RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CB23332
SOFTWARE ENGINEERING LAB

Laboratory Record Note Book

Name : . . . .

Year / Branch / Section : . . . .

Register No. : . . . .

Semester : . . . .

Academic Year : . . . .

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)
RAJALAKSHMI NAGAR, THANDALAM – 602-105

BONAFIDE CERTIFICATE

NAME:_______________________**REGISTER NO.:** _______________

ACADEMIC YEAR: 2024-25 **SEMESTER:** III **BRANCH:**_________B.E/B.Tech

This Certification is the bonafide record of work done by the above student in the

CB23332-SOFTWARE ENGINEERING - Laboratory during the year 2024 – 2025.

Signature of Faculty -in – Charge

Submitted for the Practical Examination held on ____________

Internal Examiner External Examiner

INDEX

EX NO:1	**WRITE THE COMPLETE PROBLEM STATEMENT**
DATE:	

AIM:

To prepare PROBLEM STATEMENT for Smart Grocery List Manager

ALGORITHM:

1. Initialize System and Load Data:
   - Set up user accounts and load initial pantry items (if available) into the database.
2. User Login:
   - Allow users to log in using their credentials to access shared or individual grocery lists.
3. Grocery Item Search and Add to List:
   - Allow users to search for items by category, product name, or store availability.
   - Display similar items, price comparisons, and available discounts (if connected to stores).
   - Users can add items to a new or existing list, choosing quantities and preferred brands.
4. Real-Time List Syncing:
   - Automatically sync the list across all family members' devices.
   - Send notifications for items that have been added or removed.
5. Pantry and Inventory Management:
   - Track pantry inventory based on purchases and set minimum stock levels for essential items.
   - Enable manual or barcode-based inventory updates.
6. Expiration Date Tracking:
   - Allow users to log expiration dates for perishable items.
   - Send reminders for items nearing expiration to encourage timely use or donation.
7. Budget Tracking and Suggestions:
   - Record prices of regularly purchased items and calculate weekly or monthly grocery budgets.
   - Provide budget-friendly recommendations based on frequently bought items.
8. Shopping List Optimization:
   - Suggest ideal shopping sequences based on store layout (if available) and current inventory.
   - Flag duplicate items already in the pantry or added by other family members.
9. Reminders and Notifications:
   - Send reminders for list items, restocking essentials, and deals or promotions relevant to

- Send reminders for list items, restocking essentials, and deals or promotions relevant to frequently bought items.

10. Reporting and Insights:
    - Generate reports on monthly spending, food wastage (based on expired items), and shopping frequency.
    - Provide insights into popular items and budget allocation by category.

INPUT:

1. User Information:
   - Username, password for login.
   - Contact details for notifications.
2. Grocery Item Details:
   - Item name, category, quantity, brand preference, and price.
   - Expiration dates for perishables.
3. Inventory and Pantry Information:
   - Current pantry stock, minimum stock levels, and item usage frequency.
   - Expiration dates for items in stock.
4. Notifications and Reminders:
   - Preferences for restock reminders, expiration alerts, and budget updates.
   - Inputs for automated notifications on price drops and promotions.

Problem:

Current grocery shopping processes lack efficiency and real-time collaboration features, making it challenging for households to manage grocery lists, prevent food wastage, and maintain a budget. Disorganized lists, overlooked expiration dates, and duplicate purchases contribute to inefficiencies, food waste, and budget overruns.

Background:

In busy households, coordinating grocery shopping and managing household essentials can become burdensome, leading to miscommunication, forgotten items, and unnecessary expenses. Families often need an organized way to collaborate on grocery lists and keep track of inventory and item expiration dates.

Relevance:

An efficient grocery list manager is critical for household budgeting, meal planning, and minimizing waste. A Smart Grocery List Manager (SGLM) would improve convenience by syncing lists across family members, providing real-time updates, and tracking pantry items and expiration dates.

1. **Objectives:**

   Analyzing Current Grocery Management Practices:
   - Conducting a detailed assessment of existing grocery management methods (e.g., handwritten lists, simple apps) to identify gaps and inefficiencies.
2. **Automating List Management**:
   - Implementing a system that automatically generates and updates grocery lists based on user preferences, current pantry stock, and shopping habits to reduce human intervention and errors.
3. **Real-Time List Syncing**:
   - Ensuring that grocery lists are automatically synced in real-time across all family members' devices, allowing for seamless collaboration during shopping.
4. **Tracking Pantry Inventory**:
   - Introducing features that automatically track pantry items based on purchases, set minimum stock levels for essential items, and enable manual updates using barcodes or inventory scanning.
5. **Expiration Date Tracking**:
   - Allowing users to log expiration dates for perishable items and send timely notifications for items nearing expiration to encourage better usage or donation.
6. **Budget Tracking and Suggestions**:
   - Recording prices for regularly purchased items and calculating weekly or monthly grocery budgets. Provide budget-friendly suggestions based on frequently bought items and user spending patterns.
7. **Shopping List Optimization**:
   - Suggesting the most efficient shopping sequences based on store layout (if available) and current inventory, while flagging duplicate items already in the pantry or purchased by other family members.
8. **Providing Reminders and Notifications**:
   - Sending reminders for list items, restocking essentials, and notifying users about deals, discounts, or promotions relevant to frequently purchased items.
9. **Reporting and Insights**:
   - Generating reports on monthly spending, food wastage (based on expired items), and shopping frequency. Providing insights into popular items, budget allocation by category, and opportunities to save.
10. **Ensuring User-Friendly Interface and Scalability**:

- Designing a user-friendly interface that adapts to individual needs, whether for a single user or a large family. Ensuring the system is scalable to accommodate growing user demands and integrate with third-party services like grocery delivery systems.

Result:

EX NO:2	WRITE THE SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT
DATE:	

AIM:

To do requirement analysis and develop Software Requirement Specification Sheet(SRS) for Smart Grocery List Manager

ALGORITHM:

SRS shall address are the following:

a) Functionality

The SGLM will manage grocery lists, track pantry inventory, and send expiration reminders. It will also offer budget tracking, shopping list optimization, and real-time syncing across family members. Notifications will keep users informed of inventory levels and upcoming expirations.

b) External Interfaces

The software will interface with mobile devices (iOS, Android) and web platforms. It will sync data in real-time across multiple devices and connect with third-party services for price comparison and promotions. The app will also interface with barcode scanners for inventory updates.

c) Performance

The system must respond quickly, with list updates and notifications within 1-2 seconds. The backend should ensure 99.9% uptime and handle recovery within 5 minutes after failure. Response times for data access and synchronization should be optimized for performance.

d) Attributes

The SGLM must be portable across iOS, Android, and web platforms. It should be secure with encrypted data storage and proper user authentication. The system needs to be maintainable with modular code for easy updates and modifications.

e) Design Constraints

The app will be built using React Native or platform-specific languages, and the backend with Node.js or Django. It will use a cloud-based database with ACID properties for data integrity. The system must be scalable, handling growing data and user demands effectively.

1. Introduction

1.1 Purpose

This document defines the requirements for the Smart Grocery List Manager (SGLM), which aims to automate grocery list management, track pantry inventory, and optimize budgeting. The system will enhance shopping efficiency, reduce waste, and provide real-time synchronization across family members..

1.2 Document Conventions

This document uses the following conventions:

 DB - Database

 UI - User Interface

 API - Application Programming Interface

 SGLM - Smart Grocery List Manager

 OCR - Optical Character Recognition (for barcode scanning)

1.3. Intended Audience and Reading Suggestions

This document is intended for:

- **Developers:** To understand the technical requirements for system implementation**.**
- **Project Managers:** To oversee project timelines and deliverables.
- **Stakeholders:** To ensure the system meets operational needs and user expectations**.**
- **Quality Assurance:** To validate and verify the functionality and performance of the system**.**.

1.4. Project Scope

The Smart Grocery List Manager will allow users to create grocery lists, track pantry inventory, send expiration reminders, and track budgets. It will enhance collaboration among family members, help reduce food waste, and provide valuable insights into spending patterns.

1.5. References

 Grocery shopping guidelines.

 Data privacy and cloud storage policies.

 User interface design best practices.**2. Overall Description**

2.1 Product Perspective

The system is a mobile and web-based application that syncs grocery lists and pantry inventories in real time, tracks expiration dates, and provides budget insights.

2.2 Product Features

The main features of the Smart Grocery List Manager include:

- Real-time list syncing across family members' devices.
- Pantry inventory tracking with alerts for low stock and expiration dates.
- Budget tracking and cost-effective suggestions.
- Notifications for restocking, deals, and expiration reminders.

2.3 User Class and Characteristics

- **Users**: Families and individual shoppers can create and manage grocery lists.
- **Admin**: Manages user accounts, updates, and system settings.
- **Guest**: Can view shared lists but cannot make changes..

2.4 Operating Environment

The system operates in the following environments:

- Mobile (iOS, Android) and web platforms (modern browsers like Chrome, Safari).
- Cloud-based server architecture for database and user data storage.

2.5 Design and Implementation Constraints

- **Database**: MySQL or PostgreSQL for storing user data, inventory, and transactions.
- **Mobile Platform**: React Native or Flutter for cross-platform compatibility.
- **Web Platform**: React or Vue.js for frontend development.

2.6. Assumptions and Dependencies

The following assumptions and dependencies are considered in the design and implementation of the system:

- Users will have access to smartphones with internet connectivity.
- Cloud servers and third-party payment gateways will be available and reliable.

3. Specific Requirements

Description and Priority

The system prioritizes real-time syncing, inventory tracking, and budget management. Notifications and reminders for expiration dates are essential to reducing food wastage and ensuring an efficient grocery shopping experience.

Stimulus/Response Sequence

□ **A user adds an item**: The system updates the grocery list and pantry inventory.

□ **A user checks the list**: The system fetches and displays the most recent data, syncing across devices.

□ **A user receives an expiration alert**: The system sends a reminder notification for perishable items.

Functional Requirements

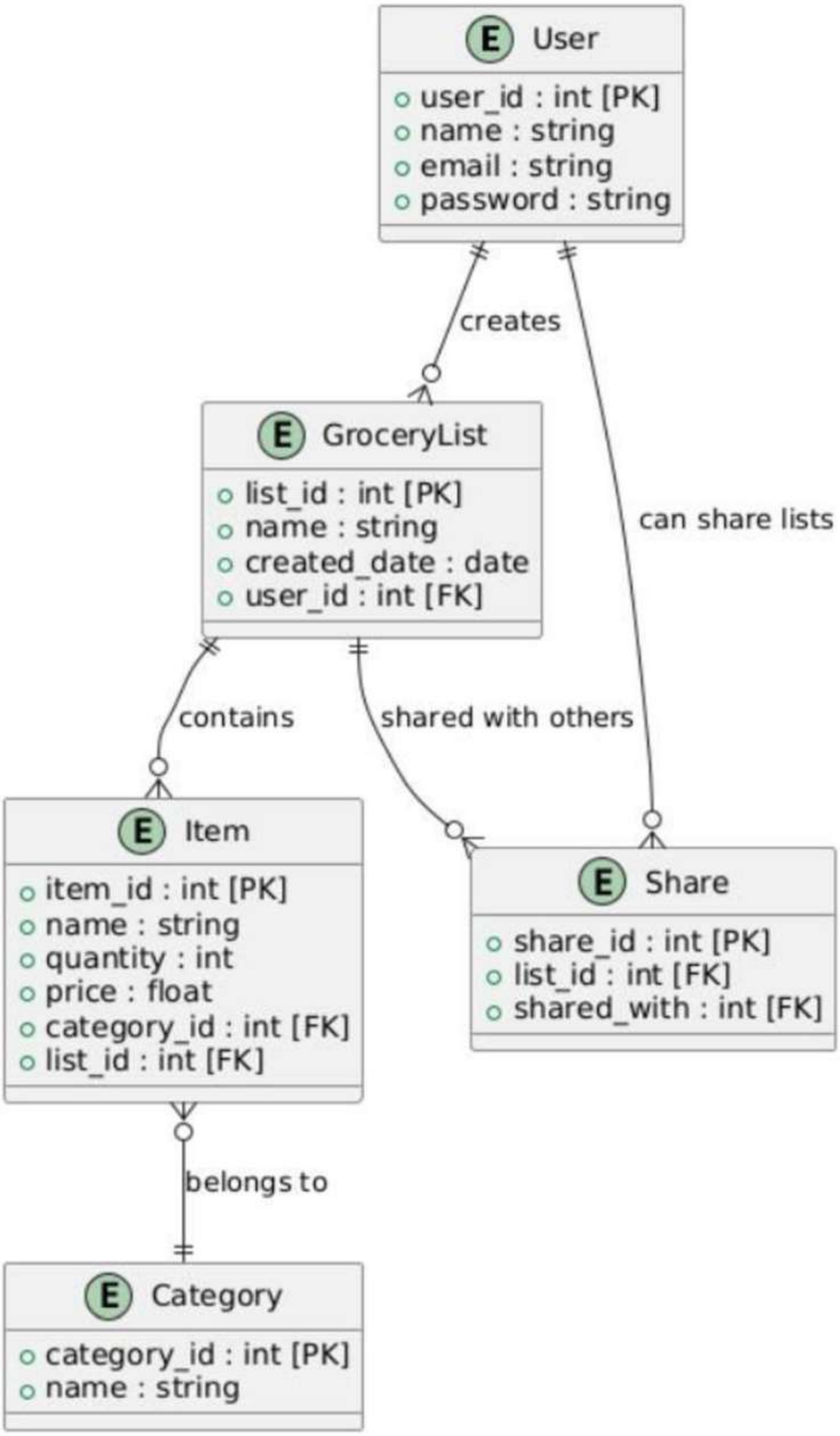
- **Real-Time Syncing**: Automatically updates lists across family members' devices.
- **Inventory Management**: Tracks pantry items and sends low-stock alerts.
- **Budget Tracking**: Calculates and tracks weekly or monthly grocery budgets.

4. External Interface Requirements

4.1 User Interfaces

□ **Front-End**: Web interface developed using React or Vue.js, mobile interface developed using React Native or Flutter.

□ **Back-End**: The backend will use Node.js or Django for API and server-side logic.


E User
user_id : int [PK]
name : string
email : string
password : string
creates
can share lists
E GroceryList
list_id : int [PK]
name : string
created_date : date
user_id : int [FK]
contains
shared with others
E Item
item_id : int [PK]
name : string
quantity : int
price : float
category_id : int [FK]
list_id : int [FK]
E Share
share_id : int [PK]
list_id : int [FK]
shared_with : int [FK]
belongs to
E Category
category_id : int [PK]
name : string

4.2 Hardware Interfaces

- □ **Mobile Devices**: iOS and Android smartphones.
- □ **Barcodes**: Support for barcode scanning via smartphone cameras to update inventory..

4.3 Software Interfaces

□ **Operating System**: iOS, Android, and modern browsers (Chrome, Safari, Firefox).
□ **Database**: MySQL or PostgreSQL for data storage.
□ **Payment Gateway**: Integration with third-party payment services for budget tracking.

4.4 Communication Interfaces

- **Web and Mobile Apps**: The system will communicate with users through a web and mobile interface for managing lists, inventory, and payments..

5. Additional Requirements

5.1 Performance Requirements

- **Response Time**: The system should process updates and sync changes within 1-2 seconds.
- **Availability**: The system should have 99.9% uptime, ensuring availability for users..

5.2 Safety Requirements

Data Recovery: In case of failure, the system will have regular backups to restore data from the last successful sync.

5.3 Security Requirements

□ **Authentication**: Users must log in securely using passwords, with optional multi-factor authentication.

□ **Data Encryption**: All sensitive user data should be encrypted in the database and during transmission.

5.4 Software Quality Attributes

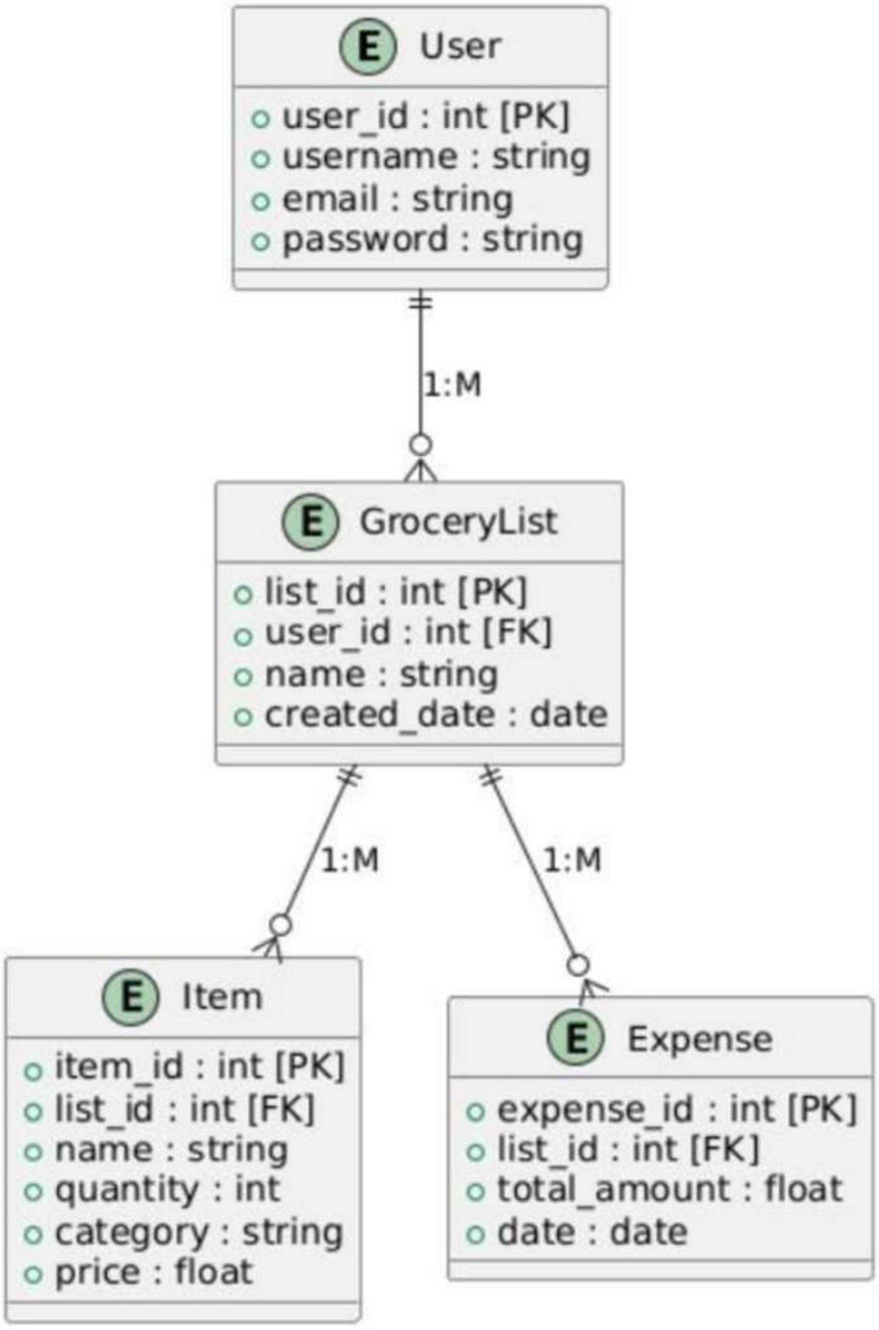
□ **Availability**: The application should be available 24/7 with minimal downtime for maintenance.

□ **Correctness**: Data on the list, inventory, and budget should always be accurate.

□ **Maintainability**: The system should be easy to update with bug fixes and new features.

□ **Usability**: The user interface should be intuitive and easy for all users to navigate.

Result:


E User
o user_id : int [PK]
o username : string
o email : string
o password : string
1:M
E GroceryList
o list_id : int [PK]
o user_id : int [FK]
o name : string
o created_date : date
1:M
1:M
E Item
o item_id : int [PK]
o list_id : int [FK]
o name : string
o quantity : int
o category : string
o price : float
E Expense
o expense_id : int [PK]
o list_id : int [FK]
o total_amount : float
o date : date

EX NO:3	**DRAW THE ENTITY RELATIONSHIP DIAGRAM**
DATE:	

AIM:

To Draw the Entity Relationship Diagram for Smart Grocery List Manager

ALGORITHM:

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types

Step 4: Mapping of Binary 1:N Relationship Types.

Step 5: Mapping of Binary M:N Relationship Types.

Step 6: Mapping of Multivalued attributes.

INPUT:

Entities

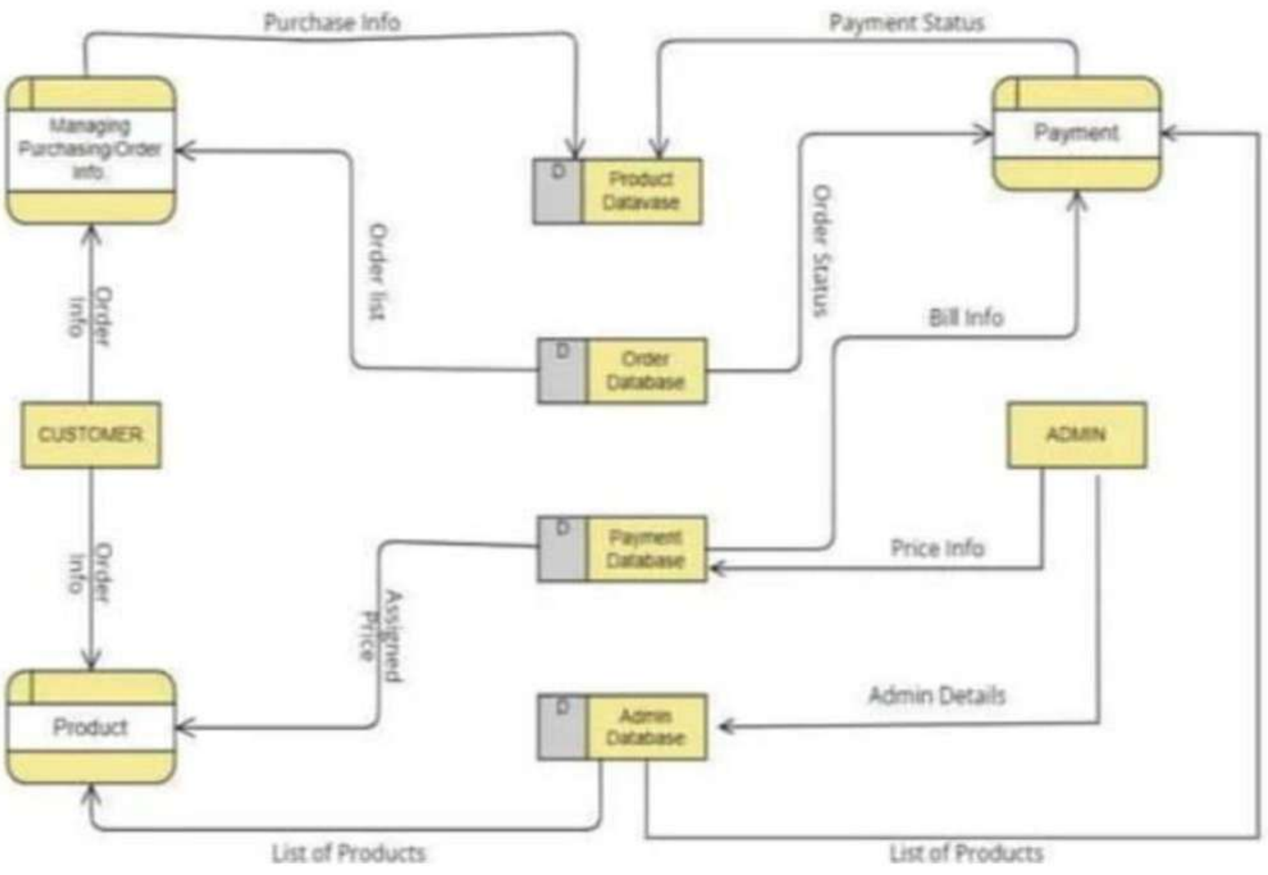
Entity Relationship Matrix

Primary Keys

Attributes

Mapping of Attributes with Entities

Result:



Purchase Info
Payment Status
Managing Purchasing/Order Info.
D
Product Datavase
Payment
Order list
Order Status
Order Info
Bill Info
D
Order Database
CUSTOMER
ADMIN
D
Payment Database
Price Info
Order Info
Assigned Price
Admin Details
Product
D
Admin Database
List of Products
List of Products

EX NO:4	DRAW THE DATA FLOW DIAGRAMS AT LEVEL 0 AND LEVEL 1
DATE:	

AIM:

To Draw the Data Flow Diagram for Smart Grocery List Manager and List the Modules in the Application.

ALGORITHM:

1. Open the Visual Paradigm to draw DFD (Ex.Lucidchart)
2. Select a data flow diagram template
3. Name the data flow diagram
4. Add an external entity that starts the process
5. Add a Process to the DFD
6. Add a data store to the diagram
7. Continue to add items to the DFD
8. Add data flow to the DFD
9. Name the data flow
10. Customize the DFD with colours and fonts
11. Add a title and share your data flow diagram

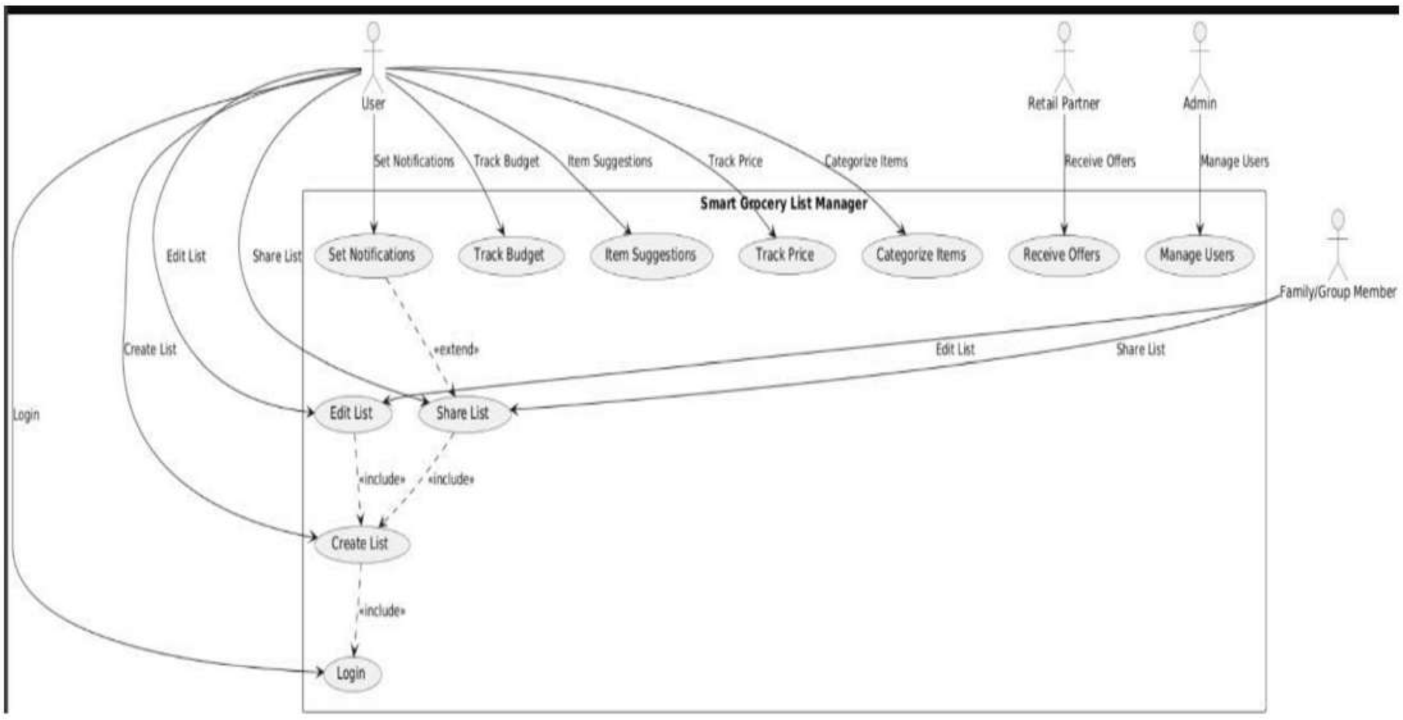
INPUT:

Processes

Datastores

External Entities

Result:



User
Retail Partner
Admin
Family/Group Member
Smart Grocery List Manager
Set Notifications
Track Budget
Item Suggestions
Track Price
Categorize Items
Receive Offers
Manage Users
Edit List
Share List
Create List
Login
«extend»
«include»
«include»
«include»

EX NO:5	DRAW USE CASE DIAGRAM
DATE:	

AIM:

To Draw the Use Case Diagram for Smart Grocery List Manager

ALGORITHM:

Step 1: Identify Actors

Step 2: Identify Use Cases

Step 3: Connect Actors and Use Cases

Step 4: Add System Boundary

Step 5: Define Relationships

Step 6: Review and Refine

Step 7: Validate

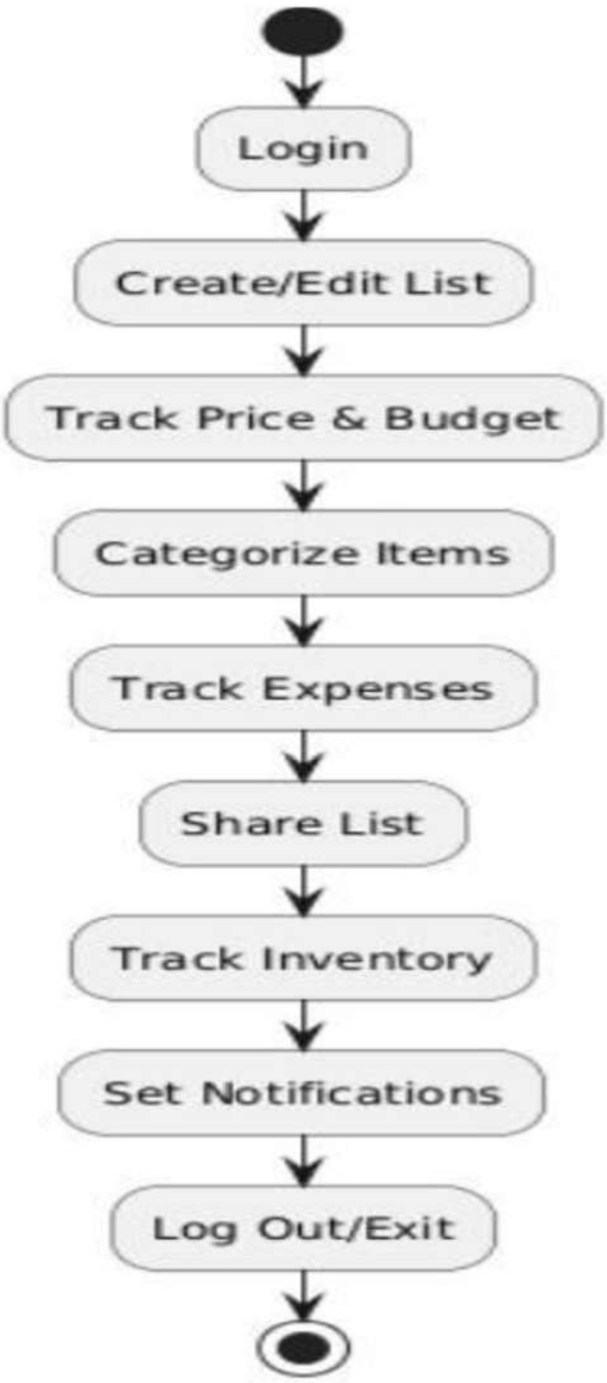
INPUTS:

Actors

Use Cases

Relations

Result:


Login
Create/Edit List
Track Price & Budget
Categorize Items
Track Expenses
Share List
Track Inventory
Set Notifications
Log Out/Exit

EX NO:6	**DRAW ACTIVITY DIAGRAM OF ALL USE CASES.**
DATE:	

AIM:

To Draw the activity Diagram for Smart Grocery List Manager

ALGORITHM:

Step 1: Identify the Initial State and Final States

Step 2: Identify the Intermediate Activities Needed

Step 3: Identify the Conditions or Constraints

Step 4: Draw the Diagram with Appropriate Notations

INPUTS:

Activities

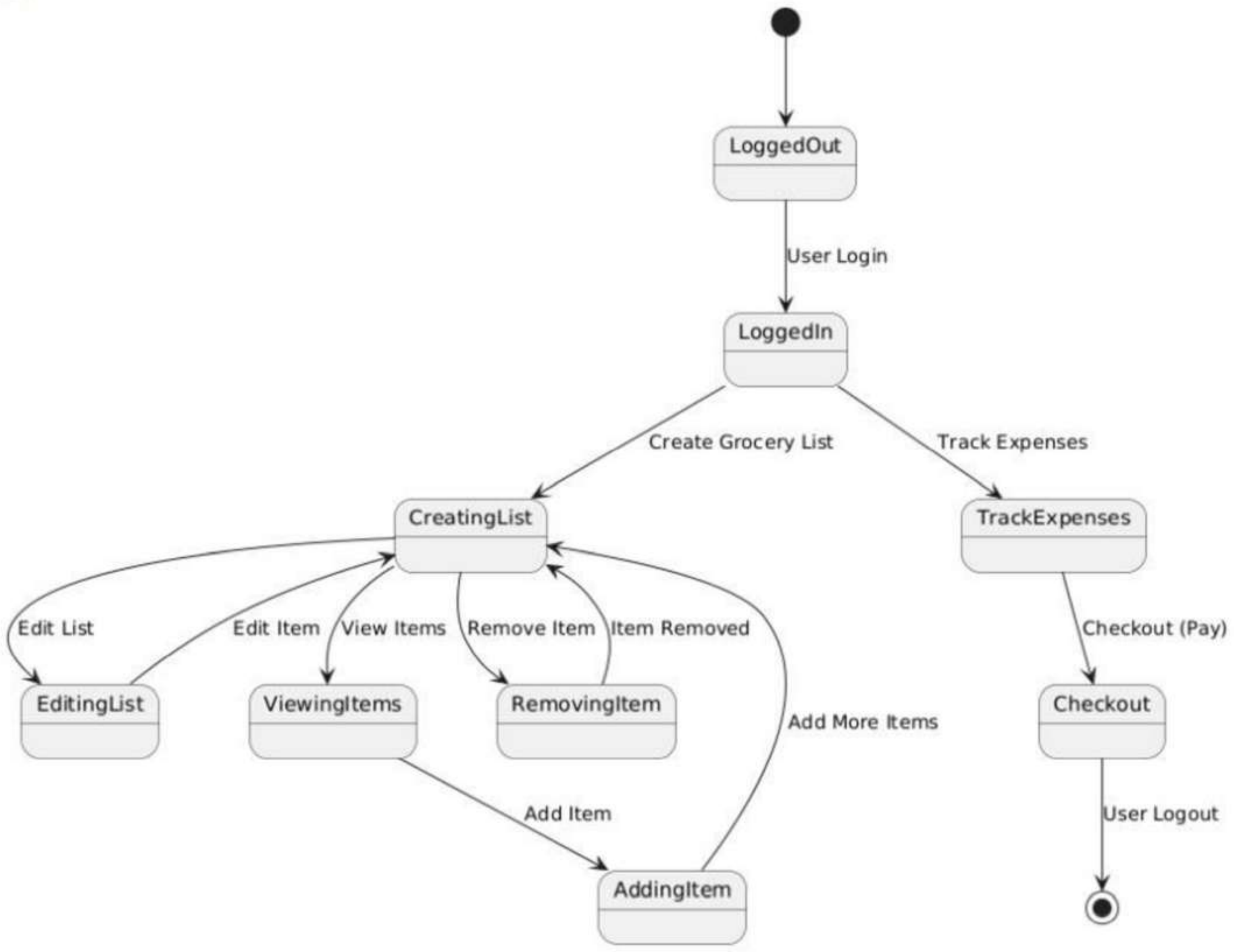
Decision Points

Guards

Parallel Activities

Conditions

Result:


LoggedOut
User Login
LoggedIn
Create Grocery List
Track Expenses
CreatingList
TrackExpenses
Edit List
Edit Item
View Items
Remove Item
Item Removed
Checkout (Pay)
EditingList
ViewingItems
RemovingItem
Add More Items
Checkout
Add Item
User Logout
AddingItem

EX NO:7	**DRAW STATE CHART DIAGRAM OF ALL USE CASES.**
DATE:	

AIM:

To Draw the State Chart Diagram for Smart Grocery List Manager

ALGORITHM:

STEP-1: Identify the important objects to be analysed.

STEP-2: Identify the states.

STEP-3: Identify the events.

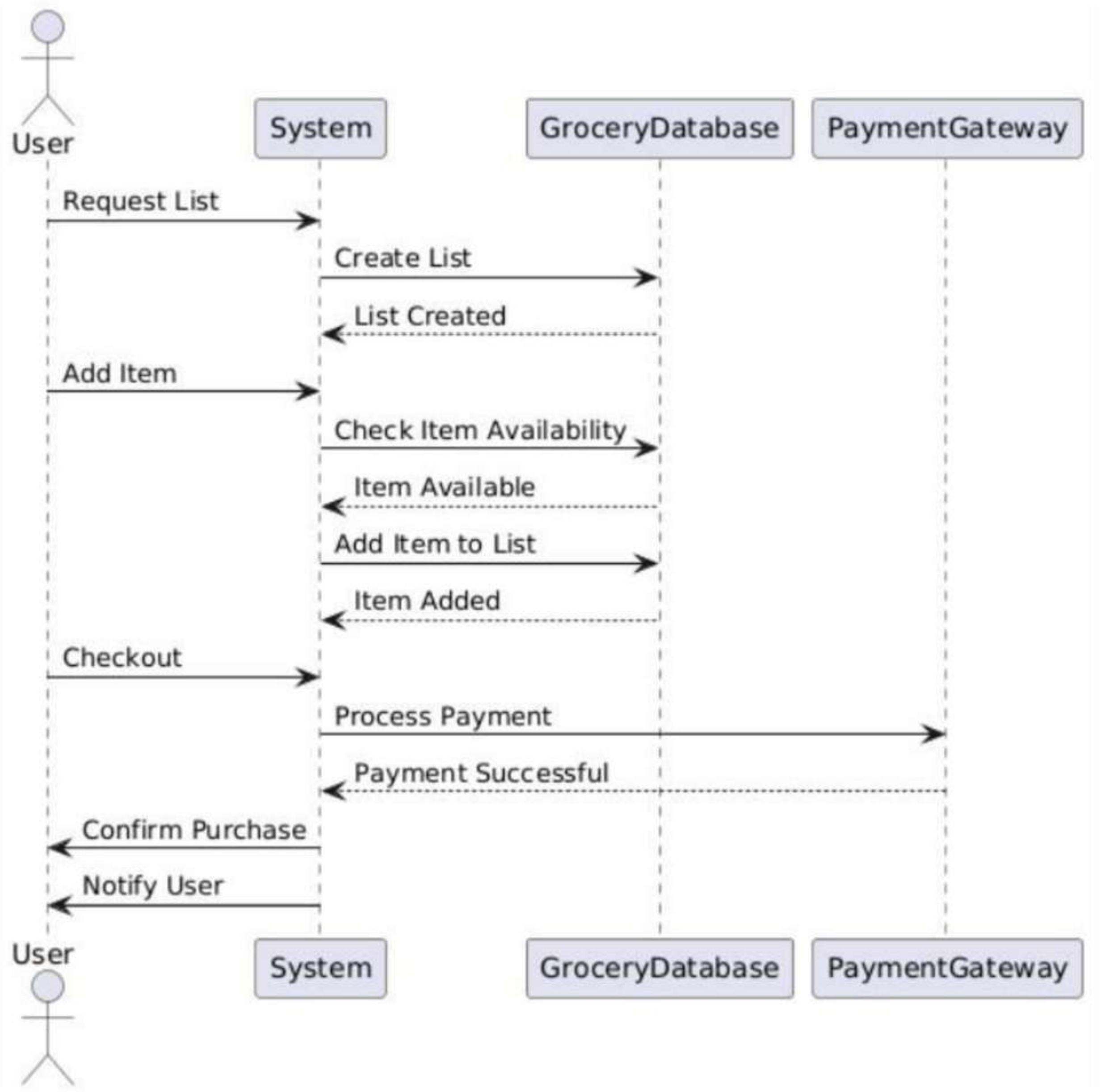
INPUTS:

Objects

States

Events

Result:


User
System
GroceryDatabase
PaymentGateway
Request List
Create List
List Created
Add Item
Check Item Availability
Item Available
Add Item to List
Item Added
Checkout
Process Payment
Payment Successful
Confirm Purchase
Notify User
User
System
GroceryDatabase
PaymentGateway

EX NO:8	**DRAW SEQUENCE DIAGRAM OF ALL USE CASES.**
DATE:	

AIM: To Draw the Sequence Diagram for Smart Grocery List Manager

ALGORITHM:

1. Identify the Scenario
2. List the Participants
3. Define Lifelines
4. Arrange Lifelines
5. Add Activation Bars
6. Draw Messages
7. Include Return Messages
8. Indicate Timing and Order
9. Include Conditions and Loops
10. Consider Parallel Execution
11. Review and Refine
12. Add Annotations and Comments
13. Document Assumptions and Constraints
14. Use a Tool to create a neat sequence diagram

INPUTS:

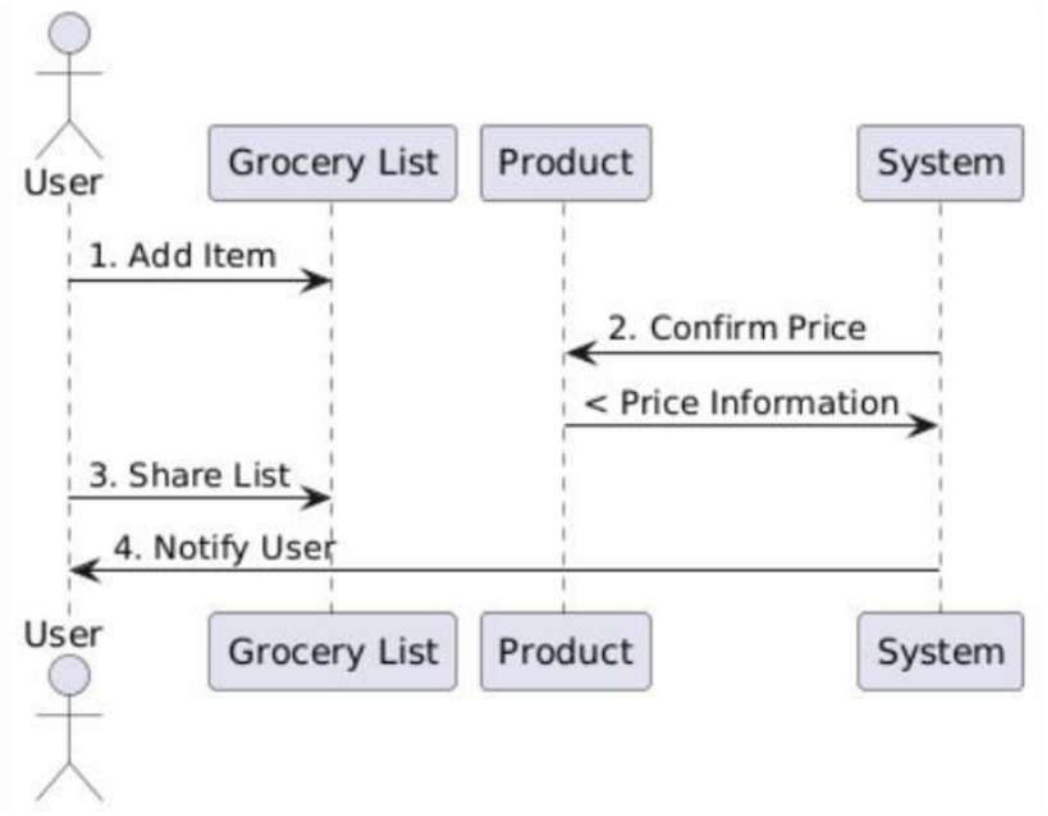
Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

Result:


User
Grocery List
Product
System
1. Add Item
2. Confirm Price
< Price Information
3. Share List
4. Notify User
User
Grocery List
Product
System

EX NO:9	**DRAW COLLABORATION DIAGRAM OF ALL USE CASES**
DATE:	

AIM:

To Draw the Collaboration Diagram for Smart Grocery List Manager

ALGORITHM:

Step 1: Identify Objects/Participants

Step 2: Define Interactions

Step 3: Add Messages

Step 4: Consider Relationships

Step 5: Document the collaboration diagram along with any relevant explanations or annotations.

INPUTS:

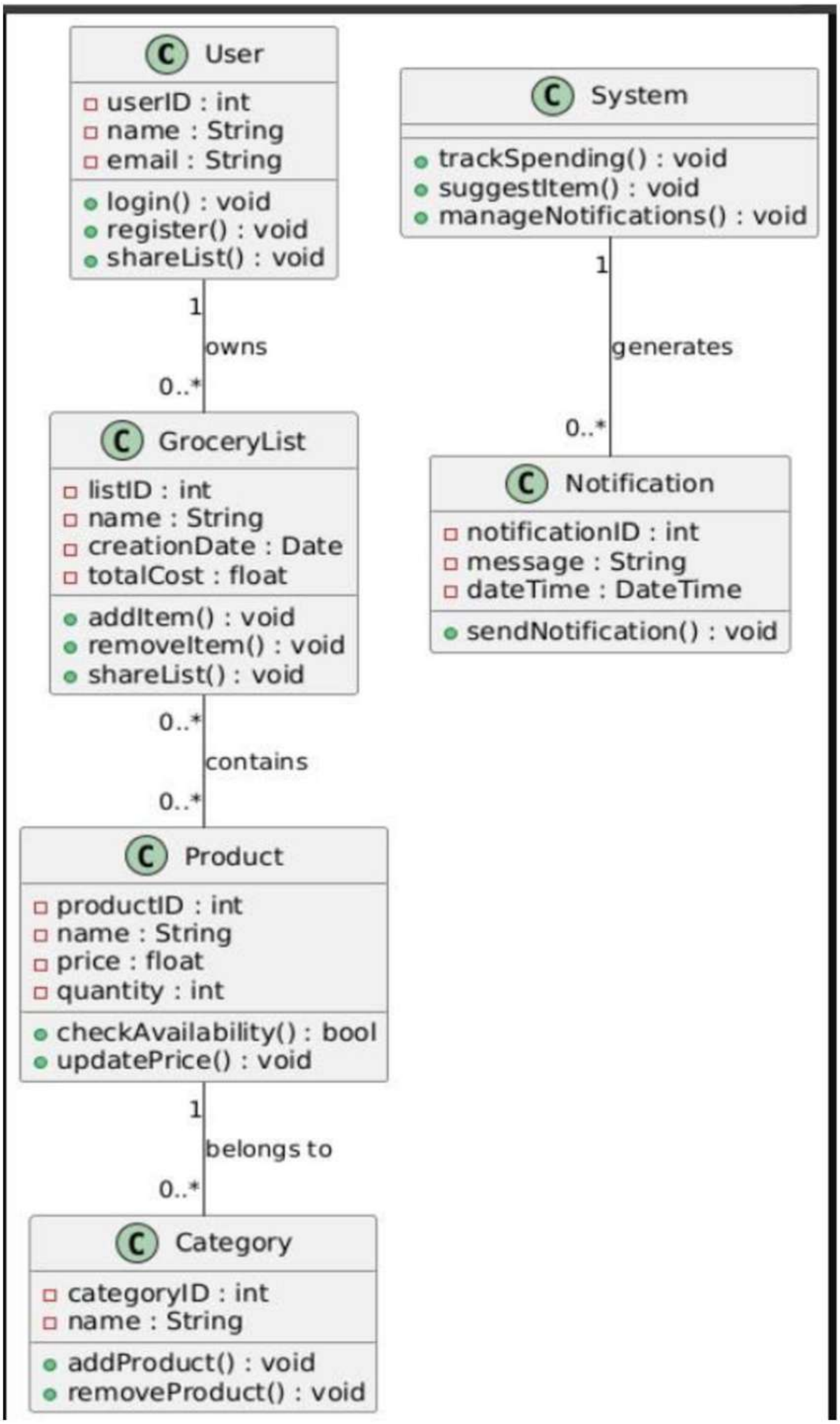
Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

Result:


User
- userID : int
- name : String
- email : String
+ login() : void
+ register() : void
+ shareList() : void
System
+ trackSpending() : void
+ suggestItem() : void
+ manageNotifications() : void
1
owns
0..*
1
generates
0..*
GroceryList
- listID : int
- name : String
- creationDate : Date
- totalCost : float
+ addItem() : void
+ removeItem() : void
+ shareList() : void
Notification
- notificationID : int
- message : String
- dateTime : DateTime
+ sendNotification() : void
0..*
contains
0..*
Product
- productID : int
- name : String
- price : float
- quantity : int
+ checkAvailability() : bool
+ updatePrice() : void
1
belongs to
0..*
Category
- categoryID : int
- name : String
+ addProduct() : void
+ removeProduct() : void

EX NO:10	**ASSIGN OBJECTS IN SEQUENCE DIAGRAM TO CLASSES AND MAKE CLASS DIAGRAM.**
DATE:	

AIM:

To Draw the Class Diagram for Smart Grocery List Manager

ALGORITHM:

1. Identify Classes
2. List Attributes and Methods
3. Identify Relationships
4. Create Class Boxes
5. Add Attributes and Methods
6. Draw Relationships
7. Label Relationships
8. Review and Refine
9. Use Tools for Digital Drawing

INPUTS:

1. Class Name
2. Attributes
3. Methods
4. Visibility Notation

RESULT:

EX NO:11	**Mini Project- Smart Grocery List Manager**
DATE	

AIM:

To develop a **Smart Grocery List Manager (SGLM)** using **Streamlit** and **MySQL** . The system will allow users to efficiently manage grocery lists, track pantry inventory, set expiration date reminders, and manage budgets. The objective is to automate grocery management, improve household efficiency, and minimize food wastage.

ALGORITHM:

Database Connection Initialization:

1. Establish a connection to the **MySQL** database to store data like user details, grocery list items, pantry inventory, expiration dates, and transaction logs.

Streamlit Interface Setup:

1. Create a simple and interactive web interface using **Streamlit** for managing grocery lists, inventory updates, and viewing relevant notifications.

Operation Selection:

1. **Add Items to List**: Allow users to input and add grocery items to the list.
2. **View Grocery List**: Display the current grocery list for the user to view and manage.
3. **Update Pantry Inventory**: Enable users to update quantities and item details in their pantry.
4. **Set Expiration Date Reminders**: Users can set reminders for expiration dates to avoid food spoilage.

Database Query Execution:

1. Perform operations like adding, updating, and deleting grocery items in the **MySQL** database based on user inputs.

Expiration Reminder Notification:

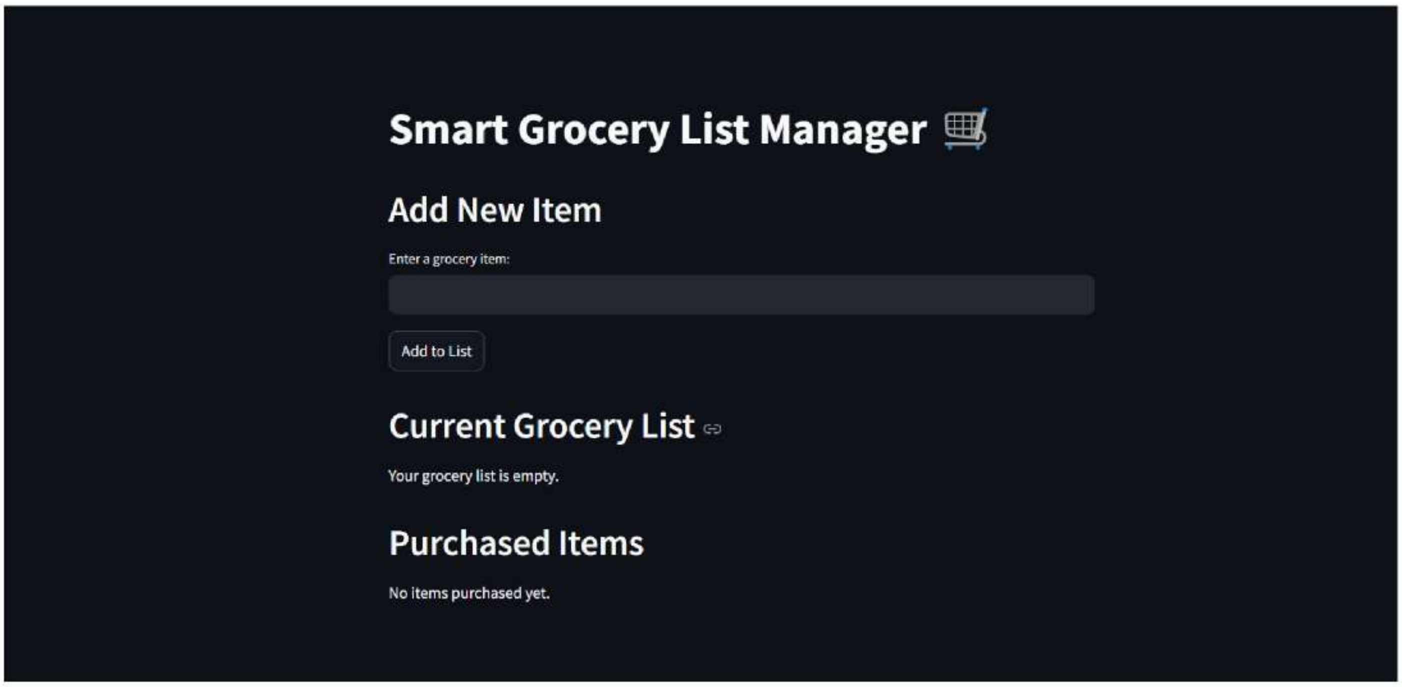
1. Set up a system to send reminders to users for items that are close to their expiration dates. This encourages timely use or donation of items to avoid wastage.

Budget Tracking & Recommendations:

1. Track user spending on groceries and provide budget summaries.
2. Suggest alternative, budget-friendly items based on user preferences and past purchases.

Feedback Display

1. Display feedback to users when adding items, updating inventory, receiving expiration reminders, or any low-stock notifications.



Smart Grocery List Manager 🛒

Add New Item

Enter a grocery item:

Current Grocery List

Your grocery list is empty.

Purchased Items

No items purchased yet.

PROGRAM:

```
import streamlit as st
```

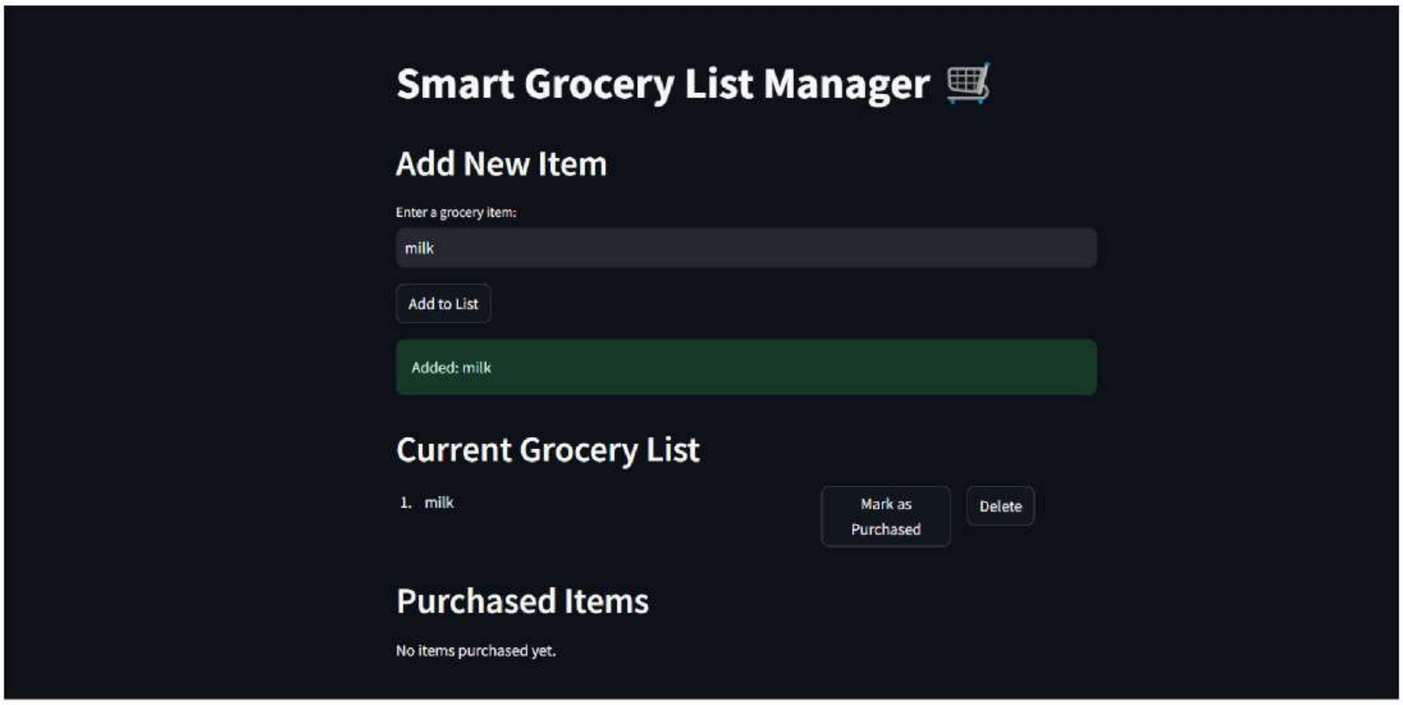
```
# Initialize session state
if "grocery_list" not in st.session_state:
    st.session_state["grocery_list"] = []

if "purchased_items" not in st.session_state:
    st.session_state["purchased_items"] = []

st.title("Smart Grocery List Manager 🛒")

# Input section to add new items
st.header("Add New Item")
item = st.text_input("Enter a grocery item:")
if st.button("Add to List"):
    if item:
        st.session_state["grocery_list"].append(item)
        st.success(f"Added: {item}")
    else:
        st.error("Please enter an item.")

# Display the current grocery list
st.header("Current Grocery List")
if st.session_state["grocery_list"]:
    for i, item in enumerate(st.session_state["grocery_list"]):
        col1, col2, col3 = st.columns([6, 2, 2])
        with col1:
            st.write(f"{i+1}. {item}")
        with col2:
            if st.button("Mark as Purchased", key=f"purchase_{i}"):
                st.session_state["purchased_items"].append(item)
                st.session_state["grocery_list"].remove(item)
                st.experimental_rerun()
```



Smart Grocery List Manager 🛒

Add New Item

Enter a grocery item:

milk

Added: milk

Current Grocery List

1. milk

Purchased Items

No items purchased yet.

with col3:

```
        if st.button("Delete", key=f"delete_{i}"):
            st.session_state["grocery_list"].remove(item)
            st.experimental_rerun()
else:
    st.write("Your grocery list is empty.")

# Display purchased items
st.header("Purchased Items")
if st.session_state["purchased_items"]:
    st.write(st.session_state["purchased_items"])
else:
    st.write("No items purchased yet.")
```