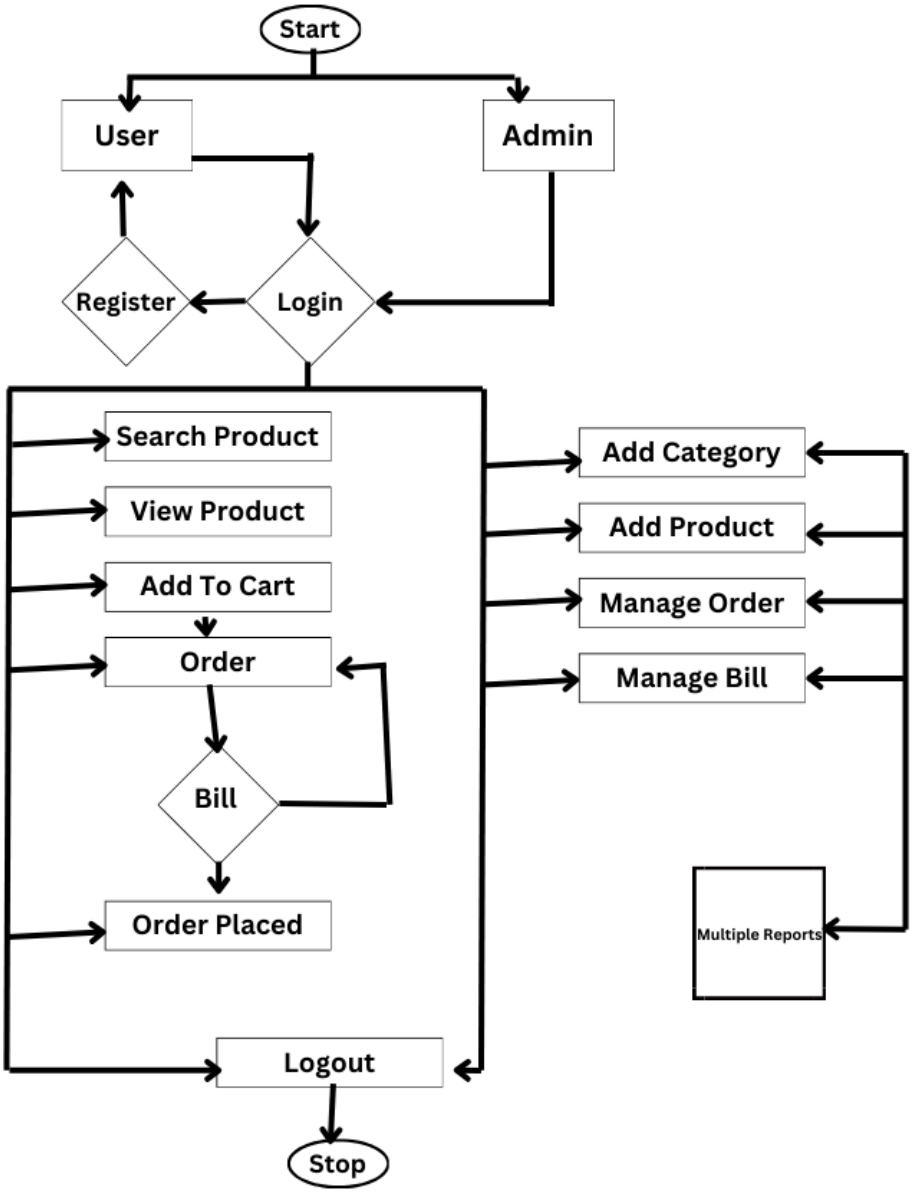
DAY 2 : Hackathon MarketPlace Ecommerce

Technical Foundation of Ecommerce



FrontEnd (Next js):

{ Name,
Product listening,
Product Details,
Cart,
Checkout,
Order Confirmation,
Shipment Tracking,
}

Backend:

{ Product Data,
Customer Data,
Order Data,
Shipment information,
Sanity CMS : { "To manage product data and orders information and shipment details". }

Third - party Api’s

Frontend	Backend	APIs	Tools
Next.js For building modern, dynamic, and server rendered Uis.	**Sanity CMS** To manage and structure content effectively.	**ShipEngine** Simplifies shipment tracking and delivery.	**GitHub** Version Control and collaborative development
Tailwind For responsive and visually appealing designs.	**Clerk** For seamless user authentication and management.	**Stripe** Handles secure and efficient online payments.	**Postman** For testing and documenting APIs efficiently
ShadcnUI For pre-designed, customize components.			**GitHub** Fast and reliable deployment for Next.js applications.

{ Payment integration : using stripe Shipment Tracking : using ship engine, shippoar integration }

Details:

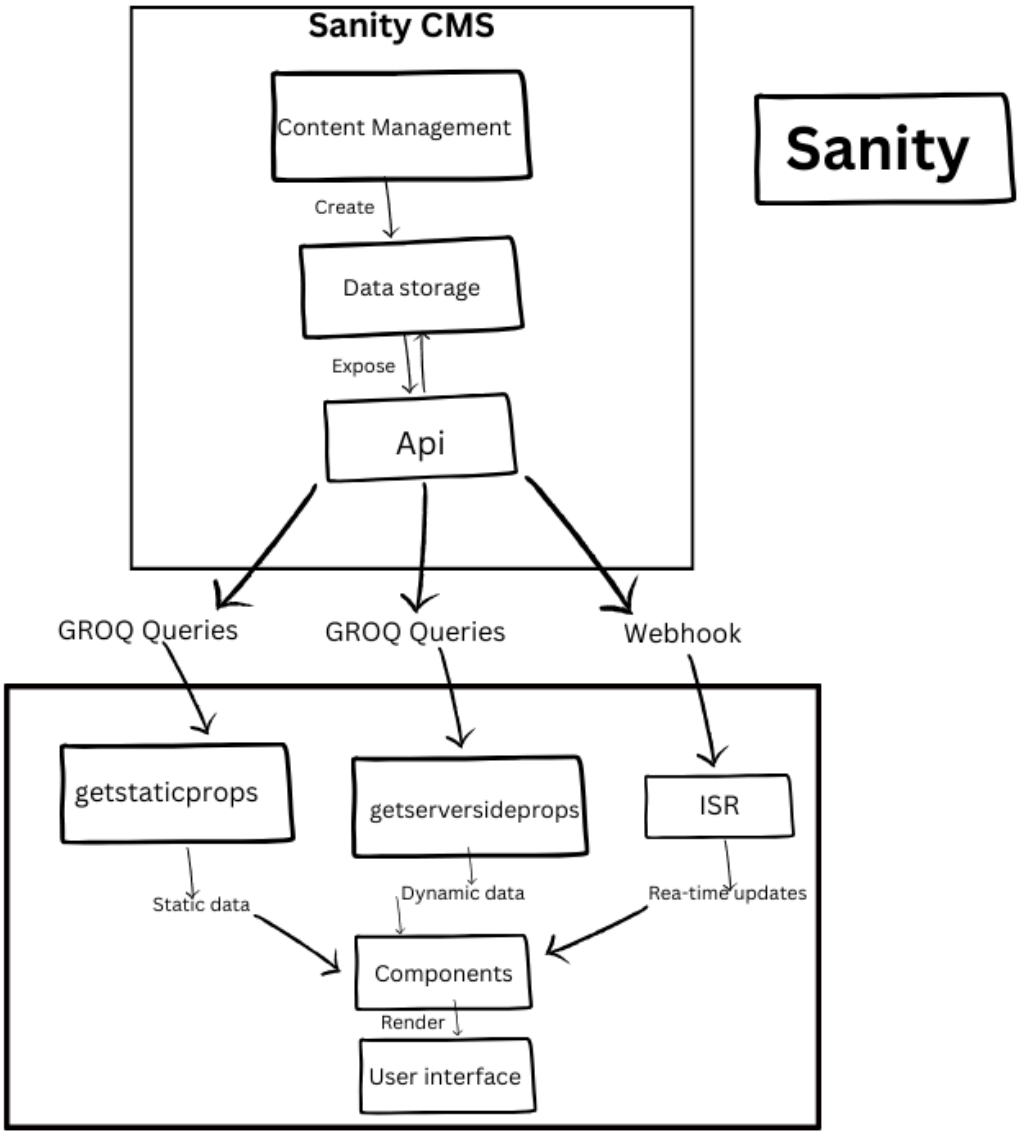
Frontend:

- Build with Next js with responsive design and responsive layouts and pages for all type of devices.
- Pages: {

Home page (with static route)
About page (with static route)

Product listening and product details (with dynamic route)
Cart page (with dynamic route)
Checkout page (with shipment Tracking integration)
404 page
Blog page (with static route)
Login and sign up page
Payment method page
}

Backend:(Sanity CMS)


Sanity CMS
Content Management
Create
Data storage
Expose
Api
Sanity
GROQ Queries
GROQ Queries
Webhook
getstaticprops
getserversideprops
ISR
Static data
Dynamic data
Rea-time updates
Components
Render
User interface

{

Product data management :{
I'd, name, price, tags, available,description and etc that other thing that I want. }

Customer Data management :{
Name, Address, Contact information, Phone number }

Order Data management :{
Items, payment, Total Amount, delivery status, shipment Status }

}

Third-party Apis:
{

Product data management :{
I'd, name, price, tags, available,description and etc that other thing that I want. }

Customer Data management :{

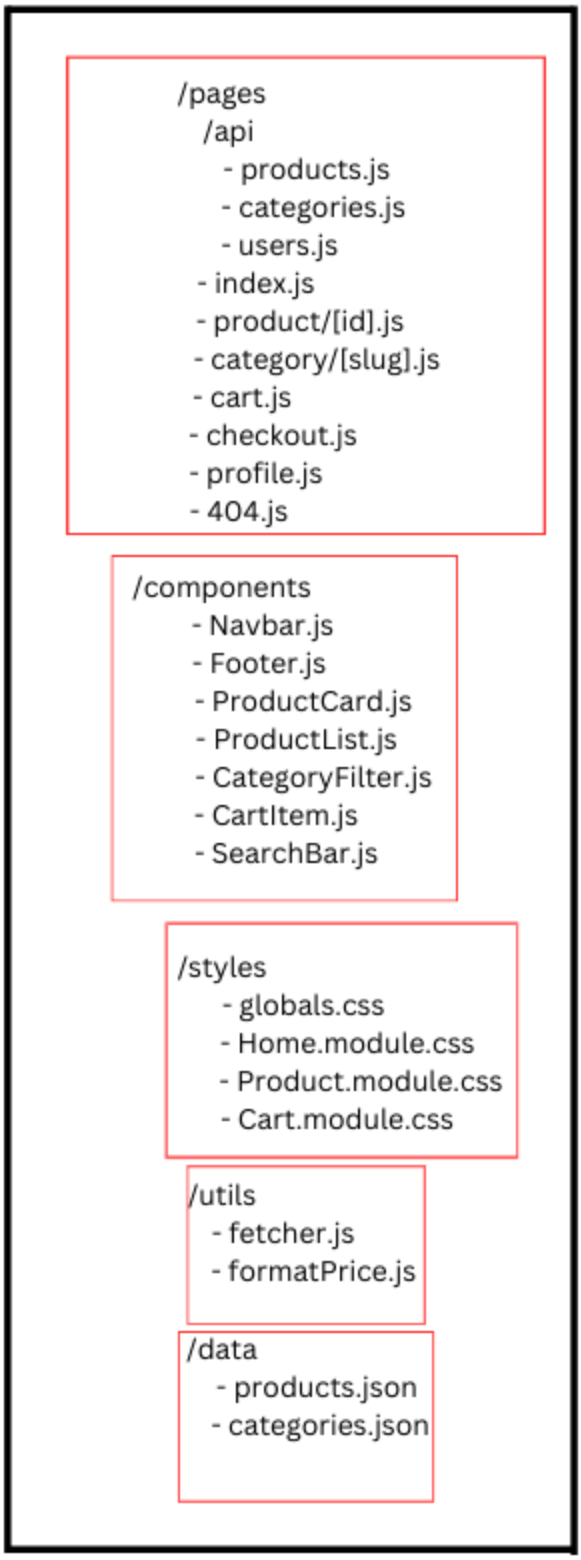
Name, Address, Contact information, Phone number
}

Order Data management :{
Items, payment, Total Amount, delivery status, shipment Status }

}

Sanity Schema:

project structure:



Product Data (products.json)


```
[
 {
  "id": 1,
  "name": "Modern Sofa",
  "price": 499.99,
  "description": "Comfortable and stylish modern sofa.",
  "image": "/images/sofa.jpg",
  "category": "Living Room",
  "stock": 10
 },
 {
  "id": 2,
  "name": "Wooden Dining Table",
  "price": 799.99,
  "description": "Elegant wooden dining table for 6.",
  "image": "/images/dining-table.jpg",
  "category": "Dining Room",
  "stock": 5
 }
]
```

Category Data (categories.json)


```
[
  {
    "id": 1,
    "name": "Living Room",
    "slug": "living-room"
  },
  {
    "id": 2,
    "name": "Dining Room",
    "slug": "dining-room"
  }
]
```

User Data (Example API Response)


```
{
 "id": 1,
 "name": "John Doe",
 "email": "john@example.com",
 "orders": [
  {
   "orderId": "12345",
   "date": "2025-01-15",
   "total": 1299.99
  }
 ]
}
```


{

Product schema :{ Name
Description
Price
Tags
Image
Available items

}}

EndPoint APIs:

Endpoint	Method	What it does
/api/rentals/create	POST	Creates a new rental listing
/api/rentals	GET	Fetches all available rental listings
/api/rentals/[rentalId]	GET	Fetches details of a specific rental
/api/rentals/[rentalId]	DELETE	Deletes a specific rental listing (admin)
/api/rentals/book	POST	Books a rental for a user
/api/rentals/user-bookings	GET	Fetches all bookings made by a user
/api/payments/create	POST	Initiates payment for a rental
/api/payments/verify	GET	Verifies the status of a payment
/api/send/confirmation-email	POST	Sends a confirmation email for a booking
/api/reviews/[rentalId]	GET	Fetches reviews for a specific rental