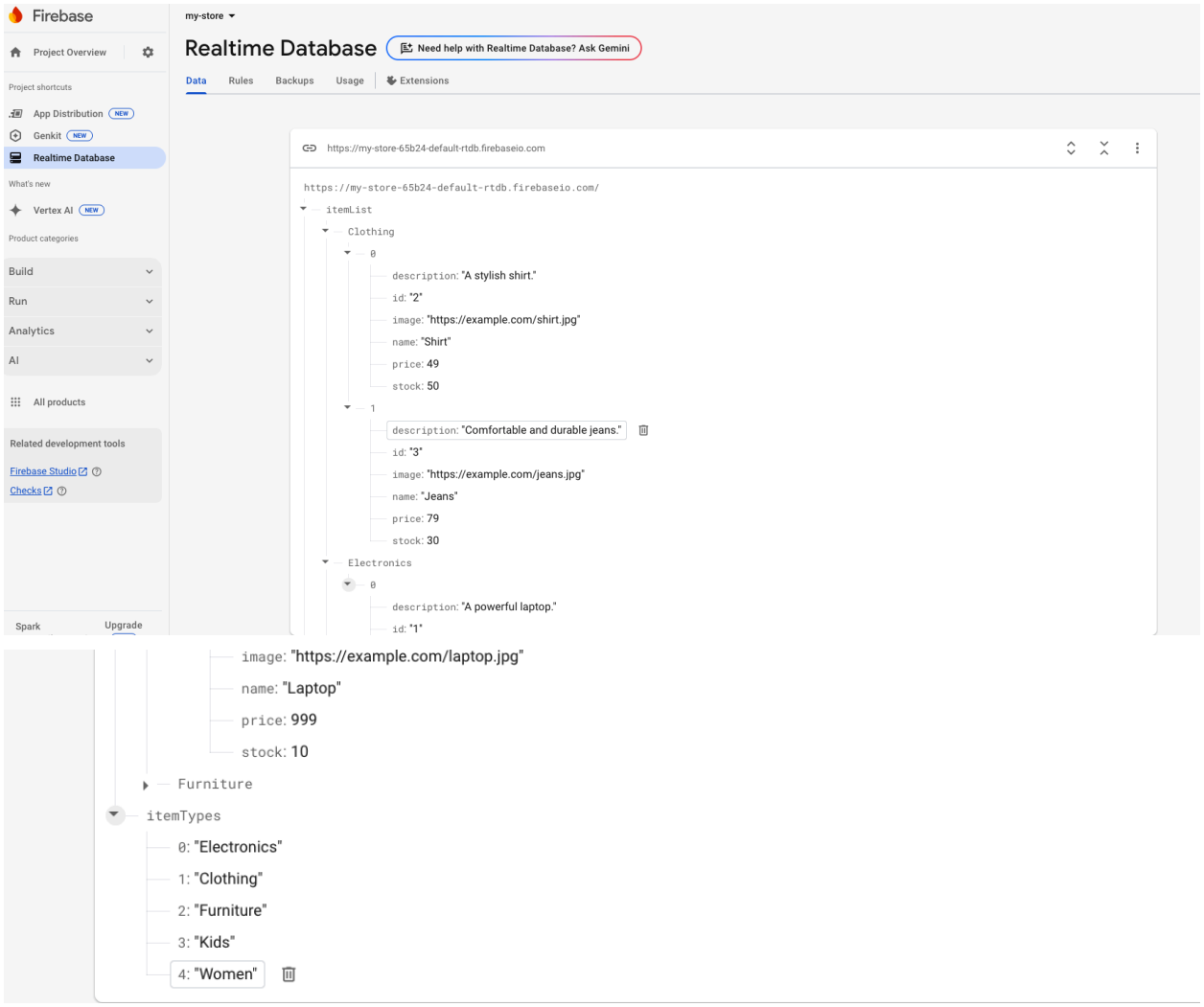
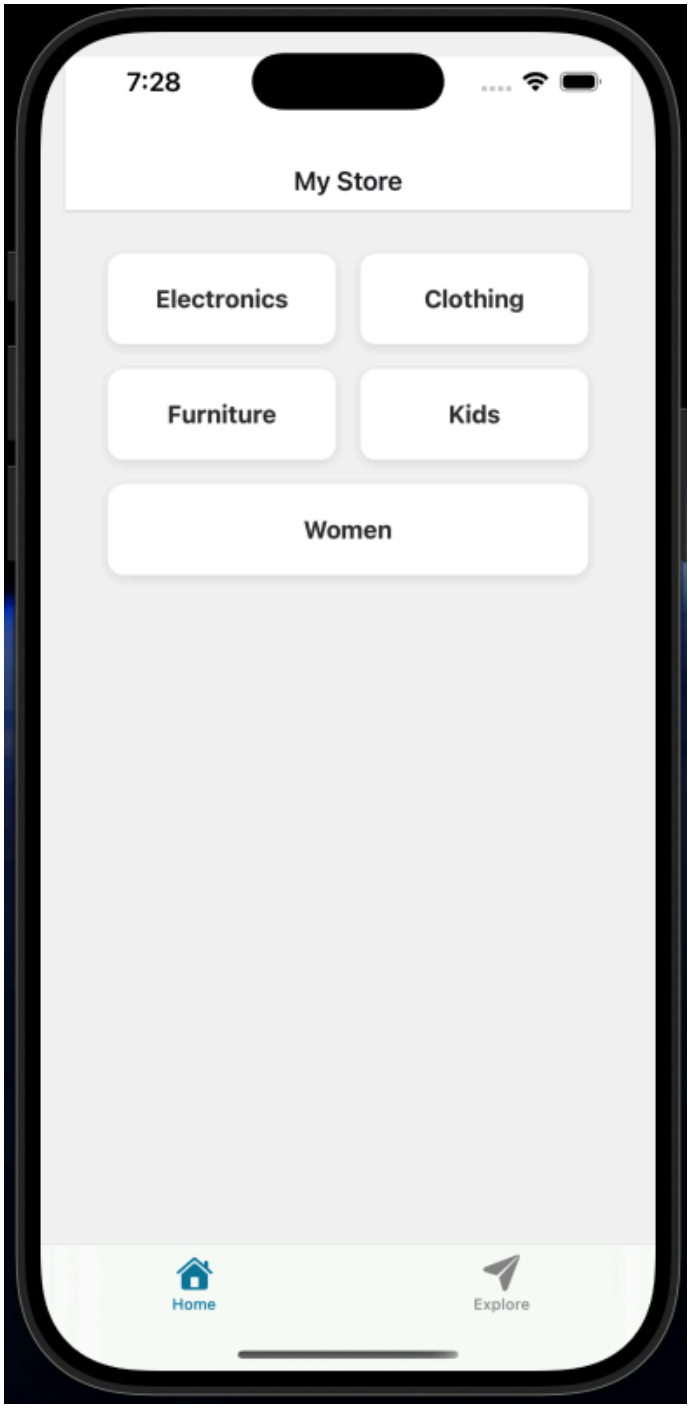
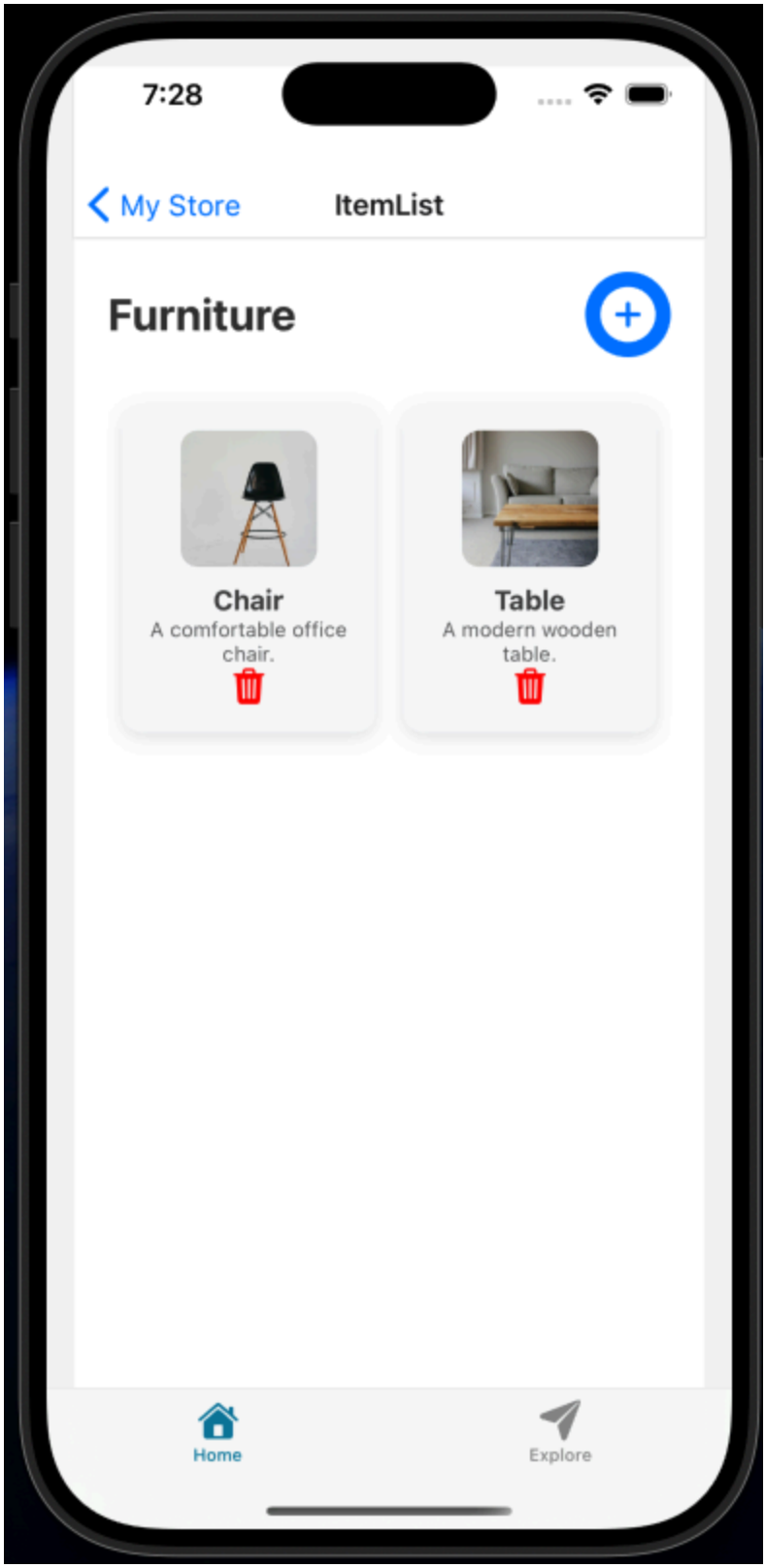
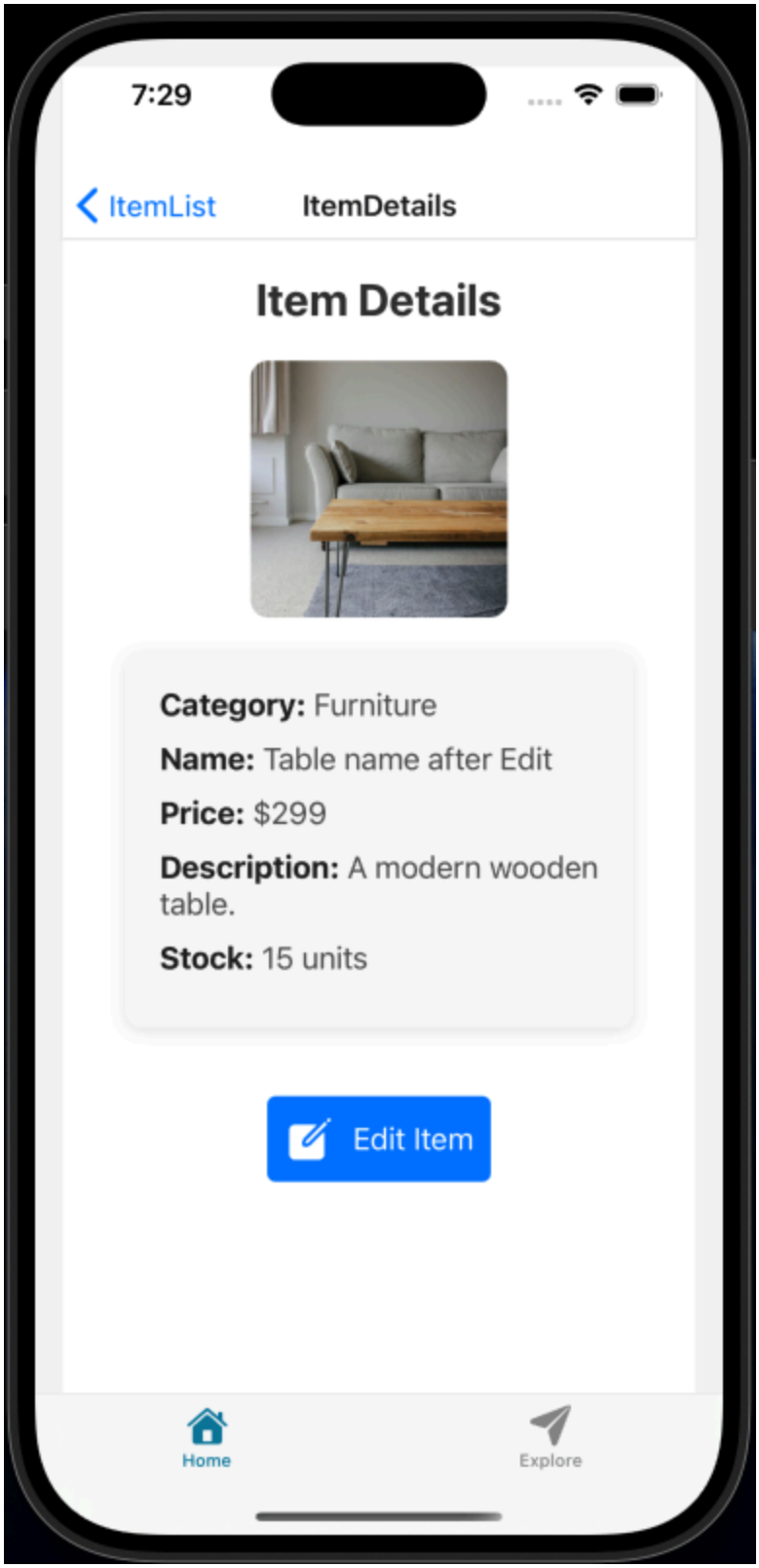
Project#05

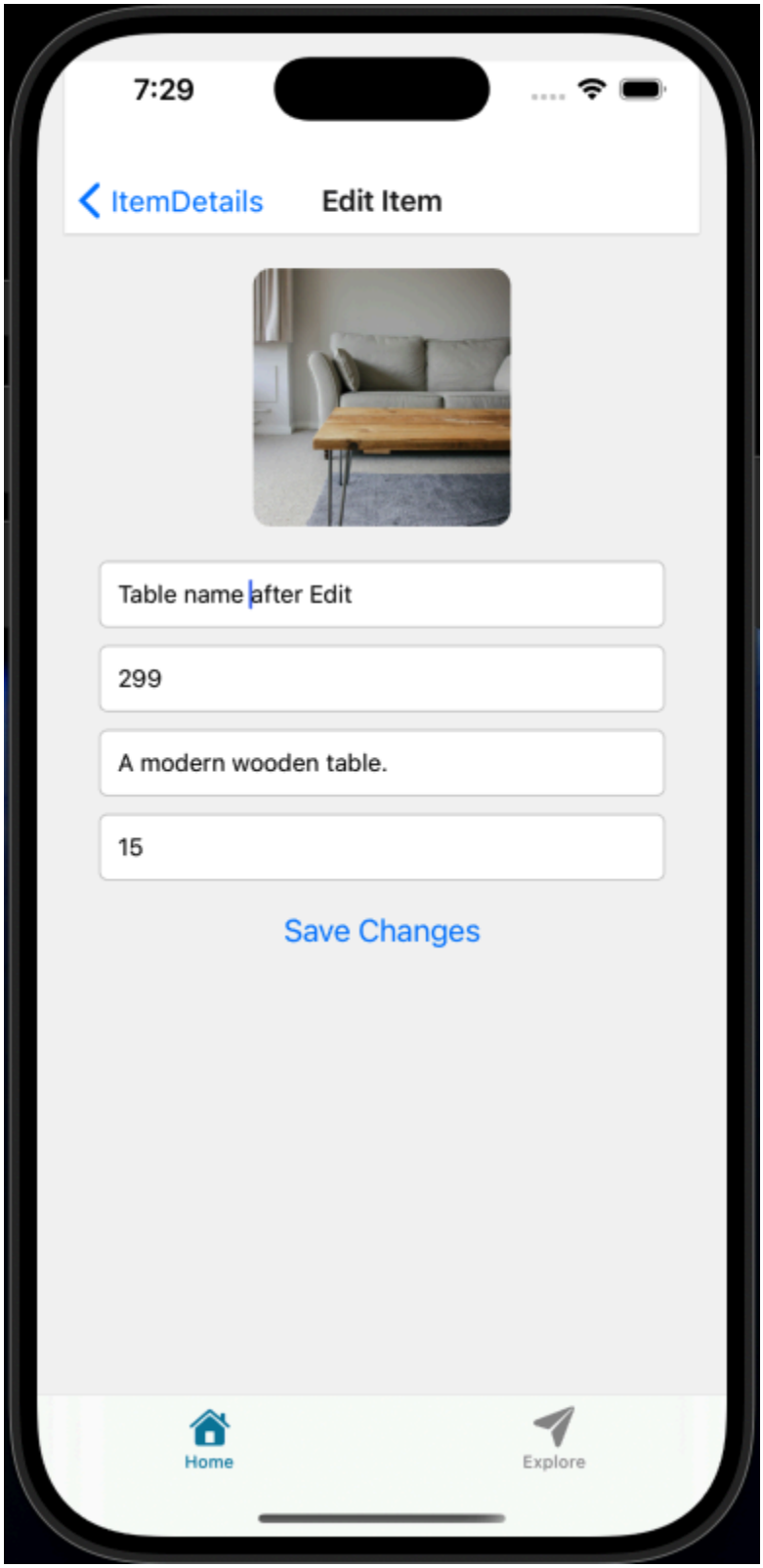
Main Page

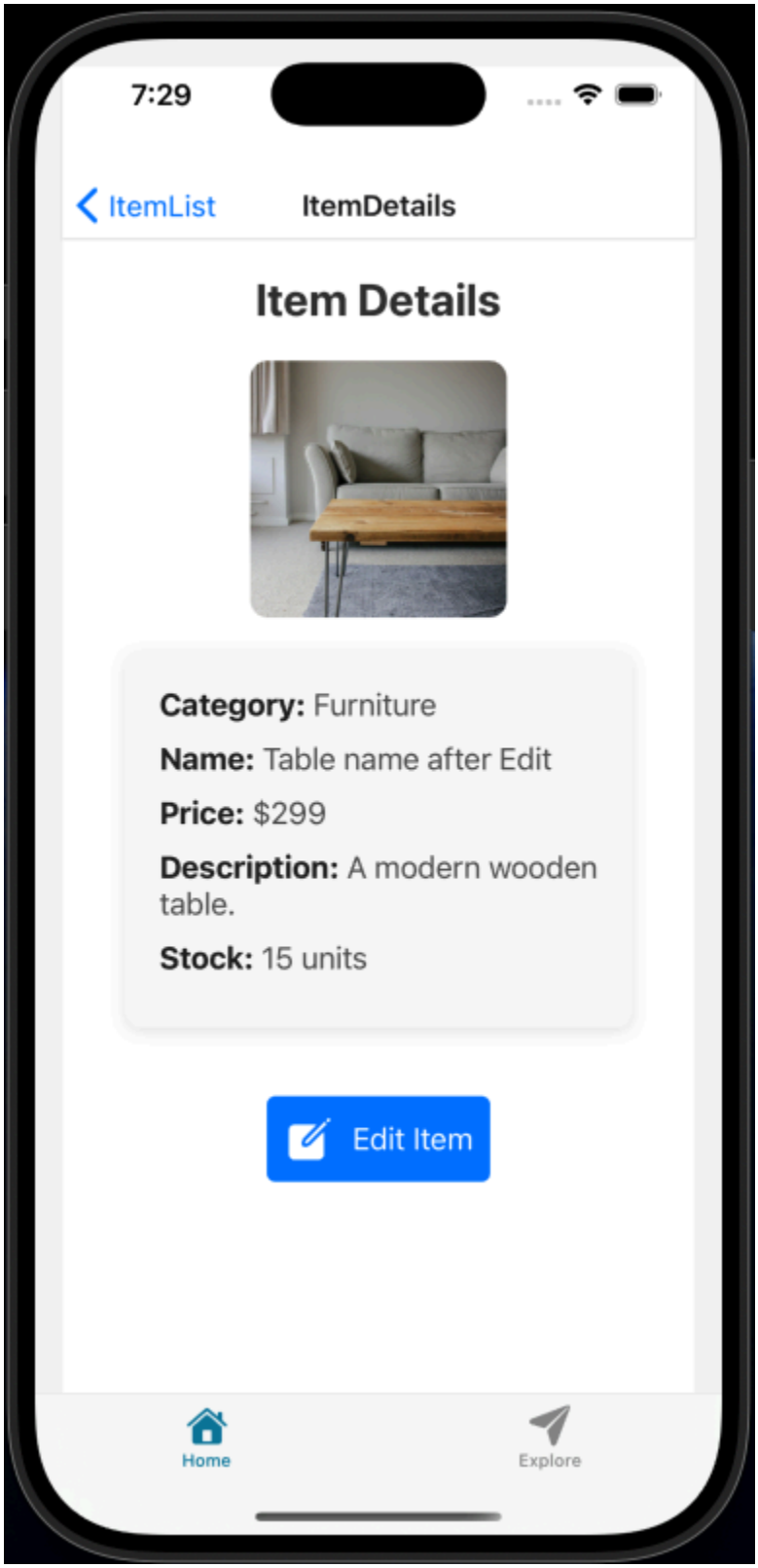


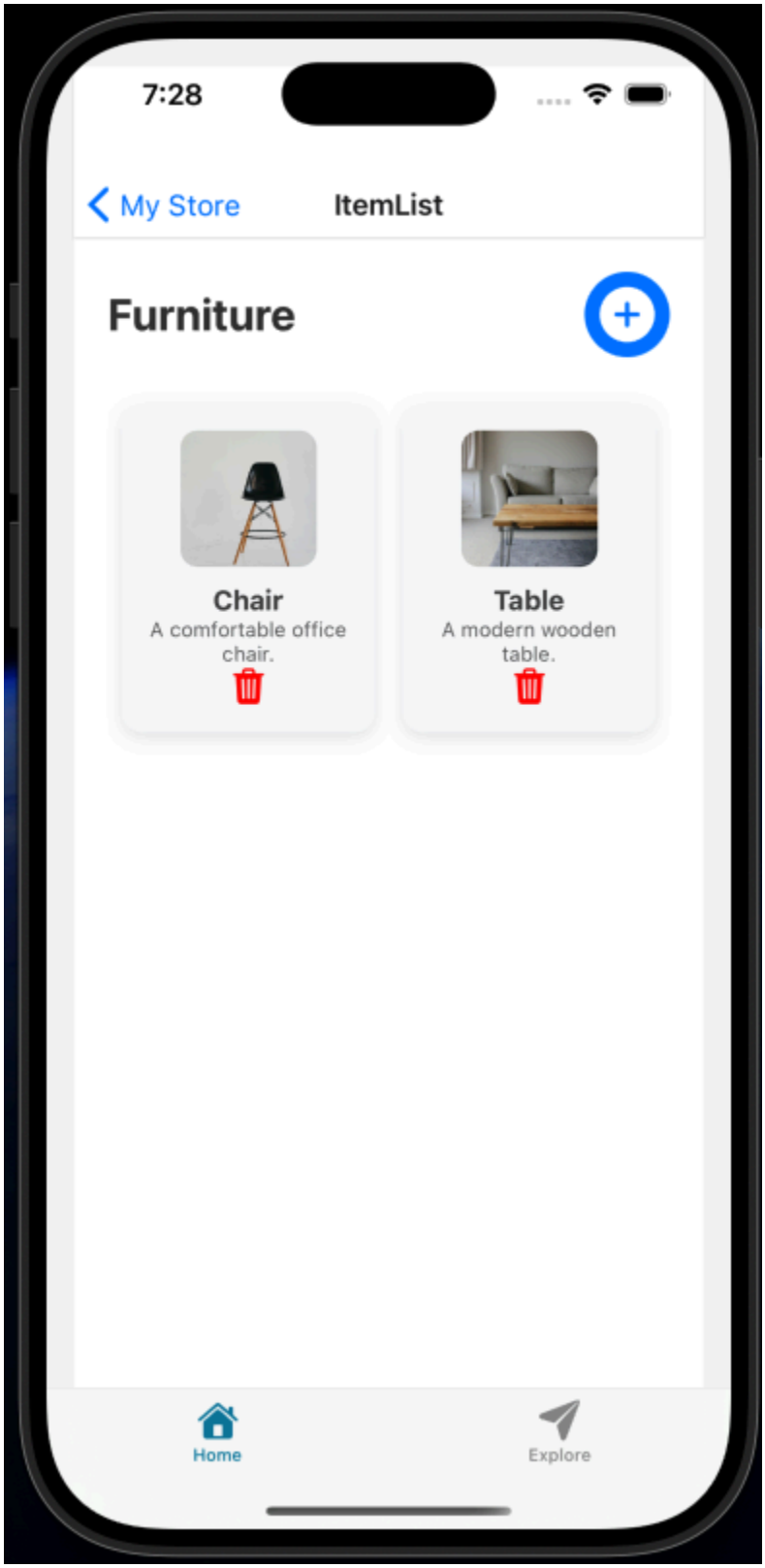

7:28
My Store
Electronics
Clothing
Furniture
Kids
Women
Home
Explore

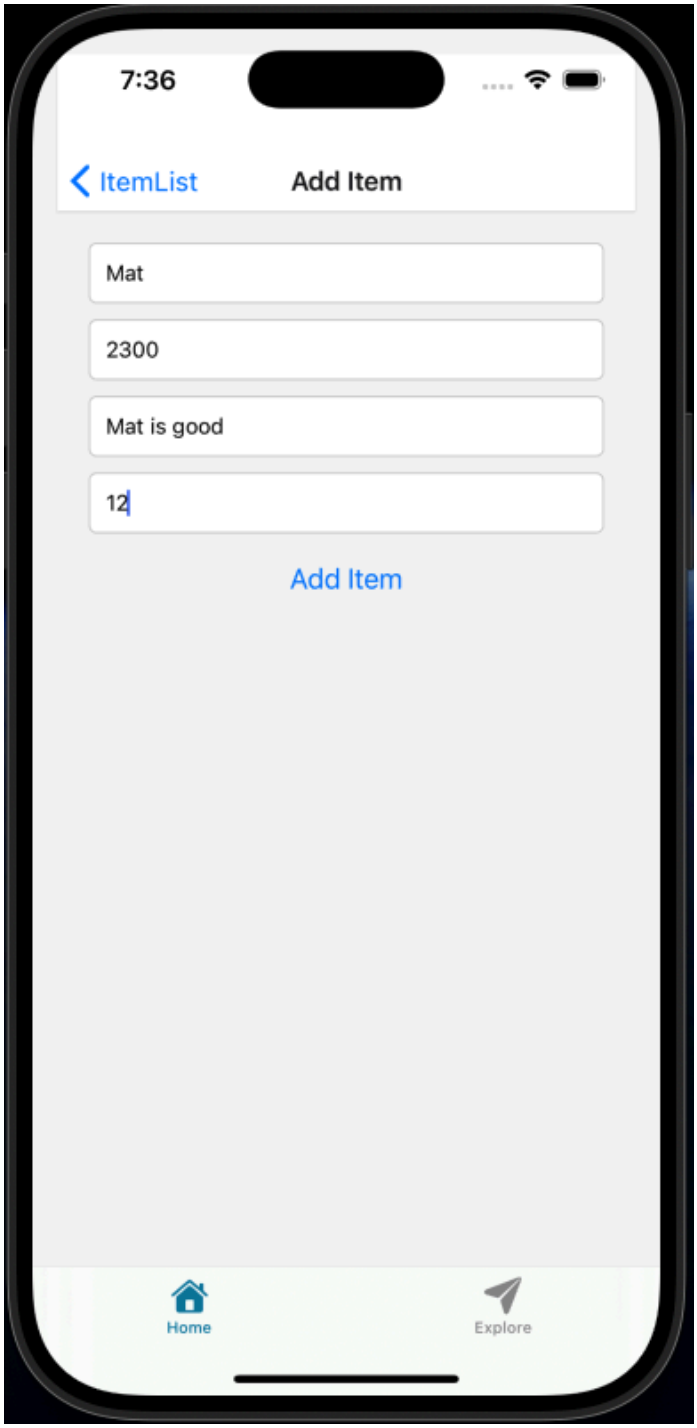

7:28
My Store
ItemList
Furniture
Chair
A comfortable office chair.
Table
A modern wooden table.
Home
Explore


7:29
ItemList
ItemDetails
Item Details
Category: Furniture
Name: Table name after Edit
Price: $299
Description: A modern wooden table.
Stock: 15 units
Edit Item
Home
Explore

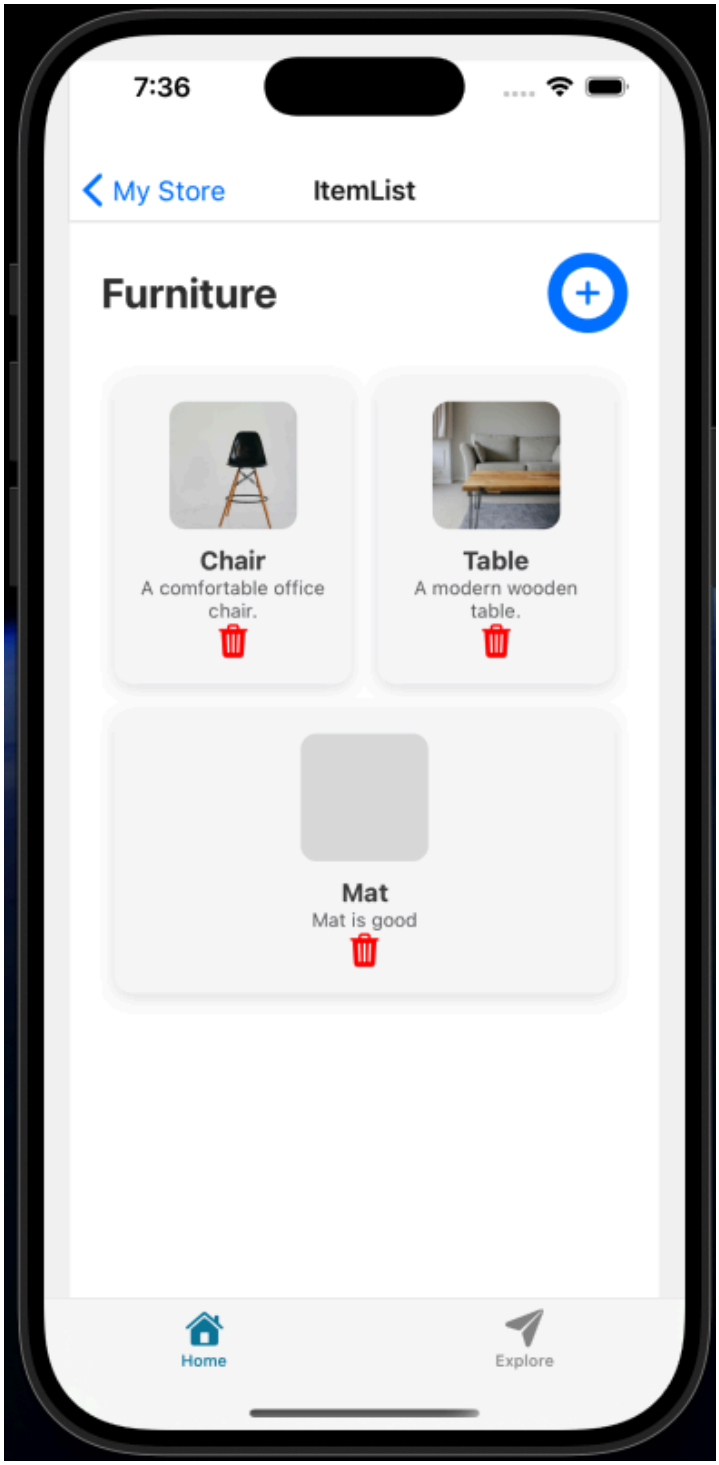

7:29
ItemDetails
Edit Item
Table name after Edit
299
A modern wooden table.
15
Save Changes
Home
Explore


7:29
ItemList
ItemDetails
Item Details
Category: Furniture
Name: Table name after Edit
Price: $299
Description: A modern wooden table.
Stock: 15 units
Edit Item
Home
Explore

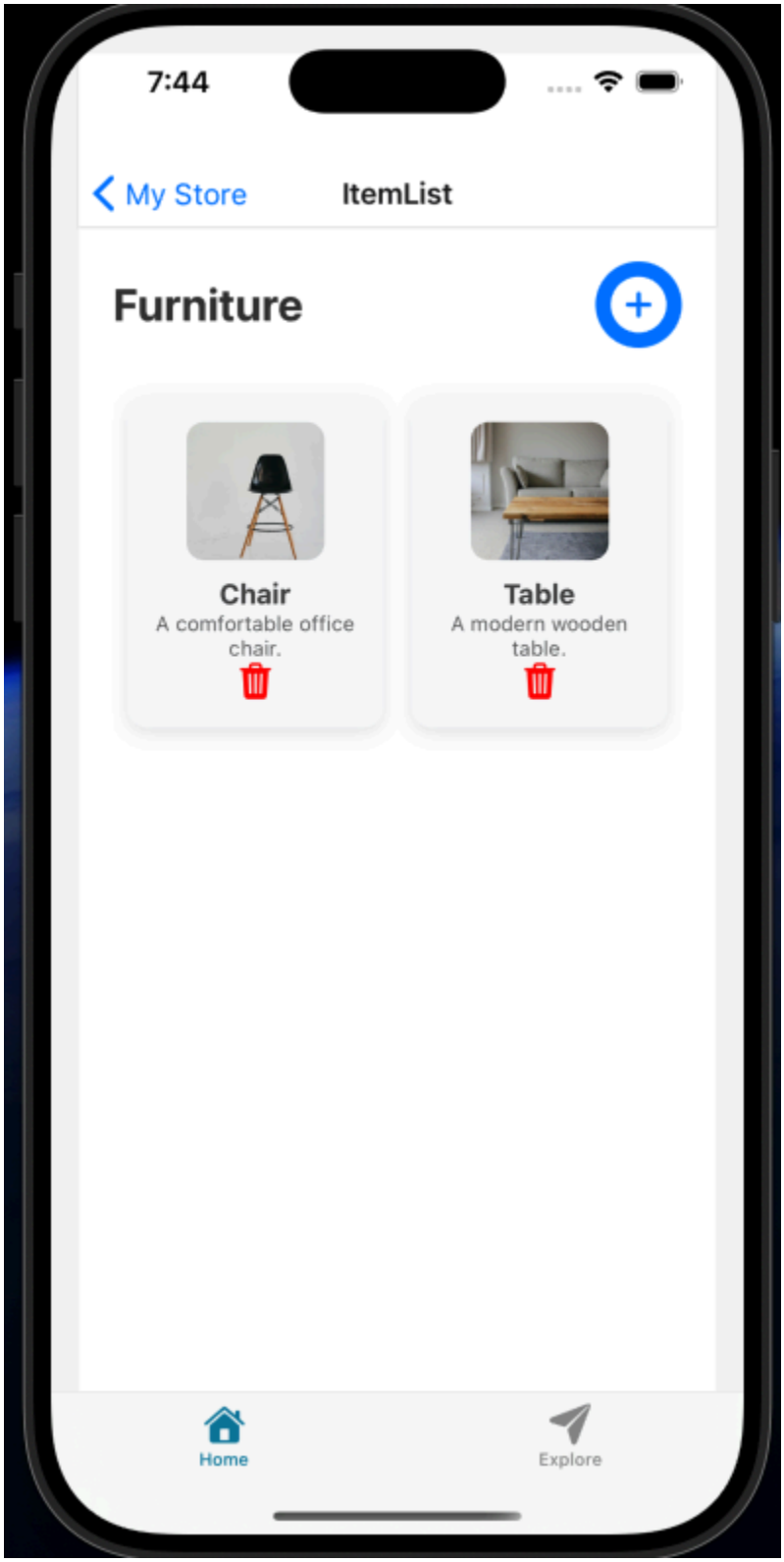

7:28
My Store
ItemList
Furniture
Chair
A comfortable office chair.
Table
A modern wooden table.
Home
Explore

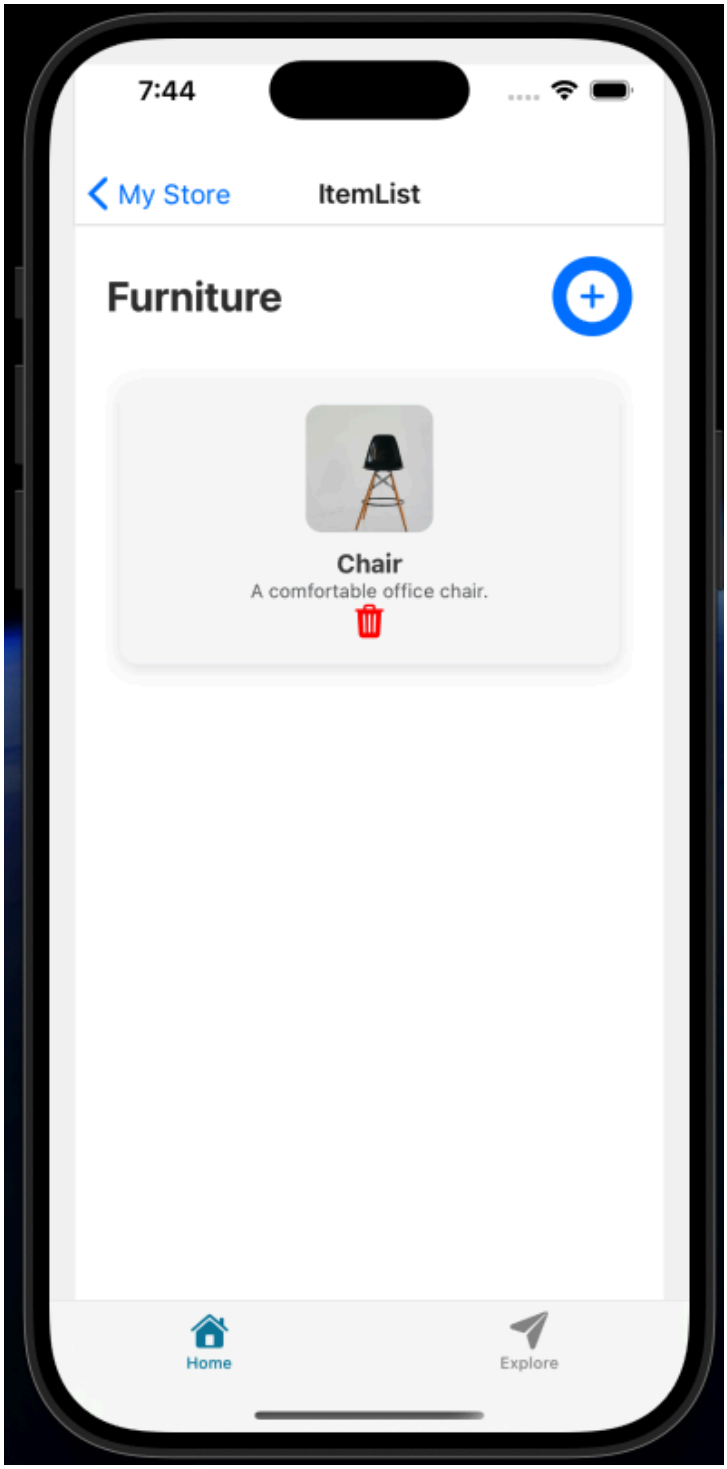


ItemList : Added item in the ItemList


7:36
My Store
ItemList
Furniture
Chair
A comfortable office chair.
Table
A modern wooden table.
Mat
Mat is good
Home
Explore

Delete


7:44
My Store
ItemList
Furniture
Chair
A comfortable office chair.
Table
A modern wooden table.
Home
Explore


7:44
My Store
ItemList
Furniture
Chair
A comfortable office chair.
Home
Explore

Code from main page App.tsx (index.tsx for others)

```
import React, { useEffect } from 'react';
import { createStackNavigator } from '@react-navigation/stack';
import ItemTypes from './components/ItemTypes';
import ItemList from './components/ItemList';
import ItemDetails from './components/ItemDetails';
import EditItem from './components/EditItem';
import AddItem from './components/AddItem';
import { View, StyleSheet } from 'react-native';


export type RootStackParamList = {
  ItemTypes: undefined;
  ItemList: { category: string };
  ItemDetails: {
    category: string;
    item: {
      id: string;
      name: string;
      price: string | number;
      description: string;
      stock: string | number;
      image?: string | any;
    };
    index: number;
  };
  EditItem: {
    category: string;
    item: {
      id: string;
      name: string;
      price: string | number;
      description: string;
      stock: string | number;
      category: string;
    };
    index: number;
  };
  AddItem: { category: string };
```

```
};

const Stack = createStackNavigator<RootStackParamList>();

const App = () => {

  return (
    <View style={styles.container}>
      <Stack.Navigator
        screenOptions={{
          headerStyle: { backgroundColor: 'white' },
          headerTitleAlign: 'center',
        }}
      >
        <Stack.Screen name="ItemTypes" component={ItemTypes} options={{ title: 'My
Store' }} />
        <Stack.Screen name="ItemList" component={ItemList} />
        <Stack.Screen name="ItemDetails" component={ItemDetails} />
        <Stack.Screen name="EditItem" component={EditItem} options={{ title: 'Edit Item'
}} />
        <Stack.Screen name="AddItem" component={AddItem} options={{ title: 'Add Item'
}} />
      </Stack.Navigator>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1, // Take full screen height
    width: '100%', // Ensures full width
  },
});

export default App;
```

Helper.ts

```
import { database } from '@/firebaseConfig'; // Import your Firebase configuration
import { ref, set, push, get, child, update, remove } from 'firebase/database'; // Firebase
Realtime Database methods
```

```
// Define and export the structure of an Item
export interface Item {
  id: string;
  name: string;
  price: string | number;
  description: string;
  stock: string | number;
  image?: string | any;
  category: string;
}

// Function to add a new item to the database
export const addItemHelper = async (item: Item): Promise<string> => {
  try {
    const itemRef = push(ref(database, 'itemList')); // Push a new item to the database
under 'itemList'
    await update(itemRef, item); // Update the reference with item data
    return itemRef.key as string; // Return the unique key of the added item
  } catch (error) {
    console.error("Error adding item:", error);
    throw new Error("Could not add item");
  }
};

// Function to fetch all items from itemList
export const getItemHelper = async (): Promise<Item[]> => {
  try {
    const snapshot = await get(child(ref(database), 'itemList')); // Get data from itemList
    const itemList = snapshot.val();

    // Flatten all items into a single array
    if (itemList) {
      let allItems: Item[] = [];

      // Loop through each category and extract items
      Object.keys(itemList).forEach(category => {
        allItems = [...allItems, ...itemList[category]]; // Flatten the items
      });
```

```
      return allItems; // Return all items
    } else {
      return []; // Return an empty array if no items are found
    }
  } catch (error) {
    console.error("Error fetching items:", error);
    throw new Error("Could not fetch items");
  }
};

// Function to fetch item categories (Types) from itemTypes
export const getTypeHelper = async (): Promise<string[]> => {
  try {
    const snapshot = await get(child(ref(database), 'itemTypes')); // Get data from
itemTypes
    const types = snapshot.val();
    return types ? types : [];
  } catch (error) {
    console.error("Error fetching types:", error);
    throw new Error("Could not fetch item types");
  }
};

// Function to edit an existing item by its ID in itemList
export const editItemHelper = async (id: string, updatedItem: Item): Promise<boolean>
=> {
  try {
    const itemRef = ref(database, `itemList/${updatedItem.category}/${id}`); // Get the
reference for the item by category and ID
    await update(itemRef, updatedItem); // Update the item data
    return true;
  } catch (error) {
    console.error("Error editing item:", error);
    throw new Error("Could not edit item");
  }
};

// Function to delete an item from itemList by its ID
export const deleteItemHelper = async (id: string, category: string): Promise<boolean>
=> {
```

```
 try {
    const itemRef = ref(database, `itemList/${category}/${id}`); // Get the reference for the
item by category and ID
    await remove(itemRef); // Remove the item from the list
    return true;
  } catch (error) {
    console.error("Error deleting item:", error);
    throw new Error("Could not delete item");
  }
};
```

ItemTypes.tsx

```
import React, { useEffect, useState } from 'react';
import { View, Text, StyleSheet, FlatList, TouchableOpacity } from 'react-native';
import { getTypeHelper } from '../utils/helper';  // Import the helper function
import { StackNavigationProp } from '@react-navigation/stack'; // Import navigation prop
type
import { RootStackParamList } from '@/App'; // Import the RootStackParamList

// Define the type for the navigation prop
type ItemTypesNavigationProp = StackNavigationProp<RootStackParamList,
'ItemTypes'>;

type Props = {
  navigation: ItemTypesNavigationProp;
};

const ItemTypes = ({ navigation }: Props) => {
  const [itemTypes, setItemTypes] = useState<string[]>([]);

  useEffect(() => {
    const fetchData = async () => {
      const types = await getTypeHelper();
      console.log(types,"------types from Item Typed-----")
      setItemTypes(types);
    };

    fetchData();
```

```
  }, []);

  return (
    <View style={styles.container}>
      <FlatList
        data={itemTypes}
        keyExtractor={(item) => item}
        renderItem={({ item }) => (
          <TouchableOpacity onPress={() => navigation.navigate('ItemList', { category: item
})}>
            <Text style={styles.itemText}>{item}</Text>
          </TouchableOpacity>
        )}
      />
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
    backgroundColor: '#f5f5f5',
  },
  itemText: {
    fontSize: 16,
    fontWeight: 'bold',
    color: '#333',
    textAlign: 'center',
  },
});

export default ItemTypes;
```

ItemList.tsx

```
import React, { useEffect, useState } from 'react';
```

```
import { View, Text, StyleSheet, FlatList, TouchableOpacity, Image } from 'react-native';
import { getItemHelper } from '../utils/helper';  // Import the helper function
import { Item } from '../utils/helper';  // Import the Item interface
import { StackNavigationProp } from '@react-navigation/stack';  // Import navigation prop
type
import { RouteProp } from '@react-navigation/native';
import { RootStackParamList } from '@/App'; // Import the RootStackParamList

// Define the type for the navigation and route props
type ItemListNavigationProp = StackNavigationProp<RootStackParamList, 'ItemList'>;
type ItemListRouteProp = RouteProp<RootStackParamList, 'ItemList'>;

type Props = {
  route: ItemListRouteProp;
  navigation: ItemListNavigationProp;
};

const ItemList = ({ route, navigation }: Props) => {
  const { category } = route.params;
  const [items, setItems] = useState<Item[]>([]);

  useEffect(() => {
    const fetchData = async () => {
      const allItems = await getItemHelper();
      console.log(allItems,"--allItems---")
      setItems(allItems.filter((item) => item.category === category));  // Filter by category
    };
    fetchData();
  }, [category]);

  return (
    <View style={styles.container}>
      <FlatList
        data={items}
        keyExtractor={(item) => item.id}
        renderItem={({ item, index }) => (
          <TouchableOpacity onPress={() => navigation.navigate('ItemDetails', { item,
category, index })}>
            <View style={styles.gridItem}>
              <Image source={{ uri: item.image }} style={styles.itemImage} />
```

```
            <Text style={styles.itemText}>{item.name}</Text>
            <Text>{item.price}</Text>
          </View>
        </TouchableOpacity>
      )}
    />
  </View>
 );
};

const styles = StyleSheet.create({
 container: {
  flex: 1,
  padding: 20,
  backgroundColor: '#ffffff',
 },
 gridItem: {
  padding: 20,
  backgroundColor: '#f8f9fa',
  borderRadius: 10,
  alignItems: 'center',
  justifyContent: 'center',
  marginBottom: 10,
 },
 itemImage: {
  width: 80,
  height: 80,
  borderRadius: 10,
 },
 itemText: {
  fontSize: 16,
  fontWeight: 'bold',
  color: '#444',
 },
});

export default ItemList;
```

ItemDetails.tsx

```
import React from 'react';
import { View, Text, StyleSheet, Image, TouchableOpacity } from 'react-native';
import { RouteProp } from '@react-navigation/native';
import { RootStackParamList } from '@/App'; // Import the RootStackParamList

// Define the type for the route prop
type ItemDetailsRouteProp = RouteProp<RootStackParamList, 'ItemDetails'>;

type Props = {
  route: ItemDetailsRouteProp;
  navigation: any;
};

const ItemDetails = ({ route, navigation }: Props) => {
  const { item, category, index } = route.params;  // Destructure the index along with item
and category

  return (
    <View style={styles.container}>
      <Text style={styles.title}>Item Details</Text>
      <Image source={{ uri: item.image }} style={styles.itemImage} />
      <View style={styles.detailContainer}>
        <Text><Text style={styles.label}>Category: </Text>{category}</Text>
        <Text><Text style={styles.label}>Name: </Text>{item.name}</Text>
        <Text><Text style={styles.label}>Price: </Text>{item.price}</Text>
        <Text><Text style={styles.label}>Description: </Text>{item.description}</Text>
        <Text><Text style={styles.label}>Stock: </Text>{item.stock}</Text>
        <Text><Text style={styles.label}>Index: </Text>{index}</Text>  {/* Display the index
*/}
      </View>
      <TouchableOpacity onPress={() => navigation.navigate('EditItem', { item, category,
index })} style={styles.editButton}>
        <Text style={styles.editText}>Edit Item</Text>
      </TouchableOpacity>
    </View>
  );
};
```

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
    alignItems: 'center',
  },
  title: {
    fontSize: 26,
    fontWeight: 'bold',
    marginBottom: 20,
    color: '#333',
  },
  itemImage: {
    width: 150,
    height: 150,
    borderRadius: 10,
  },
  detailContainer: {
    marginTop: 20,
    padding: 20,
    backgroundColor: '#f8f8f8',
    borderRadius: 10,
    shadowColor: '#000',
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.1,
    shadowRadius: 4,
  },
  label: {
    fontWeight: 'bold',
  },
  editButton: {
    marginTop: 20,
    backgroundColor: '#007bff',
    padding: 10,
    borderRadius: 5,
    alignItems: 'center',
  },
  editText: {
    color: '#fff',
    fontSize: 18,
```

```
  },
});

export default ItemDetails;
```

AddItem.tsx

```
import React, { useState } from 'react';
import { View, TextInput, Button, StyleSheet } from 'react-native';
import { addItemHelper } from '../utils/helper';  // Import the helper function
import { RouteProp } from '@react-navigation/native';
import { StackNavigationProp } from '@react-navigation/stack';
import { RootStackParamList } from '@/App';  // Import RootStackParamList

// Define the type for the route and navigation props
type AddItemRouteProp = RouteProp<RootStackParamList, 'AddItem'>;
type AddItemNavigationProp = StackNavigationProp<RootStackParamList, 'AddItem'>;

type Props = {
  route: AddItemRouteProp;
  navigation: AddItemNavigationProp;
};

const AddItem = ({ route, navigation }: Props) => {
  const { category } = route.params;  // Get category from route params
  const [itemName, setItemName] = useState('');
  const [price, setPrice] = useState('');
  const [description, setDescription] = useState('');
  const [stock, setStock] = useState('');

  const handleAddItem = async () => {
    const newItem = {
      name: itemName,
      price: parseFloat(price),  // Ensure price is a number
      description,
      stock: parseInt(stock),  // Ensure stock is a number
      category,  // Include category in the new item
      id: '',  // We'll let Firebase generate the ID
    };
```

```
    // Call the helper function to add the new item
    const itemId = await addItemHelper(newItem);

    // After adding, navigate back
    navigation.goBack();
  };

  return (
    <View style={styles.container}>
      <TextInput
        style={styles.input}
        placeholder="Item Name"
        value={itemName}
        onChangeText={setItemName}
      />
      <TextInput
        style={styles.input}
        placeholder="Price"
        value={price}
        onChangeText={setPrice}
        keyboardType="numeric"
      />
      <TextInput
        style={styles.input}
        placeholder="Description"
        value={description}
        onChangeText={setDescription}
      />
      <TextInput
        style={styles.input}
        placeholder="Stock"
        value={stock}
        onChangeText={setStock}
        keyboardType="numeric"
      />
      <Button title="Add Item" onPress={handleAddItem} />
    </View>
  );
};
```

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
    backgroundColor: '#f5f5f5',
  },
  input: {
    borderWidth: 1,
    marginBottom: 10,
    padding: 8,
  },
});

export default AddItem;
```

EditItem.tsx

```
import React, { useState } from 'react';
import { View, TextInput, Button, StyleSheet } from 'react-native';
import { editItemHelper } from '../utils/helper';  // Import the helper function
import { RouteProp } from '@react-navigation/native';
import { StackNavigationProp } from '@react-navigation/stack';
import { RootStackParamList } from '@/App';  // Import RootStackParamList
import { Item } from '../utils/helper';  // Import the Item type

// Define the type for the route prop
type EditItemRouteProp = RouteProp<RootStackParamList, 'EditItem'>;
type EditItemNavigationProp = StackNavigationProp<RootStackParamList, 'EditItem'>;

type Props = {
  route: EditItemRouteProp;
  navigation: EditItemNavigationProp;
};
```

```
const EditItem = ({ route, navigation }: Props) => {
  const { item, category, index } = route.params;  // Destructure item, category, and index
from route.params

  // Initialize state with the current item
  const [updatedItem, setUpdatedItem] = useState<Item>(item);

  const handleSaveItem = async () => {
    // Ensure the updated item has the required fields like id and category
    const updated = {
      ...updatedItem,
      price: parseFloat(updatedItem.price.toString()),  // Ensure price is a number
      stock: parseInt(updatedItem.stock.toString()),  // Ensure stock is a number
      id: item.id,  // Ensure id is included
      category: category,  // Ensure category is included
    };

    // Call the helper function to save changes
    await editItemHelper(item.id, updated);
    navigation.goBack();  // Go back to the previous screen after saving changes
  };

  return (
    <View style={styles.container}>
      <TextInput
        style={styles.input}
        value={updatedItem.name}
        onChangeText={(text) => setUpdatedItem({ ...updatedItem, name: text })}
      />
      <TextInput
        style={styles.input}
        value={String(updatedItem.price)}
        onChangeText={(text) => setUpdatedItem({ ...updatedItem, price: text })}
        keyboardType="numeric"
      />
      <TextInput
        style={styles.input}
        value={updatedItem.description}
        onChangeText={(text) => setUpdatedItem({ ...updatedItem, description: text })}
      />
```

```
      <TextInput
        style={styles.input}
        value={String(updatedItem.stock)}
        onChangeText={(text) => setUpdatedItem({ ...updatedItem, stock: text })}
        keyboardType="numeric"
      />
      <Button title="Save Changes" onPress={handleSaveItem} />
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
    backgroundColor: '#f5f5f5',
  },
  input: {
    borderWidth: 1,
    marginBottom: 10,
    padding: 8,
  },
});

export default EditItem;
```