# Code Deliverables

## 1. Code Snippet for product list:

```
  const products:Product[] = await client.fetch(`*[_type == 'product'][0...3]`);
     const builder = imageUrlBuilder(client);
   const urlFor = (source: SanityImageSource) => builder.image(source);
```

## 2. Code Snippet for individual product:

```
 const { id } = useParams();
 const [product, setProduct] = useState<Product | null>(null);
 const [loading, setLoading] = useState(true);

 useEffect(() => {
   if (!id) return;

   const query = `*[_type == 'product' && id == $id][0]`;
     client

     .fetch(query, { id })
     .then((data: Product) => {
       setProduct(data);
       setLoading(false);
     })
     .catch((error) => {
       console.error("Error fetching product:", error);
       setLoading(false);
     });
}, [id]);
```

## 3. Code Snippet for Search bar:

```
useEffect(() => {
    if (!query) return; // If there's no query, return early

    const fetchFilteredProducts = async () => {
      setLoading(true);
      try {
        // Sanity query to search for products that match the query in the name
or description
        const filterQuery = `*[_type == "product" && (name match "${query}" ||
description match "${query}")][0...12]`;
        const fetchedProducts = await client.fetch(filterQuery);
        setProducts(fetchedProducts);
      } catch (error) {
        console.error("Error fetching products:", error);
      } finally {
        setLoading(false);
      }
    };

    fetchFilteredProducts();
  }, [query]);
```

## 4. Code Snippet for API integration:

```
async function uploadImageToSanity(imagePath) {
  try {
    console.log(`Uploading image: ${imagePath}`);

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

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

    const asset = await client.assets.upload("image", bufferImage, {
      filename: imagePath.split("/").pop(),
    });
    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error("Failed to upload image:", imagePath, error);
    return null;
  }
}

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

    if (imageId) {
      const document = {
        _type: "product",
        id: product.id,
        name: product.name,
        price: product.price,
        imagePath: {
          _type: "image",
          asset: {
            _ref: imageId,
          },
        },
        discountPercentage: product.discountPercentage,
        description: product.description,
        category: product.category,
        stockLevel: product.stockLevel,
      };

      const createdProduct = await client.create(document);
      console.log(
        `Product ${product.name} uploaded successfully:`,
        createdProduct
      );
    } else {
      console.log(
        `Product ${product.name} skipped due to image upload failure.`
      );
    }
  } catch (error) {
    console.error("Error uploading product:", error);
  }
}
// Api integration
```

```javascript
async function importProducts() {
  try {
    // Fetch products from two different URLs concurrently
    const [response1, response2] = await Promise.all([
      fetch("https://template-0-beta.vercel.app/api/product"),
      fetch("https://678d1855f067bf9e24e93f90.mockapi.io/id") // Second URL here
    ]);

    // Check if both responses are ok
    if (!response1.ok) {
      throw new Error(`HTTP error for URL 1! Status: ${response1.status}`);
    }
    if (!response2.ok) {
      throw new Error(`HTTP error for URL 2! Status: ${response2.status}`);
    }

    // Parse both responses to JSON
    const products1 = await response1.json();
    const products2 = await response2.json();

    // Combine products from both sources
    const allProducts = [...products1, ...products2];

    // Upload each product
    for (const product of allProducts) {
      await uploadProduct(product);
    }
  } catch (error) {
    console.error("Error fetching products:", error);
  }
}

importProducts();
```