School of Engineering

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Computer Science and Information Systems

(SCORE)

Winter Semester 2024-2025

ISWE207P-OBJECT ORIENTED ANALYSIS AND DESIGN LAB

# System Design Document

Online Ordering System

Gugaan M

23MIS0309

L55+L56

Dr Gunasekaran G

| | |
|---|---|
| Document ID | SYSTEM DESIGN-v0.1 |
| Version Number | 0.1 |
| Issue Date | December 13, 2024 |
| Classification | Public |

**Copyright Notice**

**© COMPANYNAME, (original issue year – current issue year)**

**All Rights Reserved**

**The content in this document is the exclusive property of Zomato. Unauthorized duplication, storage in retrieval systems, or distribution of any portion of this document—whether by mechanical, electronic, or other means—is strictly prohibited without prior written permission from Zomato.This restriction includes but is not limited to translation into different languages or formats. Any unauthorized use or duplication may result in legal action.**

**The information provided here is subject to modification without prior notification and does not create any contractual obligation for Zomato. We reserve the right to make amendments to the products, services, or content described within this document at any time. Zomato assumes no liability for any direct or indirect consequences resulting from the application or use of the information contained herein.**

Revision History

| Date | Version | Description | Author (s) |
|---|---|---|---|
| 04/01/2020 | 0.1 | Draft Version | Gugaan M |
| 05/03/2025 | 0.11 | Draft Version | Gugaan M |
| | | | |

**The reviewer signoff shall signify the recommendation for acceptance of this**

| Reviewed By (Customer) | Signature | Date |
|---|---|---|
| Kaif | | 06/03/2025 |
| Arya | | 06/03/2025 |
| | | |
| | | |
| | | |

**document.**

Sign Off

| Prepared By | Acknowledged By |
|---|---|
| | |

| | |
|---|---|
| | |
| Gugaan M | Mahesh |
| Title: Developer | Title: Tester |
| Zomato | Zomato |
| Date: 06/03/2025 | Date: 06/03/2025 |

| Accepted By | Accepted By |
|---|---|
| | |
| Harini | Harshad |
| Title: Manager | Title: Manager |
| Zomato | Zomato |
| Date: 06/03/2025 | Date: 06/03/2025 |

Table of Contents

## List of Tables

## List of Figures

# Introduction

**The Online Food Ordering System is built to offer a smooth and efficient way for customers to order food online. Users can explore menus from different restaurants, place orders, complete payments, and track their deliveries. The system is designed to ensure a user-friendly interface while incorporating modern technologies to improve performance, security, and scalability.**
**By automating the ordering process, this system reduces manual errors, enhances processing speed, and provides a digital alternative to traditional phone-based food ordering. It benefits both customers and restaurant owners by simplifying operations and ensuring a higher level of service quality.**

## 1.1 Purpose of Document

**This System Design Document (SDD) outlines the design specifications of the Online Food Ordering System, including:**

- **System architecture**
- **Design approach**
- **Software components**
- **Structural and behavioral models**

**This document serves as a guide for developers, designers, and stakeholders, ensuring all design aspects align with the system's functional and non-functional requirements.**

## 1.2 Document Scope

**This document specifies the system's functionality, defining its boundaries and limitations.**

### 1.2.1 Included Features

**The system will include the following capabilities:**

- **User Registration and Authentication – Customers and administrators can sign up, log in, and manage their accounts.**
- **Menu Browsing – Users can explore food items categorized with details like pricing, ingredients, and availability.**
- **Food Ordering and Checkout – Customers can add food items to their cart and place orders.**
- **Payment Processing – Transactions will be secured using third-party gateways such as PayPal and Stripe.**
- **Order Tracking – Users can monitor their order status in real-time, from confirmation to delivery.**
- **Admin Panel for Restaurant Management – Administrators can oversee menus, pricing, restaurant details, and order statuses.**

### 1.2.2 Excluded Features

The system will not include:

- Inventory Management – The platform does not track stock levels of ingredients.
- Kitchen Workflow Tracking – The cooking process and chef assignments are beyond the system's scope.
- Delivery Management – The system does not assign delivery personnel but relies on third-party services.

### 1.2.3 Assumptions

The system operates under the following assumptions:

- Users will require an active internet connection.
- Secure third-party services will handle payment processing.
- The system will be accessible via web browsers and mobile applications.
- Restaurants will be responsible for keeping their menus updated.
- Initial support will be for a limited number of restaurants, with scalability options available.

## 1.3 Methodology, Tools, and Approach

The system is developed using Object-Oriented Analysis and Design (OOAD) principles and follows Agile methodology, employing the following technologies:

- Development Approach: OOAD, Agile
- Programming Languages: Java (Spring Boot backend), JavaScript (React.js frontend)
- Database: MySQL for structured data storage
- Tools Used:
    - PlantUML for UML diagrams
    - Visual Paradigm for advanced modeling
    - Postman for API testing
    - Docker for deployment
    - JIRA/Trello for project management

Development Strategy:

- UML diagrams will be used for structured system analysis.
- An incremental approach with iterative testing and continuous feedback will be followed.
- The architecture will be designed for scalability and adaptability to future enhancements.

## 1.4 Acronyms and Abbreviations

- GUI - Graphical User Interface
- SDD - System Design Document

- **API - Application Programming Interface**
- **DBMS - Database Management System**
- **MVC - Model-View-Controller**
- **REST - Representational State Transfer**

---

## 2. Design Overview

### 2.1 Background Information

**The rapid expansion of online food delivery services has increased the demand for fast and convenient ordering options. Traditional phone-based ordering is often inefficient, leading to errors and long wait times. This Online Food Ordering System aims to solve these challenges by offering a structured, automated, and scalable digital solution.**

**The system provides:**

- **A centralized hub for browsing restaurant menus and placing orders.**
- **Secure payment transactions through multiple options.**
- **Real-time tracking for a better customer experience.**
- **Administrative tools for restaurant owners to manage their operations.**

### 2.2 System Expansion Possibilities

**The system is designed for easy scalability, with potential future upgrades such as:**

- **AI-driven Recommendations – Custom suggestions based on order history.**
- **Voice-enabled Ordering – Integration with virtual assistants such as Alexa and Google Assistant.**
- **Customer Loyalty and Rewards – Discount programs and membership benefits.**
- **Augmented Reality (AR) Menus – Interactive previews of food items before ordering.**

### 2.3 System Requirements

**The system requires dedicated environments for development, testing, and deployment.**

**Product/Deployment Environment**

| **Product** | **Environment** |
|---|---|
| **Online Ordering System** | **Production, Development, Testing** |

**Hardware and Software Requirements**

- **Hardware:**
  - **Cloud-based hosting (AWS, Google Cloud, or Azure)**
  - **Devices: Desktop, Laptop, Tablet, Mobile**
- **Software:**
  - **Backend: Spring Boot, MySQL, REST APIs**

- Frontend: React.js, Bootstrap
- Security: SSL encryption, OAuth authentication

### 2.4 Constraints

1. **Technical Limitations:**
   - Limited server capacity may impact performance during peak usage.
   - Cloud infrastructure scaling may be necessary to handle increasing demand.
2. **User Device Compatibility:**
   - The platform must be responsive across multiple devices, including desktops, tablets, and mobile phones.
3. **Third-Party Services:**
   - The system relies on external payment processors and APIs, which may experience downtime.
4. **Security and Compliance:**
   - Compliance with regulations like PCI DSS is required for secure payment handling.
5. **Performance Considerations:**
   - The system must ensure quick response times using caching, load balancing, and optimized queries.

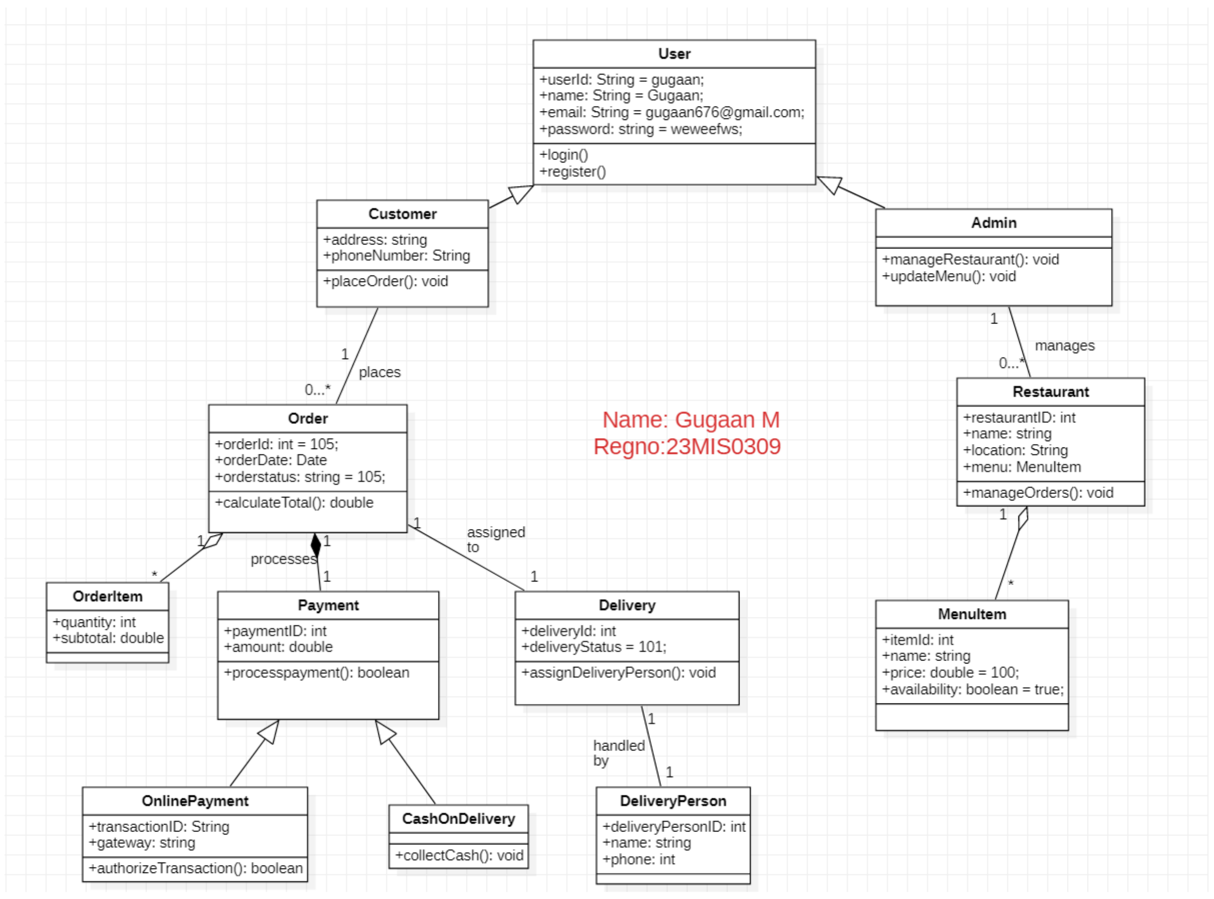
### 2.5 Design Decisions

Several trade-offs were considered during the system design process:

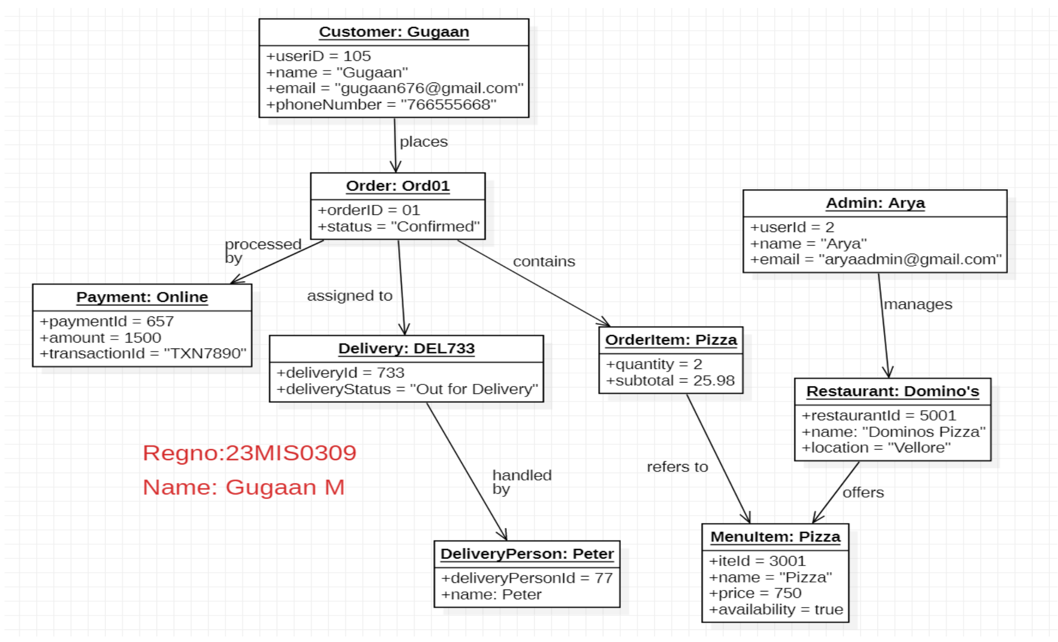
| Factor | Decision Taken |
|---|---|
| Security vs. Speed | Encryption and authentication measures were prioritized, even at the cost of slight processing delays. |
| User Simplicity vs. Features | A clean, intuitive UI was chosen over excessive customization options to improve user experience. |
| Cost vs. Expandability | Initial hosting is on a shared cloud setup, with options to transition to a dedicated environment as demand grows. |

# 3 Structural Family Diagrams

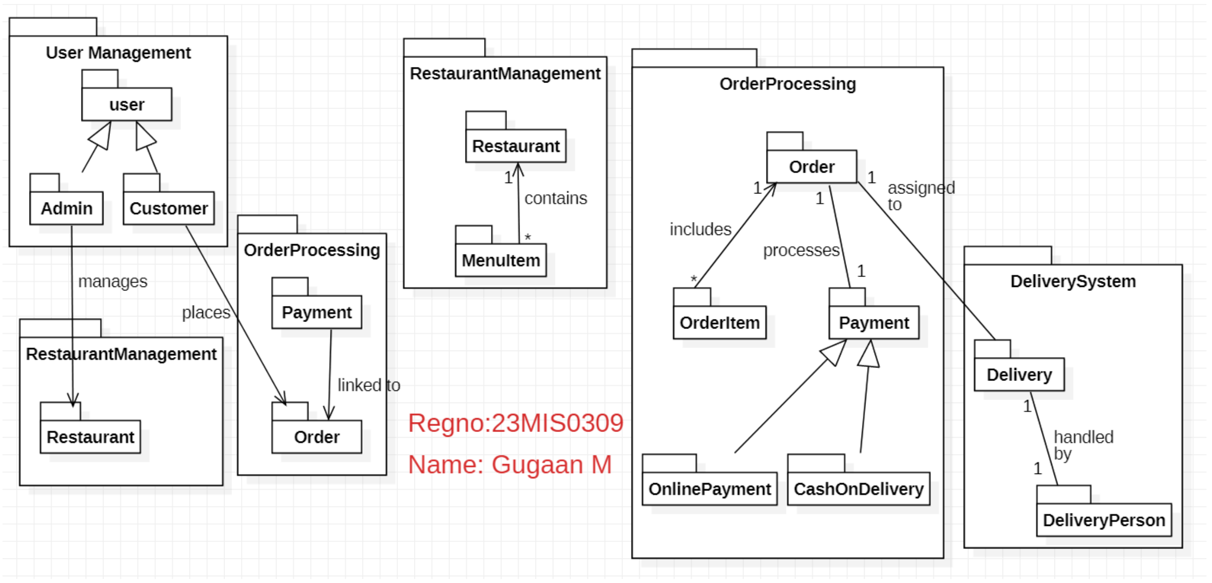
## 3.1 Class Diagram (fig no.1)



## 3.2 Object Diagram (fig no.2)

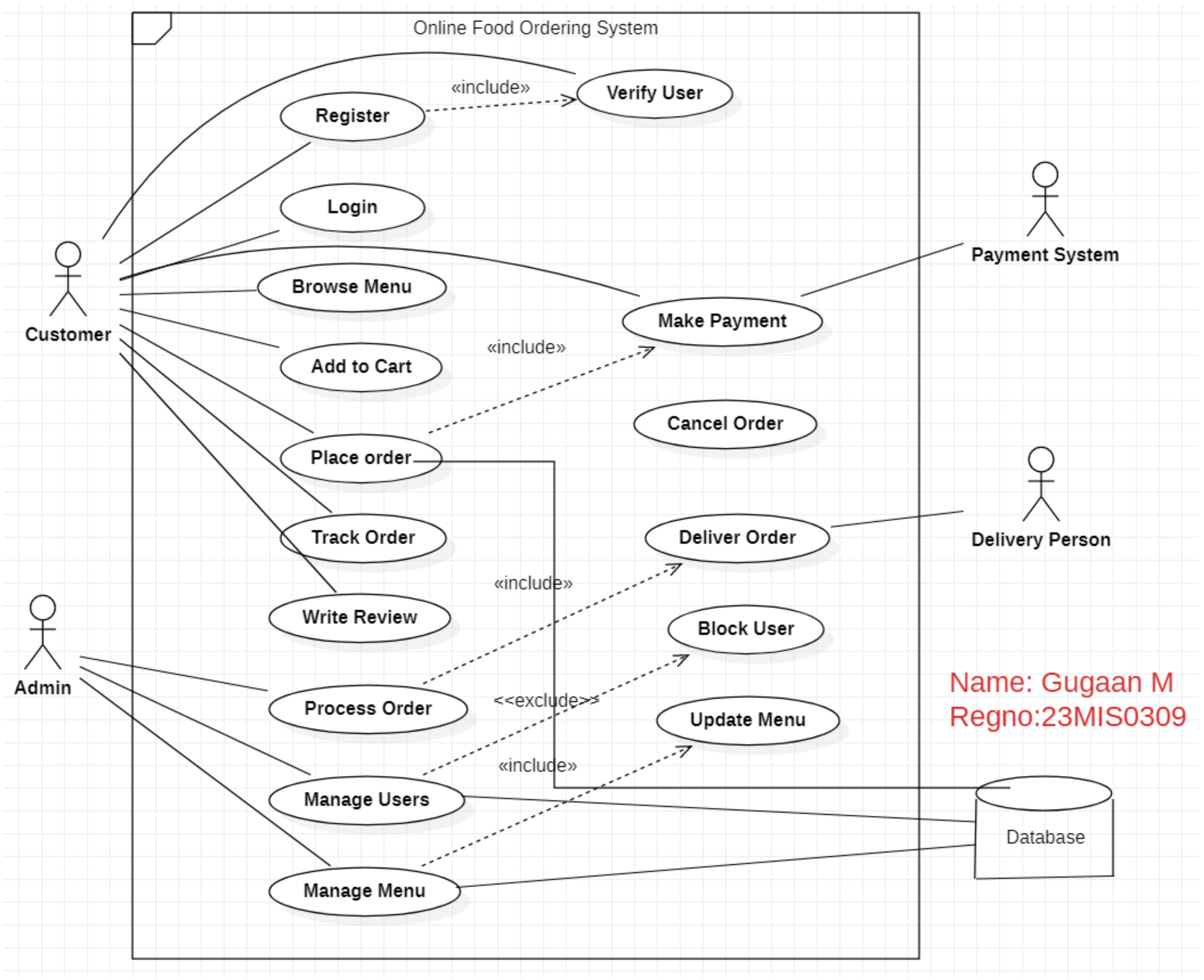


3.3 Package Diagram (fig no.3)



4 Behavioral Family Diagrams

4.1 Use Case Diagram (fig no.4)



## 4.2 Use Case Descriptions

**Use Case 1: Register**

**Name: Register a New Account**
**Brief Description: The customer can create a new account by providing personal details such as name, email, and password.**

**Actors: Customer**
**Preconditions:**

- **The system must be accessible and running.**
- **The customer must provide valid personal details.**

**Basic Flow:**

1. **The customer selects the "Register" option.**
2. **The system displays a registration form.**
3. **The customer enters their details (name, email, password).**
4. **The system validates the details and creates a new account.**
5. **The system confirms the registration and logs the customer in.**

**Alternate Flows:**

- **If the email is already registered, the system prompts the customer to use a different email.**

**Exception Flows:**

- **If the customer leaves any required field blank, the system displays an error message.**

**Postconditions:**

- **A new customer account is created.**
- **The customer is logged into the system.**

---

**Use Case 2: Verify User**

**Name: Verify User Identity**
**Brief Description: After registration, the system verifies the user’s identity.**

**Actors: Customer**
**Preconditions:**

- **The customer must have registered successfully.**

**Basic Flow:**

1. **The system sends a verification email or OTP.**
2. **The customer enters the verification code.**
3. **The system confirms the verification.**

**Alternate Flows:**

- **If the user does not receive the OTP, they can request a resend.**

**Exception Flows:**

- **If the OTP entered is incorrect, the system prompts the user to re-enter it.**

**Postconditions:**

- **The user's identity is verified.**

---

**Use Case 3: Login**

**Name: Login to the System**
**Brief Description: Customers and admins can log into the system using their credentials.**

**Actors: Customer, Admin**
**Preconditions:**

- **The user must have a registered account.**
- **The system must be accessible.**

**Basic Flow:**

1. **The user selects the "Login" option.**
2. **The system displays a login form.**
3. **The user enters their email and password.**
4. **The system verifies the credentials.**
5. **The user is granted access to their respective dashboard.**

**Alternate Flows:**

- **If the password is incorrect, the system prompts the user to re-enter it.**

**Exception Flows:**

- **If the user enters incorrect credentials three times, the account is temporarily locked.**

**Postconditions:**

- **The user gains access to their account.**

---

**Use Case 4: Browse Menu**

**Name: View Available Menu Items**
**Brief Description: Customers can browse the available food items in the restaurant menu.**

**Actors: Customer**
**Preconditions:**

- **The system must have menu items stored.**

**Basic Flow:**

1. **The customer selects the "Browse Menu" option.**
2. **The system retrieves and displays available food items with details.**
3. **The customer can view the menu and select items.**

**Alternate Flows:**

- **If no items are available, the system displays a message saying, "Currently, no items are available."**

**Postconditions:**

- **The customer can see the restaurant menu.**

---

**Use Case 5: Add to Cart**

**Name: Add Food Items to Cart**
**Brief Description: Customers can add food items to their shopping cart.**

**Actors: Customer**
**Preconditions:**

- **The customer must be logged in.**
- **The system must display available food items.**

**Basic Flow:**

1. **The customer selects a food item from the menu.**
2. **The customer clicks the "Add to Cart" button.**

3. **The system adds the item to the customer's shopping cart.**
4. **The system updates the cart total.**

**Alternate Flows:**

- **If the item is out of stock, the system notifies the customer.**

**Exception Flows:**

- **If the system fails to add the item, it displays an error message.**

**Postconditions:**

- **The selected item is added to the customer's cart.**

---

**Use Case 6: Place Order**

**Name: Place an Order**
**Brief Description: The customer places an order after adding items to the cart.**

**Actors: Customer**
**Preconditions:**

- **The customer must have items in their cart.**

**Basic Flow:**

1. **The customer proceeds to checkout.**
2. **The system displays the order summary and delivery address form.**
3. **The customer confirms the order and provides the delivery address.**
4. **The system processes the order and generates an order ID.**
5. **The system confirms the order and provides the order details.**

**Alternate Flows:**

- **If the delivery address is invalid, the system prompts the customer to enter a valid address.**

**Exception Flows:**

- **If the payment fails, the system cancels the order and notifies the customer.**

**Postconditions:**

- **The order is placed successfully.**

---

**Use Case 7: Make Payment**

**Name: Make Payment for Order**
**Brief Description: The customer makes a payment for the placed order.**

**Actors: Customer, Payment System**
**Preconditions:**

- **The order must be placed.**
- **The customer must have a valid payment method.**

**Basic Flow:**

1. **The customer selects the "Make Payment" option.**
2. **The system prompts for a payment method.**
3. **The customer provides payment details.**
4. **The system processes the payment.**
5. **The system confirms payment success.**

**Alternate Flows:**

- **If the payment system is slow, the customer is shown a loading screen.**

**Exception Flows:**

- **If the payment fails, the system notifies the customer and cancels the order.**

**Postconditions:**

- **The payment is successfully processed.**

---

**Use Case 8: Track Order**

**Name: Track Order Status**
**Brief Description: The customer can track the status of their placed order.**

**Actors: Customer**
**Preconditions:**

- **The customer must have placed an order.**

**Basic Flow:**

1. **The customer selects "Track Order".**
2. **The system asks for the order ID.**

3. **The customer enters the order ID.**
4. **The system retrieves and displays the current order status.**

**Alternate Flows:**

- **If the order ID is invalid, the system prompts the customer to enter the correct ID.**

**Exception Flows:**

- **If the system cannot retrieve the order status, it displays an error message.**

**Postconditions:**

- **The customer views the current status of their order.**

---

**Use Case 9: Cancel Order**

**Name: Cancel a Placed Order**
**Brief Description: The customer can cancel an order before it is processed.**

**Actors: Customer**
**Preconditions:**

- **The order must be in a "pending" state.**

**Basic Flow:**

1. **The customer selects "Cancel Order".**
2. **The system verifies the cancellation eligibility.**
3. **The system cancels the order.**
4. **The customer receives a cancellation confirmation.**

**Exception Flows:**

- **If the order is already dispatched, the system does not allow cancellation.**

**Postconditions:**

- **The order is canceled successfully.**

---

**Use Case 10: Write Review**

**Name: Provide Feedback on Order**
**Brief Description: The customer can review a restaurant after order completion.**

**Actors: Customer**
**Preconditions:**

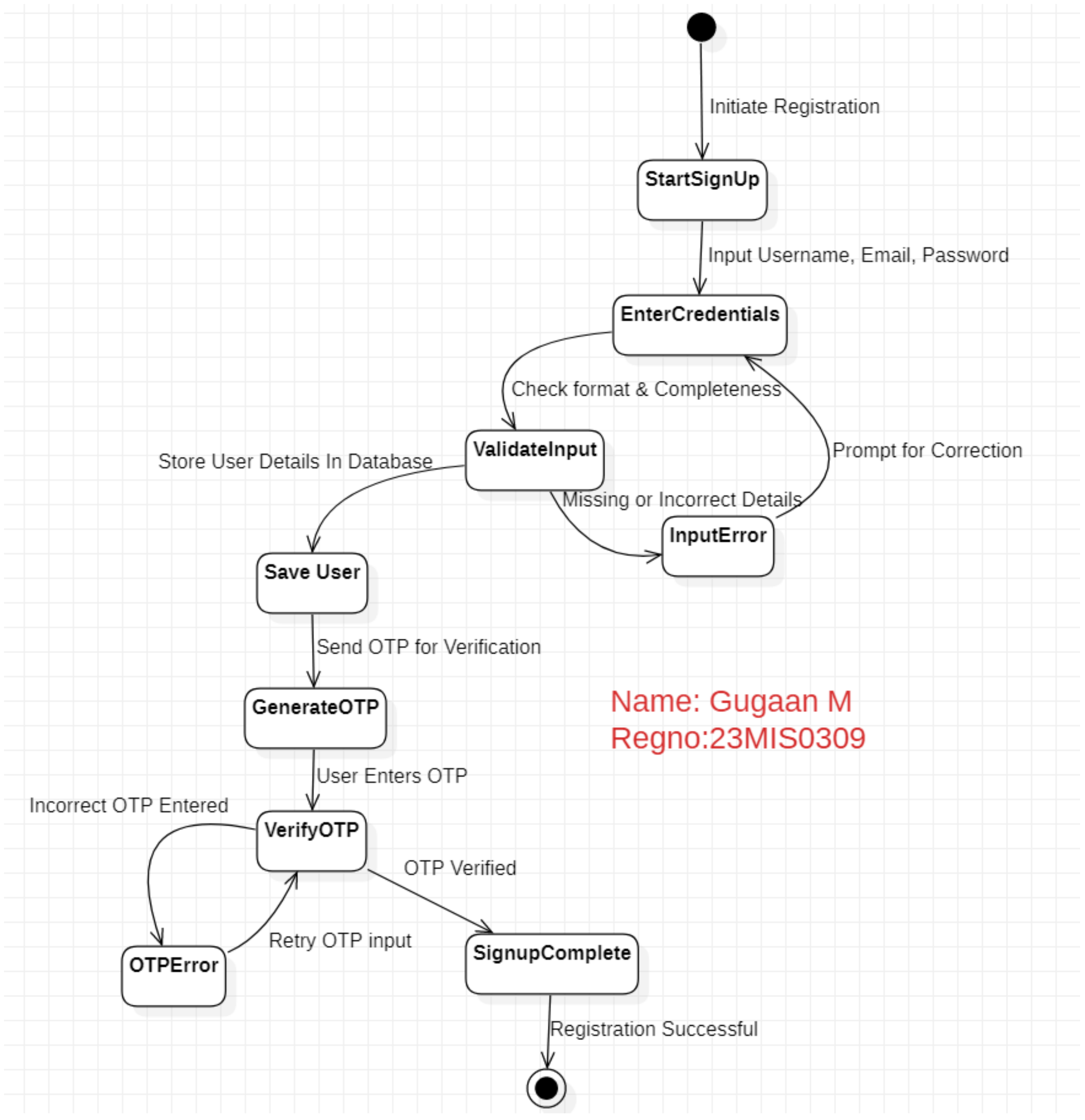
- **The customer must have completed an order.**

**Postconditions:**

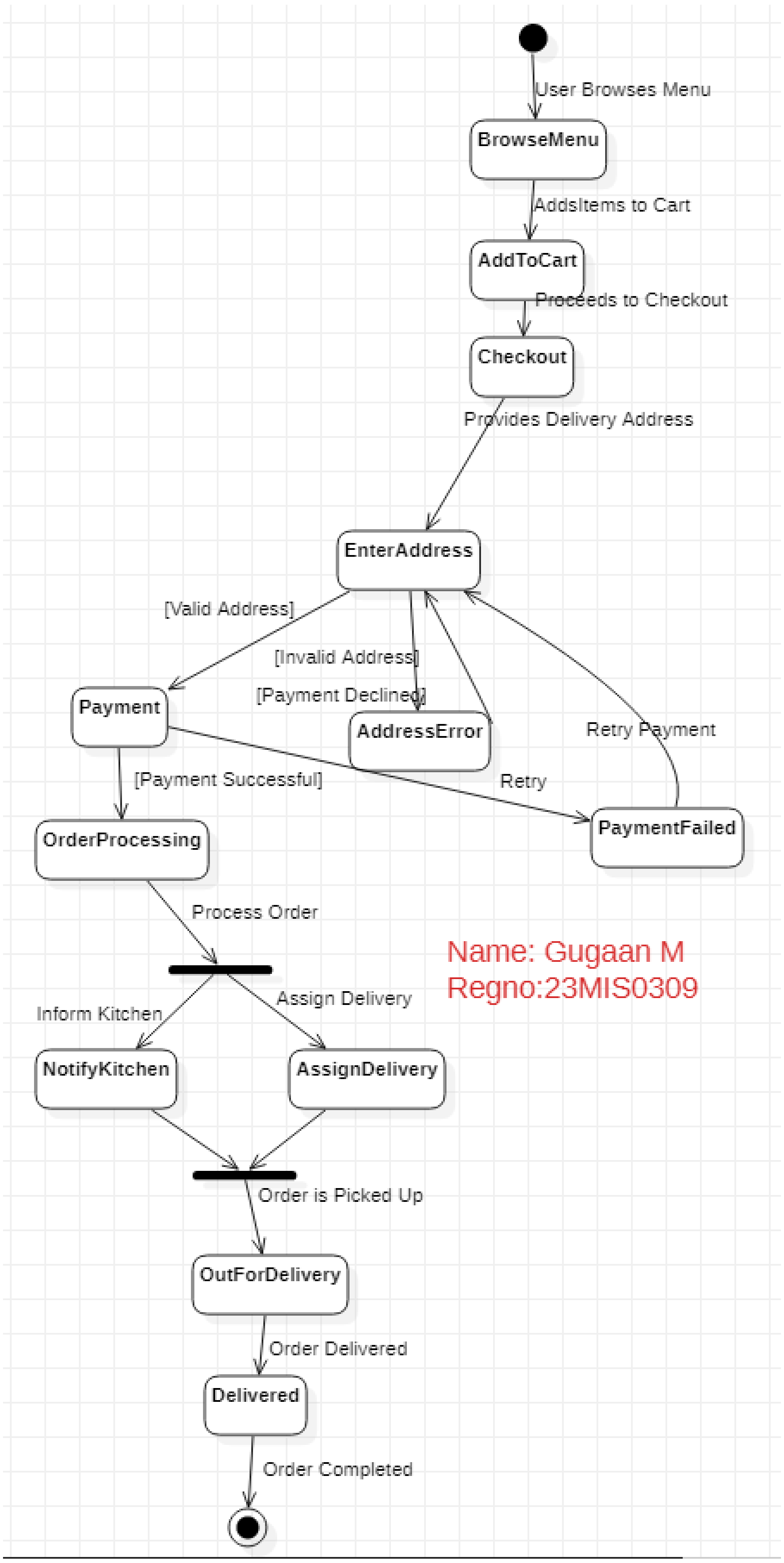
- **The review is stored in the system.**

4.3 State Chart Diagram

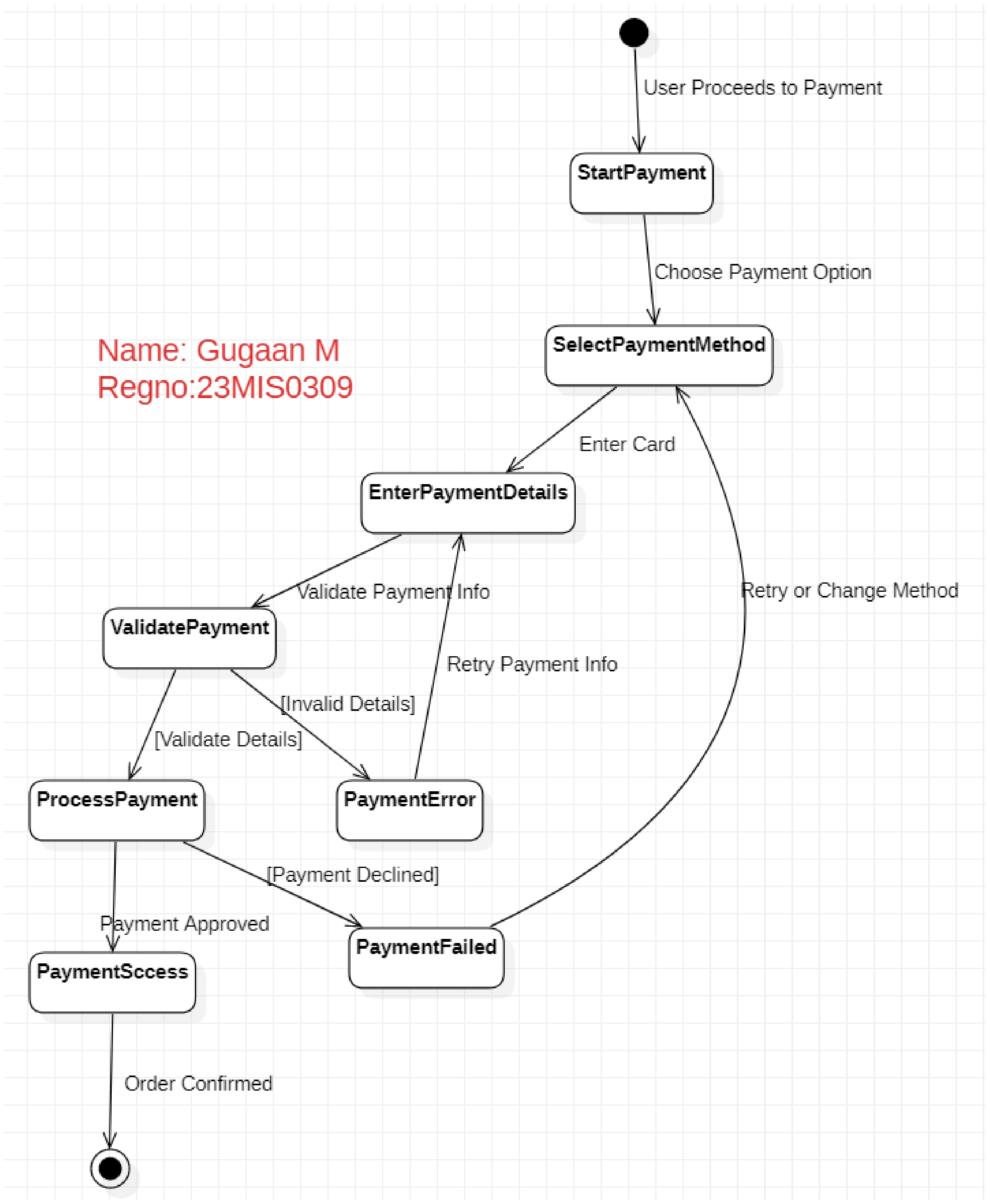
1. State Chart Diagram for Register (User Registration) (fig no.5)



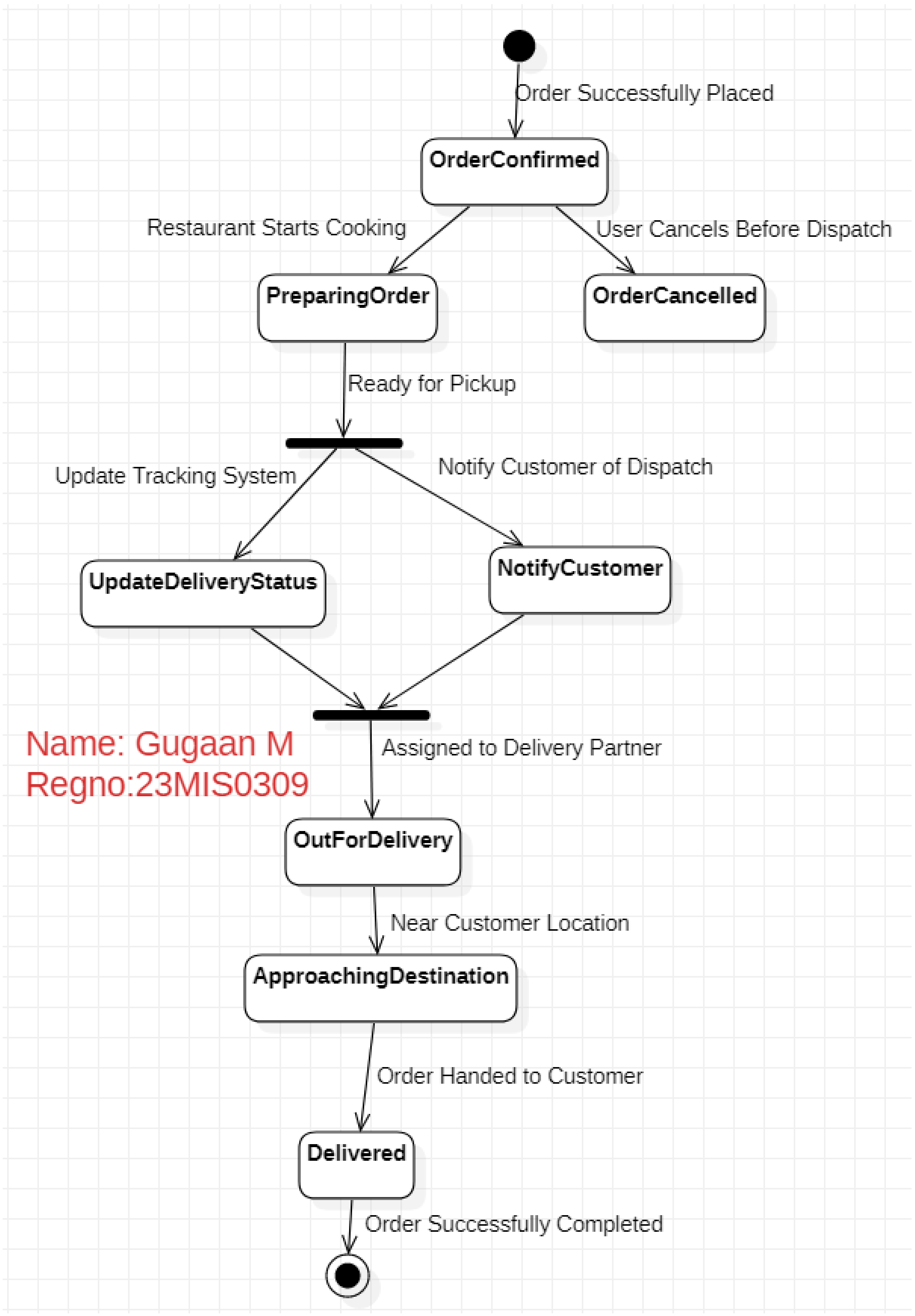
2. State Chart Diagram for Place Order (fig no.6)



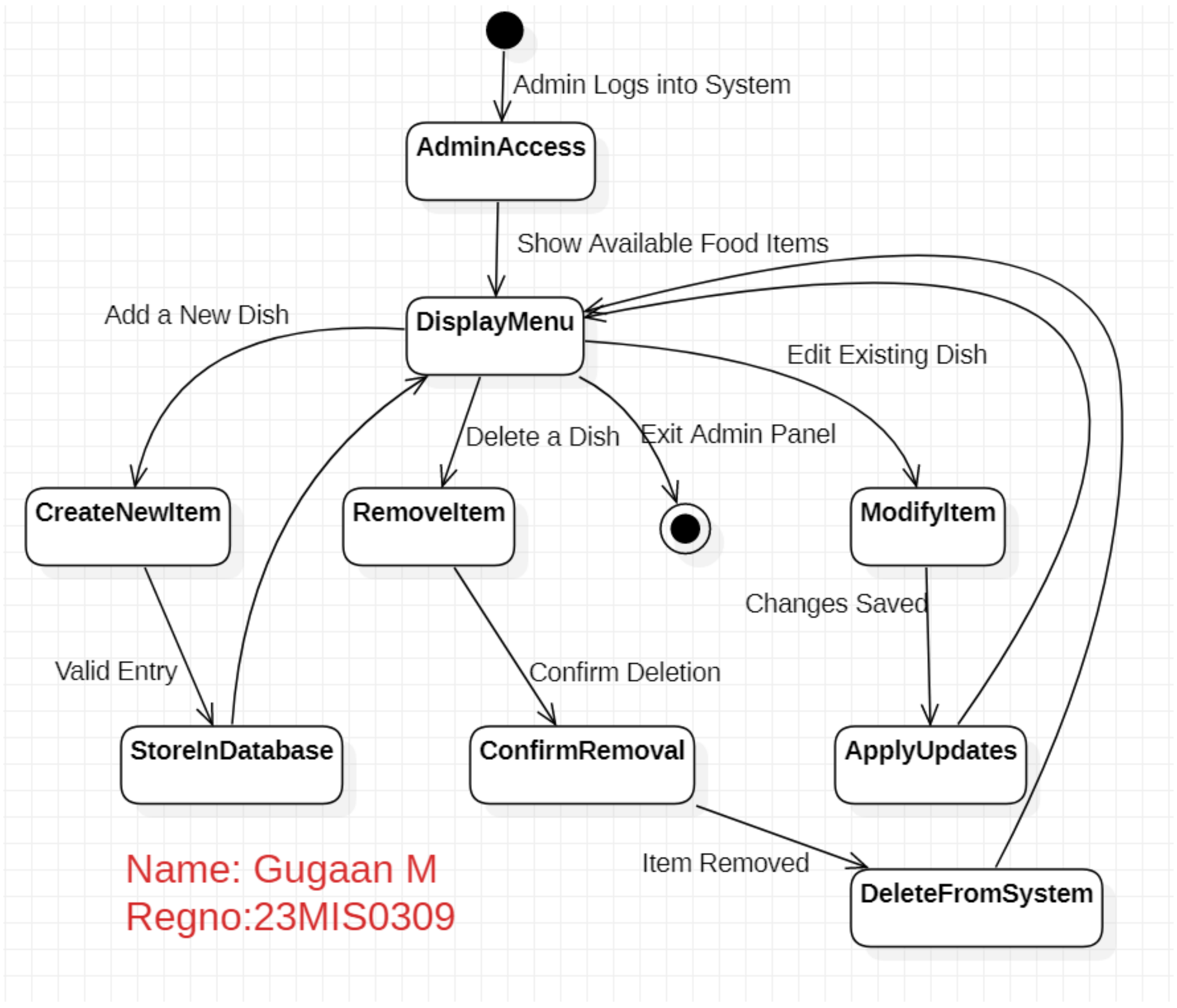
3. State Chart Diagram for Make Payment (fig no.7)



4. State Chart Diagram for Track Order (fig no.8)

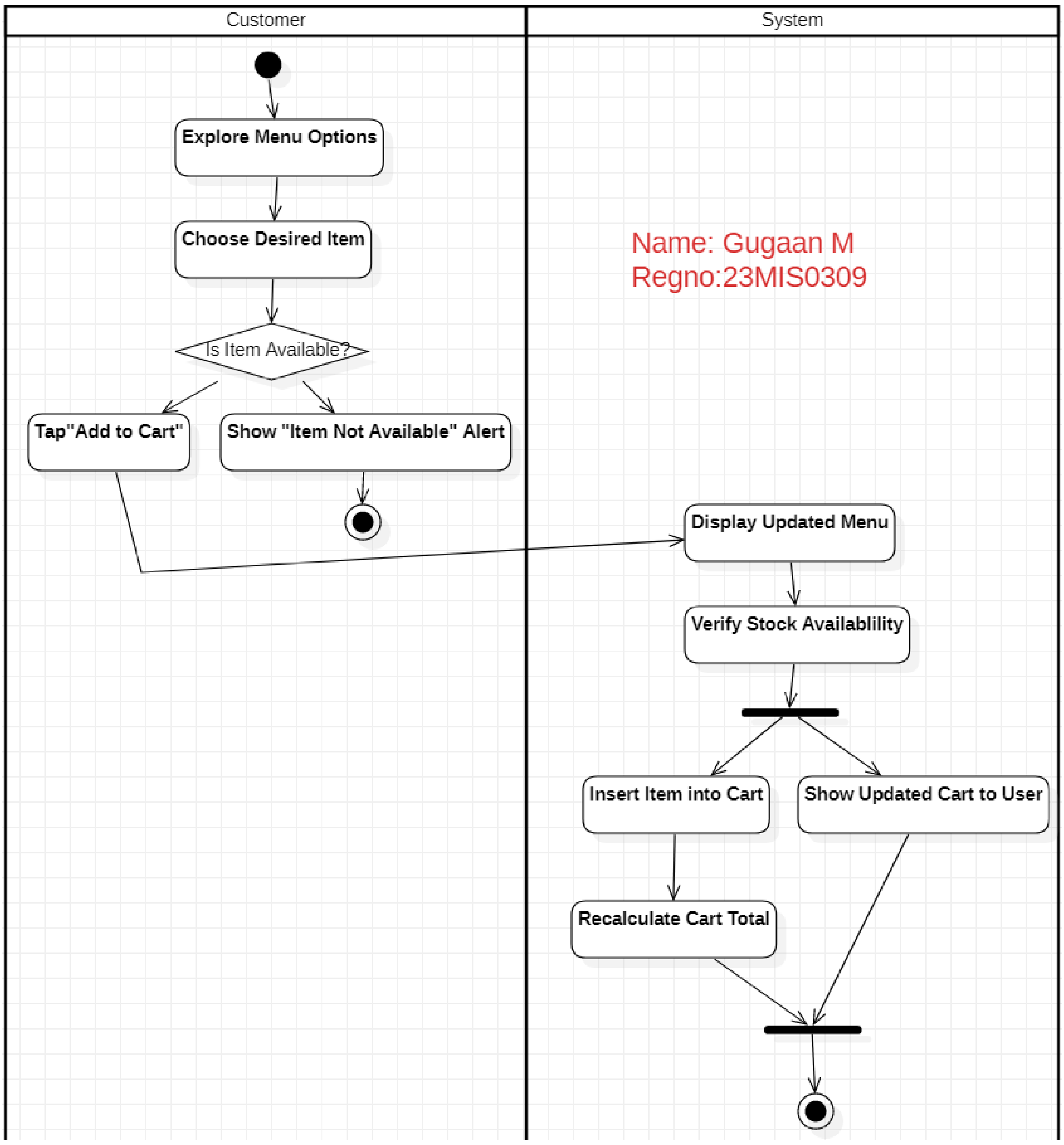


5. State Chart Diagram for Manage Food Items (Admin) (figno. 9)

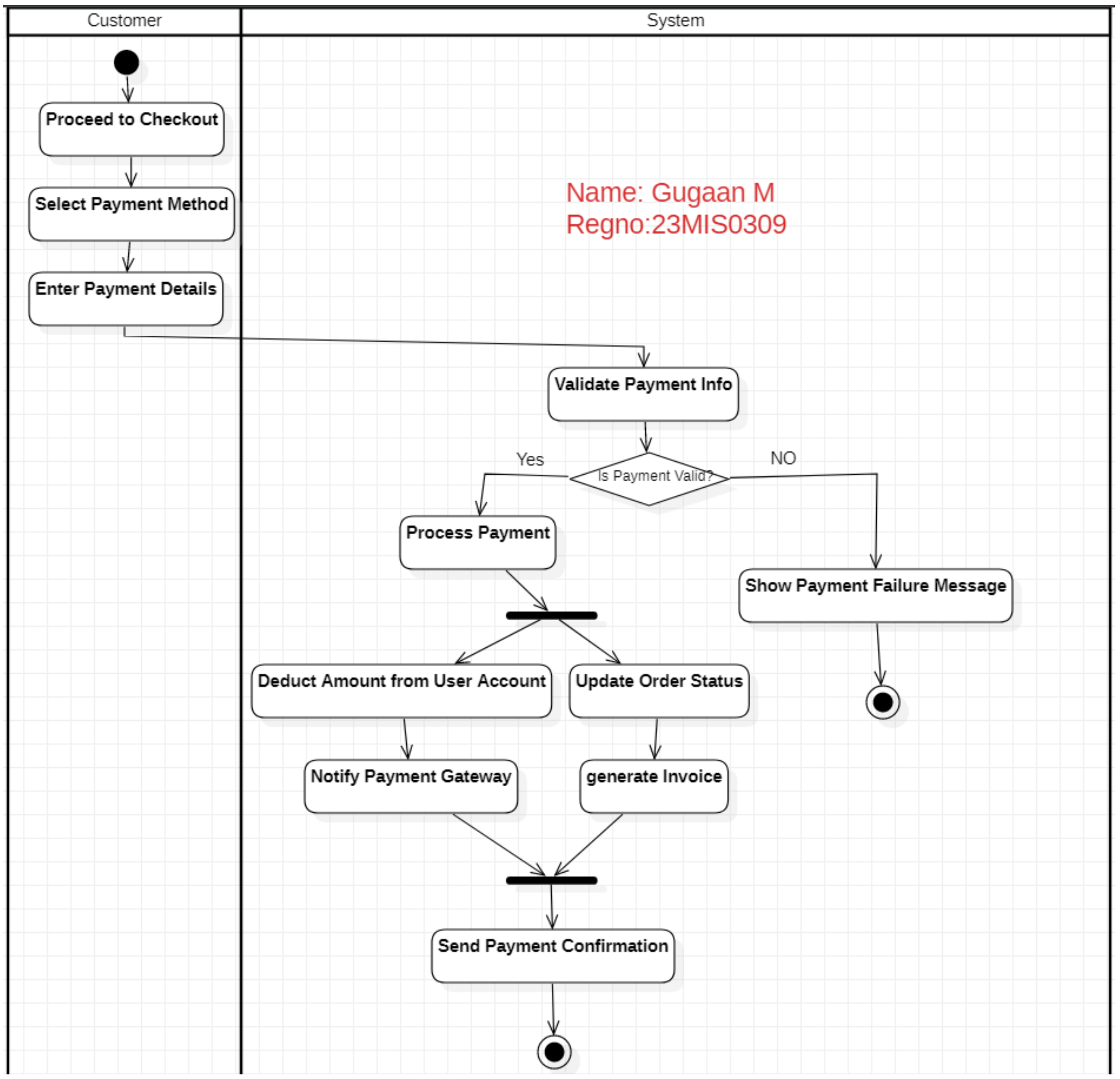


4.4 An activity Diagram

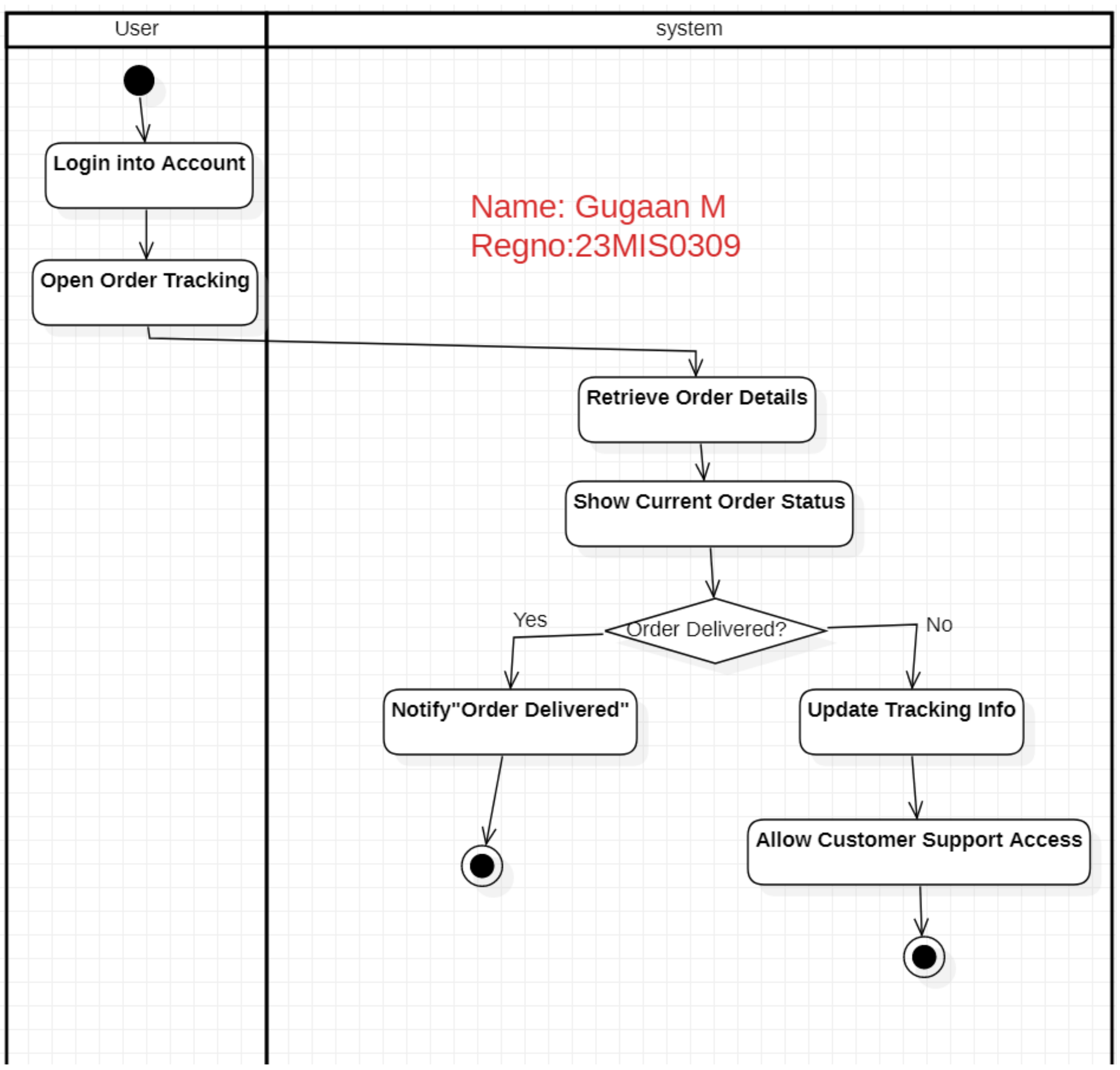
1. Activity Diagram for Add to Cart (fig no.10)



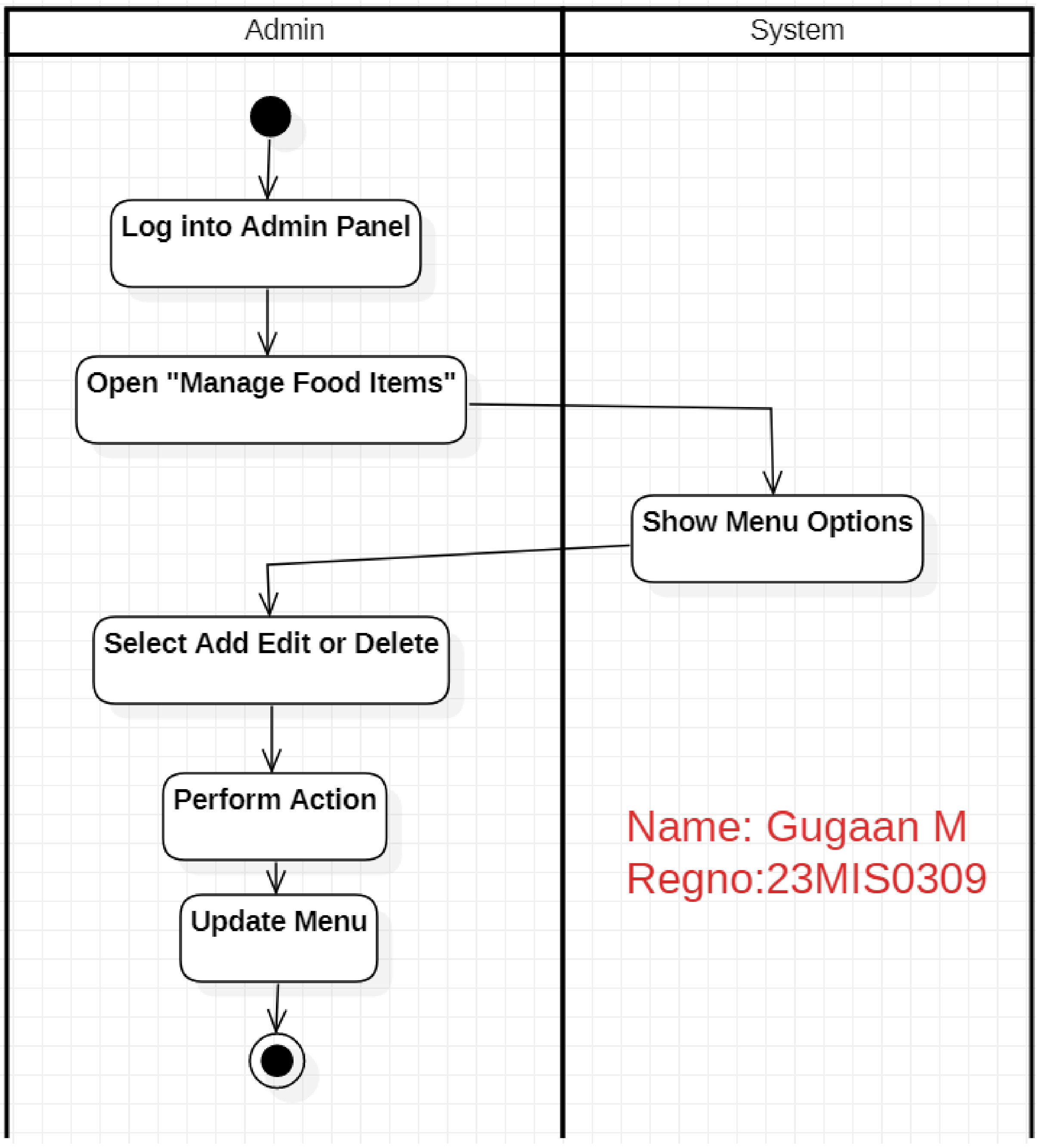
2. Activity Diagram for Payment Process (fig no. 11)



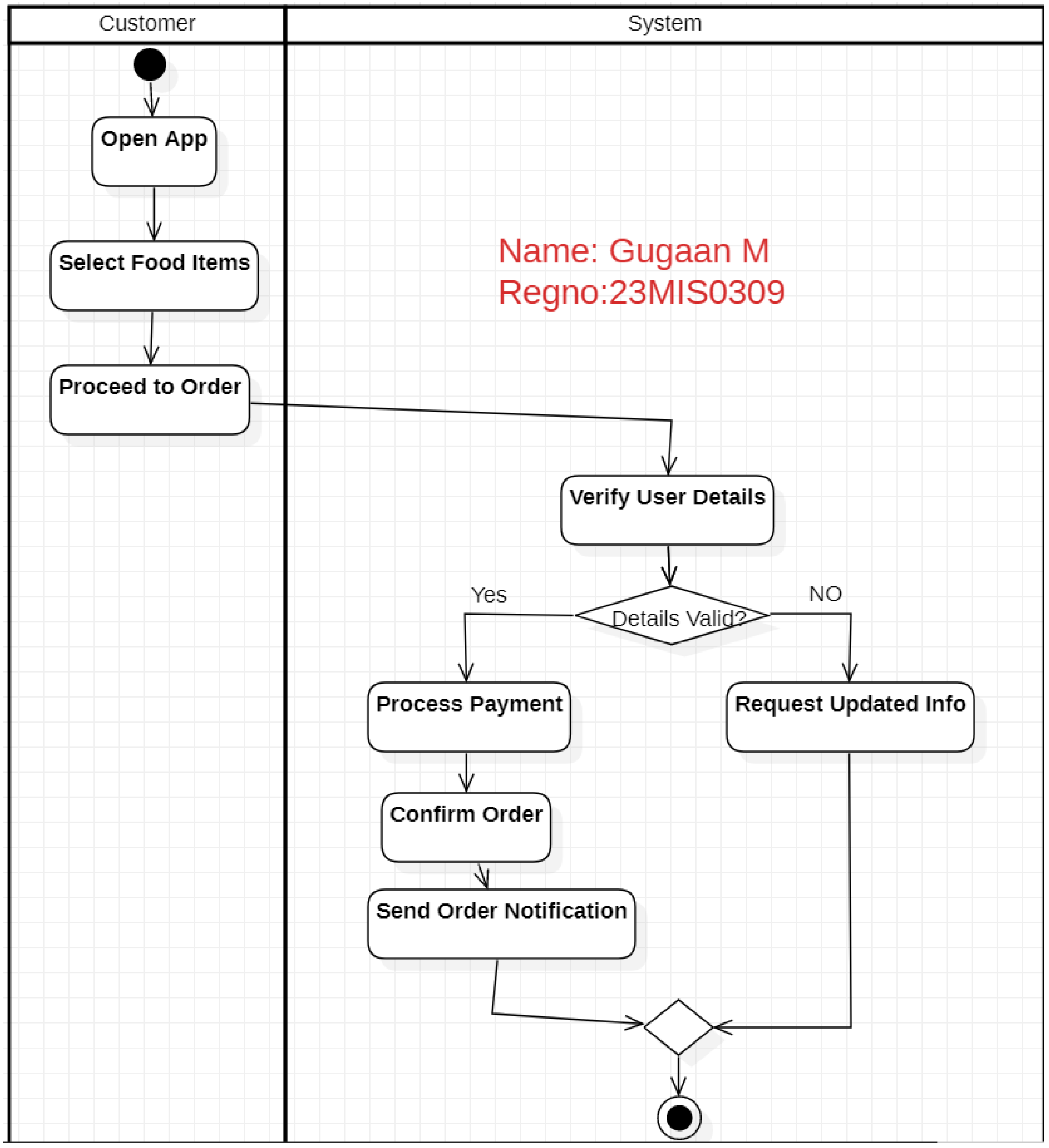
3. Activity Diagram for Order Tracking (fig no .12)



4. Activity Diagram for Managing Food Items (Admin) (fig no .13)

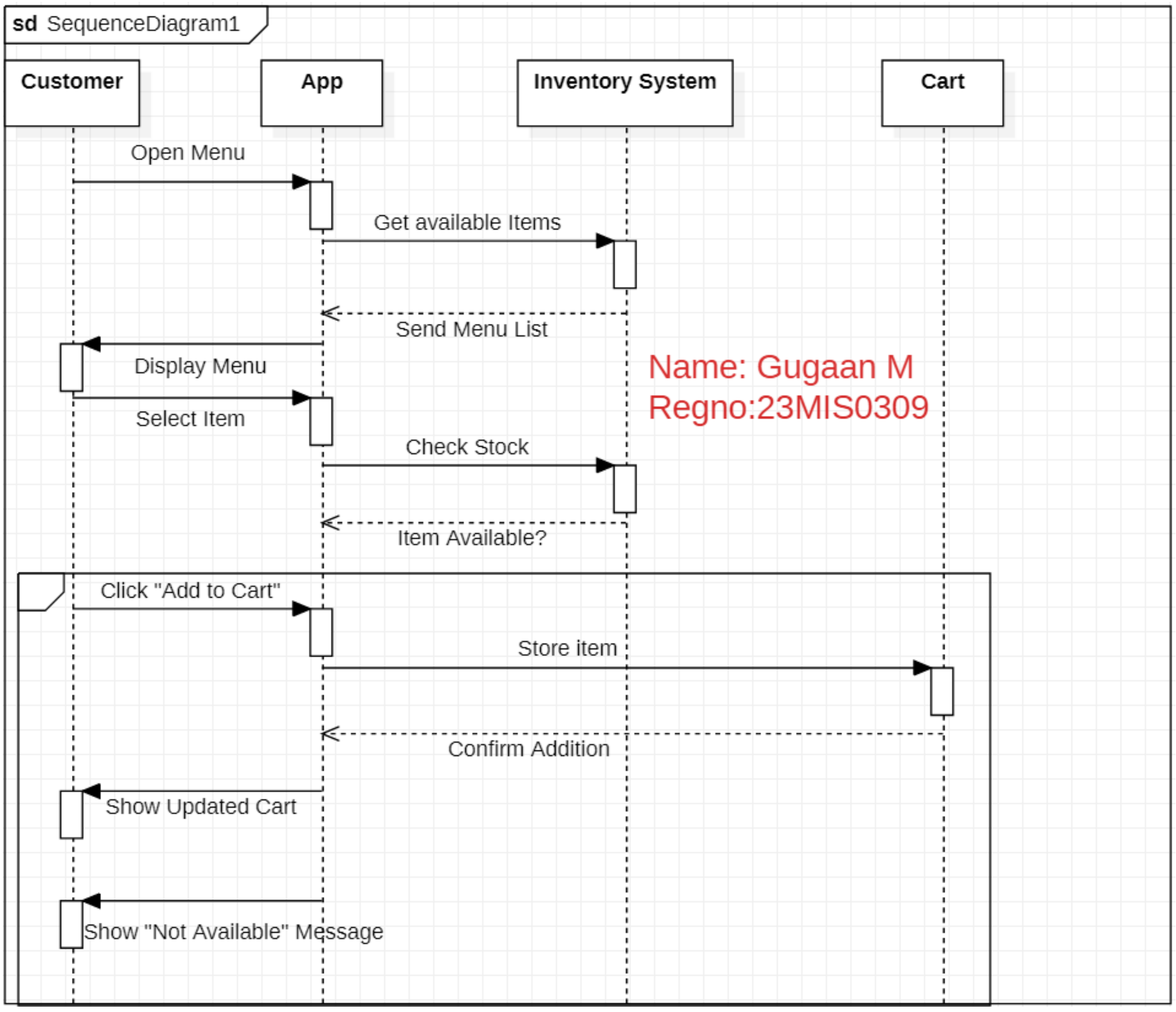


5. Activity Diagram for Order Placement (fig no.14)

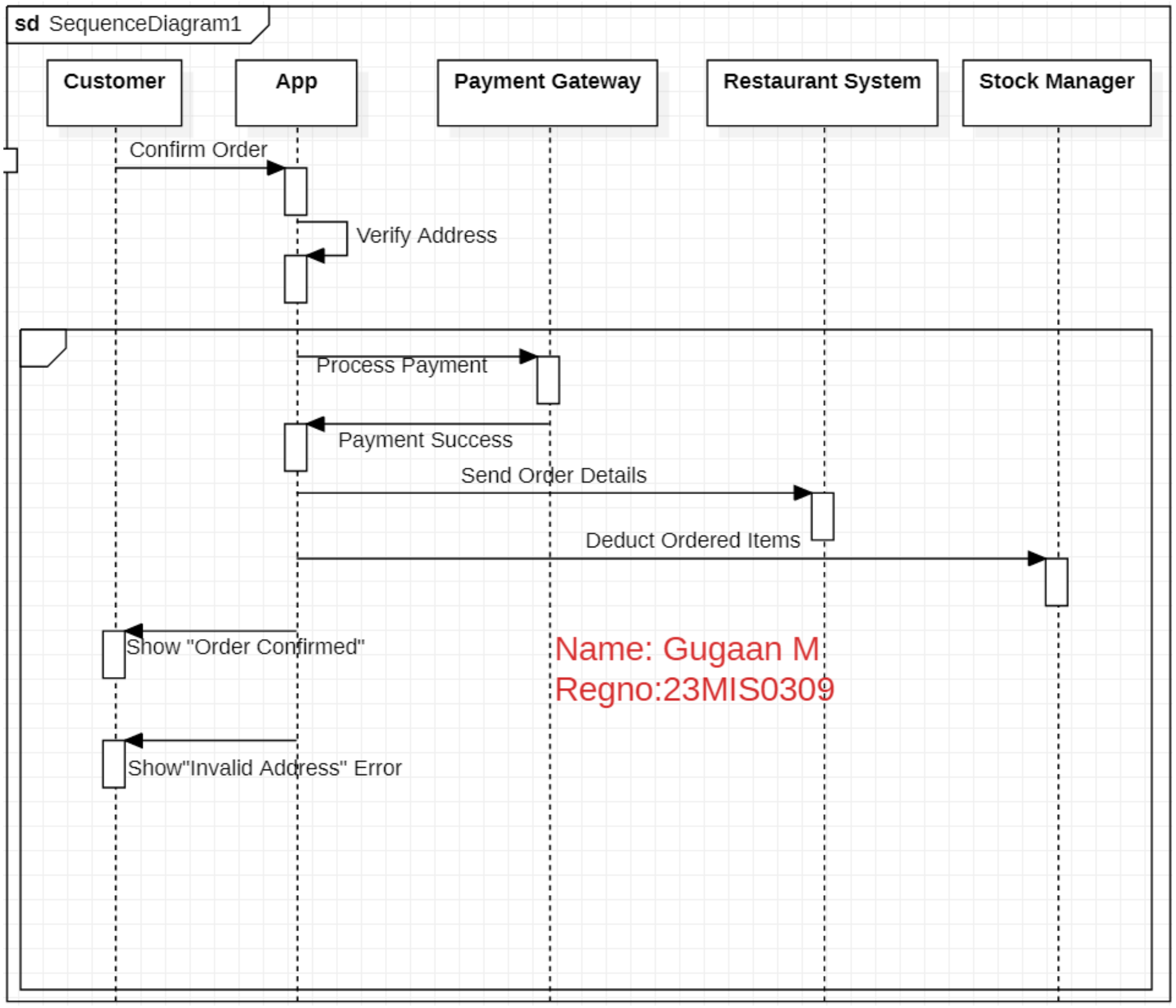


4.5 Sequence Diagram

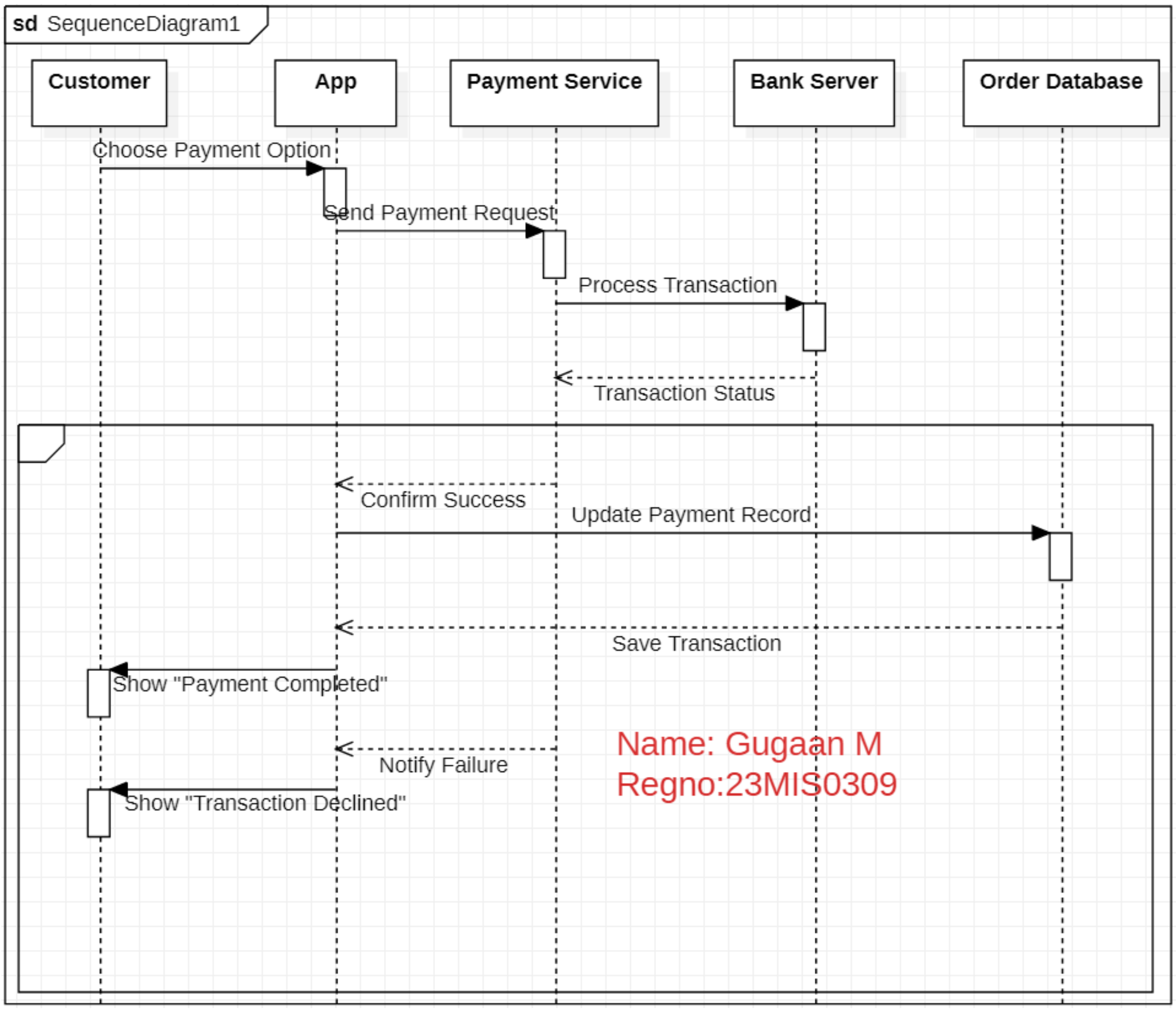
1.Sequence Diagram for Add to Cart (fig no.15)



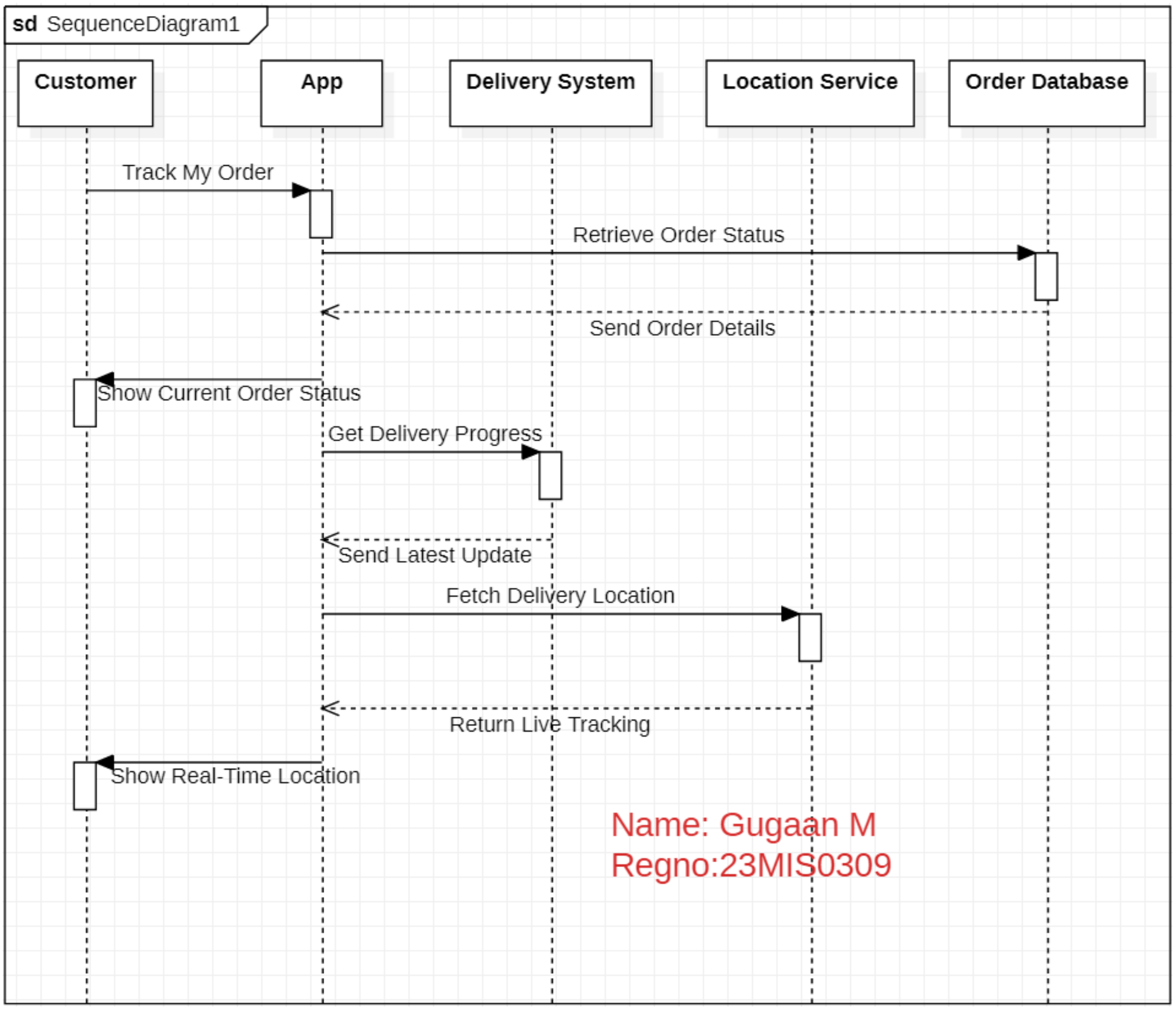
2.Sequence Diagram for Order Placement (fig no.16)



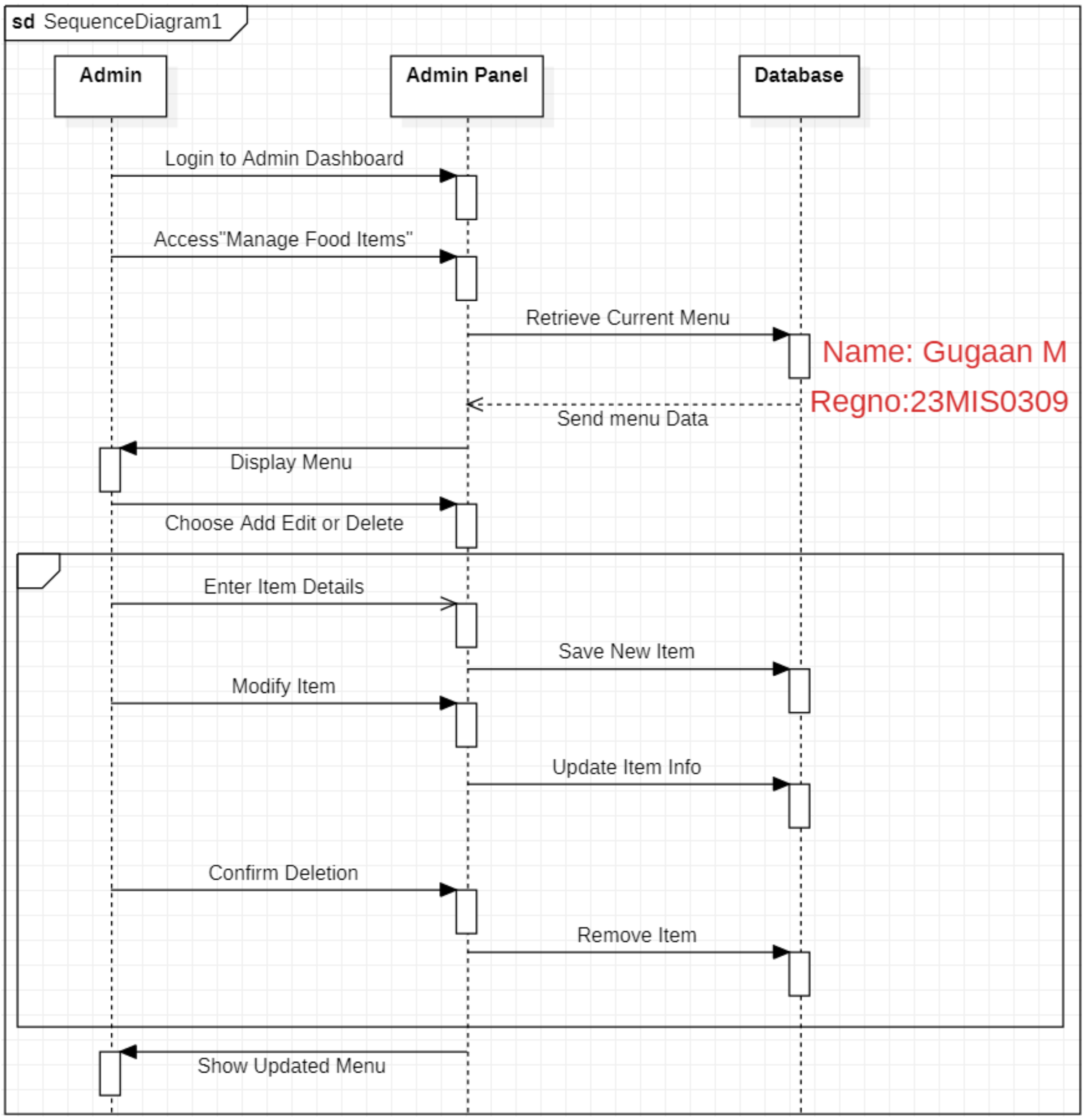
3.Sequence Diagram for Payment Processing (fig no.17)



4.Sequence Diagram for Order Tracking (fig no.18)

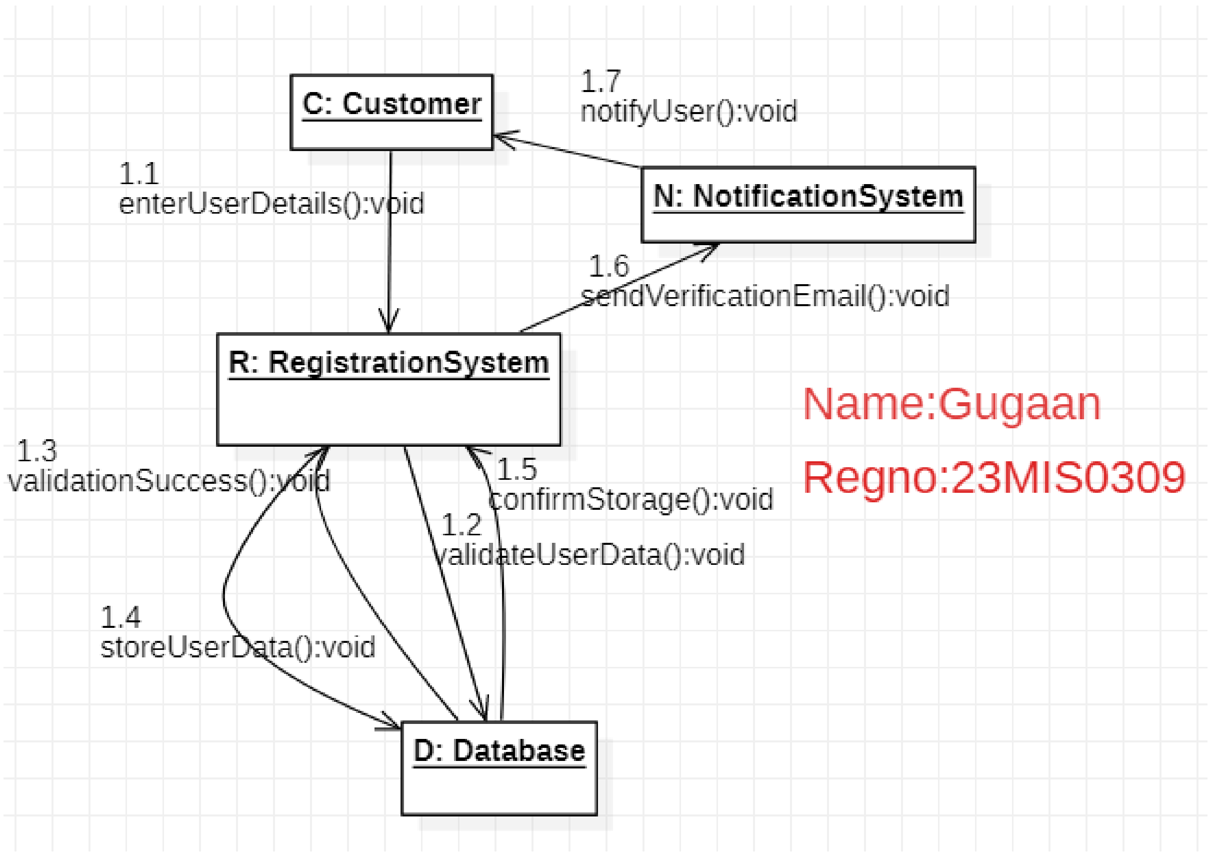


5.Sequence Diagram for Admin Managing Food Items (fig no.19)

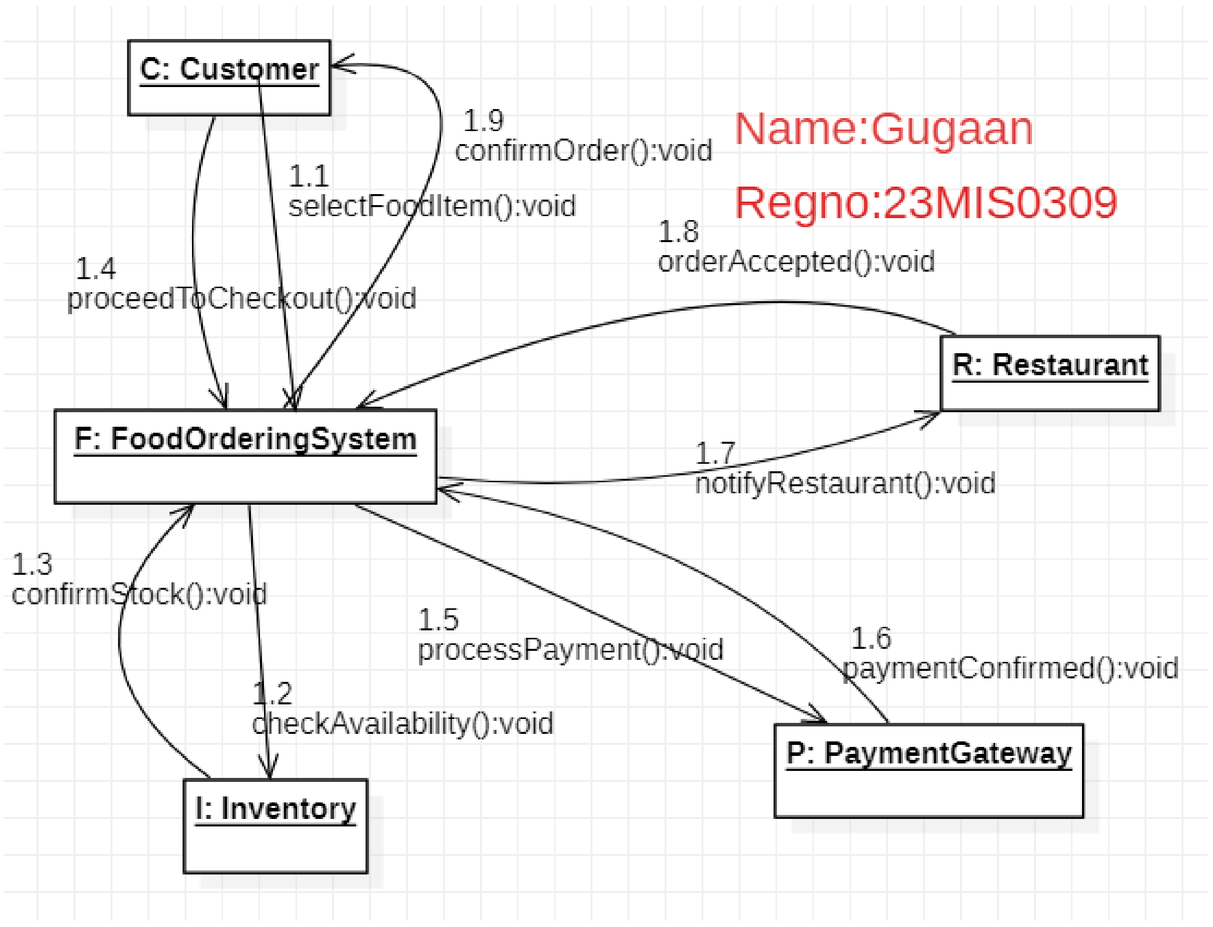


4.6 Collaboration Diagram

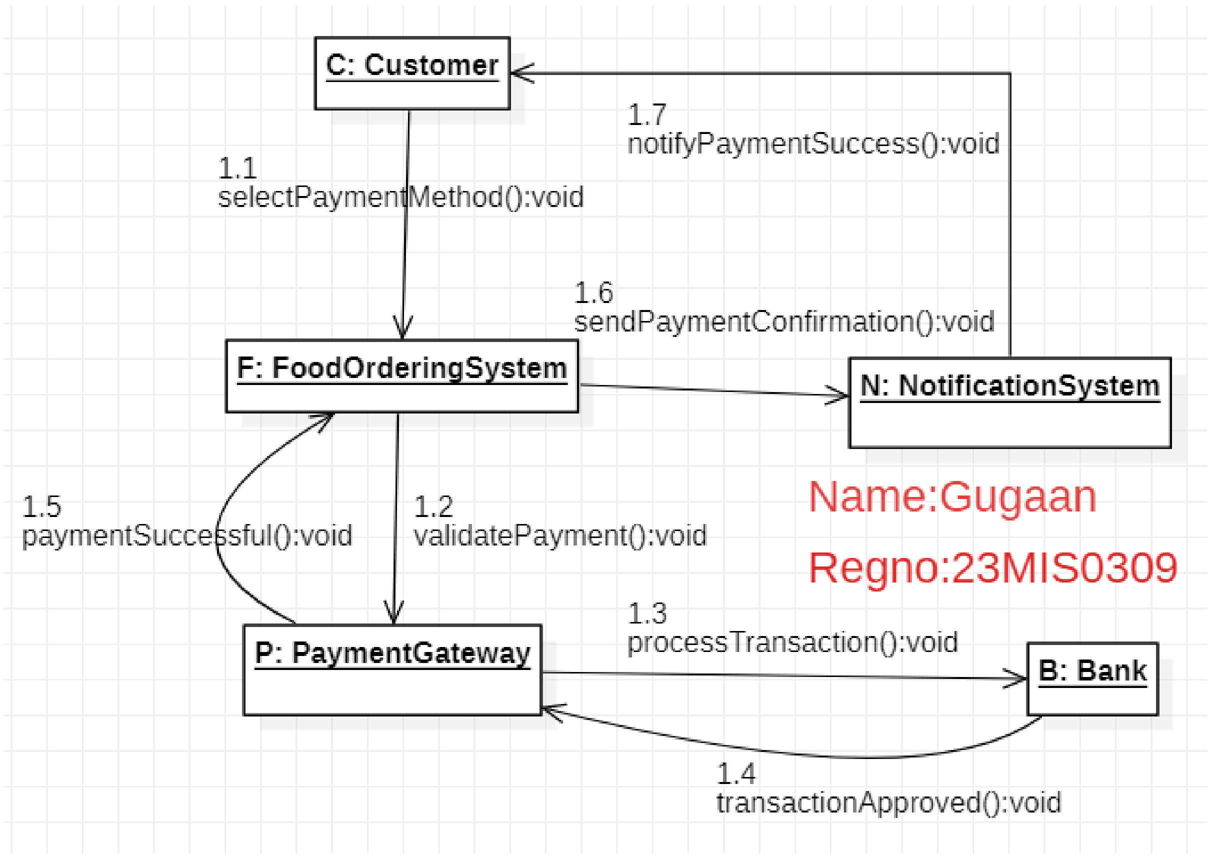
1.Collaboration Diagram for User Registration (fig no.20)



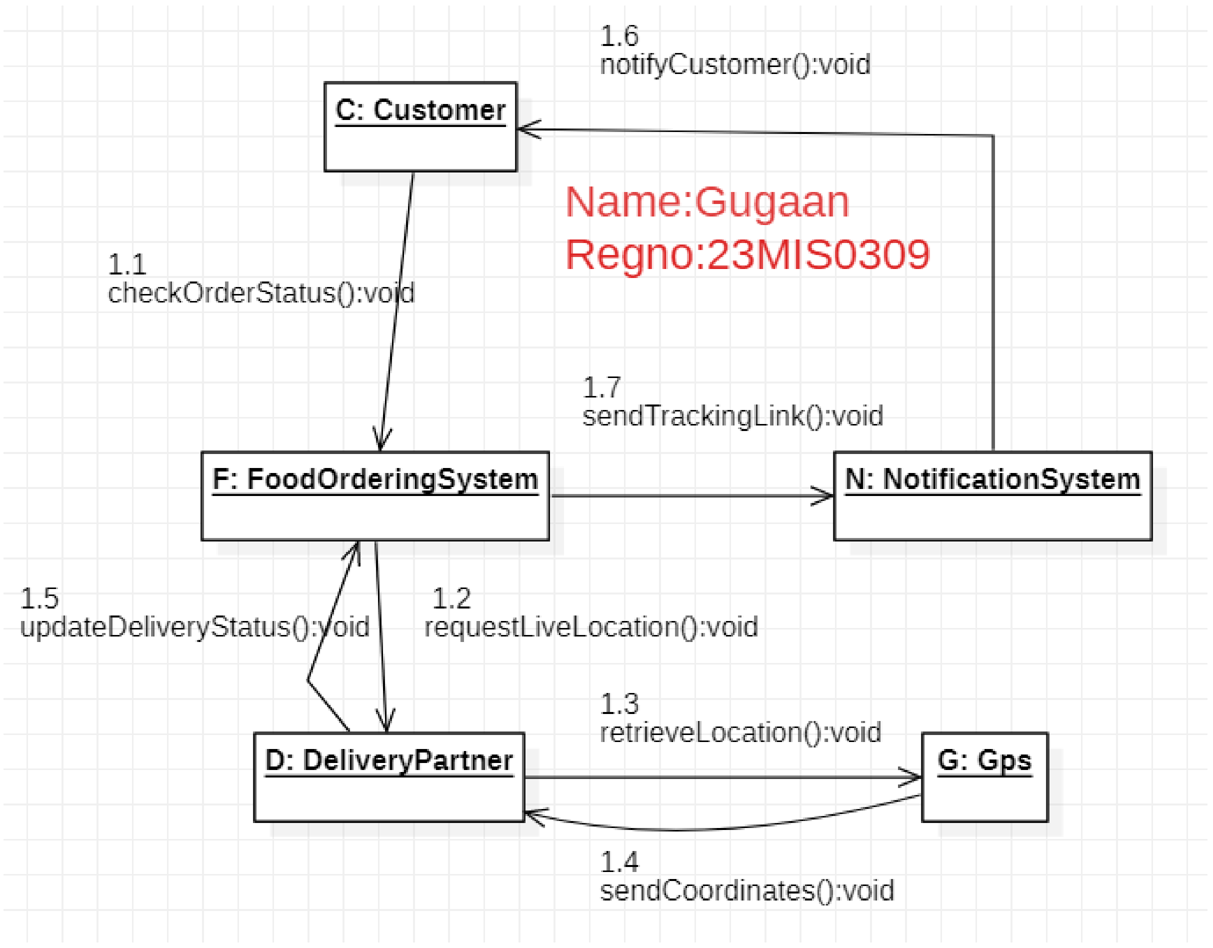
2.Collaboration Diagram for Order Placement (fig no.21)



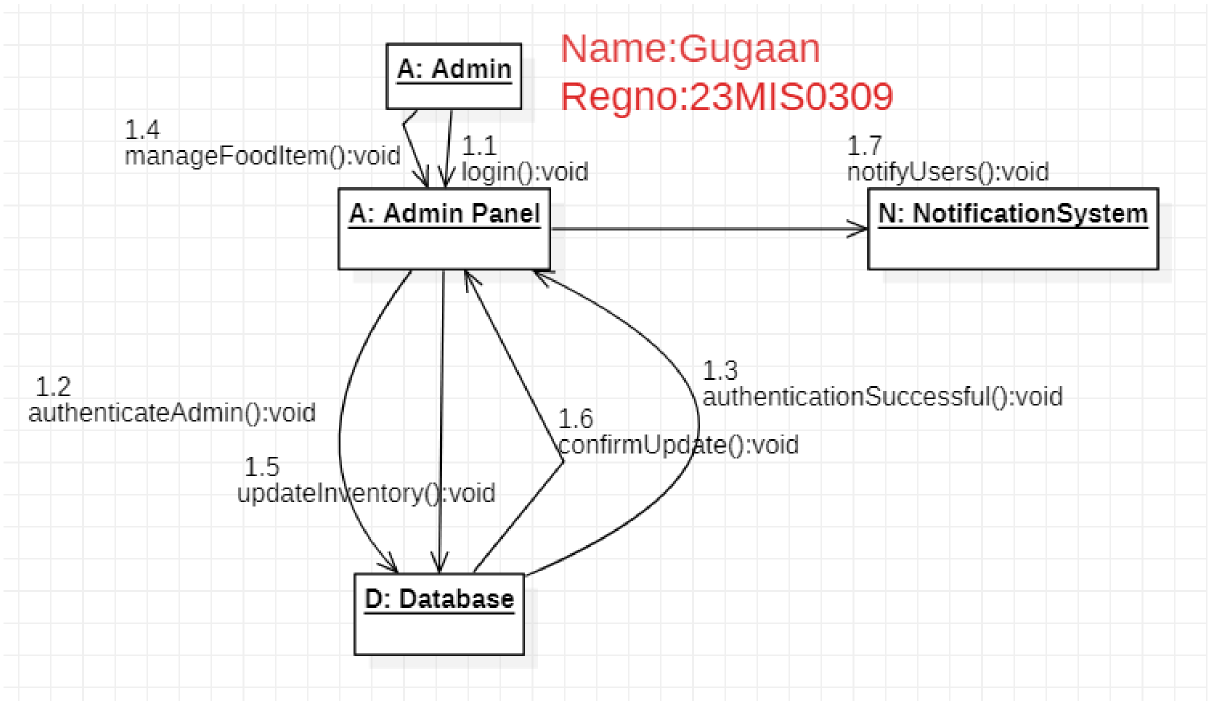
3. Collaboration Diagram for Payment Processing (fig no.22)



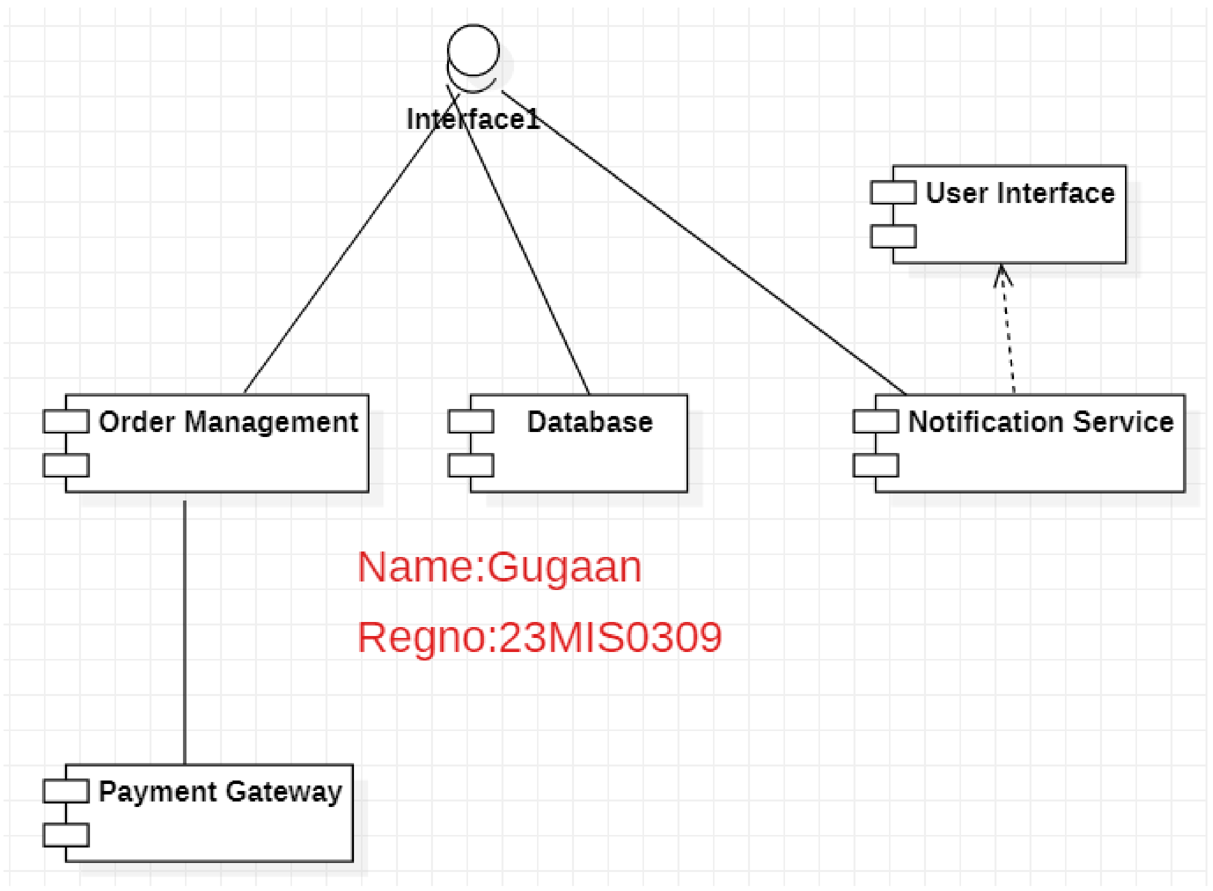
4.Collaboration Diagram for Order Tracking (fig no.23)



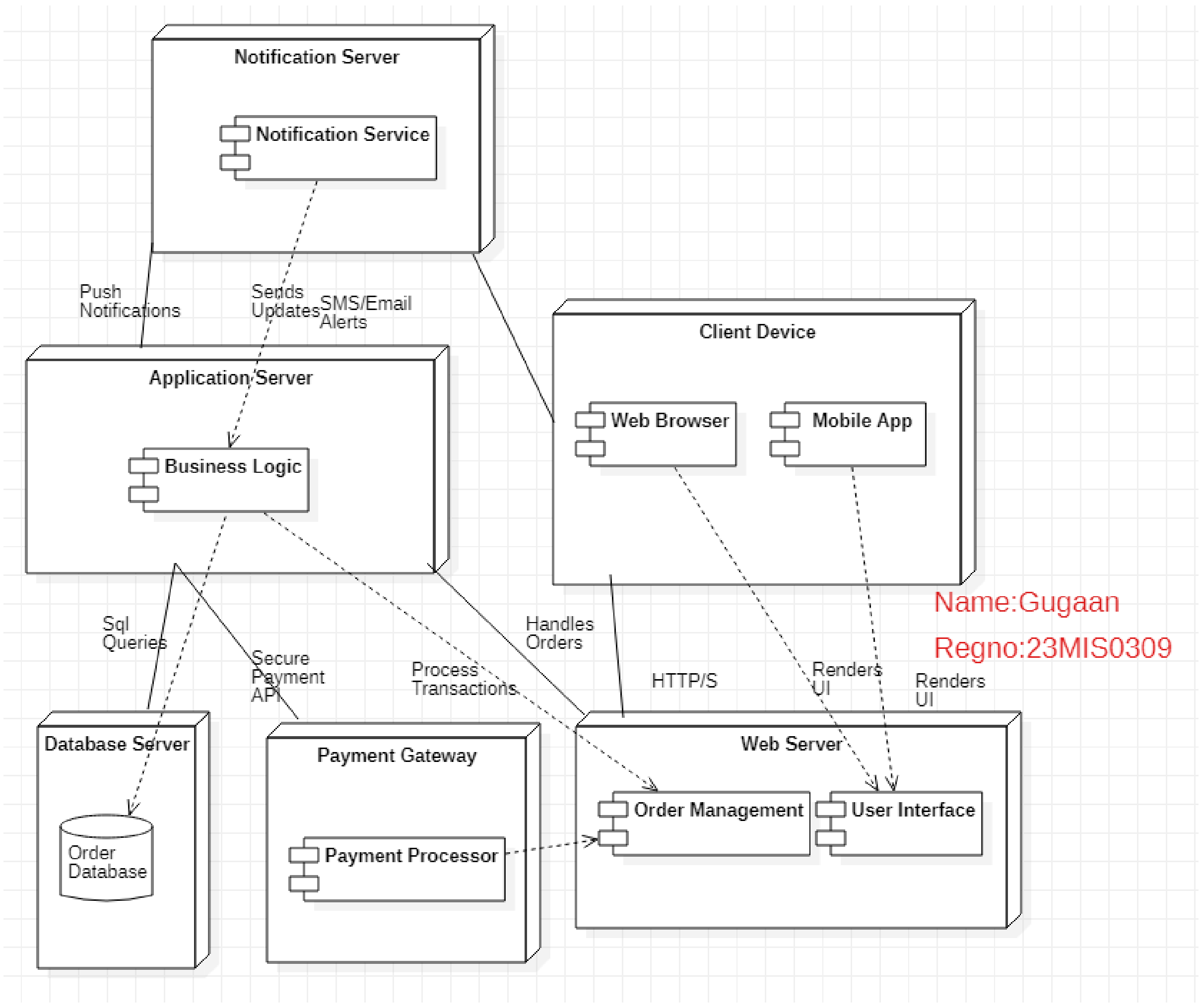
5.Collaboration Diagram for Managing Food Items (Admin) (fig no.24)



4.4 Component Diagram (fig no.25)



4.5 Deployment Diagram (fig no.26)



**Appendix**

**1. Requirement Mapping Matrix (RMM)**

The RMM ensures that all functional and non-functional system requirements are documented and mapped throughout the development lifecycle.

| **Requirement ID** | **Requirement Description** | **Use Case ID** | **Test Case ID** | **Status** |
|---|---|---|---|---|
| REQ001 | User Account Creation | UC01 | TC001 | Implemented |
| REQ002 | Secure Login & Authentication | UC02 | TC002 | Implemented |
| REQ003 | Viewing Available Food Items | UC03 | TC003 | Implemented |
| REQ004 | Adding Items to Shopping Cart | UC04 | TC004 | Implemented |
| REQ005 | Completing an Order Transaction | UC05 | TC005 | Implemented |
| REQ006 | Online Payment Processing | UC06 | TC006 | Implemented |

| Requirement ID | Requirement Description | Use Case ID | Test Case ID | Status |
|---|---|---|---|---|
| REQ007 | Tracking Live Order Status | UC07 | TC007 | Implemented |
| REQ008 | Submitting Customer Reviews | UC08 | TC008 | Implemented |
| REQ009 | Admin Control Over Menu Items | UC09 | TC009 | Implemented |
| REQ010 | Handling Delivery Partner Logs | UC10 | TC010 | Implemented |

**Legend:**

- **Requirement ID:** Unique identifier for each system requirement.
- **Requirement Description:** Summary of the required feature or functionality.
- **Use Case ID:** Corresponding use case as per the use case diagrams.
- **Test Case ID:** Associated test case for validation.
- **Status:** Implemented / In Progress / Pending.

---

## 2. Glossary of Terms

The glossary provides definitions for key concepts and components used in the system.

| Term | Definition |
|---|---|
| **Actor** | A user or external entity that interacts with the system. |
| **Admin** | A privileged user responsible for managing food items and orders. |
| **Shopping Cart** | A temporary storage area where users add selected items. |
| **Order Status** | The real-time state of a placed order (Processing, Out for Delivery, Delivered). |
| **Payment Gateway** | An external system that securely processes online payments. |
| **Delivery System** | Manages real-time tracking of orders from restaurant to customer. |
| **User Session** | A secure, time-bound login instance of a customer or admin. |
| **Inventory Manager** | Keeps track of food stock and availability at restaurants. |
| **Database** | A structured storage unit holding user, order, and restaurant data. |
| **Error Logging** | A mechanism to record system errors for debugging and security. |

---

## 3. Assumptions & Constraints

### Assumptions

1. Users will access the platform via a stable internet connection.
2. Restaurants will regularly update their menu and item availability.
3. Payments will be processed through third-party, PCI DSS-compliant payment gateways.

4. Delivery personnel will use GPS tracking for live order updates.
5. Customers will receive automated notifications at each stage of order processing.

**Constraints**

1. The system must handle peak-hour loads without performance degradation.
2. Transactions must be **secured using end-to-end encryption** to prevent data leaks.
3. The mobile application must support **Android 9+ and iOS 12+** for broad compatibility.
4. The backend database must be **scalable** to accommodate multiple restaurants and users.
5. The system should comply with **GDPR and other regional data privacy laws** for user protection.

---

## 4. Software & Hardware Specifications

### Software Requirements

| **Software Component** | **Description** |
|---|---|
| **Operating System** | Windows 10+, Ubuntu 20+, macOS 11+ |
| **Database System** | PostgreSQL / MySQL for relational data |
| **Programming Languages** | Java (Spring Boot), Python (Django), JavaScript (React.js) |
| **Cloud Deployment** | AWS / Google Cloud / Microsoft Azure |
| **API Framework** | RESTful API using Spring Boot |
| **Mobile App Framework** | Flutter / React Native for cross-platform support |
| **Security Protocols** | SSL/TLS Encryption, OAuth2 Authentication |

### Hardware Requirements

| **Hardware Component** | **Minimum Specification** |
|---|---|
| **Processor** | Intel Core i7 / Ryzen 5 or better |
| **RAM** | 16GB or higher |
| **Storage** | 1TB SSD or equivalent |
| **Network Speed** | 100 Mbps broadband or better |
| **Server Hosting** | Cloud-based (AWS, Google Cloud) |

---

## 5. Risks & Mitigation Strategies

| Risk Factor | Impact Level | Mitigation Strategy |
|---|---|---|
| **Server Downtime** | High | Implement automatic failover & cloud load balancing |
| **Payment Gateway Issues** | High | Use redundant payment gateways with retry logic |
| **Security Breach** | High | Enforce encryption, regular audits & multi-factor authentication (MFA) |
| **High Traffic Load** | Medium | Use caching, database indexing, and horizontal scaling |
| **User Errors** | Low | Implement clear UI, validation checks, and error handling |

## 6. References

- **ISO/IEC 27001** – Information Security Standards for Secure Web Applications.
- **UML 2.5** – Unified Modeling Language Standard for Software Diagrams.
- **OWASP Guidelines** – Best practices for securing web applications.
- **GDPR Compliance** – User Data Protection and Privacy Regulations.
- **IEEE 1471-2000** – Architectural Frameworks for Software System Design.