Restaurant system

content

- Software requirement
- Functional Requirements
- Non Functional Requirements

Functional Requirements

01 Admin

- **System Configuration**

1.Configure system settings, including user roles and permissions

2.Customize menu categories, pricing rules

3.Set up integrations with external systems (POS, accounting software)

- **User Management**

1.Create and manage user accounts, including staff and customers accounts.

2.Assign roles and permissions based on job responsibilities.

- **Data Management**

1.Ensure data security and compliance with privacy regulations .

2.Backup and restore data to prevent loss of critical information.

3.Manage access to sensitive data

02 Customers

*Menu Access and Ordering

- View an up-to-date menu with descriptions, prices, and available options .
- Place orders directly from the menu, including customization options and
- special requests.

*Reservation Management

- Receive confirmation and reminder notifications for reservations.
- Ability to make, modify, or cancel reservations online .

*Payment Options

- Flexible payment options, including cash, credit/debit cards, and digital wallets .
- Secure payment processing with encryption and fraud detection

03 Staff

- **Dashboard Overview**

. Access to overall dashboard providing real-time insights into key

. Ability to view financial reports, including profit and loss statements, inventory costs

- **Menu Management**

. Ability to create, edit, and remove menu items

. Set pricing, descriptions, and categorization of menu items.

. Ability to mark items as specials or promotions.

- **Table Management**

. View and manage table reservations, including assigning, modifying, or cancelling reservat

- **Staff Management**

. Monitor staff performance and track hours worked

- **Reporting and Analytics**

. Analyze trends and patterns to make data-driven decisions for business growth.

Non Functional Requirements

Performance

- Response Time: The system should respond to user interactions (e.g., order placement, menu browsing) within seconds under normal load conditions.
- Throughput: The system should support a minimum of 100 concurrent users during peak hours without degradation I performance.
- Scalability: The system should scale horizontally to accommodate an increase in users and transaction volume as the restaurant grows.

security

- Data Encryption: All sensitive data, including customer information and payment details, should be encrypted during transmission and storage.
- Access Control: Role-based access control (RBAC) should be implemented to restrict access to sensitive functionalities and data based on user roles.
- Authentication and Authorization: Users should be required to authenticate using strong passwords, and multi-factor authentication (MFA) should be available for admin accounts.

Non Functional Requirements

Usability

- User Interface: The user interface should be intuitive and easy to navigate, with clear labeling and consistent design patterns.
- Accessibility: The system should comply with accessibility standards (e.g., WCAG) to ensure that it is usable by people with disabilities.
- Multilingual Support: The system should support multiple languages to cater to diverse user demographics

Reliability

- High Availability: The system should have a minimum uptime of 99.9%, ensuring that it is available to users at all times, except for scheduled maintenance windows.
- Fault Tolerance: The system should be resilient to hardware failures and network

  outages, with automatic failover mechanisms in place to ensure uninterrupted service.

Non Functional Requirements

Scalability

- Elasticity: The system should automatically scale up or down based on demand, provisioning additional resources during peak hours and releasing them during off-peak hours.
- Load Balancing: Load balancers should distribute incoming traffic evenly across multiple servers to prevent overloading and ensure optimal Performance.

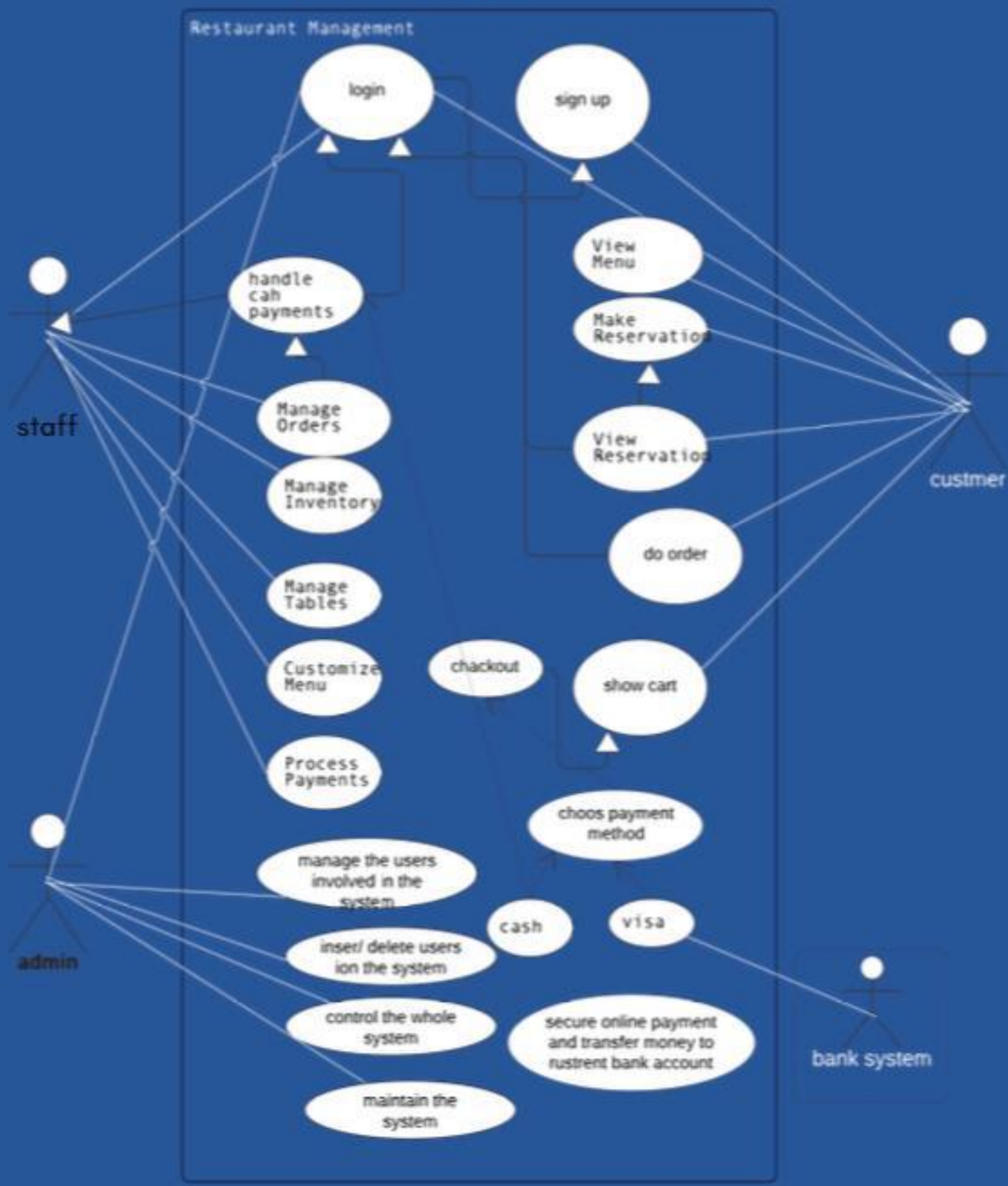
Interoperability

- Integration Capabilities: The system should support integration with third-party systems and services (e.g., POS systems, payment gateways) via standard protocols and APIs.
- Data Exchange Formats: Data should be exchanged in standardized formats (e.g., JSON, XML) to facilitate interoperability with externa systems.

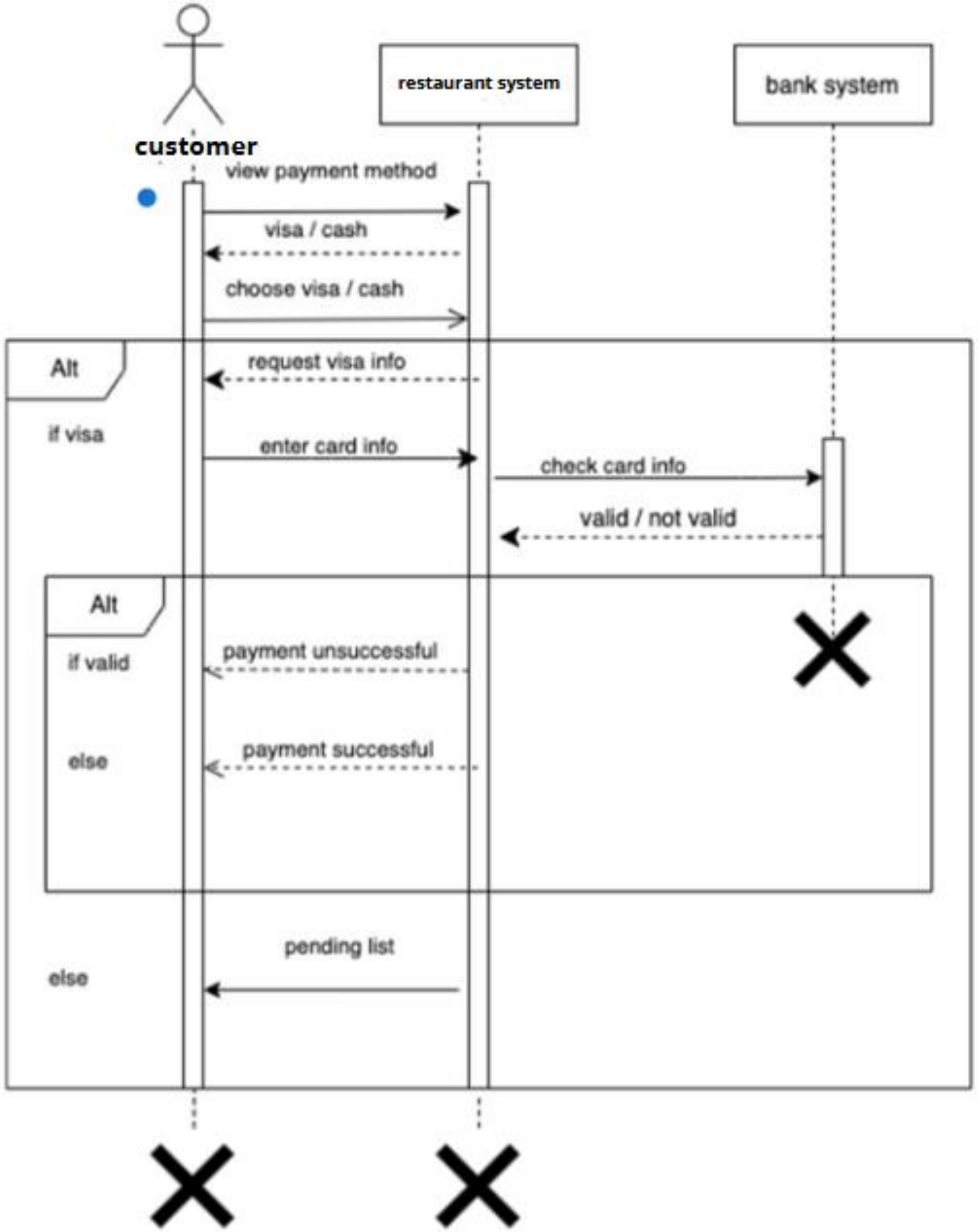
Data Management

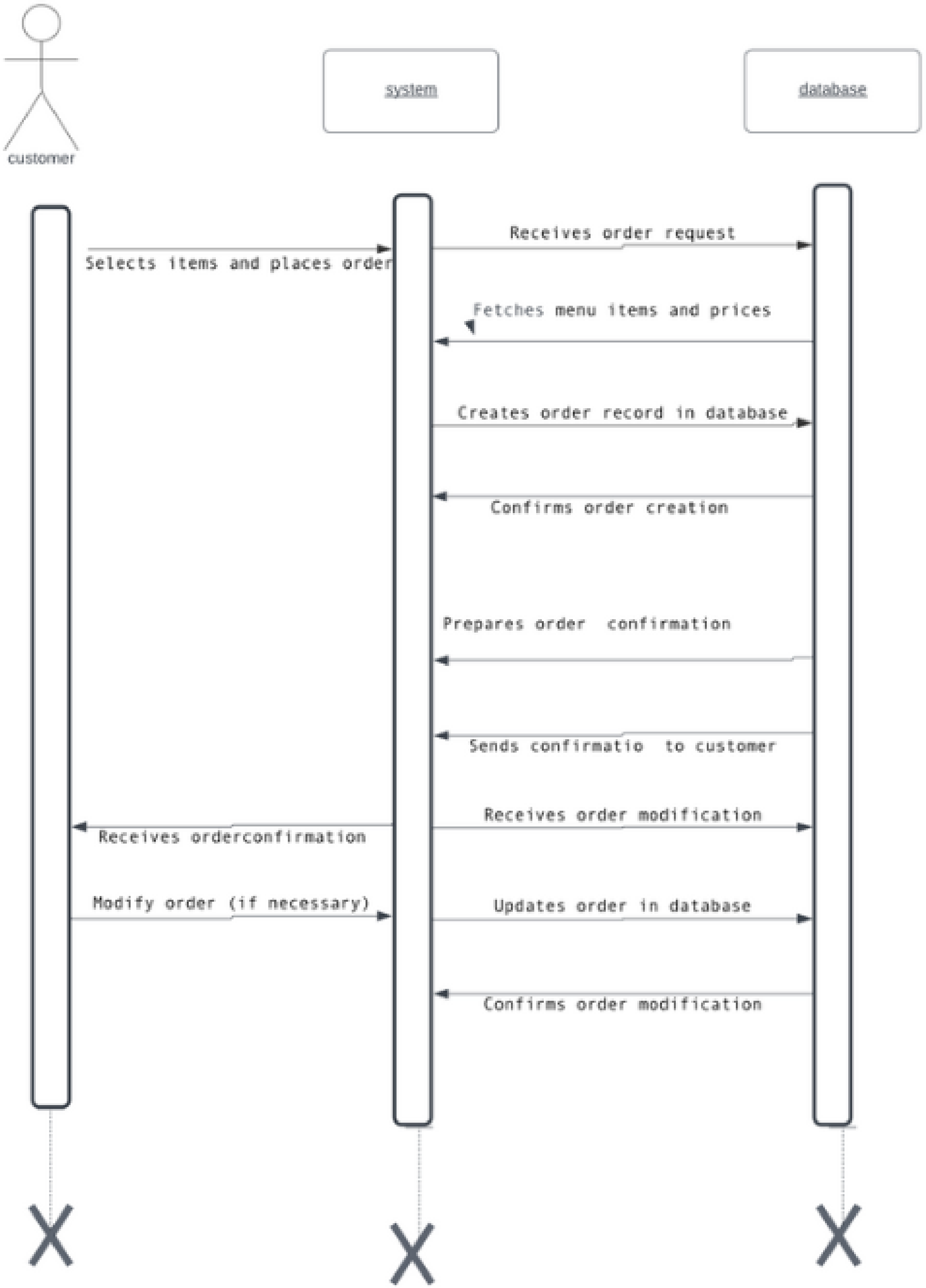
- Data Integrity: The system should maintain data integrity by enforcing referential integrity constraints and implementing error-checking mechanisms.
- Backup and Recovery: Regular backups of the system data should be performed to prevent data loss, with procedures in place for data recovery in the event of a failure.

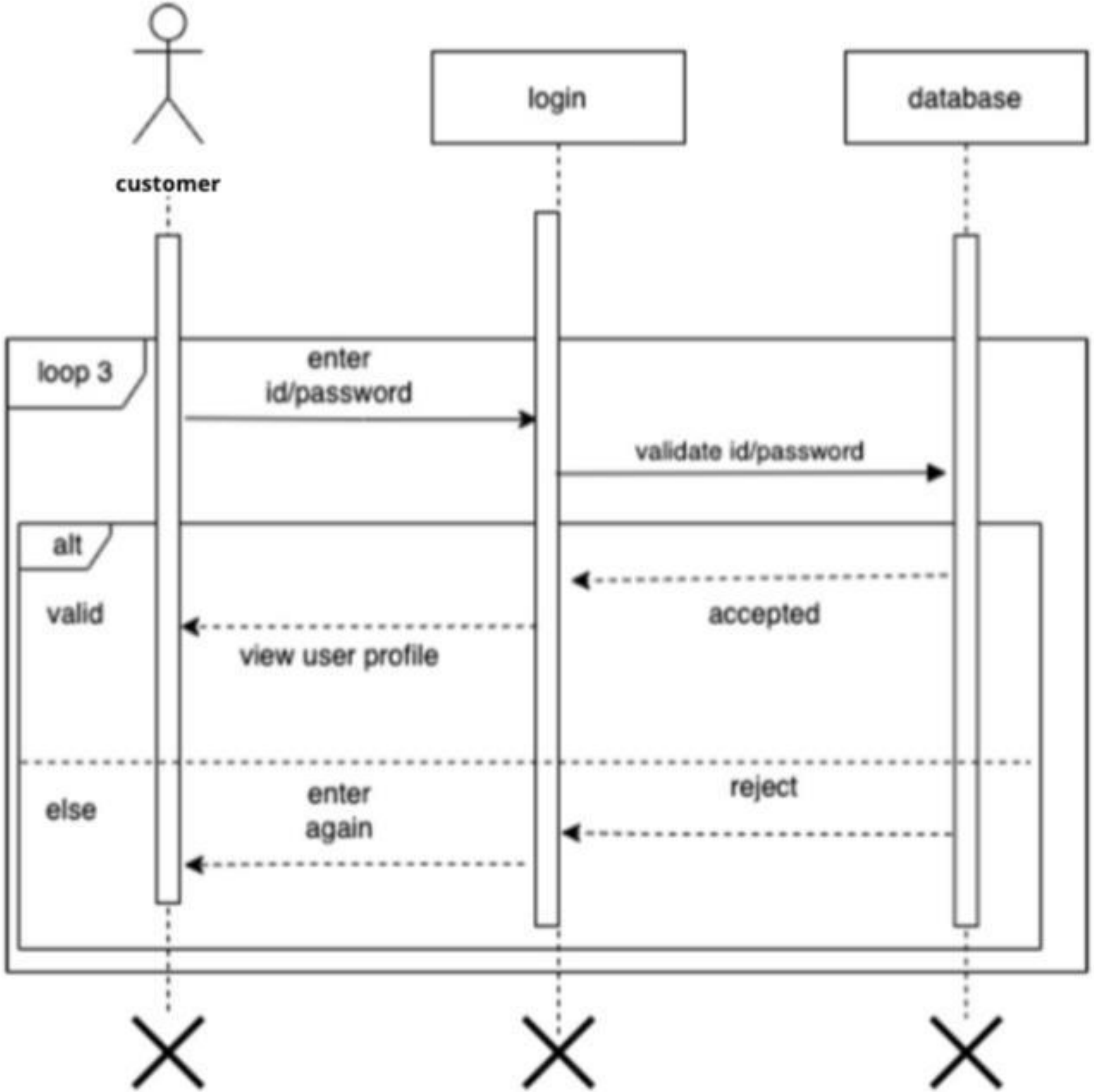
Restaurant System



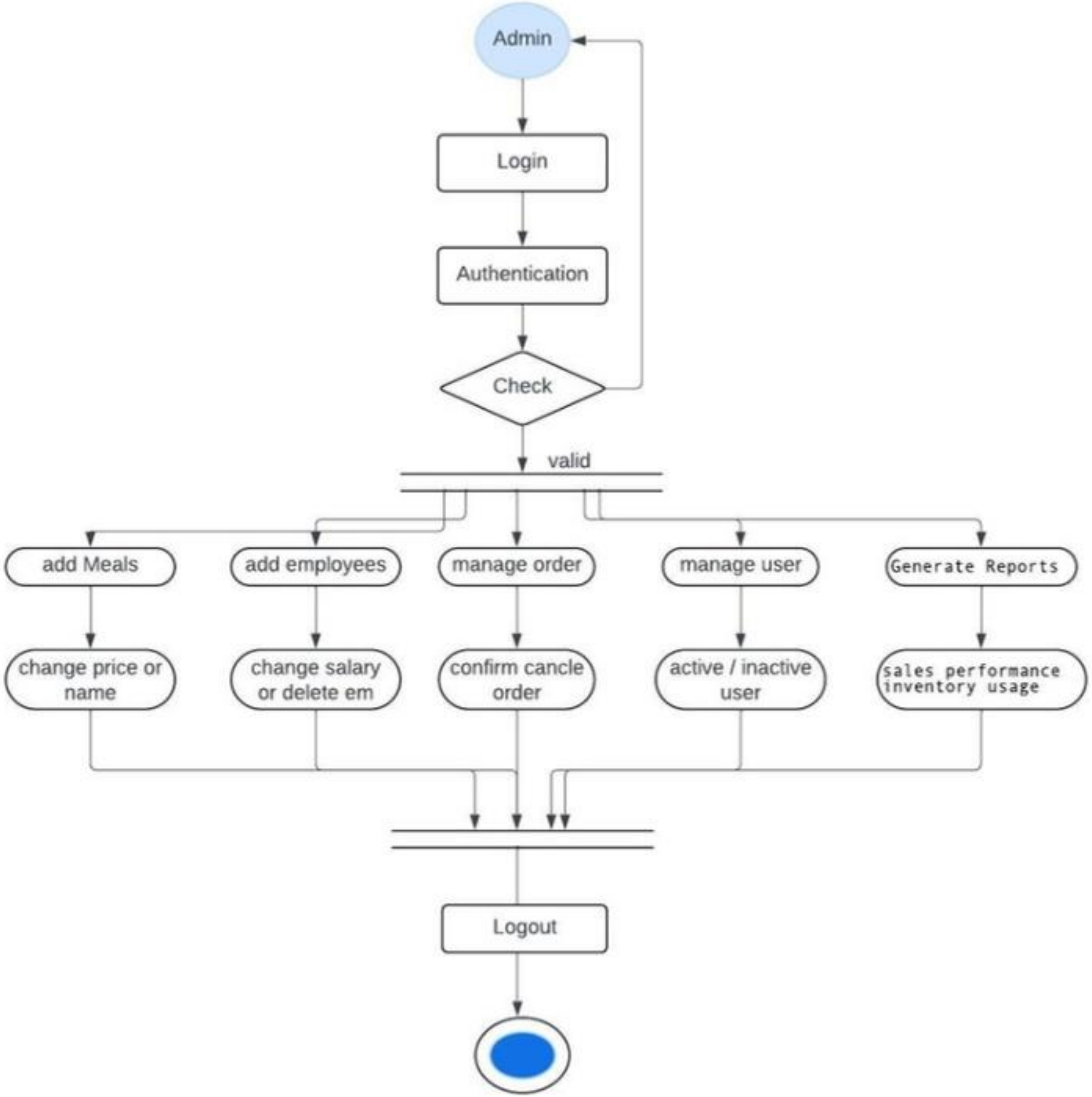
Sequence Diagram

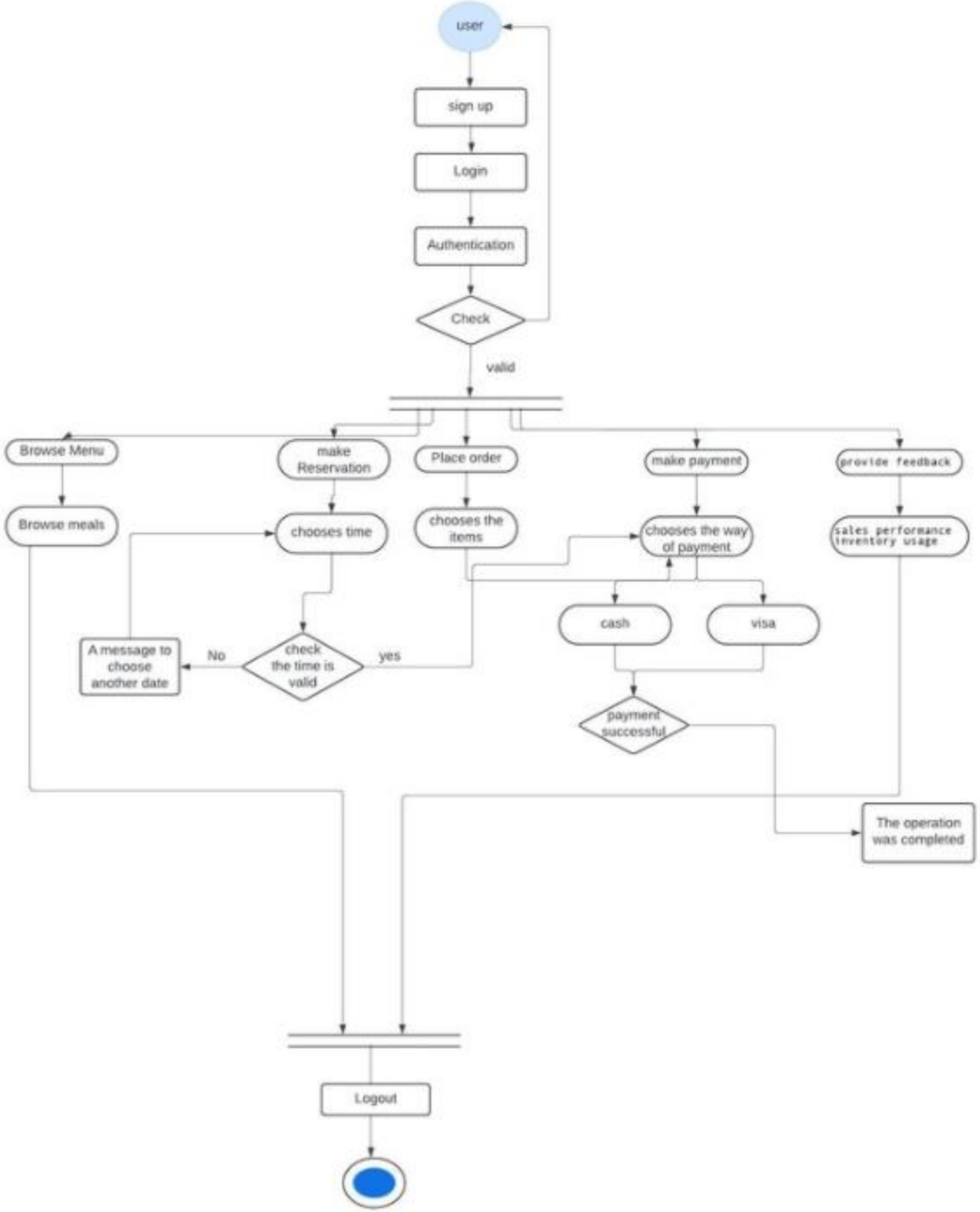

customer
restaurant system
bank system
view payment method
visa / cash
choose visa / cash
Alt
request visa info
if visa
enter card info
check card info
valid / not valid
Alt
if valid
payment unsuccessful
else
payment successful
pending list
else

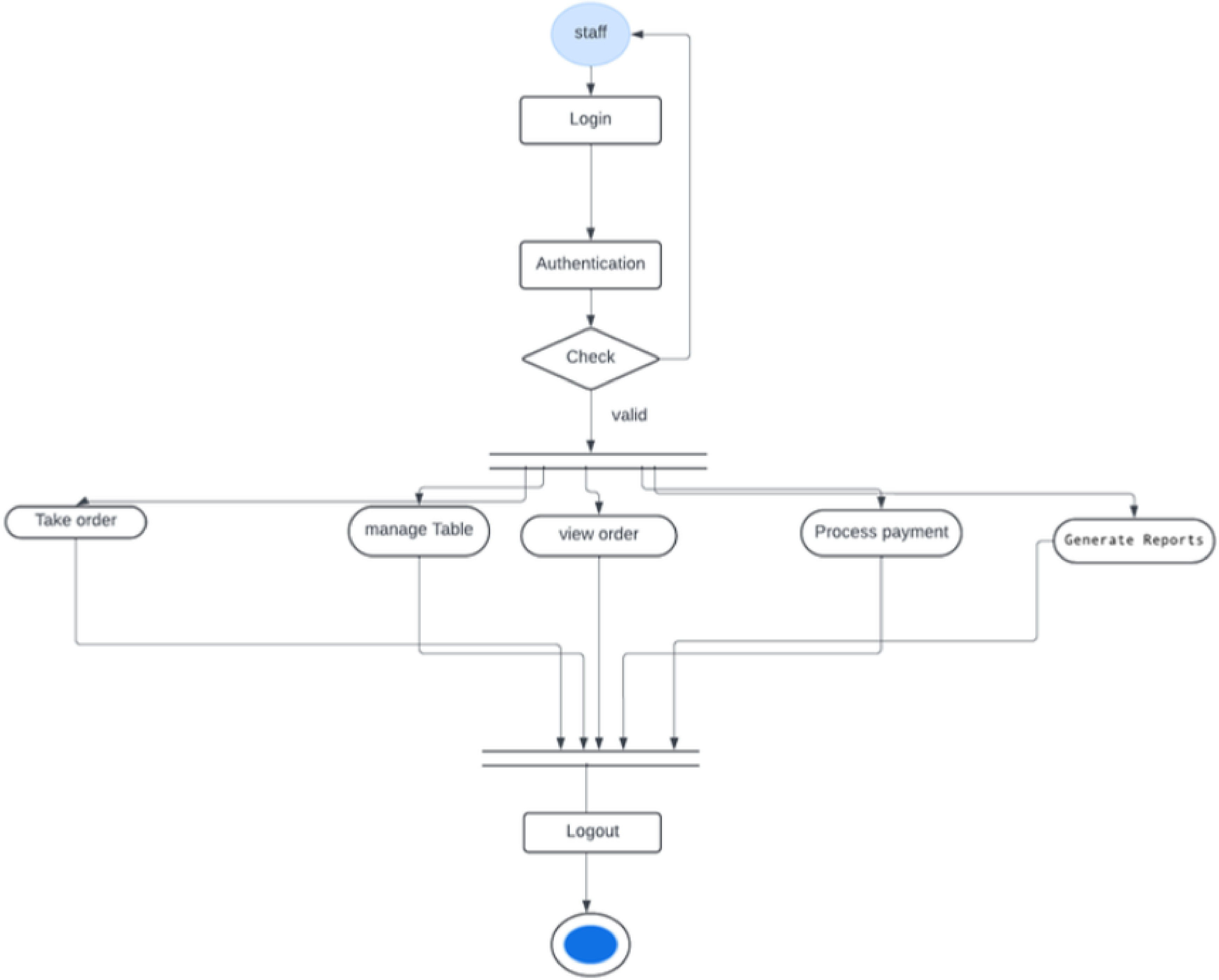

customer
system
database
Selects items and places order
Receives order request
Fetches menu items and prices
Creates order record in database
Confirms order creation
Prepares order confirmation
Sends confirmatio to customer
Receives orderconfirmation
Receives order modification
Modify order (if necessary)
Updates order in database
Confirms order modification


customer
login
database
loop 3
enter id/password
validate id/password
alt
valid
accepted
view user profile
else
reject
enter again

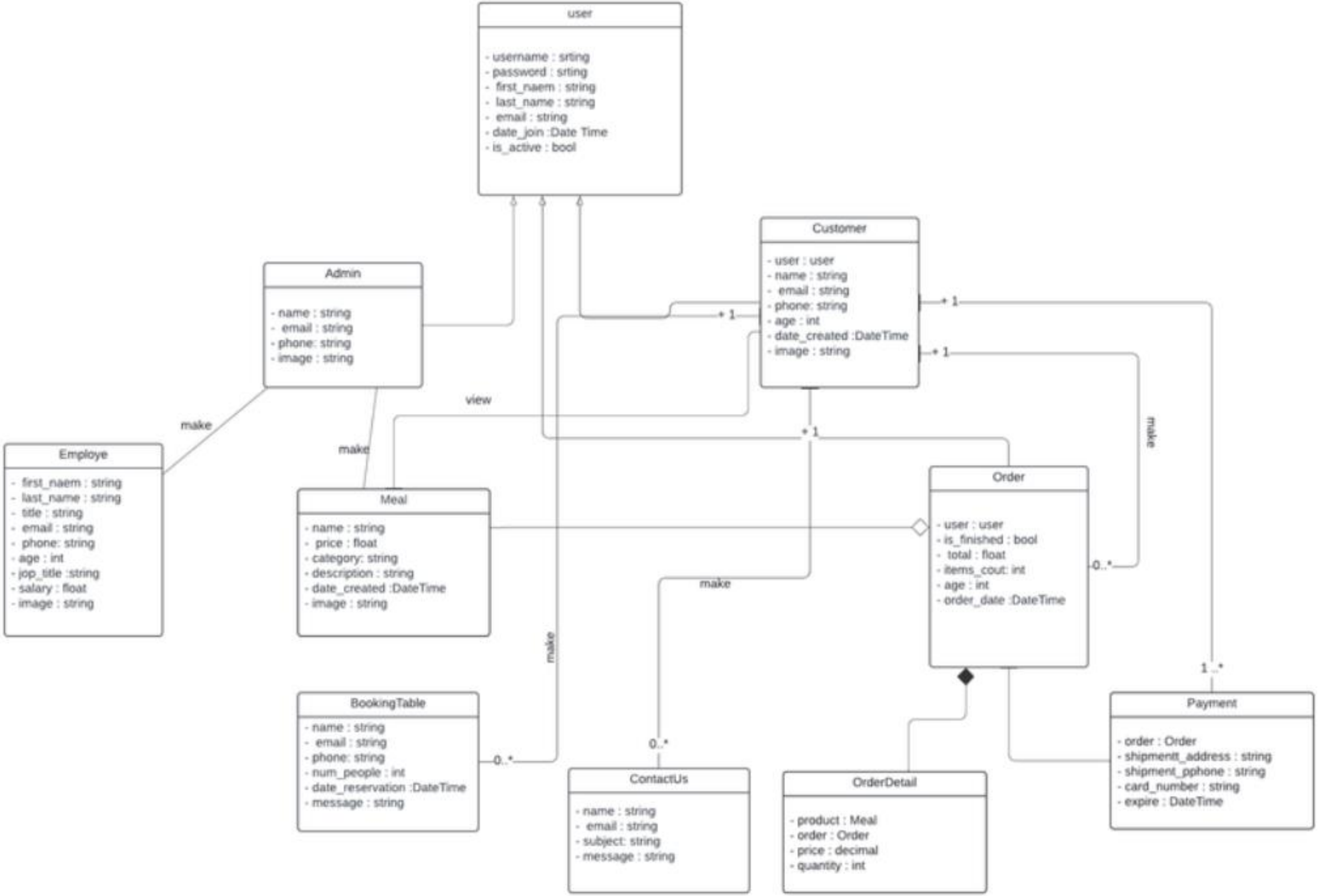
Activity Diagram


Admin
Login
Authentication
Check
valid
add Meals
add employees
manage order
manage user
Generate Reports
change price or name
change salary or delete em
confirm cancle order
active / inactive user
sales performance
inventory usage
Logout


user
sign up
Login
Authentication
Check
valid
Browse Menu
make Reservation
Place order
make payment
provide feedback
Browse meals
chooses time
chooses the items
chooses the way of payment
sales performance
inventory usage
A message to choose another date
No
check the time is valid
yes
cash
visa
payment successful
The operation was completed
Logout


staff
Login
Authentication
Check
valid
Take order
manage Table
view order
Process payment
Generate Reports
Logout

Class Diagram



user

- username : srting
- password : srting
- first_naem : string
- last_name : string
- email : string
- date_join :Date Time
- is_active : bool

Customer

- user : user
- name : string
- email : string
- phone: string
- age : int
- date_created :DateTime
- image : string

Admin

- name : string
- email : string
- phone: string
- image : string

Employe

- first_naem : string
- last_name : string
- title : string
- email : string
- phone: string
- age : int
- jop_title :string
- salary : float
- image : string

Meal

- name : string
- price : float
- category: string
- description : string
- date_created :DateTime
- image : string

Order

- user : user
- is_finished : bool
- total : float
- items_cout: int
- age : int
- order_date :DateTime

BookingTable

- name : string
- email : string
- phone: string
- num_people : int
- date_reservation :DateTime
- message : string

ContactUs

- name : string
- email : string
- subject: string
- message : string

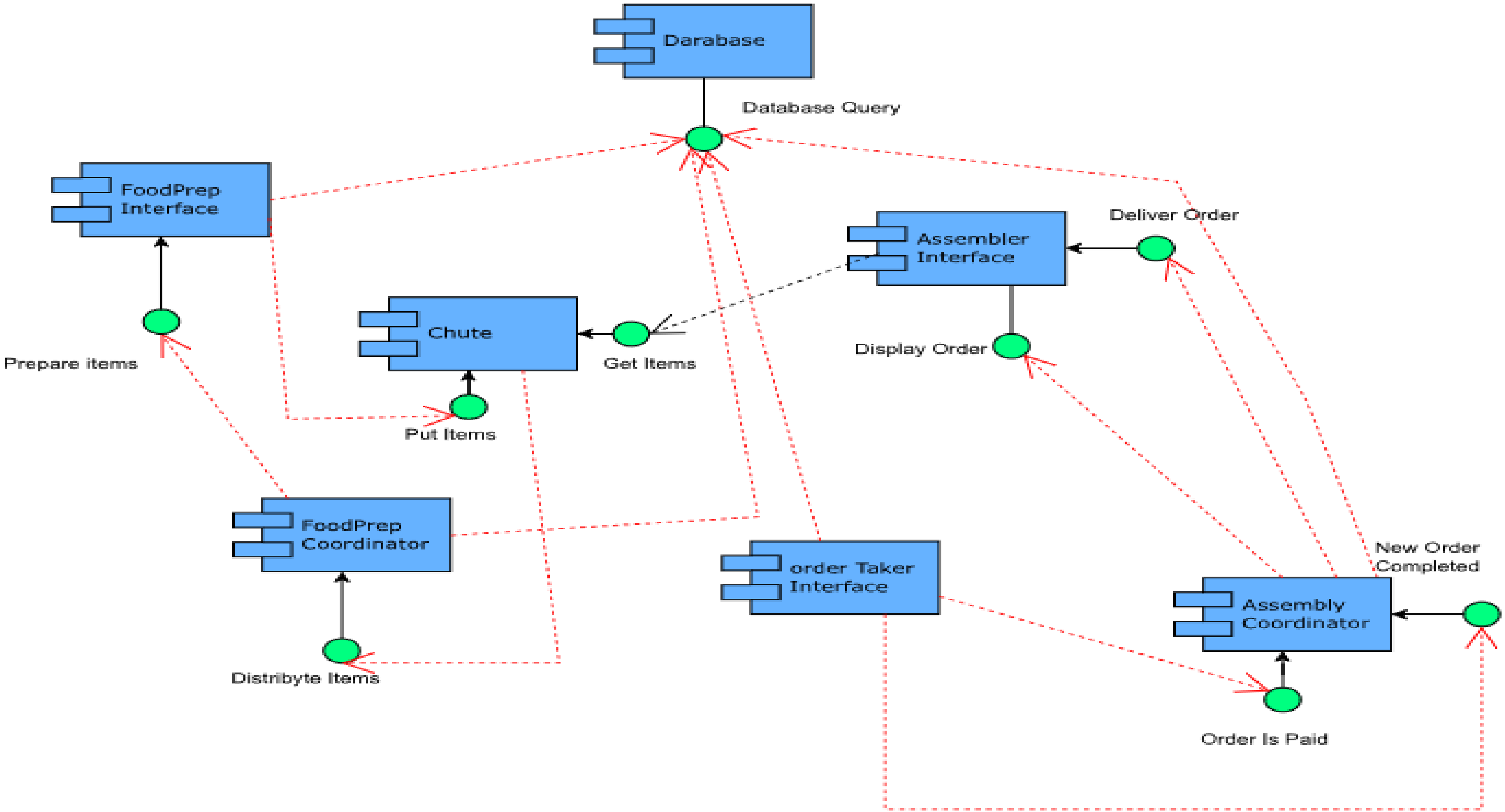
OrderDetail

- product : Meal
- order : Order
- price : decimal
- quantity : int

Payment

- order : Order
- shipmentt_address : string
- shipment_pphone : string
- card_number : string
- expire : DateTime

Component Diagram


Darabase
Database Query
FoodPrep Interface
Assembler Interface
Deliver Order
Chute
Get Items
Display Order
Prepare items
Put Items
FoodPrep Coordinator
order Taker Interface
New Order Completed
Assembly Coordinator
Distribyte Items
Order Is Paid

System Requirements

1. **Software** :

Backend:

- Language : Python (version 3.12.0) .
- Framework : Django (version 4.2.9)

  Django REST Framework (version 3.15.0).
- Database : PostgreSQL .

Frontend :

- HTML, CSS, JavaScript for basic Front-End pages.
- Bootstrap for UI styling.

Hosting and Deployment :

- Railway.
- PythonAnywhere.

Authentication :

- OAuth2 (To provide login functionality via platforms like Google or Facebook.)

Version Control :

- Git for version control.

2. **Server Hardware Requirements**:

- **CPU** : - Minimum: Intel Core i3 or AMD Ryzen 3.

  -Recommended: Intel Core i5 or AMD Ryzen 5.

- **RAM** : - Minimum: 8 GB. - Recommended: 16 GB.

- **Storage** : - Minimum: 256 GB SSD.

  - Recommended: 512 GB SSD or higher.

3.Server Software Requirements :

- **Operating System** : Windows, macOS.
- **Web Server** : IIS (Internet Information Services) for hosting the API.
- **Database** : PostgreSQL.
- **Python Environment** : - Python: (version 3.12.0) to run Django and other Python dependencies.

  - Virtual Environment: For isolating project dependencies from the system environment
- **Additional Requirements**:

  Firewall and Security: Configured for secure access (SSH, HTTPS).

  SSL/TLS: For secure communication, especially if dealing with sensitive data.