Sip n Snack v 3.0

Muhammad Hashim (BCS181070)

Spring – 2021

Supervised By

Dr. Aamer Nadeem

Department of Computer Science

Capital University of Science & Technology, Islamabad

PROJECT REPORT

Version	V 3.0	**NUMBER OF MEMBERS**	3

TITLE	**Sip n Snack**
SUPERVISOR NAME	DR. Aamer Nadeem

MEMBER NAME	**REG. NO.**	**EMAIL ADDRESS**
Muhammad Hashim	BCS181070	hashimbahi1@gmail.com

APPROVAL CERTIFICATE

This project, entitled as “Sip n Snack” has been approved for the award of

Bachelors of Science in Computer Science

DECLARATION

I/We, hereby, declare that "No portion of the work referred to, in this project has been submitted in support of an application for another degree or qualification of this or any other university/institute or other institution of learning". It is further declared that this undergraduate project, neither as a whole nor as a part there of has been copied out from any sources, wherever references have been provided.

ACKNOWLEDGEMENTS

I would like to express my very great appreciation to Dr. Aamer Nadeem for his valuable and constructive suggestions during the planning and development of this project work. His willingness to give his time so generously has been very much appreciated.

I would like to express my deepest appreciation to all those who provided us the possibility to complete this report. A special gratitude we give to our final year project coordinator, Mr. Ibrar, whose contribution in stimulating suggestions and encouragement, helped us to co-ordinate our project especially in writing this report.

We would also like to extend my thanks to the technicians and Lab assistants of the Labs of the Computer Science department for their help in offering us the resources in running the program.

Finally, we wish to thank our parents for their support and encouragement throughout our study.

DEDICATIONS

This project is dedicated to Dr. Aamer Nadeem, for his kindness and devotion, and for his endless support, his selflessness will always be remembered.

I would like to express my deepest appreciation to all those who provided us the possibility to complete this report. A special gratitude we give to our final year project coordinator, Mr. Ibrar, whose contribution in stimulating suggestions and encouragement, helped us to co-ordinate our project especially in writing this report.

We would also like to extend my thanks to the technicians and Lab assistants of the Labs of the Computer Science department for their help in offering us the resources in running the program.

Finally, we wish to thank our parents for their support and encouragement throughout our study.

Contents

List of Tables

Table of Figures

Chapter 1
Introduction

This project is basically an application that uses Android Application Development methods. The system is meant to be built for only single specific café and the café name is
'Sip n Snack'. Their existing system for food delivery is totally a manual process. The customers actually contact the café boy via internet and ask them for their order. So that the café manager said us to make an app to automate their system of food delivery. Our application features are:

- The Customers has to register themselves on our app first.
- After registering and logging in, the customers can order their food from café.
- Getting Feedback of both food quality and delivery service.
- Recommendation Engine that recommends different food products to customers on the basis of rating/stars given by other customers. It also helps out the café to check which product is likely ordered by many customers. It works on the data that is being gathered from previous customers experience so that the app recommends the food that maybe of their choice to new customers.
- Sentiment Analysis (on feedbacks given by customers) that can analyze the feedback that whether the feedback given is critical or normal or good. This method help the café to improve their food quality and delivery services more and more. Live Location, through which the customer can track their order live on map.
- Online payment gateway, that provides another option to the customer for bill payment purpose.

The project is developed on Android Application platform and is supported by a firebase database to store specific details.

1.1. Purpose of the Project:

We choose this project because all of us know that due to alarming conditions in world, many businesses even the physical businesses were going to online businesses. Food business is already popular in today's world and everyone wants that they can enjoy quality food from different cafes/restaurants without going there. Some people like only specific cafes/restaurants but due to current situations, no one wants that they can visit the café physically. We choose this project to provide such facility that everyone who wants to order from café can order online easily using their mobile phone.

- The application design will be user friendly.
- The GUI of application enable the laymen to use its all functions.

1.2. Existing Examples / Solutions:

There are many other applications that are providing such kind of functionalities and are successfully running in a market. We have selected some similar apps on the basis of most trending applications category and having at least 1 million downloads in the market. Some of these apps are:

“Foodpanda” is well rated app trending now a days and have a great reputation in a market. They manage different types of restaurant on a single plat-form and provide services to their customers.

“Cheetay” is another app that is growing from previous few days to due to current situation in the country. These are providing some other great services along with the food delivery system.

We have tried all these apps and noticed many functions that were useful in food delivery apps and these functions weren't present in these apps. The brief comparison between these apps and our proposed system is as follow:

Sr No.	Characteristics	Foodpanda	Cheetay	Sip n Snack (Proposed System)
1.	Live Location	✓		✓
2.	Online Payment			✓
3.	Feedback	✓	✓	✓
4.	Reports	✓		✓
5.	Manage Expenses			✓

Table 1: Existing Examples / Solutions

1.3. Business Scope:

The online food delivery system business is growing day by day. A good developed app with easy to use GUI can facilitate the business. Our project targets the café that provides the quality food in the town. The Business scope of our project is very wide because the online systems are now a days like a top trend in society. For advanced techniques, we must need to build an algorithm to customize the user option to choose a suitable food for them. We must use some analysis on the basis of feedback so it can help the other customers to order in a suitable way.

The proposed system (Sip n Snack) had a great impact in future on people of society. Because it provides the great functionality to let the people order the food of their choice on their door-step. This app will also help the managers of café to grow their business on large scale. It provides the additional ways to sell their products and reach more customers.

1.4. Useful Tools and Technologies:

Some of the useful tools to develop our application are:

“Android Studio” is an intelligent IDE to build and develop the beautiful android applications. It is portable tool to build android apps.

“Java” is a programming language which is used in Android App Development. Java has huge open source support, with many libraries and tools available to make developers life easier. Java allows them to create sandbox applications, and create a better security model so that one bad App can't take down your entire OS.

“Firebase Database” is a cloud based database used to manage different operations performed on data. We can use Firebase as database for our application.

1.5. Project Work Break Down:

A work-breakdown structure in project management and systems engineering is a deliverable- oriented breakdown of a project into smaller components. A work breakdown structure is a key project deliverable that organizes the team's work into a manageable section.

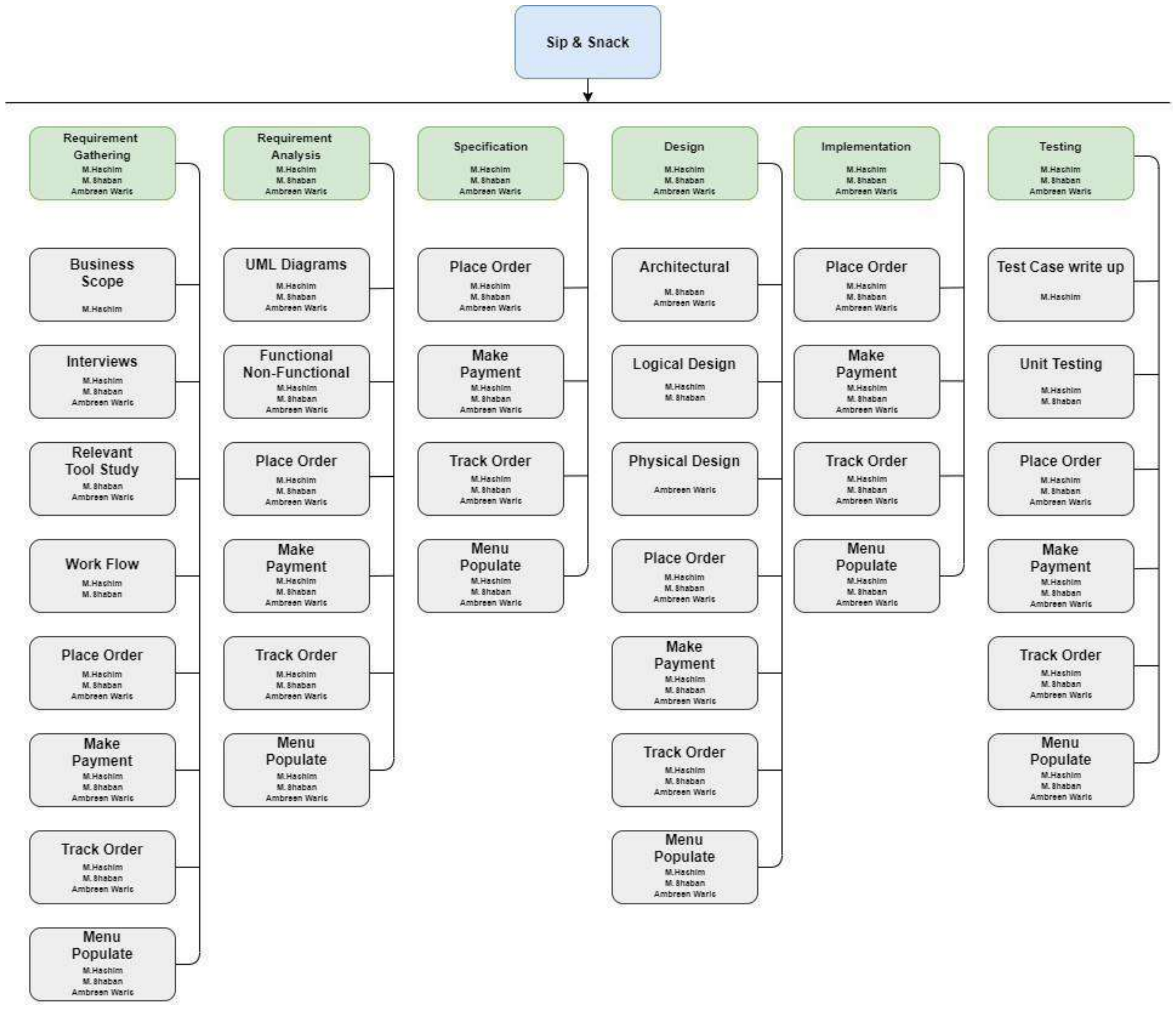


Figure 1: Project Breakdown

1.6. Project Timeline:

The Project timeline is shown in figure as follow:

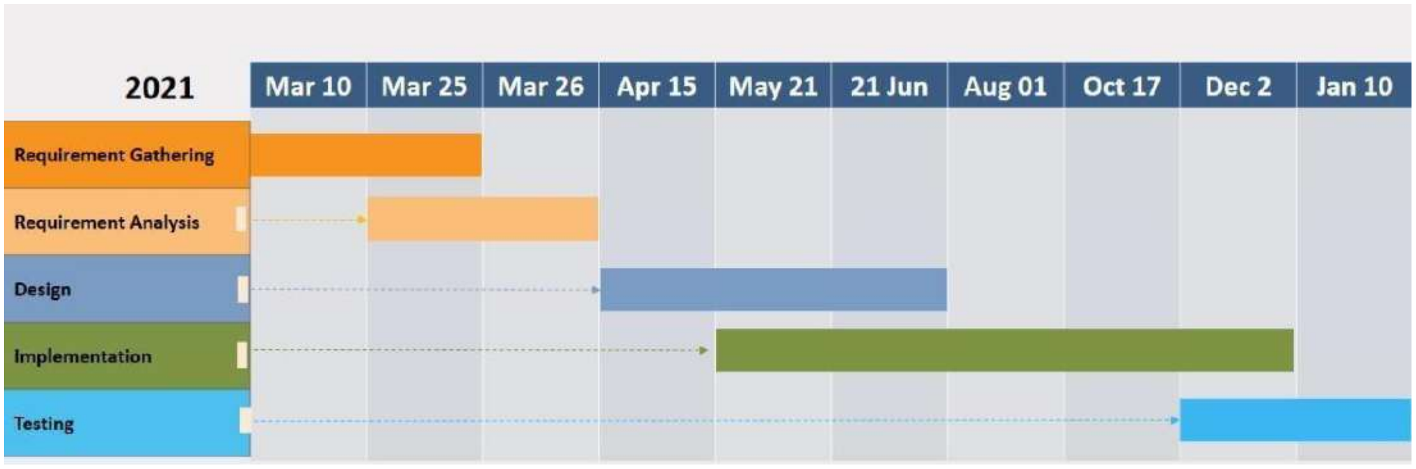


Figure 2: Project Timeline

Chapter 2

Requirement Specification and Analysis

Requirements Analysis is the method of determining consumer requirements for an application to be designed or updated. It includes all the tasks that are carried out to identify the demands of the different stakeholders. For this purpose, requirements analysis involves evaluating, recording, validating and handling software or system requirements. High quality standards are recorded, implementable, measurable, testable, and traceable, help to find business opportunities and are defined in order to facilitate the system design. In Chapter 2 we will enlist the functional and non-functional requirements and model functional requirements in the form of use case model.

2.1. Functional Requirements:

A functional requirement defines a function of a system or its component. A main functions that are going to be implemented in our system is that the customer should be able to view menu and place their order accordingly. Moreover, manager have to perform different functionalities to complete the system process.

Sr No.	Functional Requirements	Type	Status
1.	Customer should be able to Sign up into the system.	Core	Implemented
2.	Customer should be able to Login into the system.	Core	Implemented
3.	Customer should be able to edit their profile.	Core	Implemented
4.	Customer should be able to edit their password.	Core	Implemented
5.	Customer should be able to view all categories of menu.	Intermediate	Implemented
6.	Customer should be able to search across the items.	Core	Implemented
7.	Customer should be able to order food of their choice.	Core	Implemented
8.	Customer should be able to view all items from menu.	Core	Implemented
9.	Customer should be able to select any item from category.	Core	Implemented
10.	Customer should be able to select appropriate quantity of items.	Core	Implemented
11.	Customer should be able to add item to cart.	Core	Implemented
12.	Customer should be able to delete item from cart.	Core	Implemented
13.	Customer should be able to choose between online payments or cash on delivery.	Core	Implemented
14.	Customer should be able to change their information like address or phone number at the time of placing order.	Core	Implemented
15.	Customer should be able to upload image of receipt of payment in case of online payments.	Core	Implemented
16.	Customer should be able to place order.	Core	Implemented

17.	Customer should be able to give feedback for the food after order is delivered.	Core	Implemented
18.	Customer should be able to give feedback for the Biker after order is delivered.	Core	Implemented
19.	Customer should be able to give stars on delivery service.	Core	Implemented
20.	Customer should be able to give stars on food after order is being delivered.		
21.	Customer should be able to cancel the order.	Core	Implemented
22.	Customer should be able to see the status of order after confirmation.	Intermediate	Implemented
23.	Customer should be able to view popular items by the system.	Core	Implemented
24.	Customer can track the order live once the order is prepared.	Core	Implemented
25.	Customer should be able to report any issue in the system.	Intermediate	Implemeneted
26.	Customer should be able to logout from his/her account.	Core	Implemented
27.	Admin should be able to Login into the system.	Core	Implemented
28.	Admin should be able to Add manager account.	Core	Implemented
29.	Admin should be able to View manager account.	Core	Implemented
30.	Admin should be able to Delete manager account.	Core	Implemented
31.	Admin should be able to Update manager account.	Core	Implemented

32.	Admin should be able to logout from his/her account.	Core	Implemented
33.	Manager should be able to Login into the system.	Core	Implemented
34.	Manager should be able to edit their profile.	Core	Implemented
35.	Manager should be able to edit their password.	Core	Implemented
36.	Manager should be able to block customer account.	Core	Implemented
37.	Manager should be able to unblock customer account.	Core	Implemented
38.	Manager should be able to upload popular item banners.	Core	Implemented
39.	Manager should receive notifications every time a new order arrives.	Optional	Implemented
40.	Manager should be able to confirm the order given by the customer.	Core	Implemented
41.	Manager should be able to accept or reject the order given by the customer.	Core	Implemented
42.	Manager should be able to manage and view food items.	Core	Implemented
43.	Manager should be able to view both completed and pending orders.	Core	Implemented
44.	Manager should be able to change the status once the food is prepared.	Core	Implemented
45.	Manager should be able to generate bill through Bluetooth printer attached.	Core	Implemented
46.	Manager should be able to create account of biker.	Core	Implemented
47.	Manager should be able to view and manage bikers.	Core	Implemented

48.	Manager should be able to assign biker to specific order.	Core	Implemented
49.	Manager should be able to add expense.	Core	Implemented
50.	Manager should be able to view expenses recorded on specific date.	Core	Implemented
51.	Manager should be able to view daily reports.	Core	Implemented
52.	Manager should be able to view monthly reports.	Core	Implemented
53.	Manager should be able to view yearly reports.	Core	Implemented
54.	Manager should be able to generate Pdf of reports.	Core	Implemented
55.	Manager should be able to see reports that were reported by bikers and customers.	Core	Implemented
56.	Manager should be able to see feedbacks given upon orders.	Core	Implemented
57.	Biker should have necessary information about order.	Core	Implemented
58.	Biker should be able to contact the customer.	Intermediate	Implemented
59.	Biker should be able to confirm delivery once they deliver the food to customer.	Core	Implemented
60.	Biker should be able to report any issue to the system.	Core	Implemented
61.	User should be able to edit their password.	Core	Implemented
62.	User should be able to edit their profile.	Core	Implemented
63.	Biker should be able to logout from the system.	Core	Implemented
64.	Manager should be able to logot from his/her account.	Core	Implemented

2.2. Non-Functional Requirements:

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions.

S. No.	Non-Functional Requirements	Category
1.	The system should verify the information of person correctly while login to the system.	Security
2.	Only manager should be able to create delivery boys account.	Security
3.	The system should keep and retrieve record correctly.	Reliability
4.	The system's interface should contain the bright icons so that user can easily understand and choose the desired option.	Usability
5.	The system is adaptable even if additional plugins or modules are added at a later point.	Supportability
6.	All the functions of the system must be available to the user every time the system is turned on.	Accessibility
7.	The load on system depends upon the average users of system.	Performance
8.	The system should be error and crash free.	Reliability

Table 2: Non-Functional Requirement

2.4. System Use Case Modeling:

A use case is a list of actions or event steps, typically defining the interactions between a role (known in the Unified Modeling Language as an actor) and a system, to achieve a goal. The actor can be a human or other external system.

2.4.1. Use Case of Admin:

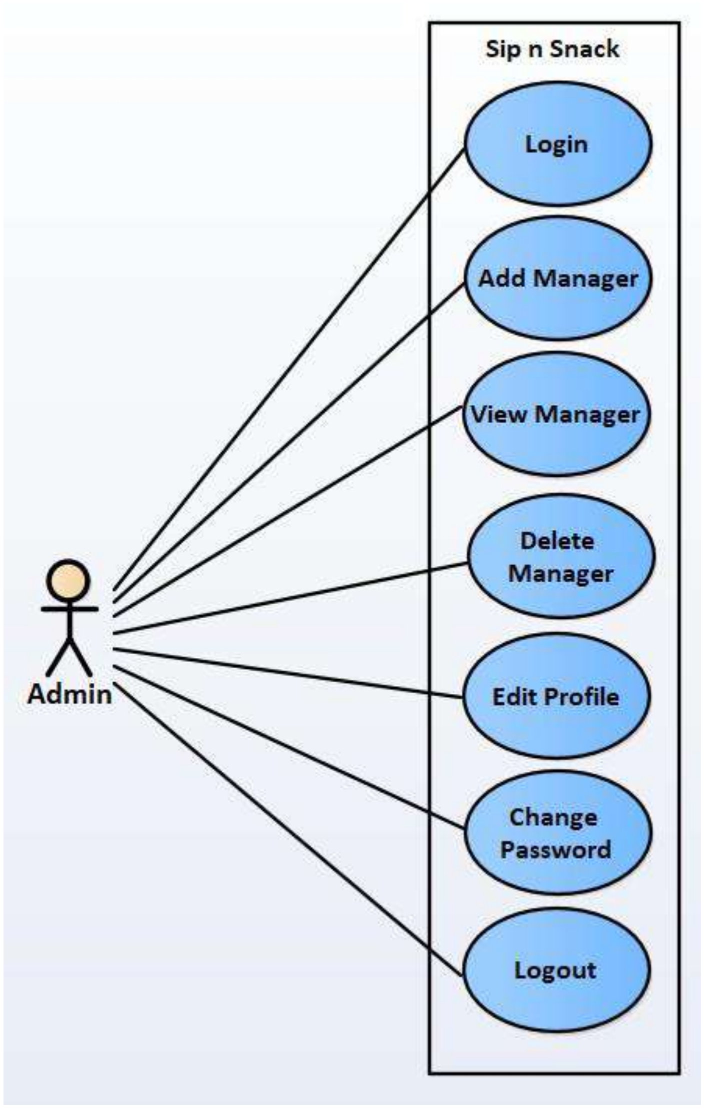


Figure 3: Use Case of Admin

2.4.2. Use Case of Manager:

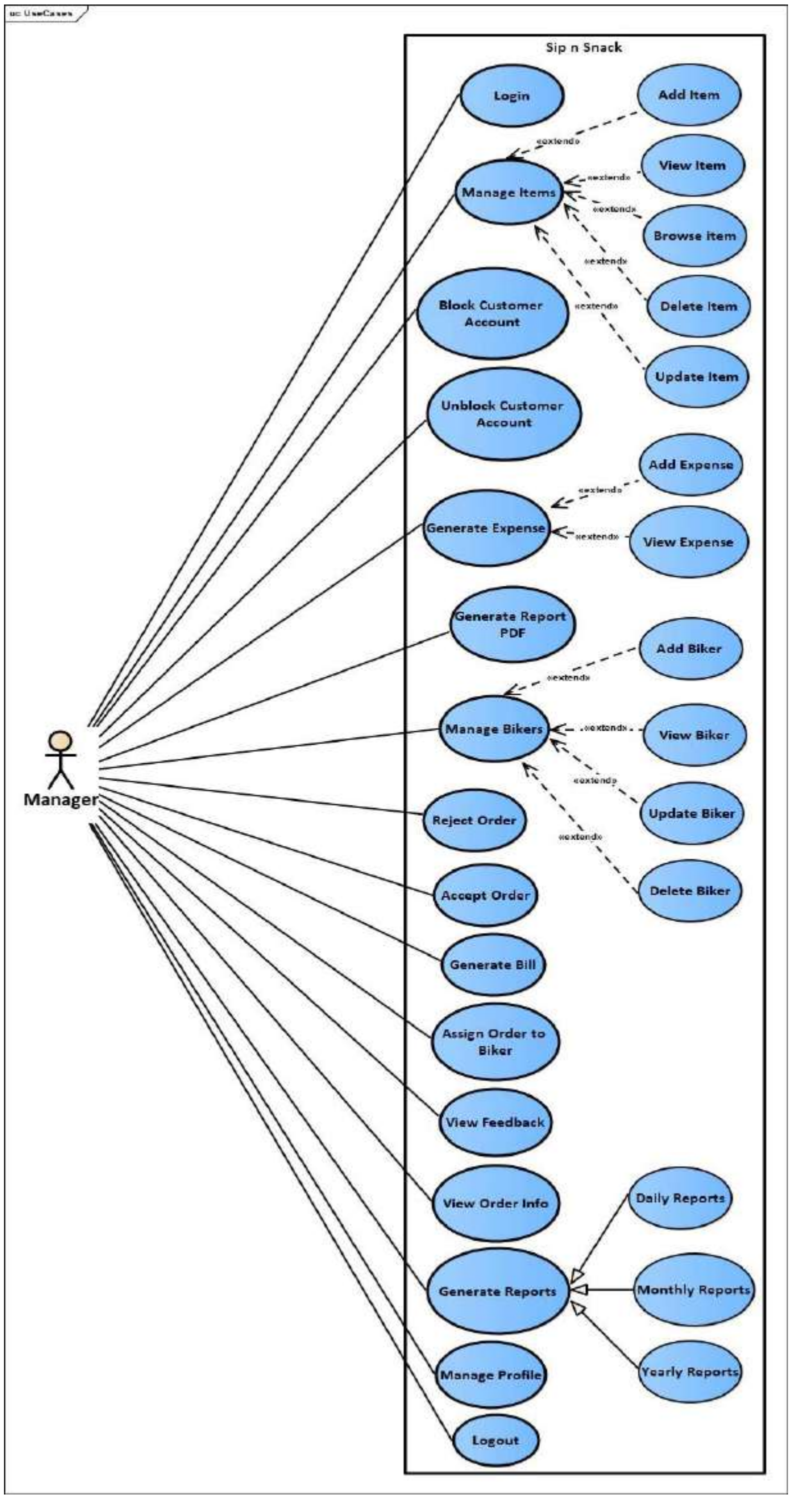


Figure 4: Use Case of Manager

2.4.3. Use Case of Customer:

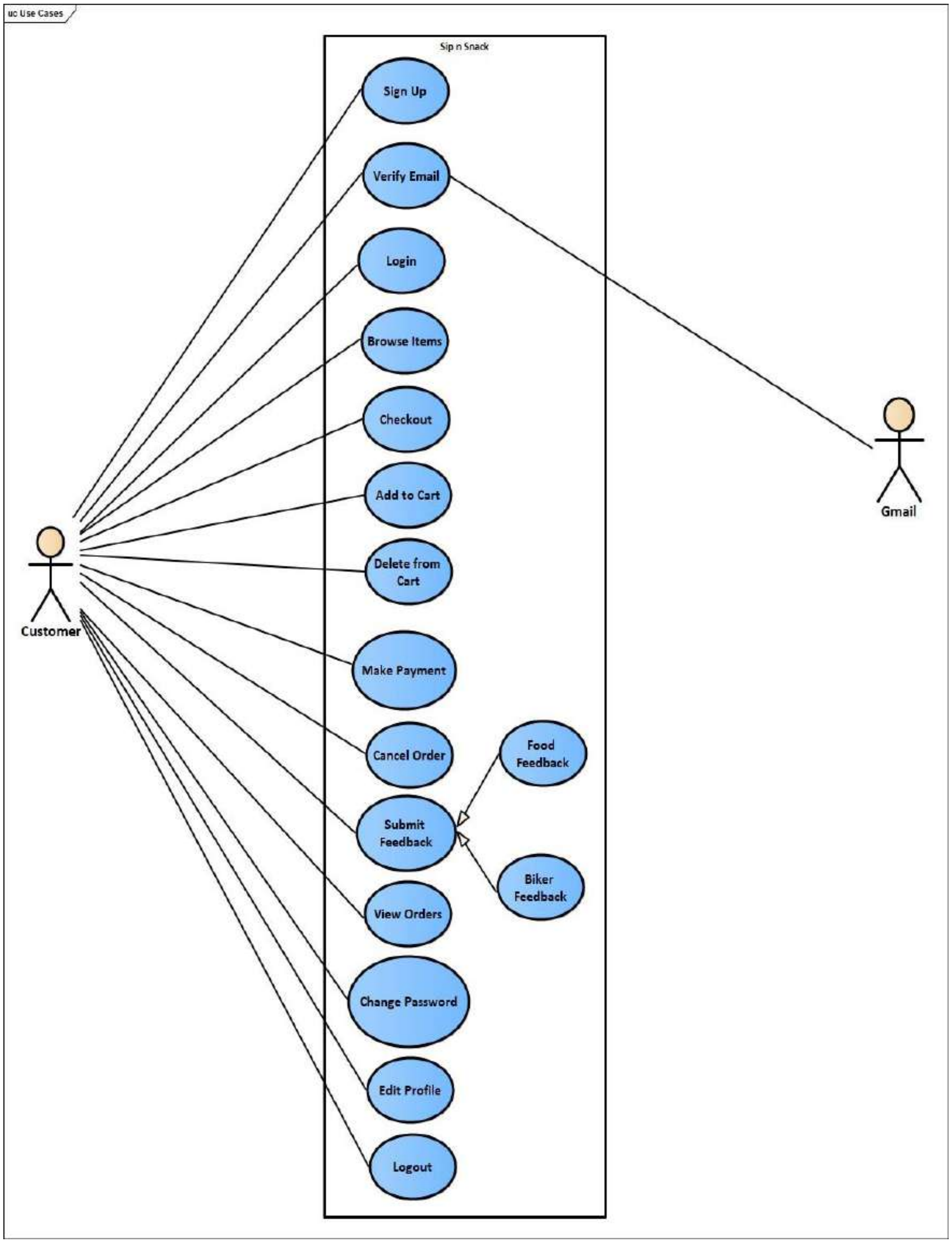


Figure 5: Use Case of Customer

2.4.4. Use Case of Biker:

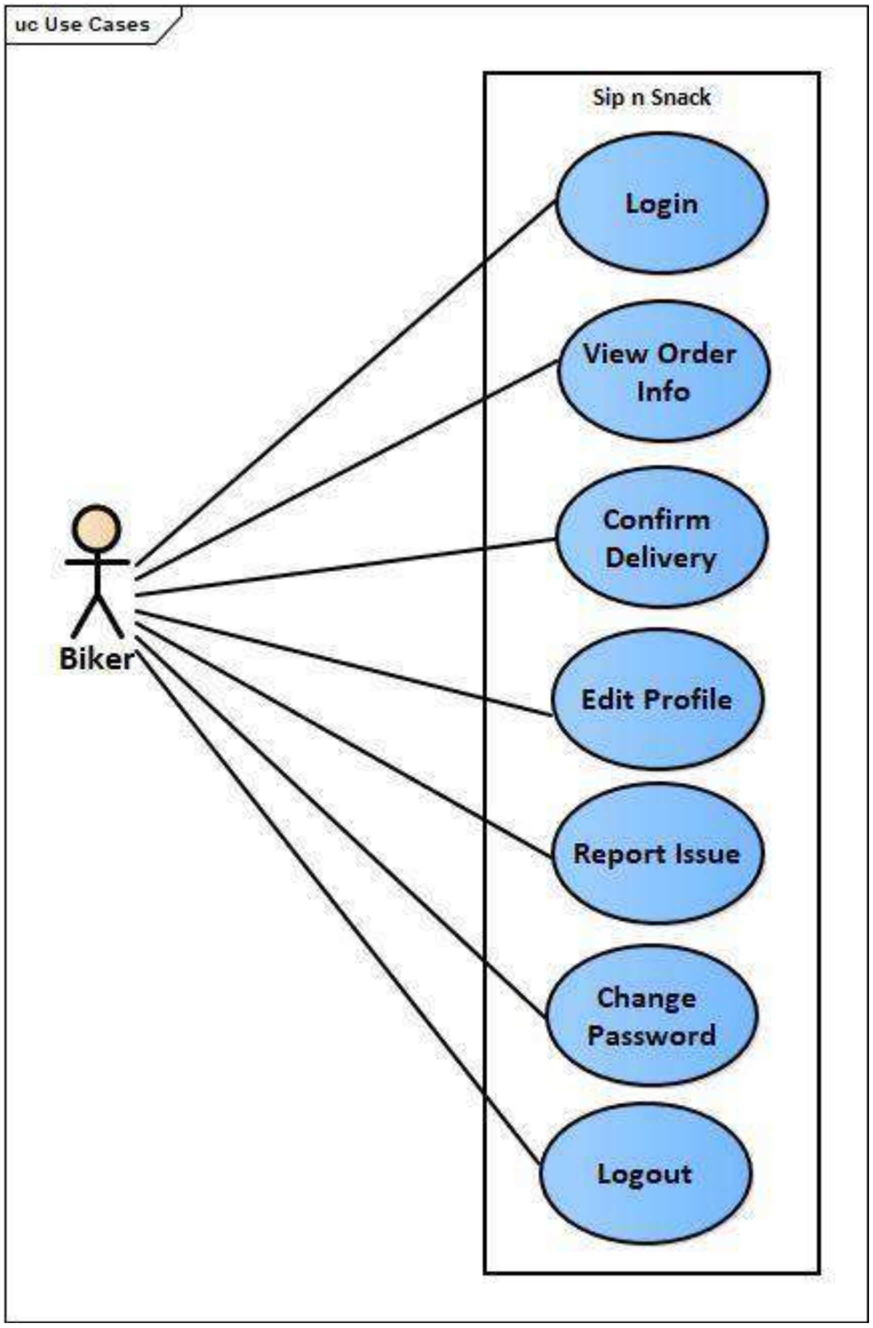


Figure 6: Use Case of Biker

2.4.5. Use Case Description:

<u>Manager Module.</u>

Use Case ID:	Uc1		
Use Case Name:	Signup		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	25 / 05 / 2021	**Last Revision Date:**	28 / 11 / 2021
Actors:	Customer		
Description:	The customer can sign up by the first time he/she uses the system by providing a name, password, address, email and mobile number.		
Trigger:	Signup button		
Preconditions:	The system must be available		
Post conditions:	Customer will be signed up and able to use the system.		
Normal Flow:	**Actor**	**System**	
	1. Customer clicks signup link to request for sign up.	2. The system provides User sign-up form.	
	3. Customer fills in form by providing name, password, address, email and mobile number.	4. System registers the actor and display greeting message.	
		5. System redirects the actor to customer view.	
Alternative Flows:	*a. Customer cancels the signup form.		

	3a. Customer leaves the name field empty. 1. System will generate error message on name textfield. 2. All the fields' data remain same and dismiss the operation. 3b. Customer leaves the username field empty. 1. System will generate error message on username textfield. 2. All the fields' data remain same and dismiss the operation. 3c. Customer leaves the phone no field empty. 1. System will generate error message on phone no textfield. 2. All the fields' data remain same and dismiss the operation. 3d. Customer leaves the password field empty. 1. System will generate error message on password textfield. 2. All the fields' data remain same and dismiss the operation. 3e. Customer leaves the address field empty. 1. System will generate error message on address textfield. 2. All the fields' data remain same and dismiss the operation. 3f. Customer enters different passwords in password & re-enter password field. 1. System will generate error message on address textfield. 2. All the fields' data remain same and dismiss the operation. 3g. Customer uses the username that already registered into the system. 1. System will generate error message. 2. All the fields' data remain same and dismiss the operation. 3h. Customer enters password with length less then 6. 1. System will generate error message on password textfield. 2. All the fields' data remain same and dismiss the operation.

Exceptions:	4a. The database is not responding. 1. Display Error message, Customer provided info remains same in fields. *a. The system is not responding. 1. Show Exception message to the actor.

Table 3: Signup

Login:

Use Case ID:	Uc2		
Use Case Name:	Login		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	22 / 05 / 2021	**Last Revision Date:**	29 / 10 / 2021
Actors:	Customer, Manager, Biker, Admin		
Description:	The actor can login for the first time he/she uses the application by providing username and password and clicking on login button. Actors for this use case are being customer, manager and biker.		
Trigger:	Login button		
Preconditions:	Actor must be already registered in system.		
Post conditions:	Actor would be able to login into system and system will load the home page on screen.		
Normal Flow:	**Actor**	**System**	
	1. Actor clicks login button to request for login operation.	2. The system provides login form that prompts actor for username & password.	
	3. Actor fills in the form by providing username and password.	4. System will verify the actor and login to the system.	
		5. System redirects the actor to main page and save the actor information so that when the actor opens the app next time, they cant have to provide credentials again and again .	
Alternative Flows:	*a. Actor cancels the login form.		

	3a. Actor leaves the username field empty. 1. System will generate error message on username textfield. 2. All the fields' data remain same and dismiss the operation. 3b. Actor leaves the password field empty. 1. System will generate error message on password textfield. 2. All the fields' data remain same and dismiss the operation. 3c. Provided credentials are not correct. 1. System will show error message of invalid credentials. 2. All the fields' data remain same and dismiss the operation.
Exceptions:	4a. The database is not responding. 1. Display Error message, actor provided info remains same in fields. *a. The system is not responding. 1. Show Exception message to the actor.

Table 4: Login

Add Items

Use Case ID:	Uc3		
Use Case Name:	Add Items		
Created By:	Muhammad Hashim	**Last Updated By:**	Muhammad Shaban
Date Created:	25 / 05 / 2021	**Last Revision Date:**	24 / 07 / 2021
Actors:	Manager		
Description:	The Manager will add the items by providing the details of item like Id, Name, Price, Category, Size, and Description.		
Trigger:	Add Items Button		
Preconditions:	Manager is identified and authenticated.		
Post conditions:	Manager should be able to add item to the system successfully.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the items button form their dashboard.	2. The system provides the activity which contains three buttons as add items, view items and browse items.	
	3. Manager clicks the add item button.	4. The system will display the form that prompts the manager to enter details of items.	
	5. Manager should select the specific category of item along with specifc size and provide details of item.	6. System will add the item in database and display the confirmation message.	
Alternative Flows:	*a. Actor cancels the add item form.		

	3a. Manager leaves the id field empty. 1. System will generate error message on Item Id textfield. 2. Provided info remains same. 3b. Manager leaves the name field empty. 1. System will generate error message on Item name textfield. 2. Provided info remains same. 3c. Manager leaves the price field empty. 1. System will generate error message on Item price textfield. 2. Provided info remains same. 3d. Manager leaves the description field empty. 1. System will generate error message on Item description textfield. 2. Provided info remains same. 3e. Manager is entering the id or name of existing item. 1. System will generate error message showing that item already exists.
Exceptions:	6a. The database is not responding. 1. Display Error message, actor provided info remains same in fields.

Table 5: Add Item

View Items:

Use Case ID:	Uc4		
Use Case Name:	View Items		
Created By:	Muhammad Hashim	**Last Updated By:**	Muhammad Shaban
Date Created:	25 / 05 / 2021	**Last Revision Date:**	19 / 07 / 2021
Actors:	Manager		
Description:	The Manager will view the items along with their details and should be able to search specific food item.		
Trigger:	View Items Button		
Preconditions:	1. Manager should be logged in. 2. There must exist some food items in system.		
Post conditions:	Manager can view and search different types of food item.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the items button form their dashboard.	2. The system provides the activity which contains three buttons as add items, view items and browse items.	
	3. Manager clicks the view item button.	4. The system will display all the existing items along with their details.	
	5. Manager can view and search along the food items.	6. System displays the items list matching with searched keyword.	
Alternative Flows:	3a. There exist no food item in database. 1. System will generate message of no item found and redirects to previous page. 5a. Manager search the invalid item.		

	1. System will display empty list of item related to search keyword.
Exceptions:	4a. The database is not responding. 1. Display Error message, actor redirects to previous page.

Table 6: View Items

Browse Items

Use Case ID:	Uc5		
Use Case Name:	Browse Items		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	16 / 07 / 2021	**Last Revision Date:**	25 / 07 / 2021
Actors:	Manager, Customer		
Description:	The Manager and customer both can browse items on the basis of category of items.		
Trigger:	Browse Items Button		
Preconditions:	1. Actror should be logged in to the system. 2. There must already exist some food items in system.		
Post conditions:	Manager and Customer can browse different types of item according to their category.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the items button form their dashboard.	2. The system provides the activity which contains three buttons as add items, view items and browse items.	
	3. Manager clicks the browse items button.		
	4. Actor select the specific category of food item.	5. System displays the items list related to specific category.	
Alternative Flows:	3a. There exist no food item in database. 1. System will generate message of no item found and redirects to previous page.		
Exceptions:	4a. The database is not responding. 1. Display Error message, actor redirects to previous page.		

Table 7: Browse Items

Update Item

Use Case ID:	Uc6			
Use Case Name:	Update Item			
Created By:	Muhammad Shaban		**Last Updated By:**	Muhammad Shaban
Date Created:	22 / 05 / 2021		**Last Revision Date:**	25 / 07 / 2021
Actors:	Manager			
Description:	The Manager will update the item details with the help of item id as unique identifier.			
Trigger:	Update Items Icon			
Preconditions:	1. Actor should be logged in to the system. 2. There must exist item with same item id that manager wants to change. 3. Manager should provide correct details of item.			
Post conditions:	Manager can successfully update the item details.			
Normal Flow:	**Actor**		**System**	
	1. Manager would click the items button form their dashboard.		2. The system provides the activity which contains three buttons as add items, view items and browse items.	
	3. Manager clicks the view item button.		4. The system will display all the existing items along with their details.	
	5. Manager click the edit icon on desired item detail that the manager wants to change.		6. The system will display the details along with delete and edit icon.	

	7. Manager would click the edit icon and then provide the updated details of item and then click the update button.	8. The system will update the item details according to specific item id and save the updated details in database.
		9. The system will show the confirmation message upon the updation of food item.
Alternative Flows:	7a. The manager changes the item id while providing updated details of items. 1. System will generate message of item id is not matching. 2. All the fields remain same and dismiss the operation. 7b. Manager leaves the id field empty. 1. System will generate error message on Item Id textfield. 2. All the fields remain same and dismiss the operation. 7c. Manager leaves the name field empty. 1. System will generate error message on Item name textfield. 2. All the fields remain same and dismiss the operation. 7c. Manager leaves the price field empty. 1. System will generate error message on Item price textfield. 2. All the fields remain same and dismiss the operation. 7d. Manager leaves the description field empty. 1. System will generate error message on Item description textfield. 2. All the fields remain same and dismiss the operation. 7e. Manager exits the application without clicking button. 1. System will reset the textfields. 2. System can’t update the food item details in database.	

Exceptions:	9a. The database is not responding. 1. Display Error message. 2. Starts loading screen. 9b. The internet stopped working. 1. Display Network Connection error message. 2. Starts loading screen.

Table 8: Update Item

Delete Item

Use Case ID:	Uc7		
Use Case Name:	Delete Item		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	16 / 05 / 2021	**Last Revision Date:**	10 / 07 / 2021
Actors:	Manager		
Description:	The Manager will delete the item details with the help of item id as unique identifier.		
Trigger:	Delete Items Icon		
Preconditions:	Actor should be logged in to the system.		
Post conditions:	Manager can successfully delete the item from system.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the items button form their dashboard.	2. The system provides the activity which contains three buttons as add items, view items and browse items.	
	3. Manager clicks the view item button.	4. The system will display all the existing items along with their details.	
	5. Manager click the delete icon on desired item detail that the manager wants to delete.	6. The system will display the details along with delete and edit icon.	
	7. Manager would click the delete icon.	8. The system will display the confirmation dialog.	

	9. The manager will confirm the deletion the of food item.	10. The system will delete the food item and redirects to view items activity.
Alternative Flows:	*a. Manager cancels the current operation. 1. System dismisses the state of application. 7a. The manager cancels the deletion while system is asking for confirmation. 1. System will dismiss the confirmation dialog and redirects to previous activity.	
Exceptions:	10a. The database is not responding. 1. Display Error message. 2. Starts loading screen.	

Table 9: Delete Item

Add Bikers

Use Case ID:	Uc8		
Use Case Name:	Add Bikers		
Created By:	Ambreen Waris	**Last Updated By:**	Muhammad Shaban
Date Created:	02 / 06 / 2021	**Last Revision Date:**	05 / 08 / 2021
Actors:	Manager		
Description:	The Manager will add the bikers account by providing the details of biker like Username, Name, Phone No, and Password etc.		
Trigger:	Add Bikers Button		
Preconditions:	1. Manager is identified and authenticated. 2. Manager should be signed in. 3. System must have an active internet.		
Post conditions:	Manager should be able to create biker account and save details to the system successfully.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the biker's button from their dashboard.	2. The system provides the activity which contains two buttons as add bikers, and view biker.	
	3. Manager clicks the add bikers button.	4. The system will display the form that prompts the manager to enter details of bikers account.	
	5. Manager should provide the details of biker like username, phone no and password etc.	6. System will create the biker account and add the biker details in database and then display the confirmation message.	
Alternative Flows:	*a. Actor cancels the add bikers form.		

	3a. Manager leaves the username field empty. 1. System will generate error message on username textfield. 2. All the fields remain same and dismiss the operation. 3b. Manager leaves the name field empty. 1. System will generate error message on name textfield. 2. All the fields remain same and dismiss the operation. 3c. Manager leaves the password field empty. 1. System will generate error message on password textfield. 2. All the fields remain same and dismiss the operation. 3d. Manager leaves the phone no field empty. 1. System will generate error message on phone no textfield. 2. All the fields remain same and dismiss the operation. 3e. Manager is entering the username of existing biker. 1. System will generate error message showing that biker already exists. 2. All the fields remain same and dismiss the operation. 3f. Manager leaves the address field empty. 1. System will generate error message on address textfield. 2. All the fields remain same and dismiss the operation.
Exceptions:	6a. The database is not responding. 1. Display Error message, actor provided info remains same in fields.

Table 10: Add Bikers

View Bikers

Use Case ID:	Uc9		
Use Case Name:	View Bikers		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	20 / 07 / 2021	**Last Revision Date:**	05 / 08 / 2021
Actors:	Manager		
Description:	The Manager will view the biker along with their details and should be able to search specific biker account.		
Trigger:	View Bikers Button		
Preconditions:	1. Manager should be logged in. 2. There must exist some biker's accounts in system.		
Post conditions:	Manager can view and search different bikers account.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the biker's button form their dashboard.	2. The system provides the activity which contains two buttons as add bikers, and view biker.	
	3. Manager clicks the view biker's button.	4. The system will display all the existing bikers account along with their details.	
	5. Manager can view and search along the bikers account.	6. System displays the searched result of biker's information.	
Alternative Flows:	3a. There exist no bikers account in database. 1. System will generate message of no biker found and redirects to previous page.		

	5a. Manager search for invalid biker acocunt. 1. System will display empty list of biker related to search keyword.
Exceptions:	4a. The database is not responding. 1. Display internet connection error dialog.

Table 11: View Bikers

Update Biker

Use Case ID:	Uc10		
Use Case Name:	Update Bikers		
Created By:	Muhammad Shaban	**Last Updated By:**	Ambreen Waris
Date Created:	20 / 07 / 2021	**Last Revision Date:**	05 / 08 / 2021
Actors:	Manager		
Description:	The Manager will update the biker details with the help of biker username as unique identifier.		
Trigger:	Update Bikers Icon		
Preconditions:	1. Manager should be logged in to the system. 2. There must exist biker account with same username that manager wants to change. 3. Manager should provide correct details of biker.		
Post conditions:	Manager can successfully update the bikers account details.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the biker's button from their dashboard.	2. The system provides the activity which contains two buttons as add bikers, and view bikers.	
	3. Manager clicks the view biker button.	4. The system will display all the existing bikers account along with their details.	
	5. Manager click the edit icon on desired biker detail that the manager wants to change.	6. The system will display the details of biker account along with delete and edit icon.	

	7. Manager would click the edit icon and then provide the updated details of biker and then click the update button.	8. The system will update the biker details according to specific biker username and save the updated details in database.
		9. The system will show the confirmation message upon the updation of biker account details.
Alternative Flows:	7a. The manager changes the biker username while providing updated details of biker account. 1. System will generate message of biker username is not matching. 2. All the fields remain same and dismiss the operation. 7b. Manager leaves the username field empty. 1. System will generate error message on username textfield. 2. All the fields remain same and dismiss the operation. 7c. Manager leaves the name field empty. 1. System will generate error message on name textfield. 2. All the fields remain same and dismiss the operation. 7d. Manager leaves the phone no field empty. 1. System will generate error message on phone no textfield. 2. All the fields remain same and dismiss the operation. 7e. Manager leaves the address field empty. 1. System will generate error message on address textfield. 2. All the fields remain same and dismiss the operation. 7f. Manager exits the application without clicking button. 1. System will reset the textfields. 2. System can't update the biker details in database.	

Exceptions:	9a. The database is not responding. 1. Display Error message. 2. Starts loading screen, 9b. The internet stopped working. 1. Display Network Connection error message. 2. Starts loading dialog.

Table 12: Update Biker

Delete Biker

Use Case ID:	Uc11		
Use Case Name:	Delete Biker		
Created By:	Muhammad Shaban	**Last Updated By:**	Ambreen Waris
Date Created:	20 / 07 / 2021	**Last Revision Date:**	05 / 08 / 2021
Actors:	Manager		
Description:	The Manager will delete the biker account along with details with the help of biker username as unique identifier.		
Trigger:	Delete Bikers Icon		
Preconditions:	Actor should be logged in to the system.		
Post conditions:	Manager can successfully delete the biker account along with details from system.		

Normal Flow:	**Actor**	**System**
	1. Manager would click the biker's button form their dashboard.	2. The system provides the activity which contains two buttons as add bikers, and view bikers.
	3. Manager clicks the view biker's button.	4. The system will display all the existing bikers account along with their details.
	5. Manager click the delete icon on desired biker detail that the manager wants to delete.	6. The system will display the details along with delete and edit icon.
	7. Manager would click the delete icon.	8. The system will display the confirmation dialog.

	9. The manager will confirm the deletion of biker account.	10. The system will delete the biker account along with details and redirects to view biker's activity.
Alternative Flows:	*a. Manager cancels the current operation. 1. System dismisses the state of application. 7a. The manager cancels the deletion while system is asking for confirmation. 1. System will dismiss the confirmation dialog and redirects to previous activity.	
Exceptions:	10a. The database is not responding. 1. Display Error message. 10b. The internet stopped working. 1. Display Network Connection error message.	

Table 13: Delete Biker

Block Customer Account

Use Case ID:	Uc12		
Use Case Name:	Block Customer Account		
Created By:	Muhammad Shaban	**Last Updated By:**	Ambreen Waris
Date Created:	25 / 09 / 2021	**Last Revision Date:**	17 / 10 / 2021
Actors:	Manager		
Description:	The Manager will block the account of customer temporarily if the customer can do some undesirable acts like replacing order and then cancel the order frequently.		
Trigger:	Block Customer Icon		
Preconditions:	1. Manager should be logged in to the system. 2. There must exist customers in system.		
Post conditions:	Manager can temporarily block the customers account when the customers cancelled orders reaches 10.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the manage customers button from their dashboard.	2. The system provides the activity in which all the customers information were displaying like their username, name, no of cancelled orders, account status etc.	
	3. Manager clicks the block button if he/she detects an undesireable acts by customer.	4. The system will ask for the confirmation of temporarily block of customer account.	

	5. Manager clicks the confirm button to confirm the temporarily block of customer account.	6. The system will block the customer account and set the account status to block.
Alternative Flows:	*a. Manager cancels the current operation. 1. System dismisses the state of application. 5a. The manager cancels the block of customer account while system is asking for confirmation. 1. System will dismiss the confirmation dialog and redirects to previous activity.	
Exceptions:	6a. The database is not responding. 1. Display Error message.	

Table 14: Block Customer Account

Update Banners

Use Case ID:	Uc13		
Use Case Name:	Update Banners		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammd Shaban
Date Created:	02 / 11 / 2021	**Last Revision Date:**	02 / 11 / 2021
Actors:	Manager		
Description:	The Manager will be able to add an images for the customer's main view. Manager should be able to update the banner if they want to change the images view.		
Trigger:	Popular Button Icon		
Preconditions:	1. Actor should be logged in to the system.		
Post conditions:	Manager can add or update the banners of customer view.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the popular icon button from their dashboards navigation bar.	2. The system provides the activity in which the images are displayed respectively and the button that upload / change the image in database.	
	3. Manager long click the image and select image of their choice from their device.		
	4. Manager clicks the button to ensure the upload of selected image.	5. The system will update the image and save the image into database.	

Alternative Flows:	*a. Manager cancels the current operation. 1. System dismisses the state of application. 4a. The manager directly clicks the upload button without selecting any image. 1. System will show the message that ask user to select an image first.
Exceptions:	5a. The database is not responding. 1. Display Error message.

Table 15: Update Banners

Admin Module.

Add Managers

Use Case ID:	Uc14		
Use Case Name:	Add Managers		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Hashim
Date Created:	29 / 08 / 2021	**Last Revision Date:**	10 / 10 / 2021
Actors:	Admin		
Description:	The Admin will add the managers account in system by providing the details of manager like Username, Name, Phone No, and Password etc.		
Trigger:	Add Managers Button		
Preconditions:	1. Admin is identified and authenticated. 2. Admin should be signed in. 3. System must have an active internet.		
Post conditions:	Admin should be able to create manager account and save details to the system successfully.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the manage managers button from their dashboard.	2. The system provides the activity which contains two buttons as add managers, and view managers.	
	3. Admin clicks the add managers button.	4. The system will display the form that prompts the admin to enter details of manager's account.	
	5. Admin should provide the details of manager like username, phone no and password etc.	6. System will create the manager account and add the manager details in database and then display the confirmation message.	

62

Alternative Flows:	*a. Admin cancels the add managers form. 3a. Admin leaves the username field empty. 1. System will generate error message on username textfield. 2. All the fields remain same and dismiss the operation. 3b. Admin leaves the name field empty. 1. System will generate error message on name textfield. 2. All the fields remain same and dismiss the operation. 3c. Admin leaves the password field empty. 1. System will generate error message on password textfield. 2. All the fields remain same and dismiss the operation. 3d. Admin leaves the phone no field empty. 1. System will generate error message on phone no textfield. 2. All the fields remain same and dismiss the operation. 3e. Admin is entering the username of existing manager. 1. System will generate error message showing that manager with username already exists.	
Exceptions:	6a. The database is not responding. 1. Display Error message, actor provided info remains same in fields.	

Table 16: Add Managers

View Managers

Use Case ID:	Uc15		
Use Case Name:	View Managers		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	30 / 10 / 2021	**Last Revision Date:**	05 / 11 / 2021
Actors:	Admin		
Description:	The Admin will view the manager along with their details and should be able to search specific manager account.		
Trigger:	View Managers Button		
Preconditions:	1. Admin should be logged in. 2. There must exist some manager's accounts in system.		
Post conditions:	Admin can view and search different managers account.		
Normal Flow:	**Actor**	**System**	
	1. Admin would click the manager's button form their dashboard.	2. The system provides the activity which contains two buttons as add manager's, and view manager.	
	3. Admin clicks the view manager's button.	4. The system will display all the existing managr's account along with their details.	
	5. Admin can view and search along the manager's account.	6. System will display the managers list related to search keyword.	
Alternative Flows:	3a. There exist no manager's account in database. 1. System will generate message of no manager found and redirects to previous page.		

	5a. Admin search for invalid manager acocunt. 1. System will display empty list of biker related to search keyword.
Exceptions:	4a. The database is not responding. 1. Display internet connection error dialog.

Table 17: View Managers

<u>Delete Manager</u>

Use Case ID:	Uc16		
Use Case Name:	Delete Manager		
Created By:	Muhammad Shaban	**Last Updated By:**	Ambreen Waris
Date Created:	31 / 10 / 2021	**Last Revision Date:**	05 / 11 / 2021
Actors:	Admin		
Description:	The Admin will delete the biker account along with details with the help of biker username as unique identifier.		
Trigger:	Delete Manager Icon		
Preconditions:	Admin should be logged in to the system.		
Post conditions:	Admin can successfully delete the manager account along with details from system.		

Normal Flow:	**Actor**	**System**
	1. Admin would click the biker's button form their dashboard.	2. The system provides the activity which contains two buttons as add managers, and view managers.
	3. Admin clicks the view manager button.	4. The system will display all the existing manager account along with their details.
	5. Admin click the delete icon on desired manager detail that the admin wants to delete.	6. The system will display the details along with delete and edit icon.
	7. Admin would click the delete icon.	8. The system will display the confirmation dialog.

	9. The Admin will confirm the deletion of manager account.	10. The system will delete the manager account along with details and redirects to view manager's activity.
Alternative Flows:	*a. Admin cancels the current operation. 1. System dismisses the state of application. 7a. The Admin cancels the deletion while system is asking for confirmation. 1. System will dismiss the confirmation dialog and redirects to previous activity.	
Exceptions:	10a. The internet stopped working. 1. Display Network Connection error message. 2. Loading dialog starts.	

Table 20: Delete Manager

Customer Module.

Add to Cart

Use Case ID:	Uc17		
Use Case Name:	Add to Cart		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	11 / 11 / 2021	**Last Revision Date:**	11 / 11 / 2021
Actors:	Customer		
Description:	The Customer will be able to see the menu categories and the food items that were added by the manager and the customer can select the food items of their choice and add that item to cart.		
Trigger:	Add to Cart Button		
Preconditions:	1. Customer should be logged in. 2. There must exist some food items already in system.		
Post conditions:	Customer can successfully add item to cart.		
Normal Flow:	**Actor**	**System**	
	1. Customer would click the menu icon or button from their bottom nav bar.	2. The system redirects the customer to item categories screen.	
	3. Customer would click on their desired menu category.	4. The system will display the list of all items belongs to that selected category along with few details.	
	5. Customer can click on their desired food item to show the further details of item and select the appropriate quantity of food according to their choice.	6. System display the details of item like its price, description, quantity etc along with add to cart button.	

	7. Customer clicks add to cart button.	8. System display message of added successfully and add that item to cart along with its total price, quantity etc.
Alternative Flows:	*a. Actor cancels the current operation. 1. System dismisses the state of application. 3a. There exist no items belonging to that specific category in database. 1. System will generate message of no item founded and redirects to previous page. 7a. It takes time to add an item to the cart. 1. System will display message to customer for wait and hold and displays the loading screen on device.	
Exceptions:	4a. The database is not responding. 1. Display loading dialog screen. 8a. The database is not responding. 1. Display loading dialog screen.	

Table 18: Add to Cart

Delete from Cart

Use Case ID:	Uc18		
Use Case Name:	Delete from Cart		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	11 / 11 / 2021	**Last Revision Date:**	11 / 11 / 2021
Actors:	Customer		
Description:	The Customer will be able to see the items that they had added to their cart and they can delete the desired item from their cart.		
Trigger:	Delete from Cart Icon		
Preconditions:	1. Customer should be logged in. 2. There must exist some items in customer cart.		
Post conditions:	Customer can successfully delete item from cart.		
Normal Flow:	**Actor**	**System**	
	1. Customer would click the Cart icon or button from their bottom nav bar.	2. The system redirects the customer to cart screen where the items are listed and displayed that were added by customer.	
	3. Customer would click on delete icon of respective item that he / she wants to delete from cart.	4. The system will display the confirmation dialog to customer.	
	5. Customer can click the yes button to confirm deletion of item from cart.	6. System will delete the item from the list of items in cart and finally from the cart of customer.	
Alternative Flows:	*a. Actor cancels the current operation. 1. System dismisses the state of application.		

	1a. There exist no items in the cart of customer. 1. System will display the text of cart is empty on screen. 5a. Customer clicks the no button at the time of confirmation. 1. System will dismiss the confirmation dialog and redirects to previous screen.
Exceptions:	4a. The system takes time to delete item from cart due to internet. 1. Display loading screen.

Table 19: Delete from Cart

Make Payment

Use Case ID:	Uc19		
Use Case Name:	Make Payment		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	14 / 11 / 2021	**Last Revision Date:**	14 / 11 / 2021
Actors:	Customer		
Description:	The Customer will be able to pay their order bill online.		
Trigger:	Online Payment Option		
Preconditions:	1. Customer should be logged in. 2. Customer should have some items in cart. 3. Customer should opted for online payment.		
Post conditions:	Customer can successfully pay their order bill online.		
Normal Flow:	**Actor**	**System**	
	1. Customer would click the Cart icon or button from their bottom nav bar.	2. The system redirects the customer to cart screen where the items are listed and displayed that were added by customer.	
	3. Customer would click confirm button to confirm their oder.	4. The system will display option for payment of bill.	
	5. Customer can click the online payment option to opt for an online payment.	6. System will provide account numbers, next button and an image view where the customer can upload the receipt of payment that they have sent to provided accounts.	

	7. Customer can pay the amount of bill to the account number displaying on screen and upload the image of receipt of payment on system.	8. System will display the message of successfully placing order.
Alternative Flows:	*a. Actor cancels the current operation. 1. System dismisses the state of application. 7a. Customer clicks the next button without uploading receipt image. 1. System will display the reminder to upload receipt image.	
Exceptions:	8a. The system takes time to upload image of receipt. 1. Display loading screen and ask user to wait.	

Table 20: Make Payment

Checkout

Use Case ID:	Uc20		
Use Case Name:	Checkout		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	14 / 11 / 2021	**Last Revision Date:**	14 / 11 / 2021
Actors:	Customer		
Description:	The Customer will be able to place the order according to their cart.		
Trigger:	Place Order Button		
Preconditions:	1. Customer should be logged in. 2. Customer must have added some items in the cart.		
Post conditions:	Customer can place order successfully.		
Normal Flow:	**Actor**	**System**	
	1. Customer would click the Cart icon or button from their bottom nav bar.	2. The system redirects the customer to cart screen where the items are listed and displayed that were added by customer.	
	3. Customer would click confirm button to confirm their order.		
	4. Customer can select the payment method and click on next and then provide the information about delivery of order and clicks the confirm button.	5. System will place their order according to their cart that contains the items list and total bill along with that.	

Alternative Flows:	*a. Actor cancels the current operation. 1. System dismisses the state of application. 3a. Customer cart is empty. 1. System will not visible the confirm button until customer has added at least one item in cart. 4a. Customer clicks the next button without uploading receipt image in case of online payment option. 1. System will display the reminder to upload receipt image.
Exceptions:	5a. The system doesn’t place order. 1. Display error message and ask user to try again.

Table 21: Place Order

Cancel Order

Use Case ID:	Uc21		
Use Case Name:	Cancel Order		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	14 / 11 / 2021	**Last Revision Date:**	14 / 11 / 2021
Actors:	Customer		
Description:	The Customer will be able to cancel the placed order.		
Trigger:	Cancel Order Button		
Preconditions:	1. Customer should be logged in. 2. Customer must have placed an order first.		
Post conditions:	Customer can cancel the placed order successfully.		
Normal Flow:	**Actor**	**System**	
	1. Customer would click the orders button or option from navigation menu.	2. The system redirects the customer to orders that they have placed in system.	
	3. Customer would click cancel order button to cancel the respective order.	4. System displays confirmation of cancellation of order.	
	5. Customer can click on yes button to confirm the cancellation of order.	6. System will cancel the order that is being requested by the customer.	
Alternative Flows:	*a. Actor cancels the current operation. 1. System dismisses the state of application.		

	5a. Customer clicks the no button at the time of confirmation 1. System will dismiss the confirmation dialog and redirects to previous screen.
Exceptions:	6a. The system tooks a time to cancel order. 1. Display loading screen and ask user to wait for a while.

Table 22: Cancel Order

View Order Status

Use Case ID:	Uc22		
Use Case Name:	View Order Status		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	14 / 11 / 2021	**Last Revision Date:**	14 / 11 / 2021
Actors:	Customer		
Description:	The Customer will be able to track the order and status of order after the order is being accepted bby the manager.		
Trigger:	Track Order Button		
Preconditions:	1. Customer should be logged in. 2. Customer must have placed an order. 3. Order must be in accepted state.		
Post conditions:	Customer can track the order and see the status of placed order successfully.		
Normal Flow:	**Actor**	**System**	
	1. Customer would click the orders button or option from navigation menu.	2. The system redirects the customer to orders that they have placed in system.	
	3. Customer would click track order button to track the respective order.	4. System displays the status and the option of tracking order live.	
	5. Customer can click on track order live option to track the live location of order.	6. System will display the location of biker who carries their order for delivery.	
Alternative Flows:	*a. Actor cancels the current operation. 1. System dismisses the state of application.		

	5a. Customer navigate back to previous page without tracking the live location. 1. System will redirect to previous page.
Exceptions:	6a. The system isn't updating the live location of biker due to connectivity issues. 1. Display the recent location of biker on screen.

Table 23: View Order Status

Submit a Feedback

Use Case ID:	Uc23		
Use Case Name:	Submit a Feedback		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	14 / 11 / 2021	**Last Revision Date:**	14 / 11 / 2021
Actors:	Customer		
Description:	The Customer will be able to submit a feedback to system at any time for the betterment of system.		
Trigger:	Feedback Button		
Preconditions:	Customer should be logged in.		
Post conditions:	Customer can successfully submit a feedback to the system.		
Normal Flow:	**Actor**	**System**	
	1. Customer would click the orders button or option from navigation menu.	2. The system redirects the customer to orders screen.	
	3. Customer would go to the delivered orders tab.		
	4. Customer would click the food or biker feedback button from respective order.	5. System will diplay a dialog that prompts customer to submit a feedback and give rating upon the order.	
	6. Customer would give the feedback and click submit button.	7. System displays the confirmation message of successfully submitted the feedback.	

Alternative Flows:	4a. Customer leaves the feedback text field empty. 1. System generates error message on feedback textfield and ask user to enter again. 2.
Exceptions:	7a. The system isn’t submitting a feedback of customer. 1. Display the error message and ask to try again and the text in feedback textfield remains same.

Table 24: Submit a Feedback

Verify Email

Use Case ID:	Uc24		
Use Case Name:	Verify Email		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	20 / 02 / 2022	**Last Revision Date:**	20 / 02 / 2022
Actors:	Customer		
Description:	The Customer will be able to generate verification link from system and then verify their email at the time of signup.		
Trigger:	Signup Button		
Preconditions:	Email should be valid and never being used already in system.		
Post conditions:	Customer can successfully verify their email.		
Normal Flow:	**Actor**	**System**	
	1. Customer would click the singup button after inserting all the required fields for signup.	2. The system redirects the customer to verification email sent dialog.	
	3. Customer would go to their email and click on verification link.	4. System login the actor when customer is authenticated.	
Alternative Flows:	2a. Customer provides invalid email. 1. System generates error Email textfield and ask user to enter again.		
Exceptions:	3a. The system isn't sending verification link due to any reason. 1. Display the error message and ask user to try again after few time.		

Table 25: Verify Email

Report an Issue

Use Case ID:	Uc25		
Use Case Name:	Submit a Feedback		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	08 / 01 / 2022	**Last Revision Date:**	21 / 01 / 2022
Actors:	Customer, Biker		
Description:	The Actor will be able to report any issue in the system for the betterement of system in future versions.		
Trigger:	Report an Issue Button		
Preconditions:	Actor should be logged in.		
Post conditions:	Actor can successfully report an issue to the system.		
Normal Flow:	**Actor**	**System**	
	1. Actor would click the report an issue button or option from navigation menu.	2. The system redirects the actor to report issue activity.	
	3. Actor fills the text field with respective issue and click on submit button.	4. System displays the confirmation message of successfully submitted report to system.	
Alternative Flows:	3a. Actor leaves the issue text field empty. 1. System generates error message on issue textfield and ask actor to enter again.		
Exceptions:	4a. The system isn't submitting a feedback of customer. 1. Display the error message and ask to try again and the text in issue textfield remains same.		

Table 26: Report Issue

Accept Order

Use Case ID:	Uc26		
Use Case Name:	Accept Order		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	15 / 11 / 2021	**Last Revision Date:**	15 / 11 / 2021
Actors:	Manager		
Description:	The Manager will be able to accept the placed order by the customer.		
Trigger:	Accept / Reject Order Button		
Preconditions:	1. Manager should be logged in. 2. Customer must have placed an order.		
Post conditions:	Manager can accept the order placed by a customer successfully.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the orders button or option from their dashboard.	2. The system redirects the manager to orders screen that contains button of accepted orders, pending orders and completed orders.	
	3. Manager would click the pending orders button in order to view the orders that are not accepted yet.	4. System displays all the orders that were placed by customers but not yet accepted.	
	5. Manager can confirm about the order from the customer by contacting them.	6. System can confirm the acceptance of order and set the status of order to in progress.	

Alternative Flows:	*a. Actor cancels the current operation. 1. System dismisses the state of application. 5a. Manager rejects the desired order that is being placed by the customer. 1. System set the respective order status to reject.
Exceptions:	6a. The database is not responding. 1. Display error dialog and ask manager to try again.

Table 27: Accept Order

Generate Bill

Use Case ID:	Uc27		
Use Case Name:	Generate Bill		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	16 / 11 / 2021	**Last Revision Date:**	17 / 11 / 2021
Actors:	Manager		
Description:	The manager will be able to generate a bill through Bluetooth printer.		
Trigger:	Update Status Button.		
Preconditions:	1. Manager should be logged in. 2. Bluetooth printer must be configured. 3. Printer must be connected to Bluetooth.		
Post conditions:	Manager can generate bill through printer successfully.		
Normal Flow:	**Actor**	**System**	
	1. Actor would click the accepted orders button.	2. The system redirects the manager to view accepted orders activity.	
	3. Manager clicks on print button in order to print the receipt.	4. System will generate receipt and the bill is being generated by printer.	
Alternative Flows:	*a. Actor cancels the current operation. 1. System dismisses the state of application. 4a. Manager select the incorrect printer device. 1. System displays message of not connected. 2. Redirects to previous page.		

	6a. Printer powers off unexpectedly. 1. System aborts the operation. 2. Prompts the manager to try again.
Exceptions:	3a. Bluetooth isn't present in device. 1. System display a message of no configuration found. 2. System redirects to previous page. 4a. Printer is not printing in correct format. 1. Repeat the process and try again.

Table 28: Generate Bill

View Order Info

Use Case ID:	Uc28		
Use Case Name:	View Order Info		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	15 / 11 / 2021	**Last Revision Date:**	17 / 11 / 2021
Actors:	Manager, Biker		
Description:	The actor will be able to view the order information.		
Trigger:	View Order Option.		
Preconditions:	1. Actor should be logged in. 2. Respective order must be placed.		
Post conditions:	Actor can see the information of respective order successfully.		
Normal Flow:	**Actor**	**System**	
	1. Actor would click the view orders option.	2. The system redirects the actor to orders screen where they can see the relevant information about order.	
	3. Actor would click the ok button to redirect to previous page.	4. System will redirect the actor to previous page.	
Alternative Flows:	1a. Actor cancels the current operation. 1. System go back to previous state.		
Exceptions:	2a. The database is not responding. 1. Display error dialog and ask actor to try again.		

Table 29: View Order Info

Assign Order to Biker

Use Case ID:	Uc29		
Use Case Name:	Assign Order to Biker		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	15 / 11 / 2021	**Last Revision Date:**	15 / 11 / 2021
Actors:	Manager		
Description:	The Manager will be able to assign the order to biker.		
Trigger:	Assign Order Button.		
Preconditions:	1. Manager should be logged in. 2. Customer must have placed an order. 3. Order must be ready to deliver. 4. There must exit some bikers account in system.		
Post conditions:	Manager can assign an order to biker successfully.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the orders button or option from their dashboard.	2. The system redirects the manager to orders screen that contains button of accepted orders, pending orders and completed orders.	
	3. Manager would click the accepted orders button in order to view the orders that are being accepted.	4. System displays all the orders that were placed by customers and are accepted by manager.	
	5. Manager can click on respective order.	6. System displays the biker list that are available for delivery.	

	7. Manager can assign and order to a biker.	8. System can successfully assign the order to a biker along wih order information like receipt of bill, customer name, phone no etc.
Alternative Flows:	*a. Actor cancels the current operation. 1. System dismisses the state of application. 5a. There is no biker available for the delivery of order. 1. System display the empty list of bikers 2. System displays the message of no biker available. 5b. There are no bikers in the system. 1. Display message of there must be some bikers in system.	
Exceptions:	6a. The system takes time to display the list of bikers. 1. Display error dialog and ask manager to try again. 8a. Database is not responding. 1. Display message of error occurred.	

Table 30: Assign Order to Biker

Generate Reports

Use Case ID:	Uc30		
Use Case Name:	Generate Reports		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	14 / 11 / 2021	**Last Revision Date:**	14 / 11 / 2021
Actors:	Manager		
Description:	The Manager will be able to generate and view reports about sale of food items.		
Trigger:	Reports / Sales Button		
Preconditions:	1. Manager should be logged in. 2. There must be some orders placed already by customers.		
Post conditions:	Manager can successfully generate and view reports of sales.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the reports button or option from navigation menu.	2. The system redirects the manager to generate reports screen where there is three options as daily, monthly and yearly reports.	
	3. Manager would click the respective option.	4. The system redirects to specific reports page and display the categories wise sale and respective pie-chart of sales.	
Alternative Flows:	3a. There is no generated report due to no sale on specific date. 1. System displays a message of no report founded and redirects to previous screen.		
Exceptions:	4a. The database is not responding. 1. Display the message of try again.		

Table 31: Generate Reports

<u>View Submitted Feedbacks</u>

Use Case ID:	Uc31		
Use Case Name:	View a Submitted Feedbacks		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	14 / 11 / 2021	**Last Revision Date:**	14 / 11 / 2021
Actors:	Manager		
Description:	The Manager will be able to view and read the submitted feedbacks by a customer.		
Trigger:	Feedbacks Button		
Preconditions:	1. Manager should be logged in. 2. There must be some feedbacks submitted by the customers.		
Post conditions:	Manager can successfully view a feedbacks submitted by customers.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the feedback button or option from navigation menu.	2. The system redirects the manager to feedback screen.	
		3. System displays the list of all submitted feedbacks by the customer.	
Alternative Flows:	1a. There is no feedbacks submitted. 1. System display a message of no feedback submitted and redirects to previous page.		
Exceptions:	2a. The system is not responding. 1. Display the message of try again.		

Table 32: View Submitted Feedbacks

Add Expense

Use Case ID:	Uc32		
Use Case Name:	Add Expense		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	10 / 01 / 2022	**Last Revision Date:**	10 / 01 / 2022
Actors:	Manager		
Description:	The Manager will add the expense that will occur on specific date.		
Trigger:	Add Expense Button		
Preconditions:	1. Manager is identified and authenticated. 2. Manager should be signed in. 3. System must have an active internet.		
Post conditions:	Manager should be able to add expense successfully.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the expense button from their dashboard.	2. The system provides the activity which contains two buttons as add expense, and view expense.	
	3. Manager clicks the add expense button.	4. The system will display the form that prompts the manager to enter details of expense account.	
	5. Manager should provide the details of expense like category and amount etc.	6. System will add the expense to system and displays confirmation message.	
Alternative Flows:	*a. Actor cancels the add expense form. 3a. Manager leaves the expense field empty. 1. System will generate error message on expense amount textfield.		

Exceptions:	6a. The database is not responding. 1. Display Error message, actor provided info remains same in fields.

Table 33: Add Expense

View Expense

Use Case ID:	Uc33		
Use Case Name:	View Expense		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	10 / 01 / 2022	**Last Revision Date:**	10 / 01 / 2022
Actors:	Manager		
Description:	The Manager will view the expense details by specifying the date that were added by the manager in past.		
Trigger:	View Expense Button		
Preconditions:	1. Manager should be logged in. 2. There must exist some manager's accounts in system.		
Post conditions:	Manager can view and search different expenses by date.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the expense button from their dashboard.	2. The system provides the activity which contains two buttons as add expense, and view expense.	
	3. Manager clicks the view expense button.	4. The system will redirect the manager to activity where the dates were displayed on which specific expense is recorded.	
	5. Manager can click on specific date to view expense.	6. System will display the expenses recorded on selected date along with pie-chart for detailed view.	
Alternative Flows:	5a. There exist no expense of selected date in database. 1. System will generate message of no expense found and redirects to previous page.		

Exceptions:	4a. The database is not responding. 1. Display internet connection error dialog.

Table 34: View Expense

Generate Reports PDF

Use Case ID:	Uc34		
Use Case Name:	Generate Reports PDF		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	10 / 01 / 2022	**Last Revision Date:**	10 / 01 / 2022
Actors:	Manager		
Description:	The Manager will generate the pdf format file of report.		
Trigger:	Generate Pdf Icon		
Preconditions:	1. Manager should be logged in. 2. There must exist some orders in the system.		
Post conditions:	Manager can generate pdf file of report successfully.		
Normal Flow:	**Actor**	**System**	
	1. Manager would click the reports button from their dashboard.	2. The system provides the activity which contains three buttons as daily, monthly and yearly reports.	
	3. Manager clicks the respective option button.	4. The system will redirect the manager to activity where the reports of specific date/month/year were displayed.	
	5. Manager can click on gernrate pdf icon to generate pdf file of report.	6. System will generate the pdf file of report and display the confirmation message.	
Alternative Flows:	3a. There exist no report of selected option in database. 1. System will generate message of no reports founded and redirects to previous page.		

Exceptions:	4a. The database is not responding. 1. Display internet connection error dialog.

Table 35: Generate Reports PDF

Edit Profile

Use Case ID:	Uc35		
Use Case Name:	Edit Profile		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	04 / 11 / 2021	**Last Revision Date:**	04 / 11 / 2021
Actors:	Customer, Manager, Biker, Admin		
Description:	The actor can be able to update their profile like their name, phone number or address etc. except for their usernames.		
Trigger:	Edit Profile Button		
Preconditions:	Actor should be logged in to the system.		
Post conditions:	Actor can successfully view the popular items by the system.		
Normal Flow:	**Actor**	**System**	
	1. Actor would login to the system.	2. The System login the actor and redirects to main screen.	
	3. Actor can switch to My Profile screen.	4. System will display the information of actor along with edit profile button.	
	5. Actor click the edit profile button.	6. System will redirect the actor to an activity where information about their account were filled in the textboxes along with update button.	
	7. Actor will prompt their desired changes and then finally click the update button.	8. System will update the information that was changed by the actor.	

Alternative Flows:	*a. Actor cancels the current operation. 1. System dismisses the state of application. 3a. Actor clikcs the back button. 1. System exits the app from device. 5a. Actor cancels the current operation. 1. System will redirects to previous screen. 7a. Actor provides invalid updated details. 1. System pop up an error message and the information remains same. 2. System prompts user to enter again.
Exceptions:	2a. The System failed to login the actor. 1. Display the message of invalid credentials and try again. 8a. The System was failed due to internet. 1. Display the message of try again.

Table 36: Edit Profile

Change Password

Use Case ID:	Uc36		
Use Case Name:	Change Password		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	04 / 11 / 2021	**Last Revision Date:**	04 / 11 / 2021
Actors:	Customer, Manager, Biker, Admin		
Description:	The actor can be able to change their password by providing the old password and good length new password.		
Trigger:	Change Password Button		
Preconditions:	Actor should be logged in to the system.		
Post conditions:	Actor can successfully change the password of their account.		
Normal Flow:	**Actor**	**System**	
	1. Actor would login to the system.	2. The System login the actor and redirects to main screen.	
	3. Actor can click the change password button.	4. System will redirect to the activity where it can prompt actor to enter old, new and re enter new password.	
	5. After providing the details, actor can click the change button.	6. System will the actor to main screen and change password of account successfully.	
Alternative Flows:	3a. Actor clikcs the back button. 1. System exits the app from device. 5a. Actor cancels the current operation. 1. System will redirects to previous screen.		

	5b. Actor entered old password didn’t matched with current password of their account. 1. System shows the error message on old password text-field. 5c. Actor entered new and re-enter new password both don’t match with each other. 1. System show the error message on new password field.
Exceptions:	2a. The System failed to login the actor. 1. Display the message of invalid credentials and try again. 6a. The System was failed due to internet. 1. Display the message of try again.

Table 37: Change Password

Logout

Use Case ID:	Uc37		
Use Case Name:	Logout		
Created By:	Muhammad Shaban	**Last Updated By:**	Muhammad Shaban
Date Created:	20 / 07 / 2021	**Last Revision Date:**	05 / 08 / 2021
Actors:	Manager, Customer, Biker, Admin		
Description:	The actor can be able to logout from the system.		
Trigger:	Logout Button.		
Preconditions:	Actor should be logged in to the system.		
Post conditions:	Manager can successfully logged out from the system.		
Normal Flow:	**Actor**	**System**	
	1. Actor would click on logout button.	2. The system pop-up a confirmation dialog.	
	3. Actor confirms the logout operation.	4. The system will logout the actor from system and can clear all its log in information from device.	
Alternative Flows:	*a. Actor cancels the current operation. 1. System dismisses the state of application. 3a. Manager cancels the current operation. 1. System dismisses the popup dialog and load the previous state of application.		
Exceptions:	4a. The System stopped working. 1. Display the message of trying the operation again.		

Table 38: Logout

2.5. System Sequence Diagram:

System sequence diagram (SSD) is a sequence diagram that shows, for a particular scenario of a use case, the events that external actors generate their order, and possible inter-system events.

SSD Login.

Here, User means: Admin, Customer, Manager and Biker.

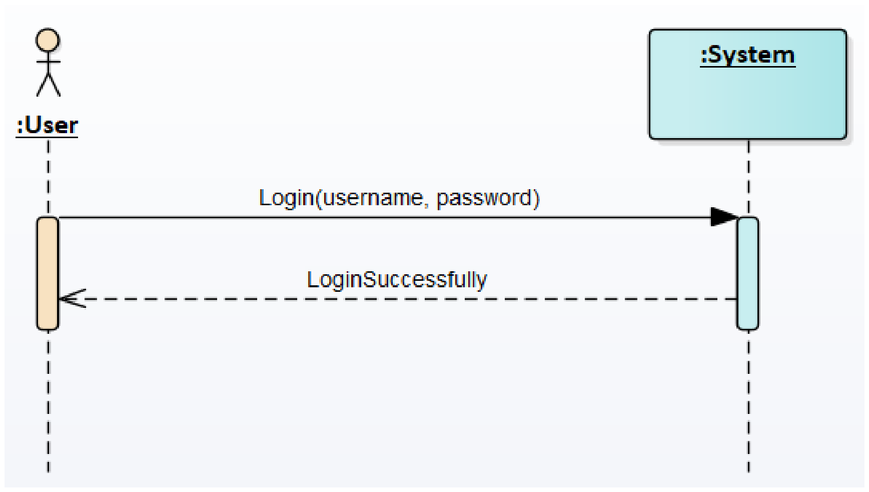


Figure 7: SSD Login

SSD Logout.

Here, User means: Admin, Customer, Manager and Biker.

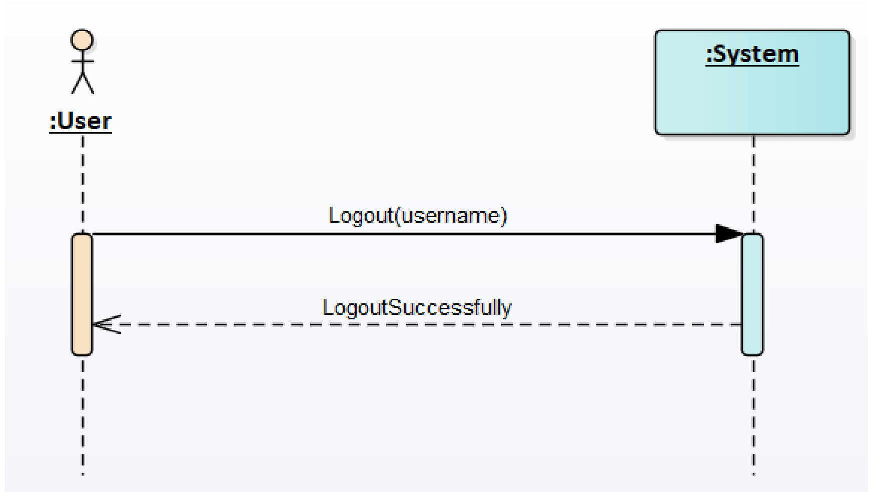
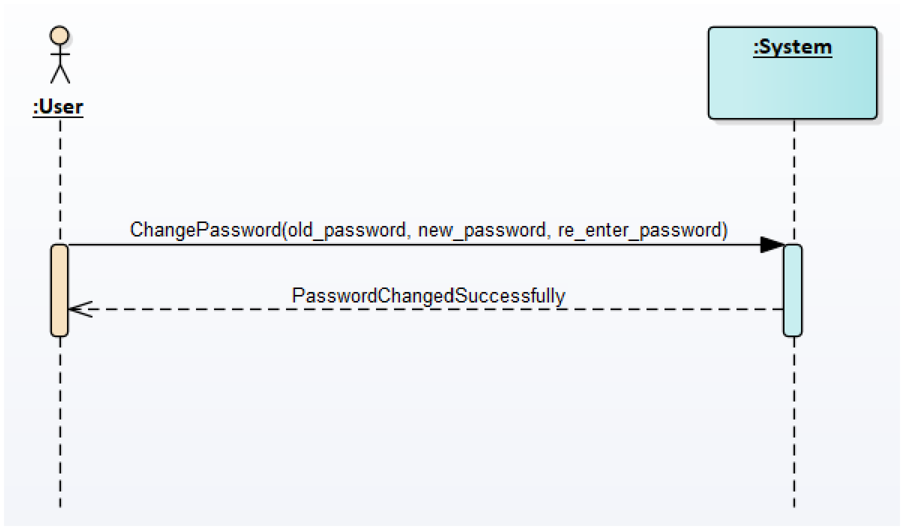


Figure 8: SSD Logout

SSD Change Password.

Here, User means: Admin, Customer, Manager and Biker.

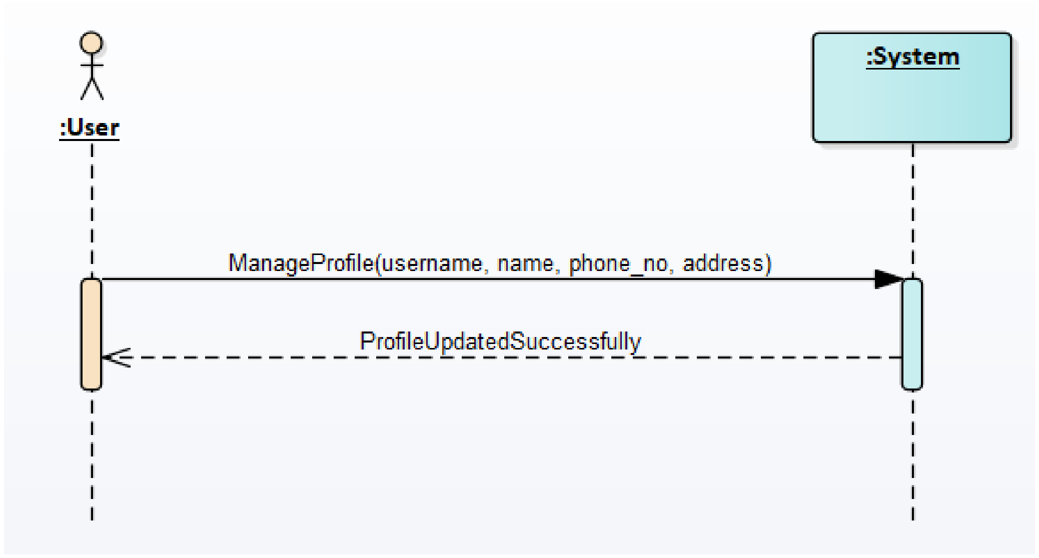


:User
:System
ChangePassword(old_password, new_password, re_enter_password)
PasswordChangedSuccessfully

Figure 9: SSD Change Password

SSD Manage Profile.

Here, User means: Admin, Customer, Manager and Biker.



:User
:System
ManageProfile(username, name, phone_no, address)
ProfileUpdatedSuccessfully

Figure 10: SSD Manage Profile

SSD Report Issue.

Here ‘Actor’ implies, Biker and Customer who can report issue related to system.

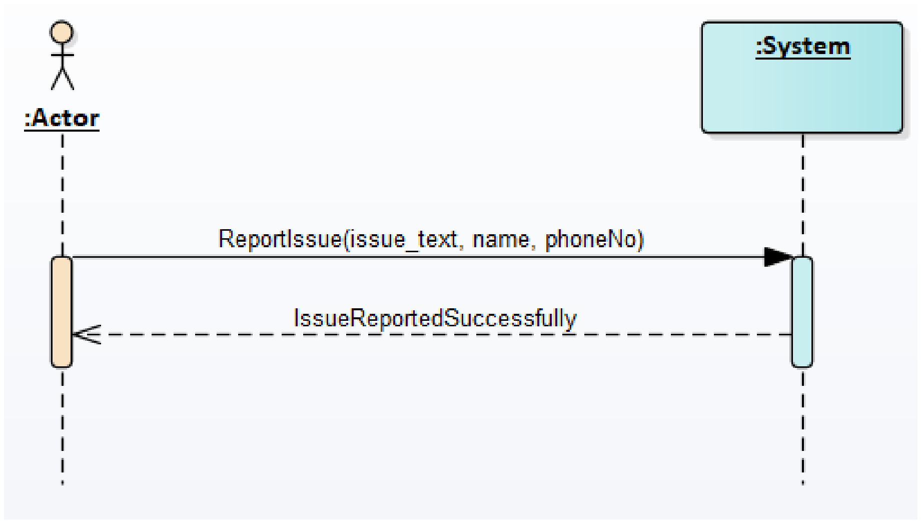

:Actor
:System
ReportIssue(issue_text, name, phoneNo)
IssueReportedSuccessfully

Figure 11: SSD Report Issue

For Admin:

SSD Add Manager.

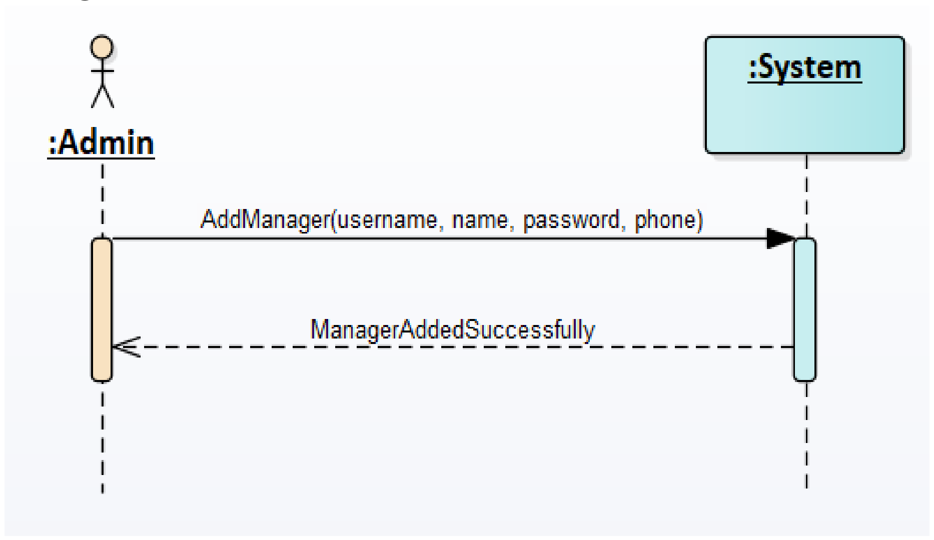


Figure 12: SSD Add Manager

SSD View Manager.

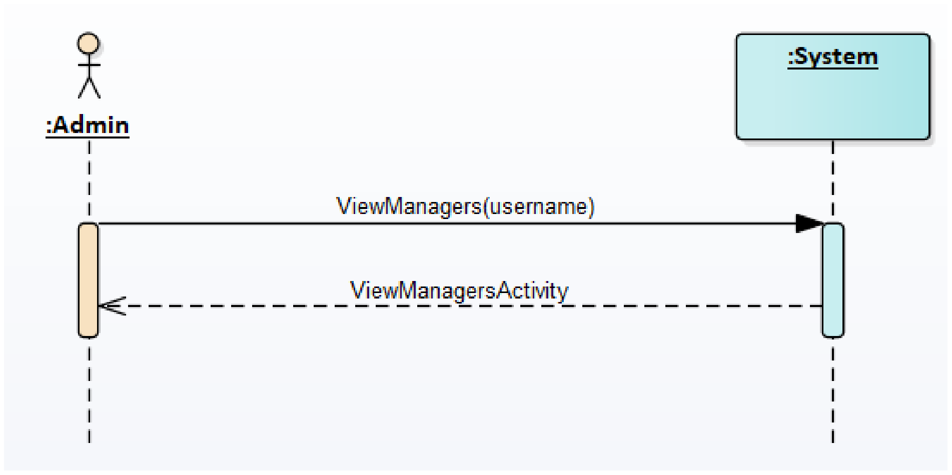


Figure 13: SSD View Manager

SSD Delete Manager.

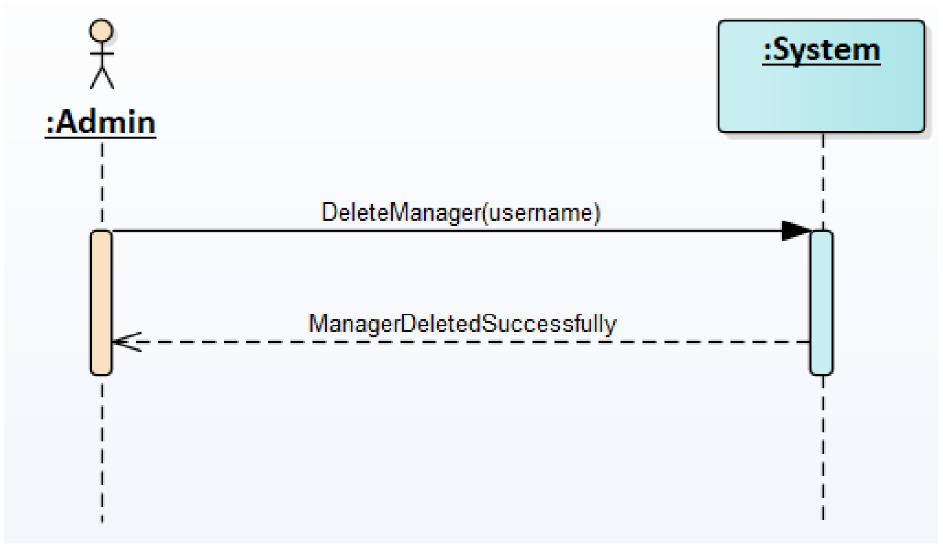


Figure 14: SSD Delete Manager

For Manager:
SSD Add Item.

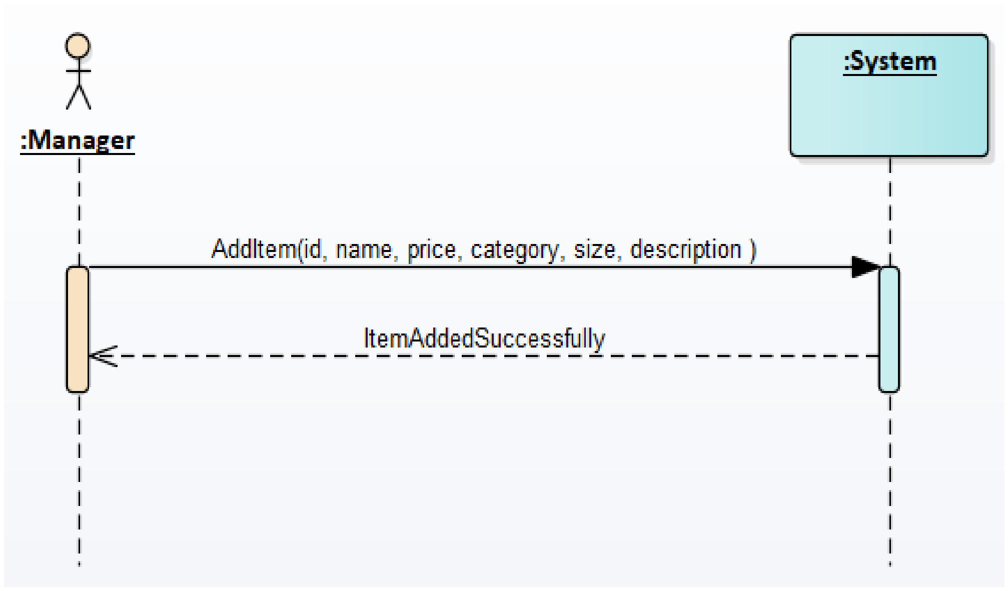


Figure 15: SSD Add item

SSD View Item.

Here, User means Customer and Manager Both.

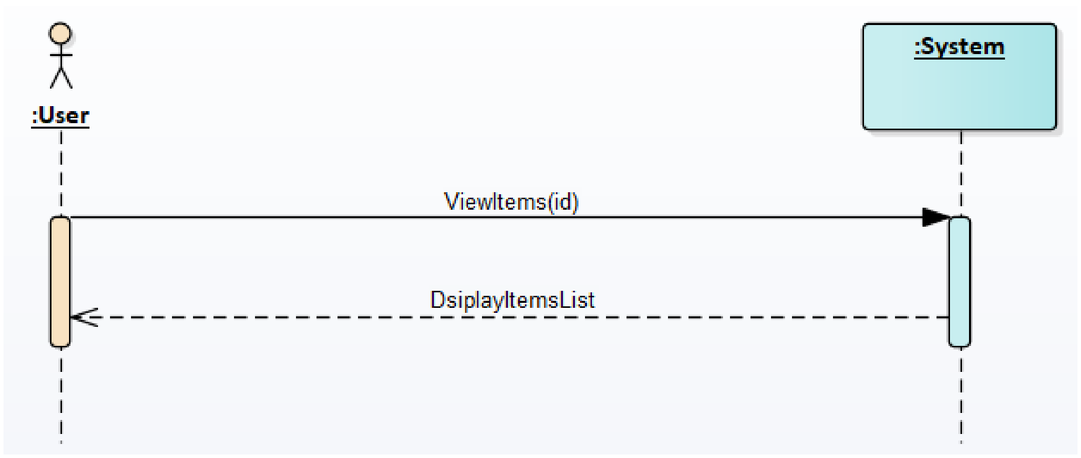


Figure 16: SSD ViewItem

SSD Browse Item.

Here, User means Customer and Manager Both.

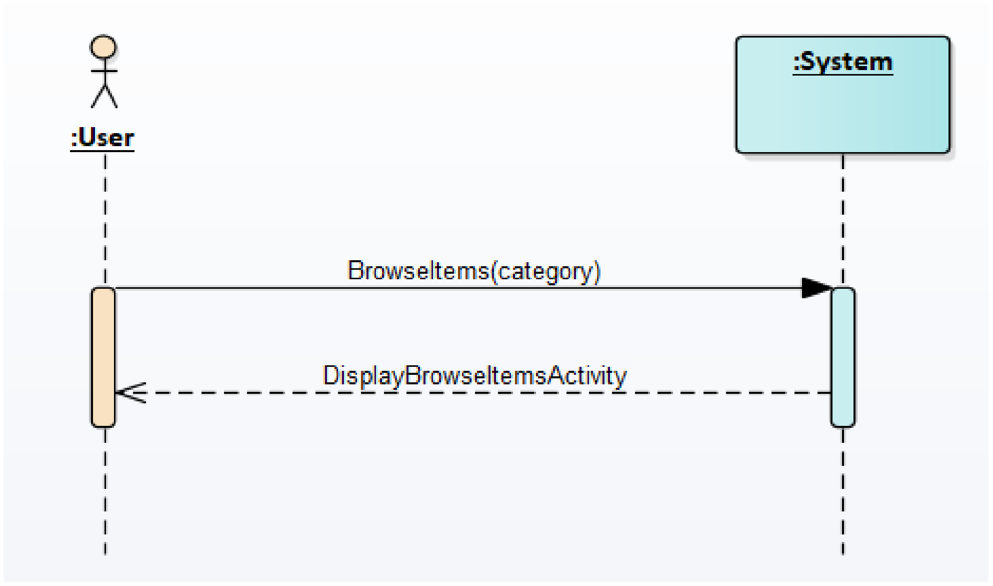


Figure 17: SSD Browse item

SSD Delete Item.

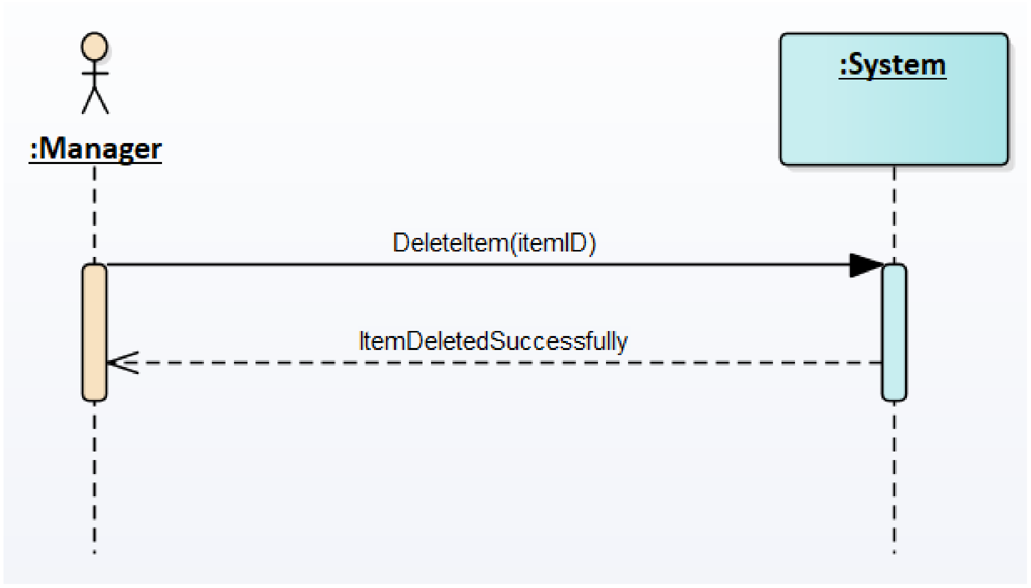


Figure 18: SSD Delete Items

SSD Update Item.

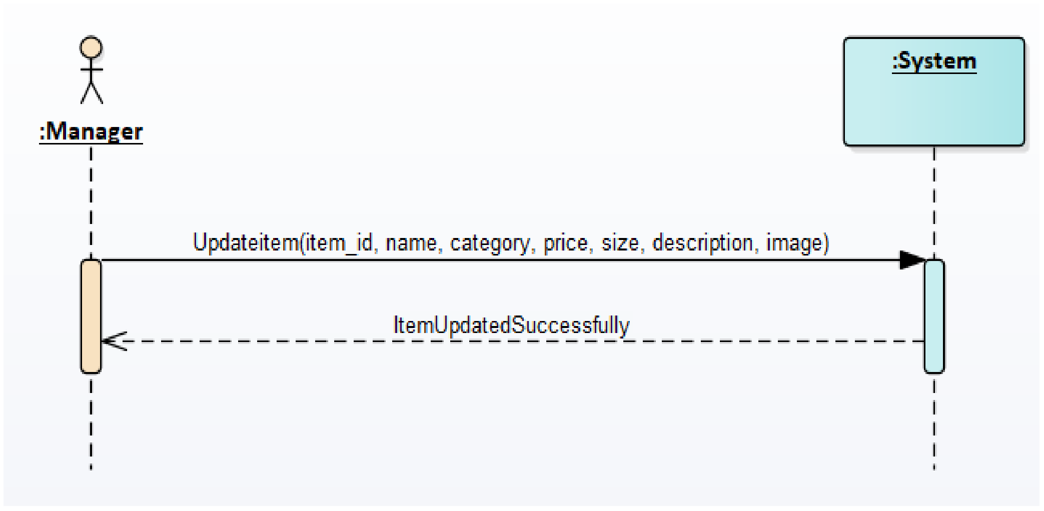

:Manager
:System
Updateitem(item_id, name, category, price, size, description, image)
ItemUpdatedSuccessfully

Figure 19: SSD Update Item

SSD Block Customer Account.

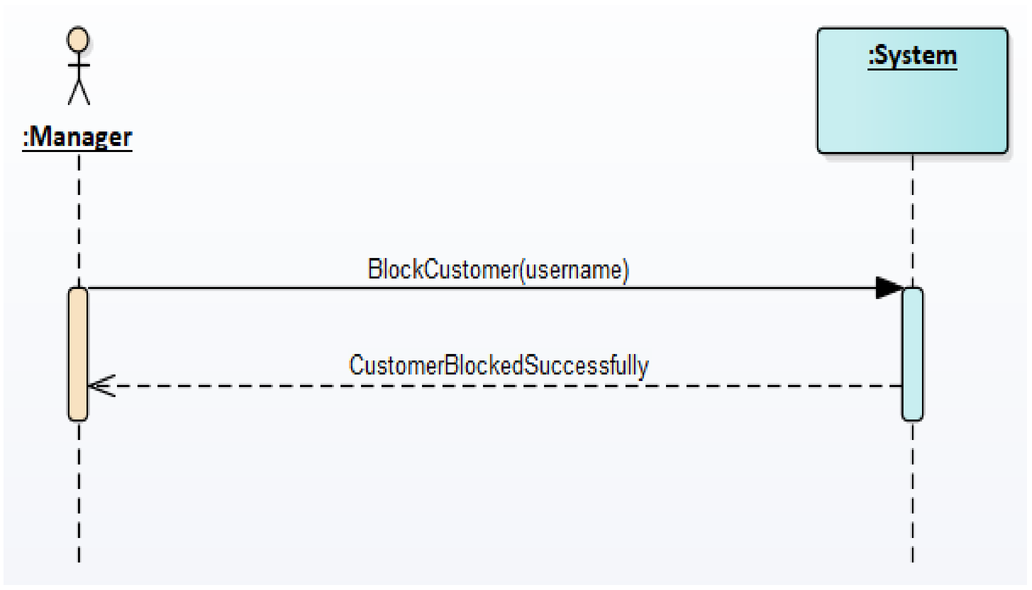


Figure 20: SSD Block Customer

SSD UnBlock Customer Account.

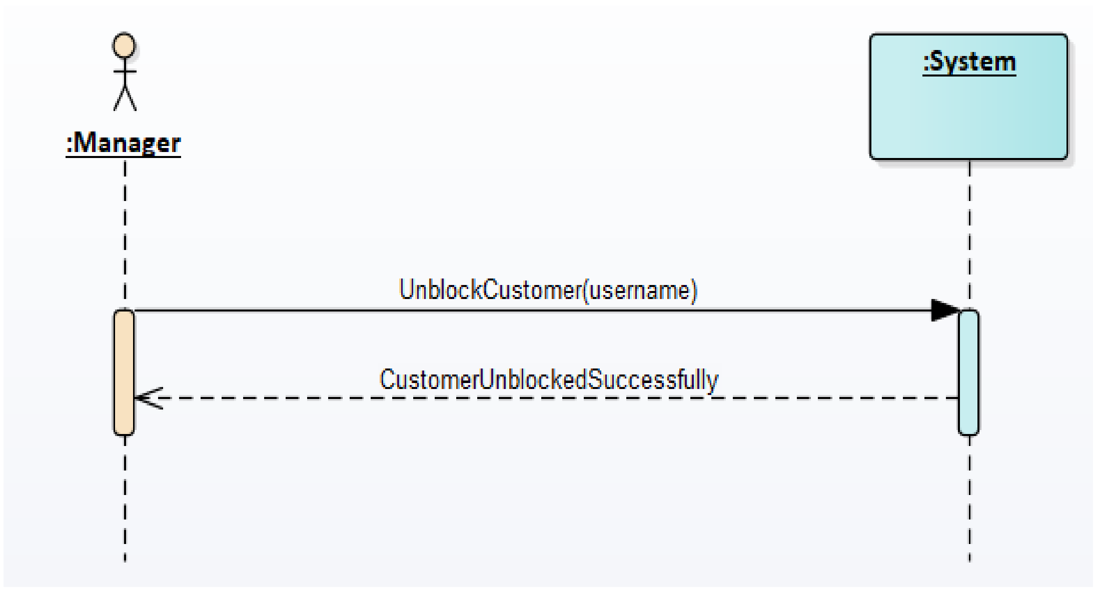


Figure 21: SSD Unblock Customer

SSD AddBiker.

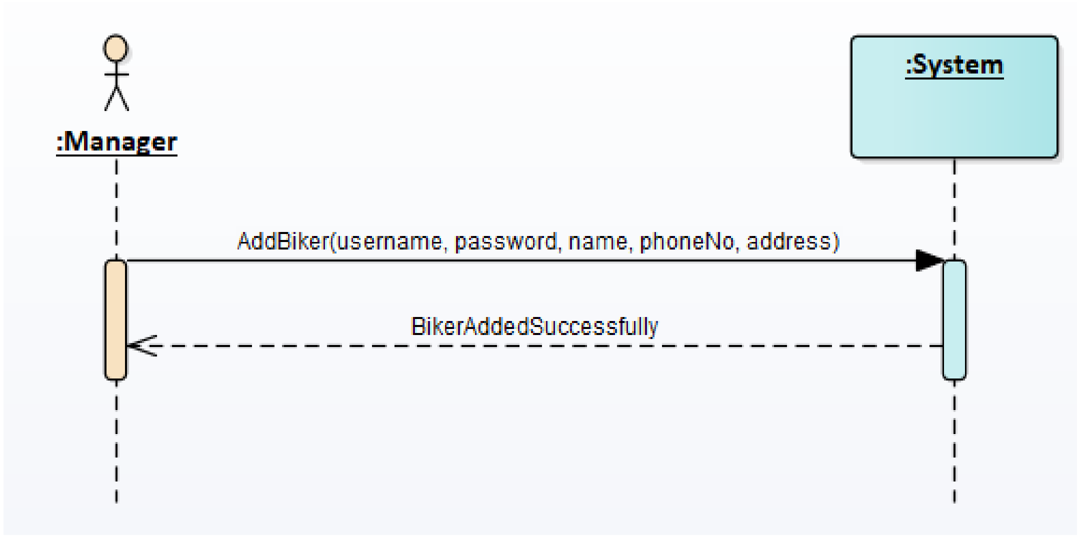


Figure 22: SSD Add Biker

SSD ViewBiker.

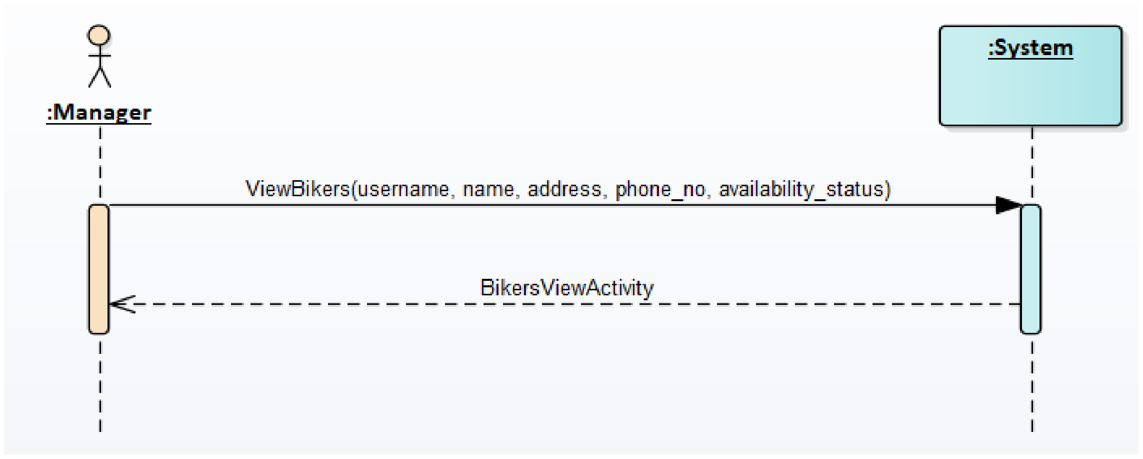

:Manager
:System
ViewBikers(username, name, address, phone_no, availability_status)
BikersViewActivity

Figure 23: SSD View Bikers

SSD DeleteBiker.

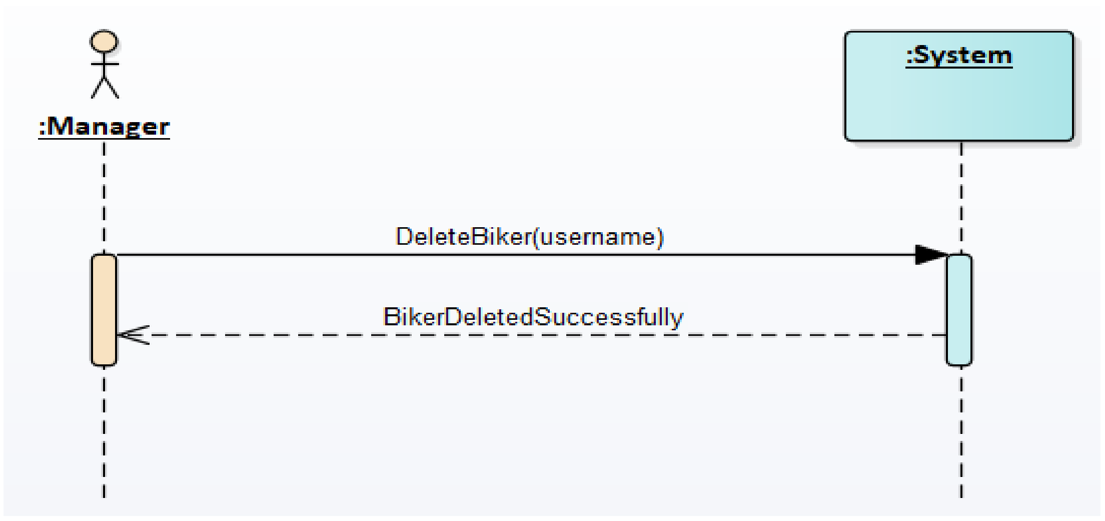


Figure 24: SSD Delete Biker

SSD UpdateBiker.

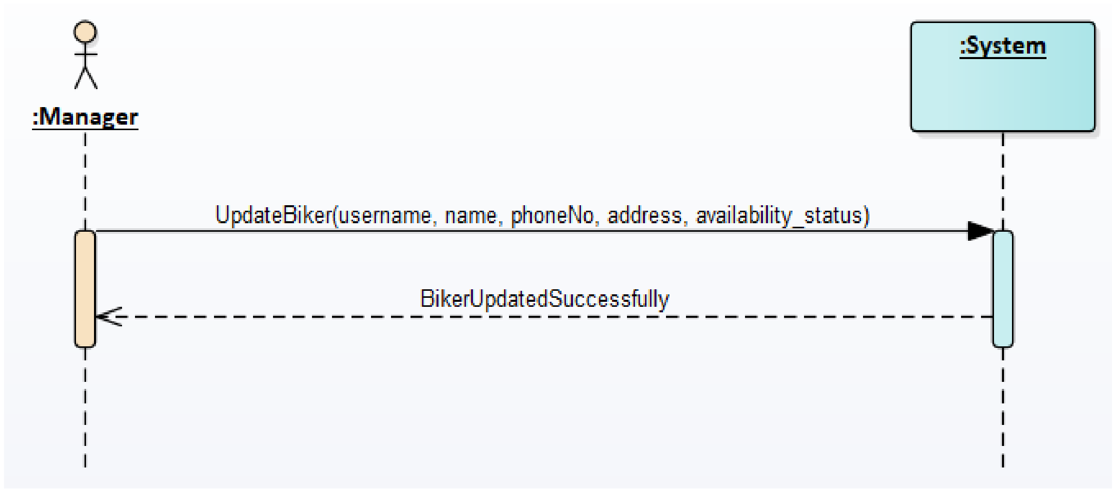

:Manager
:System
UpdateBiker(username, name, phoneNo, address, availability_status)
BikerUpdatedSuccessfully

Figure 25: SSD Update Biker

SSD Accept Order.

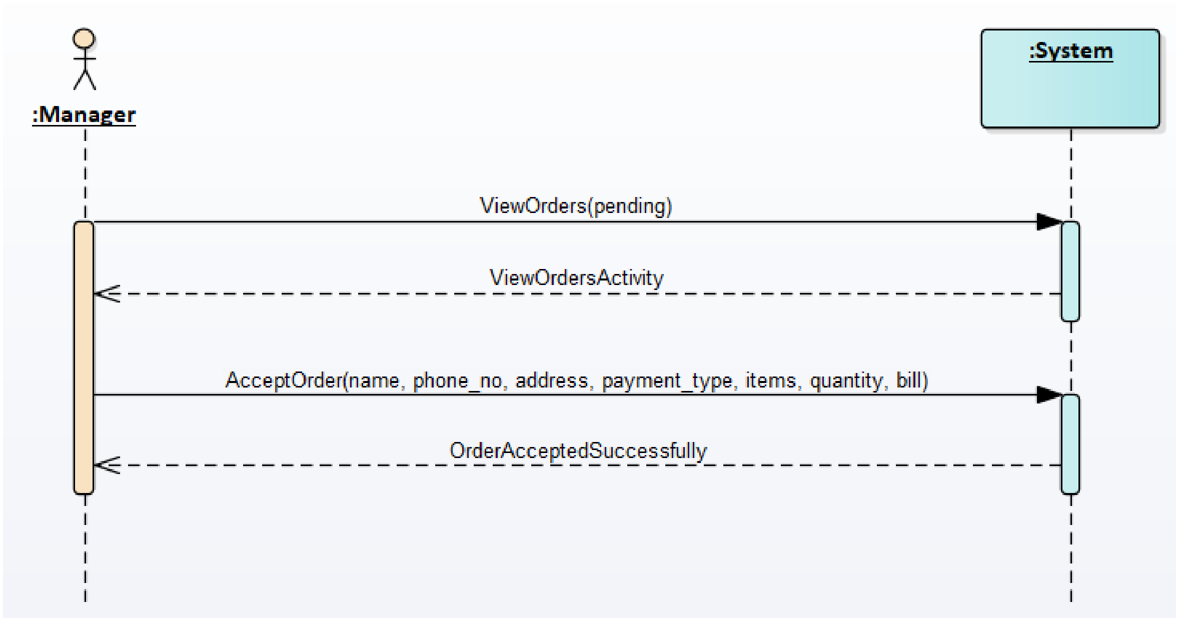

:Manager
:System
ViewOrders(pending)
ViewOrdersActivity
AcceptOrder(name, phone_no, address, payment_type, items, quantity, bill)
OrderAcceptedSuccessfully

Figure 26: SSD Accept Order

SSD Update Order Status.

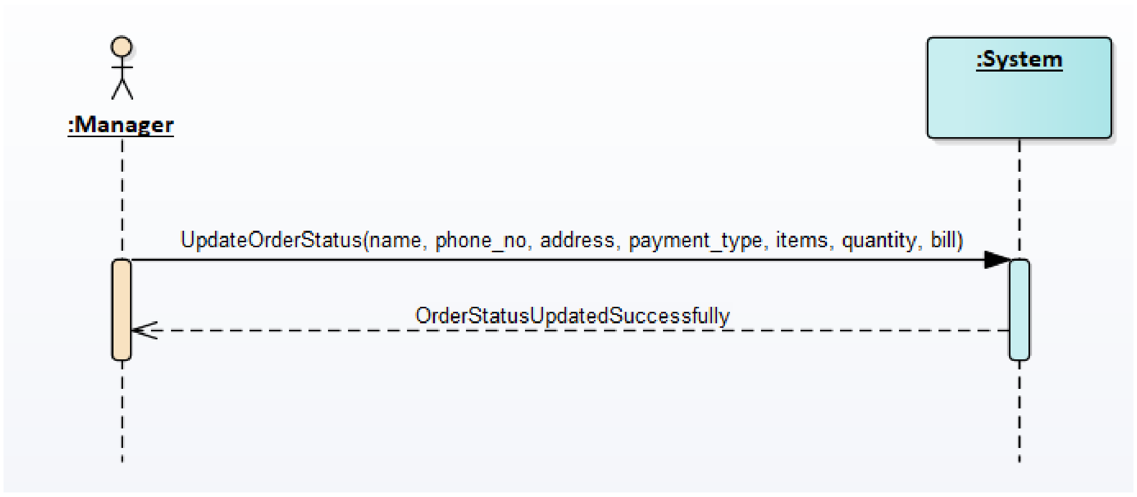

:Manager
:System
UpdateOrderStatus(name, phone_no, address, payment_type, items, quantity, bill)
OrderStatusUpdatedSuccessfully

Figure 27: SSD Update Order Status

SSD Generate Bill.

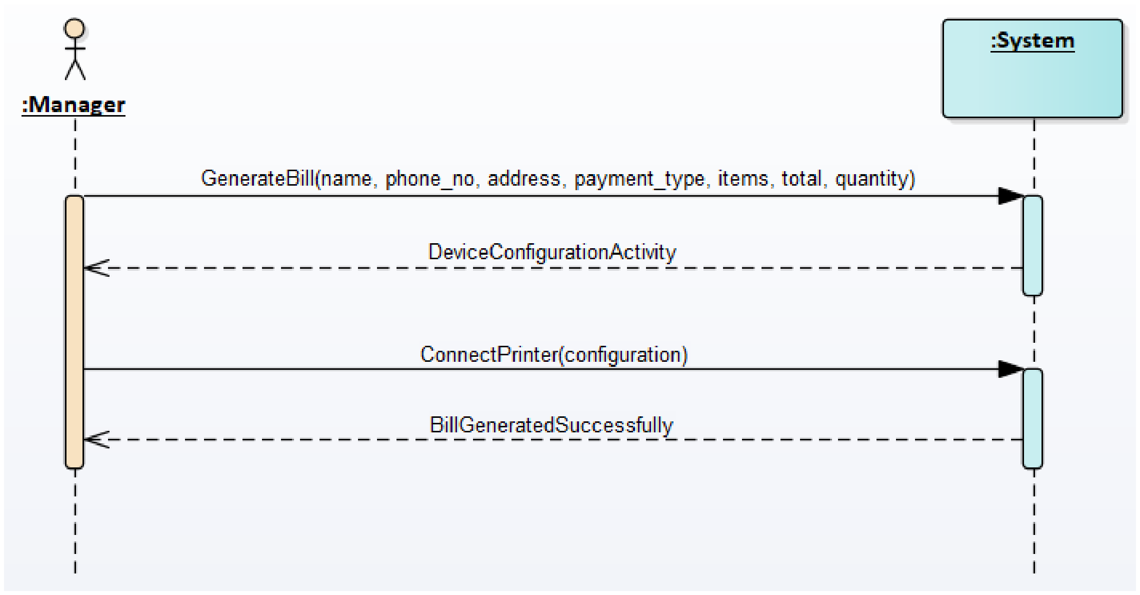

:Manager
:System
GenerateBill(name, phone_no, address, payment_type, items, total, quantity)
DeviceConfigurationActivity
ConnectPrinter(configuration)
BillGeneratedSuccessfully

Figure 28: SSD Generate Bill

SSD Assign Order to Biker.

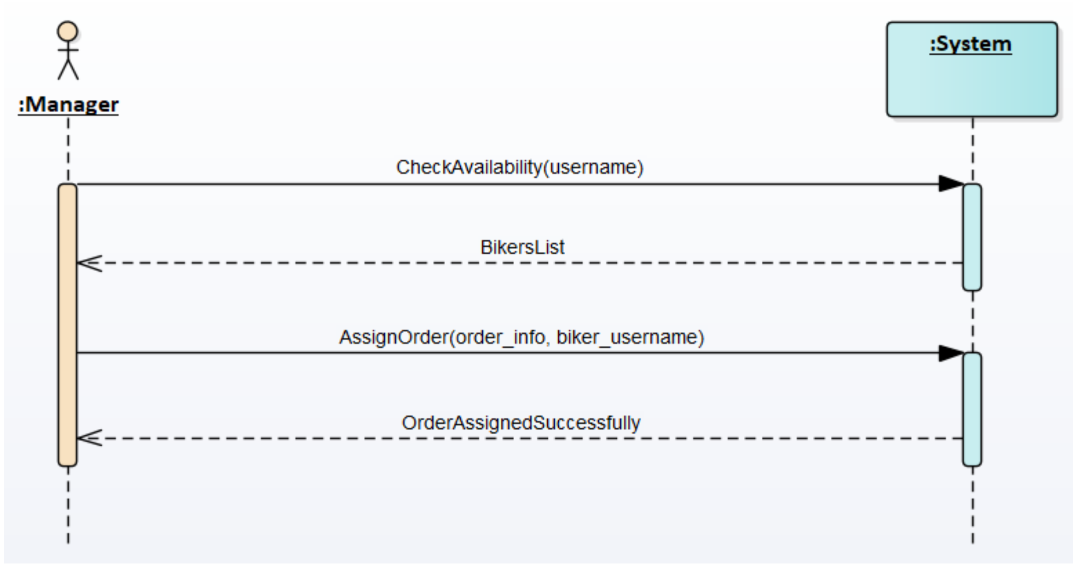

:Manager
:System
CheckAvailability(username)
BikersList
AssignOrder(order_info, biker_username)
OrderAssignedSuccessfully

Figure 29: SSD Assign Order to Biker

SSD View Feedback.

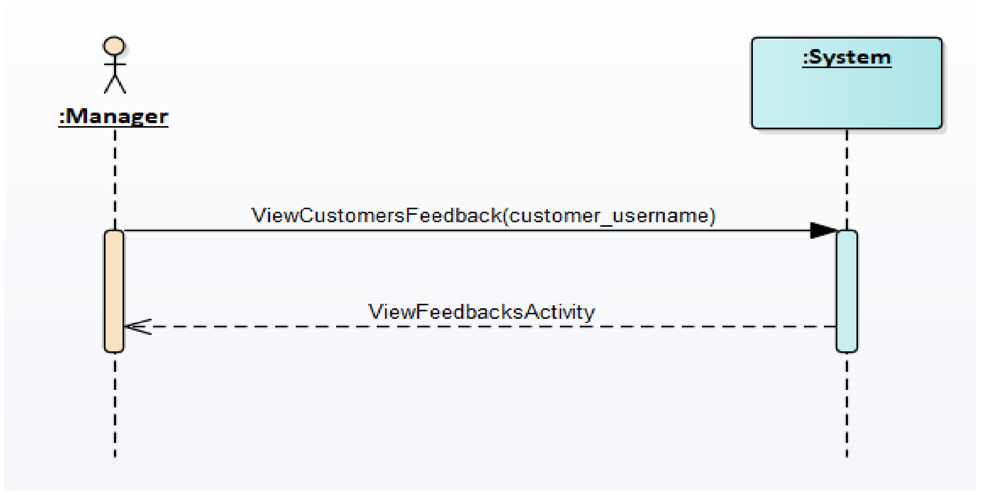


Figure 30: SSD View FeedBack

SSD View OderInfo.

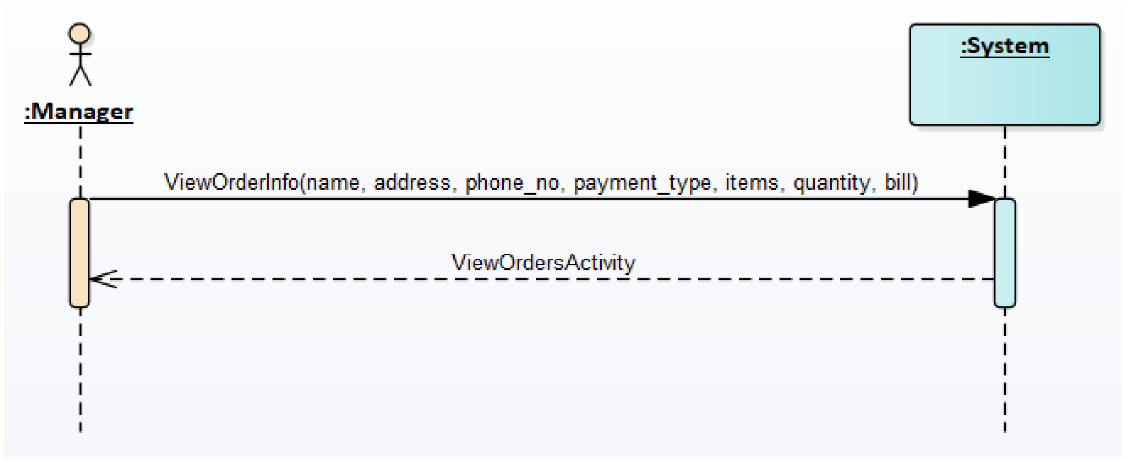

:Manager
:System
ViewOrderInfo(name, address, phone_no, payment_type, items, quantity, bill)
ViewOrdersActivity

Figure 31: SSD view Order Info

SSD Generate Reports.

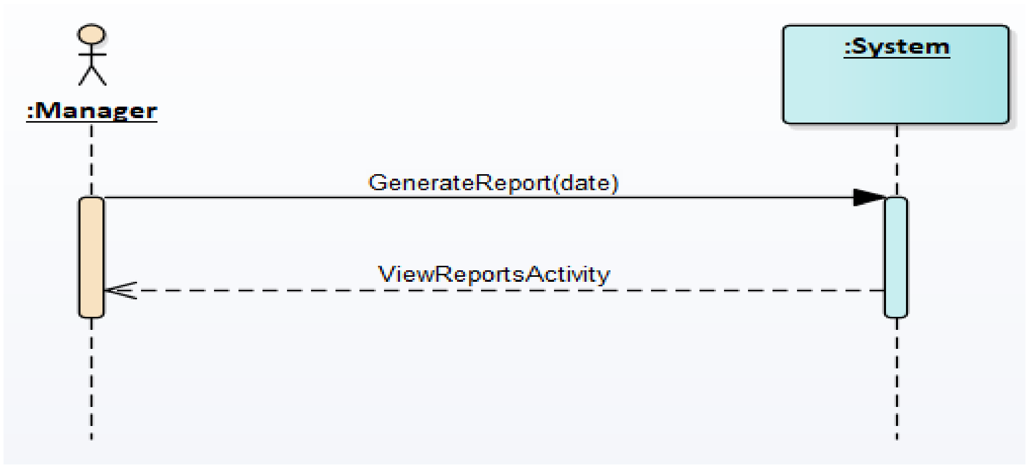


Figure 32: SSD Generate Reports

SSD Add Expense.

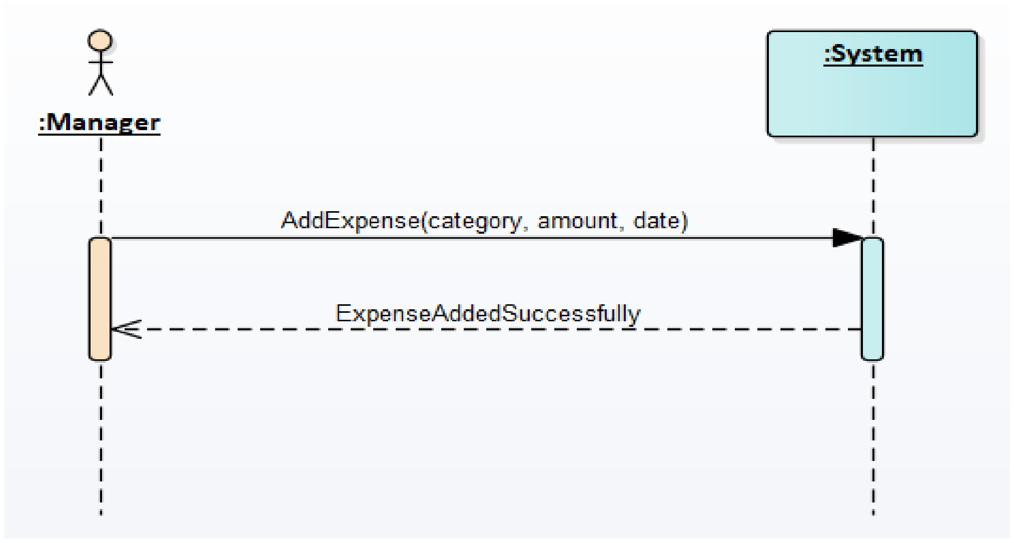


Figure 33: Add Expense

SSD View Expense.

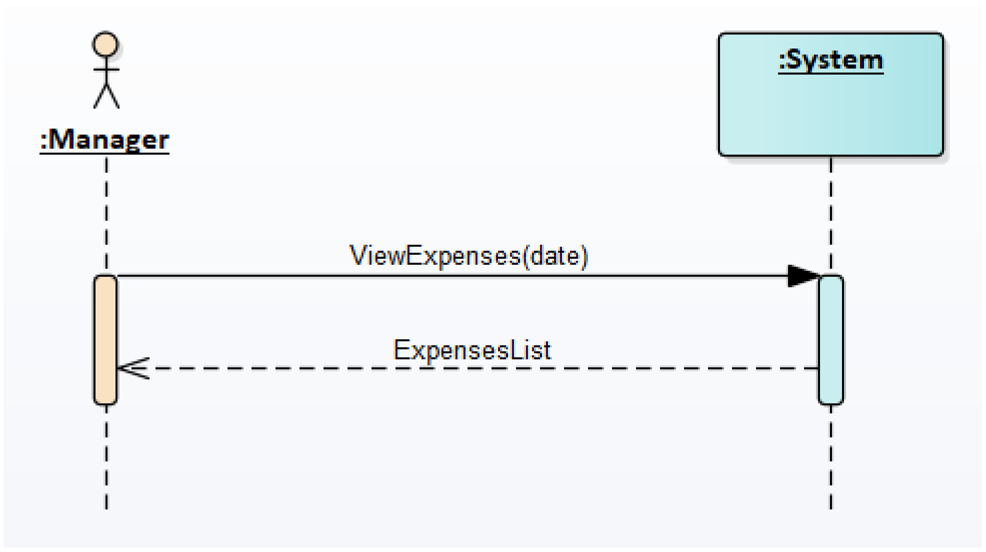


Figure 34: View Expense

SSD Generate Reports PDF.

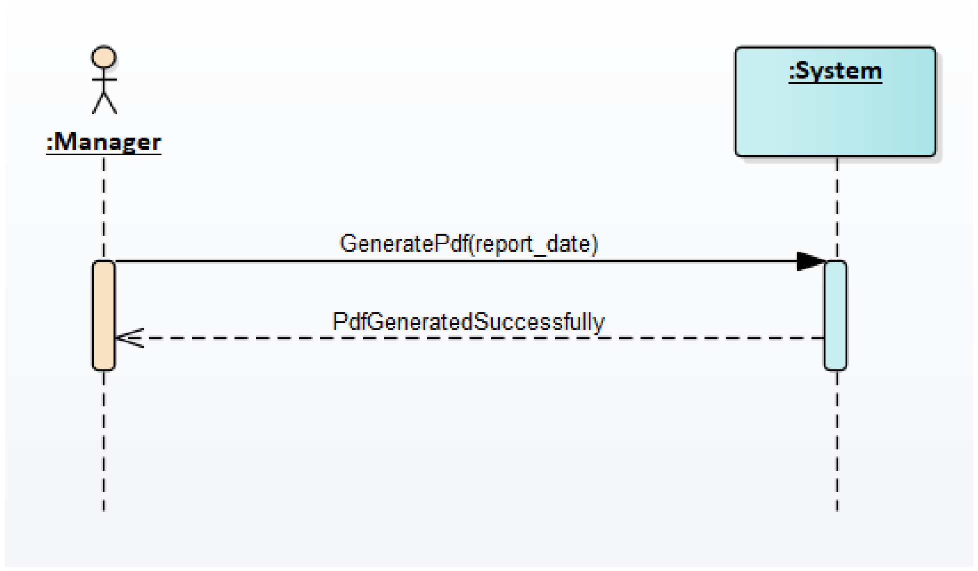


Figure 35: Generate Reports PDF

For Customer:
SSD Signup.

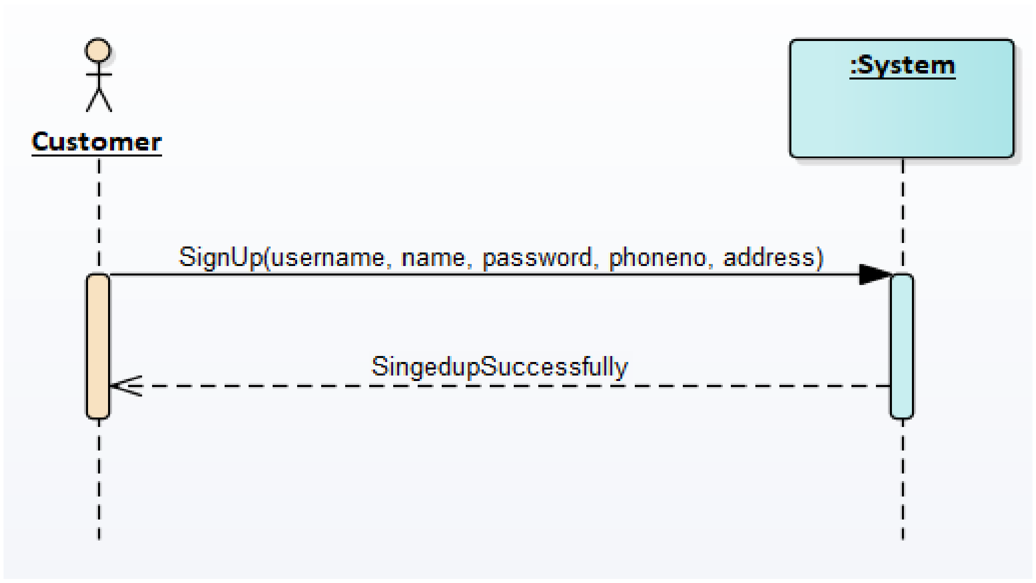


Figure 36: SSD Signup

SSD Browse item.

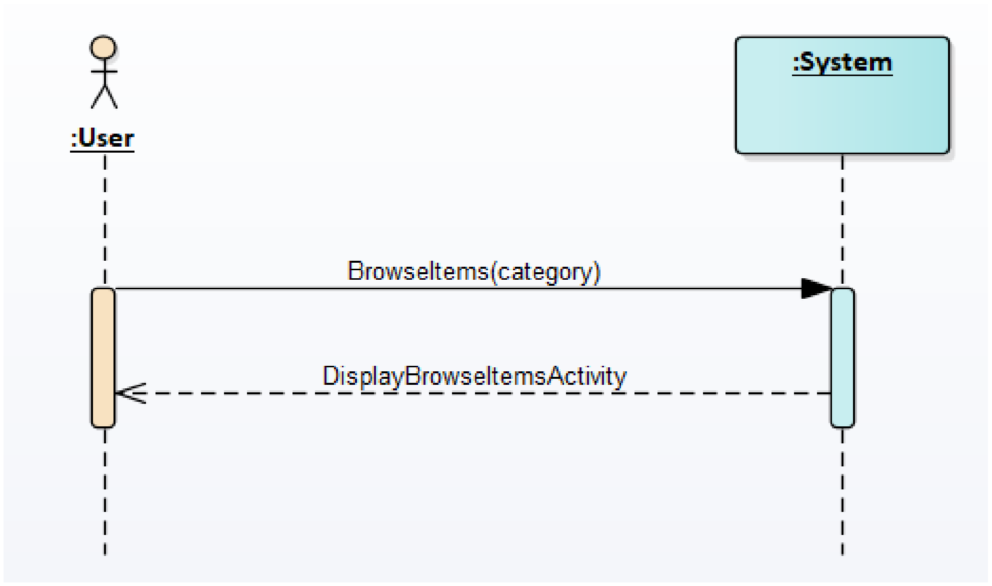


Figure 37: SSD Browse Items

SSD Checkout.

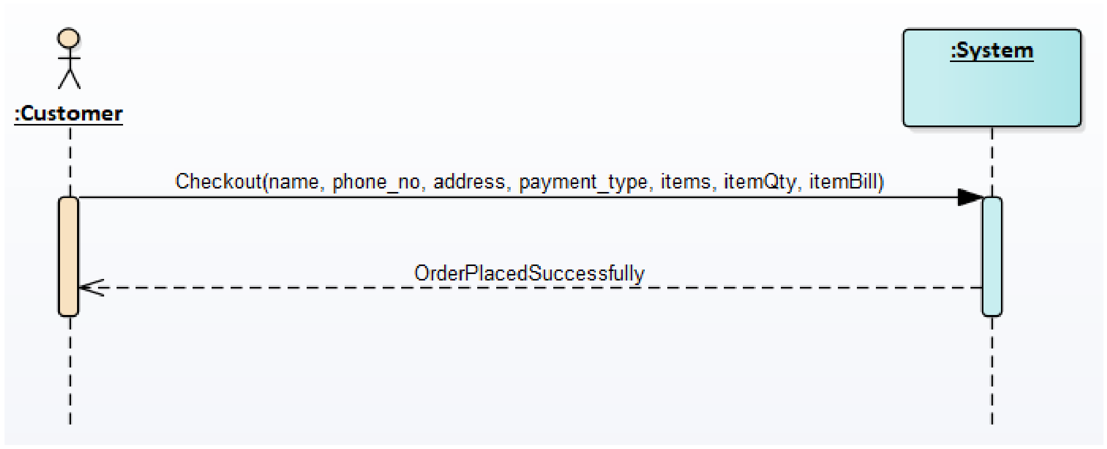

:Customer
:System
Checkout(name, phone_no, address, payment_type, items, itemQty, itemBill)
OrderPlacedSuccessfully

Figure 38: SSD Checkout

SSD Add to Cart.

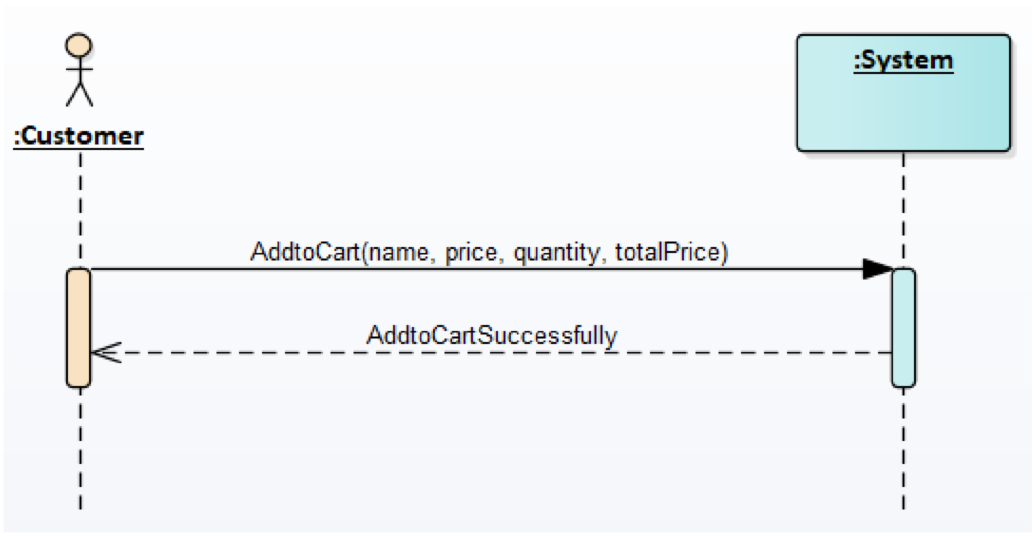


Figure 39 SSD Add to Cart

SSD Delete from Cart.

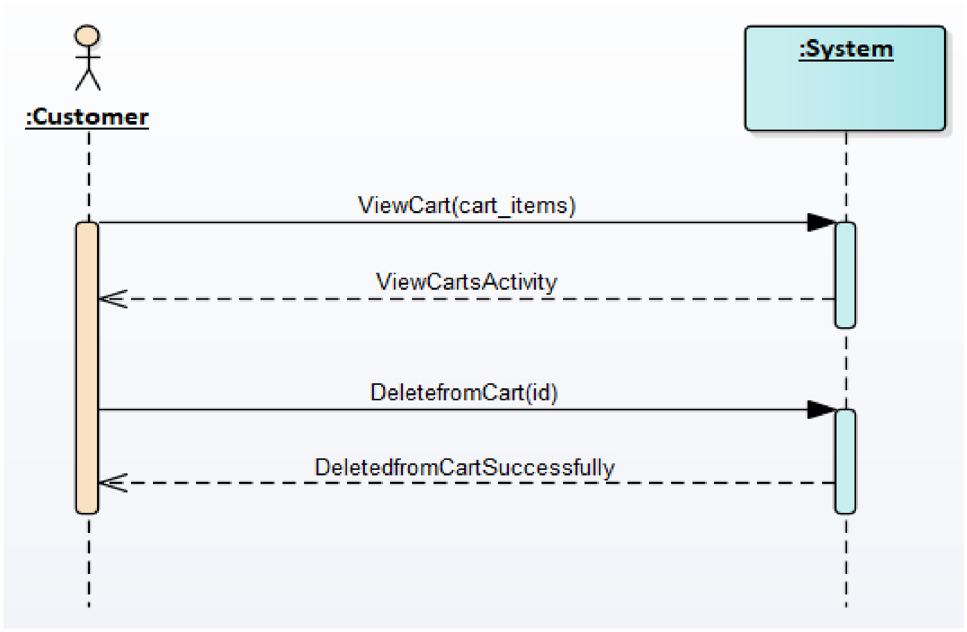


Figure 40: SSD Delete from Cart

SSD Online Payment.

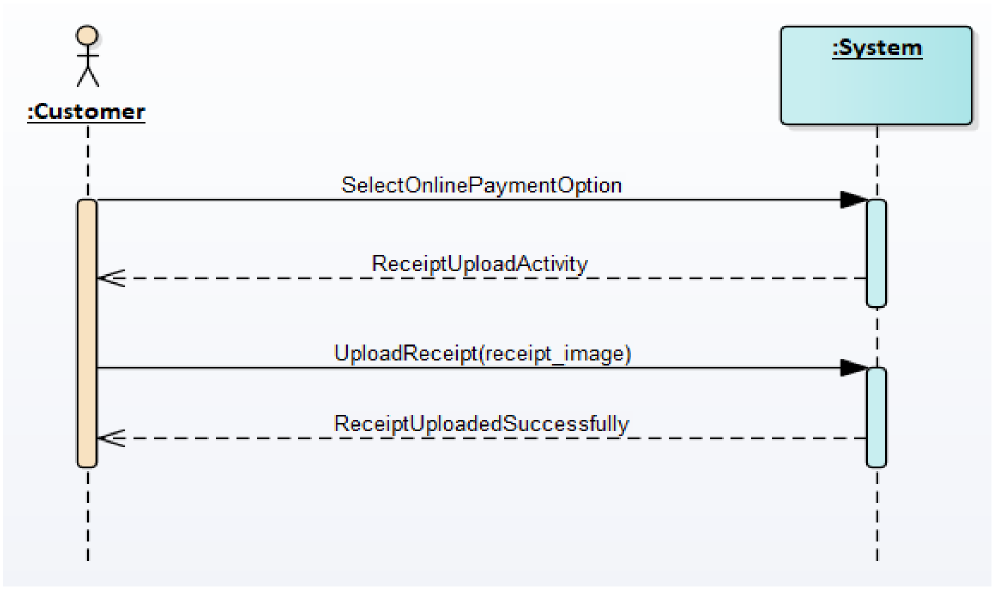

:Customer
:System
SelectOnlinePaymentOption
ReceiptUploadActivity
UploadReceipt(receipt_image)
ReceiptUploadedSuccessfully

Figure 41: SSD Online Payment

SSD Cancel Order.

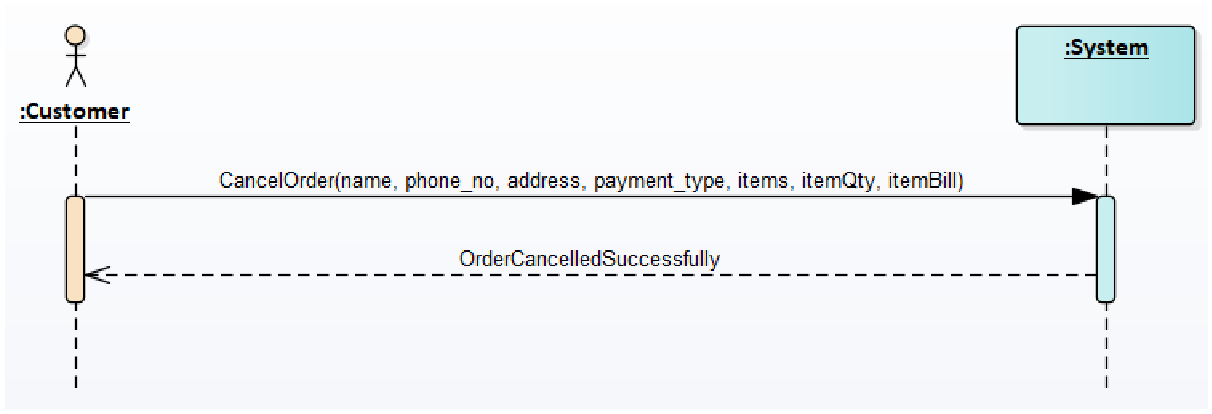

:Customer
:System
CancelOrder(name, phone_no, address, payment_type, items, itemQty, itemBill)
OrderCancelledSuccessfully

Figure 42: SSD Cancel Order

SSD Submit Feedback.

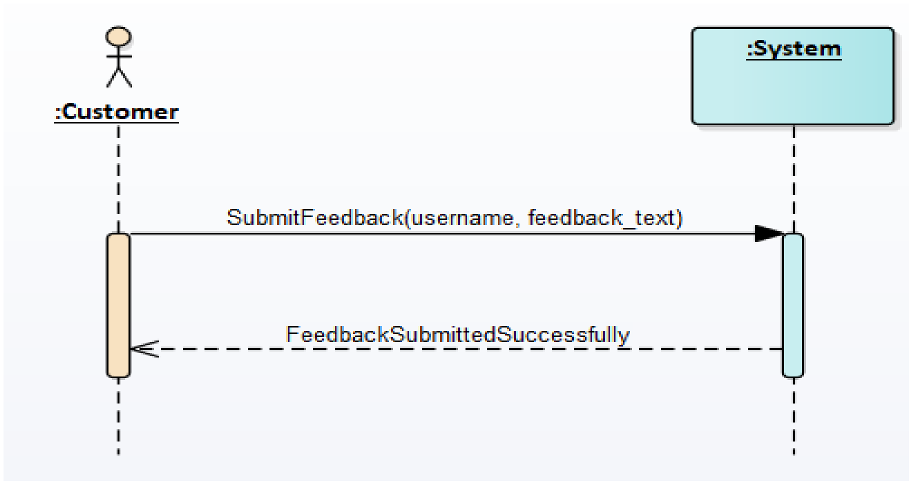

:Customer
:System
SubmitFeedback(username, feedback_text)
FeedbackSubmittedSuccessfully

Figure 43: SSD Submit Feedback

SSD Verify Email.

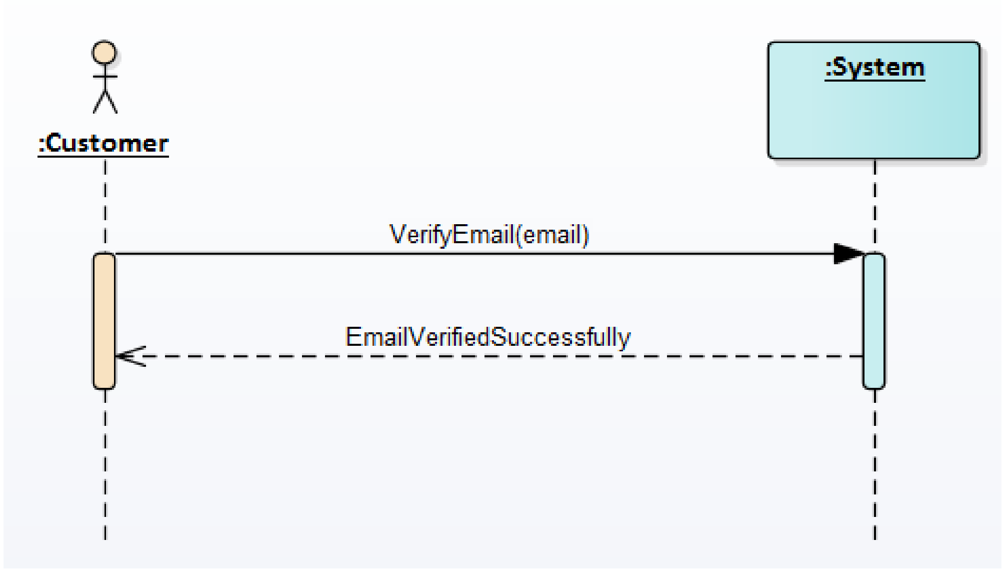


Figure 44: SSD Verify Email

For Biker:

SSD View Order Info.

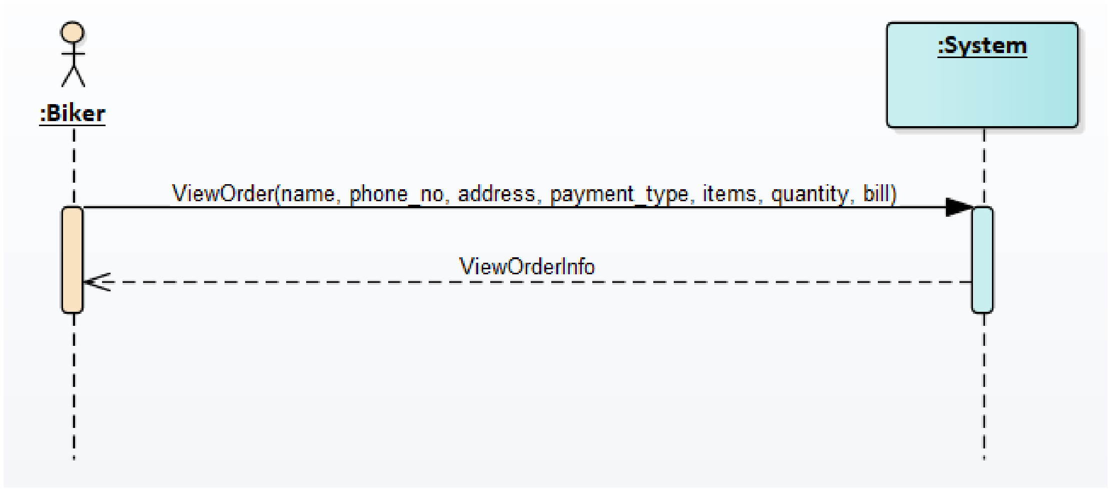

:Biker
:System
ViewOrder(name, phone_no, address, payment_type, items, quantity, bill)
ViewOrderInfo

Figure 45: SSD View Order Info

SSD Confirm Delivery.

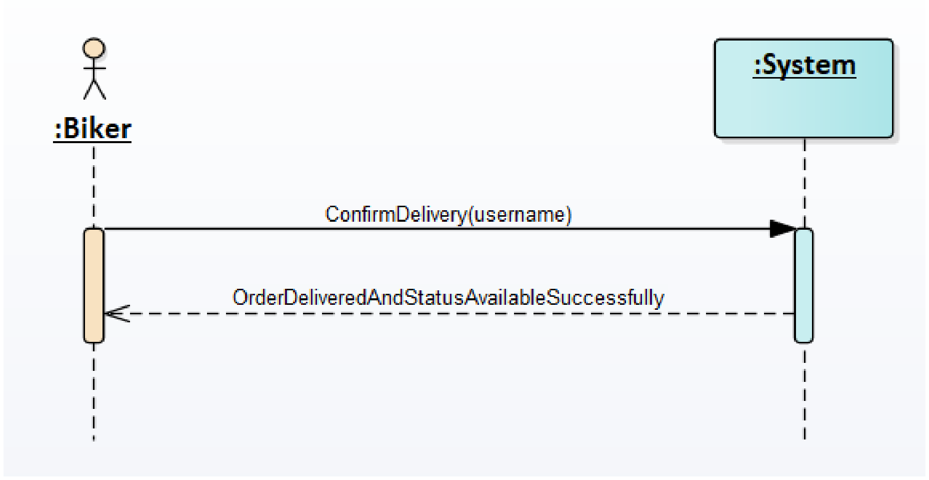

:Biker
:System
ConfirmDelivery(username)
OrderDeliveredAndStatusAvailableSuccessfully

Figure 46: SSD Confirm Delivery

2.6. Domain Model:

Domain model is a conceptual model of the domain that incorporates both behavior and data.

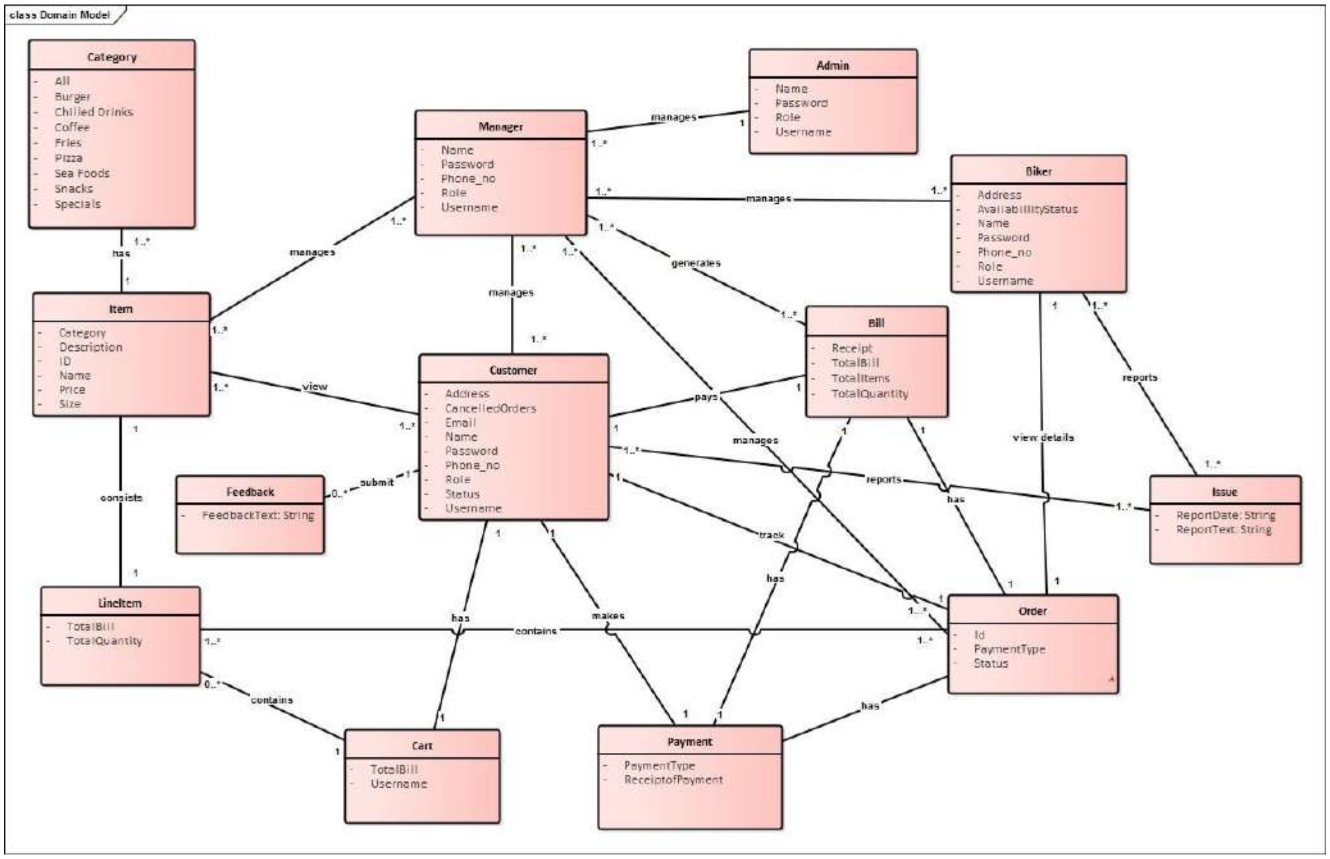


Figure 47: Domain Model

Chapter 3

System Design

3.1. Layer Definition:

3.1.1. Presentation Layer:

This layer defines how the graphical user interface interact with the business layer and the database layer. The main function of this layer is to translate tasks and results something the user can understand.

3.1.2. Business Layer:

This layer controls the system functionality by performing different processing or business rules related to the system. It also moves the data between the surrounding two layers. The main components of this layer are business rules and workflow of the system.

3.1.2. Database Layer:

This layer is used to store the any information in the database which is used in current system. This layer has own work process which handles the tasks related to the database. This layer for the permanent data storage of data.

3.2. Class Diagram:

Class diagrams describe systems by illustrating attributes, operations and relationships between classes. Unified Modeling Language (UML) calls them structure diagrams.

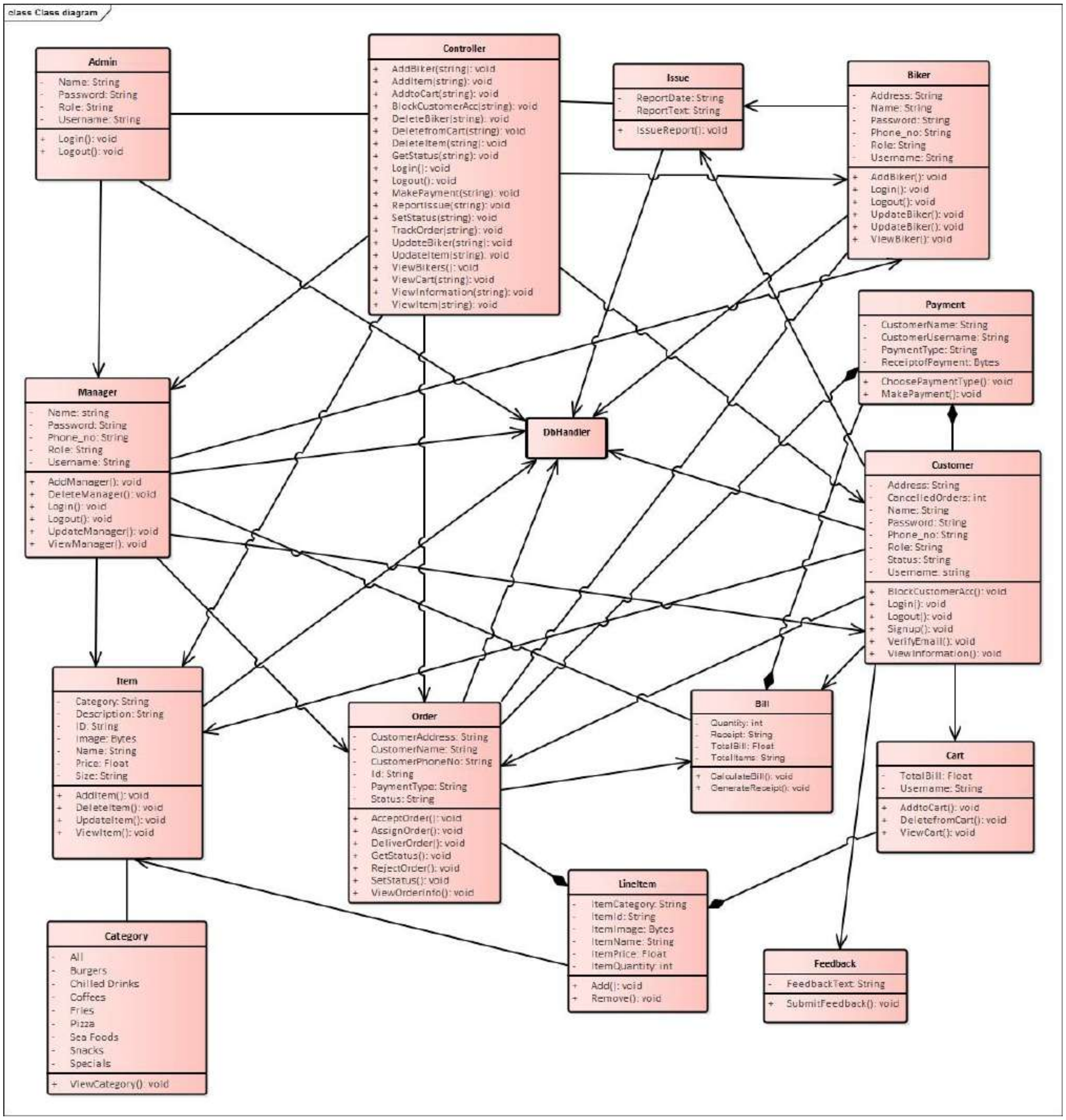


Figure 48: Class Diagram

3.2.1. Controller Class:

This class shall control the communication between classes. It gets request from UI and forward it to appropriate class.

3.2.2. DbHandler Class:

This class shall be used to handle all communication with the database. Like Creating, deleting or updating some data.

3.2.3. Manager Class:

This class shall be able to control the items, bikers and orders etc. i-e; it can manage items or bikers etc.

3.2.4. Customer Class:

This class shall be able to entertain the customer user of system. Through this class, customer can place their food item or browse items.

3.2.5. Admin Class:

This class shall be able to entertain the admin user of system. Through this class, admin can make new managers account etc.

3.2.6. Biker Class:

This class shall be able to entertain the biker user of system. Through this class, biker can view order info and other operations.

3.2.7. Cart Class:

This class shall be able to entertain the cart of customer through this class, customer can view, add and delete item from their order.

3.2.8. Items Class:

This class shall be able to entertain the food items of system. Through this class, manager can add, delete, update or view items of system.

3.2.9. Feedback Class:

This class shall be able to entertain the feedback upon the food items of order. Through this class, cutomer can submit feedback of both food and biker services.

3.2.10. Issue Class:

This class shall be able to entertain the issues that were faced by the system users. Through this class, user can report any kind of issue faced in using the system.

3.2.11. Payment Class:

This class shall be able to entertain the payments done by the customers for the orders. Through this class, customer can pay their order payment.

3.2.12. Lineitem Class:

This class shall be able to represent the single item in the cart of a customer. Through this class, customer shall be able to add new item to their cart.

3.3. Sequence Diagrams:

Given below are the Sequence Diagrams of Sip n Snack v.2.0.

SD Login.

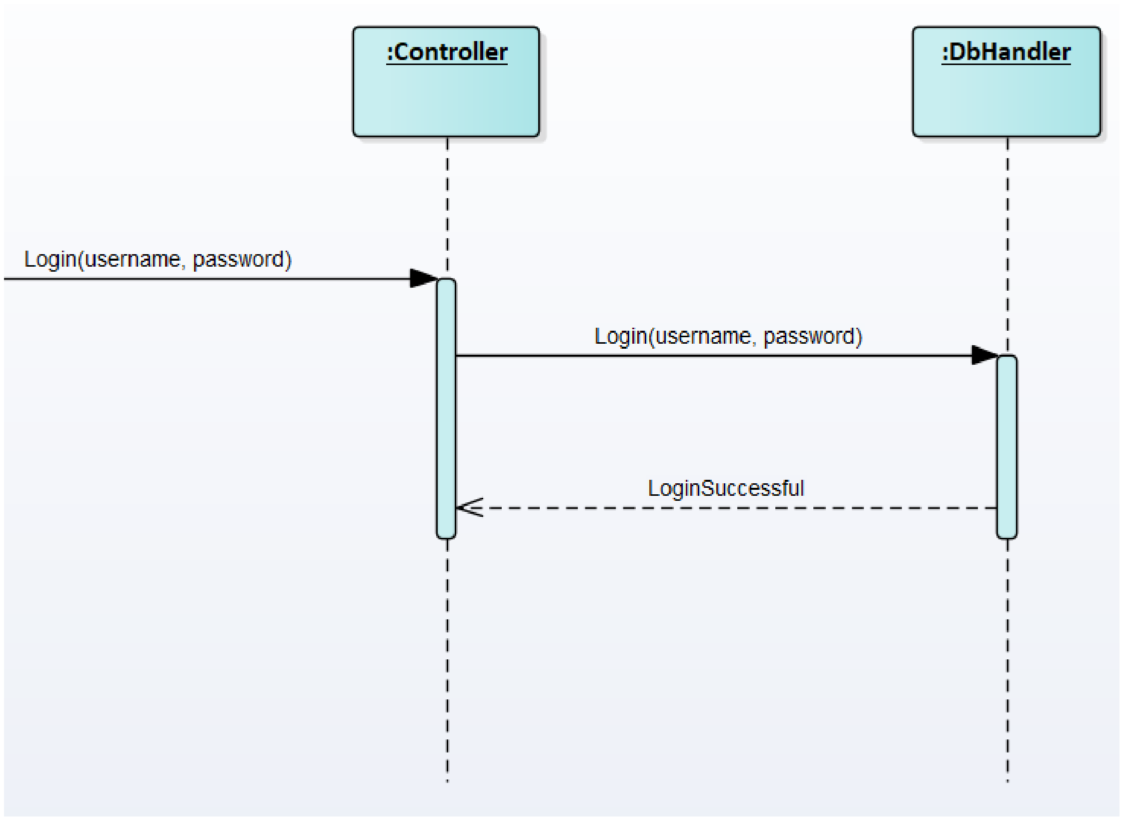


Figure 49: SD Login

SD Manage Profile.

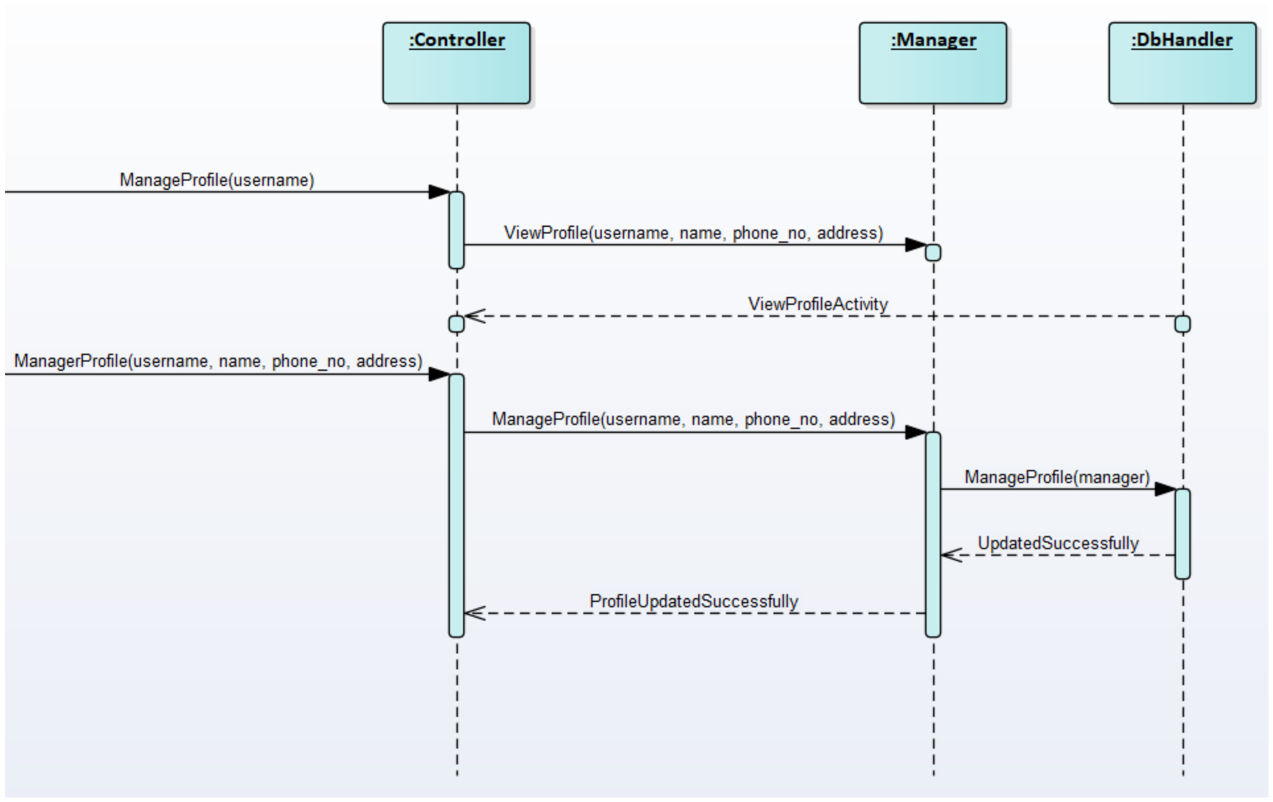

:Controller
:Manager
:DbHandler
ManageProfile(username)
ViewProfile(username, name, phone_no, address)
ViewProfileActivity
ManagerProfile(username, name, phone_no, address)
ManageProfile(username, name, phone_no, address)
ManageProfile(manager)
UpdatedSuccessfully
ProfileUpdatedSuccessfully

Figure 50: SD Manage Profile

SD Change Password:.

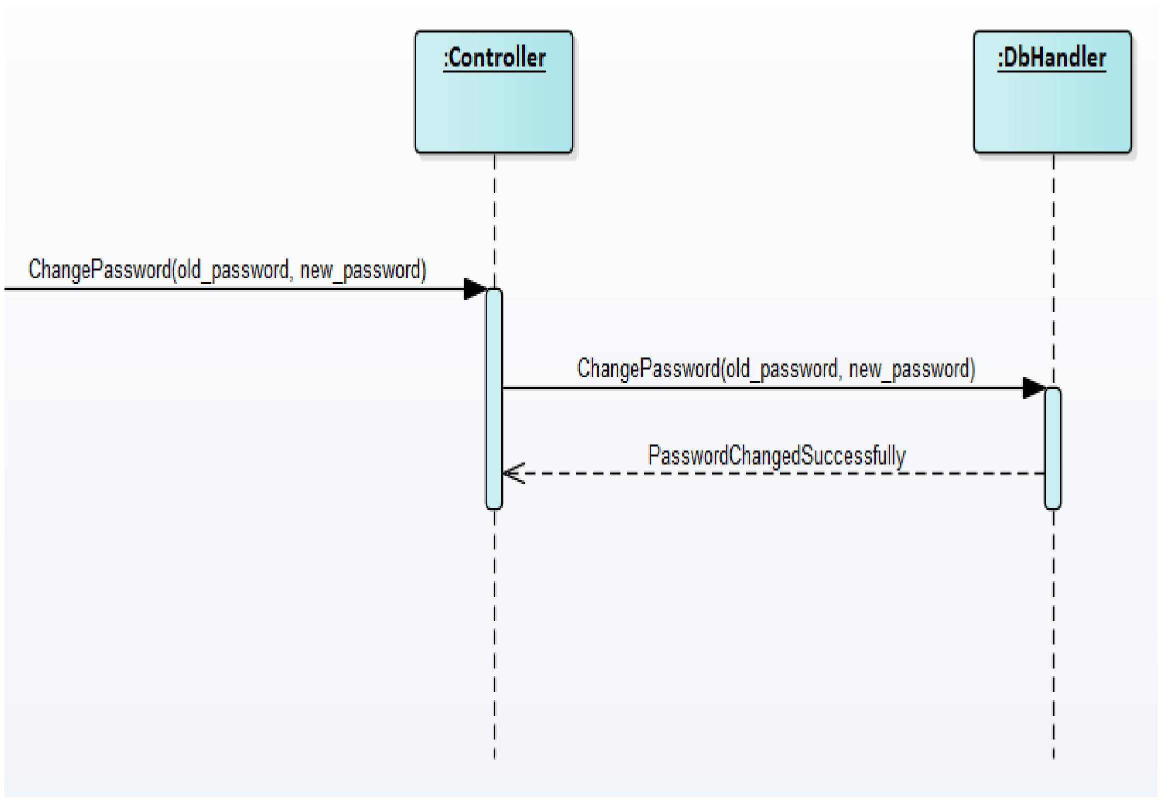

:Controller
:DbHandler
ChangePassword(old_password, new_password)
ChangePassword(old_password, new_password)
PasswordChangedSuccessfully

Figure 51: SD Change Password

SD Report Issue.

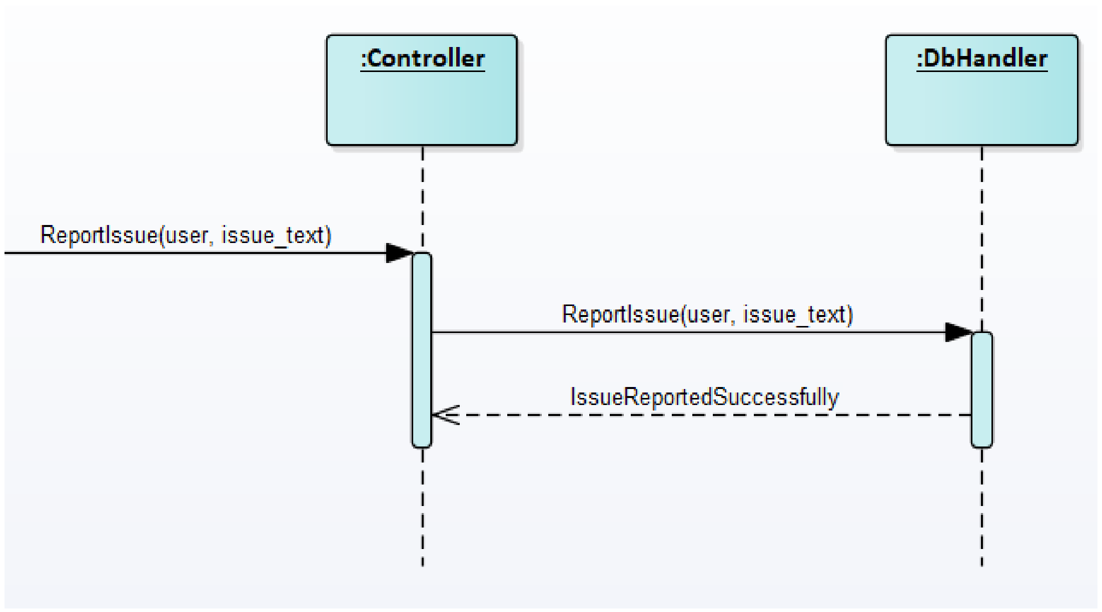

:Controller
:DbHandler
ReportIssue(user, issue_text)
ReportIssue(user, issue_text)
IssueReportedSuccessfully

Figure 52: SD Report Issue

Admin Module

SD Add Manager.

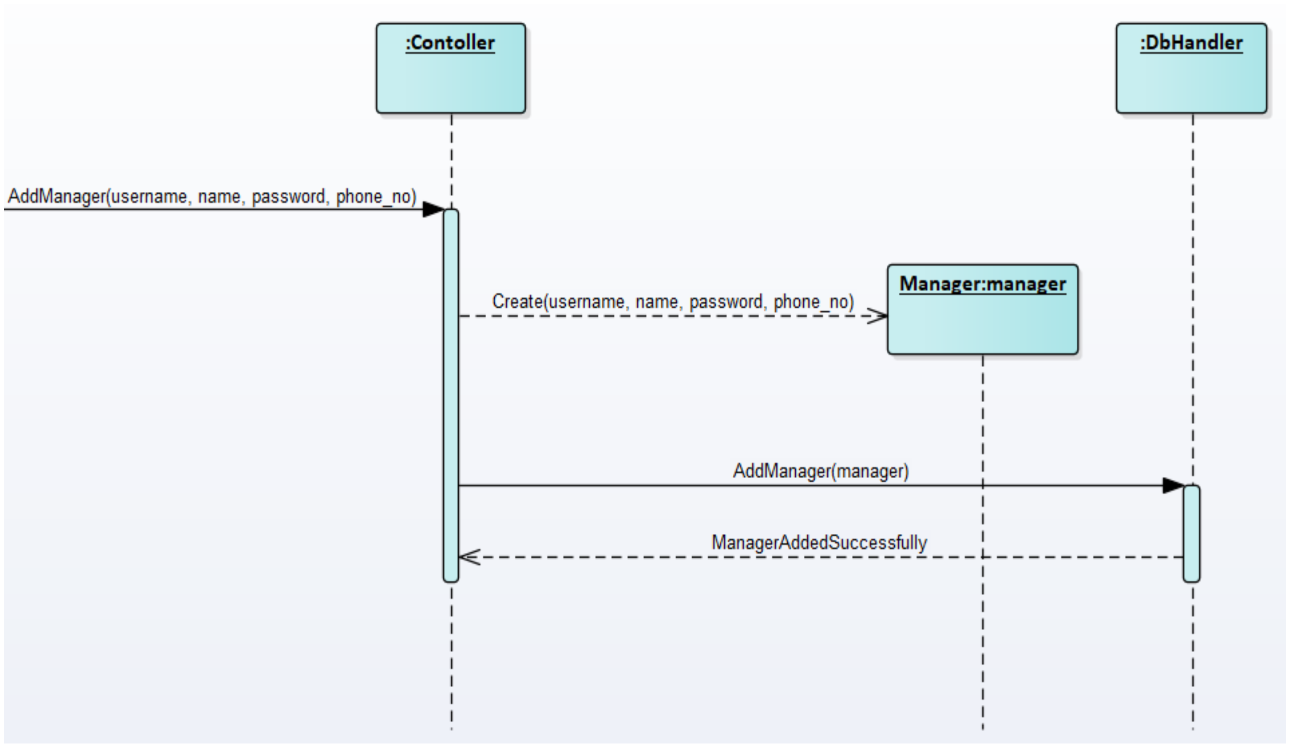

:Contoller
:DbHandler
AddManager(username, name, password, phone_no)
Manager:manager
Create(username, name, password, phone_no)
AddManager(manager)
ManagerAddedSuccessfully

Figure 53: SD Add Manager

SD View Manager.

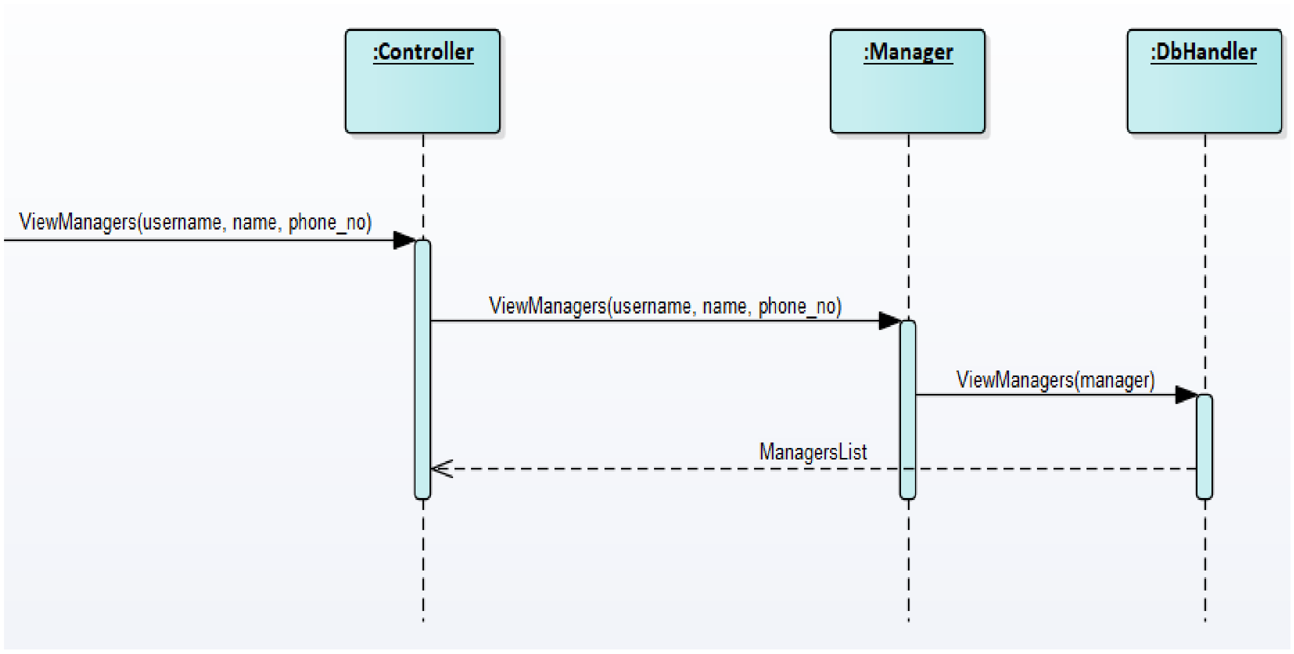

:Controller
:Manager
:DbHandler
ViewManagers(username, name, phone_no)
ViewManagers(username, name, phone_no)
ViewManagers(manager)
ManagersList

Figure 54: SD View Manger

SD Delete Manager.

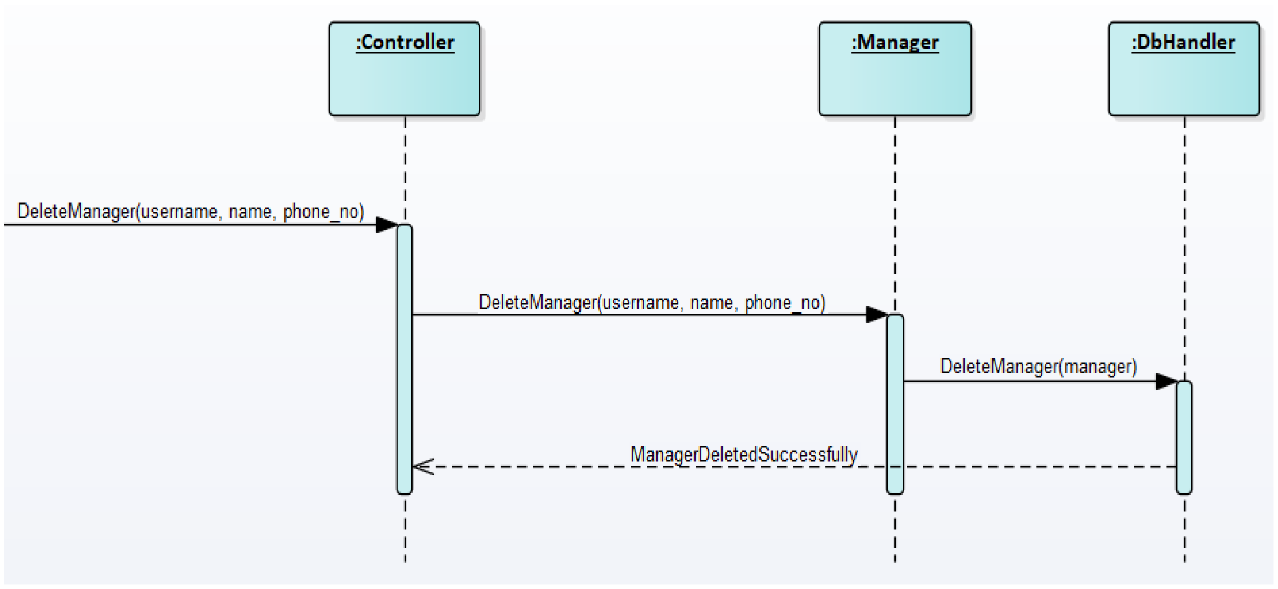

:Controller
:Manager
:DbHandler
DeleteManager(username, name, phone_no)
DeleteManager(username, name, phone_no)
DeleteManager(manager)
ManagerDeletedSuccessfully

Figure 55: SD Delete Manager

Manager Module

SD Add Item.

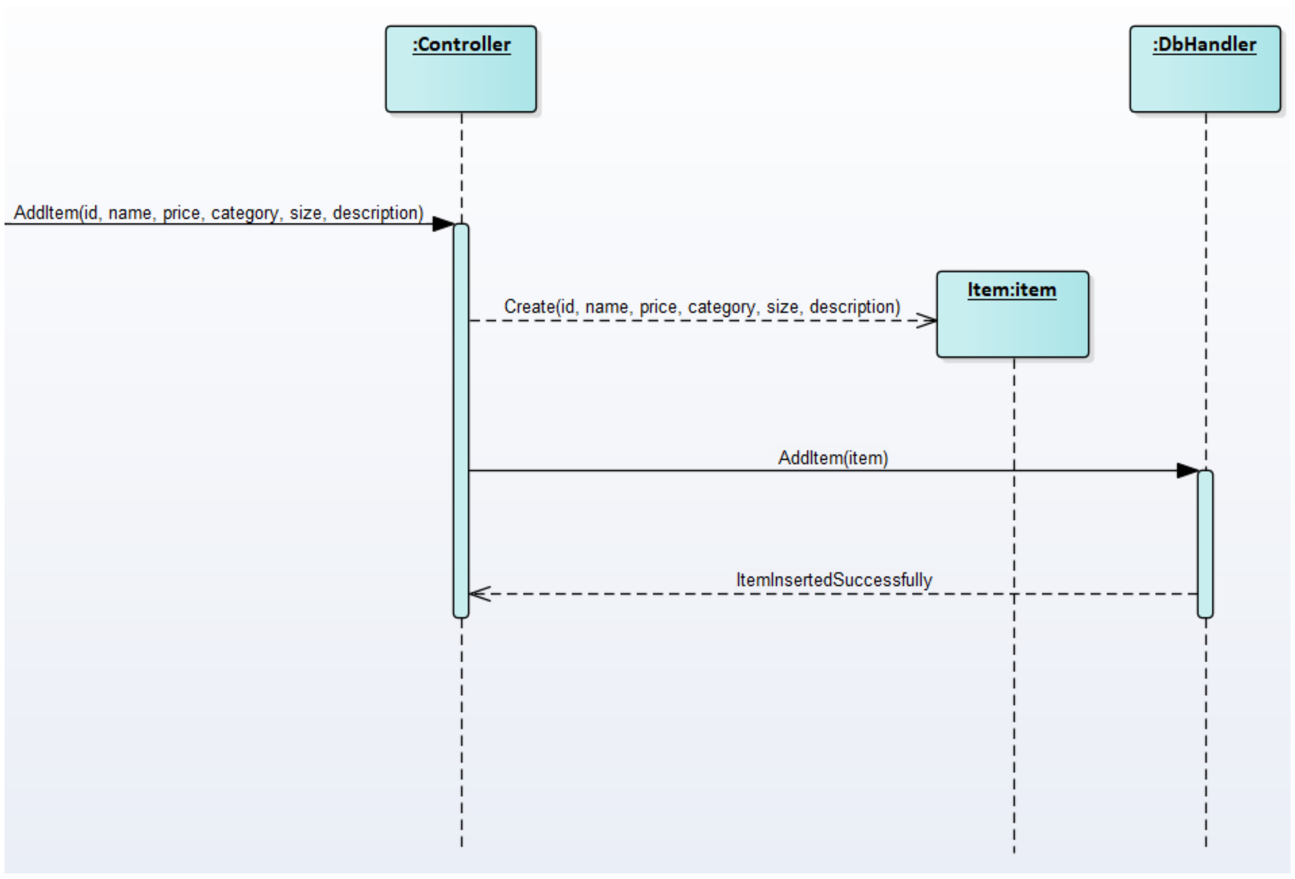

:Controller
:DbHandler
AddItem(id, name, price, category, size, description)
Item:item
Create(id, name, price, category, size, description)
AddItem(item)
ItemInsertedSuccessfully

Figure 56: SD Add Item

SD View Item.

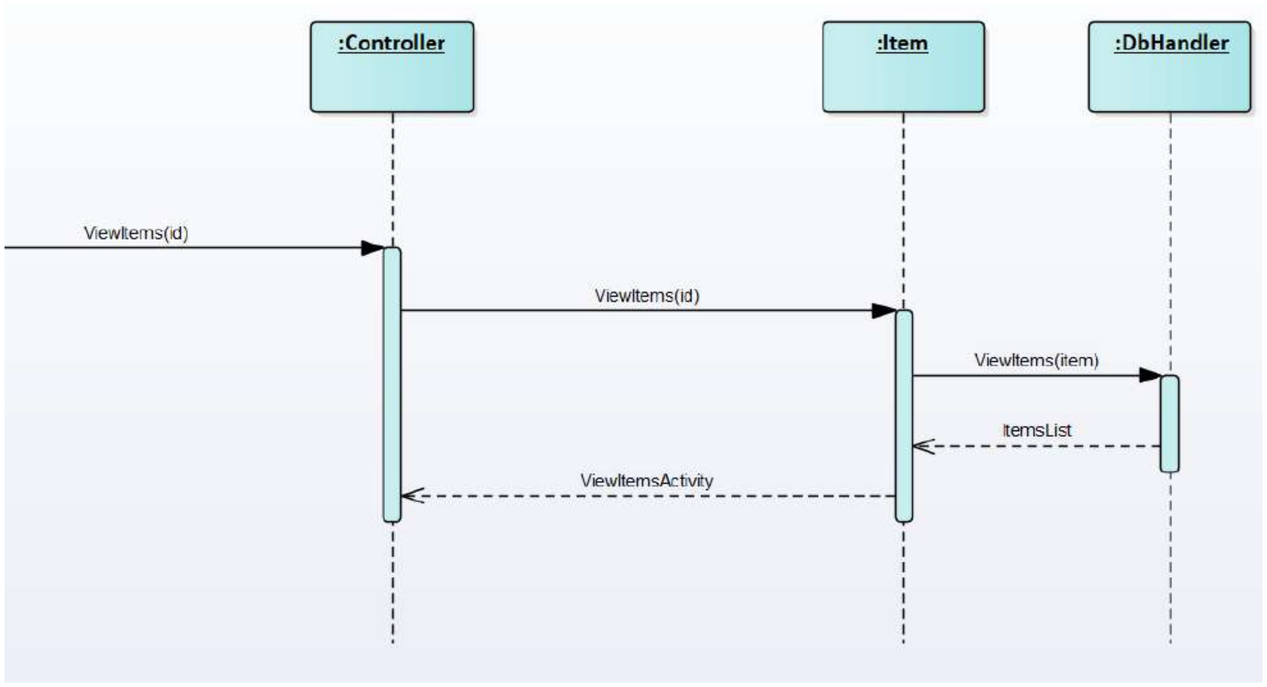


Figure 57: SD View Item

SD Browse Item.

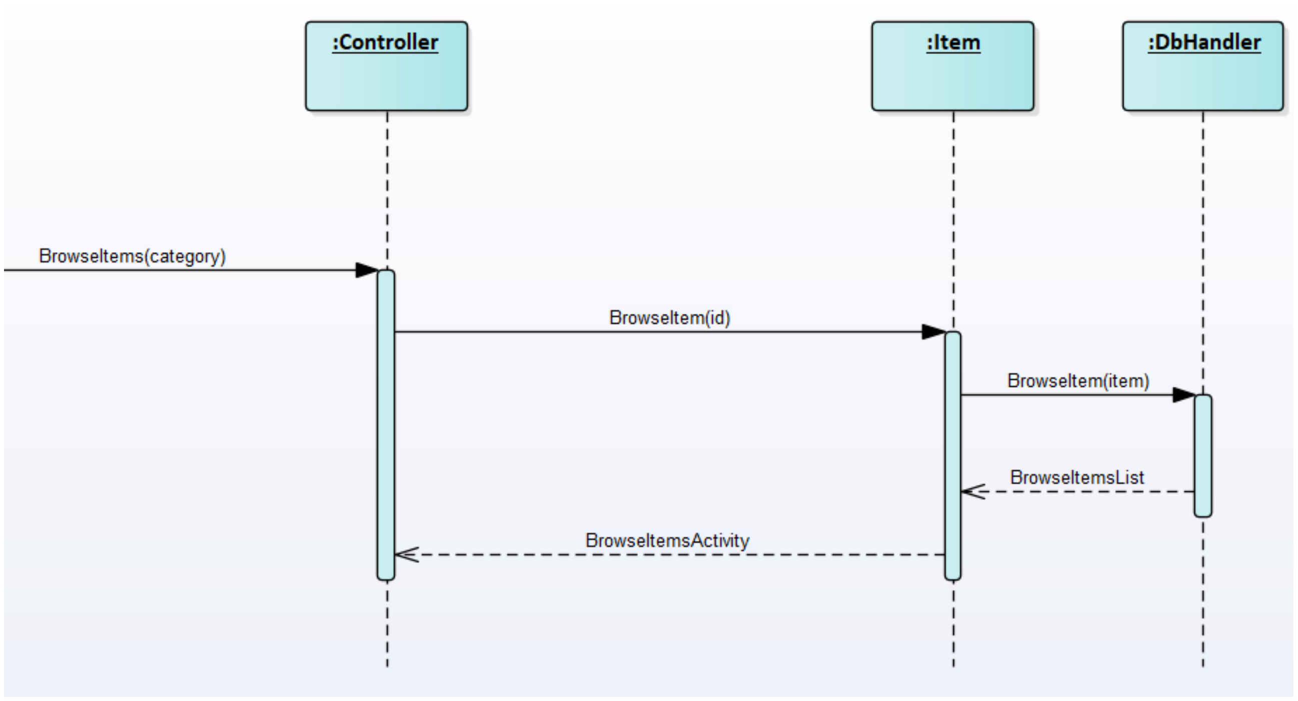


Figure 58: SD Browse Item

SD Update Item.

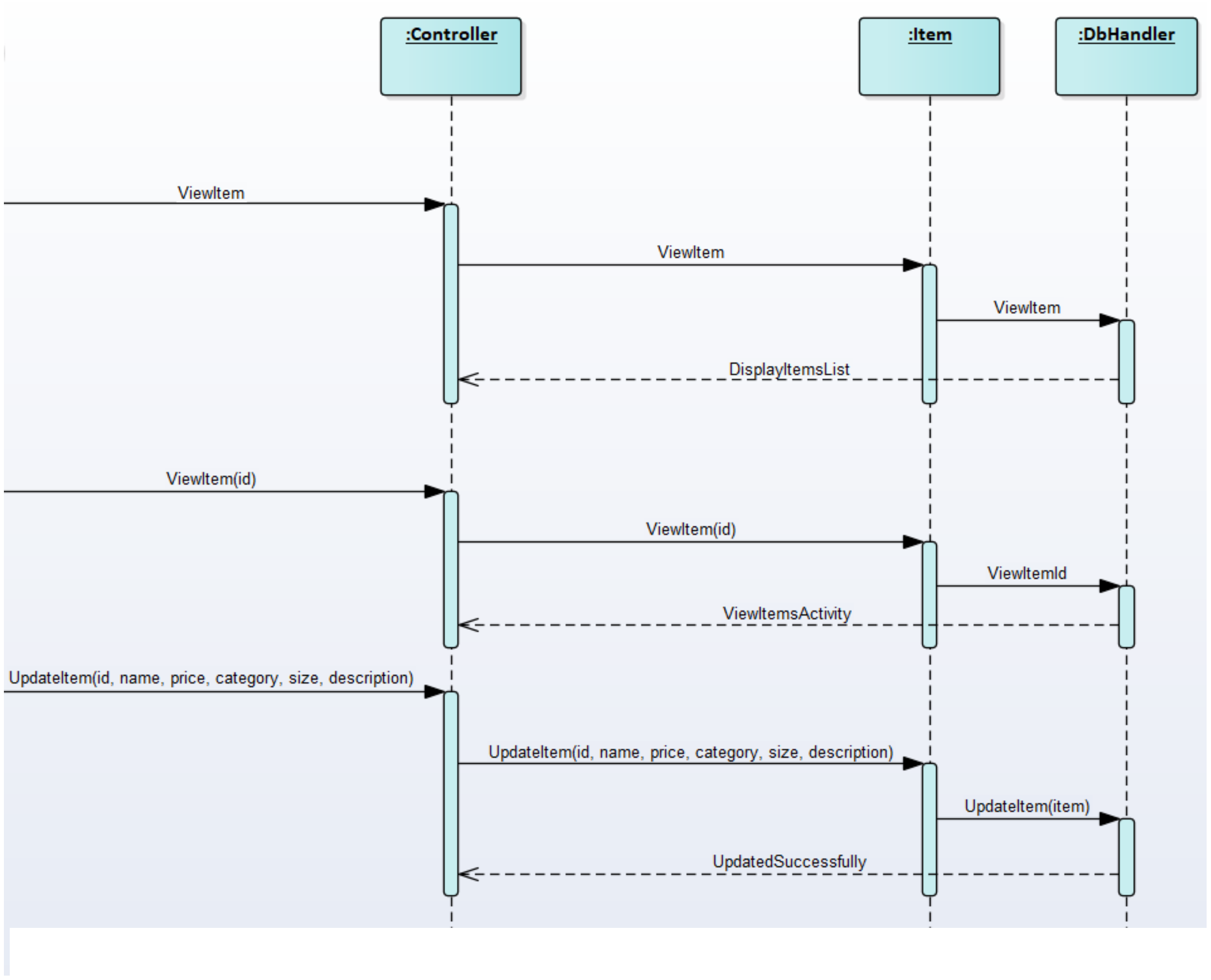


Figure 59: SD Update Item

SD Delete Item.

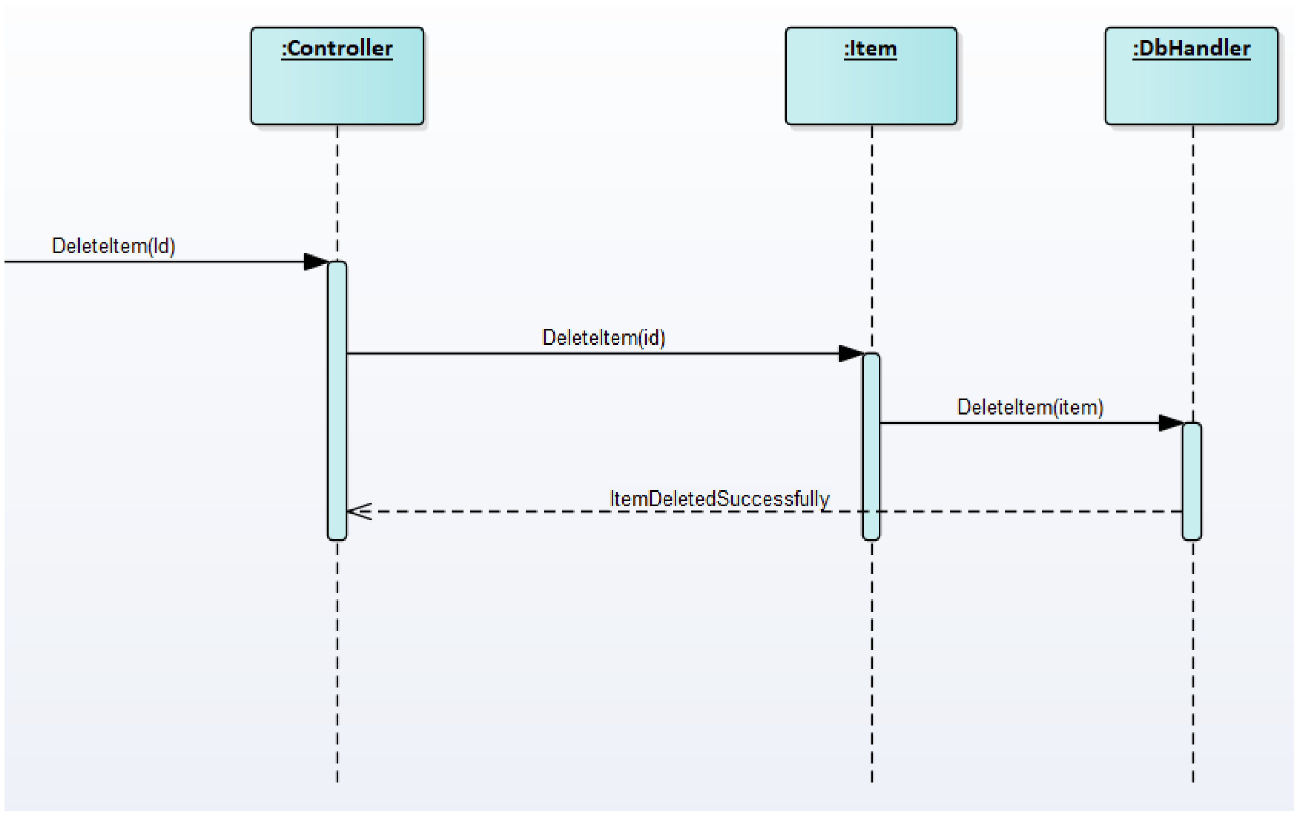


Figure 60: SD Delete Item

SD Block Customer Account.

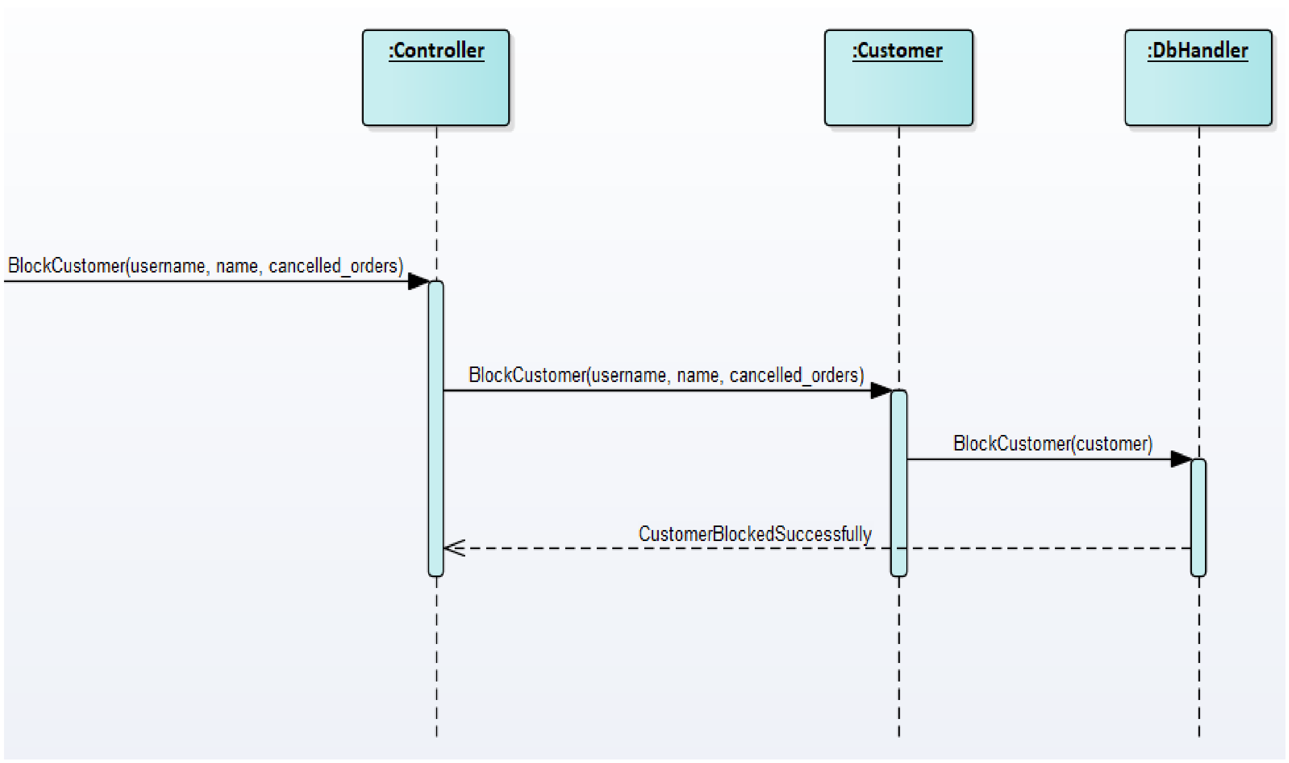

:Controller
:Customer
:DbHandler
BlockCustomer(username, name, cancelled_orders)
BlockCustomer(username, name, cancelled_orders)
BlockCustomer(customer)
CustomerBlockedSuccessfully

Figure 61: SD Block Customer Account

SD UnBlock Customer Account.

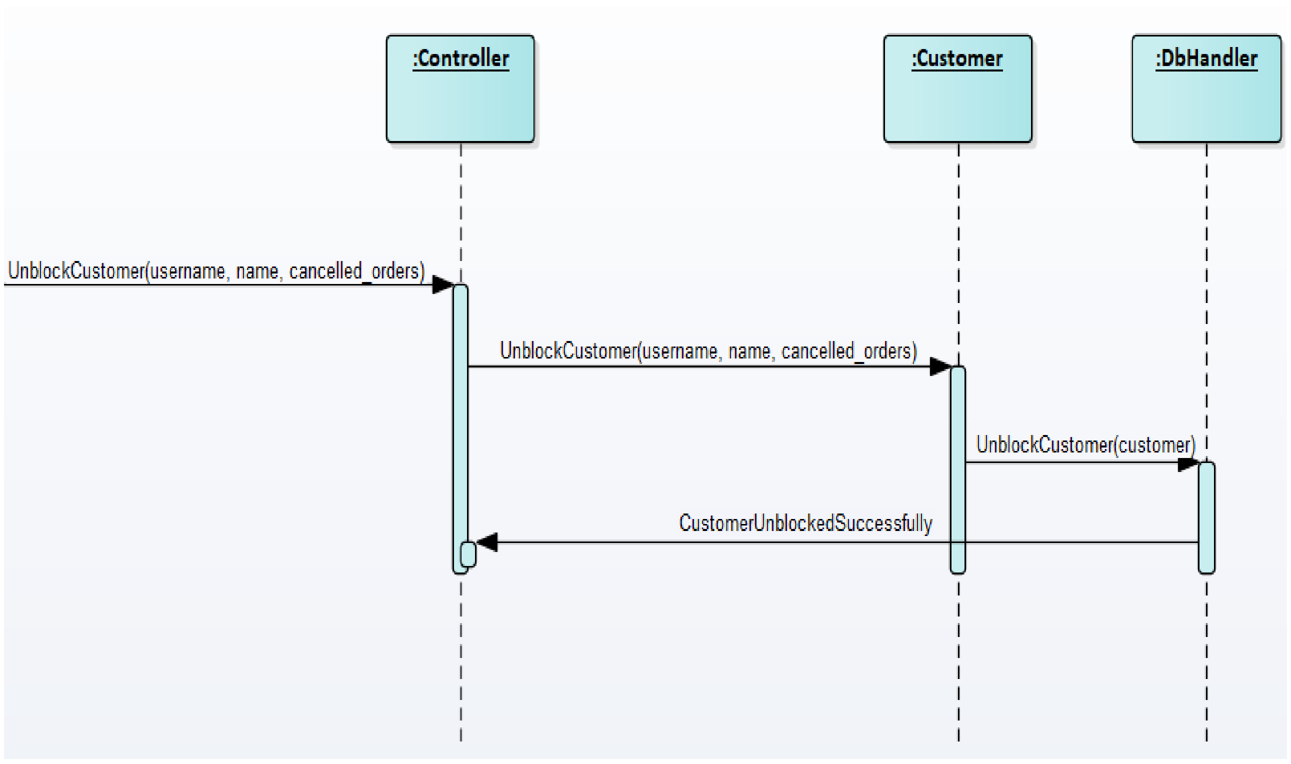

:Controller
:Customer
:DbHandler
UnblockCustomer(username, name, cancelled_orders)
UnblockCustomer(username, name, cancelled_orders)
UnblockCustomer(customer)
CustomerUnblockedSuccessfully

Figure 62: SD Unblock Customer Account

SD Add Biker.

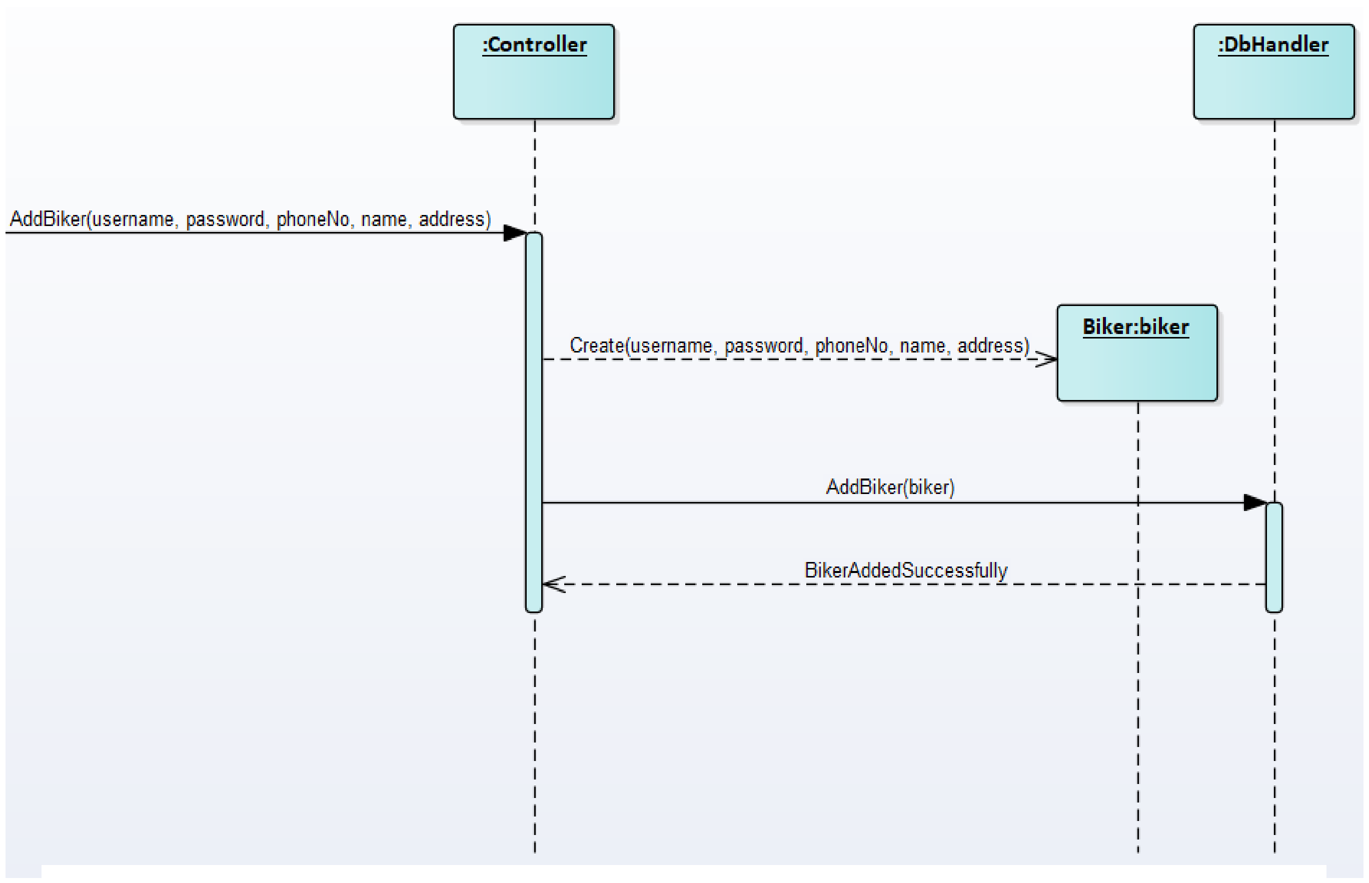

:Controller
:DbHandler
AddBiker(username, password, phoneNo, name, address)
Biker:biker
Create(username, password, phoneNo, name, address)
AddBiker(biker)
BikerAddedSuccessfully

Figure 63: SD Add Biker

SD View Biker.

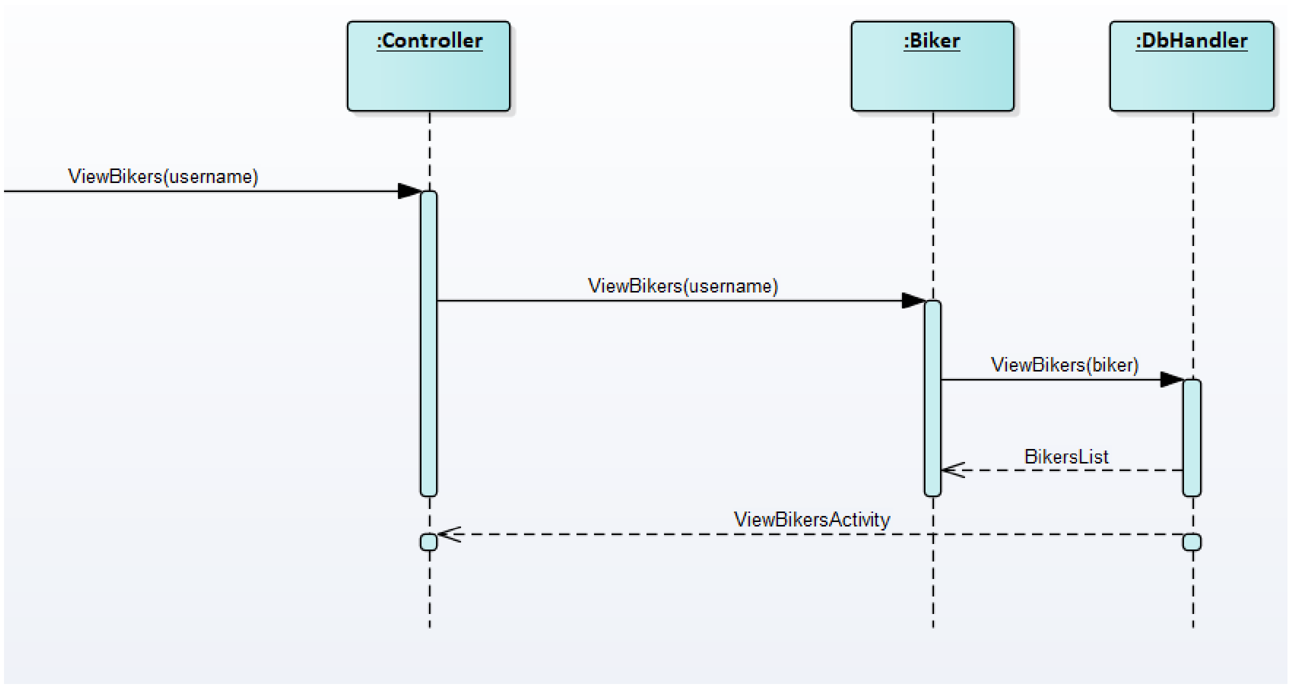


Figure 64: SD View Biker

SD Update Biker.

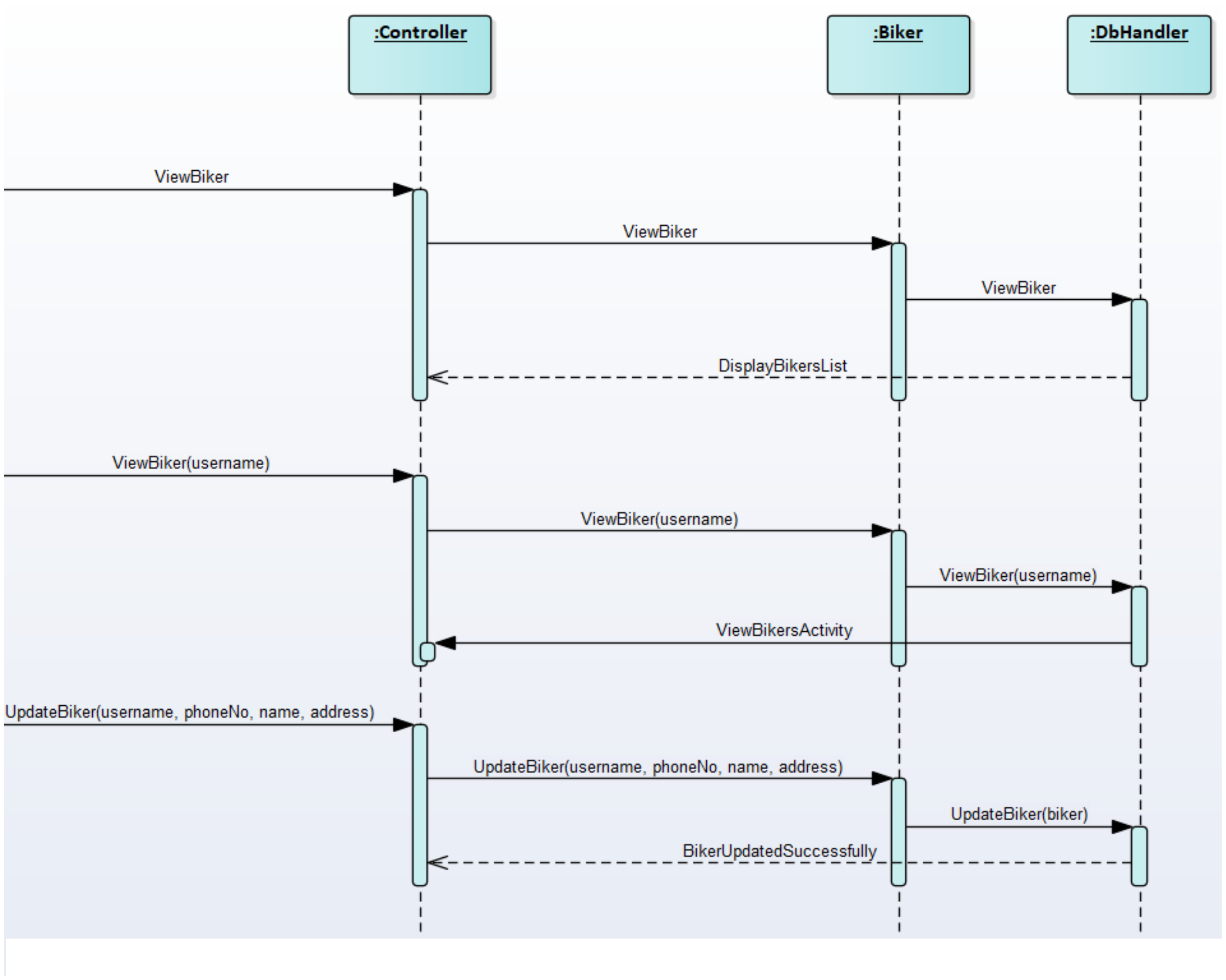


Figure 65: SD Update Biker

SD Delete Biker.

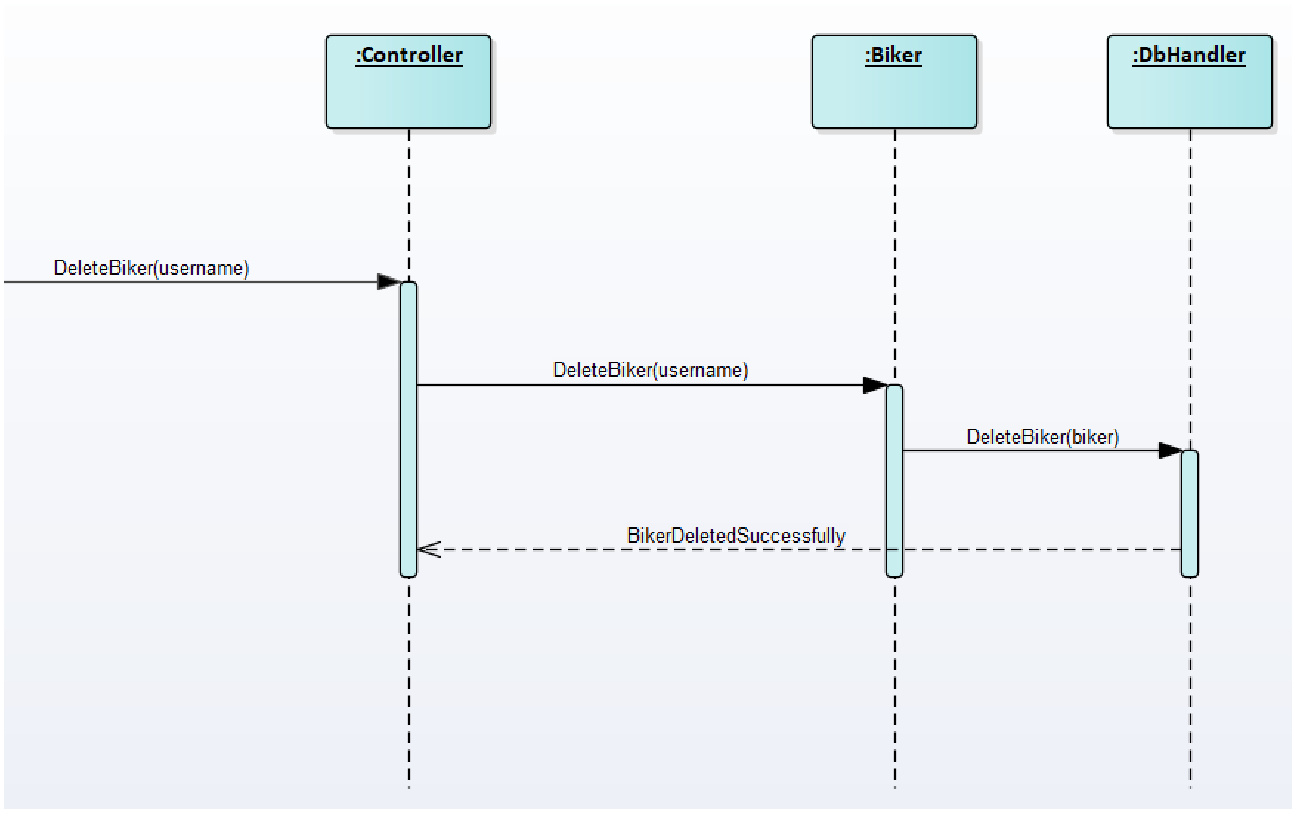


Figure 66: SD Delete Biker

SD Update Banner.

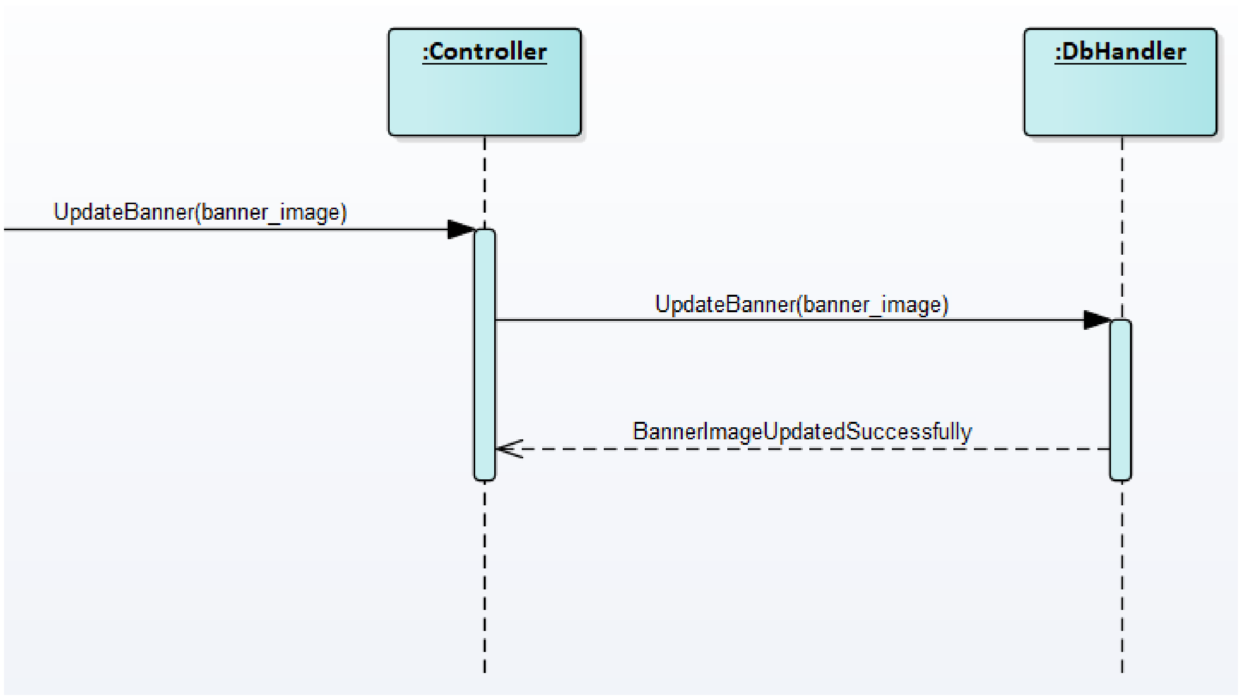

:Controller
:DbHandler
UpdateBanner(banner_image)
UpdateBanner(banner_image)
BannerImageUpdatedSuccessfully

Figure 67: SD Update Banner

SD Accept Order.

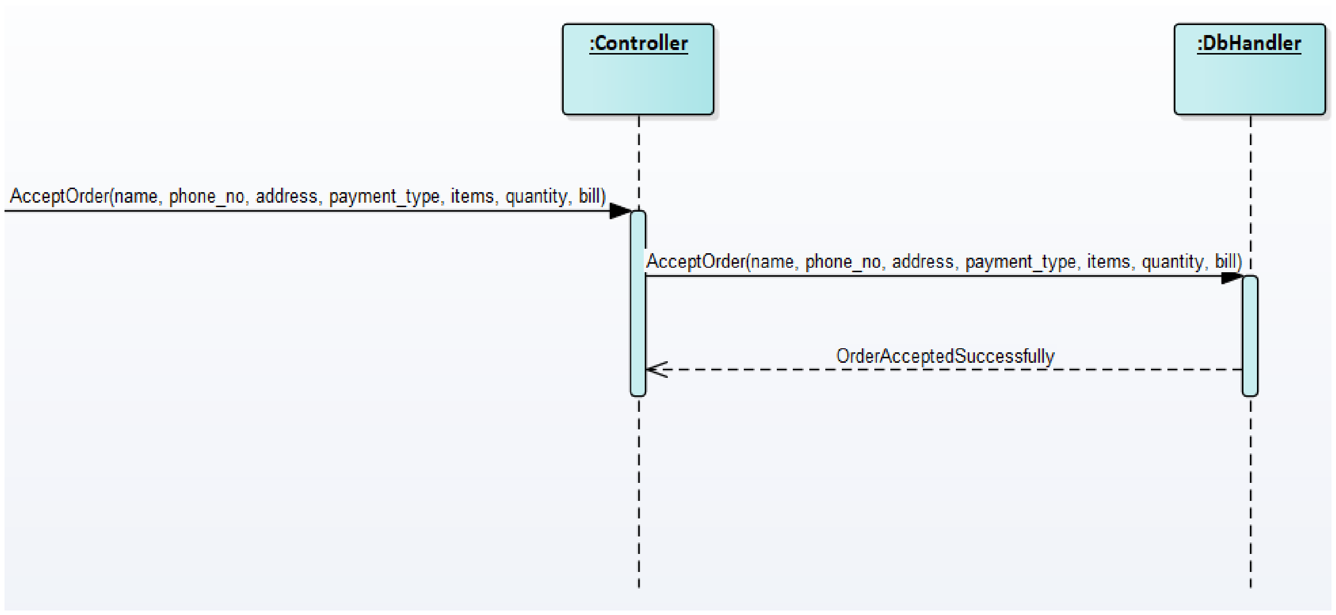

:Controller
:DbHandler
AcceptOrder(name, phone_no, address, payment_type, items, quantity, bill)
AcceptOrder(name, phone_no, address, payment_type, items, quantity, bill)
OrderAcceptedSuccessfully

Figure 68: SD Accept Order

SD Generate Bill.

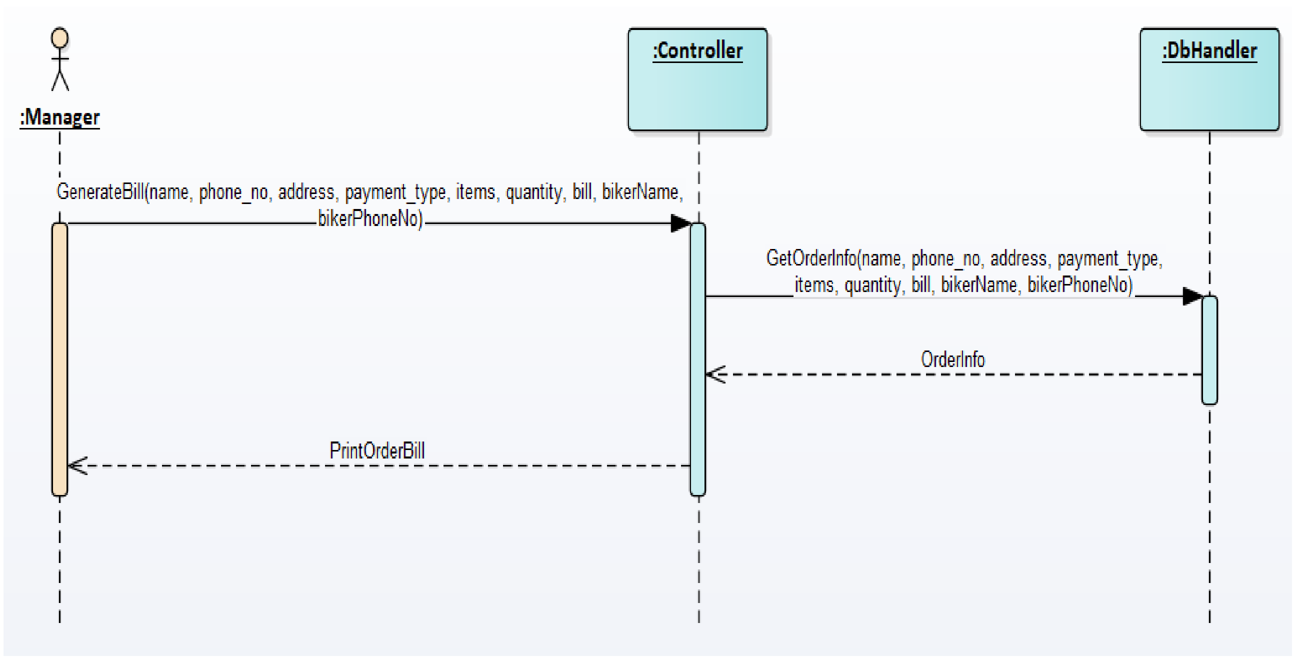

:Manager
:Controller
:DbHandler
GenerateBill(name, phone_no, address, payment_type, items, quantity, bill, bikerName, bikerPhoneNo)
GetOrderInfo(name, phone_no, address, payment_type, items, quantity, bill, bikerName, bikerPhoneNo)
OrderInfo
PrintOrderBill

Figure 69: SD Generate Bill

SD Assign Order to Biker.

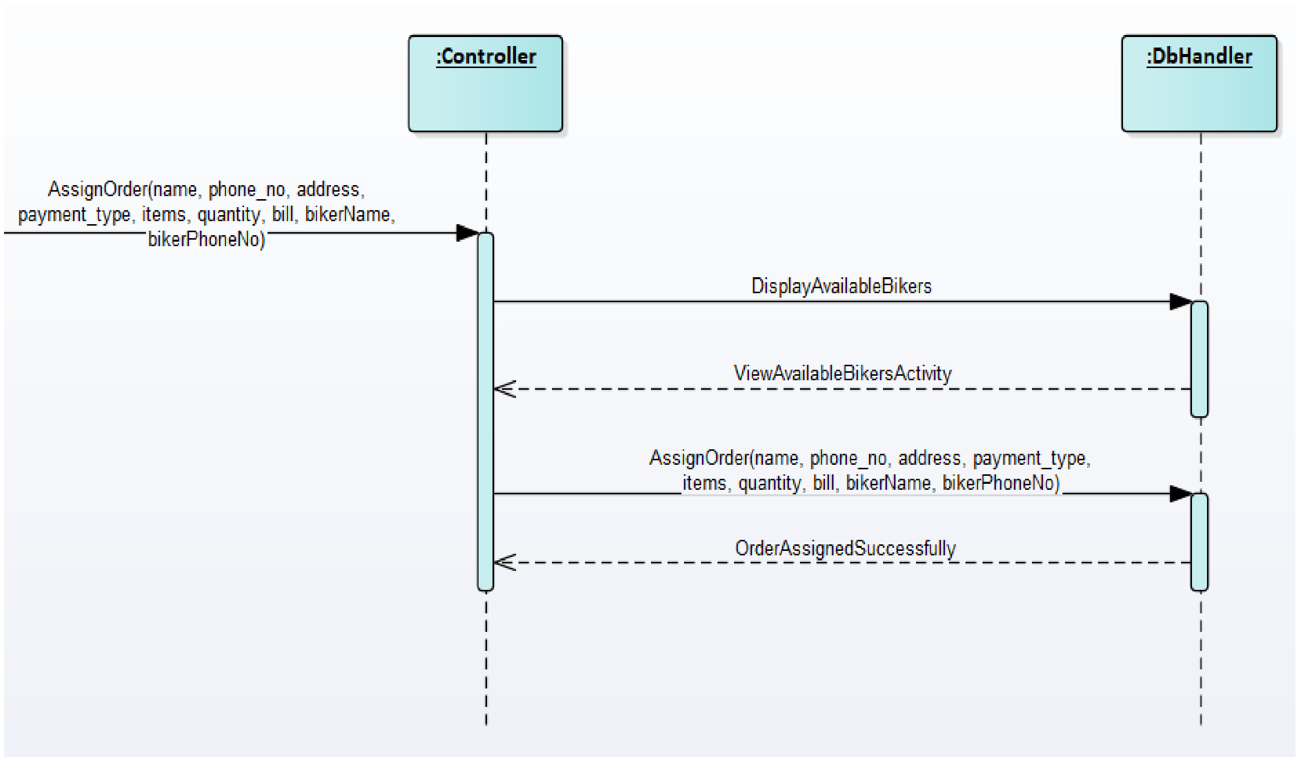

:Controller
:DbHandler
AssignOrder(name, phone_no, address, payment_type, items, quantity, bill, bikerName, bikerPhoneNo)
DisplayAvailableBikers
ViewAvailableBikersActivity
AssignOrder(name, phone_no, address, payment_type, items, quantity, bill, bikerName, bikerPhoneNo)
OrderAssignedSuccessfully

Figure 70: SD Assign Order to Biker

SD View Feedback.

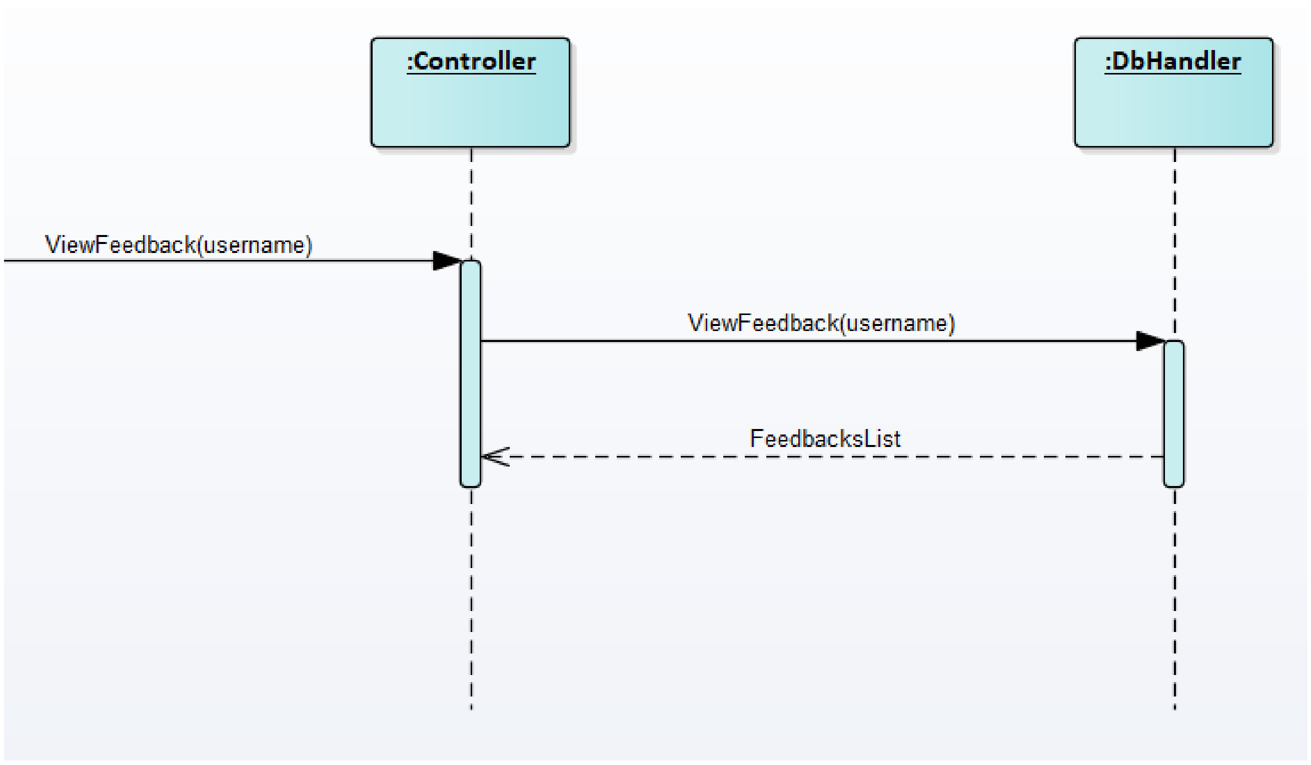


Figure 71: SD View Feedback

SD View Order Info.

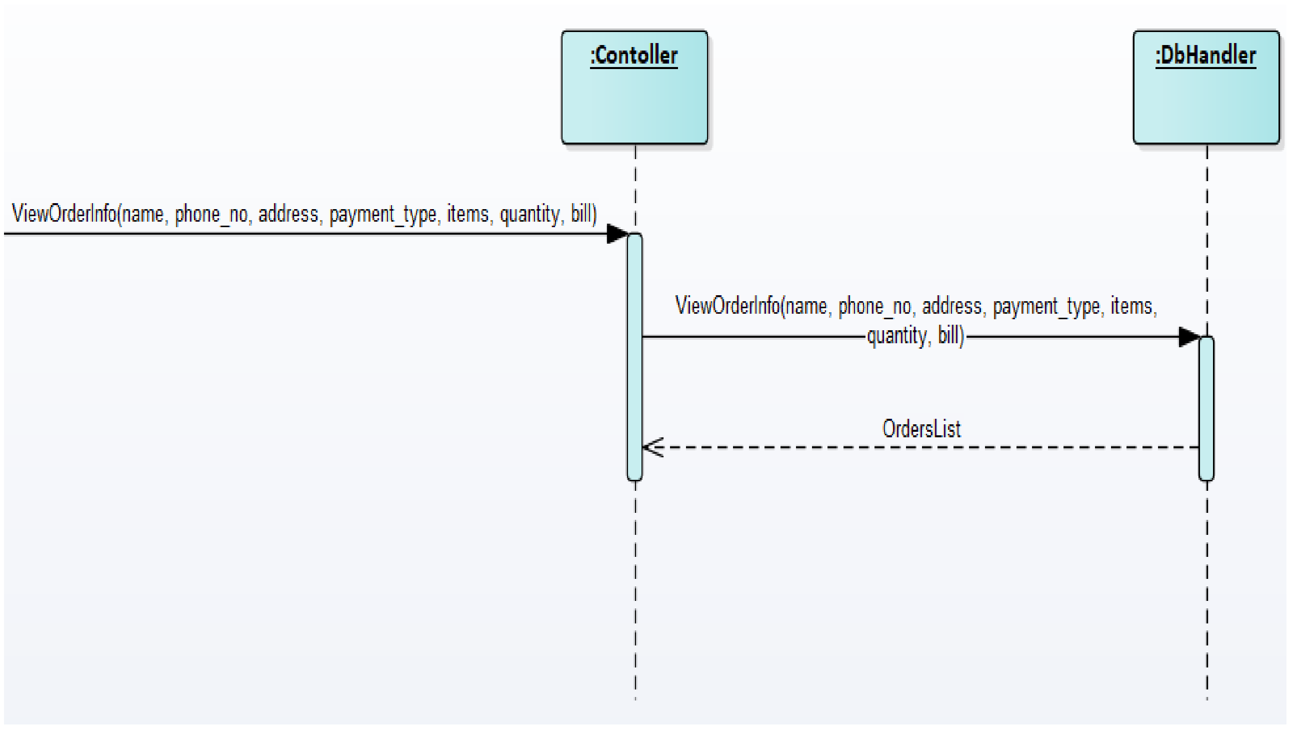

:Contoller
:DbHandler
ViewOrderInfo(name, phone_no, address, payment_type, items, quantity, bill)
ViewOrderInfo(name, phone_no, address, payment_type, items, quantity, bill)
OrdersList

Figure 72: SD View Order Info

SD Generate Reports.

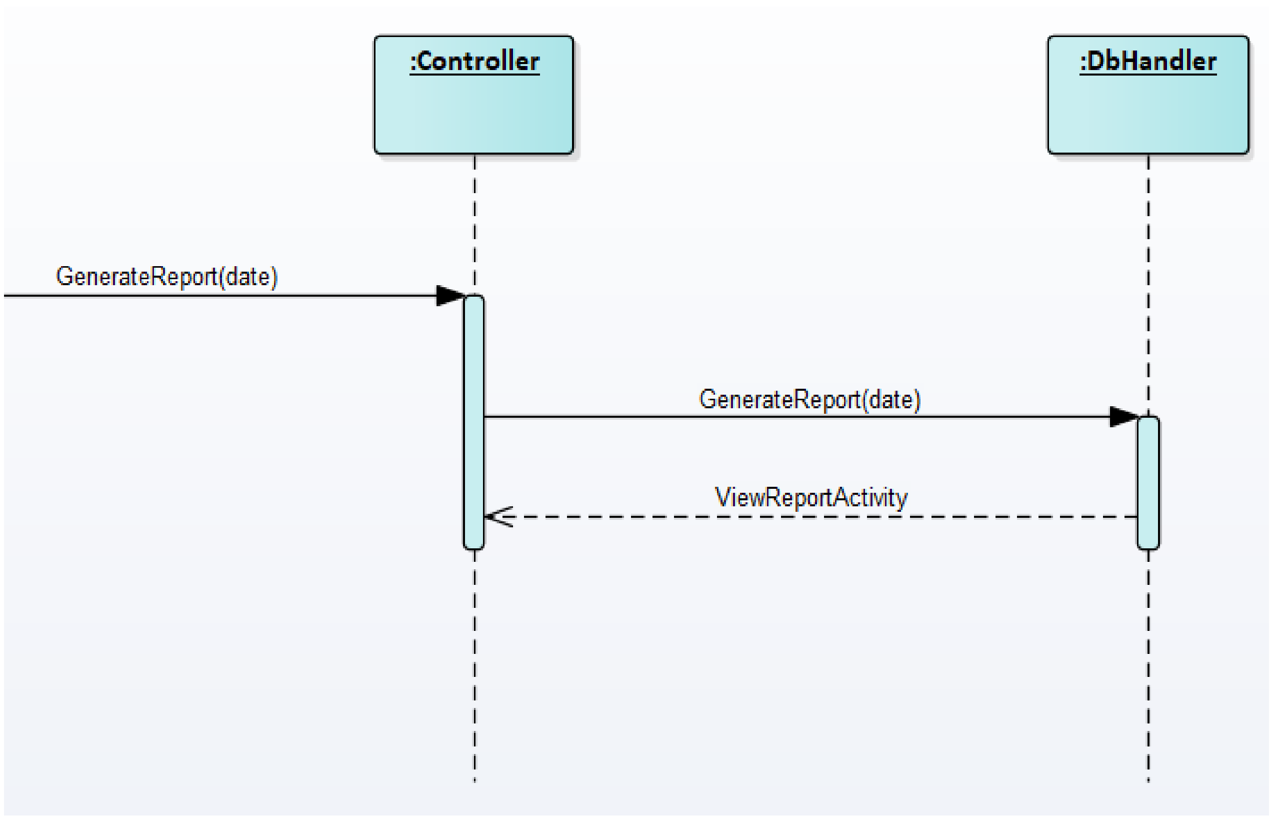


Figure 73: SD Generate Reports

SD Generate Reports PDF.

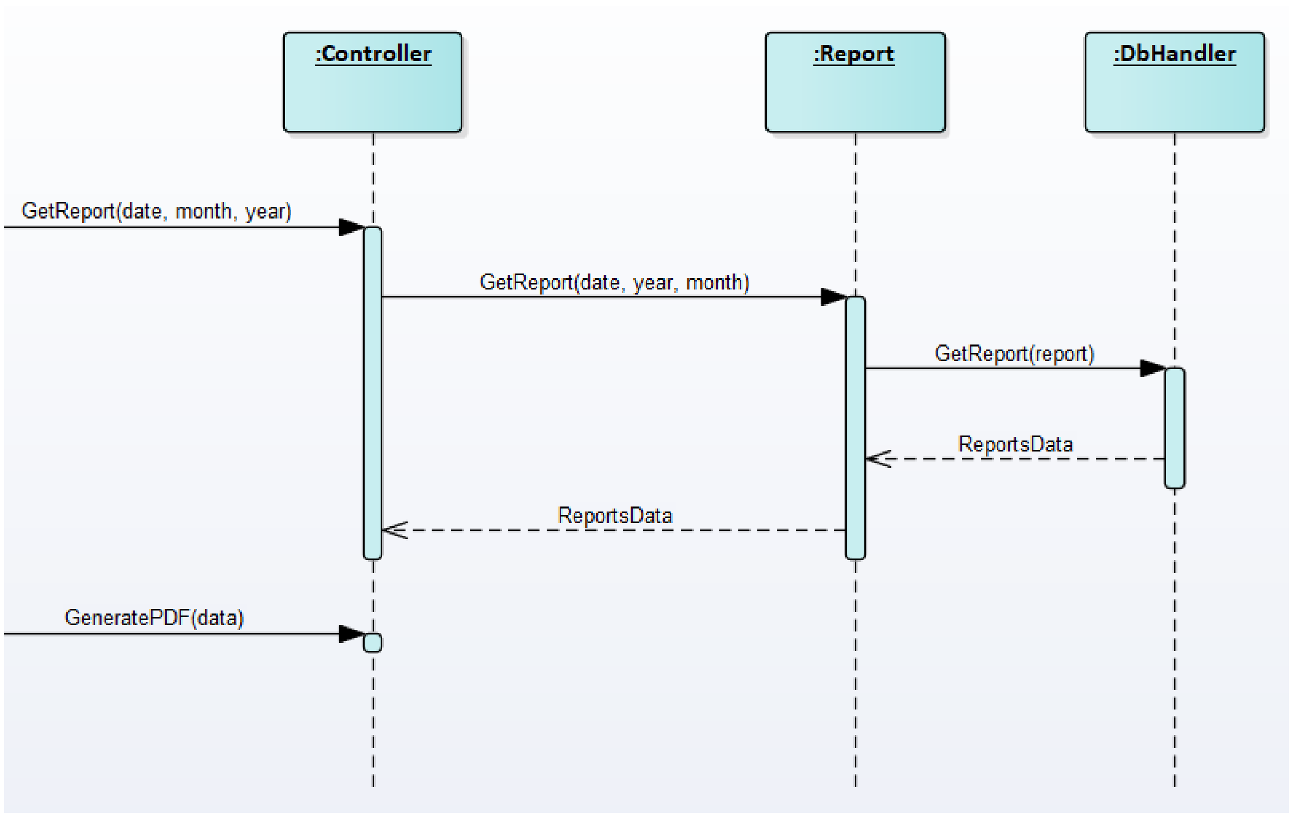


Figure 74: SD Generate Reports PDF

SD Add Expense.

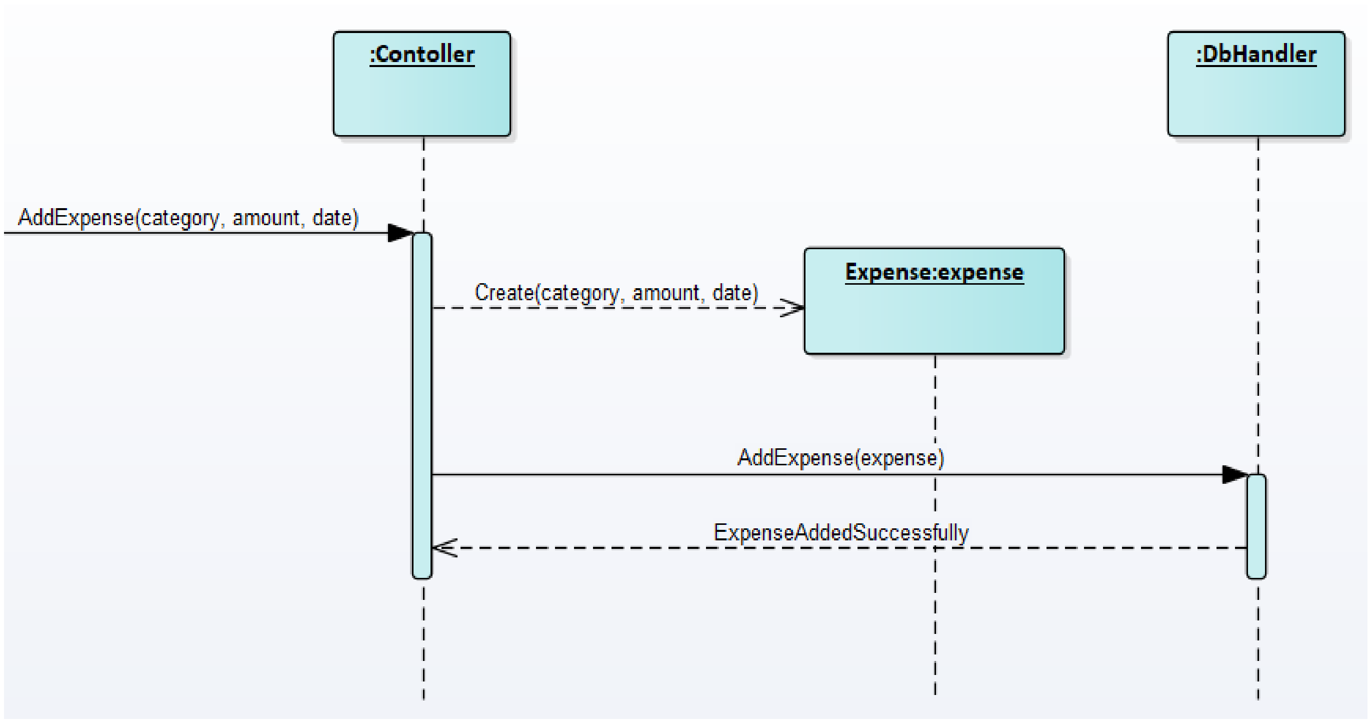


Figure 75: SD Add Expense

SD View Expense.

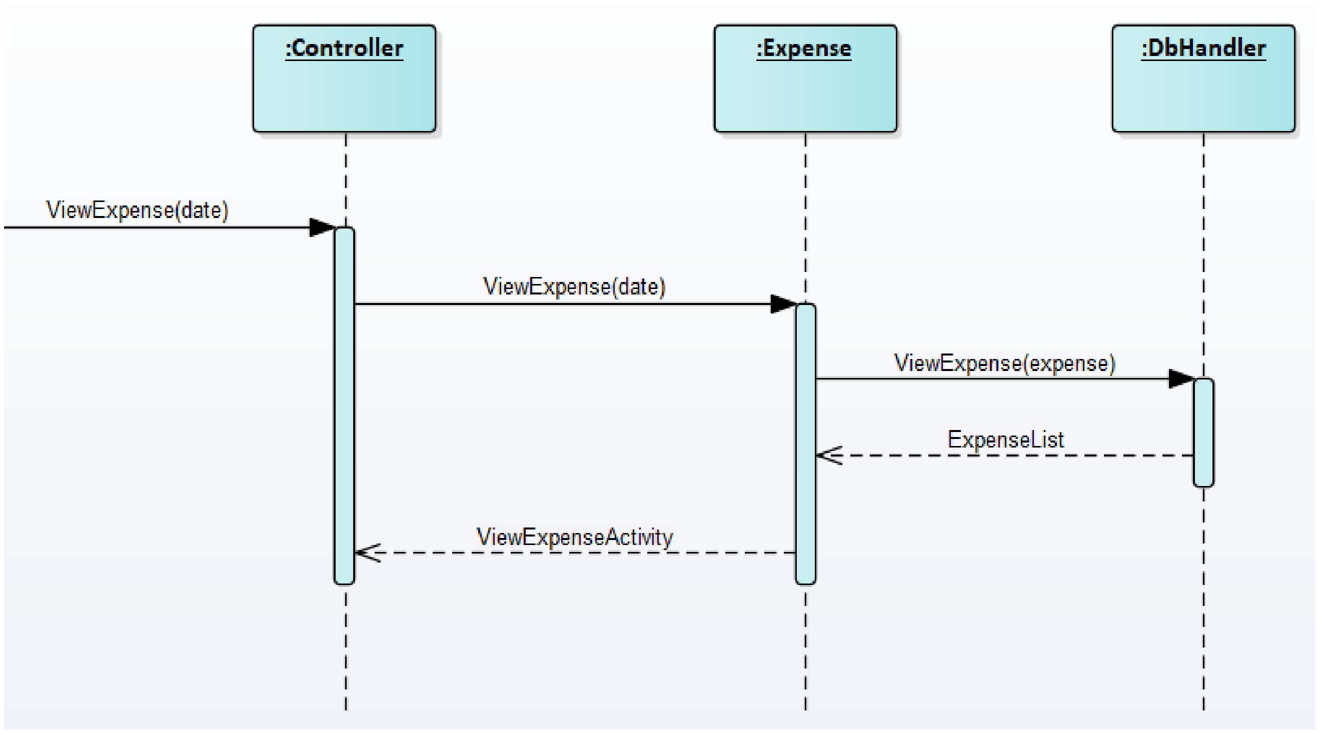


Figure 76: SD View Expense

Customer Module

SD Signup.

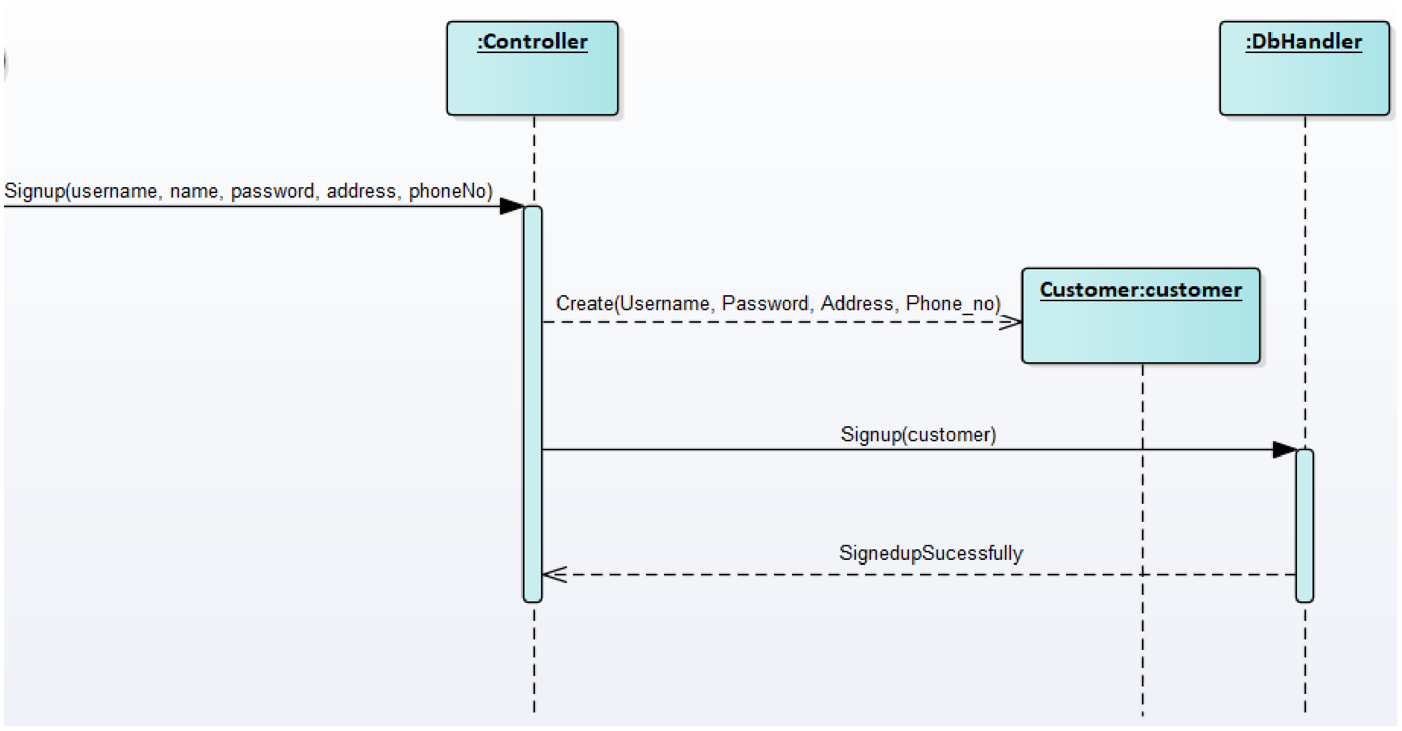

:Controller
:DbHandler
Signup(username, name, password, address, phoneNo)
Create(Username, Password, Address, Phone_no)
Customer:customer
Signup(customer)
SignedupSucessfully

Figure 77: SD Signup

SD Browse Items.

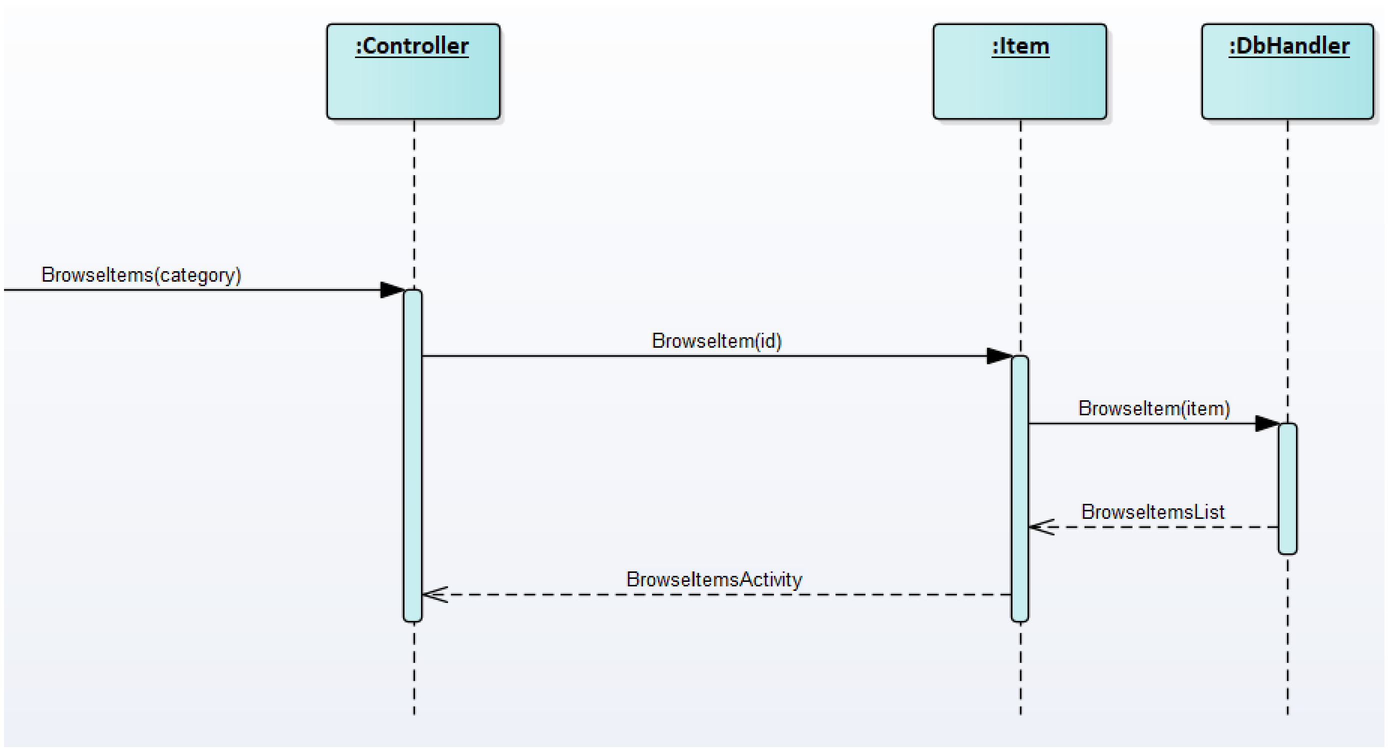


Figure 78: SD Browse Items

SD Checkout.

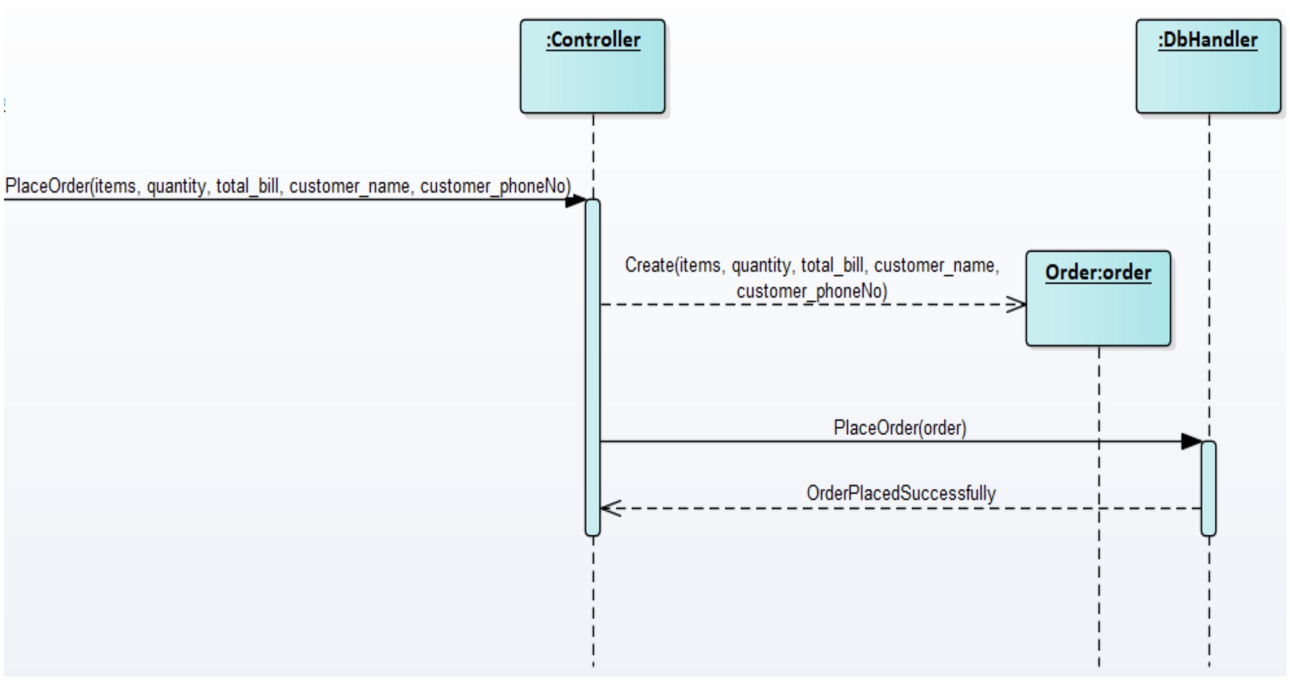

:Controller
:DbHandler
PlaceOrder(items, quantity, total_bill, customer_name, customer_phoneNo)
Create(items, quantity, total_bill, customer_name, customer_phoneNo)
Order:order
PlaceOrder(order)
OrderPlacedSuccessfully

Figure 79: SD Place Order

SD Add to Cart.

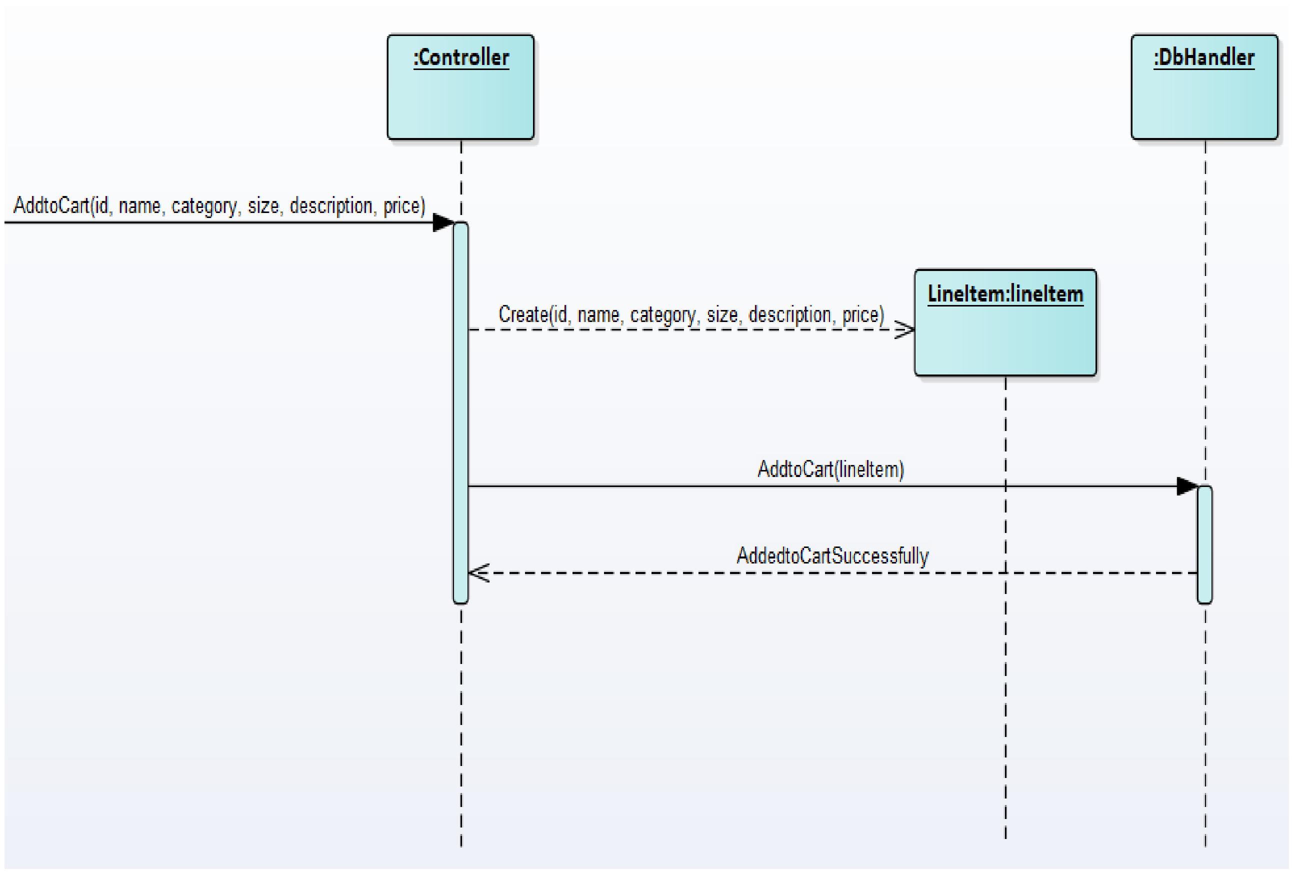
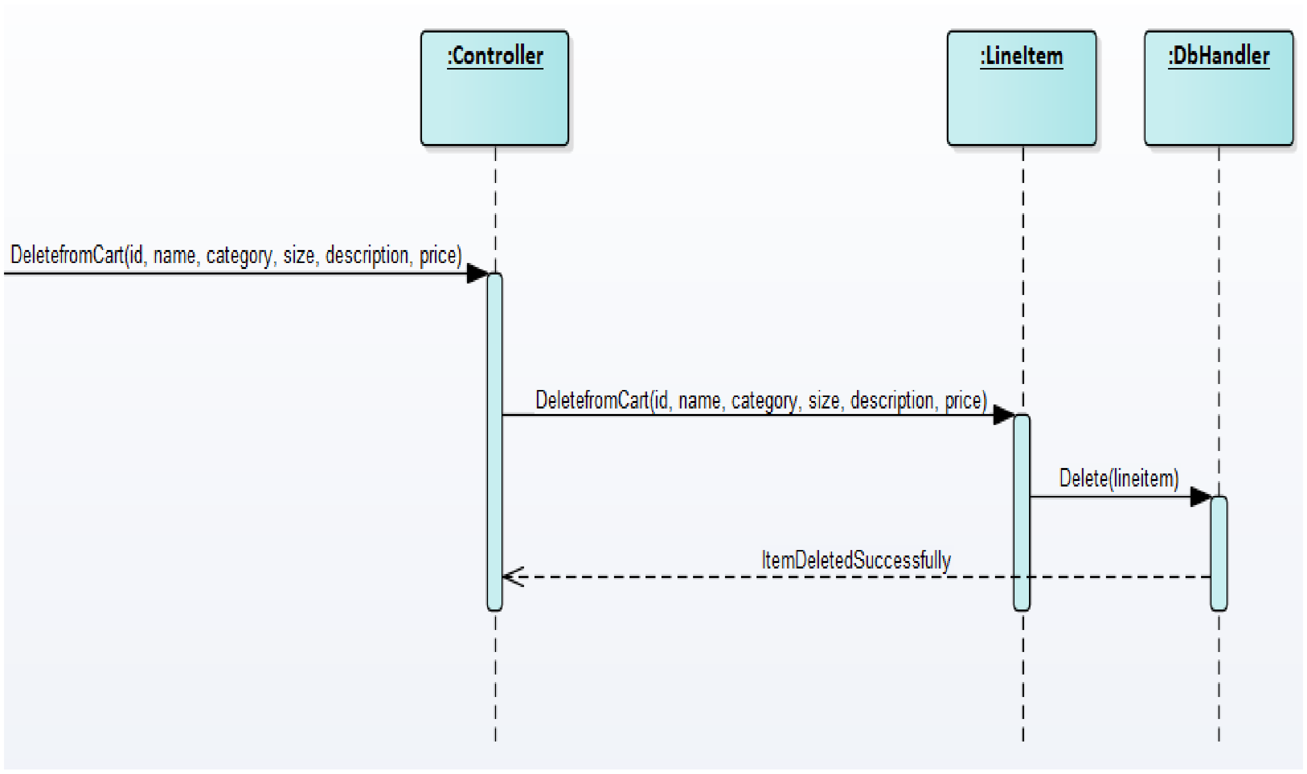

:Controller
:DbHandler
AddtoCart(id, name, category, size, description, price)
Create(id, name, category, size, description, price)
LineItem:lineItem
AddtoCart(lineItem)
AddedtoCartSuccessfully

Figure 80: SD Add to Cart

SD Delete from Cart.



:Controller
:LineItem
:DbHandler
DeletefromCart(id, name, category, size, description, price)
DeletefromCart(id, name, category, size, description, price)
Delete(lineitem)
ItemDeletedSuccessfully

Figure 81: SD Delete from Cart

SD Online Payment.

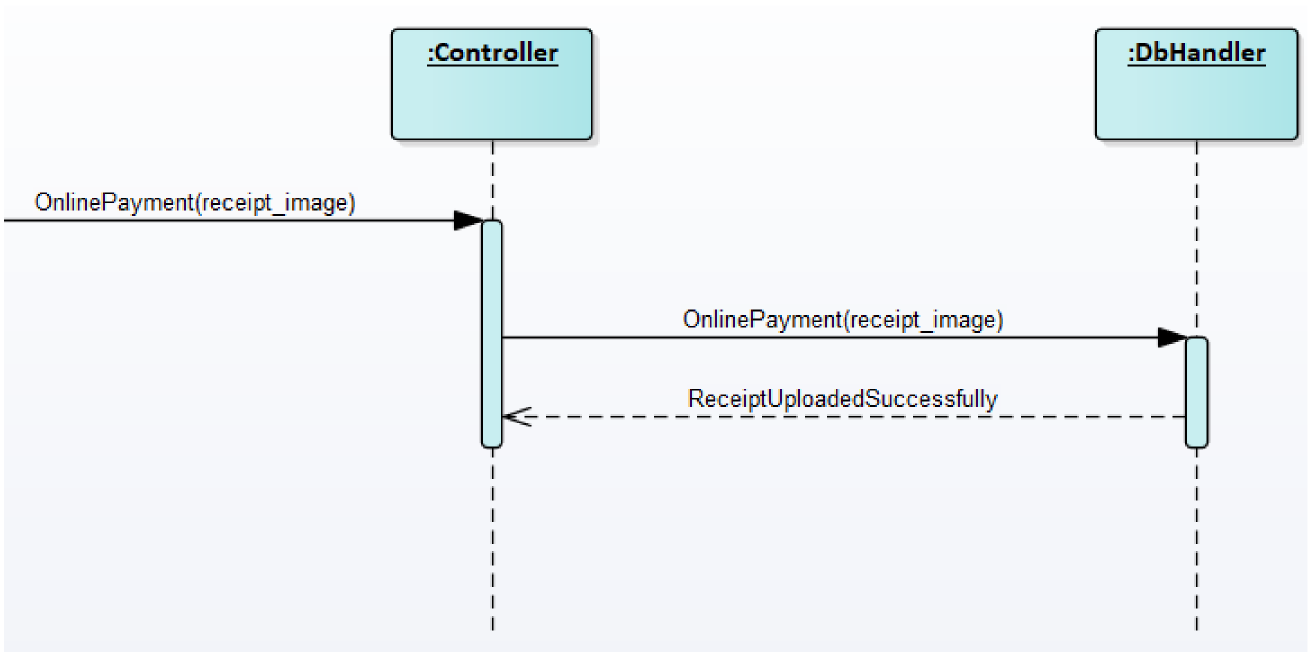

:Controller
:DbHandler
OnlinePayment(receipt_image)
OnlinePayment(receipt_image)
ReceiptUploadedSuccessfully

Figure 82: SD Online Payment

SD Cancel Order.

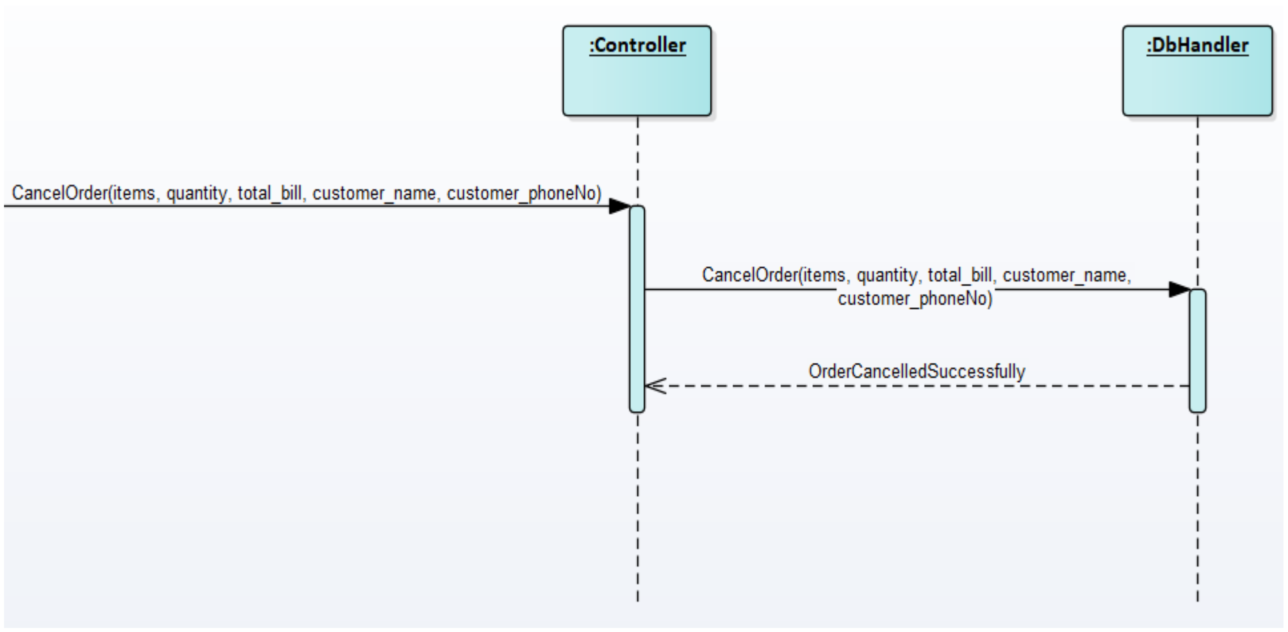

:Controller
:DbHandler
CancelOrder(items, quantity, total_bill, customer_name, customer_phoneNo)
CancelOrder(items, quantity, total_bill, customer_name, customer_phoneNo)
OrderCancelledSuccessfully

Figure 83: SD Cancel Order

SD Submit Feedback.

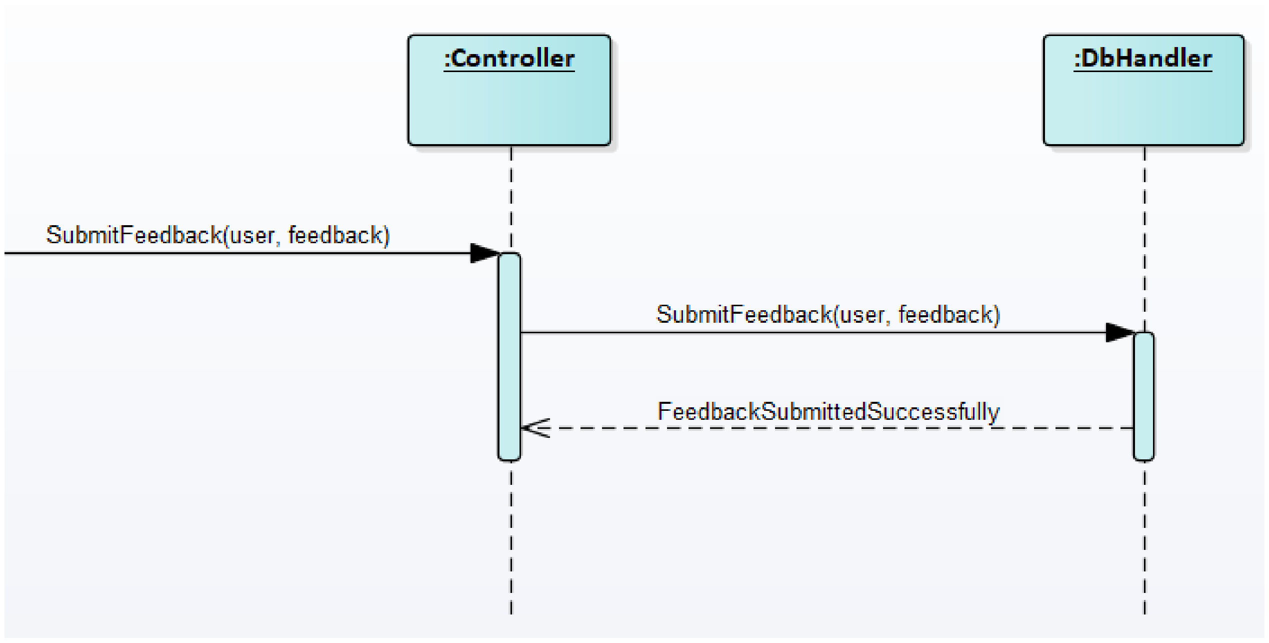

:Controller
:DbHandler
SubmitFeedback(user, feedback)
SubmitFeedback(user, feedback)
FeedbackSubmittedSuccessfully

Figure 84: SD Submit Feedback

SD Verify Email.

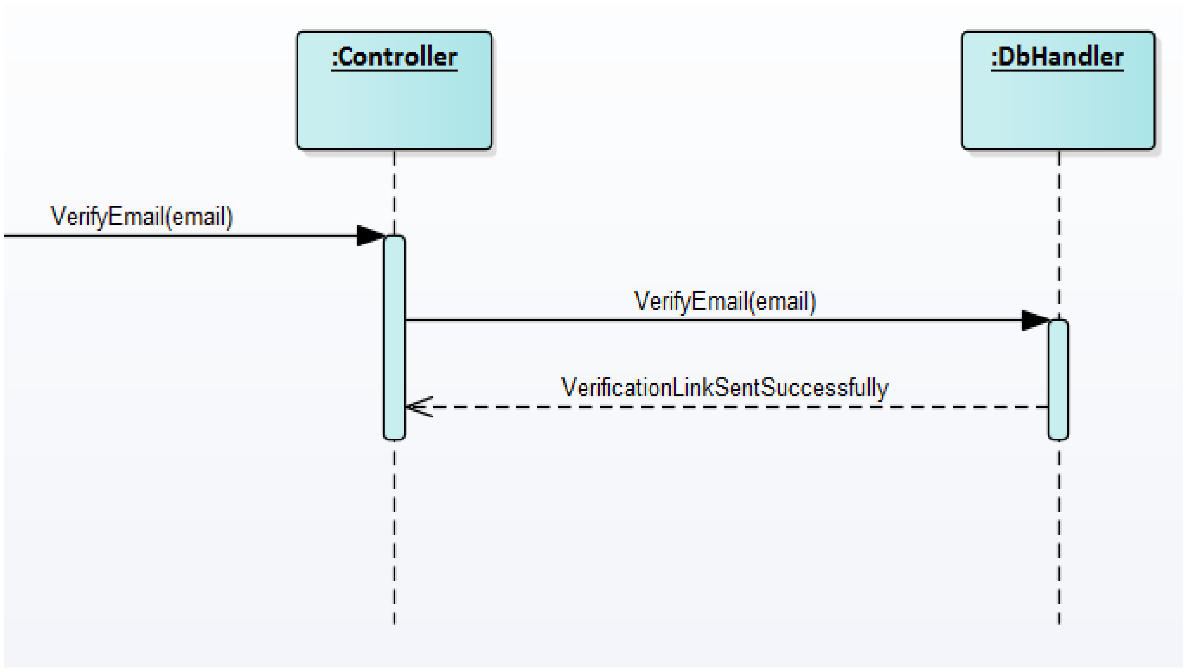


Figure 85: SD Verify Email

Biker Module

SD View Order Info.

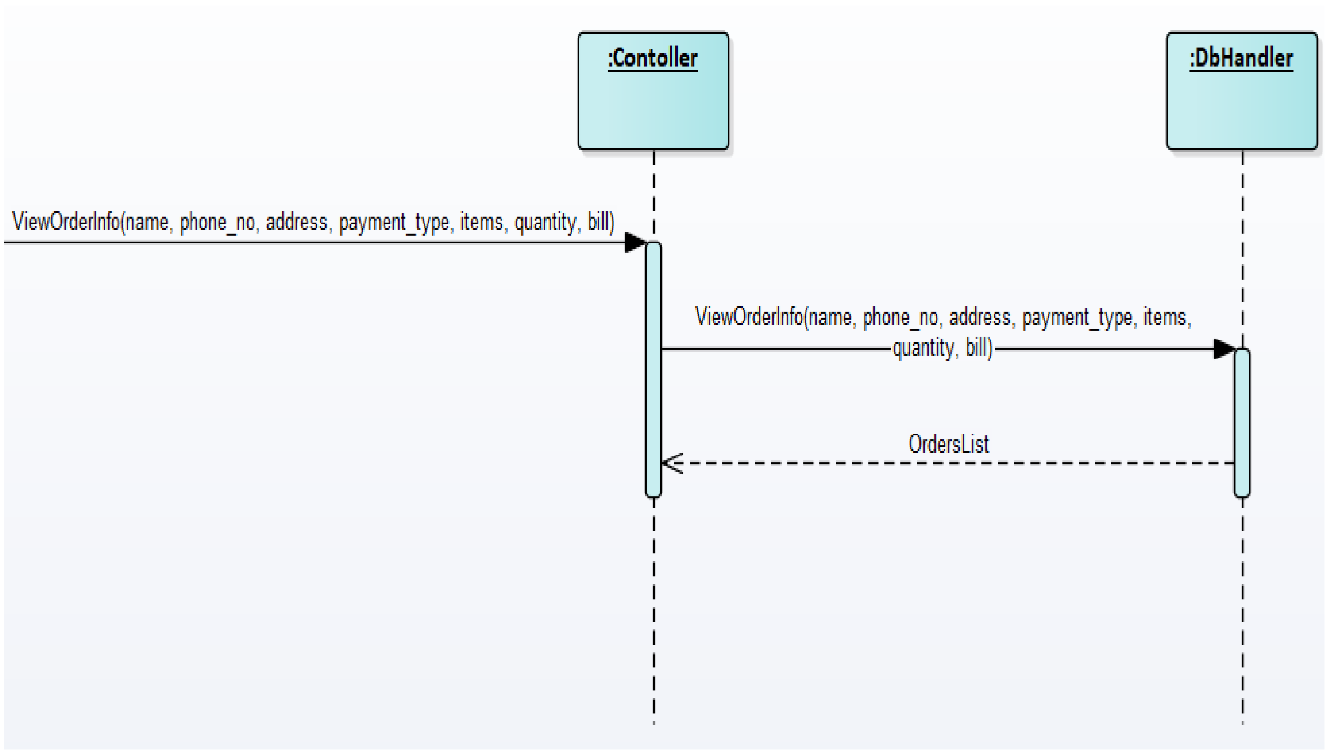

:Contoller
:DbHandler
ViewOrderInfo(name, phone_no, address, payment_type, items, quantity, bill)
ViewOrderInfo(name, phone_no, address, payment_type, items, quantity, bill)
OrdersList

Figure 86: SD View Order Ifno

SD Confirm Delivery.

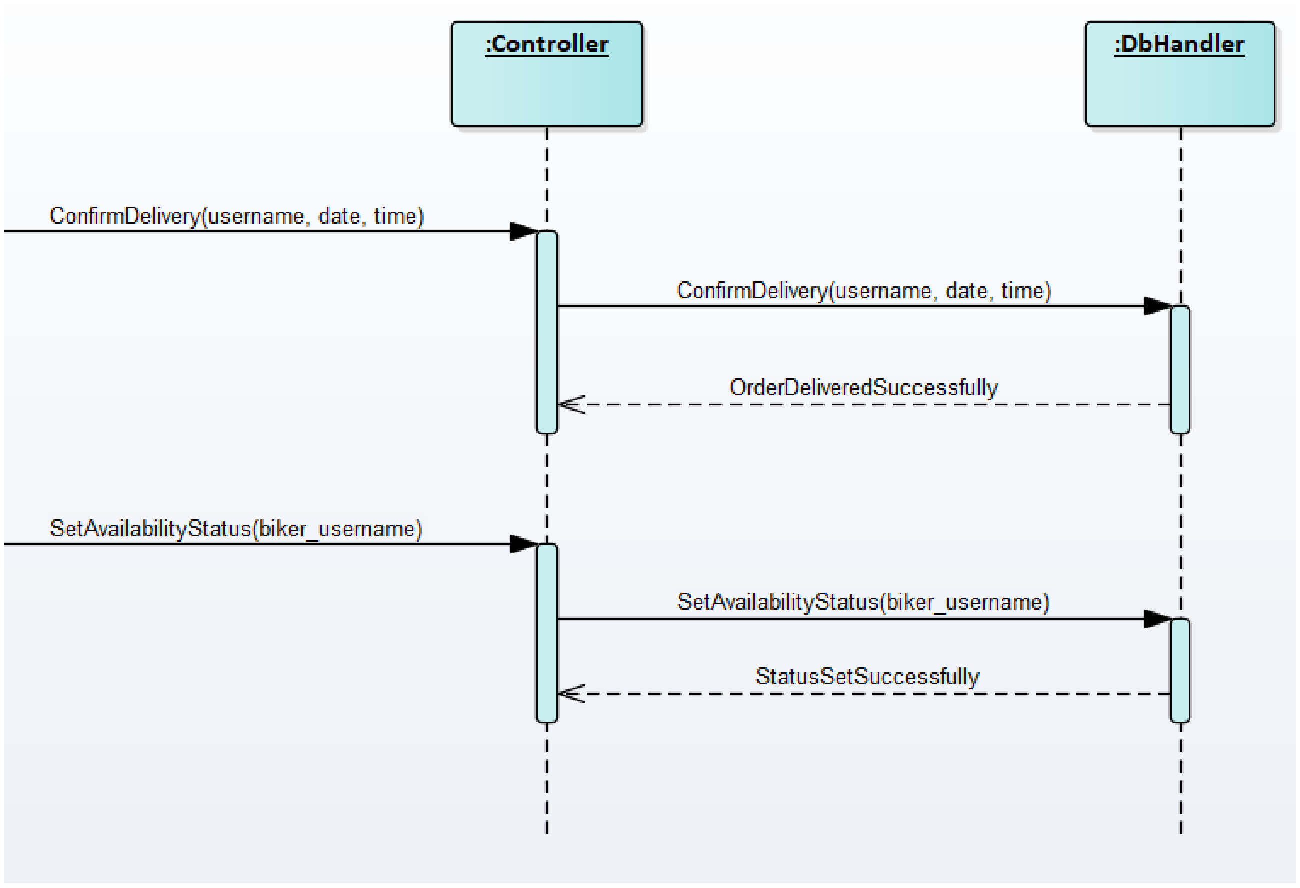

:Controller
:DbHandler
ConfirmDelivery(username, date, time)
ConfirmDelivery(username, date, time)
OrderDeliveredSuccessfully
SetAvailabilityStatus(biker_username)
SetAvailabilityStatus(biker_username)
StatusSetSuccessfully

Figure 87: SD Confirm Delivery

3.4. Architecture Diagram:

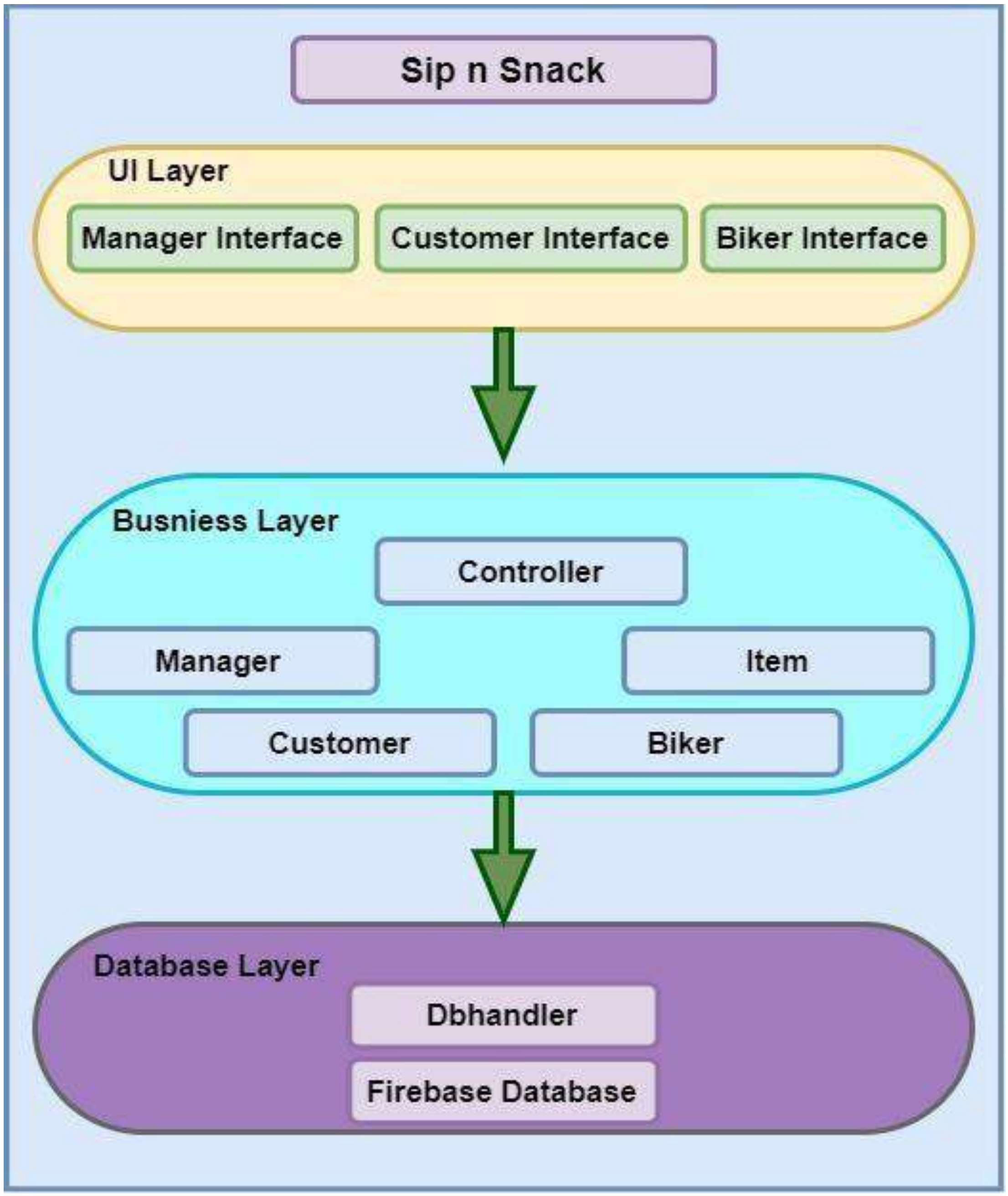
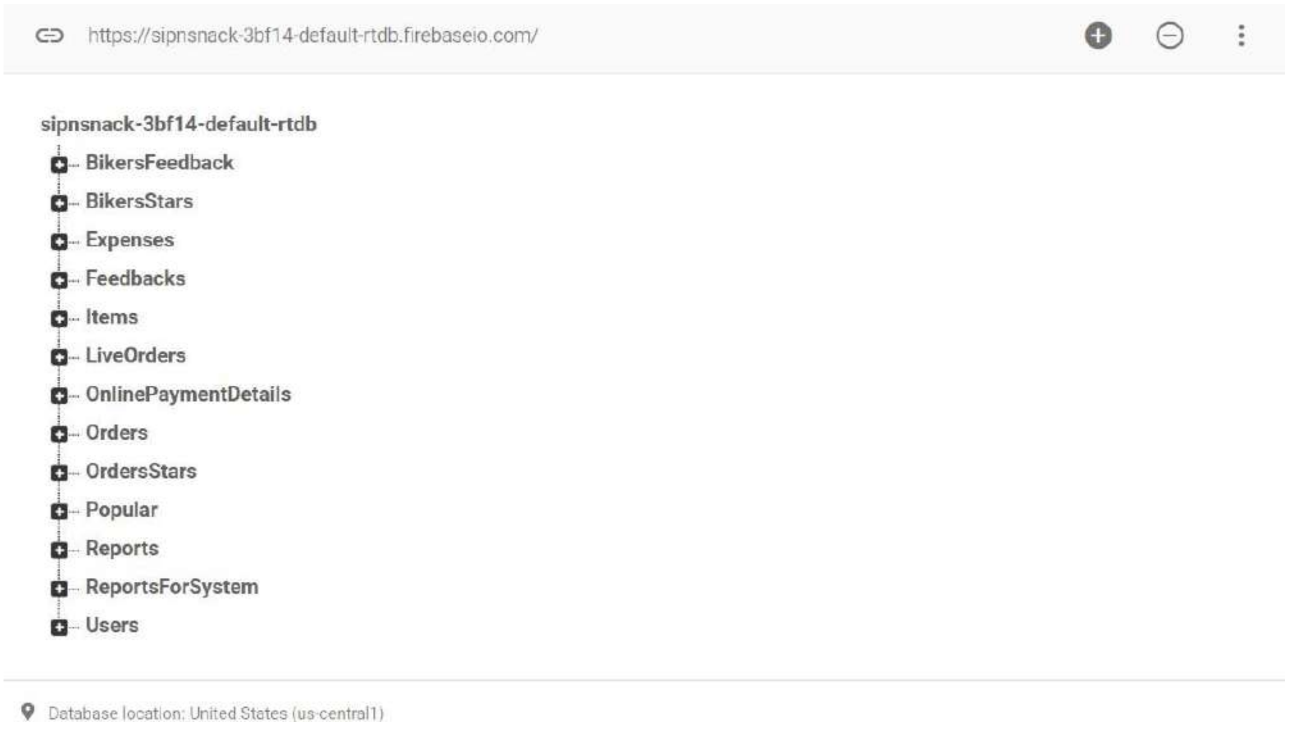


Figure 88: Architecture Diagram

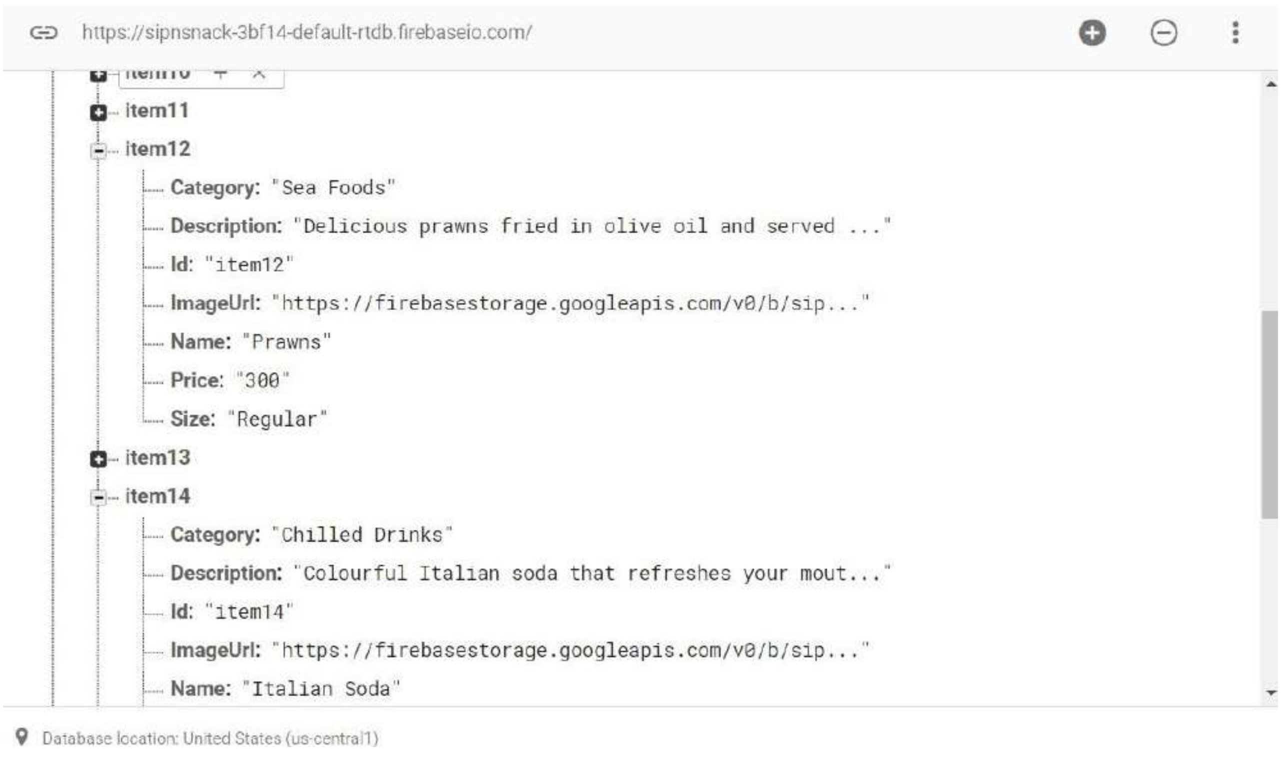
3.5. Database Schema:


https://sipnsnack-3bf14-default-rtdb.firebaseio.com/

sipnsnack-3bf14-default-rtdb
- BikersFeedback
- BikersStars
- Expenses
- Feedbacks
- Items
- LiveOrders
- OnlinePaymentDetails
- Orders
- OrdersStars
- Popular
- Reports
- ReportsForSystem
- Users

Database location: United States (us-central1)

Figure 89: Database Schema - I

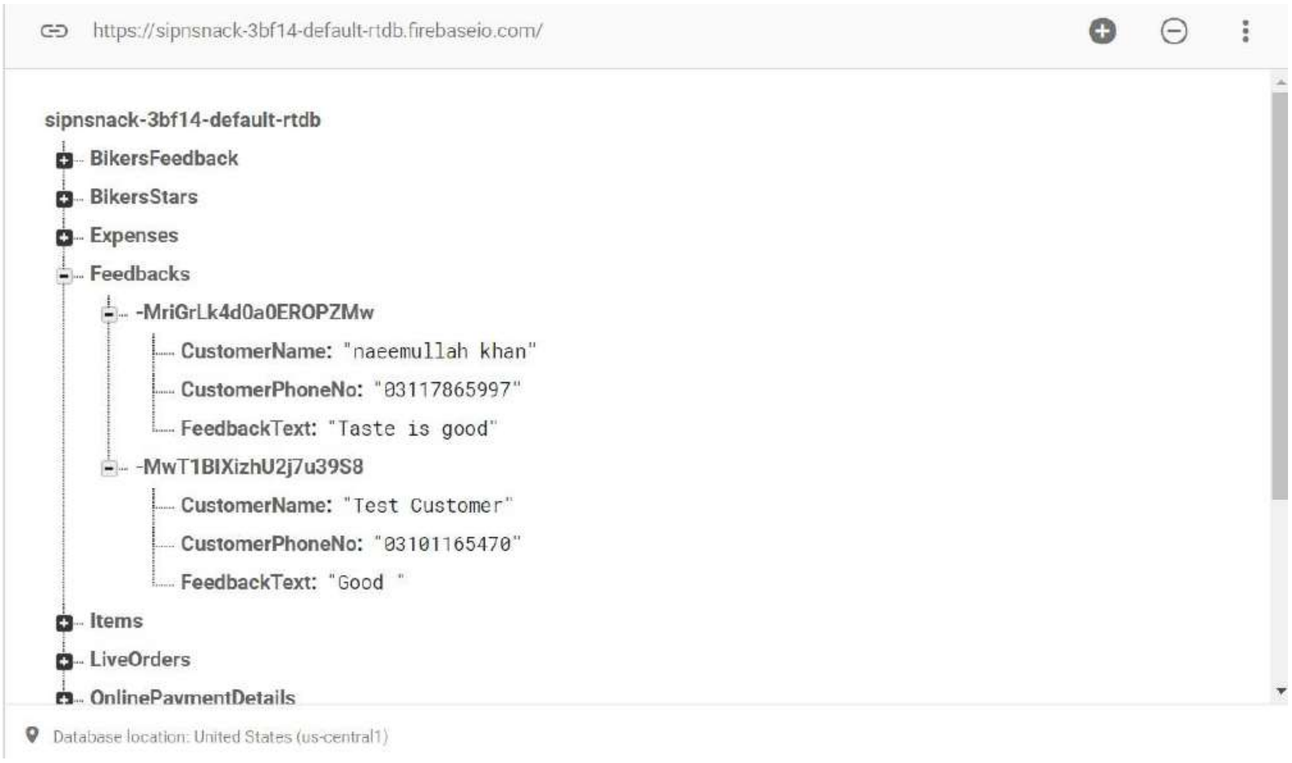


https://sipnsnack-3bf14-default-rtdb.firebaseio.com/

```
item10
item11
item12
    Category: "Sea Foods"
    Description: "Delicious prawns fried in olive oil and served ..."
    Id: "item12"
    ImageUrl: "https://firebasestorage.googleapis.com/v0/b/sip..."
    Name: "Prawns"
    Price: "300"
    Size: "Regular"
item13
item14
    Category: "Chilled Drinks"
    Description: "Colourful Italian soda that refreshes your mout..."
    Id: "item14"
    ImageUrl: "https://firebasestorage.googleapis.com/v0/b/sip..."
    Name: "Italian Soda"
```

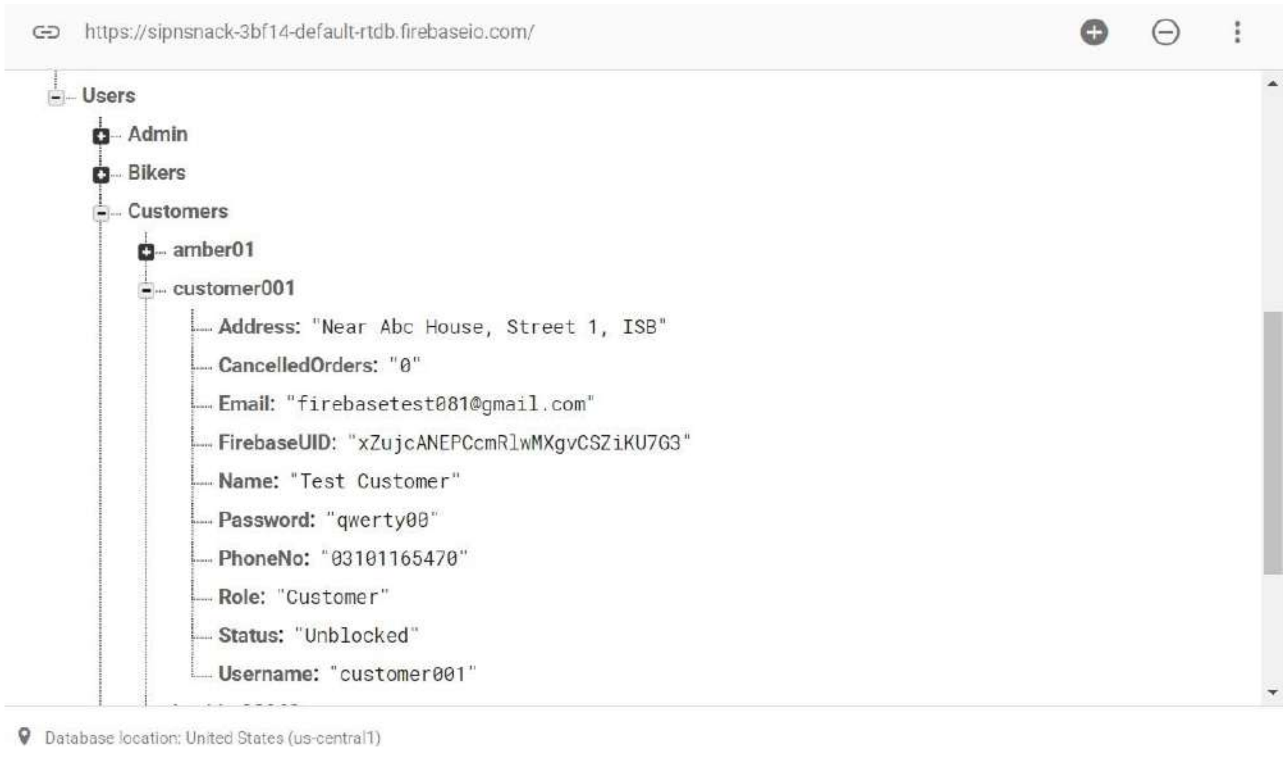
Database location: United States (us-central1)

Figure 90: Database Schema – II



```
https://sipnsnack-3bf14-default-rtdb.firebaseio.com/

sipnsnack-3bf14-default-rtdb
  BikersFeedback
  BikersStars
  Expenses
  Feedbacks
    -MriGrLk4d0a0EROPZMw
      CustomerName: "naeemullah khan"
      CustomerPhoneNo: "03117865997"
      FeedbackText: "Taste is good"
    -MwT1BIXizhU2j7u39S8
      CustomerName: "Test Customer"
      CustomerPhoneNo: "03101165470"
      FeedbackText: "Good "
  Items
  LiveOrders
  OnlinePaymentDetails

Database location: United States (us-central1)
```

Figure 91: Database Schema - III



https://sipnsnack-3bf14-default-rtdb.firebaseio.com/

```
Users
    Admin
    Bikers
    Customers
        amber01
        customer001
            Address: "Near Abc House, Street 1, ISB"
            CancelledOrders: "0"
            Email: "firebasetest081@gmail.com"
            FirebaseUID: "xZujcANEPCcmRlwMXgvCSZiKU7G3"
            Name: "Test Customer"
            Password: "qwerty00"
            PhoneNo: "03101165470"
            Role: "Customer"
            Status: "Unblocked"
            Username: "customer001"
```

Database location: United States (us-central1)

Figure 92: Database Schema – IV

3.6. User Interface Design:

User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions. UI brings together concepts from interaction design, visual design, and information architecture.

Splash Screen:

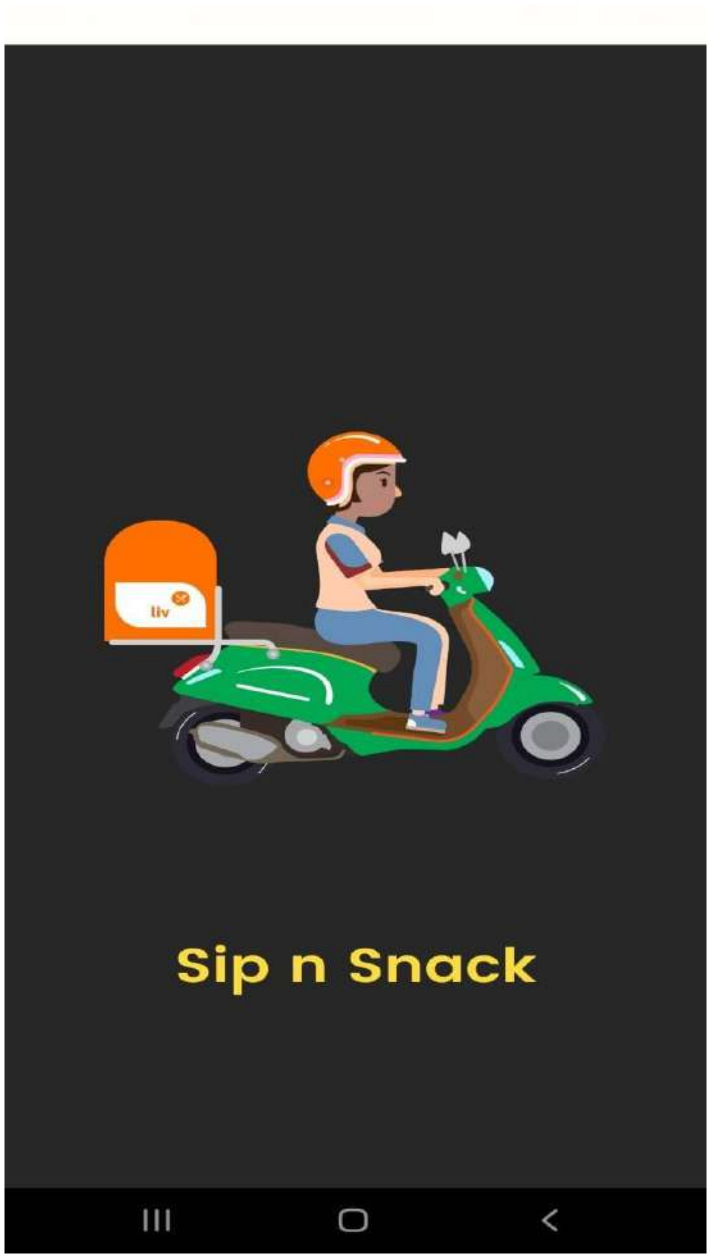


Figure 93: Splash Screen

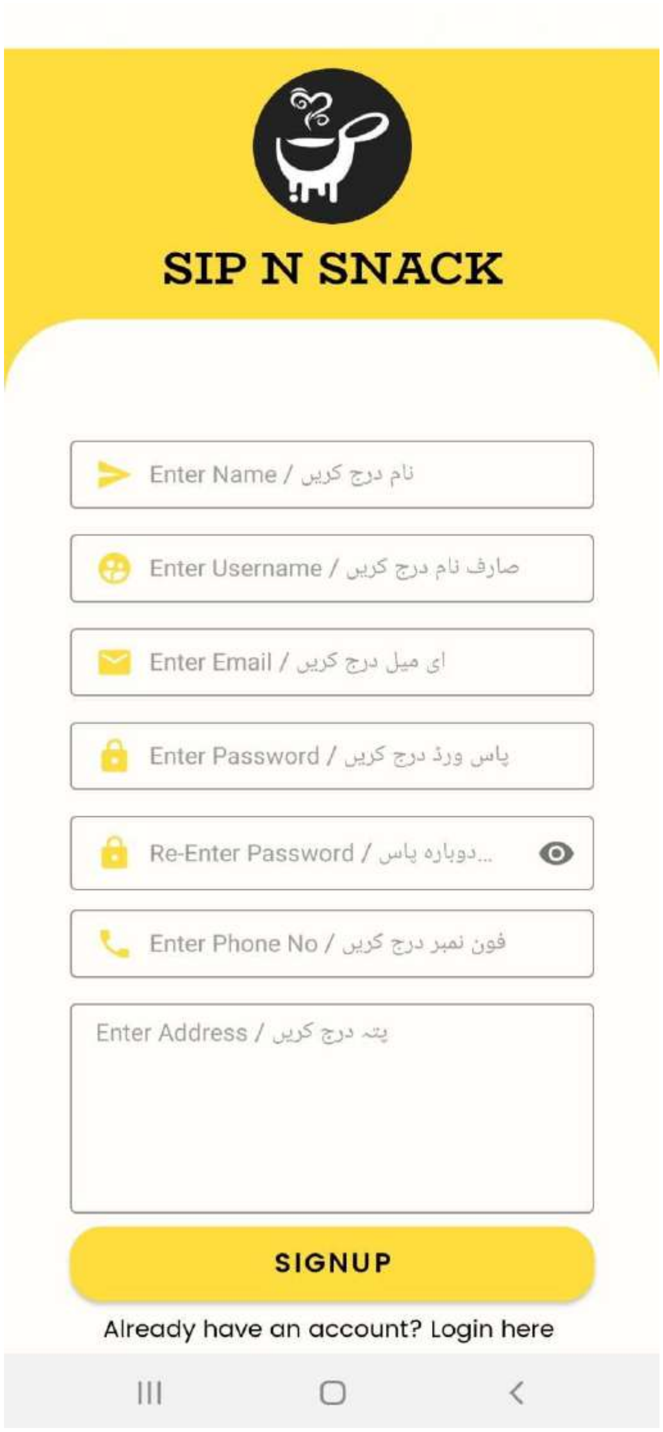

SIP N SNACK
Enter Name / نام درج کریں
Enter Username / صارف نام درج کریں
Enter Email / ای میل درج کریں
Enter Password / پاس ورڈ درج کریں
Re-Enter Password / دوبارہ پاس...
Enter Phone No / فون نمبر درج کریں
Enter Address / پتہ درج کریں
SIGNUP
Already have an account? Login here

Figure 95: Signup UI

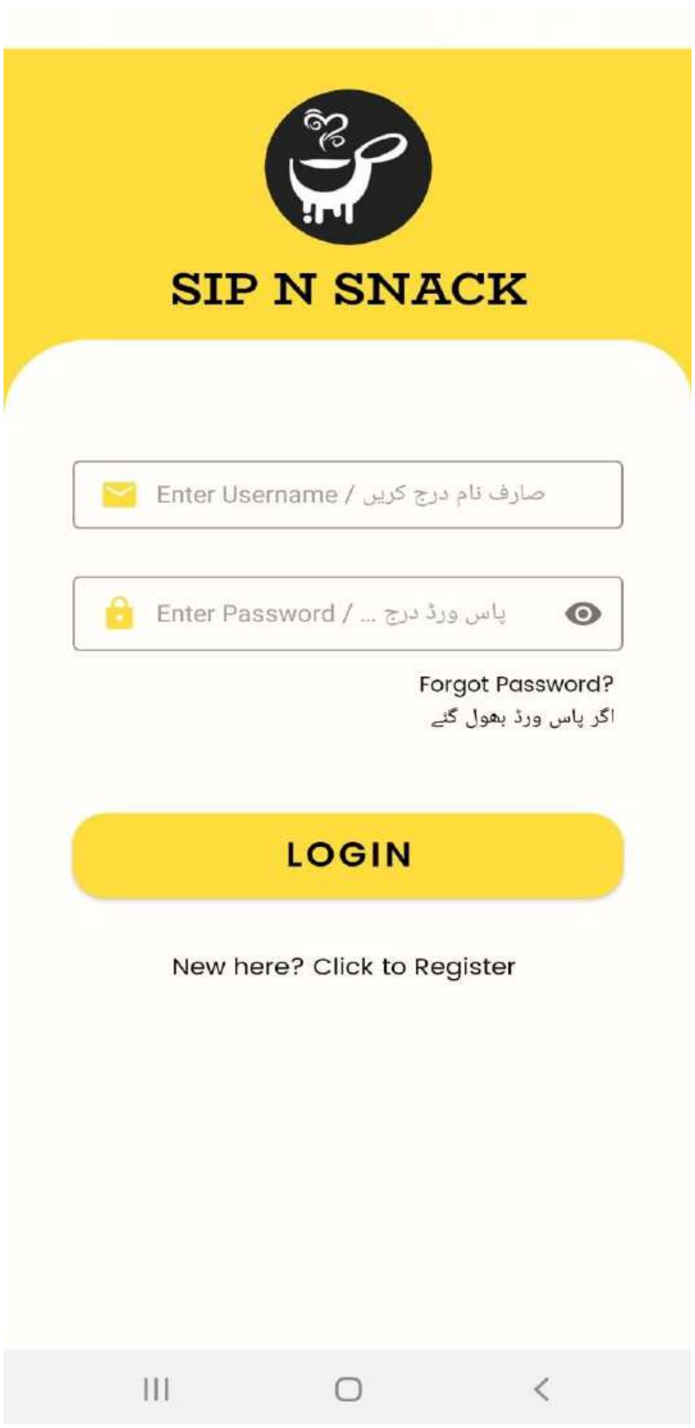

SIP N SNACK
Enter Username / صارف نام درج کریں
Enter Password / ... پاس ورڈ درج
Forgot Password?
اگر پاس ورڈ بھول گئے
LOGIN
New here? Click to Register

Figure 94: Login UI

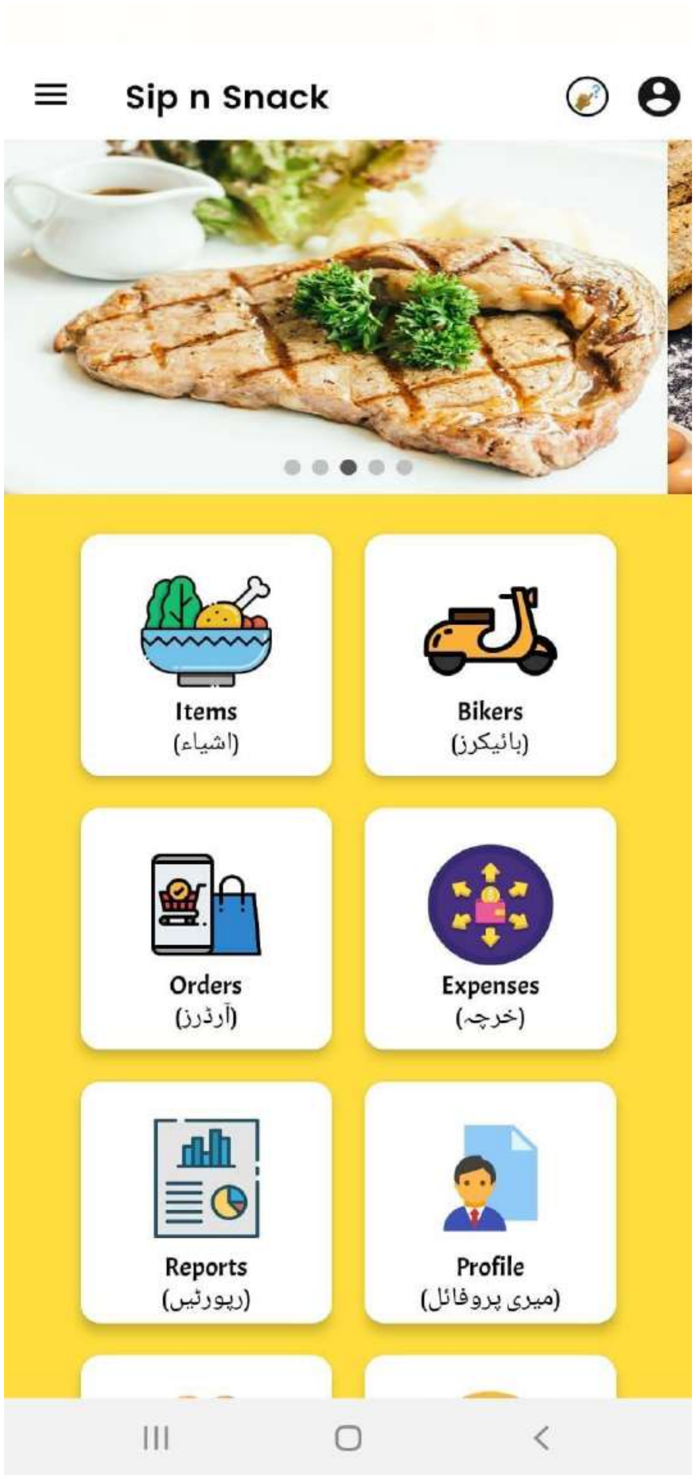


Figure 99: Dashboard UI

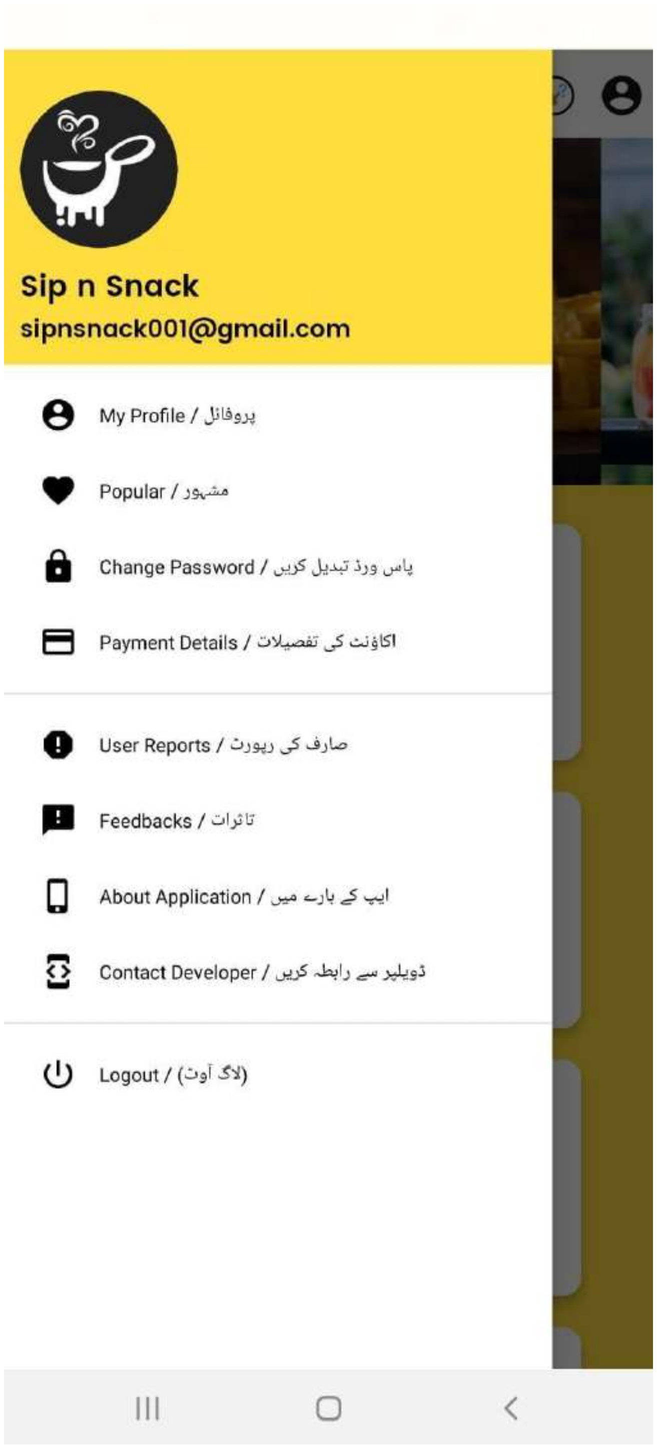
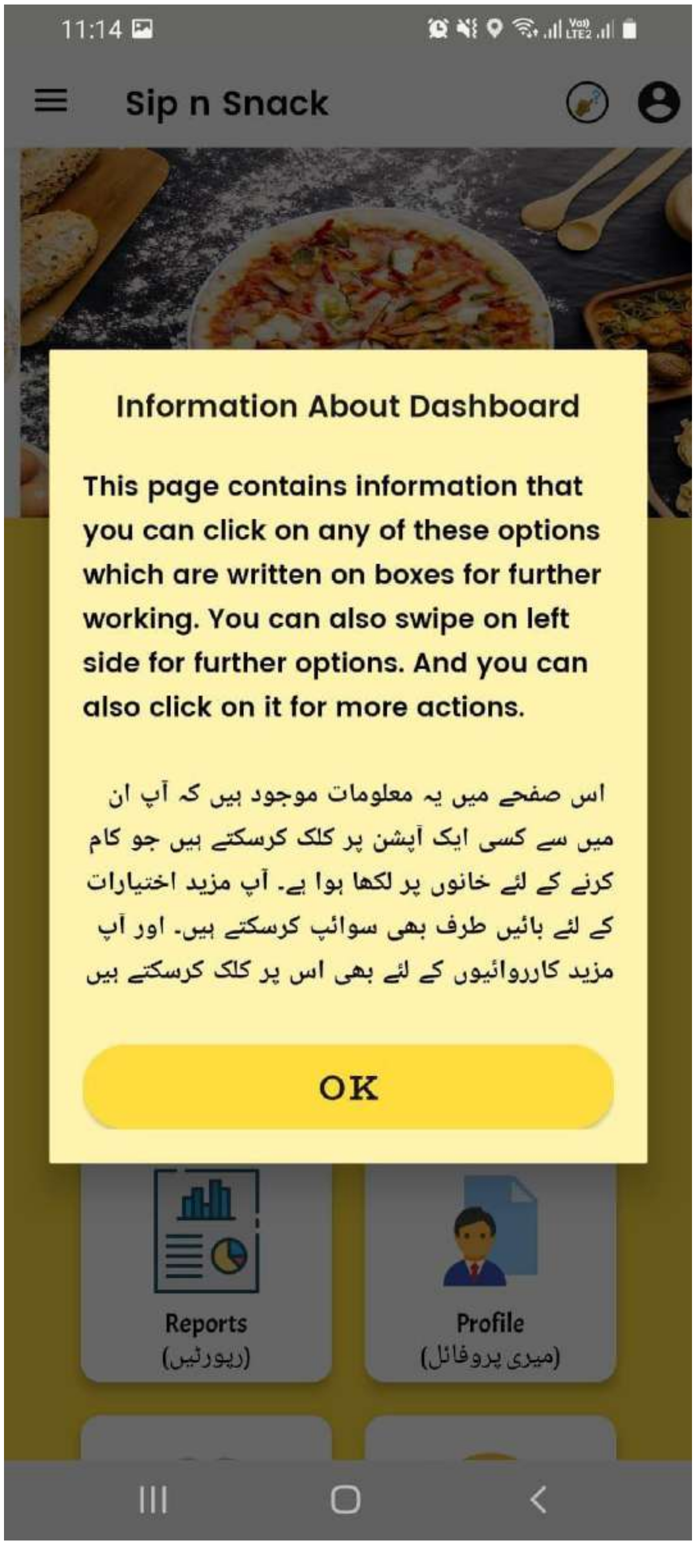

Sip n Snack
sipnsnack001@gmail.com
My Profile / پروفائل
Popular / مشہور
Change Password / پاس ورڈ تبدیل کریں
Payment Details / اکاؤنٹ کی تفصیلات
User Reports / صارف کی رپورٹ
Feedbacks / تاثرات
About Application / ایپ کے بارے میں
Contact Developer / ڈویلپر سے رابطہ کریں
Logout / (لاگ آوٹ)

Figure 98: Navigation Drawer UI



11:14

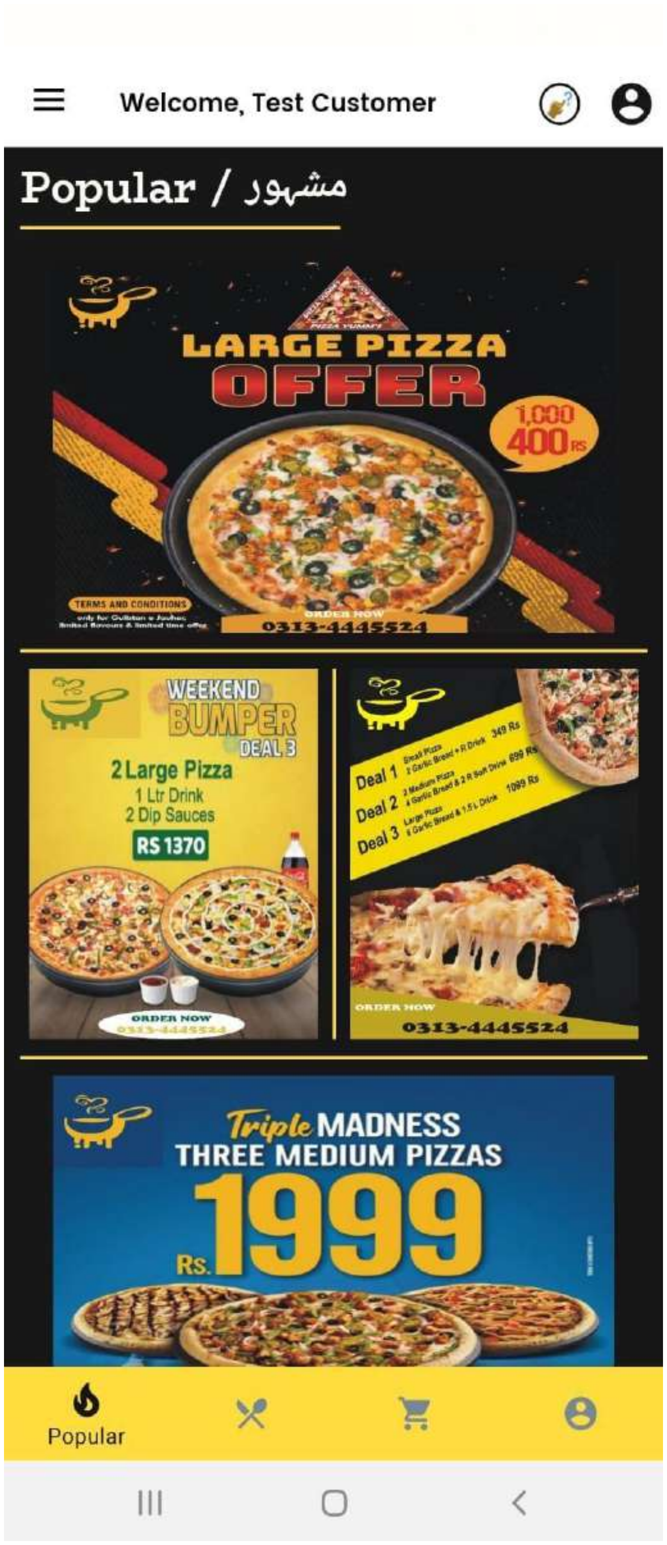
Sip n Snack

Information About Dashboard

This page contains information that you can click on any of these options which are written on boxes for further working. You can also swipe on left side for further options. And you can also click on it for more actions.

اس صفحے میں یہ معلومات موجود ہیں کہ آپ ان میں سے کسی ایک آپشن پر کلک کرسکتے ہیں جو کام کرنے کے لئے خانوں پر لکھا ہوا ہے۔ آپ مزید اختیارات کے لئے بائیں طرف بھی سوائپ کرسکتے ہیں۔ اور آپ مزید کارروائیوں کے لئے بھی اس پر کلک کرسکتے ہیں

Figure 100: About Dashboard



Welcome, Test Customer

Popular / مشہور

Figure 101: Customer View

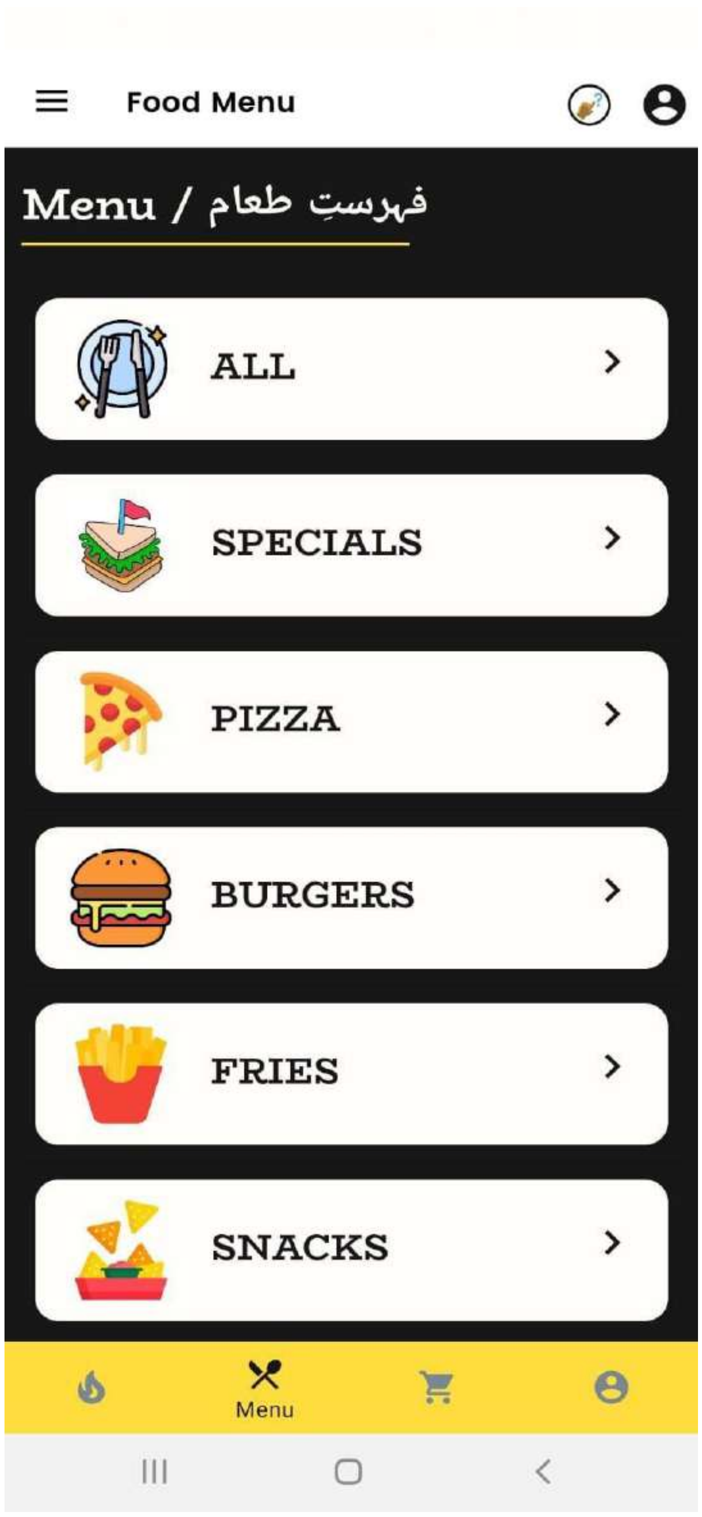
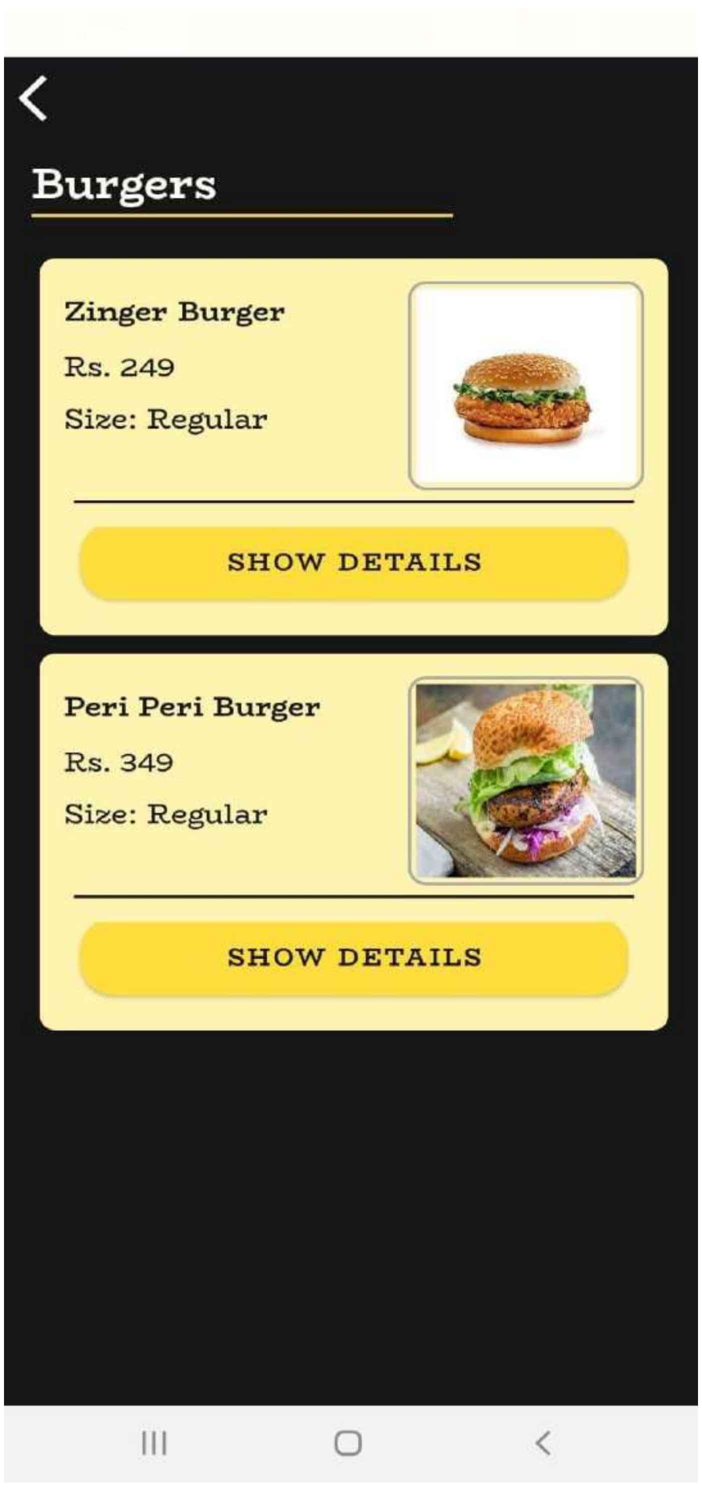


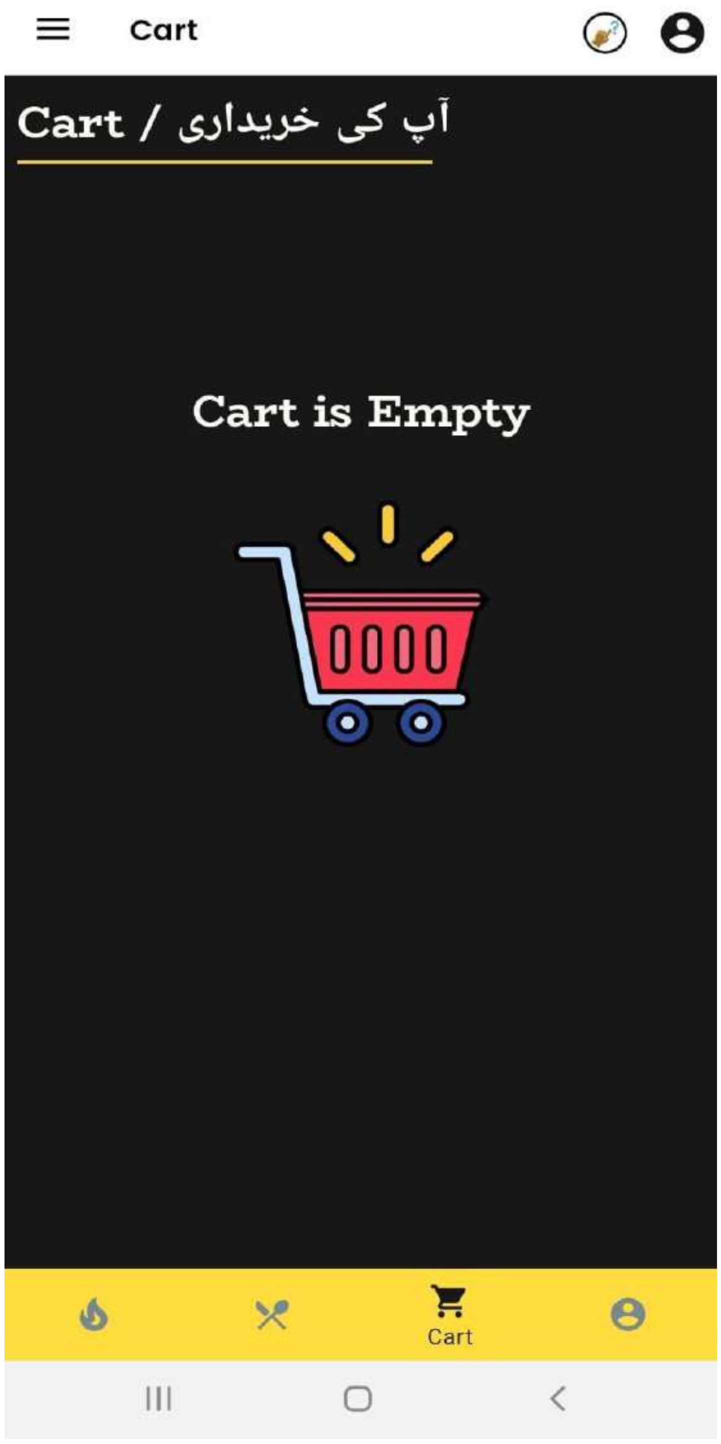
Figure 103: Menu UI

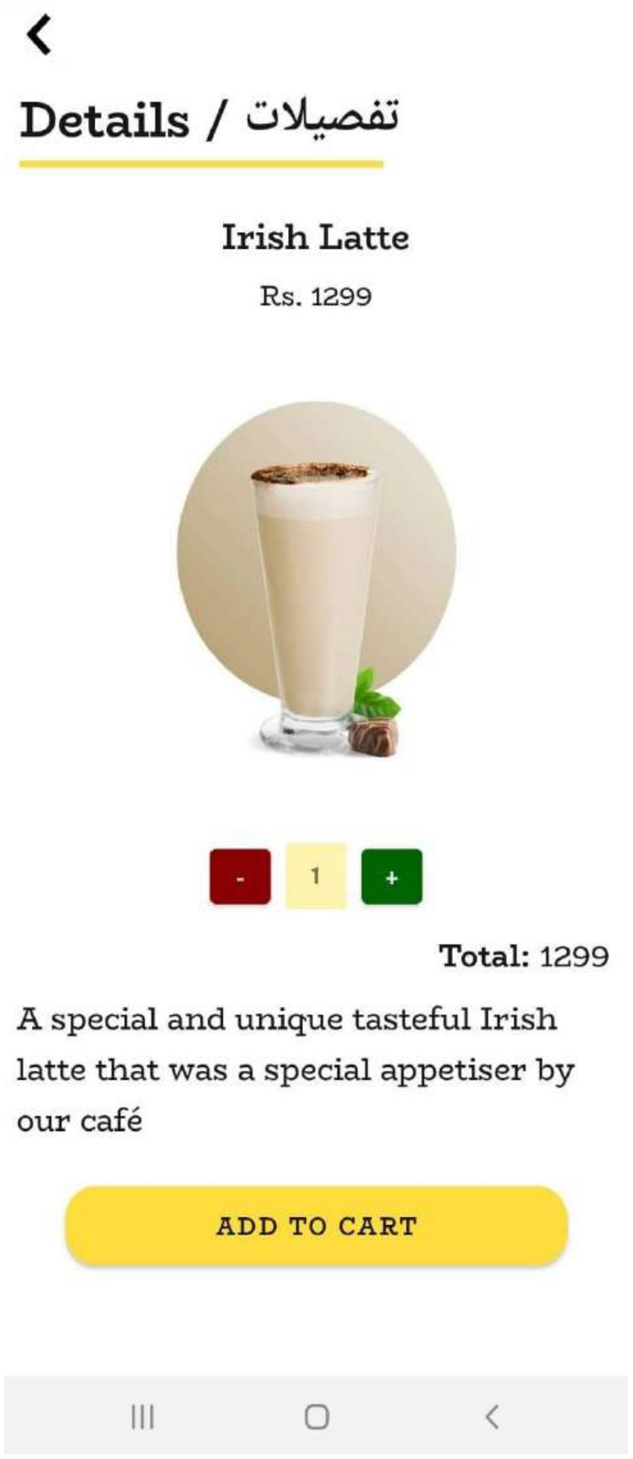


Zinger Burger
Rs. 249
Size: Regular

Peri Peri Burger
Rs. 349
Size: Regular

Figure 102: Menu Items UI

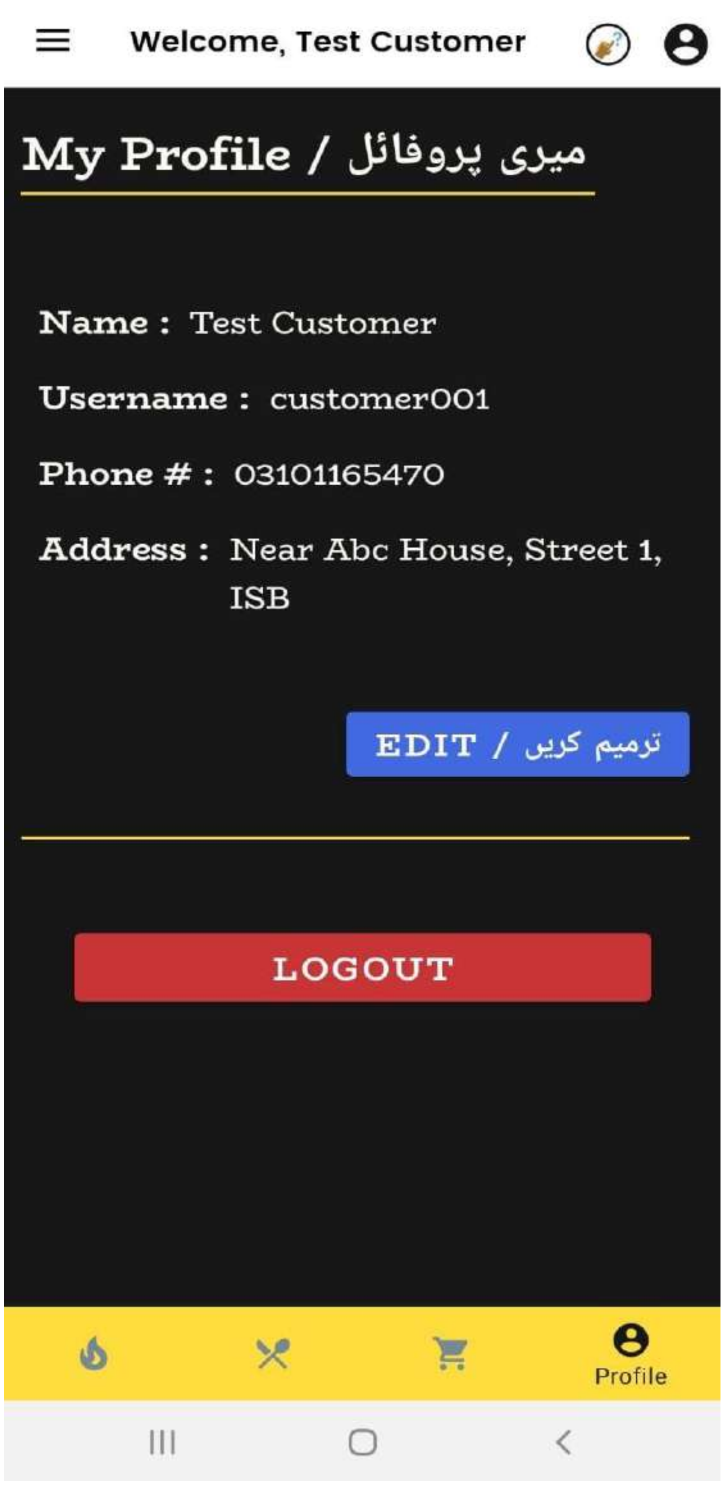




A special and unique tasteful Irish latte that was a special appetiser by our café

Figure 105: Cart UI

Figure 104: Item Details


Welcome, Test Customer

My Profile / میری پروفائل

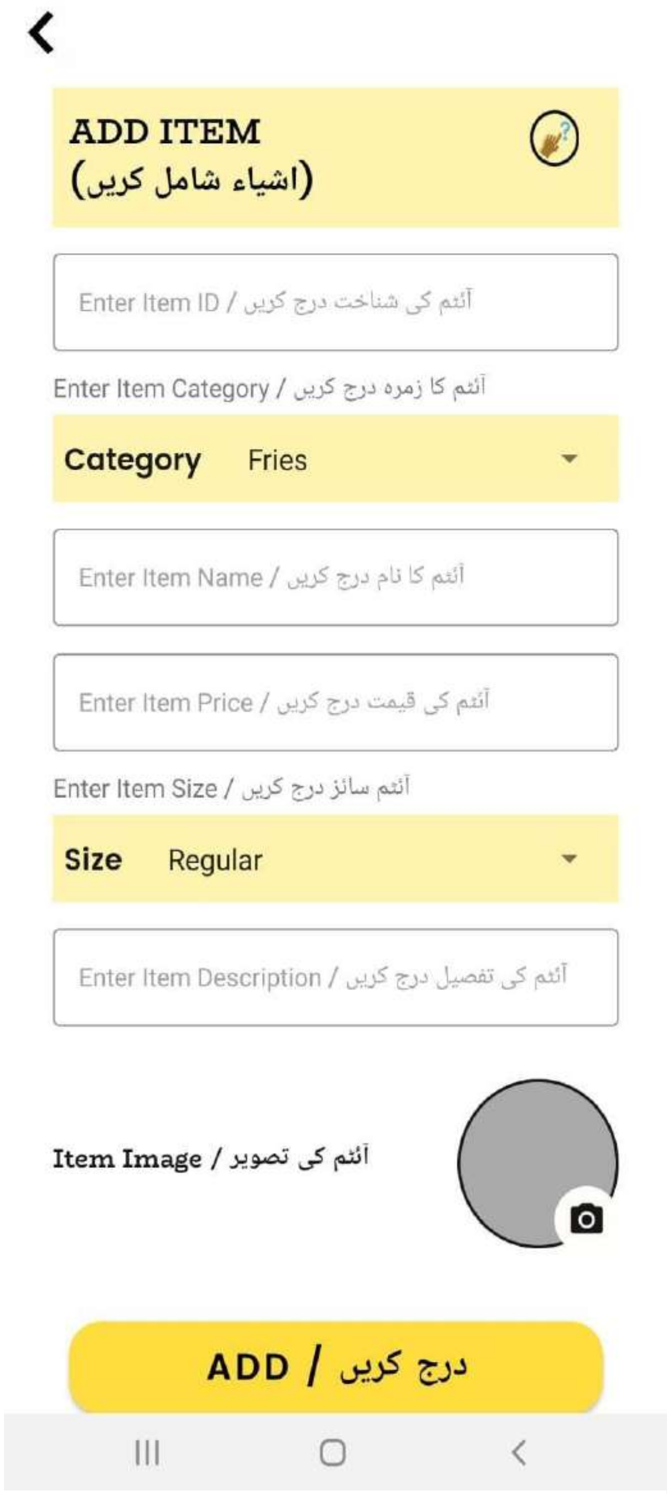
Name : Test Customer

Username : customer001

Phone # : 03101165470

Address : Near Abc House, Street 1, ISB

Figure 107: Profile UI


ADD ITEM
(اشیاء شامل کریں)

Enter Item ID / آئٹم کی شناخت درج کریں

Enter Item Category / آئٹم کا زمرہ درج کریں

Category Fries

Enter Item Name / آئٹم کا نام درج کریں

Enter Item Price / آئٹم کی قیمت درج کریں

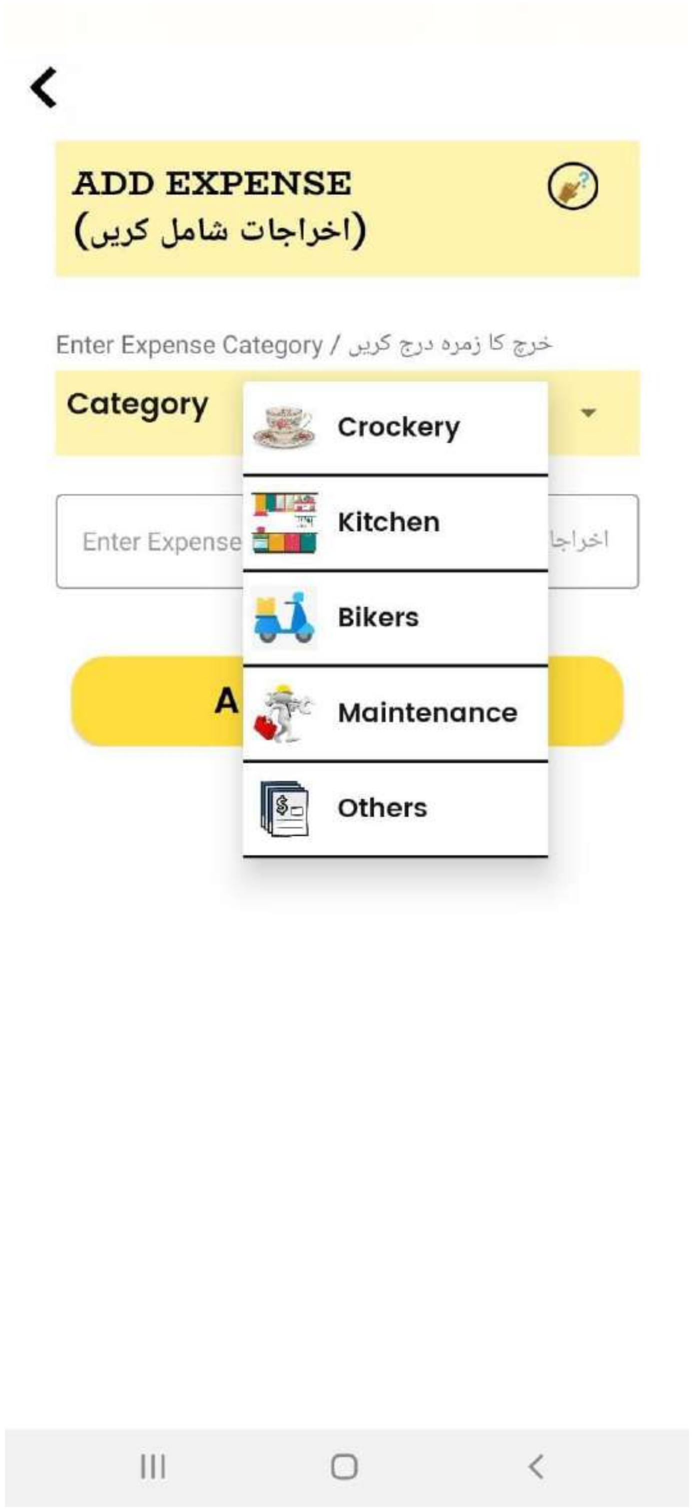
Enter Item Size / آئٹم سائز درج کریں

Size Regular

Enter Item Description / آئٹم کی تفصیل درج کریں

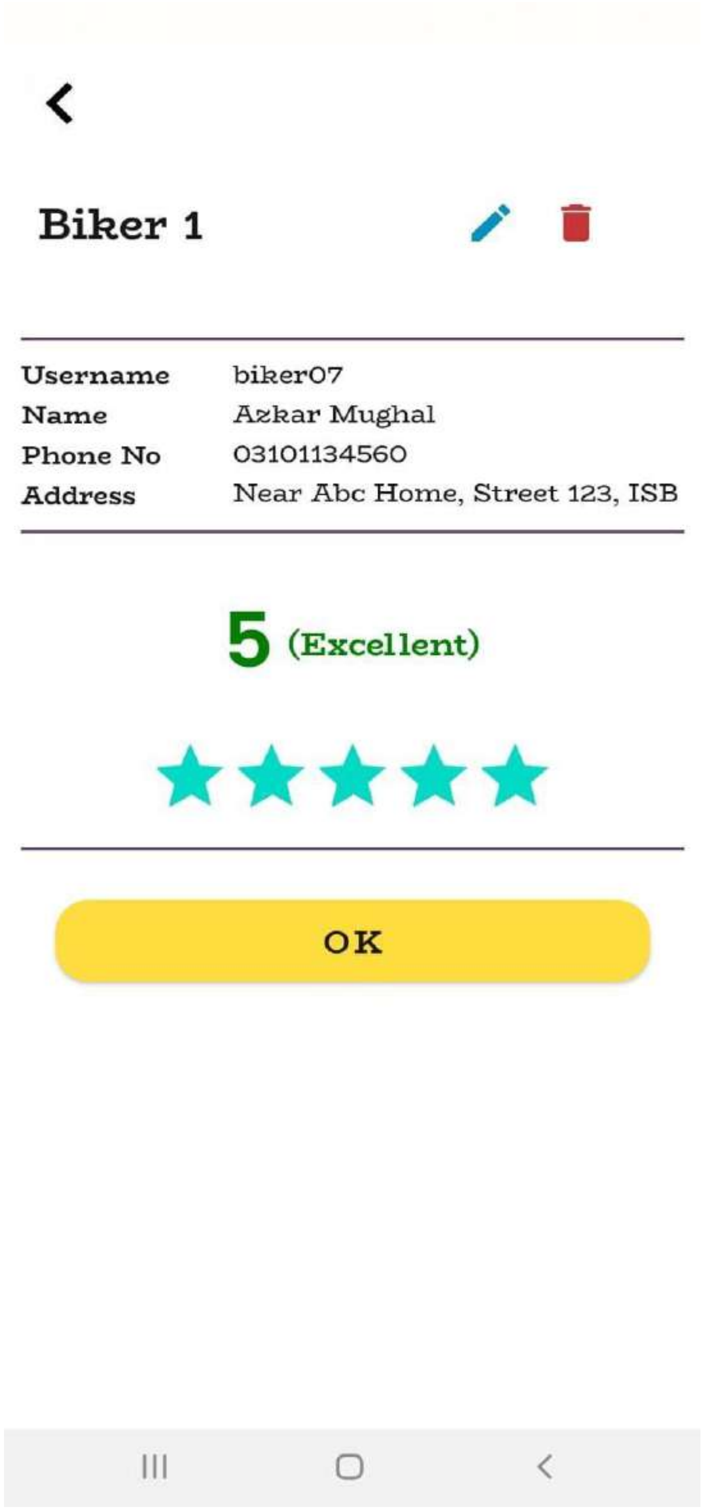
Item Image / آئٹم کی تصویر

Figure 106: Add Item UI


ADD EXPENSE
(اخراجات شامل کریں)

Enter Expense Category / خرچ کا زمرہ درج کریں

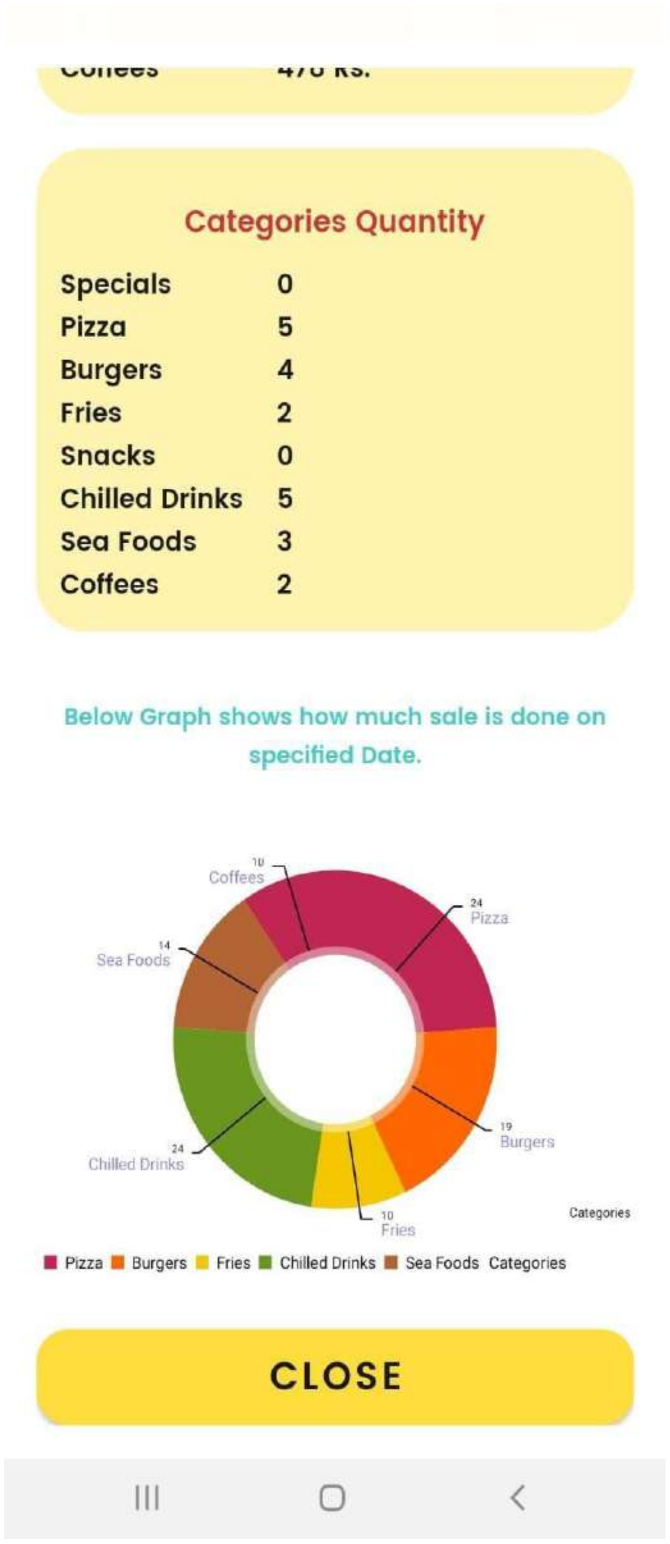
Figure 109: Add Expense


Biker 1

Username	biker07
Name	Azkar Mughal
Phone No	03101134560
Address	Near Abc Home, Street 123, ISB

5 (Excellent)

Figure 108: Biker Details

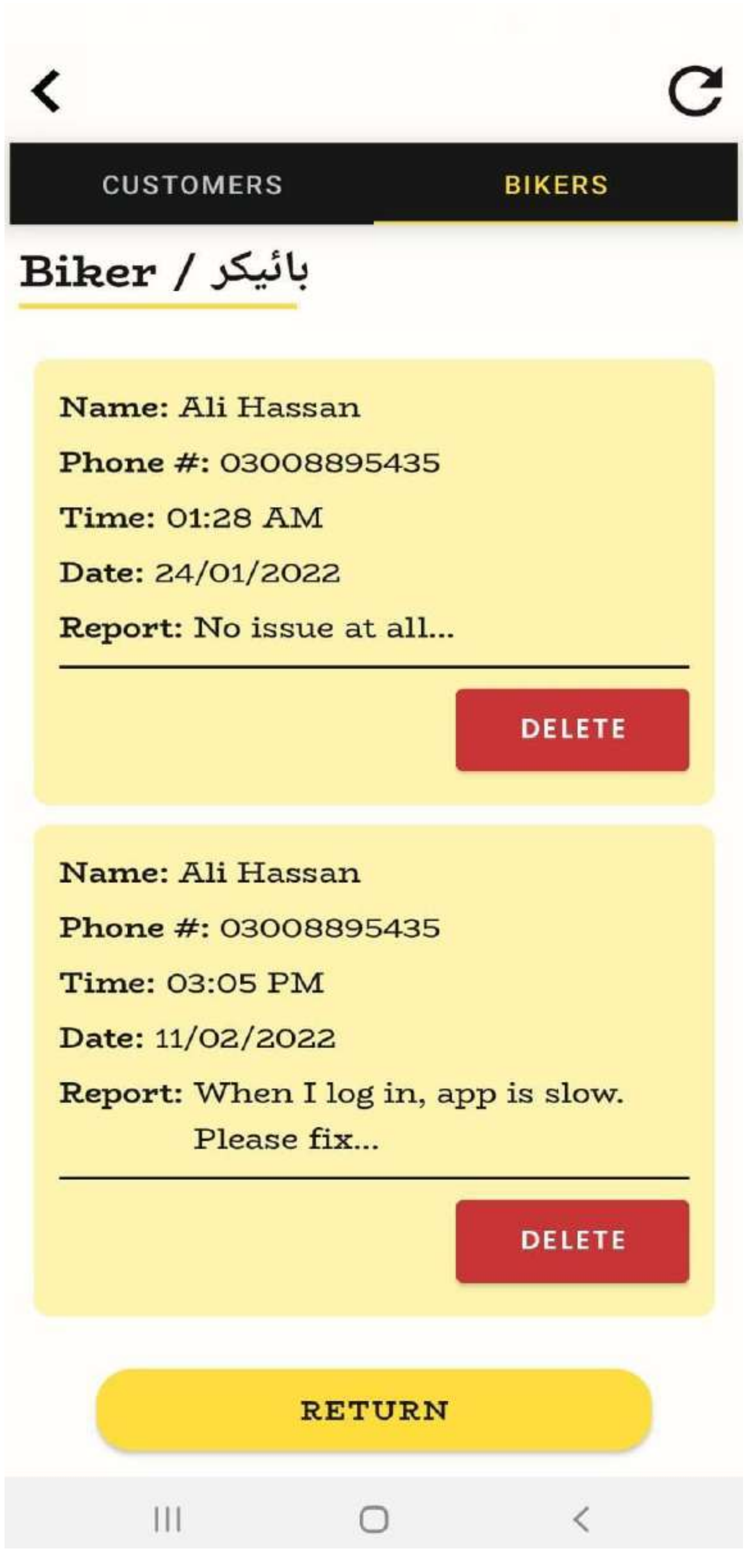


Categories Quantity

Specials	0
Pizza	5
Burgers	4
Fries	2
Snacks	0
Chilled Drinks	5
Sea Foods	3
Coffees	2

Below Graph shows how much sale is done on specified Date.

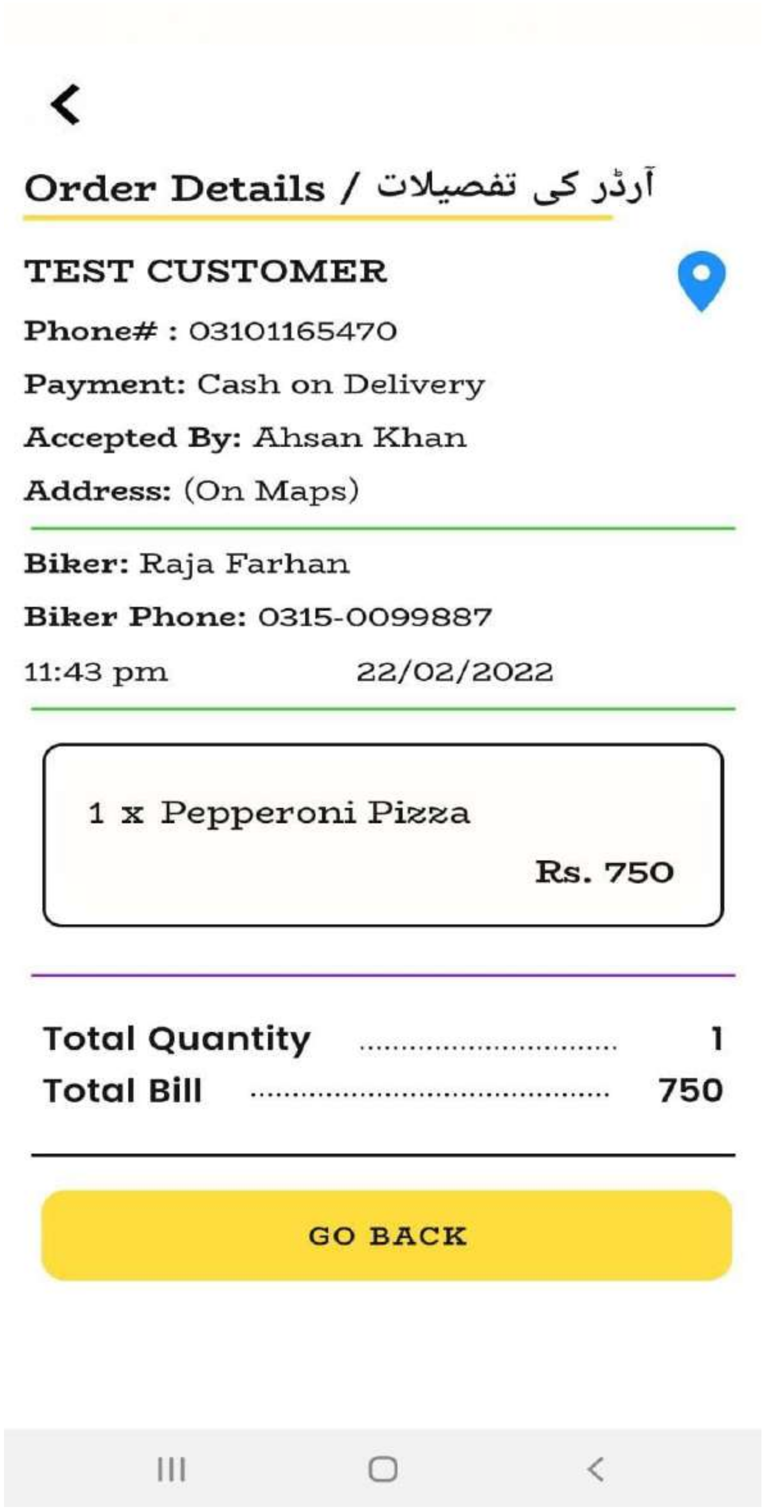
Figure 111: Sales Report



Name: Ali Hassan
Phone #: 03008895435
Time: 01:28 AM
Date: 24/01/2022
Report: No issue at all...

Name: Ali Hassan
Phone #: 03008895435
Time: 03:05 PM
Date: 11/02/2022
Report: When I log in, app is slow. Please fix...

Figure 110: View Issues UI



Order Details / آرڈر کی تفصیلات

TEST CUSTOMER

Phone# : 03101165470

Payment: Cash on Delivery

Accepted By: Ahsan Khan

Address: (On Maps)

Biker: Raja Farhan

Biker Phone: 0315-0099887

11:43 pm 22/02/2022

1 x Pepperoni Pizza

Rs. 750

Total Quantity	1
Total Bill	750

Figure 112: Bikers View

Chapter 4

Software Development

This chapter will provide the details about the coding standard, we adopted during implementation phase.

4.1. Coding Standards

The coding standard is described in the following subsection:

4.1.1. Indentation:

Four spaces are used as unit of indentation. The indentation pattern is followed consistently.

4.1.2. Declaration:

One declaration per line is used to enhance the clarity of code. The order and position of declaration is as follows:

- First the static/class variables are placed in the sequence: First public class variables, protected
- Instance variables are placed in the sequence: First public instance variables, protected
- Package level with no access modifier and then private
- Next the class constructors are declared

4.1.3. Statement Standards:

Each line contains at most one statement. While compound statements are statements that contain lists of statements enclosed in braces. The enclosed statements are indented one more level than the compound statement. The opening brace at the end of the line that begins the compound statement. The closing brace to begin a line and be indented to the beginning of the compound statement. Braces are used around all statements, even single statements, when they are part of a control structure, such as if-else or for statement. A Boolean expression / function is compared to a Boolean constant.

4.1.4. Naming Conventions:

Naming conventions make programs more understandable by making them easier to read. Following conventions are followed while naming a class or a member:

- We used full English descriptors that accurately describe the variable, method or class. Terminology applicable to the domain is used.
- Mixed case is used to make names readable with lower case letters in general capitalizing the first letter of class names and interface names.

4.2. Developing Environment:

Android Studio is the official integrated development environment (IDE) for the Android platform. It was announced on May 16, 2013 at the Google I/O conference. Android Studio is freely available under the Apache License 2.0.

The reason for using android studio was that we are going to develop an android based application. It also provides a very interactive and easy to understand interface to work with android devices. In this tool, User can test the written code on android device and that results in better outcomes.

Different services were made by us related our final year project in android studio that are currency recognition and menu recognition as well.

4.3. Software Description:

Our current selected modules are as follow:

- Manager
- Customers
- Bikers
- Items

Input (Manager):

In this module, after log in to the app, user can see different cards inside the recycler view and he/she select the specific card for which they want to perform action like managing items, managing bikers etc. Moreover the module contains the navigation drawer as well. Following is the code:

XML:

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/main_yellow"
    tools:context="com.cust.sipnsnack.ManagerDashboard.DashBoard">


    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:weightSum="3">


            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:orientation="vertical">

                <androidx.appcompat.widget.Toolbar
                    android:id="@+id/toolbar"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:background="@color/white"
                    app:subtitleTextColor="#0A0A0A"
                    app:title="Sip n Snack"
                    app:navigationIcon="@drawable/navbaricon"
                    app:titleTextColor="#111111" >

                    <LinearLayout
                        android:layout_width="match_parent"
                        android:layout_height="wrap_content">

                        <TextView
                            android:id="@+id/usernameTV"
                            android:layout_width="wrap_content"
                            android:layout_height="wrap_content"
                            android:fontFamily="@font/poppins_semibold"
                            android:textColor="@color/black"
                            android:textStyle="bold"
                            android:textSize="20dp"
                            android:text="Sip n Snack"
                            android:maxLength="15"
                            android:layout_weight="1"
                            android:layout_marginTop="5dp"
                            />

                        <LinearLayout
                            android:layout_width="wrap_content"
                            android:layout_height="wrap_content"
                            android:orientation="horizontal">

                            <ImageView
                                android:id="@+id/infoIconIVDashBoard"
                                android:layout_width="50dp"
                                android:layout_height="30dp"
                                android:layout_marginRight="3dp"
                                android:background="@drawable/blueinfologo"
                                android:layout_marginTop="5dp" />

                            <ImageView
                                android:id="@+id/imageiconIV"
                                android:layout_width="30dp"
                                android:layout_height="30dp"
                                android:layout_marginRight="5dp"
                                android:background="@drawable/profile"
                                android:layout_marginTop="5dp" />

                        </LinearLayout>

                    </LinearLayout>

                </androidx.appcompat.widget.Toolbar>

                <com.smarteist.autoimageslider.SliderView
                    android:id="@+id/sliderView"
                    android:layout_width="match_parent"
                    android:layout_height="220dp"
                    app:sliderAnimationDuration="1500"
                    app:sliderAutoCycleDirection="back_and_forth"
                    app:sliderIndicatorRadius="2dp"
                    app:sliderIndicatorSelectedColor="#5A5858"
                    app:sliderIndicatorUnselectedColor="#c1c1c1"
                    app:sliderScrollTimeInSec="1"
                    app:sliderStartAutoCycle="true" />

            </LinearLayout>


            <RelativeLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">
            </RelativeLayout>

            <GridLayout
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="15dp"
                android:alignmentMode="alignMargins"
                android:columnCount="2"
                android:columnOrderPreserved="false"
                android:rowCount="4">

                <androidx.cardview.widget.CardView
                    android:id="@+id/productCard"
                    android:layout_width="150dp"
                    android:layout_height="150dp"
                    android:layout_margin="10dp"
                    app:cardCornerRadius="12dp"
                    app:cardElevation="5dp">

                    <LinearLayout
                        android:layout_width="match_parent"
                        android:layout_height="match_parent"
                        android:layout_gravity="center"
                        android:background="@color/white"
                        android:gravity="center"
                        android:orientation="vertical" />

                    <ImageView
                        android:layout_width="80dp"
                        android:layout_height="80dp"
                        android:layout_gravity="center"
                        android:layout_marginBottom="15dp"
                        android:src="@drawable/food_ico" />


                    <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_gravity="center"
                        android:layout_marginTop="40dp"
                        android:layout_marginBottom="5dp"
                        android:fontFamily="@font/acme"
                        android:gravity="center"
                        android:text="Items"
                        android:textColor="@color/black"
                        android:textSize="16dp" />


                    <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_gravity="center"
                        android:layout_marginTop="60dp"
                        android:layout_marginBottom="5dp"
                        android:fontFamily="@font/acme"
                        android:gravity="center"
                        android:text="@string/foodItems"
                        android:textColor="@color/black"
                        android:textSize="16dp" />

                </androidx.cardview.widget.CardView>

                <androidx.cardview.widget.CardView
                    android:id="@+id/bikersCard"
                    android:layout_width="150dp"
                    android:layout_height="150dp"
                    android:layout_margin="10dp"
                    app:cardCornerRadius="12dp"
                    app:cardElevation="5dp">

                    <LinearLayout
                        android:layout_width="match_parent"
                        android:layout_height="match_parent"
                        android:layout_gravity="center"
                        android:background="@color/white"
                        android:orientation="vertical" />

                    <ImageView
                        android:layout_width="80dp"
                        android:layout_height="80dp"
                        android:layout_gravity="center"
                        android:layout_marginBottom="15dp"
                        android:src="@drawable/biker_ico" />


                    <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_gravity="center"
                        android:layout_marginTop="40dp"
                        android:layout_marginBottom="5dp"
                        android:fontFamily="@font/acme"
                        android:text="Bikers"
                        android:textColor="@color/black"
                        android:textSize="16dp" />

                    <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_gravity="center"
                        android:layout_marginTop="60dp"
                        android:layout_marginBottom="5dp"
                        android:fontFamily="@font/acme"
                        android:text="@string/bikers"
```

```
                        android:textColor="@color/black"
                        android:textSize="16dp"

                        />

                </androidx.cardview.widget.CardView>


                <androidx.cardview.widget.CardView
                    android:id="@+id/ordersCard"
                    android:layout_width="150dp"
                    android:layout_height="150dp"
                    android:layout_margin="10dp"
                    app:cardCornerRadius="12dp"
                    app:cardElevation="5dp">

                    <LinearLayout
                        android:layout_width="match_parent"
                        android:layout_height="match_parent"
                        android:layout_gravity="center"
                        android:background="@color/white"
                        android:orientation="vertical" />

                    <ImageView
                        android:layout_width="80dp"
                        android:layout_height="70dp"
                        android:layout_gravity="center"
                        android:layout_marginBottom="15dp"
                        android:src="@drawable/order_ico" />


                    <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_gravity="center"
                        android:layout_marginTop="40dp"
                        android:layout_marginBottom="5dp"
                        android:fontFamily="@font/acme"
                        android:text="Orders"
                        android:textColor="@color/black"
                        android:textSize="16dp" />

                    <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_gravity="center"
                        android:layout_marginTop="60dp"
                        android:layout_marginBottom="5dp"
                        android:fontFamily="@font/acme"
                        android:text="@string/orders"
                        android:textColor="@color/black"
                        android:textSize="16dp" />
```

```
        </androidx.cardview.widget.CardView>

        <androidx.cardview.widget.CardView
            android:id="@+id/expenseCard"
            android:layout_width="150dp"
            android:layout_height="150dp"
            android:layout_margin="10dp"
            app:cardCornerRadius="12dp"
            app:cardElevation="5dp">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_gravity="center"
                android:background="@color/white"
                android:orientation="vertical" />

            <ImageView
                android:layout_width="80dp"
                android:layout_height="80dp"
                android:layout_gravity="center"
                android:layout_marginBottom="15dp"
                android:src="@drawable/expense_ico" />


            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_marginTop="40dp"
                android:layout_marginBottom="5dp"
                android:fontFamily="@font/acme"
                android:text="Expenses"
                android:textColor="@color/black"
                android:textSize="16dp" />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_marginTop="60dp"
                android:layout_marginBottom="5dp"
                android:fontFamily="@font/acme"
                android:text="@string/expense"
                android:textColor="@color/black"
                android:textSize="16dp" />

        </androidx.cardview.widget.CardView>

        <androidx.cardview.widget.CardView
            android:id="@+id/reportsCard"
            android:layout_width="150dp"
            android:layout_height="150dp"
            android:layout_margin="10dp"
            app:cardCornerRadius="12dp"
            app:cardElevation="5dp">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_gravity="center"
                android:background="@color/white"
                android:orientation="vertical" />

            <ImageView
                android:layout_width="80dp"
                android:layout_height="80dp"
                android:layout_gravity="center"
                android:layout_marginBottom="15dp"
                android:src="@drawable/reports_ico" />


            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_marginTop="45dp"
                android:layout_marginBottom="5dp"
                android:fontFamily="@font/acme"
                android:text="Reports"
                android:textColor="@color/black"
                android:textSize="16dp" />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_marginTop="65dp"
                android:layout_marginBottom="5dp"
                android:fontFamily="@font/acme"
                android:text="@string/reports"
                android:textColor="@color/black"
                android:textSize="16dp" />

        </androidx.cardview.widget.CardView>


        <androidx.cardview.widget.CardView
            android:id="@+id/profileCard"
            android:layout_width="150dp"
            android:layout_height="150dp"
            android:layout_margin="10dp"
            app:cardCornerRadius="12dp"
            app:cardElevation="5dp">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_gravity="center"
                android:background="@color/white"
                android:orientation="vertical" />

            <ImageView
                android:layout_width="80dp"
                android:layout_height="75dp"
                android:layout_gravity="center"
                android:layout_marginBottom="15dp"
                android:src="@drawable/profile_icon" />


            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_marginTop="45dp"
                android:layout_marginBottom="5dp"
                android:fontFamily="@font/acme"
                android:text="Profile"
                android:textColor="@color/black"
                android:textSize="16dp" />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_marginTop="65dp"
                android:layout_marginBottom="5dp"
                android:fontFamily="@font/acme"
                android:text="@string/ProfileUrdu"
                android:textColor="@color/black"
                android:textSize="16dp" />

        </androidx.cardview.widget.CardView>



        <androidx.cardview.widget.CardView
            android:id="@+id/customersCard"
            android:layout_width="150dp"
            android:layout_height="150dp"
            android:layout_margin="10dp"
            app:cardCornerRadius="12dp"
            app:cardElevation="5dp">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_gravity="center"
                android:background="@color/white"
```

```
                android:orientation="vertical" />

            <ImageView
                android:layout_width="80dp"
                android:layout_height="80dp"
                android:layout_gravity="center"
                android:layout_marginBottom="15dp"
                android:src="@drawable/vendors" />


            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_marginTop="45dp"
                android:layout_marginBottom="5dp"
                android:fontFamily="@font/acme"
                android:text="Customers"
                android:textColor="@color/black"
                android:textSize="16dp" />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_marginTop="65dp"
                android:layout_marginBottom="5dp"
                android:fontFamily="@font/acme"
                android:text="@string/manageCustomer"
                android:textColor="@color/black"
                android:textSize="16dp" />

        </androidx.cardview.widget.CardView>


        <androidx.cardview.widget.CardView
            android:id="@+id/logoutCard"
            android:layout_width="150dp"
            android:layout_height="150dp"
            android:layout_margin="10dp"
            app:cardCornerRadius="12dp"
            app:cardElevation="5dp">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_gravity="center"
                android:background="@color/white"
                android:orientation="vertical" />

            <ImageView
                android:layout_width="80dp"
                android:layout_height="75dp"
```

```
                        android:layout_gravity="center"
                        android:layout_marginBottom="15dp"
                        android:src="@drawable/logout_ico" />


                    <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_gravity="center"
                        android:layout_marginTop="45dp"
                        android:layout_marginBottom="5dp"
                        android:fontFamily="@font/acme"
                        android:text="Logout"
                        android:textColor="@color/black"
                        android:textSize="16dp" />

                    <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_gravity="center"
                        android:layout_marginTop="65dp"
                        android:layout_marginBottom="5dp"
                        android:fontFamily="@font/acme"
                        android:text="@string/logout"
                        android:textColor="@color/black"
                        android:textSize="16dp" />

                </androidx.cardview.widget.CardView>

            </GridLayout>

        </LinearLayout>

    </ScrollView>

    <com.google.android.material.navigation.NavigationView
        android:id="@+id/navView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:background="@color/white"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/header_nav"
        app:itemIconTint="@color/black"
        app:itemTextAppearance="@style/HintSize"
        app:itemTextColor="@color/black"
        app:menu="@menu/main_menu" />


</androidx.drawerlayout.widget.DrawerLayout>
```

Java Snippet:

```java
package com.cust.sipnsnack.ManagerDashboard;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.cardview.widget.CardView;
import androidx.drawerlayout.widget.DrawerLayout;

import com.cust.sipnsnack.Bikers.ManageBikers;
import com.cust.sipnsnack.Items.ManageItems;
import com.cust.sipnsnack.LoginActivity;
import com.example.sipnsnack.R;
import com.google.android.material.navigation.NavigationView;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import
com.smarteist.autoimageslider.IndicatorView.animation.type.IndicatorAnimationTyp
e;
import com.smarteist.autoimageslider.SliderAnimations;
import com.smarteist.autoimageslider.SliderView;


public class DashBoard extends AppCompatActivity {

    NavigationView navView;
    private int[] images;
    private SliderAdapter adapter;
    private SliderView sliderView;
    TextView usernameTV;
    DrawerLayout drawer;
    ActionBarDrawerToggle actionBarDrawerToggle;
    CardView productsCard, bikersCard, ordersCard, expenseCard, reportsCard,
profileCard,
            customersCard, logoutCard;
    SharedPreferences sharedPreferences;
    ImageView infoIcon, profileIcon;
    static String pd, ac, ot, dl;
    int p, a, o, d;
    LoadingDialog loadingDialog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dash_board);

        loadingDialog = new LoadingDialog(DashBoard.this);

        setUpToolbar();

        drawer = findViewById(R.id.drawer_layout);
        usernameTV = findViewById(R.id.usernameTV);

        sharedPreferences = getSharedPreferences("LoginSPR", MODE_PRIVATE);

        loadingDialog.startLoadingDialog();
        getOrdersCount();

        productsCard = findViewById(R.id.productCard);
        bikersCard = findViewById(R.id.bikersCard);
        ordersCard = findViewById(R.id.ordersCard);
        expenseCard = findViewById(R.id.expenseCard);
        reportsCard = findViewById(R.id.reportsCard);
        profileCard = findViewById(R.id.profileCard);
        customersCard = findViewById(R.id.customersCard);
        logoutCard = findViewById(R.id.logoutCard);
        navView = findViewById(R.id.navView);
        infoIcon = findViewById(R.id.infoIconIVDashBoard);
        profileIcon = findViewById(R.id.imageiconIV);

        if (drawer.isDrawerOpen(navView)) {
            drawer.closeDrawer(navView);
        }

        infoIcon.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                showDialog();
            }
        });


        navView.setNavigationItemSelectedListener(new
NavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item) {

                if(item.getItemId() == R.id.nav_profile) {
                    if(drawer.isDrawerOpen(navView)) {
                        drawer.closeDrawer(navView);
                        showProfileDialog();
                    }
                }

                if(item.getItemId() == R.id.nav_about_app) {
                    if(drawer.isDrawerOpen(navView)) {
                        drawer.closeDrawer(navView);
                        aboutAppDialog();
                    }
                }

                if(item.getItemId() == R.id.nav_contactDev) {
                    if(drawer.isDrawerOpen(navView)) {
                        drawer.closeDrawer(navView);
                        developerDialog();
                    }
                }

                if(item.getItemId() == R.id.nav_logout) {
                    if(drawer.isDrawerOpen(navView)) {
                        drawer.closeDrawer(navView);
                        logoutDialog();
                    }
                }


                if(item.getItemId() == R.id.nav_popular) {
                    if(drawer.isDrawerOpen(navView)) {
                        drawer.closeDrawer(navView);
                        Intent it = new Intent(getApplicationContext(),
PopularItems.class);
                        startActivity(it);
                        finish();
                    }
                }


                if(item.getItemId() == R.id.nav_change_password) {
                    if(drawer.isDrawerOpen(navView)) {
                        drawer.closeDrawer(navView);
                        Intent it = new Intent(getApplicationContext(),
ManagerPasswordChange.class);
                        startActivity(it);
                        finish();
                    }
                }

                if(item.getItemId() == R.id.nav_reports) {
                    if(drawer.isDrawerOpen(navView)) {
                        drawer.closeDrawer(navView);
                        Intent it = new Intent(getApplicationContext(),
SystemReports.class);
                        startActivity(it);
                        finish();
                    }
                }

                if(item.getItemId() == R.id.nav_feedbacks) {
                    if(drawer.isDrawerOpen(navView)) {
                        drawer.closeDrawer(navView);
                        Intent it = new Intent(getApplicationContext(),
UserFeedbacks.class);
                        startActivity(it);
                        finish();
                    }
                }

                if(item.getItemId() == R.id.nav_online_payments) {
                    if(drawer.isDrawerOpen(navView)) {
                        drawer.closeDrawer(navView);
                        Intent it = new Intent(getApplicationContext(),
PaymentAccountDetails.class);
                        startActivity(it);
                        finish();
                    }
                }

                return false;
            }
        });


        // Product Card
        productsCard.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent it = new Intent(getApplicationContext(),
ManageItems.class);
                startActivity(it);
                finish();
            }
        });

        // Bikers Card
        bikersCard.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent it = new Intent(getApplicationContext(),
ManageBikers.class);
                startActivity(it);
                finish();
            }
        });
        // Orders Card
        ordersCard.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent it = new Intent(getApplicationContext(),
AllOrdersDetails.class);
                startActivity(it);
                finish();
            }
        });


        // Expense Card
        expenseCard.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(getApplicationContext(),
ExpenseOption.class));
                finish();
            }
        });


        // Reports Card
        reportsCard.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(getApplicationContext(),
ViewReports.class));
                finish();
            }
        });


        // Profile Card
        profileCard.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                showProfileDialog();
            }
        });


        // Customers Card
        customersCard.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                Intent it = new Intent(getApplicationContext(),
ManageCustomers.class);
                startActivity(it);
                finish();
            }
        });


        // Logout Card
        logoutCard.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                logoutDialog();
            }
        });

        profileIcon.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                showProfileDialog();
            }
        });

        images = new int[]{R.drawable.slider1, R.drawable.slider2,
R.drawable.slider3
                , R.drawable.slider4, R.drawable.slider5};
        sliderView = findViewById(R.id.sliderView);

        adapter = new SliderAdapter(images);
        sliderView.setSliderAdapter(adapter);

sliderView.setSliderTransformAnimation(SliderAnimations.SIMPLETRANSFORMATION);
        sliderView.setIndicatorAnimation(IndicatorAnimationType.DROP);
        sliderView.startAutoCycle();

    }

    public void setUpToolbar() {
        drawer = findViewById(R.id.drawer_layout);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        actionBarDrawerToggle = new ActionBarDrawerToggle(this, drawer, toolbar,
R.string.app_name, R.string.app_name);
        drawer.addDrawerListener(actionBarDrawerToggle);

actionBarDrawerToggle.getDrawerArrowDrawable().setColor(getResources().getColor(
R.color.black));
        actionBarDrawerToggle.syncState();

    }


    public void showProfileDialog() {
        Intent it = new Intent(getApplicationContext(), ManagerProfile.class);
        startActivity(it);
        finish();
    }


    public void developerDialog() {
        LayoutInflater layoutInflater = LayoutInflater.from(this);
        View view = layoutInflater.inflate(R.layout.developer_dialog, null);
        Button okBTN = view.findViewById(R.id.okBTN);
        final AlertDialog alertDialog = new
AlertDialog.Builder(this).setView(view).create();
        alertDialog.show();
        alertDialog.setCancelable(false);
        okBTN.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                alertDialog.dismiss();
            }
        });
    }


    public void aboutAppDialog() {
        LayoutInflater layoutInflater = LayoutInflater.from(this);
        View view = layoutInflater.inflate(R.layout.about_app_dialog, null);
        Button okBTN = view.findViewById(R.id.okBTN);
        final AlertDialog alertDialog = new
AlertDialog.Builder(this).setView(view).create();
        alertDialog.show();
        alertDialog.setCancelable(false);
        okBTN.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                alertDialog.dismiss();
            }
        });
    }


    public void logoutDialog() {

        LayoutInflater layoutInflater = LayoutInflater.from(this);
        View view = layoutInflater.inflate(R.layout.ask_logout_dialog, null);
        Button yesBTN = view.findViewById(R.id.yesBTN);
        Button noBTN = view.findViewById(R.id.noBTN);
        final AlertDialog alertDialog = new
AlertDialog.Builder(this).setView(view).create();
        alertDialog.show();
        yesBTN.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.putString("User", "");
                editor.putString("Status", "");
                editor.putString("Username", "");
                editor.apply();


                Intent intent = new Intent(getApplicationContext(),
LoginActivity.class);
                startActivity(intent);
                finish();
                Toast.makeText(getApplicationContext(), "Logout Successful ...",
Toast.LENGTH_SHORT).show();
                alertDialog.dismiss();
            }
        });

        noBTN.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                alertDialog.dismiss();
            }
        });

    }


    public void showDialog() {
        LayoutInflater layoutInflater = LayoutInflater.from(this);
        View view = layoutInflater.inflate(R.layout.dialog_info_dashboard,
null);
        Button okBTN = view.findViewById(R.id.okBTN);

        final AlertDialog alertDialog = new
AlertDialog.Builder(this).setView(view).create();
        alertDialog.show();

        okBTN.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                alertDialog.dismiss();
            }
        });
    }

    public void wipDialog() {
        LayoutInflater layoutInflater = LayoutInflater.from(this);
        View view = layoutInflater.inflate(R.layout.work_in_process, null);
        Button okBTN = view.findViewById(R.id.okBTN);
        final AlertDialog alertDialog = new
AlertDialog.Builder(this).setView(view).create();
        alertDialog.show();
        alertDialog.setCancelable(false);
        okBTN.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                alertDialog.dismiss();
            }
        });
    }

    @Override
    public void onBackPressed() {
        if (drawer.isDrawerOpen(navView)) {
            drawer.closeDrawer(navView);
        }
        Intent intent = new Intent(Intent.ACTION_MAIN);
        intent.addCategory(Intent.CATEGORY_HOME);
        startActivity(intent);
    }


    public void getOrdersCount() {


FirebaseDatabase.getInstance().getReference().child("Orders").child("Pending")
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    private DataSnapshot dataSnapshot;

                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                        this.dataSnapshot = dataSnapshot;

                        if (dataSnapshot.exists()) {
                            for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                                p++;
                            }
                            pd = "PND ("+p+")";
                        } else {
                            pd = "PND (0)";
                        }
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError error) {

                    }
                });




FirebaseDatabase.getInstance().getReference().child("Orders").child("Accepted")
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    private DataSnapshot dataSnapshot;

                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                        this.dataSnapshot = dataSnapshot;

                        if (dataSnapshot.exists()) {
                            for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                                a++;
                            }
                            ac = "ACT ("+a+")";
                        } else {
                            ac = "ACT (0)";
                        }
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError error) {

                    }
                });



        FirebaseDatabase.getInstance().getReference().child("Orders").child("On
the Way")
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    private DataSnapshot dataSnapshot;

                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                        this.dataSnapshot = dataSnapshot;

                        if (dataSnapshot.exists()) {
                            for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                                o++;
                            }
                            ot = "OTW ("+o+")";
                        } else {
                            ot = "OTW (0)";
                        }
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError error) {

                    }
                });



FirebaseDatabase.getInstance().getReference().child("Orders").child("Delivered")
```

```
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    private DataSnapshot dataSnapshot;

                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                        this.dataSnapshot = dataSnapshot;

                        if (dataSnapshot.exists()) {
                            for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                                d++;
                            }
                            dl = "DLV ("+d+")";
                        } else {
                            dl = "DLV (0)";
                        }

                        loadingDialog.dismissDialog();
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError error) {

                    }
                });
    }
}
```

Description:

This activity consist of navigation drawer and different cards inside cards layout. Each card has its own respective title and on clicking that specific category, system will redirect manager to another screen. In navigation drawer, manager can be able to see its profile, info about app, logout from account option etc.

Output:

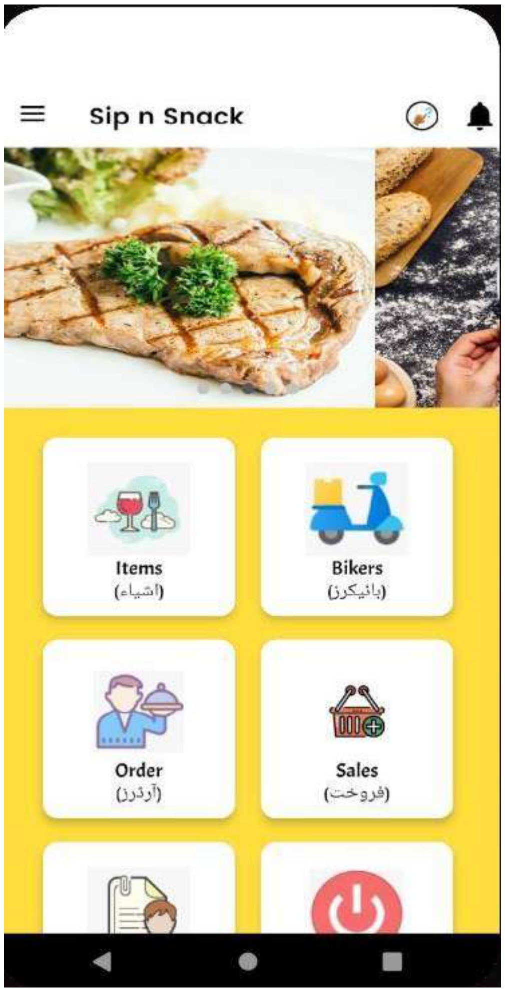


Figure 113: Dashboard Screen

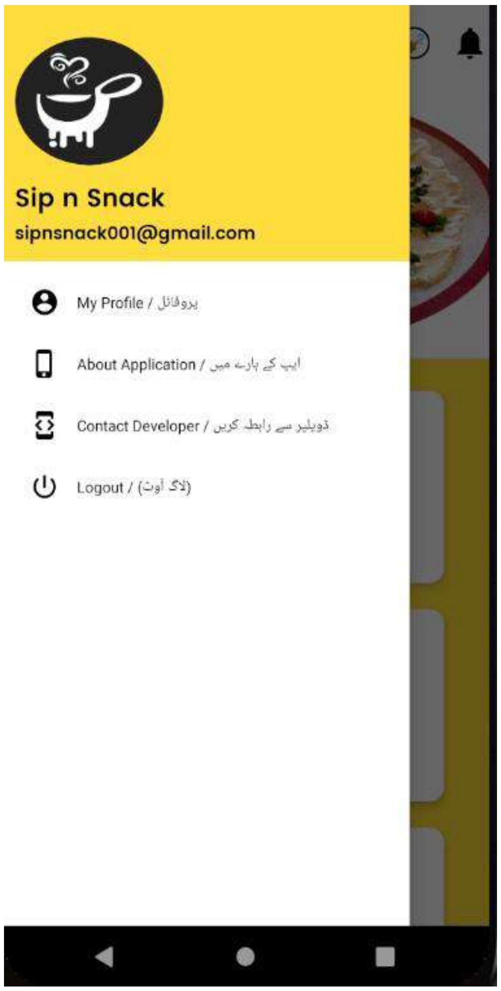

Sip n Snack
sipnsnack001@gmail.com
My Profile / پروفائل
About Application / ایپ کے بارے میں
Contact Developer / ڈویلپر سے رابطہ کریں
Logout / (لاگ آوٹ)

Figure 114: Output Navigation Drawer

Input (Bikers):

This module contains information about the assigned order information. The activity named ‘Bikers View’ consist of all necessary information needed for a biker to deliver the order to customer. There is a button of confirm delivery, after the biker confirms the delivry of order, the order is being converted to delivered state and the respective information were saved into the system data-base.

The code snippet for this module is given below:

XML:

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/half_white"
    tools:context="cust.food_delivery.sipnsnack.Bikers.BikersView">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:weightSum="3">


            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:orientation="vertical">

                <androidx.appcompat.widget.Toolbar
                    android:id="@+id/toolbar"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:background="@color/silver"
                    app:subtitleTextColor="#0A0A0A"
                    app:title="Sip n Snack"
                    app:navigationIcon="@drawable/navbaricon"
                    app:titleTextColor="#111111" >

                    <LinearLayout
                        android:layout_width="match_parent"
                        android:layout_height="wrap_content"
                        android:orientation="horizontal">

                        <TextView
                            android:id="@+id/usernameTV"
                            android:layout_width="wrap_content"
                            android:layout_height="wrap_content"
                            android:fontFamily="@font/poppins_semibold"
                            android:textColor="@color/black"
                            android:textStyle="bold"
                            android:textSize="20dp"
                            android:text="Sip n Snack"
                            android:maxLength="15"
                            android:layout_weight="1"
                            android:layout_marginTop="5dp"
                            />

                        <LinearLayout
                            android:layout_width="wrap_content"
                            android:layout_height="wrap_content"
                            android:orientation="horizontal">

                            <ImageView
                                android:id="@+id/infoIconIV"
                                android:layout_width="50dp"
                                android:layout_height="30dp"
                                android:layout_marginRight="3dp"
                                android:background="@drawable/blueinfologo"
                                android:layout_marginTop="5dp" />

                            <ImageView
                                android:id="@+id/profileIconIV"
                                android:layout_width="30dp"
                                android:layout_height="30dp"
                                android:layout_marginRight="5dp"
                                android:background="@drawable/profile"
                                android:layout_marginTop="5dp" />

                        </LinearLayout>

                    </LinearLayout>

                </androidx.appcompat.widget.Toolbar>

                <View
```

```
                android:layout_width="match_parent"
                android:layout_height="1dp"
                android:background="@color/matteBlack"/>


            <RelativeLayout
                android:id="@+id/mainRL"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="6dp"
                android:layout_marginLeft="7dp"
                android:layout_marginRight="7dp"
                android:layout_marginBottom="15dp">


                <LinearLayout
                    android:id="@+id/mainLL"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="8dp"
                    android:orientation="vertical">


                    <TextView
                        android:layout_width="match_parent"
                        android:layout_height="wrap_content"
                        android:text="@string/assignedOrder"
                        android:textSize="20sp"
                        android:fontFamily="@font/biorhyme_bold"
                        android:textColor="@color/matteBlack"
                        android:layout_marginTop="10dp"/>

                    <View
                        android:layout_width="350dp"
                        android:layout_height="3dp"
                        android:layout_marginTop="2dp"
                        android:layout_marginBottom="10dp"
                        android:background="@color/dark_magenta" />


                    <TextView
                        android:id="@+id/noOrdersTV"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="No Assigned Orders at the Moment"
                        android:textSize="18sp"
                        android:layout_gravity="center"
                        android:visibility="gone"
                        android:fontFamily="@font/poppins_semibold"
                        android:textColor="@color/matteBlack"
                        android:layout_marginTop="120dp"/>

                    <ImageView
```

```
        android:id="@+id/noOrdersIV"
        android:layout_width="180dp"
        android:layout_height="180dp"
        android:layout_marginTop="50dp"
        android:layout_gravity="center"
        android:src="@drawable/ic_no_order"
        android:visibility="gone"/>

</LinearLayout>

<RelativeLayout
    android:id="@+id/allInfoRL"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:visibility="invisible"
    android:layout_marginLeft="6dp"
    android:layout_marginRight="5dp"
    android:layout_marginTop="6dp"
    android:layout_below="@+id/mainLL">

<RelativeLayout
    android:id="@+id/infoLL"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

<TextView
    android:id="@+id/customerNameTV"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="NAEEM"
    android:textSize="18sp"
    android:layout_marginTop="10dp"
    android:textAllCaps="true"
    android:layout_marginBottom="8dp"
    android:fontFamily="@font/biorhyme_bold"
    android:textColor="@color/matteBlack"/>


    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginRight="2dp"
        android:layout_marginTop="6dp"
        android:layout_alignParentRight="true">


        <ImageView
            android:id="@+id/phoneCallIV"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:layout_marginRight="8dp"
            android:src="@drawable/ic_phone_call" />
        <ImageView
            android:id="@+id/locationIV"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:src="@drawable/ic_location" />

    </LinearLayout>



<LinearLayout
    android:id="@+id/LL1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/customerNameTV"
    android:orientation="horizontal">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Phone# : "
        android:textSize="16sp"
        android:fontFamily="@font/biorhyme_bold"
        android:textColor="@color/matteBlack"/>

    <TextView
        android:id="@+id/customerPhoneNoTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0331-6788090"
        android:textSize="16sp"
        android:fontFamily="@font/biorhyme_regular"
        android:textColor="@color/matteBlack"/>

</LinearLayout>


<LinearLayout
    android:id="@+id/LL2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/LL1"
    android:layout_marginTop="6dp"
    android:orientation="horizontal">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Payment: "
        android:textSize="16sp"
        android:fontFamily="@font/biorhyme_bold"
```

```
                android:textColor="@color/matteBlack"/>

            <TextView
                android:id="@+id/paymentTypeTV"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Online"
                android:textSize="16sp"
                android:fontFamily="@font/biorhyme_regular"
                android:textColor="@color/matteBlack"/>

        </LinearLayout>


        <LinearLayout
            android:id="@+id/LL5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/LL2"
            android:layout_marginTop="6dp"
            android:orientation="horizontal">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Address: "
                android:textSize="16sp"
                android:fontFamily="@font/biorhyme_bold"
                android:textColor="@color/matteBlack"/>

            <TextView
                android:id="@+id/customerAddressTV"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="ABC TOWN ISB"
                android:layout_marginRight="4dp"
                android:textSize="16sp"
                android:fontFamily="@font/biorhyme_regular"
                android:textColor="@color/matteBlack"/>

        </LinearLayout>

        </RelativeLayout>

        <View
            android:id="@+id/view4"
            android:layout_width="match_parent"
            android:layout_height="2dp"
            android:layout_marginLeft="4dp"
            android:layout_marginRight="4dp"
            android:layout_marginTop="12dp"
            android:layout_below="@+id/infoLL"
            android:background="@color/slate_gray" />
```

```xml
<LinearLayout
    android:id="@+id/dateTimeLL"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_below="@+id/view4"
    android:weightSum="2"
    android:layout_marginTop="7dp">

    <TextView
        android:id="@+id/timeTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Time : "
        android:textSize="16sp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="4dp"
        android:fontFamily="@font/biorhyme_regular"
        android:layout_weight="1"
        android:textColor="@color/matteBlack"/>

    <TextView
        android:id="@+id/dateTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Date : "
        android:layout_marginRight="6dp"
        android:textSize="16sp"
        android:fontFamily="@font/biorhyme_regular"
        android:layout_weight="1"
        android:textColor="@color/matteBlack"/>

</LinearLayout>

<View
    android:id="@+id/view5"
    android:layout_width="match_parent"
    android:layout_height="2dp"
    android:layout_marginLeft="4dp"
    android:layout_marginRight="4dp"
    android:layout_marginTop="12dp"
    android:layout_below="@+id/dateTimeLL"
    android:background="@color/slate_gray" />


<LinearLayout
    android:id="@+id/LL4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/view5"
    android:visibility="visible"
    android:layout_marginTop="15dp"
```

```
                        android:orientation="vertical">


                        <TextView
                            android:layout_width="wrap_content"
                            android:layout_height="wrap_content"
                            android:layout_marginLeft="6dp"
                            android:text="Items / اشیاء"
                            android:textSize="16sp"
                            android:fontFamily="@font/poppins_semibold"
                            android:textColor="@color/maroon"/>

                        <View
                            android:layout_width="110dp"
                            android:layout_height="2dp"
                            android:layout_marginLeft="6dp"
                            android:background="@color/goldenrod"/>


                        <RelativeLayout
                            android:layout_width="match_parent"
                            android:layout_height="match_parent"
                            android:layout_marginTop="15dp"
                            android:layout_marginBottom="15dp">


                            <androidx.recyclerview.widget.RecyclerView
                                android:id="@+id/assigned_items_recycler_view"
                                android:layout_width="match_parent"
                                android:layout_height="match_parent"
                                android:layout_marginLeft="6dp"
                                android:layout_marginRight="6dp"
                                android:scrollbars="vertical" />


                            <View
                                android:id="@+id/view1"
                                android:layout_width="match_parent"
                                android:layout_height="2dp"
                                android:layout_marginLeft="10dp"
                                android:layout_marginRight="10dp"
                                android:layout_marginTop="25dp"

android:layout_below="@+id/assigned_items_recycler_view"
                                android:background="@color/matteBlack" />


                            <RelativeLayout
                                android:id="@+id/RL1"
                                android:layout_width="match_parent"
                                android:layout_height="wrap_content"
                                android:layout_marginTop="22dp"
                                android:layout_below="@+id/view1">
                                <TextView
                                    android:id="@+id/qty"
                                    android:layout_width="wrap_content"
                                    android:layout_height="wrap_content"
                                    android:text="Total Quantity"
                                    android:fontFamily="@font/poppins_semibold"
                                    android:layout_marginLeft="10dp"
                                    android:textSize="18sp"
                                    android:textColor="@color/matteBlack"/>


                                <TextView
                                    android:id="@+id/qty2"
                                    android:layout_width="wrap_content"
                                    android:layout_height="wrap_content"

android:text="............................. "
                                    android:layout_toRightOf="@+id/qty"
                                    android:textColor="@color/matteBlack"
                                    android:layout_marginTop="3dp"
                                    android:layout_marginLeft="25dp"
                                    android:textSize="16sp"/>


                                <TextView
                                    android:id="@+id/totalQtyTV"
                                    android:layout_width="wrap_content"
                                    android:layout_height="wrap_content"
                                    android:text="6"
                                    android:layout_alignParentRight="true"
                                    android:layout_marginRight="10dp"
                                    android:fontFamily="@font/poppins_semibold"
                                    android:layout_marginLeft="10dp"
                                    android:textSize="18sp"
                                    android:textColor="@color/matteBlack"/>


                                <TextView
                                    android:id="@+id/qty3"
                                    android:layout_width="wrap_content"
                                    android:layout_height="wrap_content"
                                    android:text="Total Bill"
                                    android:layout_below="@+id/qty"
                                    android:fontFamily="@font/poppins_semibold"
                                    android:layout_marginLeft="10dp"
                                    android:textSize="18sp"
                                    android:textColor="@color/matteBlack"/>


                                <TextView
                                    android:id="@+id/qty4"
                                    android:layout_width="wrap_content"
                                    android:layout_height="wrap_content"

android:text="........................................... "
                                    android:layout_toRightOf="@+id/qty3"
                                    android:layout_below="@+id/qty2"
                                    android:layout_marginTop="5dp"
                                    android:textColor="@color/matteBlack"
                                    android:layout_marginLeft="25dp"
                                    android:textSize="16sp"/>


                                <TextView
                                    android:id="@+id/totalPriceTV"
                                    android:layout_width="wrap_content"
                                    android:layout_height="wrap_content"
                                    android:text="1200"
                                    android:layout_alignParentRight="true"
                                    android:layout_marginRight="10dp"
                                    android:layout_below="@+id/totalQtyTV"
                                    android:fontFamily="@font/poppins_semibold"
                                    android:layout_marginLeft="10dp"
                                    android:textSize="18sp"
                                    android:textColor="@color/matteBlack"/>

                            </RelativeLayout>


                            <View
                                android:id="@+id/view2"
                                android:layout_width="match_parent"
                                android:layout_height="2dp"
                                android:layout_marginLeft="10dp"
                                android:layout_marginRight="10dp"
                                android:layout_marginTop="22dp"
                                android:layout_below="@+id/RL1"
                                android:background="@color/matteBlack" />


                            <LinearLayout
                                android:layout_width="match_parent"
                                android:layout_height="wrap_content"
                                android:orientation="horizontal"
                                android:layout_below="@+id/view2"
                                android:layout_marginTop="20dp">

                                <Button
                                    android:id="@+id/confirmDeliveryBTN"
                                    android:layout_width="match_parent"
                                    android:layout_height="58dp"
                                    android:text="Confirm Delivery"
                                    android:layout_marginLeft="18dp"
                                    android:layout_marginRight="18dp"
                                    android:textSize="18sp"
                                    android:backgroundTint="@color/dark_green"
                                    android:textColor="@color/half_white"
                                    android:fontFamily="@font/poppins_semibold"
/>


                            </LinearLayout>


                        </RelativeLayout>


                    </LinearLayout>

                    </RelativeLayout>

                </RelativeLayout>


            </LinearLayout>

        </LinearLayout>

    </ScrollView>


    <com.google.android.material.navigation.NavigationView
        android:id="@+id/navView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:background="@color/white"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/header_nav"
        app:itemIconTint="@color/black"
        app:itemTextAppearance="@style/HintSize"
        app:itemTextColor="@color/black"
        app:menu="@menu/biker_nav" />


</androidx.drawerlayout.widget.DrawerLayout>
```

Java Snippet:

```java
package cust.food_delivery.sipnsnack.Bikers;

import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.content.Intent;
import android.content.SharedPreferences;
import cust.food_delivery.sipnsnack.ManagerDashboard.AcceptedOrderItems;
import cust.food_delivery.sipnsnack.ManagerDashboard.AcceptedOrderItemsAdapter;
import cust.food_delivery.sipnsnack.ManagerDashboard.MyAcceptedOrderData;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.core.app.NotificationCompat;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.recyclerview.widget.DefaultItemAnimator;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import cust.food_delivery.sipnsnack.LoginActivity;

import com.example.sipnsnack.R;
import com.google.android.material.navigation.NavigationView;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
```

```java
public class BikersView extends AppCompatActivity {

    NavigationView navView;
    DrawerLayout drawer;
    ActionBarDrawerToggle actionBarDrawerToggle;
    SharedPreferences spr;
    ImageView infoIcon, profileIV, locationIV;
    RelativeLayout mainRL;
    TextView noOrderTV;
    ImageView phoneCallIV, noOrderIV;
    int loopSize2;
    LoadingDialog loadingDialog;
    TextView customerNameTV, customerPhoneNoTV, customerAddressTV,
paymentTypeTV, totalQtyTV,
            totalPriceTV, dateTV, timeTV;

    String addressType, lat, lon;

    private FirebaseDatabase mDatabase = FirebaseDatabase.getInstance();
    private DatabaseReference mDatabaseRef = mDatabase.getReference();

    private FirebaseDatabase mDatabase2 = FirebaseDatabase.getInstance();
    private DatabaseReference mDatabaseRef2 = mDatabase.getReference();

    Boolean flag = false;
    String username, name, phone, address, payment, qty, price, bikerUsername,
            dbBikerUSN, acceptedBy, bikerName, bikerPhone, receipt;
    private RecyclerView.LayoutManager layoutManager;
    private static RecyclerView recyclerView;
    int loopSize;
    String iD, nAme, pRice, cAtegory, dEscription, sIze, uRl, qTy, tOtal,
billTotal, orderId;
    ArrayList<AcceptedOrderItems> myItem;
    AcceptedOrderItemsAdapter acceptedOrderItemsAdapter;
    Button confirmDelivery;
    DatabaseReference orderDeliverNodeRef;
    int specials, pizza, burgers, fries, snacks, chilled_drinks, sea_foods,
coffees, net_sale;
    int specialsAmt, pizzaAmt, burgersAmt, friesAmt, snacksAmt,
chilled_drinksAmt, sea_foodsAmt, coffeesAmt;
    String sp, pi, bu, fr, sn, ch, se, co, ns;
    String spAmt, piAmt, buAmt, frAmt, snAmt, chAmt, seAmt, coAmt, nsAmt;
    String itemCategory, itemQuantity, total, itemTotal;

    public static String getTime() {
        DateFormat dateFormat = new SimpleDateFormat("hh:mm aa");
        String dateString = dateFormat.format(new Date()).toString();
        return dateString;
    }

    public static String getTodayDate() {
        Date date;
        DateFormat setDate;
        date = Calendar.getInstance().getTime();
        setDate = new SimpleDateFormat("dd/MM/yyyy");
        String current_date = setDate.format(date);
        return current_date;
    }

    public static String getFormatDate() {
        Date date;
        DateFormat setDate;
        date = Calendar.getInstance().getTime();
        setDate = new SimpleDateFormat("dd_MM_yyyy");
        String current_date = setDate.format(date);
        return current_date;
    }

    public static String getMonth() {
        String Month;
        Date date = new Date();
        int month = date.getMonth();
        month += 1;

        Month = "January";
        if (month == 1) {
            Month = "January";
        } else if (month == 2) {
            Month = "February";
        } else if (month == 3) {
            Month = "March";
        } else if (month == 4) {
            Month = "April";
        } else if (month == 5) {
            Month = "May";
        } else if (month == 6) {
            Month = "June";
        } else if (month == 7) {
            Month = "July";
        } else if (month == 8) {
            Month = "August";
        } else if (month == 9) {
            Month = "September";
        } else if (month == 10) {
            Month = "October";
        } else if (month == 11) {
            Month = "November";
        } else if (month == 12) {
            Month = "December";
        }

        return Month;
    }

    public static String getYear() {
        int year = Calendar.getInstance().get(Calendar.YEAR);
        return String.valueOf(year);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_bikers_view);

        loadingDialog = new LoadingDialog(BikersView.this);

        setUpToolbar();

        navView = findViewById(R.id.navView);
        infoIcon = findViewById(R.id.infoIconIV);
        mainRL = findViewById(R.id.allInfoRL);
        noOrderTV = findViewById(R.id.noOrdersTV);

        customerNameTV = findViewById(R.id.customerNameTV);
        customerPhoneNoTV = findViewById(R.id.customerPhoneNoTV);
        customerAddressTV = findViewById(R.id.customerAddressTV);
        paymentTypeTV = findViewById(R.id.paymentTypeTV);
        totalQtyTV = findViewById(R.id.totalQtyTV);
        totalPriceTV = findViewById(R.id.totalPriceTV);
        dateTV = findViewById(R.id.dateTV);
        timeTV = findViewById(R.id.timeTV);
        confirmDelivery = findViewById(R.id.confirmDeliveryBTN);
        profileIV = findViewById(R.id.profileIconIV);
        phoneCallIV = findViewById(R.id.phoneCallIV);
        noOrderIV = findViewById(R.id.noOrdersIV);
        locationIV = findViewById(R.id.locationIV);

        timeTV.setText(getTime());
        dateTV.setText(getTodayDate());

        recyclerView = (RecyclerView)
findViewById(R.id.assigned_items_recycler_view);
        recyclerView.setHasFixedSize(true);
        layoutManager = new LinearLayoutManager(this);
        recyclerView.setLayoutManager(layoutManager);
        recyclerView.setItemAnimator(new DefaultItemAnimator());

        myItem = new ArrayList<AcceptedOrderItems>();

        ReadFromDB();

        confirmDelivery.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                deliverOrder(username);
            }
        });

        profileIV.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent it = new Intent(getApplicationContext(),
BikerProfile.class);
                startActivity(it);
                finish();
            }
        });

        locationIV.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent it;
                if (addressType.equals("Manual")) {
                    it = new Intent(getApplicationContext(),
LocationOfAddress.class);
                    it.putExtra("Address", address);
                } else {
                    it = new Intent(getApplicationContext(),
ShareLiveLocation.class);
                    it.putExtra("Longitude", lon);
                    it.putExtra("Latitude", lat);
                }
                it.putExtra("PhoneNo", phone);
                it.putExtra("OrderId", orderId);
                startActivity(it);
                finish();
            }
        });

        specials = pizza = burgers = fries = snacks = chilled_drinks = sea_foods
= coffees = net_sale = 0;
        specialsAmt = pizzaAmt = burgersAmt = friesAmt = snacksAmt =
chilled_drinksAmt = sea_foodsAmt = coffeesAmt = 0;

        drawer = findViewById(R.id.drawer_layout);

        if (drawer.isDrawerOpen(navView)) {
            drawer.closeDrawer(navView);
        }

        spr = getSharedPreferences("LoginSPR", MODE_PRIVATE);
        bikerUsername = spr.getString("Username", "");

        navView.setNavigationItemSelectedListener(new
NavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item) {

                if(item.getItemId() == R.id.nav_profile) {
                    if (drawer.isDrawerOpen(navView)) {
                        drawer.closeDrawer(navView);
                        Intent it = new Intent(getApplicationContext(),
BikerProfile.class);
                        startActivity(it);
                        finish();
                    }
                }

                if(item.getItemId() == R.id.nav_logout) {
                    if(drawer.isDrawerOpen(navView)) {
                        drawer.closeDrawer(navView);
                        logoutDialog();
                    }
                }


                if(item.getItemId() == R.id.nav_change_password) {
                    if(drawer.isDrawerOpen(navView)) {
                        drawer.closeDrawer(navView);
                        Intent it = new Intent(getApplicationContext(),
BikerPasswordChange.class);
                        startActivity(it);
                        finish();
                    }
                }

                if(item.getItemId() == R.id.nav_report) {
                    if(drawer.isDrawerOpen(navView)) {
                        drawer.closeDrawer(navView);
                        Intent it = new Intent(getApplicationContext(),
BikerReport.class);
                        startActivity(it);
                        finish();
                    }
                }

                if(item.getItemId() == R.id.nav_stars) {
                    if(drawer.isDrawerOpen(navView)) {
                        drawer.closeDrawer(navView);
                        Intent it = new Intent(getApplicationContext(),
BikersStars.class);
                        startActivity(it);
                        finish();
                    }
                }

                return false;
            }
        });

        phoneCallIV.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(Intent.ACTION_DIAL);
                intent.setData(Uri.parse("tel:" + phone));
                startActivity(intent);
            }
        });

    }

    public void deliverOrder(String uN) {
        LayoutInflater layoutInflater = LayoutInflater.from(this);
        View view = layoutInflater.inflate(R.layout.ask_deliver_order_dialog,
null);
        Button yesBTN = view.findViewById(R.id.yesBTN);
        Button noBTN = view.findViewById(R.id.noBTN);
        final AlertDialog alertDialog = new
AlertDialog.Builder(this).setView(view).create();
        alertDialog.show();
        yesBTN.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                loadingDialog.startLoadingDialog();

                saveToReports();

                Toast.makeText(getApplicationContext(), "Order Delivered
Successfully ...", Toast.LENGTH_SHORT).show();
                loadingDialog.dismissDialog();
                alertDialog.dismiss();
            }
        });

        noBTN.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                loadingDialog.dismissDialog();
                alertDialog.dismiss();
            }
        });
    }


    public void setUpToolbar() {
        drawer = findViewById(R.id.drawer_layout);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        actionBarDrawerToggle = new ActionBarDrawerToggle(this, drawer, toolbar,
R.string.app_name, R.string.app_name);
        drawer.addDrawerListener(actionBarDrawerToggle);

actionBarDrawerToggle.getDrawerArrowDrawable().setColor(getResources().getColor(
R.color.black));
        actionBarDrawerToggle.syncState();
    }

    @Override
    public void onBackPressed() {
        moveTaskToBack(true);
        android.os.Process.killProcess(android.os.Process.myPid());
        System.exit(1);
    }

    public void logoutDialog() {

        LayoutInflater layoutInflater = LayoutInflater.from(this);
        View view = layoutInflater.inflate(R.layout.ask_logout_dialog, null);
        Button yesBTN = view.findViewById(R.id.yesBTN);
        Button noBTN = view.findViewById(R.id.noBTN);
        final AlertDialog alertDialog = new
AlertDialog.Builder(this).setView(view).create();
        alertDialog.show();
        yesBTN.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                SharedPreferences.Editor editor = spr.edit();
                editor.putString("User", "");
                editor.putString("Status", "");
                editor.putString("Username", "");
                editor.apply();


                Intent intent = new Intent(getApplicationContext(),
LoginActivity.class);
                startActivity(intent);
                finish();
                Toast.makeText(getApplicationContext(), "Logout Successful ...",
Toast.LENGTH_SHORT).show();
                alertDialog.dismiss();
            }
        });

        noBTN.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                alertDialog.dismiss();
            }
        });

    }


    void ReadFromDB() {
        loadingDialog.startLoadingDialog();
        loopSize = 0;
        FirebaseDatabase.getInstance().getReference().child("Orders").child("On
the Way")
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    private DataSnapshot dataSnapshot;

                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                        this.dataSnapshot = dataSnapshot;

                        if (dataSnapshot.exists()) {
                            for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                                dbBikerUSN =
snapshot.child("BikerUsername").getValue().toString();
                                if (dbBikerUSN.equals(bikerUsername)) {
                                    orderId = snapshot.getKey().toString();
                                    flag = true;
                                    break;
                                }
                            }

                            if (flag) {


FirebaseDatabase.getInstance().getReference().child("Orders").child("On the
Way")

.child(orderId).addListenerForSingleValueEvent(new ValueEventListener() {

                                    DataSnapshot dataSnapshot;

                                    @Override
                                    public void onDataChange(@NonNull
DataSnapshot snapshot) {

                                        if (snapshot.exists()) {

                                            mainRL.setVisibility(View.VISIBLE);
                                            noOrderTV.setVisibility(View.GONE);
                                            noOrderIV.setVisibility(View.GONE);

                                            this.dataSnapshot = snapshot;

                                            name =
dataSnapshot.child("CustomerName").getValue().toString();
                                            phone =
dataSnapshot.child("CustomerPhoneNo").getValue().toString();
                                            payment =
dataSnapshot.child("CustomerPaymentType").getValue().toString();
                                            qty =
dataSnapshot.child("CustomerTotalQuantity").getValue().toString();
                                            price =
dataSnapshot.child("CustomerTotalBill").getValue().toString();
                                            acceptedBy =
dataSnapshot.child("AcceptedBy").getValue().toString();
                                            bikerName =
dataSnapshot.child("BikerName").getValue().toString();
                                            bikerUsername =
dataSnapshot.child("BikerUsername").getValue().toString();
                                            bikerPhone =
dataSnapshot.child("BikerPhoneNo").getValue().toString();
                                            receipt =
dataSnapshot.child("ReceiptImage").getValue().toString();
                                            username =
dataSnapshot.child("CustomerUsername").getValue().toString();
                                            addressType =
dataSnapshot.child("AddressType").getValue().toString();

                                            if (addressType.equals("Manual")) {
                                                address =
dataSnapshot.child("CustomerAddress").getValue().toString();

customerAddressTV.setText(address);
                                            } else {
                                                lat =
dataSnapshot.child("Latitude").getValue().toString();
                                                lon =
dataSnapshot.child("Longitude").getValue().toString();

                                                customerAddressTV.setText("(On
Maps)");
                                            }

                                            customerNameTV.setText(name);
                                            customerPhoneNoTV.setText(phone);
                                            paymentTypeTV.setText(payment);
                                            totalQtyTV.setText(qty);
                                            totalPriceTV.setText(price);

                                            ReadItems(orderId);
                                        }
                                    }

                                    @Override
                                    public void onCancelled(@NonNull
DatabaseError error) {

                                    }
                                });
                            } else {
                                loadingDialog.dismissDialog();
                                noOrderTV.setVisibility(View.VISIBLE);
                                noOrderIV.setVisibility(View.VISIBLE);
                                mainRL.setVisibility(View.GONE);
                            }
```

```
                    } else {
                            loadingDialog.dismissDialog();
                            noOrderTV.setVisibility(View.VISIBLE);
                            mainRL.setVisibility(View.GONE);
                            noOrderIV.setVisibility(View.VISIBLE);
                        }

                }

                    @Override
                    public void onCancelled(@NonNull DatabaseError error) {

                    }
                });
    }

    void ReadItems(String ordId) {
        loopSize = 0;
        FirebaseDatabase.getInstance().getReference().child("Orders").child("On
the Way").
                child(ordId).child("Items")
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    private DataSnapshot dataSnapshot;

                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                        this.dataSnapshot = dataSnapshot;

                        for (DataSnapshot snapshot : dataSnapshot.getChildren())
{
                            loopSize++;
                        }
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError error) {

                    }
                });

        FirebaseDatabase.getInstance().getReference().child("Orders").child("On
the Way")
                .child(ordId).child("Items")
                .addListenerForSingleValueEvent(new ValueEventListener() {

                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                        int j = 0;
                        MyAcceptedOrderData.itemId = new String[loopSize];
                        MyAcceptedOrderData.itemName = new String[loopSize];
                        MyAcceptedOrderData.itemPrice = new String[loopSize];
                        MyAcceptedOrderData.itemCategory = new String[loopSize];
                        MyAcceptedOrderData.itemDescription = new
String[loopSize];
                        MyAcceptedOrderData.itemSize = new String[loopSize];
                        MyAcceptedOrderData.itemURL = new String[loopSize];
                        MyAcceptedOrderData.itemTotalPrice = new
String[loopSize];
                        MyAcceptedOrderData.itemQuantity = new String[loopSize];

                        for (DataSnapshot snapshot : dataSnapshot.getChildren())
{

                            MyAcceptedOrderData.itemId[j] =
snapshot.child("Id").getValue().toString();
                            MyAcceptedOrderData.itemName[j] =
snapshot.child("Name").getValue().toString();
                            MyAcceptedOrderData.itemPrice[j] =
snapshot.child("Price").getValue().toString();
                            MyAcceptedOrderData.itemCategory[j] =
snapshot.child("Category").getValue().toString();
                            MyAcceptedOrderData.itemDescription[j] =
snapshot.child("Description").getValue().toString();
                            MyAcceptedOrderData.itemSize[j] =
snapshot.child("Size").getValue().toString();
                            MyAcceptedOrderData.itemURL[j] =
snapshot.child("ImageUrl").getValue().toString();
                            MyAcceptedOrderData.itemQuantity[j] =
snapshot.child("Quantity").getValue().toString();
                            MyAcceptedOrderData.itemTotalPrice[j] =
snapshot.child("TotalPrice").getValue().toString();

                            j++;
                        }

                        for (int i = 0; i < MyAcceptedOrderData.itemId.length;
i++) {
                            myItem.add(new AcceptedOrderItems(
                                    MyAcceptedOrderData.itemId[i],
                                    MyAcceptedOrderData.itemName[i],
                                    MyAcceptedOrderData.itemCategory[i],
                                    MyAcceptedOrderData.itemPrice[i],
                                    MyAcceptedOrderData.itemDescription[i],
                                    MyAcceptedOrderData.itemSize[i],
                                    MyAcceptedOrderData.itemURL[i],
                                    MyAcceptedOrderData.itemQuantity[i],
                                    MyAcceptedOrderData.itemTotalPrice[i]
                            ));
                        }

                        if (loopSize == 0) {
                            loadingDialog.dismissDialog();
                            Toast.makeText(BikersView.this, "No Item Founded
...", Toast.LENGTH_SHORT).show();
```

```
                        } else {
                            loadingDialog.dismissDialog();
                            acceptedOrderItemsAdapter = new
AcceptedOrderItemsAdapter(myItem);
                            recyclerView.setAdapter(acceptedOrderItemsAdapter);
                        }

                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError
databaseError) {

                    }
                });
    }

    public void setData(String un) {
        final DatabaseReference deliverOrderNode =
FirebaseDatabase.getInstance().getReference().
                child("Orders").child("Delivered").child(orderId);

        deliverOrderNode.child("CustomerUsername").setValue(username);
        deliverOrderNode.child("CustomerPhoneNo").setValue(phone);
        deliverOrderNode.child("AddressType").setValue(addressType);

        if (addressType.equals("Manual")) {
            deliverOrderNode.child("CustomerAddress").setValue(address);
        } else {
            deliverOrderNode.child("Latitude").setValue(lat);
            deliverOrderNode.child("Longitude").setValue(lon);
        }

        deliverOrderNode.child("CustomerName").setValue(name);
        deliverOrderNode.child("CustomerPaymentType").setValue(payment);
        deliverOrderNode.child("CustomerTotalBill").setValue(price);
        deliverOrderNode.child("CustomerTotalQuantity").setValue(qty);
        deliverOrderNode.child("AcceptedBy").setValue(acceptedBy);
        deliverOrderNode.child("BikerUsername").setValue(bikerUsername);
        deliverOrderNode.child("BikerName").setValue(bikerName);
        deliverOrderNode.child("BikerPhoneNo").setValue(bikerPhone);
        deliverOrderNode.child("OrderDate").setValue(getTodayDate());
        deliverOrderNode.child("OrderTime").setValue(getTime());
        deliverOrderNode.child("ReceiptImage").setValue(receipt);
        deliverOrderNode.child("Status").setValue("Delivered");

        copyData(orderId);
    }

    public void copyData(String keyy) {
        loopSize2 = 0;
        FirebaseDatabase.getInstance().getReference().child("Orders").child("On
the Way").
                child(keyy).child("Items")
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    private DataSnapshot dataSnapshot;

                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                        this.dataSnapshot = dataSnapshot;
                        for (DataSnapshot snapshot : dataSnapshot.getChildren())
{
                            if (snapshot.exists()) {
                                loopSize2++;
                            } else {
                                break;
                            }
                        }
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError error) {

                    }
                });


        FirebaseDatabase.getInstance().getReference().child("Orders").child("On
the Way").
                child(keyy).child("Items")
                .addListenerForSingleValueEvent(new ValueEventListener() {

                    @RequiresApi(api = Build.VERSION_CODES.O)
                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                        if (dataSnapshot.exists()) {

                            for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {

                                String id =
snapshot.child("Id").getValue().toString();

                                orderDeliverNodeRef =
FirebaseDatabase.getInstance().getReference().

child("Orders").child("Delivered").child(keyy).
                                        child("Items").child(id);

                                iD = snapshot.child("Id").getValue().toString();
                                nAme =
snapshot.child("Name").getValue().toString();
                                pRice =
snapshot.child("Price").getValue().toString();
                                cAtegory =
snapshot.child("Category").getValue().toString();
                                dEscription =
snapshot.child("Description").getValue().toString();
                                sIze =
snapshot.child("Size").getValue().toString();
                                uRl =
snapshot.child("ImageUrl").getValue().toString();
                                qTy =
snapshot.child("Quantity").getValue().toString();
                                tOtal =
snapshot.child("TotalPrice").getValue().toString();

                                orderDeliverNodeRef.child("Id").setValue(iD);

orderDeliverNodeRef.child("Name").setValue(nAme);

orderDeliverNodeRef.child("Price").setValue(pRice);

orderDeliverNodeRef.child("Category").setValue(cAtegory);

orderDeliverNodeRef.child("Description").setValue(dEscription);

orderDeliverNodeRef.child("Size").setValue(sIze);

orderDeliverNodeRef.child("ImageUrl").setValue(uRl);

orderDeliverNodeRef.child("Quantity").setValue(qTy);

orderDeliverNodeRef.child("TotalPrice").setValue(tOtal);
                            }

                            deleteOnTheWayOrder(keyy);
                        }

                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError
databaseError) {

                    }
                });
    }

    @RequiresApi(api = Build.VERSION_CODES.O)
    public void deleteOnTheWayOrder(String ordId) {
        FirebaseDatabase.getInstance().getReference().child("Orders").child("On
the Way").
                child(ordId).removeValue();
        FirebaseDatabase.getInstance().getReference().child("LiveOrders").
                child(ordId).removeValue();

        sendNotification(bikerName, payment, price);
        try {
            Thread.sleep(700);
        } catch (InterruptedException e) {
            e.printStackTrace();
        } finally {
            setStatus(bikerUsername);
        }
    }

    public void setStatus(String userName) {
        final DatabaseReference bikerStatusNode =
FirebaseDatabase.getInstance().getReference().
                child("Users").child("Bikers").child(userName);

        bikerStatusNode.child("AvailabilityStatus").setValue("Available");

        loadingDialog.dismissDialog();

        startActivity(getIntent());
        finish();
    }

    public void saveToReports() {
        String todayDate = getFormatDate();
        getCategoriesQty(todayDate);

    }

    public void getCategoriesQty(String dt) {

FirebaseDatabase.getInstance().getReference().child("Reports").child("date_wise"
).child(dt)
                .addListenerForSingleValueEvent(new ValueEventListener() {

                    DataSnapshot dataSnapshot;

                    @Override
                    public void onDataChange(@NonNull DataSnapshot snapshot) {

                        this.dataSnapshot = snapshot;

                        if (dataSnapshot.exists()) {
                            sp =
dataSnapshot.child("Specials").getValue().toString();
                            pi =
dataSnapshot.child("Pizza").getValue().toString();
                            bu =
dataSnapshot.child("Burgers").getValue().toString();
                            fr =
dataSnapshot.child("Fries").getValue().toString();
                            sn =
dataSnapshot.child("Snacks").getValue().toString();
                            ch = dataSnapshot.child("Chilled
Drinks").getValue().toString();
                            se = dataSnapshot.child("Sea
Foods").getValue().toString();
                            co =
dataSnapshot.child("Coffees").getValue().toString();
                            total = dataSnapshot.child("Net
Sale").getValue().toString();

                            spAmt =
dataSnapshot.child("SpecialsAmount").getValue().toString();
                            piAmt =
dataSnapshot.child("PizzaAmount").getValue().toString();
                            buAmt =
dataSnapshot.child("BurgersAmount").getValue().toString();
                            frAmt =
dataSnapshot.child("FriesAmount").getValue().toString();
                            snAmt =
dataSnapshot.child("SnacksAmount").getValue().toString();
                            chAmt = dataSnapshot.child("Chilled
DrinksAmount").getValue().toString();
                            seAmt = dataSnapshot.child("Sea
FoodsAmount").getValue().toString();
                            coAmt =
dataSnapshot.child("CoffeesAmount").getValue().toString();

                        } else {
                            mDatabaseRef =
mDatabase.getReference().child("Reports").
                                    child("date_wise").child(dt);

                            sp = pi = bu = fr = sn = ch = se = co = total = "0";
                            spAmt = piAmt = buAmt = frAmt = snAmt = chAmt =
seAmt = coAmt = "0";

                            mDatabaseRef.child("Specials").setValue("0");
                            mDatabaseRef.child("Pizza").setValue("0");
                            mDatabaseRef.child("Burgers").setValue("0");
                            mDatabaseRef.child("Fries").setValue("0");
                            mDatabaseRef.child("Snacks").setValue("0");
                            mDatabaseRef.child("Chilled Drinks").setValue("0");
                            mDatabaseRef.child("Sea Foods").setValue("0");
                            mDatabaseRef.child("Coffees").setValue("0");
                            mDatabaseRef.child("Net Sale").setValue("0");

mDatabaseRef.child("Date").setValue(getFormatDate());
                            mDatabaseRef.child("SpecialsAmount").setValue("0");
                            mDatabaseRef.child("PizzaAmount").setValue("0");
                            mDatabaseRef.child("BurgersAmount").setValue("0");
                            mDatabaseRef.child("FriesAmount").setValue("0");
                            mDatabaseRef.child("SnacksAmount").setValue("0");
                            mDatabaseRef.child("Chilled
DrinksAmount").setValue("0");
                            mDatabaseRef.child("Sea FoodsAmount").setValue("0");
                            mDatabaseRef.child("CoffeesAmount").setValue("0");

                        }

                        saveToDB(sp, pi, bu, fr, sn, ch, se, co, total, spAmt,
piAmt, buAmt, frAmt,
                                snAmt, chAmt, seAmt, coAmt);

                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError error) {

                    }
                });
    }

    public void saveToDB(String ss, String pa, String bs, String fs, String sn,
String cs,
                         String se, String ds, String to, String ss2, String
pa2, String bs2,
                         String fs2, String sn2, String cs2, String se2, String
ds2) {

        specials = Integer.parseInt(ss);
        pizza = Integer.parseInt(pa);
        burgers = Integer.parseInt(bs);
        fries = Integer.parseInt(fs);
        snacks = Integer.parseInt(sn);
        chilled_drinks = Integer.parseInt(cs);
        sea_foods = Integer.parseInt(se);
        coffees = Integer.parseInt(ds);
        net_sale = Integer.parseInt(to);
        specialsAmt = Integer.parseInt(ss2);
        pizzaAmt = Integer.parseInt(pa2);
        burgersAmt = Integer.parseInt(bs2);
        friesAmt = Integer.parseInt(fs2);
        snacksAmt = Integer.parseInt(sn2);
        chilled_drinksAmt = Integer.parseInt(cs2);
        sea_foodsAmt = Integer.parseInt(se2);
        coffeesAmt = Integer.parseInt(ds2);


        FirebaseDatabase.getInstance().getReference().child("Orders").child("On
the Way")
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    private DataSnapshot dataSnapshot;

                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                        this.dataSnapshot = dataSnapshot;

                        if (dataSnapshot.exists()) {
                            for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                                dbBikerUSN =
snapshot.child("BikerUsername").getValue().toString();

                                if (dbBikerUSN.equals(bikerUsername)) {
                                    orderId = snapshot.getKey().toString();
                                    billTotal =
snapshot.child("CustomerTotalBill").getValue().toString();
                                    net_sale = net_sale +
Integer.parseInt(billTotal);
                                    flag = true;
                                    break;
                                }
                            }
                            getItems(orderId);
                        }

                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError error) {

                    }
                });

    }

    // Notification
    @RequiresApi(api = Build.VERSION_CODES.O)
    void sendNotification(String nam, String pnt, String prc) {

        String textTitle = "ORDER DELIVERED !";
        String textContent = "Dear "+ nam + ", Order has been Delivered
Successfully !";

        if (pnt.equals("Online")) {
            textContent.concat("\nBill has been Paid Online.");
        } else {
            textContent.concat("\nCollect "+ prc + " Rs. from Customer.");
        }

        // Creating a notification channel
        NotificationChannel channel = new NotificationChannel("channel1",
"hello", NotificationManager.IMPORTANCE_HIGH);
        NotificationManager manager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
        manager.createNotificationChannel(channel);
        // Creating the notification object
        NotificationCompat.Builder notification = new
NotificationCompat.Builder(getApplicationContext(),"channel1");

        // Notification.setAutoCancel(true);
        notification.setContentTitle(textTitle);
        notification.setContentText(textContent);
        notification.setSmallIcon(R.drawable.cafe_main_logo);

        // Make the notification manager to issue a notification on the
notification's channel
        manager.notify(121,notification.build());

    }

    public void getItems(String id) {
        FirebaseDatabase.getInstance().getReference().child("Orders").child("On
the Way")
                .child(id).child("Items")
                .addListenerForSingleValueEvent(new ValueEventListener() {

                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{

                            for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                                itemCategory =
snapshot.child("Category").getValue().toString();
                                itemQuantity =
snapshot.child("Quantity").getValue().toString();
                                itemTotal =
snapshot.child("TotalPrice").getValue().toString();

                                if (itemCategory.equals("Specials")) {
                                    specials = specials +
Integer.parseInt(itemQuantity);
                                    specialsAmt = specialsAmt +
Integer.parseInt(itemTotal);
                                } else if (itemCategory.equals("Pizza")) {
                                    pizza = pizza +
Integer.parseInt(itemQuantity);
                                    pizzaAmt = pizzaAmt +
Integer.parseInt(itemTotal);
                                } else if (itemCategory.equals("Burgers")) {
                                    burgers = burgers +
Integer.parseInt(itemQuantity);
                                    burgersAmt = burgersAmt +
Integer.parseInt(itemTotal);
                                } else if (itemCategory.equals("Fries")) {
                                    fries = fries +
Integer.parseInt(itemQuantity);
```

```
                                    friesAmt = friesAmt +
Integer.parseInt(itemTotal);
                                } else if (itemCategory.equals("Snacks")) {
                                    snacks = snacks +
Integer.parseInt(itemQuantity);

                                    snacksAmt = snacksAmt +
Integer.parseInt(itemTotal);
                                } else if (itemCategory.equals("Chilled
Drinks")) {

                                    chilled_drinks = chilled_drinks +
Integer.parseInt(itemQuantity);

                                    chilled_drinksAmt = chilled_drinksAmt +
Integer.parseInt(itemTotal);
                                } else if (itemCategory.equals("Sea Foods")) {
                                    sea_foods = sea_foods +
Integer.parseInt(itemQuantity);

                                    sea_foodsAmt = sea_foodsAmt +
Integer.parseInt(itemTotal);
                                } else if (itemCategory.equals("Coffees")) {
                                    coffees = coffees +
Integer.parseInt(itemQuantity);

                                    coffeesAmt = coffeesAmt +
Integer.parseInt(itemTotal);
                                }
                            }
                        saveReport(specials, pizza, burgers, fries, snacks,
chilled_drinks,
                                sea_foods, coffees, net_sale, specialsAmt,
pizzaAmt, burgersAmt,
                                friesAmt, snacksAmt, chilled_drinksAmt,
sea_foodsAmt, coffeesAmt);
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError
databaseError) {

                    }
                });
    }


    public void saveReport(int a, int b, int c, int d, int e, int f, int g, int
h, int i, int j,
                            int k, int l, int m, int n, int o, int p, int q) {
        sp = String.valueOf(a);
        pi = String.valueOf(b);
        bu = String.valueOf(c);
        fr = String.valueOf(d);
        sn = String.valueOf(e);
        ch = String.valueOf(f);
        se = String.valueOf(g);
        co = String.valueOf(h);
        total = String.valueOf(i);
        spAmt = String.valueOf(j);
        piAmt = String.valueOf(k);
        buAmt = String.valueOf(l);
        frAmt = String.valueOf(m);
        snAmt = String.valueOf(n);
        chAmt = String.valueOf(o);
        seAmt = String.valueOf(p);
        coAmt = String.valueOf(q);

        mDatabaseRef = mDatabase.getReference().child("Reports").
                child("date_wise").child(getFormatDate());

        mDatabaseRef.child("Specials").setValue(sp);
        mDatabaseRef.child("Pizza").setValue(pi);
        mDatabaseRef.child("Burgers").setValue(bu);
        mDatabaseRef.child("Fries").setValue(fr);
        mDatabaseRef.child("Snacks").setValue(sn);
        mDatabaseRef.child("Chilled Drinks").setValue(ch);
        mDatabaseRef.child("Sea Foods").setValue(se);
        mDatabaseRef.child("Coffees").setValue(co);
        mDatabaseRef.child("Net Sale").setValue(total);
        mDatabaseRef.child("Date").setValue(getFormatDate());
        mDatabaseRef.child("SpecialsAmount").setValue(spAmt);
        mDatabaseRef.child("PizzaAmount").setValue(piAmt);
        mDatabaseRef.child("BurgersAmount").setValue(buAmt);
        mDatabaseRef.child("FriesAmount").setValue(frAmt);
        mDatabaseRef.child("SnacksAmount").setValue(snAmt);
        mDatabaseRef.child("Chilled DrinksAmount").setValue(chAmt);
        mDatabaseRef.child("Sea FoodsAmount").setValue(seAmt);
        mDatabaseRef.child("CoffeesAmount").setValue(coAmt);

        setMonthlySales();

    }

    public void setMonthlySales() {
        String todayDate = getMonth();
        getMonthRecord(todayDate);
    }

    public void getMonthRecord(String month) {

FirebaseDatabase.getInstance().getReference().child("Reports").child("month_wise
").
                child(month).addListenerForSingleValueEvent(new
ValueEventListener() {

                    DataSnapshot dataSnapshot;

                    @Override
                    public void onDataChange(@NonNull DataSnapshot snapshot) {
                         this.dataSnapshot = snapshot;

                         if (dataSnapshot.exists()) {
                             sp =
dataSnapshot.child("Specials").getValue().toString();
                             pi =
dataSnapshot.child("Pizza").getValue().toString();
                             bu =
dataSnapshot.child("Burgers").getValue().toString();
                             fr =
dataSnapshot.child("Fries").getValue().toString();
                             sn =
dataSnapshot.child("Snacks").getValue().toString();
                             ch = dataSnapshot.child("Chilled
Drinks").getValue().toString();
                             se = dataSnapshot.child("Sea
Foods").getValue().toString();
                             co =
dataSnapshot.child("Coffees").getValue().toString();
                             ns = dataSnapshot.child("Net
Sale").getValue().toString();

                             spAmt =
dataSnapshot.child("SpecialsAmount").getValue().toString();
                             piAmt =
dataSnapshot.child("PizzaAmount").getValue().toString();
                             buAmt =
dataSnapshot.child("BurgersAmount").getValue().toString();
                             frAmt =
dataSnapshot.child("FriesAmount").getValue().toString();
                             snAmt =
dataSnapshot.child("SnacksAmount").getValue().toString();
                             chAmt = dataSnapshot.child("Chilled
DrinksAmount").getValue().toString();
                             seAmt = dataSnapshot.child("Sea
FoodsAmount").getValue().toString();
                             coAmt =
dataSnapshot.child("CoffeesAmount").getValue().toString();

                         } else {
                             mDatabaseRef =
mDatabase.getReference().child("Reports").
                                     child("month_wise").child(month);

                             sp = pi = bu = fr = sn = ch = se = co = ns = "0";
                             spAmt = piAmt = buAmt = frAmt = snAmt = chAmt =
seAmt = coAmt = "0";

                             mDatabaseRef.child("Specials").setValue("0");
                             mDatabaseRef.child("Pizza").setValue("0");
                             mDatabaseRef.child("Burgers").setValue("0");
                             mDatabaseRef.child("Fries").setValue("0");
                             mDatabaseRef.child("Snacks").setValue("0");
```

```
                            mDatabaseRef.child("Chilled Drinks").setValue("0");
                            mDatabaseRef.child("Sea Foods").setValue("0");
                            mDatabaseRef.child("Coffees").setValue("0");
                            mDatabaseRef.child("Net Sale").setValue("0");
                            mDatabaseRef.child("Month").setValue(month);
                            mDatabaseRef.child("SpecialsAmount").setValue("0");
                            mDatabaseRef.child("PizzaAmount").setValue("0");
                            mDatabaseRef.child("BurgersAmount").setValue("0");
                            mDatabaseRef.child("FriesAmount").setValue("0");
                            mDatabaseRef.child("SnacksAmount").setValue("0");
                            mDatabaseRef.child("Chilled
DrinksAmount").setValue("0");
                            mDatabaseRef.child("Sea FoodsAmount").setValue("0");
                            mDatabaseRef.child("CoffeesAmount").setValue("0");

                        }
                        saveToDBMonthly(sp, pi, bu, fr, sn, ch, se, co, ns,
spAmt, piAmt, buAmt,
                                frAmt, snAmt, chAmt, seAmt, coAmt);

                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError error) {

                    }
                });
    }

    public void saveToDBMonthly(String ss, String pa, String bs, String fs,
String sn, String cs,
                         String se, String ds, String ne, String ss2, String
pa2, String bs2,
                                String fs2, String sn2, String cs2, String se2,
String ds2) {

        specials = Integer.parseInt(ss);
        pizza = Integer.parseInt(pa);
        burgers = Integer.parseInt(bs);
        fries = Integer.parseInt(fs);
        snacks = Integer.parseInt(sn);
        chilled_drinks = Integer.parseInt(cs);
        sea_foods = Integer.parseInt(se);
        coffees = Integer.parseInt(ds);
        net_sale = Integer.parseInt(ne);

        specialsAmt = Integer.parseInt(ss2);
        pizzaAmt = Integer.parseInt(pa2);
        burgersAmt = Integer.parseInt(bs2);
        friesAmt = Integer.parseInt(fs2);
        snacksAmt = Integer.parseInt(sn2);
        chilled_drinksAmt = Integer.parseInt(cs2);
        sea_foodsAmt = Integer.parseInt(se2);
        coffeesAmt = Integer.parseInt(ds2);


        FirebaseDatabase.getInstance().getReference().child("Orders").child("On
the Way")
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    private DataSnapshot dataSnapshot;

                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                        this.dataSnapshot = dataSnapshot;

                        if (dataSnapshot.exists()) {
                            for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                                dbBikerUSN =
snapshot.child("BikerUsername").getValue().toString();

                                if (dbBikerUSN.equals(bikerUsername)) {
                                    orderId = snapshot.getKey().toString();
                                    billTotal =
snapshot.child("CustomerTotalBill").getValue().toString();
                                    net_sale = net_sale +
Integer.parseInt(billTotal);
                                    flag = true;
                                    break;
                                }
                            }
                            getItemsMonthly(orderId);
                        }

                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError error) {

                    }
                });

    }

    public void getItemsMonthly(String id) {
        FirebaseDatabase.getInstance().getReference().child("Orders").child("On
the Way")
                .child(id).child("Items")
                .addListenerForSingleValueEvent(new ValueEventListener() {

                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{

                        for (DataSnapshot snapshot : dataSnapshot.getChildren())
```

```
{
                            itemCategory =
snapshot.child("Category").getValue().toString();
                            itemQuantity =
snapshot.child("Quantity").getValue().toString();
                            itemTotal =
snapshot.child("TotalPrice").getValue().toString();

                            if (itemCategory.equals("Specials")) {
                                specials = specials +
Integer.parseInt(itemQuantity);
                                specialsAmt = specialsAmt +
Integer.parseInt(itemTotal);
                            } else if (itemCategory.equals("Pizza")) {
                                pizza = pizza + Integer.parseInt(itemQuantity);
                                pizzaAmt = pizzaAmt +
Integer.parseInt(itemTotal);
                            } else if (itemCategory.equals("Burgers")) {
                                burgers = burgers +
Integer.parseInt(itemQuantity);
                                burgersAmt = burgersAmt +
Integer.parseInt(itemTotal);
                            } else if (itemCategory.equals("Fries")) {
                                fries = fries + Integer.parseInt(itemQuantity);
                                friesAmt = friesAmt +
Integer.parseInt(itemTotal);
                            } else if (itemCategory.equals("Snacks")) {
                                snacks = snacks +
Integer.parseInt(itemQuantity);
                                snacksAmt = snacksAmt +
Integer.parseInt(itemTotal);
                            } else if (itemCategory.equals("Chilled Drinks")) {
                                chilled_drinks = chilled_drinks +
Integer.parseInt(itemQuantity);
                                chilled_drinksAmt = chilled_drinksAmt +
Integer.parseInt(itemTotal);
                            } else if (itemCategory.equals("Sea Foods")) {
                                sea_foods = sea_foods +
Integer.parseInt(itemQuantity);
                                sea_foodsAmt = sea_foodsAmt +
Integer.parseInt(itemTotal);
                            } else if (itemCategory.equals("Coffees")) {
                                coffees = coffees +
Integer.parseInt(itemQuantity);
                                coffeesAmt = coffeesAmt +
Integer.parseInt(itemTotal);
                            }
                        }

                        saveReportMonthly(specials, pizza, burgers, fries,
snacks, chilled_drinks,
                                sea_foods, coffees, net_sale, specialsAmt,
pizzaAmt, burgersAmt,
                                 friesAmt, snacksAmt, chilled_drinksAmt,
sea_foodsAmt, coffeesAmt);
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError
databaseError) {

                    }
                });
    }

    public void saveReportMonthly(int a, int b, int c, int d, int e, int f, int
g, int h, int i, int j,
                                   int k, int l, int m ,int n, int o, int p, int
q) {
        sp = String.valueOf(a);
        pi = String.valueOf(b);
        bu = String.valueOf(c);
        fr = String.valueOf(d);
        sn = String.valueOf(e);
        ch = String.valueOf(f);
        se = String.valueOf(g);
        co = String.valueOf(h);
        ns = String.valueOf(i);

        spAmt = String.valueOf(j);
        piAmt = String.valueOf(k);
        buAmt = String.valueOf(l);
        frAmt = String.valueOf(m);
        snAmt = String.valueOf(n);
        chAmt = String.valueOf(o);
        seAmt = String.valueOf(p);
        coAmt = String.valueOf(q);

        mDatabaseRef2 = mDatabase2.getReference().child("Reports").
                child("month_wise").child(getMonth());

        mDatabaseRef2.child("Specials").setValue(sp);
        mDatabaseRef2.child("Pizza").setValue(pi);
        mDatabaseRef2.child("Burgers").setValue(bu);
        mDatabaseRef2.child("Fries").setValue(fr);
        mDatabaseRef2.child("Snacks").setValue(sn);
        mDatabaseRef2.child("Chilled Drinks").setValue(ch);
        mDatabaseRef2.child("Sea Foods").setValue(se);
        mDatabaseRef2.child("Coffees").setValue(co);
        mDatabaseRef2.child("Net Sale").setValue(ns);
        mDatabaseRef2.child("Month").setValue(getMonth());

        mDatabaseRef2.child("SpecialsAmount").setValue(spAmt);
        mDatabaseRef2.child("PizzaAmount").setValue(piAmt);
        mDatabaseRef2.child("BurgersAmount").setValue(buAmt);
        mDatabaseRef2.child("FriesAmount").setValue(frAmt);
        mDatabaseRef2.child("SnacksAmount").setValue(snAmt);
        mDatabaseRef2.child("Chilled DrinksAmount").setValue(chAmt);
        mDatabaseRef2.child("Sea FoodsAmount").setValue(seAmt);
        mDatabaseRef2.child("CoffeesAmount").setValue(coAmt);

        setYearlySales();

    }

    public void setYearlySales() {
        String currentYear = getYear();
        getYearRecord(currentYear);
    }


    public void getYearRecord(String year) {

FirebaseDatabase.getInstance().getReference().child("Reports").child("year_wise"
).
                child(year).addListenerForSingleValueEvent(new
ValueEventListener() {

            DataSnapshot dataSnapshot;

            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {

                this.dataSnapshot = snapshot;

                if (dataSnapshot.exists()) {
                    sp = dataSnapshot.child("Specials").getValue().toString();
                    pi = dataSnapshot.child("Pizza").getValue().toString();
                    bu = dataSnapshot.child("Burgers").getValue().toString();
                    fr = dataSnapshot.child("Fries").getValue().toString();
                    sn = dataSnapshot.child("Snacks").getValue().toString();
                    ch = dataSnapshot.child("Chilled
Drinks").getValue().toString();
                    se = dataSnapshot.child("Sea Foods").getValue().toString();
                    co = dataSnapshot.child("Coffees").getValue().toString();
                    ns = dataSnapshot.child("Net Sale").getValue().toString();

                    spAmt =
dataSnapshot.child("SpecialsAmount").getValue().toString();
                    piAmt =
dataSnapshot.child("PizzaAmount").getValue().toString();
                    buAmt =
dataSnapshot.child("BurgersAmount").getValue().toString();
                    frAmt =
dataSnapshot.child("FriesAmount").getValue().toString();
                    snAmt =
dataSnapshot.child("SnacksAmount").getValue().toString();
                    chAmt = dataSnapshot.child("Chilled
DrinksAmount").getValue().toString();
```

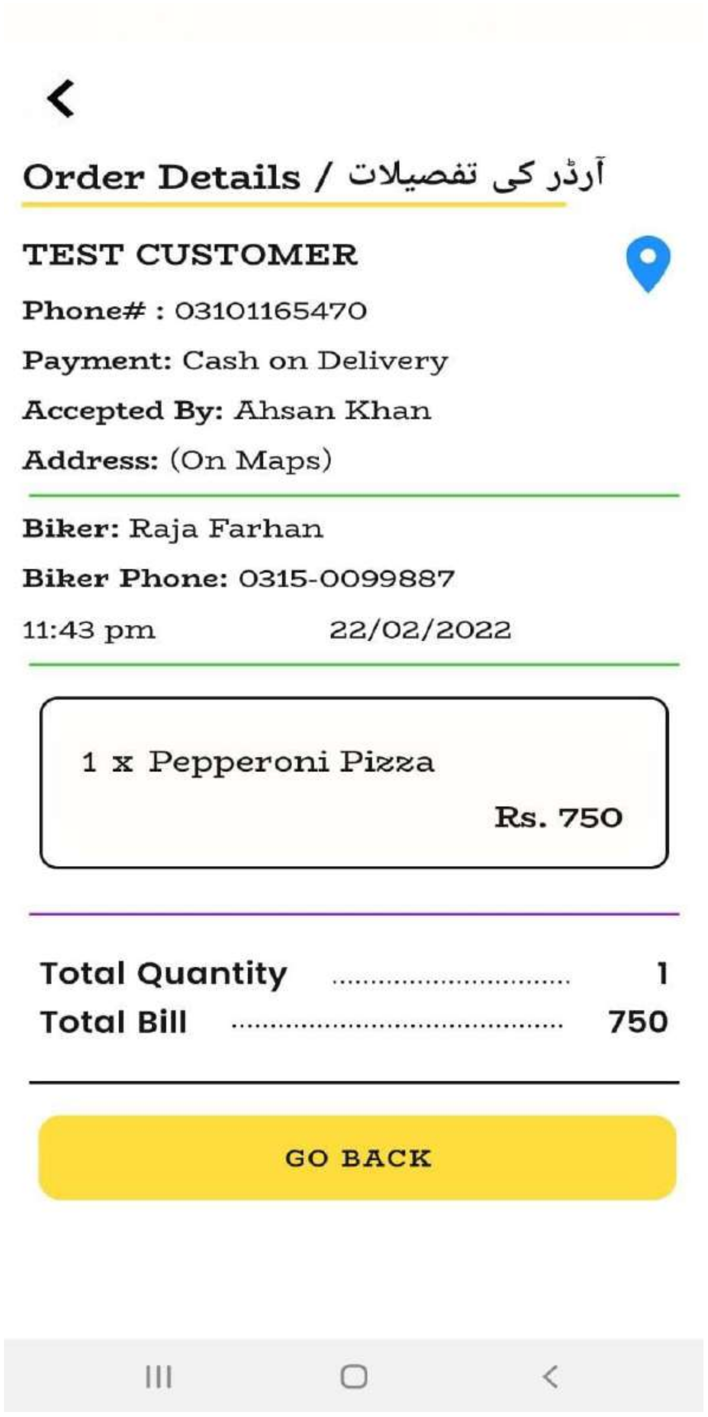
```
                    seAmt = dataSnapshot.child("Sea
FoodsAmount").getValue().toString();
                    coAmt =
dataSnapshot.child("CoffeesAmount").getValue().toString();

                } else {
                    mDatabaseRef = mDatabase.getReference().child("Reports").
                            child("year_wise").child(year);

                    sp = pi = bu = fr = sn = ch = se = co = ns = "0";
                    spAmt = piAmt = buAmt = frAmt = snAmt = chAmt = seAmt =
coAmt = "0";

                    mDatabaseRef.child("Specials").setValue("0");
                    mDatabaseRef.child("Pizza").setValue("0");
                    mDatabaseRef.child("Burgers").setValue("0");
                    mDatabaseRef.child("Fries").setValue("0");
                    mDatabaseRef.child("Snacks").setValue("0");
                    mDatabaseRef.child("Chilled Drinks").setValue("0");
                    mDatabaseRef.child("Sea Foods").setValue("0");
                    mDatabaseRef.child("Coffees").setValue("0");
                    mDatabaseRef.child("Net Sale").setValue("0");
                    mDatabaseRef.child("Year").setValue(year);

                    mDatabaseRef.child("SpecialsAmount").setValue("0");
                    mDatabaseRef.child("PizzaAmount").setValue("0");
                    mDatabaseRef.child("BurgersAmount").setValue("0");
                    mDatabaseRef.child("FriesAmount").setValue("0");
                    mDatabaseRef.child("SnacksAmount").setValue("0");
                    mDatabaseRef.child("Chilled DrinksAmount").setValue("0");
                    mDatabaseRef.child("Sea FoodsAmount").setValue("0");
                    mDatabaseRef.child("CoffeesAmount").setValue("0");

                }
                saveToDBYearly(sp, pi, bu, fr, sn, ch, se, co, ns, spAmt, piAmt,
buAmt,
                        frAmt, snAmt, chAmt, seAmt, coAmt);

            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {

            }
        });
    }


    public void saveToDBYearly(String ss, String pa, String bs, String fs,
String sn, String cs,
                               String se, String ds, String ne, String ss2,
String pa2, String bs2,
                              String fs2, String sn2, String cs2, String se2,
String ds2) {

        specials = Integer.parseInt(ss);
        pizza = Integer.parseInt(pa);
        burgers = Integer.parseInt(bs);
        fries = Integer.parseInt(fs);
        snacks = Integer.parseInt(sn);
        chilled_drinks = Integer.parseInt(cs);
        sea_foods = Integer.parseInt(se);
        coffees = Integer.parseInt(ds);
        net_sale = Integer.parseInt(ne);

        specialsAmt = Integer.parseInt(ss2);
        pizzaAmt = Integer.parseInt(pa2);
        burgersAmt = Integer.parseInt(bs2);
        friesAmt = Integer.parseInt(fs2);
        snacksAmt = Integer.parseInt(sn2);
        chilled_drinksAmt = Integer.parseInt(cs2);
        sea_foodsAmt = Integer.parseInt(se2);
        coffeesAmt = Integer.parseInt(ds2);


        FirebaseDatabase.getInstance().getReference().child("Orders").child("On
the Way")
                .addListenerForSingleValueEvent(new ValueEventListener() {
                    private DataSnapshot dataSnapshot;

                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                        this.dataSnapshot = dataSnapshot;

                        if (dataSnapshot.exists()) {
                            for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                                dbBikerUSN =
snapshot.child("BikerUsername").getValue().toString();

                                if (dbBikerUSN.equals(bikerUsername)) {
                                    orderId = snapshot.getKey().toString();
                                    billTotal =
snapshot.child("CustomerTotalBill").getValue().toString();
                                    net_sale = net_sale +
Integer.parseInt(billTotal);
                                    flag = true;
                                    break;
                                }
                            }
                            getItemsYearly(orderId);
                        }

                    }
```

```
                        @Override
                        public void onCancelled(@NonNull DatabaseError error) {

                        }
                    });
    }

    public void getItemsYearly(String id) {
        FirebaseDatabase.getInstance().getReference().child("Orders").child("On
the Way")
                    .child(id).child("Items")
                    .addListenerForSingleValueEvent(new ValueEventListener() {

                        @Override
                        public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{

                            for (DataSnapshot snapshot : dataSnapshot.getChildren())
{
                                itemCategory =
snapshot.child("Category").getValue().toString();
                                itemQuantity =
snapshot.child("Quantity").getValue().toString();
                                itemTotal =
snapshot.child("TotalPrice").getValue().toString();

                                if (itemCategory.equals("Specials")) {
                                    specials = specials +
Integer.parseInt(itemQuantity);
                                    specialsAmt = specialsAmt +
Integer.parseInt(itemTotal);
                                } else if (itemCategory.equals("Pizza")) {
                                    pizza = pizza + Integer.parseInt(itemQuantity);
                                    pizzaAmt = pizzaAmt +
Integer.parseInt(itemTotal);
                                } else if (itemCategory.equals("Burgers")) {
                                    burgers = burgers +
Integer.parseInt(itemQuantity);
                                    burgersAmt = burgersAmt +
Integer.parseInt(itemTotal);
                                } else if (itemCategory.equals("Fries")) {
                                    fries = fries + Integer.parseInt(itemQuantity);
                                    friesAmt = friesAmt +
Integer.parseInt(itemTotal);
                                } else if (itemCategory.equals("Snacks")) {
                                    snacks = snacks +
Integer.parseInt(itemQuantity);
                                    snacksAmt = snacksAmt +
Integer.parseInt(itemTotal);
                                } else if (itemCategory.equals("Chilled Drinks")) {
                                    chilled_drinks = chilled_drinks +
Integer.parseInt(itemQuantity);
                                    chilled_drinksAmt = chilled_drinksAmt +
Integer.parseInt(itemTotal);
                                } else if (itemCategory.equals("Sea Foods")) {
                                    sea_foods = sea_foods +
Integer.parseInt(itemQuantity);
                                    sea_foodsAmt = sea_foodsAmt +
Integer.parseInt(itemTotal);
                                } else if (itemCategory.equals("Coffees")) {
                                    coffees = coffees +
Integer.parseInt(itemQuantity);
                                    coffeesAmt = coffeesAmt +
Integer.parseInt(itemTotal);
                                }
                            }

                            saveReportYearly(specials, pizza, burgers, fries,
snacks, chilled_drinks,
                                    sea_foods, coffees, net_sale, specialsAmt,
pizzaAmt, burgersAmt,
                                    friesAmt, snacksAmt, chilled_drinksAmt,
sea_foodsAmt, coffeesAmt);
                        }

                        @Override
                        public void onCancelled(@NonNull DatabaseError
databaseError) {

                        }
                    });
    }

    public void saveReportYearly(int a, int b, int c, int d, int e, int f, int
g, int h, int i, int j,
                                 int k, int l, int m ,int n, int o, int p, int
q) {
        sp = String.valueOf(a);
        pi = String.valueOf(b);
        bu = String.valueOf(c);
        fr = String.valueOf(d);
        sn = String.valueOf(e);
        ch = String.valueOf(f);
        se = String.valueOf(g);
        co = String.valueOf(h);
        ns = String.valueOf(i);

        spAmt = String.valueOf(j);
        piAmt = String.valueOf(k);
        buAmt = String.valueOf(l);
        frAmt = String.valueOf(m);
        snAmt = String.valueOf(n);
        chAmt = String.valueOf(o);
        seAmt = String.valueOf(p);
        coAmt = String.valueOf(q);
        mDatabaseRef2 = mDatabase2.getReference().child("Reports").
                child("year_wise").child(getYear());

        mDatabaseRef2.child("Specials").setValue(sp);
        mDatabaseRef2.child("Pizza").setValue(pi);
        mDatabaseRef2.child("Burgers").setValue(bu);
        mDatabaseRef2.child("Fries").setValue(fr);
        mDatabaseRef2.child("Snacks").setValue(sn);
        mDatabaseRef2.child("Chilled Drinks").setValue(ch);
        mDatabaseRef2.child("Sea Foods").setValue(se);
        mDatabaseRef2.child("Coffees").setValue(co);
        mDatabaseRef2.child("Net Sale").setValue(ns);
        mDatabaseRef2.child("Year").setValue(getYear());

        mDatabaseRef2.child("SpecialsAmount").setValue(spAmt);
        mDatabaseRef2.child("PizzaAmount").setValue(piAmt);
        mDatabaseRef2.child("BurgersAmount").setValue(buAmt);
        mDatabaseRef2.child("FriesAmount").setValue(frAmt);
        mDatabaseRef2.child("SnacksAmount").setValue(snAmt);
        mDatabaseRef2.child("Chilled DrinksAmount").setValue(chAmt);
        mDatabaseRef2.child("Sea FoodsAmount").setValue(seAmt);
        mDatabaseRef2.child("CoffeesAmount").setValue(coAmt);

        setData(username);

    }
}
```

Description:

This module consist of all the information related to the order and the customer that were assigned to biker. Biker can use that information in order to deliver the order. Moreover, it consist of confirm delivery button that the biker need to press after food is being delivered.

Output:



Order Details / آرڈر کی تفصیلات

TEST CUSTOMER

Phone# : 03101165470

Payment: Cash on Delivery

Accepted By: Ahsan Khan

Address: (On Maps)

Biker: Raja Farhan

Biker Phone: 0315-0099887

11:43 pm 22/02/2022

1 x Pepperoni Pizza

Rs. 750

Total Quantity 1

Total Bill 750

Figure 115: Bikers View UI

Chapter 5

Software Testing

Software Testing is the most crucial part of Software Development Process. It is the investigation or evaluation of a software component, improving them, and finding bugs and defects. Testing is usually done by executing a system in such a way that it identifies any gaps, errors, or missing requirements in contrary to the actual requirements.

5.1. Testing Methodology:

After implementation, the process flow manager is tested for functional errors. We are going to do System Testing, which is the testing of the functional requirements implemented in our system without regard to code. The System is efficient and contains the following benefits:

1. *Examines the functionality of an application without peering into its internal structures or workings.*
2. *Can be applied virtually to every level of software testing: unit, integration, system and acceptance.*
3. *Black box tests are reproducible.*
4. *Find software bugs early.*
5. *Facilitates change.*
6. *The environment the program is running is also tested.*
7. *The invested effort can be used multiple times.*
8. *More effective on larger units of code than glass box testing.*
9. *Tester needs no knowledge of implementation, including specific programming languages.*
10. *Tests are done from a user's point of view.*
11. *Will help to expose any ambiguities or inconsistencies in the specifications.*

At this stage of our project, we had applied Black box testing method in unit testing phase of the software testing to have individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, tested to determine whether they are fit for use or not.

5.2. Testing Environment:

We have done manual testing. For this purpose, we have given some valid input to the application to get expected output and some wrong input to make sure the validity and responsiveness of the system.

5.3. Test Cases:

A test case is a specification of the inputs, execution conditions, testing procedures and expected results that define a single test to be executed to achieve particular functional requirements.

5.3.1. Test Case 1:

Scenario: This test case is generated to test the password and username format validation of system during the signup process when user is trying to get register into the system.

- **Name:** Signup
- **Activity:** Signup Activity
- **Message:** Signed Up Successfully & Verification Email has been sent.
  Result of Operation: Successful

Test Case Signup Validation

Name: Signup	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To Register a Customer Successfully by prompting the all fields as valid.	***Test ID:*** 01
Version: 2	***Test Type:*** *Functional Testing*
Input: **Name:** Tayyab Raja **Username:** tayyab6677 **Email:** tayyab786@gmail.com **Password:** tayyab007@ **Phone No:** 0312-3456789 **Address:** Park Road, Taramari Chowk, Islamabad.	
Expected Result: Signed Up Successfully & Verification Email has been Sent.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 39: Test Case Signup Validation

5.3.2. Test Case 1:

Scenario: This test case is generated to test the required field's validation while user trying to register / login to the system.

- **Name:** Signup
- **Activity:** Signup Activity
- **Message:** Please fill out all the fields first
- **Result of Operation:** Failed

Test Case for Requires Field Validation

Name: Signup	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To Register a Customer by prompting invalid fields.	***Test ID:*** 02
Version: 2	***Test Type:*** *Functional Testing*
Input: **Name:** null **Username:** null **Email:** null **Password:** null **Phone No:** null **Address:** null	
Expected Result: The Username is required. Password is required. Phone No is required. Name is required. Address is required.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 40: Test Case for Requires Field Validation

5.3.3. Test Case 1:

Scenario: This test case is generated to test the Name, Username, Password, Phone no and Address for invalid field validation of system during the signup process when user is entering the incorrect fields.

- **Name:** Signup
- **Activity:** Signup Activity
- **Message:** Password min length should be 6
- **Result of Operation:** Failed

Test Case for Invalid Field Validation

Name: Signup	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To Register a Customer by giving password of invalid length.	***Test ID:*** 03
Version: 2	***Test Type:*** *Functional Testing*
Input: **Name:** Qasim **Username:** qasim420 **Email:** qasim1@gmail.com **Password:** 1166 **Phone No:** 03411551466 **Address:** Strre 9, G-10, Garden Plaza, Islamabad.	
Expected Result: Passsword min length should be 6!	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 41: Test Case for Invalid Field Validation

5.3.4. Test Case 1:

Scenario: This test case is generated to test the validation and authorization while user trying to login into the system with the valid credentials.

- **Name:** Login
- **Activity:** Login Activity
- **Message:** Login Successful
- **Result of Operation:** Successful

Test Case Login Authorization

Name: Login	
Date: 19th November 2021	
System: Sip n Snack	
Objective: To Authorize a user and log in to the system.	***Test ID:*** 04
Version: 2	***Test Type:*** *Functional Testing*
Input: **Username:** tayyab6677 **Password:** tayyab007@	
Expected Result: Login Successful.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 42: Test Case Login Authorization

5.3.5. Test Case 1:

<u>Scenario:</u> This test case is generated to test the validation and authorization while user trying to login into the system with the invalid credentials.

- **Name:** Login
- **Activity:** Login Activity
- **Message:** Invalid Credentials Provided
- **Result of Operation:** Failed

Test Case Login Authorization (Failed)

Name: Login	
Date: 19th November 2021	
System: Sip n Snack	
Objective: Try to authorize user with invalid credentials.	***Test ID:*** 05
Version: 2	***Test Type:*** *Functional Testing*
Input: **Username:** tayyab6677 **Password:** tayyab1234	
Expected Result: Invalid Credentials Provided.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 43: Test Case Login Authorization (Failed)

5.3.6. Test Case 1:

Scenario: This test case is generated to test the field's validation while manager try to add new Item to the system.

- **Name:** Add Items
- **Activity:** Add Item Activity
- **Message:** Item Added Successfully
- **Result of Operation:** Successful

Add Items Validation

Name: Add Items	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To add a new item by the manager by giving the all fields as valid input.	***Test ID:*** 06
Version: 2	***Test Type:*** *Functional Testing*
Input: **Item Id:** item20 **Name:** Plain Masala Fries **Category:** Fries **Price:** 150 **Size:** Regular **Description:** Plain masala fries made with fresh potatoes served with tasteful mayo and ketchup.	
Expected Result: Item Added Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 44: Add Items Validation

5.3.7. Test Case 1:

Scenario: This test case is generated to test the field's validation while manager try to add new Item to the system with already existing item id.

- **Name:** Add Items
- **Activity:** Add Item Activity
- **Message:** Item already exists
- **Result of Operation:** Failed

Add Items Validation (Invalid)

Name: Add Items	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To add a new item by the manager by giving the all fields as invalid input.	***Test ID:*** 07
Version: 2	***Test Type:*** *Functional Testing*
Input: **Item Id:** item20 **Name:** Plain Masala Fries **Category:** Fries **Price:** 199 **Size:** Regular **Description:** Plain masala fries made with fresh potatoes served with tasteful mayo and ketchup.	
Expected Result: Item already exists.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 45: Add Items Validation (Invalid)

5.3.8. Test Case 1:

Scenario: This test case is generated to test the field's validation while manager try to add new Item to the system with invalid input fields.

- **Name:** Add Items
- **Activity:** Add Item Activity
- **Message:** Item Price is Required
- **Result of Operation:** Failed

Add Items Validation (Invalid)

Name: Add Items	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To add a new item by the manager by giving the field as invalid input.	***Test ID:*** 08
Version: 2	***Test Type:*** *Functional Testing*
Input: **Item Id:** item09 **Name:** Italian Soda **Category:** Chilled Drinks **Price:** null **Size:** Regular **Description:** Italian tradition soda with colorful taste that refreshes your tastebuds.	
Expected Result: Item Price is required.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 46: Add Items Validation (Invalid)

5.3.9. Test Case 1:

Scenario: This test case is generated to test deletion of specific food item from system.

- **Name:** Delete Item
- **Activity:** Delete Item
- **Message:** Item Deleted Successfully
- **Result:** Passed

Delete Items

Name: Delete Item	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To delete a specific item from the system by the manager.	***Test ID:*** 09
Version: 2	***Test Type:*** *Functional Testing*
Input: **Clicked:** Delete Item Button	
Expected Result: Item Deleted Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 47: Delete Items

5.3.10. Test Case 10:

Scenario: This test case is generated to test the field's validation while manager try to update existing Item in system.

- **Name:** Update Item
- **Activity:** Update Item Activity
- **Message:** Item Updated Successfully
- **Result:** Passed

Update Items Validation

Name: Update Item	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To update existing item in the system by providing valid inputs in the fields.	***Test ID:*** 10
Version: 2	***Test Type:*** *Functional Testing*
Input: **Item Id:** item20 **Name:** Mayo Fries **Category:** Fries **Price:** 200 **Size:** Medium **Description:** Plain masala fries made with fresh potatoes served with tasteful mayo and ketchup.	
Expected Result: Item Updated Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 48: Update Items Validation

5.3.11. Test Case 10:

Scenario: This test case is generated to test the field's validation while manager try to update existing Item in system with invalid input fields.

- **Name:** Update Item
- **Activity:** Update Item Activity
- **Message:** Item Name is required.
- **Result of Operation:** Failed

Update Items Validation (Invalid)

Name: Update Item	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To update existing item in the system by providing invalid inputs in the fields.	***Test ID:*** 11
Version: 2	***Test Type:*** *Functional Testing*
Input: **Item Id:** item20 **Name:** *null* **Category:** Fries **Price:** 200 **Size:** Regular **Description:** Plain masala fries made with fresh potatoes served with tasteful mayo and ketchup.	
Expected Result: Item Name is required.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 49: Update Items Validation (Invalid)

5.3.12. Test Case 15:

Scenario: This test case is generated to test the field's validation while manager create a new bikers account with invalid inputs.

- **Name:** Add Biker
- **Activity:** Add Biker Activity
- **Message:** Biker Name is required.
- **Result of Operation:** Failed

Add Biker Validation (Invalid)

Name: Add Biker	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To add a new biker by the manager by giving the fields as invalid input.	***Test ID:*** 12
Version: 2	***Test Type:*** *Functional Testing*
Input: **Username:** biker7 **Name:** *null* **Password:** 1122 **Phone No:** 0312-1234567 **Address:** Home 5, St # 2, ABC Town, ISB.	
Expected Result: Biker Name is required.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 50: Add Biker Validation (Invalid)

5.3.13. Test Case 15:

Scenario: This test case is generated to test the field's validation while manager try to add new biker account to the system with valid input fields.

- **Name:** Add Biker
- **Activity:** Add Biker Activity
- **Message:** Biker Added Successfully
- **Result of Operation:** Successful

Add Biker Validation

Name: Add Biker	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To add a new biker by the manager by giving the fields as invalid input.	***Test ID:*** 13
Version: 2	***Test Type:*** *Functional Testing*
Input: **Username:** biker3 **Name:** Farrukh Hussain **Password:** farrukh11 **Phone No:** 0312-1234567 **Address:** PWD, Islamabad.	
Expected Result: Biker Added Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 51: Add Biker Validation

5.3.14. Test Case 15:

<u>Scenario:</u> This test case is generated to test deletion of specific biker account from system.

- **Name:** Delete Biker
- **Activity:** Delete Biker Activity
- **Message:** Biker Deleted Successfully
- **Result of Operation:** Successful

Delete Biker

Name: Delete Biker	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To delete a specific account of biker from the system by the manager.	***Test ID:*** 14
Version: 2	***Test Type:*** *Functional Testing*
Input: **Clicked:** Delete Biker Button	
Expected Result: Biker deleted Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 52: Delete Biker

5.3.15. Test Case 15:

Scenario: This test case is generated to test the field's validation while manager try to update Biker account in system with valid inputs.

- **Name:** Update Biker
- **Activity:** Update Biker Activity
- **Message:** Biker Updated Successfully
- **Result of Operation:** Successful

Update Bikers Validation

Name: Update Biker	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To Update Bikers Detail by providing the valid field inputs.	***Test ID:*** 15
Version: 2	***Test Type:*** *Functional Testing*
Input: **Username:** biker3 **Name:** Farrukh Hassan **Phone No:** 0312-1234567 **Address:** Near Masjid Street, Gujranwala, and Islamabad.	
Expected Result: Biker Updated Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 53: Update Bikers Validation

5.3.16. Test Case 15:

Scenario: This test case is generated to test the field's validation while manager try to update existing Biker account in system with invalid input fields.

- **Name:** Update Biker
- **Activity:** Update Biker Activity
- **Message:** Biker Address is required.
- **Result of Operation:** Failed

Update Biker Validation (Invalid)

Name: Update Biker	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To Update Bikers Detail by providing the invalid field inputs.	***Test ID:*** 16
Version: 2	***Test Type:*** *Functional Testing*
Input: **Username:** biker3 **Name:** Farrukh Hassan **Phone No:** 0312-1234567 **Address:** null	
Expected Result: Biker Address is required.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 54: Update Biker Validation (Invalid)

5.3.17. Test Case 15:

Scenario: This test case is generated to test the block of customer account by manager after clicking the block button.

- **Name:** Block Customer
- **Activity:** Block Customer Activity
- **Message:** Customer Blocked Successfully
- **Result of Operation:** Successful

Block Customer

Name: Block Customer	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To block a specific account of customer.	***Test ID:*** 17
Version: 2	***Test Type:*** *Functional Testing*
Input: **Clicked:** Block Icon	
Expected Result: Customer Blocked Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 55: Block Customer

5.3.18. Test Case 15:

Scenario: This test case is generated to test the unblock customer account by manager after clicking the unblock button.

- **Name:** Unblock Customer
- **Activity:** Unblock Customer Activity
- **Message:** Customer Unblocked Successfully
- **Result of Operation:** Successful

Unblock Customer

Name: Unblock Customer	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To Unblock a specific account of customer that is being already blocked.	***Test ID:*** 18
Version: 2	***Test Type:*** *Functional Testing*
Input: **Clicked:** Unblock Icon	
Expected Result: Customer Unblocked Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 56: Unblock Customer

5.3.19. Test Case 15:

Scenario: This test case is generated to test the updation of banners of popular items by the manager after giving the image as an input.

- **Name:** Update Banners
- **Activity:** Update Banners Activity
- **Message:** Banner Uploaded Successfully
- **Result of Operation:** Successful

Update Banners

Name: Update Banners	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To update a banner by selecting an image for banner.	***Test ID:*** 19
Version: 2	***Test Type:*** *Functional Testing*
Input: **Banner Image:** image.png	
Expected Result: Banner Uploaded Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 57: Update Banners

5.3.20. Test Case 15:

Scenario: This test case is generated to test the updation of banners of popular items by the manager without selecting any image

- **Name:** Update Banners
- **Activity:** Update Banners Activity
- **Message:** Image URI is not found
- **Result of Operation:** Failed

Update Banners (failed)

Name: Update Banners	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To update a banner without selecting any image.	***Test ID:*** 20
Version: 2	***Test Type:*** *Functional Testing*
Input: **Banner Image:** *null*	
Expected Result: Image URI is not found.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 58: Update Banners (failed)

5.3.21. Test Case 15:

Scenario: This test case is generated to test add to cart functionality in which the customer hasn't added the same item in cart already.

- **Name:** Add to Cart
- **Activity:** Add to Cart Activity
- **Message:** Item Sucessfully Added to Cart.
- **Result of Operation:** Successful

Add to Cart

Name: Add to Cart	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To add a fresh item in the cart.	***Test ID:*** 21
Version: 2	***Test Type:*** *Functional Testing*
Input: **Name:** Plain Masala Fries **Category:** Fries **Price:** 150 **Size:** Regular **Quantity:** 2 **Total Price:** 300 Rs. **Description:** Plain masala fries made with fresh potatoes served with tasteful mayo and ketchup.	
Expected Result: Item Successfully added to cart.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 59: Add to Cart

5.3.22. Test Case 15:

Scenario: This test case is generated to test add to cart functionality in which the customer has added the same item in cart already.

- **Name:** Add to Cart
- **Activity:** Add to Cart Activity
- **Message:** Item already Added
- **Result of Operation:** Failed

Add to Cart (failed)

Name: Add to Cart	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To add an existing item in the cart.	***Test ID:*** 22
Version: 2	***Test Type:*** *Functional Testing*
Input: **Name:** Plain Masala Fries **Category:** Fries **Price:** 150 **Size:** Regular **Quantity:** 2 **Total Price:** 300 Rs. **Description:** Plain masala fries made with fresh potatoes served with tasteful mayo and ketchup.	
Expected Result: Item already added.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 60: Add to Cart (failed)

5.3.23. Test Case 15:

Scenario: This test case is generated to test deletion of specific item from the cart.

- **Name:** Delete from Cart
- **Activity:** Cart Activity
- **Message:** Item Deleted Successfully from Cart.
- **Result of Operation:** Successful

Delete from Cart

Name: Delete from Cart	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To delete a specific item from the cart.	***Test ID:*** 23
Version: 2	***Test Type:*** *Functional Testing*
Input: **Clicked:** Delete Icon	
Expected Result: Item Deleted Successfully from Cart.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 61: Delete from Cart

5.3.24. Test Case 15:

Scenario: This test case is generated to test the uploading of receipt of payment in case if the customer opt for online payment and upload the image of payment receipt.

- **Name:** Online Payment
- **Activity:** Payment Receipt Activity
- **Message:** Receipt Uploaded Successfully
- **Result of Operation:** Successful

Online Payment

Name: Online Payment	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To upload the image of payment receipt by the customer.	***Test ID:*** 24
Version: 2	***Test Type:*** *Functional Testing*
Input: **Banner Image:** receipt.png	
Expected Result: Receipt Uploaded Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 62: Online Payment

5.3.25. Test Case 15:

Scenario: This test case is generated to test the uploading of receipt of payment without selecting any image.

- **Name:** Online Payment
- **Activity:** Payment Receipt Activity
- **Message:** You have not selected any image.
- **Result of Operation:** Failed

Online Payment (failed)

Name: Online Payment	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To opt for online payment without attaching any image.	***Test ID:*** 25
Version: 2	***Test Type:*** *Functional Testing*
Input: **Banner Image:** *null*	
Expected Result: You have not selected any image.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 63: Online Payment (failed)

5.3.26. Test Case 15:

Scenario: This test case is generated to test the placing order of the items currently present in cart by the customer.

- **Name:** Place Order
- **Activity:** Place Order Activity
- **Message:** Order Placed Successfully
- **Result of Operation:** Successful

Place Order

Name: Place Order	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To place the order of few items that were present in cart.	***Test ID:*** 26
Version: 2	***Test Type:*** *Functional Testing*
Input: **Items:** 1. Peri Peri Burger x 2 2. Italian Pizza x 2 3. Caramel Coffee x 4 **Bill:** 2155	
Expected Result: Order Placed Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 64: Place Order

5.3.27. Test Case 15:

Scenario: This test case is generated to test the placing order with empty cart by customer.

- **Name:** Place Order
- **Activity:** Place Order Activity
- **Message:** Your cart is empty
- **Result of Operation:** Failed

Place Order (Failed)

Name: Place Order	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To place the order having empty cart.	***Test ID:*** 27
Version: 2	***Test Type:*** *Functional Testing*
Input: **Cart Items:** *null*	
Expected Result: Your cart is empty.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 65: Place Order (Failed)

5.3.28. Test Case 15:

Scenario: This test case is generated to test the cancel of order when the customer wants to cancel the current order.

- **Name:** Cancel Order
- **Activity:** Cancl Order Activity
- **Message:** Order was Cancelled
- **Result of Operation:** Successful

Cancel Order

Name: Cancel Order	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To cancel an order that is being already confirmed.	***Test ID:*** 28
Version: 2	***Test Type:*** *Functional Testing*
Input: **Clicked:** Cancel Order Button	
Expected Result: Order was Canclled.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 66: Cancel Order

5.3.29. Test Case 15:

Scenario: This test case is generated to test the submission of general feedback given by the customer to system.

- **Name:** Submit Feedback
- **Activity:** Feedback Activity
- **Message:** Feedback Submitted Successfully
- **Result of Operation:** Successful

Submit Feedback

Name: Submit Feedback	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To submit a general feedback given the customer.	***Test ID:*** 29
Version: 2	***Test Type:*** *Functional Testing*
Input: **Feedback Text:** The fries was awesome but if you add some more crisp in it, I bet it will be more delicious ☺	
Expected Result: Feedback Submitted Successfully	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 67: Submit Feedback

5.3.30. Test Case 15:

Scenario: This test case is generated to test the submission of general feedback with empty feedback text.

- **Name:** Submit Feedback
- **Activity:** Feedback Activity
- **Message:** You must need to Enter Feedback
- **Result of Operation:** Failed

Submit Feedback (failed)

Name: Submit Feedback	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To submit a general feedback given the feedback text is empty.	***Test ID:*** 30
Version: 2	***Test Type:*** *Functional Testing*
Input: **Feedback Text:** *null*	
Expected Result: You must need to Enter Feedback	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 68: Submit Feedback (failed)

5.3.31. Test Case 15:

Scenario: This test case is generated to test the field's validation while user try to change their account password with valid inputs.

- **Name:** Change Password
- **Activity:** Password Change Activity
- **Message:** Password Updated Successfully
- **Result of Operation:** Successful

Change Password Validation

Name: Change Password	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To change the password of account by providing valid inputs.	***Test ID:*** 31
Version: 2	***Test Type:*** *Functional Testing*
Input: **Old Password:** pakistan009# **New Password:** faisal77*@ **Re-enter Password:** faisal77*@	
Expected Result: Password Updated Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 69: Change Password Validation

5.3.32. Test Case 15:

Scenario: This test case is generated to test the field's validation while user try to change their account password with invalid inputs.

- **Name:** Change Password
- **Activity:** Password Change Activity
- **Message:** Old Password is not matching
- **Result of Operation:** Failed

Change Password Validation (failed)

Name: Change Password	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To change the password of account by providing invalid inputs.	***Test ID:*** 32
Version: 2	***Test Type:*** *Functional Testing*
Input: **Old Password:** alikhan007 **New Password:** alikingg1234 **Re-enter Password:** alikingg1234	
Expected Result: Old Password is not matching	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 70: Change Password Validation (failed)

5.3.33. Test Case 15:

Scenario: This test case is generated to test the field's validation while user try to update thier profile settings with valid inputs.

- **Name:** Manage Profile
- **Activity:** Manage Profile Activity
- **Message:** Profile Updated Successfully
- **Result of Operation:** Successful

Manage Profile Validation

Name: Manage Profile	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To update user profile settings with valid inputs.	***Test ID:*** 33
Version: 2	***Test Type:*** *Functional Testing*
Input: **Username:** biker3 **Name:** Farrukh Hassan **Phone No:** 0336-9866301 **Address:** Near Masjid Street, Gujranwala, and Islamabad.	
Expected Result: Profile Updated Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 71: Manage Profile Validation

5.3.34. Test Case 15:

Scenario: This test case is generated to test the field's validation while user try to update thier profile settings with invalid inputs.

- **Name:** Manage Profile
- **Activity:** Manage Profile Activity
- **Message:** Phone No is required
- **Result of Operation:** Failed

Manage Profile Validation (failed)

Name: Manage Profile	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To update user profile settings with invalid inputs.	***Test ID:*** 34
Version: 2	***Test Type:*** *Functional Testing*
Input: **Username:** biker3 **Name:** Farrukh Hassan **Phone No:** *null* **Address:** Near Masjid Street, Gujranwala, and Islamabad.	
Expected Result: Profile update failed.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 72: Manage Profile Validation (failed)

5.3.35. Test Case 15:

Scenario: This test case is generated to test the accepting of order that is being placed by the customer.

- **Name:** Accept Order
- **Activity:** Orders Activity
- **Message:** Order Accepted Successfully
- **Result of Operation:** Successful

Accept Order

Name: Accept Order	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To accept the order that was placed by customer.	***Test ID:*** 35
Version: 2	***Test Type:*** *Functional Testing*
Input: **Order Info:** Items, bill, quantity. **Clicked:** Accept Order Button.	
Expected Result: Order Accepted Successfully	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 73: Accept Order

5.3.36. Test Case 15:

Scenario: This test case is generated to test the generating of bill of order that is placed by the customer.

- **Name:** Generate Bill
- **Activity:** Generat Bill Activity
- **Message:** Bill Generated Successfully
- **Result of Operation:** Successful

Generate Bill

Name: Generate Bill	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To generate the bill of order.	***Test ID:*** 36
Version: 2	***Test Type:*** *Functional Testing*
Input: **Order Info:** Items, bill, quantity. **Clicked:** Generate Bill Button.	
Expected Result: Bill Generated Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 74: Generate Bill

5.3.37. Test Case 15:

Scenario: This test case is generated to test the generating of reports orders that are being placed on specific date.

- **Name:** Generate Report
- **Activity:** Generat Report Activity
- **Message:** Report Generated Successfully
- **Result of Operation:** Successful

Generate Report

Name: Generate Report	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To generate the reports of sales.	***Test ID:*** 37
Version: 2	***Test Type:*** *Functional Testing*
Input: **Date:** 26 / 10 / 2021 **Clicked:** Generate Report Button.	
Expected Result: Report Generated Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 75: Generate Report

5.3.38. Test Case 15:

Scenario: This test case is generated to test the field's validation while admin create a new managers account with invalid inputs.

- **Name:** Add Manager
- **Activity:** Add Manager Activity
- **Message:** Manager Name is required
- **Result of Operation:** Failed

Add Manager Validation (Invalid)

Name: Add Manager	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To add a new manager by the admin by giving the fields as invalid input.	***Test ID:*** 38
Version: 2	***Test Type:*** *Functional Testing*
Input: **Username:** mng7 **Name:** *null* **Password:** 1122 **Phone No:** 0312-1234567	
Expected Result: Manager Name is required.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 76: Add Manager Validation (Invalid)

5.3.39. Test Case 15:

Scenario: This test case is generated to test the field's validation while admin try to add new biker account to the system with valid input fields.

- **Name:** Add Manager
- **Activity:** Add Manager Activity
- **Message:** Manager Added Successfully
- **Result of Operation:** Successful

Add Manager Validation

Name: Add Manager	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To add a new manager by the admin by giving the fields as invalid input.	***Test ID:*** 39
Version: 2	***Test Type:*** *Functional Testing*
Input: **Username:** mng009 **Name:** Farrukh Hussain **Password:** farrukh11 **Phone No:** 0312-1234567	
Expected Result: Manager added Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 77: Add Manager Validation

5.3.40. Test Case 40:

Scenario: This test case is generated to test deletion of specific manager account from system.

- **Name:** Delete Manager
- **Activity:** Delete Manager Activity
- **Message:** Manager Deleted Successfully
- **Result of Operation:** Successful

Delete Manager

Name: Delete Manager	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To delete a specific account of manager from the system by the admin.	***Test ID:*** 40
Version: 2	***Test Type:*** *Functional Testing*
Input: **Clicked:** Delete Manager Button	
Expected Result: Manager deleted Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 78: Delete Manager

5.3.41. Test Case 40:

Scenario: This test case is generated to test the field's validation while admin try to update Manager Account in system with valid inputs.

- **Name:** Update Manager
- **Activity:** Update Manager Activity
- **Message:** Manager Updated Successfully
- **Result of Operation:** Successful

Update Manager Validation

Name: Update Manager	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To Update Managers detail by providing the valid field inputs.	***Test ID:*** 41
Version: 2	***Test Type:*** *Functional Testing*
Input: **Username:** mng7 **Name:** Farrukh Hassan **Phone No:** 0312-1234567 **Address:** Near Masjid Street, Gujranwala, and Islamabad.	
Expected Result: Manager Updated Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 79: Update Manager Validation

5.3.42. Test Case 40:

Scenario: This test case is generated to test the field's validation while admin try to update existing Manager Account in system with invalid input fields.

- **Name:** Update Manager
- **Activity:** Update Manager Activity
- **Message:** Phone No is required
- **Result of Operation:** Failed

Update Manager Validation (Invalid)

Name: Update Manager	
Date: 18th November 2021	
System: Sip n Snack	
Objective: To Update Managers detail by providing the invalid field inputs.	***Test ID:*** 42
Version: 2	***Test Type:*** *Functional Testing*
Input: **Username:** mng1 **Name:** Ali **Phone No:** null	
Expected Result: Phone No is required.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 80: Update Manager Validation (Invalid)

5.3.43. Test Case 40:

Scenario: This test case is generated to test the field's validation while manager add occurred expense to a system with invalid inputs.

- **Name:** Add Expense
- **Activity:** Add Expense Activity
- **Message:** Expense Amount is required.
- **Result of Operation:** Failed

Add Expense Validation (Invalid)

Name: Add Expense	
Date: 21st January 2022	
System: Sip n Snack	
Objective: To add a new occurred expense by input the invalid fields.	***Test ID:*** 43
Version: 3	***Test Type:*** *Functional Testing*
Input: **Expense Category:** Crockery **Expense Amount:** *null*	
Expected Result: Expense Amount is Required.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 81: Add Expense Validation (Inavlid)

5.3.44. Test Case 40:

Scenario: This test case is generated to test the field's validation while manager try to add new occurred expense to the system with valid input fields.

- **Name:** Add Expense
- **Activity:** Add Expense Activity
- **Message:** Expense Added Successfully
- **Result of Operation:** Successful

Add Expense Validation

Name: Add Expense	
Date: 21st January 2022	
System: Sip n Snack	
Objective: To add a new expense by the manager by giving the fields as input.	***Test ID:*** 44
Version: 3	***Test Type:*** *Functional Testing*
Input: **Expense Category:** Maintenance **Expense Amount:** 900	
Expected Result: Expense added Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 82: Add Expense Validation

5.3.45. Test Case 40:

Scenario: This test case is generated to test the field's validation while user report an issue to system with invalid inputs.

- **Name:** Report Issue
- **Activity:** Report Issue Activity
- **Message:** Issue text is required.
- **Result of Operation:** Failed

Report Issue Validation (Invalid)

Name: Report Issue	
Date: 21st January 2022	
System: Sip n Snack	
Objective: To report a new issue by the user by giving the fields as invalid input.	***Test ID:*** 45
Version: 3	***Test Type:*** *Functional Testing*
Input: **Issue Text:** null	
Expected Result: Issue text is required.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 83: Report Issue Validation (Invalid)

5.3.46. Test Case 40:

Scenario: This test case is generated to test the field's validation while user try to report issue to the system with valid input fields.

- **Name:** Report Issue
- **Activity:** Report Issue Activity
- **Message:** Issue Reported Successfully
- **Result of Operation:** Successful

Report Issue Validation

Name: Report Issue	
Date: 21st January 2022	
System: Sip n Snack	
Objective: To report an issue by the user by giving the fields as invalid input.	***Test ID:*** 46
Version: 3	***Test Type:*** *Functional Testing*
Input: **Issue Text:** There is a little lag while login to the app.	
Expected Result: Issue Reported Successfully.	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 84: Report Issue Validation

5.3.47. Test Case 40:

Scenario: This test case is generated to test the validation while customer try to give feedback about delivery or food service with valid input fields.

- **Name:** Submit Feedback
- **Activity:** View Orders Activity
- **Message:** Your Feedback Submitted Successfully
- **Result of Operation:** Successful

Submit Feedback Validation

Name: Submit Feedback	
Date: 17th February 2022	
System: Sip n Snack	
Objective: To submit a feedback about the food or delivery service by giving the fields as valid input.	***Test ID:*** 47
Version: 3	***Test Type:*** *Functional Testing*
Input: **Feedback Text:** Food taste is very good. **Rating Stars: 5** stars	
Expected Result: Feedback Submitted Successfully	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 85: Submit Feedback Validation (Valid)

5.3.48. Test Case 40:

Scenario: This test case is generated to test the validation while customer try to give feedback about delivery or food service with invalid input fields.

- **Name:** Submit Feedback
- **Activity:** View Orders Activity
- **Message:** Feedback text must be filled
- **Result of Operation:** Successful

Submit Feedback Validation

Name: Submit Feedback	
Date: 17th February 2022	
System: Sip n Snack	
Objective: To submit a feedback about the food or delivery service by giving the fields as invalid input.	***Test ID:*** 48
Version: 3	***Test Type:*** *Functional Testing*
Input: **Feedback Text:** null **Rating Stars:** null	
Expected Result: Feedback text must be filled	
Actual Result: As Expected.	
Verdict: *Passed*	

Table 86: Submit Feedback (Invalid)

Chapter 6

System Deployment

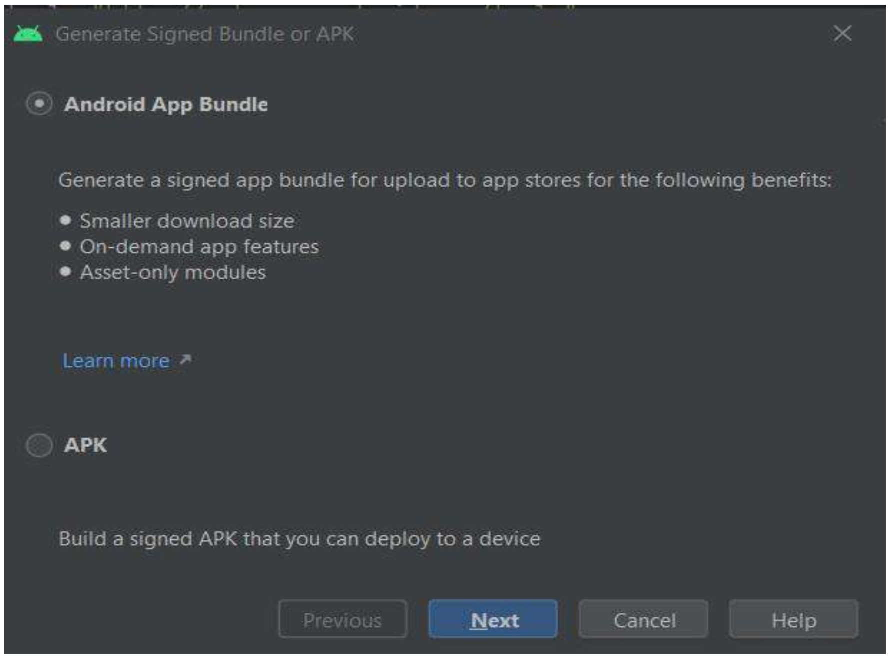
6.1. Installation / Deployment Process Description:

For deployment, we will provide user with .apk or .aab file. Through these files, user should be able to install the applications in android devices. User can simply open the .apk file in their device and install the application to use the services of system. However the credentials for restaurant side were given to customer and using these, the admin can create further accounts of managers and managers basically manage the whole ordering system etc.

The application is also being deployed on Play Store. The deployment were done by using different steps… These steps were given in sequence in below sections:

6.1.1. Choose between apk OR aab.

It’s upon us that which file we want to create and upload it on Play Store.


Generate Signed Bundle or APK

(●) **Android App Bundle**

Generate a signed app bundle for upload to app stores for the following benefits:

- Smaller download size
- On-demand app features
- Asset-only modules

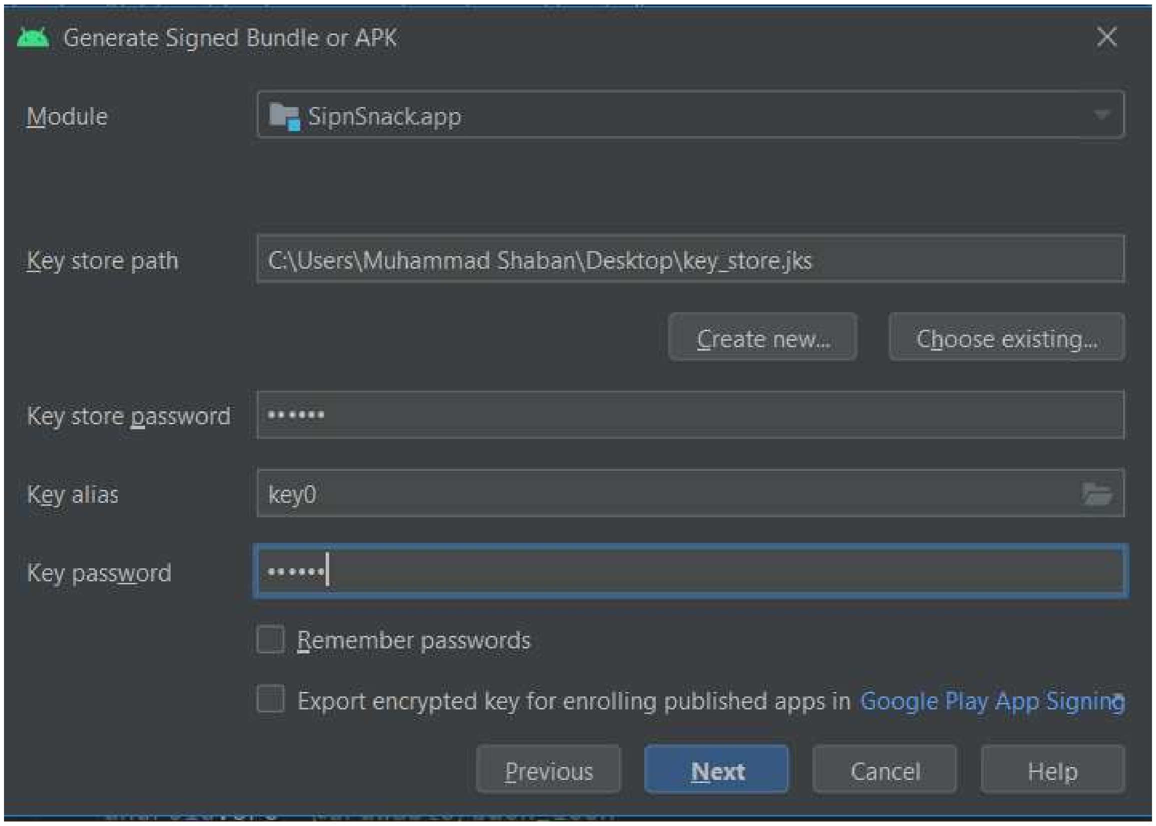
( ) **APK**

Build a signed APK that you can deploy to a device

Figure 116: Select Apk or Aab

6.1.2. Create Keystore for App.

In this step, we have to create a keystore for app and choose a suitable password for that. The keystore file with extension '.jks' and its password must be saved. Through this, we can release a new version of the application in future.


Generate Signed Bundle or APK

Module: SipnSnack.app

Key store path: C:\Users\Muhammad Shaban\Desktop\key_store.jks

Key store password: ••••••

Key alias: key0

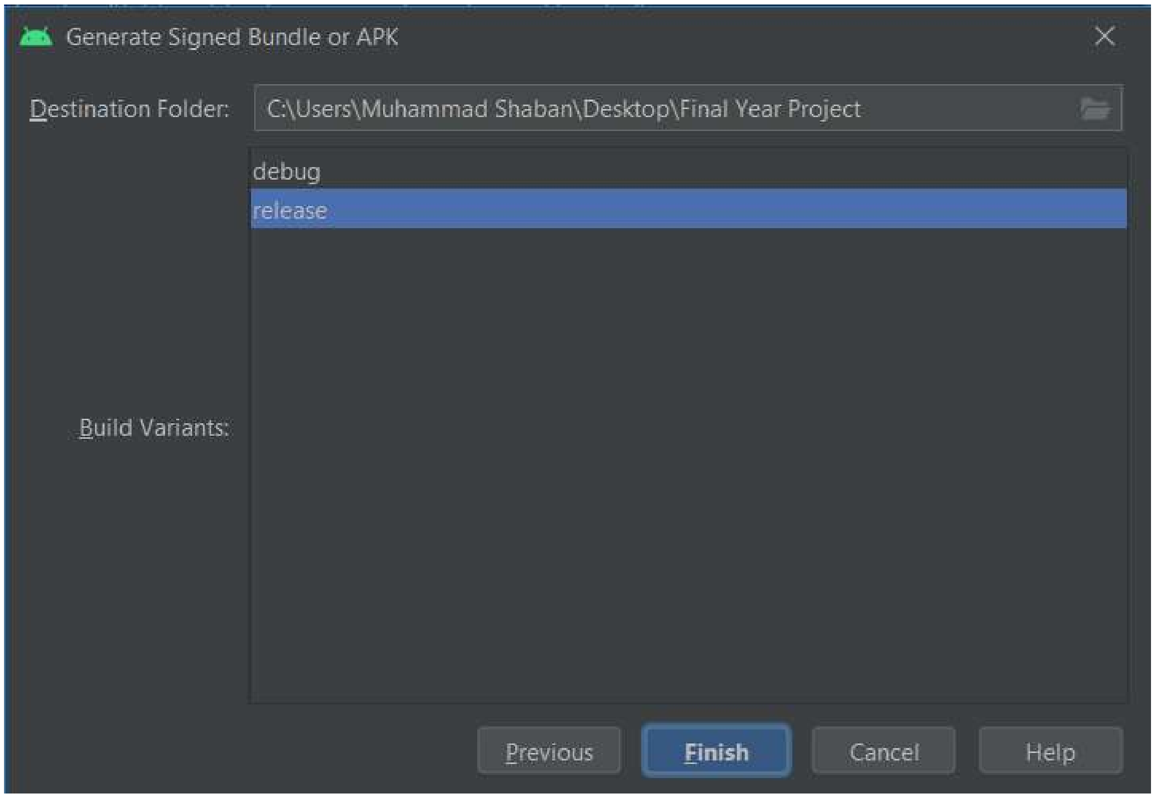
Key password: ••••••

Create new...
Choose existing...
Remember passwords
Export encrypted key for enrolling published apps in Google Play App Signing
Previous
Next
Cancel
Help

Figure 117: Create Keystore

6.1.3. Choose debug or release mode.

In this step, we have to choose mode for our app. Debug mode is being chosen when we have to launch our app just for some testing purposes and finally we launch the release mode that contains all the functionality.


Generate Signed Bundle or APK

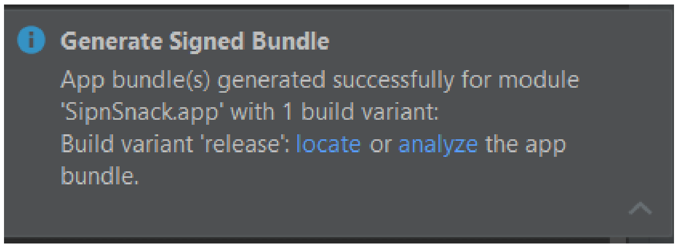
Destination Folder: C:\Users\Muhammad Shaban\Desktop\Final Year Project

Build Variants:
- debug
- release

Figure 118: Mode of Apk

6.1.4. Apk Generated Successfully.

Continued …

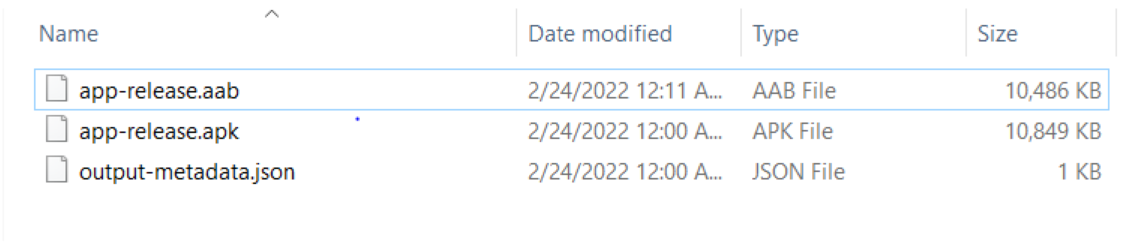

Generate Signed Bundle

App bundle(s) generated successfully for module 'SipnSnack.app' with 1 build variant:
Build variant 'release': locate or analyze the app bundle.

Figure 119: Apk Generated Successfully

6.1.5. Files for app.

As said above, we have option to choose between apk or aab file.



Name	Date modified	Type	Size
app-release.aab	2/24/2022 12:11 A...	AAB File	10,486 KB
app-release.apk	2/24/2022 12:00 A...	APK File	10,849 KB
output-metadata.json	2/24/2022 12:00 A...	JSON File	1 KB

Figure 120: Files for App