COLLEGE CODE:6107

COLLEGE NAME: GCE, BARGUR

DEPARTMENT: CSE

STUDENT NM-ID: 5a22913b2d187cc863d4b453d67408fd

ROLL NO:2303610710422047

DATE:17.09.2025

Completed the project named as Phase 2

TECHNOLOGY PROJECT NAME :

IBM-NJ-E-COMMERCE CART SYSTEM

SUBMITTED BY,

NAME: SHERIN KERENHAP.S
MOBILE NO:9487494054

1.Tech Stack Selection

When selecting a tech stack for an e-commerce cart system, you need to balance scalability, performance, security, and flexibility. Here's a breakdown of the layers and possible choices.

1. Frontend (User Interface)

Responsible for product browsing, cart management, checkout, and user experience.

Languages: HTML5, CSS3, JavaScript/TypeScript

Frameworks/Libraries:

- React.js (component-based, fast, SEO-friendly with Next.js)
- Angular (enterprise-level, structured)
- Vue.js (lightweight, flexible)

2. Backend (Business Logic & API Layer)

Handles cart operations, user sessions, inventory, and payment integrations.

Languages & Frameworks:

- Node.js (Express / NestJS) – event-driven, scalable
- Java (Spring Boot) – robust, enterprise-grade
- Python (Django / FastAPI) – quick development, good integrations

3. Database (Storage Layer)

- For storing products, users, orders, and cart sessions.
- Relational (SQL): PostgreSQL, MySQL (good for structured data like orders, users)
- NoSQL: MongoDB, DynamoDB (great for flexible product catalogs, high scalability)

4. Payment & Transaction Handling

- Gateways: Stripe, PayPal, Razorpay, Braintree
- Security: PCI-DSS compliance, HTTPS, tokenization

5. Authentication & Security

- JWT, OAuth 2.0, or Firebase Auth

- Encryption with TLS/SSL
- Role-based access control (admin vs customer)

6. Infrastructure & Deployment

- Web Servers: Nginx / Apache
- Hosting / Cloud: AWS, GCP, Azure, DigitalOcean
- Containerization: Docker, Kubernetes (for scaling microservices)

2.UI Structure/API Scheme Design

The UI should be simple, clean, and focused on the shopping flow.

1. Login / Register Page

Fields: Email, Password, Confirm Password (for register)

Buttons: Login, Register

After success → redirect to Home / Products Page

2. Home / Products Page

Header: Logo, Search Bar, Cart Icon (with count), Profile

Product Grid/List:

- Product Image
- Name
- Price
- Stock Availability
- Add to Cart button

3. Cart Page

Shows items added to the cart:

Product Image + Name

Quantity selector (+ / –)

Price per item × quantity

Remove button

4. Checkout Page

Sections:

Delivery Address Form (Name, Address, City, Pincode, Phone)

Payment Simulation (Card/UPI dummy fields)

Button: Place Order

5. Order Confirmation Page

Message: Order placed successfully

Show Order ID, summary of items, total price

3.Data Handling Approach

The following data handling approaches are:

1. Types of Data Involved
   Understanding what data is involved helps determine storage, retrieval, and processing strategies.

Data Category	Examples
User Data	User ID, session ID, authentication token
Product Data	Product ID, name, price, availability, attributes
Cart Data	Cart ID, list of items (product ID, quantity),

2. Data Model Design

The cart data model needs to support both registered and guest users, and should be flexible, scalable, and atomic.

Example Schema (Simplified)

```
Cart {
 cart_id: UUID,
 user_id: UUID or null (for guest),
 items: [
  {
   product_id: UUID,
   quantity: integer,
   price_at_addition: float
  }
 ],
 created_at: timestamp,
 updated_at: timestamp,
 is_active: boolean
}
```

3. Storage Options

Relational DB (PostgreSQL / MySQL)

Good for structured data.

Supports ACID transactions (good for consistency).

Suitable for storing persistent carts.

NoSQL (Redis / MongoDB)

Hybrid Approach

4. Security Considerations

Use HTTPS to prevent data interception.

Validate and sanitize all cart input (e.g., quantity, product ID).

Use token/session-based authentication for user carts.

Ensure price is not taken from client — verify on server side.

5. Concurrency & Consistency

Lock product inventory updates using optimistic locking or transactions.

Use atomic operations in Redis to prevent race conditions when updating cart data.

Confirm price and stock at checkout to avoid inconsistencies.

6. Performance Optimization

Cache product data (like price, name) to reduce DB calls during cart operations.

Use lazy loading or asynchronous processing for less critical data.

Expire inactive carts using TTL in Redis.

Avoid saving too much metadata in the cart object.

7. Events & Logging

Emit events for cart updates (add/remove item) for:

Analytics

Abandoned cart emails

Inventory sync

8. API Design (RESTful or GraphQL)

Typical endpoints:

Endpoint	Action
GET /cart	Fetch current cart
POST /cart/add	Add item to cart
POST /cart/remove	Remove item from cart
POST /cart/clear	Empty the cart
POST /cart/merge	Merge guest + user cart
POST /cart/apply-coupon	Apply discount

9.Testing & Validation

Unit tests for cart logic (add/remove/update).

Integration tests with inventory and checkout services.

Load testing to validate performance under high traffic.

10. Scalability & Future Proofing

Use sharding or partitioning for large-scale cart storage.

Separate microservices for cart, product, inventory, and checkout.

4.Component/Module Diagram

Components of an E-commerce Cart System

1. User Interface (UI/Frontend)

Product browsing

Search & filters

Add/Remove items from cart

Checkout & payment page

Order tracking

2. Cart Management Module

   Add/Update/Delete cart items

   Manage product quantity

   Apply coupons or discounts

   Store cart session (logged-in / guest users)

3. Product Catalog Module

   Product listing & categories

   Product details (images, price, stock, description)

   Inventory sync with cart/orders

4. User Management Module

   User registration & login

   Profile & address management

   Authentication & authorization

5. Order Management Module

   Place order from cart

   Order history & tracking

6. Payment Module

   Payment gateway integration (Stripe, Razorpay, PayPal)

   Secure transactions (PCI-DSS, SSL/TLS)

   Refunds & cancellations

7. Inventory & Stock Module

Stock availability check when adding to cart

Auto stock reduction after order confirmation

5.Basic Flow Diagram

1. User Browses Products

   View product catalog

   Selects a product

2. Add to Cart

   User clicks "Add to Cart"

   System updates cart with product info & quantity

3. View / Edit Cart

   User can see cart contents

   Modify quantity or remove items

4. Proceed to Checkout

   Login/Register (if not logged in)

   Provide shipping info

5. Payment

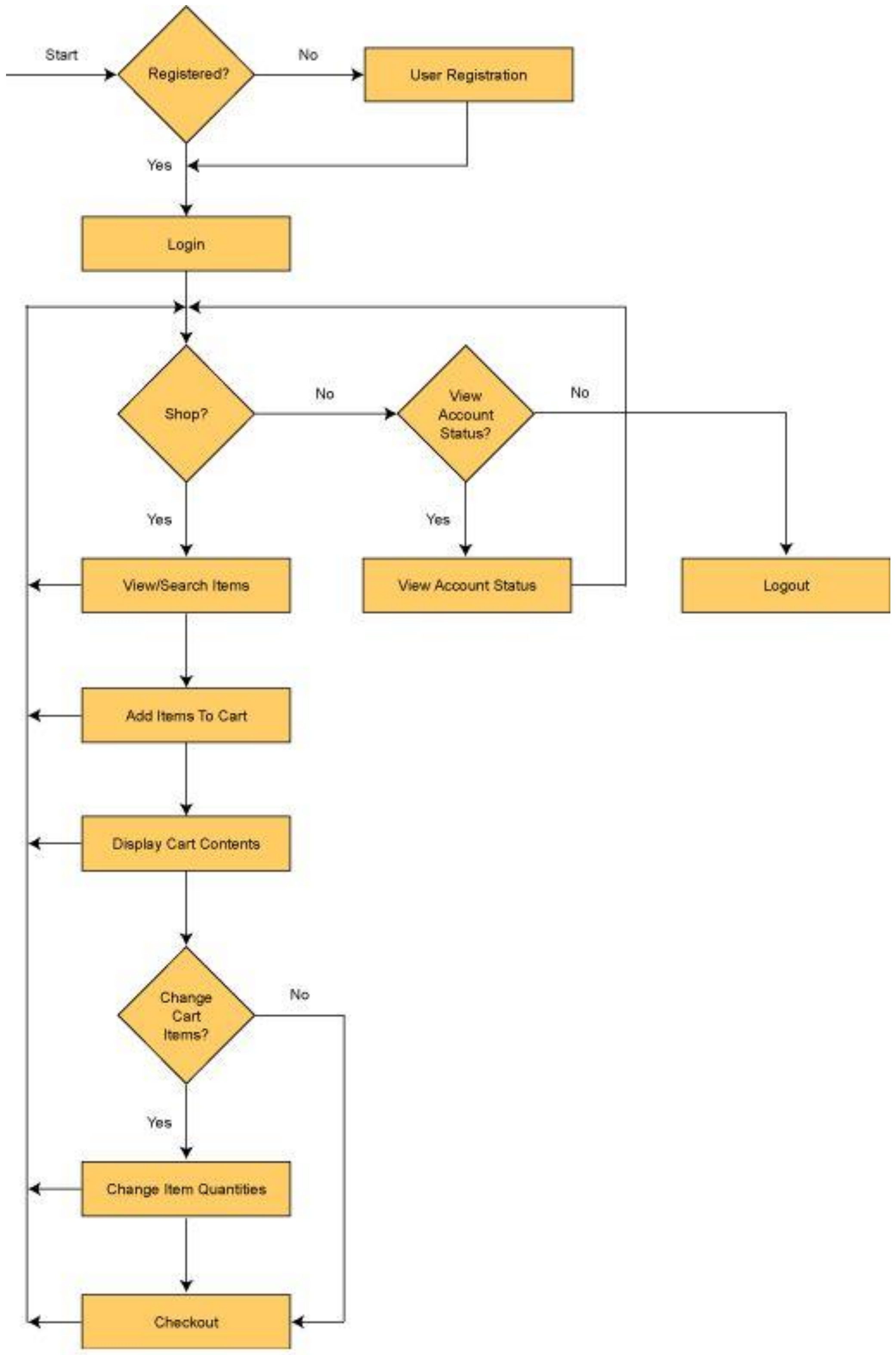
   Choose payment method

   Confirm payment

6. Order Confirmation

   System verifies payment

   Creates order


Start
Registered?
No
User Registration
Yes
Login
Shop?
No
View Account Status?
No
Yes
Yes
View/Search Items
View Account Status
Logout
Add Items To Cart
Display Cart Contents
Change Cart Items?
No
Yes
Change Item Quantities
Checkout