SWE 455 Software Maintenance and Evolution

2nd semester 2023/2024

	Name	ID
1	Nada AlQabbani	442200379
2	Jory Alhassan	442201348
3	Renad AlNumay	442202652
4	Ghadeer AlMansour	442202656
5	Najd AlNafisah	442200293
6	Danh Alkhalaf	442202340

Section#	Different Section
Group#	GROUP 7
Supervisor	Ms. Nouf AlMobarak

Table of content

Introduction

In a world where ensuring our children have fulfilling extracurricular experiences is crucial, the process has become a complex challenge for both activity centers and guardians. The struggle to find suitable programs, coupled with the cumbersome enrollment processes, has given rise to a problem that demands an innovative solution.

Introducing BAHEEJ, an application aimed at transforming the realm of children's extracurricular activities. At its core, BAHEEJ addresses the challenges faced by guardians in discovering and enrolling their children in diverse activities. Traditional methods often involve tedious searches, scattered information, and cumbersome registration procedures, leaving both guardians and centers grappling with inefficiencies.

Our project centers on developing the BAHEEJ app, aiming to solve the issue at hand. The app serves as a platform, empowering centers to easily display their programs. Simultaneously, it offers a user-friendly interface for guardians seeking enriching experiences for their children. The app streamlines the entire process, allowing guardians to not only discover and explore various activities but also to enroll their children seamlessly.

Our target users encompass center owners offering diverse programs for children and guardians seeking enriching experiences for their children.

The value that BAHEEJ adds to the market by addressing the prevalent challenges is multi-faceted. For centers, it provides an opportunity to showcase their offerings to a broader audience, expanding their reach and impact. Simultaneously, it transforms how guardians find, evaluate, and enroll their children in activities.

By seamlessly connecting these two stakeholders, BAHEEJ not only simplifies an intricate process but also enhances the overall efficiency and accessibility of extracurricular activities.

System Screenshots

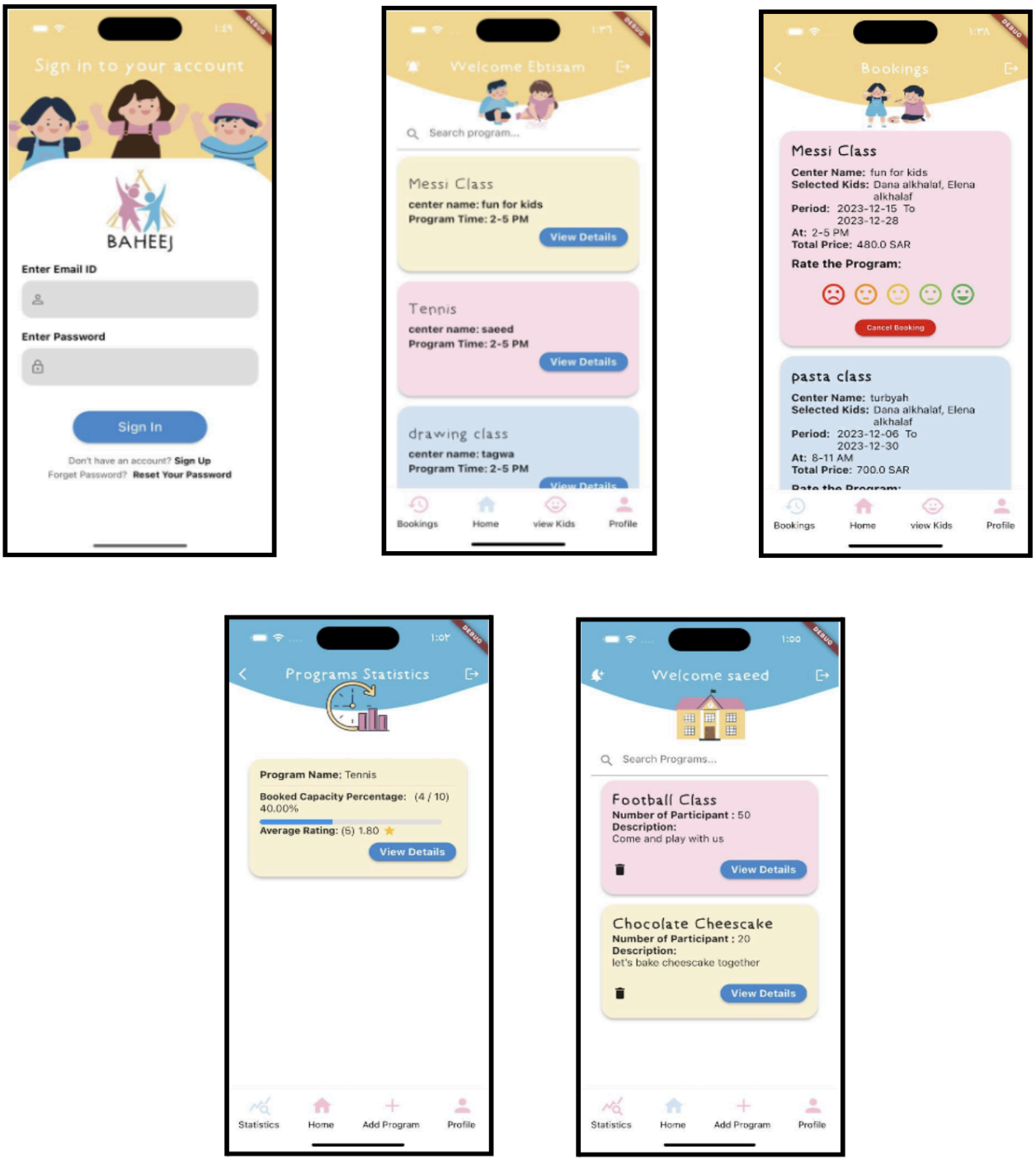

Figure 1: Baheej System Screenshots

Part 1: Program comprehension

- Involves the use of existing knowledge to acquire new knowledge about a program. Use the program comprehension technique to present a new knowledge such as (Code functionality, System architecture, Database structure, Algorithm, Control flow, Data flow ... etc)
- Each student should come up with **a unique view** of the system (with explanation)

	Existing knowledge	***Explanation***	***New Knowledge***
1	User Stories	To gain a deeper understanding of the system's features, we've rewrote the features into functional requirements.	*Functional requirements*
2		To visually represent the interactions between users and a system, aiding in understanding system functionality and requirements.	*Use Case Diagram*
3		To enhance comprehension of a fundamental feature within the system—Add Kids—we have created a use-case description to provide a detailed view.	*Use Case Description*
4	System Architecture	By transforming the box and line diagram into a deployment diagram, you gain a deeper understanding of the system's physical architecture. It unveils the real-world picture: where components physically reside (servers, cloud, devices), the hardware and software used, and communication protocols. This newfound knowledge improves system visibility, communication, and decision-making, allowing	*Deployment diagram*

		you to plan for growth, maintainability, and a robust overall architecture.	
5	Firestore Architecture	To gain a comprehensive understanding of our stored data and its interrelations. This thorough examination allows us to refine data structures, optimize query performance, and enhance security protocols, resulting in a robust and finely-tuned database system.	*Database design*
6	System Testing	To cover all missing test case scenarios for our core feature, 'Add Kids,' we should include testing where users click the 'Add' button, enter valid full names, but input invalid ages, and then click the 'Done' button."	*Test case design*

Functional requirements

No.	***System Features***	***Requirement***
1	Guardian Sign Up	The Guardian shall be able to sign up using first name, last name, email, phone number, gender, password, confirm password, so that they can create an account in the application.
2	Center Sign Up	The Center shall be able to sign up using their name, email, password, confirm password, phone number, address, commercial register, description, so that they can create an account in the application.
3	Center add program	The Center shall be able to add their programs by adding its details (program name, description, max age, min age, price, start date, end date, time slot, and capacity), so that the guardians can gain a comprehensive understanding of the center's program.

4	User Log in	The Users (Guardian, Centers) shall be able to log in using their email and password, so that they can access application features.
5	User Log out	The Users (Guardian, Centers) shall be able to log out, so that they can stop using the application features.
6	User reset password	The Users (Guardian, Centers) shall be able to reset their password when they forget it, so that they can recover their password.
7	Guardian add kids' information	The Guardian shall be able to add a kid information (full name, and age), so that they can save it as their kids.
8	Guardian view kids' information	The Guardian shall be able to view their kid information (full name, and age), so that they can see their saved kids
9	Guardian view programs	The Guardian shall be able to view the list of programs, so that they can select one of the programs.
10	Guardian search for program	The Guardian shall be able to search for a specific program, so that they can easily find the desired program.
11	Guardian view program details	The Guardian shall be able to view the selected center's program details, so that they can decide to pay to book the desired program for their kids.
12	Guardian pay for program	The Guardian shall be able to pay for the selected program by stripe, so that the program booking is confirmed.
13	Guardian view booked programs	The Guardian shall be able to view their booked programs, so that they have a perception of the programs that their kids have already enrolled in.
14	User view profile	The Users (Guardian, Centers) shall be able to view their profile, so that they can review and manage their personal information.

15	Users edit profile	The Users (Guardian, Centers) shall be able to edit their profile, so that they can edit their personal information.
16	Center sends offer notification	The Center shall be able to send offers notifications to guardians, so that the guardian can be informed by the centers.
17	Center hide program	The Center shall be able to hide their program when the start date begins, so that no one books it after the date of start.
18	Center view their programs	The Center shall be able to view the list of their own programs, so that they can decide to edit or delete a program.
19	Center delete their program	The Center shall be able to delete their programs so no one can enroll in the program.
20	Center edit program details	The Center shall be able to edit their program's details (program name, description, max age, min age, price, start date, end date, time slot, and capacity), so that they can fix any mistake or modify.
21	User delete account	The Users (Guardian, Centers) shall be able to delete their account, so that when they no longer need the application, they delete their account and data.
22	Center search for their programs	The Center shall be able to search for a specific program they own, so that they can easily find it.
23	Guardian notification history	The Guardian shall be able to view the notification history, so that they can view all notifications they received.
24	Guardian cancel subscription	The Guardian shall be able to cancel their subscription before the program start date, so that when they no longer want to book the program for their kids they can cancel it.

25	Guardian rate program	The Guardians shall be able to rate their booked programs when the start date begins, so that the center can gather insights and improve programs accordingly.
26	Center view previous programs	The Center shall be able to view the list of their previous programs when the end date finishes, so that they be aware of what programs they added that finished.
27	Center view the booking capacity percentage	The Center shall be able to view the list of their previous programs capacity, so that they be aware of each program booking capacity percentage.
28	Center view average rating	The Center shall be able to view the list of their previous programs average rating, so that they be aware of each program average ratings.

Use Case Diagram

Figure 2: Baheej Use Case Diagram

Use Case Description

<table>
<tr><th colspan="2">Use Case Description</th></tr>
<tr><td colspan="2">System: BAHEEJ</td></tr>
<tr><td colspan="2">Use Case Name: Add Kid</td></tr>
<tr><td>Primary actor: Guardian</td><td>Other actors: None</td></tr>
<tr><td colspan="2">Description: This use case shows how a guardian can add his/her child</td></tr>
<tr><td colspan="2">Relationships
Include: View Kids
Extends:</td></tr>
<tr><td colspan="2">Pre-conditions:
- The guardian is logged in.</td></tr>
<tr><td colspan="2">Basic flow:</td></tr>
<tr><td>Actor(User)</td><td>System</td></tr>
<tr><td>1- Guardian clicks on Add Kid

3- Guardian enters Kid full name
4- Guardian enters kid age
5- Guardian clicks on (Done)</td><td>2-System display “Add Kid” form asking guardian to enter kid’s full name and age

6- System verifies user input is complete and correct.
7- System adds a new record in the database for the kid.
8- system shows confirmation message (kid added successfully)</td></tr>
<tr><td colspan="2">Alternative flow:</td></tr>
<tr><td colspan="2">Invalid inputs:
6.1- The system displays an error message indicating incorrect input.
Guardian clicks on cancel:
5.1-system redirect user to view kids screen.
Kid already added:
7.1- The system displays an error message indicating this kid already added.</td></tr>
<tr><td colspan="2">Post- conditions:
Successful condition:The System will add the kid successfully
Failure condition: The System will not add the kid</td></tr>
</table>

System Architecture diagram

Here's the new knowledge that demonstrates more of the physical architecture of the system (Deployment Diagram).

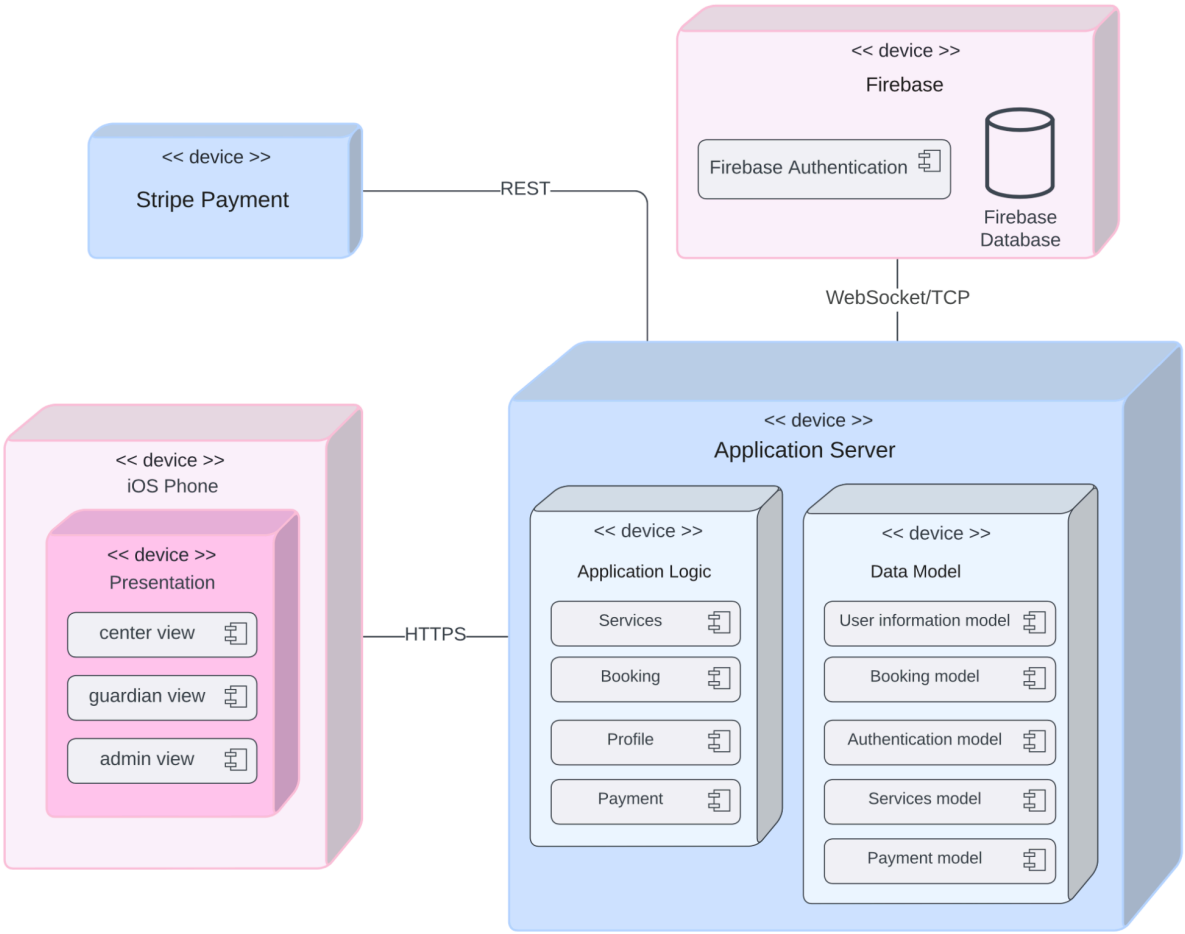

Figure 3: Baheej Deployment Diagram

Database design

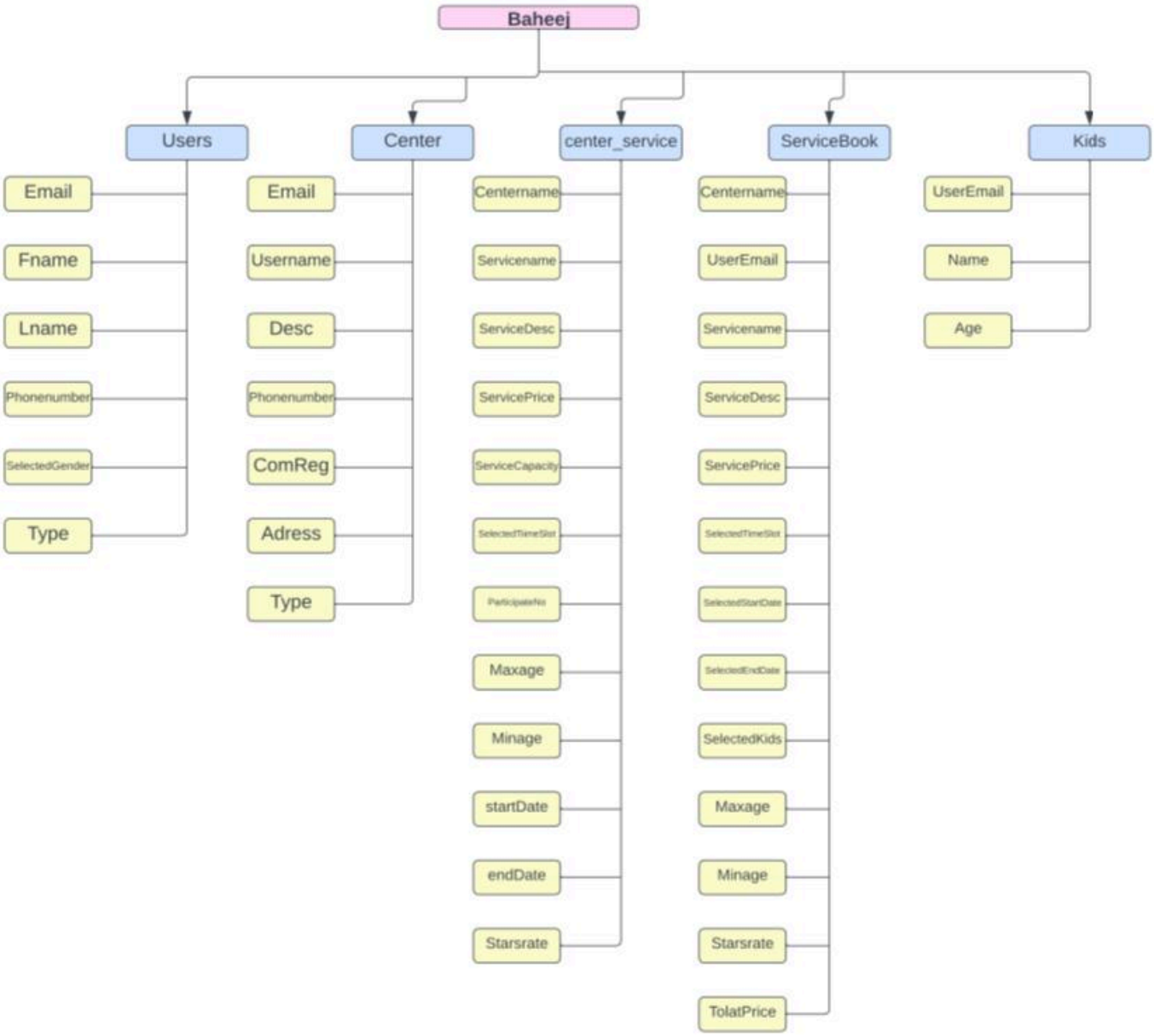

Figure 4: Baheej Database Design

Test Case design

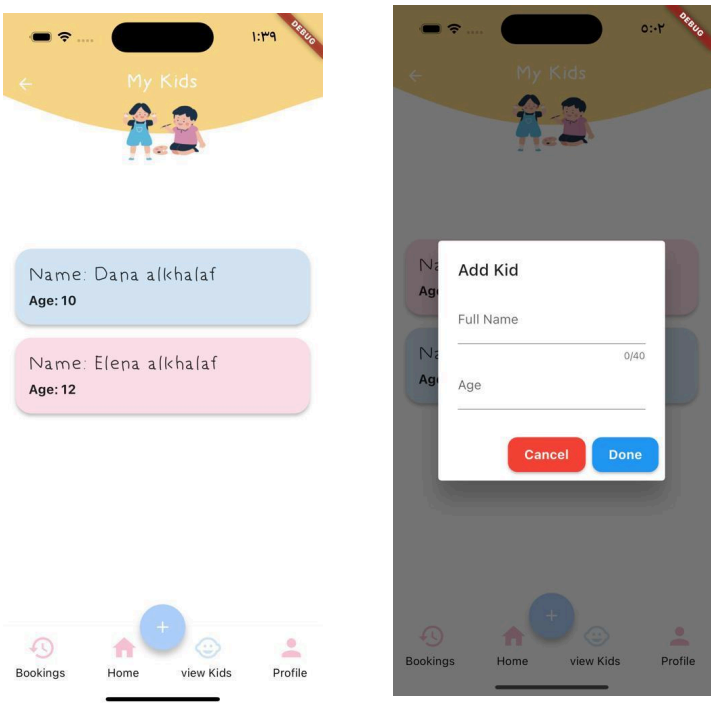

Figure 5.1: Screenshot from Baheej application , show constraint cover in test cast (valid name and age)

Sprint #	Test cases ID	Choosen Feature	Test Designed by	Test Designed Date	Test Executed By	Test Execution Date	Test Title
Sprint 2	TC_32	Add child	Jory	4/10/2023	Najd	5/10/2023	Add child with valid (full name, age)
	TC_33		Danah	4/10/2023	Najd	5/10/2023	Add child with valid full name and invalid age.
	TC_34		Renad	4/10/2023	Danah	5/10/2023	Add child with valid full name and invalid age.
	TC_35		Ghadeer	4/10/2023	Renad	5/10/2023	Add child with valid full name and empty age.
	TC_36		Nada	4/10/2023	Ghadeer	6/10/2023	Add child with invalid full name and valid age (entering integer in full name)
	TC_37		Najd	4/10/2023	Danah	6/10/2023	Add child with empty full name and valid age
	TC_38		Jory	4/10/2023	Nada	6/10/2023	Add child with empty full name and empty age
	TC_39		Renad	4/10/2023	Nada	7/10/2023	Add child with an exist full name and valid age
	TC_40		Ghadeer	4/10/2023	Jory	7/10/2023	Add child with valid full name and invalid age as entering letters
	TC_41		Danah	4/10/2023	Ghadeer	7/10/2023	Add child with invalid full name and valid age (entering special characters in full name)
	TC_42		Ghadeer	4/10/2023	Jory	7/10/2023	Add child with valid age and less than 10 letters in full name
							Add child with valid age and greater than 40 letters

Sprint #	Test cases ID	Description	Pre- conditions	Dependencies
Sprint 2	TC_32	This test case is to verify that the gaurdian can add child using valid data	1-user must login successfully as a guardian. 2-view kids	None
	TC_33	This test case is to verify that the gaurdian cannot add child using invalid age as entering age less than 4	1-user must login successfully as a guardian. 2-view kids	None
	TC_34	This test case is to verify that the gaurdian cannot add child using invalid age as entering age greater than 17	1-user must login successfully as a guardian. 2-view kids	None
	TC_35	This test case is to verify that the gaurdian cannot add child using empty age field	1-user must login successfully as a guardian. 2-view kids	None
	TC_36	This test case is to verify that the gaurdian cannot add child using invalid full name as integer	1-user must login successfully as a guardian. 2-view kids	None
	TC_37	This test case is to verify that the gaurdian cannot add child using empty full name	1-user must login successfully as a guardian. 2-view kids	None
	TC_38	This test case is to verify that the gaurdian cannot add child with empty full name and empty age	1-user must login successfully as a guardian. 2-view kids	None
	TC_39	This test case is to verify that the gaurdian cannot add child with an exist full name	1-user must login successfully as a guardian. 2-view kids	This test case depends on the presence of a successfully added child whose name is the same as the child that the guardian is trying to enter
	TC_40	This test case is to verify that the gaurdian cannot add a child using an invalid age as letter	1-user must login successfully as a guardian. 2-view kids	None
	TC_41	This test case is to verify that the gaurdian cannot add child using invalid full name as special characters	1-user must login successfully as a guardian. 2-view kids	None
	TC_42	This test case is to verify that the gaurdian cannot add a child using less than 10 letters in full name	1-user must login successfully as a guardian. 2-view kids	None

Sprint #	Test cases ID	Test Steps	Test Data	Expected Result
Sprint 2	TC_32	1-logged in 2-press "view kids" button in the bottom bar. 3-press plus button 4-Fill blanket with valid data	("Najd Alnafisah", "7")	Kid "Najd Alnafisah" added successfully!
	TC_33	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("Jory Alhassan", "2")	Invalid age input, Age must be between 4 and 17
	TC_34	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("Danh Alkhalaf", "22")	Invalid age input, Age must be between 4 and 17
	TC_35	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("May Alnumay", "")	Name and age can't be empty
	TC_36	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("Najd3333333333", "6")	Name must contain only letters.
	TC_37	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("" , "7")	Name and age can't be empty
	TC_38	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("" , "")	Name and age can't be empty
	TC_39	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("Najd Alnafisah", "16")	Kid with the same name already exists.
	TC_40	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("Luna Almanee" , "hh")	Our app will not let the guardian to enter a letter only a digit number appear in age field (no error massage will appear
	TC_41	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("!!!!!!!!!!ffffffffff" , "6")	Name must contain only letters.
	TC_42	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("dana" , "5")	Name must be between 10 and 40 characters.

Sprint #	Test cases ID	Actual Result	Status (Pass/Fail)	Notes
Sprint 2	TC_32	Kid "Najd Alnafisah" added successfully!	Pass	
	TC_33	Invalid age input, Age must be between 4 and 17	Pass	
	TC_34	Invalid age input, Age must be between 4 and 17	Pass	
	TC_35	Name and age can't be empty	Pass	
	TC_36	Name must contain only letters.	Pass	
	TC_37	Name and age can't be empty	Pass	
	TC_38	Name and age can't be empty	Pass	
	TC_39	Kid with the same name already exists.	Pass	
	TC_40	Our app will not let the guardian to enter a letter only a digit number appear in age field (no	Pass	
	TC_41	Name must contain only letters.	Pass	
	TC_42	Name must be between 10 and 40 characters.	Pass	

*Figure 5.2:*Baheej Test CaseDesign of add kids before

Sprint #	Test cases ID	Choosen Feature	Test Designed by	Test Designed Date	Test Executed By	Test Execution Date	Test Title
Sprint 2	TC_32	Add child	Jory	4/10/2023	Najd	5/10/2023	Add child with valid (full name, age)
	TC_33		Danah	4/10/2023	Najd	5/10/2023	Add child with valid full name and invalid age.
	TC_34		Renad	4/10/2023	Danah	5/10/2023	Add child with valid full name and invalid age.
	TC_35		Ghadeer	4/10/2023	Renad	5/10/2023	Add child with valid full name and empty age.
	TC_36		Nada	4/10/2023	Ghadeer	6/10/2023	Add child with invalid full name and valid age (entering integer in full name)
	TC_37		Najd	4/10/2023	Danah	6/10/2023	Add child with empty full name and valid age
	TC_38		Jory	4/10/2023	Nada	6/10/2023	Add child with empty full name and empty age
	TC_39		Renad	4/10/2023	Nada	7/10/2023	Add child with an exist full name and valid age
	TC_40		Ghadeer	4/10/2023	Jory	7/10/2023	Add child with valid full name and invalid age as entering letters
	TC_41		Danah	4/10/2023	Ghadeer	7/10/2023	Add child with invalid full name and valid age (entering special characters in full name)
	TC_42		Ghadeer	4/10/2023	Jory	7/10/2023	Add child with valid age and less than 10 letters in full name
	TC_43		Renad	4/10/2023	Danah	7/10/2023	Add child with valid age and greater than 40 letters in full name
	TC_44		Nada	4/10/2023	Danah	7/10/2023	Add child with valid full name and invalid age as entering special characters

Sprint #	Test cases ID	Description	Pre- conditions	Dependencies
Sprint 2	TC_32	This test case is to verify that the gaurdian can add child using valid data	1-user must login successfully as a guardian. 2-view kids	None
	TC_33	This test case is to verify that the gaurdian cannot add child using invalid age as entering age less than 4	1-user must login successfully as a guardian. 2-view kids	None
	TC_34	This test case is to verify that the gaurdian cannot add child using invalid age as entering age greater than 17	1-user must login successfully as a guardian. 2-view kids	None
	TC_35	This test case is to verify that the gaurdian cannot add child using empty age field	1-user must login successfully as a guardian. 2-view kids	None
	TC_36	This test case is to verify that the gaurdian cannot add child using invalid full name as integer	1-user must login successfully as a guardian. 2-view kids	None
	TC_37	This test case is to verify that the gaurdian cannot add child using empty full name	1-user must login successfully as a guardian. 2-view kids	None
	TC_38	This test case is to verify that the gaurdian cannot add child with empty full name and empty age	1-user must login successfully as a guardian. 2-view kids	None
	TC_39	This test case is to verify that the gaurdian cannot add child with an exist full name	1-user must login successfully as a guardian. 2-view kids	This test case depends on the presence of a successfully added child whose name is the same as the child that the guardian is trying to enter
	TC_40	This test case is to verify that the guardian cannot add a child using an invalid age as letter	1-user must login successfully as a guardian. 2-view kids	None
	TC_41	This test case is to verify that the gaurdian cannot add child using invalid full name as special characters	1-user must login successfully as a guardian. 2-view kids	None
	TC_42	This test case is to verify that the guardian cannot add a child using less than 10 letters in full name	1-user must login successfully as a guardian. 2-view kids	None
	TC_43	This test case is to verify that the guardian cannot add a child using greater than 40 letters in full name	1-user must login successfully as a guardian. 2-view kids	None
	TC_44	This test case is to verify that the guardian cannot add a child using an	1-user must login successfully as a guardian.	None

Sprint #	Test cases ID	Test Steps	Test Data	Expected Result
Sprint 2	TC_32	1-logged in 2-press "view kids" button in the bottom bar. 3-press plus button 4-Fill blanket with valid data	("Najd Alnafisah", "7")	Kid "Najd Alnafisah" added successfully!
	TC_33	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("Jory Alhassan", "2")	Invalid age input, Age must be between 4 and 17
	TC_34	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("Danh Alkhalaf", "22")	Invalid age input, Age must be between 4 and 17
	TC_35	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("May Alnumay", "")	Name and age can't be empty
	TC_36	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("Najd3333333333", "6")	Name must contain only letters.
	TC_37	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("" , "7")	Name and age can't be empty
	TC_38	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("" , "")	Name and age can't be empty
	TC_39	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("Najd Alnafisah", "16")	Kid with the same name already exists.
	TC_40	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("Luna Almanee" , "hh")	Our app will not let the guardian to enter a letter only a digit number appear in age field (no error message will appear
	TC_41	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("!!!!!!!!!ffffffff" , "6")	Name must contain only letters.
	TC_42	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("dana" , "5")	Name must be between 10 and 40 characters.
	TC_43	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("Danaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaa", "5")	Our application only accepts the first 40 letters and rejects the rest.
	TC_44	1-guardian open the baheej app, 2-logged in as guardian 3-press "view kids" button in the bottom bar. 4-press plus button 5-Fill blanket with data 6- click Done	("Ahmed Alhammad" , "!!")	Our app will not let the guardian to enter a special characters only a digit number appear in age field (no error message will

Sprint #	Test cases ID	Actual Result	Status (Pass/Fail)	Notes
Sprint 2	TC_32	Kid "Najd Alnafisah" added successfully!	Pass	
	TC_33	Invalid age input, Age must be between 4 and 17	Pass	
	TC_34	Invalid age input, Age must be between 4 and 17	Pass	
	TC_35	Name and age can't be empty	Pass	
	TC_36	Name must contain only letters.	Pass	
	TC_37	Name and age can't be empty	Pass	
	TC_38	Name and age can't be empty	Pass	
	TC_39	Kid with the same name already exists.	Pass	
	TC_40	Our app will not let the guardian to enter a letter only a digit number appear in age field (no	Pass	
	TC_41	Name must contain only letters.	Pass	
	TC_42	Name must be between 10 and 40 characters.	Pass	
	TC_43	only accepts the first 40 letters and	Pass	
	TC_44	Our app will not let the guardian to enter a special characters only a digit number appear in age field	Pass	

Figure 5.3: Baheej Test CaseDesign of add kids after

Part 2: Maintenance Request (MR) management

Maintenance Request management:

Efficient maintenance request management plays a pivotal role in addressing and resolving software-related issues in a timely manner. Here's a breakdown detailing the main components of our software maintenance request management system.

Define Maintenance Types: Differentiate between corrective, preventive, adaptive, and perfective maintenance. Each type requires a unique approach and planning.

Establish Priorities: Categorize maintenance tasks based on urgency and impact to minimize disruptions and ensure efficient resource allocation.

Create a Maintenance Schedule: Developing a schedule considering system downtime, user availability, and business operations and Communicate it to relevant stakeholders.

Documentation: Document the entire maintenance plan, including objectives, tasks, responsibilities, and timelines. This documentation serves as a reference for future activities.

Configuration Management:

Configuration Management involves managing and controlling changes to a system's components and ensuring consistency across the entire system

Configuration Identification: Clearly identify and label all configuration items within the system, including software, documentation, and other components.

Configuration Control: Implement controls to manage changes to configuration items and Establish a formal change control process to evaluate and approve changes.

Change Management Process: Define a structured process for requesting, evaluating, approving, and implementing changes. Ensure that changes are well-documented and tested.

Version and Release Management: Implement version control to manage different iterations of the software.

By incorporating these components into our software maintenance request management system, we can enhance the overall efficiency, transparency, and responsiveness of our maintenance operations..

MR Form

Name:		MR ID	
Date:	__/__/__	Requester	
Type:	☐ Corrective ☐ Adaptive ☐ Perfective ☐ Preventive		
Priority:	☐ High ☐ Medium ☐ Low		
Severity:	☐ High ☐ Medium ☐ Low		
Item Require Change	☐ Software: ☐ Document:		
Cost Estimate	Activity	Estimates *(person-day)*	
	Understand the problem and identify the functions that must be modified or added		
	Design the change		
	Perform impact analysis		
	Implement changes in source code		
	Change SRS, SDD, STP, configuration status		
	Compile and integrate into baseline		
	Unit Test functionality of changes		
	Perform regression testing		
	Release new baseline and report results		
	Total		
	Average		
Description			

Impact analysis	

MR #1

Name:	Special Needs Support Indicator	MR ID	MR01
Date:	24/02/2024	Requester Name:	Jory AlHassan
Type:	☐ Corrective ☐ Adaptive ✔ Perfective ☐ Preventive		
Priority:	☐ High ✔ Medium ☐ Low		
Severity:	☐ High ☐ Medium ✔ Low		
Item Require Change	✔ Software: service page, service for screen page,HomeScreenGaurdian ✔ Document:System Features, System Screenshot		
Cost Estimate	Activity	Estimates *(person-day)*	
	Understand the problem and identify the functions that must be modified or added	1	
	Design the change	2-3	
	Perform impact analysis	1-2	
	Implement changes in source code	2-3	
	Change SRS, SDD, STP, configuration status	1	
	Compile and integrate into baseline	1-2	
	Unit Test functionality of changes	1	
	Perform regression testing	1	
	Release new baseline and report results	1-3	
	Total	10-16	
	Average	1.5	

Description	Adding the ability for Centers to demonstrate that their services are designed to support children with special needs.
Impact analysis	To add a feature that allows centers to indicate support for children with special needs, several updates will be required. These updates will involve changes to the user interface, database and search functionalities. Additionally, updating documentation will need to be developed for users. Finally, thorough testing will be necessary to ensure a seamless integration of the new feature.

MR #2

Name:	Phone number ownership verification when signing up.	MR ID	MR02
Date:	28/2/2024	Requester Name:	Ghadeer AlMansour
Type:	☐ Corrective ☐ Adaptive ☐ Perfective ✔ Preventive		
Priority:	☐ High ✔ Medium ☐ Low		
Severity:	☐ High ✔ Medium ☐ Low		
Item Require Change	✔ Software:Sign up page ✔ Document:System Features, System Screenshot		
Cost Estimate	Activity	Estimates *(person-day)*	
	Understand the problem and identify the functions that must be modified or added	2	
	Design the change	1-2	
	Perform impact analysis	2	
	Implement changes in source code	1-3	
	Change SRS, SDD, STP, configuration status	1-2	
	Compile and integrate into baseline	2	
	Unit Test functionality of changes	1	
	Perform regression testing	2	

	Release new baseline and report results	1-3
	Total	13-19
	Average	1.78
Description	Implementing phone number ownership verification for users signing up to our system. A phone number is required from the users when signing up to our system . Although some preventive measures have been taken to ensure that the phone number provided is valid for format correctness, no validation is done to ensure that the users actually own the provided number that they claim they own. To address this, we plan to send a verification message containing an OTP (One-Time-Password) to each user, this will confirm the ownership of the phone number provided during sign-up. This process ensures that each user provides a genuine phone number that they own.	
Impact analysis	We need to consider specific requirements for UI updates, database changes, functionality changes, documentation updates, and testing procedures.	

MR #3

<table>
<tr><td>Name:</td><td colspan="2">Add feedback to both home guardian and center</td><td>MR ID</td><td>MR03</td></tr>
<tr><td>Date:</td><td colspan="2">27/2/2024</td><td>Requester Name:</td><td>Renad ALNumay</td></tr>
<tr><td>Type:</td><td colspan="4">☐ Corrective ☐ Adaptive ✔ Perfective ☐ Preventive</td></tr>
<tr><td>Priority:</td><td colspan="4">✔ High ☐ Medium ☐ Low</td></tr>
<tr><td>Severity:</td><td colspan="4">☐ High ✔ Medium ☐ Low</td></tr>
<tr><td>Item Require Change</td><td colspan="4">✔ Software:HomeScreenCenter , HomeScreenGaurdian
✔ Document:System Features, System Screenshot</td></tr>
<tr><td rowspan="12">Cost Estimate</td><td colspan="2">Activity</td><td colspan="2">Estimates (person-day)</td></tr>
<tr><td colspan="2">Understand the problem and identify the functions that must be modified or added</td><td colspan="2">1-1</td></tr>
<tr><td colspan="2">Design the change</td><td colspan="2">1-1</td></tr>
<tr><td colspan="2">Perform impact analysis</td><td colspan="2">1-2</td></tr>
<tr><td colspan="2">Implement changes in source code</td><td colspan="2">1-2</td></tr>
<tr><td colspan="2">Change SRS, SDD, STP, configuration status</td><td colspan="2">1-1</td></tr>
<tr><td colspan="2">Compile and integrate into baseline</td><td colspan="2">1-2</td></tr>
<tr><td colspan="2">Unit Test functionality of changes</td><td colspan="2">1-1</td></tr>
<tr><td colspan="2">Perform regression testing</td><td colspan="2">1-2</td></tr>
<tr><td colspan="2">Release new baseline and report results</td><td colspan="2">1-3</td></tr>
<tr><td colspan="2">Total</td><td colspan="2">12</td></tr>
<tr><td colspan="2">Average</td><td colspan="2">1.33</td></tr>
<tr><td>Description</td><td colspan="4">To enhance the user experience with the system, we will add a feedback feature for both users. They can write about any issues that occur or suggest improvements for our application.</td></tr>
</table>

Impact analysis	To add a feature that allows centers and guardians to write feedback, several updates will be required. These updates will involve changes to the user interface and database. Additionally, updated documentation will need to be developed for users. Finally, thorough testing will be necessary to ensure a seamless integration of the new feature.

MR #4

Name:	Add Arabic Language to the system.	MR ID	MR04
Date:	5/3/2024	Requester Name:	Najd AlNafisah
Type:	☐ Corrective ☐ Adaptive ✔Perfective ☐ Preventive		
Priority:	☐ High ✔Medium ☐ Low		
Severity:	☐ High ✔ Medium ☐ Low		
Item Require Change	✔ Software: Change the UI of all pages ✔Document:System Features, System Screenshot		
Cost Estimate	Activity	Estimates *(person-day)*	
	Understand the problem and identify the functions that must be modified or added	1-4	
	Design the change	1-6	
	Perform impact analysis	1-4	
	Implement changes in source code	4-7	
	Change SRS, SDD, STP, configuration status	2-5	
	Compile and integrate into baseline	2-5	
	Unit Test functionality of changes	2-5	
	Perform regression testing	2-4	
	Release new baseline and report results	1	
	Total		

		16-41
	Average	3.16
Description	Supporting both Arabic and English languages is crucial for the system to cater to a broader user base, as Arabic serves as the primary language for many users, ensuring their satisfaction.	
Impact analysis	When adding this future, we need to consider UI updates, functionality changes, documentation updates, and testing procedures.	

MR #5

Name:	Add filtering options to guardian search bar.	MR ID	MR05
Date:	6/3/2024	Requester Name:	Danh Alkhalaf
Type:	☐ Corrective ☐ Adaptive ✔Perfective ☐ Preventive		
Priority:	☐ High ✔ Medium ☐ Low		
Severity:	☐ High ✔ Medium ☐ Low		
Item Require Change	✔ Software: service page, serviceForm page, HomeScreenGaurdian ✔Document: System Features, System Screenshot.		
Cost Estimate	Activity	Estimates *(person-day)*	
	Understand the problem and identify the functions that must be modified or added	1-1	
	Design the change	1-2	
	Perform impact analysis	1-2	
	Implement changes in source code	1-2	
	Change SRS, SDD, STP, configuration status	1-2	
	Compile and integrate into baseline	1-2	
	Unit Test functionality of changes	1-2	
	Perform regression testing	1-1	

	Release new baseline and report results	1-2
	Total	9-16
	Average	1.388
Description	Introducing a search bar filter feature in the Baheej app to enhance user experience. The feature will allow guardians to filter the list of activities displayed on their home based on specific criteria such as the type of service offered by centers.	
Impact analysis	Implementing the search bar filter feature involves updating various aspects of the app. This includes modifying the user interface to integrate the filter options with the existing search bar smoothly, adjusting database to enable activity filtering, updating documentation, and conducting rigorous testing to ensure seamless integration.	

Part 3: Slicing

- You should show here the slicing type you have used to maintain or evolve your code and selection reason.
- Provide screenshots of the code.
- **At least 5.p**

#	*Slicing criteria*	*Original Code*	*Slicing result*	*Slicing type*	*Selection reason*
1	(843,{updatedServicePrice})	Figure 6	Figure 7 Lines in slice: 24,38,804,843	Backward	Identify and understand all the factors that contribute to its final value before it is utilized to update a Firestore document.
2	(130,{regex})	Figure 8	Figure 9 Lines in slice: 130,131	Forward	By slicing this part of the code, we can focus on understanding how email validation is performed, which is crucial for ensuring data integrity and security in the password reset functionality.
3	(958,{bookedServicesCount})	Figure 10	Figure 11 Lines in slice: 958,961	Forward	This Slicing allows us to focus on how the bookedServicesCount variable is utilized to calculate the booking ratio. By isolating this part of the code, we can effectively debug any potential issues related

					to the calculation logic and ensure its accuracy.
4	(280,{firstName })	Figure 12	Figure 13 280,283,285	Forward	By slicing this part of the code, we understand the aim to comprehend how the variable firstName is created, utilized, and to identify the statement how it influenced by the variable (firstName) and remove dead code.
5	(979,{now})	Figure 14	Figure 15 Line in slice: 979	Forward	To Identify all statements that could be influenced by the variable (now) and remove dead code

1-(843,{updatedServicePrice})

Original code:

Figure 6: Original code before applying (843,{updatedServicePrice}) criteria

Slicing Result:

```
TextEditingController _servicePriceController = TextEditingController();

_servicePriceController.text = widget.service.servicePrice.toString();

final double updatedServicePrice = double.tryParse(_servicePriceController.text) ?? 0.0;

await FirebaseFirestore.instance.collection('center-service').doc(widget.service.id).update({
    'servicePrice': updatedServicePrice,});
```

Figure 7: Slicing result after applying (843,{updatedServicePrice}) criteria

2-(130,{regex})

Original code:

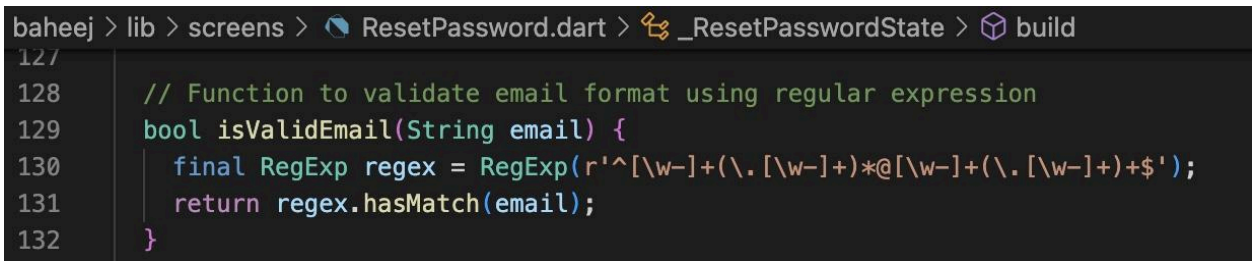

Figure 8: Original code before applying (130,{regex}) criteria

Slicing Result:

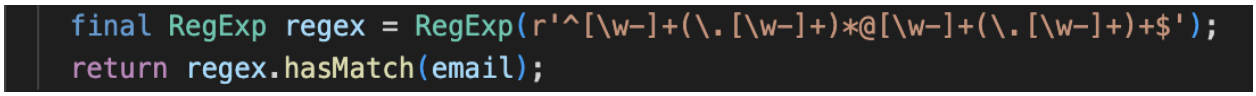

Figure 9: Slicing result after applying (130,{regex}) criteria

3-(958,{bookedServicesCount})

Original code:

baheej > lib > screens > compSerList.dart > _compSerListScreenState > getBookedServicesCount

```
955   Future<double> calculateBookingRatio(Service service) async {
956     if (service.capacityValue > 0) {
957       // Fetch the booked services for the logged-in center
958       int bookedServicesCount =  await getBookedServicesCount(service.centerName);
959
960       // Calculate the booking ratio
961       return bookedServicesCount > 0  ? bookedServicesCount / service.capacityValue  : 0.0;
962     } else {
963       return 0.0; // Handle the case where capacityValue is 0 to avoid division by zero
964     }
965   }
```

Figure 10: Original code before applying (958,{bookedServicesCount}) criteria

Slicing Result:

```
int bookedServicesCount =  await getBookedServicesCount(service.centerName);

return bookedServicesCount > 0  ? bookedServicesCount / service.capacityValue : 0.0;
```

Figure 11: Slicing result after applying (958,{bookedServicesCount}) criteria

4-(280{firstName})

```dart
void fetchName() async {
  final user = FirebaseAuth.instance.currentUser;
  if (user != null) {
    final userId = user.uid;
    final userDoc = await FirebaseFirestore.instance
        .collection('users')
        .doc(userId)
        .get();
    if (userDoc.exists) {
      final userData = userDoc.data() as Map<String, dynamic>;
      final firstName = userData['fname'] ?? '';
      final userRole = userData[
          'userType'];
      print('Fetched first name: $firstName');
      setState(() {
        FirstName = firstName;
        type = userRole;
      });
    }
  }
}
```

Figure 12: Original code before applying (280,{firstName}) criteria

```dart
        final firstName = userData['fname'] ?? '';


        print('Fetched first name: $firstName');

          FirstName = firstName;
```

Figure 13: Original code after applying (280,{firstName}) criteria

5- (979,{now})

Original code:

baheej > lib > screens > HomeScreenCenter.dart > _HomeScreenCenterState > showDeleteConfirmationDialog

```
975   }
976
977   void _handleSearch(String query) {
978     query = query.trim();
979     final now = DateTime.now();
980     setState(() {
981       filteredServices = services
982           .where((service) =>
983               service.serviceName.toLowerCase().contains(query.toLowerCase()) ||
984               service.description.toLowerCase().contains(query.toLowerCase()))
985           .toList();
986     });
987   }
```

Figure 14: _handelSearch method original code

Slicing Result:

```
final now = DateTime.now();
```

Figure 15: **Slicing Result After Applying (979,{now}) criteria**

*the only statement influenced by the criteria (979, {now}) is the line initializes variable now

Part 4: Refactoring

	Code before refactoring	***Refactoring type***	***Code after refactoring***	***Affected Class(es)***	***Explanation***
1	The variable ‘formattedPrice’ was declared in makePayment but never used **Figure 16**	Remove unused variable	The variable ‘formattedPrice ‘ is remove from makePayment method **Figure 17**	ServiceDetailsPage	There is no need for the 'formattedPrice' in the 'makePayment' method, as the price format was declared in the 'createPaymentIntent' method
2	2 Unused imports in HomeScreenCenter page. **Figure 18**	Eliminate unused Imports	Eliminate unused Imports. **Figure 19**	HomeScreenCenter page	Eliminating unused imports enhances code quality and readability by reducing the number of lines, as they are unnecessary.
3	4 Unused imports in Csignup page **Figure 20**	Eliminate unused Imports	Eliminate unused Imports. **Figure 21**	Csignup page	Eliminating unused imports enhances code quality and readability by reducing the number of lines, as they are unnecessary.
4	The variable ‘userId’ was declared in getBookedServices Count method but never used	Remove unused variable	The variable ‘userId’ was removed from the method getBookedServicesCount	compSerList	The variable ‘userId’ is unused representing dead code,where there is no need for the variable, so

	Figure 22		**Figure 23**		removing it will enhance the performance.
5	The variable ‘now’ was declared in _handelSearch method but never used **Figure 24**	Remove unused variable	The variable ‘now‘ in _handelSearch is removed **Figure 25**	HomeScreenCenter page	The declared variable 'now' remains unused representing dead code so removing it streamlines the program's behavior without any impact and also contributes to performance enhancement and minimizes maintenance overhead.

1-ServiceDetailsPage Refactoring

```dart
// create payment
  void makePayment(BuildContext context) async {
    try {
      double totalPrice = calculateTotalPrice(widget.service);
      paymentIntent = await createPaymentIntent(totalPrice);

      // ignore: prefer_const_constructors
      var gpay = PaymentSheetGooglePay(
        merchantCountryCode: "US",
        currencyCode: 'SAR',
        testEnv: true,
      );

      String formattedPrice =
          NumberFormat.currency(locale: 'en_US', symbol: '').format(totalPrice);

      Stripe.instance.initPaymentSheet(
        paymentSheetParameters: SetupPaymentSheetParameters(
          paymentIntentClientSecret: paymentIntent!["client_secret"],
          style: ThemeMode.dark,
          merchantDisplayName: "baheej",
          googlePay: gpay,
        ),
      );

      await displayPaymentSheet(context);
      // Always display payment sheet
    } catch (e) {
      print("Error: $e");
    }
  }
```

Figure 16: ServiceDetailsPage before refactoring

```dart
void makePayment(BuildContext context) async {
  try {
    double totalPrice = calculateTotalPrice(widget.service);
    paymentIntent = await createPaymentIntent(totalPrice);

    // ignore: prefer_const_constructors
    var gpay = PaymentSheetGooglePay(
      merchantCountryCode: "US",
      currencyCode: 'SAR',
      testEnv: true,
    );
    Stripe.instance.initPaymentSheet(
      paymentSheetParameters: SetupPaymentSheetParameters(
        paymentIntentClientSecret: paymentIntent!["client_secret"],
        style: ThemeMode.dark,
        merchantDisplayName: "baheej",
        googlePay: gpay,
      ),
    );

    await displayPaymentSheet(context);
    // Always display payment sheet
  } catch (e) {
    print("Error: $e");
  }
}
```

Figure 17: ServiceDetailsPage after refactoring

2- HomeScreenCenter page Refactoring

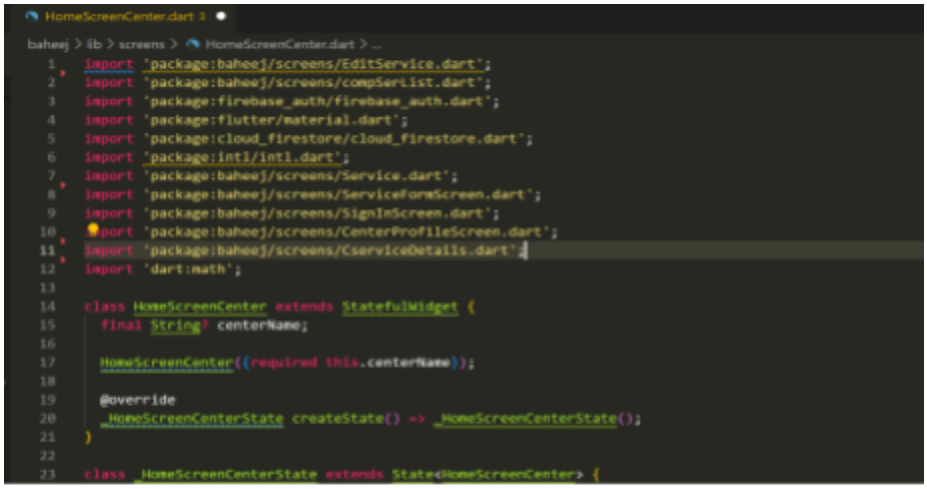

Figure 18: **HomeScreenCenter page before refactoring**

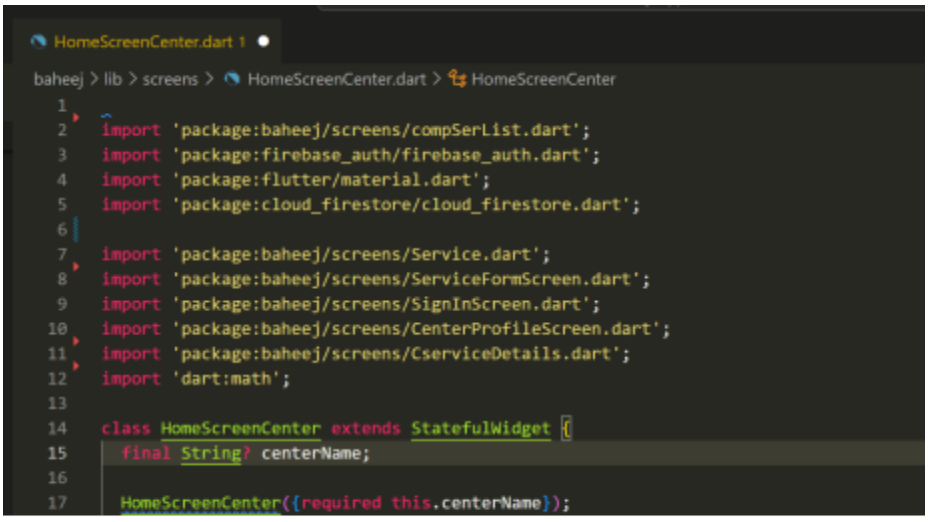

Figure 19: **HomeScreenCenter page after refactoring**

3- Csignup page Refactoring

baheej > lib > screens > Csignup.dart > ...

```dart
import 'package:baheej/screens/SignInScreen.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:baheej/utlis/utilas.dart';
import 'package:baheej/reusable_widget/reusable_widget.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'dart:convert';
import 'package:flutter/services.dart';

class CsignUpScreen extends StatefulWidget {
  const CsignUpScreen({Key? key}) : super(key: key);

  @override
  _CsignUpScreenState createState() => _CsignUpScreenState();
}
```

Figure 20: Csignup page before refactoring

baheej > lib > screens > Csignup.dart > ...

```dart
import 'package:baheej/screens/SignInScreen.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/services.dart';

class CsignUpScreen extends StatefulWidget {
  const CsignUpScreen({Key? key}) : super(key: key);

  @override
  _CsignUpScreenState createState() => _CsignUpScreenState();
}

```

Figure 21: Csignup page after refactoring

4- compSerList page Refactoring

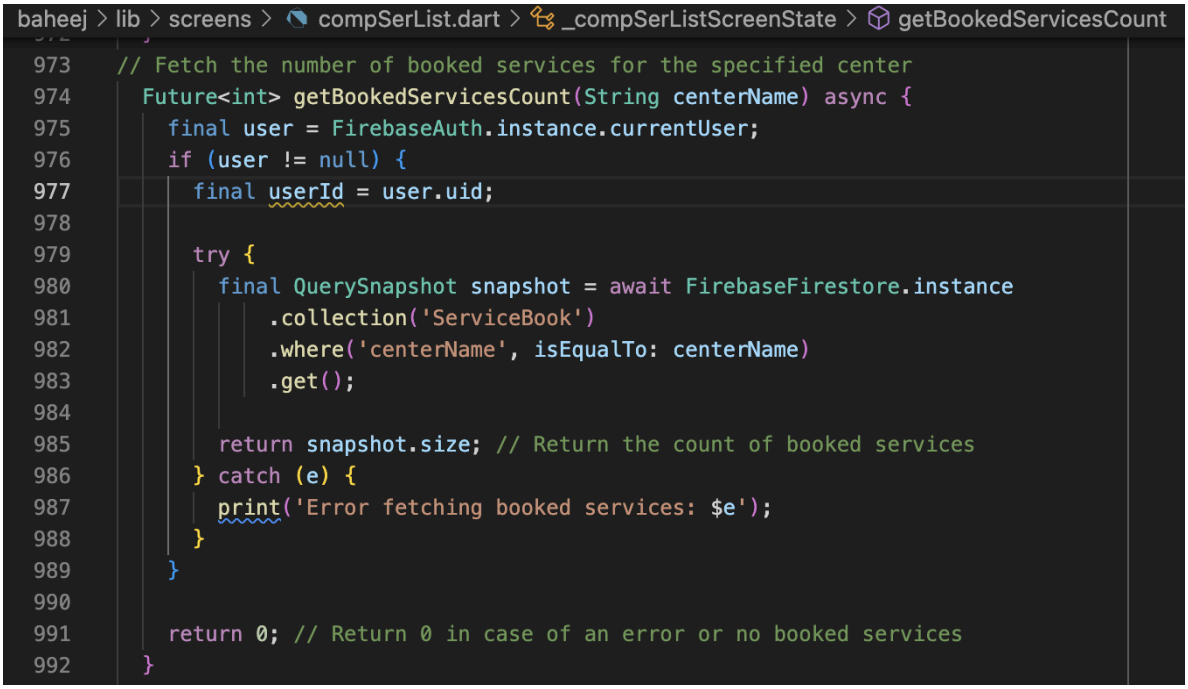

Figure 22: compSerList page before refactoring

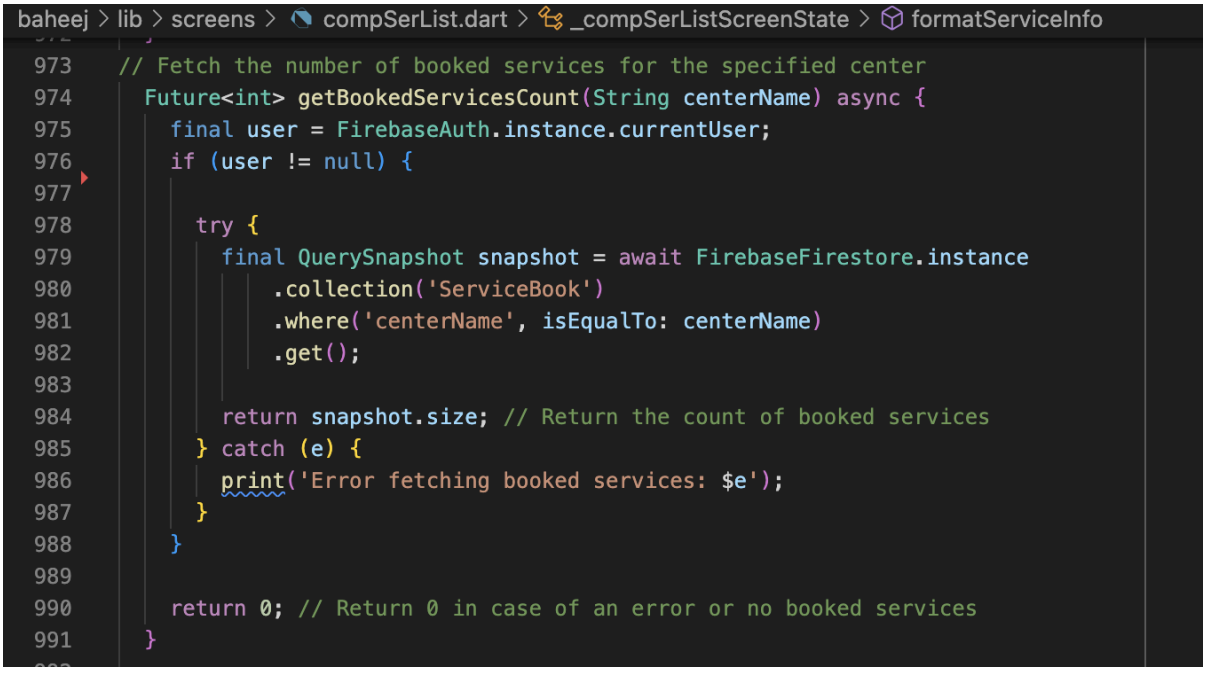

Figure 23: compSerList page after refactoring

5- _handelSearch Method Refactoring

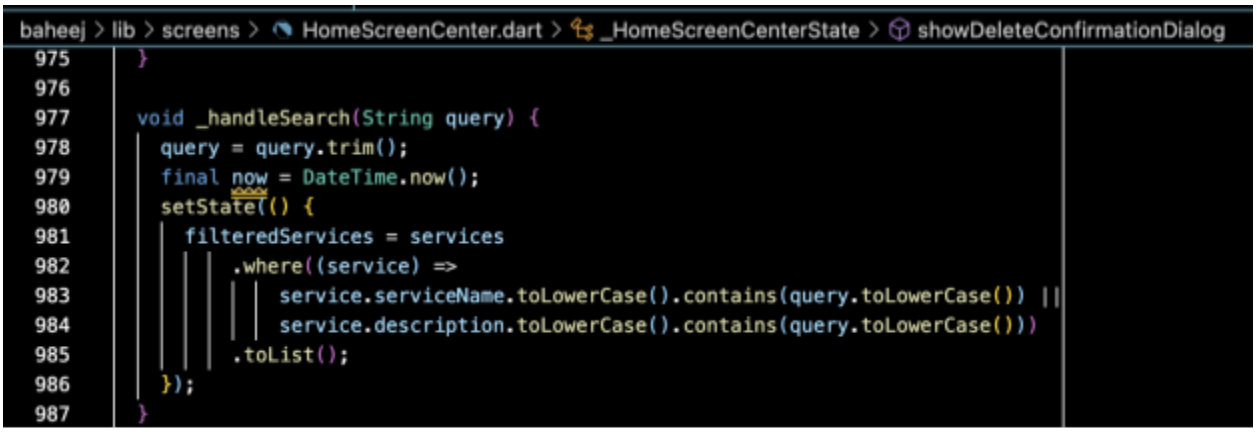

Figure 24: **_handelSearch method before refactoring**

Figure 25: **_handelSearch method after refactoring**