HACKATHON 3 (DAY 2)

TECHNICAL ANALYSIS

E-COMMERCE WEBSITE

Step 1:

- User Flow Overview This diagram outlines the user journey:

[Homepage] → [Signup/Login] → [Product Browsing] [Add to Cart] [Checkout] [Order Confirmation] [Shipment and Tracking] →

- **Homepage:** User lands on the homepage.
- **Signup/Login:** User signs up or logs in.

- **Product Browsing:** User views mobile covers by category or search.
- **Add to Cart:** User adds a product to their cart.
- **Checkout:** User completes payment.
- **Order Confirmation:** Backend processes the order and updates status.
- **Shipment & Tracking:** After order processing, the user can track the status and location of the shipment.

Step 2:

➢ **APIs**:

- **Authentication API**: Handles signup/login.
- **Product API**: Fetches mobile cover data from Sanity.

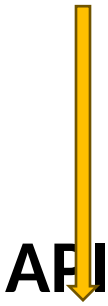
- **Cart API**: Adds/removes items in the cart.
- **Order API**: Manages order details (creation, status updates).
- **Payment API:** Processes payments (e.g. Stripe/PayPal).
- **Shipment API**: Integrates with a shipping service (e.g., UPS or FedEx) to provide tracking information.

➢ **Schemas:**

- **User Schema:** Stores user details (email, password, etc.).
- **Product Schema:** Stores product data (title, description, price, image).
- **Order Schema:** Tracks orders (user ID, product IDs, payment status).

- **Cart Schema:** Temporary storage for cart items.
- **Shipment Schema**: Stores shipment tracking information and delivery status.

Frontend (Next.js)



API Gateway (Express/Node.js)

Authentication API (JWT)

Product API (Sanity)

Cart API (MongoDB)

Order API (MongoDB)

Payment API (Stripe/PayPal)

→ **|Shipment Updates|**

ShipmentAPI[Shipment API (Tracking Service)]

Step 3:

- **Sanity Integration**

  Sanity will be used as a headless CMS to manage products:

Sanity Studio (Admin) → Product Schema → Sanity APIs

↓

Backend → Frontend

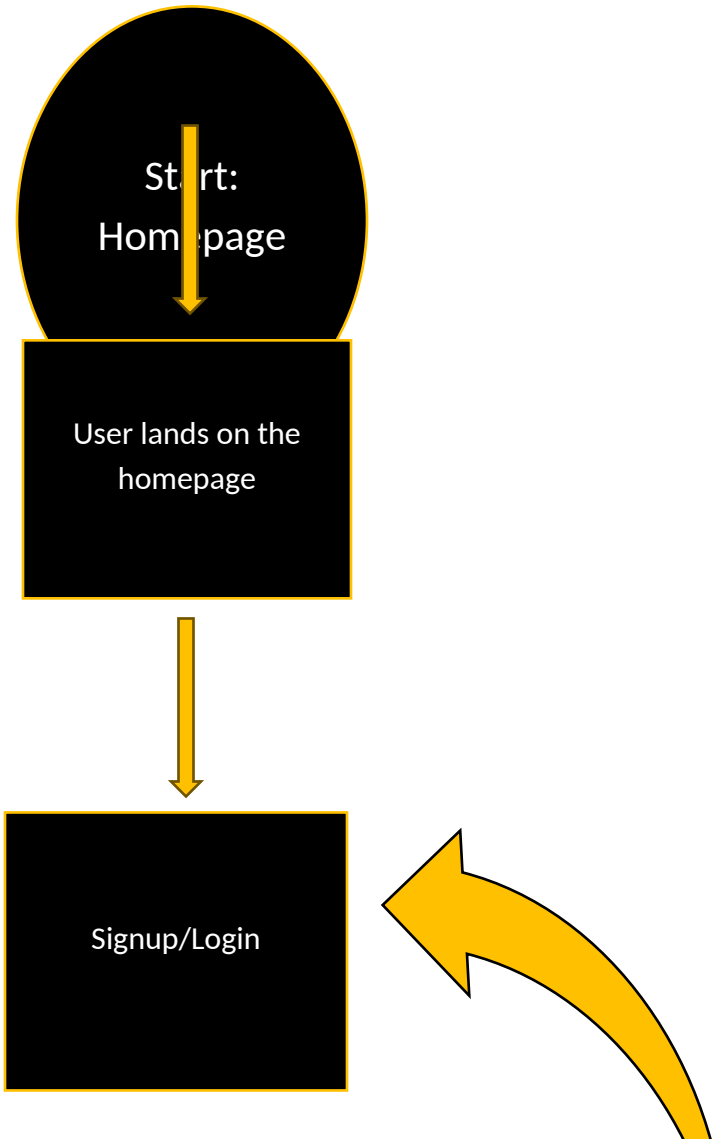
- **Sanity Studio:** Admins add/edit/delete mobile cover products.
- **Backend Fetch:** Products are fetched dynamically using Sanity APIs.

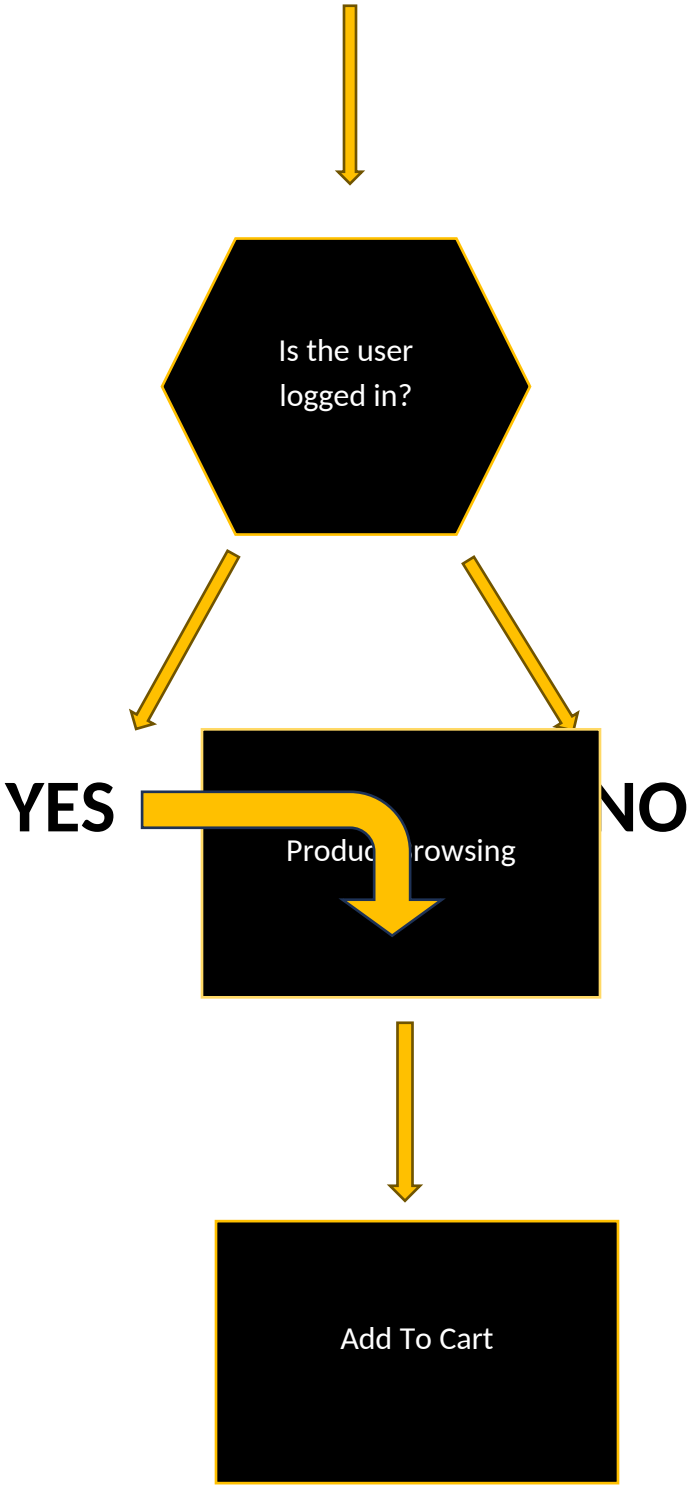
- **Frontend Display:** The website displays products retrieved from Sanity in real time.

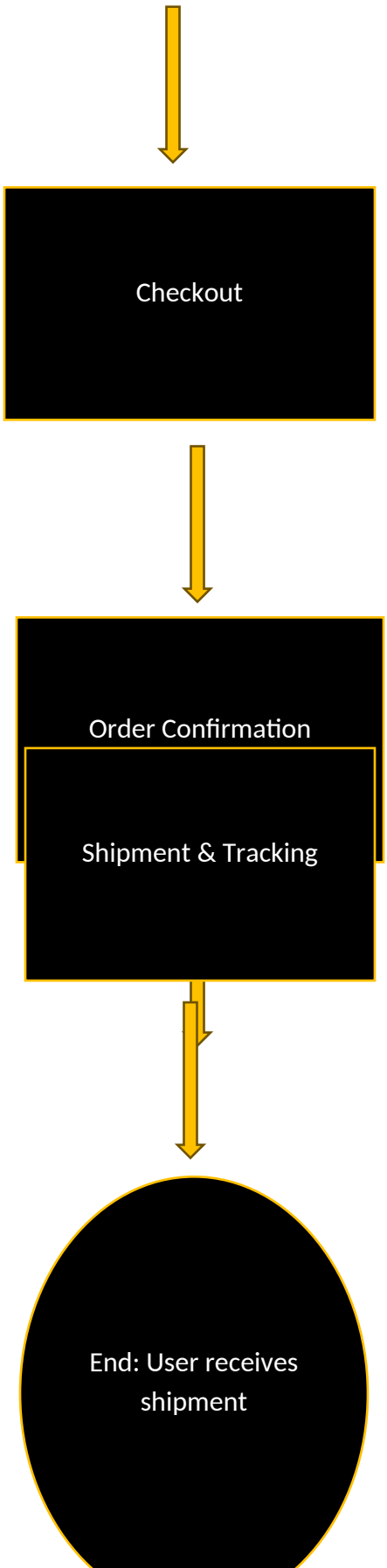
Step 4:

➢ Technical Flow

This diagram visualizes data flow:




Is the user
logged in?
YES
NO
Add To Cart


Checkout
Order Confirmation
Shipment & Tracking
End: User receives shipment

- **Frontend (Next.js):** User interacts with the UI.
- **API Gateway:** Sends/receives data between frontend and backend.
- **Backend (Node.js/Express):**
  - Authenticates users.
  - Fetches data from Sanity.
  - Stores orders in a database (MongoDB, PostgreSQL, etc.).

- **Sanity**: Serves product data.
- **Payment Processor**: Handles payment transactions securely.

(Stripe/PayPal)

- **Shipping Service**: Provides shipment tracking and delivery information.

  (Leopard/TCS/FedEx)

➢ API ENDPOINTS

1. Authentication API

- **POST /api/auth/signup**
  - **Description:** Register a new user.
  - **Request Body:** { "name": "string", "email": "string", "password": "string" }
- **POST /api/auth/login**
  - **Description:** Log in an existing user.

- **Request Body:** { "email": "string", "password": "string" }

- **GET /api/auth/logout**

  - **Description:** Log out the current user.

- **GET /api/auth/user**

  - **Description:** Get the current logged-in user's information.

  - **Response:** { "id": "string", "name": "string", "email": "string" }

2. Product API

- **GET /api/products**

- **Description:** Fetch all products or filter by category/attributes.

- **Query**
  Parameters: ?category=string&search=string

- **GET /api/products/:id**

  - **Description:** Fetch details of a specific product by ID.

- **POST /api/products**

  - **Description:** (Admin) Add a new product.

- **Request Body:** { "title": "string", "description": "string", "price": "number", "image": "string" }

- **PUT /api/products/:id**

  - **Description**: (Admin) Update an existing produt.

  - **Request Body:** { "title": "string", "description": "string", "price": "number", "image": "string" }

- **DELETE /api/products/:id**

  - **Description:** (Admin) Delete a product.

3. Cart API

- **GET /api/cart**
    - **Description:** Retrieve the current user's cart.
- **POST /api/cart**
    - **Description:** Add an item to the cart.
    - **Request Body**: { "productId": "string", "quantity": "number" }
- **PUT /api/cart/:id**
    - **Description:** Update the quantity of a specific item in the cart.
    - **Request Body:** { "quantity": "number" }

- **DELETE /api/cart/:id**
    - **Description:** Remove an item from the cart.

4. Order API

- **POST /api/orders**
    - **Description:** Create a new order.
    - **Request Body:** { "cartItems": [{ "productId": "string", "quantity": "number" }], "shippingDetails": { "address": "string", "city": "string", "zip": "string" }, "paymentMethod": "string" }

- **GET /api/orders/:id**
    - **Description:** Retrieve details of a specific order by ID.
- **GET /api/orders**
    - **Description:** Retrieve all orders for the logged-in user.

5. Payment API

- **POST /api/payments**
    - **Description:** Process a payment.
    - **Request Body:** { "orderId": "string", "paymentMethod": "string", "amount": "number" }

- **GET /api/payments/:id**

    - **Description:** Retrieve payment status for a specific order.

6. Shipment API

- **GET /api/shipments/:orderId**

    - **Description**: Retrieve shipment tracking information for a specific order.

- **POST /api/shipments**

    - **Description:** Create a shipment record after an order is processed.

- **Request Body**: { "orderId": "string", "trackingNumber": "string", "carrier": "string" }