Department of Software Engineering
College of Computer Science and Engineering
University of Jeddah

CCSE 419 Senior Project

بالتّمام

(To plan your event)

Student Name	ID
Sarah Alsubhi	1915460
Asmaa Felemban	1912424
Nojood Alamri	1912368
Sarah Ahmed	1915345
Lama Alshaikh	1913696

Supervised by:

[Dr. Safa Habibullah]

Table of Contents

Table of Figures

Table of Tables

Table of Changes

Table of Changes1 Table

Need to change	Modification
No appropriate dataset for the recommendation system	Replace the recommendation system with new features
Add new functions to compensate for the removal of the recommendation system	Add new features and functions like chat feature, Favorite and comments and a new user (sponsor) and notifications to all users
Class diagram to accommodate new application functions	Updating old class diagram
Use case diagram to accommodate new application functions	Updating use case diagram with new functions added
Admin authenticate vendors	Admin authenticate vendors services and packages

Justification

Initially, the application was based on a recommendations system. During the Data Collection and Analysis phase, we found that no dataset existed to support the application's concept, thus we proposed the creation of our own dataset. However, following the senior project 1 discussion, it became clear to us:

1. This application cannot be built without a large and reliable dataset.

2. Building the dataset will take a lot of time that we need to use on developing our application.

3. self-build dataset will result in a recommendations system with dummy results.

We decided to add more features to distinguish the application based on the reasons stated above.

CHAPTER 1

INTRODUCTION

1.1. Introduction

These days, we see that planning events have become an essential part of the success of any event. Planning events includes the organization, coordination, and provision of all the many details that include resources, places and others. Accordingly, events planners face difficulties in planning events, it is exhausting work as it takes a lot of time and effort to find the right Vendor, Sponsor for its event, also they must do multiple tasks in several things simultaneously such as communicating with the Vendors and Sponsors, signing the contract, booking, track booking and other tasks. In addition, the event planners use traditional means of planning, such as communication by phone and email, visiting the Vendors' place to complete the reservation and receipt and may have to visit several days, which makes it time-consuming.

Although there are many event planning apps, most of them lack comprehensive features such as having a sitting group of vendors in one place, booking and direct tracking, and special suggestions according to the type of event .

We came up with an idea that will have a better impact on the work of the events planners, as we propose to use technologies to develop a system, which is characterized by the technology of the Recommendation System to be capable of helping the planners in completing their work in a fast, inexpensive way in the effort or time. This project will allow to events planners build a suitable plan for their events and find the most suitable Vendors.

1.2. Problem Definition

Many event planners fail to plan an event well due to several factors they encounter during planning which causes unsuccessful events. The factor of availability of vendors and communicating with them plays a major role in planning, as the process of searching for vendors who provide the required services and are suitable for the budget and the event is cumbersome and takes a lot of time and effort of the planner, which leads to a defect in the event plan.

1.3. Proposed Solution

To solve the above-mentioned problem, we propose an application called (بالتّمام) that connects event planners with vendors so that the planners can find the vendors, sponsors, services, and places suitable for their event easily also, they can communication and reservation without effort and waste of time.

1.4. Aim

This project aims to develop an application that connects event organizers with service providers (vendors, sponsors) based on their correspond needs.

1.5. Objectives

- To introduce the vendors and their services.
- to introduce sponsors and contact with them.
- To develop a method for booking services online.
- To facilitate communication between event planners and vendors.
- To design a database to save event planners, vendor and sponsor information.

1.6. Stakeholders' Definition

There are two types of stakeholders in our project:

- Internal Stakeholders: Admin, Project Manager, Developer, Tester, Programmer, Team and Supervisor.
- External Stakeholders: Planner, Vendors, Sponsors and interested users .

1.7. Project Timeline

This project will be completed within two semesters. The first semester will include two chapters (Introduction, Literature review) and two sprints, the first sprint we will collect data, write requirements, and design UML diagrams, the second sprint we will start implementing and writing codes. The proposed timeline is presented below in (figure 1-1). The Gantt chart was created using Microsoft Project.

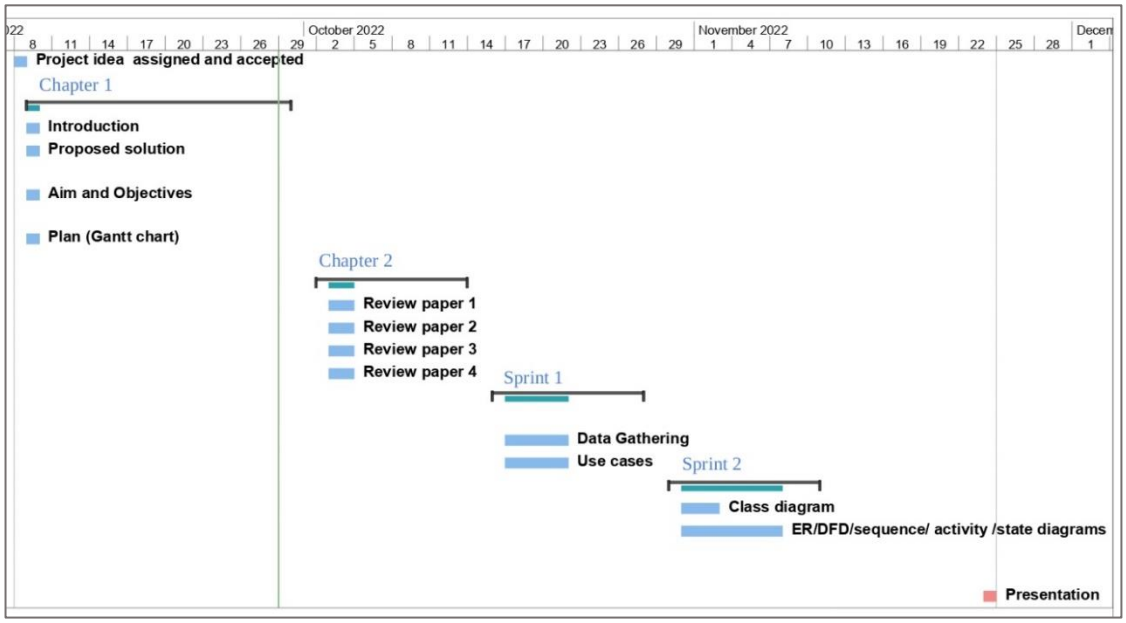


Figure 1 Gantt chart of the project plan

1.8. Conclusion

This chapter was an introduction to our project, as we covered the definition of the problem and the proposed solution, discussed the project aim and objectives, defined the scope of the project and stockholders, and presented the project plan through a Gantt chart.

In the next chapter, we will discuss and review some literature reviews, features of projects related to our project and comparison between them.

CHAPTER 2

SIMILAR APPLICATIONS REVIEW

2.1. Introduction

Our application helps event planners to find the most suitable vendors, Sponsors and service providers using different techniques. furthermore, a review and description of some similar applications to discuss the most important main features of each application and compare it to ours to maintain the limitation and maximize the potential benefits of the proposed system.

2.2. Similar applications

In this section, we will discuss similar projects that are conceptually and technologically related to our system. We will provide a list of comparable applications that match the concept and technology of our proposed system, discuss the features and limitations of each application so that we can address these limitations as much as possible.

2.2.1. Ratbli

The Ratbli application offers a solution for individuals who wish to plan their private parties. They can search for services such as hosting and logo design, as well as products such as food and decorations. In addition, it provides party planners advice for those who desire it. To reserve a service, the user looks it up and selects one of the provider's packages. Ratbli is straightforward and includes the essentials for most typical parties.

Figure 2 Home page "رتبلي"

Figure 3 Services page "رتبلي"

On the side of service providers, they register and then use the website to add their products or services, which are then approved by administrators and added to the application. There are not many features available for business owners.

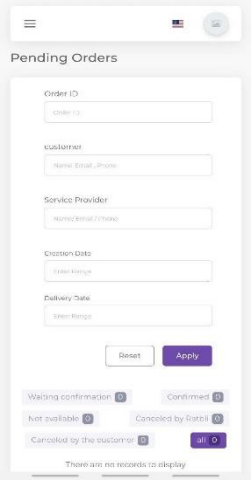


Figure4 Pending orders form "رتبلي"

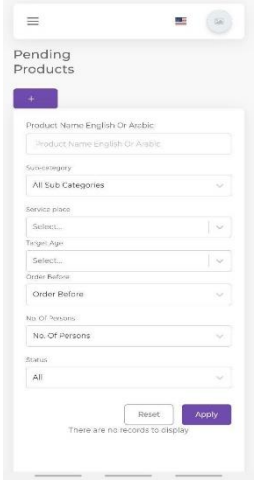


Figure5 Pending products "رتبلي"

2.2.2. Happy Season

The happy season is an application that helps you easily access what you need for your event, including halls, kitchens, buffets, cosmetics, fashion stores and all-party supplies from photographers, wedding songs and car legislation and connects you to the nearest service provider in your city.

Also, the happy season allows you to design your invitation cards for all its categories, which include graduation parties, holidays, and special and public occasions, coordinate them in terms of shape and font, and send them to your contacts via email or social media.

Home page "Happy season" 7 Figure

Figure8 One of Places with detail "Happy season"

Figure6 One of Services with detail "Happy season"

Through Happy season, you can only explore options that may be suitable for your event; you cannot connect with suppliers or book them through the application.

2.2.3. Dalel Almonasbat

The application saves users time and effort by centralizing their search for vendors of hospitality and entertainment services.

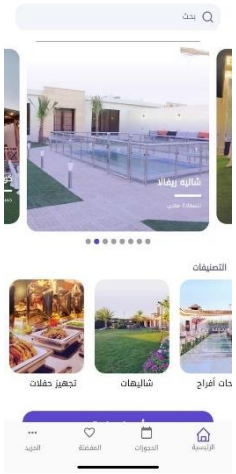


Figure 9 Home page "Dalel almonasbat"

Figure 10 Browse categoires in the home page "Dalel almonasbat"

As you can see in the figure (Figure 9), the home page is categorized and has many functions such as special pricing, as well as the option to become a partner by logging into a connected application.

Figure 14 Browse places by spesific category "Dalel almonasbat"

Figure 11 Identify the date and location of the event "Dalel almonasbat"

Figure 12 List of categories "Dalel almonasbat"

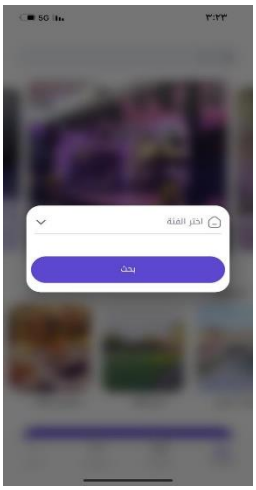


Figure 13 Search for category"Dalel almonasbat"

in the figures above, the search is categorized, and appropriate results will be displayed based on user selections. For instance, the user will select the date and city, the system will display unreserved places, after which you can reserve one, subject to the term and conditions. The vendor will then contact you via your phone number.

Figure 18 Term and conditions of the service"Dalel almonasbat"

Figure 15 Reviews of the service "Dalel almonasbat"

Figure 16 Praice of packages and detail "Dalel almonasbat"

Figure 17 Service in detail "Dalel almonasbat"

2.2.4. Eventful

Eventful is an application that offers event planners with an innovative experience. It provides several options for places and services that a user may require, as well as an organization service. In addition, it offers numerous features, that include the ability to search by event type and date, the ability to reserve a service or location immediately, and payment via MADA, visa, or installments.

Figure 21 Home page "Eventful"

Figure 19 Type of event "Eventful"

Figure 20 Browse services and offer "Eventful"

Figure 22 Show place with detail "Eventful"

In addition, venue owners and event service providers can become Eventful partners by registering to offer their venues and services.

Figure 24 Home page to vendor and business "Eventful"

Figure 25 Welcome page to vendor when provide order to be partner "Eventful"

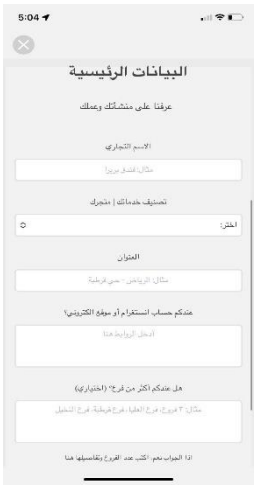


Figure 23 Vendor form page "Eventful"

2.2.5. Event

It is an application for coordinating events by searching for and contacting local service providers such as photographers, makeup artists, and event venues. The services are categorized for easy accessibility. It provides contact information such as a phone number and Instagram account for each service provider, and the application allows you to communicate with them. You can view all services and add your advertisement using the same account.

Figure 28 Home page "Event"

Figure 27 Search and section page "Event"

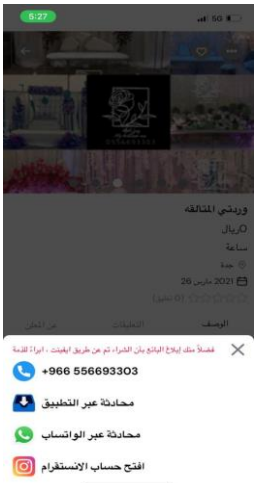


Figure 26 Contact method with service provider "Event"

Figure 29 service with detail "Event"

2.3. summary of Similar Projects

Table 2 Comparison of Similar Projects

project	search for services	booking service	services for business events	following up with requests	writing reviews	sponsoring users	smart search based on user preferences
بالتمام	yes	yes	yes	yes	yes	yes	yes
ratbliرتبلي	yes	yes	no	yes	no	no	no
eventful	yes	yes	yes	yes	no	no	yes
happy season	yes	no	no	no	no	no	no
دليل المناسبات	yes	yes	no	yes	yes	no	yes
Event	yes	no	no	no	yes	no	no

The differences between our application and other similar applications are illustrated in Table 1 by comparing significant event planner and service provider characteristics. According to the table, a comparable project supports several important features, but our application is the only one that can show a list of sponsors that can helps planners by sponsoring their events. Also, many of these applications make it more difficult for users to search for appropriate services because they require manual searching and looking, which wastes a great deal of time and effort. Our application will assist you in planning your event more efficiently and without delay.

2.6. Conclusion

This chapter We presented applications that were comparable to our system. Then compared to our own system. results based on the interests of similar users without relying on specific features.

CHAPTER 3

Data Collection and Analysis

3.1. Introduction

In this chapter we will present functional requirements, and non-functional requirements. Furthermore, we will design use case diagram of the functionality and use case specifications to identify, organize and clarify functional requirement.

3.2. Requirement Specification

3.2.1. Functional Requirements

Table3 functional Requirements

FR No.	Functional Requirement	Priority
User		
FR1	**Sign up**	**M**
FR1.1	The User shall be able to register using email and a password	
FR2	**Sign in**	**M**
FR2.1	The User must be able to sign in using their registered email and password	
FR3	**Get Notification**	**M**
FR3.1	the user shall be able to get notifications	
Admin		
FR1	**Manage Services and packages**	**H**
FR1.1	The Admin shall be able to authenticate services and packages	
Sponsor		
FR1	**Contact planners via chat messages**	**M**

Table 4 functional Requirements (cont.)

FR No.	Functional Requirement	Priority
Vendor		
FR1	**Sign up**	**M**
FR1.1	The Vendor must be able to register using email and a password	
FR1.2	Vendor must be able to create a vendor account by checking the “vendor” box at registration	
FR2	**sign in**	**M**
FR**2**.1	The Vendor must be able to sign in using their registered email and password	
FR3	**Create services**	**H**
FR3.1	The Vendor must be able to create services that provided to planners	
FR4	**Create packages**	**M**
FR4.1	The Vendor shall be able to specify customized packages of services or offers	
FR5	**Manage booking requests**	**H**
FR5.1	The Vendor must be able to approve or reject booking requests	
FR6	**Manage orders status**	**M**
FR6.1	The Vendor must be able to update order status for tracking	
FR7	**Update service and package**	**M**
FR7.1	The vendor must be able to update the details of its service or package	
FR8	**Delete service and package**	**M**
FR8.1	The vendor must be able to update the details of its service or package	

Table5 functional Requirements (cont.)

FR No.	Functional Requirement	Priority
Planner		
FR1	**Sign up**	**M**
FR1.1	The Planner shall be able to register using email and a password	
FR2	**Sign in**	**M**
FR2.1	The Planner must be able to sign in using their registered email and password	
FR3	**Browse vendors**	**H**
FR3.1	The Planner shall be able to browse through vendors based on categories and services	
FR4	**Search for services and packages**	**M**
FR4.1	The Planner shall be able to search for services and packages.	
FR5	**Book a service**	**H**
FR5.1	The Planner shall be able to book a service\package and provide the vendor with any further information	
FR6	**Track orders**	**M**
FR6.1	The Planner shall be able to track the status of orders	
FR7	**Contact sponsor**	**M**
FR7.1	The Planner shall be able to search for a suitable sponsor for their event	
FR7.2	The Planner must be able to obtain the sponsor's contact information and communicate with them	
FR8	**Add comments**	**M**
FR8.1	the planner shall be able to add comments on services and packages to give feedback	
FR9	**Add to favorites**	**M**
FR9.1	The planner shall be able to add services and packages to favorite.	

3.2.2. Non-Functional Requirements

Table6 Non-functional Requirements

NFR No.	Non-Functional Requirement
Usability	
U1	The user interface shall be GUI
U2	The system must be able to provide appropriate functionality in the user interface according to the type of user
U3	The user shall be able to understand the flow of application easily without any guide or help from manuals
U4	The user can easily navigate its interface
Security	
S1	The system must ensure the integrity of the vendor account information by having the admin check it and verify it
S2	The system must protect the personal information of all users
Portability	
P1	The application must support iPhone device running on OS iOS version 15 and iOS version 16
P2	The application must support android device running on OS android version 12.1/12L
Availability	
A1	The application must be available for the user's 99 percent of the time every month during business hours.

3.3. Use case Diagram and Specifications

3.3.1 Use case Diagram

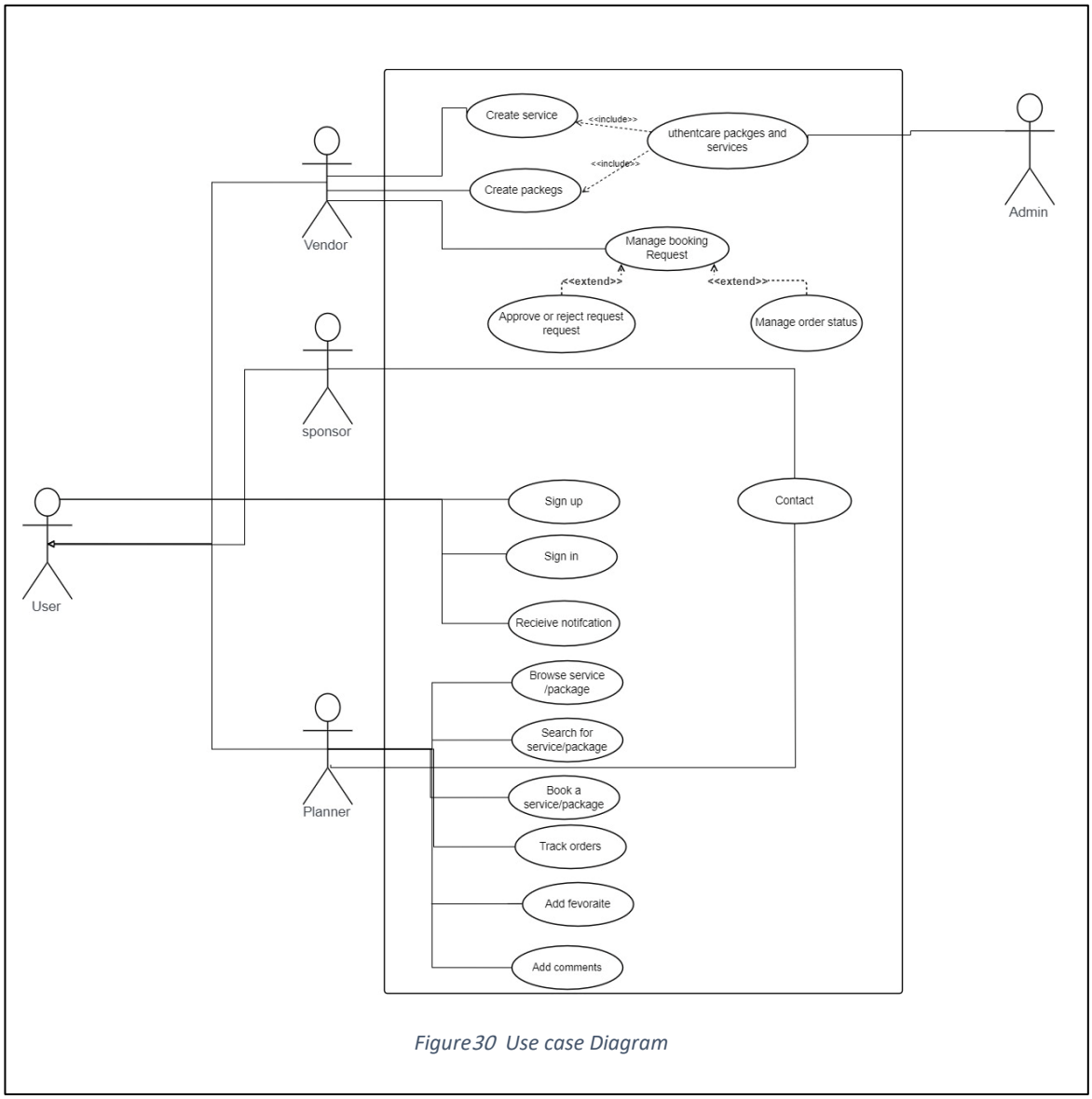


Figure30 Use case Diagram

3.3.2 Use case Specifications

Table 7 use case specification (Manage Vendor)

Use case name	Manage Requests	
Use case number	1	
brief description	Admin shall be able to verify newly registered vendors accounts	
Actors	Admin	
Related Use case	Manage Vendors account	
Preconditions	Vendor sign up	
Post condition	vendor account verified or rejected successfully	
Flow of activities	**Actor**	**System response**
	1-Admin log in	
		2-Display Vendor manage users page
	3-Admin click on verify account or reject account	
		4-Vendor account activated or rejected successfully

Table8 use case specification (Create Services)

Use case name	Create Services	
Use case number	2	
brief description	Vendor create services that offer in order to accept reservations.	
Actors	Vendor	
Related Use case	None	
Preconditions	Vendor log in	
Post condition	Vendor create servicee successfully	
Flow of activities	**Actor**	**System response**
	1-Vendor click on create service	
		2-Display create services page
	3-Vendor fills the serivice information	
	4-Vendor click on create serivce button	
		5-Service created successfully

Table9 use case specification (Browse Vendors)

Use case name	Browse Vendors	
Use case number	3	
brief description	planner browses vendors based on categories and services and can display vendor information, service packages, prices, reviews, and availability	
Actors	Planner	
Related Use case	None	
Preconditions	Planner log in	
Post condition	The planner obtained the result of browsing successfully	
Flow of activities	**Actor**	**System response**
	1-Planner enter to the home page	
		2-Display vendors
	3-Planner browse vendors based on attribute	
		4-Display vendors with this attribute

Table 10 use case specification (Book a service)

Use case name	Book a service	
Use case number	4	
brief description	The user books the service package and provides the details to the vendor	
Actors	Planner	
Related Use case	None	
Preconditions	Planner log in	
Post condition	Planner booked the service successfully	
Flow of activities	**Actor**	**System response**
	1-Planner choose the service package	
	2-Planner click on book service button	
		3-Display form to enter the details
	4-Planner enter the details	
		5-Send the booking to the vendor to review

Table 11 use case specification (Manage Request)

Use case name	Manage Booking Request	
Use case number	5	
brief description	The Vendor must be able to approve or reject booking requests	
Actors	Vendor	
Related Use case	None	
Preconditions	Vendor log in	
Post condition	Vendor manage request successfully	
Flow of activities	**Actor**	**System response**
	1- Vendor click on manage bookings	
		2- display manage bookings page
	3-Vendor click on approve request if is accept the order	
	4-Vendor click on reject request if is not accept the order	
		5-Booking request done successfully

Table 12 Use case specification (Authenticate vendor)

Use case name	Authenticate Vendor	
Use case number	6	
brief description	The Admin must be able to authenticate the vendor	
Actors	Admin	
Related Use case	Sign up	
Preconditions	Vendor Sign up	
Post condition	Vendor authenticated successfully	
Flow of activities	**Actor**	**System response**
	1- Admin will receive authenticate request from the vendor	
	2- Admin reject or approve	
		3- display massage if the admin rejects or approve
		4-ir approved, Vendor sign up and authenticated successfully

Table 13 Use case specification (Search for Vendor)

Use case name	Search for Vendor	
Use case number	7	
brief description	The Planner shall be able to search for vendors based on location, service type, or event size	
Actors	Planner	
Related Use case	none	
Preconditions	Planner log in	
Post condition	Planner found the suitable vendor successfully	
Flow of activities	**Actor**	**System response**
	1- Planner go to the home page	
	2- Click on the search bar	
	3- write location, service type, or event size that want	3- present the result that suite to Planner search successfully

Table 14 Use case specification (Track orders)

Use case name	Track orders	
Use case number	8	
brief description	The Planner shall be able to track the status of orders	
Actors	Planner	
Related Use case	Book service	
Preconditions	Planner has order/s	
Post condition	Planner could track order/s successfully	
Flow of activities	**Actor**	**System response**
	1- Planner check the status of the order	
		2- track status changes by the vendor
	3- Planner tracked the order and status successfully	

3.4. Conclusion

This chapter presented system requirements including functional and nonfunctional requirements with its priority. To clarify the functional requirements of our application, we present it in a use case diagram. And a use case specification to make it more understandable; how the functions are related to the users.

CHAPTER 4

DESIGN

4.1. Introduction

In this chapter, we present some UML Diagrams to understand and describe the system parts and functionality. We designed a Class Diagram for the whole system and Sequence Diagrams for the sign-up requirement for both the planner and the vendor to clarify how the system works and how it communicates with users' data.

4.2. Class Diagram

The class Diagram below describes the structure by exhibiting the system's classes with attributes and operations of each class. More importantly, it shows the relationships among objects.

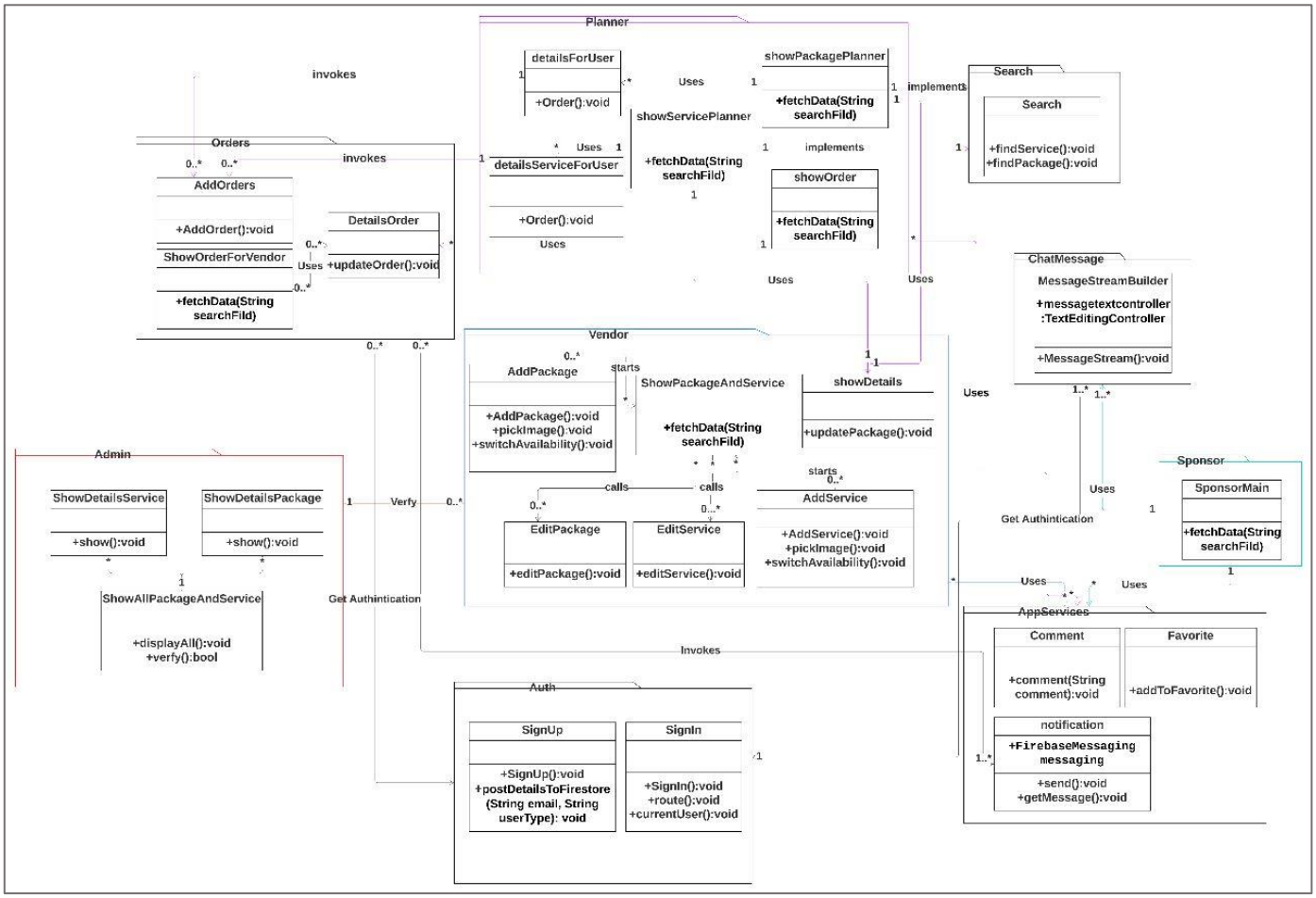


Figure31 Class Diagram of the whole application

The class diagram clearly shows the structure of our system. The classes are divided into packages based on their functionality and/or relevant user. Most classes have an association relationship that is described and labelled on them. The planner package contains classes relevant to displaying services and packages to the user and it invokes the classes in orders package to create and handle orders. The information about packages and services are created and handled in the vendor package. There are classes for other app functionalities like: search, chat messaging, notification and so on. The sponsor package contains only one class but it interacts with other classes to provide the sponsor with what they need. The Admin package contains classes that help the admin see and verify the newly added packages and services. Finally the auth package contains sign up and sign in classes but they also work on authenticating the users when they send some requests, for example when a new order is sent.

4.3. Sequence Diagrams

The Sequence Diagram shows process interactions between objects in the sequential order in which those interactions occur.

4.3.1. Signup as a Vendor

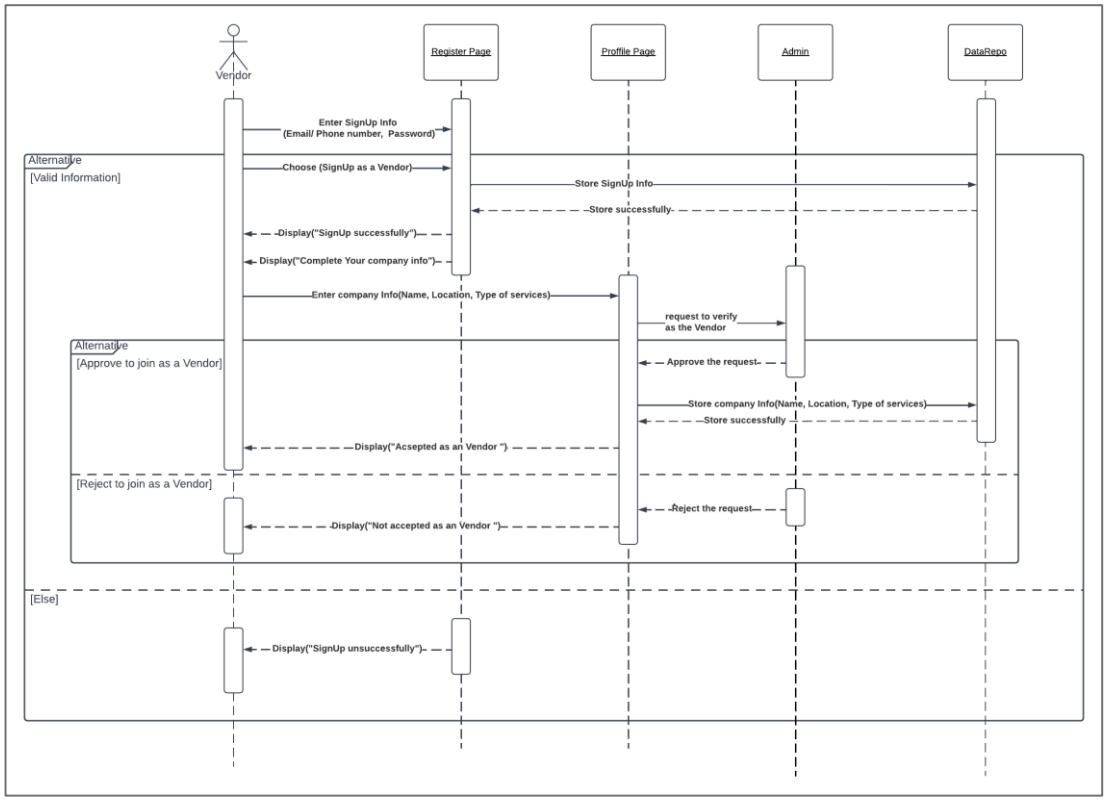


Figure 32 Sequence Diagram of vendor Sign up

The Sequence Diagram of signup as vendor shows what are the important info for the registration process and how the system will deal with this info so, if it successful the vendor will complete the process by choosing (I'm vendor) then the system will ask the vendor to add the rest information (company name….) if it verified by the admin or not if it is verified so the vendor data will stored by the data base if not the system will display("Not accepted as vendor").if the registration not success so no information will stored the system display("Signup unsuccessfully").

4.3.2. Signup as a Planner

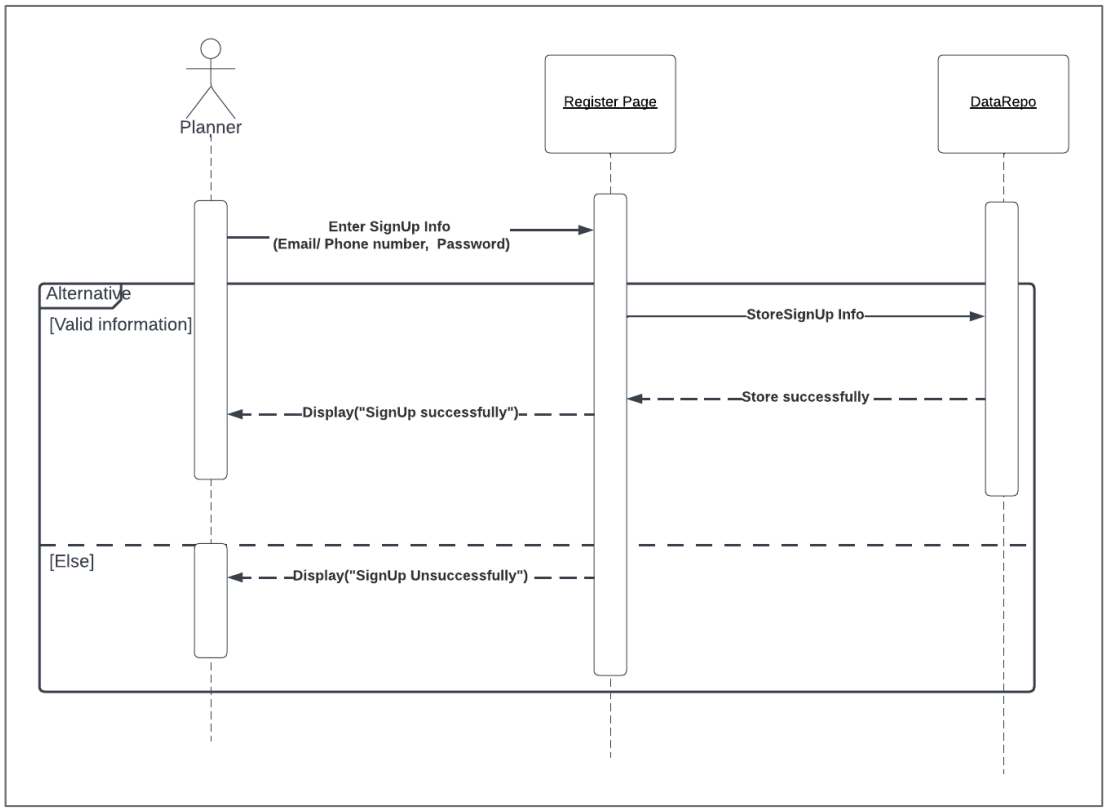


Figure 33 Sequence Diagram of Planner Sign up

The Sequence Diagram of signup as Planner shows what are the important info for the registration process and how the system will deal with this info so, if it successful the planner information will be stored by database then the planner completes the process by entering the system and complete the rest pages if it not success so no information will store the system display (“Signup unsuccessfully”).

4.3.3. Vendor edit service

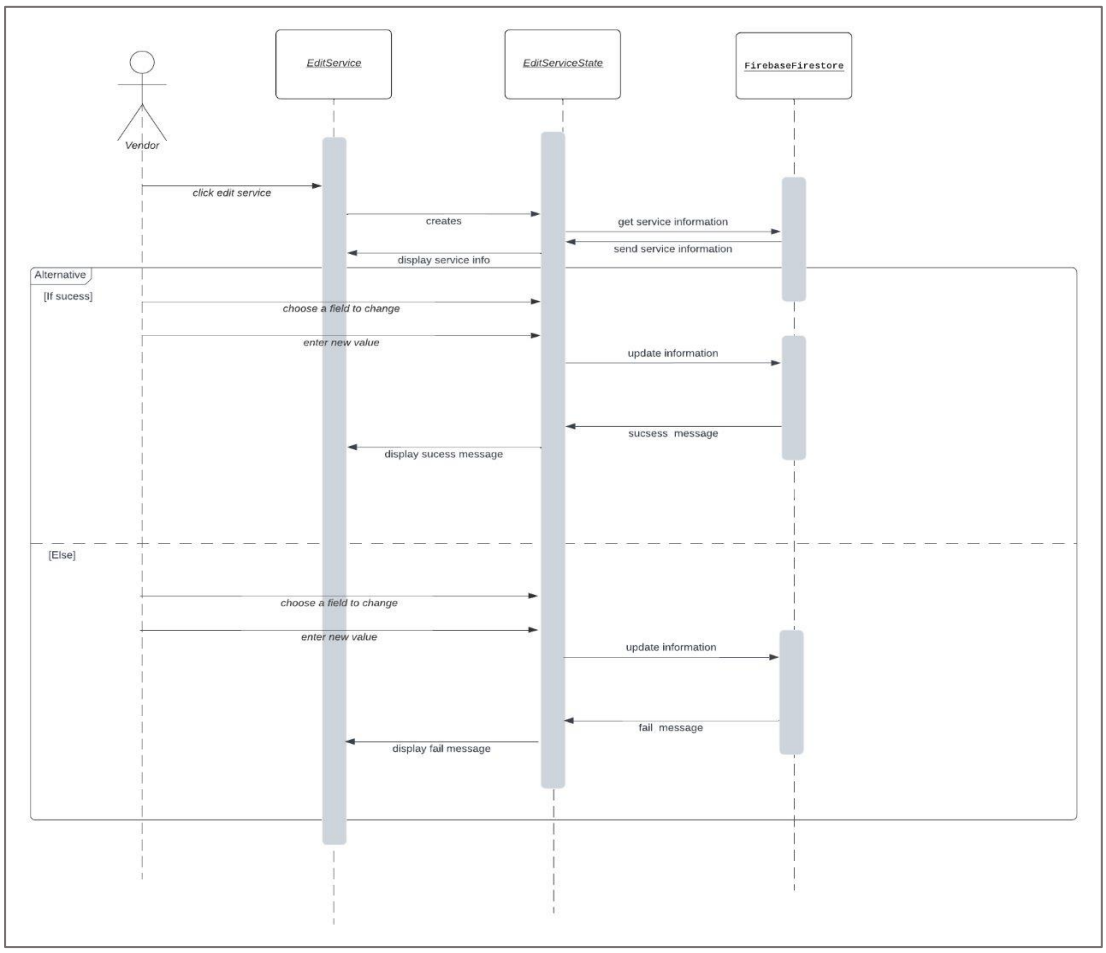


Figure 34 Sequence Diagram of Vendor edit service

The sequence diagram of "vendor edit service" shows the steps of editing a service information that already exists. The vendor can change the service name or image or availability and other information using this feature. And the diagram shows both the normal and alternative flow of events.

4.3.4. planner - place order

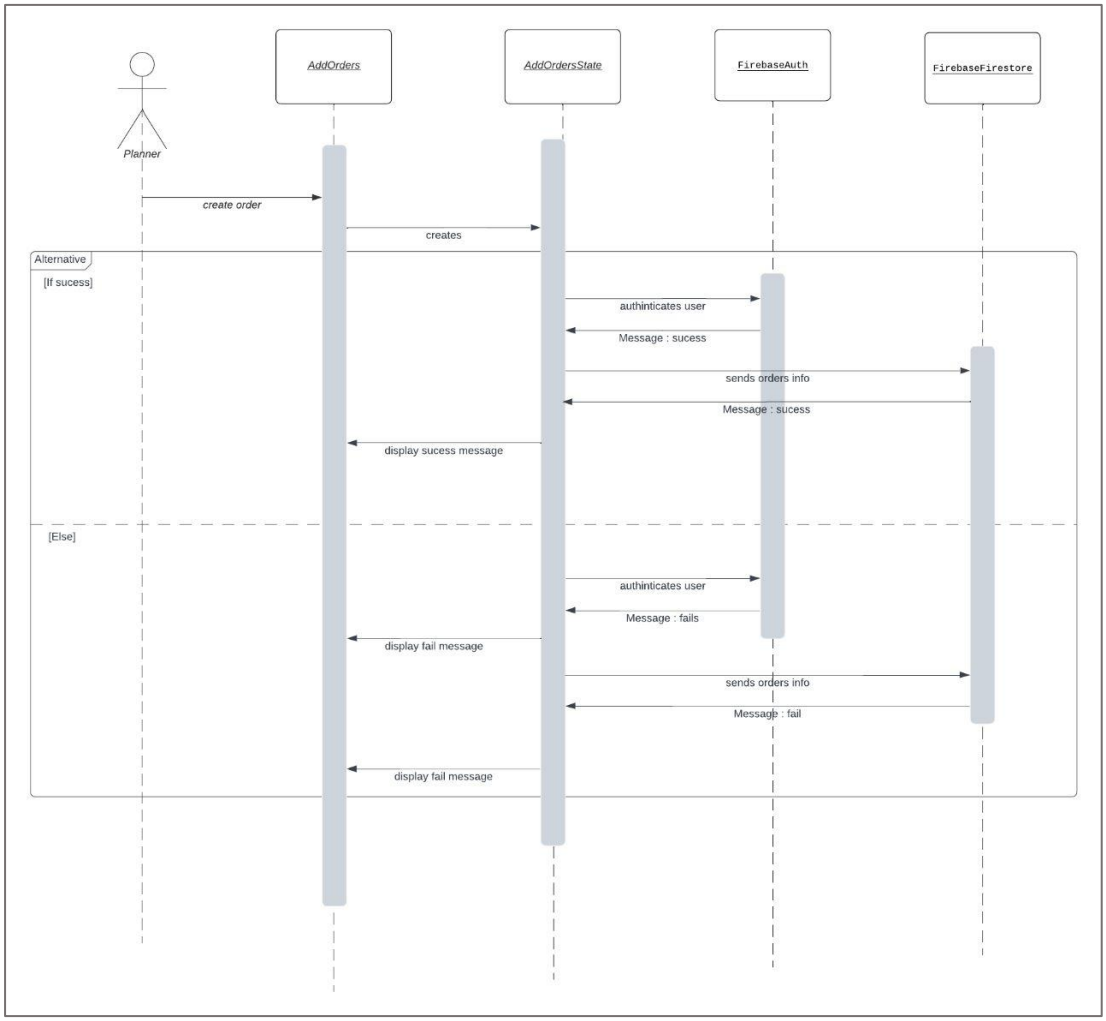


Figure 35 Sequence Diagram of planner - place order

The sequence diagram of "planner - place order" shows how an order is created and handled by the system and how it is saved in the database.

4.3.5. Chat message

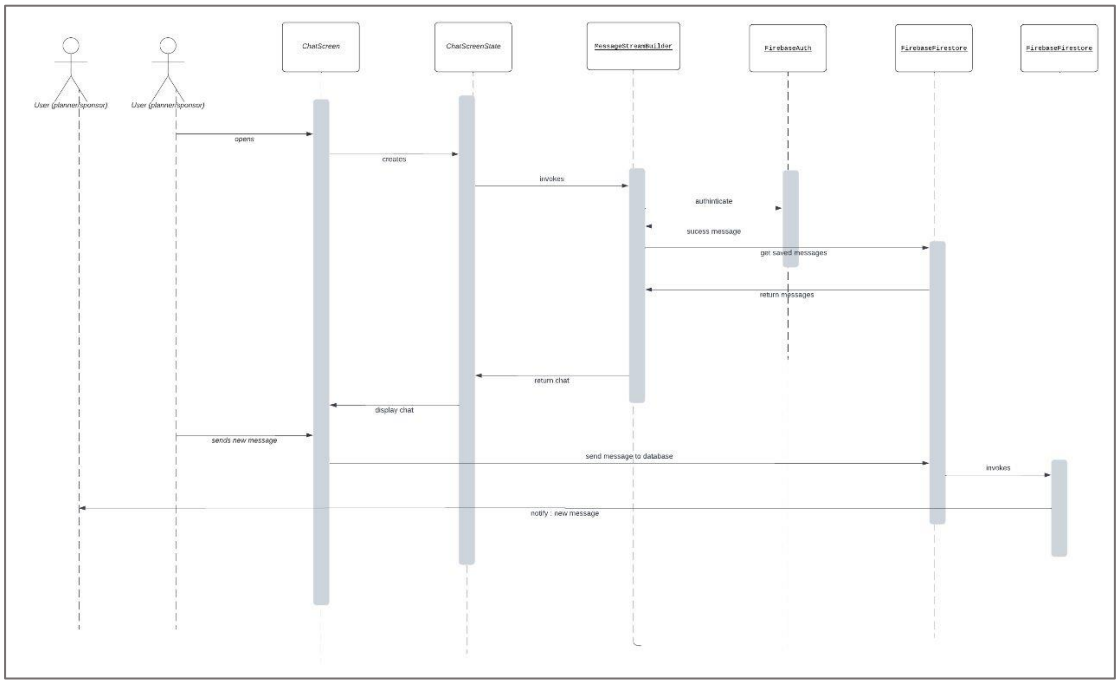


Figure 36 Sequence Diagram of Chat massage

The sequence diagram of "chat message" shows how this feature, that is available for the planner and the sponsor, is is working. For displaying the messages class "MessagestreamBuilder" handles the saved messages in the database to display them correctly. When a new message is sent from the user to the database it invokes the notification object to notify the receiver.

4.4. State Diagrams

4.4.1. Service Availability

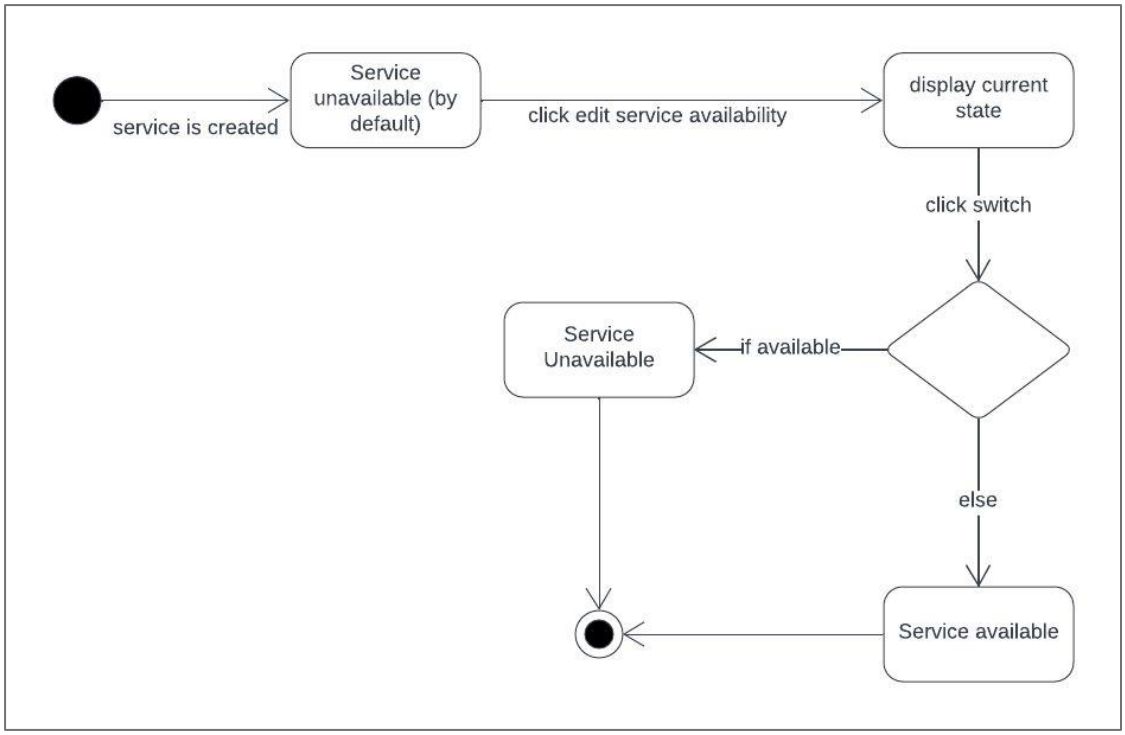


Figure 37 State Diagram of Service Availability

This state diagram of the "Service Availability" shows how the services change there state between available and unavailable depending on the vendor's input.

4.5. System Architecture

In our system we used Flutter BLoC pattern (Business Logic Component). Which is an architectural pattern based on separate components. This architecture was introduced by Google at Google I/O 2019 and it is recommended for flutter apps, so we structured our app based on it and follow it to some extent whenever possible.

We follow an architecture that has four layers:

Data layer: This layer is the one in charge of our data.

Domain layer: This is the one in charge of transforming the data that comes from the data layer.

Business logic layer: This layer manages the state.

Presentation layer: Renders UI components based on state.

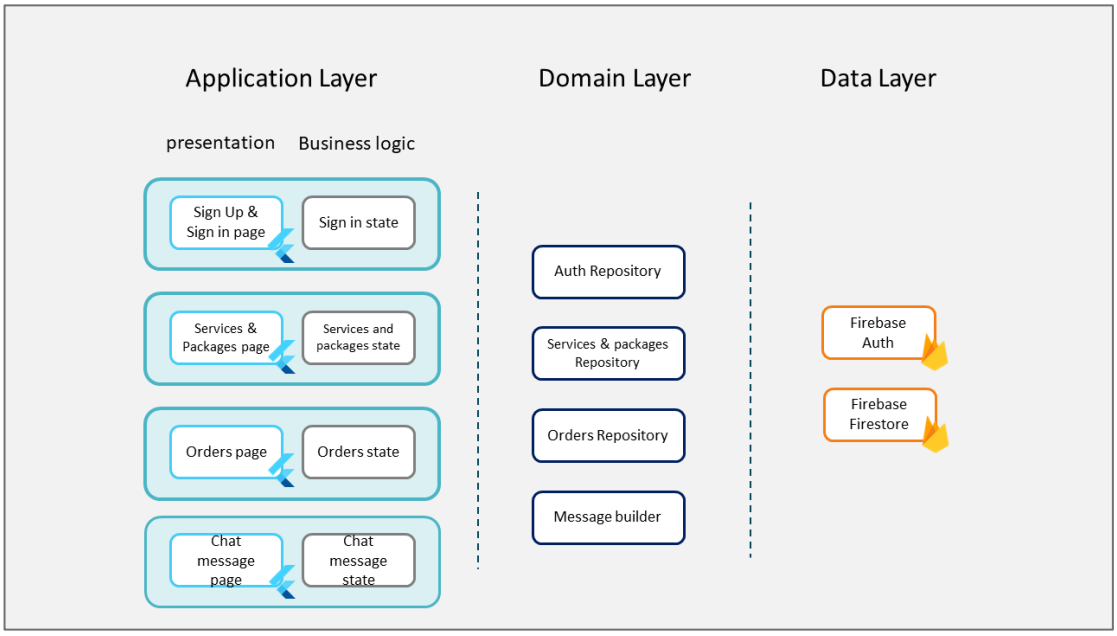


Figure 38 System Architecture (BLoC)

4.6. Prototype

The project aims to provide an easy, clear, and graphical user interface. The low prototype helps us to visualize the user interfaces and distribute the functions to achieve the goals of the application, it will clarify the work of the functions and how to navigate sequentially through the application and how users will interact with it. we will present some low prototype interfaces.

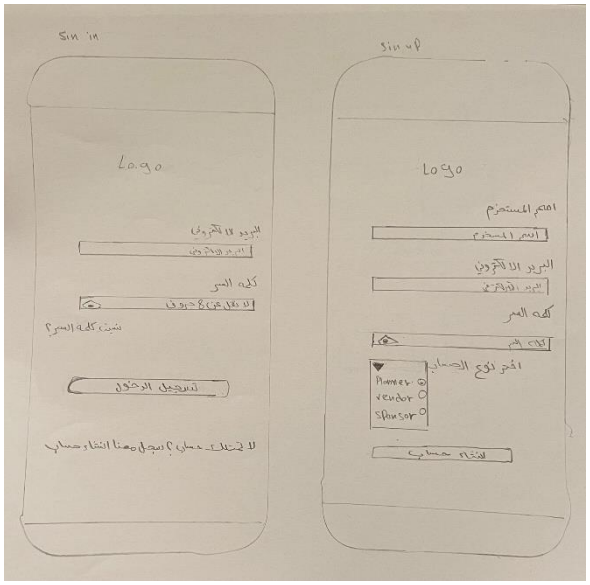


Figure41 Low Prototype of (Sign up and Sign in)

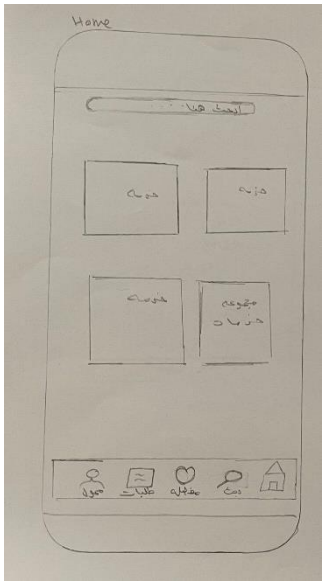


Figure40 Low Prototype of (Planner's Home page)

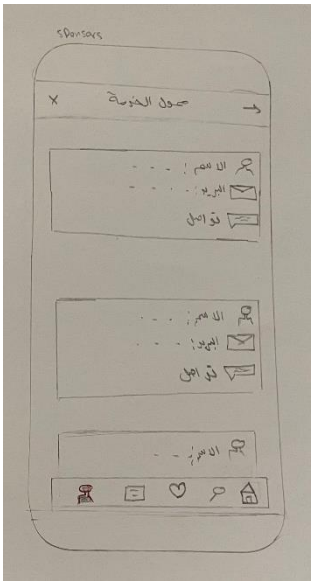


Figure39 Low Prototype of (Sponsors list)

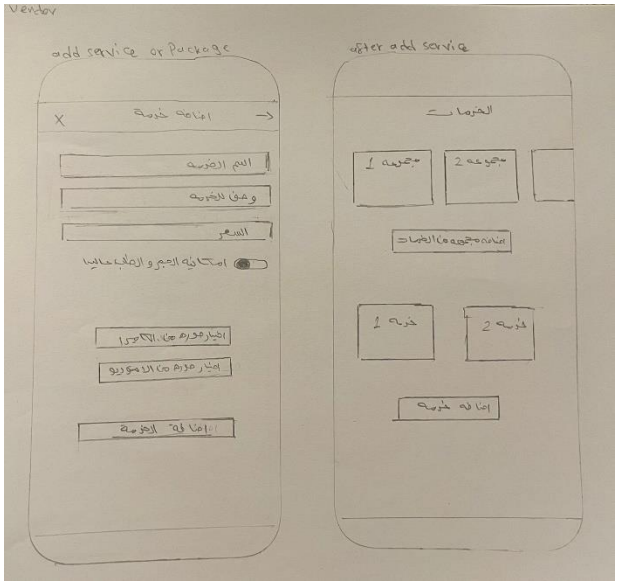


Figure42 Low Prototype of (Add service or Package, Services page)

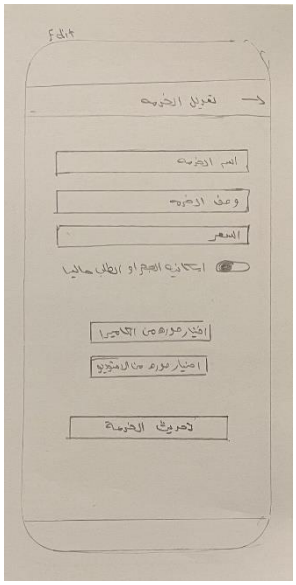


Figure43 Low Prototype of (Edit service page)

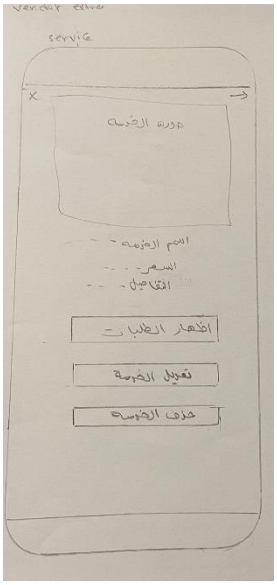


Figure46 Low Prototype of (Service details page)

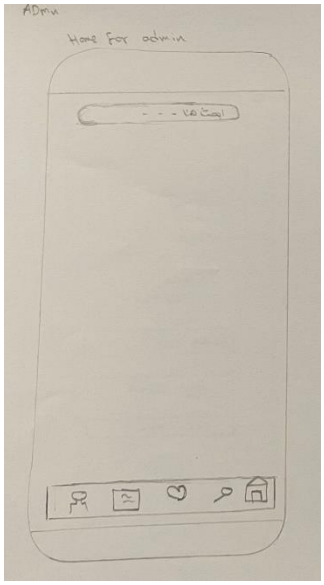


Figure44 Low Prototype of (Admin's Home page)

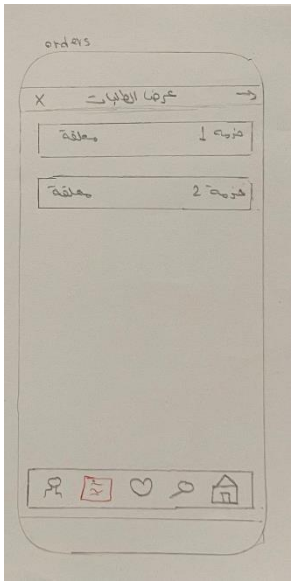


Figure45 Low Prototype of (Orders page)

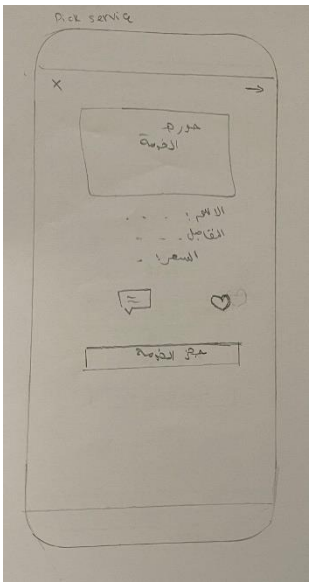


Figure49 Low Prototype of (Pick Service page)

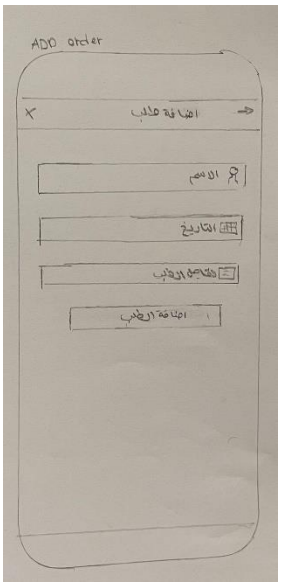


Figure48 Low Prototype of (Add order page)

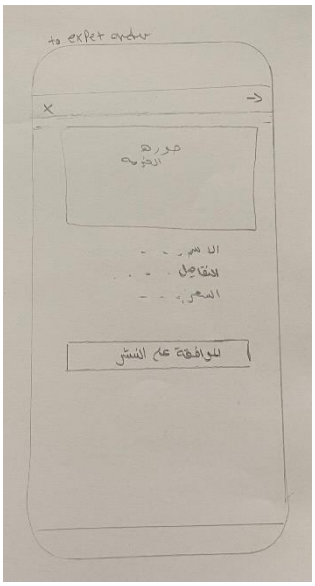


Figure47 Low Prototype of (order authentication page)

4.7. User Interfaces

4.7.1. Welcome Pages and Sign up/Sign in

Figure52 UI of (Sign up)

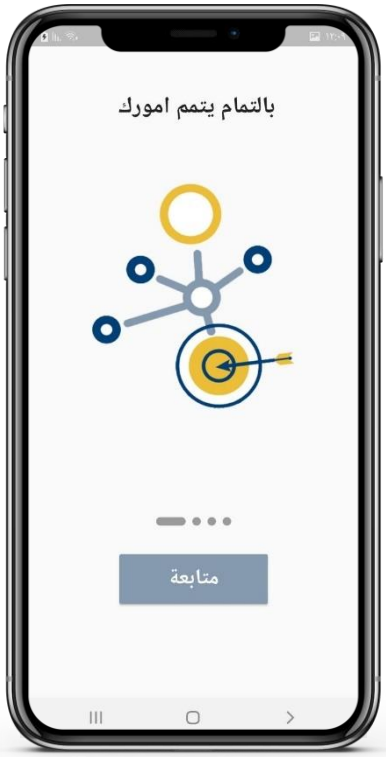


Figure51 UI of (Welcome1)

Figure50 UI of (Welcome2)

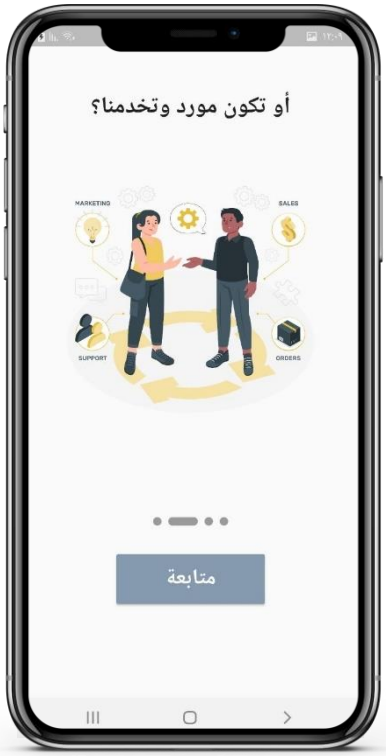


Figure54 UI of (Welcome3)

Figure55 UI of (Welcome4)

Figure53 UI of (Sign in)

4.7.2. Vendor’s Pages

Figure58 UI of (Vendor's Services)

Figure57 UI of (Service Details)

Figure56 UI of (Edit Service)

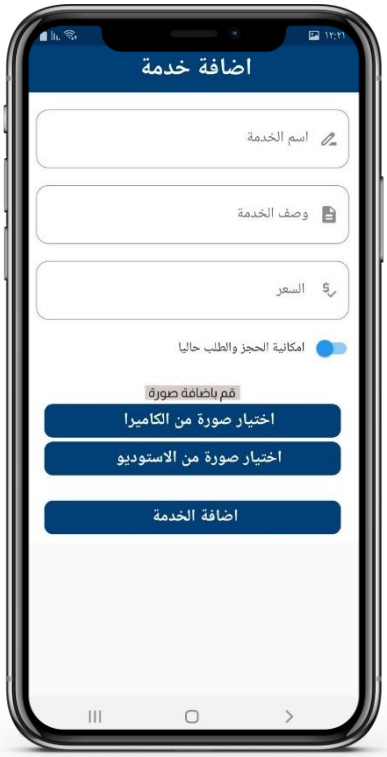


Figure60 UI of (Add Service)

Figure59 UI of (Order's Detail)

4.7.3. Admin's Pages

Figure 62 UI of (Publication approval)

Figure 61 UI of (Pending orders)

4.7.4. Sponsor's Pages

Figure 64 UI of (Sponsor's Chat)

Figure 63 UI of (Sponsor Info)

4.7.5. Planner’s Pages

Figure 67 UI of (Search on packages)

Figure 66 UI of (Favorite page)

Figure 65 UI of (Add order)

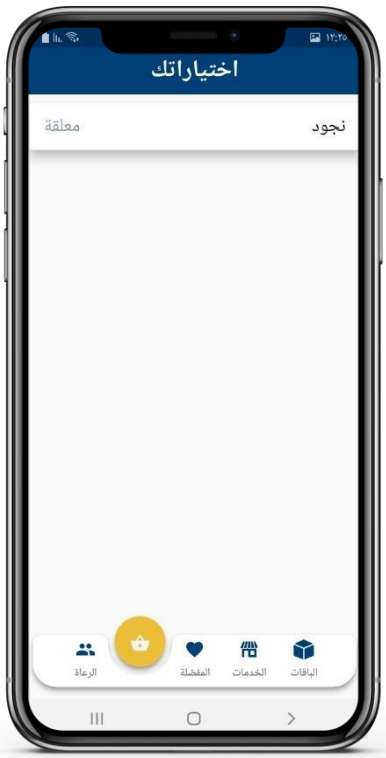


Figure 68 UI of (Planner's pending orde

Figure 70 UI of (Search on services)

Figure 69 UI of (Planner's Chat)

Figure 72 UI of (Comments)

Figure 71 UI of (Search)

Figure 74 UI of (Package's detail)

Figure 73 UI of (Sponsors list)

4.8. Conclusion

At the end of this chapter, we have covered and explained the functions of the system by designing a class diagram as well as the ordered sequential process of the system when the planner and vendor register in the system by designing a sequence diagram.

Lastly, sprint 0 has been completed, the next chapter will be representing Sprint 1 where we will identify and discuss the most important functional requirements to start building and implementing the project programmatically.

CHAPTER 5
METODOLOGY

5.1. Introduction

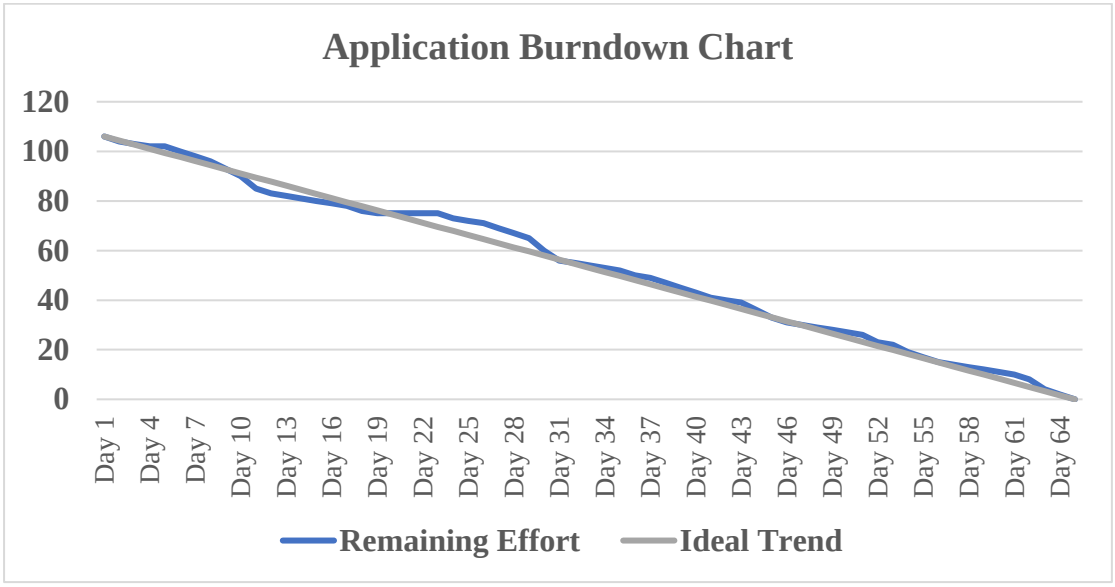
In this chapter we will present a report of our progress in the development and implementation of our software. First, the product backlog represents the milestones of our app. Second, the sprints backlog represents the tasks or issues chosen for the current sprint that we delivered. Third: the tasks allocation and the burnt down chart to represent the workflow.

5.2. Product Backlog

Table 15 Product Backlog

FR No.	Functional Requirement	Priority
User		
FR1	The User shall be able to **Sign up**	**M**
FR2	The User shall be able to **Sign in**	**M**
FR3	The User shall be able to **Get Notification**	**M**
Admin		
FR1	The Admin must be able to **Manage Services and packages**	**H**
Sponsor		
FR1	The Sponsor shall be able to **Contact planners via chat messages**	**M**
Vendor		
FR1	The Vendor shall be able to register on the app (**Sign up)**	**M**
FR2	The Vendor must be able to **sign in**	**M**
FR3	The Vendor must be able to **Create services**	**H**
FR4	The Vendor shall be able to Create **packages**	**M**
FR5	The Vendor must be able to **Manage booking requests**	**H**
FR6	The Vendor must be able to **Manage orders status**	**M**
FR7	The Vendor must be able to **Update service and package**	**M**
FR8	The Vendor must be able to **Delete service and package**	**M**

FR No.	Functional Requirement	Priority
Planner		
FR1	The Planner shall be able to register on the app (**Sign up)**	**M**
FR2	The Planner must be able to **sign in**	**M**
FR3	The Planner shall be able to **Browse vendors**	**H**
FR4	The Planner shall be able to **Search for services and packages**	**M**
FR5	The Planner shall be able to **Book a service**	**H**
FR6	The Planner shall be able to **Track orders**	**M**
FR7	The Planner shall be able to **Contact sponsor**	**M**
FR8	The Planner shall be able to **Add comments**	**M**
FR9	The Planner shall be able to **Add to favorites**	**M**



Application Burndown Chart

5.3. Sprint 0

S0| Sprint Backlog

Table 16 S0| Sprint Backlog

FR No.	Functional Requirement	Priority
Admin		
FR1	The Admin must be able to **Sign in**	**M**
FR2	The Admin must be able to **Manage Services and packages**	**H**
Vendor		
FR1	The Vendor shall be able to register on the app (**Sign up)**	**M**
FR2	The Vendor must be able to **sign in**	**M**
FR3	The Vendor must be directed to vendor's home page	**M**
Planner		
FR1	The Planner shall be able to register on the app (**Sign up)**	**M**
FR2	**The Planner must be able to sign in**	**M**
FR3	The Planner must be directed to Planner's home page	**M**

S0| Task & their Allocation

Table 17 S0| Task & their Allocation

Task No.	Tasks	Task Allocation
Report		
T1	Update Project1 Report	Everyone
T2	Document Methodology	Sarah Alsubhi
T3	Document Analysis and Design	Nojood Alamri Sarah Ahmed
T4	Document Implementation	Lama Alshaikh
T5	Document Testing	Asmaa Felemban
Coding		
T6	Create Users Classes	Sarah Ahmed
T7	Create Authentication Function	Nojood Alamri
T8	Create Repository Classes And Connect Them With Firebase	Lama Alshaikh
T9	Create Sign up Function	Sarah Alsubhi

S0| Burndown Chart

Table18 S0| Sprint Burndown Table

Backlog ID	Task	Initial Estimate	10-Dec	11-Dec	12-Dec	13-Dec	14-Dec	15-Dec	16-Dec	17-Dec	18-Dec	19-Dec	20-Dec	21-Dec	22-Dec	23-Dec	24-Dec	25-Dec	26-Dec	27-Dec
		Day 0	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14	Day 15	Day 16	Day 17	Day 18
1	Update project1 report	8	1	1	1		1	1	1	1	1									
2	Document Methodology	2	1																	1
3	Document Analysis and Design	4					1	1	1	1										
4	create users classes	1										1								
5	create authentication function	2										1	1							
6	create repository classes and connect them with firebase	3										1	1	1						
7	create sign up function	3								1	1	1								
8	create login function	2									1	1								
9	role based authentication	4													1	1	1	1		
10	Document Implementation	1																	1	
11	Document Testing	1																	1	
Remaining Effort		31	29	28	27	27	25	23	21	18	15	10	8	7	6	5	4	3	1	0
Ideal Trend		31	29.2778	27.5556	25.8333	24.1111	22.3889	20.6667	18.9444	17.2222	15.5	13.7778	12.0556	10.3333	8.61111	6.88889	5.16667	3.44444	1.72222	0

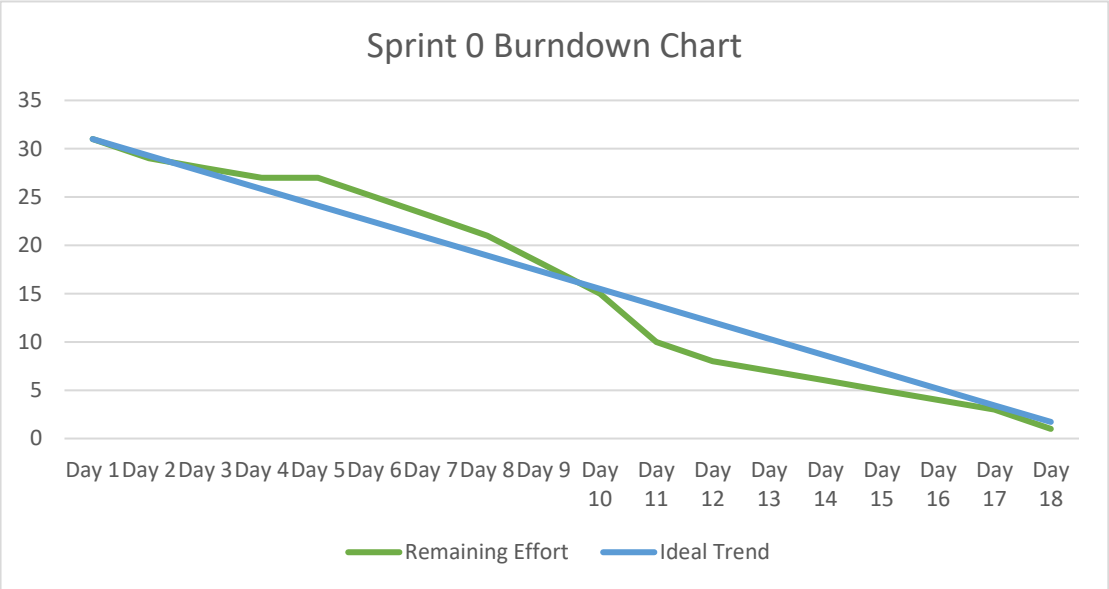


Figure 75 S0| Sprint Burndown Chart

5.4. Sprint 1

S1| Sprint Backlog

Table 19 S1| Sprint Backlog

FR No.	Functional Requirement	Priority
Vendor		
FR1	The Vendor must be able to **Create services**	**H**
FR2	The Vendor shall be able to Create **packages**	**M**
FR3	The Vendor must be able to **Update service and package**	**M**
FR4	The Vendor must be able to **Delete service and package**	**M**

S1| Task & their Allocation

Table 20 S1| Task & their Allocation

Task No.	Tasks	Task Allocation
Report		
T1	Document Methodology	Asmaa Felemban
T2	Document Analysis and Design	Sarah Alsubhi Lama Alshaikh
T3	Document Implementation	Nojood Alamri
T4	Document Testing	Sarah Ahmed
Coding		
T5	Implement Vendor Creating Services	Sarah Ahmed
T6	Implement Vendor Creating Packages	Nojood Alamri
T7	Implement Vendor Updating Service and Package	Lama Alshaikh
T8	Implement Vendor Deleting Service and Package	Asmaa Felemban

S1| Burndown Chart

Table 21 S1| Sprint Burndown Table

Backlog ID	Task	nitial Estimaite	1-Jan	2-Jan	3-Jan	4-Jan	5-Jan	6-Jan	7-Jan	8-Jan	9-Jan	10-Jan	11-Jan	12-Jan
		Day 0	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12
1	implement vendor to create services	5	1	1	1	1	1							
2	implement vendor create packages	4				1	1	1	1					
3	implement vendor update service and package	2						1	1					
4	implement vendor delete service and package	2							1	1				
5	update document analysis and design	3							1	1	1			
6	update document Implementation	2							1	1				
7	update document testing	2								1		1		
8	update document methodology	3	1										1	1
Remaining Effort		**23**	**21**	**20**	**19**	**17**	**15**	**13**	**8**	**4**	**3**	**2**	**1**	**0**
Ideal Trend		**23**	**21.0833**	**19.1667**	**17.25**	**15.3333**	**13.4167**	**11.5**	**9.58333**	**7.66667**	**5.75**	**3.83333**	**1.91667**	**0**

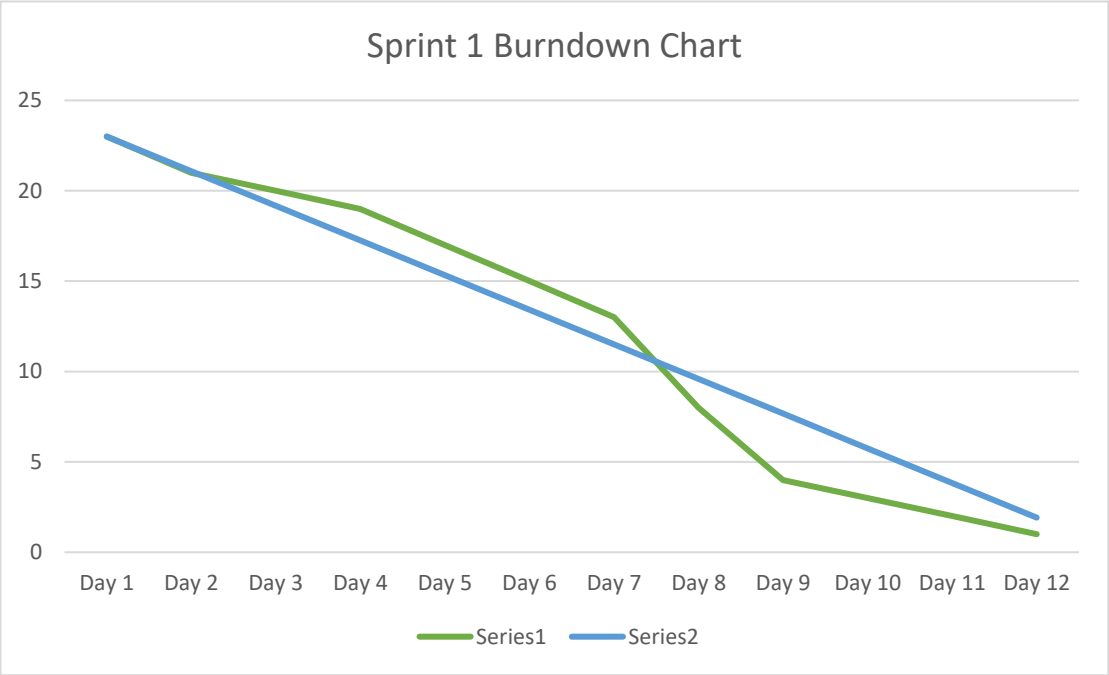


Figure 76 S1| Sprint Burndown Chart

5.5. Sprint 2

S2| Sprint Backlog

Table 22 S2| Sprint Backlog

FR No.	Functional Requirement	Priority
Vendor		
FR1	The Vendor must be able to **Manage booking requests**	**H**
FR2	The Vendor must be able to **Manage orders status**	**M**
Planner		
FR1	The Planner shall be able to **Browse vendors**	**H**
FR2	The Planner shall be able to **Search for services and packages**	**M**
FR3	The Planner shall be able to **Add comments**	**M**

S2| Task & their Allocation

Table 23 S2| Task & their Allocation

Task No.	Tasks	Task Allocation
Report		
T1	Document Methodology	Nojood Alamri
T2	Document Analysis and Design	Asmaa Felemban
T3	Document Implementation	Sarah Ahmed
T4	Document Testing	Lama Alshaikh
Coding		
T5	Implement Planner Viewing Services and Packages	Lama Alshaikh
T6	Implement Search Feature	Sarah Ahmed
T7	Implement Commenting On Services or Packages	Sarah Alsubhi
T8	Implement Booking a Services or Packages	Nojood Alamri
T9	Implement Vendor Managing Booking Requests	Asmaa Felemban
T10	Implement Vendor Updating Orders Status	Sarah Alsubhi

S2| Burndown Chart

Table 24 S2| Sprint Burndown Table

Sprint 2 Burndown Chart																				
Backlog ID	**Task**	**Initial Estimate**	**13-Jan**	**14-Jan**	**15-Jan**	**16-Jan**	**17-Jan**	**18-Jan**	**19-Jan**	**20-Jan**	**21-Jan**	**22-Jan**	**23-Jan**	**24-Jan**	**25-Jan**	**26-Jan**	**27-Jan**	**28-Jan**	**29-Jan**	**30-Jan**
		Day 0	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**	**Day 6**	**Day 7**	**Day 8**	**Day 9**	**Day 10**	**Day 11**	**Day 12**	**Day 13**	**Day 14**	**Day 15**	**Day 16**	**Day 17**	**Day 18**
1	implement planner viewing services and packages	**4**	1	1	1	1														
2	implement search feature	**4**			1	1	1	1												
3	implement commenting on services or packages	**2**					1	1												
4	book a service or package	**4**							1	1	1	1								
5	implement vendor managing booking requests	**3**									1	1	1							
6	implement vendor updating orders status	**5**									1	1	1	1	1					
7	update document analysis and design	**3**														1	1	1		
8	update document Implementation	**1**																	1	
9	update document testing	**1**																	1	
10	update document methodology	**3**	1																1	1
Remaining Effort		**30**	**28**	**27**	**25**	**23**	**21**	**19**	**18**	**17**	**14**	**11**	**9**	**8**	**7**	**6**	**5**	**4**	**1**	**0**
Ideal Trend		**30**	**28.3333**	**26.6667**	**25**	**23.3333**	**21.6667**	**20**	**18.3333**	**16.6667**	**15**	**13.3333**	**11.6667**	**10**	**8.33333**	**6.66667**	**5**	**3.33333**	**1.66667**	**0**

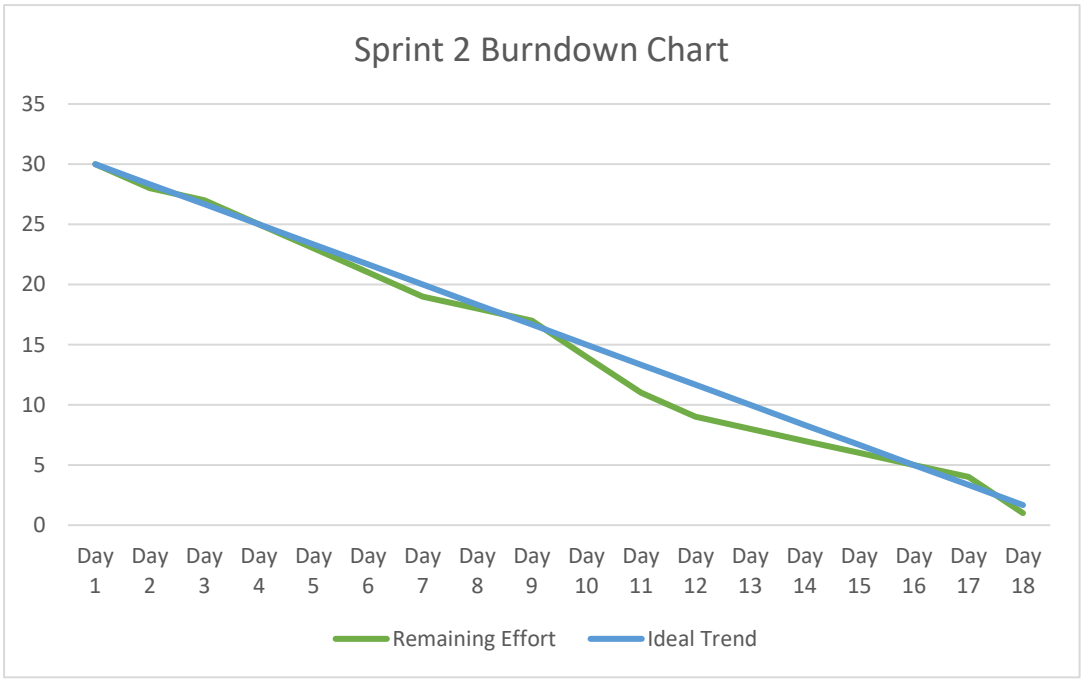


Figure 77 50 S2| Sprint Burndown Chart

5.6. Sprint 3

S3| Sprint Backlog

Table 25 S3| Sprint Backlog

FR No.	Functional Requirement	Priority
Sponsor		
FR1	The Sponsor shall be able to **Contact planners via chat messages**	**M**
Planner		
FR1	The Planner shall be able to **Track orders**	**M**
FR2	The Planner shall be able to **Contact sponsor**	**M**
FR3	The Planner shall be able to **Add to favorites**	**M**

S1| Task & their Allocation

Table 26 S3| Task & their Allocation

Task No.	Tasks	Task Allocation
Report		
T1	Document Methodology	Sarah Ahmed
T2	Document Analysis and Design	Lama Alshaikh Asmaa Felemban
T3	Document Implementation	Nojood Alamri
T4	Document Testing	Sarah Alsubhi
T5	Organizing The Whole Report	Asmaa Felemban
Coding		
T5	Implement Track Orders For Planner	Sarah Alsubhi
T6	Implement Planner Contact Sponsor Via Chat	Nojood Alamri Lama Alshaikh Asmaa Felemban
T7	Implement Adding Services or Packages to Favorites	Sarah Ahmed

S3| Burndown Chart

Table27 S3| Sprint Burndown Table

Sprint 3 Burndown Chart													
Backlog ID	**Task**	**Initial Estimaite**	**31-Jan**	**1-Feb**	**2-Feb**	**3-Feb**	**4-Feb**	**5-Feb**	**6-Feb**	**7-Feb**	**8-Feb**	**9-Feb**	**10-Feb**	**11-Feb**
		Day 0	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**	**Day 6**	**Day 7**	**Day 8**	**Day 9**	**Day 10**	**Day 11**	**Day 12**
1	implement track orders for planner	3	1	1	1									
2	implement planner contact sponsor via chat	5		1	1	1	1	1						
3	implement adding services or packages to favorites	3							1	1	1			
4	update document analysis and design	2									1	1		
5	update document Implementation	1										1		
6	update document testing	1										1		
7	update document methodology	3	1										1	1
8	organizing the whole report	4	1									1	1	1
Remaining Effort		**22**	**19**	**17**	**15**	**14**	**13**	**12**	**11**	**10**	**8**	**4**	**2**	**0**
Ideal Trend		**22**	**20.1667**	**18.3333**	**16.5**	**14.6667**	**12.8333**	**11**	**9.16667**	**7.33333**	**5.5**	**3.66667**	**1.83333**	**0**

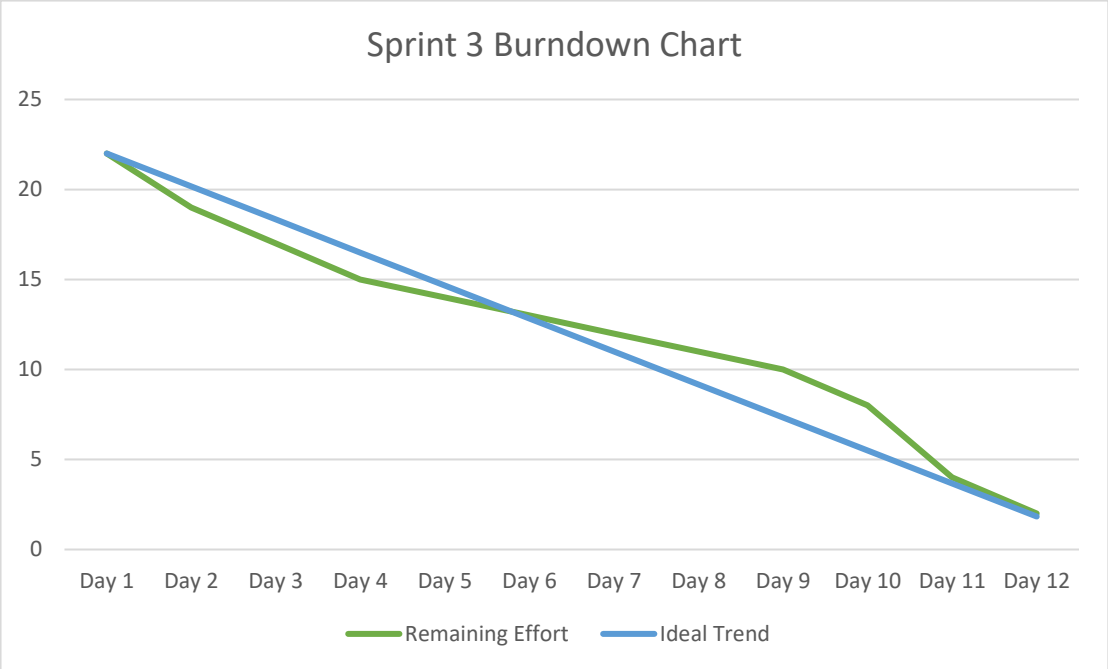


Figure 78 S3| Sprint Burndown Chart

5.7. Conclusion

To summarize, the development process of our app is going as planned in the product and sprint backlog described above.

CHAPTER 6

IMPLEMENTAION AND DEBUGGING

6.1. Introduction

The implementation of our application will be presented in this chapter.This chapter will present the implementation of our application. We will present and describe the code of for the essential functions that supporting the project's objectives. We will also present discuss the code debugging and troubleshooting issues, as well as along with its their solutions.

6.2. Code Implementation

6.2.1. Sprint 0

```
//create account for the user using email and password
firebaseAuth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                progressBar.setVisibility(View.GONE);
                if (task.isSuccessful()) {
                    //if account successfully created initialize firebaseUser
                    //and get user ID
                    firebaseUser = firebaseAuth.getCurrentUser();
                    assert firebaseUser != null;
                    userID = firebaseUser.getUid();

                    //add to database based on user type
                    if(selectedRadioButton == radioButtonPlanner){
                        user = new Planner(email);
                        user.setUserRole("planner");
                    }
                    else if (selectedRadioButton == radioButtonVendor){
                        user = new Vendor(email);
                        user.setUserRole("vendor");
                    }
                    else {
                        user = new Sponsor(email);
                        user.setUserRole("sponsor");
                    }
                    usersRepo.addNewUser(user, userID);
```

Figure 79 Create account for the user using email and password

Using firebase authentication to create the user's account by e-mail and password. If the authentication was successful, the user ID is generated automatically by firebase. It will initialize the user object based on the role

selected by the user. The user object and ID will then be provided to another method for database addition.

```
public void addNewUser(User user, String UID){
    firebaseFirestore.collection( collectionPath: "users").document(UID).set(user);
}
```

Figure 80 Add user data

This method will receive the user object and user ID as parameters, and it will create a document in the database with the specified ID and a field containing the object's values.

```
public void authorizeUser(String userID) {
    documentReference = firebaseFirestore.collection( collectionPath: "users").document(userID);
    documentReference.get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>(){
        @Override
        public void onSuccess(DocumentSnapshot documentSnapshot) {
            Log.d( tag: "TAG", msg: "OnSuccess" + documentSnapshot.getData());
            String userRole = documentSnapshot.getString( field: "userRole");
            assert userRole != null;
            switch (userRole) {
                case "planner": {
                    Intent intent = new Intent(getApplicationContext(), PlannerMain.class);
                    startActivity(intent);
                    finish();
                    break;
                }
                case "vendor": {
                    Intent intent = new Intent(getApplicationContext(), VendorMain.class);
                    startActivity(intent);
                    finish();
                    break;
                }
                case "sponsor": {
                    Intent intent = new Intent(getApplicationContext(), SponsorMain.class);
                    startActivity(intent);
                    finish();
                    break;
                }
                default:
                    Toast.makeText( context: Login.this, text: "حدث خطأ أثناء استرجاع معلومات الحساب",
                            Toast.LENGTH_SHORT).show();
                    break;
            }
```

Figure 81 Role based authorization

This method retrieve data from database. It accesses the field user role within the document with the specified ID, which is part of the collection named users and then shows a home page for each user based on his role.

In the figure below (Admin role), we have the admin class that extends Statefluwidget. In this class we approve the vendor's services. First, we look for instances where the service package publishing is "0, " in order to display it for the administrator. A publishing value of (0) indicates that the administrator has not yet provided authentication. Then, if the service or the package matches the role, show it to the adminstrator as (طلبات معلقة). The publishing value changes to "1" when the admin approves a service or package.

```
class AdminRole extends StatefulWidget {
  const AdminRole({super.key});

  @override
  State<AdminRole> createState() => _AdminRoleState();
}

class _AdminRoleState extends State<AdminRole> {
  final Stream<QuerySnapshot> _usersStreamPackage = FirebaseFirestore.instance
      .collection('PackageServices').where('Publishing',isEqualTo:"0")
      .snapshots();
  final Stream<QuerySnapshot> _usersStreamService = FirebaseFirestore.instance
      .collection('Service').where('Publishing',isEqualTo:"0")
      .snapshots();
```

82 Figure Admin role

```
@override
Widget build(BuildContext context) {
  print("==========================");
  print(sharedpref.getString("role"));
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Colors.black87,
      elevation: 0,
      title: Text('الطلبات المعلقة',style:GoogleFonts.getFont('Almarai'),),
      leading: IconButton(
        icon: Icon(Icons.arrow_back),
        onPressed: () {
      print(FirebaseAuth.instance.currentUser!.uid);
          Get.to(()=>BottomNavigation());
        },
      ), // IconButton
      centerTitle: true,
      actions: [
        IconButton(
          icon: Icon(Icons.exit_to_app),
          onPressed: () async{
           await  FirebaseAuth.instance.signOut();
           Get.to(()=>Signin());
          },
        ), // IconButton
```

83 Figure Admin role

6.2.2. Sprint 1

```
Add() async {
  if (formstate.currentState!.validate()) {
    url = await uploadImage();
    ref.add({
      "name": name.text.trim(),
      "desc": desc.text.trim(),
      "salary": salary.text.trim(),
      "booking": isSwitched.toString(),
      "imageurl": url.toString(),
      "Publishing":"0",
      "user_id": FirebaseAuth.instance.currentUser!.uid.toString()
    }).then((value) {
      Get.to(() => ShowPackage());
    }).catchError((e) {
      print(e);
    });
  }
}
}
```

84 Figure Add service/package

In this figure above (Add service or package), we have the code for adding a service or package by retreving the details, storing them in firebase and then sending it into showpackege to see it if its matches the admin role (Same steps for ADD Service).

```
edit() async {

  if (myfile == null) {
   await  ref.doc(widget.ID_doc).update({
      "name": name.text.trim(),
      "desc": desc.text.trim(),
      "salary": salary.text.trim(),
      "booking": isSwitched.toString(),

    }).then((value) {
      Get.to(() => ShowPackage());
    }).catchError((e) {
      print(e);
    });
  }
```

85 Figure Update service/package

In this figure (Update Service or package), the code for updating the services or packages is displayed. This code retrieves the new data and saves or updates NOTE:(THE SAME UPDATING STEPS WERE APPLIED TO THE DELETE).

```
  final ID_doc;
  const detilesOrder({super.key, this.data, this.ID_doc});

  @override
  State<detilesOrder> createState() => _detilesOrderState();
}

class _detilesOrderState extends State<detilesOrder> {
  CollectionReference ref = FirebaseFirestore.instance.collection("Orders");
  var options = ['معلقة', 'قيد الانجاز', 'تم الموافقة والحجز'];
  var _currentItemSelected = "";
  var updatval = "معلقة";

  @override
  void initState() {
    _currentItemSelected = widget.data['status'];
    updatval = widget.data['status'];
    super.initState();
  }
```

86 Figure Change statues

```
  updateOrder() async {
    await ref.doc(widget.ID_doc).update({
      "status": updatval,
    }).then((value) {
      Get.snackbar("تم التحديث", "", snackPosition: SnackPosition.BOTTOM);
    }).catchError((e) {
      print(e);
    });
  }
}
```

87 Figure Change statues

In the figures above **(Change statues)**, the code of changing the status of order by vendors is displayed. This code will update order status on service applicant if the vendor changes or update the order status.

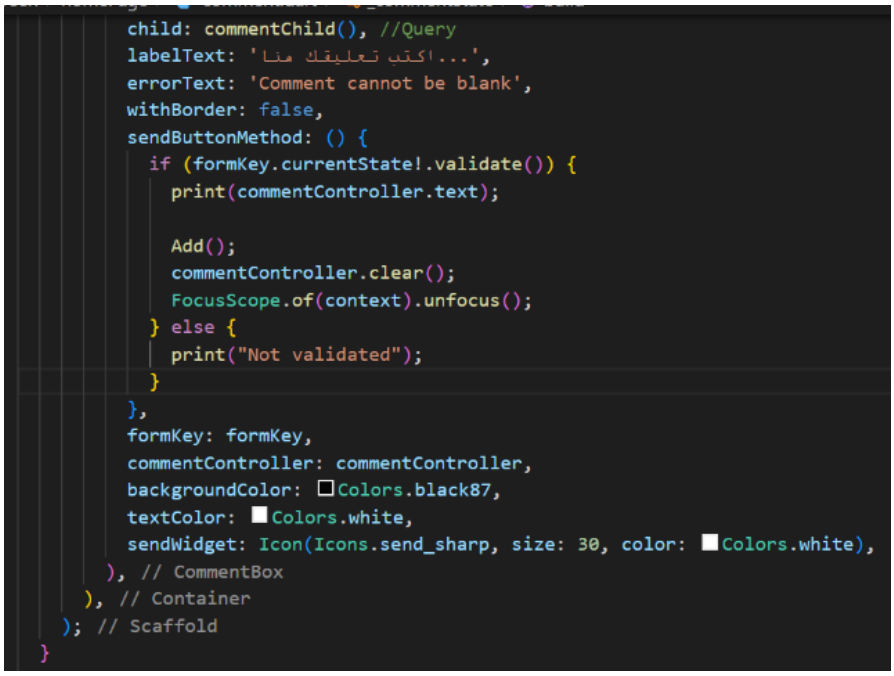
6.2.3. Sprint 2

```
Add() async {
  if (formstate.currentState!.validate()) {
    ref.add({
      "user_id": FirebaseAuth.instance.currentUser!.uid,
      "service_Id": widget.ID_doc,
      "name": name.text.trim(),
      "date": date.text.trim(),
      "datails": details.text.trim(),
      "status": "معلقة"
    }).then((value) {
      Get.snackbar("تمت العملية بنجاح  ", " ",
        colorText: Colors.white,
        backgroundColor: Colors.lightBlue,
        icon: const Icon(Icons.add_alert),
        snackPosition: SnackPosition.BOTTOM);
    }).catchError((e) {
      print(e);
    });
  }
 }
}
```

88 Figure Add order

In the above figure (**Add order**), The code of adding an order or booking a service is displayed. The service will be booked based on the userID and the servicesID including other details (name, date, details). The new order the status will be "معلقة" until the services provider accept the order and change the order status.

In the figure below (**Add comments for service or packages)**, we have the code of adding comments for services or packages. The user will add a text then the text and the service id will be added to firebase to save it and show it as a feedback.



```
        child: commentChild(), //Query
        labelText: 'اكتب تعليقك هنا...',
        errorText: 'Comment cannot be blank',
        withBorder: false,
        sendButtonMethod: () {
          if (formKey.currentState!.validate()) {
            print(commentController.text);

            Add();
            commentController.clear();
            FocusScope.of(context).unfocus();
          } else {
            print("Not validated");
          }
        },
        formKey: formKey,
        commentController: commentController,
        backgroundColor: Colors.black87,
        textColor: Colors.white,
        sendWidget: Icon(Icons.send_sharp, size: 30, color: Colors.white),
      ), // CommentBox
    ), // Container
  ); // Scaffold
}
```

89 Figure Add comment

```
fetchData(String searchFild) async {
  _usersStreamPackage = FirebaseFirestore.instance
      .collection('Comments')
      .where("service_Id", isEqualTo: searchFild)
      .snapshots();
}

Add() async {
  ref
      .add({
        "text": commentController.text,
        "service_Id": widget.ID_doc,
      })
      .then((value) {})
      .catchError((e) {
        print(e);
      });
}
}
```

90 Figure Add comment

6.2.4. Sprint 3

```
class MessageStreamBuilder extends StatelessWidget {
  const MessageStreamBuilder({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return StreamBuilder<QuerySnapshot>(
      stream: _firestore.collection('messages').orderBy('time').snapshots(),
      builder: ((context, snapshot) {
        List<Messageline> messageWidgets = [];
        if (!snapshot.hasData) {
          //add her a spinner
        }

        final messages = snapshot.data!.docs.reversed;
        for (var message in messages) {
          final Messagetext = message.get('text');
          final messageSender = message.get('sender');
```

91 Figure Chat

In this part we use stream: _firestore.collection(“massages”).orderBy(“time”) to make the order of massages between planners and sponsors by time .and get the text of the massage the sender.

```
  @override
  State<Favorite> createState() => _FavoriteState();
}

class _FavoriteState extends State<Favorite> {
  final Stream<QuerySnapshot> _usersStreamPackage =
      FirebaseFirestore.instance.collection('Favorites').where("user_id", isEqualTo: FirebaseAuth.instance.currentUser!.uid).snapshots();
        CollectionReference ref = FirebaseFirestore.instance.collection("Favorites");
```

92 Figure Add into favorite

In that figure (Favorite), we have the favorites code based on the userID add this item (service or package) into favorites collection.

6.3. Code Debugging

Table 28 Code debugging

Problem	Solution
User IDs that are automatically produced by Firebase authentication differ from the ID of the user document in the database, and Firebase does not support custom IDs in the authentication process	Authenticate the user before obtaining this ID and assigning it to the user record in the database.

Also, there was a problem with the sequence of messages on the chat screen; the messages appeared in a disorganized order. The solution is to add orderBy("time") so that the messages appear in consecutive order.

93 Figure massages order

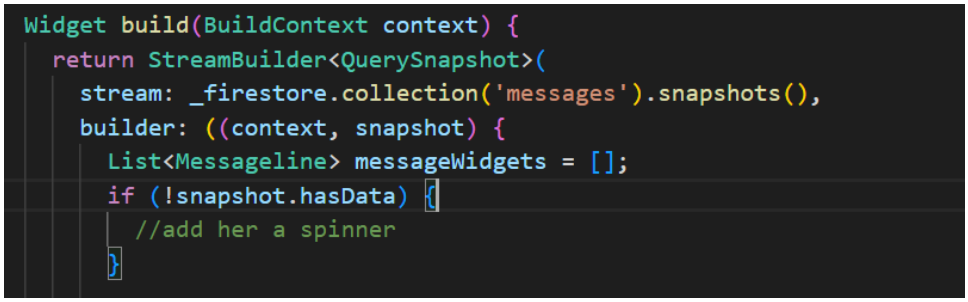


Figure 94 massages order

6.4. Conclusion

We discussed the code implementation process in this chapter. We provided code snippets of the most important functions along with their descriptions. We also presented code debugging issues and solutions.

CHAPTER 7

TESTING

7.1. Introduction

Testing is an essential component of the software development life cycle since it identifies flaws and issues that must be resolved prior to product release. In this part, we outline the unit testing, integration testing, system testing, and usability testing performed throughout each sprint.

7.2. Sprint 0

S0| Unit Testing

Table 29 S0| Unit Testing

Test case ID	Description	Steps	Data	Expected results	Actual results	pass/fail
T01	Testing registering a new user with valid data	1- run the app and choose register a new user 2- input email and password and password validation 3- choose type of user as “planner”. 4-click register	Email: “nada.plan@gmail.com” Password: “Np5767”	1-The registration successful notification appears 2-app navigate to main page.	Same as expected.	pass
T02	Testing registering an already registered user	1- run the app and choose register a new user 2- input used email and password and password validation 3- choose type of user as “planner”. 4-click register	Email: “nada.plan@gmail.com” Password: “Np5767”	1- notifying the user that they are already registered.	Same as expected	pass

S0| Integration Testing

Table 30 S0|Integration Testing

Test case ID	Description	Steps	Data	Expected results	Actual results	pass/fail
T03	Testing login for a registered user	1- run the app 2- input email and password 3- choose type of user as “planner”. 4-click register	Email: “nada.plan@gmail.com” Password: “Np5767”	1-The login successful notification appears 2-app navigate to main page.	Same as expected	pass
T04	Testing login with unregistered data	1- run the app 2- input email and password 3- choose type of user as “planner”. 4-click register	Email: “sama.plan@gmail.com” Password: “Sp5767”	1- login unsuccessful notification because you do not have an account	Log in unsuccessful notification apeared but the Reason was not clear	pass

S0| System Testing

Table 31 S0| System Testing

Test case ID	Description	Steps	Data	Expected results	Actual results	pass/fail
T05	Testing saving the data to firebase	1-opening firebase dashboard. 2- navigate to firebase database 3- check users' collection 4- verify that the data was saved successfully, and the password is hidden.	-	Data is saved in the correct format.	As expected.	pass
T06	Testing interfaces on different device Current device size 5.6 The test will be on device size 4.9	1- create a new device emulator with size off 2- run the app on the new emulator	-	Interfaces are scaled correctly	Login page is not completely appearing. Sign up page is looking ok.	pass

S0| Usability Testing

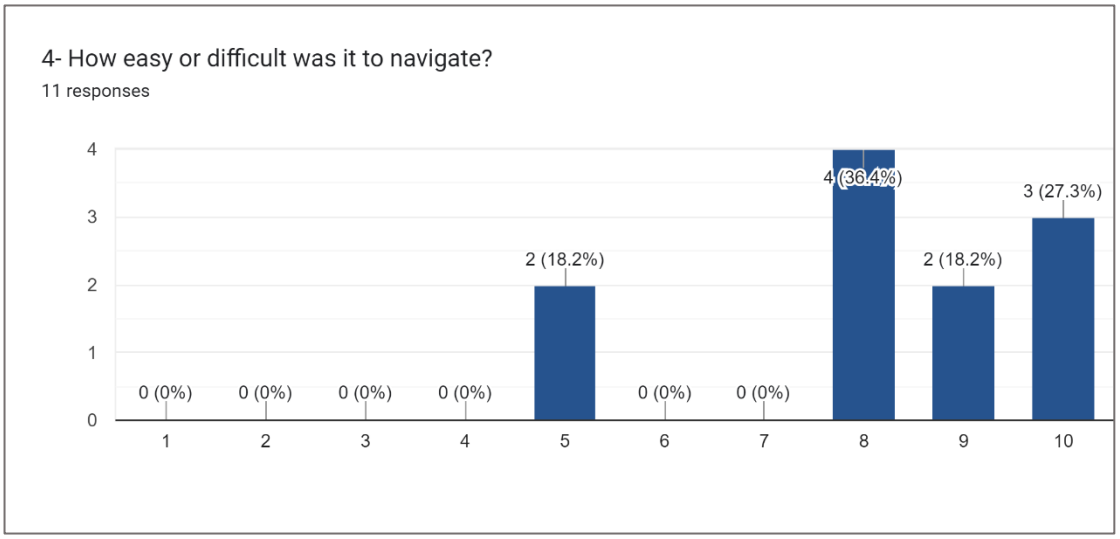


Figure 95 S0| Usability Testing(T07)

As shown in the graph above, user responses varied from 1 (difficult) to 10 (easy). The majority of users select 8 and then 10. These responses tell us that our application is simple to use for the majority of users.

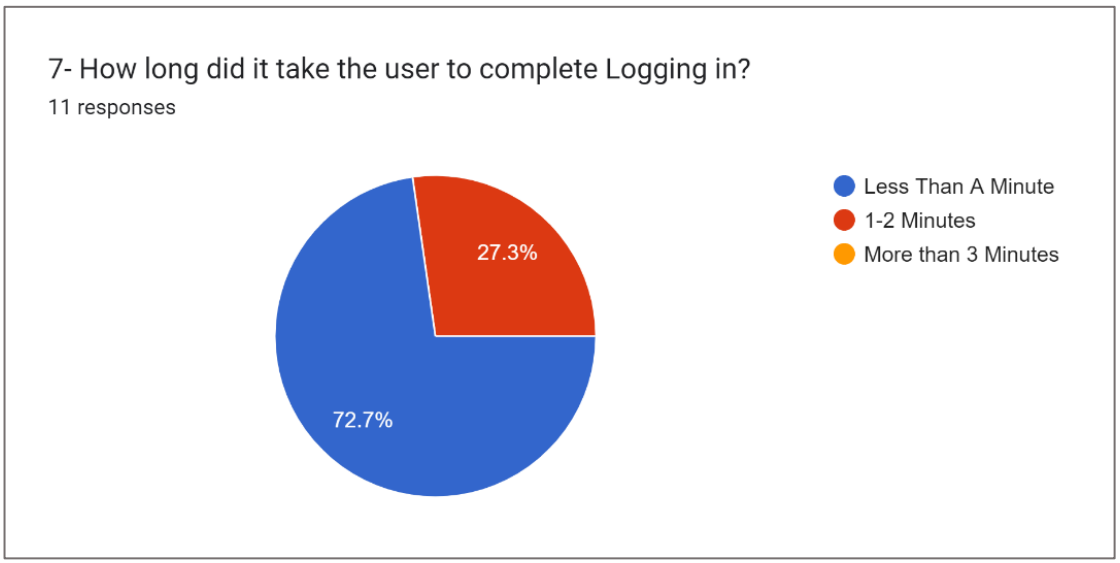


Figure 96 S0| Usability Testing(T08)

Because of the simple information required to login, users can login very quickly, as shown in the figure above.

S1| Unit Testing

Table 32 S1| Unit Testing

Test case ID	Description	Steps	Data	Expected results	Actual results	pass/fail
T01	Testing adding a service or package	1-Registering as "Vendor". 2- In vendor home page click on (إضافة خدمات). 3- Adding service or package information.	Name of service : قاعة احتفالات. Price : 20000 ريال. Details: مكان مناسب لإقامة أفخم الحفلات .	1-The adding a service or package is done 2-app navigate to home page.	Same as expected.	pass
T02	Testing adding an already existing service or package.	1-Registering as "Vendor". 2- In vendor home page click on (إضافة خدمات). 3- Adding service or package information.	Name of service : قاعة احتفالات. Price : 20000 ريال. Details: مكان مناسب لإقامة أفخم الحفلات .	1- notifying the user that they are already added this service.	Same as expected	pass

S1| Integration Testing

Table 33 S1| Integration Testing

Test case ID	Description	Steps	Data	Expected results	Actual results	pass/fail
T03	Testing updating the services or packages	1- In the "Vendor" home page click on service. 2- Service Information appears and to update it we click " تعديل الخدمة". 3- After updating Information we click "تحديث الخدمة".	Name of service : قاعة احتفالات. Price : 40000 ريال. Details: مكان مناسب لإقامة أفخم الحفلات .	1-The updating the services or packages is done 2-app navigate to service page.	Same as expected	pass

S1| System Testing

Table 34 S1| System Testing

Test case ID	Description	Steps	Data	Expected results	Actual results	pass/fail
T04	Testing deleting the services or packages	1-Sign in 1- In the "Vendor" home page click on service. 2- Service Information appears and to update it we click " حذف الخدمة".	-	Data is deleted	As expected.	pass

S1| Usability Testing

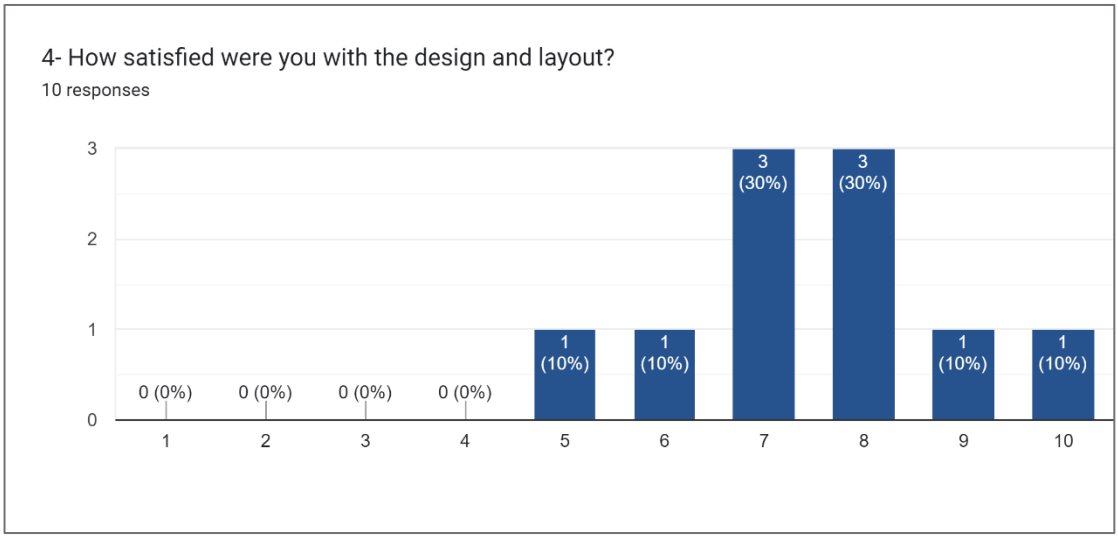


Figure97 S1| Usability Testing(T05)

As seen in the preceding graph, people ranked the design and layout as follows on a scale from 1 (difficult) to 10 (easy): 30% of users select 8 and 7, followed by 10% who select 5, 6, 9, and finally 10 These responses have revealed that the layout and design are good but might need some enhancements.

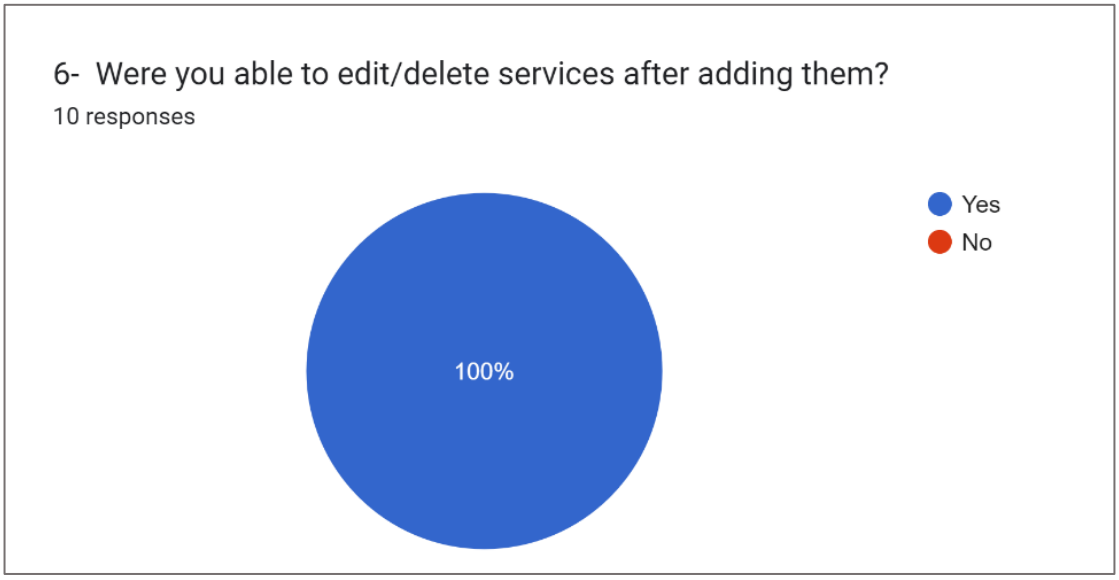


Figure98 S1| Usability Testing(T06)

The figure Above show us that all users agreed with being able to edit/delete services after adding them.

7.3. Sprint 2

S2| Unit Testing

Table 35 S2| Unit Testing

Test case ID	Description	Steps	Data	Expected results	Actual results	pass/fail
T01	Testing adding comments to service or packages	1-Registering as "Planner". 2- In planner home page click on specific service. 3- In service information page we click on "التعليقات". 4- The comments page appears and we will add comments	"I've booked this service before and I like it"	1-Comment is added and other planners can see it	Same as expected.	pass

S2| Integration Testing

Table 36 S2| Integration Testing

Test case ID	Description	Steps	Data	Expected results	Actual results	pass/fail
T01	Testing booking service or packages	1-Registering as “Planner”. 2- In planner home page click on specific service. 3- In service information page we click on “حجز الخدمة.” 4- After we wait for vendor to accept our request.	booking قاعة احتفالات	1-Vendor accepts or decline request 2-planner is notified that service is booked after vendor acceptance. 3- planner is notified that service is not booked after vendor decline.	Same as expected.	pass

S2| System Testing

Table 37 S2| System Testing

Test case ID	Description	Steps	Data	Expected results	Actual results	pass/fail
T03	Testing The search feature	1-Sign in 2- In the planner home page click on the search bar 3- you search by specific service	" قاعة احتفالات"	Data is found	As expected.	pass

S2| Usability Testing

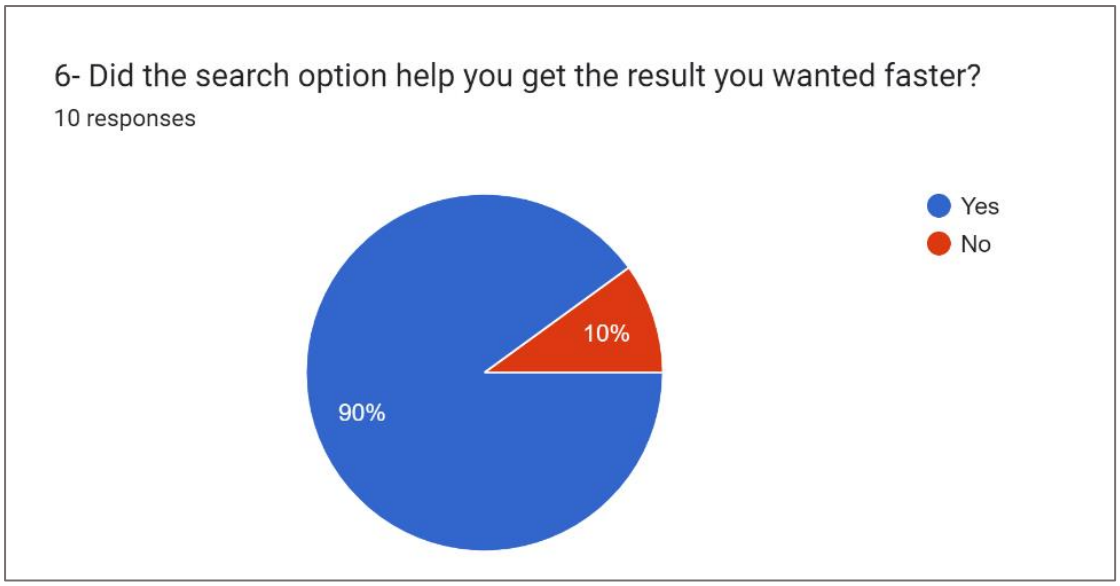


Figure 99 S2| Usability Testing(T05)

feature helped them reach required results faster.

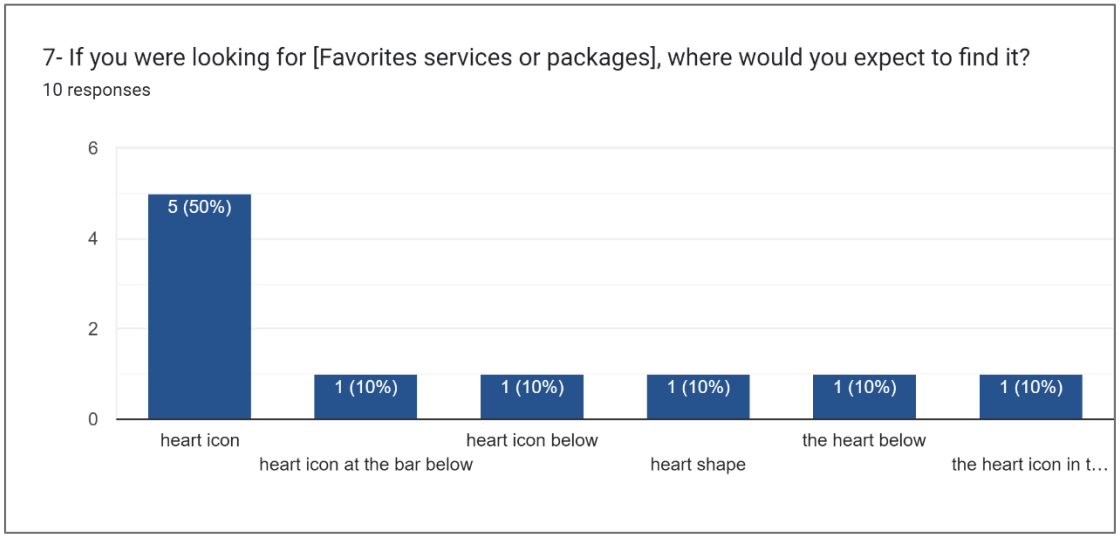


Figure 100 S2| Usability Testing(T06)

what they're looking for. All users knew where to find favorites services or packages.

7.4. Sprint 3

S3| Unit Testing

Table 38 S3| Unit Testing

Test case ID	Description	Steps	Data	Expected results	Actual results	pass/fail
T02	Testing adding the services or packages to favorites	1- In the "Planner" home page click on service. 2- Service Information appears and to add to favorites we click on " إضافة الى المفضلة".	-	1-The service or package is added to favorites.	Same as expected	pass

S3| Integration Testing

Table 39 S3| Integration Testing

Test case ID	Description	Steps	Data	Expected results	Actual results	pass/fail
T02	Testing Change the order status	1- Planner books a service 2- Vendor accepts request. 3- vendor change request status to order and update it for vendor.	-	1-The order status is changed.	Same as expected	pass

S3| System Testing

Table 40 S3| System Testing

Test case ID	Description	Steps	Data	Expected results	Actual results	pass/fail
T05	Testing saving the whole system data to firebase	1-opening firebase dashboard. 2- navigate to firebase database 3- check “Comments Favorites Orders PackageServices Service Users Message” collections 4- verify that any data entered was saved successfully	-	Data is saved in the correct format.	As expected.	pass

S3| Usability Testing

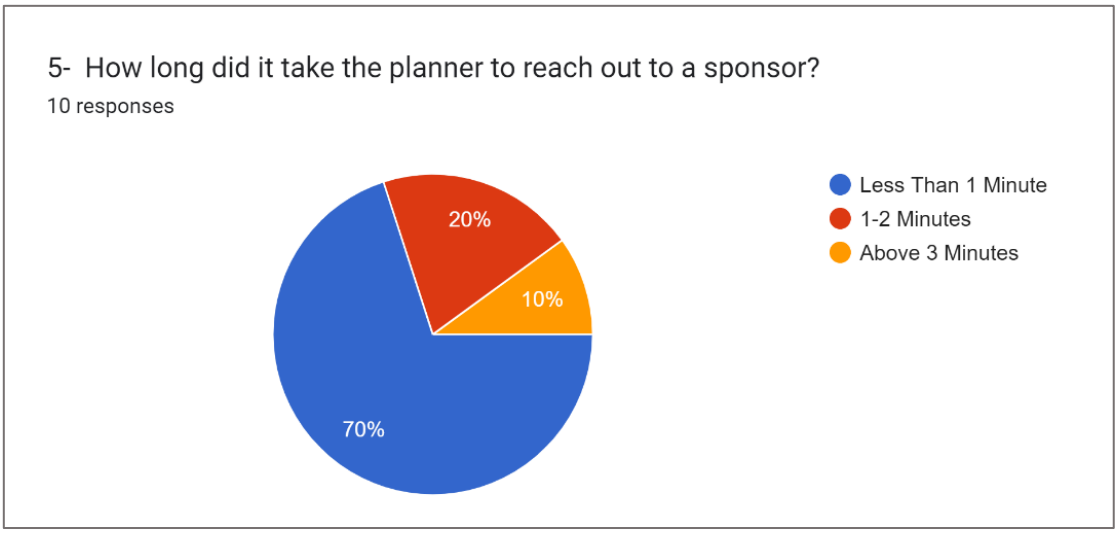


Figure 101 S3| Usability Testing(T03)

70% of users were able to initiate a chat with a sponsor in less than one minute. 20% could complete the task in an average of 1-2 minutes. 10% completed the task in three minutes.

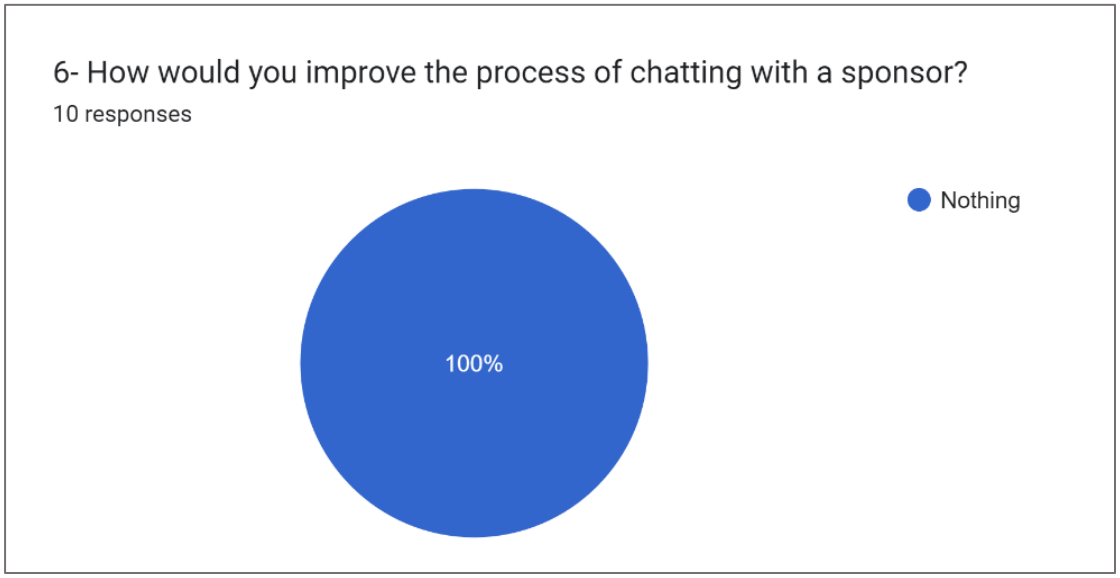


Figure 102 S3| Usability Testing(T04)

All users were satisfied with the chatting with sponsor process.

7.5. Conclusion

We observe that every scenario passed the test. Checking test cases across different forms of testing confirms that our application meets all criteria.