

Software Requirement Specification

(ABC Restaurant)

Student reg. no. – st20315315

Table of Contents

Introduction.....	4
TASK A.....	4
System Design and Development	4
UML Diagrams	4
Use Case Diagram.....	4
Class Diagram.....	6
ER Diagram	7
Sequence Diagram	8
Assumptions.....	10
TASK B.....	10
Development of the System	10
Set of Interfaces with Their Inputs and Outputs	11
Interfaces for admin	13
User management.....	14
Restaurant management	19
Gallery Management.....	24
Service management	29
Offers and promotions management	33
Manage payments	38
Reservation management.....	43
Query management	51
Interfaces for staff.....	53
Payment management	53
Reservation management.....	54
Query management	56
Interfaces for customer	57
Use of Design Patterns.....	62
MVC design pattern.....	62
Observer design pattern	64
Repository pattern.....	68
Use of OOP Concepts	69
Inheritance.....	69
Polymorphism.....	72
Database.....	74

Report Generation.....	76
Email Confirmation	90
Use of sessions and cookies.....	93
Functional Requirements	95
Non-functional Requirements.....	96
TASK C.....	97
Quality Assurance and testing.....	97
Manual test case report	97
Test cases	97
Test Automation	105
TASK D	110
Repository Management	110

Introduction

This report depicts the RESTful API developed for ABC Restaurant chain. It is designed to streamline the business activities of ABC restaurant with a seamless manner. The system is developed with the assistance of various languages and frameworks. For the front-end, Angular was used and for the back-end C# was used. As the database MySQL was chosen. This architecture is supporting various functionalities such as user management, restaurant management, service management, payment management, query management and much more.

This report wishes to cover the entire development process, initial steps that were taken to design and implement the system, the testing process etc.

TASK A

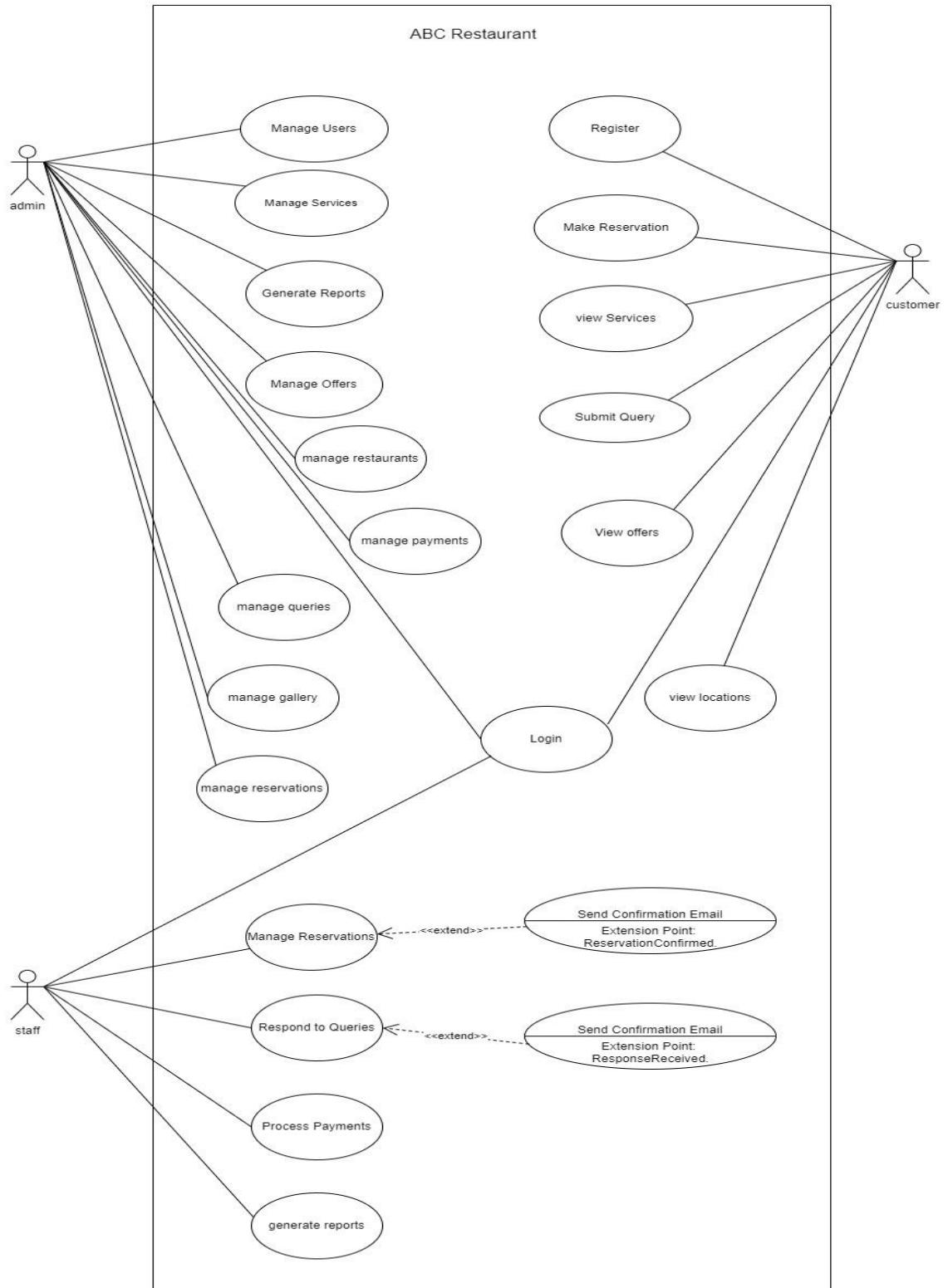
System Design and Development

UML Diagrams

Use Case Diagram

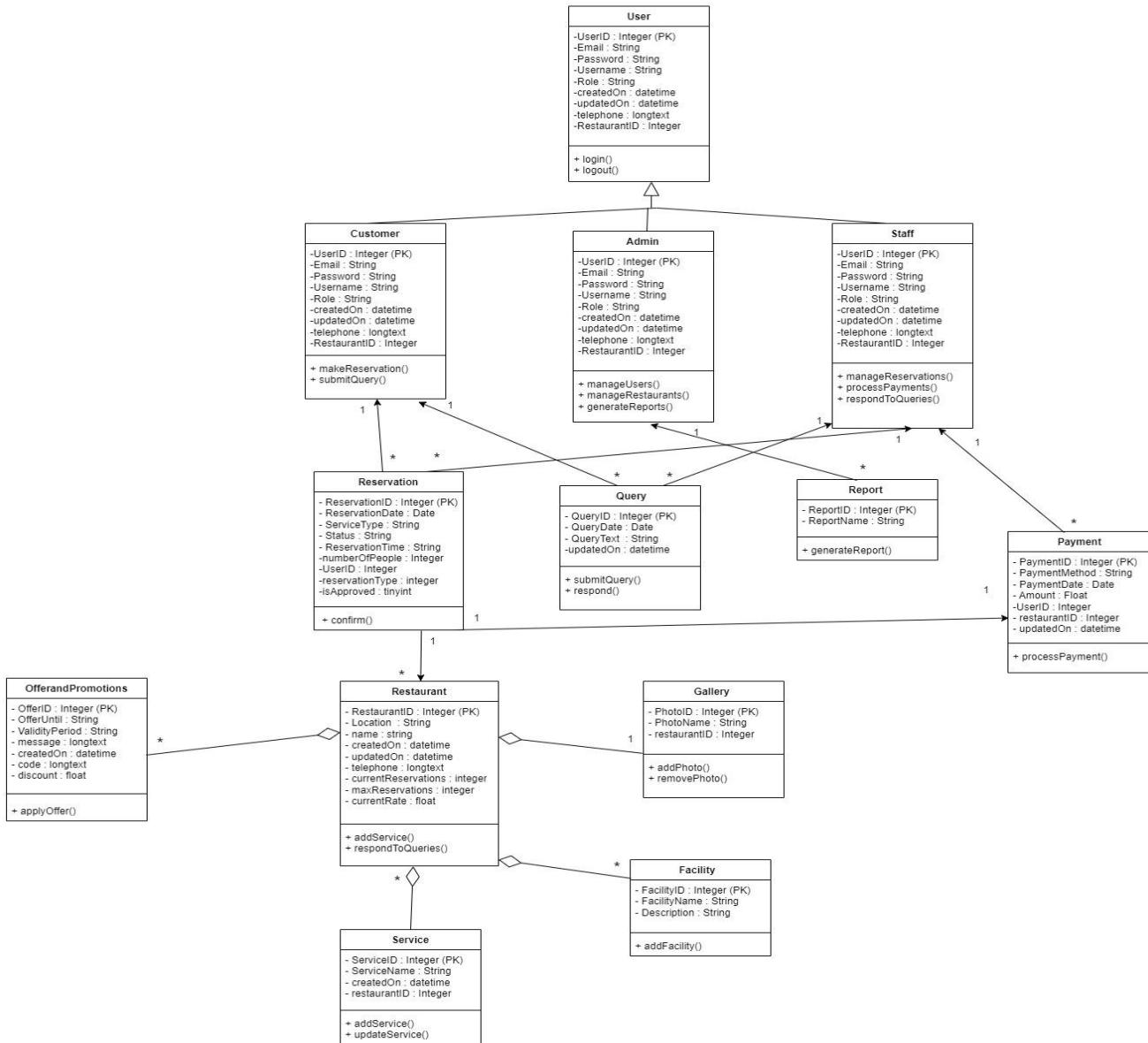
All interactions are functioning in the system according to each role. Admin possess the core management of the system having been able manage users, reservations, restaurants, offers, payments, queries and gallery. Admin is able to generate reports as well.

Staff can manage reservations, respond to customer queries, process payments and generate some of the reports as well. Staff has the responsibility to make sure that the emails are sent to the customers when they confirm a reservation and their queries are answered.



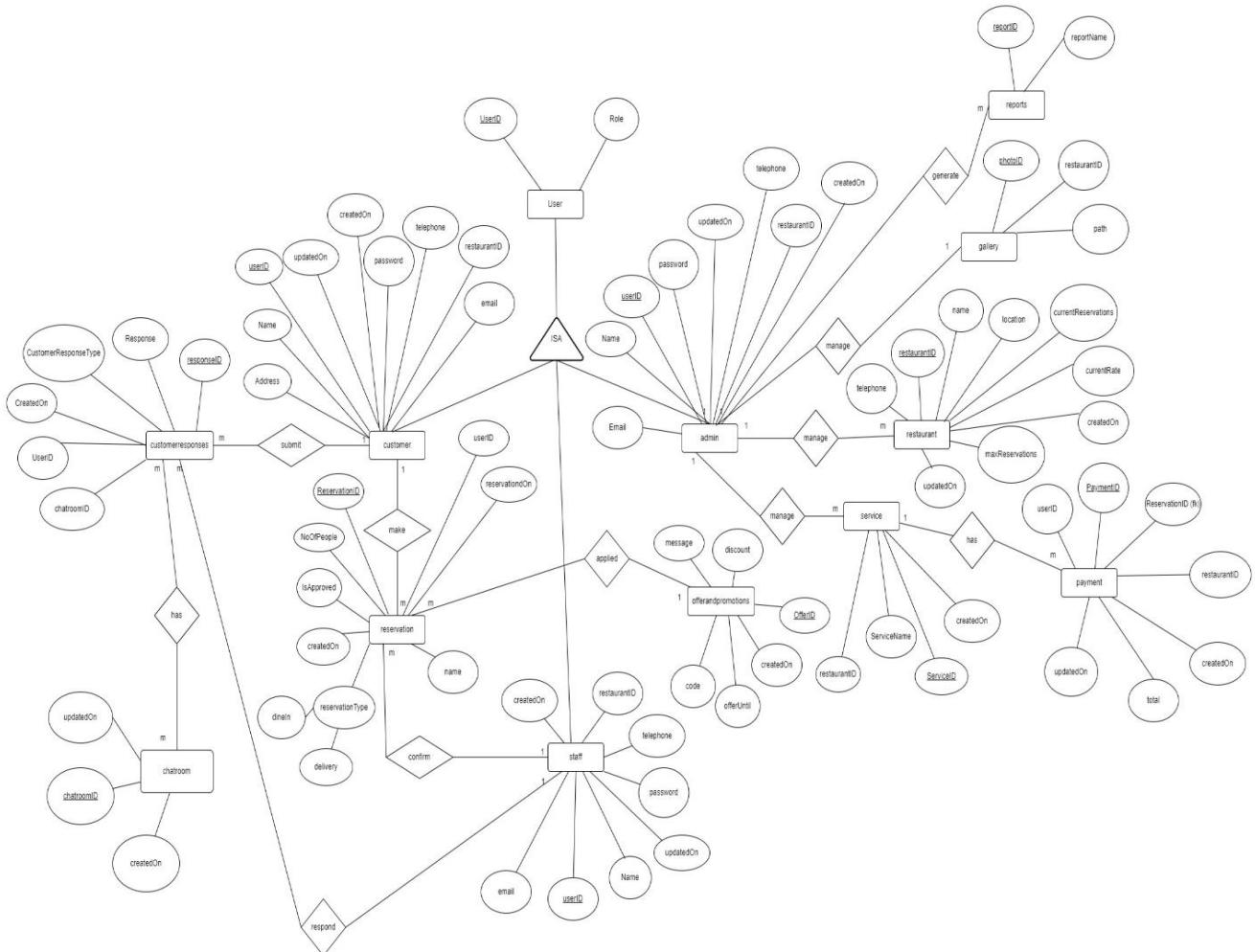
Class Diagram

User class acts as the parent class for the other three types of users. These user types are illustrated in the above diagram using inheritance. The relationships connected to restaurant through gallery, facility, service and offerandpromotions are depicted using aggregation and directed association has been used to depict all the other relationships in this diagram. Reservation class takes details about reservations such as noOfPeople, reservationDate etc. while query class is handling customer queries. Payments class manages payments and the report class is handling the report generation. Restaurant class has details about restaurants in the chain while gallery, offerandpromotion, service and facility classes are handling the additional features.



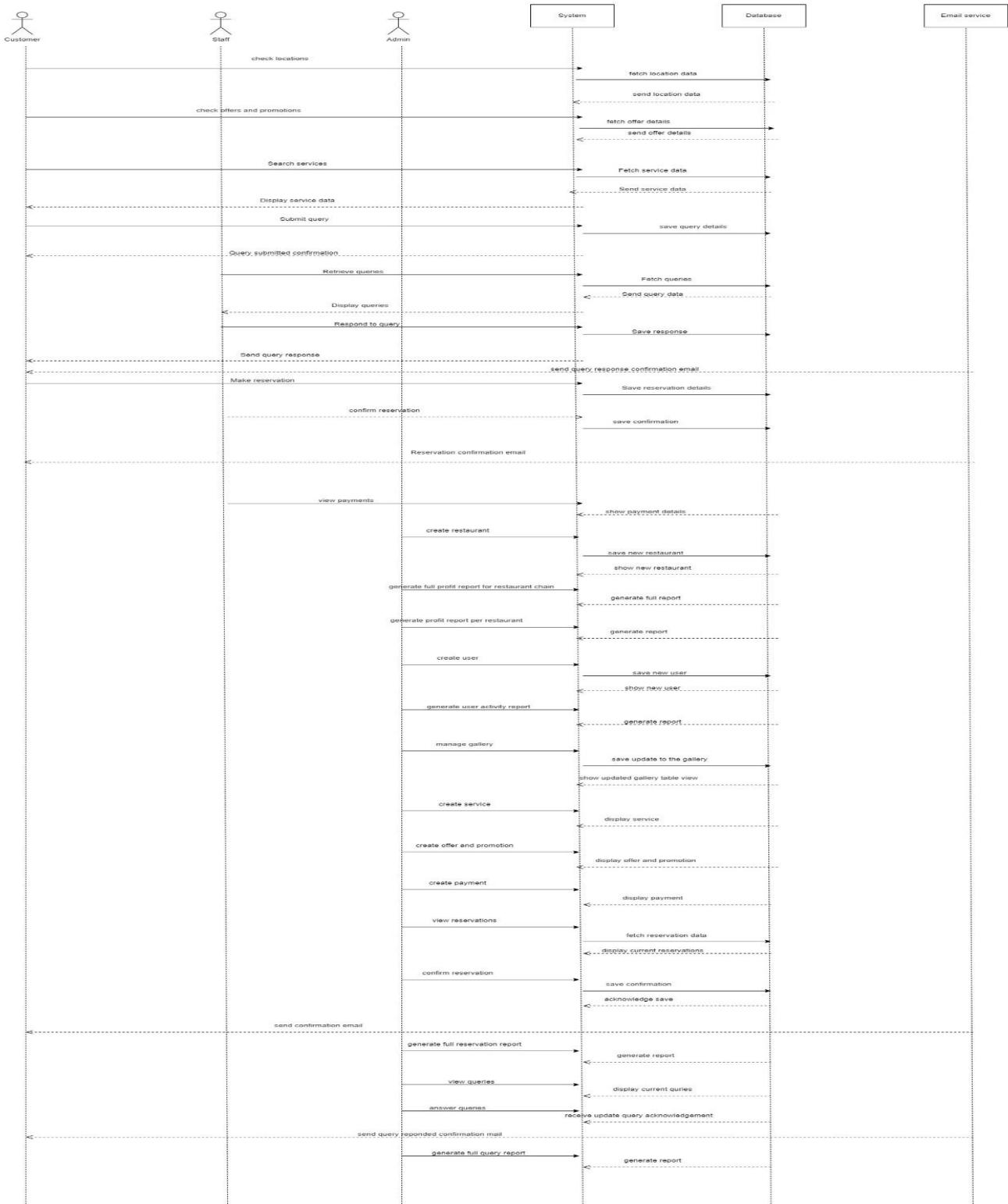
ER Diagram

All the three users are illustrated using a ISA relationship. User is a generalized entity which has inheritance as well where admin, staff and customer are specializations. The customer can make many reservations and the system includes the reservationtype entity that includes dine-in or delivery options. The reservation is connected with staff where staff can control payments and customerresponses. Admin manages the restaurant entity which is associated with offers, services and offerandpromotions etc. Admin manages the gallery. The system comes with chartroom facility for the customers to submit queries where staff respond to them.



Sequence Diagram

Sequence diagram depicts the flow of the system highlighting the interactions between three types of users with the system, database and the email system. Key interactions such as reservation management, query management, location management, user management, service management etc. are illustrated here. The sequence of requests and their responses are depicted with the information flow of the system.



Assumptions

- Admin and staff can create reservations for customers.
- Admin can enter rate per person for each restaurant
- Admin can enter a maximum number of reservations per restaurant. When the amount is exceeded, customers no longer can place reservations for the particular restaurant. Once the admin sets the max no. of reservations, the make reservation button gets enabled.
- Database has a separate table for reservations so that we can make a many-to-many relationship with restaurants.
- The chatrooms table was created so that we can take every query and make them a single question.
- Only the customers can register themselves. Staff and other admin users are created by admin.

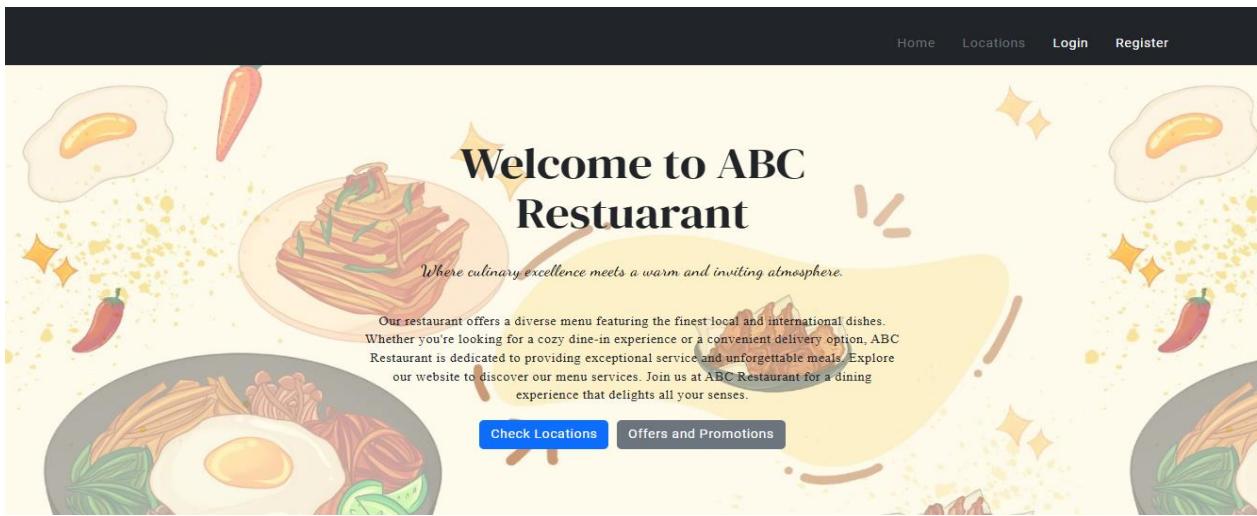
TASK B

Development of the System

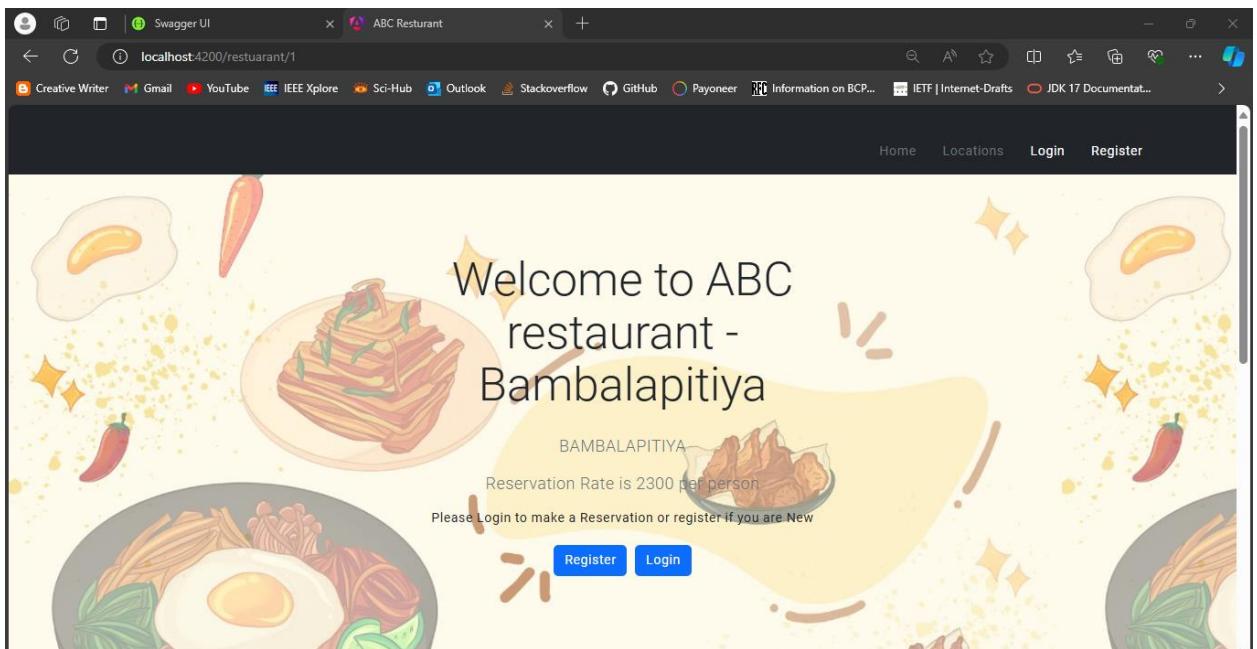
In the development process, C# and .NET 8 was used for the back-end while Angular 18 was used for the front-end. QuestPDF was the package that was used in the back-end to help with report generation and Bootstrap, Angular material and PrimeNG are the packages that were used in the front-end. Papercut SMTP desktop email server was used to send emails. Visual Studio code was used for the server-side development. Visual Studio Code was used for client-side development (UI). MySQL Workbench was used to implement the database and for query testing.

Set of Interfaces with Their Inputs and Outputs

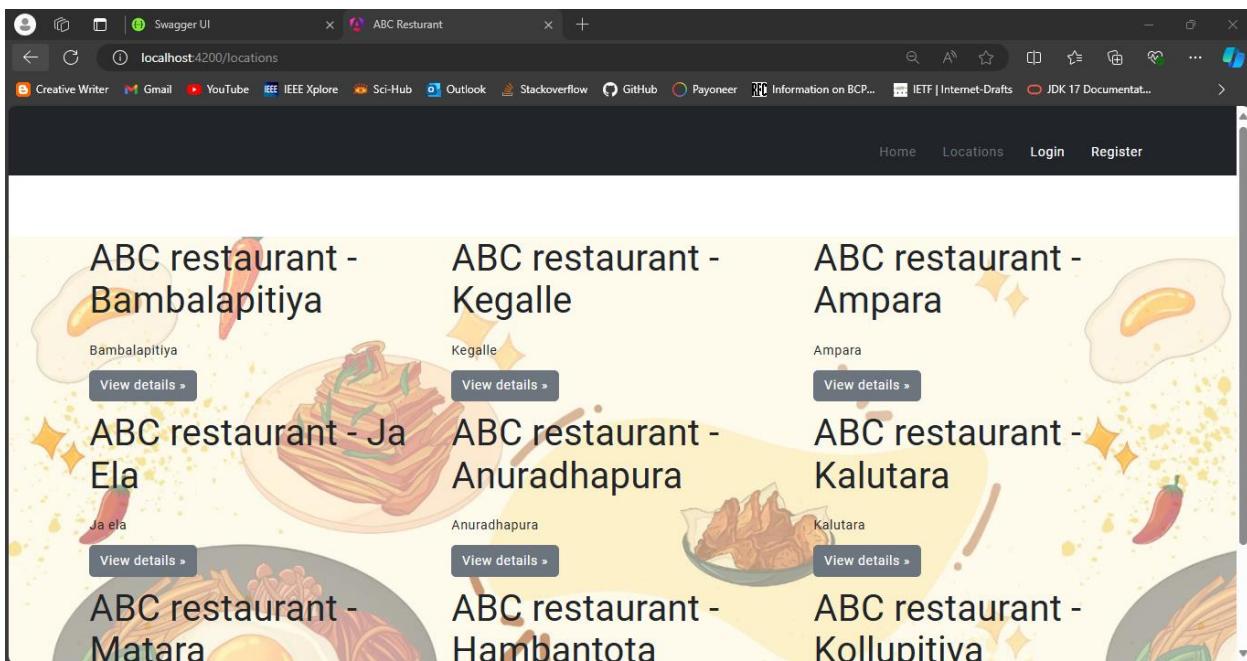
Home page



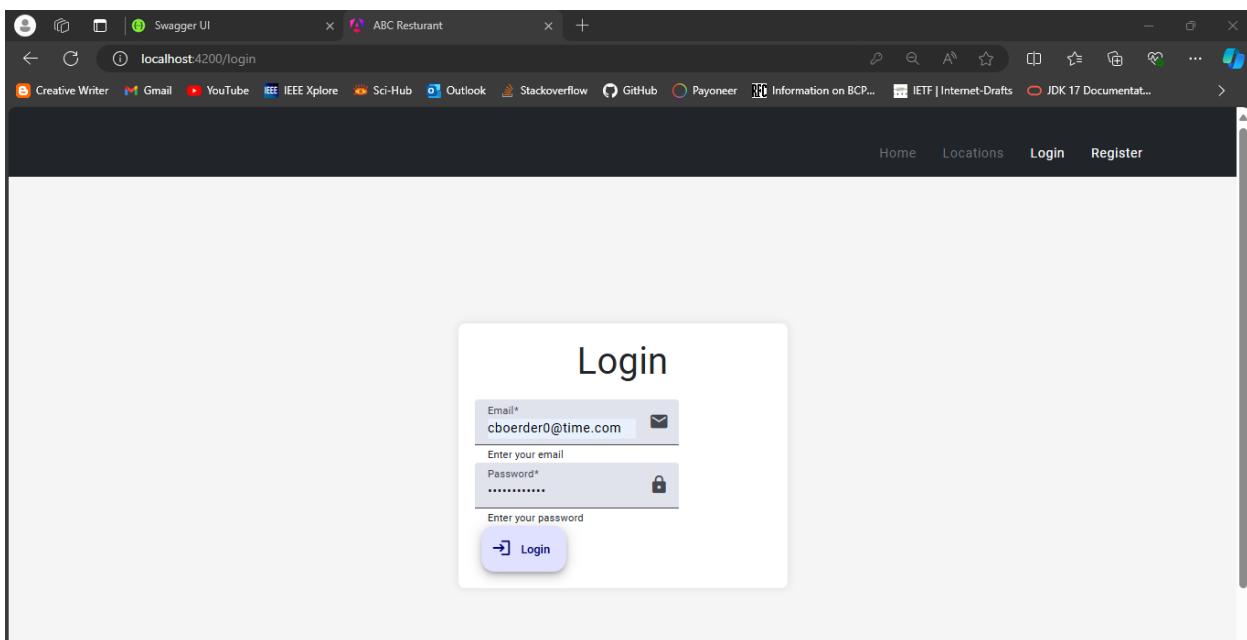
Any user can view Offers and promotions page



All users can view locations



Login interface for all users



All users can logout by clicking on the profile picture

The screenshot shows a web browser window with the title 'ABC Restaurant' at the top. In the top right corner, there is a user profile for 'Damara Hellin' with a dropdown menu containing 'Sign out'. A red box highlights this 'Sign out' button. Below the header, there is a navigation bar with links: Home, Locations, Payments, Reservations, and Queries. The main content area features a table with four rows of data. The table has columns for 'CreatedOn', 'UpdatedOn', and 'Actions'. The 'Actions' column contains three icons: a green pencil, a red trash bin, and a blue PDF icon. The data in the table is as follows:

CreatedOn	UpdatedOn	Actions
2024-08-11T14:30:08.427501	0001-01-01T00:00:00	
2024-08-11T19:43:25.953291	0001-01-01T00:00:00	
2024-08-11T19:44:03.578623	0001-01-01T00:00:00	
2024-08-11T19:44:24.135569	0001-01-01T00:00:00	

Interfaces for admin

Home page for admin

The screenshot shows the 'Overview' page for the ABC Restaurant application. At the top, there is a navigation bar with links: Home, Locations, Restaurants, Users, Gallery, Services, Offers & Promotions, Payments, Reservations, and Queries. In the top right corner, there is a user profile for 'Cleon Boerder' with a dropdown menu. A red box highlights this user profile. The main content area features a large, colorful illustration of various dishes like fried eggs, pasta, and meat. Overlaid on the illustration is a welcome message: 'Welcome to ABC Restuarant' and 'Where culinary excellence meets a warm and inviting atmosphere.' Below this, there is a paragraph of text: 'Our restaurant offers a diverse menu featuring the finest local and international dishes. Whether you're looking for a cozy dine-in experience or a convenient delivery option, ABC Restaurant is dedicated to providing exceptional service and unforgettable meals. Explore our website to discover our menu services. Join us at ABC Restaurant for a dining experience that delights all your senses.' At the bottom of the main content area, there are two buttons: 'Check Locations' and 'Offers and Promotions'.

User management

The screenshot shows a web browser window titled "ABC Restaurant" at the URL "localhost:4200/users". The page has a top navigation bar with links for Home, Locations, Restaurants, Users, Gallery, Services, Offers & Promotions, Payments, Reservations, and Queries. A user profile for "Cleon Boerder" is visible on the right. Below the navigation is a search bar with the placeholder "Search keyword". A table lists five users:

Name	Role	Telephone	CreatedOn	UpdatedOn	Actions			
Cleon Boerder	Admin	383-8490799	2024-01-13T00:00:00					
Damara Hellin	Staff	8813849155	2023-11-27T00:00:00	2024-08-12T13:21:58.336704				
Donavon Christophersen	Staff	2169890578	2024-05-11T00:00:00	2024-08-12T13:19:22.192093				
Warner Masse	Admin	3885599160	2023-11-22T00:00:00	2024-08-12T13:18:33.015546				
Elwira McQuirk	Admin	134-9992754	2023-08-27T00:00:00					

Create user

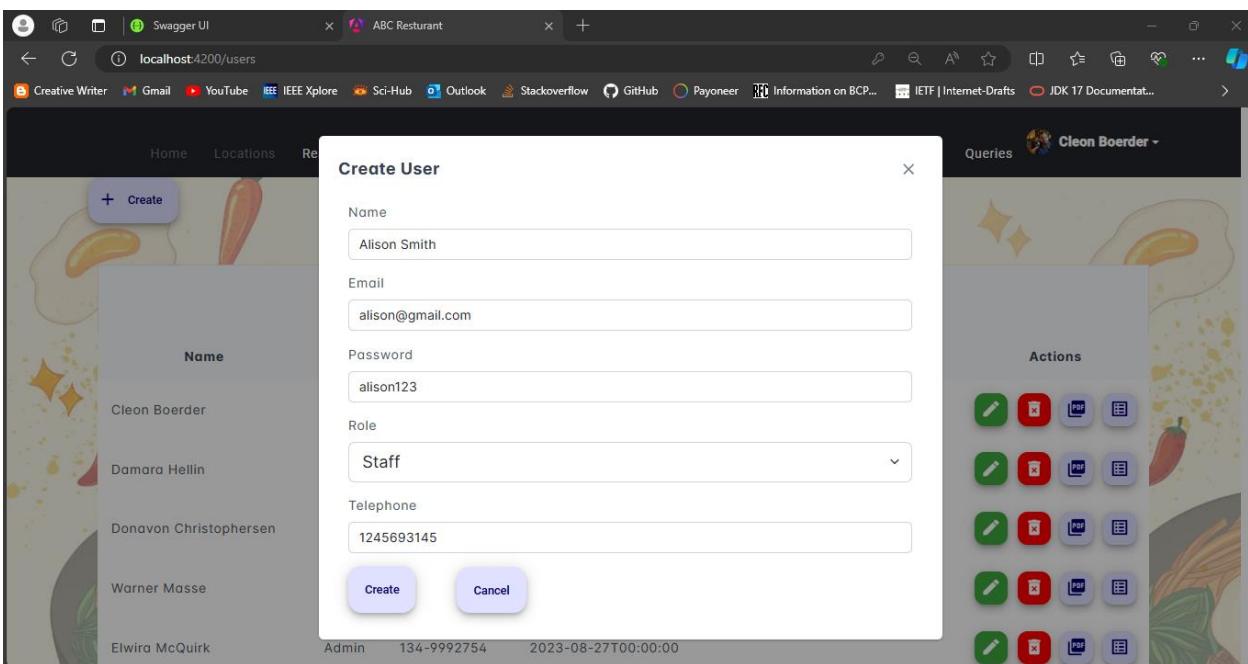
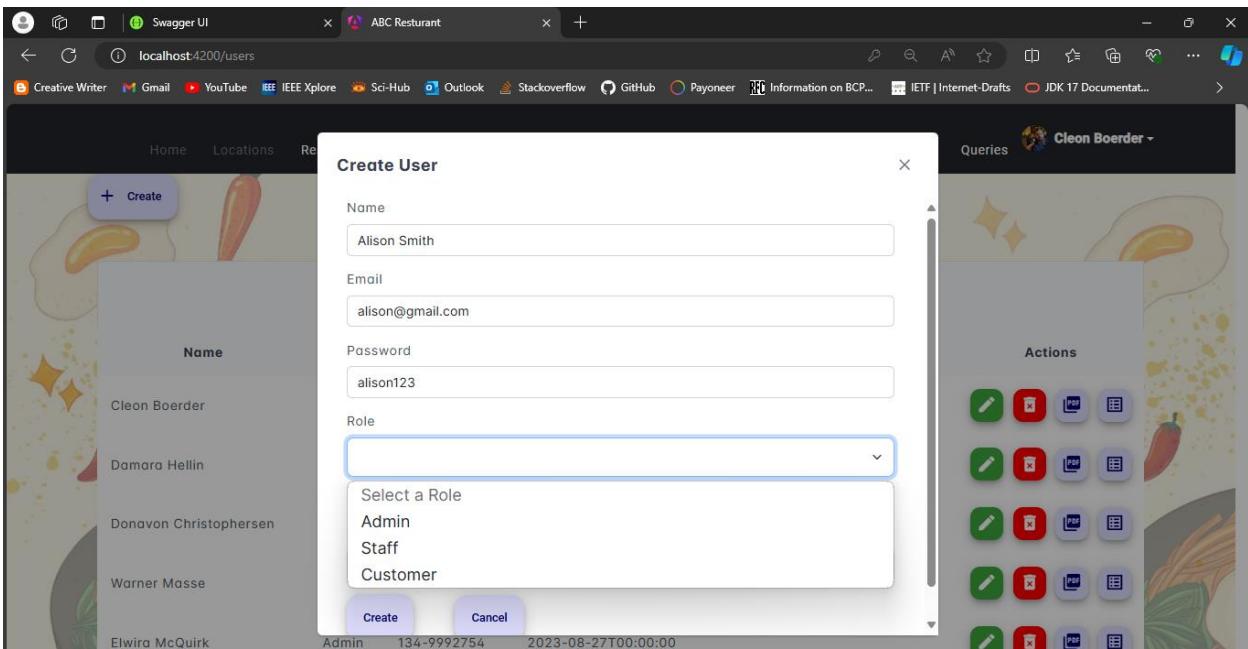
Form validations

The screenshot shows the "Create User" dialog box from the ABC Restaurant application. The dialog contains fields for Name, Email, Password, Role, and Telephone. Validation messages are displayed below each field:

- Name: "Name is required and must be at least 5 characters long."
- Email: "Email is required and must in correct format."
- Password: "Password is required and must be at least 8 characters long."
- Role: "Role is required and must be selected."
- Telephone: "Telephone is required and must be a valid 10-digit number."

Below the dialog, a partial view of the user list is visible, showing names like Cleon Boerder, Damara Hellin, Donavon Christophersen, Warner Masse, and Gal Lockyear.

Select role from drop down



Upon clicking on create button the new user is created and a notification is received to confirm it.

A screenshot of a web browser displaying the ABC Restaurant application. The page shows a list of users with columns for Name, Role, Telephone, CreatedOn, UpdatedOn, and Actions. A modal window at the top right displays a blue confirmation message: "Confirmed" and "New User Created". The user list includes Cleon Boerder, Damara Hellin, Donavon Christophersen, Warner Masse, and Elwira McQuirk. The "Actions" column for each user contains five icons: a green pencil, a red delete, a blue info, a purple print, and a blue document.

Name	Role	Telephone	CreatedOn	UpdatedOn	Actions
Cleon Boerder	Admin	383-8490799	2024-01-13T00:00:00		
Damara Hellin	Staff	8813849155	2023-11-27T00:00:00	2024-08-12T13:21:58.336704	
Donavon Christophersen	Staff	2169890578	2024-05-11T00:00:00	2024-08-12T13:19:22.192093	
Warner Masse	Admin	3885599160	2023-11-22T00:00:00	2024-08-12T13:18:33.015546	
Elwira McQuirk	Admin	134-9992754	2023-08-27T00:00:00		

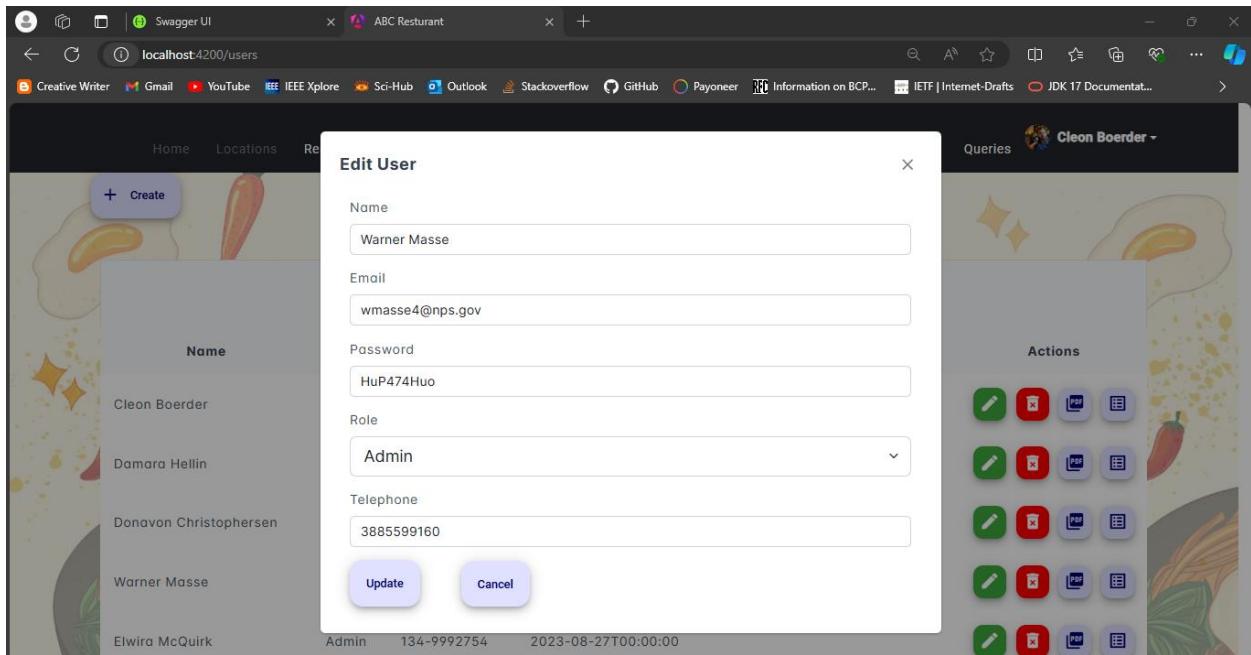
New user is created.

A screenshot of the ABC Restaurant application. The user list now includes a new entry: Alison Smith, Staff, telephone 1245693145, created on 2024-08-26T16:11:43.598727. This row is highlighted with a red rectangular box. The rest of the interface remains the same, with the "Actions" column showing the standard five icons.

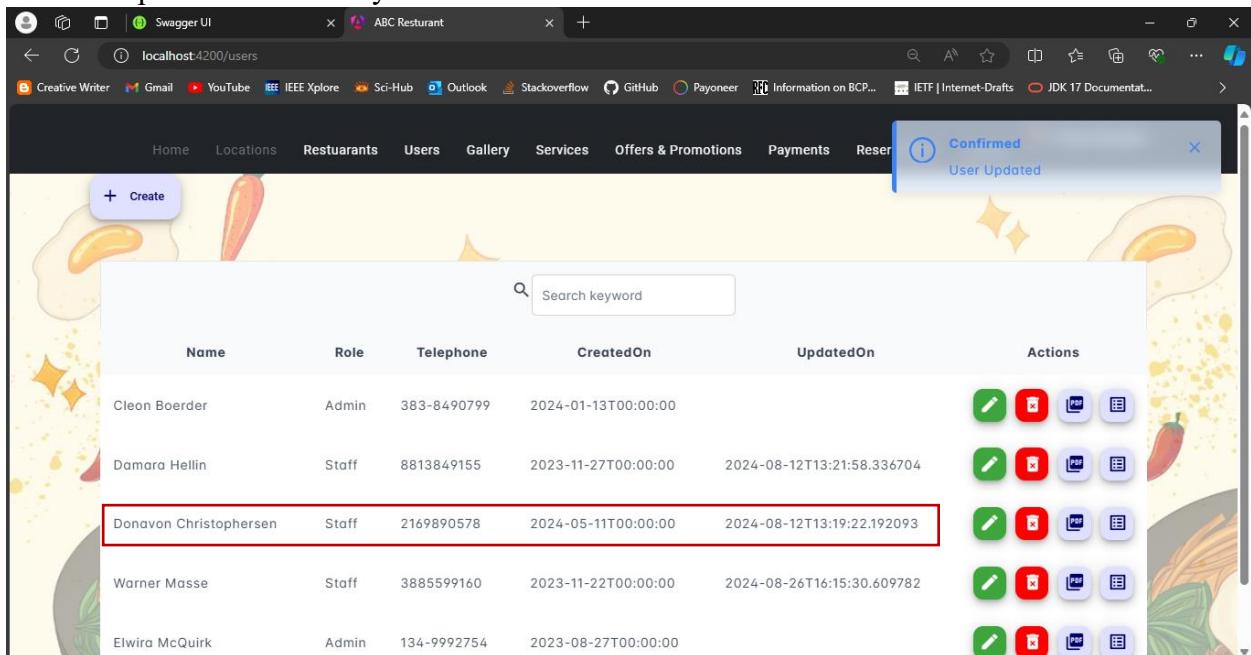
Name	Role	Telephone	CreatedOn	UpdatedOn	Actions
Amanda Seyfried Taylor	Staff	5231456984	2024-08-12T17:25:04.39983	2024-08-12T17:26:26.880789	
Alison Smith	Staff	1245693145	2024-08-26T16:11:43.598727		

Edit user

Admin can edit all user accounts

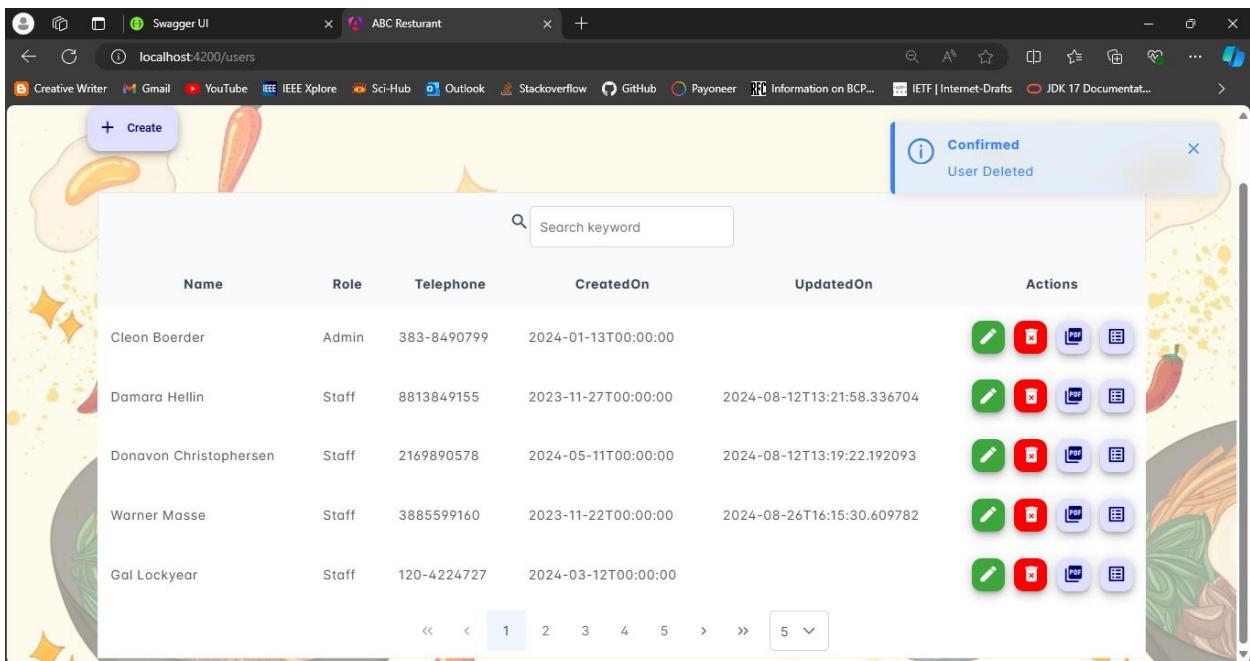
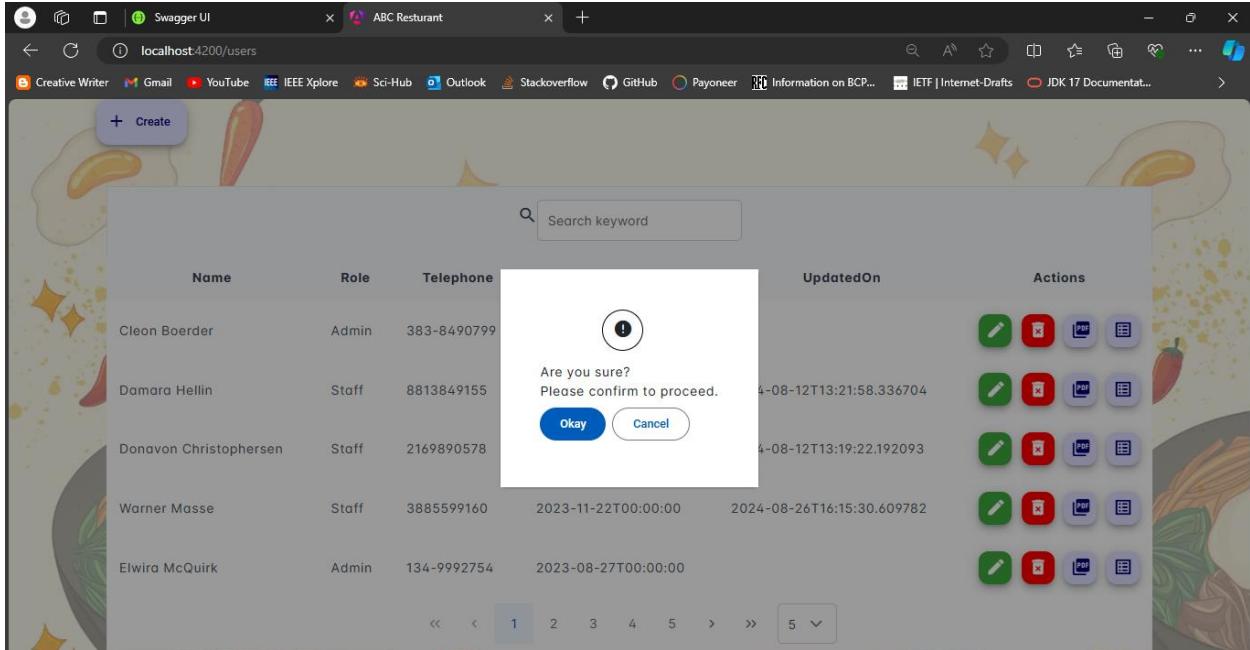


Account updated successfully



Delete user

To delete a user the delete button should be clicked under actions. And then the following pop up appears and the admin should select okay.



View user

The screenshot shows a modal window titled "Cleon Boerder" displaying user details. The modal includes fields for Name (Cleon Boerder), Email (cboerder0@time.com), Password (zcFUUrKv41iq4), Role (Admin), Telephone (383-8490799), and Created On (Saturday, January 13, 2024). Below the modal, the staff ID is listed as 120-4224727 and the updated date as 2024-03-12T00:00:00. The background shows a list of names: Cleon Boerder, Damara Hellin, Donavon Christophersen, Warner Masse, and Gal Lockyear. The top navigation bar has tabs for Home, Locations, and Restaurants.

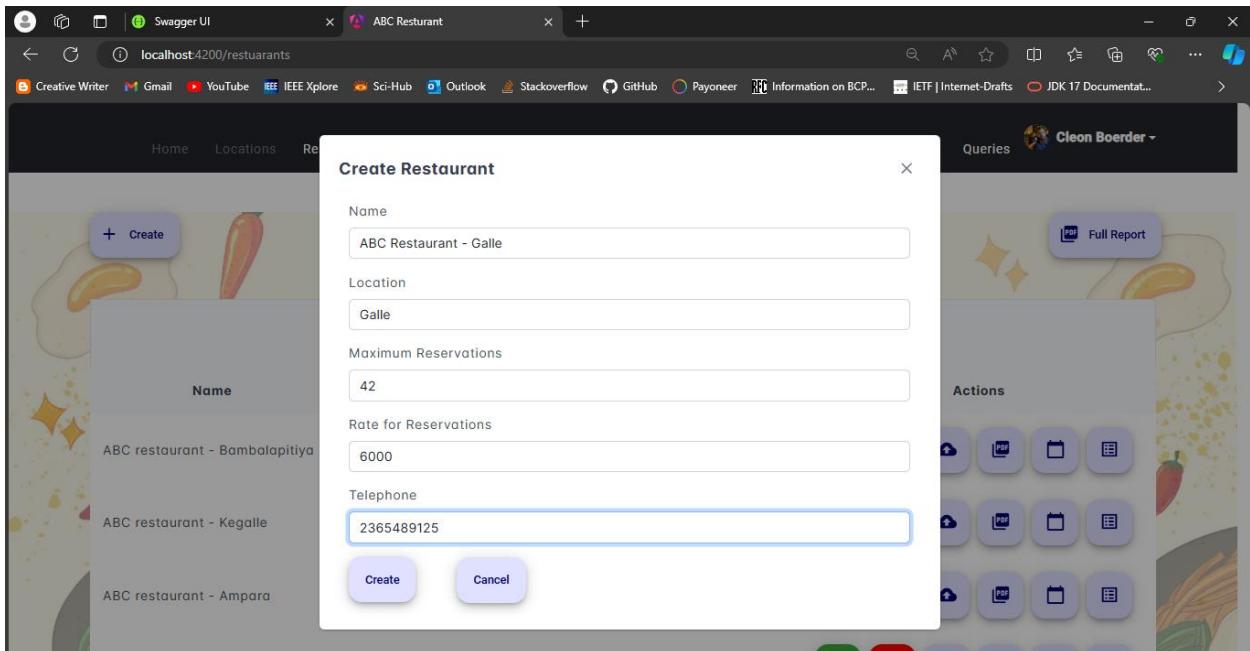
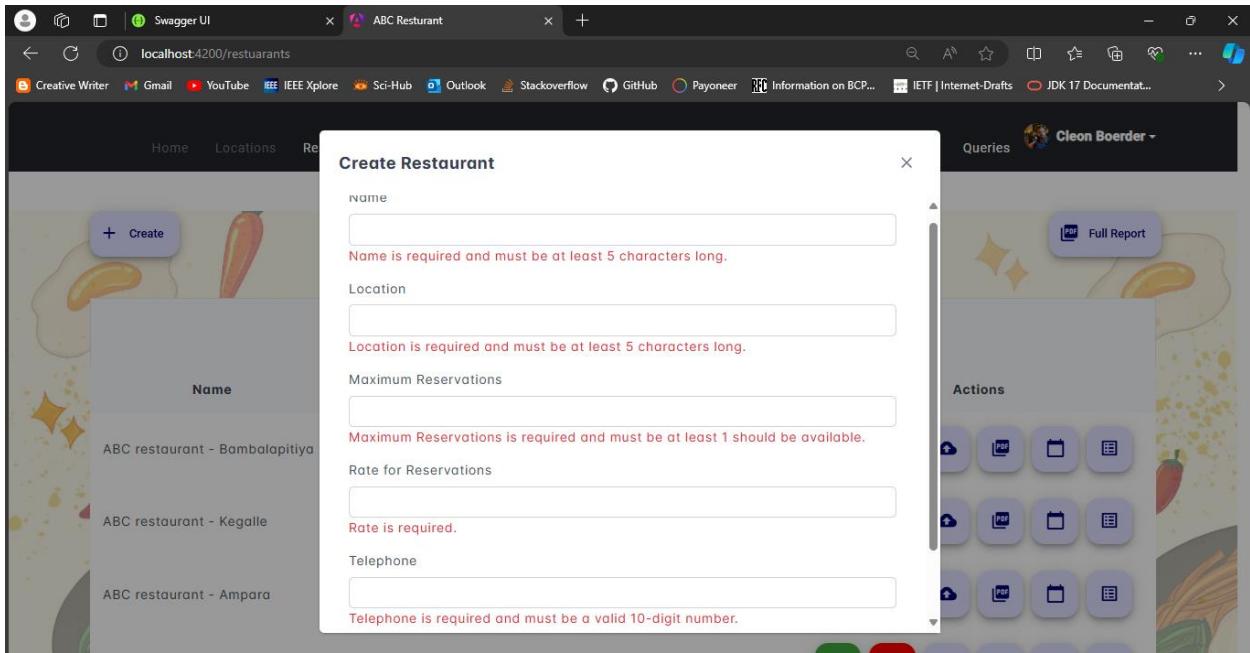
Restaurant management

The screenshot shows a list of restaurants. The table has columns for Name, Location, Telephone, and CreatedOn. The first three rows are: ABC restaurant - Bambalapitiya (Location: Bambalapitiya, Telephone: 4563124588, CreatedOn: 2024-08-11T19:02:02.022746), ABC restaurant - Kegalle (Location: Kegalle, Telephone: 4513697845, CreatedOn: 2024-08-11T19:02:34.318664), and ABC restaurant - Ampara (Location: Ampara, Telephone: 1254863594, CreatedOn: 2024-08-11T19:03:01.801478). A search bar at the top right says "Search keyword". A "Full Report" button is visible in the top right corner. The top navigation bar includes tabs for Home, Locations, Restaurants, Users, Gallery, Services, Offers & Promotions, Payments, Reservations, and Queries. The user profile "Cleon Boerder" is also visible in the top right.

Name	Location	Telephone	CreatedOn	Actions
ABC restaurant - Bambalapitiya	Bambalapitiya	4563124588	2024-08-11T19:02:02.022746	
ABC restaurant - Kegalle	Kegalle	4513697845	2024-08-11T19:02:34.318664	
ABC restaurant - Ampara	Ampara	1254863594	2024-08-11T19:03:01.801478	

Create restaurant

Form validations



localhost:4200/restaurants

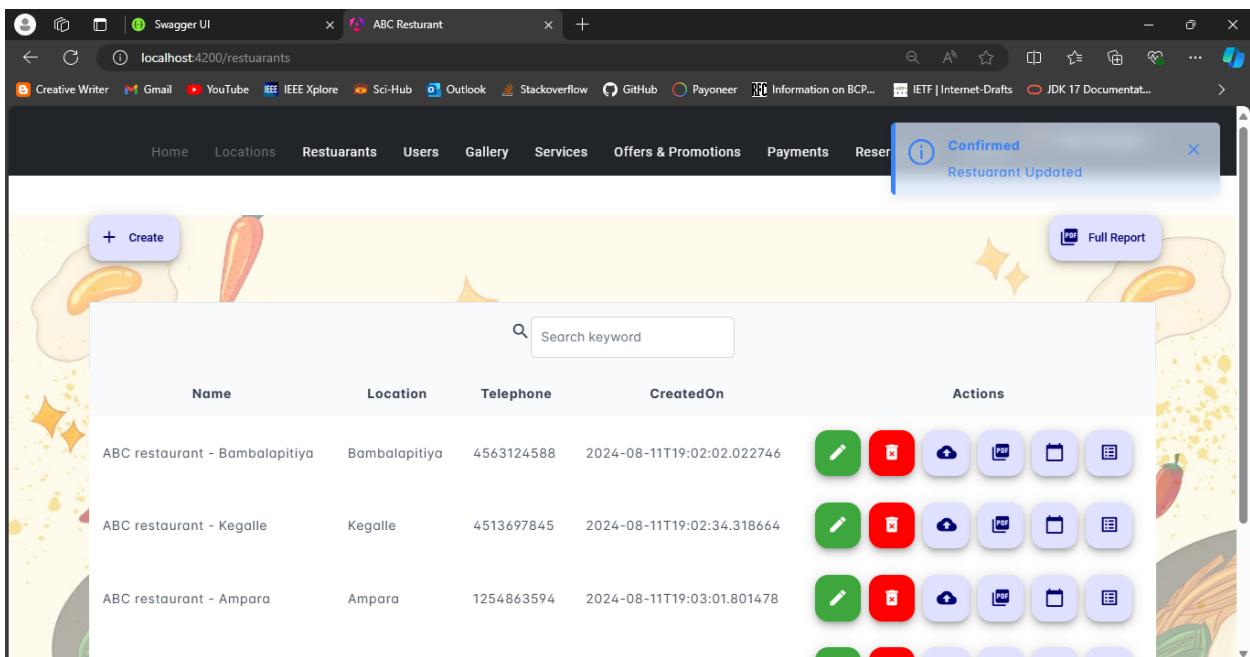
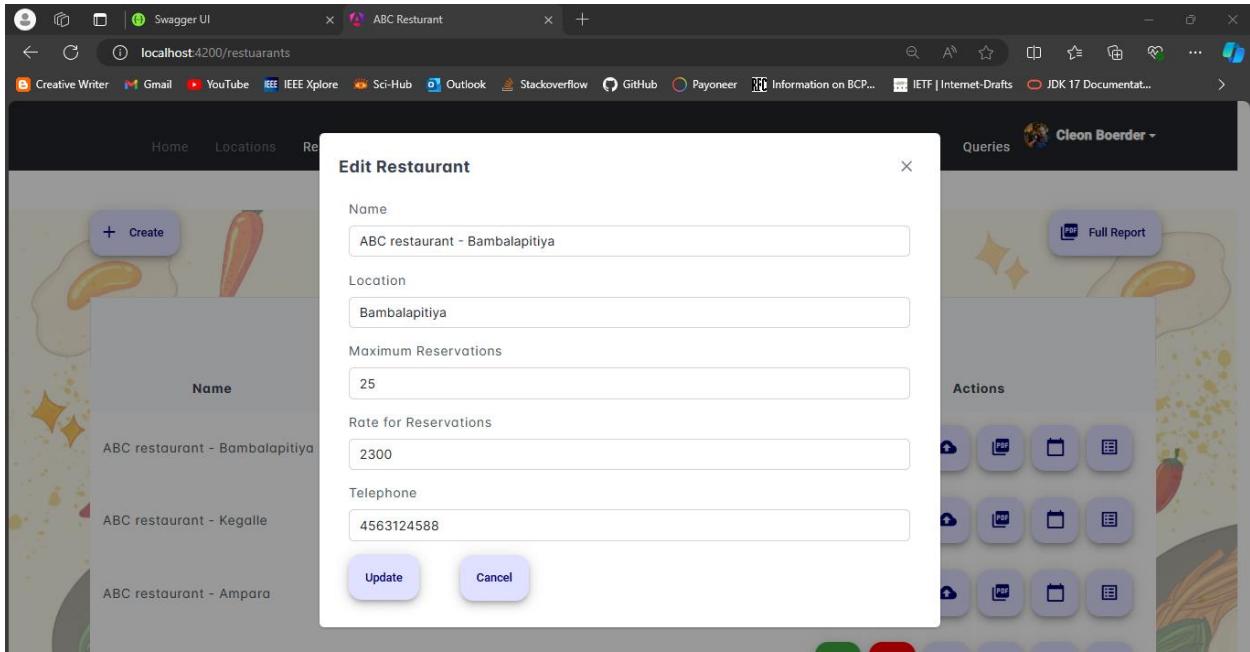
Confirmed
New Restaurant Created

Name	Location	Telephone	CreatedOn	Actions
ABC restaurant - Bambalapitiya	Bambalapitiya	4563124588	2024-08-11T19:02:02.022746	
ABC restaurant - Kegalle	Kegalle	4513697845	2024-08-11T19:02:34.318664	
ABC restaurant - Ampara	Ampara	1254863594	2024-08-11T19:03:01.801478	

localhost:4200/restaurants

Name	Location	Telephone	CreatedOn	Actions
ABC restaurant - Kalutara	Kalutara	5361247456	2024-08-11T21:11:26.534934	
ABC restaurant - Matara	Matara	5213648759	2024-08-11T21:12:31.122788	
ABC restaurant - Hambantota	Hambantota	1254963542	2024-08-11T21:13:41.033032	
ABC restaurant - Kollupitiya	Kollupitiya	5213645869	2024-08-11T21:14:36.218661	
ABC Restaurant - Galle	Galle	2365489125	2024-08-26T16:32:34.971763	

Update restaurant



Delete restaurant

The screenshot shows a list of restaurants in a table format. A modal dialog box is centered over the third row, asking for confirmation to proceed. The dialog contains a warning icon, the text "Are you sure? Please confirm to proceed.", and two buttons: "Okay" and "Cancel".

Name	Location	Telephone	CreatedOn	Actions
ABC restaurant - Bambalapitiya	Bambalapitiya	4563124588	2024-08-11T19:02:02.022746	
ABC restaurant - Kegalle	Kegalle	4513697845	2024-08-11T19:02:34.318664	
ABC restaurant - Ampara	Ampara	1254863594	2024-08-11T19:03:01.801478	
ABC restaurant - Ja Ela	Ja el a	4512789654	2024-08-11T19:03:50.825746	
ABC restaurant - Anuradhapura	Anuradhapura	5784213654	2024-08-11T19:04:27.279813	

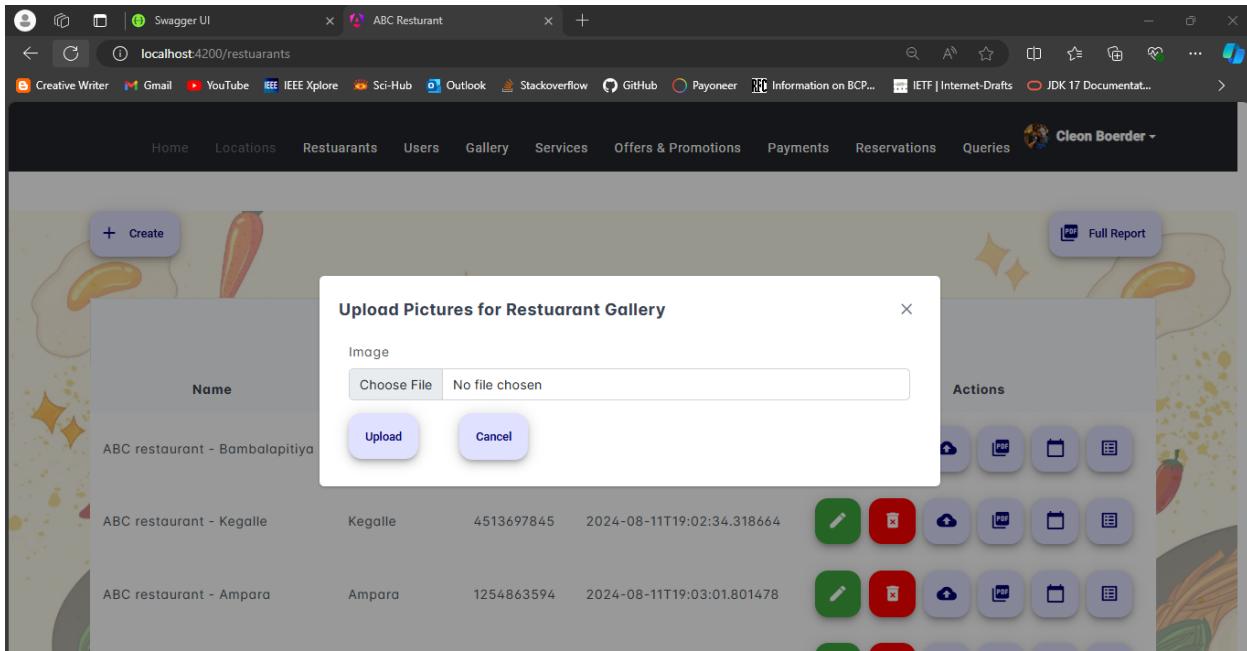
The screenshot shows the same list of restaurants as the previous one, but the modal dialog has been closed. A new message box at the top right says "Confirmed" and "Restaurant Deleted".

Name	Location	Telephone	CreatedOn	Actions
ABC restaurant - Bambalapitiya	Bambalapitiya	4563124588	2024-08-11T19:02:02.022746	
ABC restaurant - Kegalle	Kegalle	4513697845	2024-08-11T19:02:34.318664	
ABC restaurant - Ampara	Ampara	1254863594	2024-08-11T19:03:01.801478	
ABC restaurant - Ja Ela	Ja el a	4512789654	2024-08-11T19:03:50.825746	
ABC restaurant - Kalutara	Kalutara	5361247456	2024-08-11T21:11:26.534934	

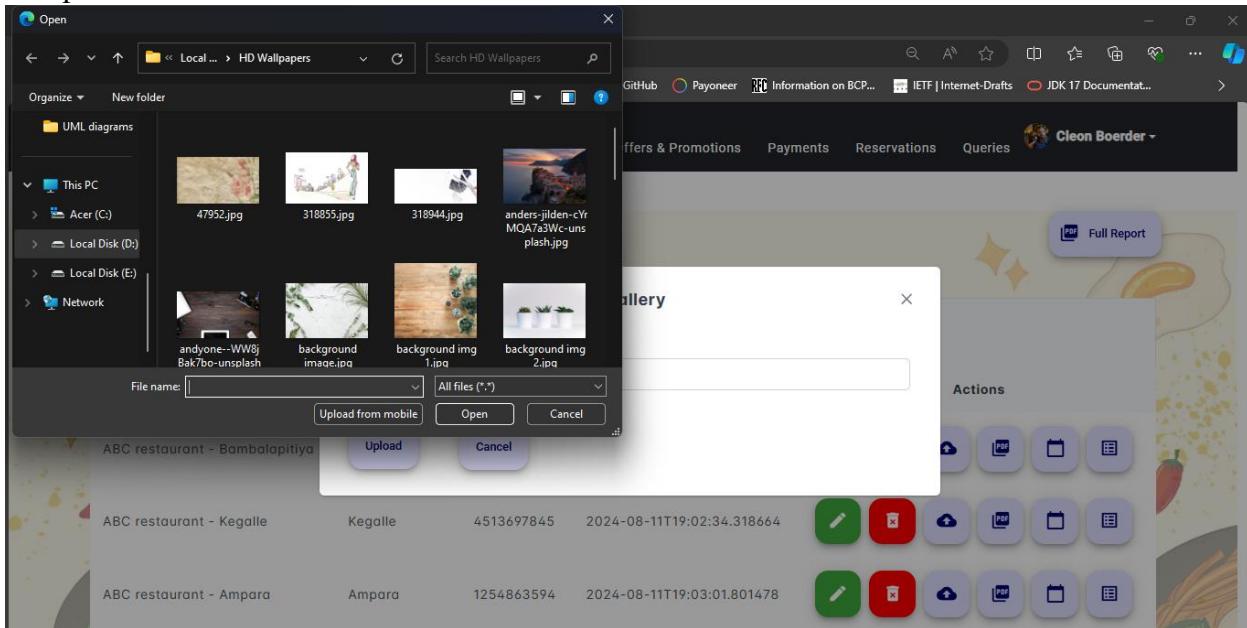
ABC restaurant – Anuradhapura is deleted.

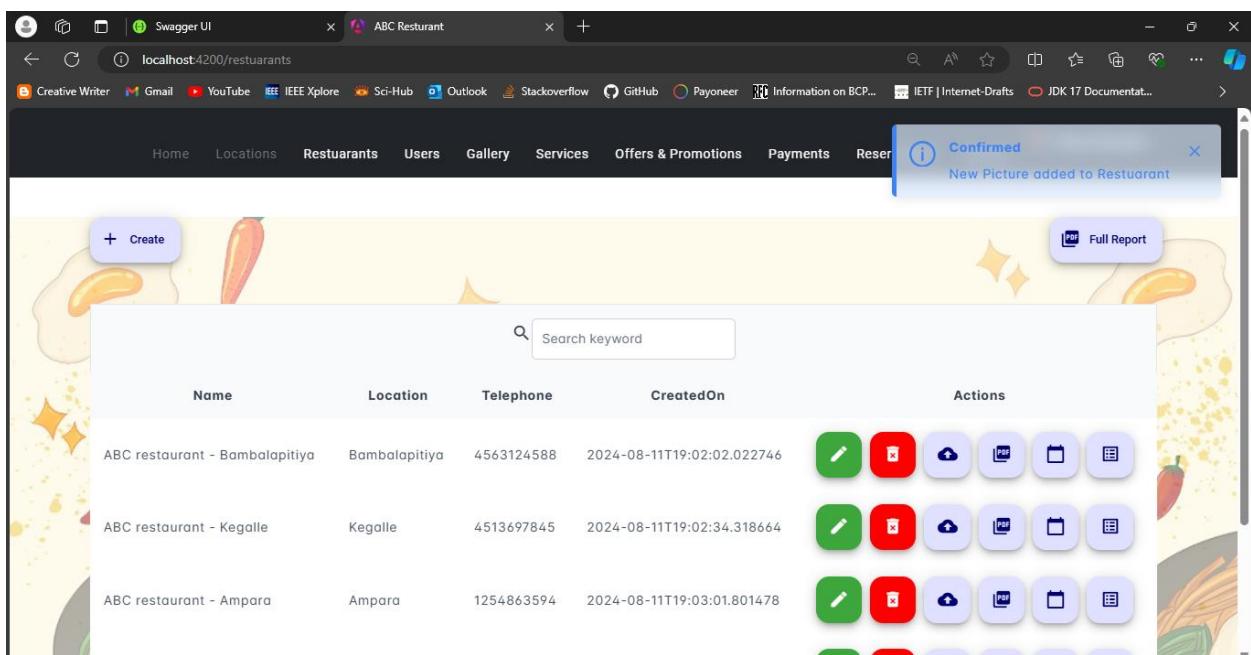
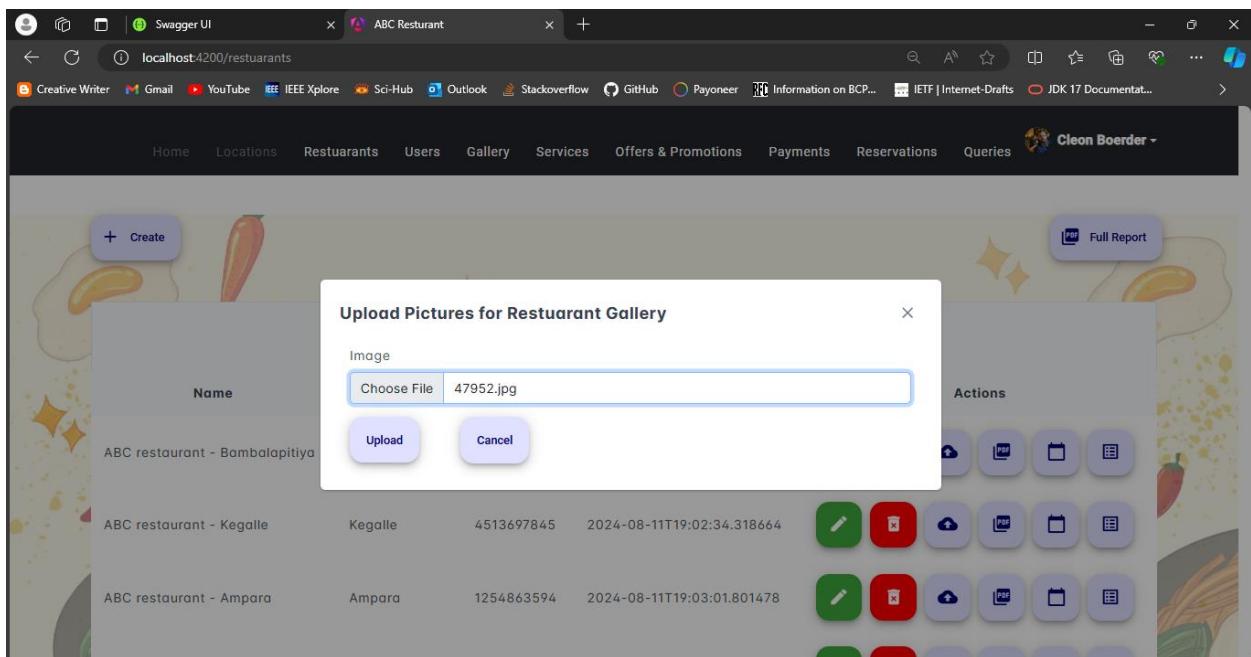
Gallery Management

Upload images to restaurant gallery



File picker





Uploaded picture in the gallery.

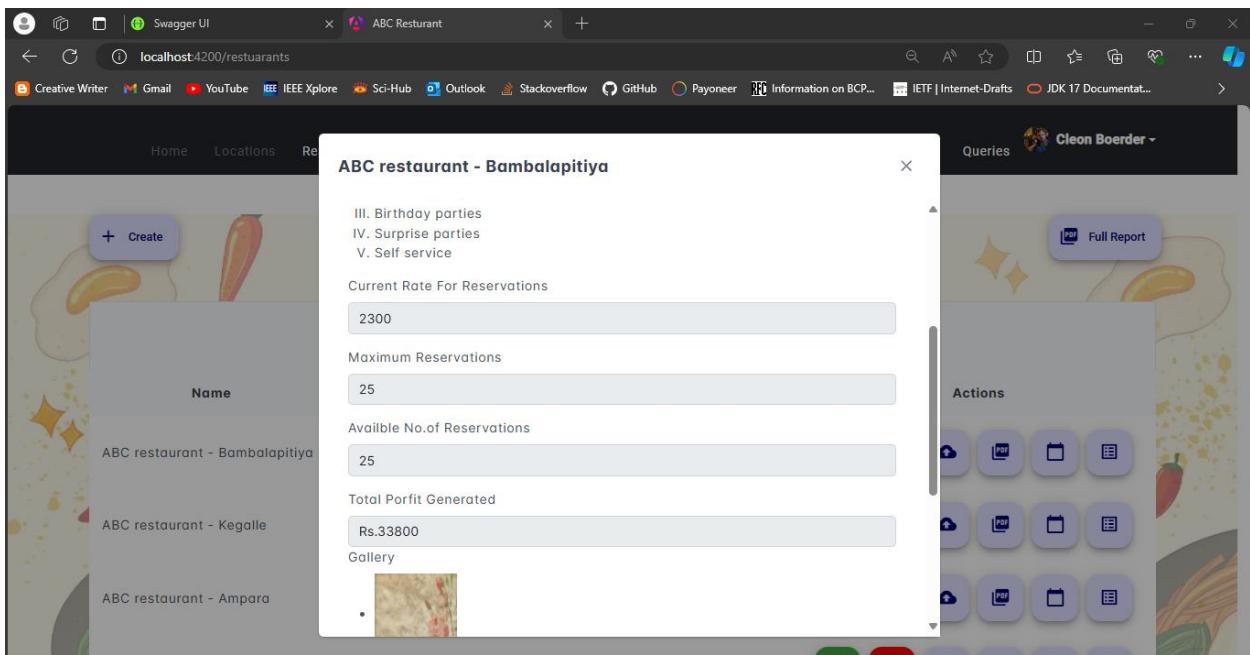
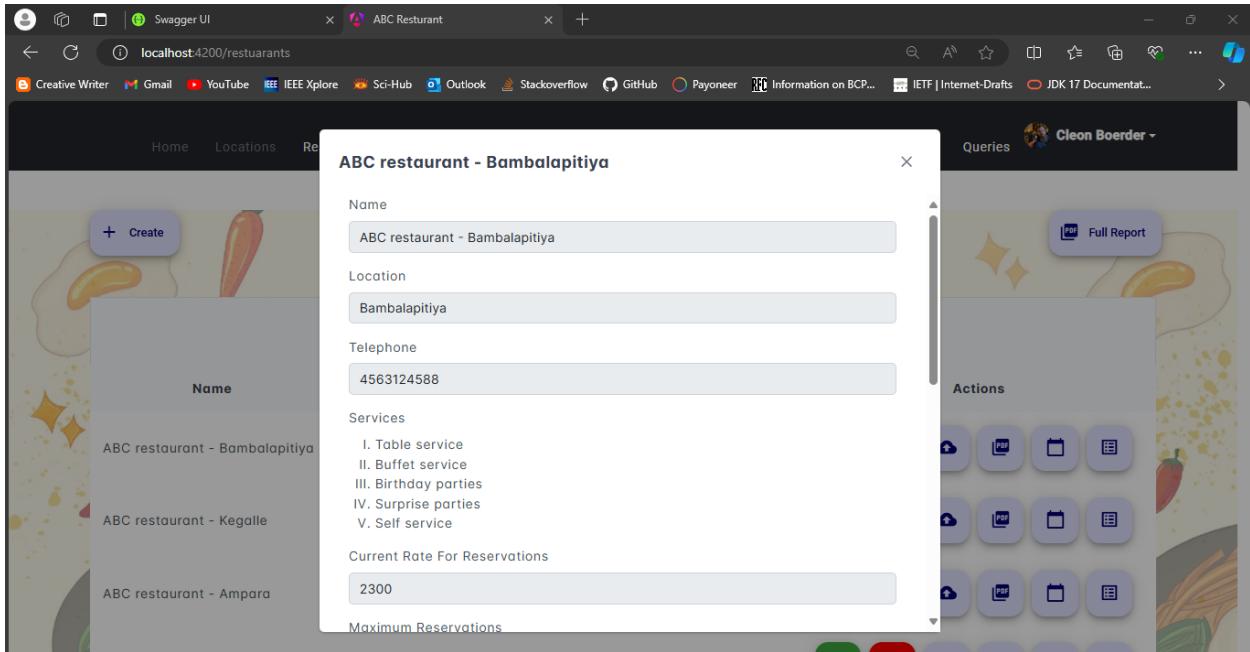
The screenshot shows a web browser window for 'ABC Restaurant' at 'localhost:4200/gallery'. The page has a header with navigation links: Home, Locations, Restaurants, Users, Gallery, Services, Offers & Promotions, Payments, Reservations, Queries, and a user profile for 'Cleon Boerder'. Below the header is a table with four columns: 'Image', 'Path', 'Restaurant Id', and 'Actions'. The first row shows an image of a landscape with a path leading to a cliff, labeled 'images\1_anders-jilden-cYrMQA7a3Wc-unsplash.jpg' and '1'. The second row shows an image of a floral arrangement, labeled 'images\1_47952.jpg' and '1'. A red rectangular box highlights the second row. At the bottom of the table is a navigation bar with arrows and a page number '5'.

And in the relevant location, it is shown also.

The screenshot shows the 'ABC Restaurant' homepage at 'localhost:4200/restaurant/1'. The main content area features a decorative background with various food illustrations like fried eggs, noodles, and chili peppers. Centered text reads 'Welcome to ABC restaurant - Bambalapitiya'. Below it is the text 'BAMBALAPITIYA' and 'Reservation Rate is 2300 per person' with a 'Make Reservation' button. A large red arrow points from the highlighted image in the previous screenshot to the same image displayed on the homepage banner.

View restaurant

View button to view details



View gallery

Under gallery tab all the uploaded pictures are shown with the restaurant ID.

Image	Path	Restaurant Id	Actions
	images\1_47952.jpg	1	
	images\2_47952.jpg	2	
	images\2_anders-jilden-cYrMQA7a3Wc-unsplash.jpg	2	
	images\4_background img 2.jpg	4	

Delete gallery items

Image	Path	Restaurant Id	Actions
	images\1_47952.jpg	1	
	images\2_47952.jpg	2	
	images\2_anders-jilden-cYrMQA7a3Wc-unsplash.jpg	2	
	images\4_background img 2.jpg	4	

The screenshot shows a web browser window for the ABC Restaurant application. The URL is `localhost:4200/gallery`. The page displays a table of images with columns: Image, Path, Restaurant Id, and Actions. A confirmation message "Confirmed Gallery Deleted" is visible in a blue toast notification at the top right. The table data is as follows:

Image	Path	Restaurant Id	Actions
	images\1_47952.jpg	1	
	images\2_47952.jpg	2	
	images\2_anders-jilden-cYrMQA7a3Wc-unsplash.jpg	2	
	images\4_background img 2.jpg	4	

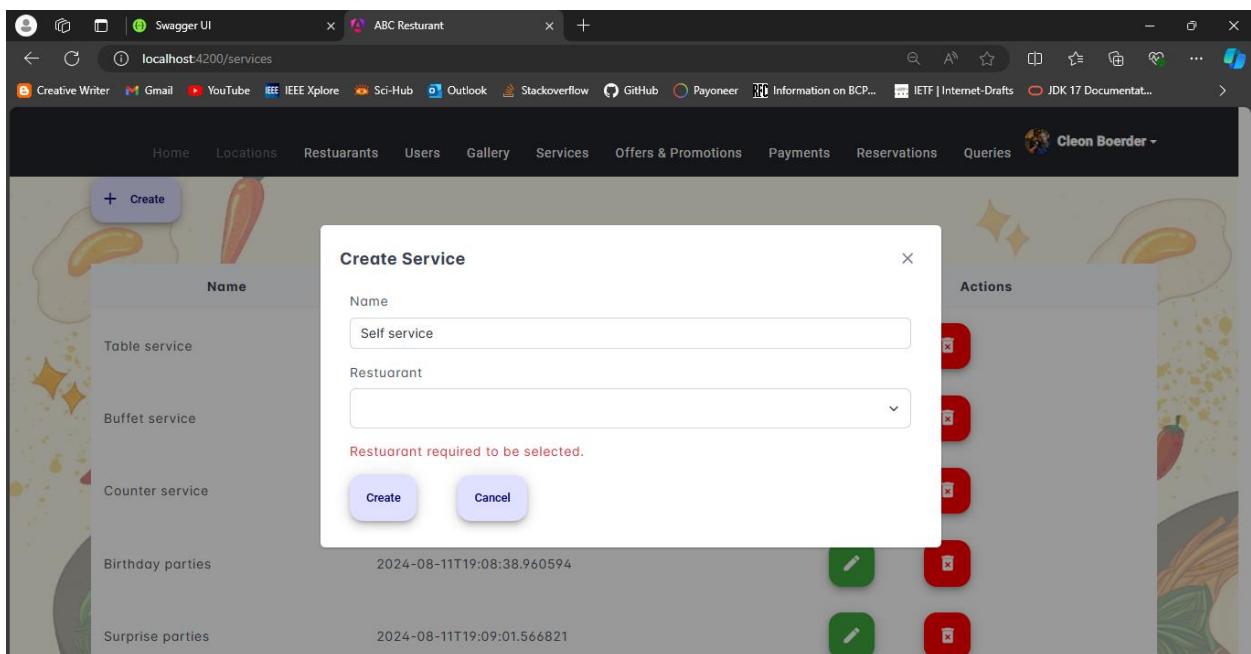
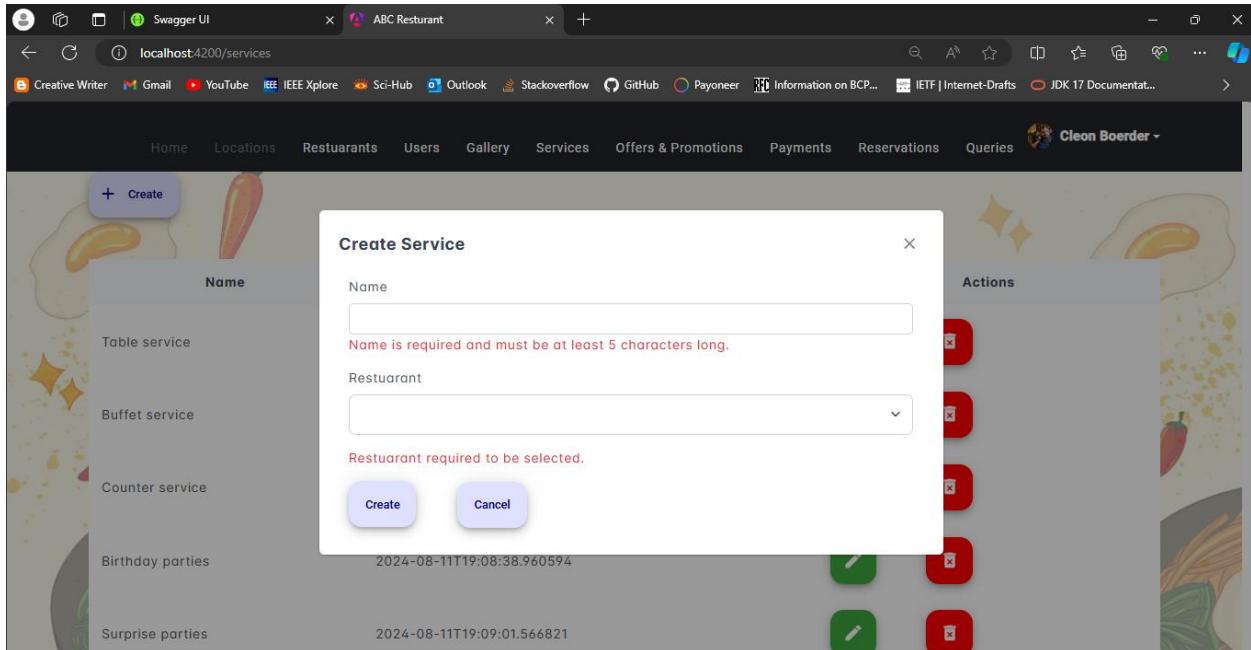
Service management

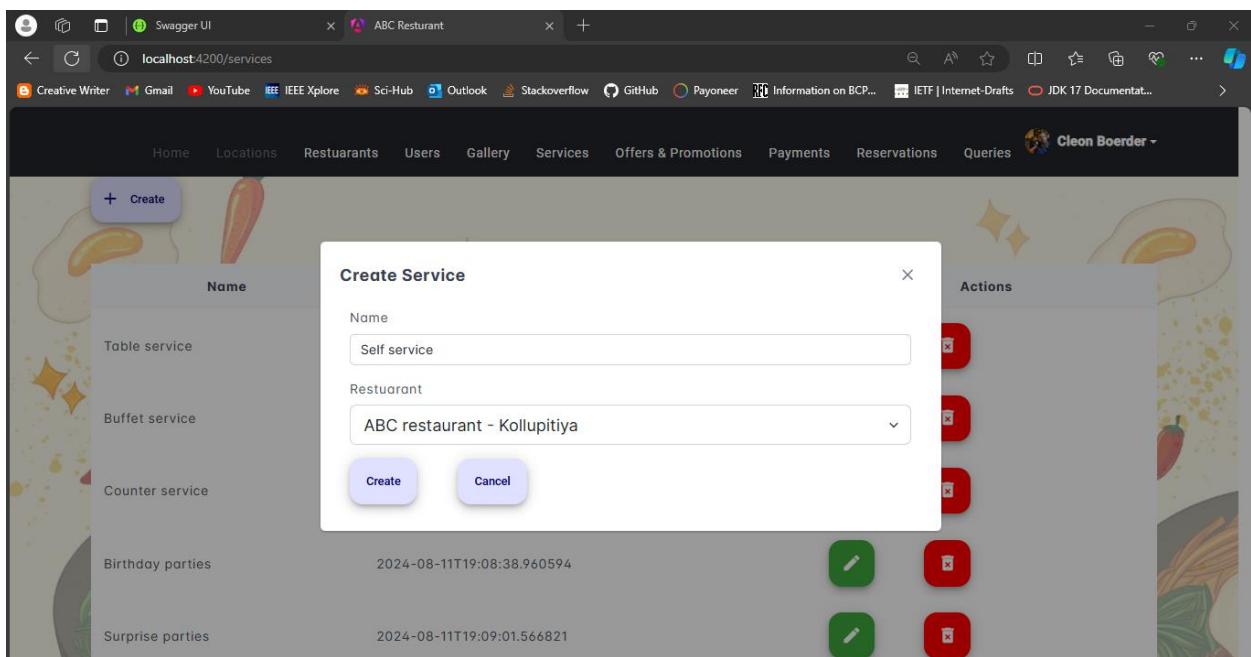
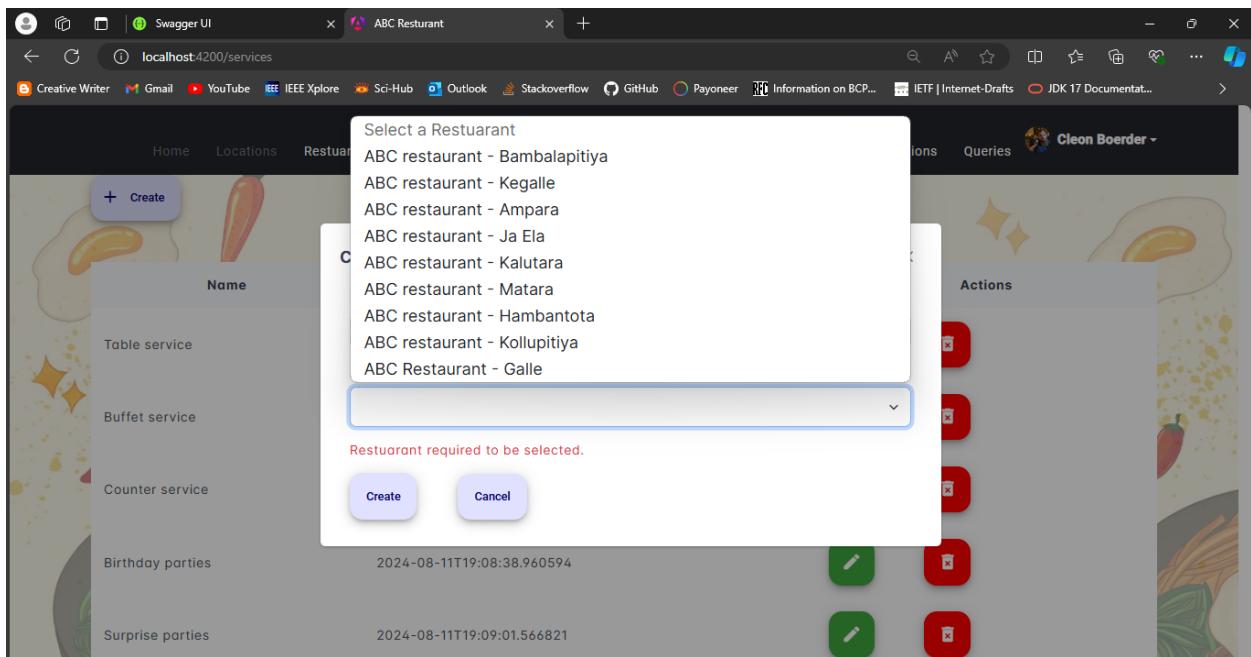
The screenshot shows a web browser window for the ABC Restaurant application. The URL is `localhost:4200/services`. The page displays a table of services with columns: Name, CreatedOn, and Actions. A user profile for "Cleon Boerder" is shown at the top right. A "Create" button is located in the top left corner. The table data is as follows:

Name	CreatedOn	Actions
Table service	2024-08-11T19:06:43.901153	
Buffet service	2024-08-11T19:07:02.523287	
Counter service	2024-08-11T19:07:19.939324	
Birthday parties	2024-08-11T19:08:38.960594	
Surprise parties	2024-08-11T19:09:01.566821	

Create service

Form validations





ABC Restaurant

New Service Created

Name	CreatedOn	Actions
Table service	2024-08-11T19:06:43.901153	
Buffet service	2024-08-11T19:07:02.523287	
Counter service	2024-08-11T19:07:19.939324	
Birthday parties	2024-08-11T19:08:38.960594	
Surprise parties	2024-08-11T19:09:01.566821	

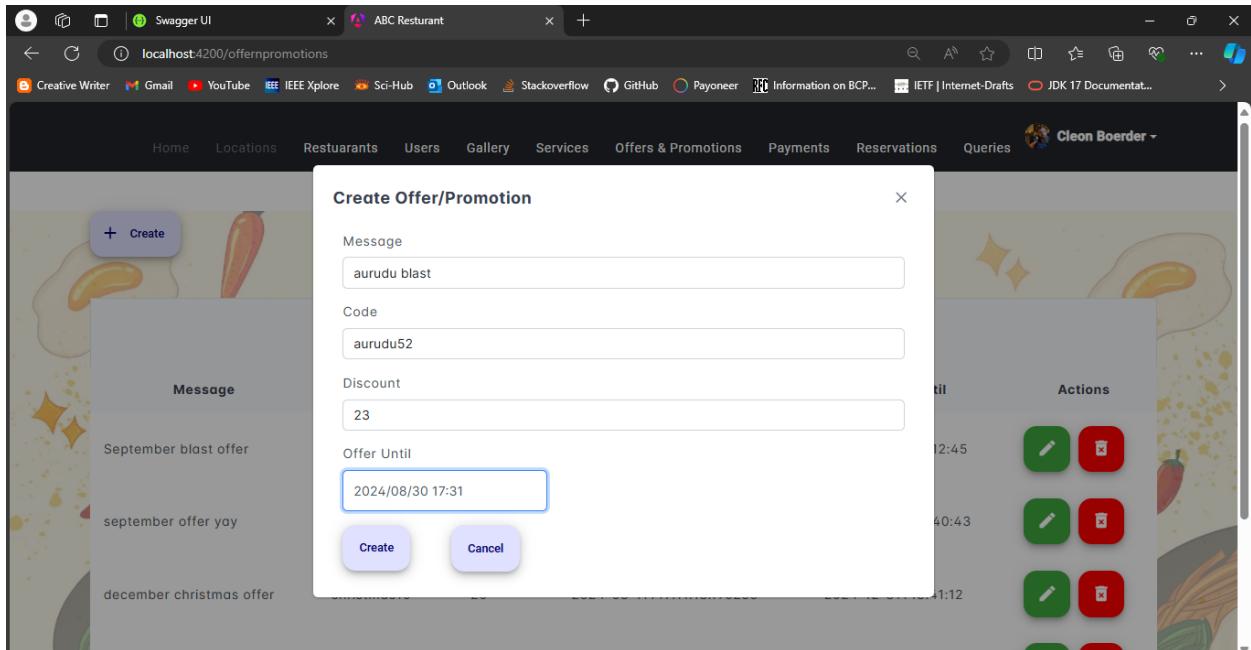
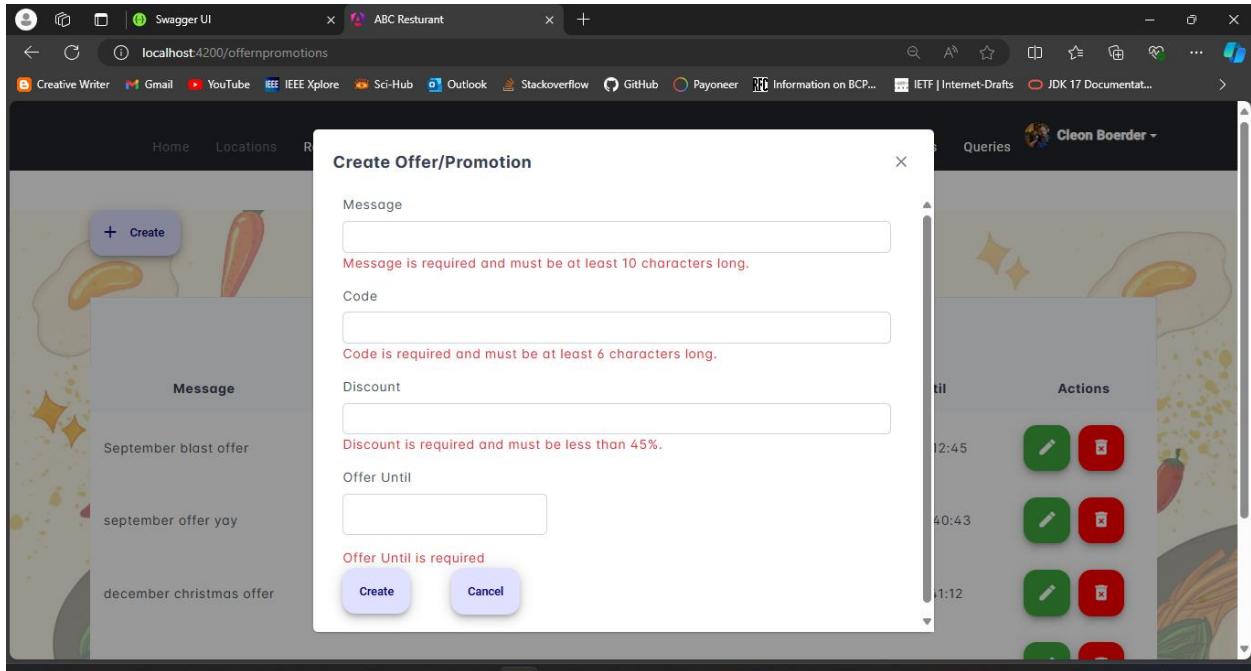
ABC Restaurant

Name	CreatedOn	Actions
Self service	2024-08-12T13:51:09.738445	
dinner buffet	2024-08-12T17:27:57.729366	
dinner buffet with salad corner	2024-08-12T17:28:08.010811	

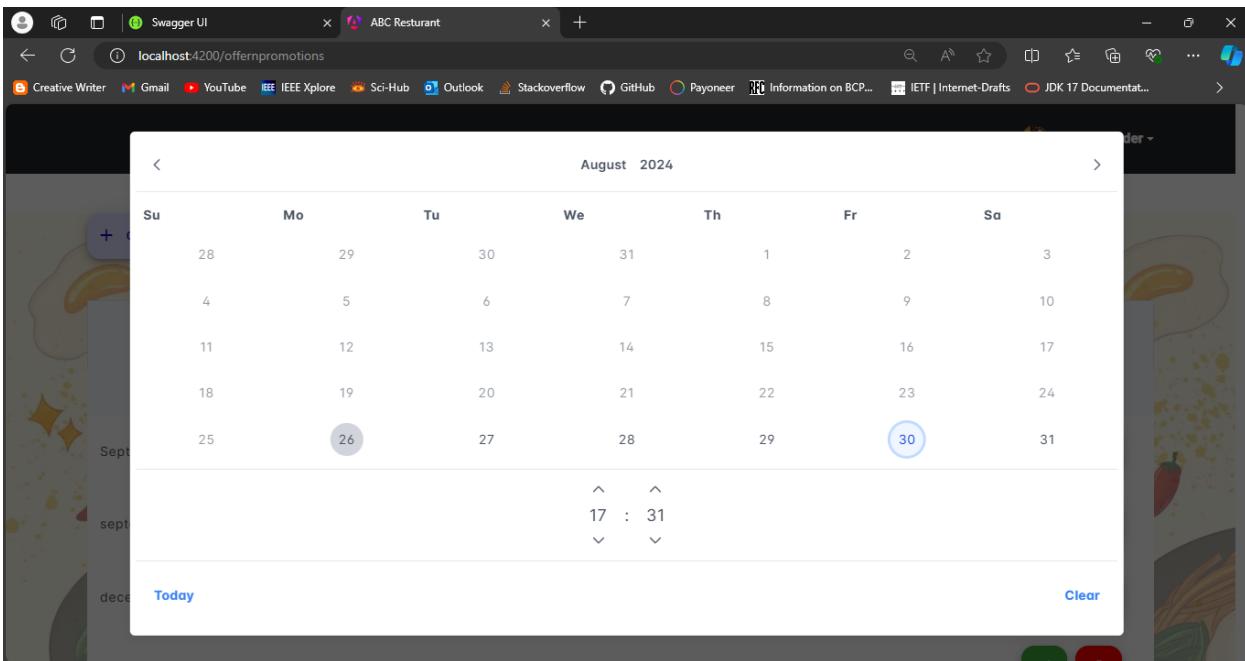
Offers and promotions management

Create offer and promotion

Form validations



Date picker



A screenshot of a dashboard titled 'ABC Restaurant'. The top navigation bar includes links for Home, Locations, Restaurants, Users, Gallery, Services, Offers & Promotions, Payments, Reservations, and Help. A search bar at the top right contains the placeholder 'Search keyword'. Below the navigation, there is a table listing three offers:

Message	Code	Discount	CreatedOn	Offer Until	Actions
September blast offer	september42	15	2024-08-11T19:10:20.965714	2024-08-28T17:12:45	
september offer yay	september301	12	2024-08-11T19:10:49.556762	2024-09-25T13:40:43	
december christmas offer	christmas10	20	2024-08-11T19:11:18.196255	2024-12-31T13:41:12	

A blue confirmation message box is overlaid on the dashboard, stating 'Confirmed' and 'New Offer Created'.

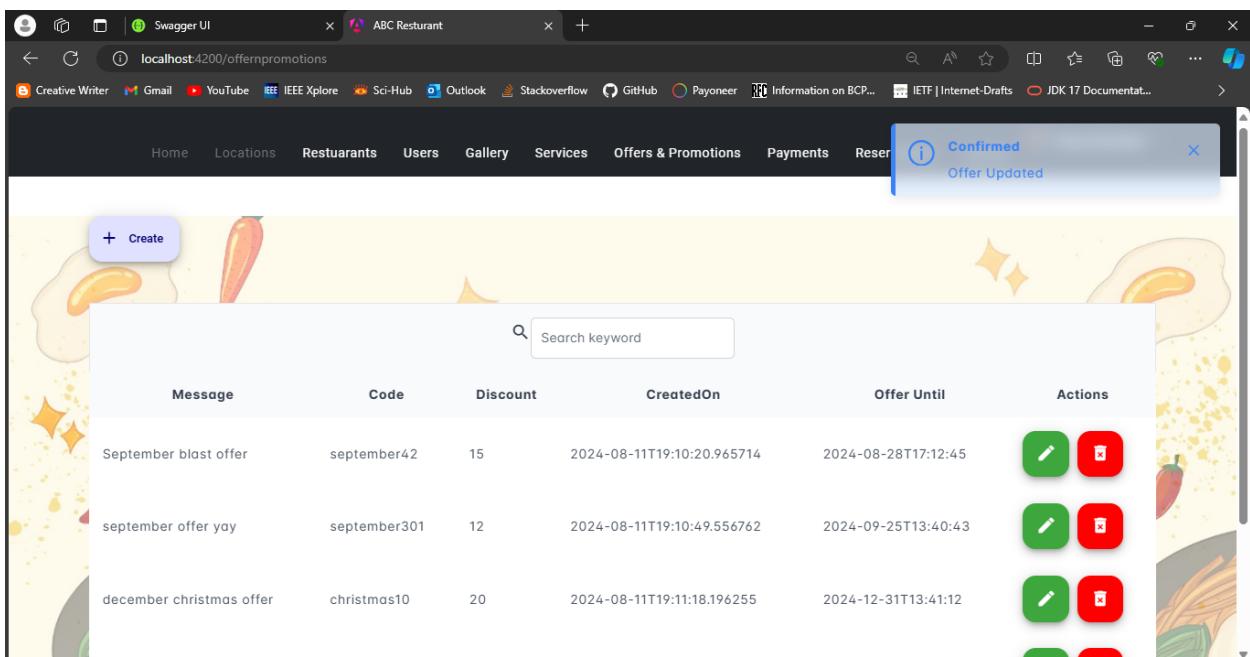
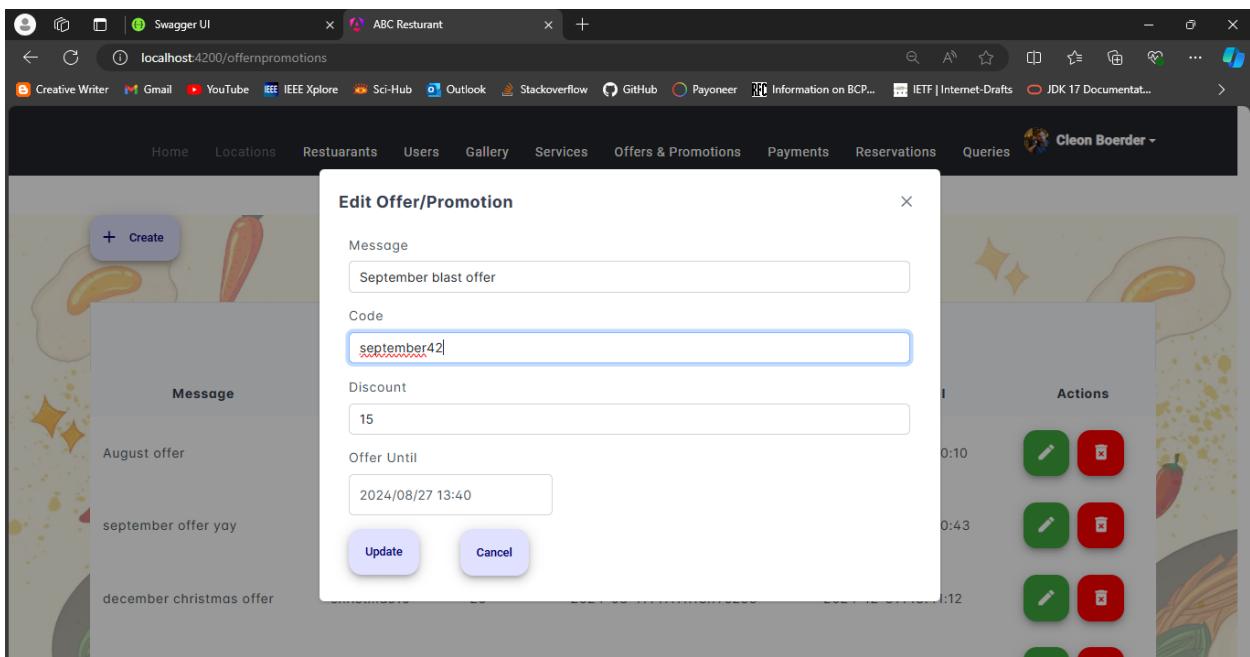
Offers & Promotions					
Message	Code	Discount	CreatedOn	Offer Until	Actions
all you can eat offer	eat2024	20	2024-08-12T17:28:49.782592	2024-08-27T11:58:47	 
aurudu blast	aurudu52	23	2024-08-26T17:32:36.834026	2024-08-30T12:01:40	 

Edit offer and promotion

Edit Offer/Promotion

Message	August offer
Code	august10
Discount	15
Offer Until	2024/08/27 13:40
Actions	 

Above offer was edited as below



Successfully updated

The screenshot shows a table of offers and promotions. One row, 'September blast offer' (Message: september42, Code: 15, CreatedOn: 2024-08-11T19:10:20.965714, Offer Until: 2024-08-28T17:12:45), is highlighted with a red border. A search bar at the top right contains 'Search keyword'. Action icons (edit and delete) are visible next to each row.

Message	Code	Discount	CreatedOn	Offer Until	Actions
September blast offer	september42	15	2024-08-11T19:10:20.965714	2024-08-28T17:12:45	
september offer yay	september301	12	2024-08-11T19:10:49.556762	2024-09-25T13:40:43	
december christmas offer	christmas10	20	2024-08-11T19:11:18.196255	2024-12-31T13:41:12	

Delete offer and promotion

A confirmation dialog box is displayed in the center of the screen, asking 'Are you sure? Please confirm to proceed.' with 'Okay' and 'Cancel' buttons. The background shows the same table of offers and promotions as the previous screenshot.

Message	Code	Discount	CreatedOn	Offer Until	Actions
September blast offer	september42	15	2024-08-11T19:10:20.965714	2024-08-28T17:12:45	
september offer yay	september301	12	2024-08-11T19:10:49.556762	2024-09-25T13:40:43	
december christmas offer	christmas10	20	2024-08-11T19:11:18.196255	2024-12-31T13:41:12	

September blast offer deleted

The screenshot shows a browser window for the ABC Restaurant application. The URL is `localhost:4200/offerpromotions`. The main content area displays a table of offers with columns: Message, Code, Discount, CreatedOn, Offer Until, and Actions. Three rows are listed:

Message	Code	Discount	CreatedOn	Offer Until	Actions
september offer yay	september301	12	2024-08-11T19:10:49.556762	2024-09-25T13:40:43	
december christmas offer	christmas10	20	2024-08-11T19:11:18.196255	2024-12-31T13:41:12	
black friday offer	friday10	10	2024-08-11T19:12:43.057076	2024-12-26T13:42:37	

A blue notification bar at the top right says "Confirmed Offer Deleted". A search bar is also visible above the table.

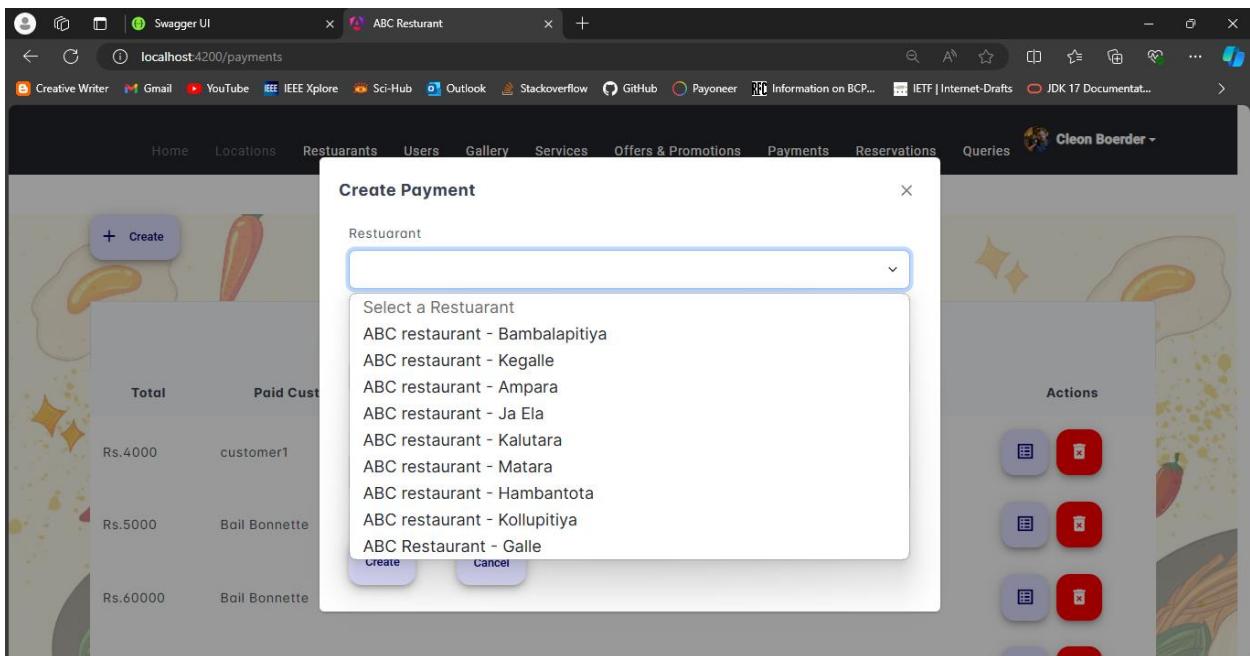
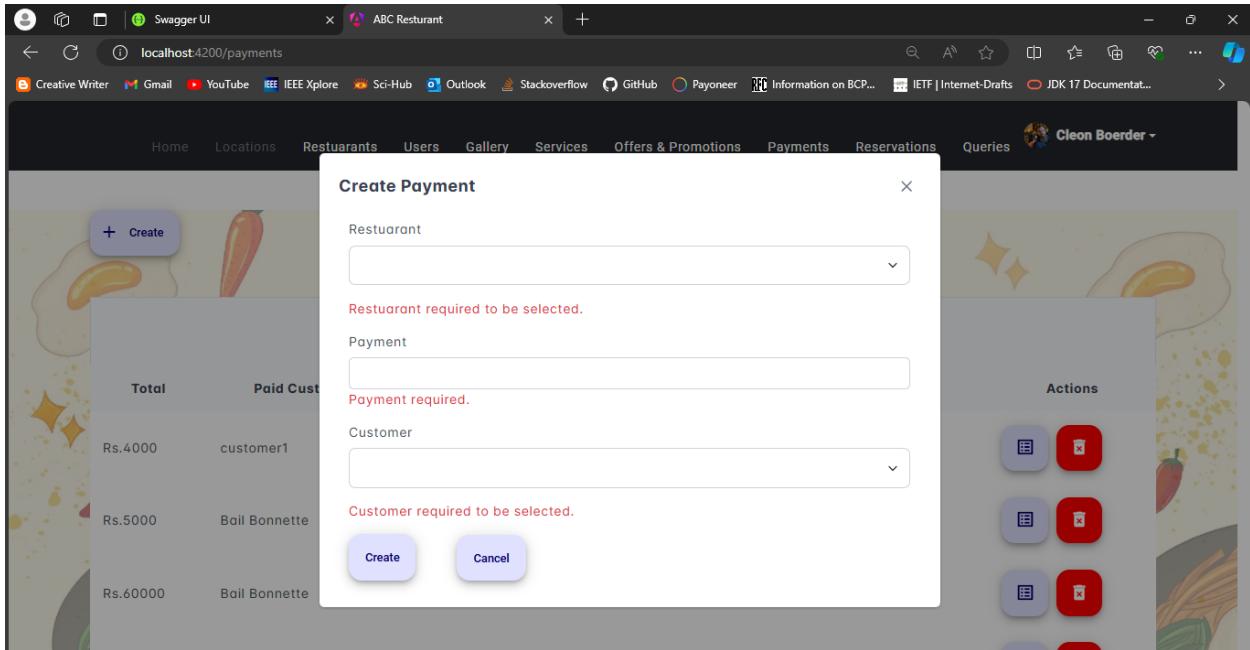
Manage payments

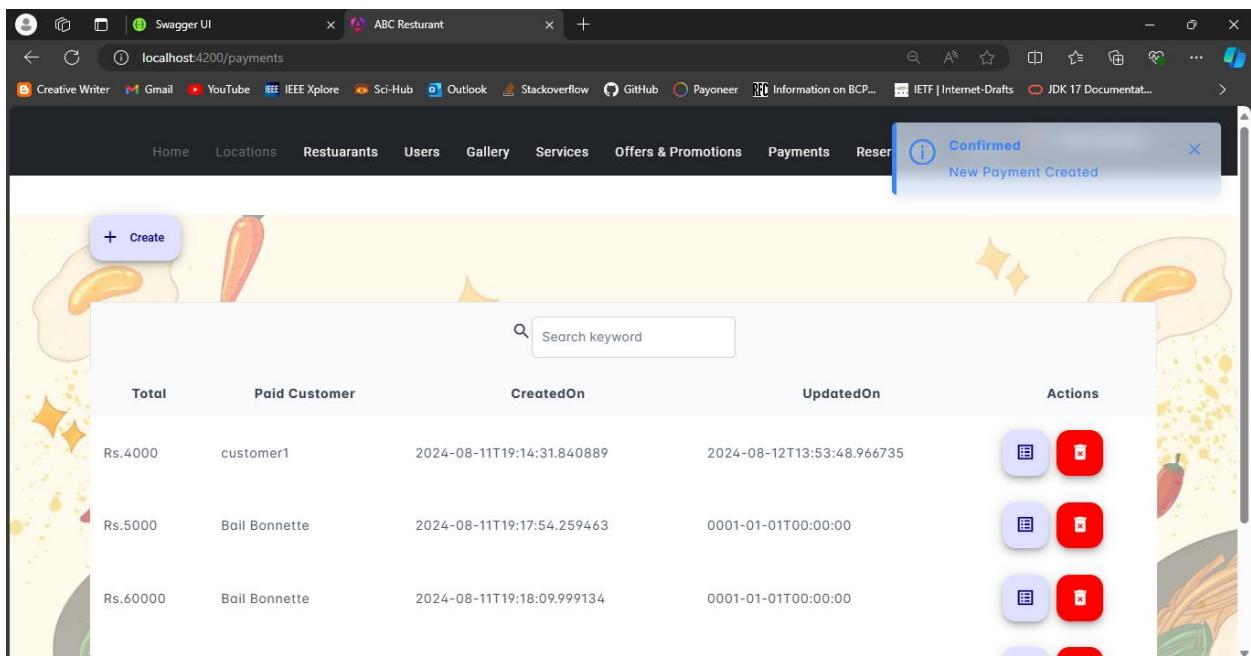
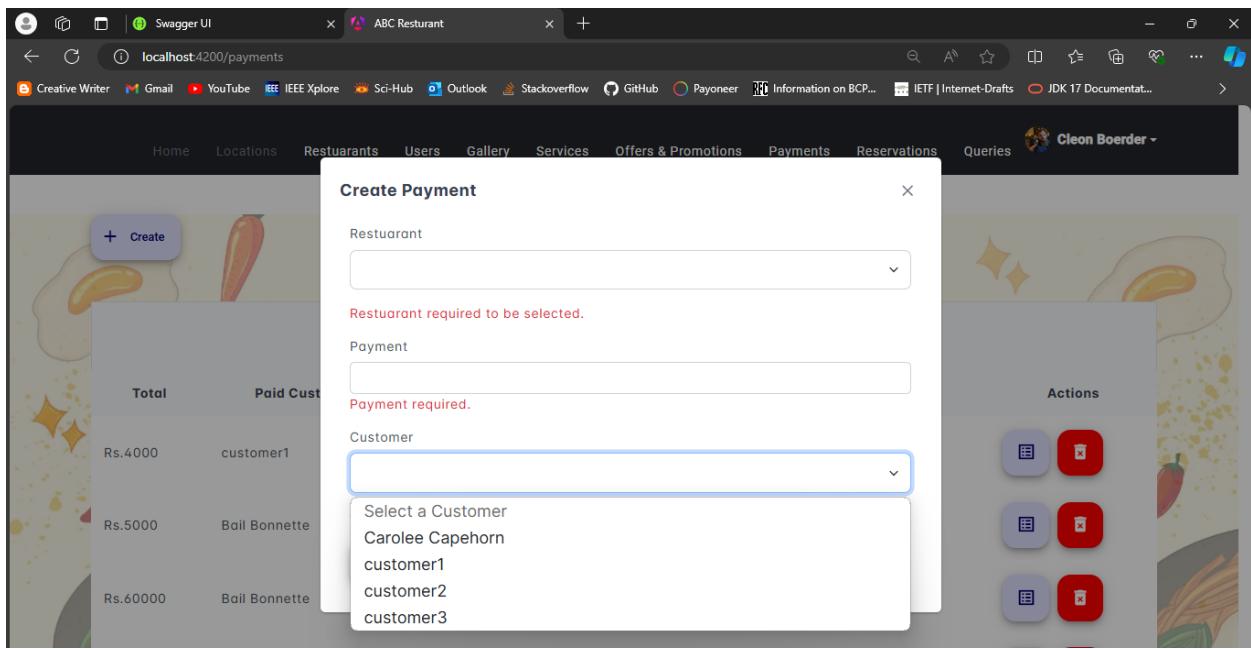
The screenshot shows a browser window for the ABC Restaurant application. The URL is `localhost:4200/payments`. The main content area displays a table of payments with columns: Total, Paid Customer, CreatedOn, UpdatedOn, and Actions. Three rows are listed:

Total	Paid Customer	CreatedOn	UpdatedOn	Actions
Rs.4000	customer1	2024-08-11T19:14:31.840889	2024-08-12T13:53:48.966735	
Rs.5000	Bail Bonnette	2024-08-11T19:17:54.259463	0001-01-01T00:00:00	
Rs.60000	Bail Bonnette	2024-08-11T19:18:09.999134	0001-01-01T00:00:00	

Create payment

Form validations





Total	Paid Customer	CreatedOn	UpdatedOn	Actions
Rs.20000	Dael Garaghan	2024-08-12T17:40:29.585999	0001-01-01T00:00:00	 
Rs.5000	Carolee Capehorn	2024-08-26T17:39:15.342264	0001-01-01T00:00:00	 

View payment details

Payment details

Total	Paid Customer
4000	customer1
Restuarant	ABC restaurant - Bambalapitiya
Paid Customer	customer1
Created On	Sunday, August 11, 2024
Updated On	Monday, August 12, 2024

Delete payment

The screenshot shows a table of payments with columns: Total, Paid Customer, CreatedOn, UpdatedOn, and Actions. A modal dialog box is centered over the table, containing a warning icon and the text: "Are you sure? Please confirm to proceed." with "Okay" and "Cancel" buttons.

Total	Paid Customer	CreatedOn	UpdatedOn	Actions
Rs.4000	customer1	2024-08-11	2024-08-12T13:53:48.966735	
Rs.5000	Bail Bonnette	2024-08-11T19:17:54.259463	0001-01-01T00:00:00	
Rs.60000	Bail Bonnette	2024-08-11T19:18:09.999134	0001-01-01T00:00:00	

Customer1's payment is deleted

The screenshot shows a table of payments with columns: Total, Paid Customer, CreatedOn, UpdatedOn, and Actions. A blue notification bar at the top right says "Confirmed Payment Deleted". A search bar is also visible above the table.

Total	Paid Customer	CreatedOn	UpdatedOn	Actions
Rs.5000	Bail Bonnette	2024-08-11T19:17:54.259463	0001-01-01T00:00:00	
Rs.60000	Bail Bonnette	2024-08-11T19:18:09.999134	0001-01-01T00:00:00	
Rs.5000	Carolee Capehorn	2024-08-11T19:18:22.883557	0001-01-01T00:00:00	

Reservation management

The screenshot shows a reservation management interface for the ABC Restaurant. At the top, there is a navigation bar with links for Home, Locations, Restaurants, Users, Gallery, Services, Offers & Promotions, Payments, Reservations, and Queries. A user profile for Cleon Boerder is visible on the right. Below the navigation bar is a decorative header featuring a stylized illustration of a sandwich and various toppings. A blue button labeled '+ Create' is located in the top-left corner of the main content area. To the right, a 'Full Report' button is shown. The main content area displays a table of reservations:

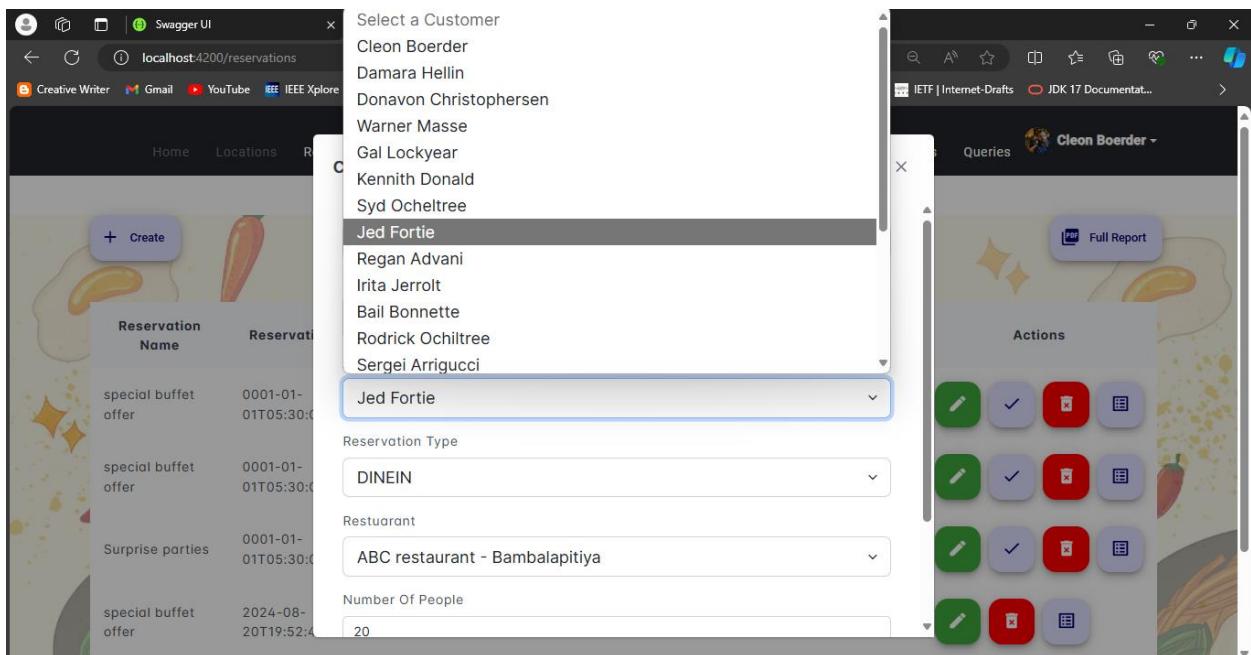
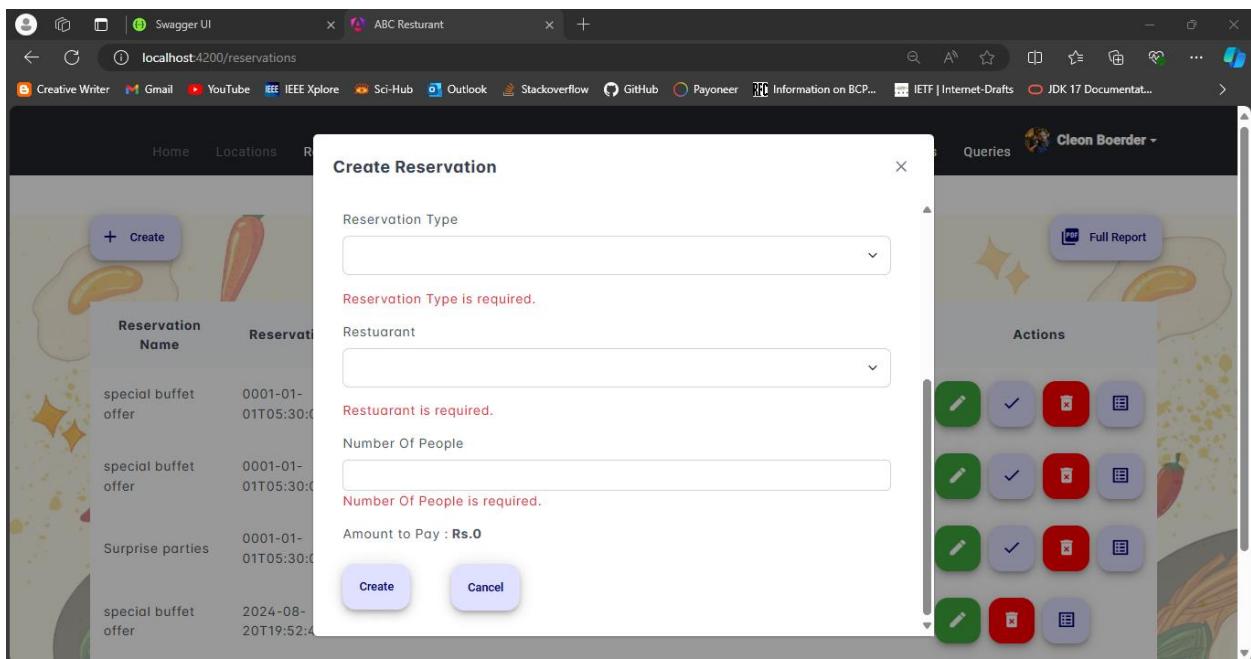
Reservation Name	Reservation On	Type	User Name	Approval Pending	Created On	Actions
special buffet offer	0001-01-01T05:30:00	Dine In	Syd Ocheltree	<input type="checkbox"/>	2024-08-11T19:40:23.506693	
special buffet offer	0001-01-01T05:30:00	Delivery	Kenneth Donald	<input type="checkbox"/>	2024-08-11T19:40:45.029187	
Surprise parties	0001-01-01T05:30:00	Dine In	Rodrick Ochiltree	<input type="checkbox"/>	2024-08-11T19:41:08.446915	
special buffet offer	2024-08-20T19:52:49	Dine In	customer3	<input checked="" type="checkbox"/>	2024-08-11T19:53:11.8017	

Create reservation

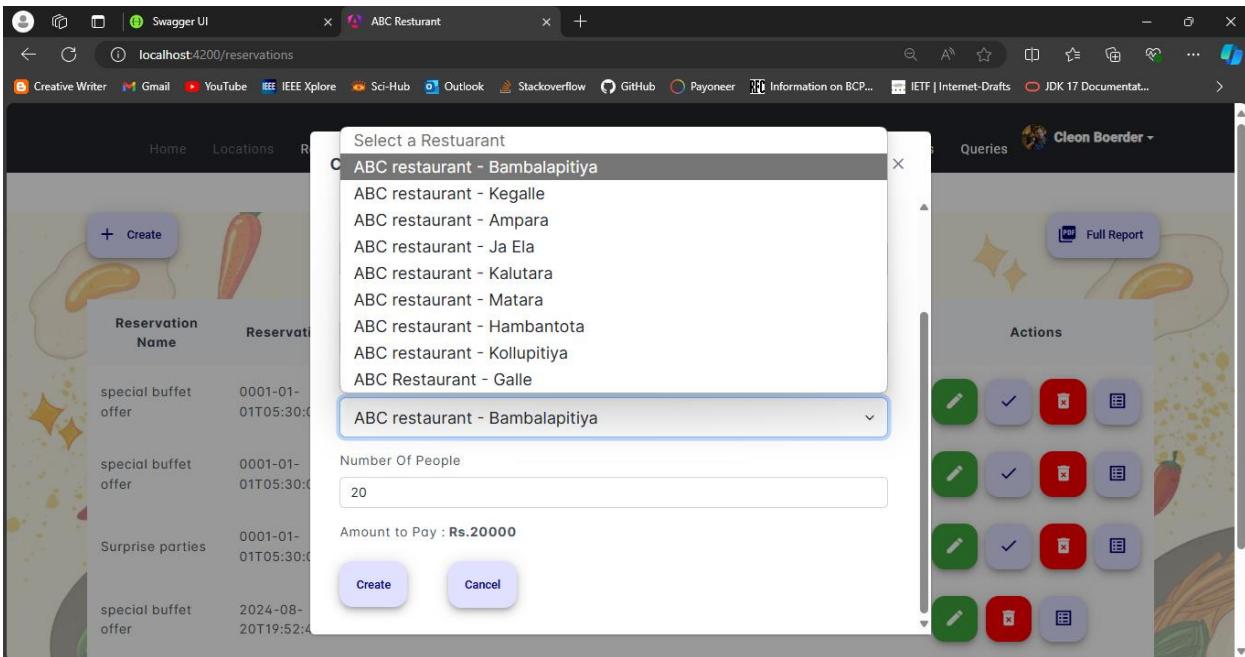
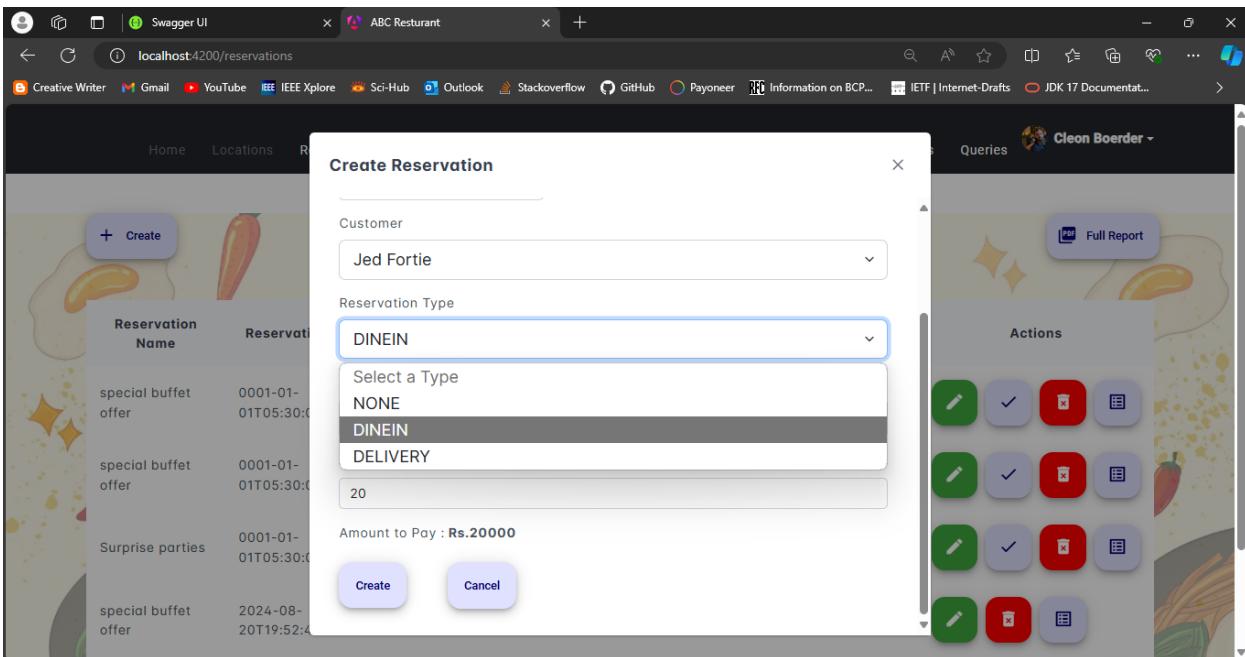
Form validations

The screenshot shows a 'Create Reservation' modal window overlaid on the reservation management interface. The modal has fields for Name, Reservation Date, Customer, Reservation Type, and Restaurant. Validation errors are displayed in red text below each field:

- Name: "Name is required and must be at least 5 characters long."
- Reservation Date: "Reservation Date is required."
- Customer: "Customer is required."
- Reservation Type: "Reservation Type is required."
- Restaurant: (No error message shown)



Reservation types are dine-in, delivery



Notification is received – new reservation created.

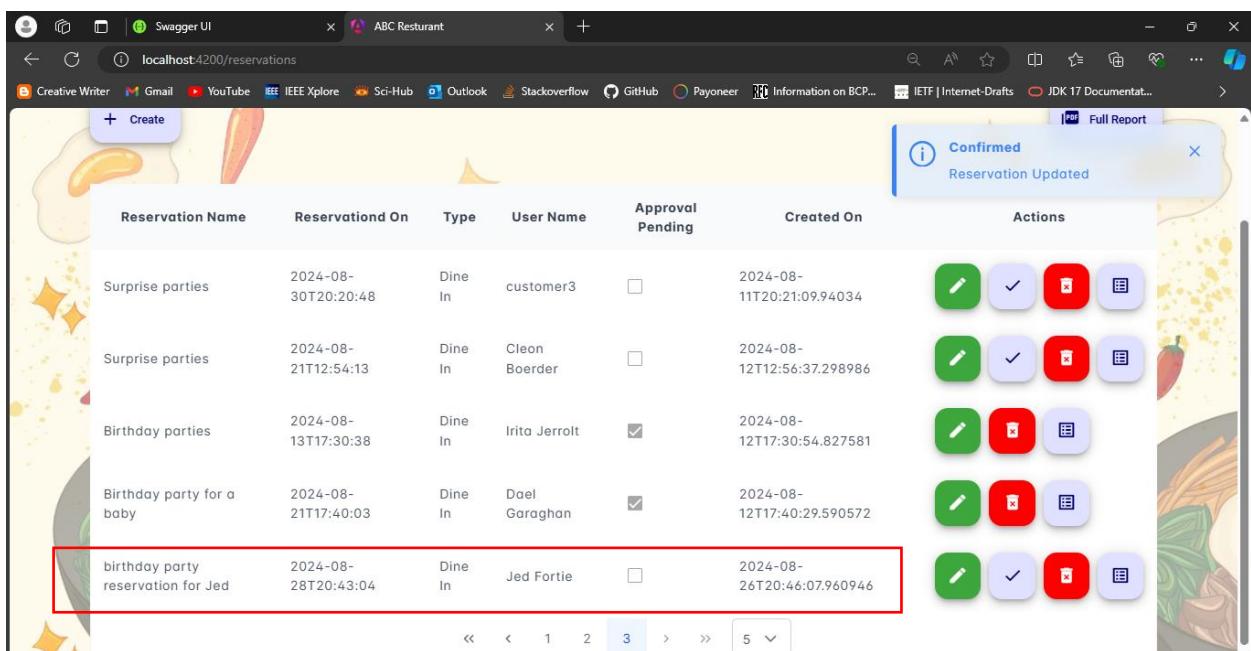
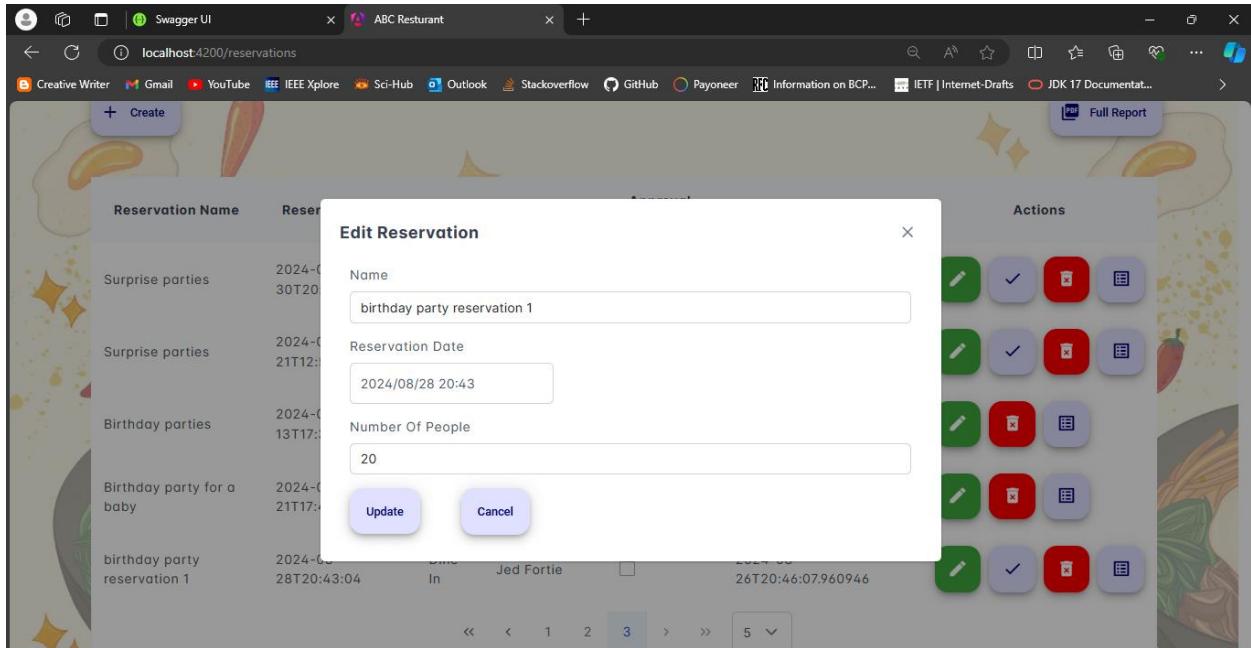
The screenshot shows a web browser window for the ABC Restaurant reservation system. At the top, there is a navigation bar with links for Home, Locations, Restaurants, Users, Gallery, Services, Offers & Promotions, Payments, Reservations, and a search bar. A prominent blue notification bubble in the top right corner says "Confirmed" and "New Reservation Created". Below the notification is a table listing four reservations:

Reservation Name	Reservation On	Type	User Name	Approval Pending	Created On	Actions
special buffet offer	0001-01-01T05:30:00	Dine In	Syd Ocheltree	<input type="checkbox"/>	2024-08-11T19:40:23.506693	
special buffet offer	0001-01-01T05:30:00	Delivery	Kenneth Donald	<input type="checkbox"/>	2024-08-11T19:40:45.029187	
Surprise parties	0001-01-01T05:30:00	Dine In	Rodrick Ochiltree	<input type="checkbox"/>	2024-08-11T19:41:08.446915	
special buffet offer	2024-08-20T19:52:49	Dine In	customer3	<input checked="" type="checkbox"/>	2024-08-11T19:53:11.8017	

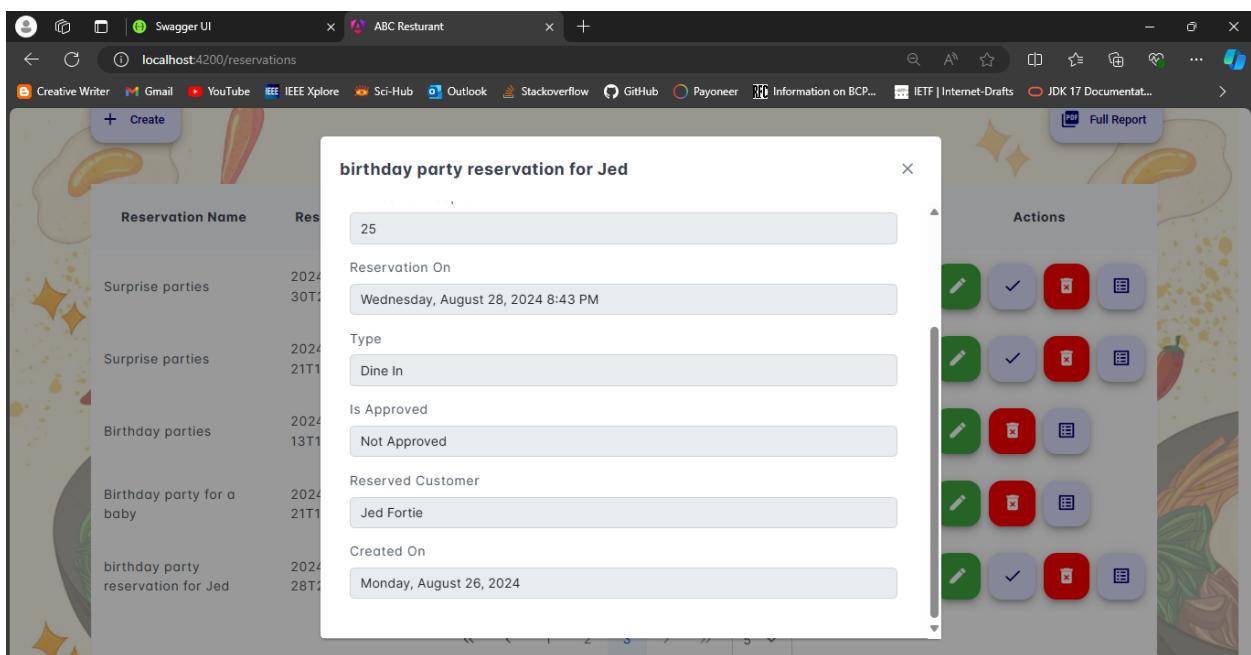
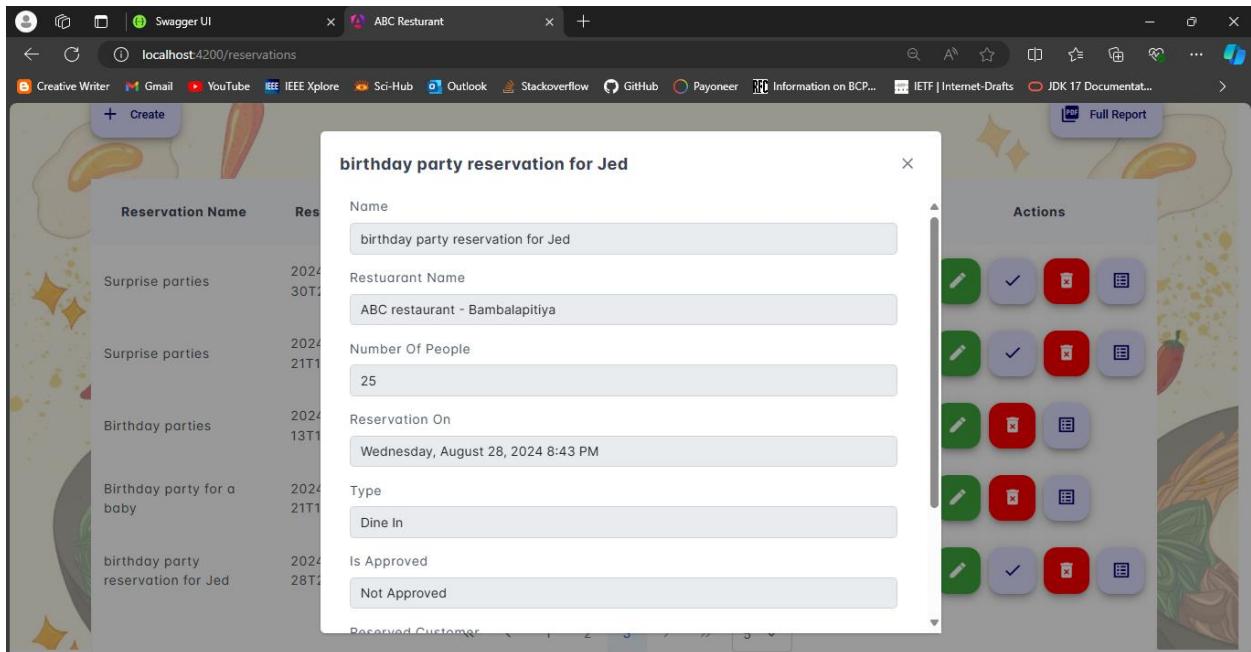
In the second screenshot, the list of reservations has grown to five entries. The fifth entry, "birthday party reservation 1", is highlighted with a red rectangular box. The table data is as follows:

Reservation Name	Reservation On	Type	User Name	Approval Pending	Created On	Actions
Surprise parties	2024-08-30T20:20:48	Dine In	customer3	<input type="checkbox"/>	2024-08-11T20:21:09.94034	
Surprise parties	2024-08-21T12:54:13	Dine In	Cleon Boerder	<input type="checkbox"/>	2024-08-12T12:56:37.298986	
Birthday parties	2024-08-13T17:30:38	Dine In	Irita Jerrott	<input checked="" type="checkbox"/>	2024-08-12T17:30:54.827581	
Birthday party for a baby	2024-08-21T17:40:03	Dine In	Dael Garaghan	<input checked="" type="checkbox"/>	2024-08-12T17:40:29.590572	
birthday party reservation 1	2024-08-28T20:43:04	Dine In	Jed Fortie	<input type="checkbox"/>	2024-08-26T20:46:07.960946	

Edit reservation



View reservation



Confirm reservation

Once the confirm button is clicked under actions, the confirm button disappears and approval pending is ticked.

The screenshot shows a table of reservations with columns: Reservation Name, Reservation On, Type, User Name, Approval Pending, and Created On. A modal window titled 'Approved' displays the message 'Reservation Approved'. A specific row for 'birthday party reservation for Jed' is highlighted with a red border. The 'Actions' column for this row contains five icons: a green pencil, a blue checkmark, a red square with a white checkmark, a blue square with a white minus sign, and a light blue square with a white plus sign. The 'Approval Pending' column for this row has a checked checkbox.

Reservation Name	Reservation On	Type	User Name	Approval Pending	Created On	Actions
Surprise parties	2024-08-30T20:20:48	Dine In	customer3	<input type="checkbox"/>	2024-08-11T20:21:09.94034	
Surprise parties	2024-08-21T12:54:13	Dine In	Cleon Boerder	<input type="checkbox"/>	2024-08-12T12:56:37.298986	
Birthday parties	2024-08-13T17:30:38	Dine In	Irita Jerrott	<input checked="" type="checkbox"/>	2024-08-12T17:30:54.827581	
Birthday party for a baby	2024-08-21T17:40:03	Dine In	Dael Garaghan	<input checked="" type="checkbox"/>	2024-08-12T17:40:29.590572	
birthday party reservation for Jed	2024-08-28T20:43:04	Dine In	Jed Fortie	<input checked="" type="checkbox"/>	2024-08-26T20:46:07.960946	

The confirmation email is sent

The screenshot shows the Papercut SMTP application interface. On the left, a log window lists numerous messages related to reservations being approved. On the right, the main window shows an email message with the following details:

From: "ABC Restaurant Support" <ABCRestaurant@Service.email>
To: "Jed Fortie" <jfortie9@artisteer.com>
Date: 8/26/2024 8:50:29 PM +05:30
Subject: You Reservation has been Approved

Message Headers Body Sections Raw

Hello Jed Fortie,
Your reservation name birthday party reservation for Jed has been approved by our staff and it will be held on 8/28/2024 8:43:04 PM.
Thank you for using our services

FORWARD **DELETE (1)** **DELETE ALL**

Delete reservation

Deleting reservation “birthday party reservation for Jed”

The screenshot shows a list of reservations in a table. A confirmation dialog box is overlaid on the screen, asking "Are you sure? Please confirm to proceed." with "Okay" and "Cancel" buttons. The table data is as follows:

Reservation Name	Reservation On	Type	User Name	Approval Pending	Created On	Actions
Surprise parties	2024-08-30T20:20:48	Dine In			2024-08-30T20:21:09.94034	
Surprise parties	2024-08-21T12:54:13	Dine In			2024-08-21T12:56:37.298986	
Birthday parties	2024-08-13T17:30:38	Dine In			2024-08-13T17:30:54.827581	
Birthday party for a baby	2024-08-21T17:40:03	Dine In	Jael Garaghan	<input checked="" type="checkbox"/>	2024-08-21T17:40:29.590572	
birthday party reservation for Jed	2024-08-28T20:43:04	Dine In	Jed Fortie	<input checked="" type="checkbox"/>	2024-08-26T20:46:07.960946	

The screenshot shows the same reservation list after one item has been deleted. A blue confirmation box at the top right says "Confirmed Reservation Deleted". The table data is as follows:

Reservation Name	Reservation On	Type	User Name	Approval Pending	Created On	Actions
Surprise parties	2024-08-30T20:20:48	Dine In	customer3	<input type="checkbox"/>	2024-08-31T20:21:09.94034	
Surprise parties	2024-08-21T12:54:13	Dine In	Cleon Boerder	<input type="checkbox"/>	2024-08-21T12:56:37.298986	
Birthday parties	2024-08-13T17:30:38	Dine In	Irita Jerrott	<input checked="" type="checkbox"/>	2024-08-13T17:30:54.827581	
Birthday party for a baby	2024-08-21T17:40:03	Dine In	Dael Garaghan	<input checked="" type="checkbox"/>	2024-08-21T17:40:29.590572	

Query management

The screenshot shows a web browser window titled "ABC Restaurant" at "localhost:4200/queries". The page has a header with navigation links: Home, Locations, Restaurants, Users, Gallery, Services, Offers & Promotions, Payments, Reservations, Queries, and a user profile for "Cleon Boerder". Below the header is a decorative header section featuring a cartoon illustration of a sandwich and some stars. The main content area displays a table of query logs:

CreatedOn	UpdatedOn	Actions
2024-08-11T14:30:08.427501	0001-01-01T00:00:00	
2024-08-11T19:43:25.953291	0001-01-01T00:00:00	
2024-08-11T19:44:03.578623	0001-01-01T00:00:00	
2024-08-11T19:44:24.135569	0001-01-01T00:00:00	

A "Full Report" button is located in the top right corner of the main content area.

Submit answer to query

Once the edit button under actions is clicked, this pop up appears. Admin/staff can answer this query by typing a message on the message text field

The screenshot shows a modal dialog titled "Edit Query" over the ABC Restaurant query management interface. The dialog contains a "Chat" section with the question "does bambalapitiya restaurant have dine-in facility?". Below it is a "Message" input field containing the text "hello, we do have dine-in facilities in Bambalapitiya restaurant.". At the bottom of the dialog are "Submit" and "Cancel" buttons. The background of the main interface shows the same query log table as the previous screenshot.

When the query is answered, an email is received.

The screenshot shows two windows side-by-side. The top window is the ABC Restaurant application's 'Queries' page, featuring a table with columns for 'CreatedOn', 'UpdatedOn', and 'Actions'. The bottom window is the Papercut SMTP inbox, displaying an incoming email message.

ABC Restaurant - Queries

CreatedOn	UpdatedOn	Actions
2024-08-11T14:30:08.427501	0001-01-01T00:00:00	
2024-08-11T19:43:25.953291	0001-01-01T00:00:00	
2024-08-11T19:44:03.578623	0001-01-01T00:00:00	
2024-08-11T19:44:24.135569	0001-01-01T00:00:00	

Papercut SMTP

New Message Received
From: "ABC Restaurant Support" <ABCRestaurant@Service.email>
To: "customer1" <customer1@gmail.com>
Date: 8/26/2024 9:03:54 PM -05:30
Subject: Your Query has a Reply

Message Headers Body Sections Raw

```

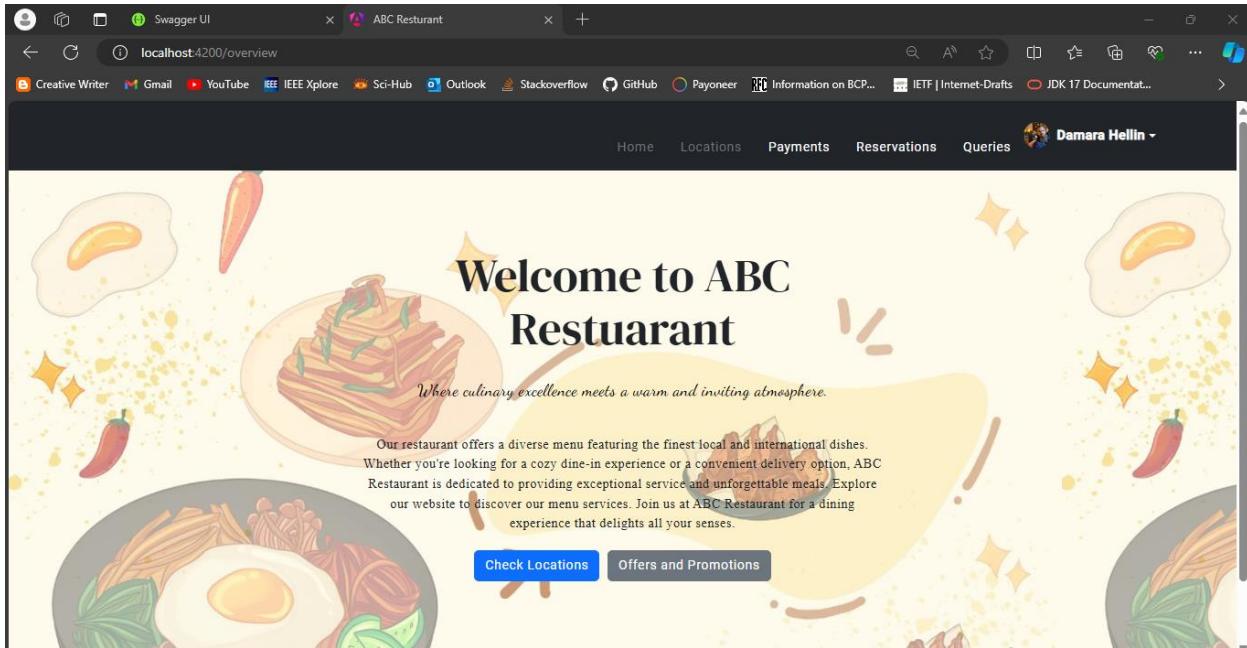
Hello customer1, My name is Cleon Boerder
You have asked : 'does bambalapitiya restaurant have dine-in facility?'
Cleon Boerder: hello, we do have dine-in facilities in Bambalapitiya restaurant.
Thank you for using our services

```

Action Buttons: FORWARD | DELETE (1) | DELETE ALL

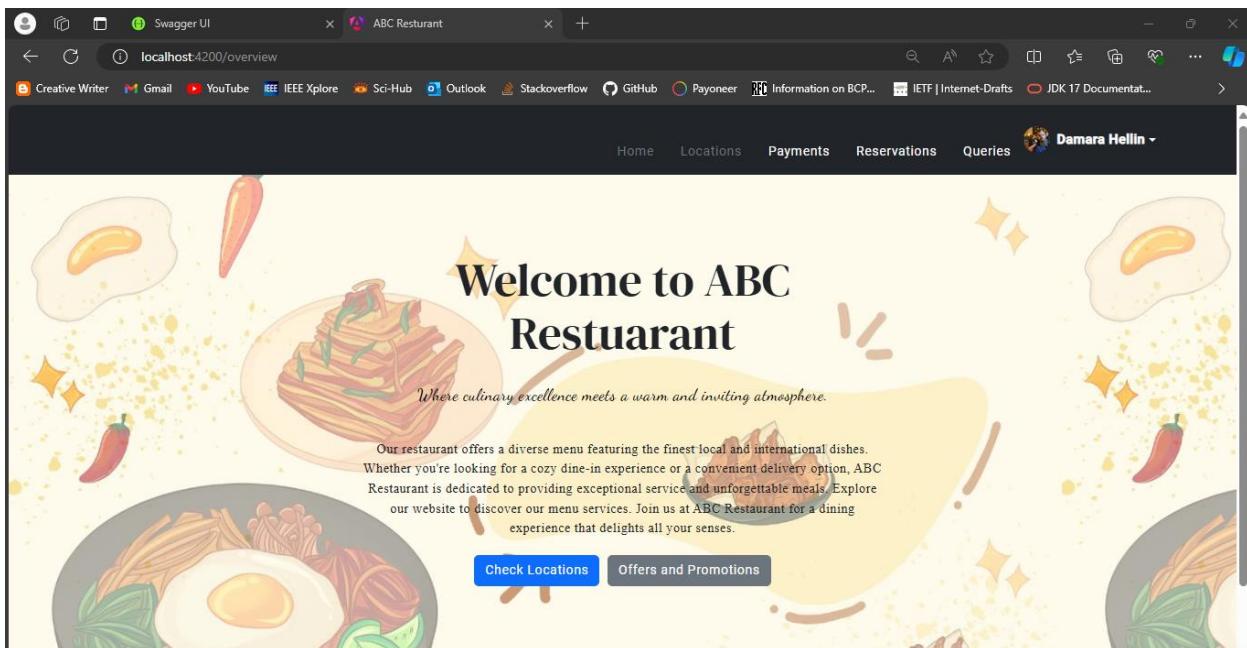
Interfaces for staff

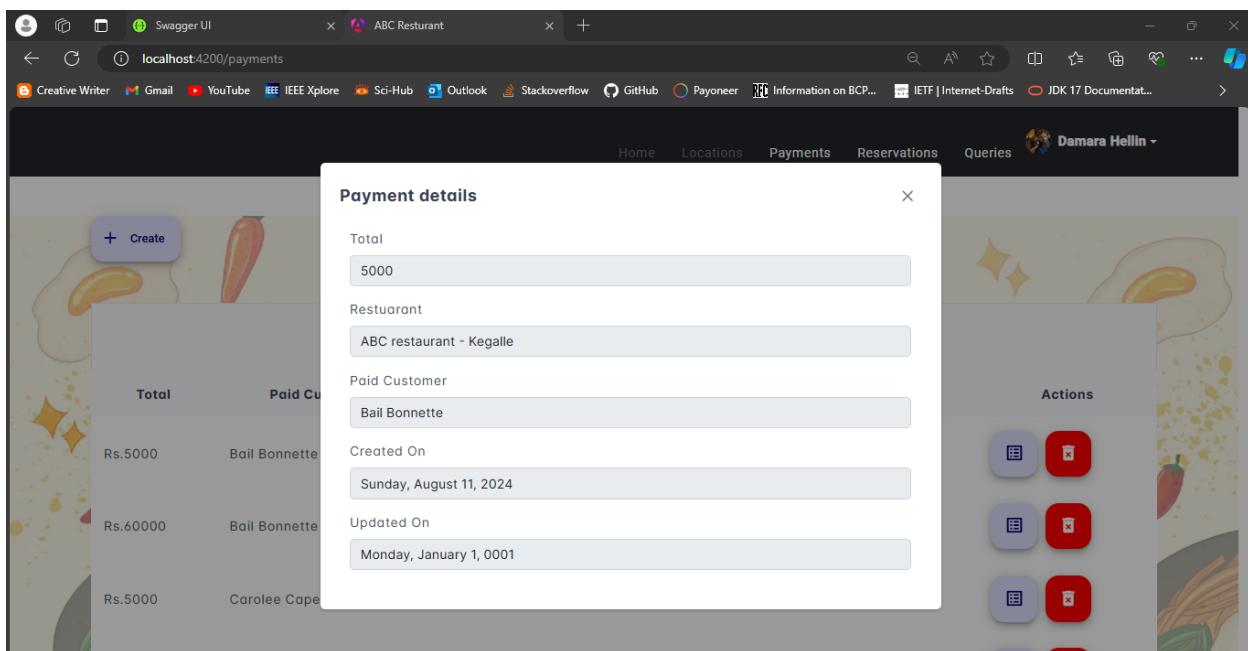
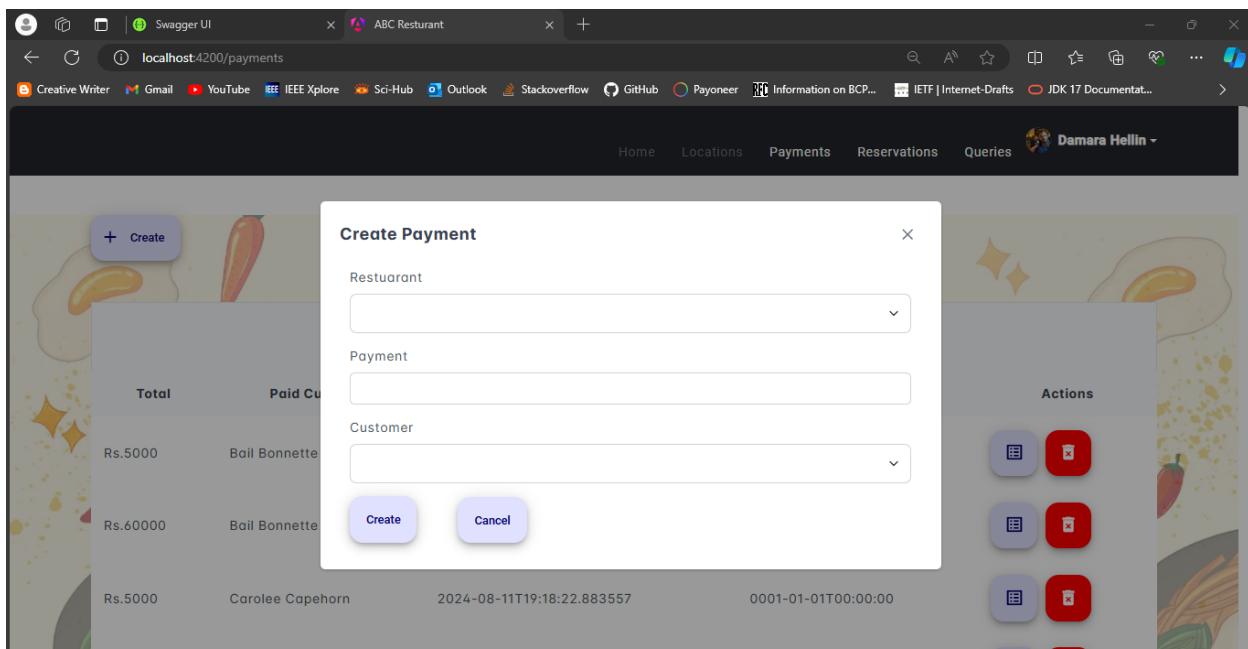
This is the homepage for staff after logging in



Payment management

Staff can create payments, view them and delete them.





Reservation management

Staff can create reservations for customers, edit, approve, view and delete them like admin does. But staff can generate only the full reservation report.

ABC Restaurant

localhost:4200/reservations

Damara Hellin

Reservation Name	Reservation On	Type	User Name	Approval Pending	Created On	Actions
black friday offer	0001-01-01T05:30:00	Delivery	Gal Lockyear	<input checked="" type="checkbox"/>	2024-08-11T19:20:58.664046	
aurudu offer	0001-01-01T05:30:00	Dine In	Rodrick Ochiltree	<input type="checkbox"/>	2024-08-11T19:21:44.773616	
december christmas offer	0001-01-01T05:30:00	Dine In	Kenneth Donald	<input checked="" type="checkbox"/>	2024-08-11T19:32:53.654468	
Surprise parties	0001-01-01T05:30:00	Dine In	Syd Ocheltree	<input checked="" type="checkbox"/>	2024-08-11T19:33:24.382049	

ABC Restaurant

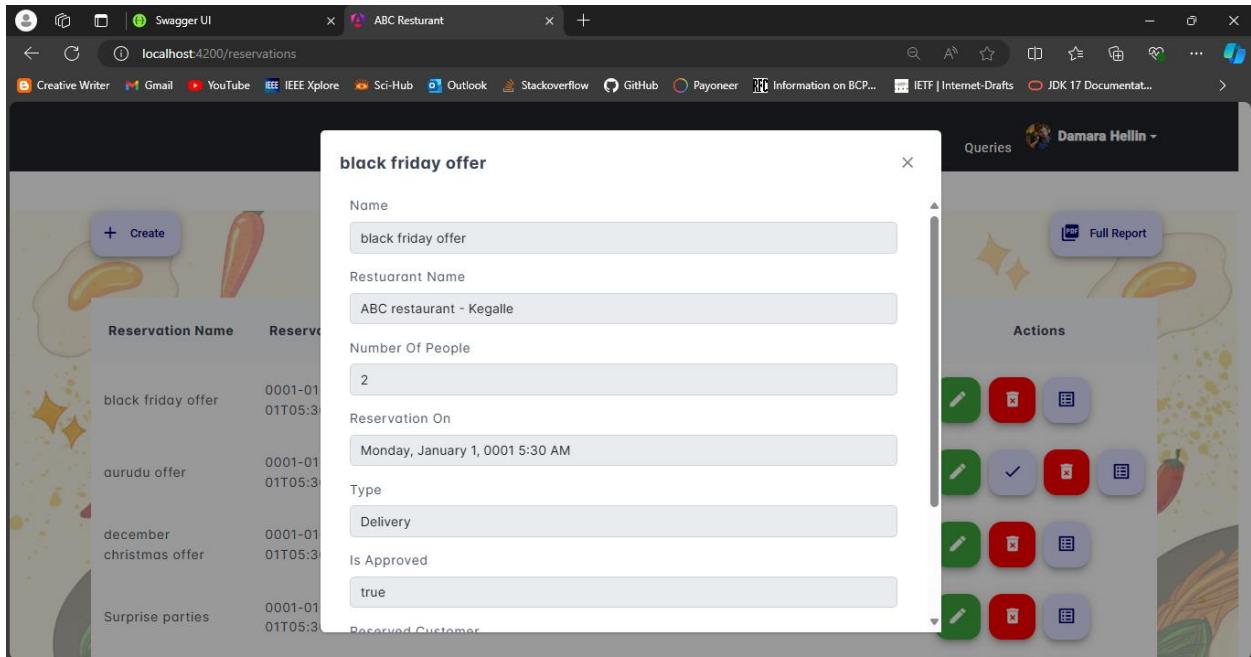
localhost:4200/reservations

Damara Hellin

Edit Reservation

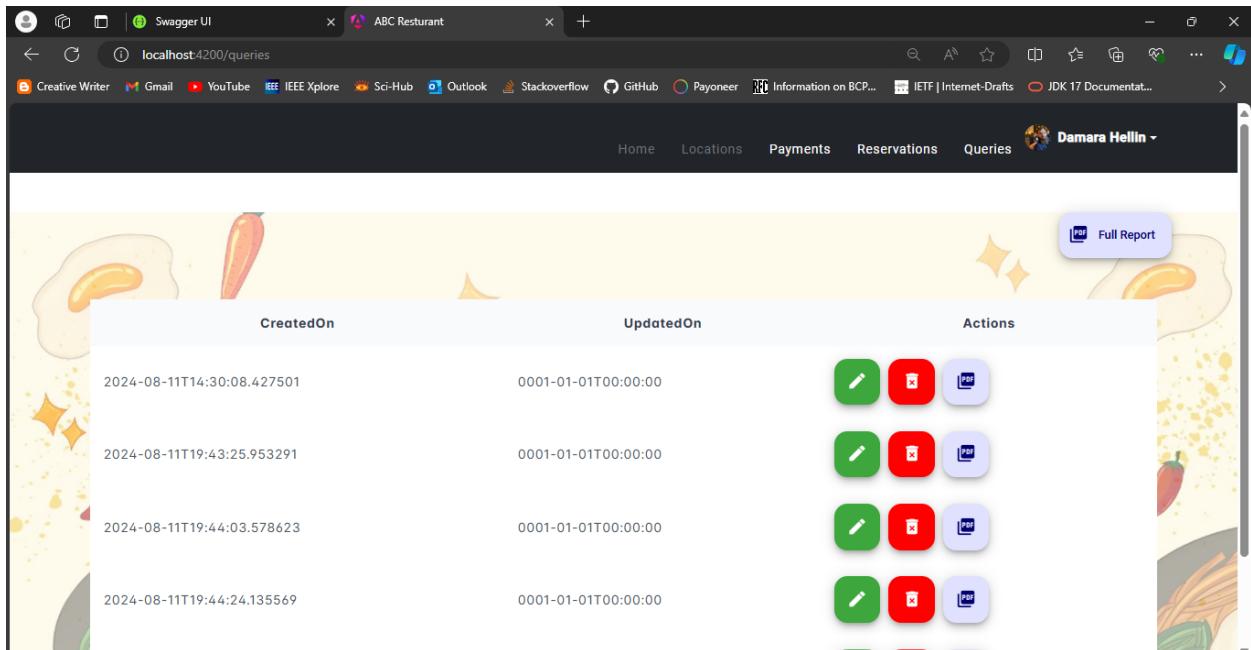
Name	black friday offer
Reservation Date	<input type="text"/>
Number Of People	2

Update **Cancel**



Query management

Staff can respond to queries, delete them and generate full report and query report per customer.



Interfaces for customer

Register

Only customers can register themselves into the system

The screenshot shows a web browser window with the title bar "ABC Restaurant". The address bar displays "localhost:4200/register". The main content area contains a registration form titled "Register". The form fields are as follows:

- Name*: An input field with placeholder text "Enter your name" and a pencil icon.
- Telephone*: An input field with placeholder text "Enter your telephone" and a phone receiver icon.
- Email*: An input field with placeholder text "Enter your email" and an envelope icon.
- Password*: An input field with placeholder text "Enter your password" and a lock icon.

A "Sign in" button is located at the bottom left of the form area.

Form validations

The screenshot shows a web browser window with the title bar "ABC Restaurant". The address bar displays "localhost:4200/register". The main content area contains a registration form titled "Register" with validation errors:

- Name***: The error message is "Name is required and must be at least 5 characters long." The input field has a red border.
- Telephone***: The error message is "Telephone is required and must be a valid 10-digit number." The input field has a red border.
- Email***: The error message is "Email is required and must in correct format." The input field has a red border.
- Password***: The error message is "Password is required and must be at least 6 characters long." The input field has a red border.

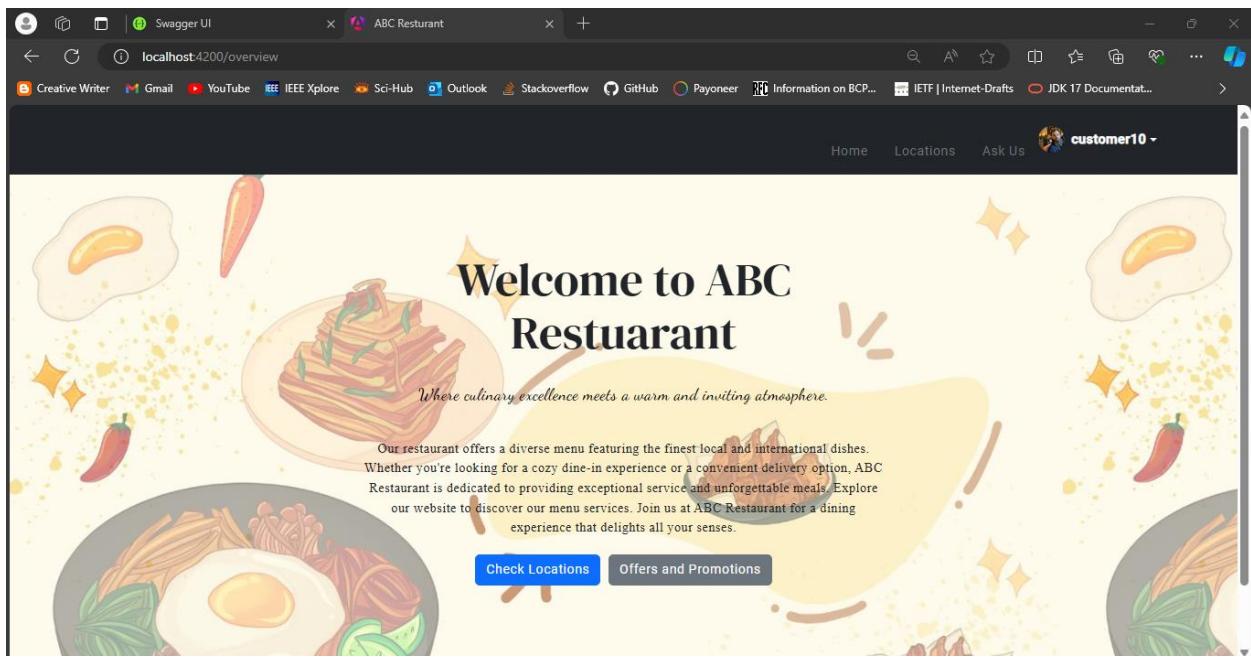
A "Sign in" button is located at the bottom left of the form area.

The screenshot shows a registration page for 'ABC Restaurant' on a web browser. The URL is 'localhost:4200/register'. The page has a dark header with 'Home', 'Locations', 'Login', and 'Register' links. A central modal window titled 'Register' contains the following form fields:

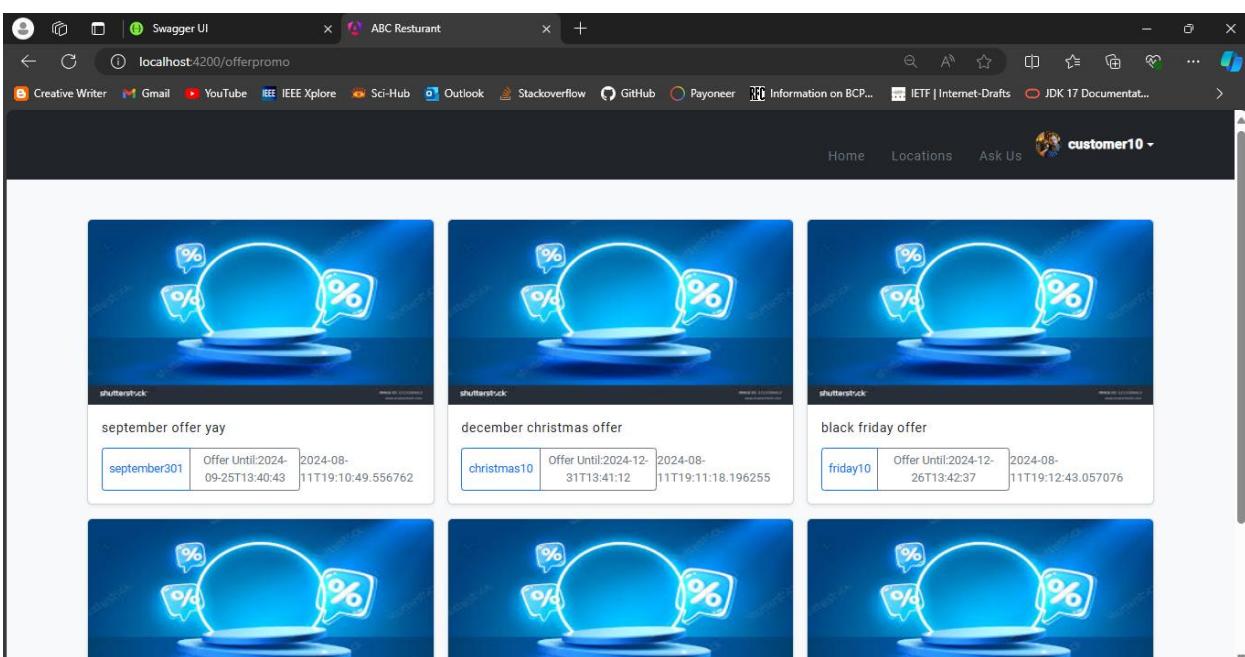
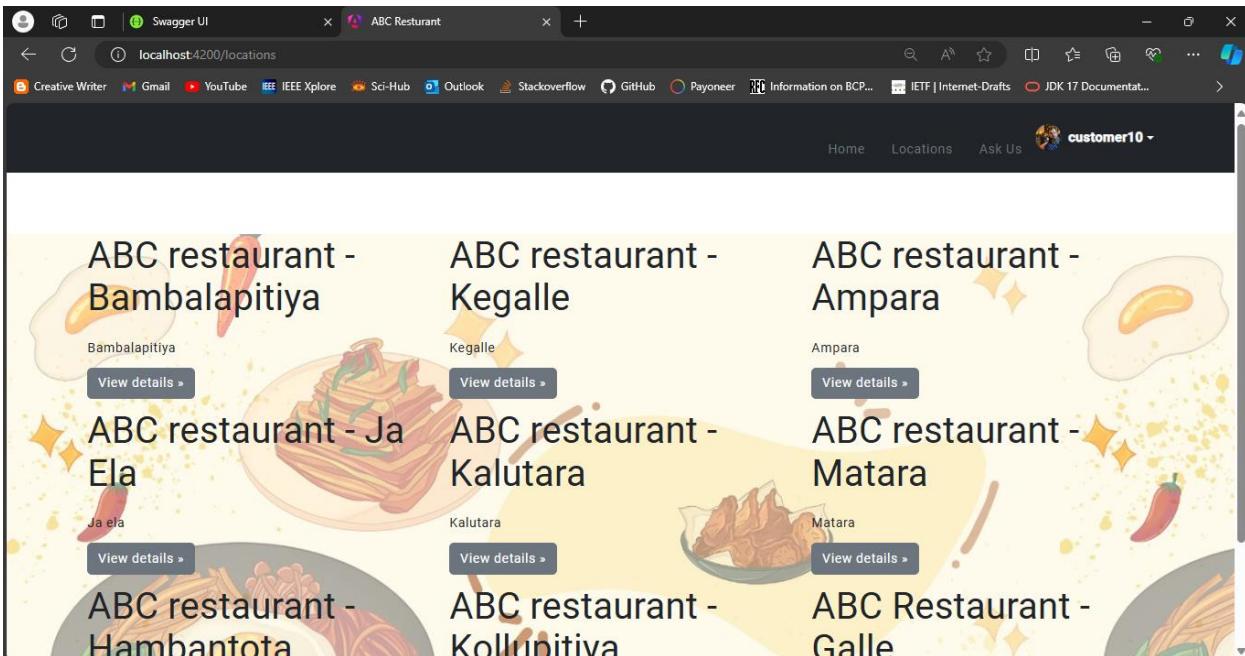
- Name*: customer10
- Telephone*: 5213649563
- Email*: customer10@gmail.com
- Password*: customer10

Below the form is a 'Sign In' button.

Upon registering, customer will be directed to home page

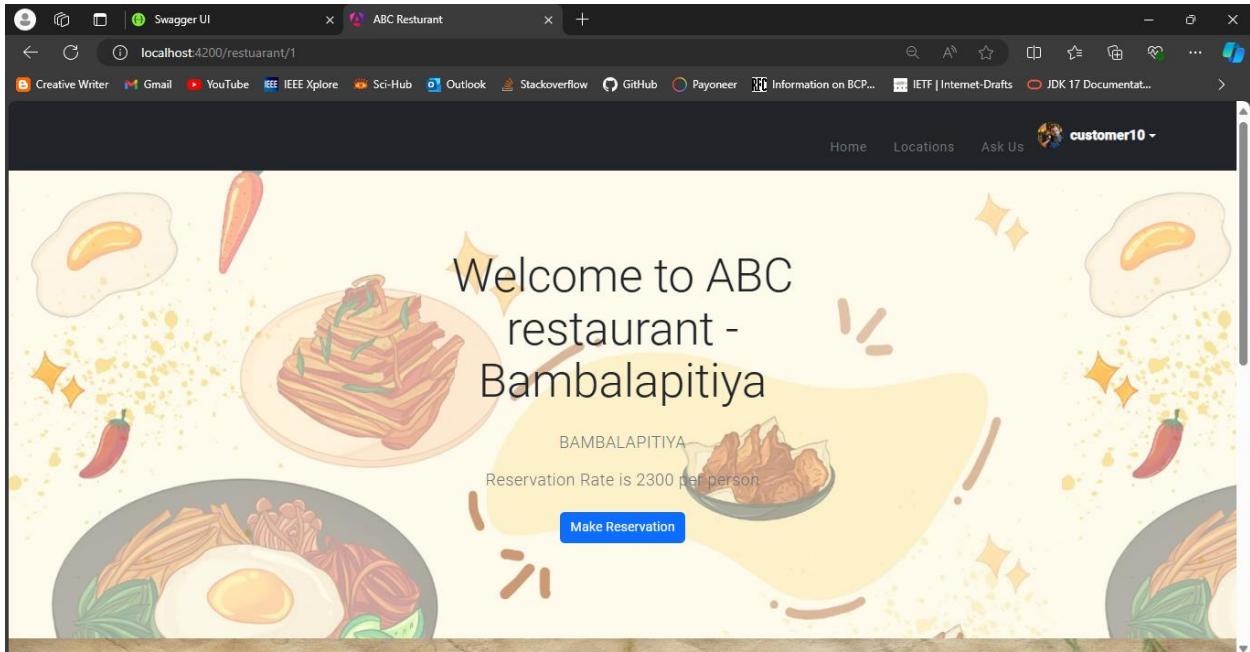


They can view locations and check offers and promotions



Make reservation

Customers can select a restaurant from locations and click on view details. Then they will be directed to the welcome page of the particular restaurant

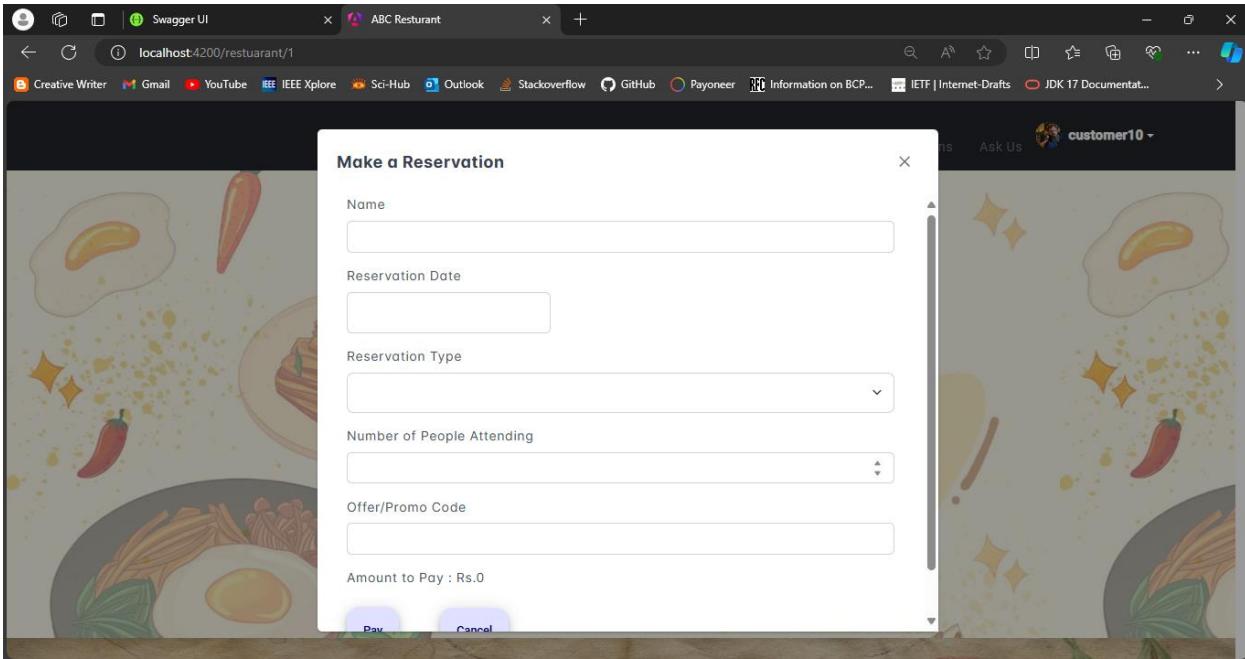


Customers can browse through services offered by restaurant.

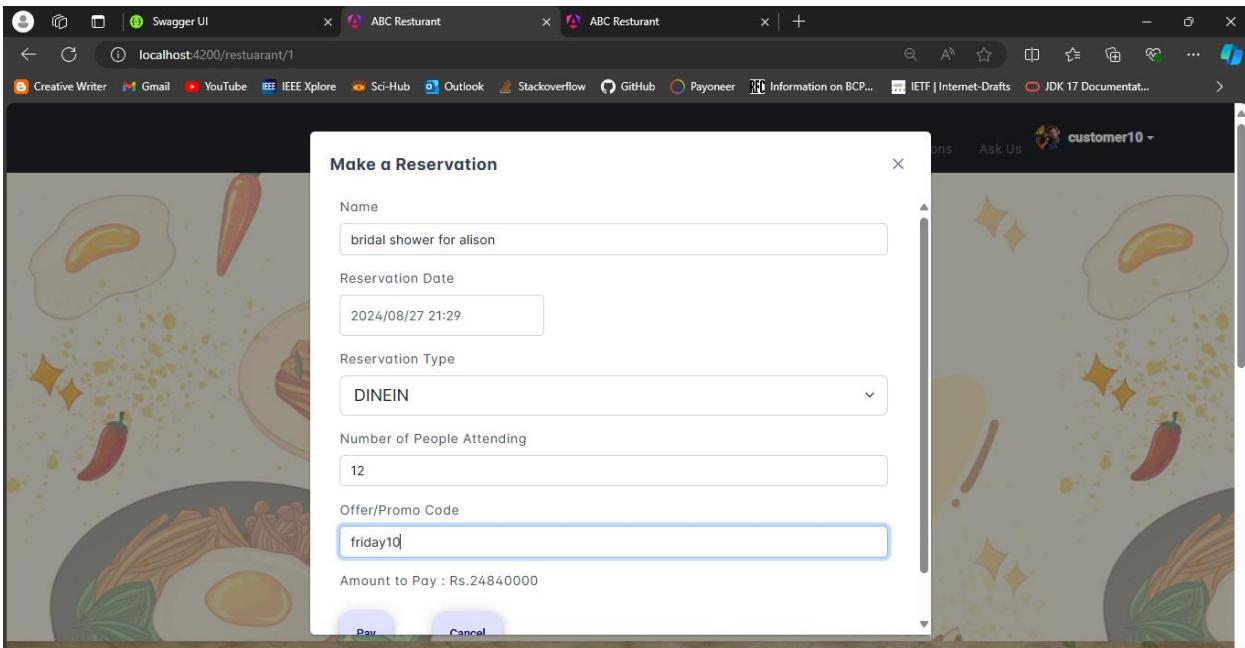
A screenshot of a web browser showing the service offerings page for ABC Restaurant. The title bar says "ABC Restaurant". The URL bar shows "localhost:4200/restaurant/1". The background features a large image of red flowers. The main heading is "Service offered by the Restuarant". Below it, the text "We go above and beyond for you." is followed by a search bar with "Search services..." placeholder text. A list of service types is shown in a dropdown menu:

- I. Table service
- II. Buffet service
- III. Birthday parties
- IV. Surprise parties
- V. Self service

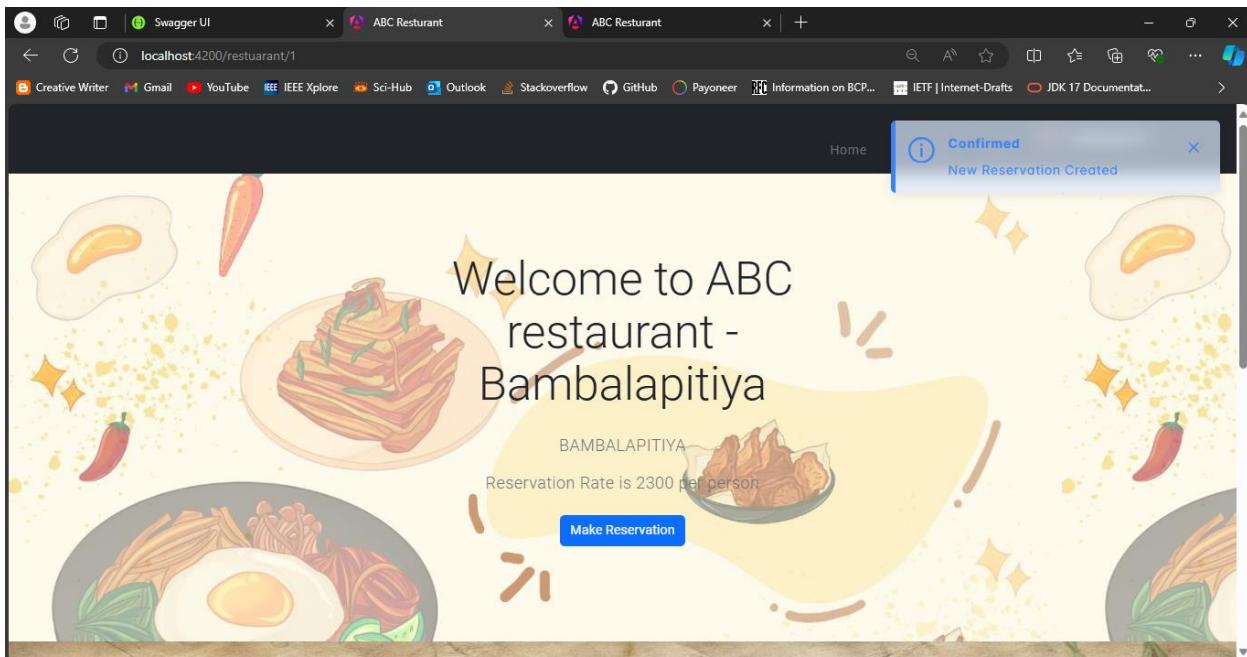
Once clicked on make reservation, this pop up appears.



When filling the form, if the customer has a promo code, they can enter it to receive a discount.



Upon making the reservation, the confirmation notification appears



Use of Design Patterns

MVC design pattern

In here, we send data into every view through a controller, and those data are bound into a model and shown in view. One controller can provide multiple views and in one view more than one model is sometimes used.

The model here shown is ChatRoom

```

29
30 // GET: api/<ChatController>
31 [HttpGet]
32 public async Task<List<ChatRoom>> Get()
33 {
34     return await _context.ChatRooms.ToListAsync();
35 }
36 [Route("GetChats")]
37 [HttpGet]
38 public async Task<List<CustomerResponse>> GetChatsById(int Id)
39 {
40     return await _context.CustomerResponses.Where(x=>x.ChatRoomId==Id).ToListAsync();
41 }
42
43 // GET api/<ChatController>/5
44 [HttpGet("{id}")]
45 public async Task<IActionResult> Get(int id)
46 {

```

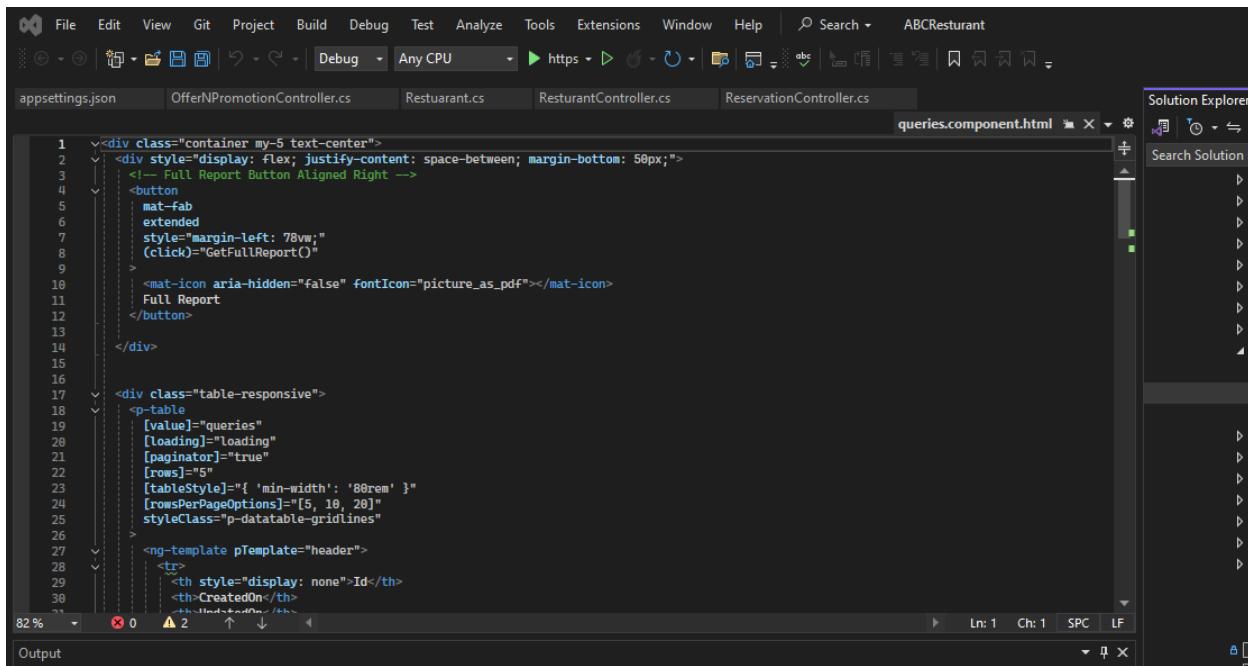
Then we bind the model ChatRoom to queries.component.ts component in the front-end

```

1 import { CommonModule } from '@angular/common';
2 import { ChangeDetectionStrategy, Component } from '@angular/core';
3 import {
4     ReactiveFormsModule,
5     FormGroup,
6     FormBuilder,
7     Validators,
8 } from '@angular/forms';
9 import { MatButtonModule } from '@angular/material/button';
10 import { MatDividerModule } from '@angular/material/divider';
11 import { MatIconModule } from '@angular/material/icon';
12 import { ConfirmationService, MessageService } from 'primeng/api';
13 import { ButtonModule } from 'primeng/button';
14 import { ConfirmDialogModule } from 'primeng/confirmdialog';
15 import { DialogModule } from 'primeng/dialog';
16 import { InputTextModule } from 'primeng/inputtext';
17 import { ProgressBarModule } from 'primeng/progressbar';
18 import { SlideMenuModule } from 'primeng/slidemenu';
19 import { TableModule } from 'primeng/table';
20 import { ToastModule } from 'primeng/toast';
21 import { QueryService } from './services/query.service';
22 import { ChatRoom, Chat, CustomerResponseType } from './models/Query';
23 import { UserService } from './services/users.service';
24 import { User } from './models/user';
25 import { UserActivityService } from './services/useractivity.service';
26 import { UserActivity } from './models/UserActivity';
27
28 @Component({
29     selector: 'app-queries',

```

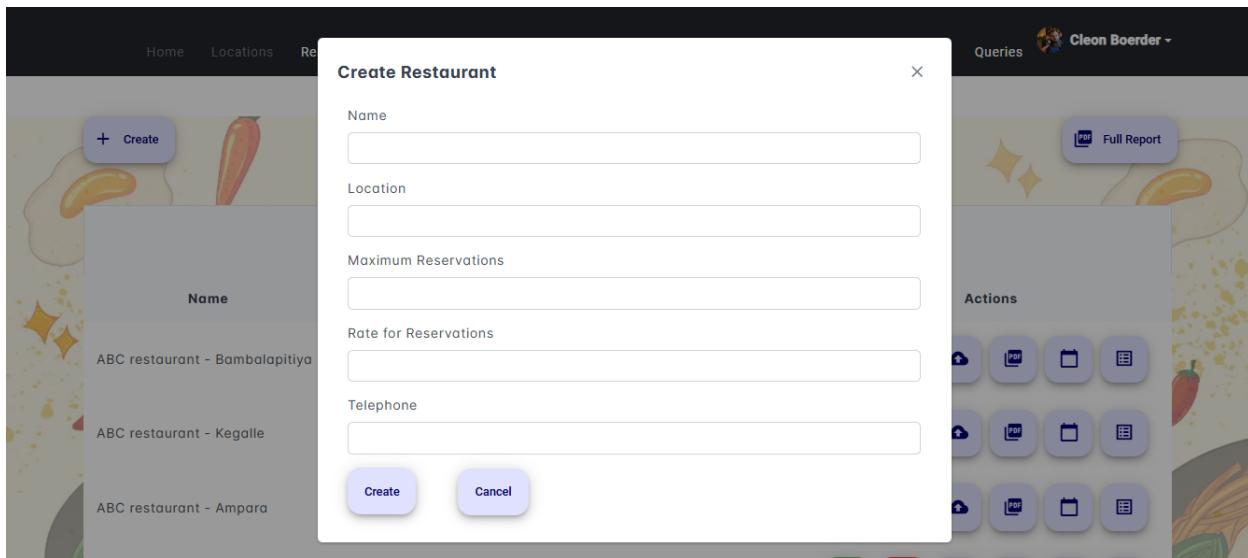
And then data that is bound are shown through the queries.component.html which acts as view.



```
1 <div class="container my-5 text-center">
2   <div style="display: flex; justify-content: space-between; margin-bottom: 50px;">
3     <!-- Full Report Button Aligned Right -->
4     <button mat-fab extended style="margin-left: 78vw;" (click)="GetFullReport()">
5       <mat-icon aria-hidden="false" fontIcon="picture_as_pdf"></mat-icon>
6       Full Report
7     </button>
8   </div>
9
10  <div class="table-responsive">
11    <p-table [value]="queries"
12      ["loading"]="loading"
13      ["paginator"]="true"
14      ["rows"]="5"
15      ["tableStyle"]="{ 'min-width': '88rem' }"
16      ["rowsPerPageOptions"]="[5, 10, 25]"
17      ["styleClass"]="p-dataTable-gridlines"
18    >
19      <ng-template pTemplate="header">
20        <tr>
21          <th style="display: none;>Id</th>
22          <th>Created On</th>
23          <th>Actions</th>
24        </tr>
25      </ng-template>
26      <tr>
27        <td><img alt="Restaurant logo" /></td>
28        <td>ABC restaurant - Bambalapitiya</td>
29        <td><button mat-fab (click)="Edit($event, item)"></button><button mat-fab (click)="Delete($event, item)"></button><button mat-fab (click)="Print($event, item)"></button></td>
30      </tr>
31      <tr>
32        <td><img alt="Restaurant logo" /></td>
33        <td>ABC restaurant - Kegalle</td>
34        <td><button mat-fab (click)="Edit($event, item)"></button><button mat-fab (click)="Delete($event, item)"></button><button mat-fab (click)="Print($event, item)"></button></td>
35      </tr>
36      <tr>
37        <td><img alt="Restaurant logo" /></td>
38        <td>ABC restaurant - Ampara</td>
39        <td><button mat-fab (click)="Edit($event, item)"></button><button mat-fab (click)="Delete($event, item)"></button><button mat-fab (click)="Print($event, item)"></button></td>
40      </tr>
41    </p-table>
42  </div>
43
```

Observer design pattern

As an example, in this project for every button you click, an event is bound to it. As an example for that when you click the create restaurant button, a pop up appears.



This works with the openForm function which works with both create and edit options in this case.

```

136     showDeleteConfirm(id: number) {
137       this.visible = true;
138     }
139   }
140   openForm(mode: 'create' | 'edit', restaurant?: any): void {
141     this.formMode = mode;
142     if (mode === 'edit' && restaurant) {
143       this.restaurantFormEdit.patchValue(restaurant);
144     } else {
145       this.restaurantFormEdit.reset();
146       this.restaurantFormNew.reset();
147     }
148     this.showForm = true;
149   }
150   const newUserActivity: UserActivity = {
151     activity: 'Open restaurant form in ' + mode,
152     userId: this.userService.getCurrentUser().id,
153   };

```

Following is the create button. With the openForm function, create parameter is passed here.

```

1 <div class="container my-5 text-center">
2   <button
3     mat-fab
4     extended
5     style="margin-bottom: 50px; margin-right: 35vw"
6     (click)="openForm('create')"
7   >
8     <mat-icon aria-hidden="false" fontIcon="add"></mat-icon>
9     Create
10    </button>
11
12   <button
13     mat-fab
14     extended
15     style="margin-bottom: 50px; margin-right: 35vw"
16     (click)="GetFullReport()"
17   >
18     <mat-icon aria-hidden="false" fontIcon="picture_as_pdf"></mat-icon>
19     Full Report
20   </button>
21 </div>

```

Once the button is clicked, it creates an event listener to respond with a pop up. When it gets true the pop up appears.

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help.
- Process:** [19096] ABCRestaurant.exe
- Editor:** restaurants.component.html (active tab), restaurants.component.ts, Admin.cs*, Staff.cs, User.cs, appsettings.json, ChatController.cs.
- Code:**

```

97     </div>
98     </td>
99     </tr>
100    </ng-template>
101   ; </p-table>
102   </div>
103 </div>
104
105  <!-- This is the create>Edit dialog -->
106 <p-dialog
107   header="{{ formMode === 'create' ? 'Create Restaurant' : 'Edit Restaurant' }}"
108   [(visible)]="showForm"
109   [modal]="true"
110   [breakpoints]={{ '1199px': '75vw', '575px': '90vw' }}
111   [style]={{ width: '50vw' }}
112   [draggable]="false"
113   [resizable]="false"
114 >
115   <div class="popup-form container mv-10">

```
- Bottom Status Bar:** Ln: 6 Ch: 33 SPC CRLF
- Sidebar:** Diagnose, Events, Processes, Summary, Events, Memory.
- Bottom Panels:** Autos, Call Stack.

Then it determines whether it's an edit or a create. The following image shows the same function for edit with the entity restaurant

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help.
- Process:** [19096] ABCRestaurant.exe
- Editor:** restaurants.component.html (active tab), restaurants.component.ts, Admin.cs*, Staff.cs, User.cs, appsettings.json, ChatController.cs.
- Code:**

```

46   <td>{{ restaurant.location }}</td>
47   <td>{{ restaurant.telephone }}</td>
48   <td>{{ restaurant.createdOn }}</td>
49   <td style="text-align: center">
50
51     <div class="actions-column">
52       <button
53         mat-fab
54         color="success"
55         (click)="openForm('edit', restaurant)">
56           <mat-icon aria-hidden="false" fontIcon="edit"></mat-icon>
57         </button>
58
59       <button
60         mat-fab
61         color="failure"
62         (click)="deleteConfirm(restaurant.id)">
63           <mat-icon></mat-icon>

```
- Bottom Status Bar:** Ln: 94 Ch: 42 SPC CRLF
- Sidebar:** Diagnose, Events, Processes, Summary, Events, Memory.
- Bottom Panels:** Autos, Call Stack.

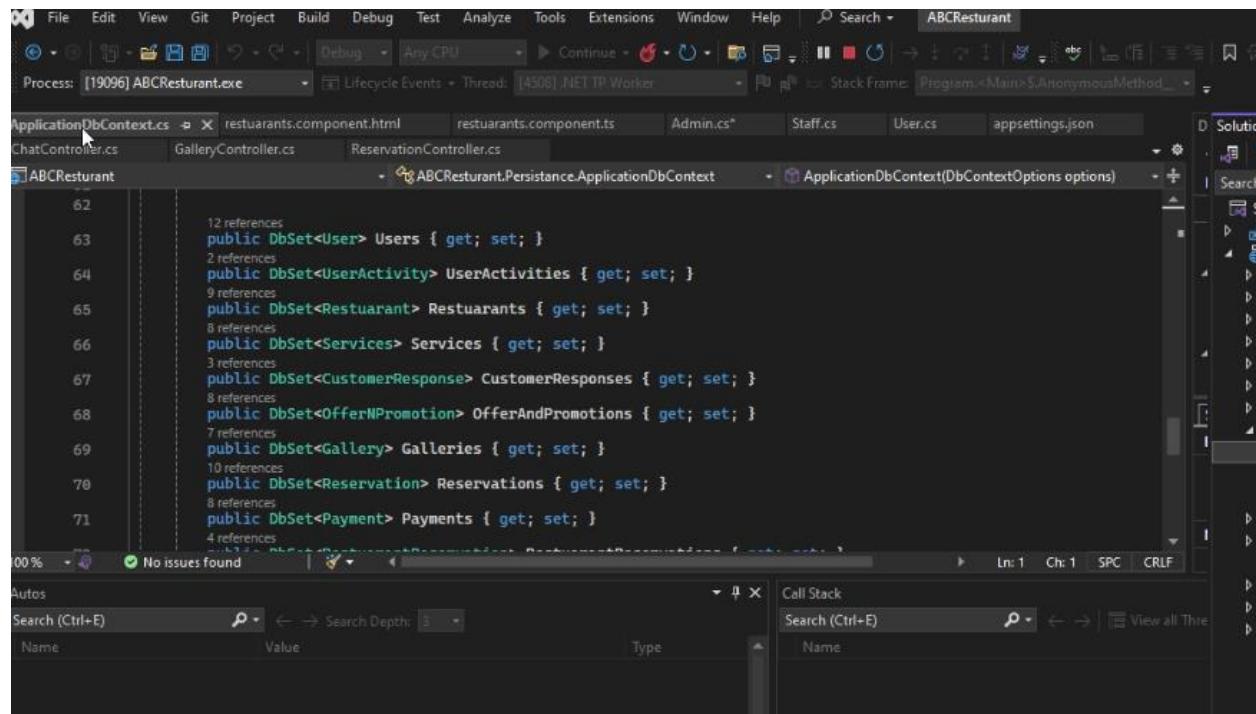
After that it determines whether it's an edit, create etc or which entity it is.

```
136     showDeleteConfirm(id: number) {
137         this.visible = true;
138     }
139
140     openForm(mode: 'create' | 'edit', restaurant?: any): void {
141         this.formMode = mode;
142         if (mode === 'edit' && restaurant) {
143             this.restaurantFormEdit.patchValue(restaurant);
144         } else {
145             this.restaurantFormEdit.reset();
146             this.restaurantFormNew.reset();
147         }
148         this.showForm = true;
149
150         const newUserActivity: UserActivity = {
151             activity: 'Open restuarant form in ' + mode,
152             userId: this.UsersService.getCurrentUser().id,
153         };
154     }

```

Repository pattern

ApplicationDbContext.cs is used for the repository pattern here. Through the repository pattern we do the changes in the database such as update, drop etc. through this class. It handles the functionality of the repository pattern.



The screenshot shows the Visual Studio IDE interface with the following details:

- Menu Bar:** File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help.
- Toolbar:** Standard toolbar with icons for file operations.
- Status Bar:** Shows "Process: (19096) ABCRestaurant.exe", "Lifecycle Events", "Thread: [4508] .NET IP Worker", and "Stack Frame: Program.<Main>\$AnonymousMethod_".
- Solution Explorer:** Shows files like ChatController.cs, GalleryController.cs, ReservationController.cs, Admin.cs, Staff.cs, User.cs, and appsettings.json.
- Code Editor:** Displays the `ApplicationDbContext.cs` file with the following code:

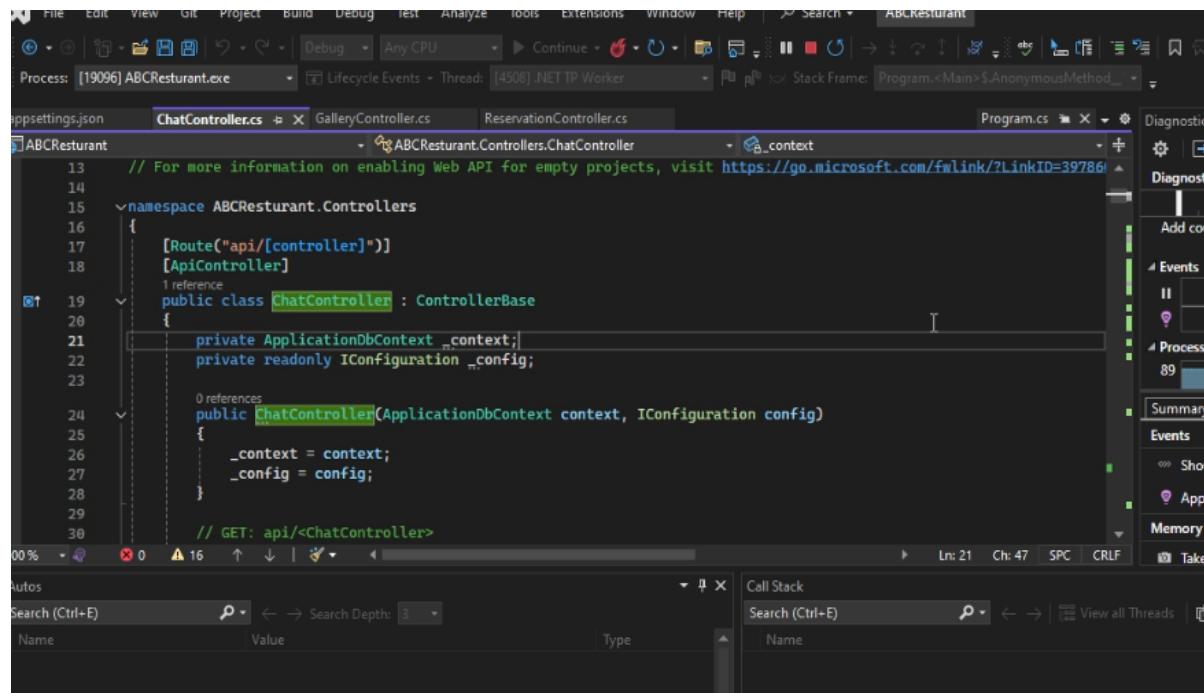
```
62      12 references
63      public DbSet<User> Users { get; set; }
64      2 references
65      public DbSet<UserActivity> UserActivities { get; set; }
66      9 references
67      public DbSet<Restuarant> Restuarants { get; set; }
68      8 references
69      public DbSet<Services> Services { get; set; }
70      3 references
71      public DbSet<CustomerResponse> CustomerResponses { get; set; }
```

Below the code editor, it says "No issues found".
- Toolbars:** Autos, Call Stack.
- Search:** Search (Ctrl+E) and Call Stack search fields.

Use of OOP Concepts

Inheritance

Inheritance OOP concept is used where ControllerBase is the parent and the ChatController has what was inherited from the parent class.



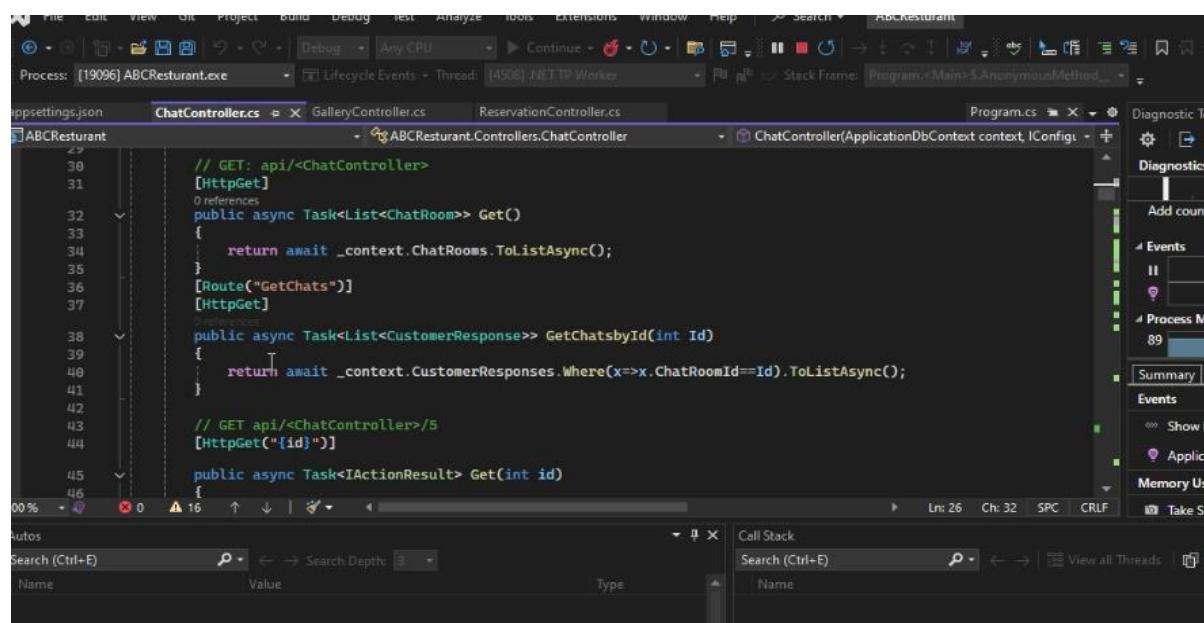
The screenshot shows the Visual Studio IDE interface with the ChatController.cs file open. The code defines a ChatController class that inherits from ControllerBase. It includes constructor injection for ApplicationDbContext and IConfiguration, and several GET routes defined using [HttpGet] and [Route] attributes.

```
// For more information on enabling Web API for empty projects, visit https://go.microsoft.com/fwlink/?LinkId=39786
namespace ABCRestaurant.Controllers
{
    [Route("api/{controller}")]
    [ApiController]
    public class ChatController : ControllerBase
    {
        private ApplicationDbContext _context;
        private readonly IConfiguration _config;

        public ChatController(ApplicationDbContext context, IConfiguration config)
        {
            _context = context;
            _config = config;
        }

        // GET: api/<ChatController>
    }
}
```

The ChatController can access HTTP get methods, route etc.



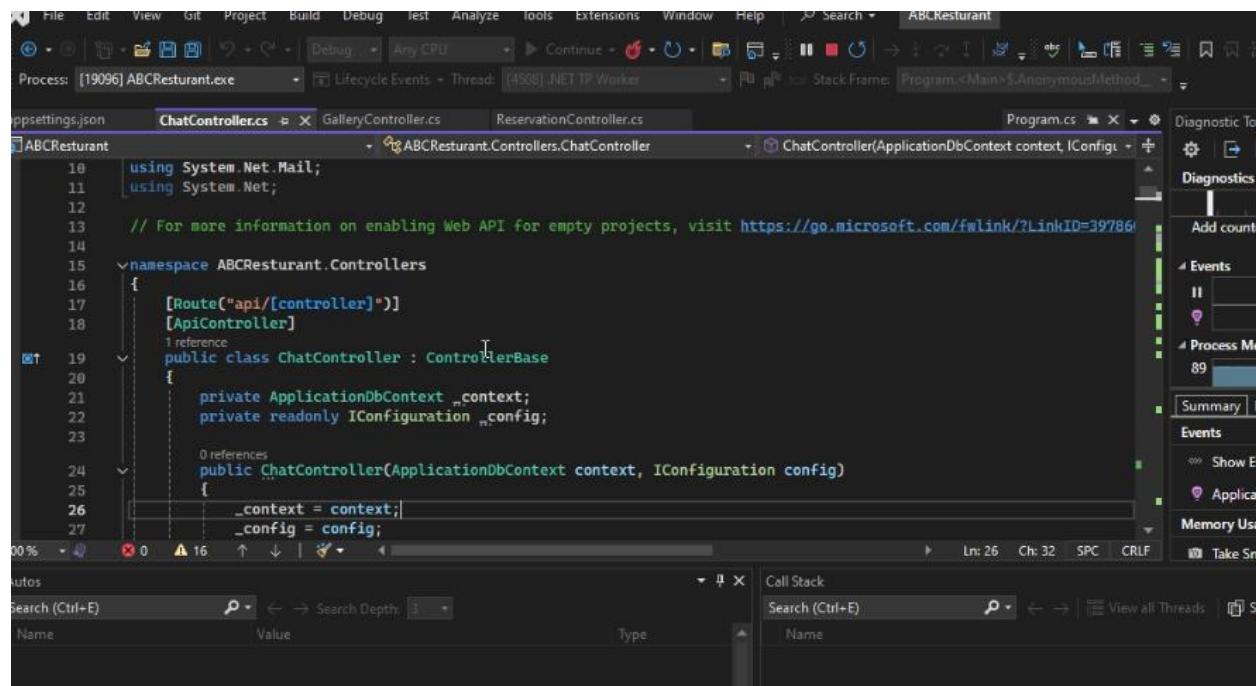
The screenshot shows the Visual Studio IDE interface with the ChatController.cs file open. The code now includes three additional GET routes: one for retrieving all ChatRooms, one for retrieving CustomerResponses by ChatRoomId, and one for retrieving a specific CustomerResponse by its ID.

```
// GET: api/<ChatController>
[HttpGet]
public async Task<List<ChatRoom>> Get()
{
    return await _context.ChatRooms.ToListAsync();
}

[Route("GetChats")]
[HttpGet]
public async Task<List<CustomerResponse>> GetChatsById(int id)
{
    return await _context.CustomerResponses.Where(x=>x.ChatRoomId==id).ToListAsync();
}

// GET api/<ChatController>/5
[HttpGet("{id}")]
public async Task<IActionResult> Get(int id)
{
}
```

The ChatController can create APIs because of the ControllerBase

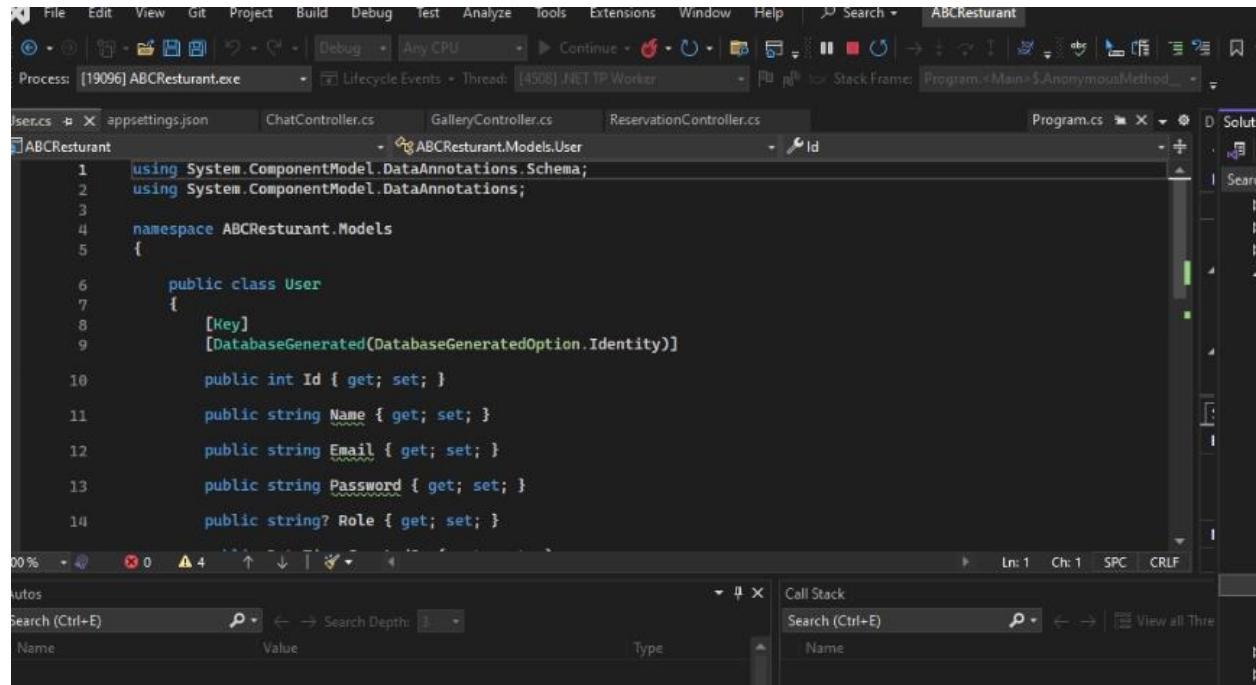


A screenshot of the Visual Studio IDE showing the ChatController.cs file. The code defines a class ChatController that inherits from ControllerBase. It includes constructor injection for ApplicationDbContext and IConfiguration. The controller has a single action method named Get that returns a string "Hello API". The code editor shows syntax highlighting and intellisense. The status bar at the bottom indicates the file is 100% complete.

```
10  using System.Net.Mail;
11  using System.Net;
12
13  // For more information on enabling Web API for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=39786
14
15  namespace ABCRestaurant.Controllers
16  {
17      [Route("api/[controller]")]
18      [ApiController]
19      public class ChatController : ControllerBase
20      {
21          private ApplicationDbContext _context;
22          private readonly IConfiguration _config;
23
24          public ChatController(ApplicationDbContext context, IConfiguration config)
25          {
26              _context = context;
27              _config = config;
28          }
29
30          [HttpGet]
31          public string Get()
32          {
33              return "Hello API";
34          }
35      }
36  }
```

This works for the user roles as well. User is the parent where admin, staff and customer inherit data from user. So admin, staff and customer become the children of the parent user.

Following is the user



A screenshot of the Visual Studio IDE showing the User.cs file. The code defines a class User with properties Id, Name, Email, Password, and Role. The Id property is annotated with [Key] and [DatabaseGenerated(DatabaseGeneratedOption.Identity)]. The code editor shows syntax highlighting and intellisense. The status bar at the bottom indicates the file is 100% complete.

```
1  using System.ComponentModel.DataAnnotations.Schema;
2  using System.ComponentModel.DataAnnotations;
3
4  namespace ABCRestaurant.Models
5  {
6      public class User
7      {
8          [Key]
9          [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
10         public int Id { get; set; }
11
12         public string Name { get; set; }
13
14         public string Email { get; set; }
15
16         public string Password { get; set; }
17
18         public string? Role { get; set; }
19     }
20 }
```

Staff

A screenshot of the Visual Studio IDE interface. The title bar says "ABCRestaurant". The menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and Search. The toolbar has icons for file operations like Open, Save, and Print. The status bar at the bottom shows "100%". The main code editor window displays the "Staff.cs" file:

```
namespace ABCRestaurant.Models
{
    public class Staff : User
    {
        public int WorkingRestaurantID { get; set; }
        public Restaurant WorkingRestaurant { get; set; }
    }
}
```

The Solution Explorer on the right shows files like "Program.cs", "WorkingRestaurantID.cs", and "ABCRestaurant.sln". The Task List and Output windows are also visible.

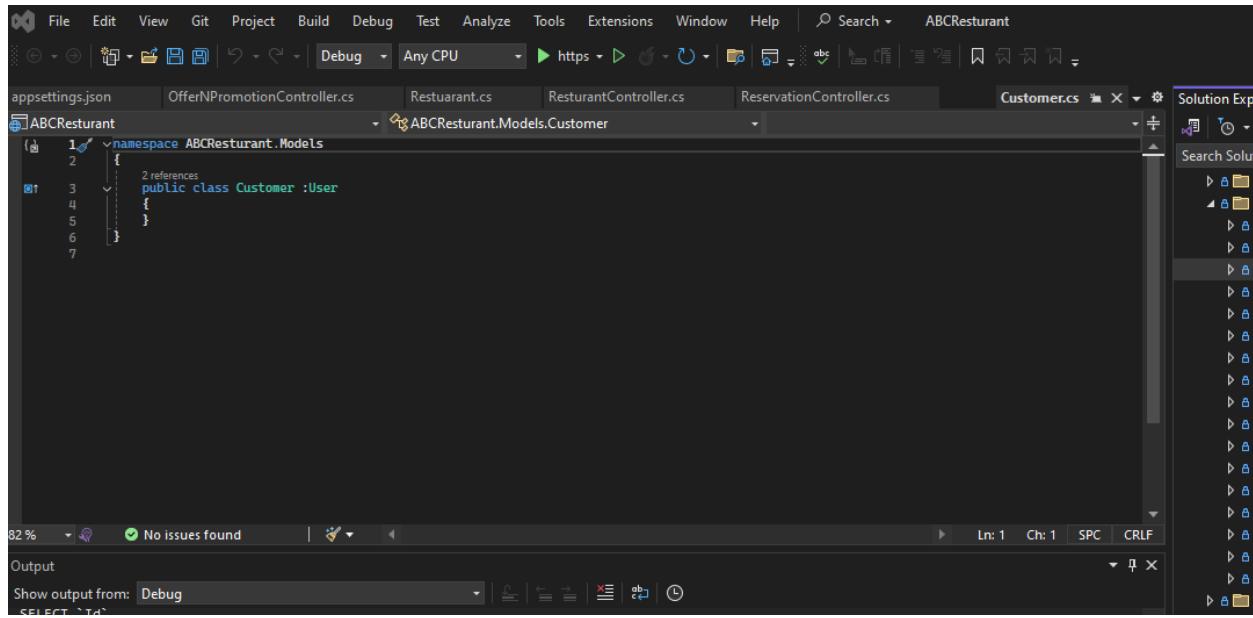
Admin

A screenshot of the Visual Studio IDE interface. The title bar says "ABCRestaurant". The menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and Search. The toolbar has icons for file operations like Open, Save, and Print. The status bar at the bottom shows "100%". The main code editor window displays the "Admin.cs" file:

```
namespace ABCRestaurant.Models
{
    public class Admin : User
    {
        public Admin()
        {
            Role = "Admin";
        }
    }
}
```

The Solution Explorer on the right shows files like "Program.cs", "WorkingRestaurantID.cs", and "ABCRestaurant.sln". The Task List and Output windows are also visible.

Customer



The screenshot shows the Microsoft Visual Studio IDE interface. The title bar reads "ABCRestaurant". The menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help. The toolbar has icons for file operations like Open, Save, and Print. The status bar at the bottom shows "82 %", "No issues found", "Ln: 1 Ch: 1 SPC CRLF", and "Output". The main code editor window displays the "Customer.cs" file with the following code:

```
namespace ABCRestaurant.Models
{
    public class Customer : User
    {
    }
}
```

The Solution Explorer on the right shows a tree view of the project structure, including files like appsettings.json, OfferNPromotionController.cs, Restaurant.cs, RestaurantController.cs, ReservationController.cs, and Customer.cs.

Polymorphism

For the same command many events are bound. As an example, when a certain button is clicked a pop up appears. The example is discussed further as follows.

When user clicks on create restaurant button a pop up appears. The openForm function works with create button which creates a listener for it and opens a pop up. The same function works with edit, delete etc. as well. Therefore, for the same function many events are bound.

```
136     showDeleteConfirm(id: number) {
137         this.visible = true;
138     }
139 }
140 openForm(mode: 'create' | 'edit', restaurant?: any): void {
141     this.formMode = mode;
142     if (mode === 'edit' && restaurant) {
143         this.restaurantFormEdit.patchValue(restaurant);
144     } else {
145         this.restaurantFormEdit.reset();
146         this.restaurantFormNew.reset();
147     }
148     this.showForm = true;
149 }
150 const newUserActivity: UserActivity = {
151     activity: 'Open restaurant form in ' + mode,
152     userId: this.UsersService.getCurrentUser().id,
153 };
this.UserService.addUserActivity(newUserActivity);
```

No issues found

```
<div class="container my-5 text-center">
    <button
        mat-fab
        extended
        style="margin-bottom: 50px; margin-right: 35vw"
        (click)="openForm('create')">
        <mat-icon aria-hidden="false" fontIcon="add"></mat-icon>
        Create
    </button>
    <button
        mat-fab
        extended
        style="margin-bottom: 50px; margin-right: 35vw"
        (click)="GetFullReport()">
        <mat-icon aria-hidden="false" fontIcon="picture_as_pdf"></mat-icon>
        Full Report
    </button>
```

It determines whether the button click was create, edit or delete etc. and opens the pop up when model becomes true here.

```

97     </div>
98     </td>
99   </tr>
100  </ng-template>
101  </p-table>
102  </div>
103</div>
104</div>
105
106  <!-- This is the create>Edit dialog -->
107  <p-dialog
108    header="{{ formMode === 'create' ? 'Create Restaurant' : 'Edit Restaurant' }}"
109    [(visible)]="showForm"
110    [modal]="true"
111    [breakpoints]={{ '1199px': '75vw', '575px': '90vw' }}
112    [style]={{ width: '50vw' }}
113    [draggable]="false"
114    [resizable]="false"
115  >
116    <div class="popup-form container mv-1A">

```

Database

MySQL was chosen as the database and MySQL workbench was used as the visual database design tool.

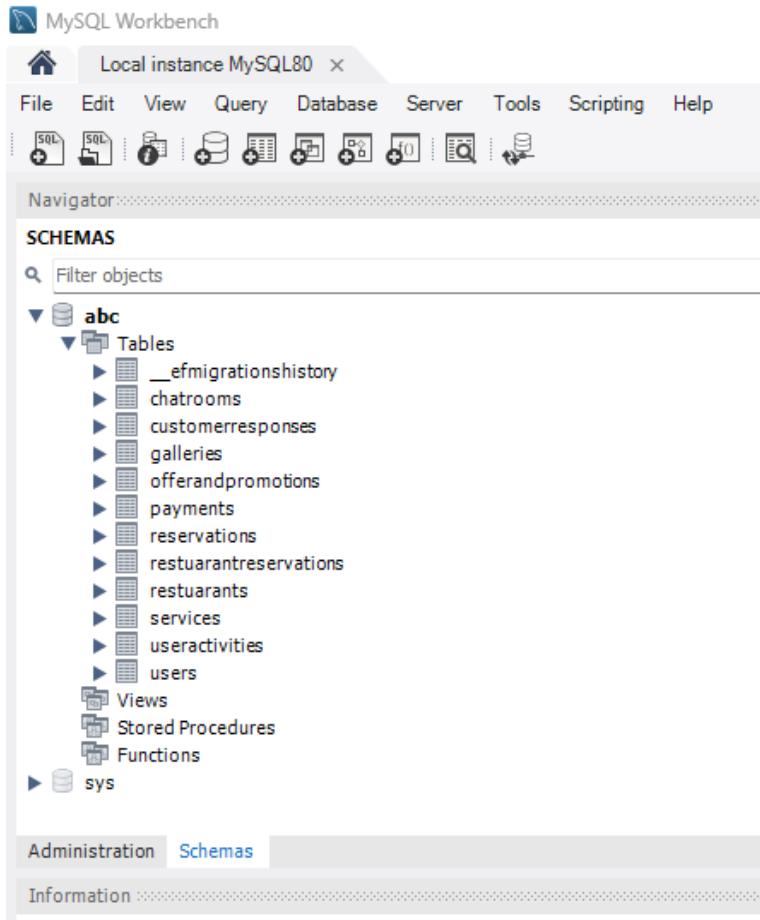
The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The `users` schema is selected.
- Result Grid:** A query `SELECT * FROM abc.users;` is run, displaying the following data:

ID	Name	Email	Password	Role	CreatedOn	UpdatedOn	Telephone	Restaurant
1	Cleon Boeder	cboeder0@time.com	zFLUkV41q4	Admin	2024-01-13 00:00:00.000000	NULL	383-8490799	NULL
2	Damara Hellin	dhellin1@usa.gov	cbj0xEGG7r	Staff	2023-11-27 00:00:00.000000	2024-08-12 13:21:58.336704	88138491515	NULL
4	Donavon Christopersen	dchristopersen3@scindencedirect.com	iWFAC74bp	Staff	2024-05-11 00:00:00.000000	2024-08-12 13:19:22.192093	2169890578	NULL
5	Warner Masse	wmasse4@rps.gov	HUp474#uo	Staff	2023-11-22 00:00:00.000000	2024-08-26 16:15:30.609782	3885599160	NULL
7	Gal Lockyear	glodyear6@google.cn	SnSlpx4k	Staff	2024-03-12 00:00:00.000000	NULL	120-4224727	NULL
8	Kenneth Donald	kdonald7@cornell.edu	53TauRN20m	Admin	2023-10-26 00:00:00.000000	NULL	630-6093047	NULL
9	Syd Ocheltree	socheltree8@constantcontact.com	D8o1ob890	Staff	2024-02-21 00:00:00.000000	NULL	500-9399118	NULL

- Action Output:** The log shows the following actions and their results:

#	Time	Action	Message	Duration / Fetch
6	21:39:37	SELECT * FROM abc.offerandpromotions LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
7	21:39:38	SELECT * FROM abc.restaurantreservations LIMIT 0, 1000	15 row(s) returned	0.000 sec / 0.000 sec
8	21:39:40	SELECT * FROM abc.services LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec
9	21:39:42	SELECT * FROM abc.restaurantreservations LIMIT 0, 1000	15 row(s) returned	0.000 sec / 0.000 sec
10	21:39:48	SELECT * FROM abc.users LIMIT 0, 1000	22 row(s) returned	0.000 sec / 0.000 sec



In appsettings.json ConnectionStrings are included.

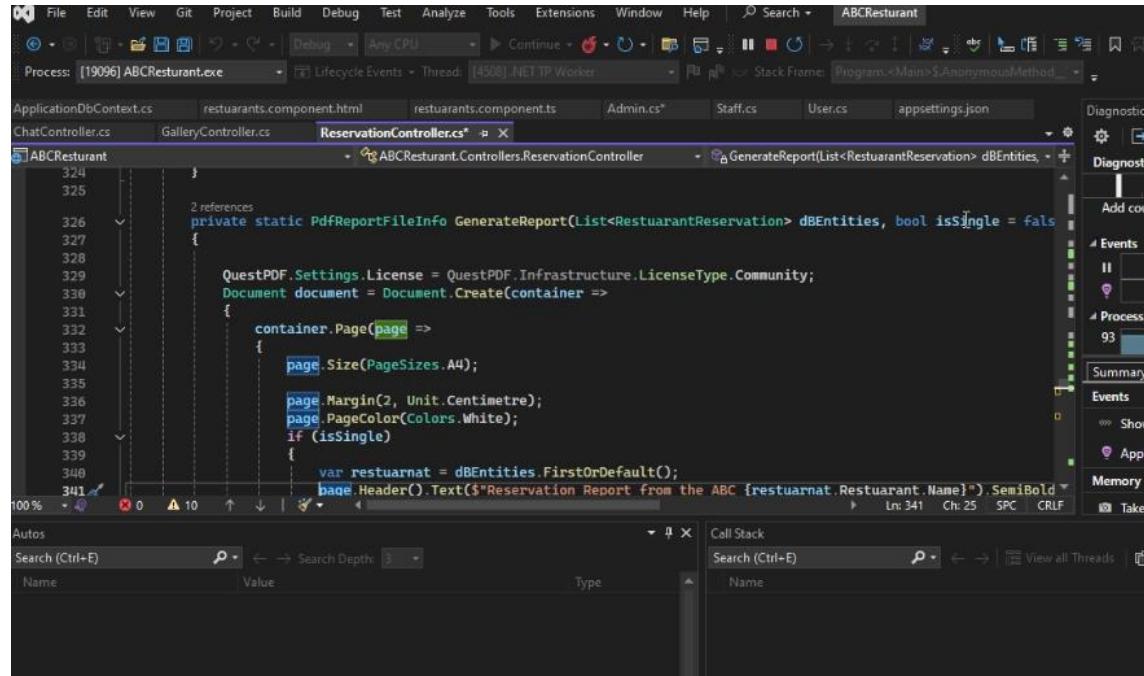
```

1  {
2
3    "EmailHost": "smtp.ethereal.email",
4    "EmailUsername": "michaela.lind69@ethereal.email",
5    "EmailPassword": "WzFFJhXbuKaTfDGU7",
6
7    "MailSettings": {
8      "Server": "Live.smtp.mailtrap.io",
9      "Port": "587",
10     "SenderName": "ABC",
11     "SenderEmail": "mailtrap@demomailtrap.com",
12     "UserName": "api",
13     "Password": "cd211c80e81660a322d6170829198bb5"
14   },
15
16   "ConnectionStrings": {
17     "DefaultConnectionMSQLNoCred": "Server=DEVESTATION\\MSSQLSERVER2017;Database=TutorialDB;Trusted_Connection=True;MultipleActiveResultSets=true",
18     "DefaultConnectionMySQL": "server=localhost; port=3306; database=ABC; user=root; password=",
19   },
20
21   "Logging": {
22     "LogLevel": {
23       "Default": "Information",
24       "Microsoft.AspNetCore": "Warning"
25     }
26   },
27   "AllowedHosts": "*"
28 }
  
```

Report Generation

QuestPDF library was used for report generation.

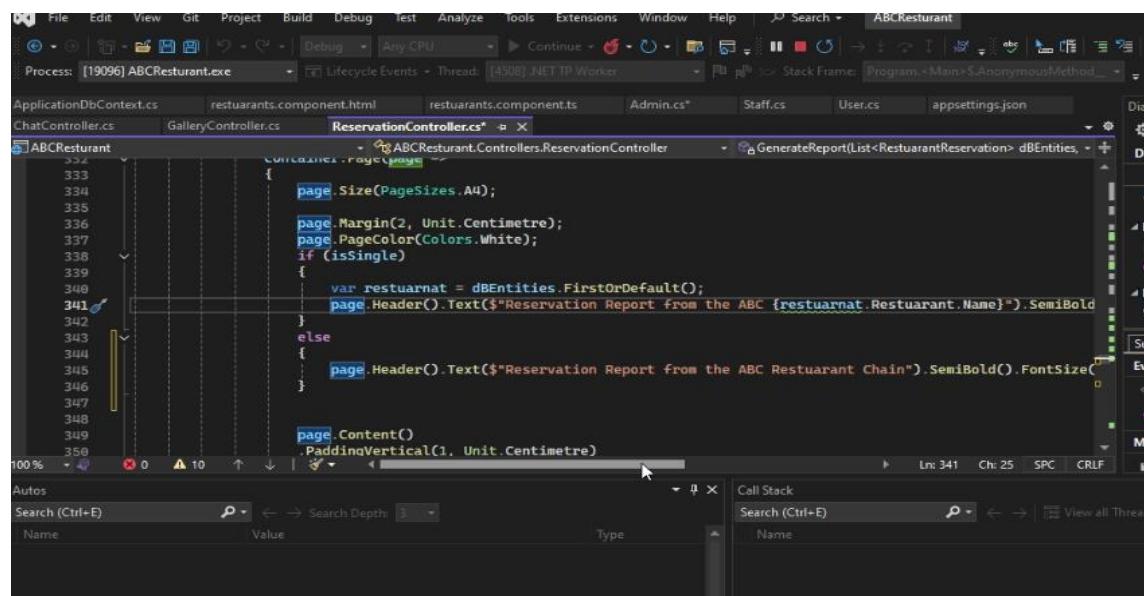
Following shows the function for it. First, it creates a document. Through the container in the following image, we define the page properties such as page size, margins, PageColor etc.



The screenshot shows the Visual Studio IDE with the code editor open. The file being edited is `ReservationController.cs`. The code defines a static method `GenerateReport` that takes a list of `RestaurantReservation` entities and a boolean `isSingle`. It sets up a `PdfReportFileInfo` object with a license and a `Document` object. The `Document` is created with a `container` that specifies a page size of A4, a margin of 2 units, and a white page color. If `isSingle` is true, it sets the header to "Reservation Report from the ABC [restuarant.Restaurant.Name]" in semi-bold font. The code editor includes standard VS tools like Autos and Call Stack at the bottom.

```
private static PdfReportFileInfo GenerateReport(List<RestaurantReservation> dBEntities, bool isSingle)
{
    QuestPDF.Settings.License = QuestPDF.Infrastructure.LicenseType.Community;
    Document document = Document.Create(container =>
    {
        container.Page[page =>
        {
            page.Size(PageSizes.A4);
            page.Margin(2, Unit.Centimetre);
            page.PageColor(Colors.White);
            if (isSingle)
            {
                var restuarant = dBEntities.FirstOrDefault();
                page.Header().Text($"Reservation Report from the ABC {restuarant.Restaurant.Name}").SemiBold();
            }
        }];
    });
}
```

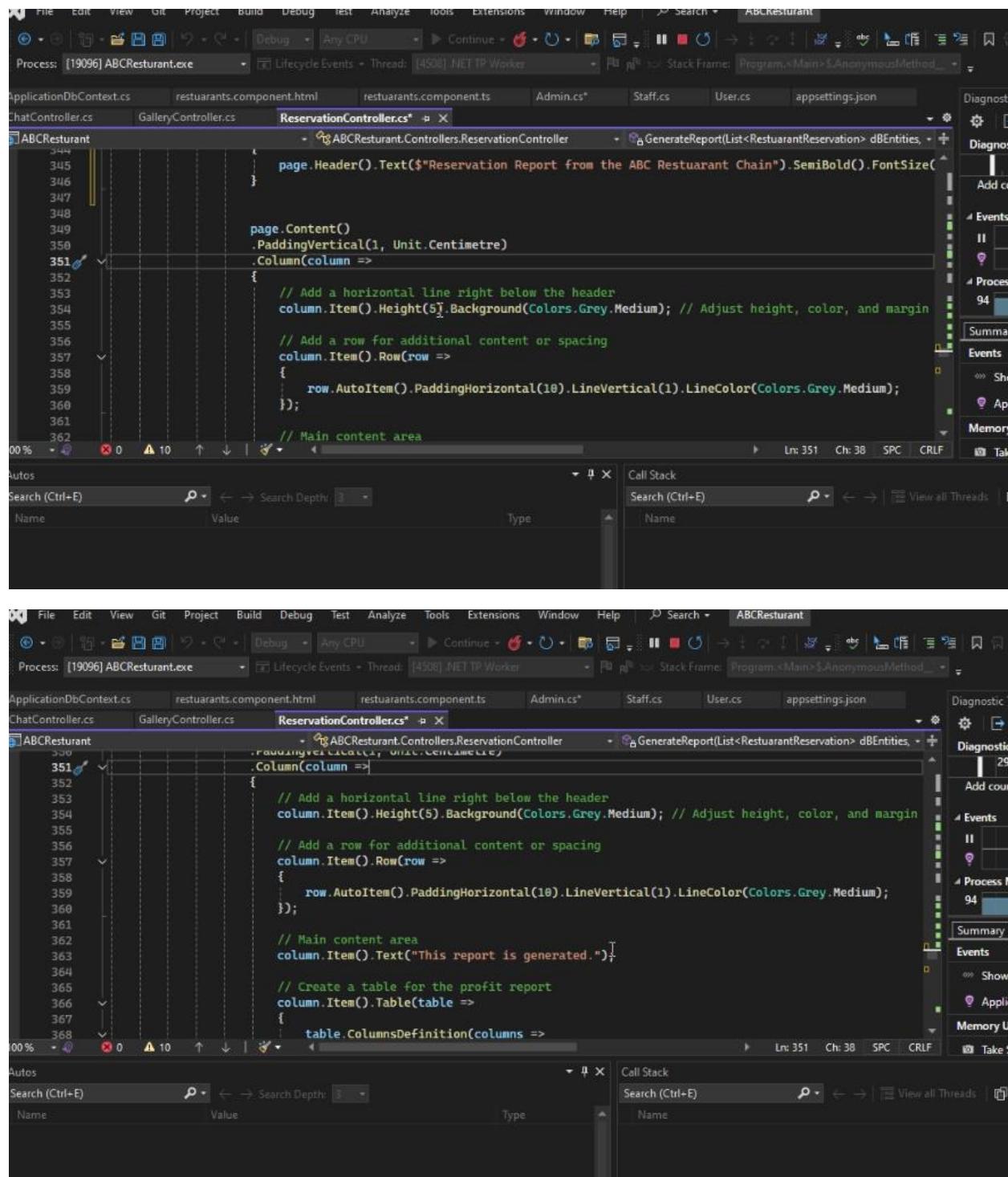
In the following image we can define the document header.



This screenshot shows the same `ReservationController.cs` file in Visual Studio. The code has been modified to include an else block in the `GenerateReport` method. The new code adds a header for a chain of restaurants if `isSingle` is false, otherwise it uses the individual restaurant name. The code editor interface remains consistent with the previous screenshot.

```
if (isSingle)
{
    var restuarant = dBEntities.FirstOrDefault();
    page.Header().Text($"Reservation Report from the ABC {restuarant.Restaurant.Name}").SemiBold();
}
else
{
    page.Header().Text($"Reservation Report from the ABC Restuarant Chain").SemiBold().FontSize(14);
}
```

page.Content() defines the appearance of the document.



```
ABCRestaurant
    344
    345
    346
    347
    348
    349
    350
    351
    352
    353
    354
    355
    356
    357
    358
    359
    360
    361
    362
    363
    364
    365
    366
    367
    368

    page.Header().Text($"Reservation Report from the ABC Restaurant Chain").SemiBold().FontSize(14);
}

page.Content()
.PaddingVertical(1, Unit.Centimetre)
.Column(column =>
{
    // Add a horizontal line right below the header
    column.Item().Height(5).Background(Colors.Grey.Medium); // Adjust height, color, and margin

    // Add a row for additional content or spacing
    column.Item().Row(row =>
    {
        row.AutoItem().PaddingHorizontal(10).LineVertical(1).LineColor(Colors.Grey.Medium);
    });

    // Main content area
    column.Item().Text("This report is generated.");
}

// Create a table for the profit report
column.Item().Table(table =>
{
    table.ColumnsDefinition(columns =>
```

The screenshot shows the Visual Studio IDE with the ABCRestaurant project open. The current file is ReservationController.cs. The code is defining a report structure:

```
    .PaddingVertical(2, Unit.Centimetre)
    .Column(column =>
    {
        // Add a horizontal line right below the header
        column.Item().Height(5).BackgroundColor(Colors.Grey.Medium); // Adjust height, color, and margin

        // Add a row for additional content or spacing
        column.Item().Row(row =>
        {
            row.AutoItem().PaddingHorizontal(10).LineVertical(1).LineColor(Colors.Grey.Medium);
        });

        // Main content area
        column.Item().Text("This report is generated.");
    });

    // Create a table for the profit report
    column.Item().Table(table =>
    {
        table.ColumnsDefinition(columns =>
```

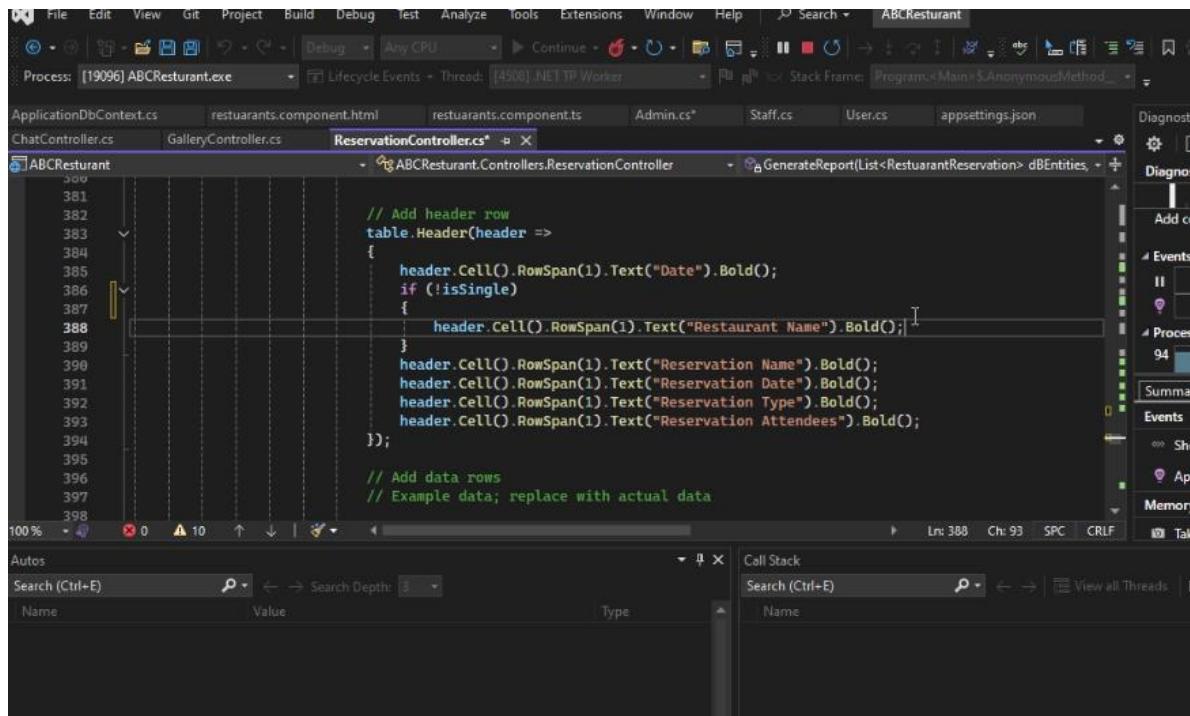
After that a table is created for the report

The screenshot shows the Visual Studio IDE with the ABCRestaurant project open. The current file is ReservationController.cs. The code continues from the previous snippet:

```
// Create a table for the profit report
column.Item().Table(table =>
{
    table.ColumnsDefinition(columns =>
    {
        columns.ConstantColumn(3, Unit.Centimetre); // Width for dates
        if (!isSingle)
        {
            columns.ConstantColumn(70);[]
        }
        columns.ConstantColumn(70);
        columns.ConstantColumn(70);
        columns.ConstantColumn(70);
        columns.ConstantColumn(70);
    });

    // Add header row
    table.Header(header =>
```

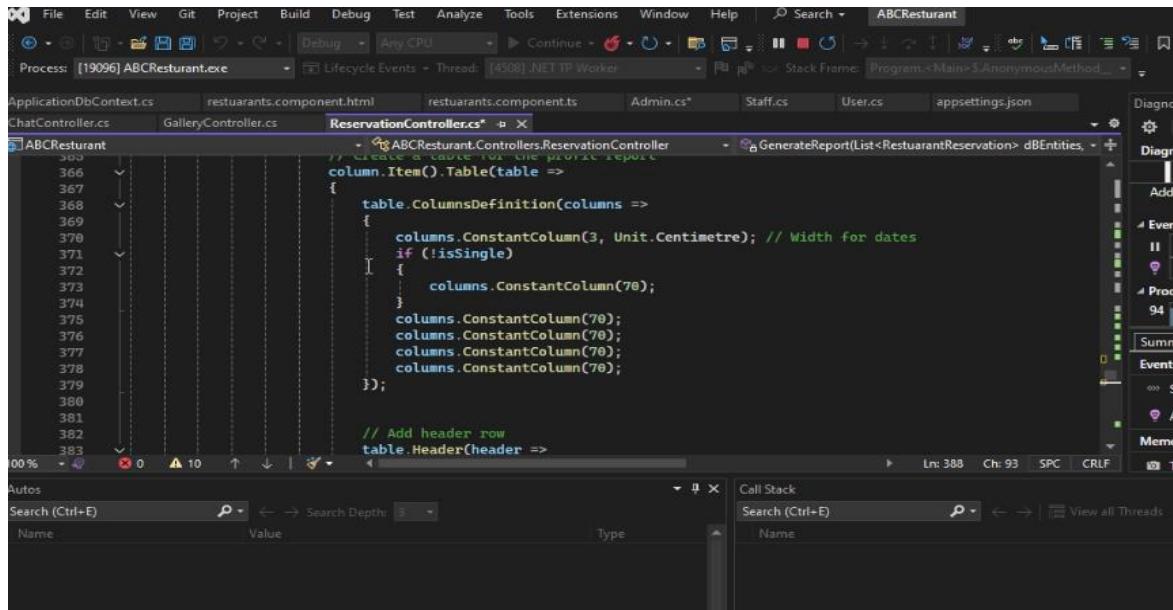
This takes restaurant name, date, attendees



A screenshot of the Visual Studio IDE showing the `ReservationController.cs` file. The code is generating a report for a single restaurant. It includes logic to add a header row with columns for Date, Reservation Name, Reservation Date, Reservation Type, and Reservation Attendees. The code uses `RowSpan(1)` to span these columns across multiple rows of data.

```
381
382
383     // Add header row
384     table.Header(header =>
385     {
386         header.Cell().RowSpan(1).Text("Date").Bold();
387         if (!isSingle)
388             header.Cell().RowSpan(1).Text("Restaurant Name").Bold();
389         header.Cell().RowSpan(1).Text("Reservation Name").Bold();
390         header.Cell().RowSpan(1).Text("Reservation Date").Bold();
391         header.Cell().RowSpan(1).Text("Reservation Type").Bold();
392         header.Cell().RowSpan(1).Text("Reservation Attendees").Bold();
393     });
394
395     // Add data rows
396     // Example data; replace with actual data
397
398
```

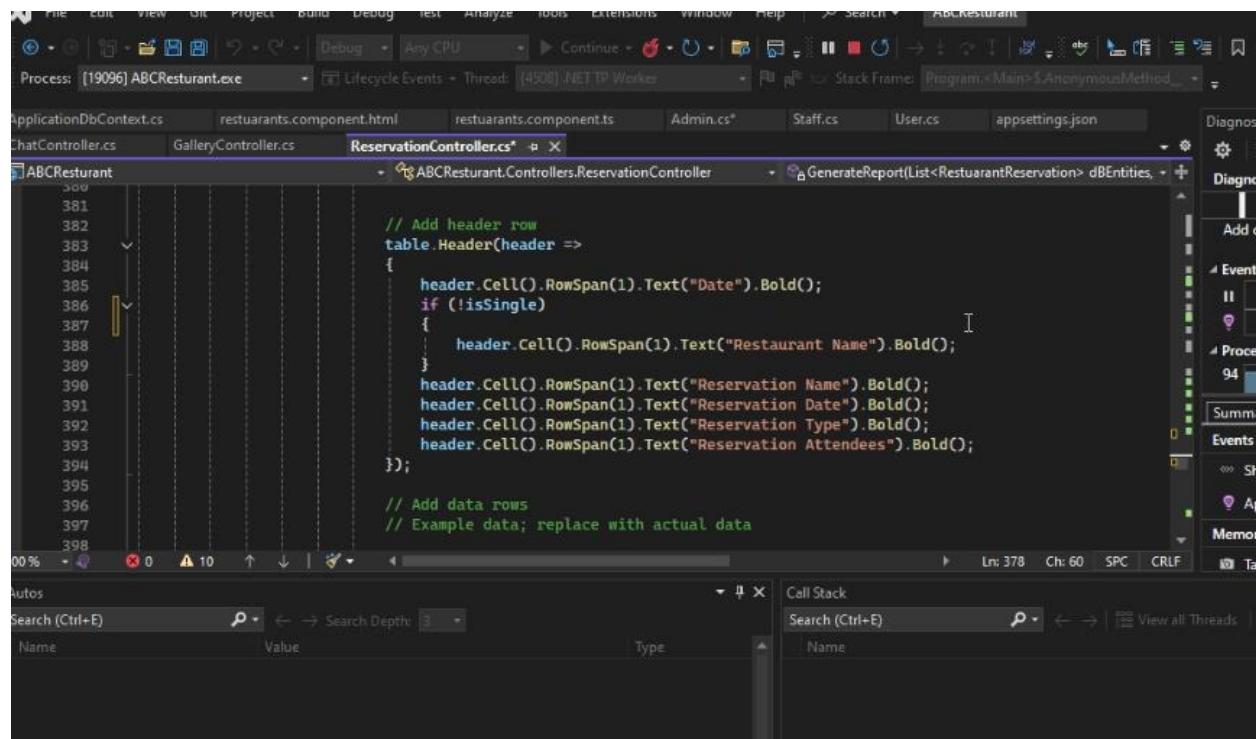
In case this report doesn't depend on one restaurant but the entire restaurant chain, (eg: reservation list of the entire restaurant chain) as below it creates an additional column in the table to get the restaurant names.



A screenshot of the Visual Studio IDE showing the `ReservationController.cs` file. The code has been modified to include an additional column for the restaurant name. It uses `TableDefinition` to define the columns and `Table` to define the rows, adding a new column for the restaurant name before the date.

```
363
364
365     // Add header row
366     table.Header(header =>
367     {
368         column.Item().Table(table =>
369         {
370             table.ColumnsDefinition(columns =>
371             {
372                 columns.ConstantColumn(3, Unit.Centimetre); // Width for dates
373                 if (!isSingle)
374                 {
375                     columns.ConstantColumn(70);
376                     columns.ConstantColumn(70);
377                     columns.ConstantColumn(70);
378                     columns.ConstantColumn(70);
379                 }
380             });
381
382             // Add header row
383             table.Header(header =>
```

This is how we get the column headers of the report.

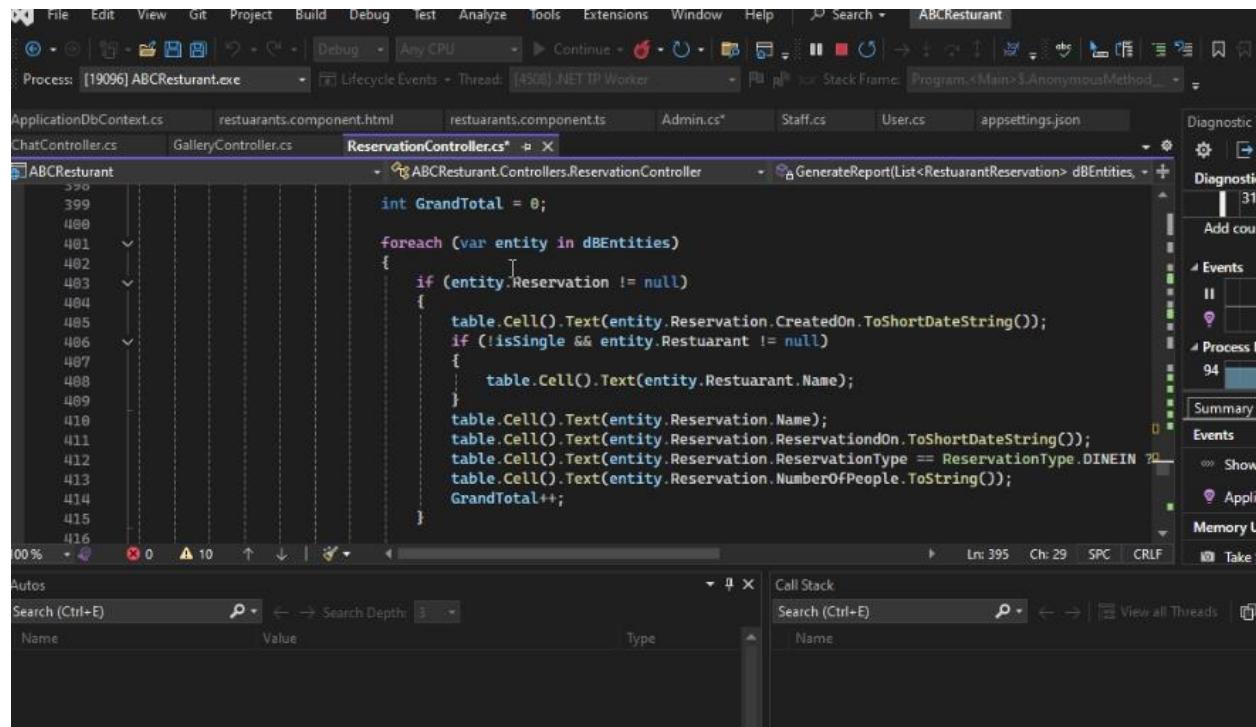


A screenshot of the Visual Studio IDE showing the code editor for `ReservationController.cs`. The code is part of the `ABCRestaurant.Controllers.ReservationController` class. It contains logic to generate a table header with five columns: Date, Restaurant Name, Reservation Name, Reservation Date, and Reservation Type. The code uses `RowSpan(1)` and `Bold()` methods to format the cells. The code editor shows lines 381 to 398. The status bar at the bottom indicates line 378, character 60, and other file paths like `appsettings.json`.

```
    // Add header row
    table.Header(header =>
    {
        header.Cell().RowSpan(1).Text("Date").Bold();
        if (!isSingle)
        {
            header.Cell().RowSpan(1).Text("Restaurant Name").Bold();
        }
        header.Cell().RowSpan(1).Text("Reservation Name").Bold();
        header.Cell().RowSpan(1).Text("Reservation Date").Bold();
        header.Cell().RowSpan(1).Text("Reservation Type").Bold();
        header.Cell().RowSpan(1).Text("Reservation Attendees").Bold();
    });

    // Add data rows
    // Example data; replace with actual data
```

With the help of a foreach loop, it fills the data on the document. GrandTotal has the total number of reservations.



A screenshot of the Visual Studio IDE showing the continuation of the `GenerateReport` method. The code iterates through each entity in `dBEntities`. For each entity, it checks if `entity.Reservation` is not null. If so, it adds the reservation details to the table: CreatedOn, Restaurant Name, Reservation Name, Reservation Date, and Reservation Type. It also increments the `GrandTotal` variable. The code editor shows lines 399 to 416. The status bar at the bottom indicates line 395, character 29, and other file paths like `appsettings.json`.

```
int GrandTotal = 0;

foreach (var entity in dBEntities)
{
    if (entity.Reservation != null)
    {
        table.Cell().Text(entity.Reservation.CreatedOn.ToString("yyyy-MM-dd"));
        if (!isSingle && entity.Restaurant != null)
        {
            table.Cell().Text(entity.Restaurant.Name);
        }
        table.Cell().Text(entity.Reservation.Name);
        table.Cell().Text(entity.Reservation.ReservationDate.ToString("yyyy-MM-dd"));
        table.Cell().Text(entity.Reservation.ReservationType == ReservationType.DINEIN ? "Dine-in" : "Takeout");
        table.Cell().Text(entity.Reservation.NumberOfPeople.ToString());
        GrandTotal++;
    }
}
```

```
    table.Cell().Text(entity.Restaurant.Name);
}
table.Cell().Text(entity.Reservation.Name);
table.Cell().Text(entity.Reservation.ReservationOn.ToShortDateString());
table.Cell().Text(entity.Reservation.ReservationType == ReservationType.DINEIN ? "Dine In" : "Take Out");
table.Cell().Text(entity.Reservation.NumberOfPeople.ToString());
GrandTotal++;

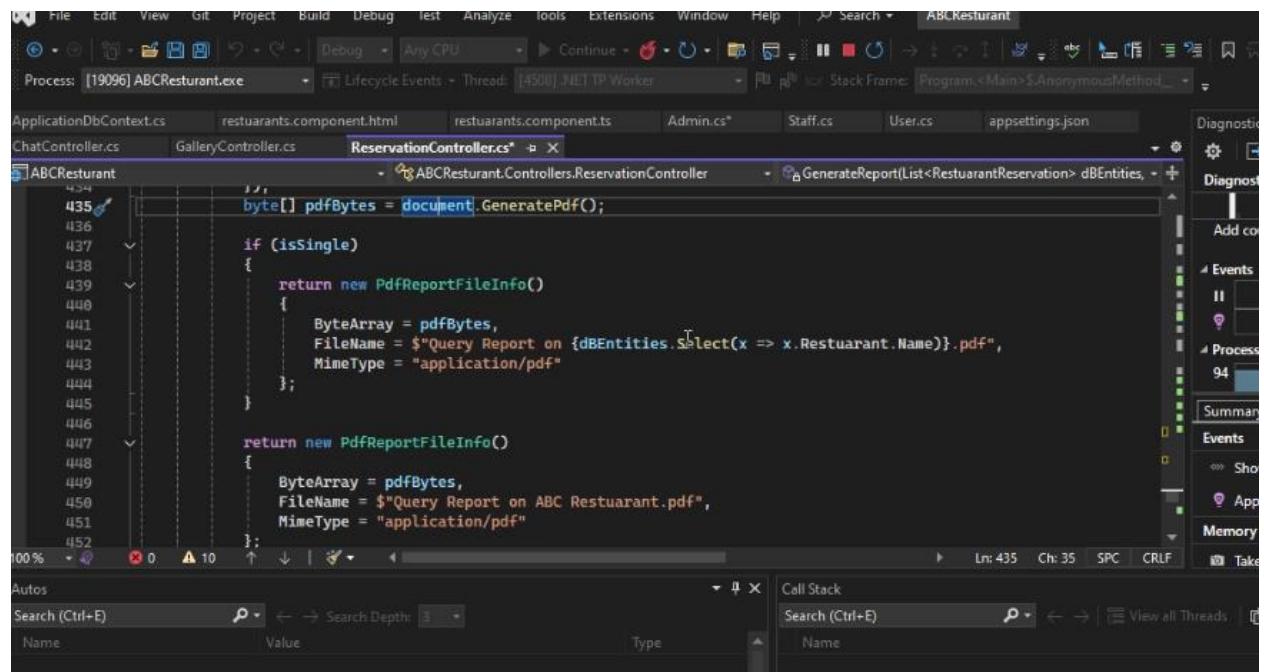
        table.Cell().Text("Total Reservations ").Bold();
        table.Cell().Text($"{GrandTotal}").Bold();
    });
}

pane.Footer()
```

It further describes the content of the document. This is the code for page footer

```
ABCRestaurant
423     });
424 
425     page.Footer()
426         .AlignCenter()
427         .Text(x =>
428     {
429         x.Span("Page ");
430         x.CurrentPageNumber();
431     });
432 
433 });
434 }
435 byte[] pdfBytes = document.GeneratePdf();
436 
437 if (isSingle)
438 {
439     return new PdfReportFileInfo()
440 {
```

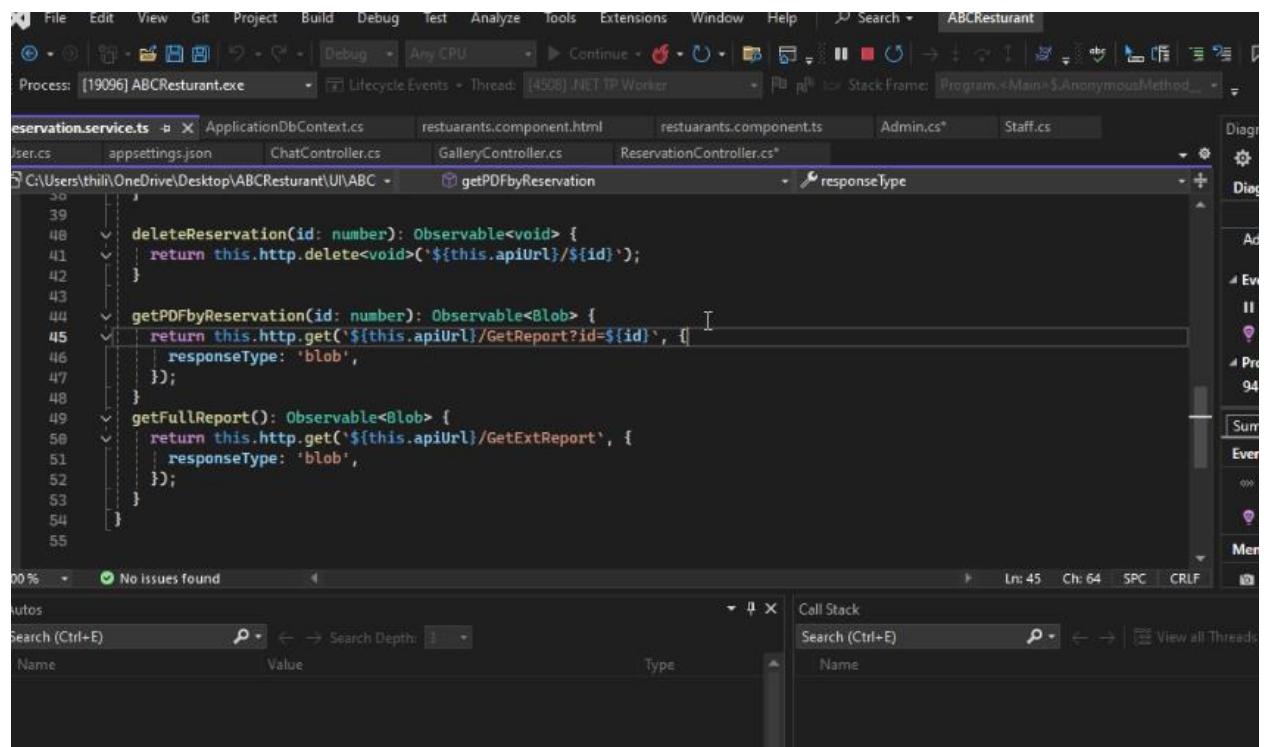
After that it generates the document



Screenshot of Visual Studio showing the `ReservationController.cs` file. The code is part of the `ABCRestaurant.Controllers` namespace. It contains logic for generating PDF reports. The `GenerateReport` method takes a list of `RestaurantReservation` objects and returns a `byte[]` representing the PDF bytes. The code handles both single and multiple reservations, setting `ByteArray`, `FileName`, and `MimeType` accordingly.

```
434
435     byte[] pdfBytes = document.GeneratePdf();
436
437     if (isSingle)
438     {
439         return new PdfReportFileInfo()
440         {
441             ByteArray = pdfBytes,
442             FileName = $"Query Report on {dBEntities.Select(x => x.Restuarant.Name)}.pdf",
443             MimeType = "application/pdf"
444         };
445
446     }
447     return new PdfReportFileInfo()
448     {
449         ByteArray = pdfBytes,
450         FileName = $"Query Report on ABC Restuarant.pdf",
451         MimeType = "application/pdf"
452     };

```



Screenshot of Visual Studio showing the `Reservation.service.ts` file. This is a TypeScript file containing service methods for managing reservations. It includes methods for deleting a reservation and getting a PDF report by reservation ID. The `getPDFbyReservation` method uses the `http.get` method with `responseType: 'blob'`.

```
39
40
41     deleteReservation(id: number): Observable<void> {
42         return this.http.delete<void>(`${this.apiUrl}/${id}`);
43     }
44
45     getPDFbyReservation(id: number): Observable<Blob> {
46         return this.http.get(`${this.apiUrl}/GetReport?id=${id}`, {
47             responseType: 'blob',
48         });
49     }
50     getFullReport(): Observable<Blob> {
51         return this.http.get(`${this.apiUrl}/GetExtReport`, {
52             responseType: 'blob',
53         });
54     }
55 
```

The same logic is used for all the types of reports that gets generated in this API.

Once you click on the PDF or generate report buttons it downloads the document.

The screenshot shows a web browser window for 'ABC Restaurant' at 'localhost:4200/restaurants'. The page displays a list of three restaurants with columns for Name, Location, Telephone, and CreatedOn. Each row has a set of actions icons. A context menu is open over the third row, showing three PDF files: 'report-3 (2).pdf', 'report-2 (1).pdf', and 'report-1 (9).pdf', each with an 'Open file' option.

Name	Location	Telephone	CreatedOn	Actions
ABC restaurant - Bambalapitiya	Bambalapitiya	4563124588	2024-08-11T19:02:02.022746	
ABC restaurant - Kegalle	Kegalle	4513697845	2024-08-11T19:02:34.318664	
ABC restaurant - Ampara	Ampara	1254863594	2024-08-11T19:03:01.801478	

Profit report per restaurant –

This report includes the date and the profit received for the particular day and the total. Admin only can generate this report.

To generate this, the admin has to click on the PDF icon under actions for the particular restaurant under restaurants tab.

The screenshot shows the same 'ABC Restaurant' application interface. A 'Full Report' button is visible in the top right corner of the main content area. The restaurant list and actions are identical to the previous screenshot.

Name	Location	Telephone	CreatedOn	Actions
ABC restaurant - Bambalapitiya	Bambalapitiya	4563124588	2024-08-11T19:02:02.022746	
ABC restaurant - Kegalle	Kegalle	4513697845	2024-08-11T19:02:34.318664	
ABC restaurant - Ampara	Ampara	1254863594	2024-08-11T19:03:01.801478	

Profit Report from ABC restaurant - Ampara

This report is generated.

Date	Profit
8/11/2024	Rs.60000
8/11/2024	Rs.8000
8/11/2024	Rs.20000
8/11/2024	Rs.20000
8/11/2024	Rs.8000
8/12/2024	Rs.9600
Total	Rs.125600

Profit report from ABC restaurant chain –

This report includes the full profit report of the whole restaurant chain. To generate it the admin has to click on the full report button.

localhost:4200/restaurants

Cleon Boerder

Name	Location	Telephone	CreatedOn	Actions
ABC restaurant - Bambalapitiya	Bambalapitiya	4563124588	2024-08-11T19:02:02.022746	
ABC restaurant - Kegalle	Kegalle	4513697845	2024-08-11T19:02:34.318664	
ABC restaurant - Ampara	Ampara	1254863594	2024-08-11T19:03:01.801478	

+ Create

Profit Report from the ABC Restaurant Chain

This report is generated.

	Profit
ABC restaurant - Bambalapitiya	Rs.33800
ABC restaurant - Kegalle	Rs.55200
ABC restaurant - Ampara	Rs.125600
ABC restaurant - Ja Ela	Rs.61000
ABC restaurant - Anuradhapura	Rs.2100
ABC restaurant - Kalutara	Rs.0
ABC restaurant - Matara	Rs.20000
ABC restaurant - Hambantota	Rs.20000
ABC restaurant - Kollupitiya	Rs.0
Total	Rs.317700

Reservation report per restaurant –

Admin can generate reservation report per restaurant using the report icon in reservations tab.

localhost:4200/restaurants

Cleon Boeder -

+ Create Full Report

Search keyword

Name	Location	Telephone	CreatedOn	Actions
ABC restaurant - Bambalapitiya	Bambalapitiya	4563124588	2024-08-11T19:02:02.022746	
ABC restaurant - Kegalle	Kegalle	4513697845	2024-08-11T19:02:34.318664	
ABC restaurant - Ampara	Ampara	1254863594	2024-08-11T19:03:01.801478	

Reservation Report from the ABC ABC restaurant - Bambalapitiya

This report is generated.

Date	Reservation Name	Reservation Date	Reservation Type	Reservation Attendees
8/11/2024	special buffet offer	1/1/0001	Dine In	2
8/11/2024	special buffet offer	1/1/0001	Delivery	20
8/11/2024	Surprise parties	8/28/2024	Delivery	2
8/11/2024	Surprise parties	8/30/2024	Dine In	1
Total Reservations	4			

User activity report –

This report includes the date, who the user is and their activity. This can be generated per user by clicking on PDF icon.

Cleon Boerder

+ Create

Search keyword

Name	Role	Telephone	CreatedOn	UpdatedOn	Actions
Cleon Boerder	Admin	383-8490799	2024-01-13T00:00:00		
Damara Hellin	Staff	8813849155	2023-11-27T00:00:00	2024-08-12T13:21:58.336704	
Donavon Christophersen	Staff	2169890578	2024-05-11T00:00:00	2024-08-12T13:19:22.192093	
Warner Masse	Admin	3885599160	2023-11-22T00:00:00	2024-08-12T13:18:33.015546	
Elwira McQuirk	Admin	134-9992754	2023-08-27T00:00:00		

This is the report for the admin user

The screenshot shows a Microsoft Edge browser window displaying a PDF document. The title of the PDF is "User Activity Report from the Cleon Boerder". The content of the PDF includes a header stating "This report is generated." followed by a table of user activity logs. The table has columns for Date, User, and Activity.

Date	User	Activity
8/9/2024	Cleon Boerder	User Dashboard Loaded
8/11/2024	Cleon Boerder	User Dashboard Loaded
8/11/2024	Cleon Boerder	User Dashboard Loaded
8/11/2024	Cleon Boerder	Offer/Promotion Dashboard Loaded
8/11/2024	Cleon Boerder	User Dashboard Loaded
8/11/2024	Cleon Boerder	Restuarant Dashboard Loaded
8/11/2024	Cleon Boerder	Open restaurant form in create

Reservation report from ABC restaurant chain –

Admin and staff have permissions to generate this report by clicking on full report button in reservations.

The screenshot shows a web-based reservation management system for the ABC Restaurant chain. The interface features a top navigation bar with links for Home, Locations, Restaurants, Users, Gallery, Services, Offers & Promotions, Payments, Reservations, and Queries. A user profile for "Cleon Boerder" is visible on the right. Below the navigation, there is a decorative header with food-related illustrations. A "Create" button is located in the top left corner of the main content area. On the right side, there is a "Full Report" button. The main content area displays a table of reservations with columns for Reservation Name, Reservation On, Type, User Name, Approval Pending, Created On, and Actions.

Reservation Name	Reservation On	Type	User Name	Approval Pending	Created On	Actions
black friday offer	0001-01-01T05:30:00	Delivery	Gal Lockyear	<input checked="" type="checkbox"/>	2024-08-11T19:20:58.664046	
aurudu offer	0001-01-01T05:30:00	Dine In	Rodrick Ochiltree	<input type="checkbox"/>	2024-08-11T19:21:44.773616	
december christmas offer	0001-01-01T05:30:00	Dine In	Kenneth Donald	<input checked="" type="checkbox"/>	2024-08-11T19:32:53.654468	
Surprise parties	0001-01-01T05:30:00	Dine In	Syd Ocheltree	<input checked="" type="checkbox"/>	2024-08-11T19:33:24.382049	

This includes date, restaurant name, reservation name, reservation date, type and no.of attendees.

Date	Restaurant Name	Reservation Name	Reservation Date	Reservation Type	Reservation Attendees
8/11/2024	ABC restaurant - offer	black friday	1/1/0001	Delivery	2
8/11/2024	ABC restaurant - Ja Ela	aurudu offer	1/1/0001	Dine In	5
8/11/2024	ABC restaurant - Ja Ela	december christmas offer	1/1/0001	Dine In	10
8/11/2024	ABC restaurant - Amnara	Surprise parties	1/1/0001	Dine In	20

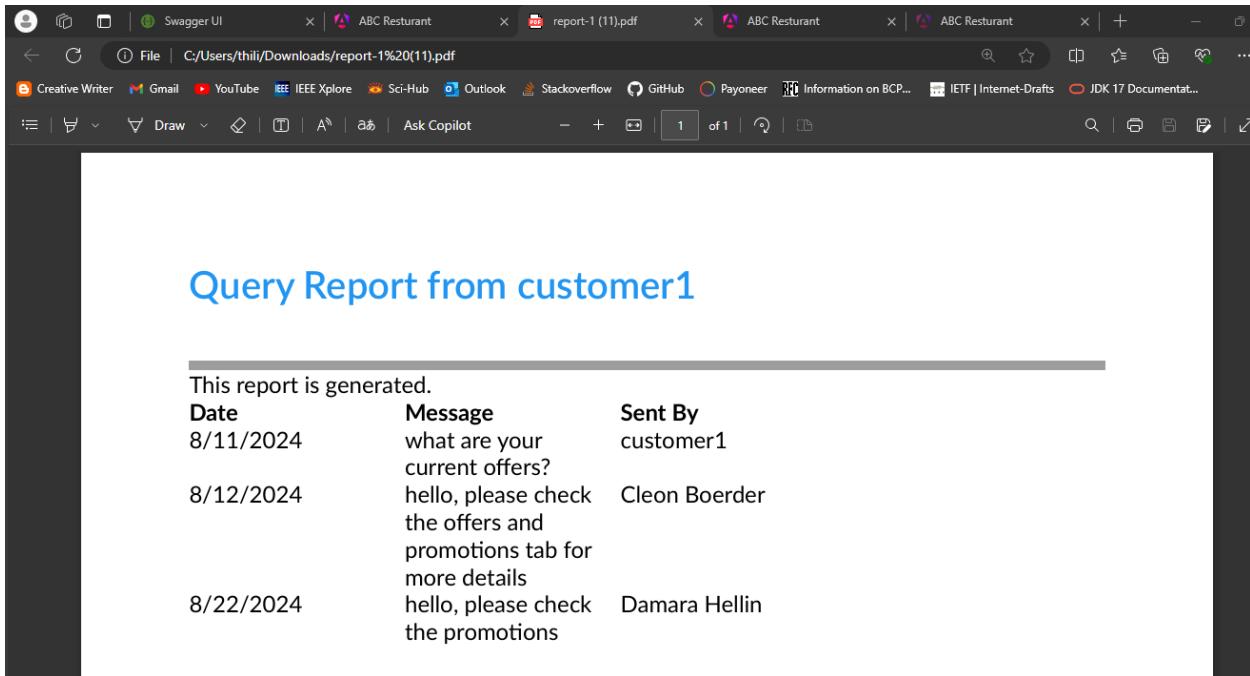
Query reports –

Query reports can be generated per customer and the full report as well. To generate the query report per customer user has to click on the PDF icon under actions in queries tab. Both admin and staff can generate these.

CreatedOn	UpdatedOn	Actions
2024-08-11T14:30:08.427501	0001-01-01T00:00:00	
2024-08-11T19:43:25.953291	0001-01-01T00:00:00	
2024-08-11T19:44:03.578623	0001-01-01T00:00:00	
2024-08-11T19:44:24.135569	0001-01-01T00:00:00	

This includes the date, message and replies from the staff and the name of the person who responded.

Query report per customer -

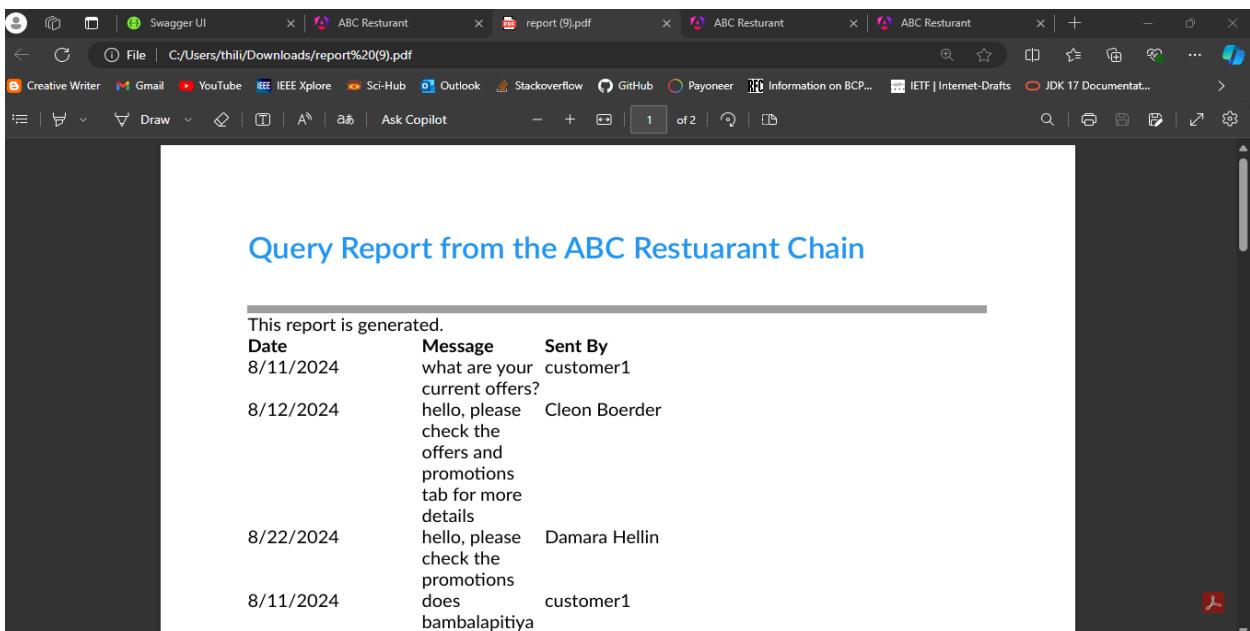


The screenshot shows a web browser window with multiple tabs open. The active tab displays a query report titled "Query Report from customer1". The report begins with a statement: "This report is generated." Below this, there is a table with three columns: Date, Message, and Sent By. The data in the table is as follows:

Date	Message	Sent By
8/11/2024	what are your current offers?	customer1
8/12/2024	hello, please check the offers and promotions tab for more details	Cleon Boerder
8/22/2024	hello, please check the promotions	Damara Hellin

To generate the full query report, user needs to click on full report button. This includes all the queries and their responses with dates. Admin and staff can generate this.

Full query report -



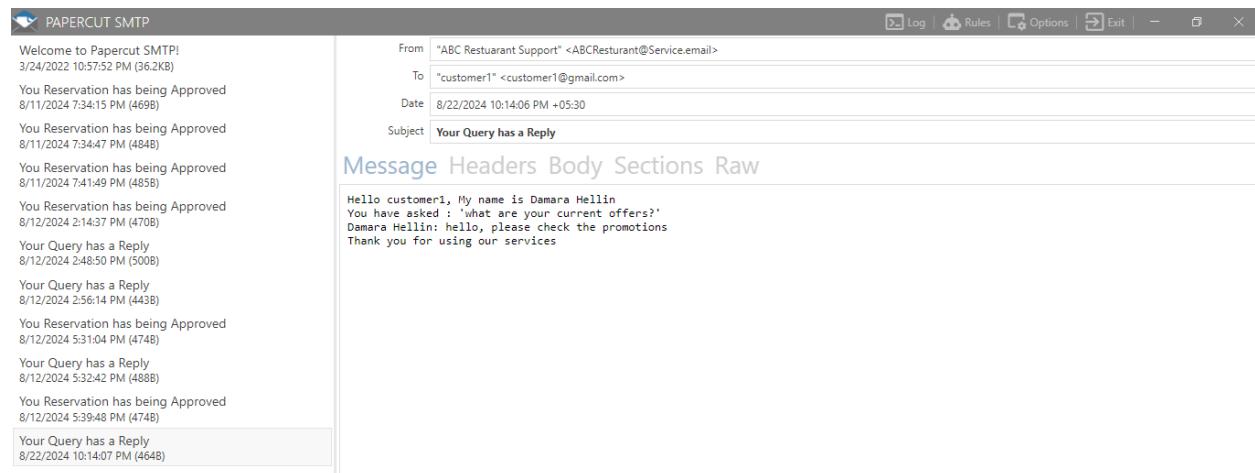
The screenshot shows a web browser window with multiple tabs open. The active tab displays a full query report titled "Query Report from the ABC Restaurant Chain". The report begins with a statement: "This report is generated." Below this, there is a table with three columns: Date, Message, and Sent By. The data in the table is as follows:

Date	Message	Sent By
8/11/2024	what are your current offers?	customer1
8/12/2024	hello, please check the offers and promotions tab for more details	Cleon Boerder
8/22/2024	hello, please check the promotions	Damara Hellin
8/11/2024	does bampalapitiya	customer1

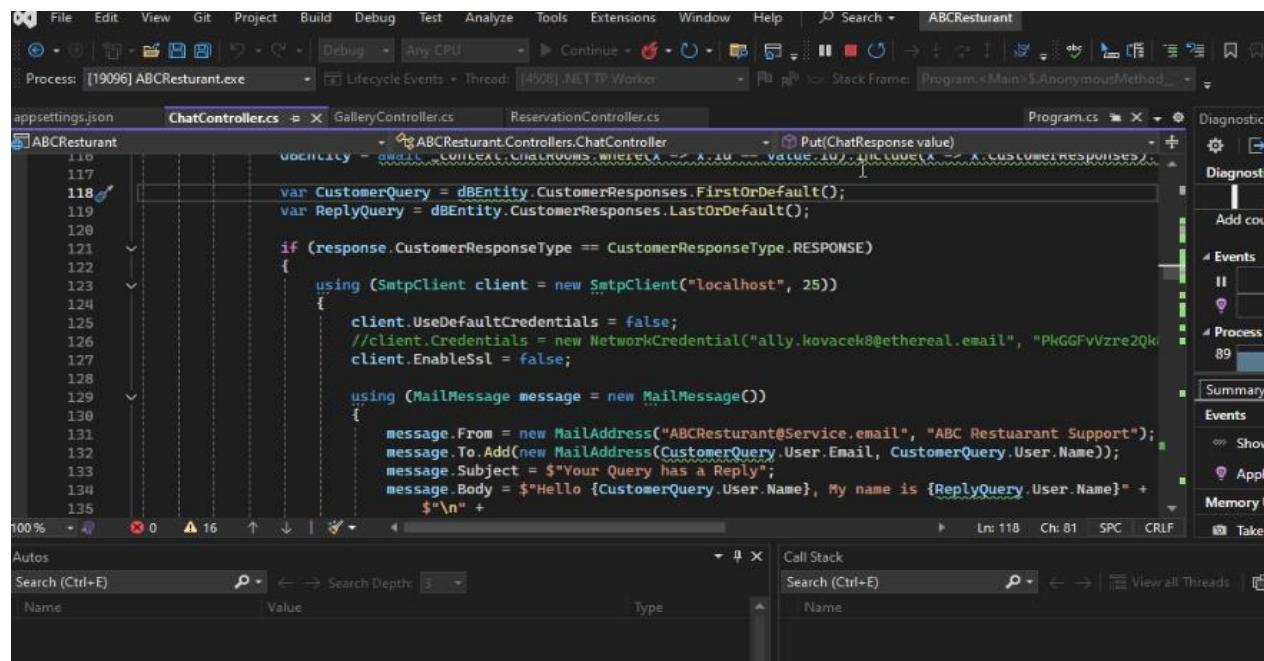
Email Confirmation

Papercut SMTP mail server was used for emailing. This application works as a quick email viewer and a built in SMTP server. The customer receives emails when their reservations are confirmed and once the staff responds to their queries.

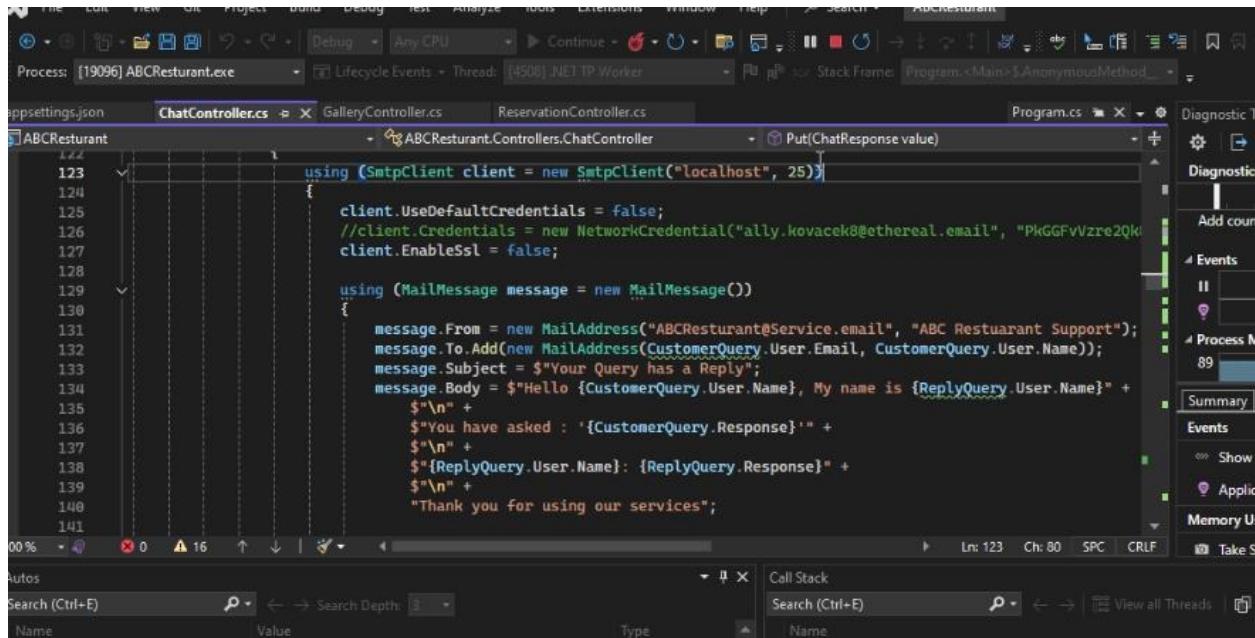
Overview of Papercut SMTP.



In the ChatController.cs it catches the customer query and the staff response.



Next it opens the SMTP mail application. The host is localhost and the port is 25. It asks not to use the default credentials and asks to disable the SSL as well since we're using the localhost for this.

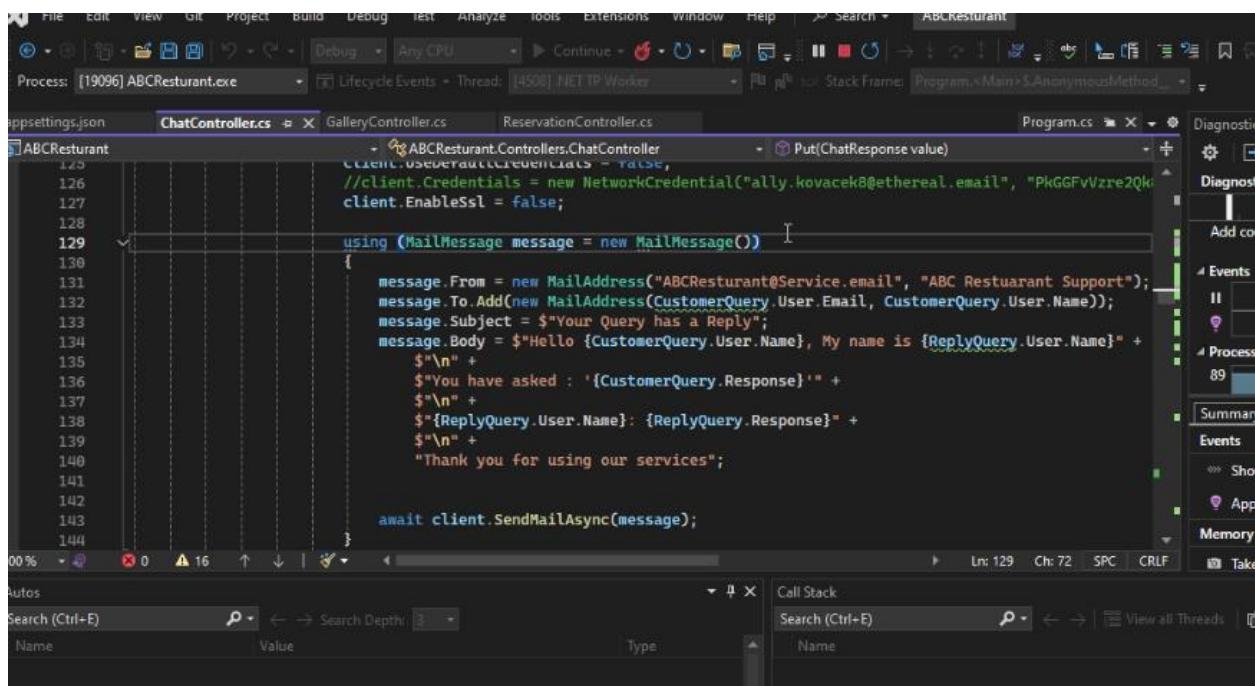


```

123     using (SmtpClient client = new SmtpClient("localhost", 25))
124     {
125         client.UseDefaultCredentials = false;
126         //client.Credentials = new NetworkCredential("ally.kovacek8@ethereal.email", "PkgGFvVzre2Qk");
127         client.EnableSsl = false;
128
129         using (MailMessage message = new MailMessage())
130         {
131             message.From = new MailAddress("ABCRestaurant@Service.email", "ABC Restaurant Support");
132             message.To.Add(new MailAddress(CustomerQuery.User.Email, CustomerQuery.User.Name));
133             message.Subject = $"Your Query has a Reply";
134             message.Body = $"Hello {CustomerQuery.User.Name}, My name is {ReplyQuery.User.Name}" +
135                 $"{Environment.NewLine}" +
136                 $"You have asked : '{CustomerQuery.Response}'" +
137                 $"{Environment.NewLine}" +
138                 $"'{ReplyQuery.User.Name}: {ReplyQuery.Response}'" +
139                 $"{Environment.NewLine}" +
140                 "Thank you for using our services";
141
142         }
143
144         await client.SendMailAsync(message);
145     }

```

Next it describes what includes in the mail in MailMessage(). It includes From, To, Subject and Body fields. Finally it calls for client.MailAsync(message) and wait for the email to be received.



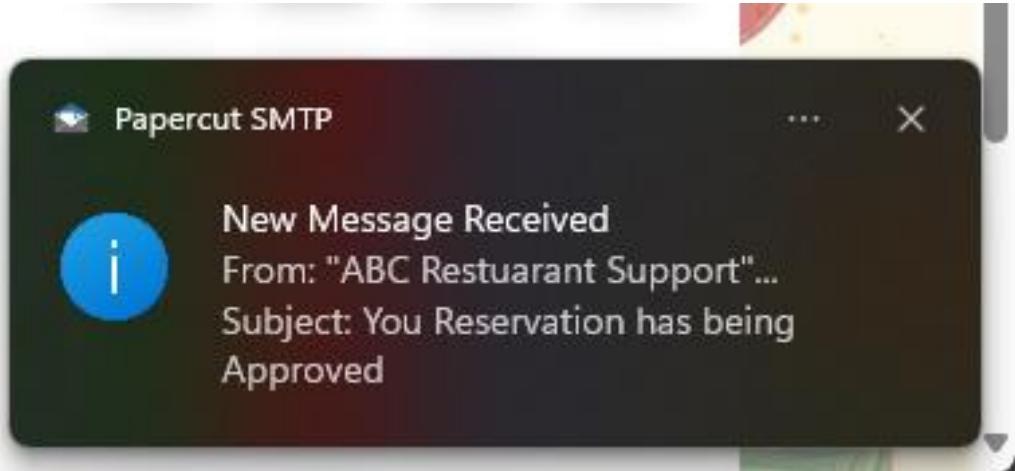
```

123     using (SmtpClient client = new SmtpClient("localhost", 25))
124     {
125         client.UseDefaultCredentials = false;
126         //client.Credentials = new NetworkCredential("ally.kovacek8@ethereal.email", "PkgGFvVzre2Qk");
127         client.EnableSsl = false;
128
129         using (MailMessage message = new MailMessage())
130         {
131             message.From = new MailAddress("ABCRestaurant@Service.email", "ABC Restaurant Support");
132             message.To.Add(new MailAddress(CustomerQuery.User.Email, CustomerQuery.User.Name));
133             message.Subject = $"Your Query has a Reply";
134             message.Body = $"Hello {CustomerQuery.User.Name}, My name is {ReplyQuery.User.Name}" +
135                 $"{Environment.NewLine}" +
136                 $"You have asked : '{CustomerQuery.Response}'" +
137                 $"{Environment.NewLine}" +
138                 $"'{ReplyQuery.User.Name}: {ReplyQuery.Response}'" +
139                 $"{Environment.NewLine}" +
140                 "Thank you for using our services";
141
142         }
143
144         await client.SendMailAsync(message);
145     }

```

Once a reservation is approved by staff, the particular user gets an email confirmation saying their reservation is confirmed.

The notification -



The message appears like in the below image.

The screenshot shows the Papercut SMTP application interface. On the left, there's a list of recent messages. The main area displays an incoming email from "ABC Restaurant Support" to "Gal Lockyear" on 8/22/2024 at 10:20:44 PM. The message subject is "You Reservation has being Approved". The message body contains a welcome message and a note about a black Friday offer.

From: "ABC Restaurant Support" <ABCRestaurant@Service.email>
To: "Gal Lockyear" <glockyear6@google.cn>
Date: 8/22/2024 10:20:44 PM +05:30
Subject: You Reservation has being Approved

Hello Gal Lockyear,
Your reservation name black friday offer has being approved by our staff and it will be held on 1/1/0001 5:30:00 AM.
Thank you for using our services

Message Headers Body Sections Raw

Recent messages (partial list):

- Welcome to Papercut SMTP!
- 3/24/2022 10:57:52 PM (36.2KB)
- You Reservation has been Approved
- 8/11/2024 7:34:15 PM (469B)
- You Reservation has been Approved
- 8/11/2024 7:34:47 PM (484B)
- You Reservation has been Approved
- 8/11/2024 7:41:49 PM (485B)
- You Reservation has been Approved
- 8/12/2024 2:14:37 PM (470B)
- Your Query has a Reply
- 8/12/2024 2:48:50 PM (500B)
- Your Query has a Reply
- 8/12/2024 2:56:14 PM (443B)
- You Reservation has been Approved
- 8/12/2024 5:31:04 PM (474B)
- Your Query has a Reply
- 8/12/2024 5:32:42 PM (488B)
- You Reservation has been Approved
- 8/12/2024 5:39:48 PM (474B)
- Your Query has a Reply
- 8/22/2024 10:14:07 PM (464B)
- You Reservation has been Approved
- 8/22/2024 10:20:44 PM (474B)

Once a response is received for a query, customer gets an email like below.

The screenshot shows the Papercut SMTP interface. On the left, there's a list of incoming emails from "ABC Restaurant Support". On the right, a specific email is selected, showing its details:

From: "ABC Restaurant Support" <ABCRestaurant@Service.email>
To: "customer1" <customer1@gmail.com>
Date: 8/22/2024 10:14:06 PM +0530
Subject: Your Query has a Reply

Message Headers:

```
Hello customer1, My name is Damara Hellin
You have asked : 'what are your current offers?'
Damara Hellin: hello, please check the promotions
Thank you for using our services
```

Body:

Message Headers Body Sections Raw

Sections:

Message Headers Body Sections Raw

Raw:

```
Hello customer1, My name is Damara Hellin
You have asked : 'what are your current offers?'
Damara Hellin: hello, please check the promotions
Thank you for using our services
```

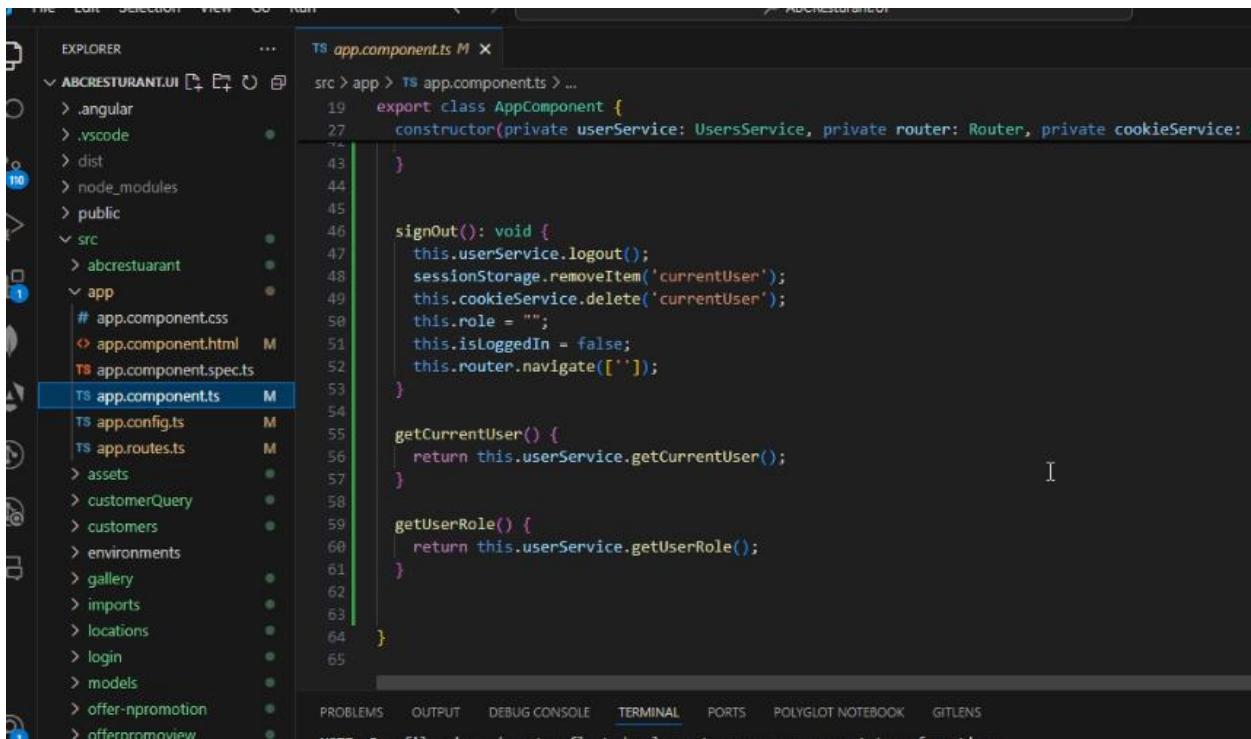
Use of sessions and cookies

In the users.service.ts, through getCurrentUser(), currentUser session is accessed from the session storage. Same happens with cookies also since the same currentUser is accessing from the cookieService.

The screenshot shows the VS Code interface with the "users.service.ts" file open in the editor. The code is as follows:

```
src > services > users.service.ts > UserService > getCurrentUser
10 export class UserService {
11   updateUser(User: User): Observable<User> {
12     return this.http.put<User>(`${this.apiUrl}`, User);
13   }
14
15   deleteUser(id: number): Observable<void> {
16     return this.http.delete<void>(`${this.apiUrl}/${id}`);
17   }
18
19   login(User: any): Observable<any> {
20     return this.http.post<any>(`${this.apiUrl}/login`, User);
21   }
22
23   logout(): void {
24     this.responseSubject.next(null);
25     sessionStorage.removeItem('currentUser');
26     this.http.get(`${this.apiUrl}/logout`);
27   }
28
29   getCurrentUser(): any {
30     const sessionUser = safeJsonParse(sessionStorage.getItem('currentUser'));
31     const cookieUser = safeJsonParse(this.cookieService.get('currentUser'));
32     return sessionUser ? sessionUser : cookieUser;
33   }
34
35   getUserRole(): string {
36     const user = this.getCurrentUser();
37     return user ? user.role : null;
38   }
39 }
```

When the getCurrentUser is returned, it checks whether the user has a session or cookies. If the user has a cookie, the user is not logged out and they can log back in without entering username and password. If the user has logged out, the cookie should be destroyed. When the sign out is pressed, both the cookie and the session is destroyed.



The screenshot shows the VS Code interface with the 'app.component.ts' file open in the editor. The code is written in TypeScript and defines an AppComponent class. It includes methods for signOut, getCurrentUser, and getUserRole. The 'getCurrentUser' method returns a promise from the userService. The 'signOut' method removes the 'currentUser' item from sessionStorage and deletes it from the cookieService. The 'getUserRole' method returns a promise from the userService. The code uses arrow functions and the async/await pattern.

```
export class AppComponent {
  constructor(private userService: UserService, private router: Router, private cookieService: CookieService) { }

  signOut(): void {
    this.userService.logout();
    sessionStorage.removeItem('currentUser');
    this.cookieService.delete('currentUser');
    this.role = "";
    this.isLoggedIn = false;
    this.router.navigate(['']);
  }

  getCurrentUser(): Promise<any> {
    return this.userService.getCurrentUser();
  }

  getUserRole(): Promise<any> {
    return this.userService.getUserRole();
  }
}
```

In the login once the response is set sessionStorage is asked to create a new session called currentUser. If the login is successful (status 200) user's data is stored in cookieService and sessionStorage.

```

File Edit Selection View Go Run ...
EXPLORER ... TS login.component.ts U X
src > login > TS login.components.ts > LoginComponent > handleUpdateResponse
29   export class LoginComponent {
30
31   handleUpdateResponse(response: any): void {
32     this.UserService.setResponse = response;
33     console.log(response.statusCode);
34     if (response.statusCode === 200) {
35       console.log(response.data.name);
36       const userName: string = response.data.name.toString();
37       sessionStorage.setItem('currentUser', JSON.stringify(response.data));
38       this.cookieService.set('currentUser', JSON.stringify(response.data));
39
40       if(response.data.role === "Admin"){
41         this.router.navigate(['users']);
42       }
43       if(response.data.role === "Staff"){
44         this.router.navigate(['reservations']);
45       }
46       if(response.data.role === "Customer"){
47         this.router.navigate(['']);
48       }
49     } else {
50       alert(response.Message);
51     }
52     this.messageService.add({
53       severity: 'success',
54       summary: 'Success',
55       detail: `Welcome ${response.data.name}`;
56   }

```

Functional Requirements

Admin operations – admin must be able to do the following.

- Restaurants – view, create, delete, edit, generate reports on restaurants
- Users - view, create, delete, edit, generate reports on users
- Gallery – view, upload, delete gallery
- Services - view, create, delete, edit services
- Offers and promotions – view, create, delete, edit promotions
- Payments – create, view, delete payments
- Reservations – view, create, delete, edit, generate reports, confirm reservations
- Queries – view, respond, delete, generate reports on queries

Staff operations – staff must be able to do the following.

- Payments – create, view, delete payments
- Reservation - view, create, delete, edit, generate reports, confirm reservations
- Queries – view, respond, delete, generate reports on queries

Customer operations – customers must be able to do the following

- Check locations
- Check rates and availability for reservations of each restaurant
- Make reservations
- Search for services
- Check offers and promotions
- Submit queries
- Receive email confirmations for query responses and reservations

All the forms must contain form validations.

Search facility for each table view. (example: search bar in reservations to search reservation records)

Non-functional Requirements

- The system tracks user activities of all the users and generate reports on it.
- The system is scalable in the future and can handle large sets of datasets.
- The system consists of user-friendly interfaces to handle user interactions seamlessly.
- Error messages and confirmation notifications are displayed to quickly resolve issues.
- System is integrated with third-party services (Papercut SMTP for emails)
- System is able to adapt to new business features

TASK C

Quality Assurance and testing

Test driven development was followed. First the test cases were written, then development was done. Once the development is done, the test cases were executed to check whether the system is working as intended.

Manual test case report

Test environment details –

Operating system - Windows 11

Web browser – Microsoft Edge version 128.0.2739.42

Total number of test cases executed – 38

Passed – 38

Failed – 0

Test cases

Test case ID	Module	Test scenario	Test steps	Test data	Expected results	Actual result	Pass/fail
1	Restaurants	Verify create restaurant-admin	1. click restaurants 2. click create restaurant 3. fill form	Name – ABC restaurant - Negombo Location – Negombo	A new restaurant should be created	A new restaurant is created	Pass

			4. click on create	Maximum reservations - 250 Telephone - 1203254693			
2		Verify update restaurant – admin	1. click restaurants 2. click edit restaurant icon 3. edit form fields 4. click on update	Name – ABC restaurant - Hikkaduwa Location – Hikkaduwa Maximum reservations - 100 Telephone - 4521369845	The record should be updated	The record is updated	Pass
3		Verify delete restaurant – admin	1 click restaurants 2 click delete icon 3 confirm deletion by clicking on okay		The restaurant should be deleted	The restaurant is deleted	Pass
4		Verify upload image to restaurant record – admin	1 click restaurant table view 2 click image upload icon 3 select image from computer 4 click upload		The image should be uploaded. Image should be visible in gallery table view and in location.	Image uploaded and visible in gallery table view and location	Pass
5		Verify generate profit report for a restaurant – admin	1 click restaurants 2 click PDF icon for the restaurant you need to generate the report for		The report should be generated and download ed	The report is generated and downloaded	Pass
6		Verify generate reservation	1 click restaurants 2 click PDF icon for the restaurant		The report should be generated	The report is generated and downloaded	Pass

		report for a restaurant – admin	you need to generate the report for		and download ed		
7		Verify view restaurant – admin	1 click restaurants 2 click view icon		The view pop up should be visible and all the test fields should be disabled	View pop up appears and the text fields are disabled	Pass
8		Verify generate Profit Report from the ABC Restaurant Chain-admin	1 click restaurants 2 click full report button		The report should be generated and downloaded	The report is generated and downloaded	Pass
9		Verify search option – admin	1 click restaurants 2 enter search term on the search bar	Search term – ABC restaurant - Bambalapitiya	Correct search results should be visible	Correct search results are shown	Pass
10	Users	Verify create customer – admin	1 click users 2 click create 3 fill in details 4 click create	Name – Caroline Perera Email – Caroline@mail.com Password – caroline@45 Role – customer Telephone - 1234561245	A new customer should be created and only the customer permissions should be available when logged in	A new customer is created and only the customer permissions are available when logged in	Pass
11		Verify create staff – admin	1 click users table view 2 click create 3 fill in details	Name – Freddie Perera	A new staff user should be created	A new staff user is created and only the staff	Pass

			4 click create	Email – fred@gmail.com Password – fred@4521 Role – staff Telephone - 5213601245	and only the staff permissions should be available when logged in	permissions are available when logged in	
12		Verify create admin – admin	1 click users table view 2 click create button 3 fill in details 4 click create	Name – Nelly Nelson Email – nel@gmail.com Password – nelly\$nelson Role – staff Telephone - 5236489412	A new admin should be created	The admin is created	Pass
13		Verify edit user – admin	1 click users 2 click edit button 3 edit details 4 click update	Name – Nelly Manson Email – nelmanson@gmail.com Password – nelly\$manson Role – staff Telephone - 5236489412	The role should be updated	The role is updated	Pass
14		Verify delete user – admin	1 click users 2 click delete 3 confirm deletion by clicking okay		The user record should be deleted	The user record is deleted	Pass
15		Verify generating user activity report - admin	1 click users 2 click PDF icon next to the user		User activity report should be downloaded	User activity report is downloaded	Pass
16		Verify view users – admin	1 click users 2 click view button		View pop up should be visible. All the	View pop up is visible. All the text fields are disabled	Pass

					text fields should be disabled		
17	Gallery	Verify delete image - admin	1 click gallery 2 click delete button next to image 3 confirm deletion		Image should be deleted	Image is deleted	Pass
18	Service s	Verify create service – admin	1 click services 2 click create 3 fill details 4 click create	Name – self service Restaurant – ABC Restaurant- Bambalapitiy a	New service should be created. It should be added to the particular restaurant	New service is created. It is added to restaurant	Pass
19		Verify edit service - admin	1 click services 2 click edit service 3 edit details 4 click update	Name – all- you-can-eat- service Restaurant – ABC Restaurant- Galle	Service should be updated	Service is updated	Pass
20		Verify delete service – admin	1 click services 2 click delete button 3 confirm deletion		Service should be deleted	Service is deleted	Pass
21	Offers and promoti ons	Verify create offer/pro motion – admin	1 click offers and promotions 2 click create 3 fill details 4 click create	Message – lunch buffet offer Code – lunch20 Discount – 20 Offer until – 2024.09.30	Offer should be created	Offer is created	Pass
22		Verify edit offer/pro motion – admin	1 click offers and promotions 2 click edit 3 edit details 4 click update	Message - buffet offer Code – lunch25 Discount – 25	Offer should be updated	Offer is updated	Pass

				Offer until – 2024.10.30			
23		Verify delete offer/promotion – admin	1 click offers and promotions 2 click delete 3 confirm deletion		Offer should be deleted	Offer is deleted	Pass
24	Payments	Verify create payment – admin, staff	1 click payments 2 click create	Restaurant – ABC Restaurant-Bambalapitiya Payment – 3200 Customer – customer1	Payment should be created	Payment is created	Pass
25		Verify delete payment – admin, staff	1 click payments 2 click delete		Payment should be deleted	Payment is deleted	Pass
26	Reservations	Verify create reservation – admin, staff	1 click reservations 2 click create 3 fill details 4 click create	Name – birthday party Date – 2024.09.01 Customer – customer1 Reservation type – dine-in Restaurant No.of people - 25	Reservation should be created. Make reservation button should be enabled for the particular restaurant	Reservation is created. Make reservation button is available	Pass
27		Verify edit reservation – admin, staff	1 click reservations 2 click edit 3 edit fields 4 click update	Name – bridal shower Date – 2024.09.02 Customer – customer2 Reservation type – dine-in Restaurant No.of people - 30	Reservation should be updated	Reservation is updated	Pass

28		Verify delete reservation – admin, staff	1 click reservations 2 click delete 3 confirm deletion		Reservation should be deleted	Reservation is deleted	Pass
29		Verify confirm reservation – admin, staff	1 click reservations 2 click confirm button		Reservation should be confirmed, confirmation email should be sent to customer	Reservation is confirmed, confirmation email is sent	Pass
30		Verify delete reservation – admin, staff	1 click reservations 2 click delete 3 confirm deletion		Reservation should be deleted	Reservation is deleted	Pass
31		Verify generate Reservation Report from the ABC Restaurant Chain – admin, staff	1 click reservations 2 click full report		Report should be downloaded	Report is downloaded	Pass
32	Queries	Verify respond to query – admin, staff	1 click queries 2 click respond icon 3 write response 4 click submit	Response – please check our website	Response should be submitted ; email should be sent to customer	Response is submitted, email is sent	Pass
33		Verify delete query – admin, staff	1 click queries 2 click delete 3 confirm deletion		Query should be deleted	Query is deleted	Pass

34		Verify generate query report per customer – admin, staff	1 click queries 2 click PDF icon		Report should be downloaded	Report is downloaded	Pass
35		Verify generate Query Report from the ABC Restaurant Chain – admin, staff	1 click queries 2 click full report icon		Report should be downloaded	Report is downloaded	Pass
36	Locations	Verify check locations – admin, staff, customer	1 click locations 2 choose location and click view details		Users should be able to view details of location	Details can be viewed	Pass
37	Ask us	Verify submit query – customer	1 click ask us 2 type query 3 click submit	Query – do you deliver food?	Query should be submitted and should be viewed by staff	Query is submitted and can be viewed by admin and staff	Pass
38	Make reservation	Verify make reservation – customer	1 click locations 2 choose location, click view details 3 click make reservation 4 fill form 5 click pay	Name – engagement party Date – 2024.10.12 Reservation type – dine-in No.of people attending – 50	Reservation should be made successfully. Amount to pay should be calculated correctly based on	Reservation is made and amount is calculated correctly	Pass

				Offer/promo code – september20	no.of people, current rate, promo code		
--	--	--	--	--------------------------------	--	--	--

Test Automation

Operating system - Windows 11

Web browser – Microsoft Edge version 128.0.2739.42

Test automation tool – Katalon studio 9.5.0

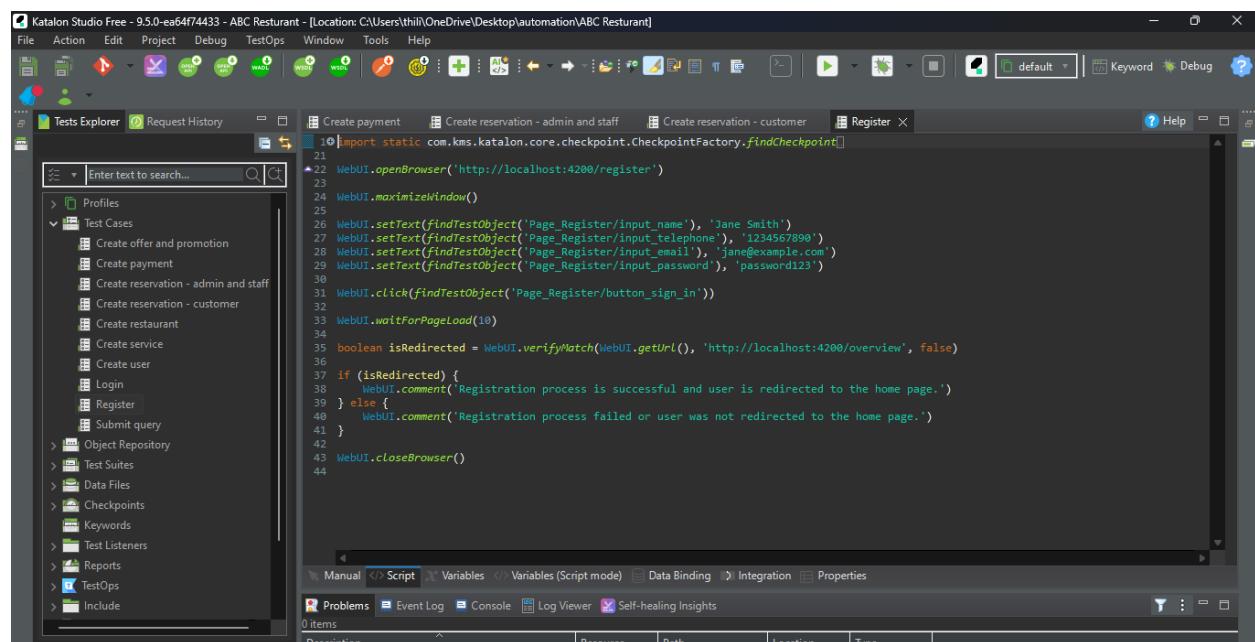
Total number of test cases executed – 10

Passed – 10

Failed – 0

Following are the test scripts for every test case coded in Groovy in Katalon studio.

Register

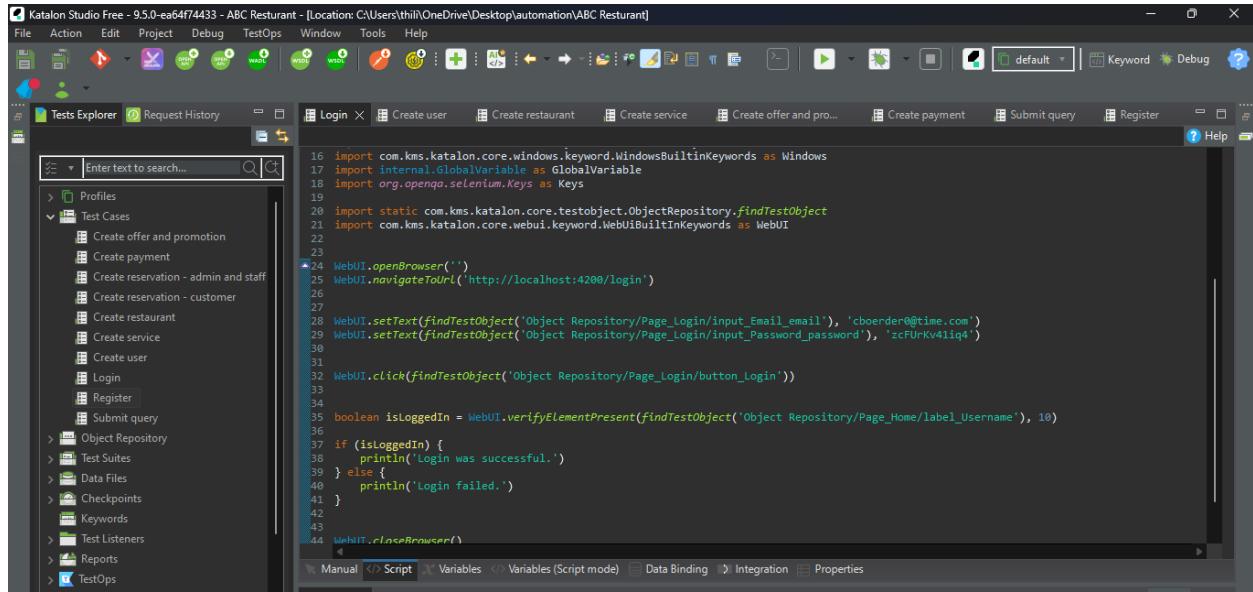


```

Katalon Studio Free - 9.5.0-ea64f74433 - ABC Restaurant - [Location: C:\Users\thii\OneDrive\Desktop\automation\ABC Restaurant]
File Action Edit Project Debug TestOps Window Tools Help
Tests Explorer Request History
Enter text to search...
> Profiles
> Test Cases
  > Create offer and promotion
  > Create payment
  > Create reservation - admin and staff
  > Create reservation - customer
  > Create restaurant
  > Create service
  > Create user
  > Login
  > Register
  > Submit query
> Object Repository
> Test Suites
> Data Files
> Checkpoints
> Keywords
> Test Listeners
> Reports
> TestOps
> Include
Create payment Create reservation - admin and staff Create reservation - customer Register X
10 import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint()
11 WebUI.openBrowser('http://localhost:4200/register')
12 WebUI.maximizeWindow()
13 WebUI.setText(findTestObject('Page_Register/input_name'), 'Jane Smith')
14 WebUI.setText(findTestObject('Page_Register/input_telephone'), '1234567890')
15 WebUI.setText(findTestObject('Page_Register/input_email'), 'jane@example.com')
16 WebUI.setText(findTestObject('Page_Register/input_password'), 'password123')
17 WebUI.click(findTestObject('Page_Register/button_sign_in'))
18 WebUI.waitForPageLoad(10)
19 boolean isRedirected = WebUI.verifyMatch(WebUI.getUrl(), 'http://localhost:4200/overview', false)
20 if (isRedirected) {
21     WebUI.comment('Registration process is successful and user is redirected to the home page.')
22 } else {
23     WebUI.comment('Registration process failed or user was not redirected to the home page.')
24 }
25 WebUI.closeBrowser()

```

Login

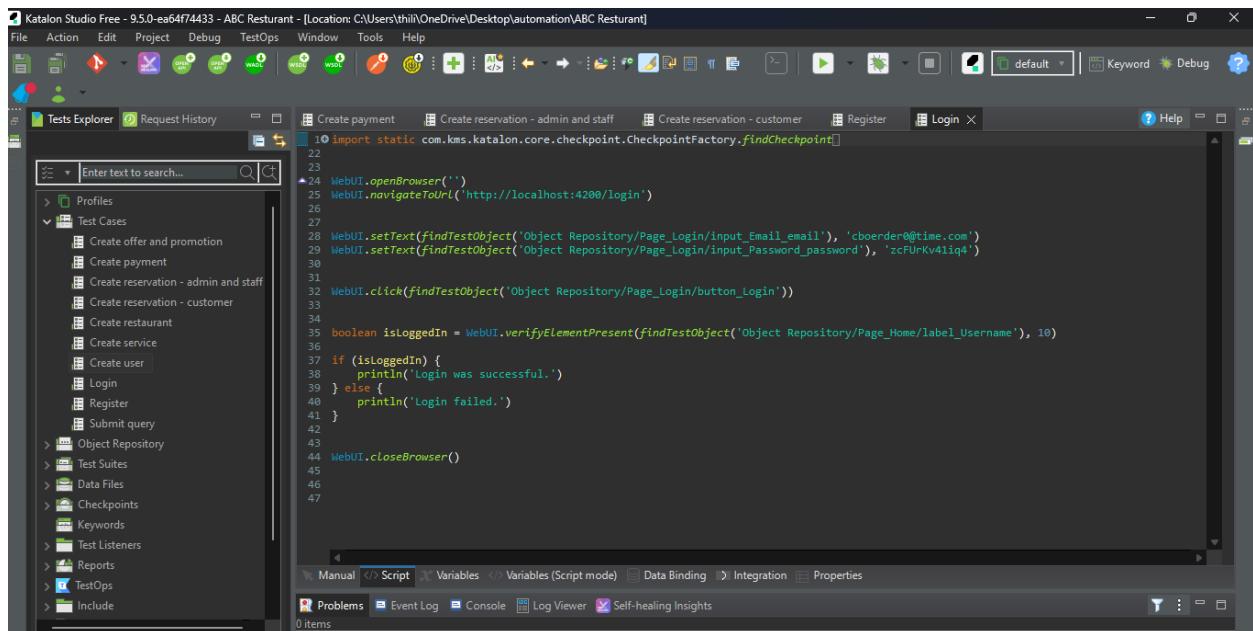


Katalon Studio Free - 9.5.0-ea64f74433 - ABC Restaurant - [Location: C:\Users\thili\OneDrive\Desktop\automation\ABC Restaurant]

The screenshot shows the Katalon Studio interface with the 'Login' test case selected in the Tests Explorer. The main pane displays a Groovy script for performing a login operation. The script uses Katalon's built-in keywords like WebUI to open a browser, navigate to the login page, set email and password, click the login button, and verify if the user is logged in by checking for a specific label. The code is as follows:

```
16 import com.kms.katalon.core.windows.keyword.WindowsBuiltinKeywords as Windows
17 import internal.GlobalVariable as GlobalVariable
18 import org.openqa.selenium.Keys as Keys
19
20 import static com.kms.katalon.core.testobject.ObjectRepository.findTestObject
21 import com.kms.katalon.core.webui.keyword.WebUiBuiltInKeywords as WebUI
22
23
24 WebUI.openBrowser('')
25 WebUI.navigateToUrl('http://localhost:4200/login')
26
27
28 WebUI.setText(findTestObject('Object Repository/Page_Login/input_Email_email'), 'cboerder0@time.com')
29 WebUI.setText(findTestObject('Object Repository/Page_Login/input_Password_password'), 'zcFUrKv4liq4')
30
31
32 WebUI.click(findTestObject('Object Repository/Page_Login/button_Login'))
33
34
35 boolean isLoggedIn = WebUI.verifyElementPresent(findTestObject('Object Repository/Page_Home/label_Username'), 10)
36
37 if (isLoggedIn) {
38     println('Login was successful.')
39 } else {
40     println('Login failed.')
41 }
42
43
44 WebUI.closeBrowser()
```

Create user

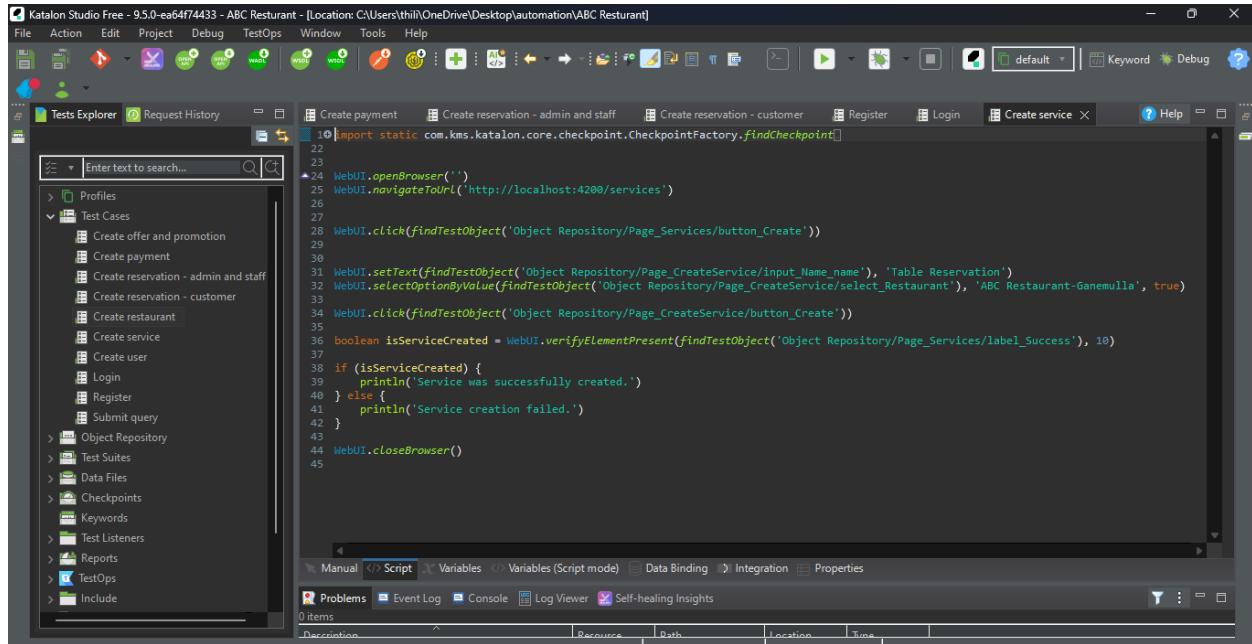


Katalon Studio Free - 9.5.0-ea64f74433 - ABC Restaurant - [Location: C:\Users\thili\OneDrive\Desktop\automation\ABC Restaurant]

The screenshot shows the Katalon Studio interface with the 'Create user' test case selected in the Tests Explorer. The main pane displays a Groovy script for creating a new user. The script follows a similar structure to the 'Login' script, using WebUI keywords to open a browser, navigate to the login page, log in, and then use the 'Create user' keyword from the Object Repository to perform the creation. The code is as follows:

```
10 import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
22
23
24 WebUI.openBrowser('')
25 WebUI.navigateToUrl('http://localhost:4200/login')
26
27
28 WebUI.setText(findTestObject('Object Repository/Page_Login/input_Email_email'), 'cboerder0@time.com')
29 WebUI.setText(findTestObject('Object Repository/Page_Login/input_Password_password'), 'zcFUrKv4liq4')
30
31
32 WebUI.click(findTestObject('Object Repository/Page_Login/button_Login'))
33
34
35 boolean isLoggedIn = WebUI.verifyElementPresent(findTestObject('Object Repository/Page_Home/label_Username'), 10)
36
37 if (isLoggedIn) {
38     println('Login was successful.')
39 } else {
40     println('Login failed.')
41 }
42
43
44 WebUI.closeBrowser()
```

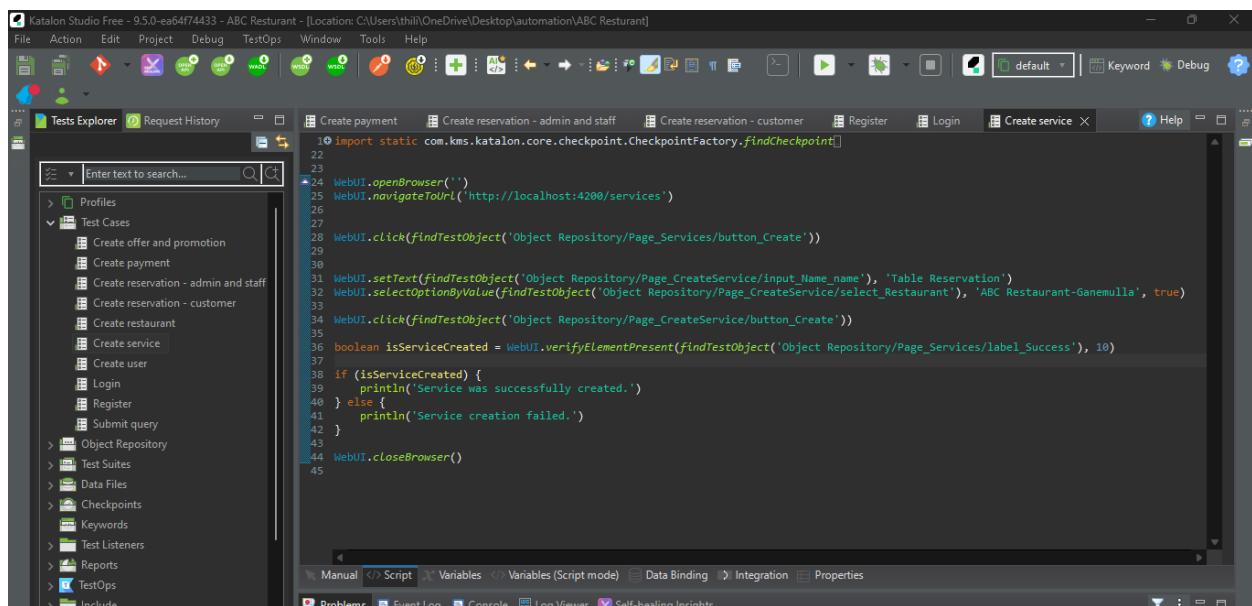
Create restaurant



The screenshot shows the Katalon Studio Free interface with a project titled "ABC Restaurant". The left sidebar displays the Test Explorer, showing a tree structure with "Profiles", "Test Cases", and various test items like "Create offer and promotion", "Create payment", etc. The main area shows a script editor with the following Java code:

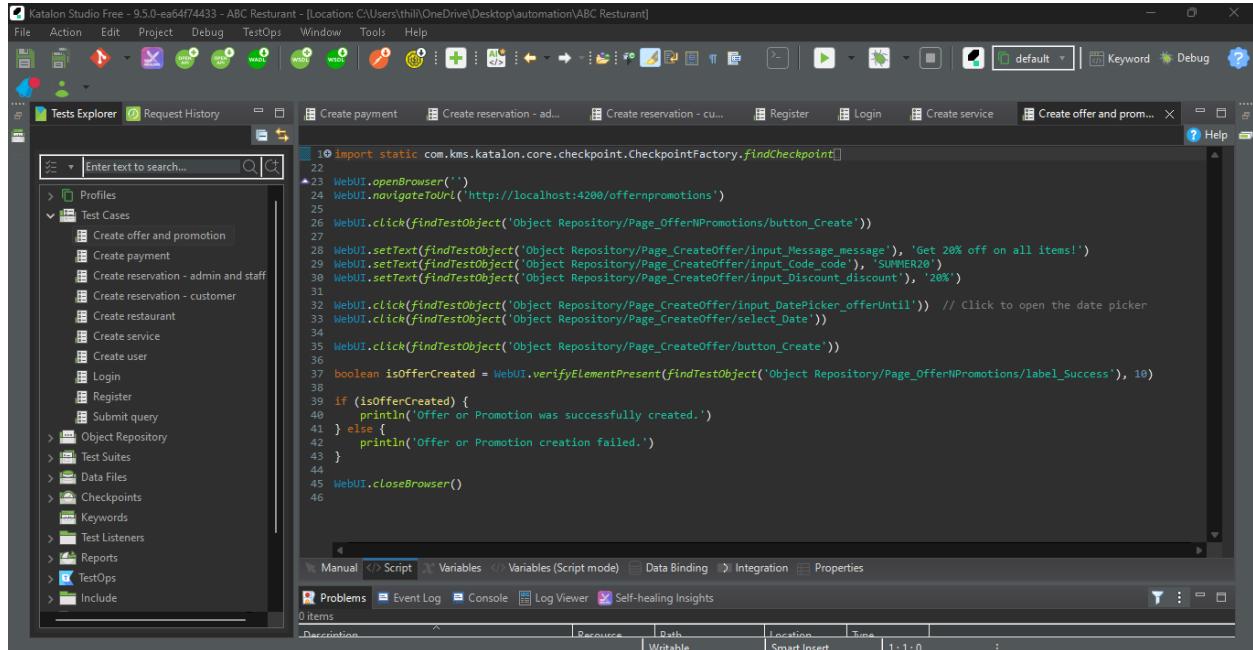
```
1@import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint()
2
3
4WebUI.openBrowser('')
5WebUI.navigateToUrl('http://localhost:4200/services')
6
7
8WebUI.click(findTestObject('Object Repository/Page_Services/button_Create'))
9
10
11WebUI.setText(findTestObject('Object Repository/Page_CreateService/input_Name_name'), 'Table Reservation')
12WebUI.selectOptionByValue(findTestObject('Object Repository/Page_CreateService/select_Restaurant'), 'ABC Restaurant-Ganemulla', true)
13
14WebUI.click(findTestObject('Object Repository/Page_CreateService/button_Create'))
15
16
17boolean isServiceCreated = WebUI.verifyElementPresent(findTestObject('Object Repository/Page_Services/label_Success'), 10)
18
19if (isServiceCreated) {
20    println('Service was successfully created.')
21} else {
22    println('Service creation failed.')
23}
24
25WebUI.closeBrowser()
```

Create service



The screenshot shows the Katalon Studio Free interface with the same project "ABC Restaurant". The left sidebar shows the Test Explorer with the "Create service" item selected. The main area shows the same Java script as the previous screenshot, demonstrating how to create a service named "Table Reservation" under the "ABC Restaurant-Ganemulla" category.

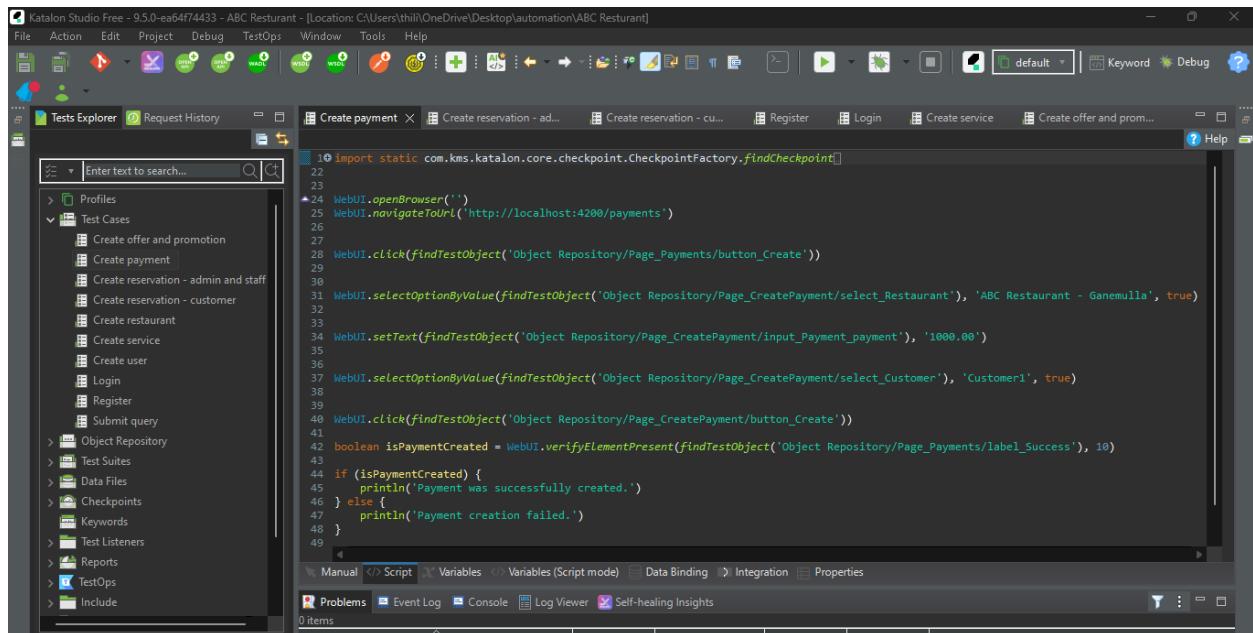
Create offer and promotion



The screenshot shows the Katalon Studio Free interface with the title bar "Katalon Studio Free - 9.5.0-ea64f74433 - ABC Restaurant - [Location: C:\Users\thii\OneDrive\Desktop\automation\ABC Restaurant]". The main area displays a Java script test case for creating an offer or promotion. The code uses WebUI methods to open a browser, navigate to a specific URL, click on a create button, set text values for message, code, and discount, and verify if the offer was created successfully. The test case is part of a project named "ABC Restaurant" located at "C:\Users\thii\OneDrive\Desktop\automation\ABC Restaurant".

```
10 import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint()
11
12 WebUI.openBrowser('')
13 WebUI.navigateToUrl('http://localhost:4200/offernpromotions')
14
15 WebUI.click(findTestObject('Object Repository/Page_OfferNPromotions/button_Create'))
16
17 WebUI.setText(findTestObject('Object Repository/Page_CreateOffer/input_Message_message'), 'Get 20% off on all items!')
18 WebUI.setText(findTestObject('Object Repository/Page_CreateOffer/input_Code_code'), 'SUMMER20')
19 WebUI.setText(findTestObject('Object Repository/Page_CreateOffer/input_Discount_discount'), '20%')
20
21 WebUI.click(findTestObject('Object Repository/Page_CreateOffer/input_DatePicker_offerUntil')) // Click to open the date picker
22 WebUI.click(findTestObject('Object Repository/Page_CreateOffer/select_Date'))
23
24 WebUI.click(findTestObject('Object Repository/Page_CreateOffer/button_Create'))
25
26 boolean isOfferCreated = WebUI.verifyElementPresent(findTestObject('Object Repository/Page_OfferNPromotions/label_Success'), 10)
27
28 if (isOfferCreated) {
29     println('Offer or Promotion was successfully created.')
30 } else {
31     println('Offer or Promotion creation failed.')
32 }
33
34 WebUI.closeBrowser()
```

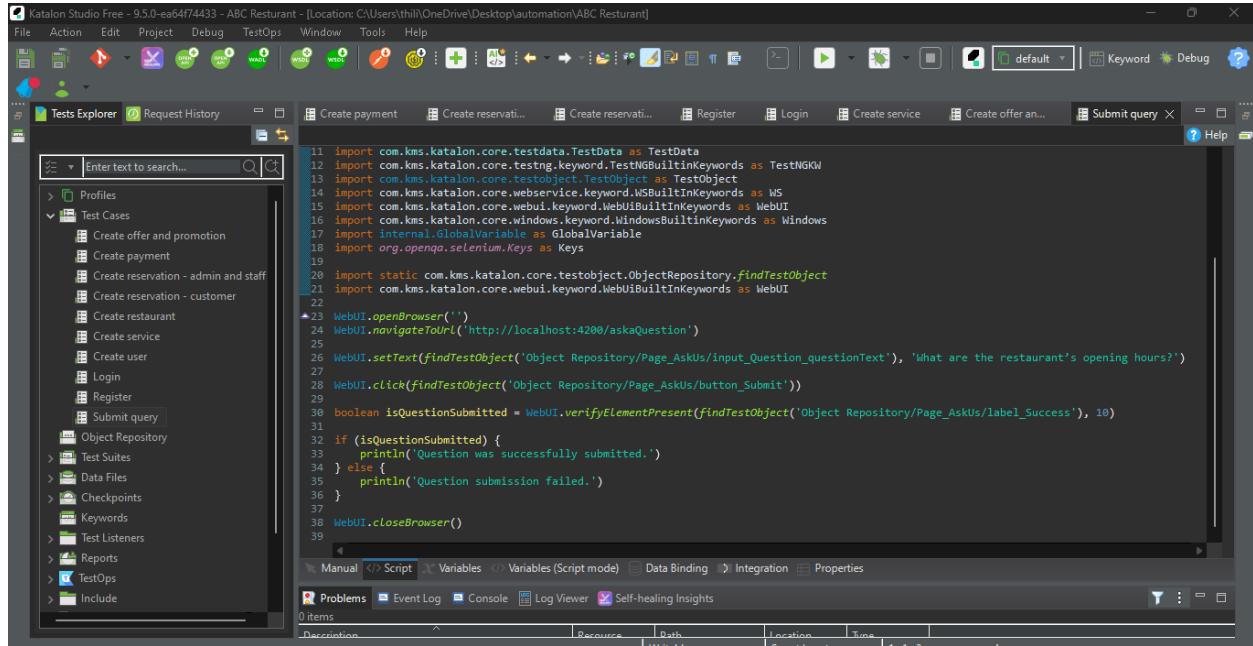
Create payment



The screenshot shows the Katalon Studio Free interface with the title bar "Katalon Studio Free - 9.5.0-ea64f74433 - ABC Restaurant - [Location: C:\Users\thii\OneDrive\Desktop\automation\ABC Restaurant]". The main area displays a Java script test case for creating a payment. The code uses WebUI methods to open a browser, navigate to a specific URL, click on a create button, select a restaurant, set a payment amount, select a customer, and verify if the payment was created successfully. The test case is part of a project named "ABC Restaurant" located at "C:\Users\thii\OneDrive\Desktop\automation\ABC Restaurant".

```
10 import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint()
11
12 WebUI.openBrowser('')
13 WebUI.navigateToUrl('http://localhost:4200/payments')
14
15 WebUI.click(findTestObject('Object Repository/Page_Payments/button_Create'))
16
17 WebUI.selectOptionByValue(findTestObject('Object Repository/Page_CreatePayment/select_Restaurant'), 'ABC Restaurant - Ganemulla', true)
18
19 WebUI.setText(findTestObject('Object Repository/Page_CreatePayment/input_Payment_payment'), '1000.00')
20
21 WebUI.selectOptionByValue(findTestObject('Object Repository/Page_CreatePayment/select_Customer'), 'Customer1', true)
22
23 WebUI.click(findTestObject('Object Repository/Page_CreatePayment/button_Create'))
24
25 boolean isPaymentCreated = WebUI.verifyElementPresent(findTestObject('Object Repository/Page_Payments/label_Success'), 10)
26
27 if (isPaymentCreated) {
28     println('Payment was successfully created.')
29 } else {
30     println('Payment creation failed.')
31 }
```

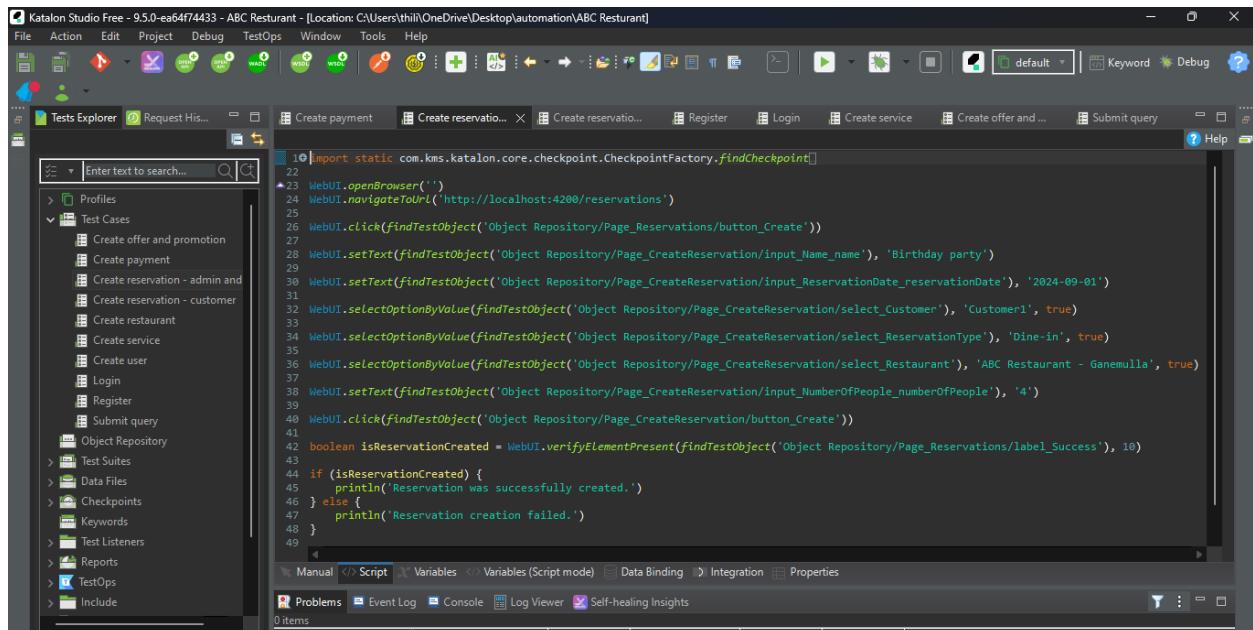
Submit query



The screenshot shows the Katalon Studio interface with a test script titled "Submit query". The script uses WebUI keywords to open a browser, navigate to a URL, set text in an input field, click a button, and verify if the submission was successful. The code is as follows:

```
11 import com.kms.katalon.coretestdata.TestData as TestData
12 import com.kms.katalon.core.testng.keyword.TestNGBuiltInKeywords as TestNGKW
13 import com.kms.katalon.core.testobject.TestObject as TestObject
14 import com.kms.katalon.core.webservice.keyword.WebUIBuiltInKeywords as WS
15 import com.kms.katalon.core.webui.keyword.WebUIBuiltInKeywords as WebUI
16 import com.kms.katalon.core.windows.keyword.WindowsBuiltInKeywords as Windows
17 import internal.GlobalVariable as GlobalVariable
18 import org.openqa.selenium.Keys as Keys
19
20 import static com.kms.katalon.core.testobject.ObjectRepository.findTestObject
21 import com.kms.katalon.core.webui.keyword.WebUIBuiltInKeywords as WebUI
22
23 WebUI.openBrowser('')
24 WebUI.navigateToUrl('http://localhost:4200/askaQuestion')
25
26 WebUI.setText(findTestObject('Object Repository/Page_AskUs/input_Question_questionText'), 'What are the restaurant's opening hours?')
27
28 WebUI.click(findTestObject('Object Repository/Page_AskUs/button_Submit'))
29
30 boolean isQuestionSubmitted = WebUI.verifyElementPresent(findTestObject('Object Repository/Page_AskUs/label_Success'), 10)
31
32 if (isQuestionSubmitted) {
33     println('Question was successfully submitted.')
34 } else {
35     println('Question submission failed.')
36 }
37
38 WebUI.closeBrowser()
39
```

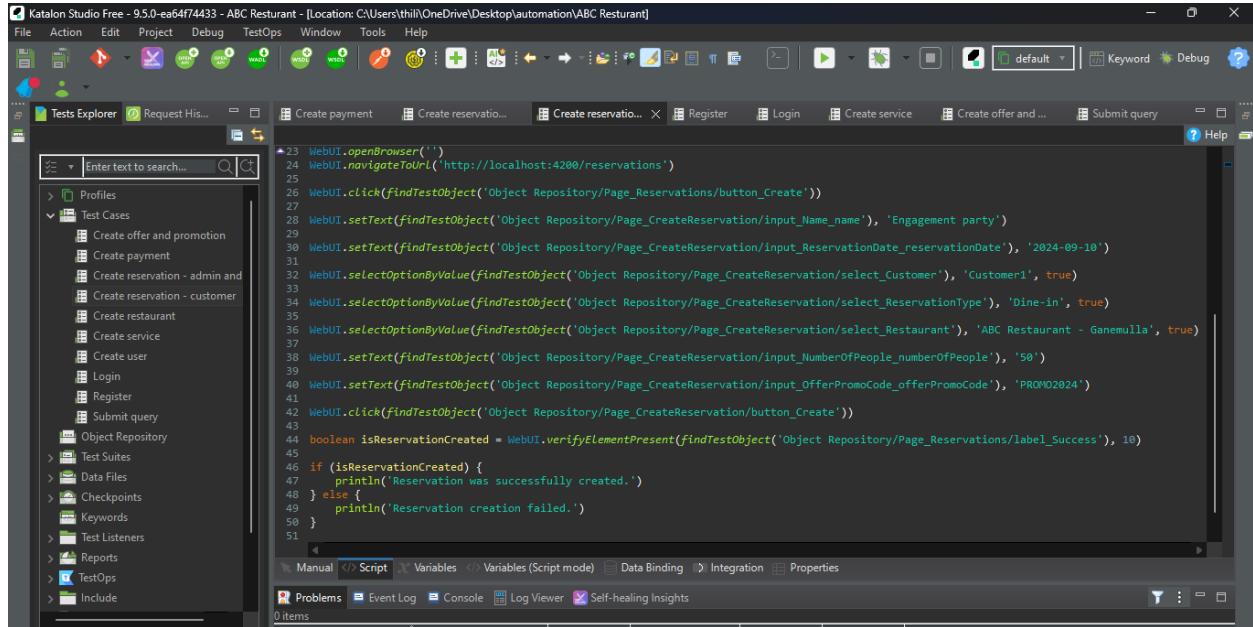
Create reservation – admin, staff



The screenshot shows the Katalon Studio interface with a test script titled "Create reservation – admin, staff". The script uses WebUI keywords to open a browser, navigate to a URL, click a create button, enter reservation details, and verify if the creation was successful. The code is as follows:

```
10 import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
22
23 WebUI.openBrowser('')
24 WebUI.navigateToUrl('http://localhost:4200/reservations')
25
26 WebUI.click(findTestObject('Object Repository/Page_Reservations/button_Create'))
27
28 WebUI.setText(findTestObject('Object Repository/Page_CreateReservation/input_Name_name'), 'Birthday party')
29
30 WebUI.setText(findTestObject('Object Repository/Page_CreateReservation/input_ReservationDate_reservationDate'), '2024-09-01')
31
32 WebUI.selectOptionByValue(findTestObject('Object Repository/Page_CreateReservation/select_Customer'), 'Customer1', true)
33
34 WebUI.selectOptionByValue(findTestObject('Object Repository/Page_CreateReservation/select_ReservationType'), 'Dine-in', true)
35
36 WebUI.selectOptionByValue(findTestObject('Object Repository/Page_CreateReservation/select_Restaurant'), 'ABC Restaurant - Ganemulla', true)
37
38 WebUI.setText(findTestObject('Object Repository/Page_CreateReservation/input_NumberOfPeople_numberOfPeople'), '4')
39
40 WebUI.click(findTestObject('Object Repository/Page_CreateReservation/button_Create'))
41
42 boolean isReservationCreated = WebUI.verifyElementPresent(findTestObject('Object Repository/Page_Reservations/label_Success'), 10)
43
44 if (isReservationCreated) {
45     println('Reservation was successfully created.')
46 } else {
47     println('Reservation creation failed.')
48 }
49
```

Create reservation - customer



The screenshot shows the Katalon Studio Free interface. The left sidebar displays a tree view of test cases, including 'Create reservation - customer'. The main area shows a script editor with the following code:

```
23 WebUI.openBrowser('')
24 WebUI.navigateToUrl('http://localhost:4200/reservations')
25
26 WebUI.click(findTestObject('Object Repository/Page_Reservations/button_Create'))
27
28 WebUI.setText(findTestObject('Object Repository/Page_CreateReservation/input_Name_name'), 'Engagement party')
29
30 WebUI.setText(findTestObject('Object Repository/Page_CreateReservation/input_ReservationDate_reservationDate'), '2024-09-10')
31
32 WebUI.selectOptionByValue(findTestObject('Object Repository/Page_CreateReservation/select_Customer'), 'Customer1', true)
33
34 WebUI.selectOptionByValue(findTestObject('Object Repository/Page_CreateReservation/select_ReservationType'), 'Dine-in', true)
35
36 WebUI.selectOptionByValue(findTestObject('Object Repository/Page_CreateReservation/select_Restaurant'), 'ABC Restaurant - Ganemulla', true)
37
38 WebUI.setText(findTestObject('Object Repository/Page_CreateReservation/input_NumberOfPeople_numberOfPeople'), '50')
39
40 WebUI.setText(findTestObject('Object Repository/Page_CreateReservation/input_OfferPromoCode_offerPromoCode'), 'PRONO2024')
41
42 WebUI.click(findTestObject('Object Repository/Page_CreateReservation/button_Create'))
43
44 boolean isReservationCreated = WebUI.verifyElementPresent(findTestObject('Object Repository/Page_Reservations/label_Success'), 10)
45
46 if (isReservationCreated) {
47     println('Reservation was successfully created.')
48 } else {
49     println('Reservation creation failed.')
50 }
51
```

TASK D

Repository Management

Git and Github were used for version control.

GitHub repository (including report, UML diagrams) –

<https://github.com/Thilini-Perera/ABC-Restaurant>

GitHub repository for automation –

https://github.com/Thilini-Perera/ABC_Restaurant---automation

Local Git repository

```
MINGW64:/c/Users/thili/OneDrive/Desktop/ABCRestaurant
thili@Thilini_Perera MINGW64 ~/OneDrive
$ cd Desktop
thili@Thilini_Perera MINGW64 ~/OneDrive/Desktop
$ cd ABCRestaurant
thili@Thilini_Perera MINGW64 ~/OneDrive/Desktop/ABCRestaurant (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.
nothing to commit, working tree clean

thili@Thilini_Perera MINGW64 ~/OneDrive/Desktop/ABCRestaurant (main)
$ git branch
* main

thili@Thilini_Perera MINGW64 ~/OneDrive/Desktop/ABCRestaurant (main)
$ git log
commit cce900c9431429dbaa3f12f3634dc29d0bed6838 (HEAD -> main, origin/main)
Merge: 5a47b7e ae38d43
Author: Thilini Perera <thiliniruwanthi.perera@gmail.com>
Date:   Sat Aug 31 16:23:00 2024 +0530

  Merge branch 'main' of https://github.com/Thilini-Perera/ABC-Restaurant

commit 5a47b7eefdeb9114e1163c70d18ed3189bb9c25a
Author: Thilini Perera <thiliniruwanthi.perera@gmail.com>
Date:   Sat Aug 31 16:20:47 2024 +0530

  committing project documents folder

commit ae38d431f24e144d518b9f78e63d0535664e1669
Author: Thilini Perera <75115428+Thilini-Perera@users.noreply.github.com>
Date:   Sat Aug 31 10:49:55 2024 +0530

  Update README.md

commit 7f0912777ed82daf1d2325c5881ce61addaa6c38
Merge: b9e362d 29fb105
Author: Thilini Perera <thiliniruwanthi.perera@gmail.com>
Date:   Fri Aug 30 16:47:43 2024 +0530

  Merge branch 'main' of https://github.com/Thilini-Perera/ABC-Restaurant

commit b9e362deec291053a7f747162e53eaf807bd3be4
Author: Thilini Perera <thiliniruwanthi.perera@gmail.com>
Date:   Fri Aug 30 16:46:06 2024 +0530
```

```
MINGW64:/c/Users/thili/OneDrive/Desktop/ABCRestaurant
commit 5a47b7eefdeb9114e1163c70d18ed3189bb9c25a
Author: Thilini Perera <thiliniruwanthi.perera@gmail.com>
Date:   Sat Aug 31 16:20:47 2024 +0530

    committing project documents folder

commit ae38d431f24e144d518b9f78e63d0535664e1669
Author: Thilini Perera <75115428+Thilini-Perera@users.noreply.github.com>
Date:   Sat Aug 31 10:49:55 2024 +0530

    Update README.md

commit 7f0912777ed82daf1d2325c5881ce61addaa6c38
Merge: b9e362d 29fb105
Author: Thilini Perera <thiliniruwanthi.perera@gmail.com>
Date:   Fri Aug 30 16:47:43 2024 +0530

    Merge branch 'main' of https://github.com/Thilini-Perera/ABC-Restaurant

commit b9e362deec291053a7f747162e53eaf807bd3be4
Author: Thilini Perera <thiliniruwanthi.perera@gmail.com>
Date:   Fri Aug 30 16:46:06 2024 +0530

    changing report name to query report

commit 29fb105ef30128dc5d820c8d623005feb4fb90e0
Author: Thilini Perera <75115428+Thilini-Perera@users.noreply.github.com>
Date:   Fri Aug 30 16:21:50 2024 +0530

    committing Restuarant.cs

commit f0f22926aba3dba6f51fbf966fd1dccf01991eeb
Author: Thilini Perera <thiliniruwanthi.perera@gmail.com>
Date:   Fri Aug 30 15:58:09 2024 +0530

    committing weatherforecast.cs

commit 9151d95e73d4e649f359dca5f2ed7254db3fea11
Author: Thilini Perera <thiliniruwanthi.perera@gmail.com>
Date:   Fri Aug 30 15:56:06 2024 +0530
```

```

MINGW64:/c/Users/thili/OneDrive/Desktop/ABCRestaurant
Date: Fri Aug 30 15:56:06 2024 +0530

thili@Thilini_Perera MINGW64 ~/OneDrive/Desktop/ABCRestaurant (main)
$ git log --oneline
cce900c (HEAD -> main, origin/main) Merge branch 'main' of https://github.com/Thilini-Perera/ABC-Restaurant
5a47b7e committing project documents folder
ae38d43 Update README.md
7f09127 Merge branch 'main' of https://github.com/Thilini-Perera/ABC-Restaurant
b9e362d changing report name to query report
29fb105 committing Restuarant.cs
f0f2292 committing weatherforecast.cs
9151d95 committing wwwroot folder
1f5de9e adding extra files
cc2b22b Committing UI folder - commit 1
c020c83 committing images and ViewModels
5eb464d committing user activity, users, weather forecast controllers
a08ae26 committing controllers
a8d99ff committing persistence folder
2aa3639 committing migrations
831040f committing models
142e8cc committing payment.cs
8a5ded8 committing models - commit 1
09e9ddc core config files
c73fe03 Merge branch 'main' of https://github.com/Thilini-Perera/ABC-Restaurant
6e0b2fc Initial commit
4a22de2 initial project setup

thili@Thilini_Perera MINGW64 ~/OneDrive/Desktop/ABCRestaurant (main)
$ git branch
* main

thili@Thilini_Perera MINGW64 ~/OneDrive/Desktop/ABCRestaurant (main)
$ git branch -r
origin/main

thili@Thilini_Perera MINGW64 ~/OneDrive/Desktop/ABCRestaurant (main)
$ git remote -v
origin https://github.com/Thilini-Perera/ABC-Restaurant.git (fetch)
origin https://github.com/Thilini-Perera/ABC-Restaurant.git (push)

thili@Thilini_Perera MINGW64 ~/OneDrive/Desktop/ABCRestaurant (main)
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl

```

```
MINGW64:/c/Users/thili/OneDrive/Desktop/ABCRestaurant
origin https://github.com/Thilini-Perera/ABC-Restaurant.git (push)

thili@Thilini_Perera MINGW64 ~/OneDrive/Desktop/ABCRestaurant (main)
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=Thilini Perera
user.email=thiliniruwanthi.perera@gmail.com
core.editor="C:\Users\thili\AppData\Local\Programs\Microsoft VS Code\bin\code" --wait
core.autocrlf=true
init.default=branch
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
remote.origin.url=https://github.com/Thilini-Perera/ABC-Restaurant.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.main.remote=origin
branch.main.merge=refs/heads/main

thili@Thilini_Perera MINGW64 ~/OneDrive/Desktop/ABCRestaurant (main)
$ git remote show origin
* remote origin
  Fetch URL: https://github.com/Thilini-Perera/ABC-Restaurant.git
  Push URL: https://github.com/Thilini-Perera/ABC-Restaurant.git
  HEAD branch: main
  Remote branch:
    main tracked
  Local branch configured for 'git pull':
    main merges with remote main
  Local ref configured for 'git push':
```

GitHub repository

The screenshot shows the GitHub repository page for `Thilini-Perera/ABC-Restaurant`. The repository is public and has 22 commits. The main branch is `main`, with 1 branch and 0 tags. The repository description is "RESTful API for ABC Restaurant - Uswatta Liyanage Thilini Ruwanthi Perera (st20315315)". The commit history includes changes like "changing report name to query report" and "adding extra files". The sidebar on the right provides information about the repository, including activity, releases, and packages.

The screenshot shows the GitHub repository page for `Thilini-Perera/ABC-Restaurant` with the "Code" tab selected. The repository contains several files and folders, including `UI/ABCRestaurantUI`, `ViewModels`, `images`, and `wwwroot/images`. The commit history shows initial project setup and configuration files like `.gitignore`, `ABCRepository.csproj`, and `ABCRepository.csproj.user`. The sidebar on the right displays language statistics (C# 40.7%, TypeScript 34.1%, HTML 23.1%, CSS 2.1%) and suggested workflows for .NET, Datadog Synthetics, and SLSA Generic generator.

Commits - Thilini-Perera/ABC-R

github.com/Thilini-Perera/ABC-Restaurant/commits/main

Thilini-Perera / ABC-Restaurant

Type to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Commits

main

All users All time

Commits on Aug 31, 2024

Merge branch 'main' of https://github.com/Thilini-Perera/ABC-Restaurant
Thilini-Perera committed 19 hours ago

committing project documents folder
Thilini-Perera committed 19 hours ago

Update README.md
Thilini-Perera committed yesterday

Commits on Aug 30, 2024

Merge branch 'main' of https://github.com/Thilini-Perera/ABC-Restaurant
Thilini-Perera committed 2 days ago

changing report name to query report

Commits - Thilini-Perera/ABC-R

github.com/Thilini-Perera/ABC-Restaurant/commits/main

Thilini-Perera / ABC-Restaurant

Type to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Commits

changing report name to query report
Thilini-Perera committed 2 days ago

committing Restaurant.cs
Thilini-Perera committed 2 days ago

committing weatherforecast.cs
Thilini-Perera committed 2 days ago

committing wwwroot folder
Thilini-Perera committed 2 days ago

adding extra files
Thilini-Perera committed 2 days ago

Committing UI folder - commit 1
Thilini-Perera committed 2 days ago

Commits on Aug 29, 2024

committing images and ViewModels
Thilini-Perera committed 3 days ago

Commits on Aug 28, 2024

committing user activity, users, weather forecast controllers
Thilini-Perera committed 4 days ago

The screenshot shows two GitHub commit history pages for the repository `Thilini-Perera/ABC-Restaurant`.

Top Commit History:

- committing user activity, users, weather forecast controllers** (Thilini-Perera committed 4 days ago) - SHA: 5eb464d
- committing controllers** (Thilini-Perera committed 4 days ago) - SHA: a08ae26
- > Commits on Aug 27, 2024
- committing persistence folder** (Thilini-Perera committed 5 days ago) - SHA: a8d99ff
- committing migrations** (Thilini-Perera committed 5 days ago) - SHA: 2aa3639
- > Commits on Aug 26, 2024
- committing models** (Thilini-Perera committed last week) - SHA: 831040f
- committing payment.cs** (Thilini-Perera committed last week) - SHA: 142e8cc
- committing models - commit 1** (Thilini-Perera committed last week) - SHA: 8a5ded8
- core config files** (Thilini-Perera committed last week) - SHA: 09e9ddc

Bottom Commit History:

- > Commits on Aug 26, 2024
- committing models** (Thilini-Perera committed last week) - SHA: 831040f
- committing payment.cs** (Thilini-Perera committed last week) - SHA: 142e8cc
- committing models - commit 1** (Thilini-Perera committed last week) - SHA: 8a5ded8
- core config files** (Thilini-Perera committed last week) - SHA: 09e9ddc
- Merge branch 'main' of https://github.com/Thilini-Perera/ABC-Restaurant** (Thilini-Perera committed last week) - SHA: c73fe03
- Initial commit** (Thilini-Perera committed last week) - Verified - SHA: 6e0b2fc
- initial project setup** (Thilini-Perera committed last week) - SHA: 4a22de2

GitHub repository for test automation

The screenshot shows the GitHub repository page for `ABC_Restaurant---automation`. The repository is public and has 4 commits from `Thilini-Perera`. The commit history shows several commits to various files including `.settings`, `Include/config`, `Profiles`, `Scripts`, `Test Cases`, `settings`, `.gitignore`, and `ABC Restaurant.prj`, all committed 2 days ago. The repository has 0 stars, 1 watching, and 0 forks. The `About` section describes the repository as including test automation code base of ABC restaurant RESTful API - Uswatta Liyanage Thilini Ruwanthi Perera (st20315315). The `Releases` section indicates no releases published and provides a link to [Create a new release](#).

Code

ABC_Restaurant---automation Public

main 1 Branch 0 Tags

Go to file Add file <> Code

About

This repository includes test automation code base of ABC restaurant RESTful API - Uswatta Liyanage Thilini Ruwanthi Perera (st20315315)

Readme Activity 0 stars 1 watching 0 forks

Releases

No releases published [Create a new release](#)

Packages

No packages published [Publish your first package](#)

Languages

Groovy 100.0%

Suggested workflows

Based on your tech stack

Java with Gradle Configure Build and test a Java project using a Gradle wrapper script.

README

ABC_Restaurant---automation by Uswatta Liyanage Thilini Ruwanthi Perera (st20315315)

This repository includes test automation code base of ABC restaurant RESTful API