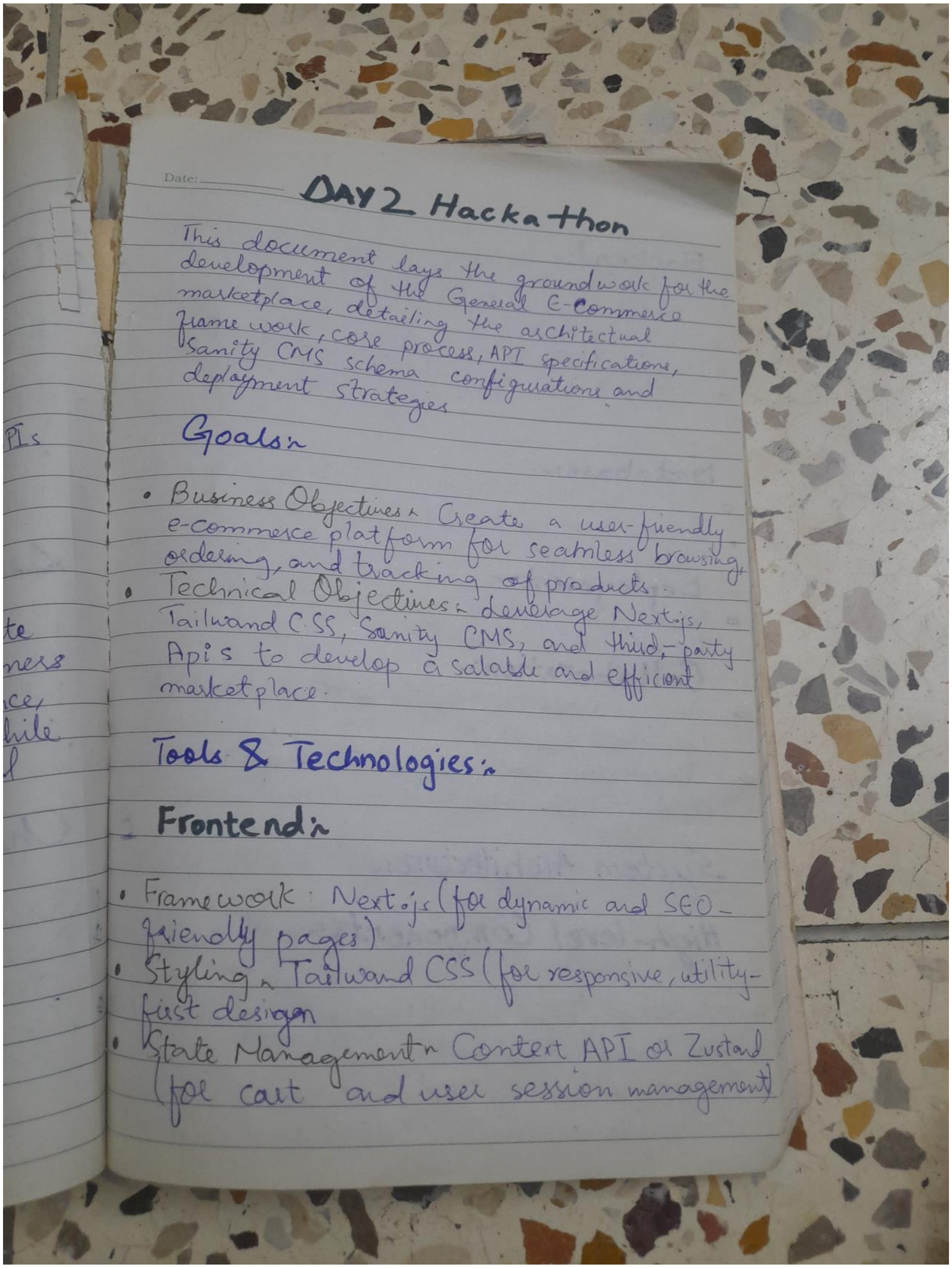
DAY 2



Date:

DAY 2 Hackathon

This document lays the groundwork for the development of the General E-Commerce marketplace, detailing the architectual frame work, core process, API specifications, Sanity CMS schema configurations and deployment strategies

Goals:

- Business Objectives: Create a user-friendly e-commerce platform for seamless browsing, ordering, and tracking of products.
- Technical Objectives: leverage Next.js, Tailwand CSS, Sanity CMS, and third-party Apis to develop a salable and efficient marketplace.

Tools & Technologies:

Frontend:

- Framework: Next.js (for dynamic and SEO-friendly pages)
- Styling: Tailwand CSS (for responsive, utility-first desigan
- State Management: Context API or Zustand (for cart and user session management)

Date: ________

Backend:-

- CMS:- Sanity CMS (to manage product, customer, and order data)
- Third-Party APIs:
- Payment Gateway:- Stripe (for secure transcation)
- Shipment Tracking:- EasyPost API (for tracking order status)

Database:-

- Managed by Sanity CMS with schema definitions for products, orders, and users.

Deployment:-

- Platform:- Vercel (for deploying the Next.js Application)

Collaboration:-

- Version Control:- Github (to manage project code).
- Diagram Tools:- excalidraw (for architecture and workflow design).

System Architecture:-

High-level Components:-

- Frontend (Next.js):- Handles user interaction and communicates with APIS.
- Backend (Sanity CMS):- Stores and retrieves data for products, orders, and customers-

roduct,

n secure transcation)
(for tracking
)

schema
and

the

nage

l architecture

interaction

d retrieves
stomers-

Date:

- Third-Party APIs:- Manages payments and Shipment tracking

Dataflow:-

1) User requests product data from the frontend
2) The frontend fetches product information from Sanity CMS.
3) User places an order, data is sent to Sanity CMS and payment is processed via Stripe
4) Shipment tracking information is retrieved from EasyPOST APT and displayed to the user.

Architecture Diagram:-

```
[Frontend (Next.js)] <--> [Sanity CMS]
          |                     |
[Payment Gateway (Stripe)]  [Shipment APT (EasyPost)]
```

Key Workflows:-

1 User Registration:-

1 User signs up on the platform.
2 Data is stored in Sanity CMS.
3 A Confirmation email is sent to the user.

Date:

2 Product Browsing:-

1 User view product categories and details
2 Sanity CMS provides product data via API
3 Products are displayed dynamically on the frontend

3 Order Placement:-

1 User adds items to the cart and proceeds to checkout.
2 Order details are sent to Sanity CMS.
3 Payment is processed via Stripe.
4 Confirmation is sent to the user and stored in the database.

4 Shipment Tracking:-

1 Order status is updated in Sanity CMS.
2 EasyPost API fetches real-time tracking details
3 Tracking status is displayed on the user dashboard.

API Requirements:-

Endpoints:-

1 Fetch Products

- Endpoint: /api/products.
- Method: GET
- Description:- Fetch all available products
- Response Example.

Date:

[
{
"id
"na
"pr
"st
"im
}
]

2 Cre

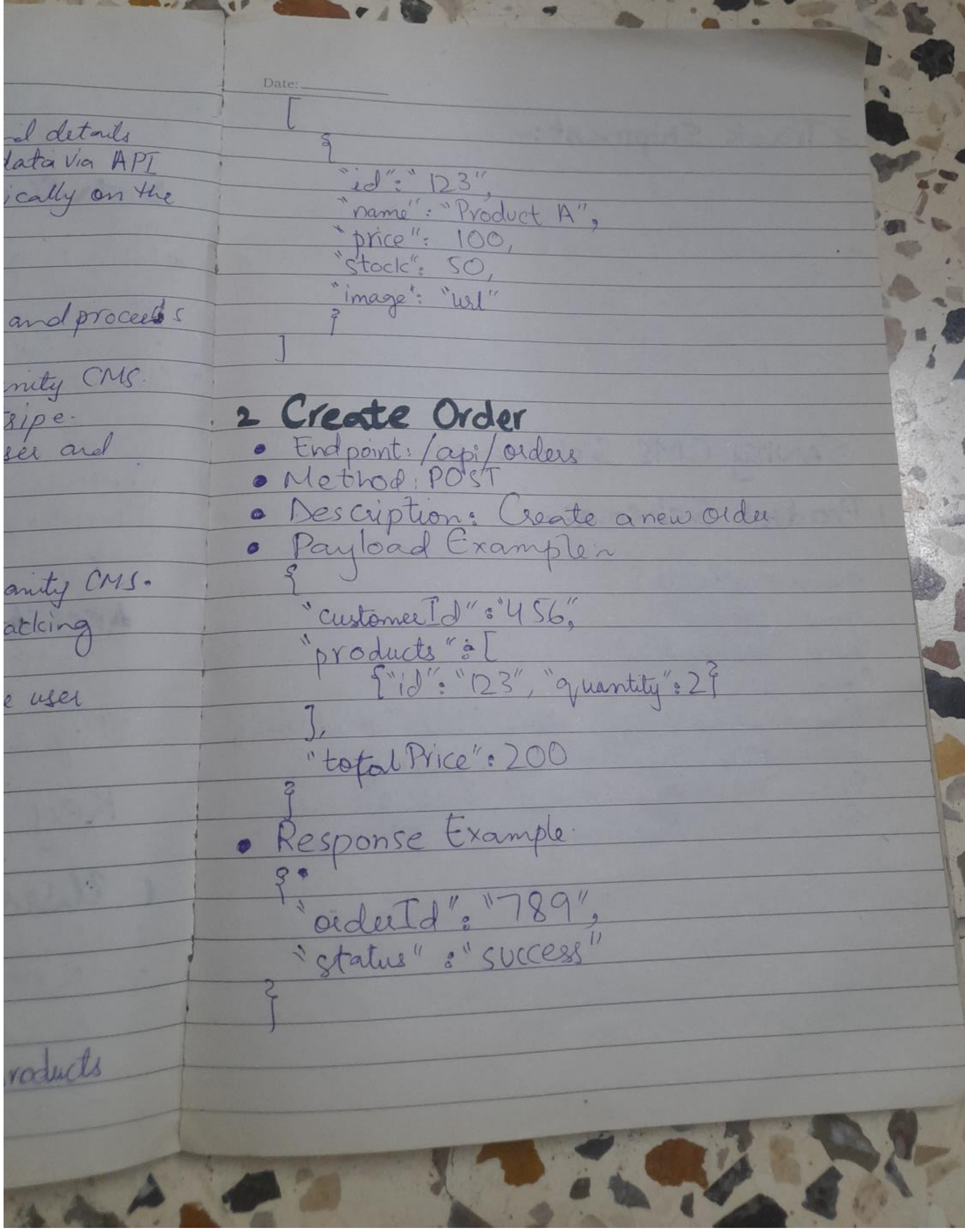
- End
- Met
- Des
- Pa

{
"c
"p
],
"
}
}

- Re

{
"

}
}



d details
data via API
ically on the

and proceeds

nity CMS.
ripe.
ser and

anity CMS.
acking

e user

roducts

Date:

```
[
  {
    "id": "123",
    "name": "Product A",
    "price": 100,
    "stock": 50,
    "image": "url"
  }
]
```

2 Create Order

- Endpoint: /api/orders
- Method: POST
- Description: Create a new order
- Payload Example~

```
{
  "customerId": "456",
  "products": [
      {"id": "123", "quantity": 2}
  ],
  "totalPrice": 200
}
```

- Response Example

```
{
  "orderId": "789",
  "status": "success"
}
```

Date:

3 Track Shipment:-

- Endpoint: /api/shipments
- Method: GET
- Description: Fetch shipment status
- Response Example

```
{
 "orderId": "789"
 "status": "In Transit",
 "ETA": "2025-01-20"
}
```

SANity CMS Schemas:-

1 Product Schema:-

```
export default {
name: 'product',
type: 'document',
fields: [
 { name: 'name', type: 'string', title: 'Product Name' },
 {name: 'price', type: 'number', title: 'Price' },
 {name: 'stock', type: 'number', title: 'Stock Level' }
 {name: 'image', type: 'image', title: 'Product Image' }
 ]
};
```

Date:

2 Ord

```
expo
nam
type
fiel
 {na
 {na
  [{
 {na
 {na
]
};
```

De

1) F
2) B
 S
3) (

t status

oduct Name'},
rice'},
ock Level'},
t Image'}

Date:

2 Order Schema:-

```
export default{
 name:'order',
 type: 'document',
fields: [
 {name : 'customerId', type:'string', title:'CustomerID'},
 {name: 'products', type:'assay', of:[{type:'reference', to:
  [{type: 'product'}]}]},
 {name: 'totalPrice', type: 'number', title: 'Total Price'},
 {name: 'status', type: 'string', title: 'Order Status'}
]
};
```

Deployment Strategy:-

1) Frontend:- Deploy on Vercel.
2) Backend:- Sanity CMS hosted on Sanity Studio
3) Enviroment Variable:- Configure API keys for stripe and EasyPost securely

Conclusion:-

This document ensures the technical foundation aligns with the business goals of creating a scalable General E-Commerce platform.