### 

**Object Oriented Analysis**

**Team members**

1. HMBS Induwara (cs/2020/037)
2. Sasith Avinda (CS/2020/030)
3. Kasun chathuranga (CS/2020/068)
4. Kusal Senanyake (CS/2020/060)
5. Ashan Lokuge (CS/2020/19)
6. Pruthuvi Samod (CS/2020/28)

### **1)**

### **1. Actors and their Types**

**Primary Actors:**

* Customer
* Restaurant Manager

**Secondary Actors:**

* Administrator

### **2. Use Cases**

1. Create new account
2. Log in
3. verify password
4. display login error
5. Manage personal information
6. Manage Payment Information
7. Search for Restaurants
8. Make Reservation
9. Cancel Reservation
10. Browse Menu
11. Place Food Order
12. Manage Reservations
13. Update Menu
14. Process Food Orders
15. Manage User Accounts
16. Access System Settings
17. View Order/Reservation History
18. maintain application
19. update application

### 

### **3. Relationships**

**Associations:**

* Customer:
  + Register as Customer
  + Log in
  + Manage Profile Information
  + Manage Payment Information
  + Search for Restaurants
  + Make Reservation
  + Cancel Reservation
  + Browse Menu
  + Place Food Order
  + View Order/Reservation History
* Restaurant Manager:
  + Log in
  + Manage Profile Information
  + Manage Payment Information
  + Manage Reservations
  + Update Menu
  + Process Food Orders
  + View Order/Reservation History
* Administrator:
  + Manage User Accounts
  + Access System Settings

Includes**:**

* verify password

Extends**:**

* display login error
* manage personal details
* manage payment information
* cancel reservation
* update status

A white paper with black writing

Description automatically generated

2)

Classes

1. Customer
   1. attributes
      1. Name -private,string
      2. ContactInformation-private,string
      3. Id -private,string
   2. Methods
      1. SearchingForRestaurant() :Restaurant
      2. MakingReservation():Reservation
      3. PlacingFoodOrders():Order
      4. Providing feedback():void
      5. getName():string,setName():string
      6. getContact():string,setContact():void
2. Restaurant
   1. Attributes
      1. Name-private,string
      2. Contact-private,string
      3. Id-private,string
   2. Methods
      1. UpdateMenus():void
      2. ManageReservation()
      3. ProcessingFoodOrder()
3. Reservation
   1. Attributes
      1. Id -private,string
      2. DateAndTime -private,Date
      3. Status-private,string
   2. Methods
      1. Confirm reservation()
      2. HandleCancellation()
      3. getStatus():string
      4. getDate():Date
4. Order
   1. Attributes
      1. OrderId -private,string
      2. ItemOrdered -private ,List<Menu>
      3. TotalCost -private,int
      4. Status-private,string
   2. Methods
      1. UpdateOrderStatus()
      2. Calculate total cost()
      3. getOrderStatus():string
      4. getOderedItems():List<Menu>
5. Menu
   1. Attributes
      1. Name -private,string
      2. Price -private,int
      3. Description-private,string
   2. Methods
6. Authentication
   1. Attributes
      1. authId -private,int
   2. Methods
      1. Login()
      2. RegisterCustomer(): Customer
      3. Register Restaurant(): Restaurant
      4. Log out restaurant()
      5. Logout customer()
7. Payment
   1. Attributes
      1. paymentID -private, int
      2. amount -private, double
      3. Date -private,date
      4. Status private, String
      5. customerID private, int
      6. orderID - private, int ,nullable
      7. reservationID -private, int nullable
   2. Methods
      1. processPaymentOfOrder()
      2. processPaymentOfReservation()
      3. getPaymentDetails()

Relationships, Types, and Multiplicity

1. Customer - Reservation

- Type: Association

- Multiplicity: `1..\*` Customers can have `0..\*` Reservations

2. Customer - Order

- Type: Association

- Multiplicity: `1..\*` Customers can place `0..\*` Orders

3. Restaurant - Reservation

- Type: Association

- Multiplicity: `1` Restaurant can have `0..\*` Reservations

4. Restaurant - Order

- Type: Association

- Multiplicity: `1` Restaurant can have `0..\*` Orders

5. Restaurant - Menu

- Type: Composition

- Multiplicity: `1` Restaurant has `1..\*` Menu

6. Reservation - Payment

- Type: Association

- Multiplicity: `1` Reservations can have `1` Payment

7. Order - Payment

- Type: Association

- Multiplicity: `1` Orders can have `1` Payment

8. Authentication - Customer

- Type: Association

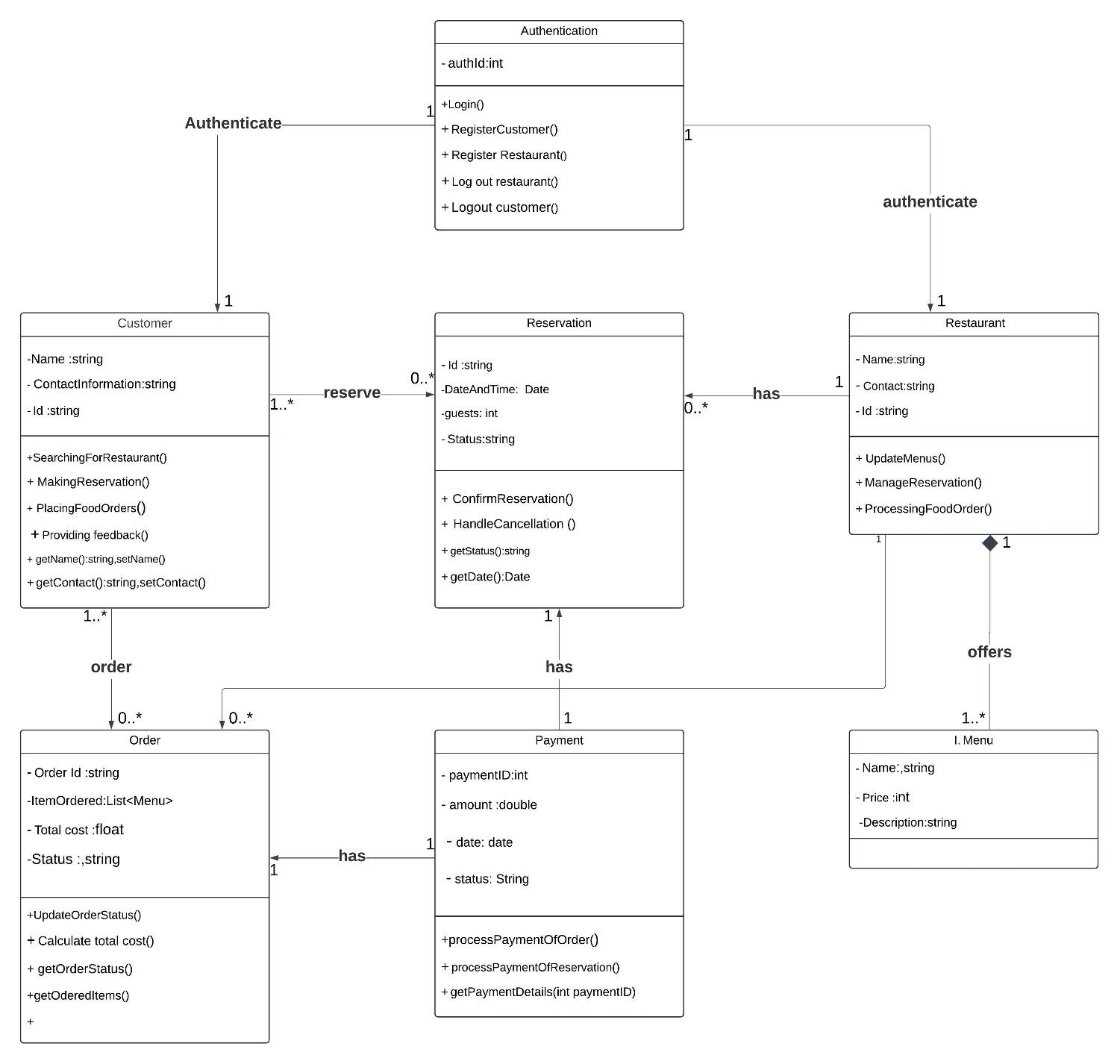
- Multiplicity: `1` Authentication authenticates `1` Customers

9. Authentication - Restaurant

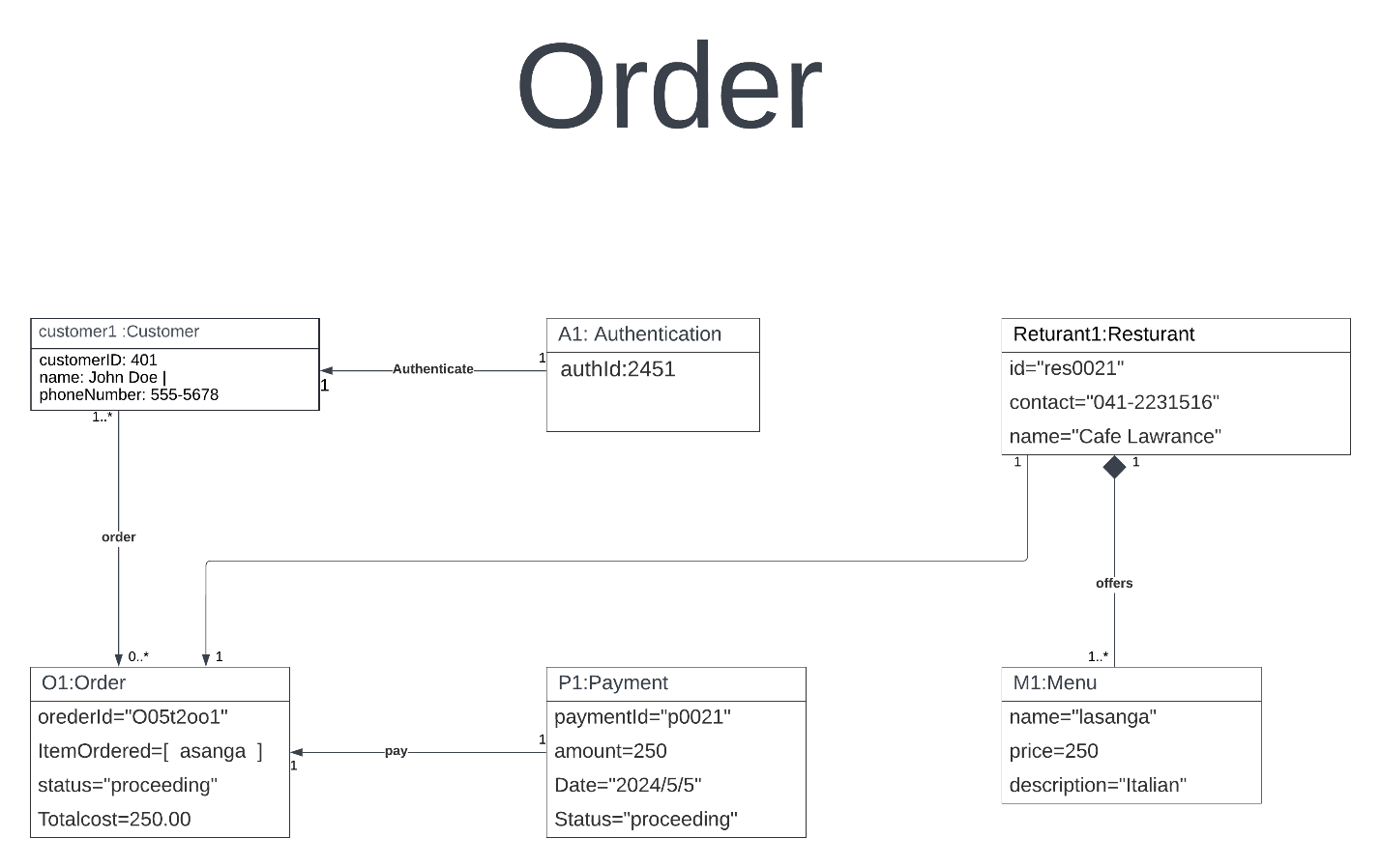
- Type: Association

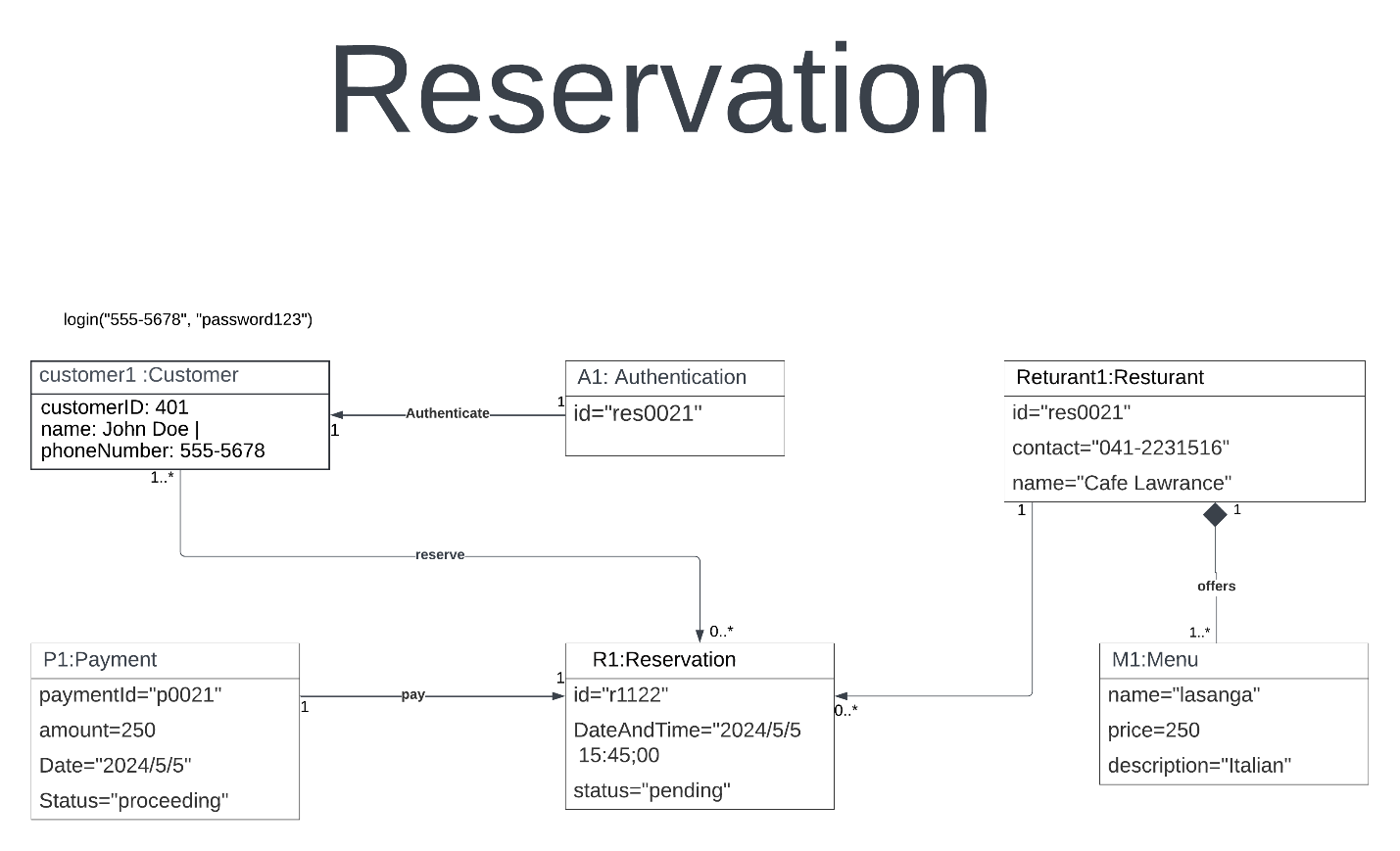
- Multiplicity: `1` Authentication authenticates `1` Restaurants

Class Diagram



Object Diagram





3)

1. All the actors and objects.

* Actors :

o User (Nimal)

o Restaurant Manager (Mahela)

* Objects :

o FoodieFinds App

o FoodieFinds Server

o Restaurant System

2. All the messages and the alternative paths.

* Messages :
* User opens the FoodieFinds App.

o User -> FoodieFinds App : Open the app

* User searches for the restaurant.

o User -> FoodieFinds App : Search for the restaurant

o FoodieFinds App -> FoodieFinds Server : Search restaurant list

o FoodieFinds Server -> FoodieFinds App : Available restaurant list

o FoodieFinds App -> User : Show the restaurants

* User selects the desired date and time for the reservation.

o User -> FoodieFinds App : Select the date & time

o FoodieFinds App -> FoodieFinds Server : Request reservation

o FoodieFinds Server -> Restaurant System : Check the availability

o Resturant System -> FoodieFinds Server : Confirm the availability

o FoodieFinds Server -> FoodieFinds App : Confirm reservation

o FoodieFinds App -> User : Notify reservation is confirmed

* Restaurant manager (Mahela) sees the new reservation.

o Restaurant System -> Restaurant Manager : Confirm new reservation

* User receives the reminder notification.

o FoodieFinds App -> User : Send the reminder notification

* User provides a feedback.

o User -> FoodieFinds App : Provide feedback

o FoodieFinds App -> FoodieFinds Server : Submit feedback

* Alternative paths :

* Reservation can’t be found.

o Restaurant System -> FoodieFinds Server : Reservation not available

o FoodieFinds Server -> FoodieFinds App : Reservation not available

o FoodieFinds App -> User : Reservation failed

Sequence Diagram

