Day 3 - API Integration Report - [Furniro]"

Prepared by: ZOHAIB MEMON

1. API Integration Process

1. **Understanding API Requirements**:
   - Analyzed the API documentation for authentication, endpoints, and payload structures.
2. **Setup**:
   - Configured the environment by installing necessary dependencies (e.g., Fetch for API calls).
   - Established environment variables for API keys and sensitive credentials.
3. **API Endpoints**:
   - Integrated multiple endpoints for fetching and posting data.
4. **Error Handling**:
   - Added robust error handling mechanisms to manage failed calls and unexpected responses.
5. **Testing**:
   - Used tools like Postman and Thunder Client to validate requests and responses before integrating them into the codebase.
6. **Final Integration**:
   - Implemented API calls within the respective functions, ensuring seamless interaction between frontend and backend.

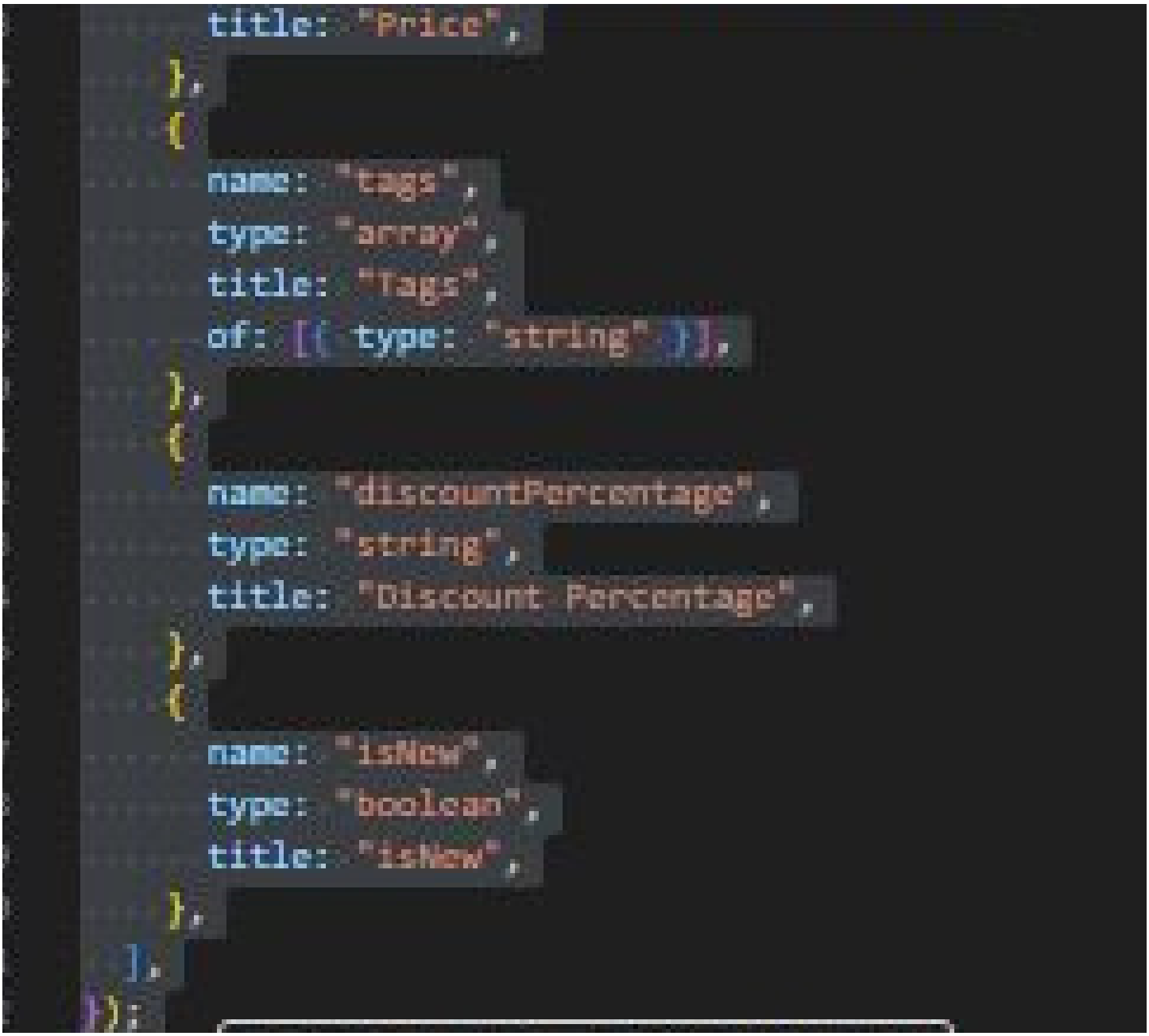
2. Adjustments Made to Schemas

Updated Sanity CMS Schema:

- **Product Schema:**
  - Added fields for title, description, price, imageUrl, tags, isNew, and discountPercentage.
  - Ensured validation rules for mandatory fields.

```
import { defineType } from "sanity";

export const product = defineType({
  name: "product",
  title: "Product",
  type: "document",
  fields: [
    {
      name: "title",
      title: "Title",
      validation: (rule) => rule.required(),
      type: "string",
    },
    {
      name: "description",
      type: "text",
      validation: (rule) => rule.required(),
      title: "Description",
    },
    {
      name: "productImage",
      type: "image",
      title: "Product Image",
      validation: (rule) => rule.required(),
      options: {
        hotspot: true,
      },
    },
    {
      name: "price",
      type: "number",
      validation: (rule) => rule.required(),
```



3. Migration Steps and Tools Used

Migration Process:

1. **Data Backup**:
   - Backed up existing data from Sanity CMS using the Sanity CLI export tool.
2. **Data Transformation**:
   - Used scripts to map legacy data formats to the new schema.
   - Applied validation rules to ensure all records adhered to the updated schema.
3. **Data Import**:
   - Imported data into Sanity CMS using the sanity dataset import command.
   -

Tools Utilized:

- **Postman** for API testing.

- **Node.js** scripts for data transformation.
- **Sanity CLI** for data export and import.

5. Code Snippets

Migration Script:

```
  } catch (error) {
    console.error('Error deleting existing products:', error);
  }
}


async function uploadImageToSanity(imageUrl) {
  try {
    const response = await fetch(imageUrl);

    if (!response.ok) {
      throw new Error(`Failed to fetch image: ${imageUrl}`);
    }

    console.log(`Image fetched successfully from: ${imageUrl}`);

    const buffer = await response.arrayBuffer();
    const bufferImage = Buffer.from(buffer);

    const asset = await client.assets.upload('image', bufferImage, {
      filename: imageUrl.split('/').pop() ?? 'default-image-name.jpg',
    });

    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
  }
}


async function uploadProduct(product) {
  try {
    const imageId = await uploadImageToSanity(product.imageUrl);

    if (imageId) {
      const document = {
        _type: 'product',
        title: product.title,
        price: product.price,
        productImage: {
          _type: 'image',
          asset: {
            _ref: imageId,
          },
        },
        tags: product.tags,
        discountPercentage: product.discountPercentage || 0,
        description: product.description,
        isNew: product.isNew ? true : false,
      };

      const createdProduct = await client.create(document);
      console.log(`Product ${product.title} uploaded successfully:`, createdProduct);
    } else {
      console.log(`Product ${product.title} skipped due to image upload failure.`);
    }
```

API Integration:

src > app > components > ProductPage > page.tsx > ...

```tsx
interface SanityProduct {
  title: string;
  description: string;
  price: string;
  oldPrice?: string;
  discountPercentage?: string;
  imageUrl: string;
  isNew: boolean;
  tags: string[];
}

const ProductPage: React.FC = () => {
  const [products, setProducts] = useState<SanityProduct[]>([]);
  const [error, setError] = useState<boolean>(false);
  const [loading, setLoading] = useState<boolean>(false);

  useEffect(() => {
    const fetchProducts = async () => {
      setLoading(true);
      try {
        const res = await fetch("http://localhost:3000/api/products");
        if (!res.ok) {
          throw new Error("Failed to fetch products");
        }
        const data = await res.json();

        console.log(data);
        setProducts(data);
        setError(false);
      } catch (err) {
        setError(true);
      } finally {
        setLoading(false);
      }
    };

    fetchProducts();
  }, []);

  if (error) {
    return <Custom500 />;
  }

  if (loading) {
    return (
      <div className="flex justify-center items-center h-screen">
        <div ><LoadingPage/></div>
      </div>
    );
  }

  return (
    <div className="container mx-auto py-6 max-w-7xl">
      <h1 className="text-3xl font-bold text-center mb-8">Our Products</h1>
      <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4 gap-8 mt
        {products.slice(0, 8).map((product) => (
          <ProductCard
            key={product._id}
```

```
product={
```