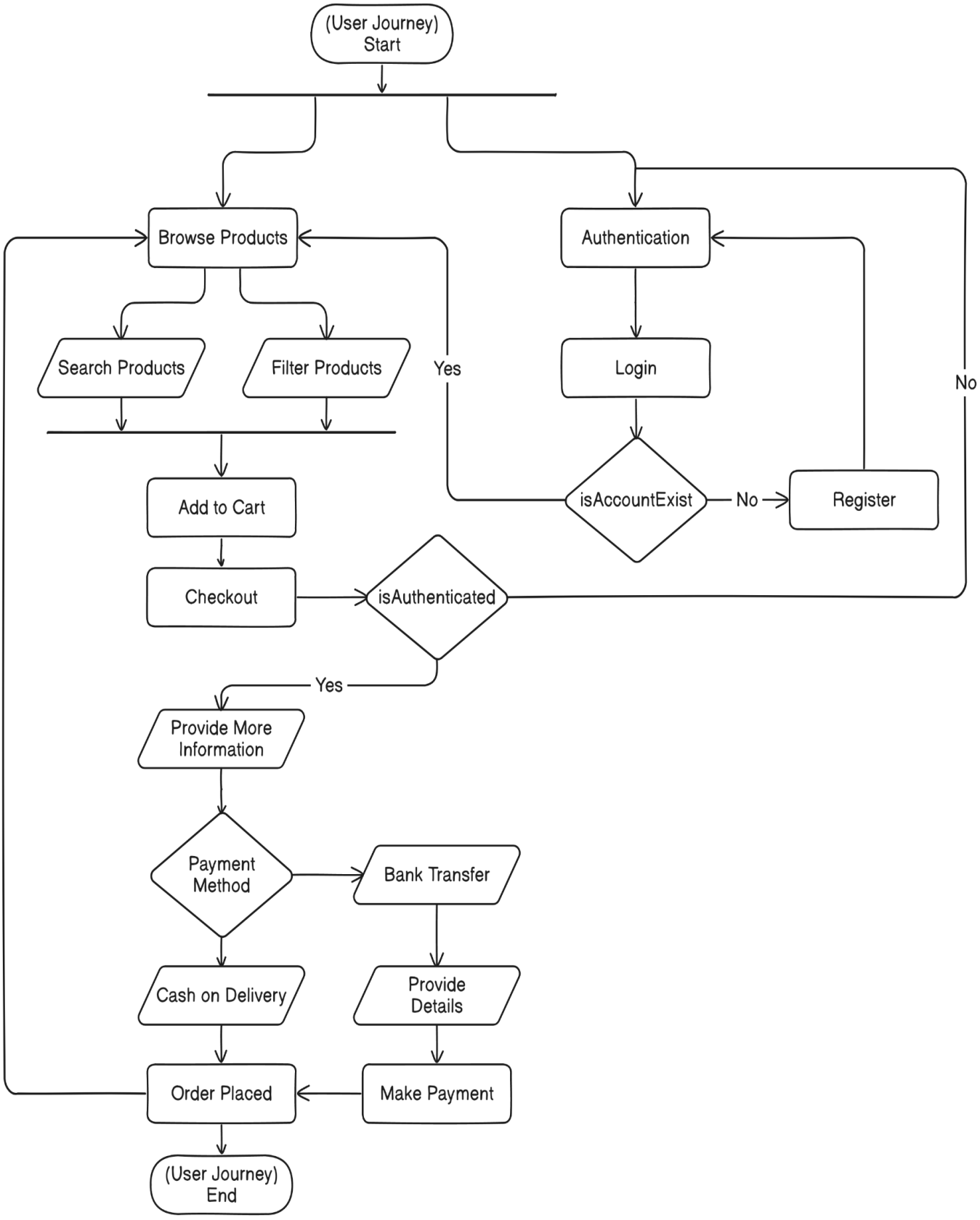
Marketplace Technical Foundation - Hiperstar

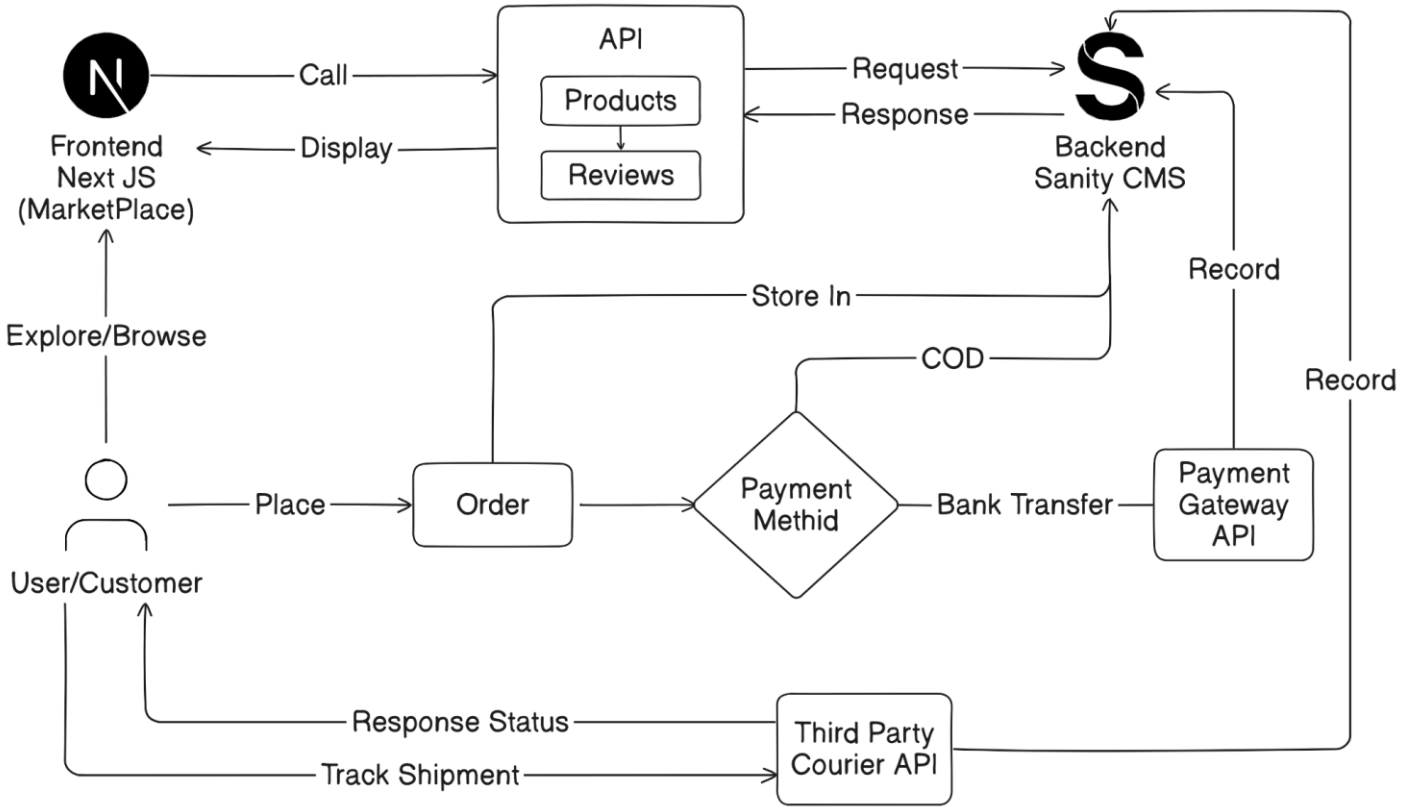
MARKETPLACE BUILDER HACKATHON 2025

DAY 2: PLANNING THE TECHNICAL FOUNDATION - Hiperstar

Key User Workflow



System Architecture Overview



API Endpoints Table

Endpoint Name: /products
Method: GET
Description: Fetch all available products from Sanity.
Payload: -
Response: {

product_id: 1,
name: "Product Name",
category_id: 1,
description: "Product Description",
product_image:"https://encrypted-tbn0.gstatic.com/imagesq=tbn:ANd9GcRmCy16nhIbV3pI1qLYHMJKwbH2458oiC9EmA&s",
price: 2600,
stock: 10,
rating: 4.5,
added_date:"20-1-2025 12:08:57 PM"

}

Endpoint Name: /order
Method: POST
Description: Create a new order in Sanity.
Payload: {

Customer_id:1,

Total_price:3000,
order_note:"Order Note",
order_date:"20-1-2025 9:43:50 PM",
Order_status:"pending"
}
Response: {
Status:"success",
message:"Order Created Successfully"
}

Endpoint Name: /track-shipment:order_id
Method: GET
Description: Track order status via third-party API.
Payload: -
Response: {
Shipment_id:1,
Order_id:1,
status:"Arrived at facility",
expected_delivery_date:"25-1-2025 4:50:00 PM"
}

Endpoint Name: /create-review
Method: POST
Description: Create a new review of specific product.
Payload: {
Product_id:1,
Customer_id:1,
Rating:4.5,
comment:"Good Product"
}
Response: {
status:"Success",
message:"Review created successfully"
}

Endpoint Name: /reviews
Method: GET
Description: Fetch reviews of specific products.
Payload: -
Response: [
{
Review_id:1,
Product_id:1,
Customer_id:1,
Rating:4.5,
comment:"Good Product",

review_date:"20-1-2025 10:20:00 PM"
}
]

Endpoint Name: /payment
Method: GET
Description: Process online payment
Payload: -
Response: {
Status:"completed".
message:"Transaction completed"
}

Endpoint Name: /customers
Method: GET
Description: Fetch all registered customers
Payload: -
Response: [
{
Customer_id:1,
firstname:"Muhammad",
lastname:"Anus",
Username:"anusmemon226",
email:"anusm226@gmail.com",
Phone:"03302626644",
country:"Pakistan",
city:"Karachi",
Postal_code:74800,
address:"XYZ near famous, Karachi",
account_creation_date:"20-1-2025 9:45:00 PM"
}
]

Endpoint Name: /single-customer:id
Method: GET
Description: Fetch single customer
Payload: -
Response:{
Customer_id:1,
firstname:"Muhammad",
lastname:"Anus",
Username:"anusmemon226",
email:"anusm226@gmail.com",
Phone:"03302626644",
country:"Pakistan",
city:"Karachi",

Postal_code:74800,
address:"XYZ near famous, Karachi",
account_creation_date:"20-1-2025 9:45:00 PM"
}

Endpoint Name: /create-customer
Method: POST
Description: Register a new customer
Payload:{
Customer_id:1,
firstname:"Muhammad",
lastname:"Anus",
Username:"anusmemon226",
email:"anusm226@gmail.com",
Phone:"03302626644",
country:"Pakistan",
city:"Karachi",
Postal_code:74800,
address:"XYZ near famous, Karachi",
account_creation_date:"20-1-2025 9:45:00 PM"
}

Response: {
Status:"success",
message:"User registered successfully"
}

Sanity Schema

1. Customer Schema

```
import { defineField, defineType } from 'sanity'
export const customerType = defineType({
  name: 'customer',
  title: 'Customer',
  type: 'document',
  fields: [
    defineField({
      name: 'firstname',
      type: 'string',
      placeholder: "Enter Firstname",
      validation: (rule) => rule.required(),
    }),
    defineField({
      name: 'lastname',
      type: 'string',
      placeholder: "Enter Lastname",
      validation: (rule) => rule.required(),
```

```
}),
defineField({
   name: "username",
   type: "string",
   placeholder: "Enter Username",
   validation: (rule) => rule.required()
}),
defineField({
   name: "email",
   type: "email",
   placeholder: "Enter Email",
   validation: (rule) => rule.required()
}),
defineField({
   name: "phone_number",
   type: "string",
   placeholder: "Enter Phone Number",
   validation: (rule) => rule.required()
}),
defineField({
   name: "country",
   type: "reference",
   to: [{
      type: "country"
   }]
}),
defineField({
   name: "city",
   type: "reference",
   to: [{
      type: "city",
   }],
}),
defineField({
   name: "postal_code",
   type: "string",
   placeholder: "Enter Postal Code",
   validation: (rule) => rule.required()
}),
defineField({
   name: "address",
   type: "string",
   placeholder: "Enter Address",
   validation: (rule) => rule.required()
}),
```

```
      defineField({
        name: 'account_creation_date',
        type: 'datetime',
        initialValue: () => new Date().toISOString(),
        validation: (rule) => rule.required(),
      }),
    ],
})
```

2. Product Schema:

```
import { defineField, defineType } from 'sanity'
export const productType = defineType({
  name: 'product',
  title: 'Product',
  type: 'document',
  fields: [
    defineField({
      name: 'product_name',
      type: 'string',
      validation: (rule) => rule.required(),
    }),
    defineField({
      name: 'slug',
      type: 'slug',
      options: { source: 'product_name' },
      validation: (rule) => rule.required(),
    }),
    defineField({
      description: "Select Category",
      name: 'category',
      type: 'reference',
      to: [
        {
          type: "category"
        }],
      validation: (Rule) => Rule.required()
    }),
    defineField({
      name: "variation_details",
      type: "array",
      of: [
        {
          type: "object",
          fields: [
            { name: "variation_name", type: "string" },
            {
```

```
                    name: "variation_options",
                    type: "array",
                    of: [
                      { type: "string" }
                    ],
                    validation: (Rule:any) =>
                      Rule.custom((options: string[] | undefined, context: { parent: { variation_name?:
string } }) => {
                        // Check if variation_name is provided
                        const parent = context.parent; // Access the parent object
                        if (parent.variation_name && (!options || options.length === 0)) {
                          return "Variation options cannot be empty when variation name is provided";
                        }
                        return true; // Valid if no variation_name or options are present
                      }),
                  }
                ]
              }
            ]
          }),
          defineField({
            name: "main_image",
            type: "image",
            validation: (rule) => rule.required()
          }),
          defineField({
            name: 'product_images',
            type: 'array',
            of: [
              {
                type: "image"
              }
            ],
            validation: (rule) => rule.required()
          }),
          defineField({
            name: 'description',
            type: 'array',
            of: [{ type: 'block' }],
            validation: (rule) => rule.required()
          }),
          defineField({
            name: 'price',
            type: 'string',
            validation: (rule) => rule.required()
```

```
    }),
    defineField({
      name: "stock",
      type: "number",
      validation: (rule) => rule.required()
    }),
    defineField({
      name: "rating",
      type: "number",
      initialValue: 0,
      validation: (rule) => rule.min(0).max(5).required()
    }),
    defineField({
      name: 'added_at',
      type: 'datetime',
      initialValue: () => new Date().toISOString(),
      validation: (rule) => rule.required(),
    }),

  ],
})
```

3. Category Schema:

```
import { defineField, defineType } from 'sanity'
export const categoryType = defineType({
  name: 'category',
  title: 'Category',
  type: 'document',
  fields: [
    defineField({
      name: 'category_name',
      type: 'string',
      validation: (rule) => rule.required(),
    }),
    defineField({
      name: 'slug',
      type: 'slug',
      options: { source: 'category_name' },
      validation: (rule) => rule.required(),
    }),
    defineField({
      name: 'category_image',
      type: 'image',
    }),
    defineField({
      name: 'description',
```

```
      type: 'array',
      of: [{ type: 'block' }],
    }),
    defineField({
      name: 'added_at',
      type: 'datetime',
      initialValue: () => new Date().toISOString(),
      validation: (rule) => rule.required(),
    }),
  ],
})
```

4. **Country Schema:**

```
import { defineField, defineType } from 'sanity'
export const countryType = defineType({
  name: 'country',
  title: 'Country',
  type: 'document',
  fields: [
    defineField({
      name: 'country_name',
      type: 'string',
      validation: (rule) => rule.required(),
    }),
    defineField({
      name: 'slug',
      type: 'slug',
      options: { source: 'country_name' },
      validation: (rule) => rule.required(),
    }),
  ],
})
```

5. **City Schema:**

```
import { defineField, defineType } from 'sanity'
export const cityType = defineType({
  name: 'city',
  title: 'City',
  type: 'document',
  fields: [
    defineField({
      name: 'city_name',
      type: 'string',
      validation: (rule) => rule.required(),
    }),
    defineField({
      name: "country",
```

```
      type: "reference",
      to: [{
        type: "country"
      }]
    }),
    defineField({
      name: 'slug',
      type: 'slug',
      options: { source: 'city_name' },
      validation: (rule) => rule.required(),
    }),
  ],
})
```

6. Review Schema:

```
import { defineField, defineType } from 'sanity'
export const reviewType = defineType({
  name: 'review',
  title: 'Review',
  type: 'document',
  fields: [
    defineField({
      name: 'product',
      type: 'reference',
      to: {
        type: "product"
      },
      validation: (rule) => rule.required(),
    }),
    defineField({
      name: 'customer',
      type: 'reference',
      to: { type: 'customer' },
      validation: (rule) => rule.required(),
    }),
    defineField({
      name: 'rating',
      type: 'number',
      validation: (rule) => rule.min(1).max(5).required(),
    }),
    defineField({
      name: 'comment',
      type: 'text',
      validation: (rule) => rule.required(),
    }),
    defineField({
```

```
      name: 'review_date',
      type: 'datetime',
      initialValue: () => new Date().toISOString(),
      validation: (rule) => rule.required(),
    }),
  ],
})
```

7. Order Schema:

```
import { defineField, defineType } from 'sanity'
export const orderType = defineType({
  name: 'order',
  title: 'Order',
  type: 'document',
  fields: [
    defineField({
      name: 'customer',
      type: 'reference',
      to: {
        type: "customer"
      },
      validation: (rule) => rule.required(),
    }),
    defineField({
      name: "order_items",
      type: "array",
      of: [
        {
          type: "object",
          fields: [
            {
              name: "product",
              type: "reference",
              to: [{ type: "product" }],
              validation: (rule) => rule.required()
            },
            {
              name: "quantity",
              type: "number",
              validation: (rule) => rule.min(1).required()
            },
            {
              name: "price",
              type: "string",
              validation: (rule) => rule.required()
            }
```

```
            ]
          }
        ]
      }),
      defineField({
        name: 'total_price',
        type: 'string',
        validation: (rule) => rule.required(),
      }),
      defineField({
        name: "order_note",
        type: "text",
      }),
      defineField({
        name: "order_status",
        type: "string",
        options: {
          list: [
            { title: "Pending", value: "pending" },
            { title: "Fulfilled", value: "fulfilled" },
            { title: "Completed", value: "completed" },
            { title: "Cancelled", value: "cancelled" }
          ]
        },
        validation: (rule) => rule.required()
      }),
      defineField({
        name: "order_date",
        type: "datetime",
        initialValue: () => new Date().toISOString(),
        validation: (rule) => rule.required(),
      })
    ],
})
```

8. Payment Schema:

```
import { defineField, defineType } from 'sanity'
export const paymentType = defineType({
  name: 'payment',
  title: 'Payment',
  type: 'document',
  fields: [
    defineField({
      name: 'order',
      type: 'reference',
      to: {
```

```
            type: "order"
          },
          validation: (rule) => rule.required(),
        }),
        defineField({
          name: 'payment_method',
          type: 'string',
          options: {
            list: [
              {title:"Cash On Delivery",value:"COD"},
              {title:"Bank Transfer",value:"bank_transfer"}
            ]
          },
          validation: (rule) => rule.required(),
        }),
        defineField({
          name: "amount",
          type: "string",
          placeholder: "Enter Transaction Amount",
          validation: (rule) => rule.required()
        }),
        defineField({
          name: "payment_status",
          type: "string",
          options: {
            list:[
              {title:"Pending",value:"pending"},
              {title:"Failed",value:"failed"},
              {title:"Completed",value:"completed"}
            ]
          }
        }),
        defineField({
          name: 'transaction_date',
          type: 'datetime',
          initialValue: () => new Date().toISOString(),
          validation: (rule) => rule.required(),
        }),
    ],
})
```