



Jahezli Application

CPIT-251 | GROUP 4

Prepared By:
Abeer Abdullah
Jana Kurdi
Renad Ghalib
Razan Alshaikh
Rawan Alzahrany

ID:
1905681
1906167
1908460
1906813
1905271

Prepared For :
I. Hind Hamdi

Section:
DAR



Table of Contents

INTRODUCTION	4
PROBLEM STATEMENT.....	4
SOLUTION	5
APPROACH.....	5
REQUIREMENTS	5
FUNCTIONAL REQUIREMENTS:	5
NON-FUNCTIONAL REQUIREMENTS:.....	5
SCHEDULE FOR THE ACTIVITIES OF THE PROJECT	6
USE CASE DIAGRAM	7
USE CASES DESCRIPTIONS	8
CLASS DIAGRAM:	14
SEQUENCE DIAGRAM:.....	15
ACTIVITY DIAGRAM	20
IMPLEMENTATION	21
LOGIN:	21
REGISTRATION:	21
.....	22
RESERVE TABLE:.....	23
.....	23
MODIFY RESERVATION:	23
CANCEL RESERVATION:	24
<i>DISPLAY RESERVATION:</i>	24
MAKE PAYMENT:.....	25
PAY BY CREDIT CARD:	26
PAY BY CASH:	26
REVIEW:	27
FEEDBACK:	28
TEST	28
CHALLENGES:	29
FUTURE WORK:.....	29
CONCLUSION	29

Table of figures and Tables

Table 1 Login use case description	8
Table 2 Search for table use case	9
Table 3 reserve table use case	10
Table 4 display reservation use case	11
Table 5 Make payment use case.....	12
Table 6 review and read feedback use case	13
Figure 1 MS project schedule(1)	6
Figure 2 MS project schedule (2)	6
Figure 3 use case diagram	7
Figure 4 Class Diagram	14
Figure 5 Search for a table sequence diagram	15
Figure 6 Reserve table sequence diagram.....	16
Figure 7 Display reservation sequence diagram	17
Figure 8 Make payment sequence diagram	18
Figure 9 Review and read feedback sequence diagram	19
Figure 10 Activity diagram	20
Figure 11 Login source code.....	21
Figure 12 Registration source code (1)	21
Figure 13 Registration source code (2)	22
Figure 14 Registration source code (3)	22
Figure 15 Reserve table source code	23
Figure 16 modify reservation	23
Figure 17 cancel reservation	24
Figure 18 Display reservation source code.....	24
Figure 19 make payment source code.....	25
Figure 20 pay by credit card source code part2	26
Figure 21 pay by credit card source code part1	26
Figure 22 pay by cash source code	26
Figure 23 Review source code (1)	27
Figure 24 Review source code (2)	27
Figure 25 Feedback source code.....	28
Figure 26 Test	28

Abstract

In our daily lives dealing in various fields have become easy. So, on many occasions, we need applications and sites that make it easier for us to search and reserve tables at restaurants that we need. Because of this, the idea of our system came, which is that customers can search and choose what they want through the app, and the table will be equipped with various payment methods to avoid losing time.

The system we have chosen is the Jahezli Application, which offers many options of restaurant tables and a variety of table types for various occasions if the person wants a casual dinner or has an important meeting

The report discusses the solution to some problems we noticed. We identified the problem statement the people face while identifying the best solutions. Also, we have outlined the methodology that we will follow which is the Waterfall methodology. Lastly, we set the requirements that must be present in this system

In the end, we achieved our goal by planning, analysing, implementing ,designing, and testing our system well and deeply by completing all the required tasks. The main points in our report for each phase will be discussed in detail.

Introduction

The application is for table reservations. This app aims to help customers reserve a suitable table before arriving at the restaurant. The application provides a variety of services. In addition to that, the customer can then pay for the service/s in different ways.

Problem statement

Nowadays, restaurants and cafes are experiencing overcrowding issues, causing customers to wait for a long time for a table, wasting the customer's time upon arrival.

Solution

Our solution is to create an application that allows customers to reserve a table before arriving. To avoid congestion and long waits as well as processing the order in advance, which helps customers not to waste their time such as important meetings held in the place. So, they can reserve and pick the tables and the orders as they want.

Approach

Waterfall methodology This method was chosen because of several reasons one of them is the application is not complicated. Also, this method focuses on determining the requirements which clarify them. Lastly, the model develops systematically from one phase to another in downward motion without going backward.

Requirements

Functional requirements:

- Search for table
- Reserve table
- Modify the reservation
- Display the reservation
- Review and feedback

Non-functional requirements:

- Availability.
- Security.
- Reability.

Schedule for the activities of the project

		Task Mode	Task Name	Duration	Start	Finish	Predecessor
1			analysis	4 days	Mon 04/04/22	Thu 07/04/22	
2			interview	2 days	Mon 04/04/22	Tue 05/04/22	
3			collection requirment	2 days	Wed 06/04/22	Thu 07/04/22	2
4			requirement documentation	0 days	Thu 07/04/22	Thu 07/04/22	3
5			design	12 days	Fri 08/04/22	Mon 25/04/22	
6			use case	2 days	Fri 08/04/22	Mon 11/04/22	
7			sequence digram	3 days	Tue 12/04/22	Thu 14/04/22	
8			activety digram	3 days	Fri 15/04/22	Tue 19/04/22	
9			class digram	4 days	Wed 20/04/22	Mon 25/04/22	
10			check digram	0 days	Mon 25/04/22	Mon 25/04/22	
11			implemntetion	8 days	Tue 26/04/22	Thu 05/05/22	
12			coding	4 days	Tue 26/04/22	Fri 29/04/22	
13			debug	4 days	Mon 02/05/22	Thu 05/05/22	12
14			code documentation	0 days	Thu 05/05/22	Thu 05/05/22	13
15			test	7 days	Fri 06/05/22	Mon 16/05/22	
16			test the code	4 days	Fri 06/05/22	Wed 11/05/22	
17			user feedback	3 days	Thu 12/05/22	Mon 16/05/22	
18			check the code	0 days	Mon 16/05/22	Mon 16/05/22	17, 16
19			maintenance	0 days	Sun 03/04/22	Sun 03/04/22	
20			new releases	0 days	Sun 03/04/22	Sun 03/04/22	

Figure 1 MS project schedule (1)

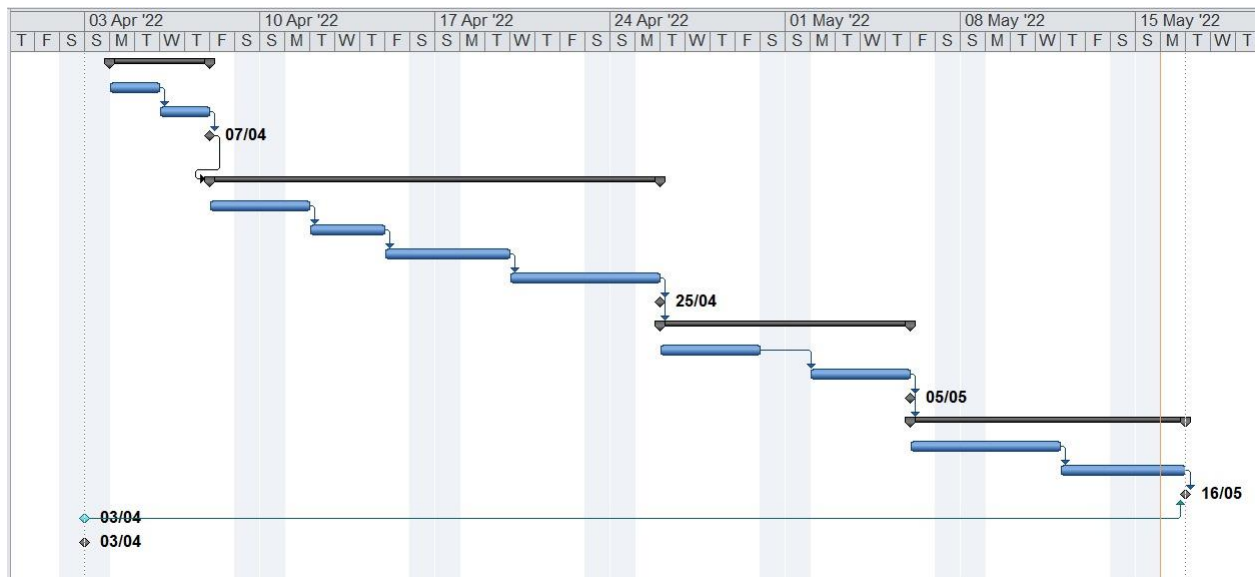


Figure 2 MS project schedule (2)

Use Case diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system.

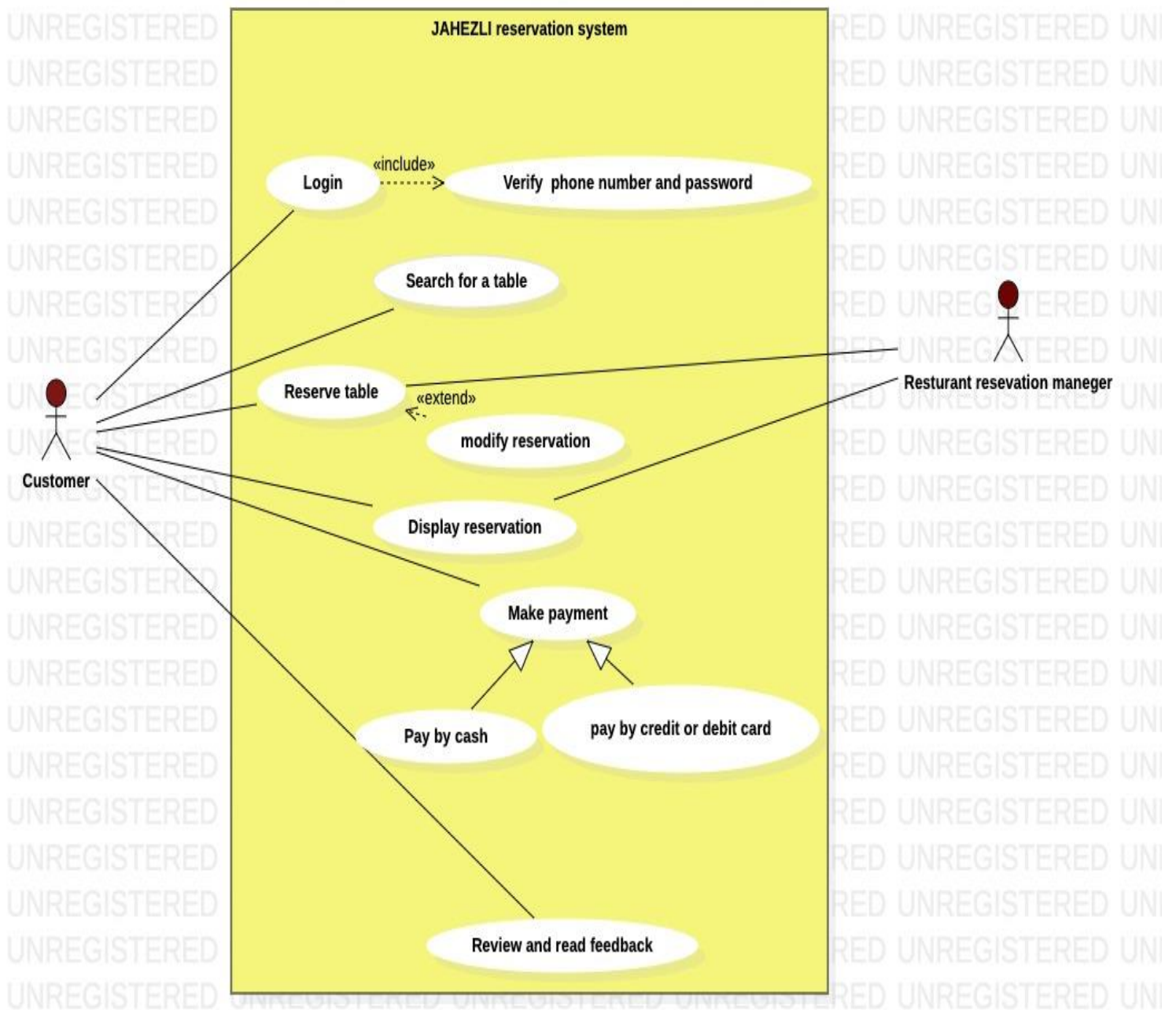


Figure 3 use case diagram

Use cases descriptions

Table1 Shows login use case in detail.

Table 1 Login use case description

Use case title: <i>Login</i>
Primary actor: Customer
Level: Kite (Summary).
Stakeholders: Customer.
Precondition: The system must connect to the internet.
Minimal Guarantee: Rollback to any uncompleted
Success Guarantee: Access to home page.
Trigger: User requests to login.
Main Success Scenario: 1. Customers must sign up for the first time. <ul style="list-style-type: none">• The customer sets a username, phone number, e-mail, and password. 2. The customer must login if it is not the first time. <ul style="list-style-type: none">• The system validates the username and password. 3. The customer successfully accesses the home page.
Extensions: 1a. Invalid username or password(or both). <ul style="list-style-type: none">1a1. The system shows an error message containing “ Please try again “.1a2. Either customer tries again or quits the application. 2a. Invalid password for three times. <ul style="list-style-type: none">2a1. System notifies the user by sending a message to the E-mail. 3a. Customer forgets the password. <ul style="list-style-type: none">3a1. Click on forget password button.3a2. Reset a new password. 4.a. Customer does not fill in the required information. <ul style="list-style-type: none">4a1. System shows an error message.

Table 2 Shows Search for table use case in detail.

Table 2 Search for table use case

Use case title: <i>Search for table</i>
Primary actor: Customer
Level: Kite (Summary).
Stakeholders: Customer, manager.
Precondition: customer must select the place, time, date, and number of people
Minimal Guarantee: Show Places Menu.
Success Guarantee: available tables displayed to customer and the customer choose table.
Trigger: choose a place and table.
Main Success Scenario: <ol style="list-style-type: none"> 1. Customer login to the account. 2. The customer opens the menu & chooses a place. 3. Customer set date, time, and the number of people. 4. The customer chooses a table after the available tables are displayed.
Extensions: <ol style="list-style-type: none"> 1a. Unable to access the account. <ol style="list-style-type: none"> 1a1. The system shows an error message containing “ Please try again “. 1a2. Either customer tries again or quits the application. 1a3. Customer reset account password, username. 2a. No places shown on the menu. <ol style="list-style-type: none"> 2a1. The system shows an error message, Customer refresh the page. 2a2. The customer quit the app. 3a. Unable to set a date, time, and the number of people. <ol style="list-style-type: none"> 3a1. The system shows an error message, Customer refresh the page. 4a. There are no tables displayed to the customer. <ol style="list-style-type: none"> 4a1. The system Shows the reason for no available tables. 4a2. The system shows the message “ Please try another date and time “. 4a3. Customer selects another date and time, if available they will reserve. 4a4. If the customer selects another date and time and no available tables shown; they will quit the application.

Table 3 Shows reserve table use case in detail.

Table 3 reserve table use case

Use case title: <i>Reserve table</i>
Primary actor: Customer
Level: Kite (Summary).
Stakeholders: Customer, Manager
Precondition: Select the table, place, date and time, number of people
Minimal Guarantee: Show the available tables.
Success Guarantee: Inform the customer that the table is available, and the reservation has been confirmed.
Trigger: The customer found the available table suitable for their reservation details.
Main Success Scenario: <ol style="list-style-type: none"> 1. Customer login to the account 2. The customer opens the menu & chooses a place. 3. Customer set date, time, and the number of people. 4. Customer search for a table 5. The customer chooses a table after the available tables are displayed. 6. New reservation created. 7. The customer confirms the reservation with a guarantee that the reservation details are appropriate. 8. The manager checks the reservation by ensuring that no one has previously booked the same table. 9. The manager confirms the reservation. 10. The system sends a confirmation and detailed information about their reservation.
Extensions: <ol style="list-style-type: none"> 1a. Unable to access the account. <ol style="list-style-type: none"> 1a1. The system shows an error message containing “ Please try again “. 1a2. Either customer tries again or quits the application. 1a3. Customer reset account password, username. 2a. Couldn't find a table <ol style="list-style-type: none"> 2a1. The system Shows the reason for no available tables 2a2. The system shows the message “ Please try another date and time “ 2a3. Customer selects another date and time, if available they will reserve. 2a4. If the customer selects another date and time and no available tables are shown; they will quit the application. 3a. No reservation created. <ol style="list-style-type: none"> 3a1. The system shows an error message. 3a2. The customer makes a new reservation. 4a. Manager cancel reservation. <ol style="list-style-type: none"> 4a1. The system shows a message containing the reason for canceling the reservation; the customer makes a new reservation. 5a. No confirmation was displayed for the customer. <ol style="list-style-type: none"> 5a1. The system shows an error message, Customer refresh page. 5a2. The customer re-checks the history of reservations in the account.

Table 4 Shows display reservation use case in detail.

Table 4 display reservation use case

Use case title: <i>Display reservation</i>
Primary actor: customer.
Level: Kite (Summary).
Stakeholders: Customer.
Precondition: customer should login.
Minimal Guarantee: display the reservation page.
Success Guarantee: inform the customer that the modification has been successful.
Trigger: customer want to modify the reservation.
Main Success Scenario: <ol style="list-style-type: none"> 1. Customer login to the account 2. Customer open menu. 3. The customer makes modifications delete or adds. 4. Display the table reservation information to the customer. 5. The customer confirms the reservation.
Extensions: <ol style="list-style-type: none"> 1a. Unable to access the account. <ol style="list-style-type: none"> 1a1. The system shows an error message containing “ Please try again “. 1a2. Either customer tries again or quits the application. 1a3. Customer reset account password, username. 2a. Couldn't open the reservation page. <ol style="list-style-type: none"> 2a1. Customer re-fresh the page. 3a. Customers do confirm the reservation after the modification. <ol style="list-style-type: none"> 3a1. The system shows a message that the customer must confirm. 4a. Require information is not correct. <ol style="list-style-type: none"> 4a1. The system shows an error message and re-enters.

Table 5 Shows Make payment use case in detail.

Table 5 Make payment use case

Use case title: <i>Make payment</i>
Primary actor: Customer.
Level: Kite (Summary).
Stakeholders: Customer.
Precondition: Customer must select the payment method.
Minimal Guarantee: Rollback to any uncompleted payment details.
Success Guarantee: Inform the customer that the payment has been successful, confirm the reservation and send invoice.
Trigger: Customer want to confirm reservation.
Main Success Scenario: <ol style="list-style-type: none"> 1. Customer login to the account 2. The customer opens the menu & chooses a place. 3. Customer set date, time, and the number of people. 4. Customer search for a table 5. The customer chooses a table after the available tables are displayed. 6. New reservation created. 7. The customer confirms the reservation with a guarantee that the reservation details are appropriate. 8. The customer selects the payment method. 9. The customer fills in all required information. 10. Show the "payment has been successful" message to the customer. 11. The manager checks the reservation by ensuring that no one has previously booked the same table. 12. The manager confirms the reservation. 13. The system sends a confirmation and detailed information about their reservation.
Extensions: <ol style="list-style-type: none"> 1a. Unable to access the account. <ol style="list-style-type: none"> 1a1. The system shows an error message containing " Please try again " . 1a2. Either customer tries again or quits the application. 1a3. Customer reset account password, username. 2a. there is no table selected. <ol style="list-style-type: none"> 2a1. The system shows an error message. 2a2. The customer must select something. 3a. No method of payment is selected. <ol style="list-style-type: none"> 3.a1. The system shows an error message. 3a2. The customer must select a method. 4a. Require information is not correct. <ol style="list-style-type: none"> 4a1. The system shows an error message and re-enters.

Table 6 Shows review and read feedback use case in detail.

Table 6 review and read feedback use case

Use case title: <i>Review and feedback.</i>
Primary actor: Customer
Level: Kite (Summary).
Stakeholders: Customer.
Precondition: Reservation via the application and visit the restaurant.
Minimal Guarantee: Display the review page.
Success Guarantee: Write the review on the review page.
Trigger: The customer wants to read the reviews or write their feedback.
Main Success Scenario: <ol style="list-style-type: none"> 1. Customer login to the account 2. The customer opens the menu & chooses a place. 3. The customer can read the reviews. 4. After the system sends a confirmation and detailed information about the reservation the customer writes opinion and experience.
Extensions: <ol style="list-style-type: none"> 1a. Unable to access the account. <ol style="list-style-type: none"> 1a1. The system shows an error message containing “ Please try again “. 1a2. Either customer tries again or quits the application. 1a3. Customer reset account password, username. 2a. The review page does not appear to the customer. <ol style="list-style-type: none"> 2a1. The customer must select a place. 3a. The feedback page does not appear to the customer. <ol style="list-style-type: none"> 3a1. The customer must confirm the reservation.

Class diagram:

Use to represent classes, attributes, methods, and relationship between classes.

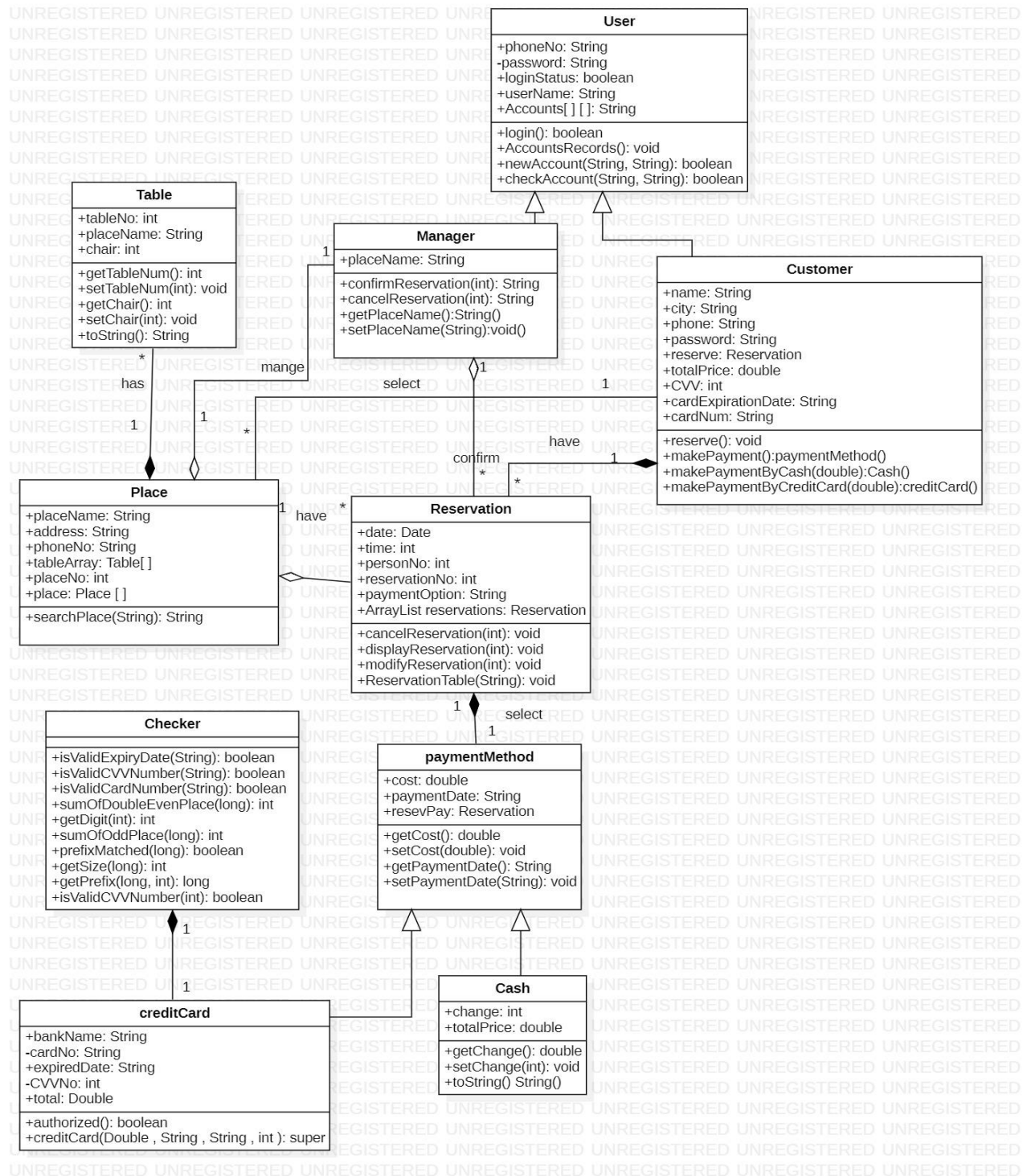


Figure 4 Class Diagram

Sequence diagram:

Sequence diagram used to depict the interaction among objects during a period of time.

I. Search for a table sequence diagram

This diagram shows the order of operation when customer search a table to reserve.

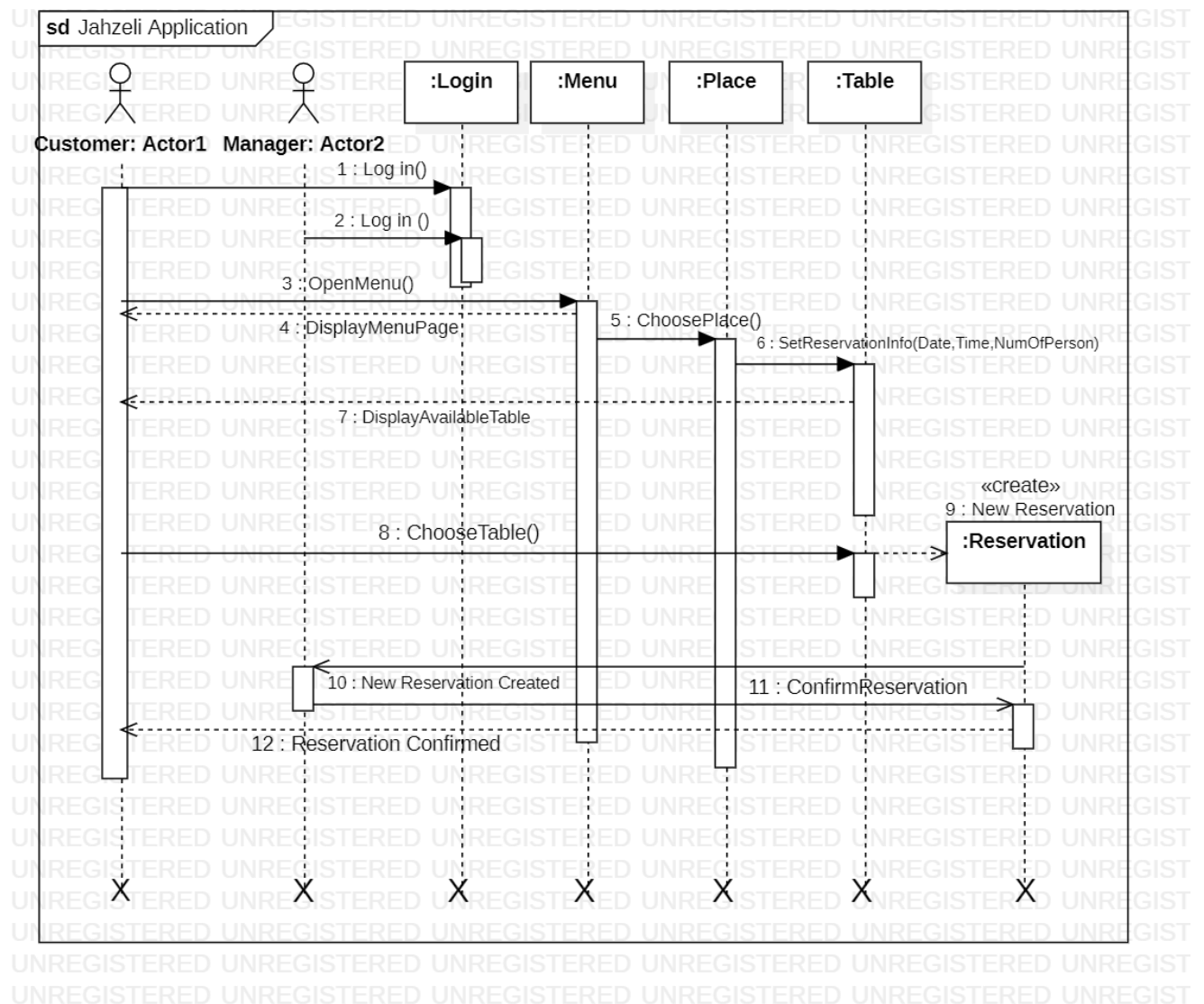


Figure 5 Search for a table sequence diagram

II. Reserve table sequence diagram

This diagram shows the operations when a customer reserve table.

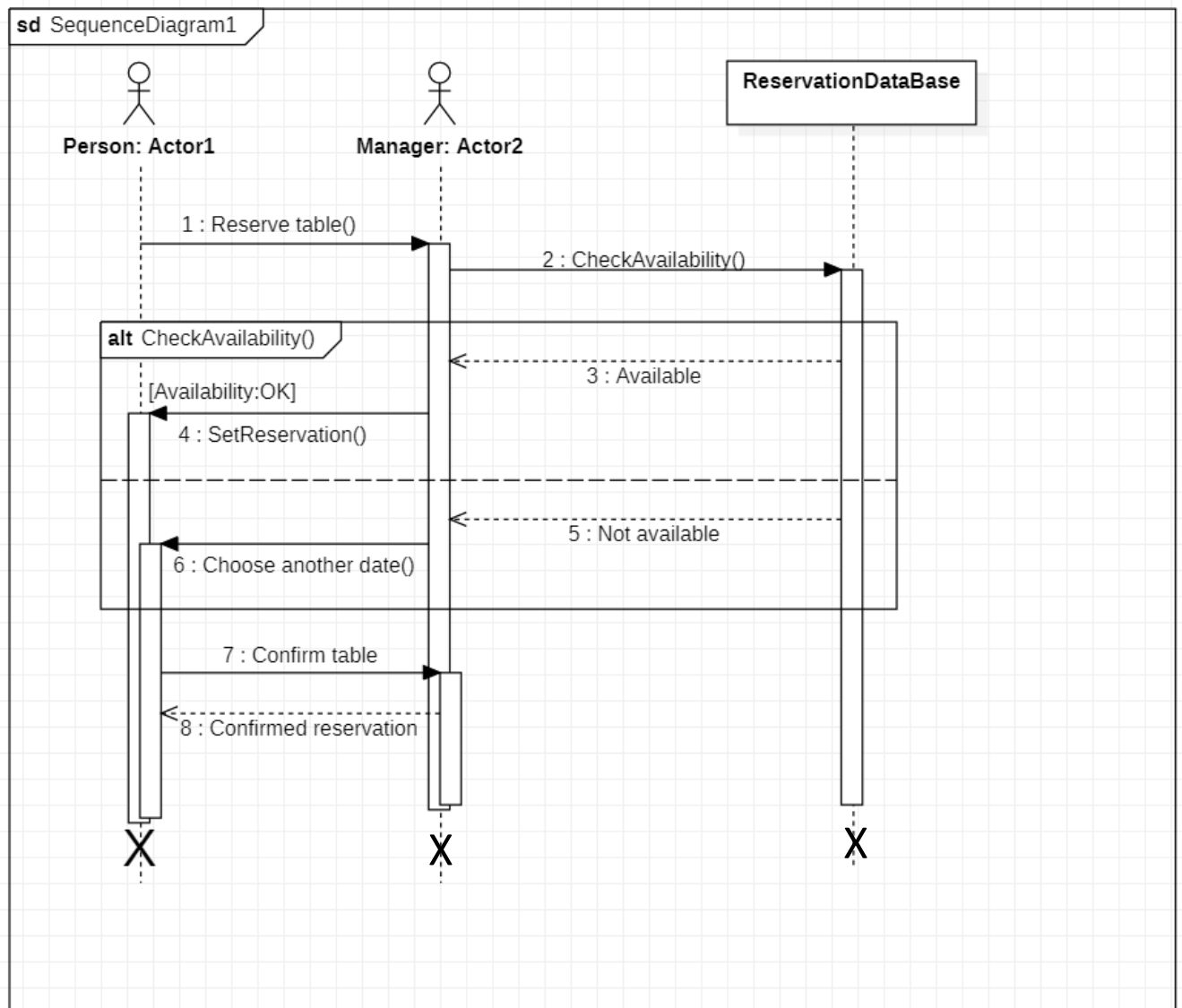


Figure 6 Reserve table sequence diagram

III. Display reservation sequence diagram

This diagram shows the order of the operations when a customer display reservation for a table

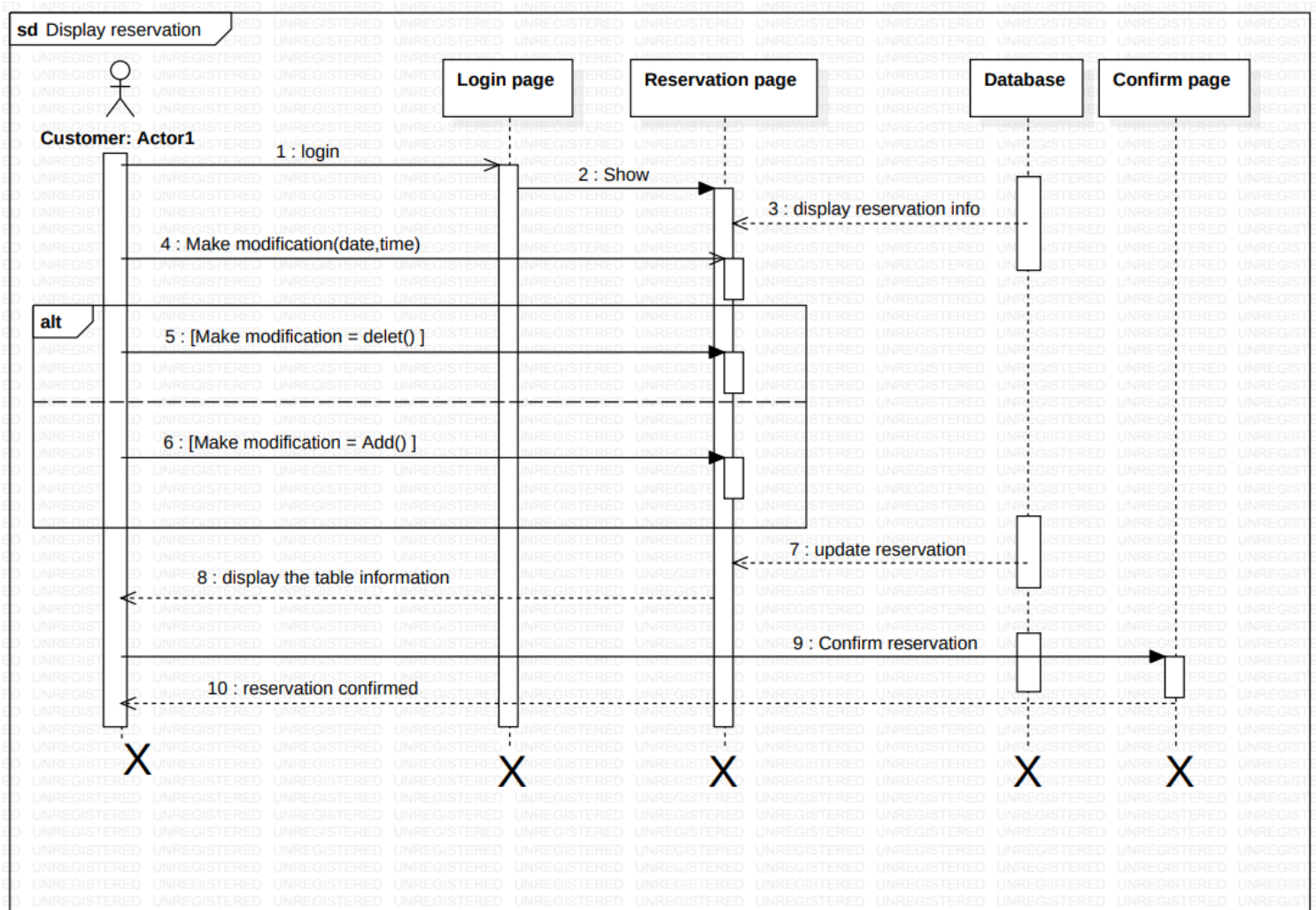


Figure 7 Display reservation sequence diagram

IV. Make payment sequence diagram

This diagram shows the order of the operations when a customer pay for reserved a table.

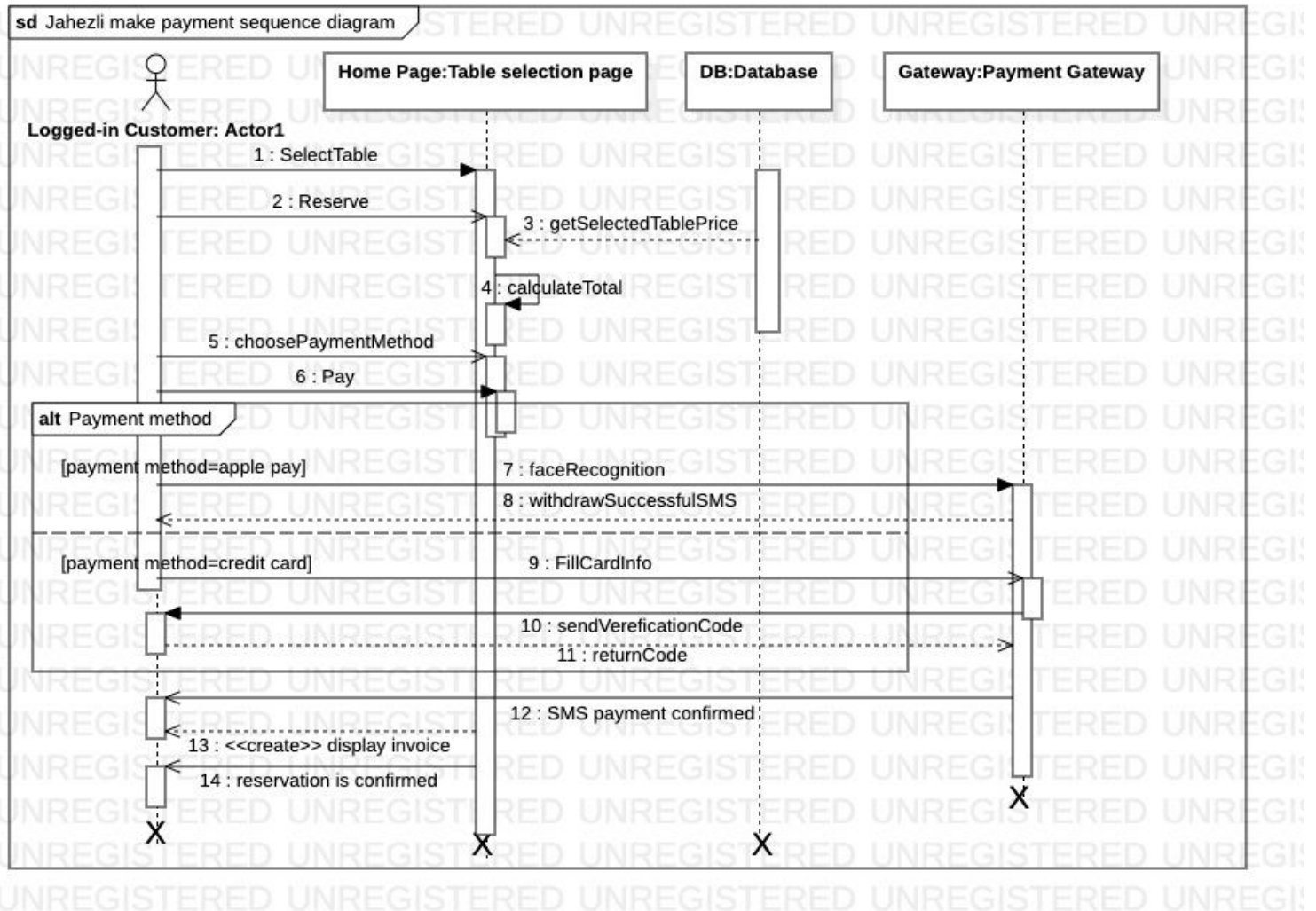


Figure 8 Make payment sequence diagram

V. Review and read feedback sequence diagram

This diagram shows the order of the operations when a customer displays and writes Reviews and feedback.

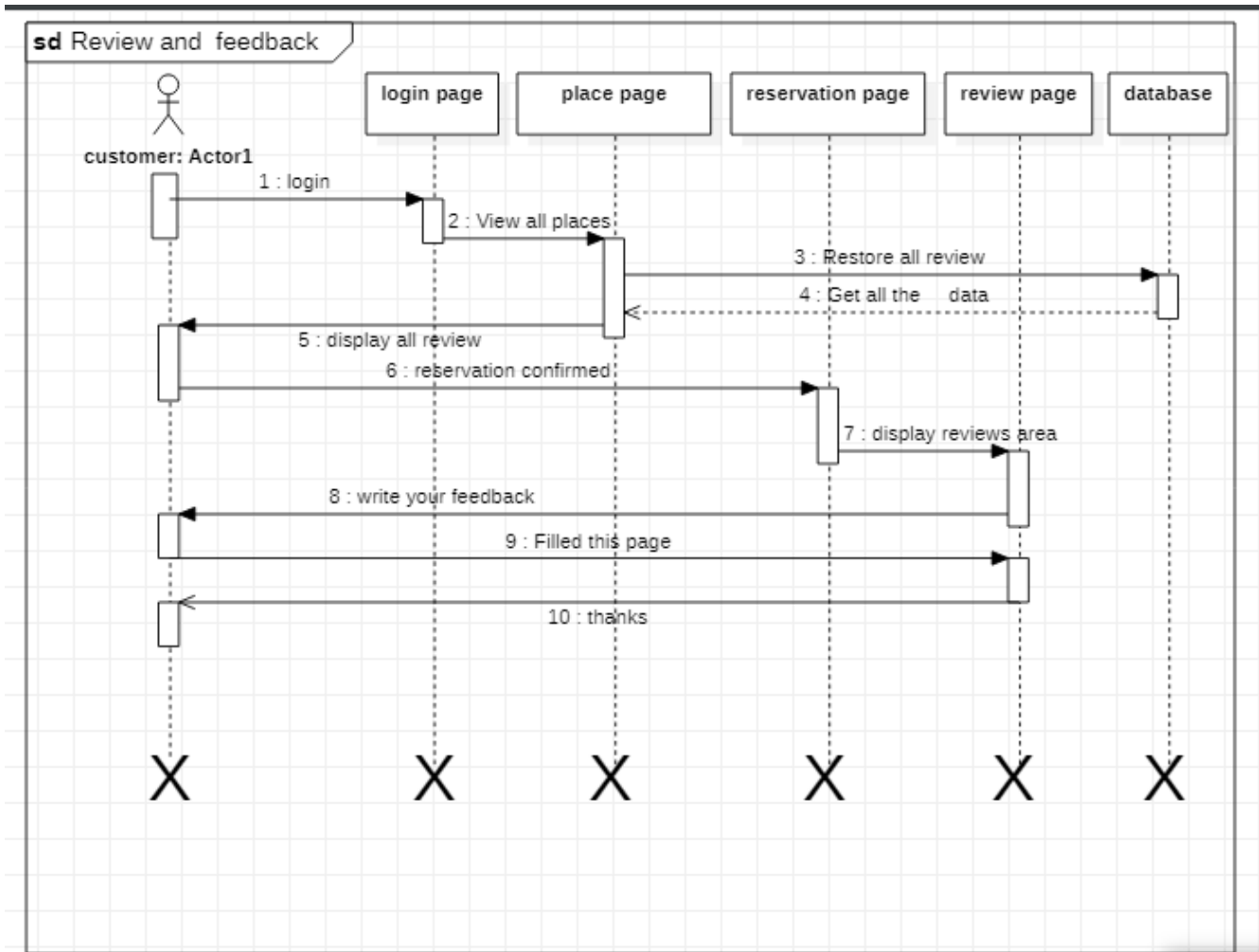


Figure 9 Review and read feedback sequence diagram

Activity diagram

Activity diagram used to display graphical representations of workflows of stepwise activities and actions with support for choice, iteration, and concurrency.

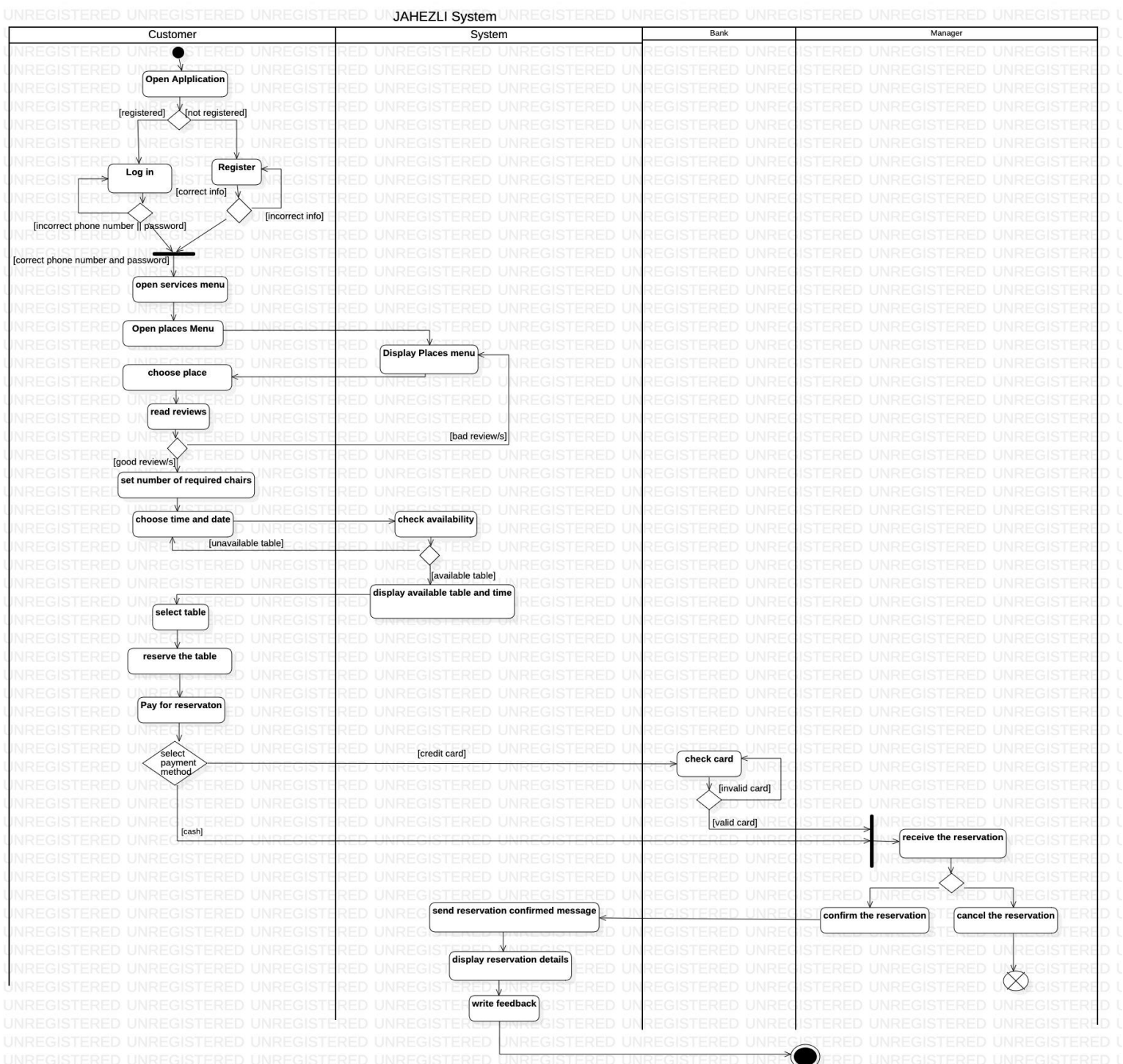


Figure 10 Activity diagram

Implementation

Login:

```
// this method takes login info
public static boolean login(Scanner input) throws FileNotFoundException {
    User user = new User();
    // log in ## not completed
    System.out.print("Enter your user name: ");
    String userName = input.next();
    System.out.print("Enter your password: ");
    String Password = input.next();
    boolean login_statue = user.login(userName, Password);
    if (login_statue == true) {
        System.out.println("WELCOME TO JahezliApp! ");
        customersatute = true;
    } else {
        System.out.println("incorrect password or username");
    }
    return login_statue;
}
```

Figure 11 Login source code

Registration:

```
public static void RegisterAccount(Scanner input) throws FileNotFoundException {
    String phone;
    String city;
    String password;
    // registration
    System.out.println("-----");
    System.out.println("                Registration Page                ");
    System.out.println("-----");
    String name;
    System.out.println("Enter your full name :");
    name = input.next();
    // Phone Number
    while (true) {
        // take phone and check if it correct
        System.out.println("Enter Phone Number Starting with 05:");
        phone = input.next();
        if (phone.length() == 10 && phone.startsWith("05")) {
            System.out.println("Phone number is correct ");
            break;
        }
    }
    // password i didn't put any conditions on the password
    System.out.println("Enter password:");
    password = input.next();
}
```

Figure 12 Registration source code (1)

```

System.out.println("Enter password:");
password = input.next();
// CITY
//this is a menu so user select city
System.out.println("-----");
System.out.println("                Select your City:                ");
System.out.println("-----");
System.out.println("Jeddah Enter 1");
System.out.println("Mecca Enter 2");
System.out.println("Riyadh Enter 3");
System.out.println("Dammam Enter 4");
city = input.next();
//check if it right
while (true) {
    if (city.equals("1") || city.equals("2") || city.equals("3") || city.equals("4")) {
        break;
    } else {
        System.out.println("-----");
        System.out.println("                Select your City:                ");
        System.out.println("-----");
        System.out.println("Jeddah Enter 1");
        System.out.println("Mecca Enter 2");
        System.out.println("Riyadh Enter 3");
        System.out.println("Dammam Enter 4");
        city = input.next();
    }
}

```

Figure 13 Registration source code (2)

```

// according to selection add the name of the city
switch (city) {
    case "1":
        city = "Jeddah";
        break;
    case "2":
        city = "Mecca";
        break;
    case "3":
        city = "Riyadh";
        break;
    case "4":
        city = "Dammam";
        break;
}

// insert the customer method not completed
// here i create an object of customer
//String phone, String city,String password
Customer customer = new Customer(name, phone, city, password);
arrayOfCustomers.add(customer);
System.out.println("thank you!" + customer.getName());
System.out.println("Account Created Successfully!");
}

```

Figure 14 Registration source code (3)

Reserve table:

```
public void ReservationTable(String tableN) throws IOException {

    FileReader f = new FileReader("TableReserved.txt");
    BufferedReader br = new BufferedReader(f);

    FileWriter file = new FileWriter("TableReserved.txt");//To save to file you need to create a File object.
    PrintWriter writer = new PrintWriter(file);//Then you need a way to print to the file. You can use PrintWriter
    int Tables[] = new int[2];

    String line;
    while ((line = br.readLine()) != null) {
        String[] token = line.split(",");
        for (int i = 0; i < Tables.length; i++) {
            if (token[i].equals(tableN)) {
                System.out.println("That table has already been reserved" + "Please select another one or try another time");

            } else {

                System.out.println("That table reserved");
                token[i] = tableN;
                writer.println(tableN);

            }
        }
    }
    br.close();
    writer.close();
}
```

Figure 15 Reserve table source code

Modify reservation:

```
public void modifyReservation(int reservationNo) throws ParseException {
    Scanner input = new Scanner(System.in);
    String date = input.nextLine();
    Date date1 = new SimpleDateFormat("dd/mm/yyyy").parse(date);
    Place placeName = null;
    String placeName1 = input.nextLine();
    paymentMethod total = null;

    System.out.println("Your reservation number is: " + getReservationNo());
    System.out.println("Your reservation date is: ");
    setDate(date1);
    System.out.println("Place name is: ");
    placeName.setPlaceName(placeName1);
    System.out.println(" Your reservation has been modified successfully ");
    System.out.println();

}
```

Figure 16 modify reservation

Cancel reservation:

```
//check parameter and return values
public String cancelReservation(int reservationNo) {
    for (int i = 0; i < reservations.size(); i++) {
        Reservation Temp = reservations.get(i);
        if (Temp.getReservationNo() == reservationNo) {
            reservations.remove(i);
            return Temp.getReservationNo() + " Your reservation has been cancelled successfully";
        }
    }
    return "This reservation is not Assigned to you";
}
```

Figure 17 cancel reservation

Display reservation:

```
public void displayReservation(int reservationNo) {
    Place placeName = null;
    paymentMethod total = null;

    System.out.println("Welcome");
    System.out.println("Your reservation number is: " + getReservationNo());
    System.out.println("Your reservation date is: " + getDate());
    System.out.println("Place name is: " + placeName.getPlaceName());
    System.out.println("Total amount is: " + total.getCost());
    System.out.println(" Enjoy. ");
    System.out.println();
}
```

Figure 18 Display reservation source code

Make Payment:

```
6    package jahezli.app;
7
8    /**
9     *
10    * @author mac
11    */
12    public class paymentMethod {
13        double cost;
14        String paymentDate;//date or String
15
16        public paymentMethod(){
17
18        }
19
20        public double getCost() {
21            return cost;
22        }
23
24        public void setCost(double cost) {
25            this.cost = cost;
26        }
27
28        public String getPaymentDate() {
29            return paymentDate;
30        }
31
32        public void setPaymentDate(String paymentDate) {
33            this.paymentDate = paymentDate;
34        }
35
```

Figure 19 make payment source code

Pay by credit card:

```
6 package jahezli.app;
7
8 /**
9  *
10  * @author mac
11  */
12 public class creditCard extends paymentMethod {
13
14     String bankName;
15     private String cardNo;
16     String expiredDate;
17     private int CVVNo;
18
19     public creditCard() {
20         super();
21     }
22
23     public String getBankName() {
24         return bankName;
25     }
26
27     public void setBankName(String bankName) {
28         this.bankName = bankName;
29     }
30
31     public String getCardNo() {
32         return cardNo;
33     }
34
35     public void setCardNo(String cardNo) {
36         this.cardNo = cardNo;
37     }
38
39     public String getExpiredDate() {
40         return expiredDate;
41     }
42
43     public void setExpiredDate(String expiredDate) {
44         this.expiredDate = expiredDate;
45     }
46
47     public int getCVVNo() {
48         return CVVNo;
49     }
50
51     public boolean authorized() {
52         boolean authorizedcheck;
```

```
26
27     public void setBankName(String bankName) {
28         this.bankName = bankName;
29     }
30
31     public String getCardNo() {
32         return cardNo;
33     }
34
35     public void setCardNo(String cardNo) {
36         this.cardNo = cardNo;
37     }
38
39     public String getExpiredDate() {
40         return expiredDate;
41     }
42
43     public void setExpiredDate(String expiredDate) {
44         this.expiredDate = expiredDate;
45     }
46
47     public int getCVVNo() {
48         return CVVNo;
49     }
50
51     public void setCVVNo(int CVVNo) {
52         this.CVVNo = CVVNo;
53     }
54
55     public boolean authorized() {
56         boolean authorizedcheck;
57
58         if ((cardNo == null) || (cardNo.length() < 13) || (cardNo.length() > 19)) {
59             System.out.println("failed length check");
60             authorizedcheck = false;
61             return authorized();
62         } else {
63             System.out.println("valid");
64             authorizedcheck = true;
65         }
66         return authorizedcheck;
67     }
68
69 }
70
```

Figure 21 pay by credit card source code part1

Figure 20 pay by credit card source code part2

Pay by cash:

```
6 package jahezli.app;
7
8 /**
9  *
10  * @author mac
11  */
12 public class Cash extends paymentMethod{
13     int change;
14
15     public Cash(){
16         super();
17     }
18
19     public int getChange() {
20         return change;
21     }
22
23     public void setChange(int change) {
24         this.change = change;
25     }
26
27
28
29 }
```

Figure 22 pay by cash source code

Review:

```
public static void getReviews(String placeName) throws FileNotFoundException {
    File inputFile = new File("C:\\Users\\اسامه بايونس\\Desktop\\Reviews.txt");
    if (!inputFile.exists()) {
        System.out.println("Input files does not exist !!");
        System.exit(0);
    }
    Scanner input = new Scanner(inputFile);
    String currPlaceName = "";
    String review = "";
    String temp1 = "";
    String temp2 = "";
    boolean notFound = true;
    int placeCounter = 0;
    ArrayList<String> reviews = new ArrayList<>();
    while (input.hasNext()) {
        temp1 = input.next();
        if (!temp1.equals("Place_Name:")) {
            continue;
        }
        currPlaceName = input.nextLine().trim();
        if (currPlaceName.equals(placeName)) {
            temp2 = input.next();
            if (!temp2.equals("Review:")) {
                continue;
            }
            review = input.nextLine().trim();
            reviews.add(review);
            notFound = false;
            placeCounter++;
        }
    }
}
```

Figure 23 Review source code (1)

```
    }
    System.out.println();
    if (notFound) {
        System.out.println("Not found any place with the required name \"" + placeName + "\"\n");
    } else {
        if (placeCounter == 1) {
            System.out.println("There is 1 review found for the place \"" + placeName + "\"\n");
        } else {
            System.out.println("There are " + placeCounter + " reviews found for the place \"" + placeName + "\"\n");
        }
        for (int i = 0; i < placeCounter; i++) {
            System.out.println("Review " + (i + 1) + ": " + reviews.get(i));
        }
        System.out.println();
    }
}
```

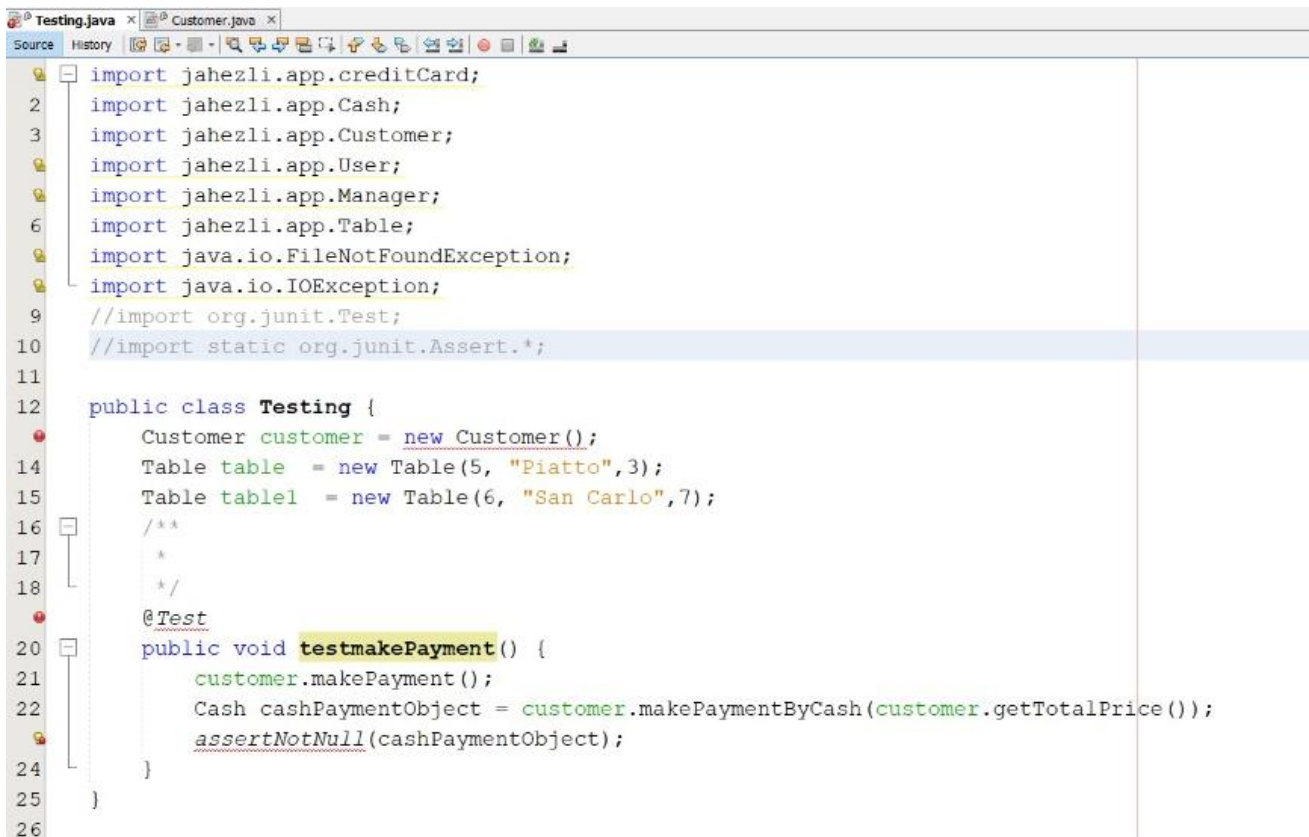
Figure 24 Review source code (2)

Feedback:

```
public static void printFeedback(String placeName) throws FileNotFoundException {  
  
    Scanner in = new Scanner(System.in);  
  
    File outputFile = new File("C:\\Users\\اسامه بايونس\\Desktop\\Feedback.txt");  
    PrintWriter output = new PrintWriter(outputFile);  
  
    System.out.print("Enter your feedback about the place \"" + placeName + "\"");  
    String feedback = in.nextLine();  
  
    output.println(feedback);  
    output.flush();  
  
}
```

Figure 25 Feedback source code

Test



The screenshot shows an IDE with two tabs: 'Testing.java' and 'Customer.java'. The 'Testing.java' tab is active, displaying the following code:

```
1 import jahezli.app.creditCard;  
2 import jahezli.app.Cash;  
3 import jahezli.app.Customer;  
4 import jahezli.app.User;  
5 import jahezli.app.Manager;  
6 import jahezli.app.Table;  
7 import java.io.FileNotFoundException;  
8 import java.io.IOException;  
9 //import org.junit.Test;  
10 //import static org.junit.Assert.*;  
11  
12 public class Testing {  
13     Customer customer = new Customer();  
14     Table table = new Table(5, "Piatto", 3);  
15     Table table1 = new Table(6, "San Carlo", 7);  
16     /**  
17      *  
18      */  
19     @Test  
20     public void testmakePayment() {  
21         customer.makePayment();  
22         Cash cashPaymentObject = customer.makePaymentByCash(customer.getTotalPrice());  
23         assertNotNull(cashPaymentObject);  
24     }  
25 }  
26
```

Figure 26 Test

Challenges:

It was hard for us to implement this project as a code, because this is the first project we implement from the first phase, which is planning to the last phase testing.

It was hard to deal with github because it is a new for us.

Future work:

Our application covered certain cities Jeddah, Mecca, Riyadh, and Dammam, so we hope to make for all cities in Saudi Arabia. Also we used file as way to store our data, to improve the application we will use a database to store all data.

Conclusion

In the end, the main idea of the project is to develop an application for the customer to choose and reserve a table in restaurants for the available classes without waiting a lot of time. Moreover, for managers, can check the availability and can respond to their customer. the customer has two way to pay which make it easier for them. The system also has a feature that allows the customer to modify a reservation or cancel it.