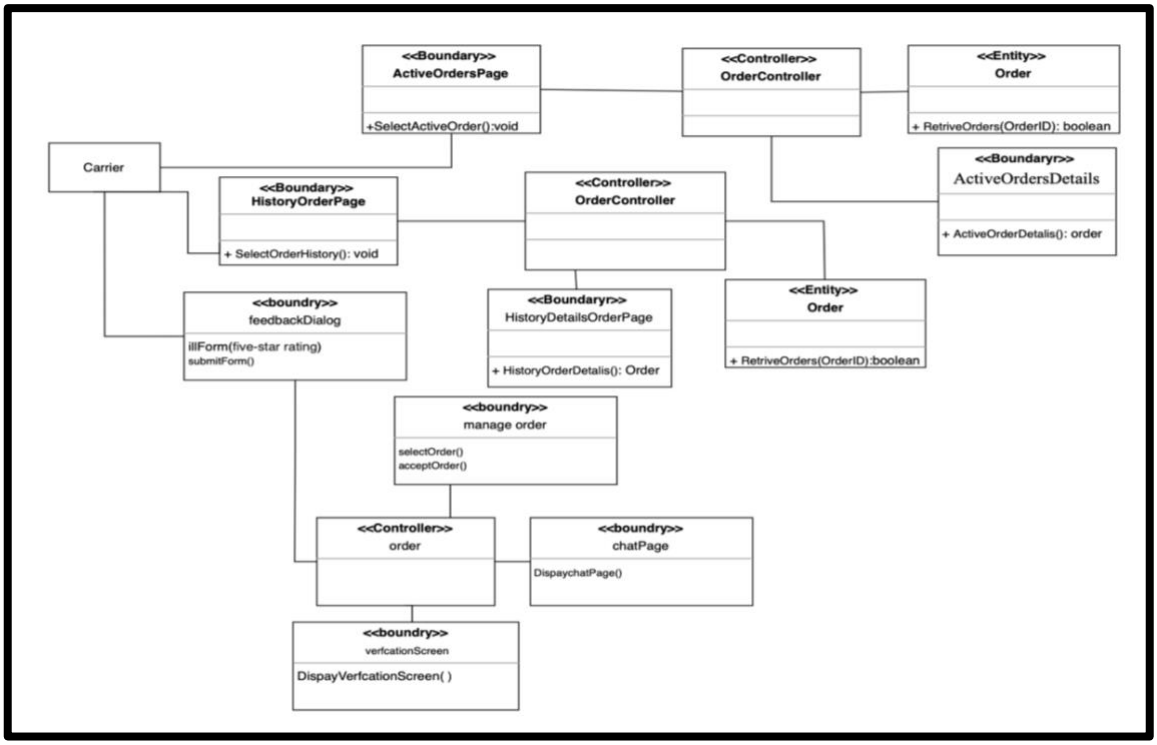


Figure 34 unified part2 VOPC

11. Class Diagram

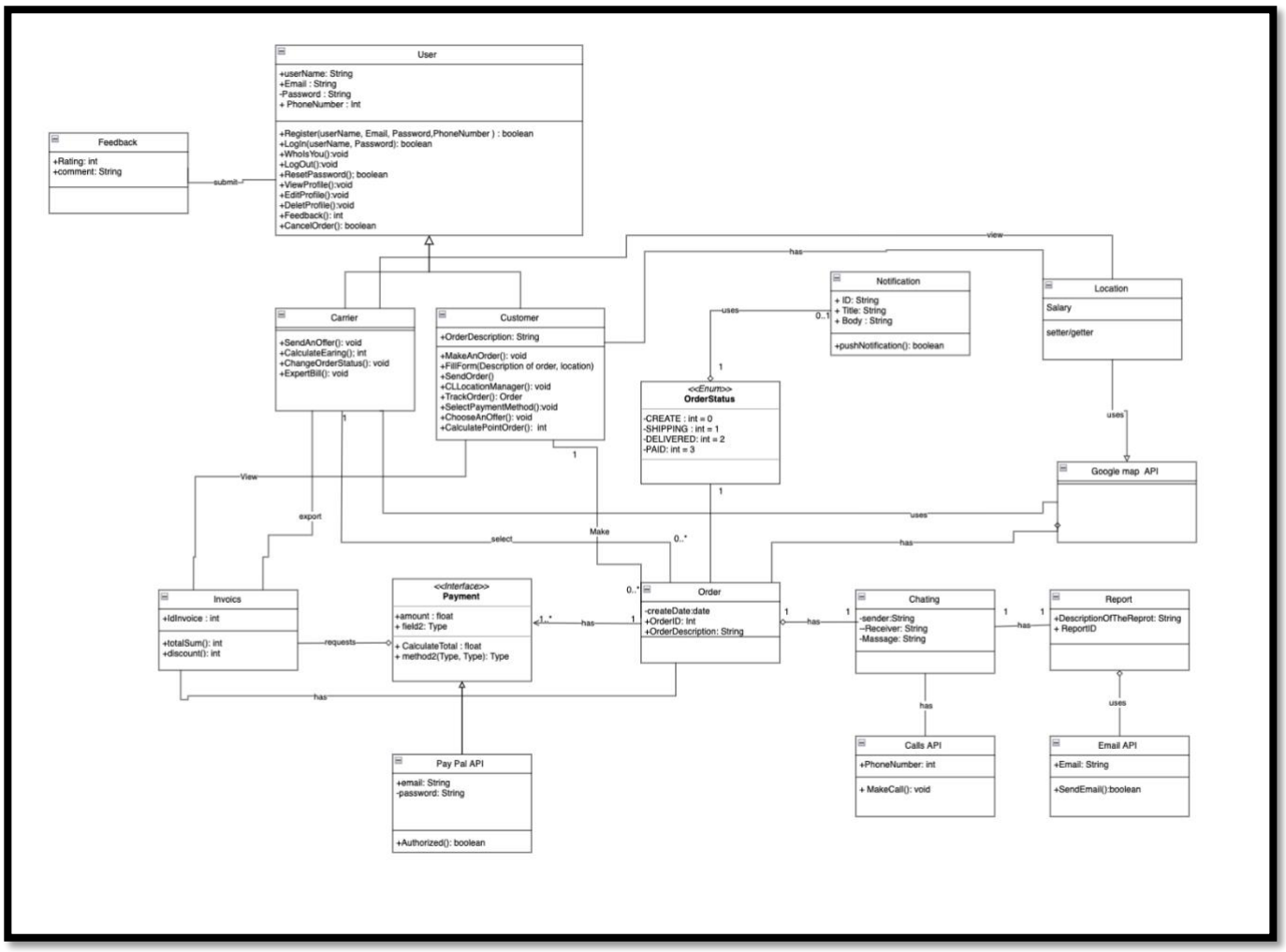


Figure 35 Class diagram

12.System Architecture

Systems Architecture is a generic discipline to handle objects (existing or to be created) called “systems", in a way that supports reasoning about the structural properties of these objects. Systems Architecture is a response to the conceptual and practical difficulties of the description and the design of complex systems. [20]

12.1 Goal of Architecture:

The primary goal of the architecture is to identify requirements that would affect the structure of the application. A well-laid architecture reduces the risks associated with building a technical solution derived from the nonfunctional requirements. [21] These are the most important design goals that the system will focus on:

12.1.1 Availability:

Availability describes how likely the system is accessible for a user at a given point in time. It is the most business-critical requirement, even in our application, the app should be available all the time for customers to order at any time. [22]

12.1.2 Usability:

Specifies how easy the system must be to use. It focuses on the usability quality attribute that measures how well a customer can use our application to achieve a defined goal effectively, efficiently, and satisfactorily. [23]

12.1.3 Security:

The intent of security is to ensure that systems are initially configured to a secure state, and that they remain in that state over the life of the system used to protect sensitive data, it specifies how the system will be protected against malware attacks or unauthorized access, it could be achieved by using encryption in the password.[24]

12.2 Architectural Style:

The architectural style is a particular solution to specific software, which how to organize the code created for the software. It focuses on building the and allowing an appropriate interaction between the various modules for giving the right results upon implementation. [25]

The system architecture of KSU Delivery Service is a MVVM (Model View View Model) architecture, considering it is the most popular architectural pattern for designing iOS apps. The MVVM architecture separates the business and presentation logic from the UI and improves the testability and maintainability of the system it contains:

12.2.1 View Model:

Represents the data as it should be presented in the view and contains presentation and business logic. It Manages UI behavior and state, interprets user inputs into actions upon business rules and data and prepares the data retrieved from a model to be presented to the user. It is located between the View and Model layers. This is where the controls for interacting with View are housed, while binding is used to connect the UI elements in View to the controls in View Model.

12.2.2 Model:

represents simple data, it only holds the data and has nothing to do with the business logic, the model just sends and retrieves data from the database. It houses the logic for the program, which is retrieved by the View Model upon its own receipt of input from the user through View.

12.2.3 View:

is the collection of visible elements, which also receives user input. This includes user interfaces (UI), animations and text. The content of View is not interacted with directly to change what is presented.

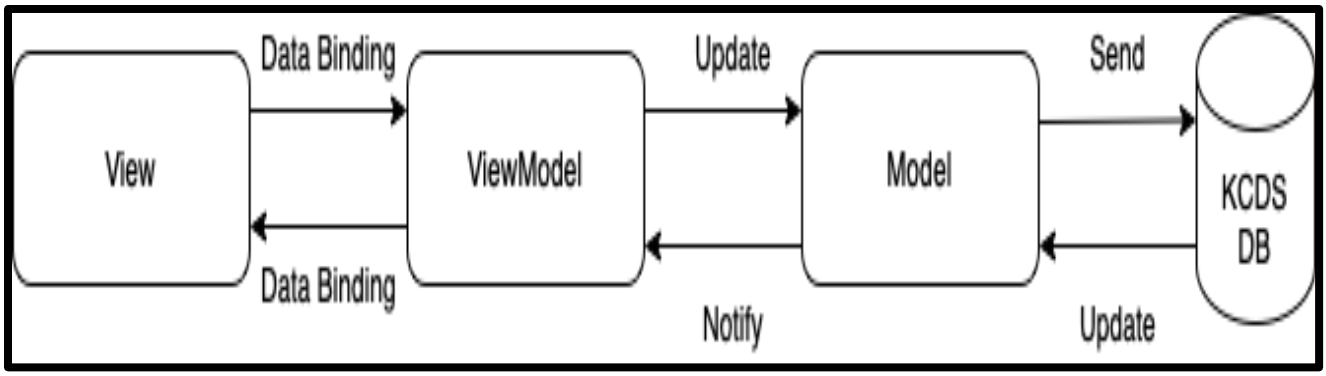


Figure 36 MVVM architecture

12.3 Component diagram

Component diagram is a special kind of diagram in UML. The purpose of the component diagram is also different from all other diagrams discussed so far. It does not describe the functionality of the system, but it describes the components used to make those functionalities.[26]

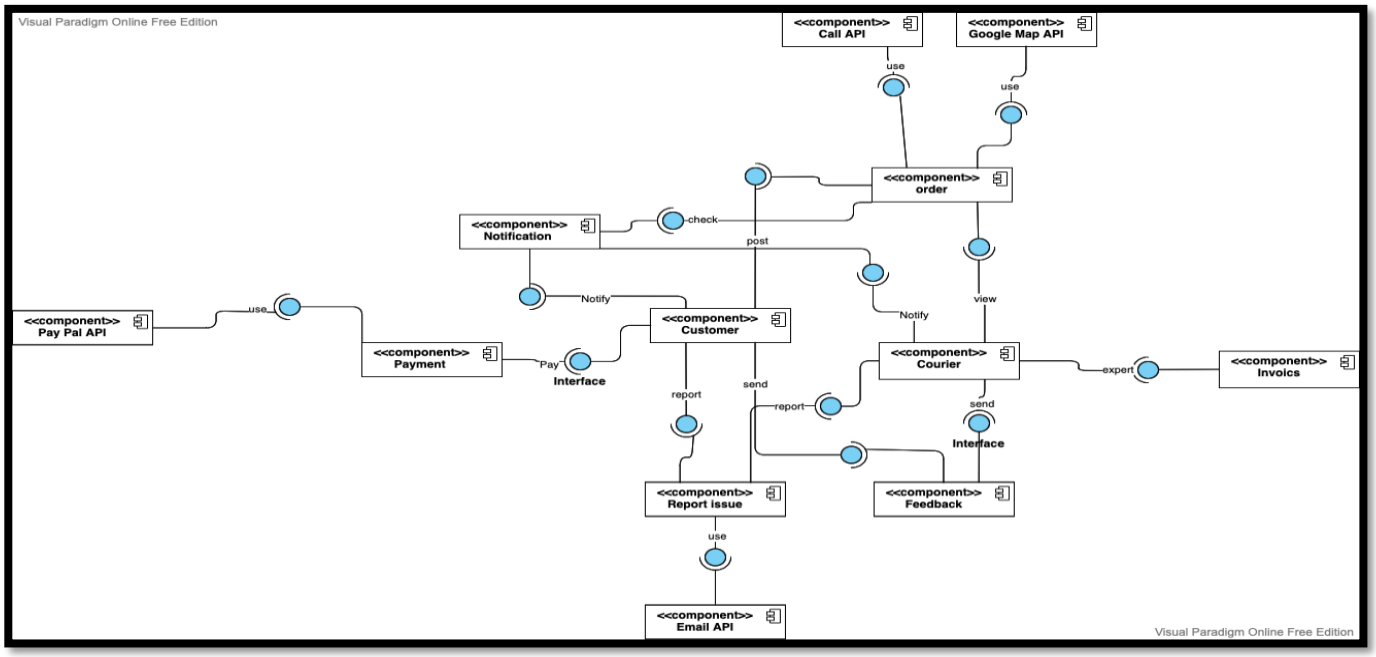


Figure 37 Component diagram

13. User Interface Mockup

This part shows how the system looks and feel by implementing the design of the basic functionalities and illustrating the user interface of our application KSU Campus Delivery Service. The mock-up was designed using Figma which is an application that helps to build prototypes without the need of using any code. The mock-ups will help us to have a view of the main functionalities and how the functionalities will be provided to the user as well as helping the front-end developer with defining the system's structure. The below figures are some examples of KSU Campus Delivery Service.

13.1 Basic UI

13.1.1 Forget password

The figures below show the Customer forgot password interfaces. By entering KSUDS Application, Customer selects "Forgot password", and will be directed to the "Forgot password" page. Customer should enter his registered email to receive the change password mail. By clicking "Send" customer should receive the mail and change the password to access KSUDS Application.

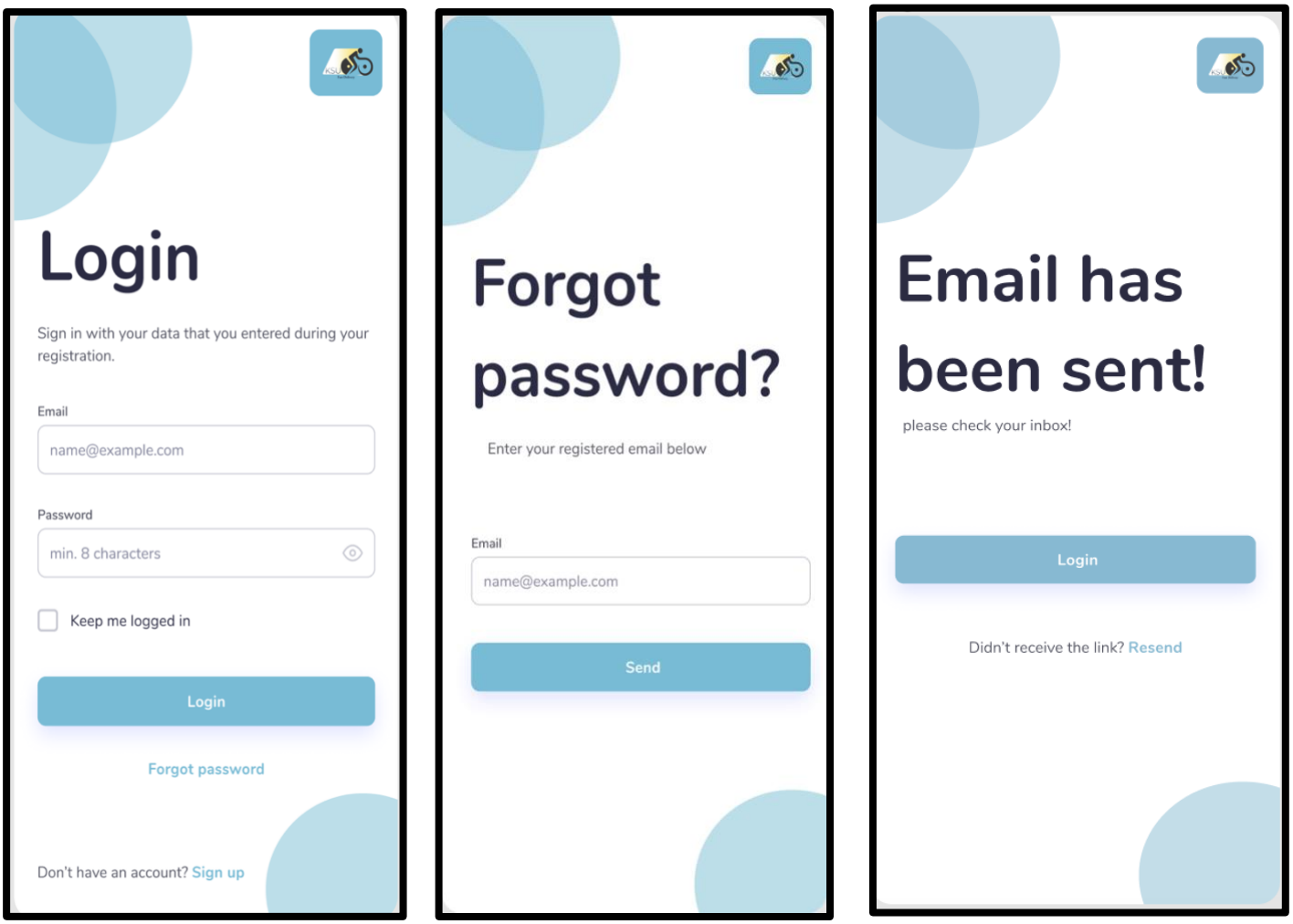


Figure 38 forget password scenario

13.1.2 Customer profile

The figure below shows the Customer profile interface. Customer can view and edit his/her profile, he/she can also check his/her earnings from past orders in order to use it in future orders.

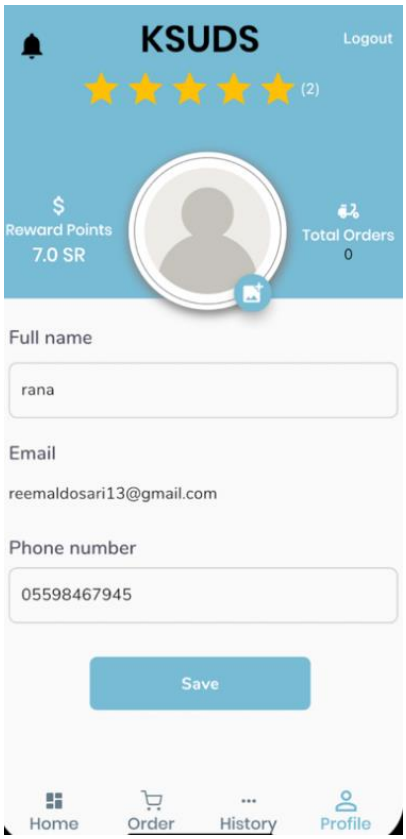


Figure 39 customer profile

13.1.3 Sign up

The figures below show the Welcome and Sign-up interfaces. User will first select whether he/she is "Customer" or "Courier". Then the user will be able to sign up to KSUDS Application by filling the form with his/her information such as "Name", "Email", "Phone number" and "Password". After that the user will be directed to the log in page.

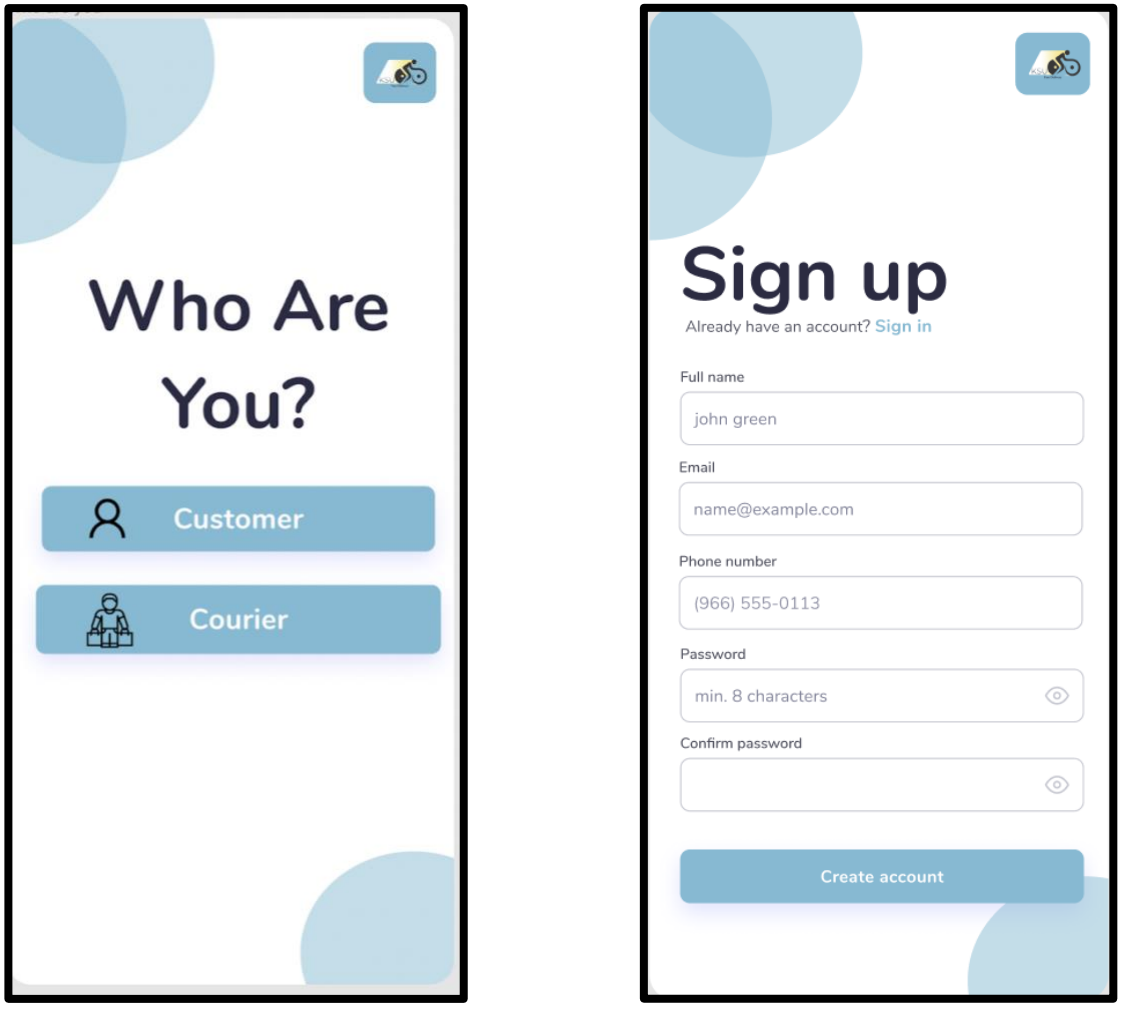


Figure 40 sign up flow

13.1.4 History and active orders

The figures below show the history order and the active order interfaces. The customer views his order history including (store name, status, date, price, order ID, order details) by clicking on the history button also the customer can view his active order including (store name, status, price, order ID, order details) by clicking on the Order button.

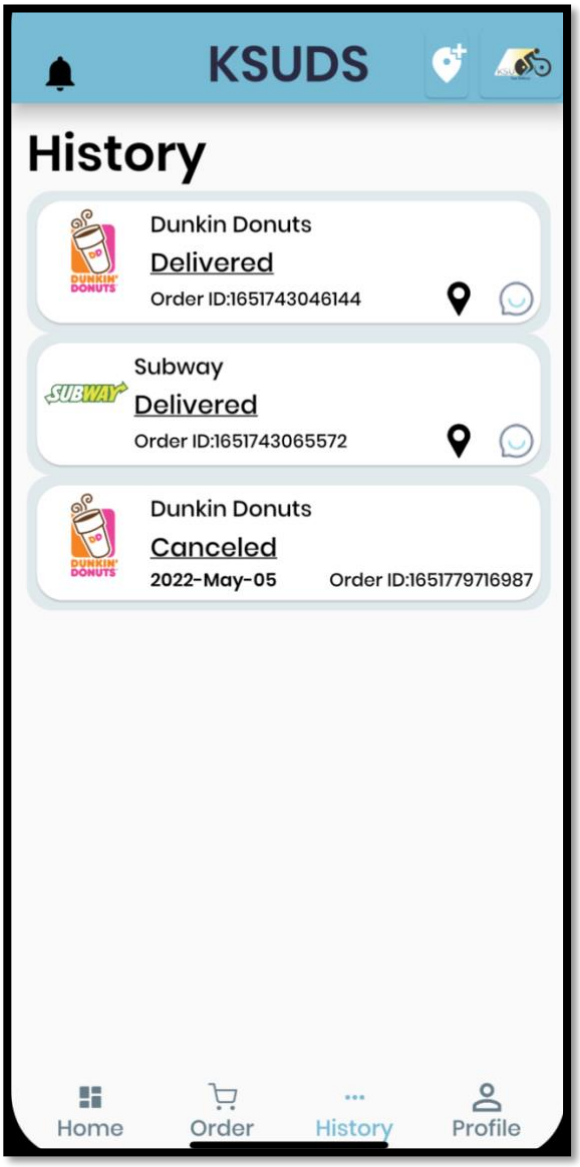


Figure 41 history and active orders figure

13.1.5 Report issue

The figures below show the help and support center and report an issue interface. The Customer views the help and support icon , the customer fills the title and the message then click on the “send” button.

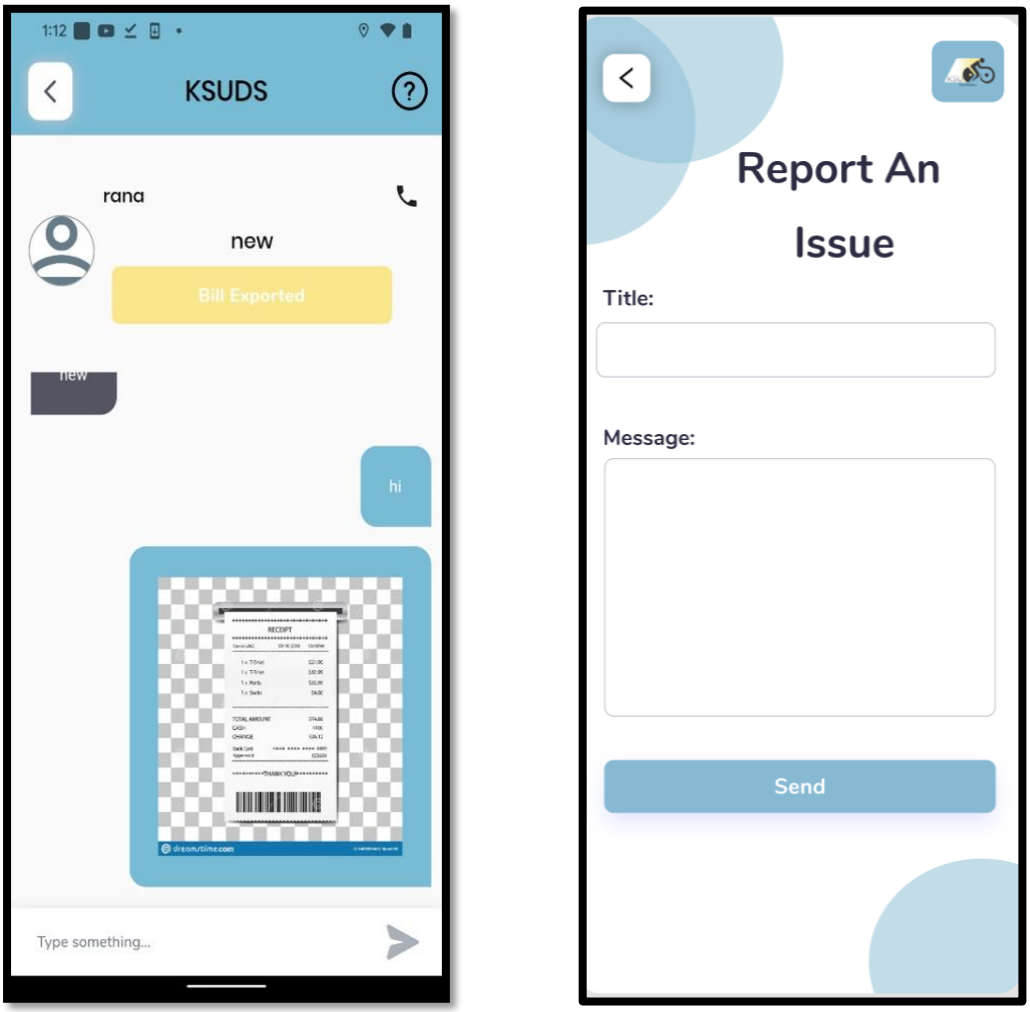


Figure 42 report issue flow

13.2 Customer UI

13.2.1 Order view

The figures below show the customer home page, and the store menu interfaces. The customer selects the desired store type including (food, bookstore, coffee). Then customer selects the desired store and views the store menu.

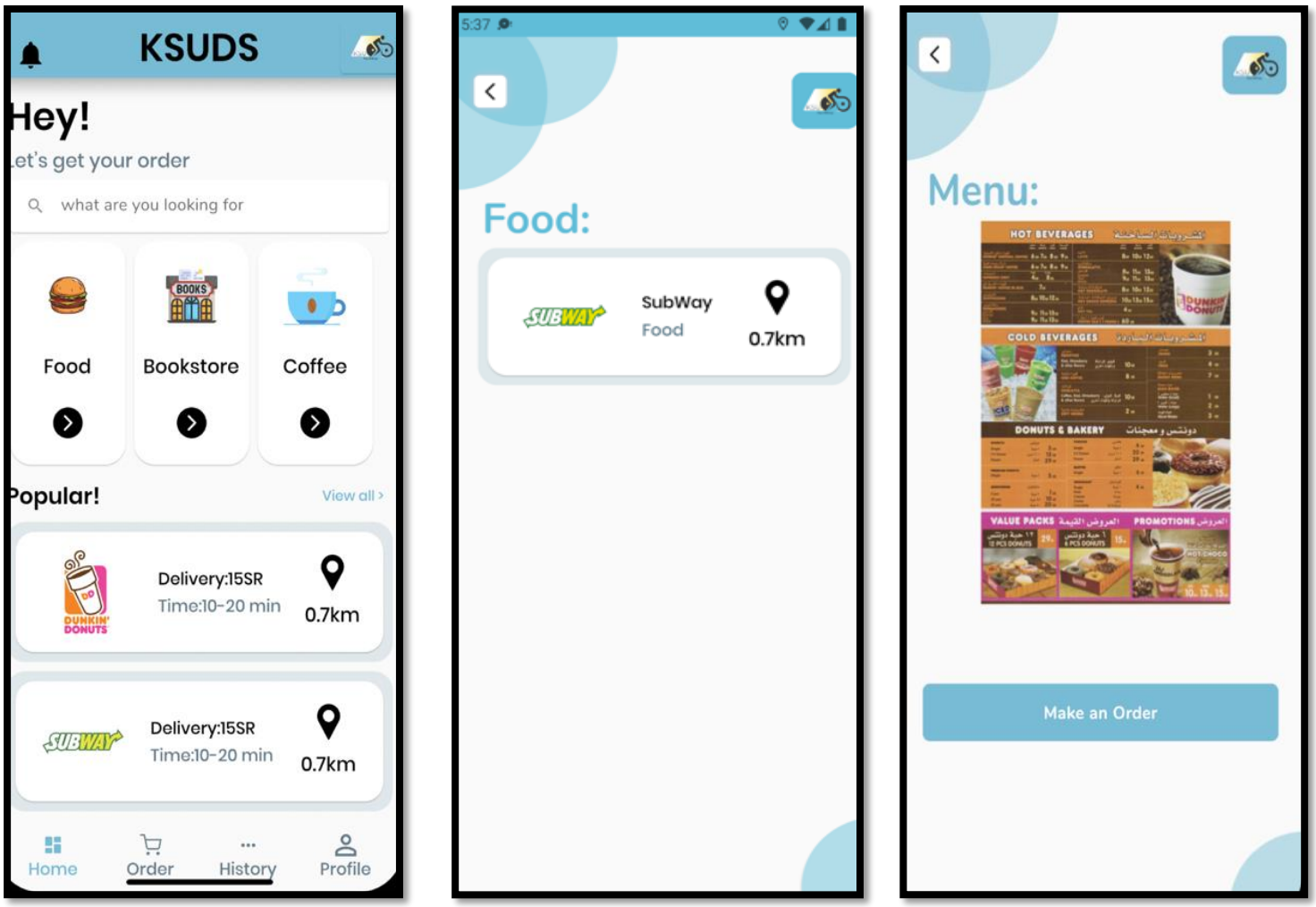


Figure 43 order view flow

13.2.2 Make an order

The figures below show the order and the offers interfaces. The customer types of his order and then press “locate” button, Customer set his location and then press confirm, customer views the offers and select from them by pressing the “send” icon.

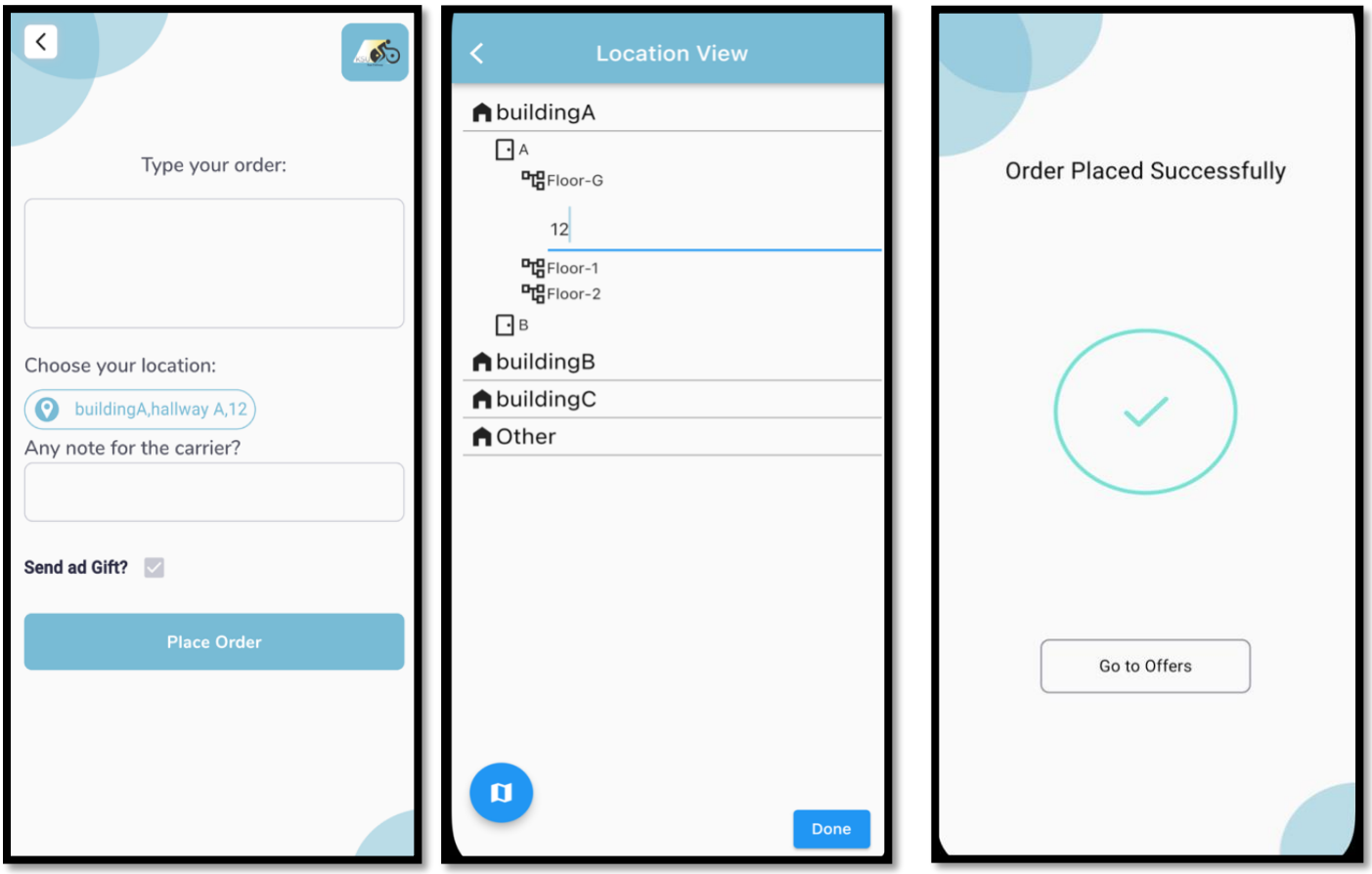


Figure 44 make an order flow

13.2.3 Select offer customer

The figures below show the offers,. The customer views the offers and clicks on the “Select offer” button, Customer chat with the courier by typing or calling him.

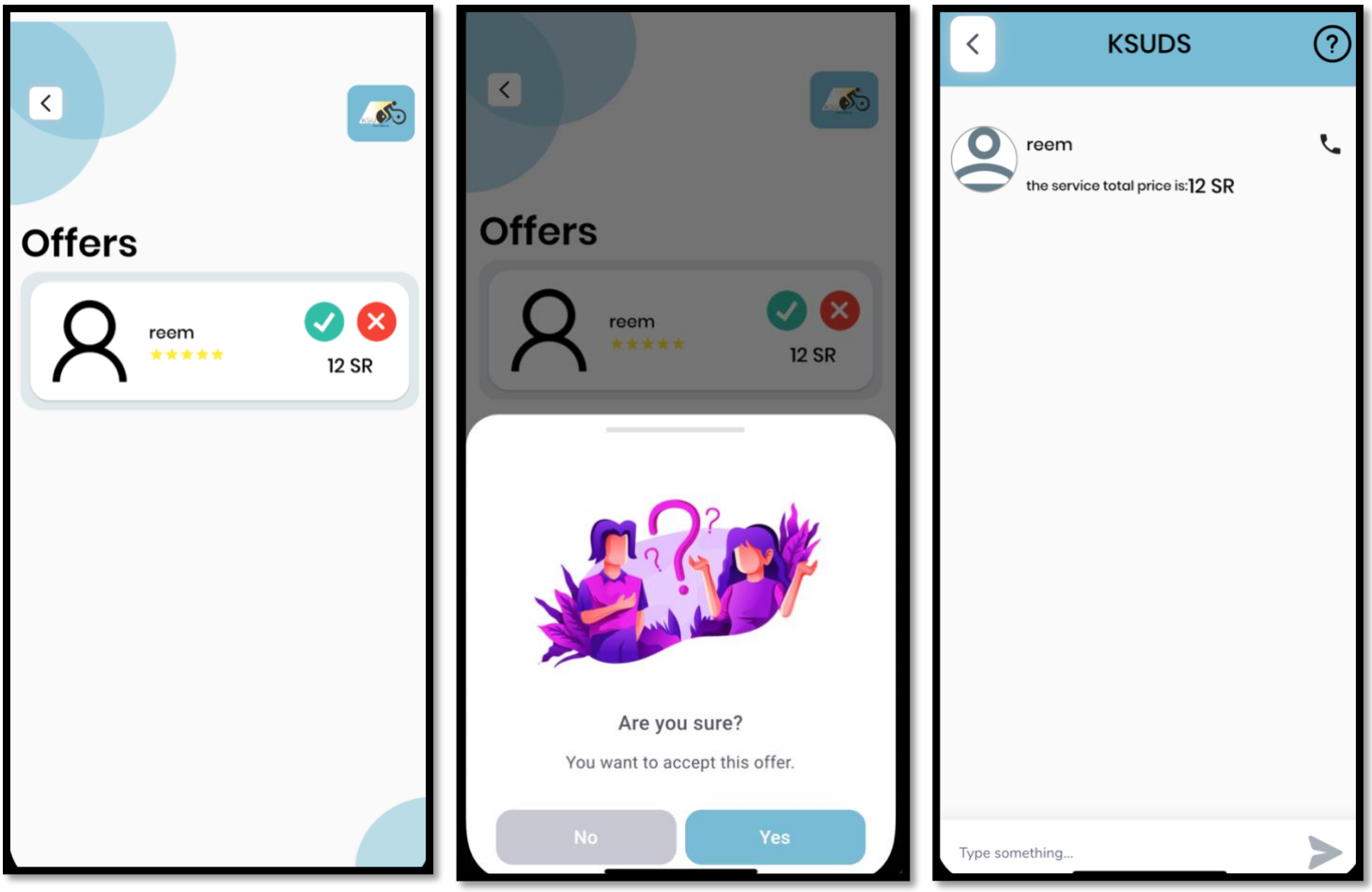


Figure 45 select offer customer flow

13.2.4 Payment

The figures below show the payment, rating order interfaces, The customer fill his card information including (name, card number, expiry date, cvv) and then selects the “add card” button, customer rate the order and selects “submit” button, Customer selects “home” button to return.

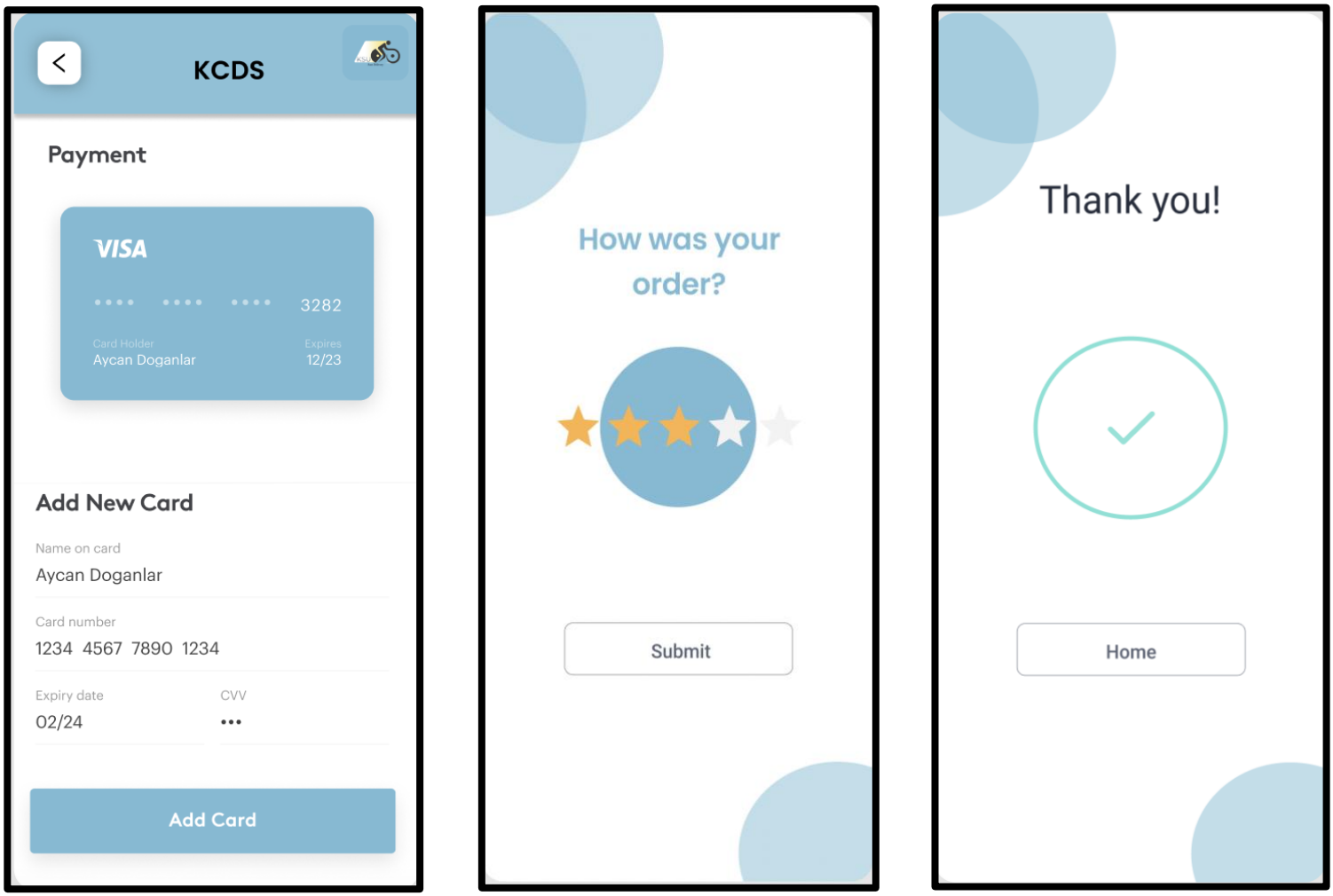


Figure 46 payment flow

13.3 Courier UI

13.3.1 Select offer courier

The figures below show the courier home page interface. The Courier view all the orders and then selects the desired order by pressing “Select order” button, courier type his delivery price and selects “send offer” button, the courier view order information and can select “pick order” button or “cancel” button

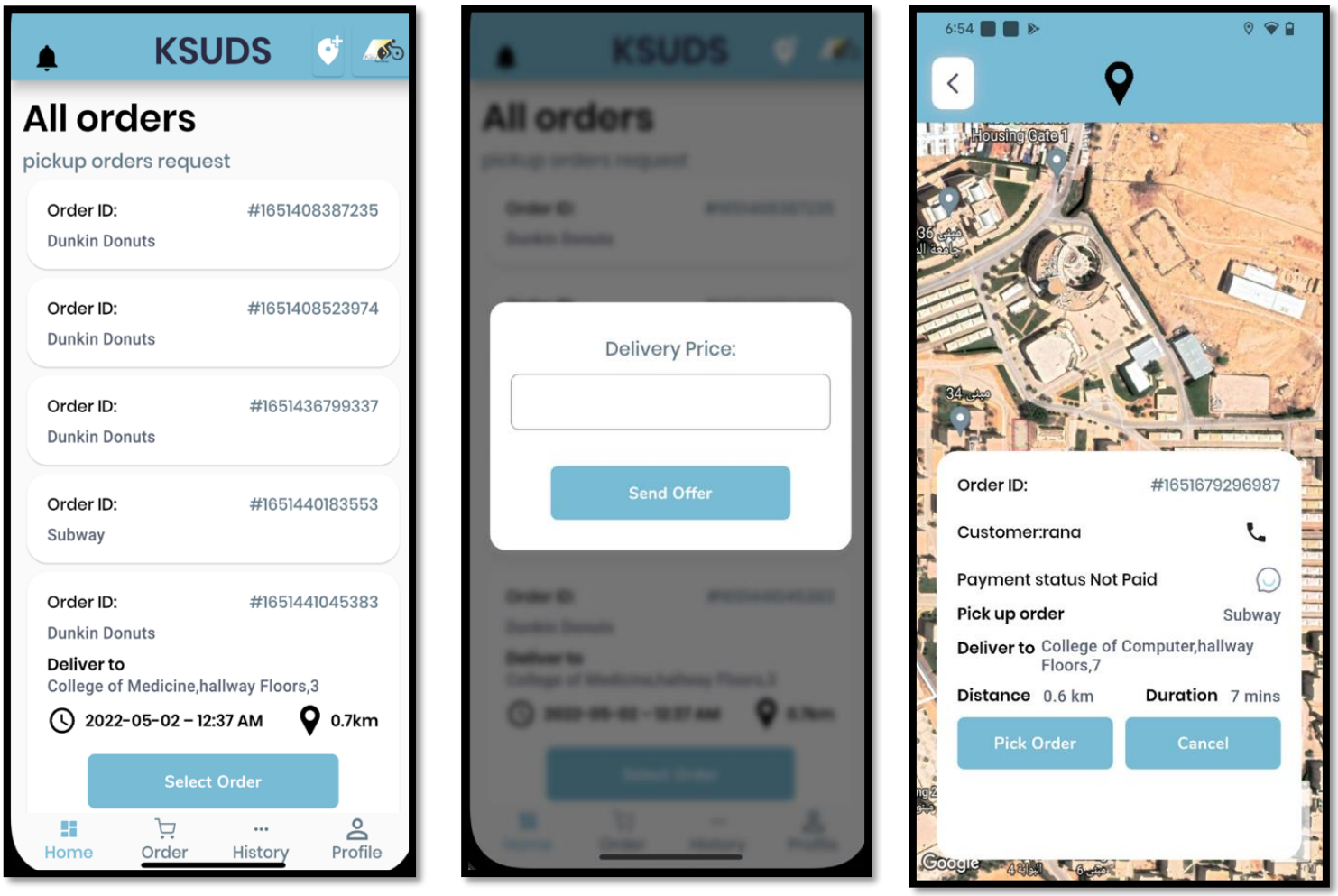


Figure 47 select offer courier flow

13.3.2 Courier profile

The figure below shows the courier profile interface. The courier can view and edit his/her profile, also the courier could view his total earnings.

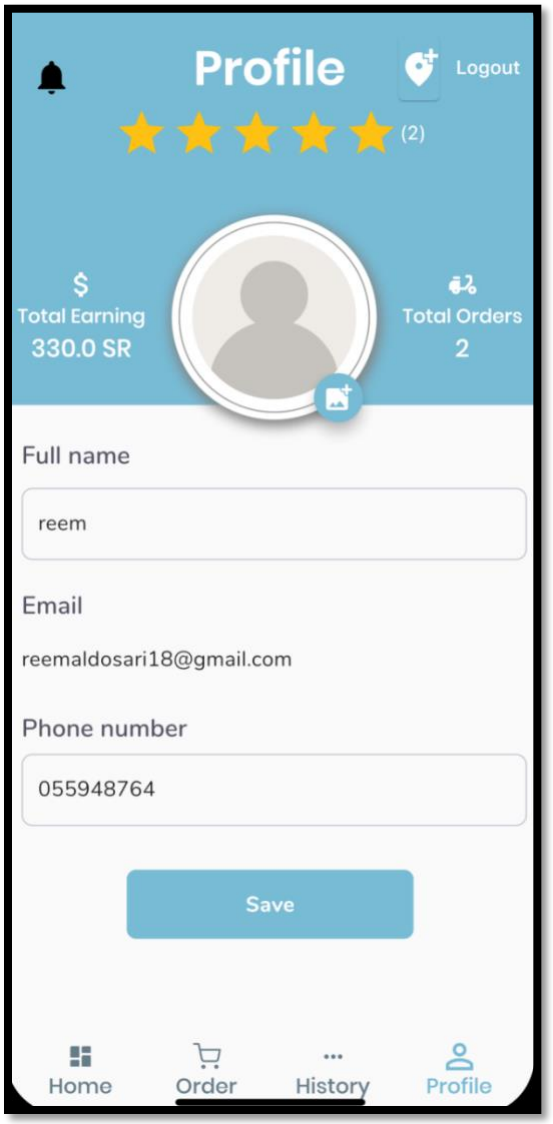


Figure 48 courier profile

13.3.3 Submitting bill

The figures below show the courier chat, export bill and the rating delivery experience interfaces. The courier can chat with the customer by typing or calling him, then the courier press the "export bill" button, the courier views export bill including "Customer name, delivery price) and then type the order price and click on the "Calculate" button to view the Total price, and click on the "upload receipt" after that courier selects "Export bill" button, Courier rate his delivery experience and then select "submit" button.

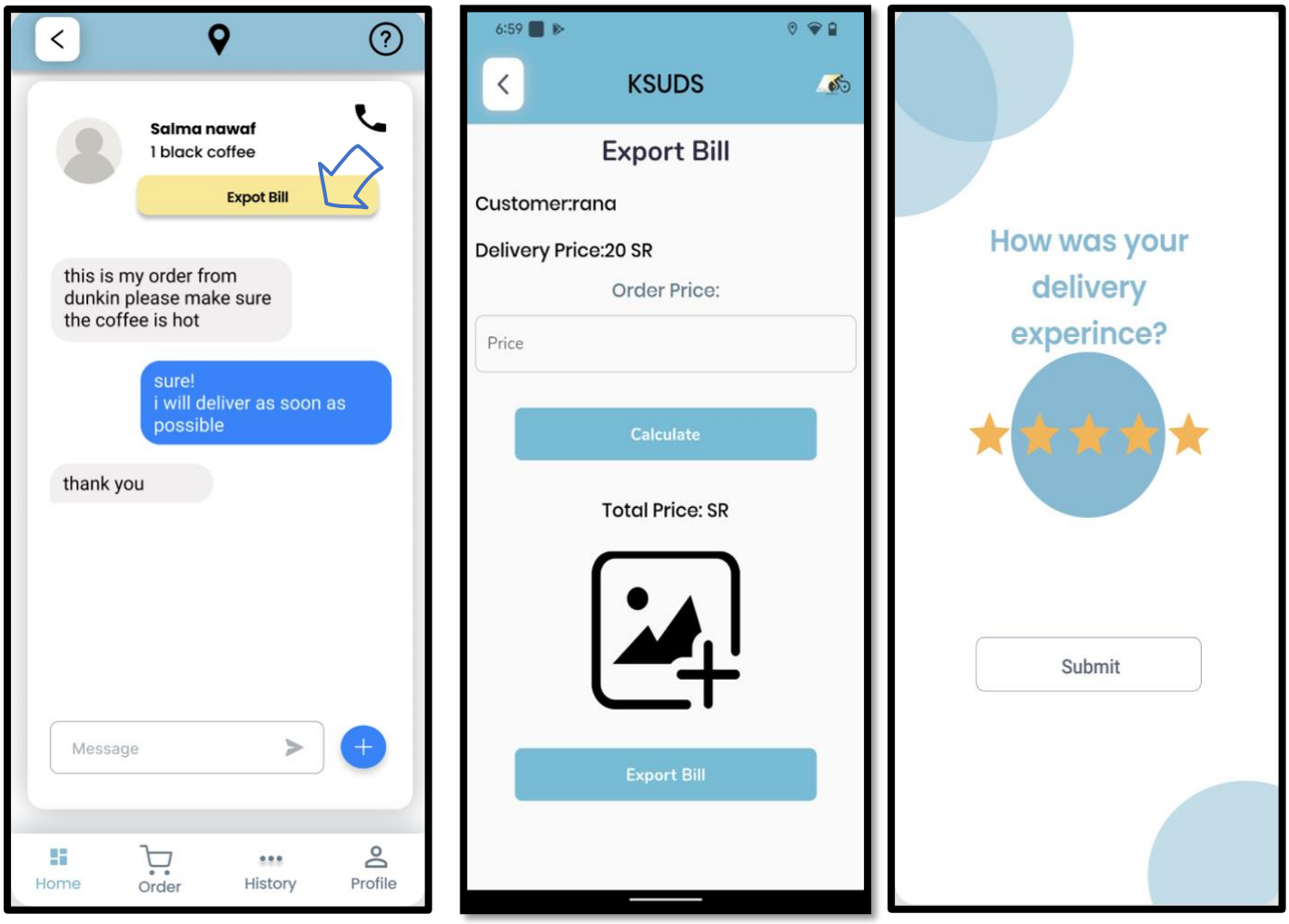


Figure 49 submitting bill flow

14. Database Schema

14.1 Database Schema

The database used for the KSUDS system is Cloud Firestore provided by Firebase. Cloud Firestore is a NoSQL document database that lets you easily store, sync, and query data for mobile and web apps on a global scale. [27] The characteristics lead to choose Cloud Firestore as follow:

14.2 Flexibility

The Cloud Firestore data model supports flexible and hierarchical data structures. Save the data in a document organized in a collection. Documents can contain complex nested objects in addition to subcollections.

14.3 Uses collections and documents to structure and query data

This data model is familiar and intuitive for many developers. It also allows for expressive queries. Queries scale with the size of your result set, not the size of your data set, so you'll get the same performance fetching 1 result from a set of 100, or 100,000,000.

14.4 Scale globally

Cloud Firestore Powered by Google's storage infrastructure automatic multi-region data replication, strong consistency guarantees, atomic batch operations, and real transaction support.

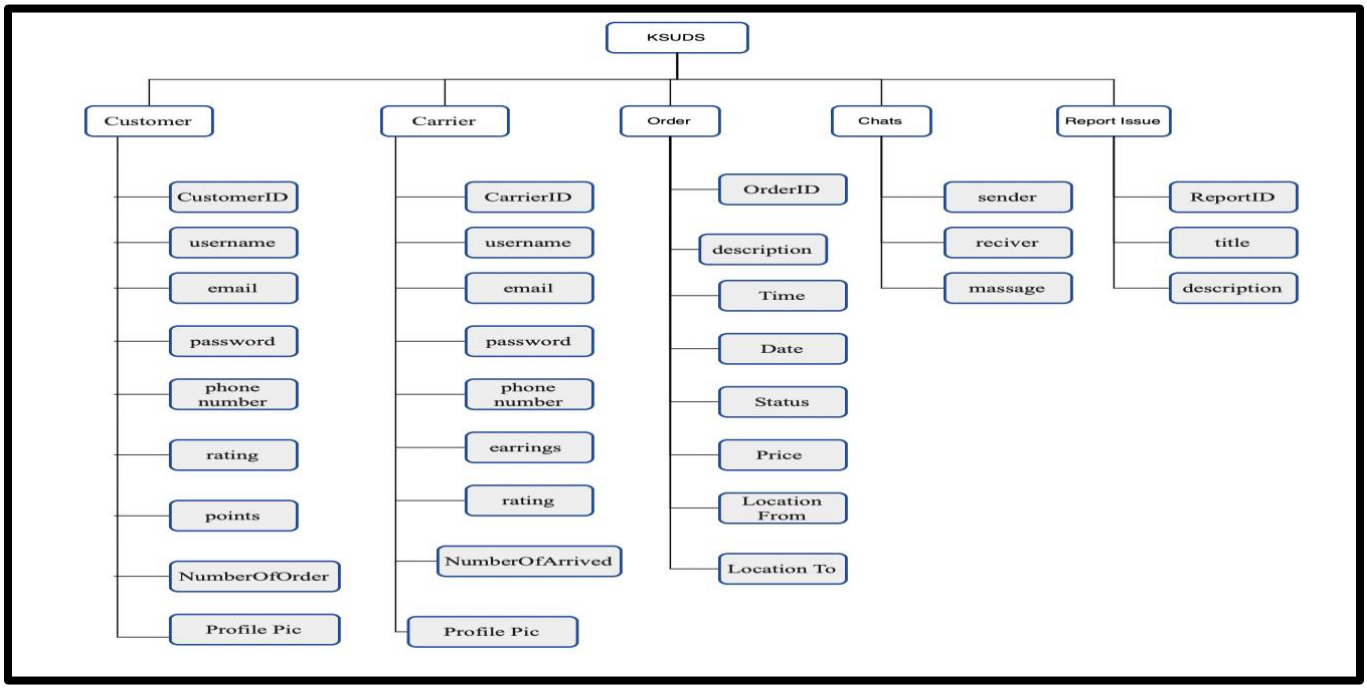


Figure 50 Database Schema

15.Algorithms

An algorithm is a set of instructions for solving a problem or accomplishing a task.[28] This section contains used algorithms in the system with a brief description of their functions.

15.1 Calculate discount amount

```
discounted_price=0
discount=0
Read CustomerPoint;
Read original_price;

if CustomerPoint <25
Print "Sorry, You Should Have More Than 25 Point to Take the
Discount";

Then CustomerPoint = CustomerPoint;


if CustomerPoint == 25
Then discount=25;
Then discounted_price = original_price - (original_price *
discount / 100);
Print "Congratulations You Have 25% discount";
Then CustomerPoint = 0;

if CustomerPoint == 50
Then discount=50;
Then discounted_price = original_price - (original_price *
discount / 100);
Print "Congratulations You Have 50% discount";
Then CustomerPoint = 0;

if CustomerPoint == 100
Then discount=100;
Then discounted_price = original_price - (original_price *
discount / 100);
Print "Congratulations Your order for free";
Then CustomerPoint = 0;

 if CustomerPoint > 100
Then discount= CustomerPoint-100;
Then discounted_price = original_price - (original_price *
discount / 100);
Print "Congratulations Your order for free";
Then CustomerPoint = CustomerPoint;

END IF;
END IF;
END IF;
END IF;
END IF;
```

Figure 51 Calculate discount amount algorithm

15.2 Shortest path

We will use Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph.[29][30]

```
// Let v1 be the origin vertex,
//   and initialize W and ShortDist[u] as
W := {v1}
ShortDist[v1] :=0
FOR each u in V - {v1}
ShortDist[u] := T[v1,u]‹
// Now repeatedly enlarge W
//   until W includes all verticies in V
WHILE W <> V
// Find the vertex w in V - W at the minimum distance
//   from v1
MinDist := INFINITE
FOR each v in V - W
IF ShortDist[v] < MinDist
MinDist = ShortDist[v]
w := v
END IF
END FOR
// Add w to W
W := W U {w}
// Update the shortest distance to vertices in V - W
FOR each u in V - W
ShortDist[u] := Min(ShorDist[u],ShortDist[w] + T[w,u])
END while
```

Figure 52 shortest path algorithm

16.Expected Deployment

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them.[31] In this section we show how we organized the system hardware or software in the deployment diagram.

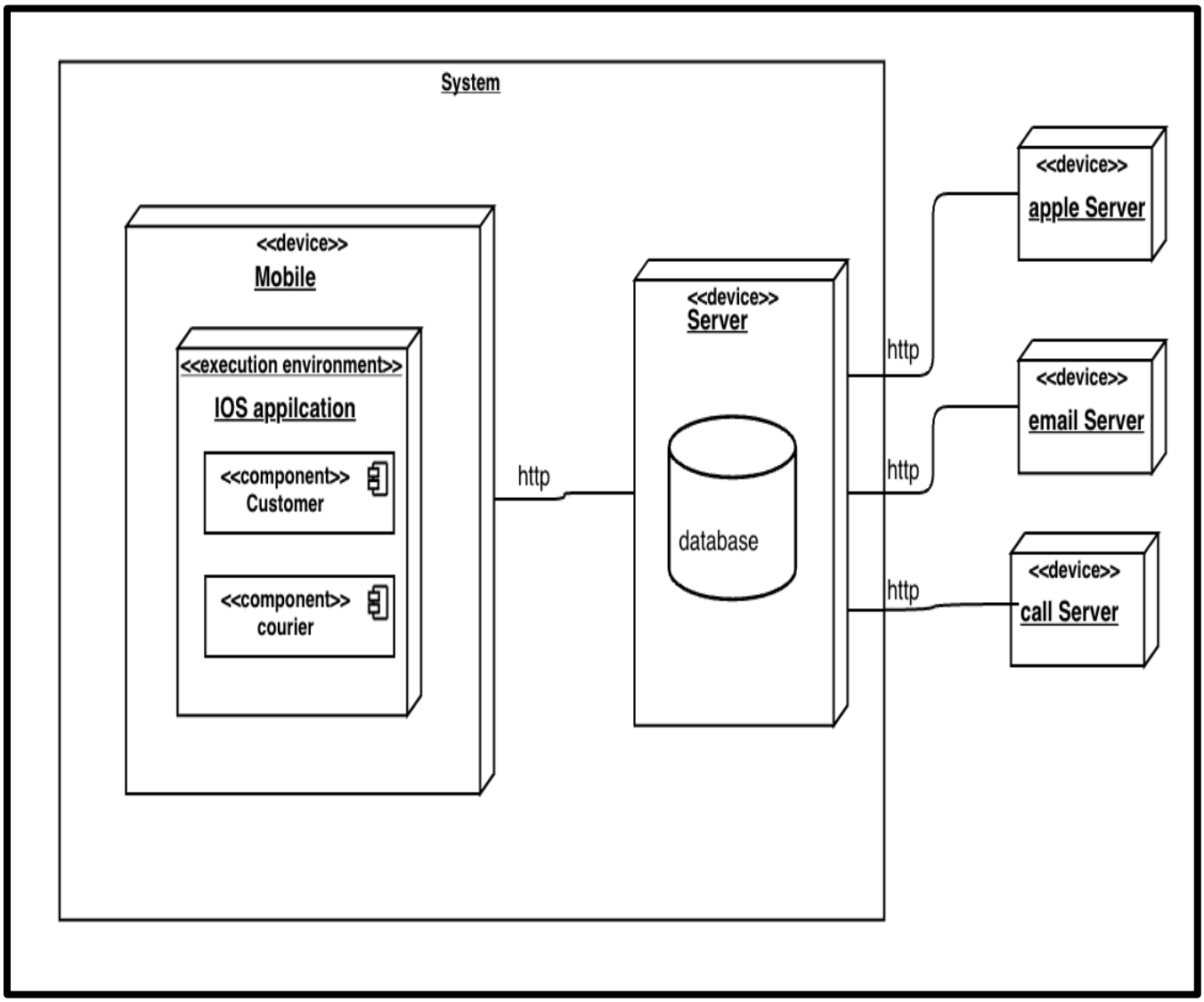


Figure 53 deployment diagram

17.Test Scenario

Test Case ID: TC1			
Test Designed by: Lama AlSanie			
Test Designed date: 25/11/2021			
Test Title: Make an order			
Description: This test case aims to test how a customer makes an order.			
Pre-Condition: Customer Logged-in successfully.			
Dependencies: TC(Log in)			
Steps	**Test steps**	**Test data**	**Expected results**
1	Customer selects the restaurant he/she wants.	Dunkin Donuts	System will view the restaurant page.
2	Customer selects “make an order”	-	System will view the “make an order” form.
3	Customer fills the form (Description of the order, location).	Black coffee medium size, “get location”.	Form is filled
4	Customer clicks confirm	-	System accepted order.

Table 14 Make an order test scenario

Test Case ID: TC2			
Test Designed by: Lama AlSanie			
Test Designed date: 25/11/2021			
Test Title: Make an order with empty fields.			
Description: This test case aims to test how a customer makes an order with empty fields.			
Pre-Condition: Customer Logged-in successfully.			
Dependencies: TC (Log in).			
Steps	**Test steps**	**Test data**	**Expected results**

1	Customer selects the restaurant he/she wants.	Dunkin Donuts	System will view the restaurant page.
2	Customer selects “make an order”	-	System will view “make an order” form.
3	Customer clicks “confirm”.	-	Error message “form cannot be empty”

Table 15 Make an order with empty fields test scenario

Test Case ID: TC3			
Test Designed by: Alanoud Alsuwailem			
Test Designed date: 25/11/2021			
Test Title: Pay an order.			
Description: This test case aims to test how a customer pays his/her order.			
Pre-Condition: Customer Logged-in successfully.			
Dependencies: TC (Log in), TC (Make an order), TC (Export bill).			
Steps	**Test steps**	**Test data**	**Expected results**
1	Customer selects “Pay order”.	-	System will view the “pay order” page.
2	Customer selects payment method.	-	System will view payment method options.
3	Customer clicks suitable payment method.	-	System accepted payment method.

Table 16 Pay an order test scenario

Test Case ID: TC4			
Test Designed by: Aljoury bin osseil			
Test Designed date: 25/11/2021			
Test Title: Select an order.			
Description: This test case aims to test how a courier selects an order.			
Pre-Condition: Courier Logged-in successfully.			
Dependencies: TC (Log in)			
Steps	**Test steps**	**Test data**	**Expected results**
1	Courier selects “all orders”	-	System will view all orders.
2	Courier selects the order to deliver.	-	System will view the chat page with the customer.

Table 17 Select an order test scenario

Test Case ID: TC5			
Test Designed by: Fay Almutairi			
Test Designed date: 04/12/2021			
Test Title: View an active order.			
Description: This test case aims to test how a customer view an active order.			
Pre-Condition: Customer Logged-in successfully.			
Dependencies: TC (Log in), TC (Make an order).			
Steps	**Test steps**	**Test data**	**Expected results**
1	Customer click on “Active Order” tap locate on homepage.	-	System will view the “Active order” page.

2	Select one of the active orders.	Chat with the courier.	The system will the chat of the order she/he wants.

Table 18 View an active order test scenario

Test Case ID: TC6			
Test Designed by: Fay Almutairi			
Test Designed date: 04/12/2021			
Test Title: Test choose courier request			
Description: This test case aims to test how a customer view courier request in order.			
Pre-Condition: Customer Logged-in successfully.			
Dependencies: TC (Log in), TC (Make an order).			
Steps	**Test steps**	**Test data**	**Expected results**
1	-	**Name**: Fay Almutairi **Price:**25 SR.	System will view all requested courier.
2	Customer selects the courier he/she wants.	**Name:** Fay Almutairi **Price:**25 SR call button.	System will view the conformation massage.

Table 19 choose courier request test scenario

Test Case ID: TC7			
Test Designed by: Reem aldosari			
Test Designed date: 5/12/2021			
Test Title: Register			
Description: This test case aims to test how a customer Register.			
Pre-Condition: None.			
Dependencies: None.			
Steps	**Test steps**	**Test data**	**Expected results**

1	Customer selects “Register”	-	System will view the “Register” form.
2	Customer fills the form with his/her information including (name, email, phone number, password).	Amal, amal@gmail.com,0559837483, Female, amal12345	Form is filled.
3	Customer clicks “Register”.	-	System create new account.

Table 20 Register test scenario

Test Case ID: TC8			
Test Designed by: Alanoud Alsuwailem			
Test Designed date: 11/12/2021			
Test Title: Rating an order			
Description: This test case aims to show how the customer rates an order			
Pre-Condition: 1- Customer Logged-in successfully. **2-** Type his/her order and submit it successfully			
Dependencies: TC (Log in), TC (Make an order).			
Steps	**Test steps**	**Test data**	**Expected results**
1	Customer paid for the order successfully.	-	The system displays the pop-up screen for the rating.
2	Customer reviews the service with the desired rate using a five-star rating.	**Rate:** 3 stars	The system will fill the stars.

3	Customer submits the rating.	**Rate:** 3 stars	The system displays feedback submitted successfully message.

Table 21 Rating order scenario

Test Case ID: TC9			
Test Designed by: Aljoury bin osseil			
Test Designed date: 11/12/2021			
Test Title: View history order			
Description: this test case aims to test how the courier can view his/her history orders.			
Pre-Condition: Customer Logged-in successfully.			
Dependencies: TC (Log in), TC (Make an order).			
Steps	**Test steps**	**Test data**	**Expected results**
1	The courier selects the “history order” tab.	-	The system displays all past orders.
2	The courier selects the order he/she wants.	Starbucks	The system will display the details of the desired order (name of the customer, phone number, date, time, price, location).

Table 22 View history scenario

Test Case ID: TC10			
Test Designed by: Reem Aldosari			
Test Designed date: 12/12/2021			
Test Title: update account.			
Description: This test case aims to test how a customer update her account.			
Pre-Condition: 1- Customer Logged-in successfully. **2-**Customer views the profile.			
Dependencies: TC (log in).			
Steps	**Test steps**	**Test data**	**Expected results**
1	Customer selects “Profile” icon.	Lena Alsubaie, lena@gmail.com,0559384799	System will view the “Profile” form.
2	Customer fills the form with her information including (name, email, phone number).	Sara Al Qahtani, sara@gmail.com,0559384756	Form is filled.
3	Customer clicks “Save” button.	-	System updates the profile.

Table 23 update account scenari

18. Project Status

in the KSU Delivery Service we will talk about the project statues and the problems we faced first our project is following the waterfall method it includes five phases we will describe each phase and how the team solved the problems and helped each other:

18.1 Project Proposal

The work for the Project Proposal started from (2021-9-2) and ended on (2021-9-21) the team faced some difficulties in the risk/constraints since we should include all of the risks that the project will face, the team delivered the Project Proposal including (scope, technology, domain analysis, introduction, risk/constraints, project plan, quality assurance plan (QAP)) on time.

18.2 Requirement analysis

The work for the Requirement analysis started from(2021-9-27)and ended on (2021-10-25) the team faced some difficulties in the sequence diagram as the resources was insufficient since the sequence diagram was different from what we learned, we solved it by returning to our instructor and searching in depth about it, the team delivered the Requirement analysis including(functional, nonfunctional, design constraint, system boundary, problem complexity, use case diagram, use case description, analysis class(VOPC), interaction diagram(Sequence)) on time.

18.3 Design

The work for the Design started from (2021-11-1) and ended on (2021-12-12) the team faced some difficulties in the algorithm since it was written in a pseudo code, the team delivered the Design including (user interface mockup, system architecture, design class, database, expected deployment, algorithm, test scenario, project statues) on time.

18.4 Implementation

The team will code the project based on the previous phases (Project Proposal, Requirement analysis, Design). This phase will be implemented in the second semester for the academic year (2021-2022).

18.5 Testing

The team will do several testing methods to assure the quality of the project. This phase will be implemented in the second semester for the academic year (2021-2022).

19. Prototype Description

This section describes the prototype of the KSU Delivery System. Depending on the roles played by the user either Customer or Courier, the system's interfaces and functionalities will vary.

19.1 Implementation Platform

The list below shows the hardware / software technologies that were used for the development of KSUDS Application:

- Visual Studio Code
- Firebase Database
- Flutter
- dart
- Trello
- Blackboard
- GitHub
- Hardware components
- IOS 12 simulator
- XCode
- Android Studio

19.2 Mapping Between Requirement and Implementation Functions

The following table shows a mapping between the projected functionalities of the system and the corresponding implemented modules.

19.2.1 Functional Requirements of Customer:

Functional requirements	Modules/Functions/Class that implemented this feature
6.3.1 The customer shall be able to register by entering username, email, phone number, and password.	**Classes:** -auth_view.dart - auth_vm.dart **Functions:** -registerUser
6.3.2 The customer shall be able to log in by entering username and password.	**Classes:** -auth_view.dart -auth_vm.dart **Functions:** -SignInNow

	-handleSignIn
6.3.3 The customer shall be able to log out from the application.	**Classes:** -Base_view.dart -signout_sheet.dart **Functions:** -signOut
6.3.4 The customer shall be able to view his/her information profile.	**Classes:** -profile_view.dart -base_view.dart
6.3.5 The customer shall be able to edit his/her account information (First name, Last name, Picture, Email, Phone number).	**Classes:** -profile_view.dart -validator.dart **Function:** -validateName -validatePhoneNumber
6.3.6 The customer shall be able to view the courier profile (Name,Email,picture,Phone number, Ratings,total orders).	**Classes:** Other_user_profile.dart
6.3.7 The customer shall be able to make an order by writing a description of his/her order.	**Classes:** -type_order_view.dart **Function:** Checkempty-order
6.3.8 The customer shall be able to locate his/her location.	**Classes:** -location_view.dart -location_viewmodal.dart **Function:** update
6.3.9 The customer shall be able to track his/her order by receiving the status of the order (Going to pickup,Picked food, Arrived at drop location, delivered).	**Classes:** -Order_details_page.dart **Function:** -_goStore -goToPickup -pickOrder -deliverOrder

6.3.10 The customer shall be able to rate the courier.	**Classes:** -Rate_user.dart **Functions:** -rateNow -RateUser
6.3.11 The customer shall be able to pay his/her order.	**Classes:** -Pay_pal_payment.dart –pay_pal.dart -order.dart -order_vm.dart -chat_view.dart - **Functions:** **-p**ayNow
6.3.12 The customer shall be able to view all delivery offers received.	**Classes:** -View_offers.dart
6.3.13 The customer shall be able to choose a suitable delivery offer.	**Classes:** -View_offers.dart **Functions:** -AcceptRejectSheet - accceptOffer -rejectOffer
6.3.14 The customer shall be able to contact the courier via call.	Classes: -Order_details_page.dart -chat_view.dart Functions: -_makePhoneCall
6.3.15 The customer shall be able to contact the courier via chat.	**Classes:** -Chat_room_model.dart -chat_service.dart -chat_view.dart -chat_vm.dart **Function:** -ChatView
6.3.16 The customer shall be able to view his/her active order.	**Classes:** -Order_view.dart
6.3.17 The customer shall be able to view his/her orders history.	**Classes:** -history_view.dart **Functions:**

	-HistoryView
6.3.17.1 The customer shall be able to view his/her orders history details (status, courier name , order id, total bill, order details, date , time, payment status).	**Classes:** -order_details_page.dart
6.3.18 The customer shall be able to report an issue in the order.	**Classes:** -Report_issue.dart
6.3.19 The customer shall be able to browse through all available shops on campus.	**Classes:** -home_view.dart -base_view.dart
6.3.20 The customer shall be able to view points earned from ordering.	**Classes:** -profile_view.dart
6.3.21The customer shall be able to cancel an order delivery.	**Classes:** Order_details_page.dart Order _widget. dart **Function:** -cancelOrderByCustomer

Table 24 mapping Requirement for the customer function

19.2.2 Functional Requirements of courier:

Functional requirements	Modules/Functions/Class that implemented this feature
6.3.22 The courier shall be able to register by entering username, email, number phone, and password.	**Classes:** -auth_view.dart -auth_vm.dart **Functions:** -registerUser
6.3.23 The courier shall be able to log in by entering username and password.	**Classes:** -auth_view.dart -auth_vm.dart **Functions:** -SignInNow -handleSignIn

6.3.24 The courier shall be able to log out from the application.	**Classes:** -Base_view.dart -signout_sheet.dart **Functions:** -signOut
6.3.25 The courier shall be able to view active orders.	**Classes:** -Order_view.dart
6.3.26 The courier shall be able to view his/her account information.	**Classes:** -profile_view.dart -base_view.dart
6.3.27 The courier shall be able to edit his/her account information (First name, Last name, Picture, Email, Phone number).	**Classes:** -profile_view.dart -validator.dart **Function:** -validateName -validatePhoneNumber
6.3.28 The courier shall be able to view orders requests.	**Classes:** -Order_view.dart
6.3.29 The courier shall be able to select an order.	**Classes:** -Order_view.dart
6.3.29.1 The courier shall be able to view the order's information.	**Classes:** -Order_details_page.dart
6.3.29.2 The courier shall be able to cancel an order delivery.	**Classes:** Order_details_page.dart Order_widget. dart **Function:** -cancelOrder -cancel
6.3.29.3 The courier shall be able to send the offer.	**Classes:** -Send_offer_dialog.dart **Function:** -sendOffer

6.3.30 The courier shall be able to contact the customer via call.	**Classes:** -Order_details_page.dart - chat_view.dart **Functions:** -_makePhoneCall
6.3.31 The courier shall be able to contact the customer via chat.	**Classes:** -Chat_room_model.dart - chat_service.dart -chat_view.dart -chat_vm.dart **Function:** -ChatView
6.3.32 The courier shall be able to view the customer profile (Name,Email,Phone number, Ratings,total orders).	**Classes:** Other_user_profile.dart
6.3.33 The courier shall be able to report an issue in the order.	**Classes:** -Report_issue.dart **Function:** -sendEmail
6.3.34 The courier shall be able to rate the customer.	**Classes:** -Order_details_page.dart **Function:** -rateNow -RateUser
6.3.35 The courier shall be able to view his/her orders history.	**Classes:** -Courier_history _view.dart
6.3.35.1 The courier shall be able to view his/her orders history details (date, time, order description , name of the customer, location, total bill, order id , payment status ,chat, call).	**Classes:** -Courier_history _view.dart **Function:** -makePhoneCall -ChatView

6.3.36 The courier shall be able to view his/her rating.	**Classes:** -Profile_view.dart
6.3.37 The courier shall be able to view his/her earnings from the orders he/she deliver.	**Classes:** -Profile_view.dart
6.3.38 The courier shall be able to view the location of the place he/she will pick the order from.	**Classes:** -Order_details_page.dart -order_service **Function:** -customerOrders -courierOrders
6.3.39 The courier shall be able to view the location of the place he/she will deliver the order to.	**Classes:** -Order_details_page.dart -order_service **Function:** -customerOrders -courierOrders
6.3.40 The courier shall be able to change order status (Going for pickup, Picked Food, Arrived at drop location , Delivered).	**Classes:** -Order_details_page.dart **Function:** -goStore -goToPickup -pickOrder -deliverOrder
6.3.41 The courier shall be able to view the ID for the order.	**Classes:** -Order_details_page.dart
6.3.42 The courier shall be able to export the bill by the chat when he/she pick-up the order.	**Classes:** -Export_bill.dart **Function:** -exportBill

6.3.43 The courier shall be able to access the camera to upload the bill picture.	**Classes:** -Export_bill.dart **Function:** -getImage -setState

Table 25 mapping Requirement for the courier function

19.2.3 Functional Requirements of System:

Functional requirements	Modules/Functions/Class that implemented this feature
6.3.44 The system shall be able to calculate the average rating for the customer or the courier.	**Classes:** -Profile_view.dart -Other_user_profile.dart
6.3.45 The system shall be able to calculate the total price including the delivery of the order.	**Classes:** -Export_bill.dart
6.3.46 The system shall be able to generate ID for the order.	**Classes:** -Order_vm.dart **Function:** -Paypalpayment
6.3.47 The system shall be able to show a confirmation payment message to the customer after paying.	**Classes:** -Payment_sucessful.dart
6.3.48 The system shall be able to validate the payment information before accepting it.	**Classes:** -Pay_pal.dart **Function:** -getAccessToken

6.3.49 The system shall be able to notify the receiver when the sender sends a message.	**Classes:** -app_notification.dart -Notification_page.dart -order_details_page.dart -enums.dart
6.3.50 The system shall be to calculate points earned after each order to get a discount.	**Classes:** -Profile_view.dart **Function:** -_convertToLatLng
6.3.51 The system shall be able to notify the customer when order arrives.	**Classes:** -app_notification.dart -Notification_page.dart -order_details_page.dart -enums.dart **Function:** -reachDropLocation
6.3.52 The system shall be able to notify the customer/courier when the customer/courier canceled the order.	**Classes:** -app_notification.dart -Notification_page.dart -enums.dart **Function:** -cancelOrder

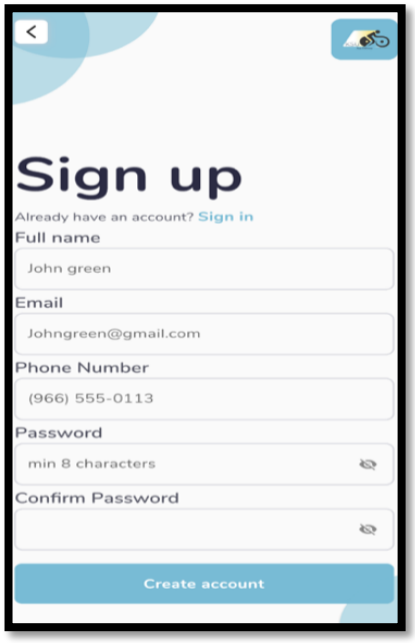
Table 26 mapping Requirement for system function

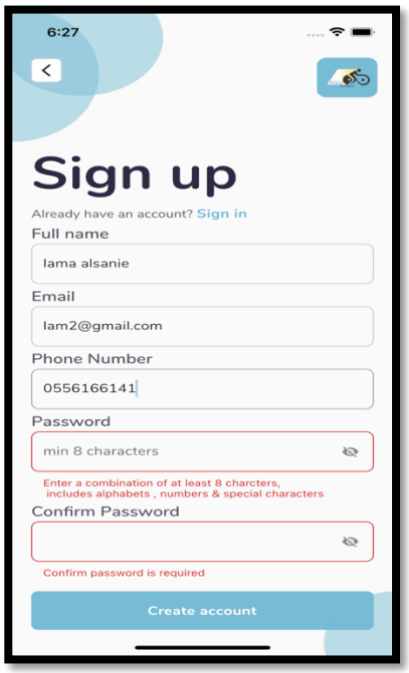
19.3 Implementation Details

This section shows the code segment of the most critical functions of the system. Functions include Registration, for a customer select a restaurant , make an order process ,select an offer and reject offer. For the courier select an order request, export bill, and change order status.

19.3.1 Registration Authentication:

Step	Code Segment	Associated interface

To make sure of the account information when the user enters them when he wants to register	class FieldValidator {
	static String? validateEmail(String value) {
	if (value.isEmpty) {
	return 'Please Enter email address';
	}
	if (!RegExp(
	r"^[a-zA-Z0-9.a-zA-Z0-9.!#$%&'*+-/=?^_`{|}~]+@[a-zA-Z0-9]+\.[a-zA-Z]+")
	.hasMatch(value)) {
	return 'Invalid Email';
	}
	if (value.contains(" ")) {
	return 'Invalid Email';
	}
	return null;
	}
		

The ValidatePassword method will make sure of the format of the password and shows a massage if the inserted password is wrong.	static String? validatePassword(String value) {
	if (value.isEmpty) return 'Please enter password';
	if (value.length < 6) {
	return "Enter a combination of at least 6 numbers, \nalphabets & special characters";
	}
	if (!RegExp(r"^(?=.*?[0-9])").hasMatch(value)) {
	return 'Enter a combination of at least 6 numbers, \nalphabets & special characters';
	}
	if (!RegExp(r'^(?=.?[!@#\$&~])').hasMatch(value.trim())) {
		

	return 'Enter a combination of at least 6 numbers, \nalphabets & special characters';
	}
	return null;
	}
	
	static String? validatePasswordMatch(String value, String pass2) {
	if (value.isEmpty) return "Confirm password is required";
	if (value != pass2) {
	return 'Password didn\'t match';
	}
	return null;
	}	

The validateName method will make sure of the format of the name and shows a massage if the inserted name is wrong.	static String? validateName(String value) {
	if (value.isEmpty) {
	return 'Please enter your full name';
	}
	
	if (!RegExp(r"^[A-Z a-z]{2,25}$").hasMatch(value)) {
	return 'Incorrect full name';
	}
	
	return null;
	}	

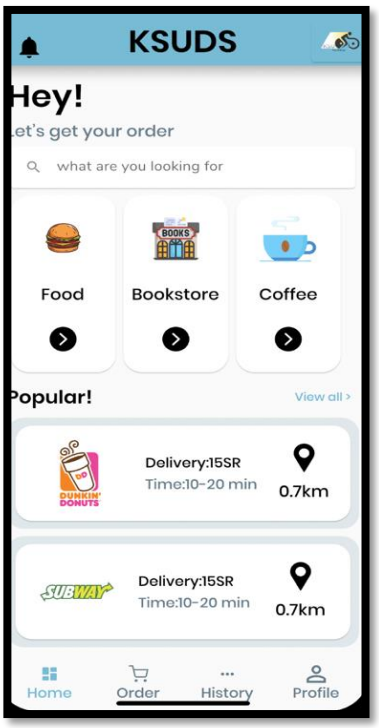
The validateConfirmPassword method will make sure of the format of the password and shows a massage if the inserted password is wrong.	static String? validateConfirmPassword(String value) {
	if (value.isEmpty) return "Confirm password is required";
	if (value.length < 6) {
	return "Enter a combination of at least 6 numbers, \nalphabets & special characters";
	}	

	if (!RegExp(r"^(?=.*?[0-9])").hasMatch(value)) {
	return 'Enter a combination of at least 6 numbers, \nalphabets & special characters';
	}
	if (!RegExp(r'^(?=.?[!@#\$&~])').hasMatch(value.trim())) {
	return 'Enter a combination of at least 6 numbers, \nalphabets & special characters';
	}
	return null;
	}
	
	static String? validatePhoneNumber(String value) {
	print("validatepassword : $value ");
	
	if (value.isEmpty) return "Please enter your phone number";
	if (value.length <= 8) {
	return "Invalid number";
	}
	
	String pattern = r'(^(?:[+0]9)?[0-9]{9,15}$)';
	RegExp regex = new RegExp(pattern);
	if (!regex.hasMatch(value.trim())) {
	return "Invalid Number";
	}
	return null; }}
		The validatePhoneNumber method will make sure of the format of the phone number and shows a massage if the inserted phone number is wrong.

Table 27 Implementation detail for Registration Authentication

19.3.2 Select restaurant:

Step	Code Segment	Associated interface

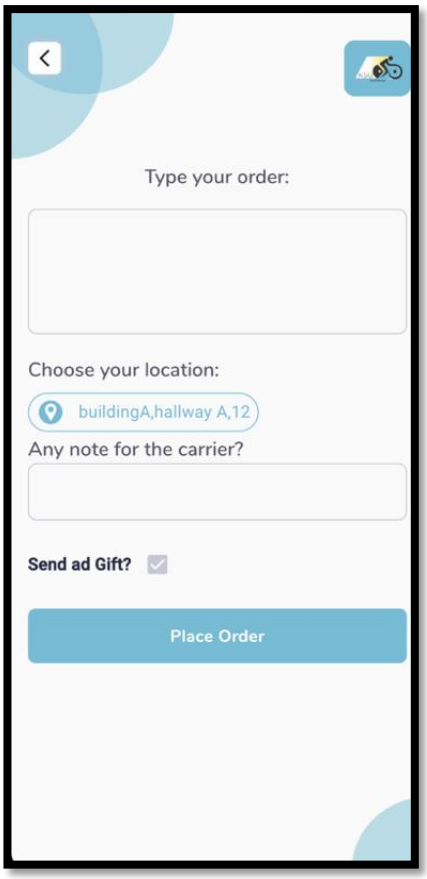
The customer will have variety of ways to select a restaurant he can view them according to the category using itemBuilder	class HomeView extends StatefulWidget {static String route = '/HomeView';
	const HomeView({Key? key}) : super(key: key);
	@override
	State<HomeView> createState() => _HomeViewState();}
	Widget build(BuildContext context) {
	return Consumer<HomeVM>(builder: (context, model, _) {
	return SingleChildScrollView(
	child:Padding(padding:EdgeInsets.symmetric(vertical:1.h, horizontal: 3.w),
	child:Column(crossAxisAlignment: CrossAxisAlignment.start,
	children: [Text("Hey!",)
	style: R.textStyles.poppinsHeading1(),),
	Text("Let's get your order",
	style: R.textStyles.poppinsTitle1()
	.copyWith(color:
	const Color(0xff667C8A)),
	Card(child:TextFormField(decoration: R.decorations.appFieldDecoration2(
	"what are you looking for",),
	controller: searchTC,
	keyboardType: TextInputType.name,
	textInputAction: TextInputAction.next,
	style: R.textStyles.textFormFieldStyle(),
	),),
	SizedBox(height: 25.h,
	child: ListView.builder(
	scrollDirection: Axis.horizontal,
	itemCount: model.categories.length,
	itemBuilder:(context, index) {
	Categorycategory= model.categories[index];
	return CategoryWidget(
	category: category,);}),),
	Row(mainAxisAlignment: MainAxisAlignment.spaceBetween,
	children: [Text("Popular!",
	style: R.textStyles.poppinsHeading1()
	.copyWith(fontSize: 16.sp),),
	TextButton(
	onPressed: () {},
	child: Text("View all >",
	style: R.textStyles.poppinsTitle1().copyWith(fontSize: 10.sp, color: R.colors.theme),),),],),
	GestureDetector(
		
		 

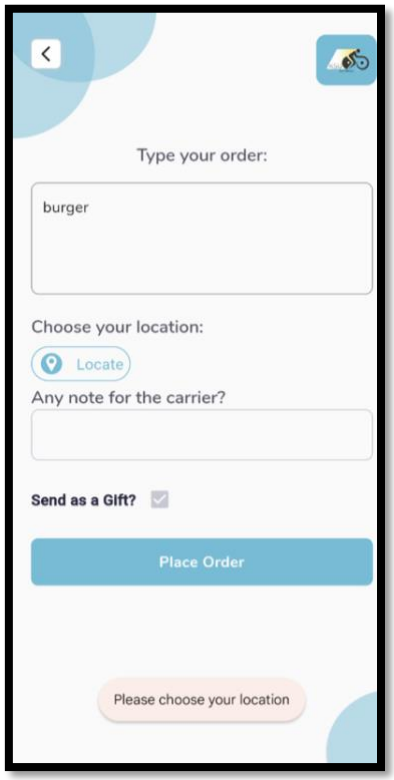
Or select a restaurant from the view all restaurants and what ever the user taps it will route him/her to the menu of the selected store .	onTap: () {
	_roomProvider.selectedStore=0;
	Get.toNamed(DunkinView.route);},
	child: ProductWidget(
	image: R.images.cup,),),
	R.sizedBox.sizedBox1h(),
	GestureDetector(
	onTap: () {
	_roomProvider.selectedStore=1;
	Get.toNamed(SubwayView.route);},
	child: ProductWidget(
	image: R.images.subway,
	),),],),),);});}}	

Table 28 Implementation detail for select a restaurant

19.3.3 Make an order:

Step	Code Segment	Associated interface

First the customer needs to fill his/her order description then the controller will validate the field if its empty with checkEmpty method .
 	
The customer will have to press the location button then Get.toNamed() it will route the customer to locate his/her location and then validate the field if its empty.
	classTypeOrder extends StatefulWidget{
	static String route = '/TypeOrderView';
	const TypeOrder({Key? key}) : super(key: key);
	@override
	State<TypeOrder> createState() => _TypeOrderState();}
	
	Padding(padding: EdgeInsets.symmetric(horizontal: 5.w),
	child:Column(crossAxisAlignment: CrossAxisAlignment.start,
	children: [Center(child: const Heading(title: 'Type your order:',),),
	R.sizedBox.sizedBox2h(),
	TextFormField(decoration: R.decorations.appFieldDecoration(null, "", "), controller: orderTC,
	validator:(value)=> FieldValidator.checkEmpty(orderTC.text),
	keyboardType: TextInputType.text,
	textInputAction:TextInputAction.done,
	maxLines: 4,
	minLines: 4,
	style: R.textStyles.textFormFieldStyle(),),
	R.sizedBox.sizedBox2h(),
	const Heading(title: 'Choose your location:',),
	R.sizedBox.sizedBox1h(),
	InkWell(onTap:(){ Get.toNamed(LocationView.route); },
	child: Container(child: Row(
	mainAxisSize: MainAxisSize.min,
	children:[Container(padding: EdgeInsets.all(3), child: Image.asset(
	R.images.pin,
	height: 16,
	color: R.colors.white),
	decoration:BoxDecoration(shape: BoxShape.circle,
	color: R.colors.theme),),
	R.sizedBox.sizedBox3w(), Text(roomProvider.selectedRoom.isEmpty
		

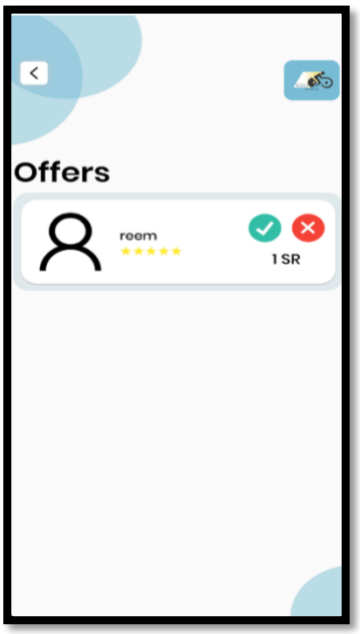
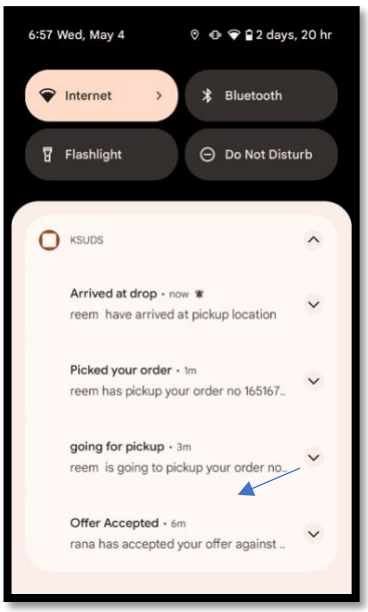
Afterward the courier will have the option to fill if he/she has any note for the courier and weather the purchase is a gift.	?"Locate": "${roomProvider.selectedBuilding},hallway ${roomProvider.selectedHallway},${roomProvider.selectedRoom[0].label}", style: R.textStyles .robotoRegular() .copyWith(color: R.colors.theme),),],), padding: EdgeInsets.all(6), decoration: BoxDecoration(borderRadius: BorderRadius.circular(50), border:Border.all(color: R.colors.theme)),),), const Heading(title: 'Any note for the courier? ',), TextFormField(decoration: R.decorations.appFieldDecoration(null, "", ' '), controller: noteTC, keyboardType: TextInputType.text, textInputAction: TextInputAction.done, style:R.textStyles.textFormFieldStyle(),) R.sizedBox.sizedBox2h(), Row(children: [Text("Send as a Gift?", style: R.textStyles.robotoMedium().copyWith(fontWeight: FontWeight.bold, color: R.colors.themeBlack),), IconButton(onPressed: () {setState(() { gift = !gift;}); }, icon:Icon(gift?Icons.check_box_outline_blank_rounded : Icons.check_box_rounded, color: R.colors.grey,),)],), R.sizedBox.sizedBox2h(), MyButton(onTap: () async { if (formKey.currentState!.validate()) { if(roomProvider.selectedRoom.isEmpty) {ShowMessage.toast('Please choose your location');}else {	
Finally Place order method placeOrder will place the order and send the information couriers side after pressing place order button	vm.placeOrder(order: Order(status: 0, storeId: roomProvider.selectedStore, createdAt: Timestamp.now(), id:Timestamp.now() .millisecondsSinceEpoch .toString(), requestExpiry: Timestamp.fromDate(DateTime(DateTime.now().year, DateTime.now().month, DateTime.now().day+1, DateTime.now().hour, DateTime.now().minute,	

	DateTime.now().second,)).millisecondsSinceEpoch, orderDetail: orderTC.text, note: noteTC.text, courierId: '', pickLat: rest.lat, pickLng:rest.long,customerId: authVm.appUser?.email, dropLat: roomProvider.selectedRoom[0].lat, dropLng: roomProvider.selectedRoom[0].long, pickAddress: roomProvider.selectedStore == 0 ? 'Dunkin Donuts''Subway', dropAddress: '${roomProvider.selectedBuilding},hall way ${roomProvider.selectedHallway},${roo mProvider.selectedRoom[0].label}', price: '')); } }}, buttonText: 'Place Order',)]),)],),),),	

Table 29 Implementation detail for make an order

19.3.4 Select an offer:

Step	Code Segment	Associated interface

The acceptOffer method ,if the customer selected an offer and accepted based on the price the courier offered and his information , acceptOffer() will notify the courier and the chat will initiate between them	Future<void> acceptOffer(
	DocumentSnapshot offerDoc,
	BaseVM baseVM,
	) async {
	setState(() {
	isLoading = true;
	});
	await FBCollections.orders.doc(widget.orderId).update({
	'status': 1,
	'deliveryPrice': offerDoc['price'],
	'courierId': offerDoc['courierId'],
	});
	AppNotification notify = AppNotification(
	title: "Offer Accepted",
	message: "${authVm.appUser?.fullName} has accepted your offer against order no ${widget.orderId}",
	isSeen: false,
	notificationType: NotificationType.acceptedOffer,
	sender: authVm.appUser?.email,
	receiver: offerDoc['courierId'],
	createdAt: Timestamp.now());
	await fbNotification.notifyUser(notify, sendPush: true, sendInApp: true);
	await FBCollections.users.doc(offerDoc['courierId']).update({
	'onRide': true,
	});
	await InitiateChat(peerId: offerDoc['courierId'], roomId: widget.orderId)
	.now()
	.then((value) async {
	log("my room id ${value.roomId}");
	setState(() {
	isLoading = false;
	});
	MessageModel msg = MessageModel(
		
		
		 

	message: offerDoc['orderDetail'], receiver: offerDoc['courierId'], sender: offerDoc['customerId'], roomId: offerDoc['orderId'], createdAt: Timestamp.now(), isSeen: false,); Provider.of<ChatVM>(Get.context!, listen: false).sendMessage(msg,); await Get.to(() => ChatView(isFirstTime: true, price: offerDoc['price'], peerId: offerDoc['courierId'], orderId: widget.orderId,),);}); }	

Table 30 Implementation detail for select an offer

19.3.5 Reject an offer:

Step	Code Segment	Associated interface

The rejectOffer method ,the customer can reject an offer and a notifier will appear to the courier of the rejection	Future<void> rejectOffer(
	DocumentSnapshot offerDoc,
	String offerId,
	) async {
	setState(() {isLoading = true});
	await FBCollections.offers.doc(offerId).update({ 'status': 2});
	AppNotification notify= AppNotification(title: "Offer Rejected", message: "${authVm.appUser?.fullName} has rejected your offer against order no ${widget.orderId}", isSeen: false,
	notificationType: NotificationType.acceptedOffer,
	sender: authVm.appUser?.email,
	receiver: offerDoc['courierId'],
	createdAt: Timestamp.now());
	await fbNotification.notifyUser(notify, sendPush: true, sendInApp: true);
	setState(() { isLoading = false; }); }
		

Table 31 Implementation detail for Reject an order

19.3.6 Select order request:

Step	Code Segment	Associated interface
The courier can browse through order requests which contains information about the order so he can select the desired option. If the courier selected an offer and he/she received an acceptance from the customer the chat between them will initiate.	Row(crossAxisAlignment: CrossAxisAlignment.start, mainAxisAlignment: MainAxisAlignment.spaceBetween, children: [Text('Order ID:', style: R.textStyles.poppinsTitle1().copyWith(fontSize: 12.sp),), Text('#${widget.order.id}', style: R.textStyles.poppinsTitle1().copyWith(fontSize:12.sp,color:const Color(0xff667C8A)),),],), R.sizedBox.sizedBox1h(), Row(mainAxisAlignment: MainAxisAlignment.spaceBetween, children: [Text(widget.order.pickAddress ?? "", style: R.textStyles.robotoMedium(),), if (widget.order.status > 1) IconButton(onPressed: () {Get.to(() => ChatView(orderId: widget.order.id!, peerId: widget.order.customerId, firstMessage: widget.order.orderDetail,),); }, icon:Icon(Icons.chat_outlined,size:20.sp, color: R.colors.darkGrey,),)else SizedBox(),],)],), collapsed: SizedBox(), expanded: Column(crossAxisAlignment: CrossAxisAlignment.start, children: [R.sizedBox.sizedBox1h(), Text("Deliver to", style: R.textStyles.poppinsHeading1().copyWith(fontSize: 12.sp),), Text(widget.order.dropAddress ?? "", style: R.textStyles.robotoMedium(),), R.sizedBox.sizedBox1h(), Row(crossAxisAlignment: CrossAxisAlignment.start, mainAxisAlignment: MainAxisAlignment.spaceBetween, children:[Row(crossAxisAlignment: CrossAxisAlignment.center, children: [Image.asset(	

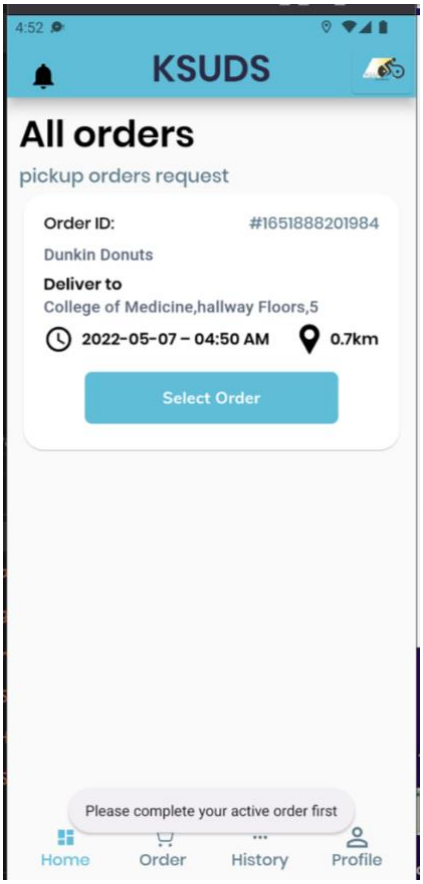
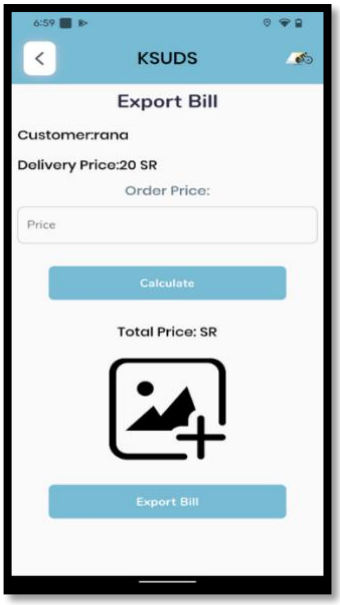
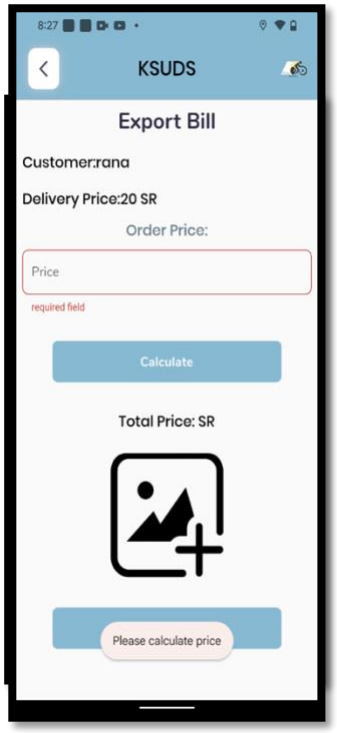
Streambuilder will make sure that the courier doesn't have an active order, if he/she does there will be a notifier that requests him/her to complete active order first else it will accept it.	R.images.accessTime,height: 22.sp), R.sizedBox.sizedBox2w(), Text(DateFormat('yyyy-MM-dd – hh:mm a') .format((widget.order.createdAt)!.toDate ()), style: R.textStyles .poppinsHeading1() .copyWith(fontSize: 12.sp),),],), Row(crossAxisAlignment: CrossAxisAlignment.center, children: [Image.asset(R.images.pin, height: 22.sp,) ,Text('0.7km', style: R.textStyles.poppinsHeading1() .copyWith(fontSize: 12.sp),),],)],), R.sizedBox.sizedBox1h(), StreamBuilder(stream: FBCollections.offers.where("orderId", isEqualTo: widget.order.id) .where("courierId", isEqualTo: authVm.appUser?.email) .snapshots(), builder:(context, AsyncSnapshot<QuerySnapshot> snapshot) { if (!snapshot.hasData) { return MyLoader(); } else if (snapshot.data!.docs.isEmpty) { return Padding(padding: EdgeInsets.symmetric(horizontal: 10.w, vertical: 1.h), child: MyButton(onTap: () { if (authVm.appUser?.onRide == false) { Get.dialog(SendOfferDialog(order: widget.order,));} else { ShowMessage.toast('Please complete your active order first'); }},buttonText: "Select Order"),);} else { returnPadding(padding: EdgeInsets.symmetric(horizontal:10.w, vertical: 1.h), child:Text("OfferSent: ${snapshot.data?.docs[0]['price']}", style: R.textStyles .poppinsTitle1() .copyWith(color:const Color(0xff667C8A)),),); }}],)),));}}	

Table 31 Implementation Reject an offer

19.3.7 Export bill:

Step	Code Segment	Associated interface
The courier can export the bill after picking up the order by adding the price and importing a picture of the bill The courier can view the information previously added like the delivery price as a recall.	body:Padding)padding:const EdgeInsets.all(8.0), child: Form(key: formKey, autovalidateMode: AutovalidateMode.disabled, child: ListView(children: [Center(child: Text("Export Bill", style: MyTextStyle().heading1(),),), R.sizedBox.sizedBox2h(), Text("Customer:${widget.name}", style: MyTextStyle().poppinsTitle1(),), R.sizedBox.sizedBox2h(), Text("Delivery Price:${widget.order.deliveryPrice}SR", style: MyTextStyle().poppinsTitle1(),), Padding(padding:const EdgeInsets.symmetric(vertical: 10), child: Center(child:Text("Order Price:",style:R.textStyles.poppinsTitle1().copyWith(color:const Color(0xff667C8A)),),),), TextFormField(decoration: R.decorations.appFieldDecoration(null, "price", 'Price'),controller: priceTC, keyboardType: TextInputType.number, textInputAction: TextInputAction.done,	 

The validator will check and make sure the price is added. The total price of the order and the delivery will be calculated through pressing the button. The courier must insert an image getimage() will be used to validate and return weather the user inserted the image or not and it will show an error massage.	style: R.textStyles.textFormFieldStyle(), validator:(value)=> FieldValidator.checkEmpty(priceTC.text),), R.sizedBox.sizedBox3h(),Padding(padding: EdgeInsets.symmetric(horizontal: 10.w, vertical: 1.h),child: MyButton(onTap:(){if (formKey.currentState!.validate()) { widget.order.totalPrice= (double.parse(widget.order.deliveryPrice!)+double.parse(priceTC.text)).toString(); widget.order.price= priceTC.text;setState(() {}); } }, buttonText: "Calculate"),), R.sizedBox.sizedBox3h(), Center(child: Text("Total Price:${widget.order.totalPrice ?? ""} SR", style: MyTextStyle().poppinsTitle1()),), R.sizedBox.sizedBox3h(), Align(alignment: Alignment.topCenter, child:InkWell(onTap:(){ getImage(true);}, child:ClipRRect(borderRadius: BorderRadius.circular(20), child: selected_image != null ? Image.file(selected_image!, fit:BoxFit.cover,height:250,): Image.asset(R.images.upload, height: 150,),),),), R.sizedBox.sizedBox3h(),	

Validate all fields for the courier and shows an error massage .	Padding(padding: EdgeInsets.symmetric(horizontal: 10.w, vertical: 1.h), child:MyButton(onTap:(){if (formKey.currentState!.validate() && widget.order.totalPrice!=null){if (selected_image!=null){ exportBill(imageURL);} else { ShowMessage.toast("Please add a bill receipt");}}else{ ShowMessage.toast("Please calculate price"); } }, buttonText: "Export Bill"),),],),),),),); Future getImage(bool gallery) async { ImagePicker picker = ImagePicker(); PickedFile? pickedFile; if (gallery) { pickedFile = await picker.getImage(source: ImageSource.gallery,);} else {pickedFile= await picker.getImage(source: ImageSource.camera,); } setState(() { if (pickedFile != null) { _images.add(File(pickedFile.path)); selected_image = File(pickedFile.path);e } }); } DocumentReference usersRef = FirebaseFirestore.instance.collection('users').doc(); Future<String?> uploadFile(File _image)async{Reference storageReference = FirebaseStorage.instance.ref() .child('orders/${Path.basename(_image.path)}');	

	await storageReference.putFile(_image);
	print('File Uploaded');
	String? returnURL;
	await storageReference.getDownloadURL().then((fileURL) {returnURL = fileURL; });
	return returnURL;}String imageURL = '';
	Future<void>uploadImage(File _image, DocumentReference ref) async {
	if (_image != null) {imageURL=(await uploadFile(_image))!;
	MessageModel msg= MessageModel(type: 1,
	message: imageURL,
	receiver: widget.order.customerId,
	sender: widget.order.courierId,
	roomId: widget.order.id,
	createdAt: Timestamp.now(),
	isSeen: false,);
	Provider.of<ChatVM>(context, listen: false).sendMessage(
	msg,);} else { imageURL = "";}}	

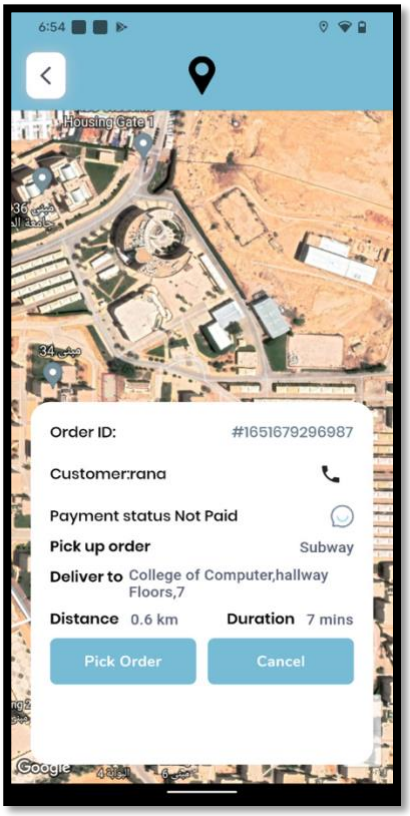
The UploadImage method sends the data to the fire base.		
The ExportBill method sets all the information the firebase	Future<void> exportBill(String image) async {setState(() { isLoading = true;}); awaituploadImage(selected_image!, usersRef);await FBCollections.orders.doc(widget.order.id).update({ 'payment_status':1,'price': priceTC.text,	

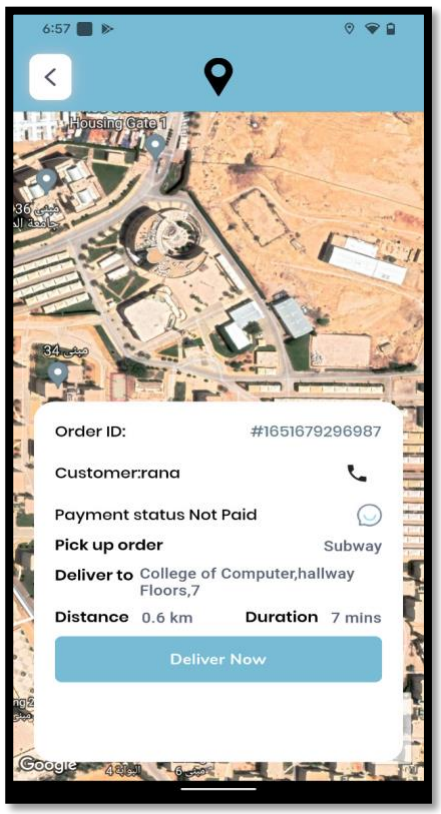
	'totalPrice':widget.order.totalPrice,});
	widget.order.paymentStatus = 1;
	setState(() { isLoading = false; });
	Get.back();}}	

Table 32 Implementation Expert bill

19.3.8 Change order status:

Step	Code Segment	Associated interface

The courier must change his status through his delivery process so the customer will be aware of the couriers' movements
 	
The goToPickup method will change the status of the order, change the button text to picked order and will send a notification to the customer that the courier is on his way to pick the order.
 	
The pickOrder method will change the status of the order, change the button text to arrived at destination and will send a notification to the customer that the courier has picked your order.
	Future<void> goToPickup(Order order, AuthVM authVm) async {
	setState(() {
	isLoading = true;});
	await FBCollections.orders.doc(order.id).update({'status': OrderStatus.goingToPickup.index,});
	AppNotification notify= AppNotification(title:"going for pickup", message:"${authVm.appUser?.fullName} is going to pickup your order no ${order.id}",
	isSeen: false,
	notificationType: NotificationType.goingToPickup,
	sender: authVm.appUser?.email,
	receiver: order.customerId,
	createdAt: Timestamp.now());
	await fbNotification.notifyUser(notify, sendPush: true, sendInApp: true);
	setState(() {isLoading = false; });}
	Future<void> pickOrder(Order order, authVm) async {setState(() {
	isLoading = true; });
	await FBCollections.orders.doc(order.id).update({'status':OrderStatus.pickedFood.index, });
		

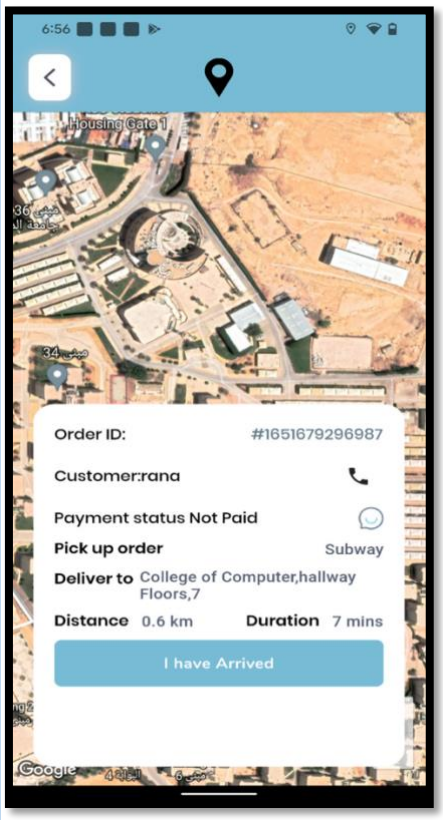
The DeliverOrder method will change the status of the order, change the button text to delivered and will send a notification to the customer that the courier has delivered your order.
 	
The cancleOrder method will change the status of the order, change the button text to Order cancelled and will send a notification to the customer that the courier has canceled your order.
	AppNotification notify = AppNotification(
	title: "Picked your order",
	message:"${authVm.appUser?.fullName} has pickup your order no ${order.id}",
	isSeen: false,
	notificationType: NotificationType.pickedFood,
	sender: authVm.appUser?.email,
	receiver: order.customerId,
	createdAt: Timestamp.now());
	await fbNotification.notifyUser(notify, sendPush: true, sendInApp: true);
	setState(() { isLoading = false; });}
	
	Future<void> deliverOrder(Order order, authVm) async {setState(() { isLoading=true;});await FBCollections.orders.doc(order.id).update({ 'status': OrderStatus.delivered.index, });await FBCollections.users.doc(order.courierId).update({ 'onRide': false, });
	authVm.appUser?.onRide = false;
	authVm.update();
	AppNotification notify = AppNotification(title: "Order Delivered",
	message:"${authVm.appUser?.fullName} has delivered your order",
	isSeen: false,
	notificationType: NotificationType.delivered,
	sender: authVm.appUser?.email,
	receiver: order.customerId,
	createdAt: Timestamp.now());
	await fbNotification.notifyUser(notify, sendPush: true, sendInApp: true);
	setState(() { isLoading = false;});
	Get.offAll(SuccessPage(
	title:'OrderDelivered \nSuccessfully',));}
	Future<void> cancelOrder(Order order, authVm) async {
	setState(() { isLoading = true;});
	await FBCollections.orders.doc(order.id).update({ 'status': OrderStatus.canceled.index, });
	AppNotification notify = AppNotification(title: "Order Canceled",
	message:"${authVm.appUser?.fullName} has canceled your order",
		

The reachDropLocation method will change the status of the order, change the button to text go to pick up and will send a notification to the customer that the courier arrived at pick up location..

The goToPickup method, helps when courier presses the button on his way to the store which at first appears as go to pick up button.

The cancelOrder method, makes the courier has the option to cancel the order before arriving to the store.

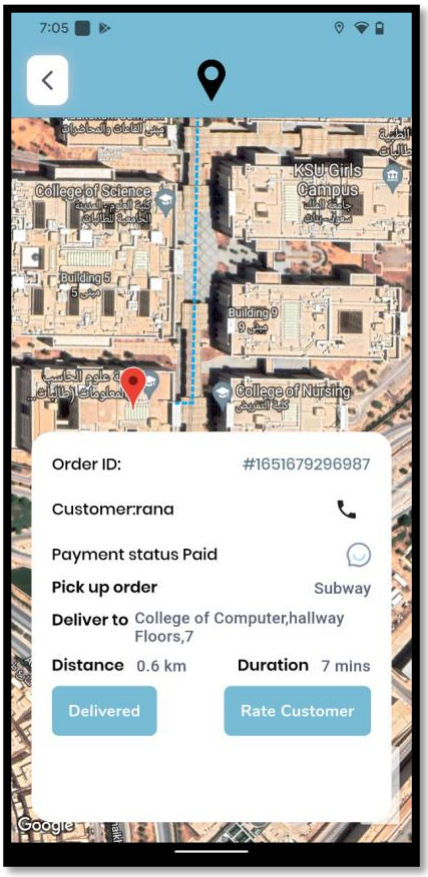
```
        isSeen: false,
        notificationType:
NotificationType.canceledOrder,
        sender: authVm.appUser?.email,
        receiver: order.customerId,
        createdAt: Timestamp.now());
    await
fbNotification.notifyUser(notify,
sendPush: true, sendInApp: true);
    setState(() { isLoading = false;});}
 Future<void> reachDropLocation(Order
order, authVm) async { setState(() {
      isLoading = true;});
 await
FBCollections.orders.doc(order.id).upda
te({'status':
OrderStatus.arrivedAtDrop.index, });
    AppNotification notify =
AppNotification( title: "Arrived at drop",
message:"${authVm.appUser?.fullName
}  have arrived at pickup location",
        isSeen: false,
notificationType:
NotificationType.arrivedAtDrop,
        sender: authVm.appUser?.email,
        receiver: order.customerId,
        createdAt: Timestamp.now());
    await
fbNotification.notifyUser(notify,
sendPush: true, sendInApp: true);
 setState(() { isLoading = false;}); }
lWidget statusWidget(OrderVM
orderVM, AuthVM authVm) {
    switch (orderVM.streamOrder?.status)
{ case 1:return Row(children: [
          Expanded(
            child: MyButton(
                onTap:(){
goToPickup(orderVM.streamOrder!,
authVm); },
 buttonText: "Go To Pickup"),  ),
          R.sizedBox.sizedBox3w(),
          Expanded(
            child: MyButton(
                onTap:(){
cancelOrder(orderVM.streamOrder!,
authVm); },
                buttonText: "Cancel"),)  ], );
      case 2:
        return Row(
          children: [
```



The pickOrder method, helps After the courier presses the go to pick up the button massage will change to pick order which the courier can press after receiving the order.

The reachDropLocation method button will change to arrive and when the courier presses the button it will first validate weather the customer paid or not and it will show a massage.

```
Expanded(child:MyButton(onTap:(){
pickOrder(orderVM.streamOrder!,
authVm); },
              buttonText: "Pick Order"),),
          R.sizedBox.sizedBox3w(),
          Expanded(
            child: MyButton(
               onTap:(){
cancelOrder(orderVM.streamOrder!,
authVm); },
   buttonText: "Cancel"), ) ], );
     case 3:
return MyButton(
onTap:(){
reachDropLocation(orderVM.streamOrd
er!, authVm); },
          buttonText: "I have Arrived");
            case 4:
       return MyButton(
          onTap:(){switch
(orderVM.streamOrder?.paymentStatus)
{   case 0:
 ShowMessage.toast('Please export bill to
customer first');
               break;
             case1:
ShowMessage.toast('Please ask customer
to pay bill first');
               break; case 2:
    deliverOrder(orderVM.streamOrder!,
authVm)  break;} },
          buttonText: "Deliver Now");
     case 5:
       return StreamBuilder(
         stream: FBCollections.ratings
            .where('rate_by',isEqualTo:
authVm.appUser?.email)
.where('order_id',isEqualTo:
orderVM.streamOrder?.id)
            .snapshots(),
         builder:(context,
AsyncSnapshot<QuerySnapshot>
snapshot) {if (!snapshot.hasData) {
return Center(child: MyLoader(), );}
else
if (snapshot.data!.docs.isEmpty) {
 Return Row(
mainAxisAlignment:
MainAxisAlignment.spaceBetween,
children: [MyButton(onTap:(){},
buttonText: "Delivered"),
MyButton(
 onTap: () { Get.to(() => RateUser(
```



	orderId: orderVM.streamOrder?.id, userId: orderVM.streamOrder?.customerId,)); }, buttonText: "Rate Customer")],); } else { return Column(children: [MyButton(onTap: () {}, buttonText: "Delivered"), RatingBar.builder(ignoreGestures: true, initialRating: double.parse('${snapshot.data!.docs[0]['rating']}'), direction: Axis.horizontal, allowHalfRating: true, itemCount: 5, itemPadding: EdgeInsets.symmetric(horizontal: 4.0), itemBuilder:(context,_)=>Icon(Icons.star, color: Colors.amber,), onRatingUpdate: (rating) {},)],); } },);default: return SizedBox(); } }}	

Table 33 Implementation Change order status

19.4 Actual Database Schema

This section captures a snapshot of KSUDS data structure on the Cloud Firestore. Data is stored in documents organized in collections. Documents are stored in collections, and each document represents a lightweight record with fields that map to values. The design decision of utilizing Firestore, a NoSQL document-oriented database, instead of traditional Real time database is because of its support for customizable, hierarchical data structures, as well as its powerful expressive and efficient querying capabilities. The following figures illustrates the final data schema of KSUDS.

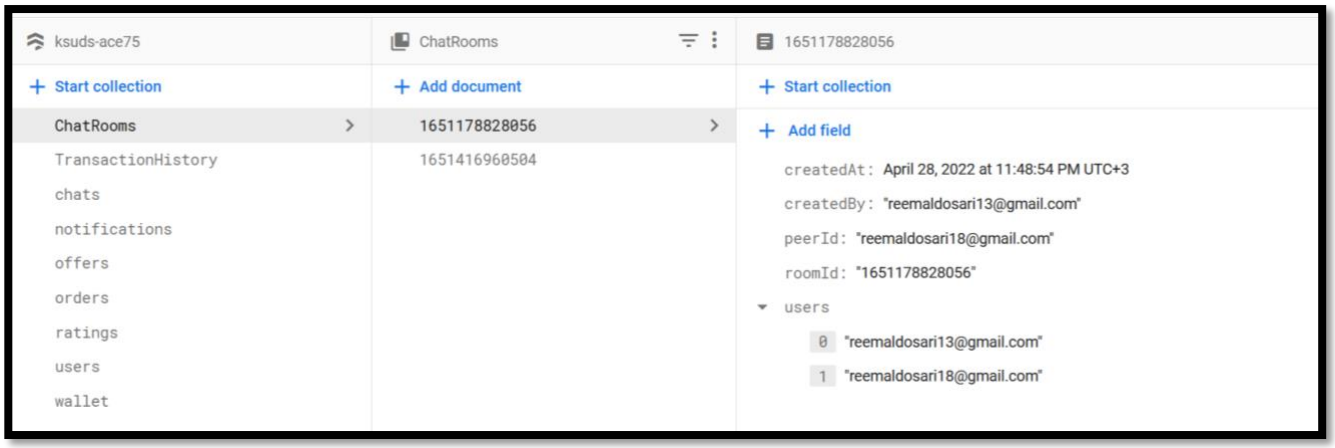


Figure 54 Actual database schema

The data schema organized into multiple collections consisting of Chat Rooms, Transaction History, chats, notifications, offers, orders, rating, users, and the wallet. The Chat Rooms collection contain documents of each chat along with their information as shown in Figure above. Furthermore, in the same manner, Figure 55,56,57,58,59,60,61,62, and 63 illustrates the Transaction History, chats, notifications, offers, orders, rating, users, and the wallet.

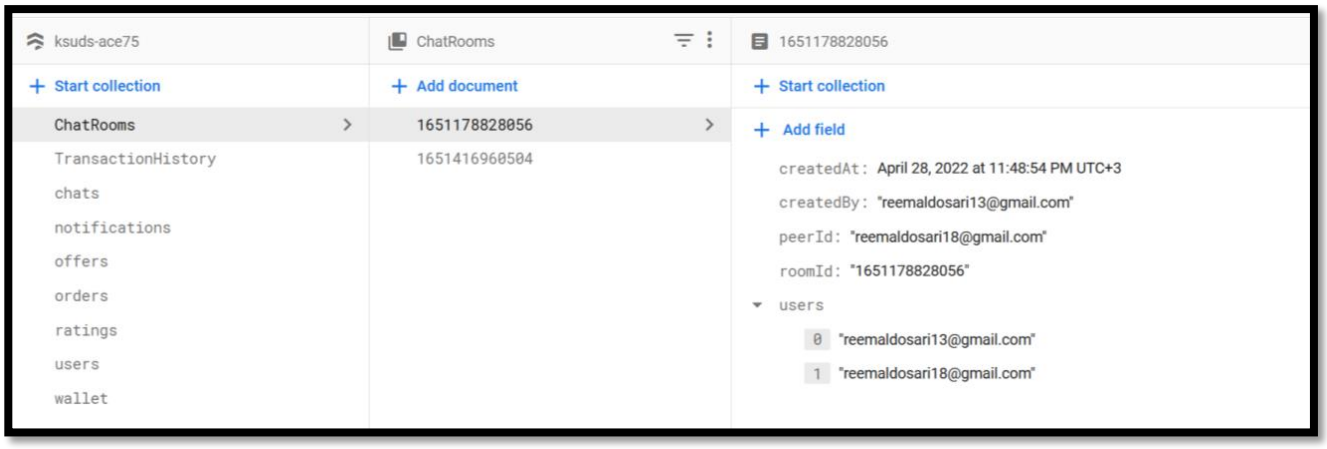


Figure 55 Chat Rooms collection

The collection Chat Rooms which consist of the created At, created By, peer Id, room Id, and the users.

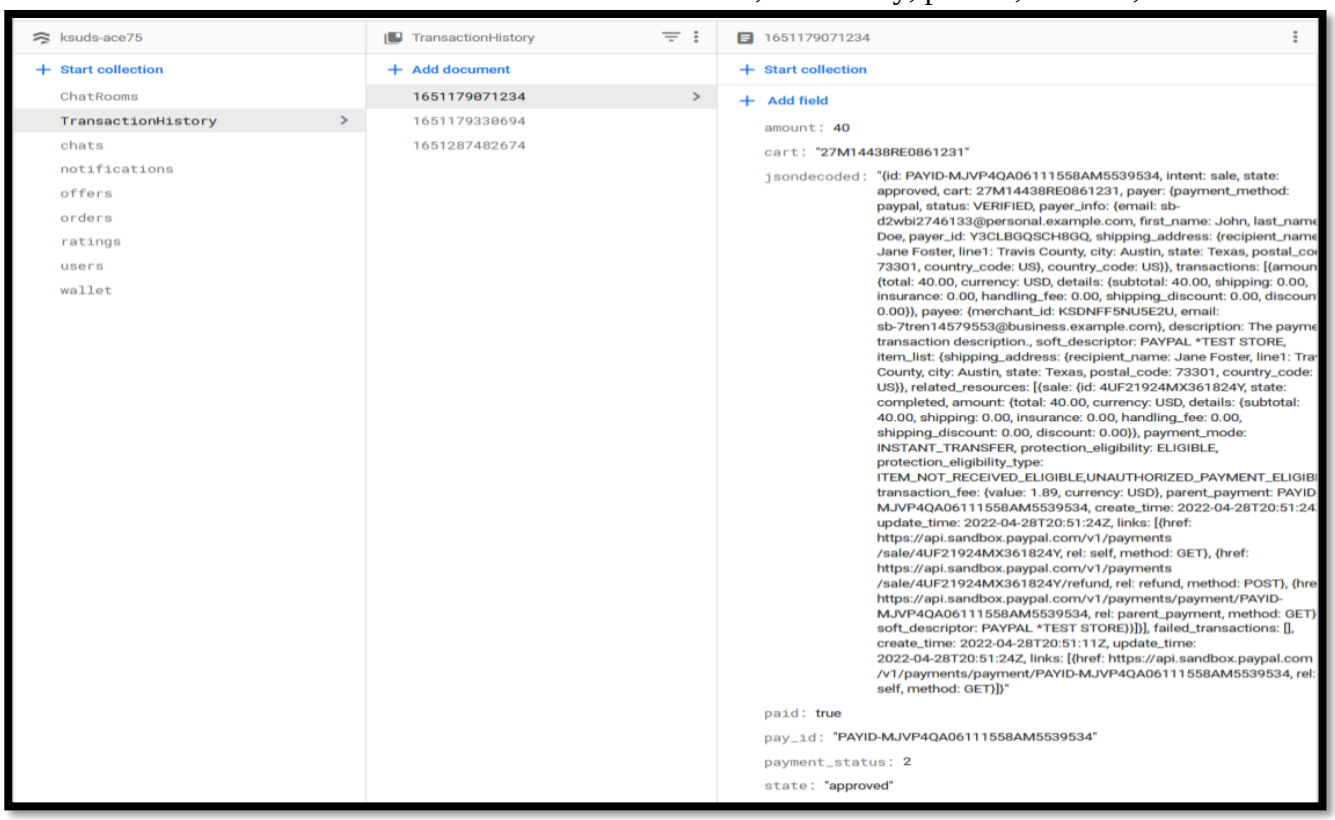


Figure 56 Transaction History collection

The collection Transaction History which consists of the amount, cart, jsondecoded, paid, pay_id, payment_status, and the state.

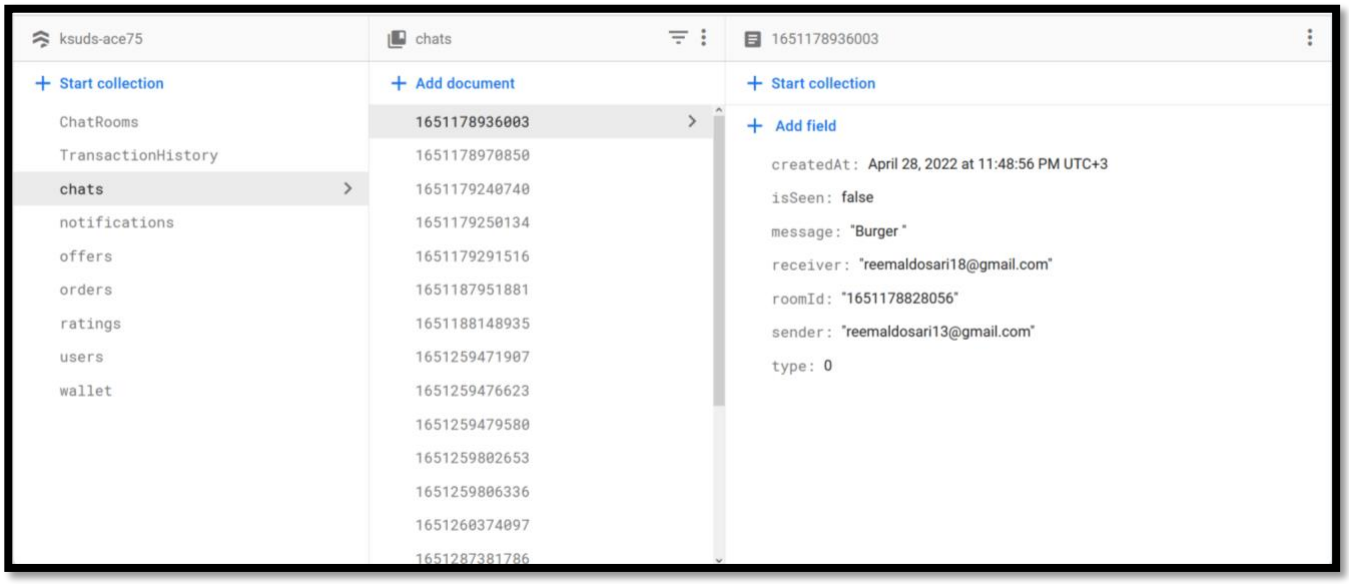


Figure 57 chats collection

Collection chats consist of the createAt, isSeen, message, receiver, roomID, sender, and type.

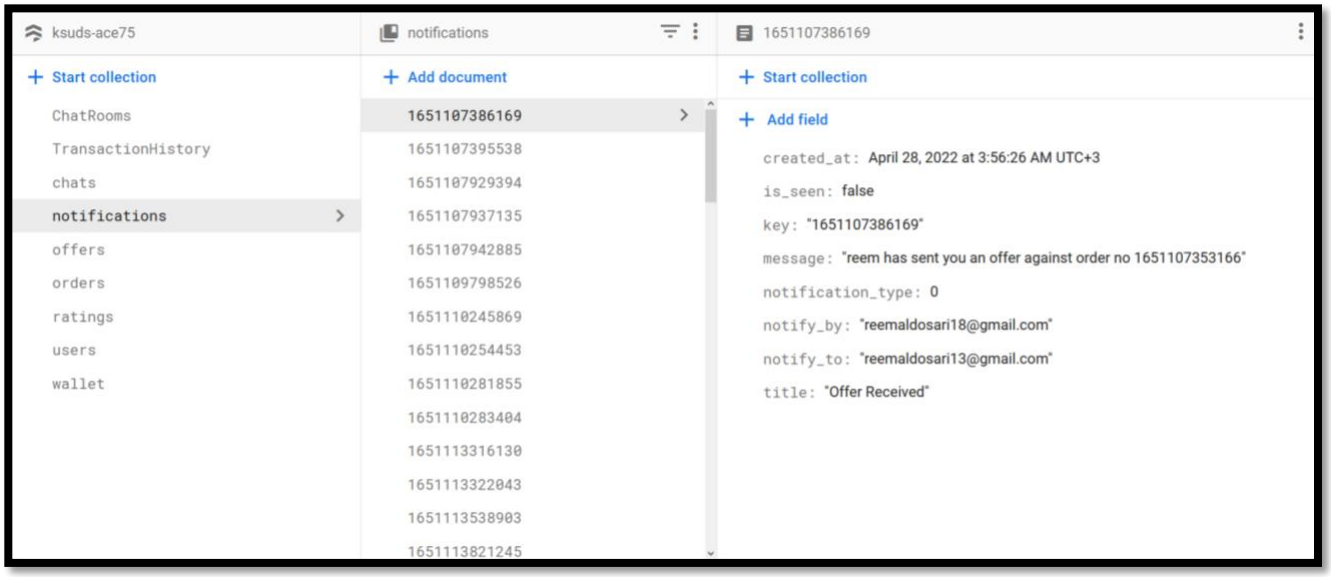


Figure 58 notifications collection

collection notifications consist of the createAt, isSeen, key, message, notification_type, notify_by, notify_to, and the title.

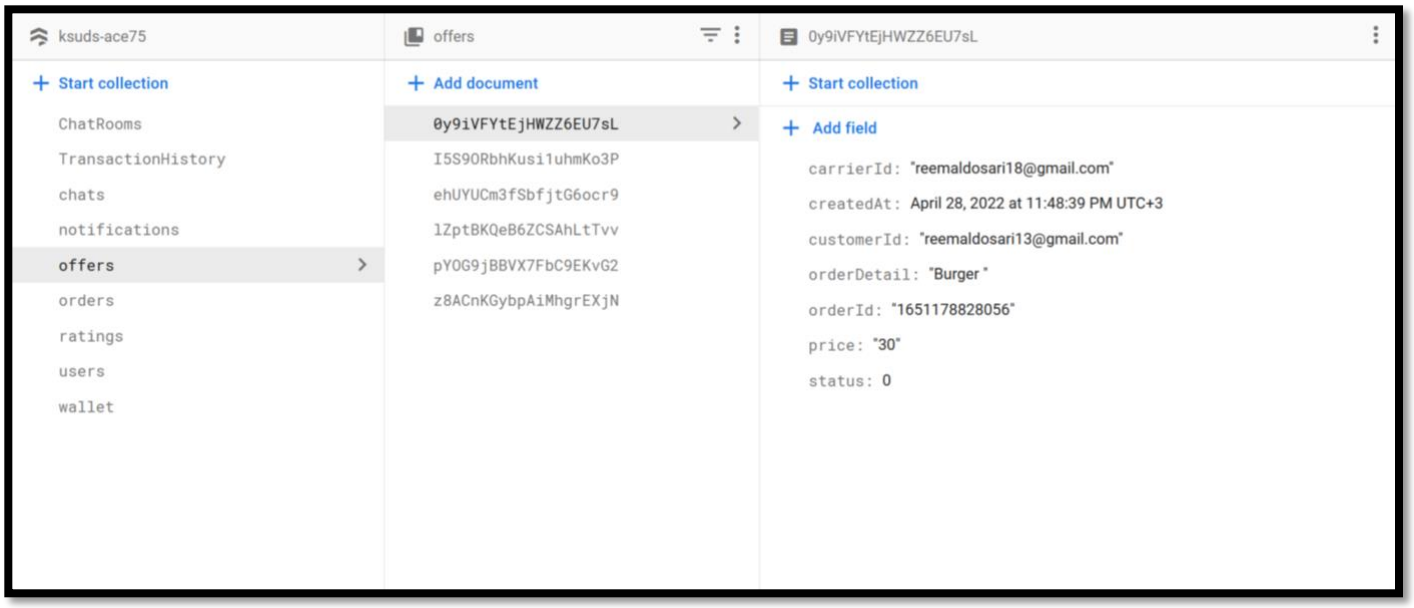


Figure 59 offers collection

Collection offers consist of the courierId, createdAt, customerId, orderId, price, and the Status.

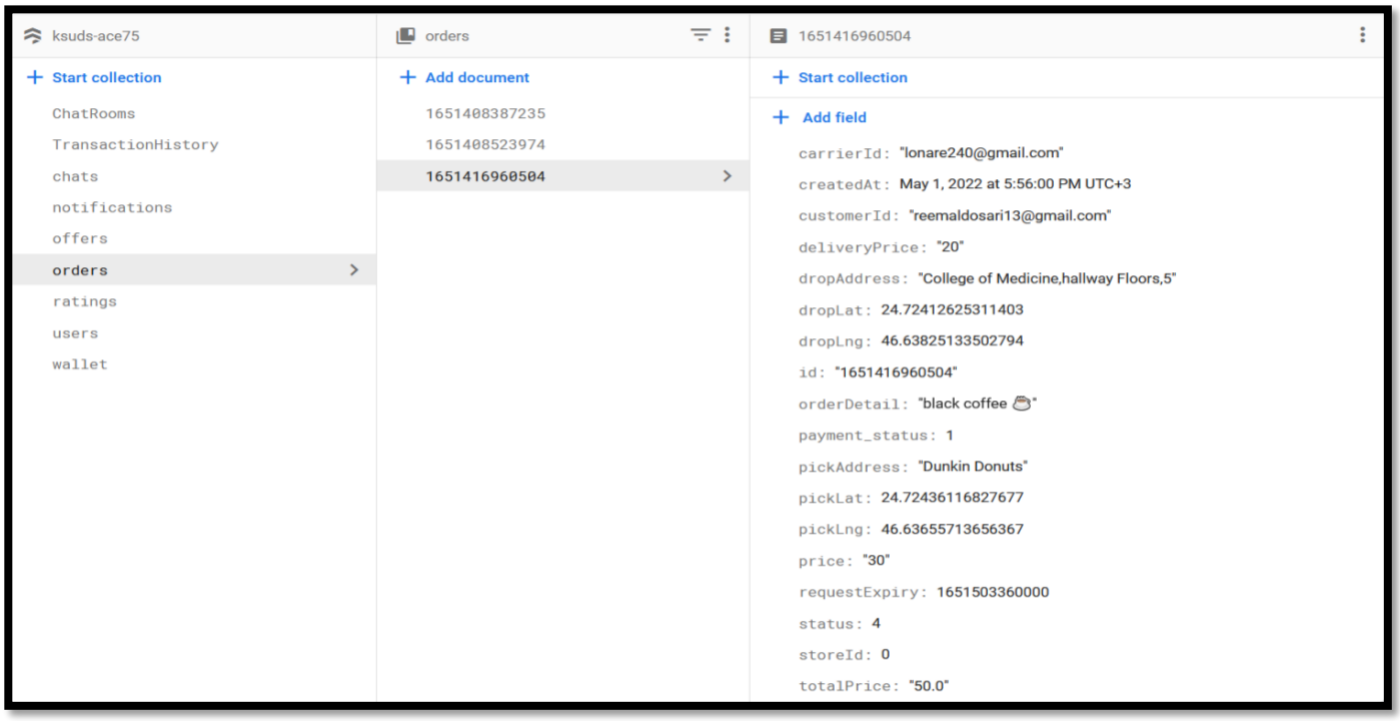


Figure 60 orders collection

Collection orders consist of the courierId,createdAt, customerId, deliveryPrice, dropAddress, dropLat, dropLng, id, orderDetail, payment_status, pickAddress, pickLat, pickling, price, requestExpiry, status, storeId, and the totalPrice.

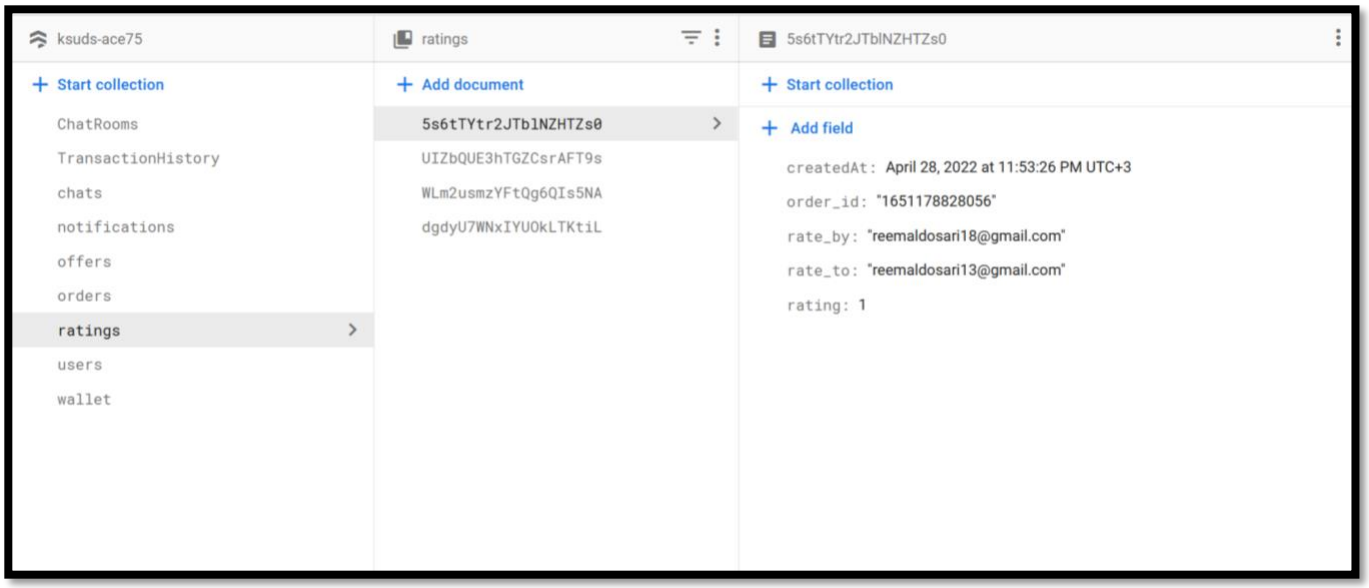


Figure 61 ratings collection

Collection ratings consist of the createdAt, orderId, rate_by, rate_to, and the rating.

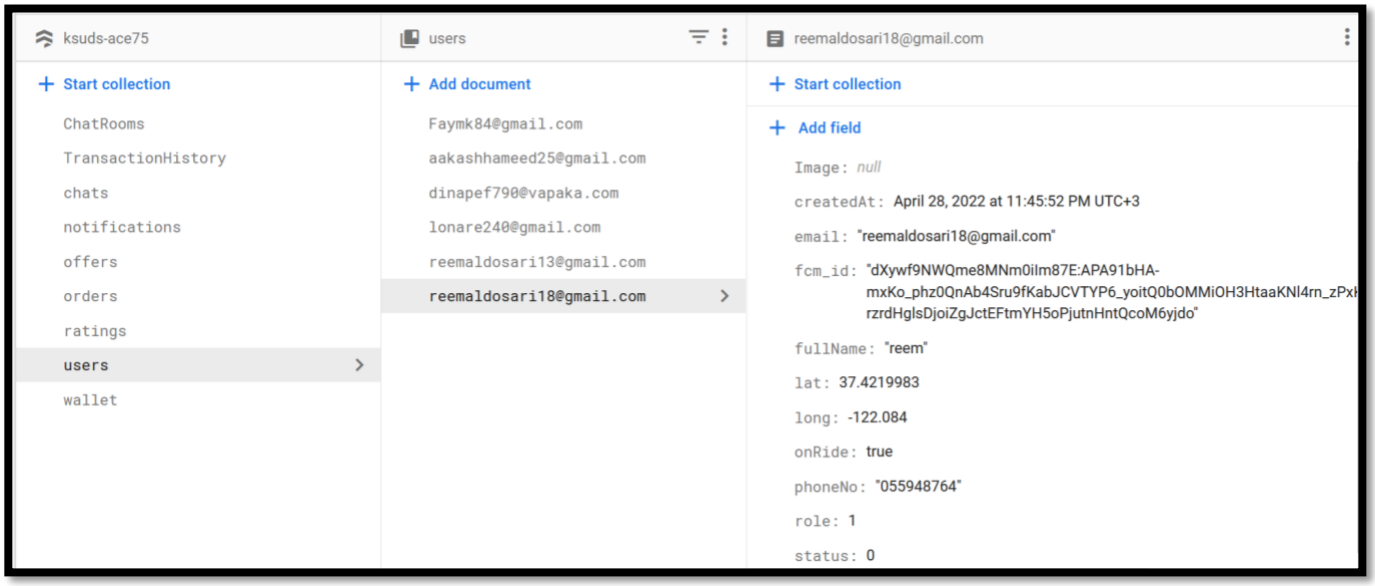


Figure 62 users collection

Collection users consist of the Image, createdAt, email, fcm_id, fullName, onRide, phoneNo, role, and the status

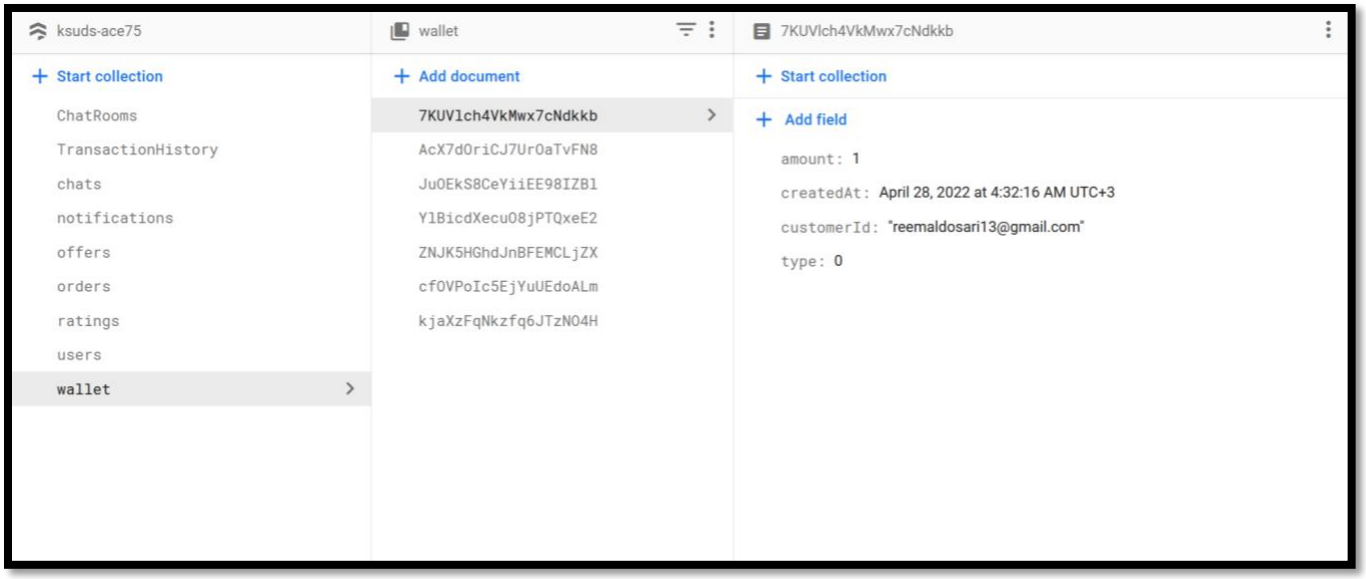


Figure 64 wallet collection

Collection wallet consist of the amount, createdAt, customerId, and the type.

19.5 User Interface

This section describes the critical scenarios and illustrates the user interface of KSUDS's application where we've ensured that the interface has elements that are easy to access and understand to native users. The tables below present a walkthrough of critical scenarios with a brief description.

Critical Scenario #1: Registration, Login, and Edit profile		
Starting with the registration for a new account a customer/courier must fill the following fields with the right format information to create an account.	When the customer/ courier wants to login into the application, they'll have to provide an already exciting account, filling a correct email and password into the right text fields.	The customer/ courier can view their profile information, and manage their accounts by editng their information in the profile tab.

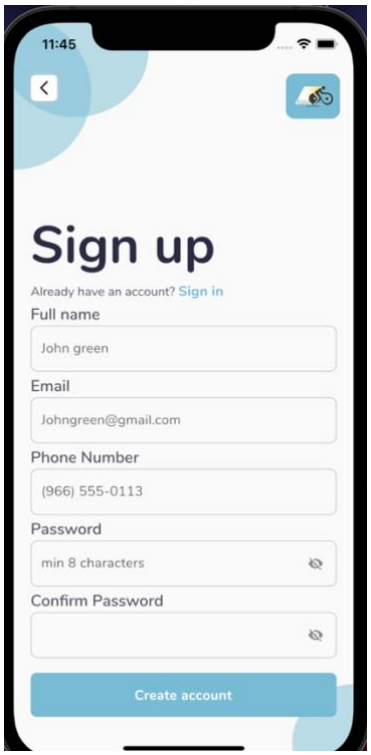
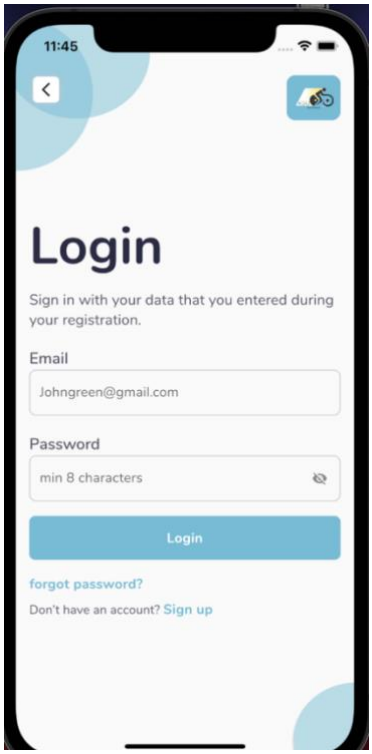
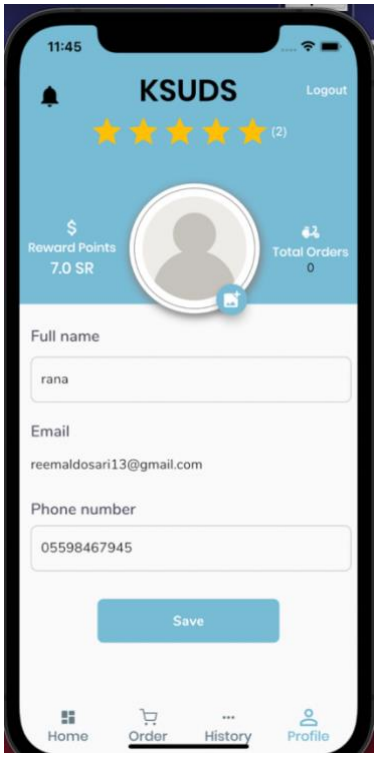

		

Figure 65 Actual sing up UI	*Figure 66 Actual Log in UI*	*Figure 67 Actual Edit profile UI*

Table 34 UI Critical Scenario #1

Critical Scenario #2: Home page, view menu, and type order		
When the customer login the first page that will be displayed is the home screen page which has all the listed places, he/she can order from.	The customer can view the menu, the he/she can select and write their order in the make an order in the order page.	When the customer wants to make an order based on the place, he/ she selected, they can locate the location, and write his/her order.

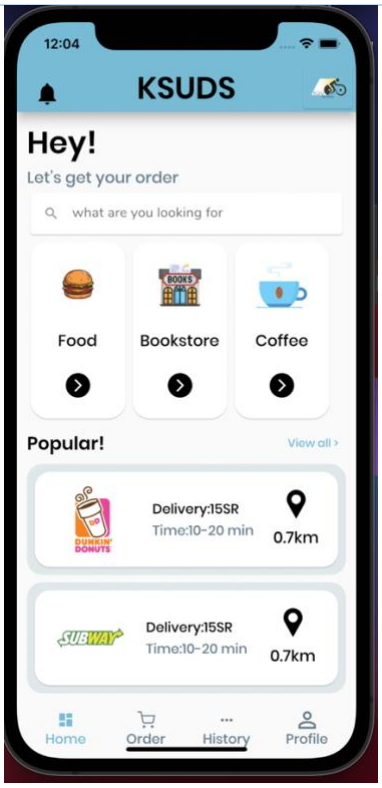
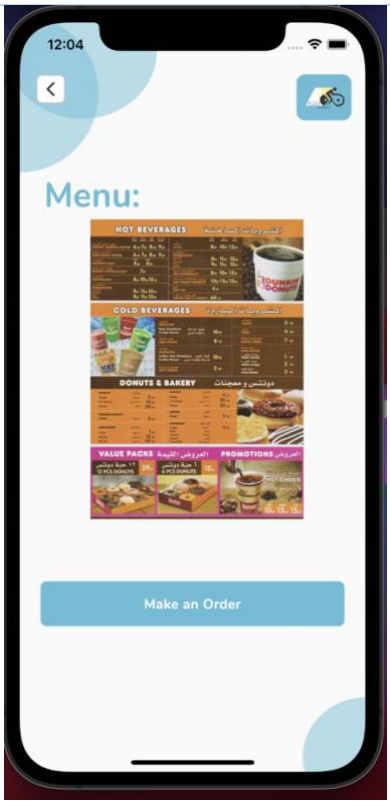
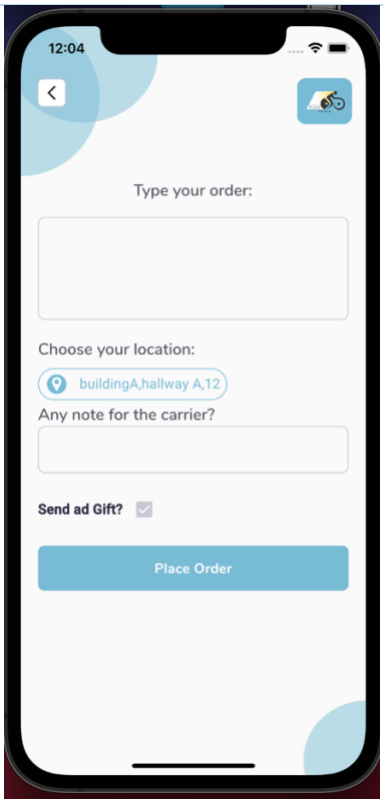

		

Figure 68 Actual Home page for customer UI	*Figure 69 Actual Menu UI*	*Figure 70 Actual Type order UI*

Table 35 UI Critical Scenario #2

Critical Scenario #3 Send offer, and Accept offer		
When the customer sends an offer to select courier.	Here all offers will be shown to the customer, courier information rating/ price.	After selecting an offer a confirmation message will be shown.

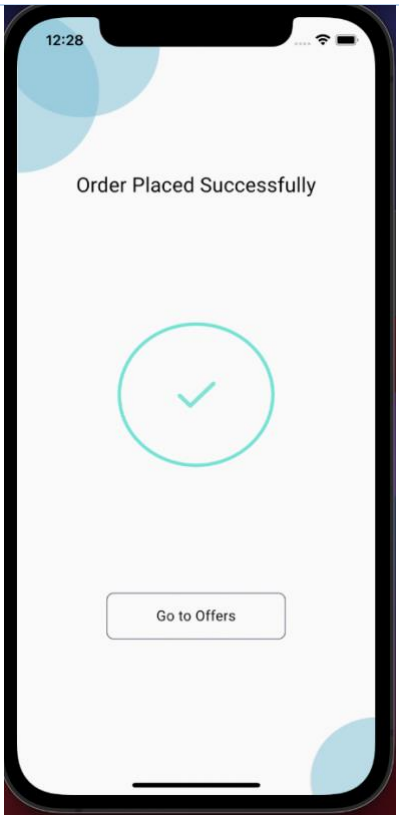
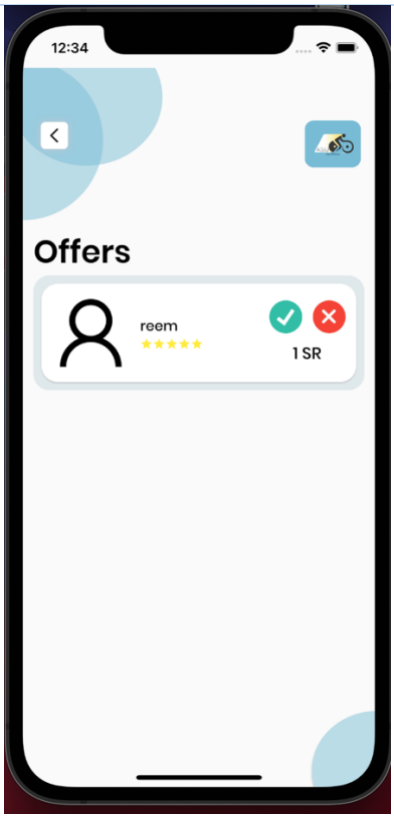
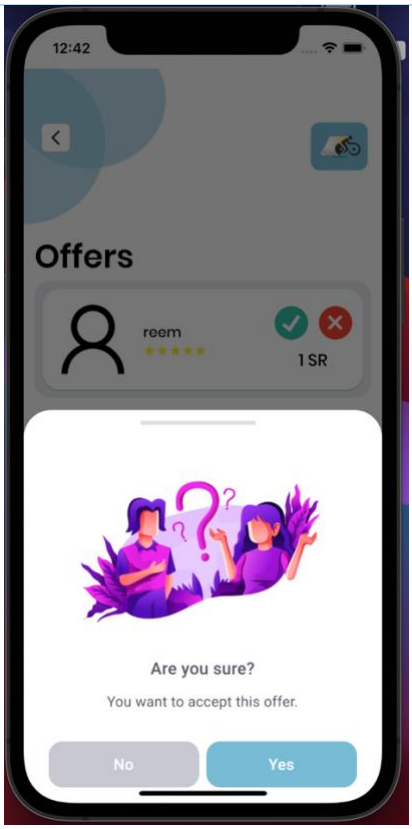

		

Figure 71 Actual Place an order UI	*Figure 72 Actual offers UI*	*Figure 73 Actual conformation offers UI*

Table 36 UI Critical Scenario #3

Critical Scenario #4: Home page courier, Send offer, and chat		
When the courier login the first page that will be displayed is the home screen page which has all the listed orders come from customers.	When the courier selects an order, a box will be shown where he/she can fill the desired price.	After customer accept the offer, it will be shown in current order for both customer/ couriers.

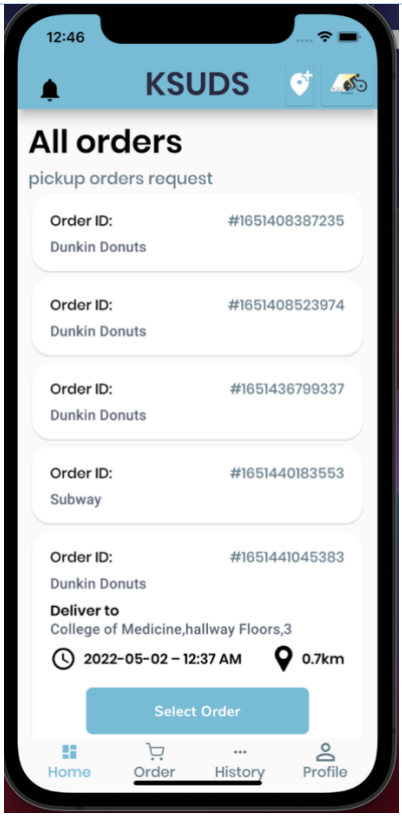


Figure 71 Actual Home page for courier UI

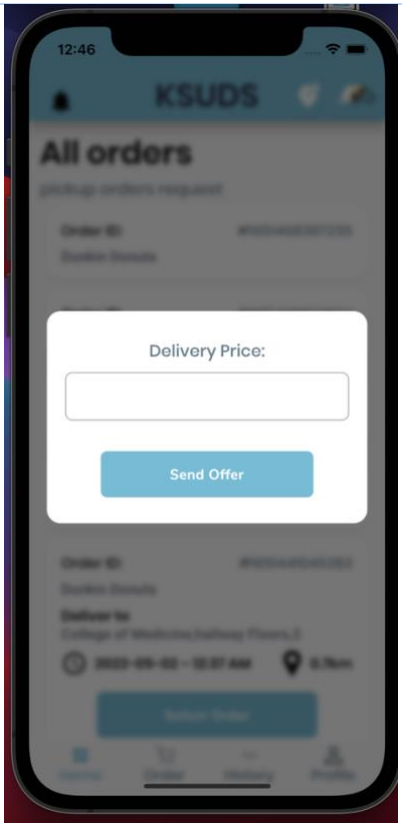


Figure 72 Actual Send price UI

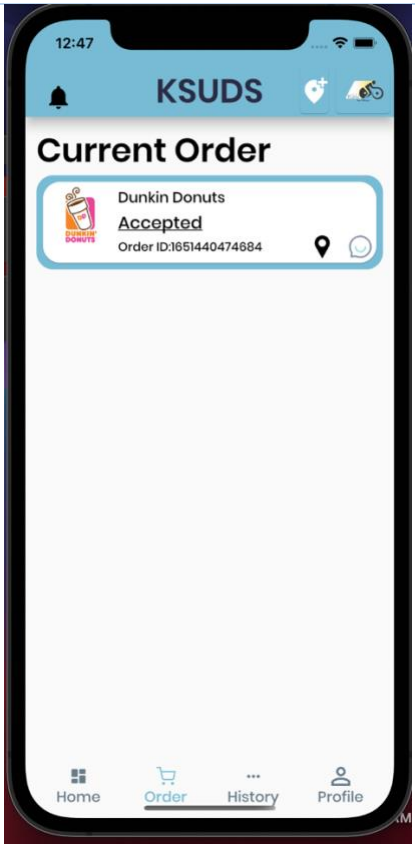


Figure 73 Actual current order UI

Table 37 UI Critical Scenario #4

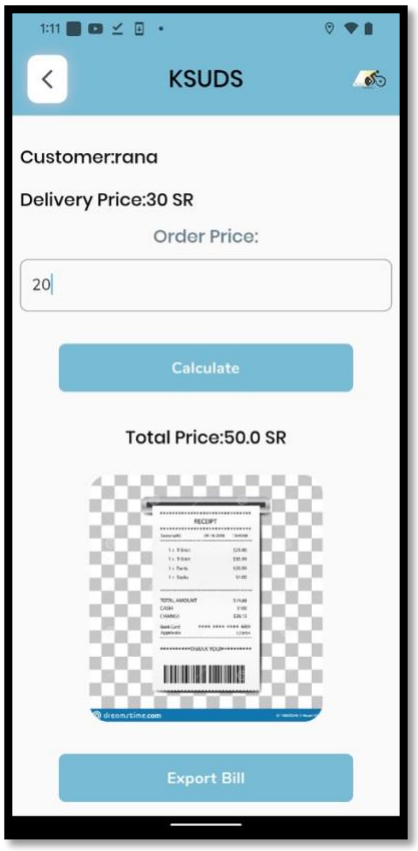
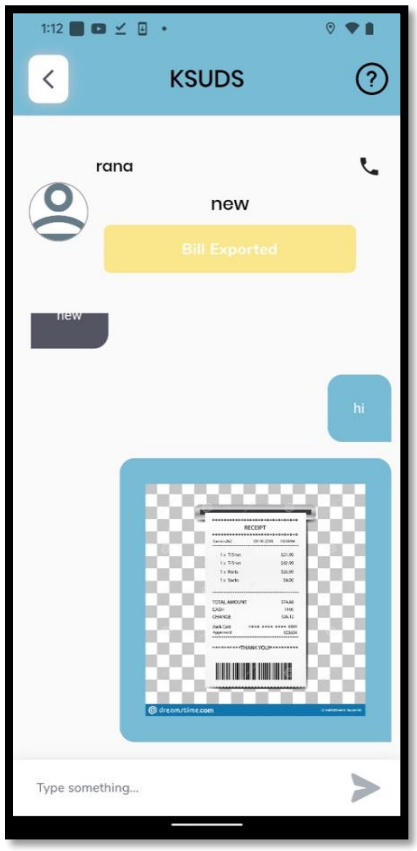
Critical Scenario #5: Expert bill, Rate customer, and point in profile	
When the courier changes the order status to “I have arrived” he/she clicks the “Export bill” button	When the courier clicks “export bill” button the system will upload the bill to the chat including the photo and the price
 *Figure 74 Actual Export bill page*	 *Figure 75 Actual Export bill in the chat*

Table 38 UI Critical Scenario #5

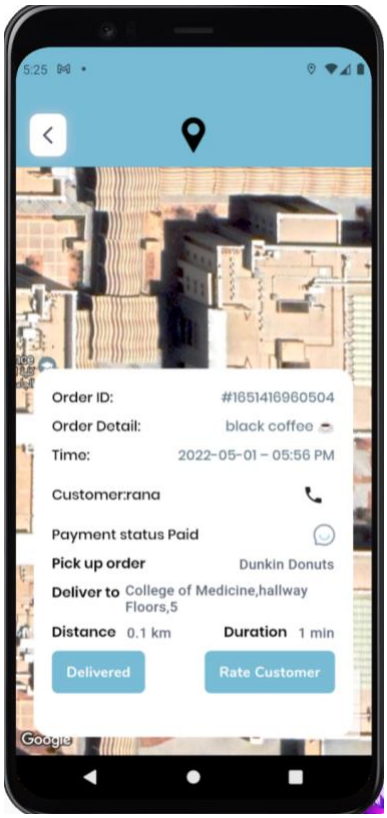
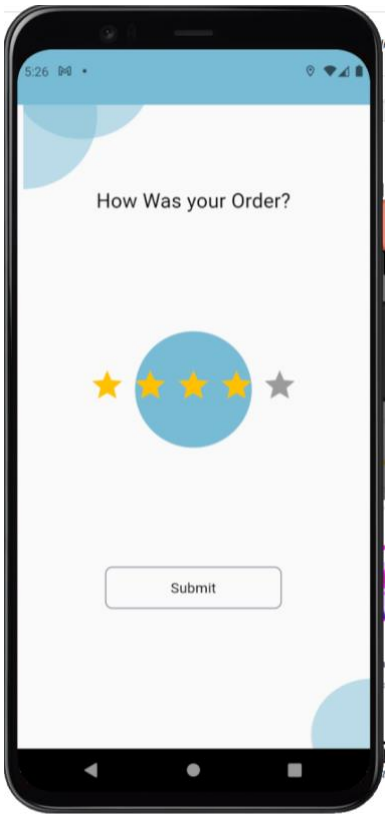
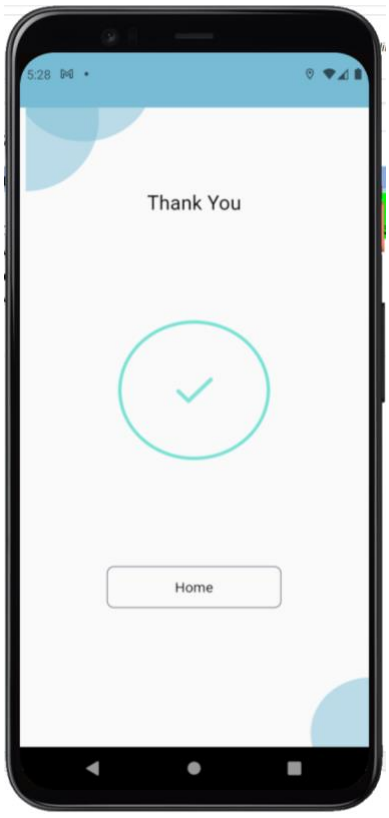
Critical Scenario #6: pay bill, Rate customer, and point in profile		
When the courier delivered the order, he/she can rate the customer by clicking the “Rate Customer” button.	The System will direct the courier to the rate page, and he/she will click the stars icon to rate the customer after that the courier clicks the “Submit” button	The system will direct the courier to the confirmation page and then the courier selects the “home” button to return to the home page
 *Figure 77 Actual Rate Customer for the courier UI*	 *Figure 78 Actual Rating customer UI*	 *Figure 79 Actual confirmation page UI*

Table 39 UI Critical Scenario #6

20. Testing

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.[32] Testing is an important part of quality assurance. It is concerned with verifying the genuine running software and approving its compliance with the requirements. This section describes the methods for validating a system's compliance with its specifications as defined in the Quality Assurance plan.

- **Unit testing:** To ensure the quality of each software component the team used unit testing to make sure that it performs as expected and it is free of any possible defects.
- **Functional testing:** The team used the "black box testing" to test the major functionality of KSUDS system.
- **Usability testing:** Usability testing is conducted to determine whether the KSUDS system satisfy its desired end-user needs or not by involving end-users in the testing process to gather their feedback.

20.1 Test Scenario

The following list represents the tested functional aspects of the system from end-user perspective scenarios are different for each user (customer, courier). The test cases are listed in the table below.

#	Test cases for customer	Testing technique
1	Register	Equivalence partitioning
2	Log in	Unit testing / Equivalence partitioning
3	Log out	Usability testing
4	Edit Profile Information	Unit testing / Usability testing
5	Make an order	Unit testing / Usability testing
6	View map	Unit testing / Usability testing
7	Track Order	Usability testing
8	Rate Order	Equivalence partitioning
9	Pay bill	Test Case
10	Report an issue	Unit testing / Usability testing
11	Rate courier	Equivalence partitioning
12	View ratings	Unit testing
13	Use points	Equivalence partitioning
14	Filter restaurant	Test case

15	View history	Usability testing
16	Select an offer	Scenario based testing /Usability testing
17	Forget password	Unit testing / Usability testing

Table 40 Test case for customer and technique

#	**Test Cases for courier**	**Testing technique**
1	Register	Equivalence partitioning
2	Log in	Unit testing / Equivalence partitioning
3	Log out	Usability testing
4	Edit profile information	Test case/ Unit testing
5	View map	Test case/ Unit testing
6	Report an issue	Unit testing / Usability testing
7	Export bill	Equivalence partitioning
8	Rate customer	Test case
9	Contact customer	Test case / Usability testing
10	Change order status	Scenario based testing/ Usability testing
11	View earnings	Unit testing
12	View ratings	Unit testing
13	View active order	Usability testing
14	View history order	Usability testing

Table 41 Test case for courier r and technique

20.2 Unit Testing:

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit tests are handy for verifying the behavior of a single function, method, or class. The test package provides the core framework for writing unit tests, and the flutter_test package provides additional utilities for testing widgets.[33]

20.2.1 Edit profile information

Test ID	Function Name	Description	Inputs	Expected Output	Actual Output	V erdict
1	validateName	This test cases tests the name field is left empty	Name=null	Error massage “Please enter your full name”	Error massage “Please enter your full name”	pass
2	validateName	This test case tests the name field when courier/custom er enter correct name	Name= aljoury ibrahim	Will successfully place name	Will successfully place name	Pass
3	validateName	This test case test the name when it has invalid characters	Name=jour y/^^	Error massage “incorrect full name”	Error massage “incorrect full name”	pass
4	Validate phone number	This test case test the phone number field is left empty	phoneNumb er=null	Error massage “Please enter your phone number”	Error massage “Please enter your phone number”	Pass
5	Validate phone number	This test case test the correct phone number	PhoneNumb er=5524839 9	Will successfully phone number	Will successfully phone number	Pass

6	Validate phone number	This test case test when Phone number less than or equal 8 error stirng	PhoneNumber= 5524899	Error massage “Invalid number“	Error massage “Invalid number“	Pass
7	Validate phone number	This test case test when Phone number with wrong pattren	PhoneNumber= 55248//99	Error massage “Invalid number“	Error massage “Invalid number“	Pass

Test Result:

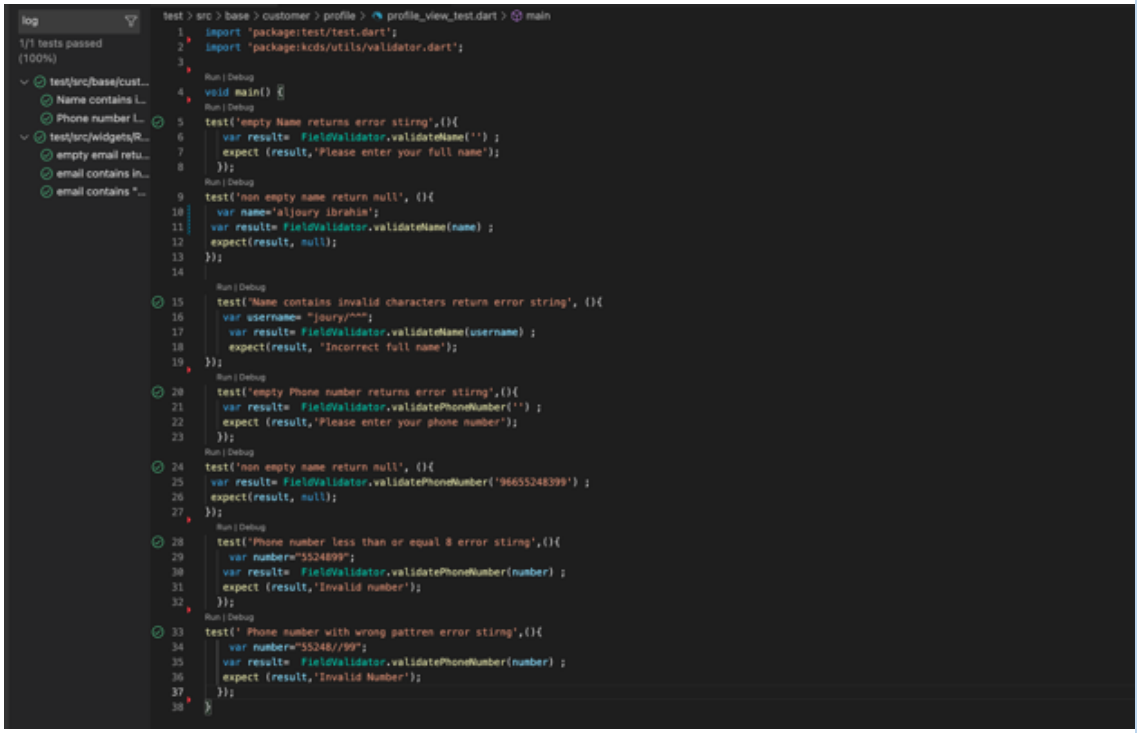


Figure 80 Edit information unit test result

Table 42 Edit information unit test

20.2.2 Reset password

Test ID	Function Name	Description	Inputs	Expected Output	Actual Output	V erdict
8	validateEmail	This test cases tests the email field is left empty	Email=null	Error massage “Please enter email address”	Error massage “Please enter email address”	pass
9	validateEmail	This test case tests the name field when courier/customer enter correct email	Email= email2gmail.com	Will successfully place email adress	Will successfully place email adress	Pass
10	validateEmail	This test case test the email contains invalid characters	Email= joury^^^^gmail.com	Error massage “Invalid number“	Error massage “Invalid number“	Pass

Test result:

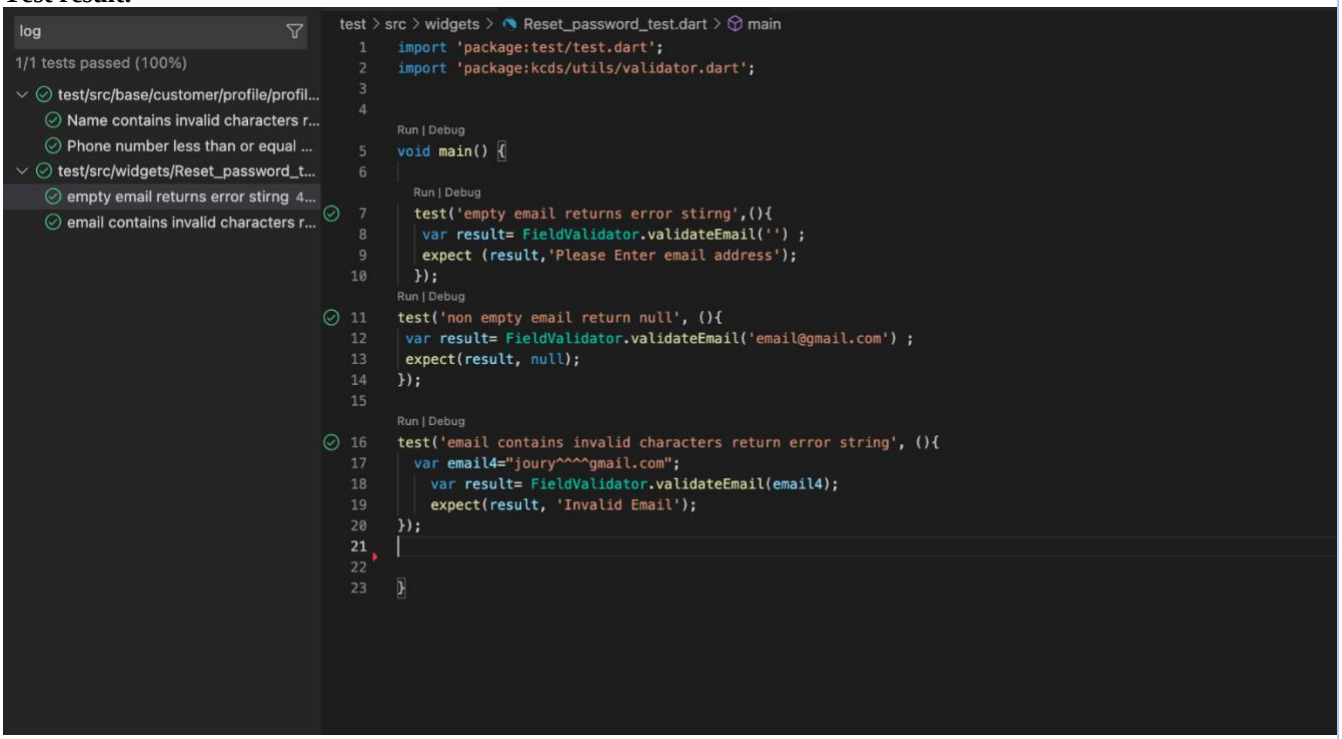


Figure 81 Rest password unit test result

Table 43 Rest password unit test

20.3 Functional Testing:

is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.[34]

20.3.1 Black box testing

is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications, and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.[35]

20.3.1.1 Registration equivalence classes and boundary value analysis

a. Registration equivalence classes

Input Condition	Valid equivalence classes	Invalid equivalence classes
Full name	- -Full name ∈ [A-Z] **(1)** - - Full name ∈ [a-z] **(2)** - - Full name ∈ [white space] **(3)**	- Full name ∉ [0-9] **(4)** - Full name ∉ [!#$%&'*+-/=?^_{|}~] **(5)** - Empty **(6)**
Email ***Emailname@Domain.com***	- Email name ∈ [a ... z] **(7)** - Email name ∈ [A ... Z] **(8)** - Email name ∈ [0 ... 9] **(9)** - Email names have special characters **(10)** - Email name is present **(11)** - Domain name∈ [a ... z] **(12)** - Domain name∈ [A ... Z] **(13)** - Domain name∈ [0 ... 9] **(14)** - Domain name is present **(15)** - If email name exists. Then, it must be followed by @ -character-**(16)** - Correct sequence. (**17**)	- Email name is absent **(18)** - Domain names have special characters **(19)** - Domain name is absent **(20)** - If email name exists. Then, it's not followed by @ -character-**(21)** - Incorrect sequence. **(22)**
Phone Number	Phone number length = 10. **(23)** Phone number starts with '05'. **(24)** Phone Number ∈ [0-9]. **(25)**	- Phone number length ≠10. **(25)** - Phone number ∈ [A-Z/a-z]. **(26)** - Phone number ∈ [!#$%&'*+-/=?^_{|}~]. **(27)**

Password	- password ∈ [a ... z] **(28)** - password ∈ [A ... Z] **(29)** - password∈ [0 ... 9] **(30)** - password has special characters (3**1)** - password length=< 8 **(32)** - password is present **(33)**	- password has a non-English character. **(34)** - password length< 8 character **(35)** - Password is absent **(36)**
Form	- Input fields are filled. **(37)** - Email address is not registered. **(38)** - Phone number is not registered. **(39)** - Password fields are matching. **(40)**	- Incomplete fields. **(41)** - Email address is registered. **(42)** - Mobile number is registered. **(43)** - Password fields don't match. **(44)** - Empty form. **(45)**

Table 43 Registration equivalence classes table

b. Identify Use Cases for Valid equivalence classes

Use- Case Data	Cover equivalence classes
T01 : Name: Lama Alsanie Email: lamalsanie6@hotmail.com Phone number: 0556165555 Password: Lama1655@ Confirm Password: Lama1655@	(1), (2), (3), (7), (8), (9), (10), (11), (12), (13), (14), (15), (16), (17), (23), (24), (25), (28), (29), (30), (31), (32), (33), (37), (38), (39) (40)

Table 44 valid use case Registration equivalence classes

c. Identify Use Cases for Invalid equivalence classes

Test Case 2	Covered EC
T02 : Name: La9a Alsanie Email: lamalsanie6@hotmail.com Phone number: 0556165555 Password: Lama1655@ Confirm Password: Lama1655@	(4)
T03 : Name: La*a Alsanie Email: lamalsanie6@hotmail.com Phone number: 0556165555 Password: Lama1655@ Confirm Password: Lama1655@	(5)

T04 : Name: Email: lamalsanie6@hotmail.com Phone number: 0556165555 Password: Lama1655@ Confirm Password: Lama1655@	(6)
T05 : Name: Lama Alsanie Email: Phone number: 0556165555 Password: Lama1655@ Confirm Password: Lama1655@	(18)
T06 : Name: Lama Alsanie Email: lamalsanie6@hotm&il.com Phone number: 0556165555 Password: Lama1655@ Confirm Password: Lama1655@	(19)
T07 : Name: Lama Alsanie Email: lamalsanie6@ Phone number: 0556165555 Password: Lama1655@ Confirm Password: Lama1655@	(20)
T08 : Name: Lama Alsanie Email: lamalsanie6%hotmail.com Phone number: 0556165555 Password: Lama1655@ Confirm Password: Lama1655@	(21)
T09 : Name: Lama Alsanie Email: hotmail.com@lamalsanie6 Phone number: 0556165555 Password: Lama1655@ Confirm Password: Lama1655@	(22)
T10 : Name: Lama Alsanie Email: lamalsanie6@hotmail.com Phone number: 05561655511 Password: Lama1655@ Confirm Password: Lama1655@	(25)
T11 : Name: Lama Alsanie Email: lamalsanie6@hotmail.com Phone number: 0556165555F Password: Lama1655@ Confirm Password: Lama1655@	(26)

T12 : Name: Lama Alsanie Email: lamalsanie6@hotmail.com Phone number: 0556165555% Password: Lama1655@ Confirm Password: Lama1655@	(27)
T13 : Name: Lama Alsanie Email: lamalsanie6@hotmail.com Phone number: 0556165555 Password: لama1655@ Confirm Password: لama1655@	(34)
T14 : Name: Lama Alsanie Email: lamalsanie6@hotmail.com Phone number: 0556165555 Password: Lama1@ Confirm Password: Lama1@	(35)
T15 : Name: Lama Alsanie Email: lamalsanie6@hotmail.com Phone number: 0556165555 Password: Confirm Password:	(36)
T16 : Name: Email: lamalsanie6@hotmail.com Phone number: 0556165555 Password: Lama1655@ Confirm Password:	(41)
T17 : Name: Lama Alsanie Email: lamaalsanie1@gmail.com Phone number: 0556165555 Password: Lama1655@ Confirm Password: Lama1655@	(42)
T18 : Name: Lama Alsanie Email: lamalsanie6@hotmail.com Phone number: 0556165511 Password: Lama1655@ Confirm Password: Lama1655@	(43)

T19 : Name: Lama Alsanie Email: lamalsanie6@hotmail.com Phone number: 0556165555 Password: Lama1655@ Confirm Password: Lama1655&	(44)
T20 : Name: Email: Phone number: Password: Confirm Password:	(45)

Table 45 Invalid use case Registration equivalence classes

d. Registration boundary value analysis

This test complements the preceding test and is designed to test the boundary values of valid and invalid input domains of the KSUDS registration feature. Table 18 below identifies the boundary values to each input condition of the registration form given its corresponding constraint. The length of the input Phone Number, and Password are tested at the boundaries of the constraint. Test results are given in Table 19,20, and 21 respectively.

Input condition	Constraint	Boundary Value
Phone Number	Length = 10	[9] [10] [11]
Password	Length>=8	[7] [8] [9]

Table 46 Registration boundary value analysis table

1. Phone number

Test Value	Boundary Value
T01 : Name: Lama Alsanie Email: lamalsanie6@hotmail.com Phone number: 055616555 Password: Lama1655@ Confirm Password: Lama1655@	9
T02 : Name: Lama Alsanie Email: lamalsanie6@hotmail.com Phone number: 05561655551 Password: Lama1655@ Confirm Password: Lama1655@	10

T03 : Name: Lama Alsanie Email: lamalsanie6@hotmail.com Phone number: 05561655551 Password: Lama1655@ Confirm Password: Lama1655@	11
Test Result: Test cases with invalid length of 9 and 11 digits pass successfully by displaying an error message to the user. Whereas a test case with a valid length of 10 digits passes successfully and resumes the registration process.	

Table 48 Phone Number Boundary Test

2. Password

Test Value	Boundary Value
T01 : Name: Lama Alsanie Email: lamalsanie6@hotmail.com Phone number: 0556165555 Password: Lama16@ Confirm Password: Lama16@	7
T02 : Name: Lama Alsanie Email: lamalsanie6@hotmail.com Phone number: 0556165555 Password: Lama165@ Confirm Password: Lama165@	8
T03 : Name: Lama Alsanie Email: lamalsanie6@hotmail.com Phone number: 0556165555 Password: Lama1655@ Confirm Password: Lama1655@	9
Test Result: Test cases with invalid length of 7 pass successfully by displaying an error message to the user. Whereas a test case with valid length of 8 and above passes successfully and resumes the registration process.	

Table 49 Password Field Boundary Test

20.3.1.1 Log in equivalence classes and boundary value analysis

a. Login Equivalence Classes

Input Conditions	Valid equivalence classes	Invalid equivalence classes
Email	- Email name ∈ [a ... z] **(1)** - Email name ∈ [A ... Z] **(2)** - Email name ∈ [0 ... 9] **(3)**	- Email name is absent**(6)** - Domainnamehasspecial character **(10)**

Emailname@Domain.com	- Email name has special characters**(4)** - Email name is present **(5)** - Domain name∈ [a ... z] **(7)** - Domain name∈ [A ... Z] **(8)** - Domain name∈[0 ... 9] **(9)** - Domain name is present **(11)** - If email nameexists. Then, it's must be followed by @ -character-**(13)** - Correct sequence.**(15)**	- Domain name is absent**(12)** - If email nameexists. Then, it's not followed by @ -character-**(14)** - Incorrect sequence.**(16)**
Password	- password ∈ [a ... z] **(17)** - password ∈ [A ... Z] **(18)** - password∈ [0 ... 9] **(20)** - password has special characters**(21)** - passwordlength=> 8 **(22)** - passwordis present **(24)** - email and password match an existing user's information. **(26)** - email and password fields are both filled. **(28)**	- password has a non-English character. **(19)** - passwordlength< 8 character **(23)** - passwordisabsent **(25)** - email and password do not match an existing user's information. **(27)** - Input fields are missing email or password or both. **(29)**

Table 50 log in equivalence classes

b. Identify Use Cases for Valid equivalence classes

Use-Case Data	Covers Equivalence Classes
T01 :Email : reemaldosari13@gmail.com Password : Reem12345&	(1),(2),(3),(4),(5),(7),(8),(9),(11),(13),(15),(17),(18),(20),(21),(22),(24),(26),(28)

Table 51 log in valid equivalence class

c. Identify Use Cases for Invalid equivalence classes

Use-Case Data	Covers Equivalence Classes
T02 :Email : @reemgmail.com Password : Reem12345&	(6)
T03 :Email : Reem@$$gm.com Password : Reem12345&	(10)
T04 :Email : Reem@	(12)

Password : Reem12345&	
T05: Email: Reem&Reem.com Password: Reem12345&	(14)
T06 :Email : gmail.com @reem Password : Raghad1@1	(16)
T07 :Email : Reemaldosari13@gmail.com Password : @ـ133	(19)
T08 :Email : Reemaldosari13@gmail.com Password : ree11	(23)
T09 :Email : Reemaldosari13@gmail.com Password :	(25)
T10 :Email : Reemaldosari13@gmail.com Password : reem12345$	(27)
T11 :Email : Password : reem12345&	(29)

Table 52 log in invalid equivalence class

d. Log in boundary value analysis

Input condition	Constraint	Boundary Value
Password	Length>=8	[7] [8] [9]

Table 52 log in boundary value analysis

1. Password:

Test Value	Password Input Length
T18 :Email : Reem@gmail.com Password : reemah0	7
T19 :Email : Reem@gmail.com Password : reeeema@	8
T20 :Email : Reem@gmail.com Password : Reemah12@	9

Table 53 Password Field Boundary Test

a. Export bill Equivalence Classes

Input Condition	Valid class	Invalid Class

Delivery_price	- Delivery_price ∈ [0-9]. **(1)**	- Delivery_price ∈ [A-Z/a-z]. **(2)** - Delivery_price ∈ [!#$%&'*+- /=?^_{|}~]. **(3)** - Empty **(4)**
Order_price	- Order_price ∈ [0-9]. **(5)**	- Order_price ∈ [A-Z/a-z]. **(6)** - Order_price ∈ [!#$%&'*+- /=?^_{|}~]. **(7)** - Empty **(8)**
TotalPrice	- Totalprice ∈ [0-9]. **(9)**	- Totalprice ∈ [A-Z/a-z]. **(10)** - Totalprice ∈ [!#$%&'*+- /=?^_{|}~]. **(11)** - Empty **(12)**
Image	- Image ∈ [a-z]. **(13)** - Image ∈ [A-Z]. **(14)** - Image ∈ [0-9]. **(15)** - Image ∈ [!#$%&'*+-/=?^_{|}~].**(16)**	- Empty **(17)**
Form	- Input fields are filled. **(18)**	- Incomplete fields. **(19)** - Empty form. **(20)**

Table 54 Export bill equivalence classes

b. Identify Use Cases for Valid equivalence classes

Use-Case Data	Covers Equivalence Classes
T01:deliveryPrice:30 orderPrice:40 totalPrice:70 image:Image0$.png	**(1,5,9,13,14,15,18)**

Table 55 Export bill valid equivalence classes

c. Identify Use Cases for Invalid equivalence classes

Use-Case Data	Covers Equivalence Classes
T02: deliveryPrice: three riyals orderPrice: 40 totalPrice: 70 image: Image0$.png	**(2)**
T03: deliveryPrice: 40$ orderPrice: 40 totalPrice: 70 image:Image0$.png	**(3)**
T04: deliveryPrice:Null orderPrice:40 totalPrice:70 image:Image0$.png	**(4)**
T05: deliveryPrice:30 orderPrice:threeriyals totalPrice:70 image:Image0$.png	**(6)**
T06:deliveryPrice:30 orderPrice:40$ totalPrice:70 Image:Image0$.png	**(7)**
T07:deliveryPrice:30 orderPrice:Null totalPrice:70 image:Image0$.png	**(8)**
T08:deliveryPrice:30 orderPrice:40 totalPrice:1riyal image:Image0$.png	**(10)**
T09:deliveryPrice:30 orderPrice:40 totalPrice:70$ image:Image0$.png	**(11)**
T10:deliveryPrice:30 orderPrice:40 totalPrice:Null image:Image0$.png	**(12)**

T11:deliveryPrice:30 orderPrice:40 totalPrice:70 image:Null	**(17)**
T12:deliveryPrice:30$ orderPrice:40$ totalPrice:70 image:Image0$.png	**(19)**
T13:deliveryPrice:Null orderPrice:Null totalPrice:Null image:Null	**(20)**

Table 56 Export bill invalid equivalence classes

d. Export bill boundary value analysis

Input condition	Constraint	Boundary Value
DeliveryPrice	500>=Length>=0	[-1] [30] [1000]
OrderPrice	500>=Length>=0	[-1] [30] [1000]

Table 57 Expert bill boundary value analysis table

1. Delivery Price:

Test Value	Boundary Value
T14: deliveryPrice: -1 orderPrice: 40 totalPrice:39 image: image.png	-1
T15: deliveryPrice:30 orderPrice:40 totalPrice:70 image:image.png	30
T16: deliveryPrice1001 orderPrice40 totalPrice:1041 Image:image.png	1001
Test Result: Test cases with invalid length of -1 and 500 for the Delivery price pass successfully by displaying an error message to the user. Whereas a test case with a valid length between the range 500>=Length>=0 passes successfully and resumes the export bill process.	

Table 58 Export bill boundary analysis for delivery price

1. Order Price:

Test Value	Boundary Value
T17: deliveryPrice: 40 orderPrice: -1 TotalPrice: 39 Image:image.png	-1
T18: deliveryPrice: 40 orderPrice: 30 totalPrice: 70 -image.png	30
T19: deliveryPrice:40 orderPrice:1001 totalPrice:1041 image: image.png	1001
Test Result: Test cases with invalid length of -1 and 500 for the Order price pass successfully by displaying an error message to the user. Whereas a test case with a valid length between the range 500>=Length>=0 passes successfully and resumes the export bill process.	

Table 59 Export bill boundary analysis for order price

20.3.2 Scenario based testing

Scenario-based testing is one method of documenting software specifications and requirements for the project. Scenario-based testing is used for writing tests for individual user scenario, which would check their work. Scenarios concentrate on the principal objectives and requirements. If the scenario runs from start to finish, then it passes.[36]

20.3.2.1 Change order status

Use case name	Change order status
Actor	Courier
Goal	The courier changes order status by notifying the customer
Participants	Customer
Preconditions	• Courier log in • Customer accepted offer • Customer Pay bill
Steps	1- The order will be shown in “Order tap”. 2- Order first status will be “Go to Pick up”. 3- Courier click on the status to move to second status “Pickup order”. 4- Courier click on the status to move to third status “I have arrived”. 5- Courier should be export bill 6- Courier upload the picture of the bill 7- Customer should pay the bill

	8- Courier click on the status to move to last one status “Deliver now” 9- The order will be moved to “History tap”.
Alternatives	In step 2,3 courier can cancel the order 1.Courier click on the cancel button. 2.The status will change to “canceled”. 3.The order will be in “history tap”.
Post-condition	The order status changed successfully

Table 60 change order status use case

Scenario Graph:

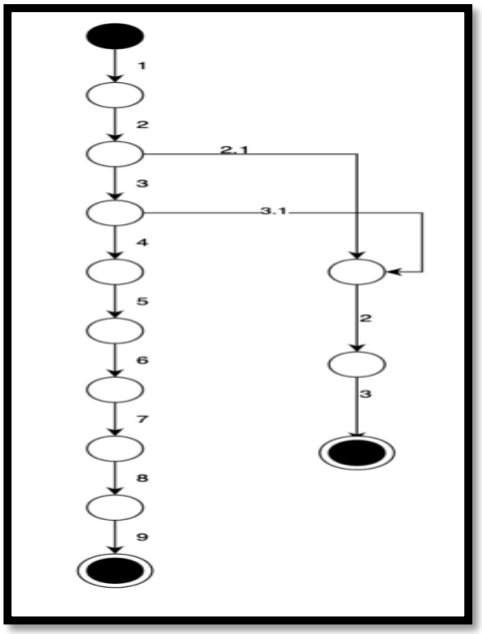


Figure 82 change order status scenario flow graph

Scenario id	Events	Description
1	1-2-3-4-5-6-7-8-9	**Normal Scenario** The order will be in “order tap" the courier change the status of the order from” Go to pick up”, click to be “Pickup order”, click to be “I have arrived”, Courier Expert bill, Customer pay bill, Courier click to be “Deliver now” then the order will be in “History tap”.
2	1-2-3-3.1-3.2-3.3	**Courier cancel order** The order will be in “order tap”, the courier change the status of the order from” Go to pick up”, click to be “Pickup order”, courier decide click on “cancel” then the order will be in “History tap”.

Table 61 change order status possible scenarios

	Test case id	TC1
	Description	Test case corresponding to main course of events.
	Goal	Change order status.
	Scenario Reference	1
	Pass Criteria	The order status is updated to deliver now.
	Set-up	-Courier log in -Courier send an offer -Customer paid bill
	Course of Events	
#	**External Event**	**Reaction**
1	-	System display order in current order page the status will be “Go to pick up”
2	Courier click on “Go to Pick up”.	System updates trip status to “Pick up order”.
3	Courier click on “Pickup order”.	System updates trip status to “I have arrived”.

4	Courier click on “I have arrived”.	-
5	Courier should be export bill	-
6	Courier upload the picture of the bill	-
7	Customer should pay the bill	System updates trip status to “Deliver now”.
8	Courier click on “Deliver now”	The order will be moved to “History tap”.

Test Result: Test passed successfully, and order has been completed

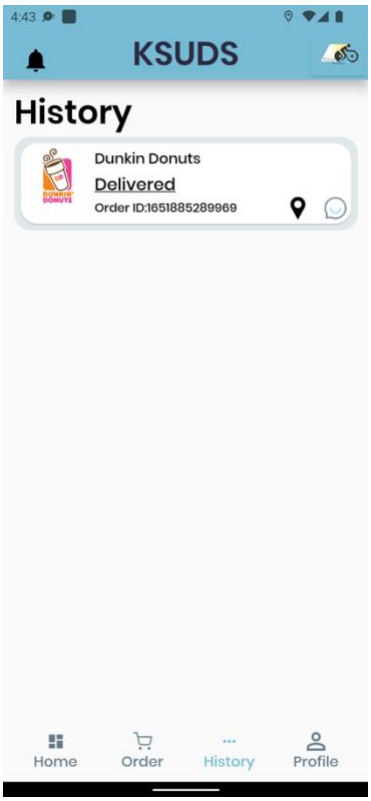


Table 62 change order status Test Case 1

	Test case id	**TC2**
	Description	Test case corresponding to cancellation of order.
	Goal	Canceled an order
	Scenario Reference	2

	Pass Criteria	The order status is updated to a cancelled.	
	Set-up	-Courier log in -Courier send an offer	
	Course of Events		
#	**External Event**	**Reaction**	
1	-	System display order in current order page the status will be “Go to pick up”	
2	Courier click on “Go to Pick up”.	System updates trip status to “Pick up order”.	
3	Courier click on “Cancel”.	The order will be moved to “History tap”.	

Test Result: Test passed successfully, and order status is updated to active on the “canceled”.

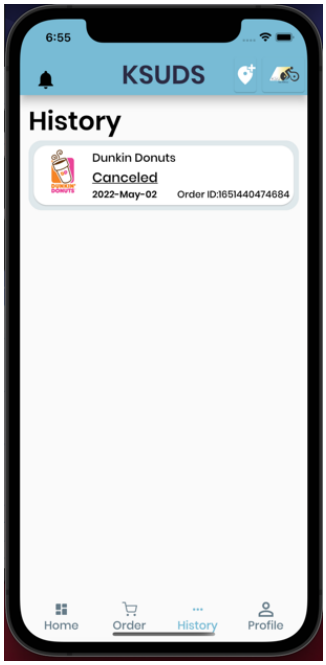


Figure 84 Actual Canceled UI

Table 63 change order status Test Case 2

20.3.2.2 Select an offer

Use case name	Select an offer
Actor	Courier
Goal	The courier sends an offer to customer.
Participants	Customer
Preconditions	• Courier log in • Order has been sent from the customer

	successfully
Steps	1- Order shown to the courier in the home page with (order id, place name). 2- Courier click on the order it will be shown more detail (deliver to, time). 3- Courier click on “Select order”. 4- System will display Send offer box. 5- Courier enter the price. 6- System will display offer sent
Alternatives	In step 2: 1-if order sent before one hour and no one select it, it will be expired.
Post-condition	Courier can select an offer successfully

Table 64 Select an offer use case

Scenario Graph:

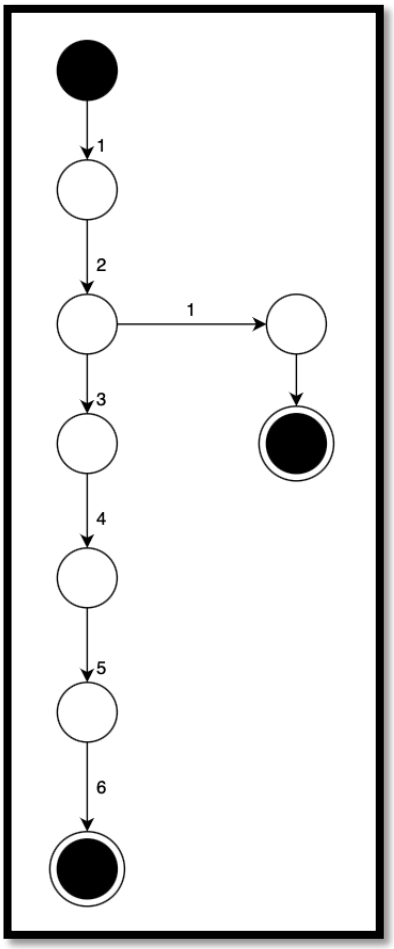


Figure 85 Select an offer scenario flow graph

Scenario id	Events	Description
1	1-2-3-4-5-6	**Normal flow:** the customer sends an order and the courier sends the offer within an hour
2	1-2.1	**Courier ignored the order:** The customer sends an order, and the courier does not reply for more than one hour.

Table 65 select an offer possible scenario

	Test case id	TC1	
	Description	Test case corresponding to send an offer.	
	Goal	Courier sent an offer	
	Scenario Reference	1	
	Pass Criteria	Courier sent an offer successfully	
	Set-up	-Courier log in	
	Course of Events		
#	**External Event**	**Reaction**	
1	-	System displays all orders for the courier.	
2	Courier clicks on the order.	System display box sent an offer.	
3	Courier enter the price delivery.	System display “Offer sent successfully”	
Test result: Test passed successfully, and offer has been sent.			

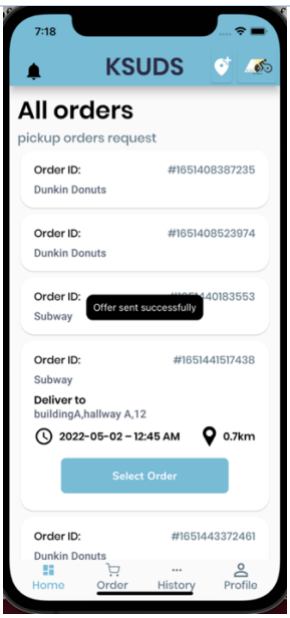


Figure 86 Actual Sent an offer UI

Table 66 Select an offer Test Case 1

	Test case id	**TC2**
	Description	Test case corresponding to expired order.
	Goal	Order expired.
	Scenario Reference	2
	Pass Criteria	Order has expired.
	Set-up	-Courier log in
	Course of Events	
#	**External Event**	**Reaction**
1	-	System displays all orders for the courier.
2	Courier ignores order	System will delete the offer from home page and make the status "Expired"
Test result: Test passed successfully, and order has been delete**d** it.		

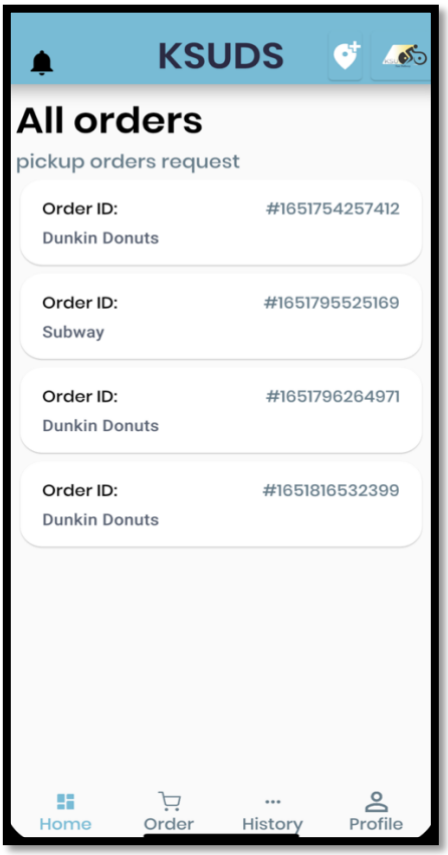


Figure 87 Actual Sent an offer UI

Table 67 Select an offer Test Case 2

20.4 Test cases:

This section contains test cases details that cover some of the functional requirements.

Filter restaurants:

Test Case ID	TC#1
Description	Test Filter restaurants based on “Food” category.

Steps	**Test Data**	**Expected Results**	**Actual Results**	**Test Results**

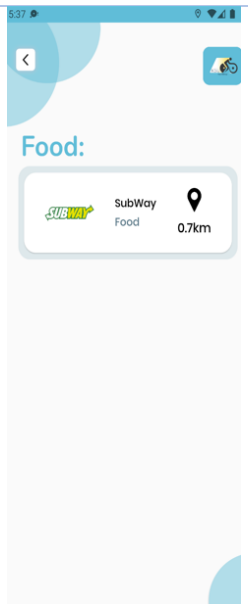
1- Customer selects “Home” 2- System displays all restaurants regardless of their categories. 3- Child select "Food" Category. 4- The system displays restaurants with "Food” category only.	None	The system displays the selected category restaurants only.		Pass

Table 68 filter restaurant function test case.

Test Case ID	TC#2
Description	Test select an order.

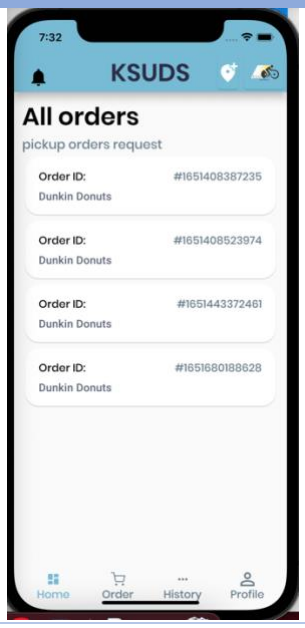
Steps	**Test Data**	**Expected Results**	**Actual Results**	**Test Results**
1- Courier selects “Home” 2- System displays all available orders. 3- Courier selects the desired order. 4- The system displays the chat.	None	The system displays the selected order.		Pass

Table 69 select an order function test case

Test Case ID	TC#3
Description	Test view earnings

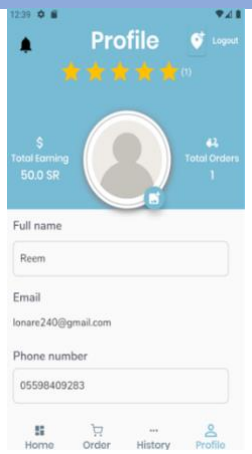
Steps	Test Data	Expected Results	Actual Results	Test Results
1- Courier selects “profile” tab 2- System displays the total earning for the courier.	deliveryPrice=30SR orderPrice=20 SR	The system displays the Total earning which is equal to totalPrice=50 SR in the profile page for the courier		Pass

Table 69 view earing function test case.

Test Case ID	TC#4
Description	Test view rating for the courier

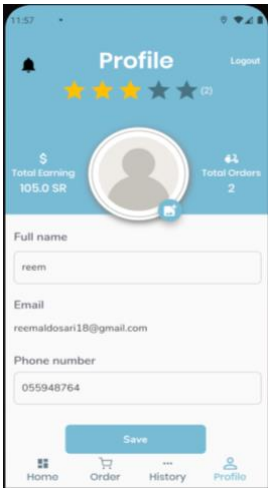
Steps	Test Data	Expected Results	Actual Results	Test Results
1- Courier selects “profile” tab 2- System displays the rating for the courier.	Rating from the first customer: 4 Rating from second customer:2	The system displays the rating for the courier which is equal to 3 in the profile page		Pass

Table 70 view rating function test case.

Test Case ID	TC#5
Description	Test view rating for the customer

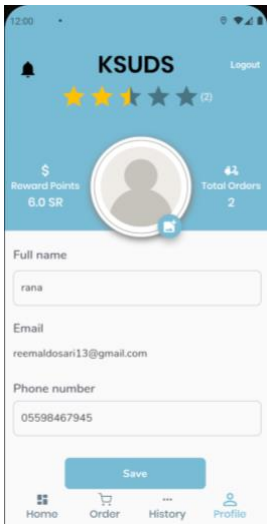
Steps	Test Data	Expected Results	Actual Results	Test Results
1- Customer selects “profile” tab 2- System displays the rating for the Customer.	Rating from the first courier: 1 Rating from second courier:4	The system displays the rating for the customer which is equal to 2.5 in the profile page		Pass

Table 71 view rating function test case

Test Case ID	TC#6
Description	Test report an issue for the customer

Steps	Test Data	Expected Results	Actual Results	Test Results
1- Customer selects “chat” button 2- Customer selects “question mark” button 3- Customer write the title, and description of his/her problem 4- Customer clicks ”send” button	Title= didn’t receive my order Description= didn’t receive my order until 2 hours	The system directs the customer to the email of the KSUDS admin with the title and description of his/her problem		Pass

Table 72 report an issue function test case.

Test Case ID	TC#7
Description	Test report an issue for the courier

Steps	**Test Data**	**Expected Results**	**Actual Results**	**Test Results**
1- Courier selects “chat” button 2- Courier selects “question mark” button 3- Courier write the title, and description of his/her problem 4- Courier clicks” send” button	Title= can’t view all the orders Description= I can’t view all the orders from the customer	The system directs the courier to the email of the KSUDS admin with the title and description of his/her problem		Pass

Table 73 report an issue function test case.

Test Case ID	TC#8

Description	Test pay order’s bill

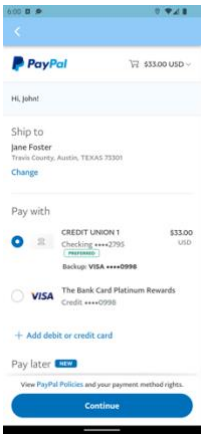
Steps	Test Data	Expected Results	Actual Results	Test Results
1- Customer selects “chat” button 2- Customer selects “Proceed to payment” button 3- Customer selects “PayPal” method. 4- Customer enters email and password for his/her account. 5- Customer selects the card. 6-Customer clicks “Continue”.	Email: sb-d2wbi2746133@personal.example.com Password: test1234A	The system directs the customer to PayPal page, and customer selects the card and pay.		Pass

Table 74 pay order bill function test case.

Test Case ID	TC#9
Description	Contact courier

Steps	Test Data	Expected Results	Actual Results	Test Results

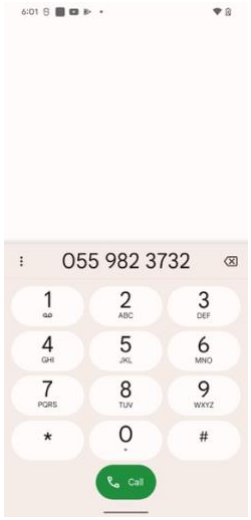
1- Customer selects “order” tab. 2- Customer press the order he/she wants then enter to the chat. 2- Customer press the icon of the call.	Active order.	The system directs the customer to a real call.		Pass

Table 75 call courier function test case.

Test Case ID	TC#10			
Description	Test view map			
Steps	**Test Data**	**Expected Results**	**Actual Results**	**Test Results**
1- Customer/courier selects “order” tab. 2- Customer/courier presses the icon of the map or the active order.	Active order.	The system directs the customer/courier to the map page.		Pass

Table 76 view map function test case.

20.5 Non-functional Testing

The non-functional requirements testing is conducted to ensure that the system operates as specified in the requirements. The effectiveness and performance of the system is evaluated in the following non-functional testing aspects.

20.5.1 Integrity, access control

KSUDS sends an email to the registered user before login to make sure that user has entered the right email.

20.5.2 Security

KSUDS contains restrictions for the user's password to elevate the level of security in the system. The user's password should contain at least eight characters containing numbers, small letters, capital letters, and special characters.

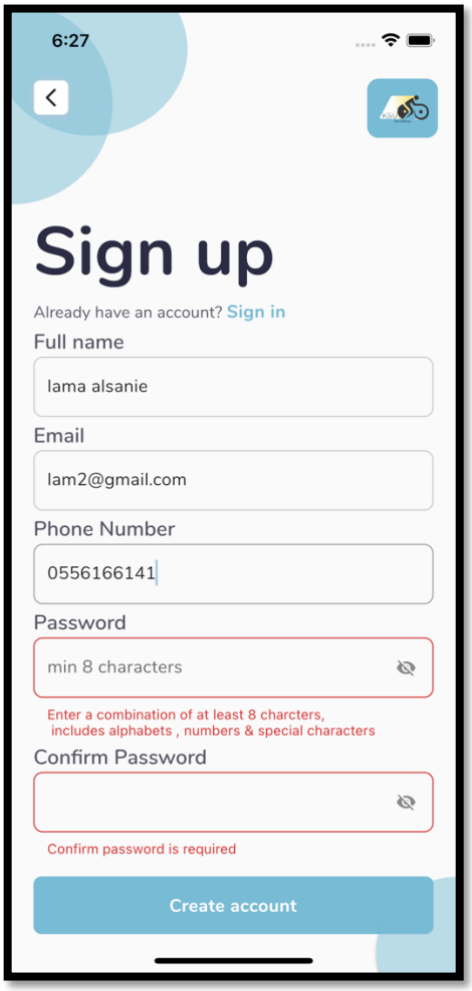


Figure 88 Actual Sign up security test UI

```
16
17  static String? validatePassword(String value) {
18    if (value.isEmpty)
19      return 'Enter a combination of at least 8 charcters, \n includes alphabets , numbers & special characters';
20    if (value.length < 8) {
21      return "Enter a combination of at least 8 charcters, \n includes alphabets , numbers & special characters";
22    }
23    if (!RegExp(r"^(?=.*?[0-9])").hasMatch(value)) {
24      return 'Enter a combination of at least 8 charcters, \n includes alphabets , numbers & special characters';
25    }
26    if (!RegExp(r'^(?=.*?[!@#\$&*~])').hasMatch(value.trim())) {
27      return 'Enter a combination of at least 8 charcters, \n includes alphabets , numbers & special characters';
28    }
29    return null;
30  }
31
```

Figure 89 code security for the sign up

20.5.3 Performance

The performance of the system was measured using Firebase Performance. The figure below shows the app start, app-in-foreground, and app-in-background performance data, where the:

- [App start] trace — A trace that measures the time between when the user opens the app and when the app is responsive
- [App-in-foreground] trace — A trace that measures the time when the app is running in the foreground and available to the user
- [App-in-background] trace — A trace that measures the time when the app is running in the background

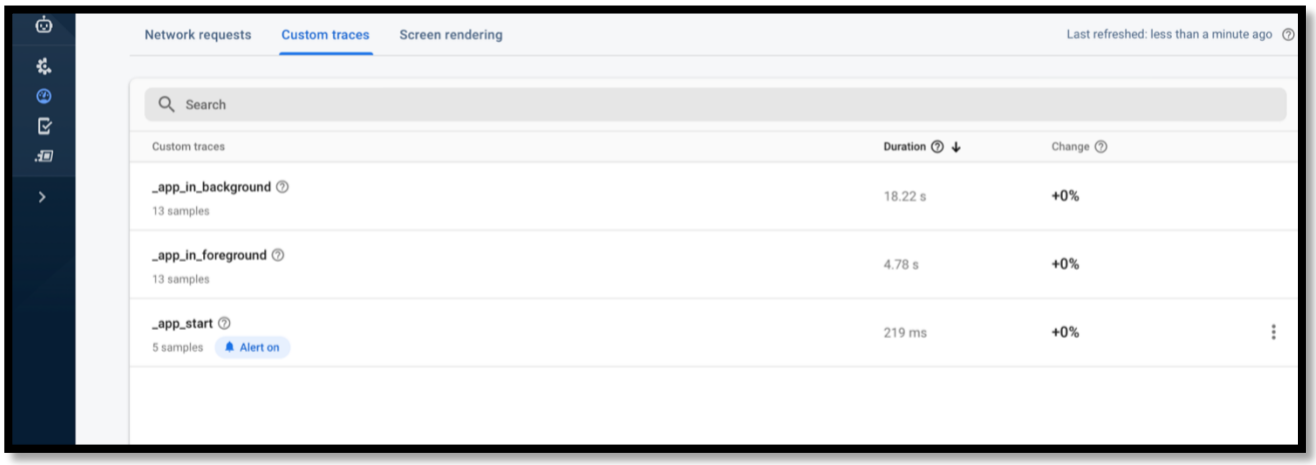


Figure 90 Performance Data Duration

20.5.4 Usability, software ease of use

20.5.4.1 Usability Testing

Usability testing is a technical method for determining how easy a system is to use and identifying any usability issues or design flaws.

Set Up:

For usability testing, the team asks participants to perform specific tasks. Participants in the study are instructed to think aloud while using the app. Simply so individuals are verbalizing their thoughts as they navigate the user interface. The sessions are documented and saved on Google Drive. they can be found in.

Participant Profile

This section contains information about the people who took part in usability testing. The participants come from various fields and range in age from 19 to 40.To finish, each participant is given a set of tasks.

#	Participant name	Age	Gender
1	Joud aloqla	22	Female
2	Shaden alfawzan	22	Female
3	Rahaf alshali	22	Female
4	Aljazi bin osseil	24	Female
5	Aljouhara bin osseil	28	Female
6	Ghyda alsuwailem	24	Female
7	Sara alsuwailem	20	Female
8	Seba aloin	21	Female
9	Rana alhajri	22	Female
10	Noura aldosari	22	Female
Mean		22.7	
Median		22	

Table 77 Participant Profiles (customer)

#	Participant name	Age	Gender
1	Moneera alsuwailem	27	Female
2	Abeer aba alkhail	22	Female
3	Yara almalik	23	Female
4	Shaden alsyari	22	Female
5	Sara almehoum	23	Female
6	Raghad almutairi	21	Female
7	Rahaf alenazi	22	Female
8	Rabab alqausi	22	Female
9	Sara althunaian	22	Female
10	Lamees alshahrani	22	Female
Mean	22.4		
Median	22		

Table 78 Participant Profiles (courier)

Usability Study Design

In this section. We intended to offer each participant a series of activities that were not fully explained in order to give them the opportunity to perform the tasks successfully without assistance in this section.
A list of the steps that were followed is as follows:

1. Describing the idea of KSUDS and its goals.
2. Explaining the purpose of the usability testing and what is expected.
3. Providing sample data to use during the testing and instructions.
4. Giving each participant a device where they can test the application.
5. Asking to complete the following task:

#	Task
1	Make an order
2	Track Order
3	Contact courier
4	Report an issue
5	View history
6	Log out

Table 79 Usability Testing Tasks (customer)

6. Requesting that each participant complete the following post-testing questionnaire using the google form in Appendix I. Participants' replies are gauged using a Likert scale, which ranges from agree to disagree.

#	Questions
1	I would like to use this application frequently.
2	The application was easy to use.
3	The application’s functions were unclear.
4	The application’s functions were unnecessarily difficult.
5	I need a technical support to be able to use the application.
6	The application UI was inconsistent.
7	I would recommend naive people to use the application.
8	Using the application was frustrating.
9	The order process was clear and smooth.
10	I had no issues contacting the courier.

Table 78 Usability Questionnaire (customer)

#	Task
1	View active order
2	Change order status
3	Contact customer
4	Report an issue
5	View history
6	Log out

Table 79 Usability Testing Tasks (courier)

#	Questions
1	I would like to use this application frequently.
2	The application was easy to use.
3	The application's functions were unclear.
4	The application's functions were unnecessarily difficult.
5	I need a technical support to be able to use the application.
6	The application UI was inconsistent.
7	I would recommend naive people to use the application.
8	Using the application was frustrating.
9	The report an issue process was clear and smooth.
10	I had no issues contacting the customer.

Table 80 Usability Questionnaire (courier)

Usability Results

The performance and feedback were measured using the following metrices: **Efficiency, satisfaction, and effectiveness.** The outcomes of the Google form, as well as observer thoughts and feedback on the application, were gathered, and action was taken. The results are displayed in the table below.

Efficiency:

Measures how long it takes a user to finish a job. Time on Task = (task completion time - task start time) The observers used a rudimentary timer tool to time each user's time from start to finish. e.g.: phone timer then the average was calculated based on the time taken on each task. (2+2+3+4+2+2.6+3+3+2.1+2) = 25.7mins / 10 users = 2.57 minutes per user.

This step measured the efficiency of reporting an issue the average time that took participants to complete the task which resulted in 2.57 minutes, The time it took each user to finish the task is indicated on the Y axis, and the user is indicated on the X axis in the bar graph below.

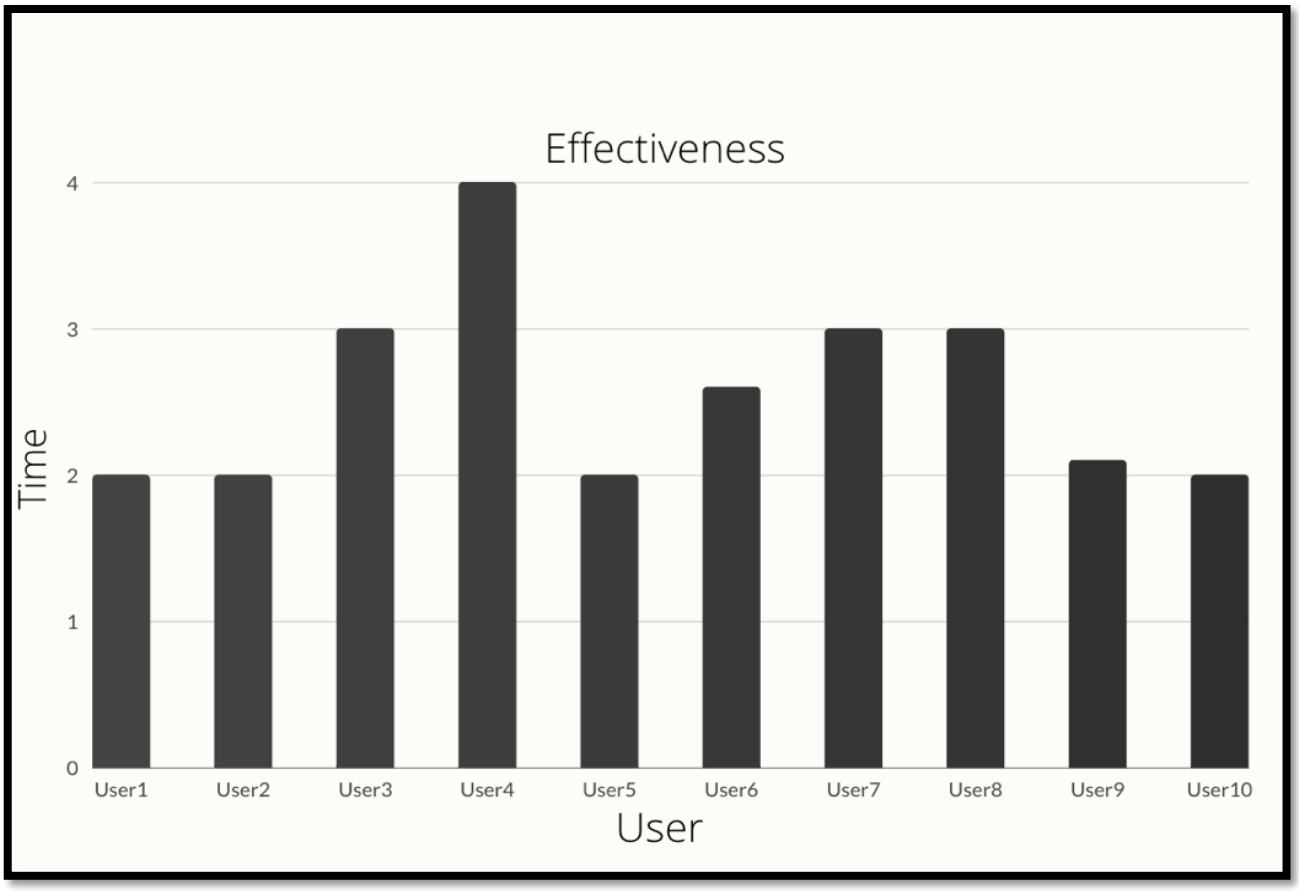


Figure 91 Effectiveness Bar Chart

Satisfaction:

The overall impression and judgment of the software's usability are measured. After completing the exercises, participants are given a questionnaire to gauge their satisfaction. The following pie chart illustrates the satisfaction of participants for the courier side of our application and indicates that 80% of people would agree to use the application, 10% had Neutral responses and 10% would disagree to use the application.

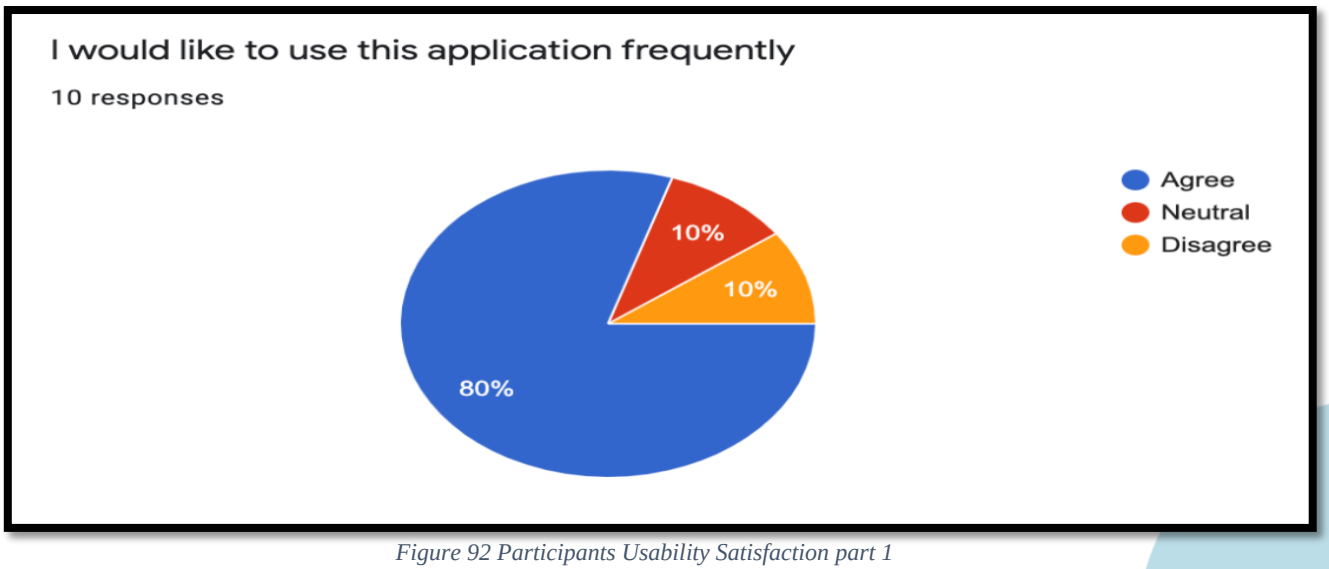


Figure 92 Participants Usability Satisfaction part 1

And the following pie chart illustrates the satisfaction of participants for the customer side of our application and indicates that 90% of people would agree to use the application, 10% had Neutral responses.

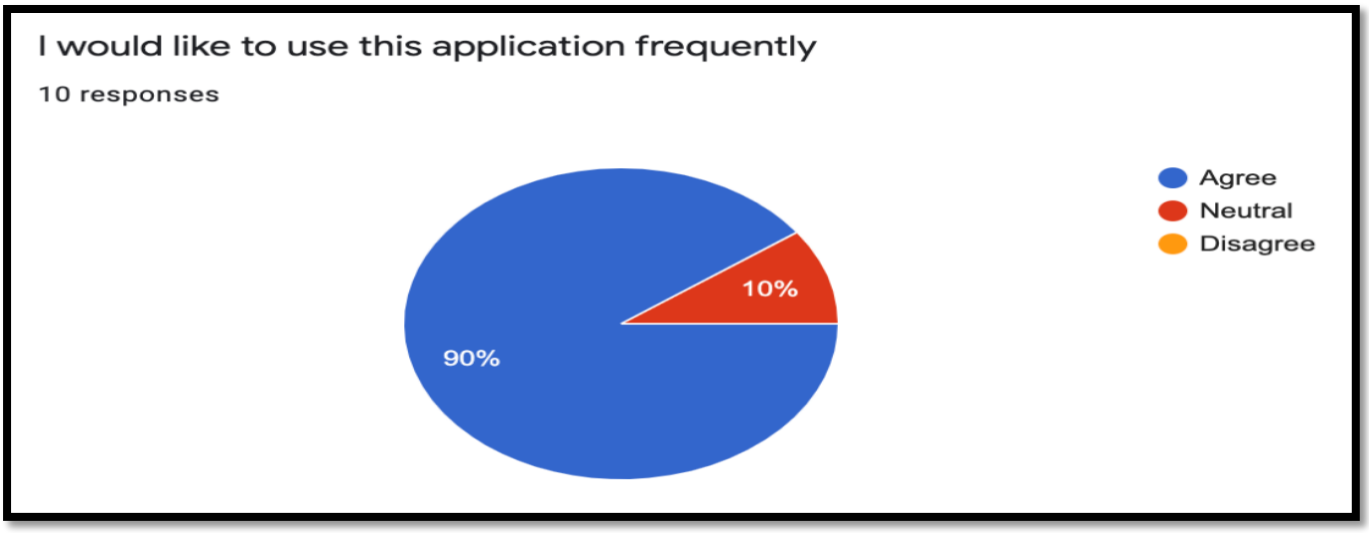


Figure 93 Participants Usability Satisfaction part 2

Effectiveness:
The rate at which a task is completed successfully. The number of successfully completed tasks is estimated based on task success, which was determined using overserves observation, in which each observer had a checklist of each user and checked whenever the step was completed successfully.

Task	User1	User2	User3	User4	User5	User6	User7	User8	User9	User10
Make an order	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Track Order	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Contact courier	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Report an issue	✓	✓	✗	✓	✓	✓	✓	✗	✓	✗
View history	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Log out	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 81 Observers Checklist Of User's Tasks (customer)

Task	User1	User2	User3	User4	User5	User6	User7	User8	User9	User10
View active order	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Change order status	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Contact customer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Report an issue	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
View history	✓	✗	✗	✓	✓	✓	✗	✓	✓	✓
Log out	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 82 Observers Checklist Of User's Tasks (courier)

Post Usability Test Results Feedback

The issues, comments, and feedback observed and gathered from participants while using the program, as well as the actions performed in response to each feedback, are listed in the table below.

Location	Category	Feedback	Action
View history	UI	Participants were confused in where to find the history page	The history page was moved to the menu bar
Report issue	Functionality	The participants were asked to report an issue and after writing the issue the application took them to the email page and they had to write the issue all over again	The report an issue functionality was fixed and users don't have to rewrite the issue all over again

Table 83 Post Usability Test Results And Feedback

21. Limitation of the system

KSUDS was successfully developed and tested. However, there were some constraints in the **development and testing process.**

One of the constraints was integrating Apple pay and Stc Pay, since it requires payment, uploading an Apple Pay certificate to the Square Developer Portal, so it wasn't possible to be integrated.

Another challenge was updating courier's location every second in the app, which means a background service it wasn't an easy method to be found, as we found one method that wasn't compatible with our code, so we found an easier way where courier's location is updating every 5 seconds when courier enters the application.

Moreover, at the present time the customer is able to have one order at a time, and the courier is allowed to deliver one order at a time, since having multiple orders at a time mainly crashing the location and chatting code, due to the project duration we couldn't find a way to make it possible. Hoping to do it in the future.

Finally, the team has decided to cut out these functions due to the inability to use libraries, as the team wants to deliver the project on time.

22. Conclusion and Future Work

The KSUDS application was built to help students and staff at King Saud University to order anything from anywhere on campus through a flexible system "KSUDS". Customers also have the opportunity to earn points for discounts on future orders. King Saud University students can even send gifts using the KSUDS application.

To realize the project's vision, a team effort was required, starting with a detailed review that included input from our clients, then modeling the findings to stabilize the system and provide a common ground for development and testing. The team members created the system with the goal of not only hiring all of their prior knowledge, but also exploring, learning, and searching to complete the tasks listed above, which were distributed among the team members to achieve the final desired result. However, this has been a valuable chance for the team to learn about patience, resilience, and struggle.

The team members sincerely hope that KSUDS can achieve the following features in the future:
-Adding admin user and features.
- Enable one Customer to have Multiple orders at a time.
- Enable one Courier to deliver Multiple orders at a time.
- Support Arabic language so it would have localization as a non-functional requirement.
-Update courier location in the background service.
-Develop KSUDS website.
-Integrate Apple Pay for an enhanced payment user experience.
-Add more KSU restaurants, coffee shops, and bookstores in the application.
-Collaborate with restaurants, coffee shops, and bookstores to add their items in the application.

23. Reference

[1] Hirschberg, C., Rajko, A., Schumacher, T. and Wrulich, M., 2016. *The changing market for food delivery*. [online] mckinsey. Available at: <https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/the-changing-market-for-food-delivery> [Accessed 21 September 2021].

[2] Nordic APIs. 2021. *The Importance of APIs for Payment Platforms | Nordic APIs |*. [online] Available at: <https://nordicapis.com/the-importance-of-apis-for-payment-platforms/> [Accessed 14 September 2021].

[3] En.wikipedia.org. 2021. *Google maps - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Apple_Maps> [Accessed 14 September 2021].

[4] iThink Logistics | Blogs. 2021. *How does the courier tracking system work? | iThink Logistics | Blogs*. [online] Available at: <https://ithinklogistics.com/blog/how-does-the-courier-tracking-system-work/amp/> [Accessed 14 September 2021].

[5] Mrsool.co. 2021. *Mrsool*. [online] Available at: <https://mrsool.co/> [Accessed 13 September 2021].

[6]Toyou.io. 2021. *ToYou*. [online] Available at: <https://toyou.io/ar> [Accessed 13 September 2021].

[7] Wssel.com. 2021. *Wssel*. [online] Available at: <https://www.wssel.com/> [Accessed 13 September 2021].

[8] Nana.sa. 2021. تطبيق نعناع - شراء وتوصيل المقاضي. [online] Available at: <https://www.nana.sa/ar/> [Accessed 13 September 2021].

[9] Jahez.net. 2021. تطبيق جاهز لتوصيل الطلبات. [online] Available at: <https://www.jahez.net/> [Accessed 13 September 2021].

[10] Pik.sa. 2021. *Pik*. [online] Available at: <https://pik.sa/pages/home> [Accessed 13 September 2021].

[11] Lvivity. 2021. *Waterfall Methodology: Advantages and Disadvantages, And When to Use It?*. [online] Available at: <https://lvivity.com/waterfall-model> [Accessed 16 September 2021].

[12] App.teamgantt.com. 2021. *TeamGantt*. [online] Available at: <https://app.teamgantt.com/> [Accessed 21 September 2021].

[13] Sharma, L., 2021. *What is Configuration Management System?*. [online] TOOLSQA. Available at: <https://www.toolsqa.com/software-testing/configuration-management/.> [Accessed 17 September 2021].

[14] Wahlbeck, M., 2016. *iOS 10 & Flutter 3: From Beginner to Paid Professiona*. [online] udemy. Available at: <https://www.udemy.com/course/devslopes-ios10/> [Accessed 21 September 2021].

[15] SearchSoftwareQuality. 2021. *What is a Use Case?*. [online] Available at: <https://searchsoftwarequality.techtarget.com/definition/use-case> [Accessed 28 November 2021].

[16] En.wikipedia.org. 2021. *Use case diagram - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Use_case_diagram> [Accessed 28 November 2021].

[17] Guru99. 2021. *Interaction, Collaboration & Sequence Diagrams with Examples*. [online] Available at: <https://www.guru99.com/interaction-collaboration-sequence-diagrams-examples.html> [Accessed 28 November 2021].

[18] Youtube.com. 2021. [online] Available at: <https://www.youtube.com/watch?v=_Mzi1rYtI5U> [Accessed 28 November 2021].

[19] Rationalrose.com. 2021. *View Of Participating Classes*. [online] Available at: <http://www.rationalrose.com/scripts/vopc.htm> [Accessed 28 November 2021].

[20] B. Golden, "lix.polytechnique.fr," [Online]. Available: https://www.lix.polytechnique.fr/~golden/systems_architecture.html. [Accessed 25 November 2021]

[21]"tutorialspoint,"[Online].Available:https://www.tutorialspoint.com/software_architecture_design/introduction.htm#:~:text. [Accessed 25 November 2021]

[22]"altexsoft.com,"[Online].Available:https://www.altexsoft.com/blog/non-functional-requirements/. [Accessed 25 November 2021].

[23] "interaction-design.org," [Online]. Available: https://www.interaction-design.org/literature/topics/usability#:~:text=Usability%20is%20a%20measure%20of,deliverable%E2%80%94to%20ensure%20maximum%20usability. [Accessed 25 November 2021].

[24] Security Boulevard. 2021. *Building Non-Functional Requirements Framework - RequirementsCategories*.[online]Availableat: <https://securityboulevard.com/2020/12/building-non-functional-requirements-framework-requirements-categories/> [Accessed 26 November 2021].

[25] "study.com," [Online]. Available: https://study.com/academy/lesson/software-architecture-styles-patterns-components.html. [Accessed 25 November 2021].

[26] Tutorialspoint.com. 2021. *UML - Component Diagrams*. [online] Available at: <https://www.tutorialspoint.com/uml/uml_component_diagram.htm> [Accessed 28 November 2021].

[27] Firebase. 2021. *Cloud Firestore | Store and sync app data at global scale | Firebase*. [online] Available at: <https://firebase.google.com/products/firestore> [Accessed 4 December 2021].

[28] Investopedia. 2021. *What Is an Algorithm?*. [online] Available at: <https://www.investopedia.com/terms/a/algorithm.asp> [Accessed 4 December 2021].

[29] En.wikipedia.org. 2021. *Dijkstra's algorithm - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm> [Accessed 9 December 2021].

[30] Cc.gatech.edu. 2021. *Pseudo-Code for Shortest Path Algorithm*. [online] Available at: <https://www.cc.gatech.edu/aimosaic/students/Psych-students/Shikano/Projects/AlgoNet/mst/code.html> [Accessed 10 December 2021].

[31] 2021. [online] Available at: <https://creately.com/blog/diagrams/deployment-diagram-tutorial/> [Accessed 4 December 2021]

[32] Guru99. 2022. *What is Software Testing? Definition, Basics & Types in Software Engineering*. [online] Available at: <https://www.guru99.com/software-testing-introduction-importance.html> [Accessed 26 April 2022].

[33] Docs.flutter.dev. 2022. *An introduction to unit testing*. [online] Available at: <https://docs.flutter.dev/cookbook/testing/unit/introduction> [Accessed 26 April 2022].

[34] Guru99. 2022. *What is Functional Testing? Types & Examples (Complete Tutorial)*. [online] Available at: <https://www.guru99.com/functional-testing.html> [Accessed 26 April 2022].

[35] Guru99. 2022. *What is BLACK Box Testing? Techniques, Example & Types*. [online] Available at: <https://www.guru99.com/black-box-testing.html> [Accessed 26 April 2022].

[36] Qatestlab.com. 2022. *What Is Scenario-Based Testing and When Use It? – QATestLab*. [online] Available at: <https://qatestlab.com/resources/knowledge-center/scenario-based-part-1/> [Accessed 26 April 2022].

24. Appendix

Appendix 1:

Walkthrough table

#	Section	Done by	Reviewed by	comments
1	Introduction	Alanoud ,Aljoury,Lama	Fay, Reem	Add more details about the features and be more specific about the idea
2	Domain analysis	All	All	Provide more information about the chosen similar application
3	Risk/constrains	Fay, Reem,lama	Aljoury,Alanoud	Revise the type, severity and likelihood for each.
4	Project plan	Fay	Lama	Make sure that the format has been implemented as specified
5	Quality assurance plan	Reem, Lama	Alanoud	Try adding checklist it would be suitable
6	Requirements	All	All	Update survey results Re-check the requirement
7	Problem complexity	Fay,aljoury	Reem	No comments
8	System use-case	All	All	Make all the description use cases have the same format and revise the flow
9	Interaction diagram and analysis classes	All	All	Add the Opt to the sequences to make it more justifiable and add the name of each sequence in the picture
10	Unified VOPC	All	All	Make VOPC for each function spirited and join them together
11	Design class	Fay,Lama	Aljoury,Alanoud	Add the API for the payment

12	System architecture	Lama,Reem	Fay	Provide the references
13	User interface mockup	Aljoury,Alanoud	Lama	Make sure to add the new pictures after the changes
14	Database schema	Fay,Alanoud	Lama,Reem,Aljoury	Add report issue
15	Algorithms	Fay	Alanoud	No comments
16	Expected deployment	Aljoury,reem	Fay,Alanoud	Add introduction paragraph
17	Test scenario	All	All	Make sure that all the scenarios have the same format
18	Project status	Lama	Fay	No comment
19	Conclusion and future work	Alanoud	Aljoury	Mention the admin
20	Unit Testing	All team members	All team members	-
21	Functional testing	All team members	All team members	-
22	Usability Testing	All team members	All team members	-
23	Deployment of the system	Fay, alanoud	lama	-
24	Limitation of the system	aljoury	reem	-
25	Future work	lama	reem	-

26	conclusion	reem	fay	-

Table 84 Walkthrough table

Appendix 2:

Questioner's question and answer:
The questions are provided in the following link:
https://forms.gle/jBs1Mf2XwZM4adfRA

The Result:

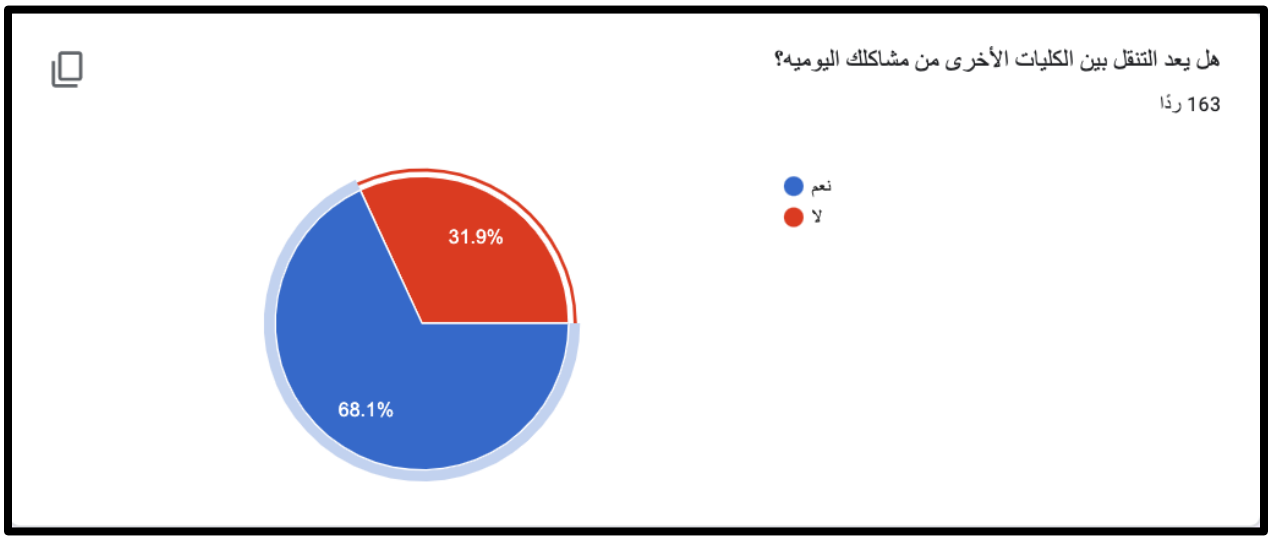


Figure 91 question 1

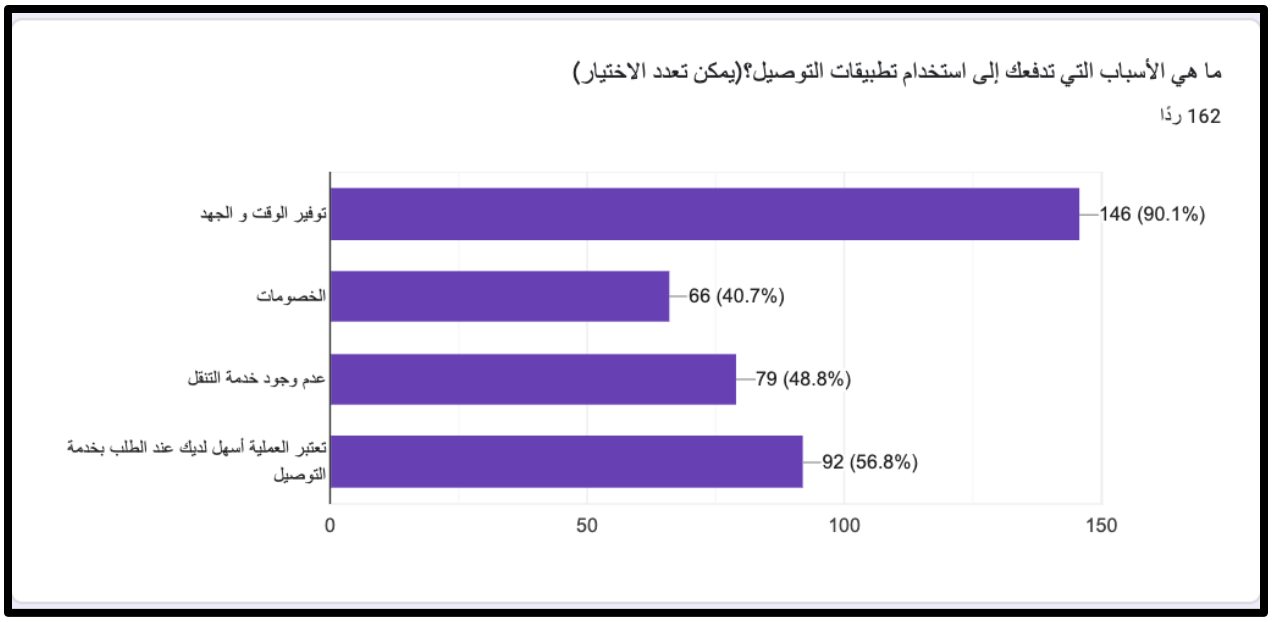


Figure 92 question 2

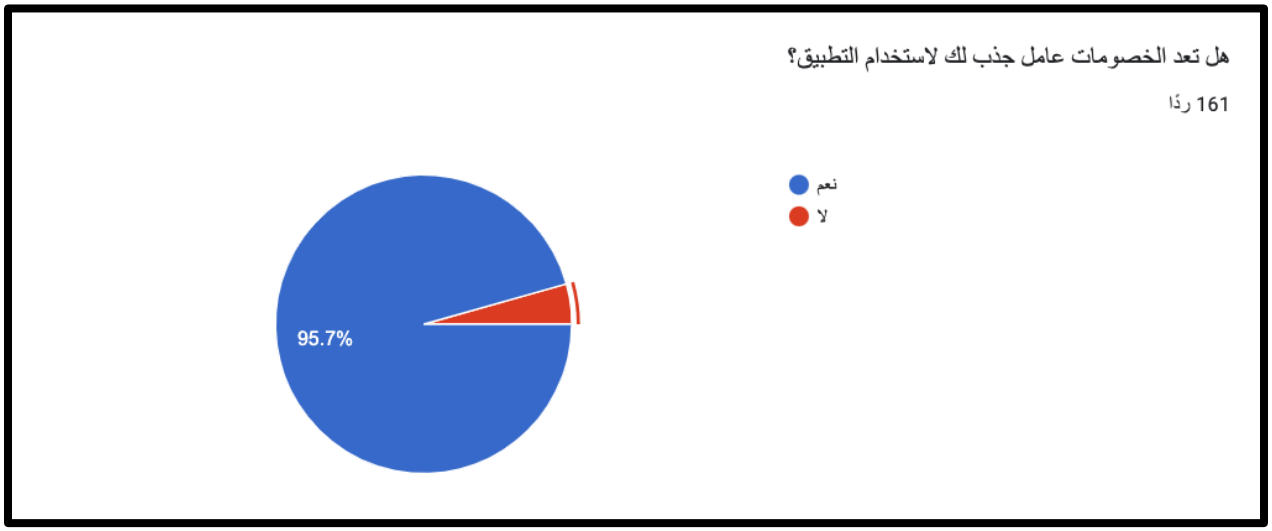


Figure 93 question 3

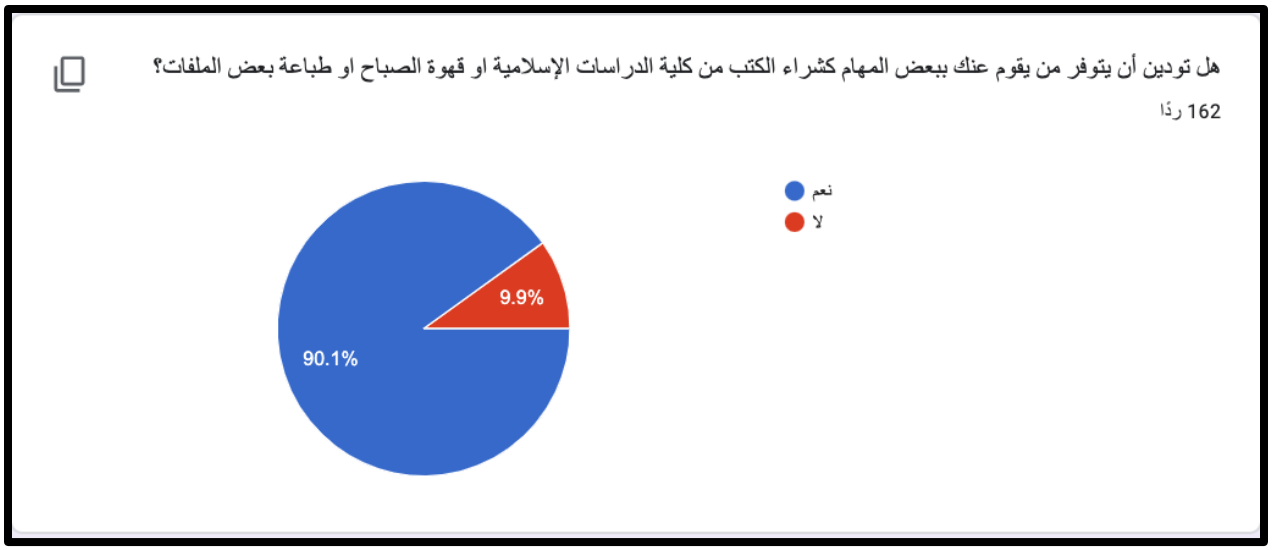


Figure 94 question 4

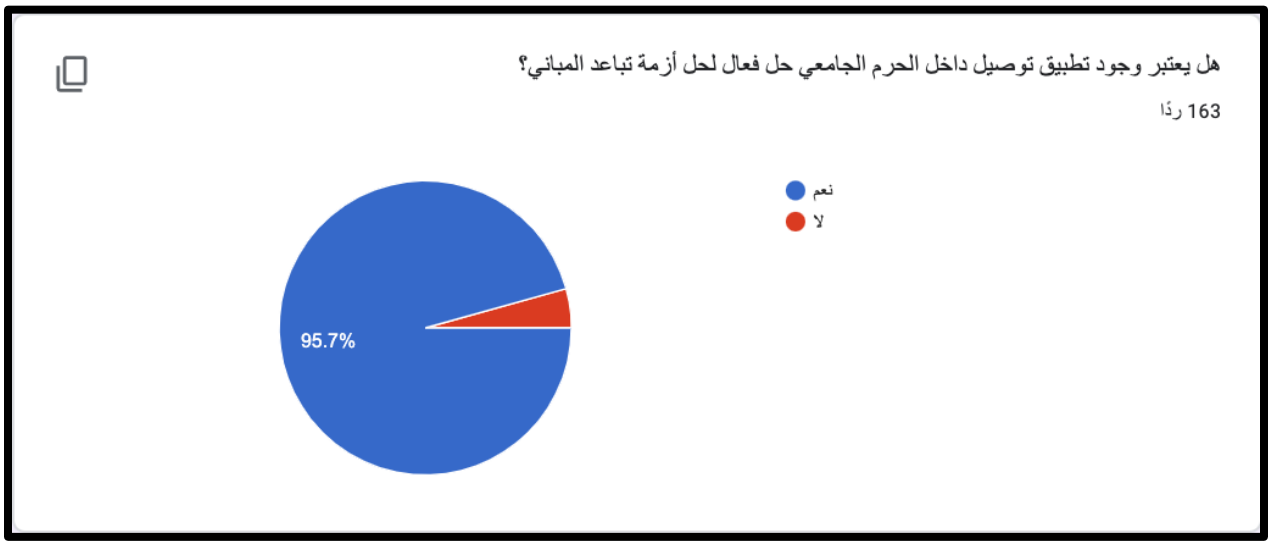


Figure 95 question 5

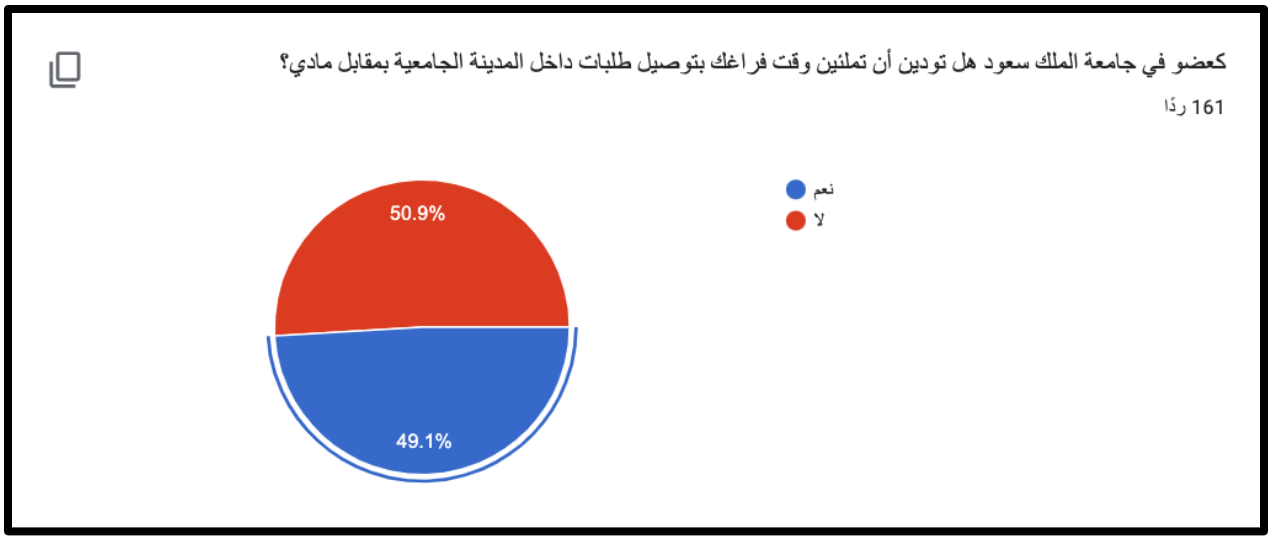


Figure 96 question 6

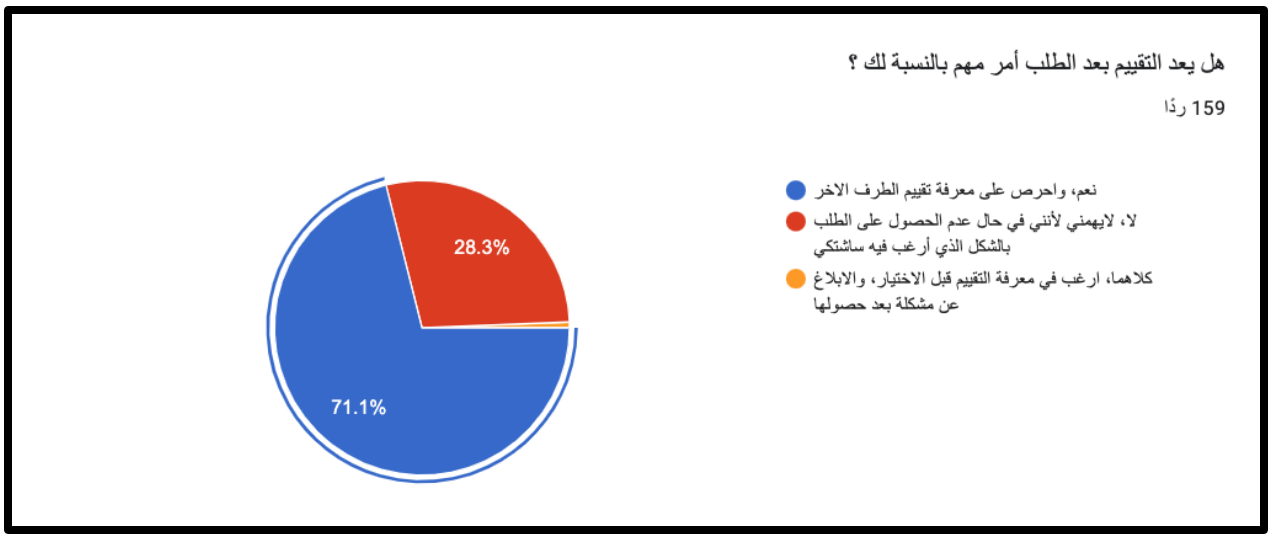


Figure 97 question 7

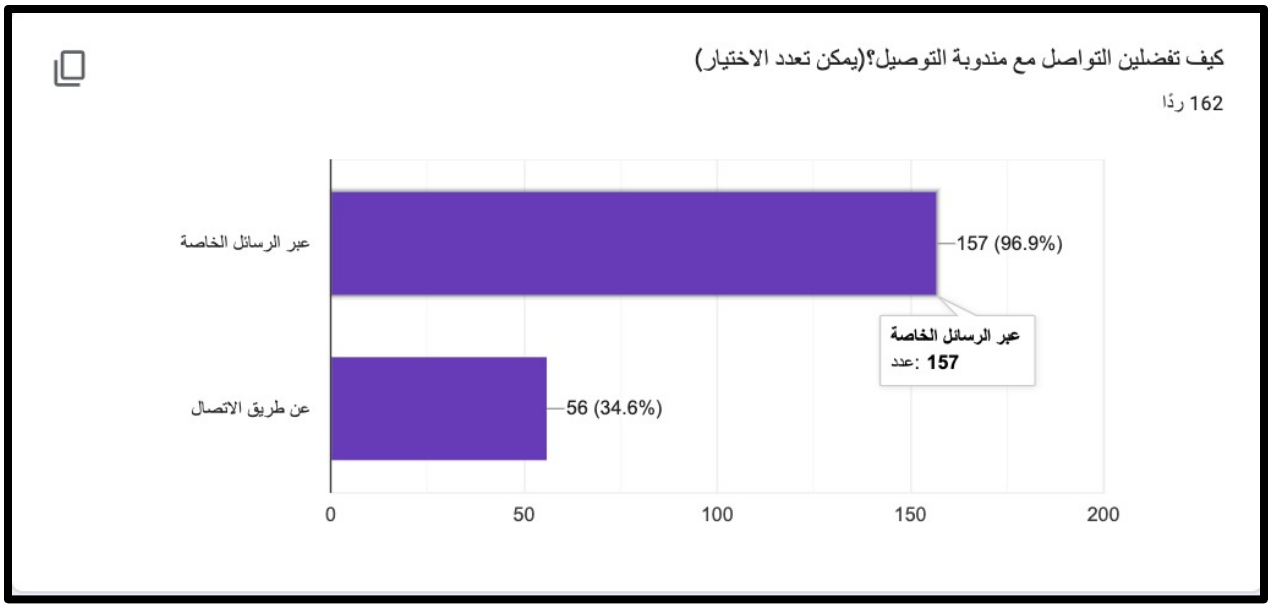


Figure 98 question 8

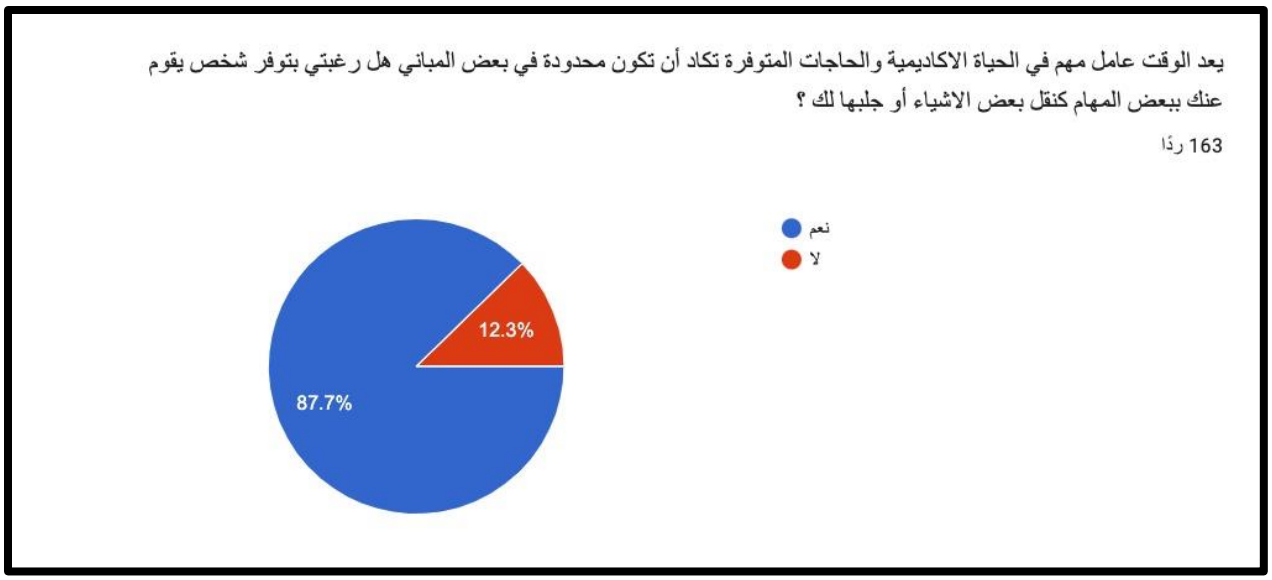


Figure 99 question 9

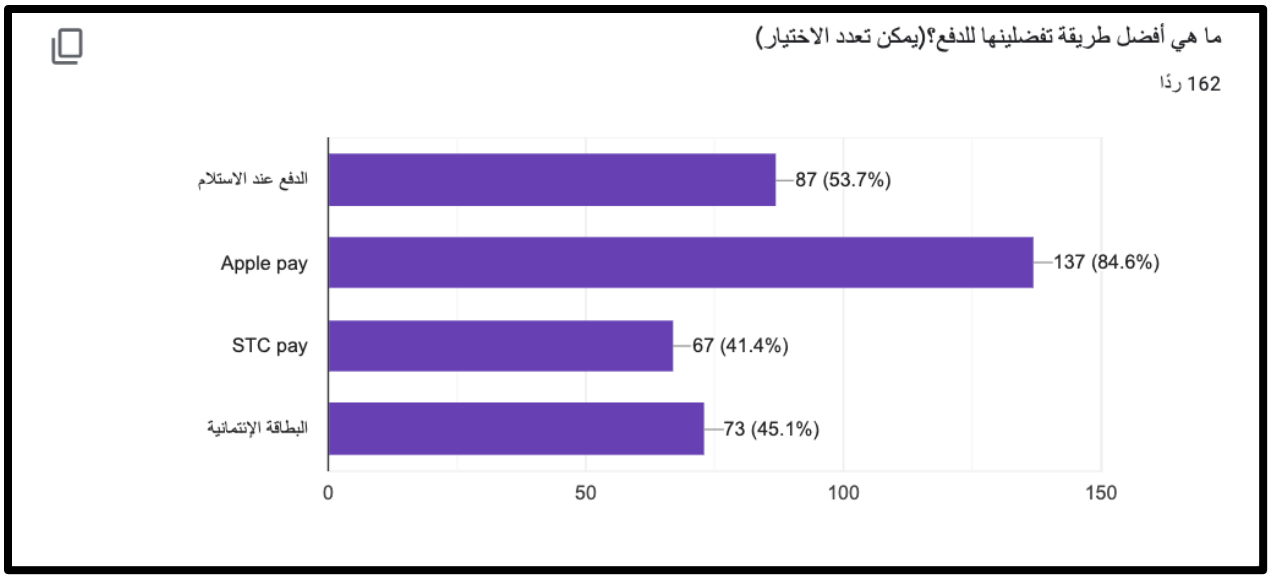


Figure 100 question 10