Document 6- Please prepare a use case diagram, activity diagram and a use case specification document.

Answer:

A use case diagram is a type of Unified Modeling Language (UML) diagram.

It is used in software and system design to show how different types of users (called actors) interact with a system to achieve their goals.

It visually represents the functional requirements of a system, showing the system's use cases, actors, and the relationships among them.

Component of use case:

1.Actor: It can be primary or secondary actor who interact with the system.

- **Primary Actor**: The main entity that initiates and interacts with the system to achieve a goal.
- **Secondary Actor**: A supporting entity that assists the system but does not initiate interactions.

2.System boundry: Defines the scope of the system and what's included within it, usually shown as a rectangle that contains all use cases.

3.Use case: It is the function or activity perform by the actor.

1. Main Use Case

- A **core** or **primary** process that achieves a key goal.

2. Supporting Use Case

- A **sub-process** that helps complete the main use case.

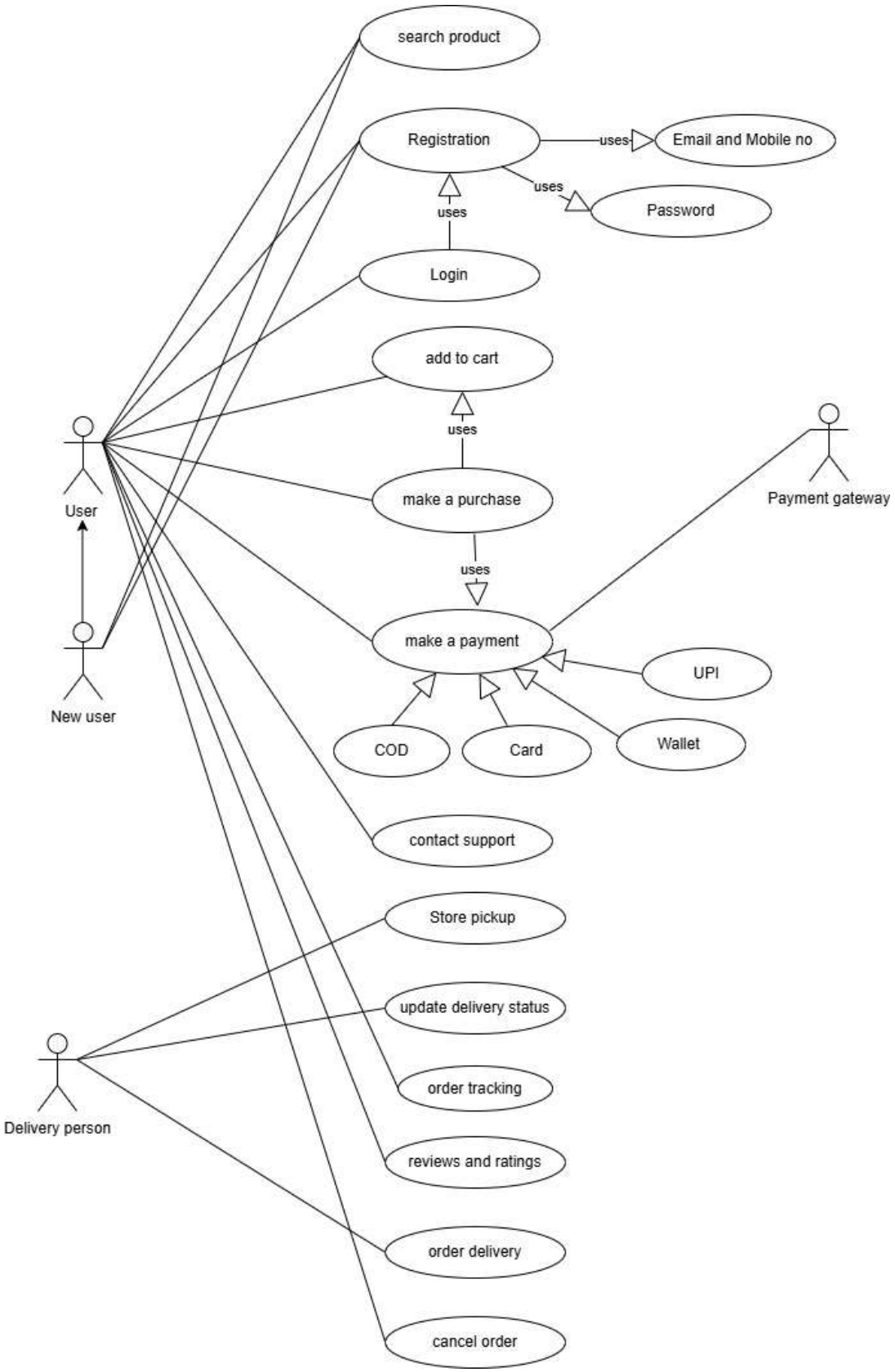
4.Relationship:

1. Association: It is relationship between actor and use case.

2.Include: Shows that one use case is always included with another usecase.

3.Extend: Indicates an optional or conditional relationship.

4.Generalization: Used when actors or use cases have a hierarchical relationship, showing inheritance or commonalities among actors.


search product
Registration
uses
Email and Mobile no
uses
Password
uses
Login
add to cart
uses
User
make a purchase
Payment gateway
uses
make a payment
UPI
New user
COD
Card
Wallet
contact support
Store pickup
update delivery status
Delivery person
order tracking
reviews and ratings
order delivery
cancel order

Activity diagram on use cases: Activity diagram is basically a flow chart to represent the flow from one activity to another activity. An **activity diagram** is a type of diagram commonly used in **Unified Modeling Language (UML)** to visually represent the flow of activities, decisions, and processes within a system.

Activity diagram includes:

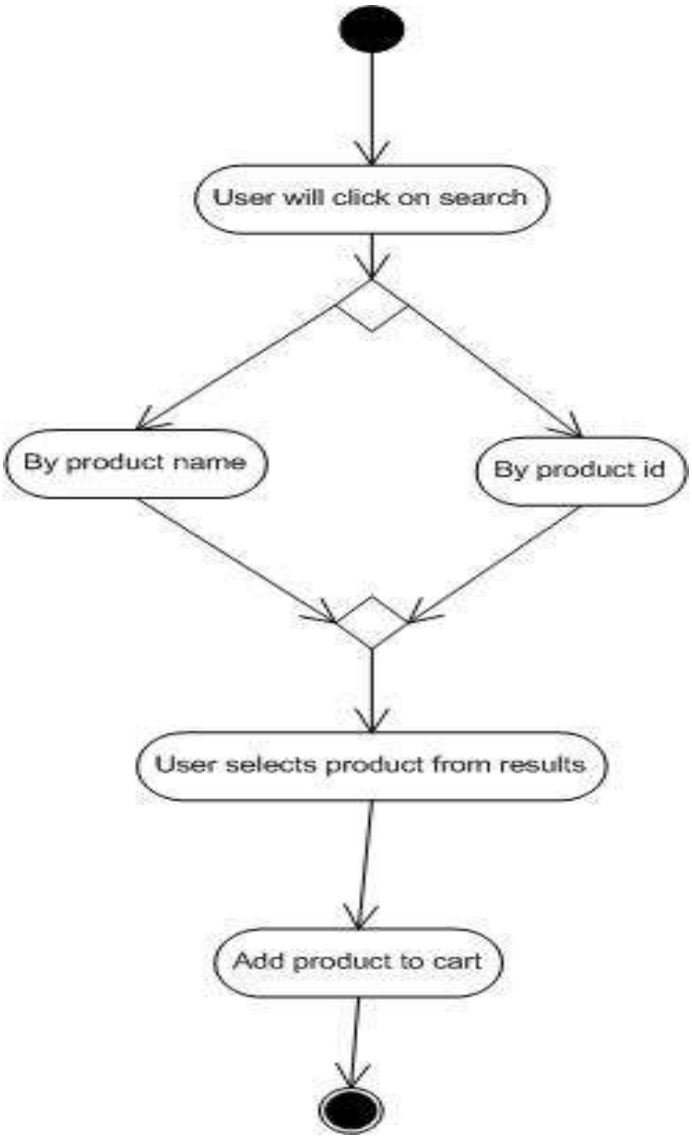
- **Initial Node**: The starting point of the process, shown as a filled black circle.
- **Activity/Action Nodes**: Steps or tasks in the process, represented by rounded rectangles.
- **Decision Node**: A point where a choice is made, shown as a diamond.
  Leads to different paths based on conditions (like "yes" or "no").
- **Guard condition**: A guard condition is a rule that decides if a path in a flow can be taken. It's written in brackets, like [condition], and must be true for the flow to continue on that path.
  Example: [Password is correct] leads to "Login successful."
- **Branch node** : Represents a decision point where the flow **splits** into multiple alternative paths based on conditions.
- **Example**: After **"Make Payment"**, the system decides whether the **payment is successful** or **failed**.
- **Merge Node**: Brings multiple paths back into one, also shown as a diamond.
- After handling **successful** and **failed** payments separately, both paths merge back into a common action like **"Show Order Status"**.
- **Fork and Join Nodes:**

  Fork: Splits the flow into parallel actions.

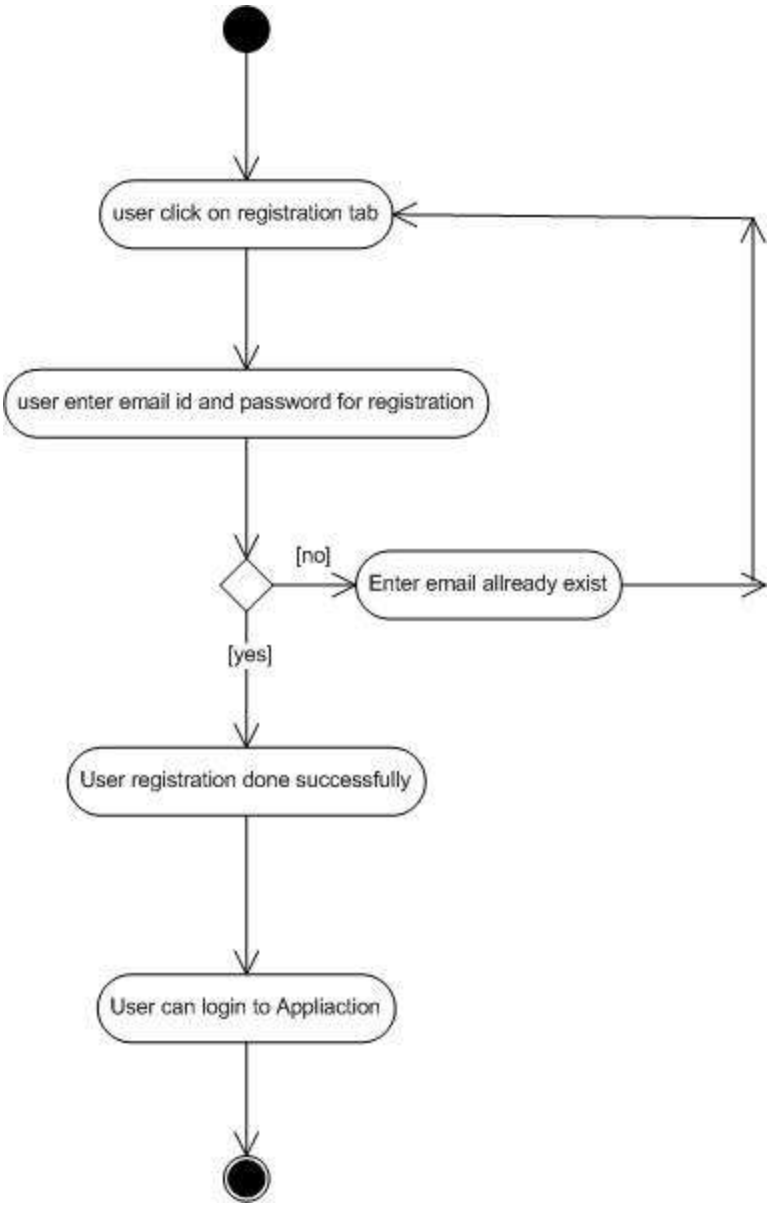
  Join: Combines parallel actions back into one flow.

- **Flow/Control Arrows**: Arrows show the direction of the flow between actions.
- **Final Node**: The end of the process, shown as a black circle with a surrounding ring.

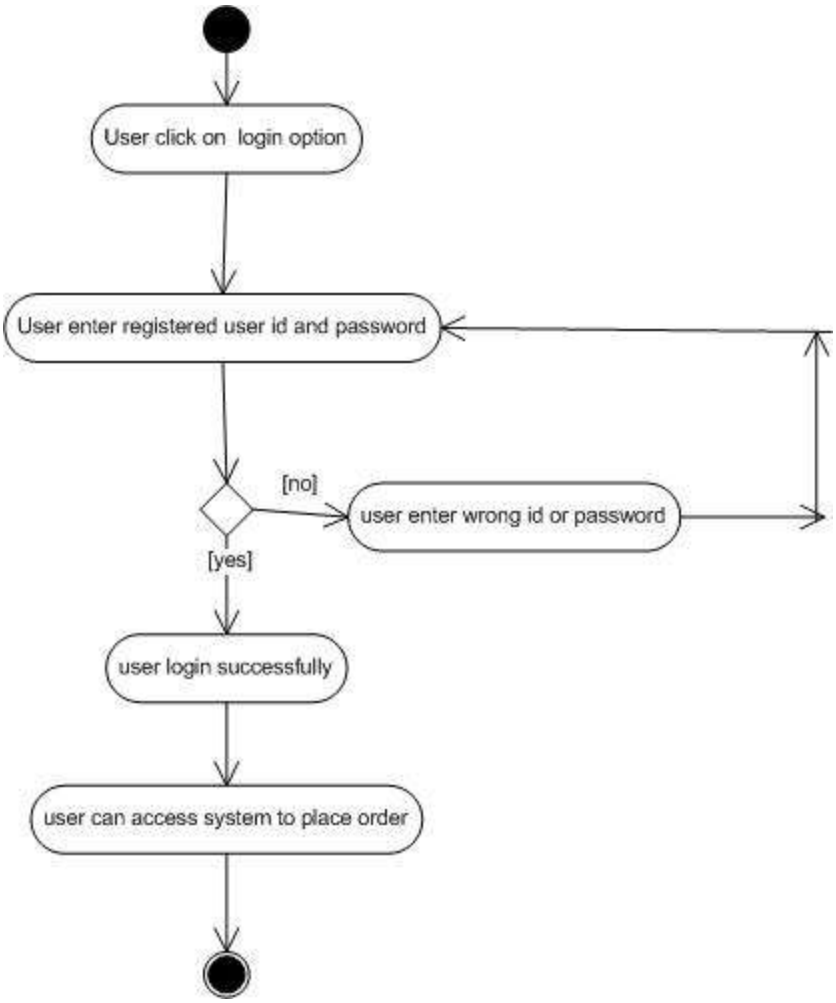
1.Activity diagram on search:



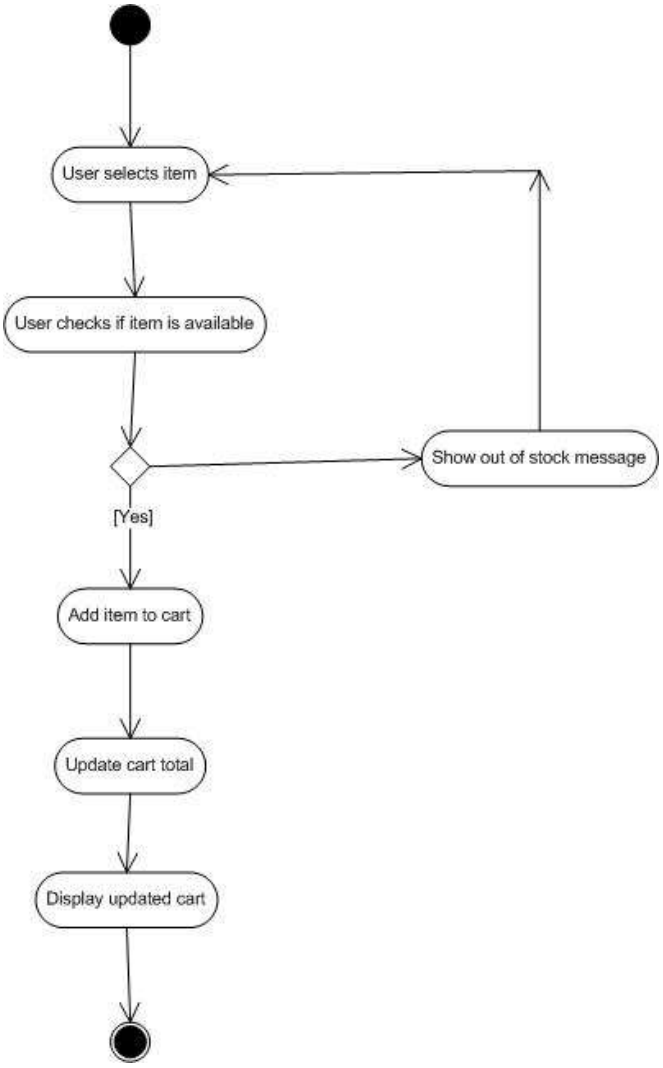
2.Activity diagram on registration:



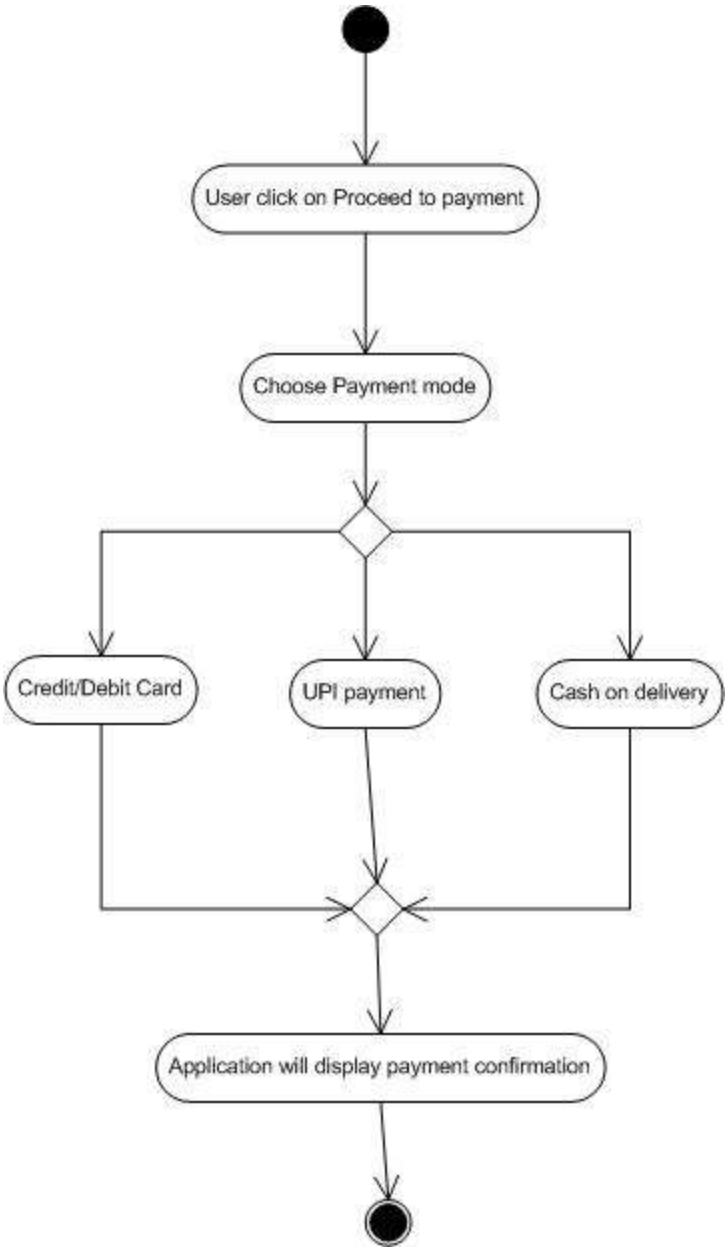
3.**Activity diagram on login:**



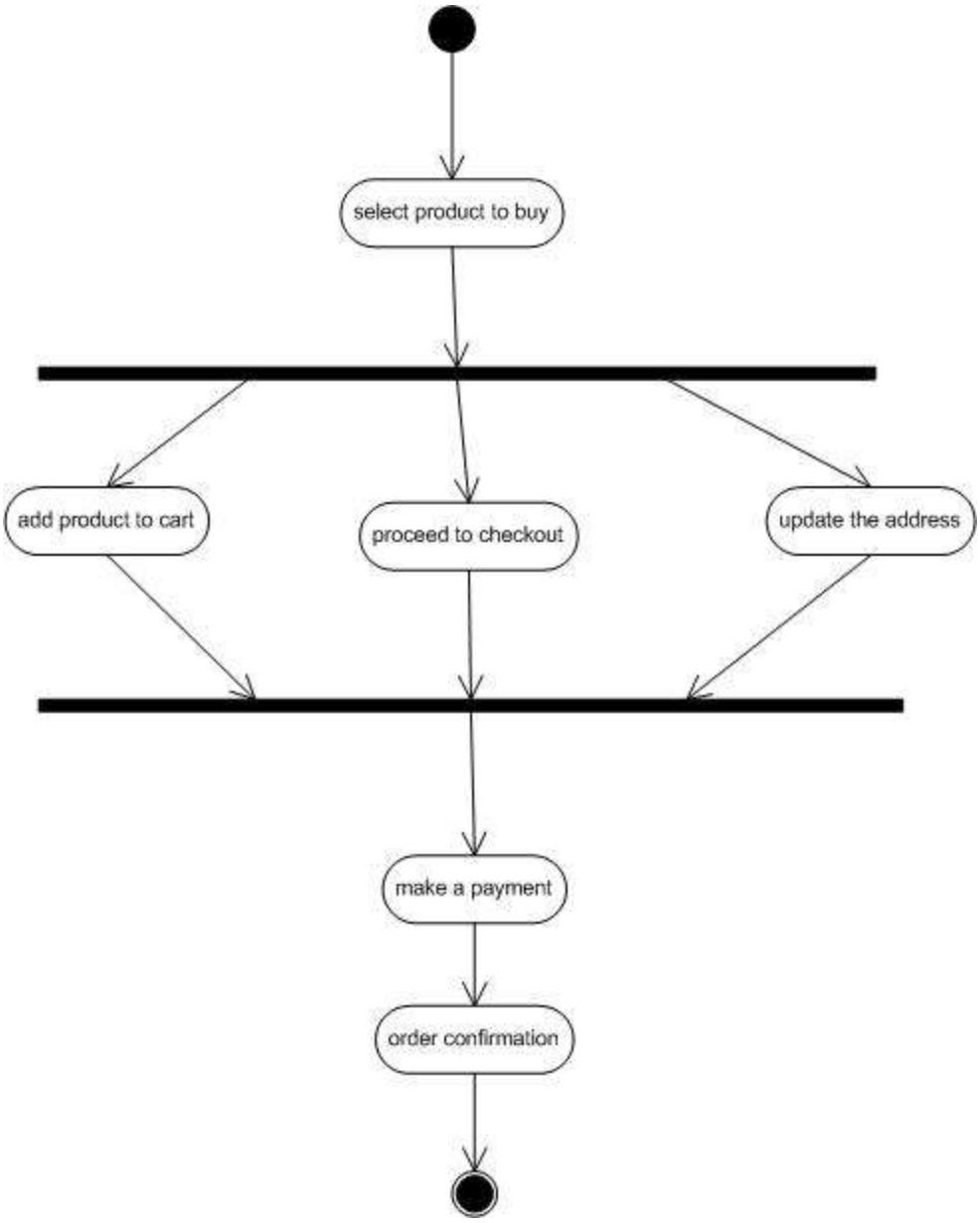
4. Activity diagram for add to cart:



5.Activity diagram on make payment:



6.Activity diagram on buy a product:



1.Use Case Name: Search Product

Use Case Description:

This use case allows a user to search for products in the Fraazo online store by entering keywords, product names, or categories. The system retrieves relevant results and displays them to the user.

Actors:

- **Primary Actor:** User
- **Secondary Actor:** None

Basic Flow:

1. The user accesses the Fraazo online store.
2. The user enters a search query (product name, category, or keyword).
3. The system processes the query and retrieves matching products from the database.
4. The system displays the search results, including product names, images, and prices.
5. The user can click on a product for more details or refine the search.

Alternate Flow:

- **AF1: No Matching Products Found**
    1. The system does not find any products matching the search query.
    2. The system displays a message stating, "No products found. Please try different keywords."
    3. The user can enter a new search query.

Exceptional Flow:

- **EF1: System Error**
    1. If there is an issue retrieving products from the database, the system displays an error message.
    2. The user can retry the search after some time.

Pre-Conditions:

- The user must have access to the Fraazo online store.
- The product database must be available.

Post-Conditions:

- The system displays the search results to the user.

Assumptions:

- The user knows what they are searching for.
- The system has an efficient search algorithm.

Constraints:

- The search function should return results within a few seconds.
- The system should handle a large number of simultaneous searches.

Dependencies:

- The search feature depends on the product database and indexing service.

Inputs and Outputs:

- **Input:** Search query entered by the user.
- **Output:** A list of matching products with images, prices, and details.

Business Rules:

- The system should rank search results based on relevance.
- Popular or trending products may be displayed first.

Miscellaneous Information:

- The system may provide search suggestions based on user input.
- Filters and sorting options can enhance the search experience.

2. Use Case Name: Registration

Use Case Description:

This use case allows a new user to create an account on the Fraazo online store by providing necessary details such as email, mobile number, password, and OTP verification. Once registered, the user can log in and access the store's features.

3. Actors:

- **Primary Actor:** New User
- **Secondary Actor:** None

4. Basic Flow:

1. The new user selects the "Register" option on the Fraazo online store.
2. The system prompts the user to enter their **email address** and **mobile number**.
3. The system sends an **OTP (One-Time Password)** to the provided mobile number.
4. The user enters the received OTP for verification.
5. The system verifies the OTP and prompts the user to create a **password**.
6. The user enters and confirms their password.
7. The system successfully registers the user and displays a confirmation message.
8. The user can now log in with the registered credentials.

5. Alternate Flow:

- **AF1: User Enters Incorrect OTP**

1. The system notifies the user that the OTP is incorrect.
2. The user can request a new OTP and re-enter it.

- **AF2: User Already Registered**
    1. If the user tries to register with an existing email or mobile number, the system displays an error message.
    2. The user is prompted to log in instead or recover their password if forgotten.

6. Exceptional Flow:

- **EF1: OTP Not Received**
    1. The user does not receive the OTP.
    2. The system allows the user to request a new OTP.
    3. If multiple OTP requests fail, the system suggests checking the mobile network or email spam folder.
- **EF2: Weak Password**
    1. The user enters a weak password (e.g., too short, missing special characters).
    2. The system prompts the user to create a stronger password following security guidelines.

7. Pre-Conditions:

- The user should not have an existing account.
- The user should have a valid mobile number and email address.
- The system should be able to send OTP messages.

8. Post-Conditions:

- The new user is successfully registered in the system.
- The user can log in using the provided credentials.

9. Assumptions:

- The user enters valid information (email, mobile number).
- The OTP system is functioning correctly.
- The system ensures data security during registration.

10. Constraints:

- The OTP should expire after a certain time (e.g., 5 minutes).
- Password should meet security criteria (minimum length, special characters, etc.).
- Multiple failed OTP attempts may temporarily block further attempts.

11. Dependencies:

- The registration process depends on an SMS/email service for OTP verification.
- The database must store user credentials securely.

12. Inputs and Outputs:

- **Input:** Email, mobile number, OTP, password.
- **Output:** Confirmation of successful registration or error messages for issues.

13. Business Rules:

- The system must verify that the email and mobile number are unique.
- The password must follow security guidelines.
- OTP should be time-limited and valid for single use.

14. Miscellaneous Information:

- The system may allow social media or Google login as an alternative registration method.
- The registration form should be user-friendly and mobile-compatible.

3. Use Case Name: Login

2. Use Case Description:

This use case allows a registered user to log in to the Fraazo online store using their email/mobile number and password. Once authenticated, the user gains access to the store's features, including searching for products, adding them to the cart, and making purchases.

3. Actors:

- **Primary Actor:** User (Registered User)
- **Secondary Actor:** None

4. Basic Flow:

1. The user selects the **"Login"** option on the Fraazo online store.
2. The system prompts the user to enter their **email or mobile number** and **password**.
3. The user enters the required credentials.
4. The system verifies the credentials against the stored database.
5. If the credentials are correct, the system grants access to the user's account.
6. The system redirects the user to the homepage/dashboard.

5. Alternate Flow:

- **AF1: User Enters Incorrect Password**

1. The system notifies the user that the password is incorrect.
2. The user is given the option to re-enter the password or reset it.

- **AF2: User Chooses "Forgot Password"**
    1. The user clicks on the "Forgot Password" link.
    2. The system prompts the user to enter their registered email or mobile number.
    3. The system sends a **password reset OTP** or **reset link** to the user's email/mobile.
    4. The user follows the instructions to reset their password.
    5. The user logs in using the new password.

6. Exceptional Flow:

- **EF1: User Not Registered**
    1. The user enters an email or mobile number that is not registered.
    2. The system notifies the user that no account exists with the entered details.
    3. The system suggests registering as a new user.
- **EF2: Account Locked Due to Multiple Failed Attempts**
    1. The user enters incorrect login credentials multiple times (e.g., 5 failed attempts).
    2. The system temporarily locks the account for security reasons.
    3. The user receives a notification to reset their password or try again later.

7. Pre-Conditions:

- The user must be registered in the system.
- The system must have the user's credentials stored securely.

8. Post-Conditions:

- If successful, the user is logged into their account.
- If unsuccessful, the user may try again, reset the password, or register.

9. Assumptions:

- The user remembers their correct login credentials.
- The internet connection is stable.
- The authentication system is functioning correctly.

10. Constraints:

- The password must meet security requirements (e.g., minimum length, special characters).
- Multiple failed login attempts may result in a temporary account lock.

11. Dependencies:

- The login process depends on the authentication system and database verification.
- The system may integrate with third-party authentication providers (e.g., Google, Facebook).

12. Inputs and Outputs:

- **Input:** Email/Mobile number, Password.
- **Output:** Successful login confirmation or error messages.

13. Business Rules:

- Only registered users can log in.
- The system should securely encrypt and store passwords.
- Users must have a valid password reset mechanism.

14. Miscellaneous Information:

- Two-factor authentication (2FA) can be introduced for added security.
- Users may stay logged in using a **"Remember Me"** option.

1. Use Case Name:

Add to Cart

2. Use Case Description:

This use case allows a registered user to add selected products to their shopping cart. The user can modify the quantity, remove items, and proceed to checkout from the cart.

3. Actors:

- **Primary Actor:** User (Registered User)
- **Secondary Actor:** None

4. Basic Flow:

1. The user searches for or browses a product on the Fraazo online store.
2. The user selects a product and views its details.
3. The user clicks the **"Add to Cart"** button.
4. The system checks product availability.
5. If the product is available, the system adds the product to the user's shopping cart.
6. The system confirms the addition of the product and updates the cart.
7. The user can continue shopping or proceed to checkout.

5. Alternate Flow:

- **AF1: User Modifies Quantity in Cart**
    1. The user goes to the cart and increases/decreases the quantity of an item.
    2. The system updates the cart and recalculates the total price.
- **AF2: User Removes an Item from Cart**
    1. The user selects an item in the cart and chooses the "Remove" option.
    2. The system removes the item from the cart and updates the total price.

6. Exceptional Flow:

- **EF1: Product Out of Stock**
    1. The system checks availability before adding to the cart.
    2. If the product is out of stock, the system notifies the user and does not add the item.
- **EF2: User Not Logged In**
    1. If a guest user attempts to add an item to the cart, the system prompts them to log in or continue as a guest.
    2. The system may provide an option to save the cart for later.

7. Pre-Conditions:

- The user must be logged in (for registered users).
- The selected product must be available in stock.

8. Post-Conditions:

- If successful, the product is added to the shopping cart.
- If unsuccessful, the system notifies the user of any issues (e.g., out of stock, login required).

9. Assumptions:

- Users may add multiple products to the cart.
- The system updates cart details in real time.

10. Constraints:

- The cart should have a reasonable limit for the number of items.
- Some products may have quantity restrictions (e.g., max 5 per order).

11. Dependencies:

- The use case depends on the product catalog and stock management system.

- The system must track cart items for logged-in users across sessions.

12. Inputs and Outputs:

- **Input:** Product selection, quantity.
- **Output:** Updated cart, confirmation message.

13. Business Rules:

- Only available products can be added to the cart.
- The cart should auto-update when modifications are made.
- Price changes should reflect in real time.

14. Miscellaneous Information:

- Users may save the cart for later purchases.
- Promotions or discounts may apply to cart items dynamically.

4. Use Case Name: Make a Purchase

2. Use Case Description:

This use case allows a registered user to proceed with purchasing the selected items from their shopping cart by selecting a payment method and confirming the order.

3. Actors:

- **Primary Actor:** User (Registered User)
- **Secondary Actors:** Payment Gateway

4. Basic Flow:

1. The user navigates to the shopping cart.
2. The user reviews the items in the cart and modifies them if needed.
3. The user clicks on the **"Proceed to Checkout"** button.
4. The system displays available delivery options (e.g., home delivery, store pickup).
5. The user selects the preferred delivery method.
6. The system displays available payment options (e.g., UPI, Wallet, Card, Cash on Delivery).
7. The user selects a payment method.
8. The user confirms the order.
9. The system processes the payment (if applicable) via the payment gateway.
10. The system confirms the order placement and generates an order ID.
11. The system sends an order confirmation notification to the user via email/SMS.

5. Alternate Flow:

- **AF1: User Applies a Coupon Code**
    1. The user enters a discount or promotional code.
    2. The system validates the coupon and applies the discount if valid.
- **AF2: User Selects Store Pickup**
    1. The user chooses the **"Store Pickup"** option instead of home delivery.
    2. The system reserves the items and provides store details for pickup.

6. Exceptional Flow:

- **EF1: Payment Failure**
    1. If the payment transaction fails, the system notifies the user.
    2. The user is given the option to retry payment or select another payment method.
- **EF2: Item Becomes Unavailable During Checkout**
    1. The system checks product availability before order confirmation.
    2. If an item is out of stock, the system removes it from the cart and notifies the user.

7. Pre-Conditions:

- The user must have at least one item in the shopping cart.
- The selected products must be in stock.
- The user must have valid delivery details.

8. Post-Conditions:

- If successful, the order is placed, and payment (if applicable) is processed.
- If unsuccessful, the system notifies the user of the issue and provides options to retry.

9. Assumptions:

- Users can choose between multiple payment methods.
- Discounts or promotions may be applied at checkout.

10. Constraints:

- Some payment methods may not be available for specific locations.
- Cash on Delivery might not be applicable for high-value orders.

11. Dependencies:

- This use case depends on **Add to Cart** and **Make a Payment** use cases.
- The system must be able to communicate with the **Payment Gateway** for online payments.

12. Inputs and Outputs:

- **Inputs:** Selected products, delivery details, payment details.
- **Outputs:** Order confirmation, payment transaction details, order tracking information.

13. Business Rules:

- Orders can only be placed if the total cart value meets a minimum purchase requirement (if applicable).
- Discounts and promotions apply based on predefined rules.
- The system should automatically calculate applicable taxes and shipping fees.

14. Miscellaneous Information:

- The system may provide an estimated delivery date at checkout.
- Users should be able to cancel the order within a limited time if needed.

5. Use Case Name: Make a Payment

2. Use Case Description:

This use case allows the user to make a payment for an order using the available payment methods. The system processes the transaction and confirms the payment status.

3. Actors:

- **Primary Actor:** User (Registered User)
- **Secondary Actors:** Payment Gateway

4. Basic Flow:

1. The user proceeds to the checkout page.
2. The system displays available payment options (UPI, Wallet, Card, Cash on Delivery).
3. The user selects a preferred payment method.
4. If the user selects **Card, UPI, or Wallet**, they enter the required payment details.
5. The system redirects the user to the payment gateway for transaction processing.
6. The user completes the payment authentication (OTP, PIN, etc.).
7. The payment gateway confirms the transaction success and notifies the system.
8. The system updates the order status as **Payment Received**.
9. The system sends a payment confirmation notification via email/SMS.
10. If the user selects **Cash on Delivery**, the system marks the order for payment upon delivery.

5. Alternate Flow:

- **AF1: User Uses a Wallet or UPI**
    1. The user selects **Wallet/UPI** as the payment option.
    2. The system redirects to the respective payment service.
    3. The user completes the transaction and is redirected back to the system.
- **AF2: User Chooses Cash on Delivery**
    1. The user selects **Cash on Delivery (COD)**.
    2. The system marks the order as pending payment.
    3. Payment is collected at the time of delivery.

6. Exceptional Flow:

- **EF1: Payment Failure**
    1. If the payment fails, the system displays an error message.
    2. The user is given the option to retry or select a different payment method.
- **EF2: Insufficient Wallet Balance**
    1. If the wallet balance is insufficient, the system notifies the user.
    2. The user is prompted to add funds or use another payment method.
- **EF3: Payment Gateway Timeout**
    1. If the payment gateway does not respond, the system notifies the user.
    2. The user can retry or choose another payment method.

7. Pre-Conditions:

- The user must have an active internet connection for online payments.
- The total order amount must be calculated before payment.
- The selected payment method should be available for the user's location.

8. Post-Conditions:

- If successful, the payment is processed, and the order is confirmed.
- If unsuccessful, the system prompts the user to retry payment.

9. Assumptions:

- Users have valid payment methods.
- Payment gateway services are operational.

10. Constraints:

- Some payment options may not be available based on location or order value.
- Payment retries may be limited after multiple failures.

11. Dependencies:

- This use case depends on the **Make a Purchase** and **Payment Gateway**.

12. Inputs and Outputs:

- **Inputs:** Selected payment method, user payment details, order amount.
- **Outputs:** Payment confirmation, order status update, payment receipt.

13. Business Rules:

- Orders cannot be placed without a successful payment (except COD).
- Refunds must follow the store's return and refund policy.
- Transaction failures should allow multiple retry attempts.

14. Miscellaneous Information:

- Users may receive discounts or cashback based on payment method.
- Payment history should be accessible in the user's account.

6. Use Case Name: Contact Support

2. Use Case Description:

This use case enables users to contact customer support for assistance regarding orders, payments, delivery issues, or general inquiries. The system provides various support options, including chat, email, and phone support.

3. Actors:

- **Primary Actor:** User (Registered or Guest)
- **Secondary Actor:** Customer Support Representative

4. Basic Flow:

1. The user navigates to the "Contact Support" section of the online store.
2. The system displays available support options (Live Chat, Email, Phone Call).
3. The user selects a preferred support option.
4. The system prompts the user to enter details about their query (e.g., order number, issue description).
5. The user submits the request.
6. The system routes the request to the customer support team.

7. A customer support representative responds to the query via the chosen support channel.
8. The user receives a resolution or further instructions.
9. The system allows the user to rate the support experience.

5. Alternate Flow:

- **AF1: User Opts for Live Chat**
    1. The user selects the **Live Chat** option.
    2. A chat window opens, and a support agent joins the conversation.
    3. The user and agent communicate in real-time to resolve the issue.
- **AF2: User Opts for Email Support**
    1. The user selects the **Email** option.
    2. The system provides a form to enter details about the issue.
    3. The user submits the request.
    4. A support agent responds via email within a specified time.
- **AF3: User Opts for Phone Support**
    1. The user selects the **Phone Call** option.
    2. The system displays the support hotline number.
    3. The user dials the number and speaks with a representative.

6. Exceptional Flow:

- **EF1: No Available Support Agent**
    1. If no support agents are available for live chat, the system notifies the user.
    2. The system offers an option to leave a message or send an email instead.
- **EF2: User Provides Incomplete Information**
    1. If required fields (e.g., order number) are missing, the system prompts the user to complete them.
- **EF3: Phone Support Line Busy**
    1. If all phone lines are busy, the system suggests alternative contact methods.

7. Pre-Conditions:

- The user must have access to the support section.
- Internet connectivity is required for chat and email support.

8. Post-Conditions:

- The user's issue is logged and responded to by the support team.
- If resolved, the ticket is closed; otherwise, it remains open for follow-up.

9. Assumptions:

- The support team is available during business hours.

- The user provides accurate details for issue resolution.

10. Constraints:

- Response times may vary based on query complexity.
- Some support channels may have limited availability based on location.

11. Dependencies:

- This use case depends on **User Account Management** (for retrieving user details).

12. Inputs and Outputs:

- **Inputs:** User query, order details (if applicable), contact details.
- **Outputs:** Support ticket ID, confirmation message, resolution response.

13. Business Rules:

- Users should receive a response within the defined SLA (e.g., 24 hours for email).
- Support should be available during business hours.
- Issue escalation is required if unresolved within a specific time frame.

14. Miscellaneous Information:

- Users may be offered self-help FAQs before connecting to a support agent.
- Support interactions should be logged for quality assurance.

7. Use Case Name: Store Pickup

2. Use Case Description:

This use case describes the process where a delivery person picks up an order from a designated store location to deliver it to the customer.

3. Actors:

- **Primary Actor:** Delivery Person
- **Secondary Actors:** Store Staff, System

4. Basic Flow:

1. The system assigns a delivery request to the delivery person.
2. The delivery person receives pickup details, including store location and order ID.
3. The delivery person reaches the store and verifies the order details.

4. The store staff prepares the order and hands it over to the delivery person.
5. The delivery person confirms receipt of the order in the system.
6. The system updates the order status to "Out for Delivery."
7. The delivery person proceeds to deliver the order to the customer.

5. Alternate Flow:

- **AF1: Customer Cancels Order Before Pickup**
    1. If the customer cancels the order before pickup, the system notifies the delivery person.
    2. The delivery request is removed from the delivery person's queue.
- **AF2: Store Delay in Order Preparation**
    1. If the order is not ready at the store, the delivery person waits or contacts support.
    2. The system updates the estimated delivery time.

6. Exceptional Flow:

- **EF1: Delivery Person Cannot Locate Store**
    1. If the delivery person cannot find the store, they can contact the support team for assistance.
- **EF2: Order Items Are Missing or Damaged**
    1. If the order has missing or damaged items, the delivery person reports the issue to store staff.
    2. Store staff resolves the issue or escalates it.

7. Pre-Conditions:

- The order is ready for pickup at the store.
- The delivery person has access to the order details via the system.

8. Post-Conditions:

- The delivery person successfully picks up the order.
- The system updates the status to "Out for Delivery."

9. Assumptions:

- The store staff prepares orders on time.
- The delivery person follows the assigned route.

10. Constraints:

- The delivery person must pick up the order within the designated time frame.
- Order handover must follow verification procedures.

11. Dependencies:

- This use case depends on **Order Management** and **Delivery Tracking.**

12. Inputs and Outputs:

- **Inputs:** Order details, store location, verification information.
- **Outputs:** Order status update, confirmation of pickup.

13. Business Rules:

- The delivery person must verify the order details before pickup.
- If the order is not picked up within a set timeframe, reassignment may occur.

14. Miscellaneous Information:

- Delivery persons may receive real-time navigation to the store.
- The system may record proof of pickup via digital confirmation.

8. Use Case Name: Order Tracking

2. Use Case Description:

This use case allows customers to track the status and location of their orders in real-time from the moment of purchase until delivery. The system provides updates at different stages of the order fulfillment process.

3. Actors:

- **Primary Actor:** Customer
- **Secondary Actors:** System, Delivery Person

4. Basic Flow:

1. The customer logs into the application and navigates to the **"Order Tracking"** section.
2. The system displays a list of recent orders.
3. The customer selects an order to view its current status.
4. The system retrieves and displays the real-time status of the order, including:
    - Order placed
    - Order confirmed
    - Order being prepared
    - Order ready for pickup
    - Order picked up by the delivery person
    - Order out for delivery

    - Order delivered
5. If available, the system provides real-time tracking on a map with the estimated delivery time.
6. The system sends notifications to the customer at key stages (e.g., "Your order is on the way!").

5. Alternate Flow:

- **AF1: Customer Requests More Details**
    1. If the customer needs further information, they can contact support directly from the tracking page.
    2. The system provides contact options (chat, email, or call).
- **AF2: Delivery is Rescheduled**
    1. If the delivery is rescheduled, the system updates the estimated arrival time.
    2. The customer receives a notification about the new delivery schedule.

6. Exceptional Flow:

- **EF1: System Fails to Retrieve Tracking Information**
    1. If tracking data is unavailable, the system displays an error message.
    2. The customer is prompted to refresh or contact support for assistance.
- **EF2: Order is Delayed**
    1. If the order is delayed due to unforeseen circumstances (traffic, weather), the system updates the estimated delivery time.
    2. The customer is notified of the delay.

7. Pre-Conditions:

- The customer must have placed an order.
- The system must have real-time tracking enabled.

8. Post-Conditions:

- The customer successfully tracks their order until delivery.
- The order status updates correctly in the system.

9. Assumptions:

- The delivery person regularly updates order status.
- The tracking system functions without interruptions.

10. Constraints:

- Real-time tracking may depend on GPS and internet availability.

- Some areas may not support live tracking.

11. Dependencies:

- This use case depends on **Order Management, Delivery Tracking, and Notifications.**

12. Inputs and Outputs:

- **Inputs:** Order ID, customer request, GPS location (if applicable).
- **Outputs:** Order status updates, real-time location (if enabled), estimated delivery time.

13. Business Rules:

- Customers must be notified of significant order status changes.
- Tracking information must be updated in real time.
- If an order is delayed beyond a threshold, the customer must receive a compensation option (if applicable).

14. Miscellaneous Information:

- Customers may receive delivery verification (e.g., OTP, signature).
- The system may allow customers to contact the delivery person if needed.

9. Use Case Name: Delivers the Order

2. Use Case Description:

This use case describes how a delivery person successfully delivers an order to the customer, verifies the delivery, and updates the order status in the system.

3. Actors:

- **Primary Actor:** Delivery Person
- **Secondary Actors:** Customer, System

4. Basic Flow:

1. The delivery person picks up the order from the store or warehouse.
2. The system provides the delivery person with the customer's address and contact details.
3. The delivery person navigates to the customer’s location using the in-app GPS.
4. Upon arrival, the delivery person notifies the customer (via call or app notification).
5. The customer receives the order and verifies the items.
6. The delivery person marks the order as **"Delivered"** in the system.
7. The system updates the order status and notifies the customer.

8. The customer may provide feedback or rate the delivery.

5. Alternate Flow:

- **AF1: Customer Requests Contactless Delivery**
    1. The customer requests a contactless delivery option.
    2. The delivery person leaves the order at the designated location.
    3. The delivery person takes a confirmation photo and uploads it to the system.
    4. The system updates the status as **"Delivered – Contactless."**
- **AF2: Customer is Unavailable at Delivery Time**
    1. The delivery person attempts to contact the customer.
    2. If the customer is unreachable, the system notifies the customer about the missed delivery.
    3. The delivery person marks the status as **"Customer Unavailable – Delivery Attempted."**
    4. The system may allow rescheduling or return the order to the store.

6. Exceptional Flow:

- **EF1: Delivery is Delayed**
    1. If the delivery is delayed due to traffic or weather, the delivery person updates the status as **"Delayed – In Transit."**
    2. The system notifies the customer with a new estimated delivery time.
- **EF2: Order is Damaged During Transit**
    1. The delivery person reports the damage in the system.
    2. The system notifies the store and customer.
    3. The customer may choose to accept or reject the order.
    4. If rejected, the delivery person returns the order to the store.

7. Pre-Conditions:

- The delivery person has accepted the order for delivery.
- The system has provided accurate customer details and address.

8. Post-Conditions:

- The order is successfully delivered and marked as **"Completed."**
- If delivery fails, the system records the reason and updates the status.

9. Assumptions:

- The customer is available at the specified address.
- The delivery person follows the assigned route.

10. Constraints:

- The delivery person must complete the delivery within the expected time frame.
- The order must be verified before being marked as delivered.

11. Dependencies:

- This use case depends on **Order Tracking, Customer Notifications, and Delivery Management.**

12. Inputs and Outputs:

- **Inputs:** Order ID, customer details, GPS location.
- **Outputs:** Order status updates, customer notifications, delivery confirmation.

13. Business Rules:

- The delivery person must verify the recipient before handing over the order.
- If the customer refuses the order, the system must record the reason.
- Contactless deliveries require photographic proof.

14. Miscellaneous Information:

- The customer may be required to enter an OTP to confirm delivery.
- The system may allow real-time chat between the customer and delivery person.

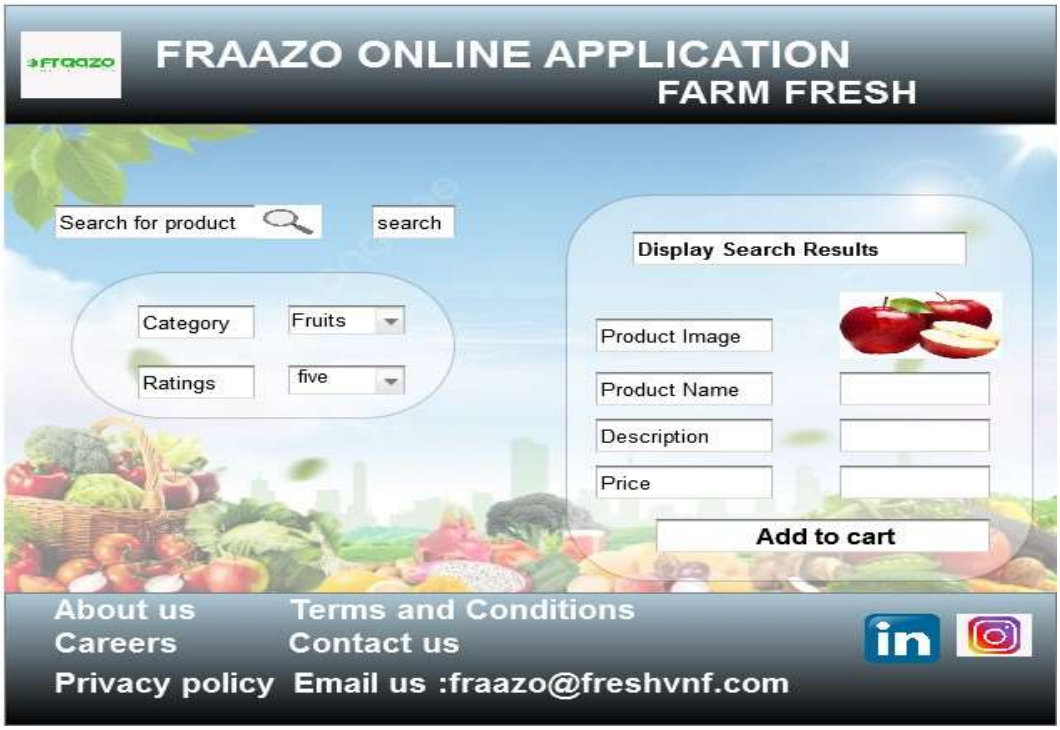
Document 7- Screens and pages

Please follow the following steps to create the mock-ups

Answer:

- **Wireframe**: A basic sketch that shows the layout of a page without colors or images.
- **Mockup**: A detailed picture of how the final design will look, but it's not clickable.
- **Prototype**: A clickable model that shows how the app or website will work.

1. Search page:



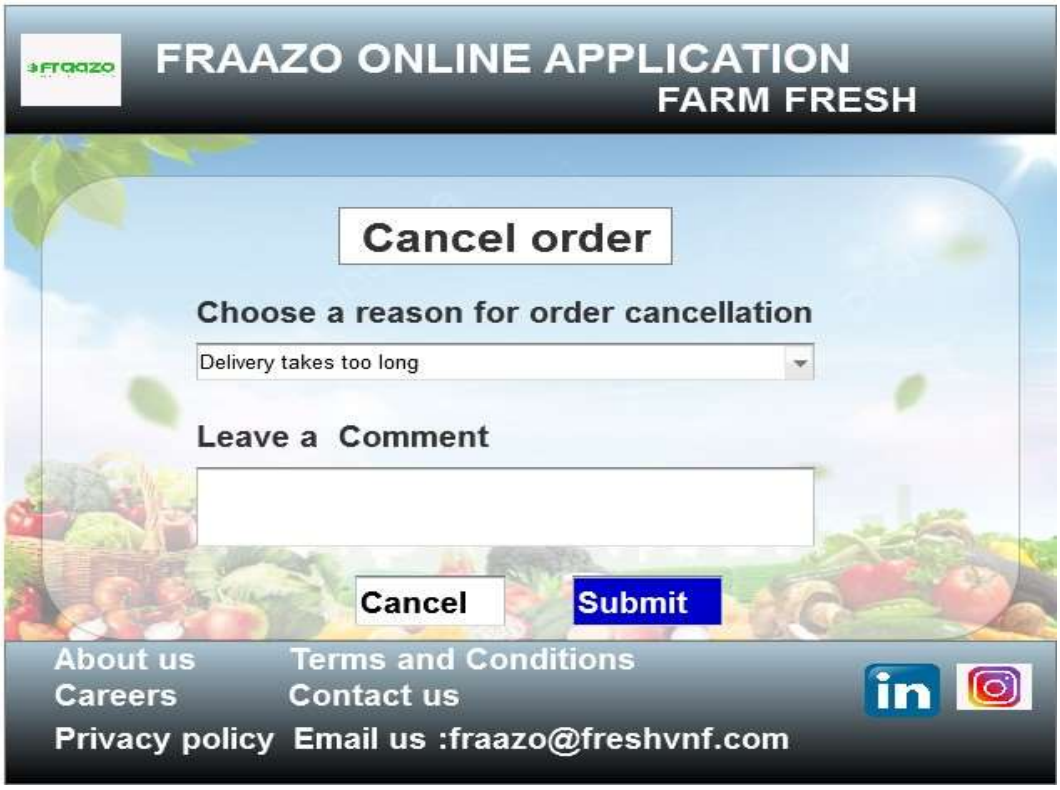
2.Registration page:

3.Login page:

4.Add to cart page:

5.Make a payment page:

6.Cancel order page:



Document 8- Tools-

Visio and Axure Write a paragraph on your experience using Visio and Axure for the project.

Microsoft Visio: A Detailed Overview

Microsoft **Visio** is a professional diagramming and vector graphics application that helps users create a wide range of visual representations, such as flowcharts, organizational charts, process diagrams, network diagrams, and engineering schematics. It is widely used in **business, IT, engineering, project management, and software development** to map out complex systems and workflows in an easy-to-understand manner.

Visio is part of the **Microsoft Office family** but is sold separately. It is available in both **desktop and web versions**, with cloud-based collaboration features in Microsoft 365.

Key Features of Microsoft Visio

As a Business Analyst, MS Visio is one of the tools I often use to create diagrams that help explain business processes, system flows, and relationships in a clear and visual way.

Using MS Visio is quite helpful because:

1. **Easy to Create Diagrams**:
   Visio allows me to drag and drop shapes and symbols to quickly build flowcharts, use case diagrams, and process maps.
2. **Helps Explain Complex Ideas**:
   Sometimes, explaining a process or a system in words can be confusing. Visio helps me turn those ideas into diagrams so that both technical and non-technical people can understand easily.
3. **Supports Different Diagram Types**:
   I can create many types of diagrams like:
   - Flowcharts
   - Use Case Diagrams
   - Data Flow Diagrams (DFD)
   - Organizational Charts
   - Swimlane Diagrams
4. **Collaboration Made Simple**:
   often work with developers, testers, and stakeholders. With Visio, I can share diagrams that clearly show what needs to be built or improved.
5. **Professional and Neat**:
   The diagrams look clean and professional, which is great for presentations and documents.
6. **Time-Saving Templates**:
   Visio has built-in templates, so I don't have to start from scratch every time.

What is Axure RP?

Axure RP is a **powerful UX/UI design and prototyping tool** used to create **wireframes, interactive prototypes, and documentation** for websites, apps, and software systems. Unlike simple wireframing tools, Axure allows designers to add **dynamic interactions, conditional logic, and animations** without coding.

It is widely used by **UX designers, product managers, developers, and business analysts** to design user experiences, test workflows, and communicate ideas effectively with stakeholders.

As a **Business Analyst (BA)**, using **Axure** has been a game-changer for creating wireframes and prototypes. Here's my experience in simple terms:

1. Easy to Create Wireframes

Axure makes it easy to **draw screens and layouts** for websites or apps. You don't need coding skills—just drag and drop elements like buttons, text boxes, and images.

2. Interactive Prototypes

Unlike basic tools, Axure allows me to create **clickable prototypes**. This helps stakeholders see how the system will work instead of just looking at static images.

3. Clear Communication with Teams

Instead of long documents, I show my wireframes to **developers, designers, and stakeholders**. It reduces confusion because everyone can see the flow visually.

4. Saves Time & Reduces Rework

Since I can test the user journey early, we **catch problems** before development starts. This avoids costly fixes later.

5. Easy Documentation & Annotations

Axure allows me to add **notes** to elements, making it easy to explain logic, requirements, and interactions without separate documents.

Overall Experience

Axure is an excellent tool for **BAs** because it helps **visualize ideas, gather feedback, and improve user experience** before actual development begins. It makes my job **faster, clearer, and more efficient**.

Document 9- BA experience My experience as BA in following phases:

1. **Requirement gathering:**

My Experience as a Business Analyst in Requirement Gathering

As a Business Analyst, I played a crucial role in gathering and validating requirements to ensure the project's success. Here's how I handled different aspects of this phase:

1. **Requirement Prioritization Using MoSCoW**
    - To effectively gather and categorize requirements, I used the **MoSCoW** technique (Must-have, Should-have, Could-have, Won't-have). This helped me prioritize requirements based on business needs and feasibility, ensuring that critical features were identified early.
2. **Managing Client Unavailability**
    - During the requirement-gathering phase, the client was unavailable for some time. To avoid delays, I took the initiative to identify and engage alternative points of contact from the client's side. I ensured that I collected the necessary information as quickly as possible to keep the project on track.

3. **Validating Requirements Using FURPS**
    - I applied the **FURPS** model (Functionality, Usability, Reliability, Performance, and Supportability) to validate requirements. This ensured that all gathered requirements were clear, feasible, and aligned with business goals.
4. **Eliminating Duplicate and Redundant Requirements**
    - While reviewing requirements, I noticed many duplicates and repetitive requests. I took the necessary steps to eliminate them immediately, ensuring a clean and efficient requirement list. This helped streamline development efforts and reduced ambiguity.
5. **Using Prototyping for Clarity**
    - To refine and clarify requirements, I utilized **prototyping** techniques. This approach helped stakeholders visualize features and functionalities, leading to more specific and detailed requirements. By providing wireframes and mockups, I minimized misinterpretations and ensured alignment between business and technical teams.
6. **Creating the BRD:**
    - I compiled all approved requirements into a Business Requirements Document (BRD), which served as a reference point for all stakeholders throughout the project.

Through these methods, I effectively managed requirement gathering, ensuring that the project had a solid foundation for the subsequent phases.

2. Requirement Analysis:

In the **Requirement Analysis** phase, my role was to break down and refine gathered requirements to ensure they were well-structured, clear, and aligned with business goals. My key activities included:

1. **Creating UML Diagrams for Visual Representation**
    - I utilized **UML diagrams** to visually describe the system's requirements, ensuring a structured representation of system behavior and interactions. These diagrams helped the team understand the system architecture effectively.
2. **Using Activity Diagrams for Process Flow**
    - To illustrate business workflows, I created **Activity Diagrams** that depicted the step-by-step flow of processes. These diagrams helped in identifying dependencies, bottlenecks, and potential improvements in the system’s design.
3. **Collaborating with the Team for Diagram Validation**
    - After preparing the diagrams, I communicated them to the development and testing teams for validation.
4. **Preparing Business Requirement Specification (BRS) and Software Requirement Specification (SRS)**

- I documented all business needs in the **Business Requirement Specification (BRS)** to ensure alignment with stakeholder expectations.
- The **Business Requirement Specification (BRS)** is a high-level document that outlines the **business needs, goals, and expectations** for the project
- The **Software Requirement Specification (SRS)** is a comprehensive document that translates the business requirements into detailed **functional and non-functional specifications** for the development and testing teams.

Through this structured approach, I ensured that the requirements were thoroughly analyzed, documented, and validated before moving into the design and development phases.

3. Design:

During the **Design Phase**, I played a key role in ensuring that the proposed system design met all business and functional requirements. My primary responsibilities included:

1. **Deriving Test Cases from Use Case Diagrams**
   - I analyzed **Use Case Diagrams** to derive comprehensive test cases that covered various system functionalities and user interactions. This ensured that all scenarios were well-documented for the testing phase.
2. **Communicating Design and Solution Documents with Clients**
   - I actively engaged to review and validate the **design and solution documents**, ensuring that the proposed system aligned with their business expectations. Feedback from the client was incorporated into the design to avoid any gaps.
3. **Writing Both Positive and Negative Test Cases**
   - I prepared a detailed set of test cases, including:
     - **Positive Test Cases** to verify expected system behavior.
     - **Negative Test Cases** to test how the system handles invalid or unexpected inputs.
   - This approach helped identify potential issues early in the development cycle.
4. **Ensuring No Test Cases Are Missed**
   - I emphasized **completeness** in test case preparation, as missing even a single test case could lead to critical defects in later development stages. I followed a structured approach to cover all possible scenarios.
5. **Preparing Test Data for Testing**
   - I created and maintained relevant **test data** required for validating different test scenarios. This included both valid and invalid data sets to ensure comprehensive testing coverage.
6. **Updating the Requirements Traceability Matrix (RTM)**
   - I consistently updated the **Requirements Traceability Matrix (RTM)** to track each requirement against corresponding test cases. This ensured that all business requirements were accounted for and met through the testing process.

4. Development:

During the **Development Phase**, my role as a BA was to facilitate seamless communication between the technical team and stakeholders while ensuring the development aligned with business requirements. My key contributions included:

1. **Organizing Joint Application Development (JAD) Sessions**
    - I conducted **JAD sessions** to bring together business stakeholders and the development team to discuss and finalize requirements, design, and implementation details.
    - These sessions helped bridge the gap between business needs and technical execution, reducing misinterpretations.
2. **Clarifying Queries from the Tech Team During Coding**
    - As development progressed, I actively supported the technical team by **resolving any requirement-related queries**.
    - I referred to requirement documents, diagrams, and past discussions to provide precise clarifications, ensuring that development stayed on track.
3. **Handling Team Conflicts and Resistance in JAD Sessions**
    - Some team members had differing opinions or were hesitant to cooperate during JAD sessions.
    - As a BA, I **managed these situations diplomatically** by having **one-on-one discussions** with concerned members. I explained how their collaboration impacted the project's success and encouraged a **positive, solution-oriented environment** within the team.
4. **Referring Diagrams to Code the Unit**
    - I ensured that **UML diagrams, process flows, and system designs** were used effectively as references during coding.
    - This helped developers stay aligned with the agreed-upon structure and logic.
5. **Conducting Regular Meetings with the Tech Team and Client**
    - Regular meetings with the development team and the client were crucial for progress tracking and issue resolution.
    - However, some team members were occasionally unavailable. To **ensure no one missed important updates**, I:
        - Recorded the meetings and shared them with absent members.
        - Followed up with **one-on-one discussions** to address their concerns and keep them aligned with the project's progress.

By implementing these strategies, I helped maintain **clear communication, resolve challenges proactively, and support the development team in building a system that met business needs.**

5. Testing:

In the **Testing Phase**, my role as a BA was to ensure that the system was thoroughly validated, aligned with business requirements, and ready for client approval. My key responsibilities included:

1. **Preparing Test Cases from Use Cases**
    - Based on the **Use Case Diagrams**, I derived **detailed test cases** covering functional and non-functional requirements.
    - Ensuring comprehensive test coverage helped identify defects early and improved the overall quality of the system.
2. **Performing High-Level Testing**
    - I conducted **high-level testing** to validate system behavior before it moved to dedicated testing teams.
    - This included **sanity checks, workflow validation, and business rule verification** to ensure major functionalities were working as expected.
3. **Requesting Test Data from Clients**
    - For **realistic test scenarios**, I coordinated with the client to obtain the necessary **test data** (e.g., sample records, configurations, or real-world inputs).
    - This ensured that testing closely mimicked actual business scenarios, making results more reliable.
4. **Updating the Requirements Traceability Matrix (RTM)**
    - I continuously updated the **RTM** to ensure that every requirement was mapped to corresponding test cases and validated during testing.
    - This helped in **requirement coverage analysis** and ensured no business requirement was left untested.
5. **Taking Sign-Off from the Client**
    - Once the testing phase was successfully completed, I facilitated the **sign-off process** from the client.
    - This involved **demonstrating key functionalities**, addressing any last-minute concerns, and ensuring stakeholder approval before proceeding to deployment.
6. **Preparing the Client for User Acceptance Testing (UAT)**
    - I assisted the client in preparing for **UAT** by:
        - Providing a **step-by-step walkthrough** of the system.
        - Creating and sharing **UAT scenarios** for structured testing.
        - Addressing **any queries or clarifications** to ensure smooth execution.

By meticulously handling these tasks, I ensured that the system was **validated, approved, and ready for deployment** with minimal defects and maximum stakeholder confidence.

7. **Deployment:**

In the **Deployment Phase**, my role as a BA was to ensure a smooth transition of the system to end-users by handling documentation, training, and coordination. My key responsibilities included:

1. **Forwarding the Requirements Traceability Matrix (RTM) to the Client**
    - I ensured that the **RTM was finalized and sent to the client**, demonstrating that all requirements were successfully implemented and tested.
    - The RTM was **attached to the project closure document**, serving as proof that all business needs were met.
2. **Coordinating and Sharing End-User Manuals**
    - I worked with the documentation team to **create and distribute end-user manuals** to guide users in system usage.
    - Ensured that the manuals were **clear, concise, and user-friendly**, addressing common queries and processes.
3. **Planning and Organizing Training Sessions**
    - To help end-users adapt to the new system, I planned **training sessions** tailored to different user roles.
    - The training covered system navigation, key functionalities, and troubleshooting common issues.
4. **Ensuring Full Attendance in Training Sessions**
    - I made sure that **all relevant users attended the training** by:
        - Sending calendar invites and reminders.
        - Following up with absent participants for **one-on-one training** if needed.
        - Recording sessions and sharing them for future reference.

By efficiently managing these tasks, I helped ensure a **seamless deployment, well-trained users, and a structured project closure process.**