TABLE OF CONTENTS

1.PROBLEM STATEMENT:

The online food ordering system works in such a way that the customers are presented with an interactive up-to-date menu simplifying the ordering process for both the customers and restaurants.

2.SRS: SOFTWARE REQUIREMENT SPECIFICATION:

2.1.PREFACE:

The system helps improve the management aspect by utilising computerised system to co-ordinate each and every food ordering transactions instead of traditional methods.

The system facilitates in reducing human errors and providing good quality customer service.

2.2.INTRODUCTION:

The online food ordering system allows the customer to register, log in, log out, edit profile perform optimised search , view up-to-date menu, share and track live location, place orders , personalise and edit the cart, avail offers and make payments.

2.3.GLOSSARY:

- vb.net
- Star UML
- PHP
- sql+ database
- GPS

2.4.USER REQUIREMENTS:

2.4.1 REGISTRATION:

INPUT: username , password , email id ,phone number.

PROCESS: Verifying email id, phone number.

OUTPUT: Account created.

2.4.2 LOGGING IN:

INPUT: Username, password.

PROCESS: Authenticating the user.

OUTPUT: Access to system.

2.4.3 OPTIMIZED SEARCH:

INPUT: Database of hotels with all details.

PROCESS: Queries to select relevant data.

OUTPUT: Displays relevant information.

2.4.4 UP-TO-DATE MENU:

INPUT: Detailed menu of hotels with availability

PROCESS: Retrieving unambiguous information.

OUTPUT: Displays appropriate menu.

2.4.5 LOCATION SHARING AND LIVE TRACKING:

INPUT: Device location via GPS

PROCESS: Tracks the location and updates in live map.

OUTPUT: Visualise the order location and provide delivery address.

2.4.6 FACILITATING FAVOURITES:

INPUT: User providing personal favourites information

PROCESS: Saves the favourites.

OUTPUT: Notifies offers and availability on favourites.

2.4.7 PERSONALIZED CART:

INPUT: Selection of items for delivery.

PROCESS: Visually verify order and edit them if required.

OUTPUT: Puts orders together.

2.4.8 PAYMENT OPTIONS:

INPUT: User card details/payTM details.

PROCESS: Direct to bank website/payTM server and detect amount.

OUTPUT:Makes payment and notifies user or applies COD.

2.4.9 OFFERS AND COUPONS:

INPUT: Coupon code.

PROCESS: Reduces bill amount.

OUTPUT: Optimised cost is paid.

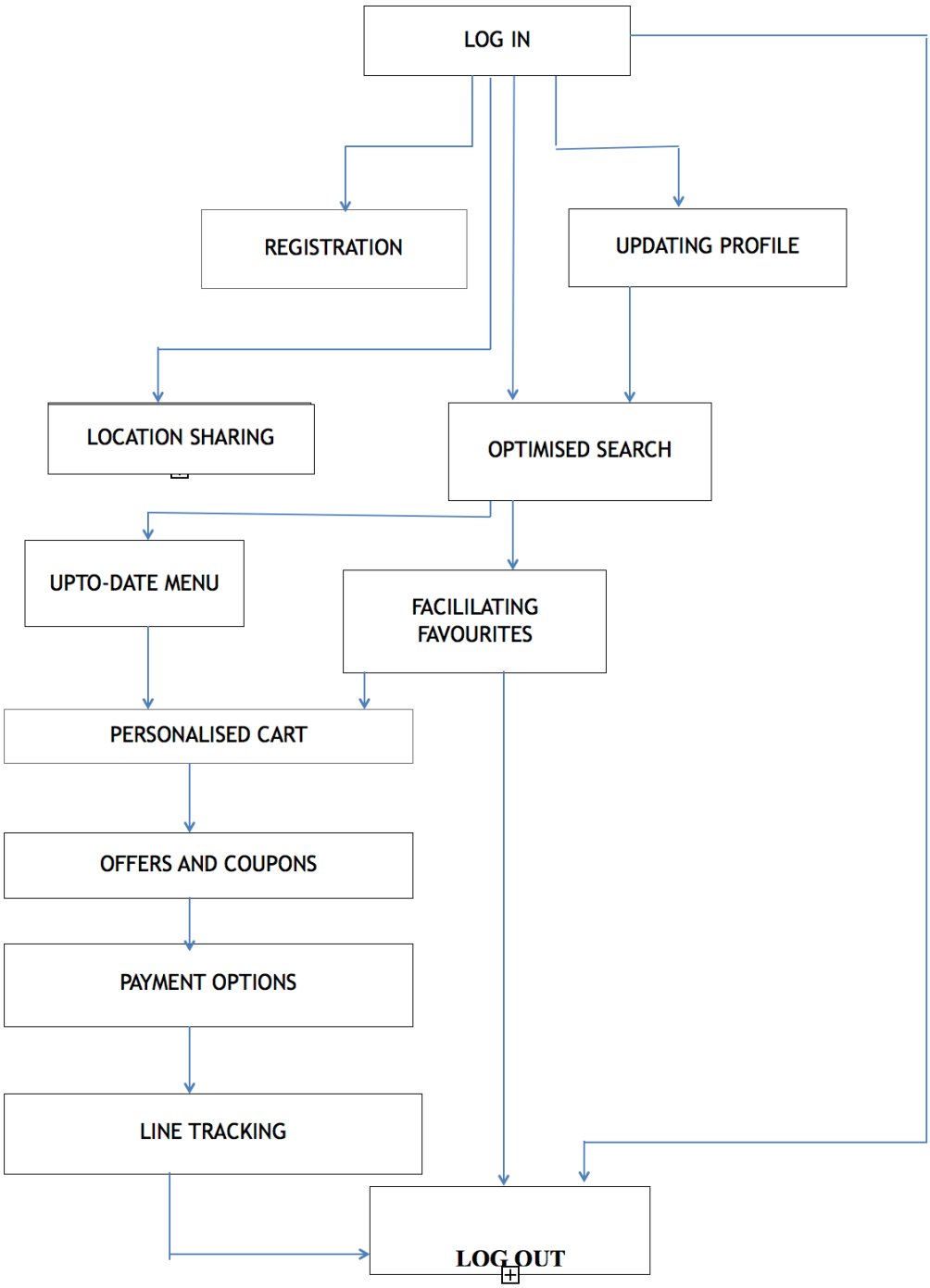
2.4.10 UPDATING PROFILE:

INPUT: Change of credentials.

PROCESS: Verifying the details.

OUTPUT: Account updation

2.5. SYSTEM ARCHITECTURE :

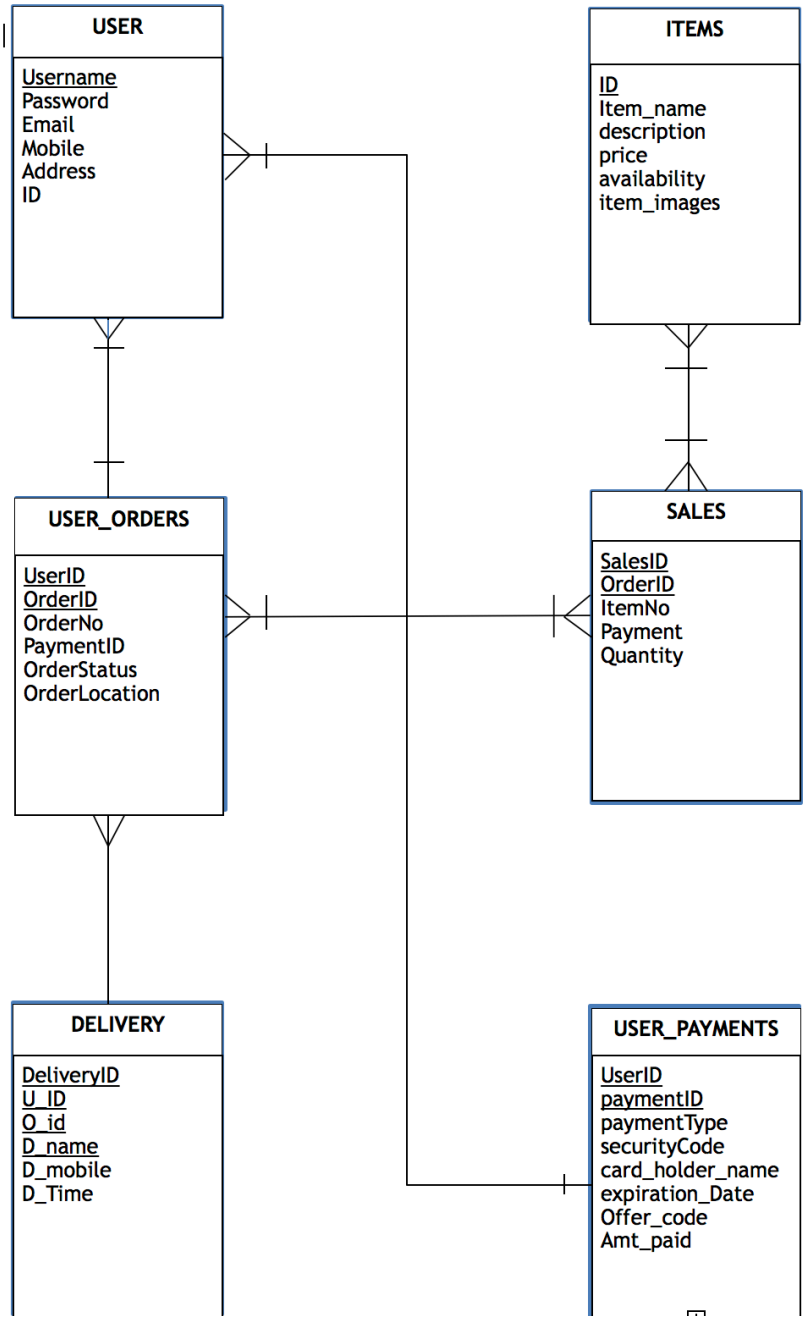


2.6. SYSTEM REQUIREMENTS:

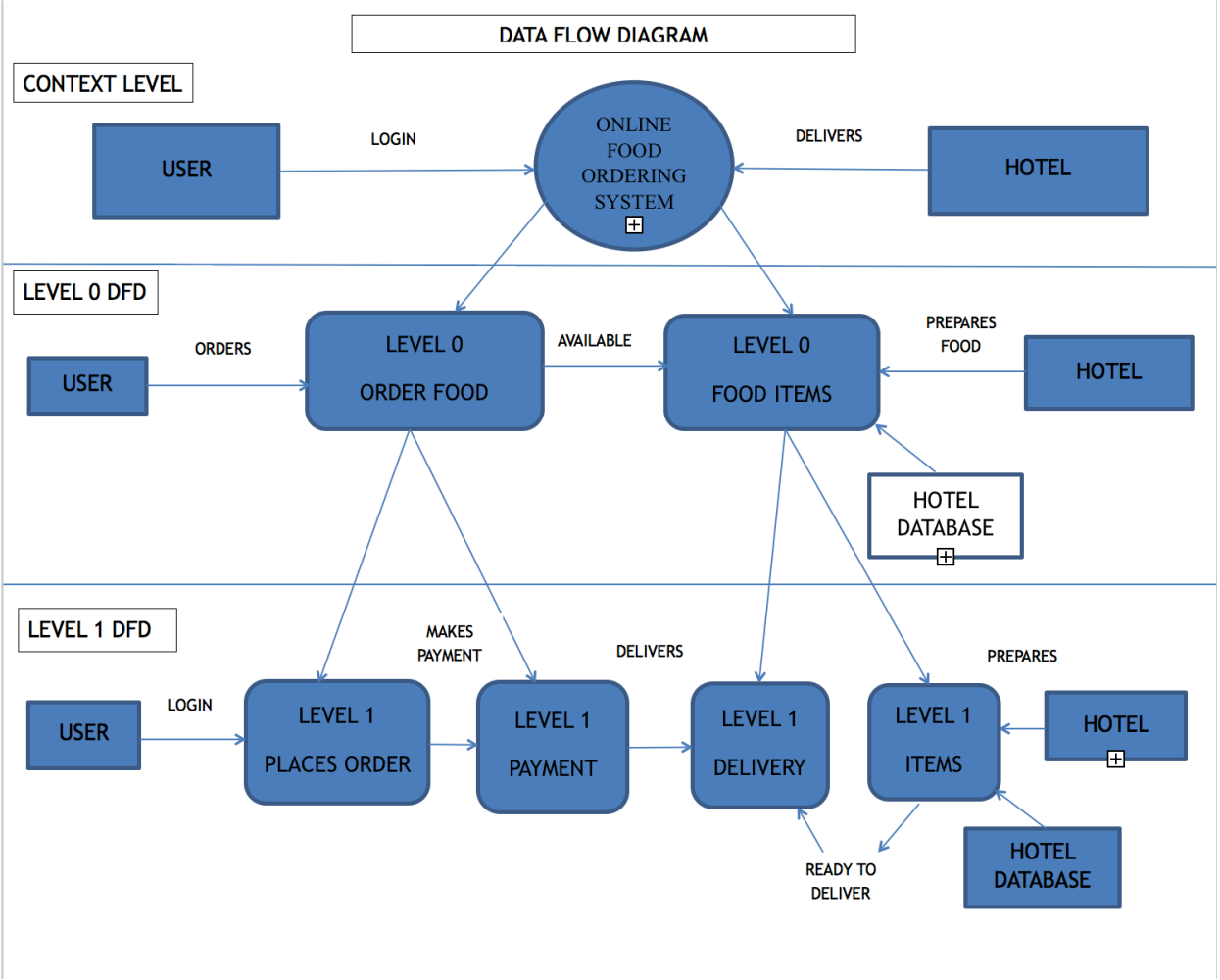
- Operating System : Windows, MacOSX
- Platforms : iOS, android , vb.net/Java/PhP
- Database : sql+database
- Tool : StarUML

2.7. SYSTEM MODELS:

2.7.1 ER MODEL:



2.7.2 DATA FLOW DIAGRAM:



2.8. SYSTEM EVOLUTION:

- This is the first version being developed.

The proposed system differs from the existent system by the following.

- No minimum order constraints.
- Scales a larger geographic area for service.
- Removes spams and Trivial Promotions.
- 24x7 service.

2.9.REFERENCES:

i. Roger S.Pressman,Software Engineering:A Practitioner's Approach.

ii. https://www.inflectra.com/ideas/topic/requirementsdefinition.aspx -SRS.

iii. https://www.smartdraw.com/data-flow-diagram.aspx -DFD

iv. https://www.lucidchart.com/pages/ER-diagram-symbols-and-meaning.aspx -ER model

3.StarUML:

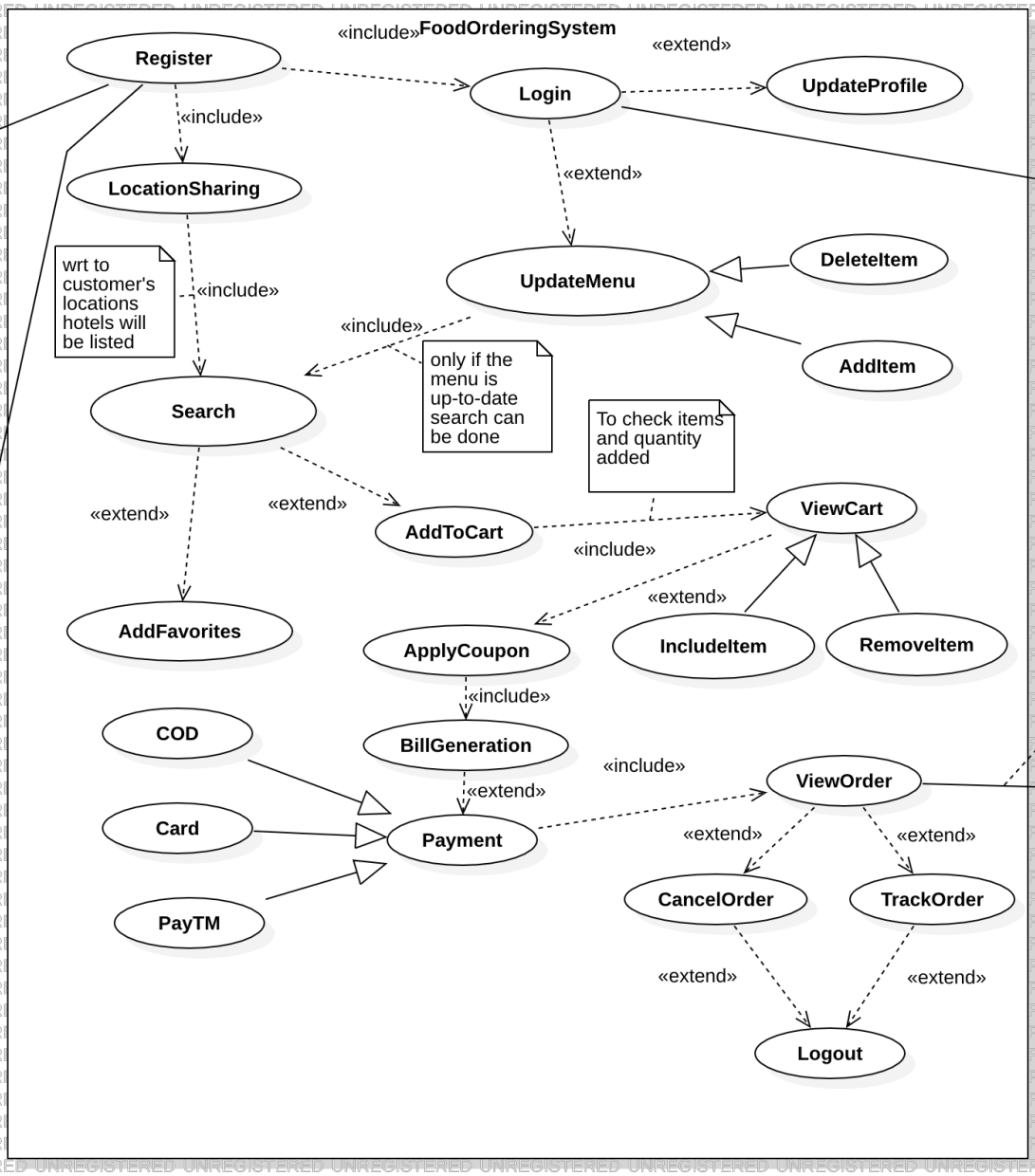
StarUML is a UML tool by MKLab. It is an open software modelling tool that supports UML (Unified Modeling Language). It is based on UML version 1.4, provides eleven different types of diagrams and it accepts UML 2.0 notation. It actively supports the MDA (Model Driven Architecture) approach by supporting the UML profile concept and allowing to generate code for multiple languages.

StarUML supports the following diagrams types

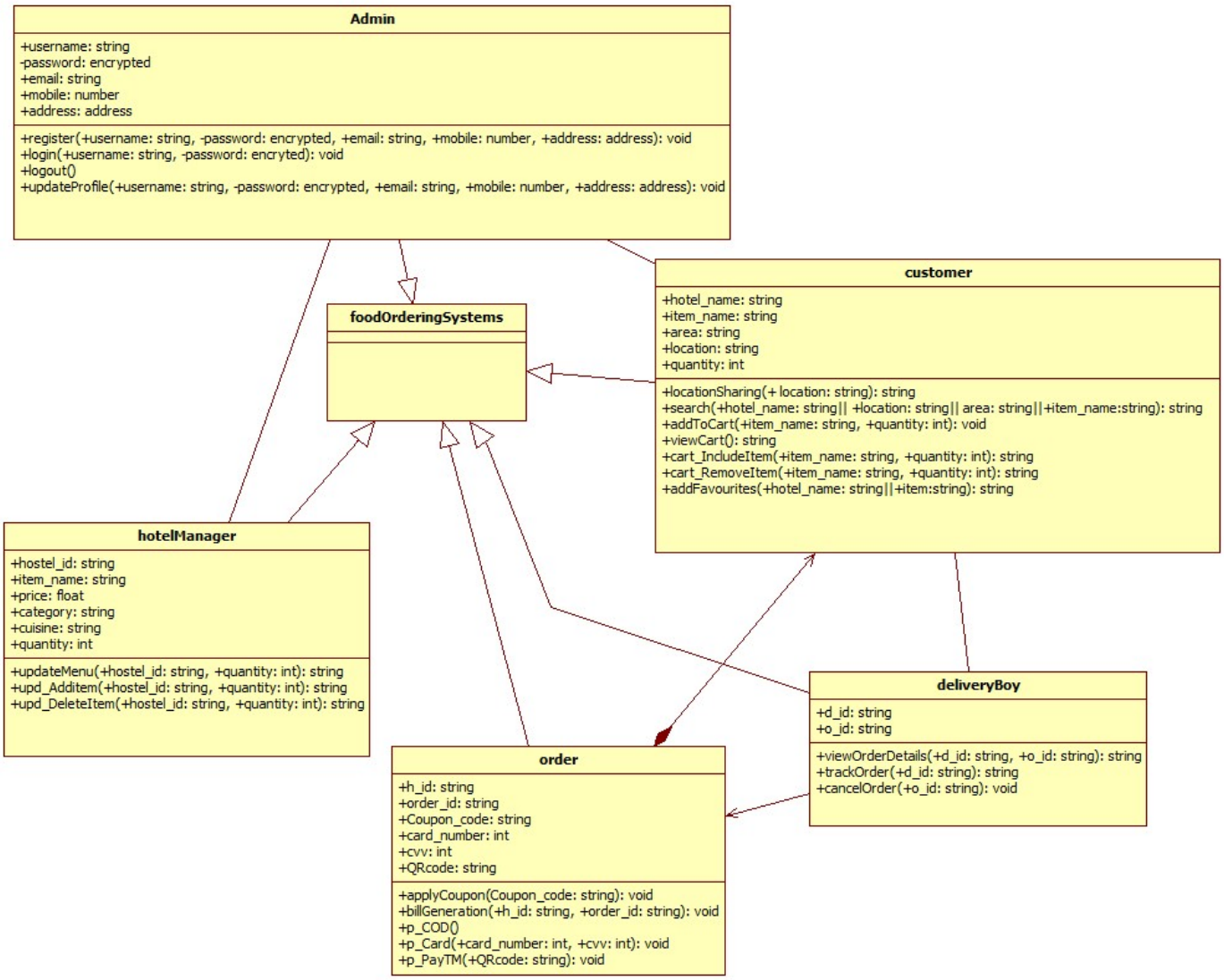
- Use Case Diagram
- Class Diagram
- Sequence Diagram
- Collaboration Diagram
- Activity Diagram
- Component Diagram
- Deployment Diagram
- Composite Structure Diagram

4.DIAGRAMS

4.1 UML USECASE DIAGRAM



4.2 UML CLASS DIAGRAM



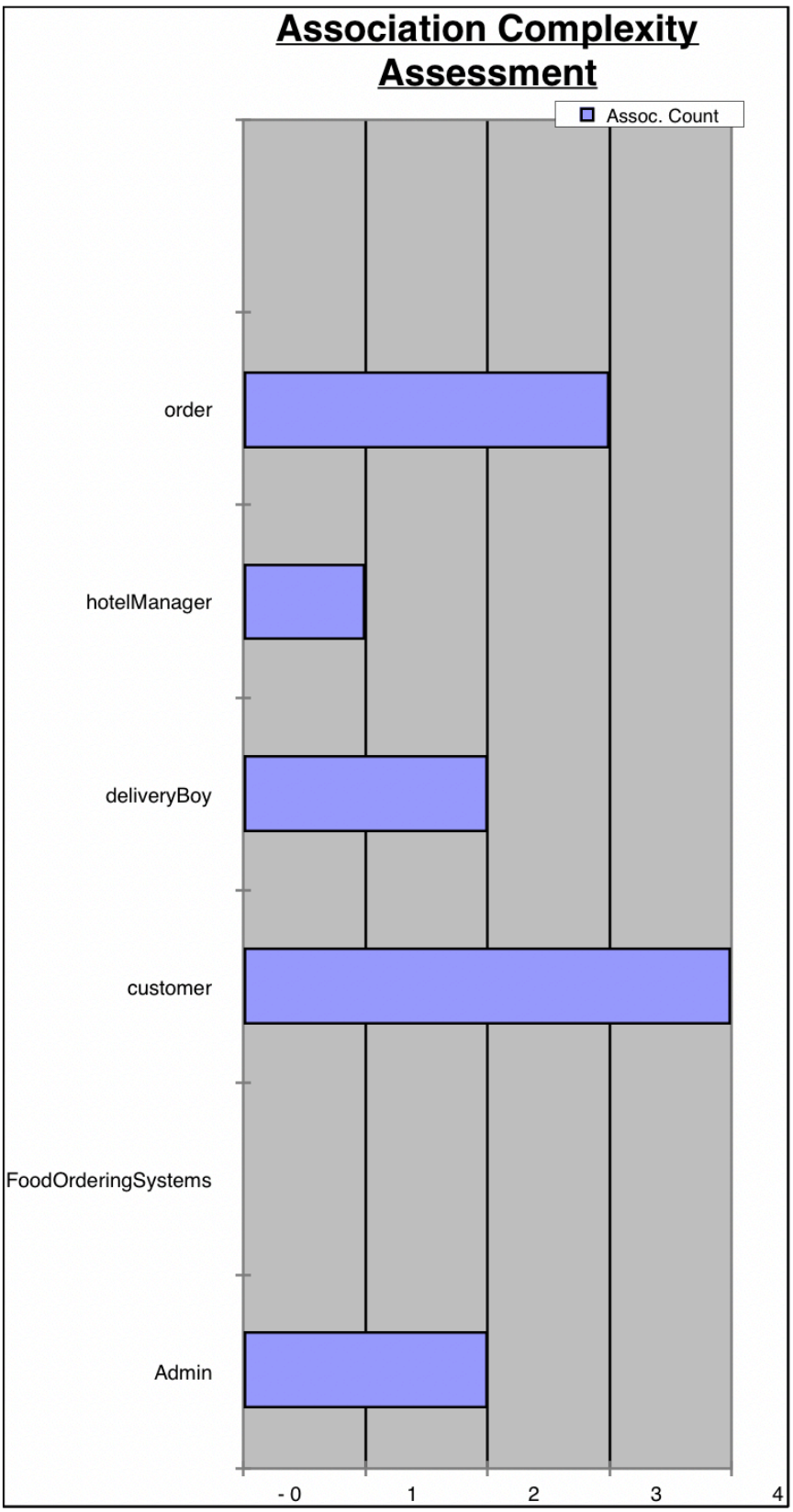
4.2.1. DataSheet:

Class Complexity Assessment

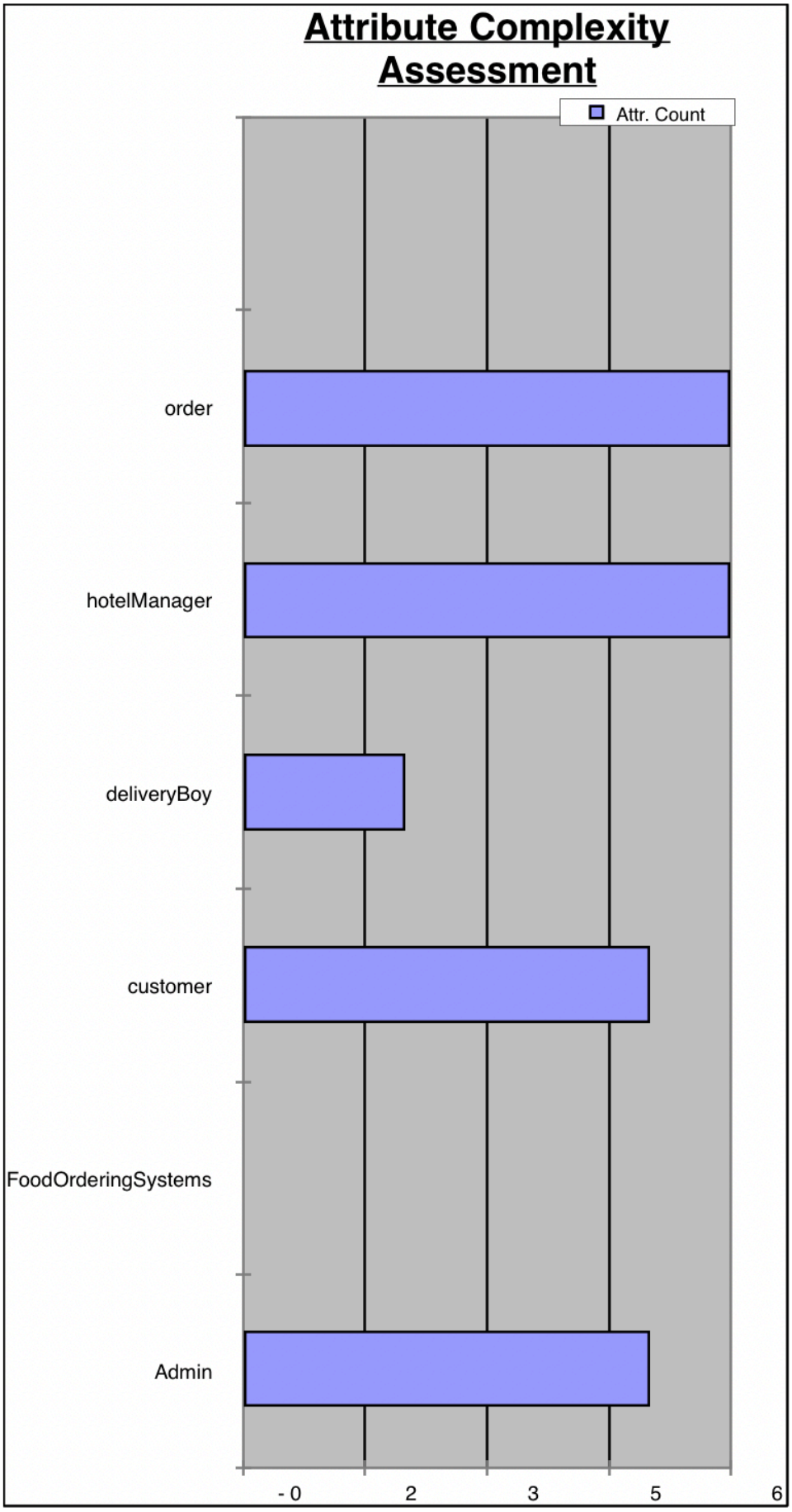
Class	Package	Assoc. Count	Attr. Count	Oper. Count
Admin	::Design Model	2	5	4
FoodOrderingSystems	::Design Model	- 0	- 0	- 0
customer	::Design Model	4	5	7
deliveryBoy	::Design Model	2	2	3
hotelManager	::Design Model	1	6	3
order	::Design Model	3	6	5

All Count of Classes	1
All Count of Associatio	12
All Count of Attributes	24
All Count of Operation	22

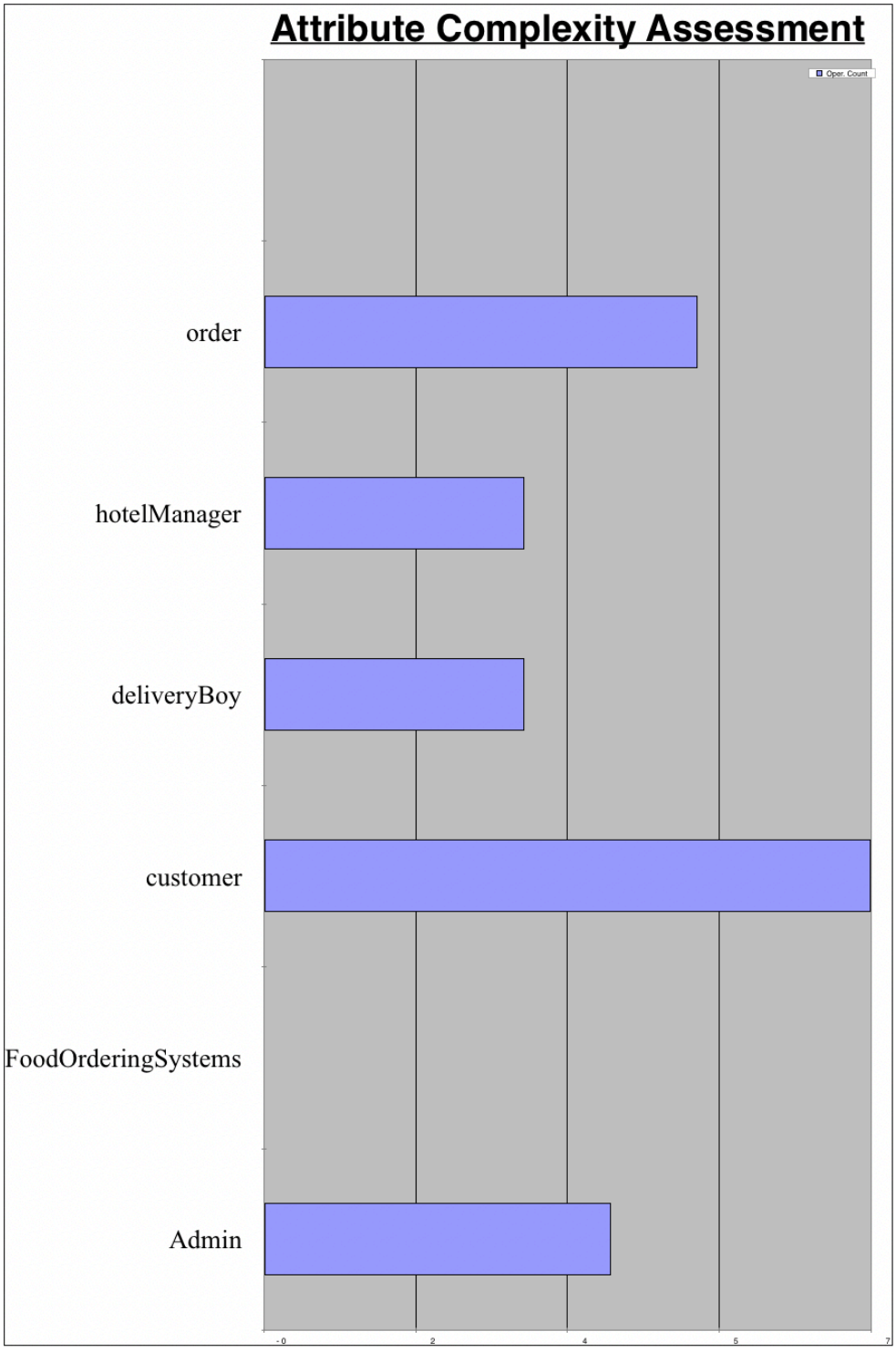
4.2.2. Association Assessment :



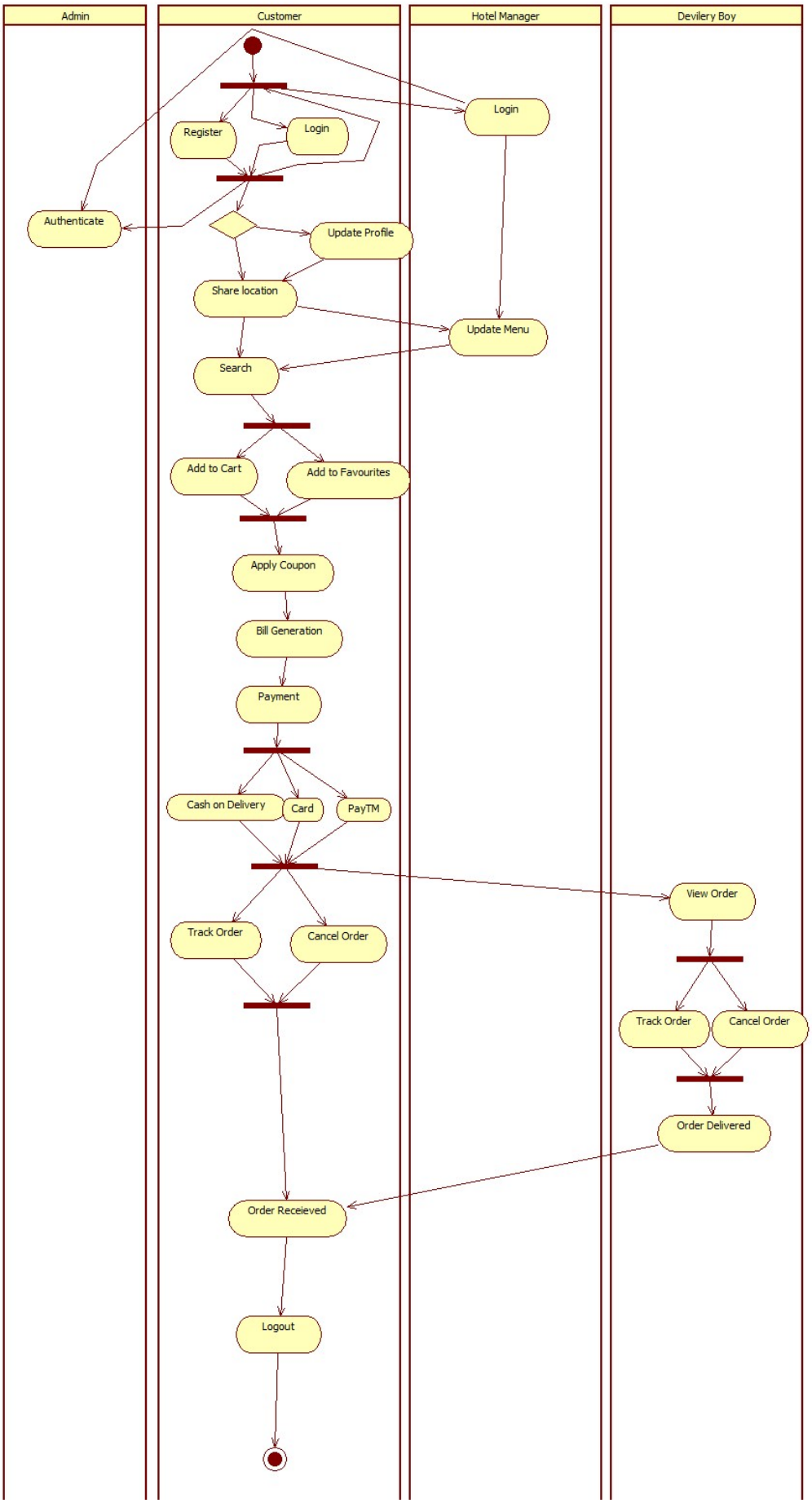
4.2.3. Attribute Assessment



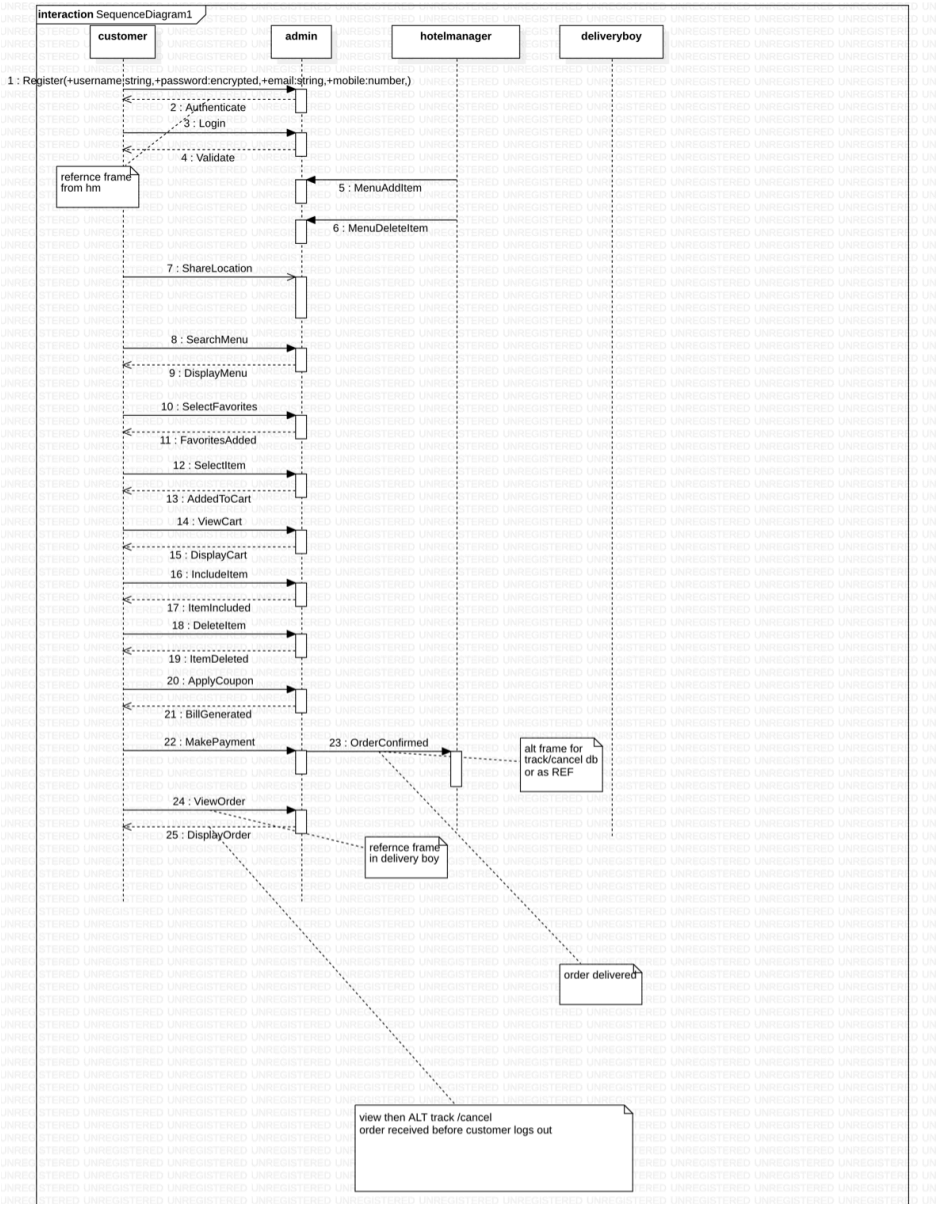
4.2.3 Operation Assessment



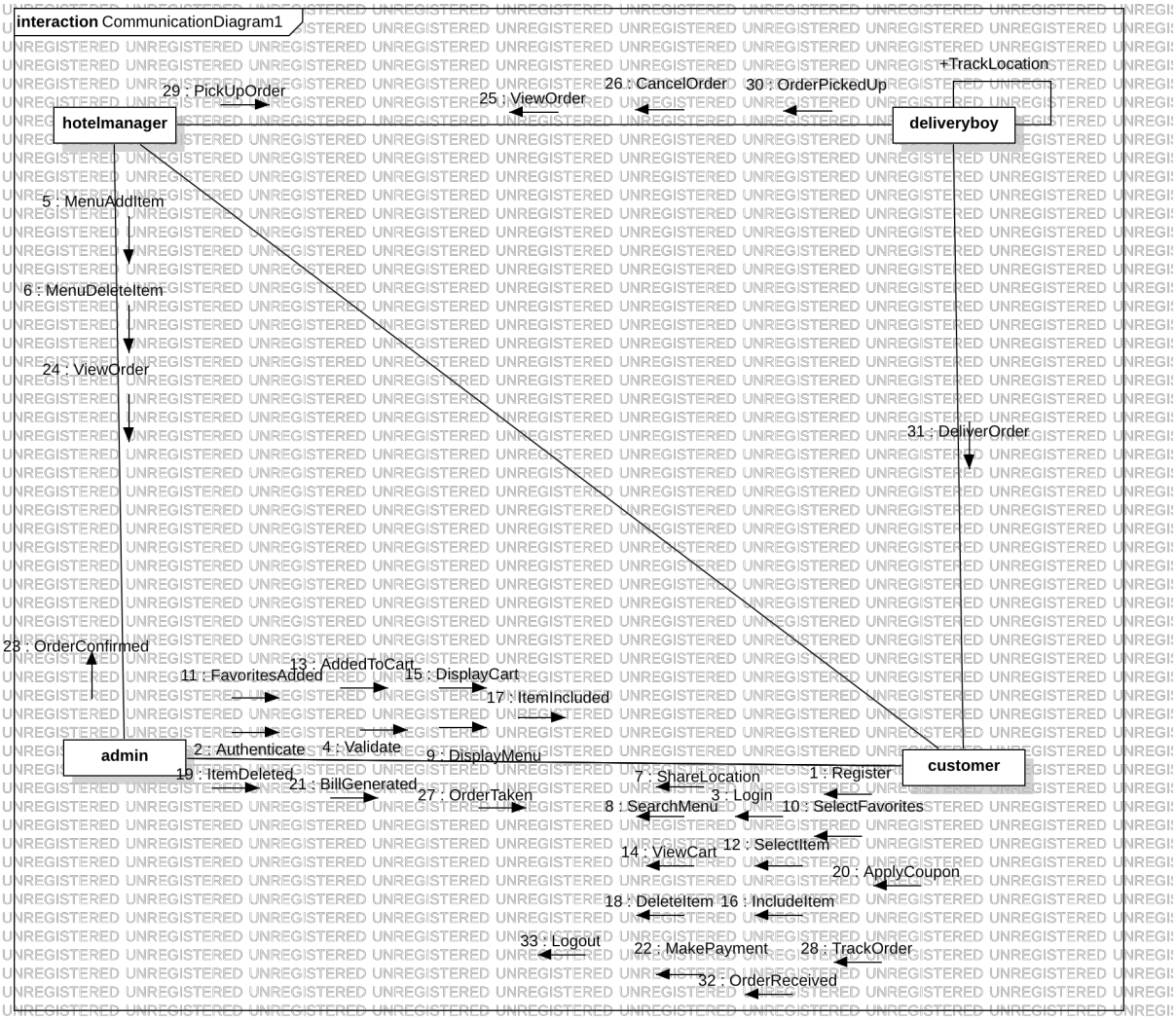
4.3. UML ACTIVITY DIAGRAM



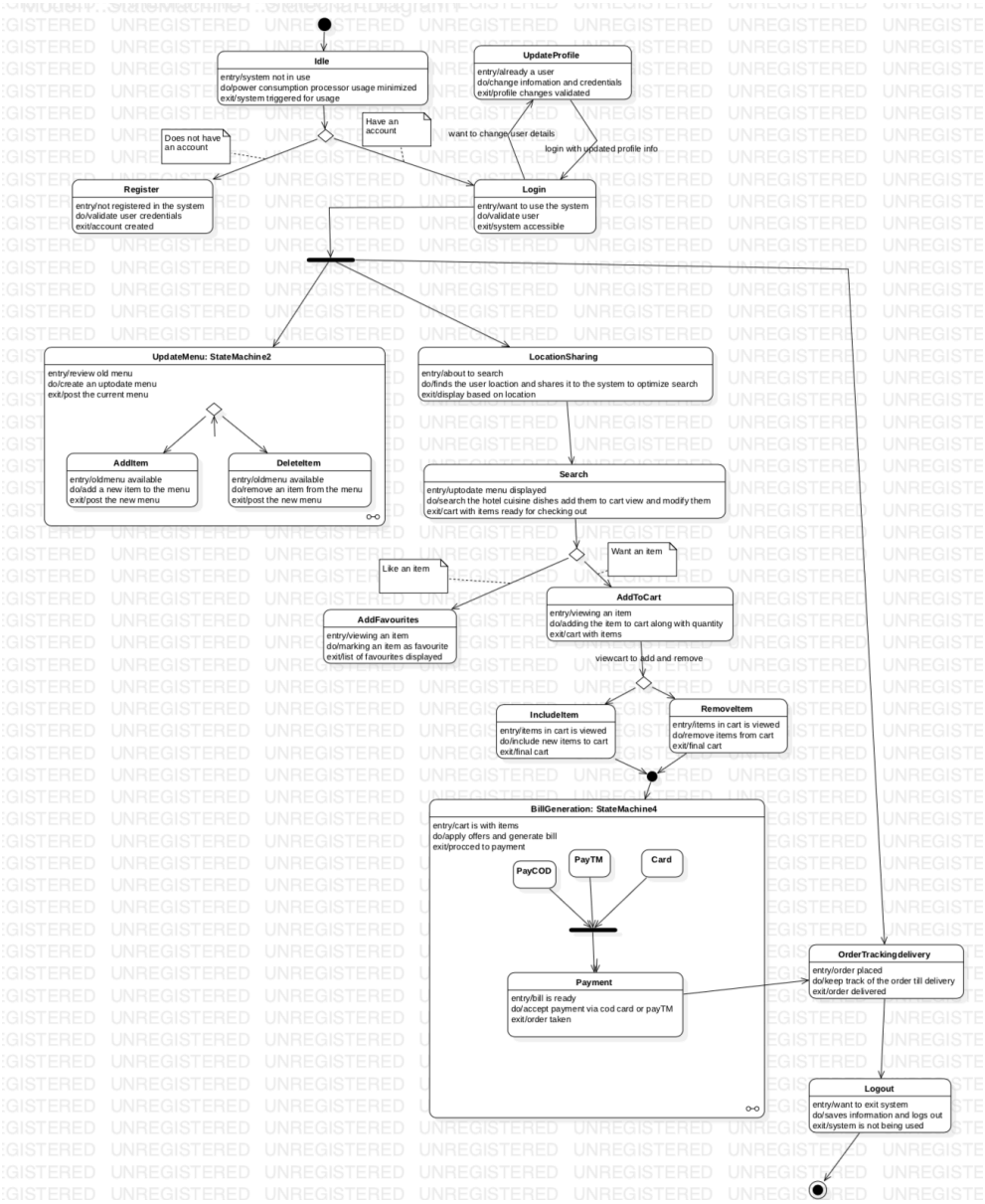
4.4. UML SEQUENCE DIAGRAM



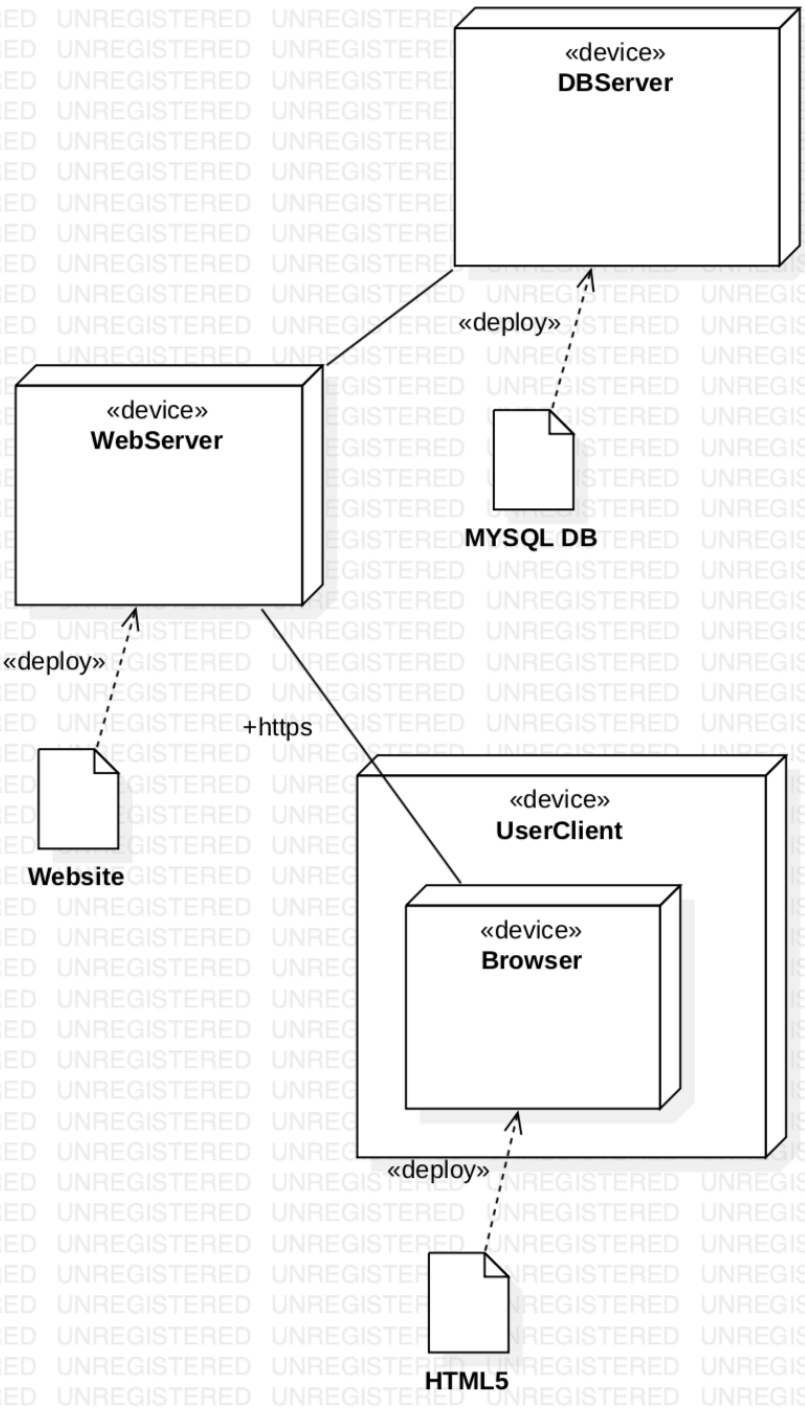
4.5. UML COLLABORATION DIAGRAM



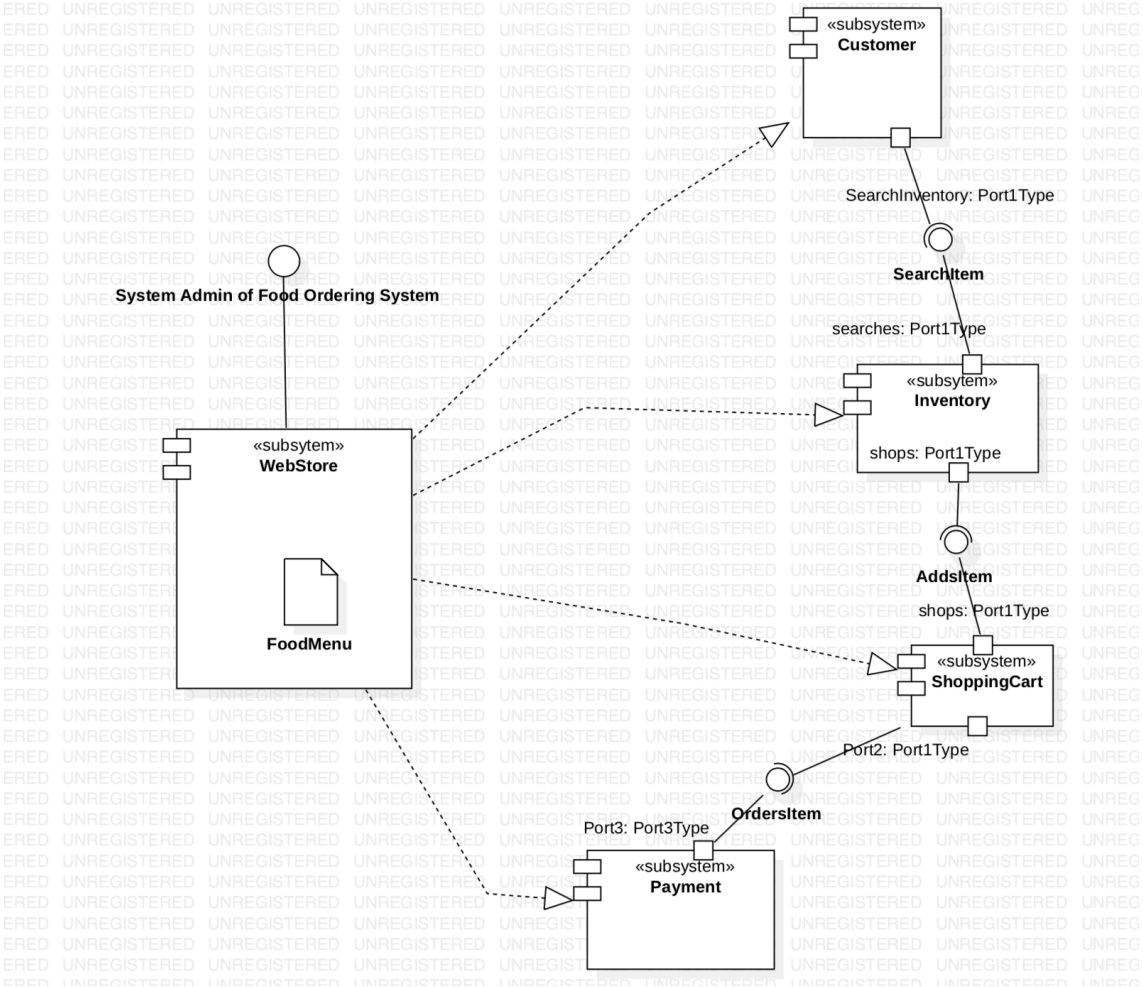
4.6 UML STATE MACHINE DIAGRAM



4.7 UML COMPONENT DIAGRAM



4.8. UML DEPLOYMENT DIAGRAM



5. SAMPLE CODE GENERATED:

Class:

5.1 Admin source:

```
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : FoodOrderingSystems.h
//  @ Date : 10/14/2018
//  @ Author :
//
//


#if !defined(_FOODORDERINGSYSTEMS_H)
#define _FOODORDERINGSYSTEMS_H


class FoodOrderingSystems {
};

#endif  //_FOODORDERINGSYSTEMS_H
```

5.2Admin

```
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : Admin.h
//  @ Date : 10/14/2018
//  @ Author :
//
//


#if !defined(_ADMIN_H)
#define _ADMIN_H

#include "FoodOrderingSystems.h"

class Admin : public FoodOrderingSystems {
public:
      string usecase;
      string email;
      number mobile;
```

```
      address address;
      void register(string +usecase, encrypted -password, string
+email, number +mobile, address +address);
      void login(string +usecase, encryted -password);
      void logout();
      void updateProfile(string +username, encryted -password,
string +email, number +mobile, address +adress);
private:
      encrypted password;
};

#endif  //_ADMIN_H
```

5.3 Customer

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : customer.h
//  @ Date : 10/14/2018
//  @ Author :
//
//


#if !defined(_CUSTOMER_H)
#define _CUSTOMER_H

#include "FoodOrderingSystems.h"

class customer : public FoodOrderingSystems {
public:
      string hotel_name;
      string item_name;
      string location;
      int quantity;
      string locationSharing(string +location);
      string search(string||+location:string||-area:string||
+item_name:string +hotel_name);
      void addToCart(string +item_name, int +quantity);
      string viewCart();
      string cart_includeItem(string +item_name, int +quantity);
      string cart_RemoveItem(string +item_name, int +quantity);
      string addFavorites(string||+Item_name:string
+Hotel_name);
private:
      string area;
};

#endif  //_CUSTOMER_H
```

5.4 Delivery Boy

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : deliveryBoy.h
//  @ Date : 10/14/2018
//  @ Author :
//
//


#if !defined(_DELIVERYBOY_H)
#define _DELIVERYBOY_H

#include "FoodOrderingSystems.h"

class deliveryBoy : public FoodOrderingSystems {
public:
      string d_id;
      string o_id;
      string viewOrderDetails(string +d_id, string +o_id);
      string trackOrder(string +d_id);
      void cancelOrder(string +o_id);
};

#endif  //_DELIVERYBOY_H
```

5.5 Hotel Manager

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Untitled
//  @ File Name : hotelManager.h
//  @ Date : 10/14/2018
//  @ Author :
//
//


#if !defined(_HOTELMANAGER_H)
#define _HOTELMANAGER_H

#include "FoodOrderingSystems.h"

class hotelManager : public FoodOrderingSystems {
public:
      string Hotel_id;
      string Item_name;
      float price;
      string category;
      string cuisine;
```

```
      int quantity;
      string updateMenu(string +hotel_id, int +quantity);
      string upd_AddItem(string +hostel_id, int +quantity);
      string upd_DeleteItem(string +hostel_id, int +quantity);
};

#endif  //_HOTELMANAGER_H
```

6. CONCLUSION:

Thus , the Online Food Ordering System has been clearly analysed and designed using StarUML with the required functionalities .