

1.create screens and components folder

Installing react-navigation

1)npm install @react-navigation/native

2)npm expo install react-native-screens react-native-safe-area-context

3)npm install @react-navigation/native-stack

4)npm install @react-navigation/bottom-tabs

In screens create 3 js files named as AllExpenses.js,ManageExpense.js,RecentExpenses.js

In allexpenses.js

```
import { Text } from 'react-native';

function AllExpenses() {
  return <Text>AllExpenses Screen</Text>;
}

export default AllExpenses;
```

in recentexpenses.js

```
import { Text } from 'react-native';

function RecentExpenses() {
  return <Text>RecentExpenses Screen</Text>;
}

export default RecentExpenses;
```

In manageexpense.js

```
import { Text } from 'react-native';

function ManageExpense() {
  return <Text>ManageExpense Screen</Text>;
}

export default ManageExpense;
```

2) adding-navigation

In app.js

```
import { StatusBar } from 'expo-status-bar';
import { NavigationContainer } from '@react-navigation/native';
```

```

import { createNativeStackNavigator } from
 '@react-navigation/native-stack';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';

import ManageExpense from '../screens/ManageExpense';
import RecentExpenses from '../screens/RecentExpenses';
import AllExpenses from '../screens/AllExpenses';

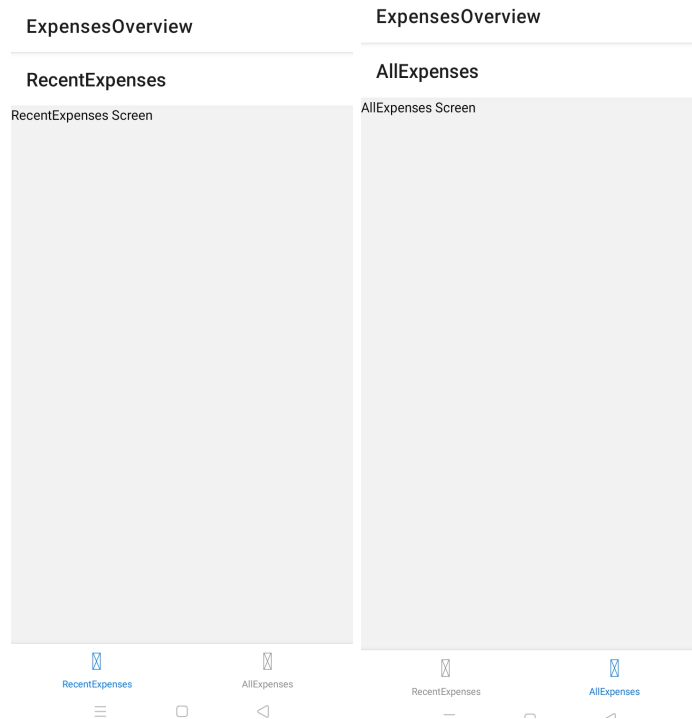
const Stack = createNativeStackNavigator();
const BottomTabs = createBottomTabNavigator();

function ExpensesOverview() {
  return (
    <BottomTabs.Navigator>
      <BottomTabs.Screen name="RecentExpenses" component={RecentExpenses}
    />
      <BottomTabs.Screen name="AllExpenses" component={AllExpenses} />
    </BottomTabs.Navigator>
  );
}

export default function App() {
  return (
    <>
      <StatusBar style="auto" />
      <NavigationContainer>
        <Stack.Navigator>
          <Stack.Screen name="ExpensesOverview"
component={ExpensesOverview} />
          <Stack.Screen name="ManageExpense" component={ManageExpense} />
        </Stack.Navigator>
      </NavigationContainer>
    </>
  );
}

```

Output:



### 3)adding -global-colors

#### In app.js

```
import { StatusBar } from 'expo-status-bar';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from
 '@react-navigation/native-stack';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { Icons } from '@expo/vector-icons';

import ManageExpense from '../screens/ManageExpense';
import RecentExpenses from '../screens/RecentExpenses';
import AllExpenses from '../screens/AllExpenses';

import { GlobalStyles } from '../constants/styles';

const Stack = createNativeStackNavigator();
const BottomTabs = createBottomTabNavigator();
```

```

function ExpensesOverview() {
  return (
    <BottomTabs.Navigator
      screenOptions={{
        headerStyle: { backgroundColor: GlobalStyles.colors.primary500 },
        headerTintColor: 'white',
        tabBarStyle: { backgroundColor: GlobalStyles.colors.primary500 },
        tabBarActiveTintColor: GlobalStyles.colors.accent500,
      }}
    >
      <BottomTabs.Screen
        name="RecentExpenses"
        component={RecentExpenses}
        options={{
          title: 'Recent Expenses',
          tabBarLabel: 'Recent',
          tabBarIcon: ({ color, size }) => (
            <Ionicons name="hourglass" size={size} color={color} />
          ),
        }}
      />
      <BottomTabs.Screen
        name="AllExpenses"
        component={AllExpenses}
        options={{
          title: 'All Expenses',
          tabBarLabel: 'All Expenses',
          tabBarIcon: ({ color, size }) => (
            <Ionicons name="calendar" size={size} color={color} />
          ),
        }}
      />
    </BottomTabs.Navigator>
  );
}

export default function App() {
  return (
    <>

```

```

<StatusBar style="auto" />
<NavigationContainer>
  <Stack.Navigator>
    <Stack.Screen
      name="ExpensesOverview"
      component={ExpensesOverview}
      options={{ headerShown: false }}
    />
    <Stack.Screen name="ManageExpense" component={ManageExpense} />
  </Stack.Navigator>
</NavigationContainer>
</>
);
}

```

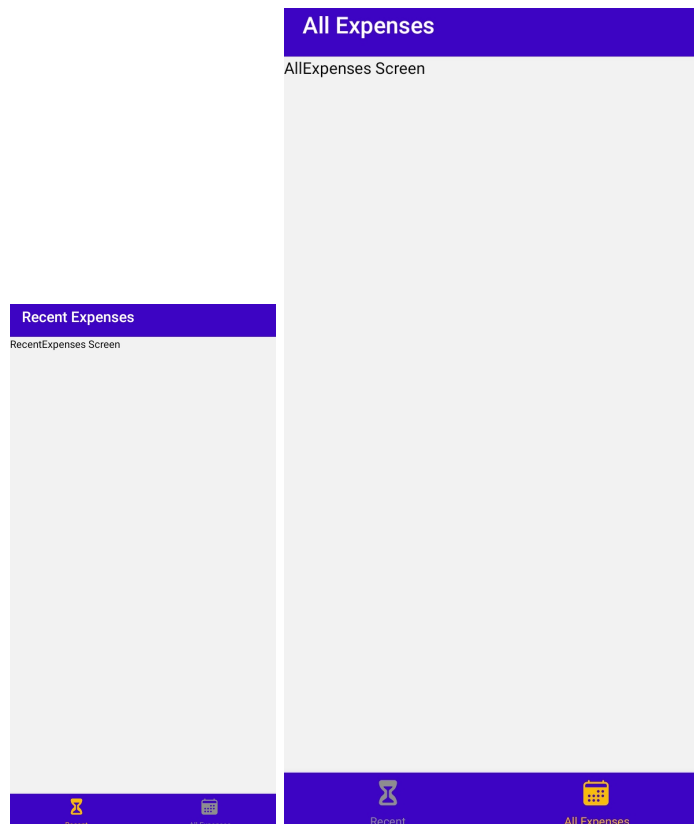
## In style.js(constants folder)

```

export const GlobalStyles = {
  colors: {
    primary50: '#e4d9fd',
    primary100: '#c6affc',
    primary200: '#a281f0',
    primary400: '#5721d4',
    primary500: '#3e04c3',
    primary700: '#2d0689',
    primary800: '#200364',
    accent500: '#f7bc0c',
    error50: '#fcc4e4',
    error500: '#9b095c',
    gray500: '#39324a',
    gray700: '#221c30',
  },
};

```

Output:



## 4)adding dummy expense data

### 1)In ExpensesSummary.js(components/expenses folder)

```
import { View, Text } from 'react-native';

function ExpensesSummary({ expenses, periodName }) {
  const expensesSum = expenses.reduce((sum, expense) => {
    return sum + expense.amount
  }, 0);

  return (
    <View>
      <Text>{periodName}</Text>
      <Text>${expensesSum.toFixed(2)}</Text>
    </View>
  );
}

export default ExpensesSummary;
```

## 2)in ExpensesList.js(components/expenses folder)

```
import { FlatList } from 'react-native';

function ExpensesList() {
  return <FlatList />;
}

export default ExpensesList;
```

## 3)in expensesoutput.js(components/expenses folder)

```
import { View } from 'react-native';

import ExpensesList from '../ExpensesList';
import ExpensesSummary from '../ExpensesSummary';

const DUMMY_EXPENSES = [
  {
    id: 'e1',
    description: 'A pair of shoes',
    amount: 59.99,
    date: new Date('2021-12-19')
  },
  {
    id: 'e2',
    description: 'A pair of trousers',
    amount: 89.29,
    date: new Date('2022-01-05')
  },
  {
    id: 'e3',
    description: 'Some bananas',
    amount: 5.99,
    date: new Date('2021-12-01')
  },
  {
    id: 'e4',
    description: 'A book',
```

```

    amount: 14.99,
    date: new Date('2022-02-19')
  },
  {
    id: 'e5',
    description: 'Another book',
    amount: 18.59,
    date: new Date('2022-02-18')
  }
];

function ExpensesOutput({ expenses, expensesPeriod }) {
  return (
    <View>
      <ExpensesSummary expenses={DUMMY_EXPENSES}
periodName={expensesPeriod} />
      <ExpensesList />
    </View>
  );
}

export default ExpensesOutput;

```

#### 4)in allexpenses.js(screens folder)

```

import ExpensesOutput from '../components/ExpensesOutput/ExpensesOutput';

function AllExpenses() {
  return <ExpensesOutput expensesPeriod="Total" />;
}

export default AllExpenses;

```

#### 5)in recentexpenses.js(screens)

```

import ExpensesOutput from '../components/ExpensesOutput/ExpensesOutput';

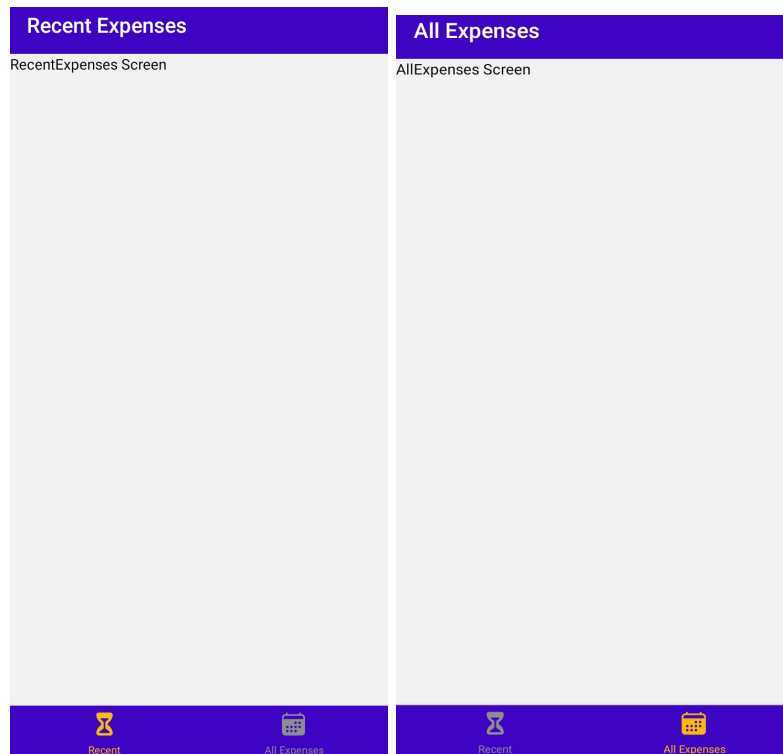
function RecentExpenses() {
  return <ExpensesOutput expensesPeriod="Last 7 Days" />;
}

```



```
export default RecentExpenses;
```

## Output:



## 5)outputting a list of expenses

In expenseslist.js(components/expenses folder)

```
import { FlatList, Text } from 'react-native';

function renderItem(itemData) {
  return <Text>{itemData.item.description}</Text>;
}

function ExpensesList({ expenses }) {
  return (
    <FlatList
      data={expenses}
      renderItem={renderItem}
      keyExtractor={(item) => item.id}
    />
  );
}
```

```
export default ExpensesList;
```

## In expenseoutput.js(components/expenses)

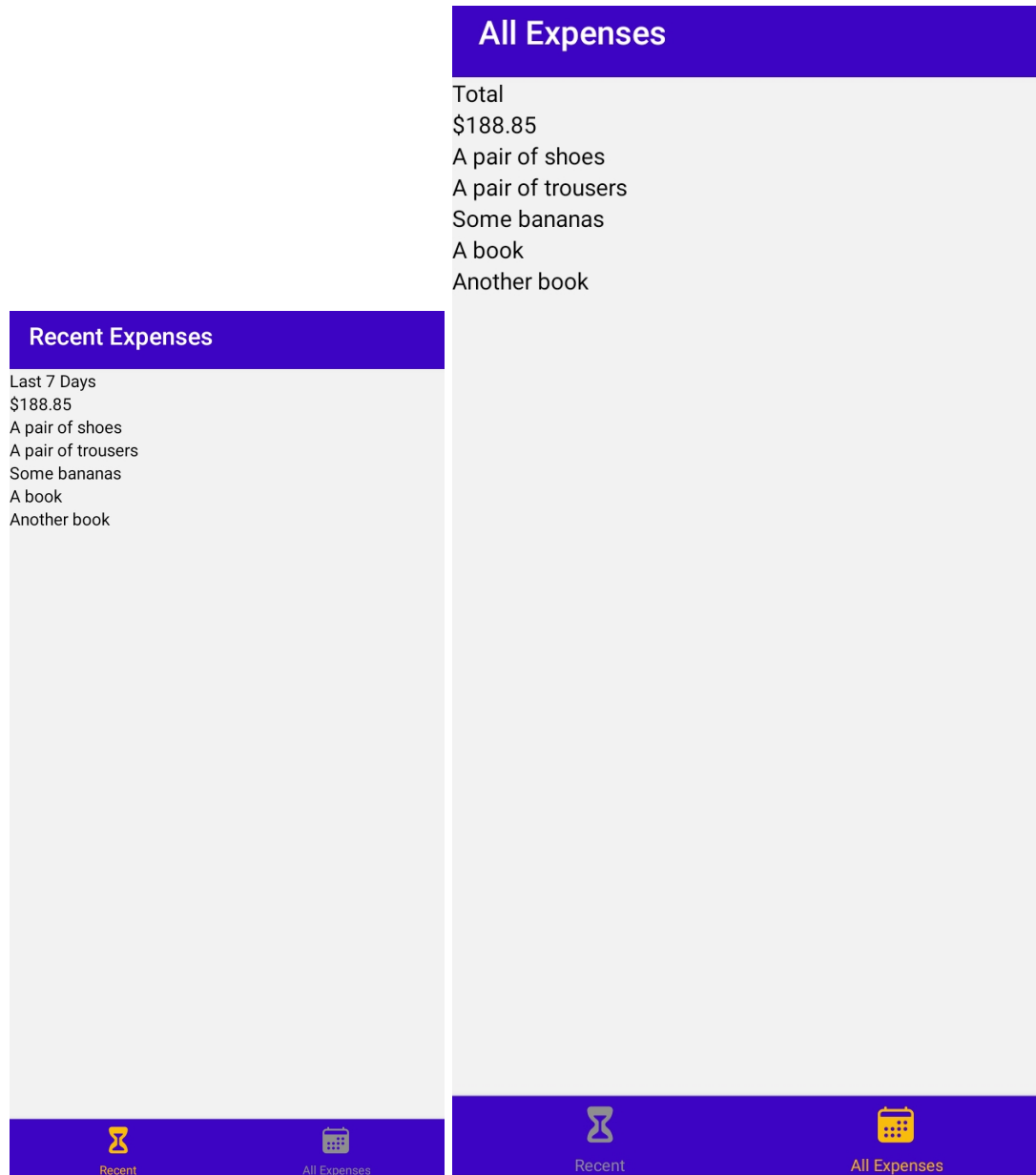
```
import { View } from 'react-native';

import ExpensesList from './ExpensesList';
import ExpensesSummary from './ExpensesSummary';

const DUMMY_EXPENSES = [
  {
    id: 'e1',
    description: 'A pair of shoes',
    amount: 59.99,
    date: new Date('2021-12-19')
  },
  {
    id: 'e2',
    description: 'A pair of trousers',
    amount: 89.29,
    date: new Date('2022-01-05')
  },
  {
    id: 'e3',
    description: 'Some bananas',
    amount: 5.99,
    date: new Date('2021-12-01')
  },
  {
    id: 'e4',
    description: 'A book',
    amount: 14.99,
    date: new Date('2022-02-19')
  },
  {
    id: 'e5',
    description: 'Another book',
    amount: 18.59,
    date: new Date('2022-02-18')
  }
]
```

```
];  
  
function ExpensesOutput({ expenses, expensesPeriod }) {  
  return (  
    <View>  
      <ExpensesSummary expenses={DUMMY_EXPENSES}  
periodName={expensesPeriod} />  
      <ExpensesList expenses={DUMMY_EXPENSES} />  
    </View>  
  );  
}  
  
export default ExpensesOutput;
```

Output:



## 6)improving app-layout and styling

### In expensesummary.js(components/expenses)

```
import { View, Text, StyleSheet } from 'react-native';

import { GlobalStyles } from '../constants/styles';

function ExpensesSummary({ expenses, periodName }) {
  const expensesSum = expenses.reduce((sum, expense) => {
    return sum + expense.amount;
  }, 0);
```

```

    return (
      <View style={styles.container}>
        <Text style={styles.period}>{periodName}</Text>
        <Text style={styles.sum}>${expensesSum.toFixed(2)}</Text>
      </View>
    );
  }

export default ExpensesSummary;

const styles = StyleSheet.create({
  container: {
    padding: 8,
    backgroundColor: GlobalStyles.colors.primary50,
    borderRadius: 6,
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
  },
  period: {
    fontSize: 12,
    color: GlobalStyles.colors.primary400,
  },
  sum: {
    fontSize: 16,
    fontWeight: 'bold',
    color: GlobalStyles.colors.primary500,
  },
});

```

## In expensesoutput.js(components/expenses)

```

import { StyleSheet, View } from 'react-native';

import { GlobalStyles } from '../../constants/styles';
import ExpensesList from './ExpensesList';
import ExpensesSummary from './ExpensesSummary';

const DUMMY_EXPENSES = [
  {

```

```

        id: 'e1',
        description: 'A pair of shoes',
        amount: 59.99,
        date: new Date('2021-12-19')
    },
    {
        id: 'e2',
        description: 'A pair of trousers',
        amount: 89.29,
        date: new Date('2022-01-05')
    },
    {
        id: 'e3',
        description: 'Some bananas',
        amount: 5.99,
        date: new Date('2021-12-01')
    },
    {
        id: 'e4',
        description: 'A book',
        amount: 14.99,
        date: new Date('2022-02-19')
    },
    {
        id: 'e5',
        description: 'Another book',
        amount: 18.59,
        date: new Date('2022-02-18')
    }
];

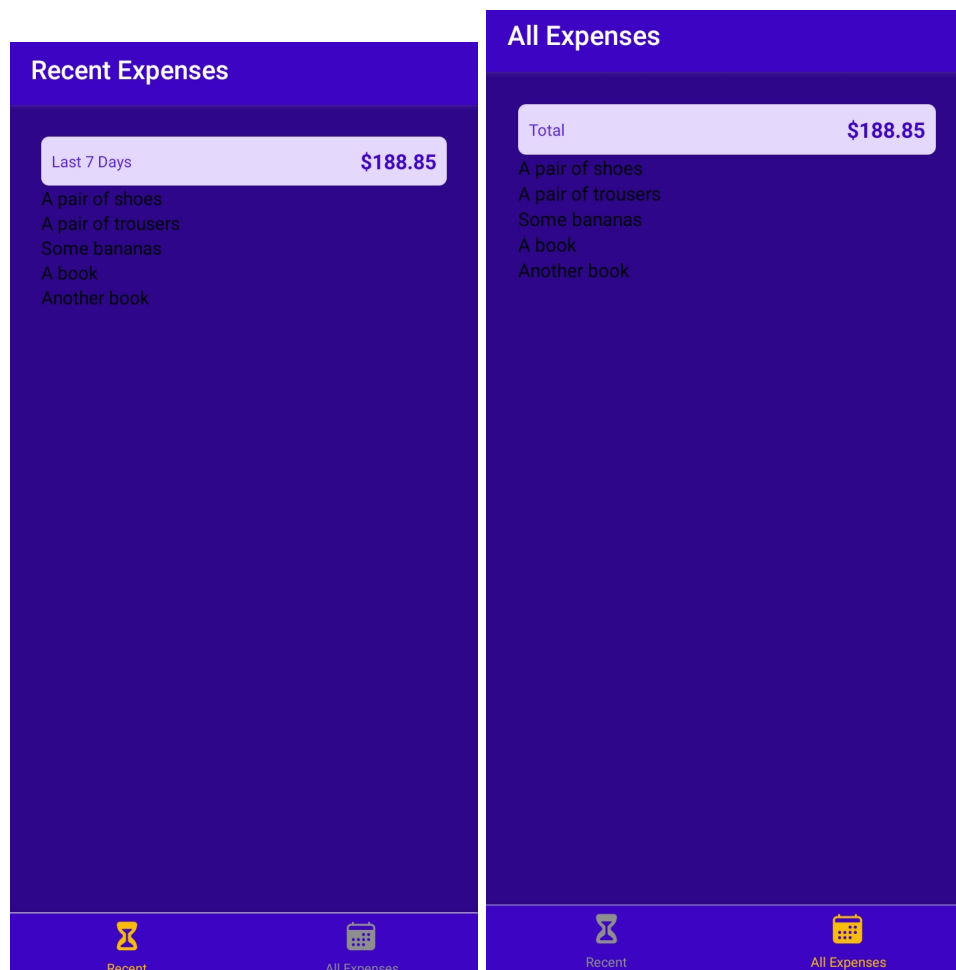
function ExpensesOutput({ expenses, expensesPeriod }) {
    return (
        <View style={styles.container}>
            <ExpensesSummary expenses={DUMMY_EXPENSES}
periodName={expensesPeriod} />
            <ExpensesList expenses={DUMMY_EXPENSES} />
        </View>
    );
}

```

```
export default ExpensesOutput;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 24,
    backgroundColor: GlobalStyles.colors.primary700
  }
});
```

Output:



7)working-on-expenses-list-items

In expenseitem.js(components/expenses)

```
import { Pressable, StyleSheet, Text, View } from 'react-native';
```

```

import { GlobalStyles } from '../../constants/styles';

function ExpenseItem({ description, amount, date }) {
  return (
    <Pressable>
      <View style={styles.expenseItem}>
        <View>
          <Text style={[styles.textBase, styles.description]}>
            {description}
          </Text>
          <Text style={styles.textBase}>{date.toString()}</Text>
        </View>
        <View style={styles.amountContainer}>
          <Text style={styles.amount}>{amount}</Text>
        </View>
      </View>
    </Pressable>
  );
}

export default ExpenseItem;

const styles = StyleSheet.create({
  expenseItem: {
    padding: 12,
    marginVertical: 8,
    backgroundColor: GlobalStyles.colors.primary500,
    flexDirection: 'row',
    justifyContent: 'space-between',
    borderRadius: 6,
    elevation: 3,
    shadowColor: GlobalStyles.colors.gray500,
    shadowRadius: 4,
    shadowOffset: { width: 1, height: 1 },
    shadowOpacity: 0.4,
  },
  textBase: {
    color: GlobalStyles.colors.primary50,
  },
  description: {

```



```

    fontSize: 16,
    marginBottom: 4,
    fontWeight: 'bold',
  },
  amountContainer: {
    paddingHorizontal: 12,
    paddingVertical: 4,
    backgroundColor: 'white',
    justifyContent: 'center',
    alignItems: 'center',
    borderRadius: 4,
  },
  amount: {
    color: GlobalStyles.colors.primary500,
    fontWeight: 'bold',
  },
});

```

## 2)in expenseslist.js(components/expenses)

```

import { FlatList } from 'react-native';

import ExpenseItem from './ExpenseItem';

function renderExpenseItem(itemData) {
  return <ExpenseItem {...itemData.item} />;
}

function ExpensesList({ expenses }) {
  return (
    <FlatList
      data={expenses}
      renderItem={renderExpenseItem}
      keyExtractor={(item) => item.id}
    />
  );
}

export default ExpensesList;

```

### 3)in expensesoutput.js(components/expenses)

```
import { StyleSheet, View } from 'react-native';

import { GlobalStyles } from '../../constants/styles';
import ExpensesList from '../ExpensesList';
import ExpensesSummary from '../ExpensesSummary';

const DUMMY_EXPENSES = [
  {
    id: 'e1',
    description: 'A pair of shoes',
    amount: 59.99,
    date: new Date('2021-12-19')
  },
  {
    id: 'e2',
    description: 'A pair of trousers',
    amount: 89.29,
    date: new Date('2022-01-05')
  },
  {
    id: 'e3',
    description: 'Some bananas',
    amount: 5.99,
    date: new Date('2021-12-01')
  },
  {
    id: 'e4',
    description: 'A book',
    amount: 14.99,
    date: new Date('2022-02-19')
  },
  {
    id: 'e5',
    description: 'Another book',
    amount: 18.59,
    date: new Date('2022-02-18')
  },
  {
    id: 'e6',
```

```

        description: 'A pair of trousers',
        amount: 89.29,
        date: new Date('2022-01-05')
    },
    {
        id: 'e7',
        description: 'Some bananas',
        amount: 5.99,
        date: new Date('2021-12-01')
    },
    {
        id: 'e8',
        description: 'A book',
        amount: 14.99,
        date: new Date('2022-02-19')
    },
    {
        id: 'e9',
        description: 'Another book',
        amount: 18.59,
        date: new Date('2022-02-18')
    }
];

function ExpensesOutput({ expenses, expensesPeriod }) {
    return (
        <View style={styles.container}>
            <ExpensesSummary expenses={DUMMY_EXPENSES}
periodName={expensesPeriod} />
            <ExpensesList expenses={DUMMY_EXPENSES} />
        </View>
    );
}

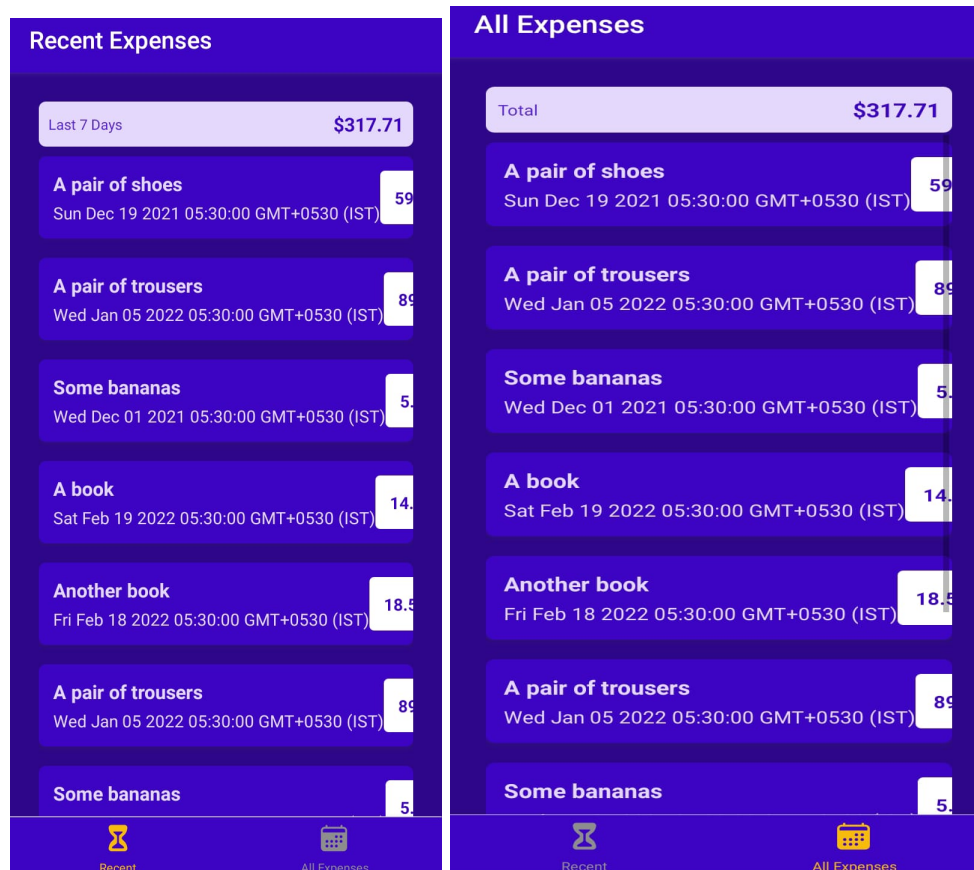
export default ExpensesOutput;

const styles = StyleSheet.create({
    container: {
        flex: 1,
        paddingHorizontal: 24,

```

```
paddingTop: 24,
paddingBottom: 0,
backgroundColor: GlobalStyles.colors.primary700
}
});
```

## Output:



## 8)formatting-dates

### In expenseitem.js(components/expense)

```
import { Pressable, StyleSheet, Text, View } from 'react-native';

import { GlobalStyles } from '../../constants/styles';
import { getFormattedDate } from '../../util/date';

function ExpenseItem({ description, amount, date }) {
  return (
    <Pressable>
      <View style={styles.expenseItem}>
```

```

    <View>
      <Text style={[styles.textBase, styles.description]}>
        {description}
      </Text>
      <Text style={styles.textBase}>{getFormattedDate(date)}</Text>
    </View>
    <View style={styles.amountContainer}>
      <Text style={styles.amount}>{amount.toFixed(2)}</Text>
    </View>
  </View>
</Pressable>
);
}

```

```
export default ExpenseItem;
```

```

const styles = StyleSheet.create({
  expenseItem: {
    padding: 12,
    marginVertical: 8,
    backgroundColor: GlobalStyles.colors.primary500,
    flexDirection: 'row',
    justifyContent: 'space-between',
    borderRadius: 6,
    elevation: 3,
    shadowColor: GlobalStyles.colors.gray500,
    shadowRadius: 4,
    shadowOffset: { width: 1, height: 1 },
    shadowOpacity: 0.4,
  },
  textBase: {
    color: GlobalStyles.colors.primary50,
  },
  description: {
    fontSize: 16,
    marginBottom: 4,
    fontWeight: 'bold',
  },
  amountContainer: {
    paddingHorizontal: 12,
  },
});

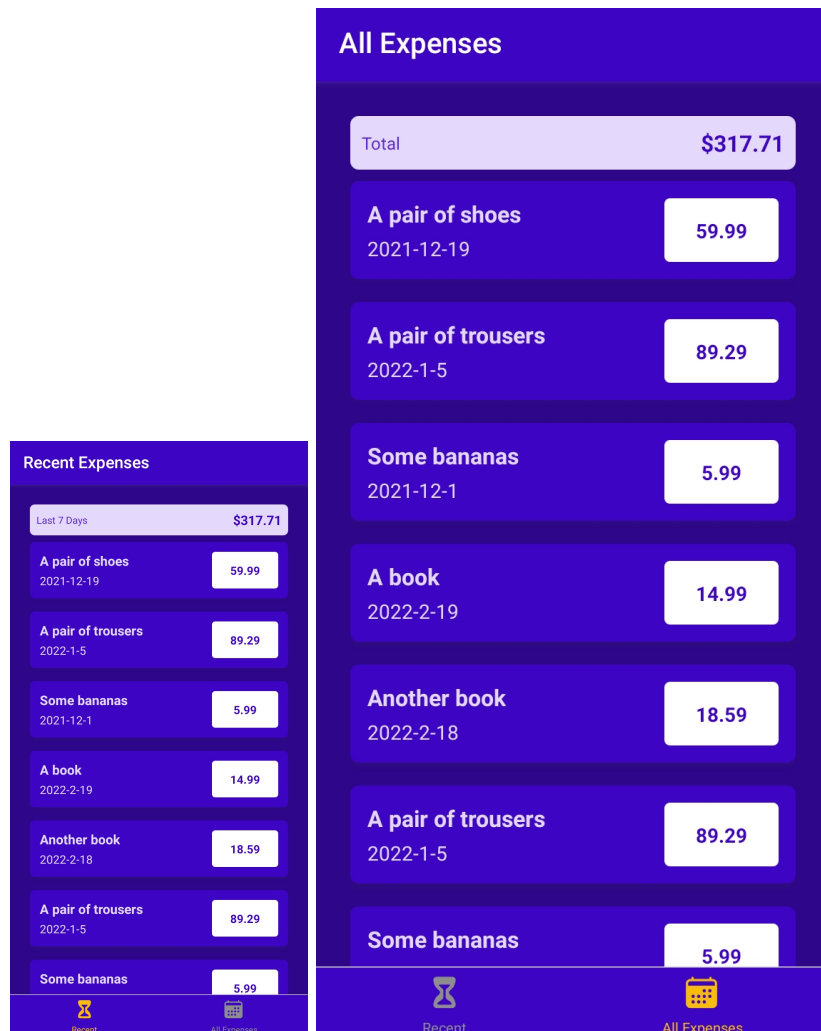
```

```
paddingVertical: 4,  
backgroundColor: 'white',  
justifyContent: 'center',  
alignItems: 'center',  
borderRadius: 4,  
minWidth: 80  
},  
amount: {  
  color: GlobalStyles.colors.primary500,  
  fontWeight: 'bold',  
},  
));
```

## In date.js(util)

```
export function getFormattedDate(date) {  
  return `${date.getFullYear()}-${date.getMonth() +  
1}-${date.getDate()}`;  
}
```

## Output:



9)adding a header button and making expense items tappable  
In app.js

```
import { StatusBar } from 'expo-status-bar';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from
 '@react-navigation/native-stack';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { Icons } from '@expo/vector-icons';

import ManageExpense from '../screens/ManageExpense';
import RecentExpenses from '../screens/RecentExpenses';
import AllExpenses from '../screens/AllExpenses';

import { GlobalStyles } from '../constants/styles';
```

```

import IconButton from '../components/UI/IconButton';

const Stack = createNativeStackNavigator();
const BottomTabs = createBottomTabNavigator();

function ExpensesOverview() {
  return (
    <BottomTabs.Navigator
      screenOptions={({ navigation }) => ({
        headerStyle: { backgroundColor: GlobalStyles.colors.primary500 },
        headerTintColor: 'white',
        tabBarStyle: { backgroundColor: GlobalStyles.colors.primary500 },
        tabBarActiveTintColor: GlobalStyles.colors.accent500,
        headerRight: ({ tintColor }) => (
          <IconButton
            icon="add"
            size={24}
            color={tintColor}
            onPress={() => {
              navigation.navigate('ManageExpense');
            }}
          />
        ),
      })
    >
    <BottomTabs.Screen
      name="RecentExpenses"
      component={RecentExpenses}
      options={{
        title: 'Recent Expenses',
        tabBarLabel: 'Recent',
        tabBarIcon: ({ color, size }) => (
          <Ionicons name="hourglass" size={size} color={color} />
        ),
      }}
    />
    <BottomTabs.Screen
      name="AllExpenses"
      component={AllExpenses}
      options={{

```



```

        title: 'All Expenses',
        tabBarLabel: 'All Expenses',
        tabBarIcon: ({ color, size }) => (
          <Ionicons name="calendar" size={size} color={color} />
        ),
      }}
    />
  </BottomTabs.Navigator>
);
}

export default function App() {
  return (
    <>
      <StatusBar style="auto" />
      <NavigationContainer>
        <Stack.Navigator>
          <Stack.Screen
            name="ExpensesOverview"
            component={ExpensesOverview}
            options={{ headerShown: false }}
          />
          <Stack.Screen name="ManageExpense" component={ManageExpense} />
        </Stack.Navigator>
      </NavigationContainer>
    </>
  );
}

```

### In iconbutton.js(ui folder)

```

import { Pressable, StyleSheet, View } from 'react-native';
import { Ionicons } from '@expo/vector-icons';

function IconButton({ icon, size, color, onPress }) {
  return (
    <Pressable
      onPress={onPress}
      style={({ pressed }) => pressed && styles.pressed}
    >
      <View style={styles.buttonContainer}>

```

```

        <Icons name={icon} size={size} color={color} />
      </View>
    </Pressable>
  );
}

export default IconButton;

const styles = StyleSheet.create({
  buttonContainer: {
    borderRadius: 24,
    padding: 6,
    marginHorizontal: 8,
    marginVertical: 2
  },
  pressed: {
    opacity: 0.75,
  },
});

```

## In expenseitem.js

```

import { Pressable, StyleSheet, Text, View } from 'react-native';
import { useNavigation } from '@react-navigation/native';

import { GlobalStyles } from '../../constants/styles';
import { getFormattedDate } from '../../util/date';

function ExpenseItem({ description, amount, date }) {
  const navigation = useNavigation();

  function expensePressHandler() {
    navigation.navigate('ManageExpense');
  }

  return (
    <Pressable
      onPress={expensePressHandler}
      style={({ pressed }) => pressed && styles.pressed}
    >
      <View style={styles.expenseItem}>

```

```

    <View>
      <Text style={[styles.textBase, styles.description]}>
        {description}
      </Text>
      <Text style={styles.textBase}>{getFormattedDate(date)}</Text>
    </View>
    <View style={styles.amountContainer}>
      <Text style={styles.amount}>{amount.toFixed(2)}</Text>
    </View>
  </View>
</Pressable>
);
}

```

```
export default ExpenseItem;
```

```

const styles = StyleSheet.create({
  pressed: {
    opacity: 0.75,
  },
  expenseItem: {
    padding: 12,
    marginVertical: 8,
    backgroundColor: GlobalStyles.colors.primary500,
    flexDirection: 'row',
    justifyContent: 'space-between',
    borderRadius: 6,
    elevation: 3,
    shadowColor: GlobalStyles.colors.gray500,
    shadowRadius: 4,
    shadowOffset: { width: 1, height: 1 },
    shadowOpacity: 0.4,
  },
  textBase: {
    color: GlobalStyles.colors.primary50,
  },
  description: {
    fontSize: 16,
    marginBottom: 4,
    fontWeight: 'bold',
  },
});

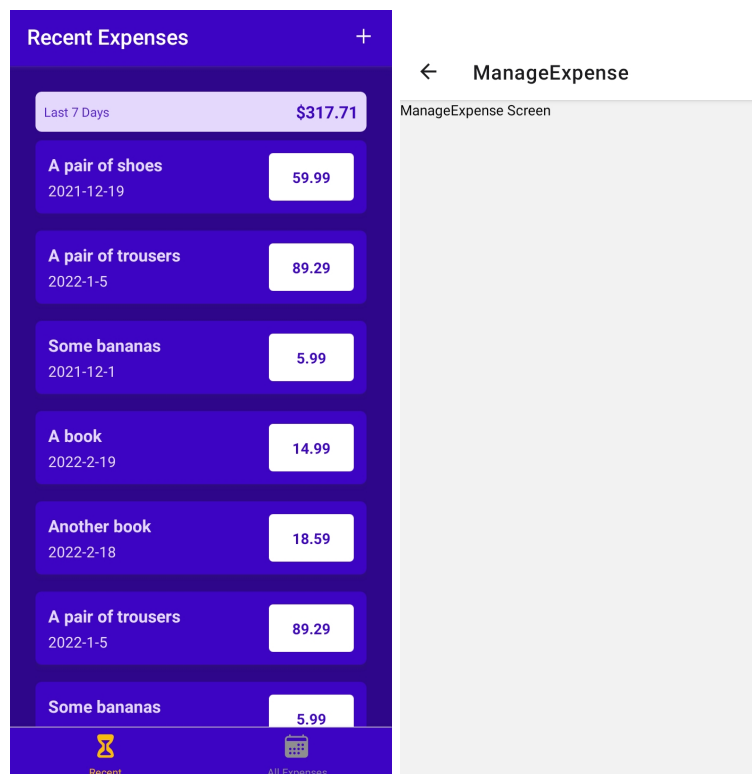
```

```

    },
    amountContainer: {
      paddingHorizontal: 12,
      paddingVertical: 4,
      backgroundColor: 'white',
      justifyContent: 'center',
      alignItems: 'center',
      borderRadius: 4,
      minWidth: 80,
    },
    amount: {
      color: GlobalStyles.colors.primary500,
      fontWeight: 'bold',
    },
  },
});

```

## Output:



## 10.supporting -different-editing-models

### In app.js

```
import { StatusBar } from 'expo-status-bar';
```

```

import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from
  '@react-navigation/native-stack';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { Ionicons } from '@expo/vector-icons';

import ManageExpense from '../screens/ManageExpense';
import RecentExpenses from '../screens/RecentExpenses';
import AllExpenses from '../screens/AllExpenses';

import { GlobalStyles } from '../constants/styles';
import IconButton from '../components/UI/IconButton';

const Stack = createNativeStackNavigator();
const BottomTabs = createBottomTabNavigator();

function ExpensesOverview() {
  return (
    <BottomTabs.Navigator
      screenOptions={({ navigation }) => ({
        headerStyle: { backgroundColor: GlobalStyles.colors.primary500 },
        headerTintColor: 'white',
        tabBarStyle: { backgroundColor: GlobalStyles.colors.primary500 },
        tabBarActiveTintColor: GlobalStyles.colors.accent500,
        headerRight: ({ tintColor }) => (
          <IconButton
            icon="add"
            size={24}
            color={tintColor}
            onPress={() => {
              navigation.navigate('ManageExpense');
            }}
          />
        ),
      })}
    >
    <BottomTabs.Screen
      name="RecentExpenses"
      component={RecentExpenses}
      options={{

```

```

        title: 'Recent Expenses',
        tabBarLabel: 'Recent',
        tabBarIcon: ({ color, size }) => (
          <Ionicons name="hourglass" size={size} color={color} />
        ),
      }}
    />
    <BottomTabs.Screen
      name="AllExpenses"
      component={AllExpenses}
      options={{
        title: 'All Expenses',
        tabBarLabel: 'All Expenses',
        tabBarIcon: ({ color, size }) => (
          <Ionicons name="calendar" size={size} color={color} />
        ),
      }}
    />
  </BottomTabs.Navigator>
);
}

export default function App() {
  return (
    <>
      <StatusBar style="auto" />
      <NavigationContainer>
        <Stack.Navigator
          screenOptions={{
            headerStyle: { backgroundColor: GlobalStyles.colors.primary500
},
            headerTintColor: 'white',
          }}
        >
          <Stack.Screen
            name="ExpensesOverview"
            component={ExpensesOverview}
            options={{ headerShown: false }}
          />
          <Stack.Screen

```

```

        name="ManageExpense"
        component={ManageExpense}
        options={{
          presentation: 'modal',
        }}
      />
    </Stack.Navigator>
  </NavigationContainer>
</>
);
}

```

## In expenseitem.js

```

import { Pressable, StyleSheet, Text, View } from 'react-native';
import { useNavigation } from '@react-navigation/native';

import { GlobalStyles } from '../../constants/styles';
import { getFormattedDate } from '../../util/date';

function ExpenseItem({ id, description, amount, date }) {
  const navigation = useNavigation();

  function expensePressHandler() {
    navigation.navigate('ManageExpense', {
      expenseId: id
    });
  }

  return (
    <Pressable
      onPress={expensePressHandler}
      style={({ pressed }) => pressed && styles.pressed}
    >
      <View style={styles.expenseItem}>
        <View>
          <Text style={[styles.textBase, styles.description]}>
            {description}
          </Text>
          <Text style={styles.textBase}>{getFormattedDate(date)}</Text>
        </View>
      </View>
    </Pressable>
  );
}

```

```

        <View style={styles.amountContainer}>
          <Text style={styles.amount}>{amount.toFixed(2)}</Text>
        </View>
      </View>
    </Pressable>
  );
}

```

```
export default ExpenseItem;
```

```

const styles = StyleSheet.create({
  pressed: {
    opacity: 0.75,
  },
  expenseItem: {
    padding: 12,
    marginVertical: 8,
    backgroundColor: GlobalStyles.colors.primary500,
    flexDirection: 'row',
    justifyContent: 'space-between',
    borderRadius: 6,
    elevation: 3,
    shadowColor: GlobalStyles.colors.gray500,
    shadowRadius: 4,
    shadowOffset: { width: 1, height: 1 },
    shadowOpacity: 0.4,
  },
  textBase: {
    color: GlobalStyles.colors.primary50,
  },
  description: {
    fontSize: 16,
    marginBottom: 4,
    fontWeight: 'bold',
  },
  amountContainer: {
    paddingHorizontal: 12,
    paddingVertical: 4,
    backgroundColor: 'white',
    justifyContent: 'center',
  },
});

```



```
      alignItems: 'center',
      borderRadius: 4,
      minWidth: 80,
    },
    amount: {
      color: GlobalStyles.colors.primary500,
      fontWeight: 'bold',
    },
  });
});
```

### In manageexpense.js(screens)

```
import { useEffect } from 'react';
import { Text } from 'react-native';

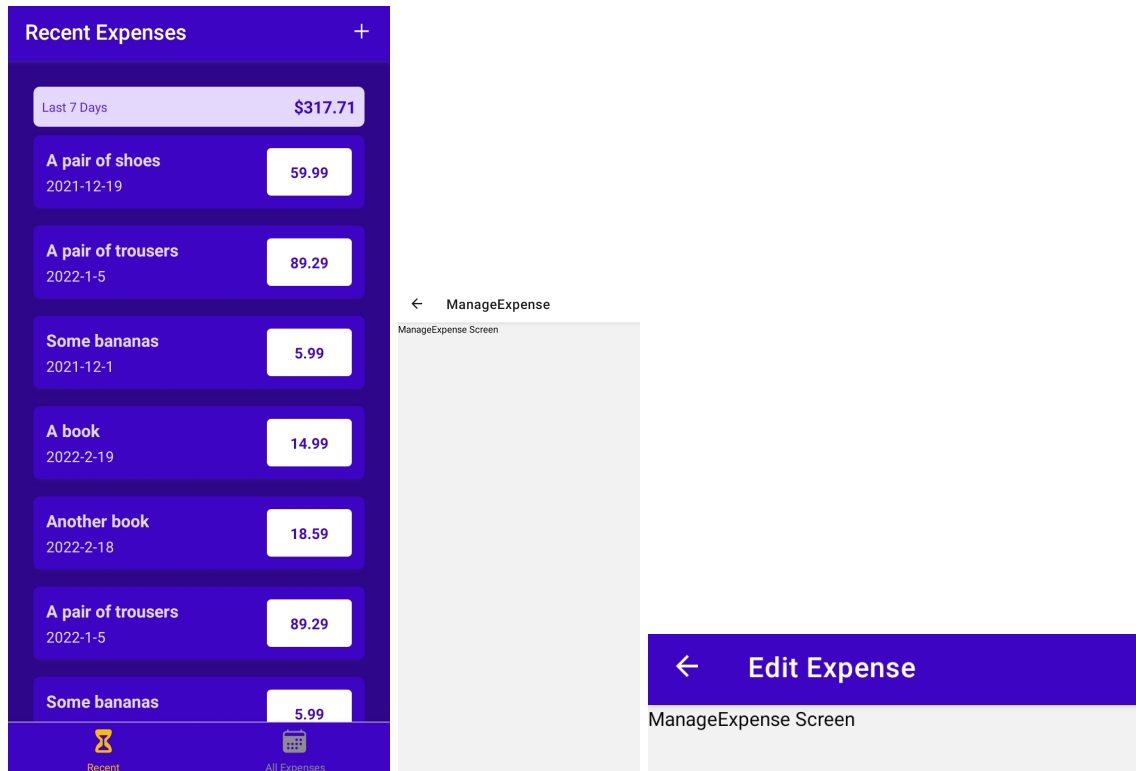
function ManageExpense({ route, navigation }) {
  const editedExpenseId = route.params?.expenseId;
  const isEditing = !!editedExpenseId;

  useEffect(() => {
    navigation.setOptions({
      title: isEditing ? 'Edit Expense' : 'Add Expense',
    });
  }, [navigation, isEditing]);

  return <Text>ManageExpense Screen</Text>;
}

export default ManageExpense;
```

Output:



## 11.adding-custom-buttons

### In manageexpense.js

```
import { useEffect } from 'react';
import { StyleSheet, View } from 'react-native';

import Button from '../components/UI/Button';
import IconButton from '../components/UI/IconButton';
import { GlobalStyles } from '../constants/styles';

function ManageExpense({ route, navigation }) {
  const editedExpenseId = route.params?.expenseId;
  const isEditing = !!editedExpenseId;

  useEffect(() => {
    navigation.setOptions({
      title: isEditing ? 'Edit Expense' : 'Add Expense',
    });
  }, [navigation, isEditing]);

  function deleteExpenseHandler() {}
}
```

```

function cancelHandler() {}

function confirmHandler() {}

return (
  <View style={styles.container}>
    <View style={styles.buttons}>
      <Button style={styles.button} mode="flat" onPress={cancelHandler}>
        Cancel
      </Button>
      <Button style={styles.button} onPress={confirmHandler}>
        {isEditing ? 'Update' : 'Add'}
      </Button>
    </View>
    {isEditing && (
      <View style={styles.deleteContainer}>
        <IconButton
          icon="trash"
          color={GlobalStyles.colors.error500}
          size={36}
          onPress={deleteExpenseHandler}
        />
      </View>
    )}
  </View>
);
}

export default ManageExpense;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 24,
    backgroundColor: GlobalStyles.colors.primary800,
  },
  buttons: {
    flexDirection: 'row',
    justifyContent: 'center',

```

```

    alignItems: 'center',
  },
  button: {
    minWidth: 120,
    marginHorizontal: 8,
  },
  deleteContainer: {
    marginTop: 16,
    paddingTop: 8,
    borderTopWidth: 2,
    borderTopColor: GlobalStyles.colors.primary200,
    alignItems: 'center',
  },
});

```

## In button.js(ui)

```

import { Pressable, StyleSheet, Text, View } from 'react-native';
import { GlobalStyles } from '../../constants/styles';

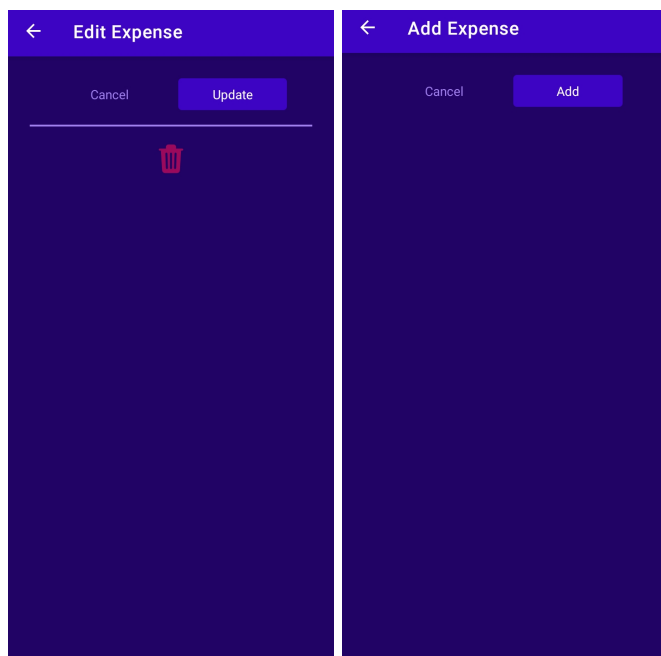
function Button({ children, onPress, mode, style }) {
  return (
    <View style={style}>
      <Pressable
        onPress={onPress}
        style={({ pressed }) => pressed && styles.pressed}
      >
        <View style={[styles.button, mode === 'flat' && styles.flat]}>
          <Text style={[styles.buttonText, mode === 'flat' &&
styles.flatText]}>
            {children}
          </Text>
        </View>
      </Pressable>
    </View>
  );
}

export default Button;

```

```
const styles = StyleSheet.create({
  button: {
    borderRadius: 4,
    padding: 8,
    backgroundColor: GlobalStyles.colors.primary500,
  },
  flat: {
    backgroundColor: 'transparent',
  },
  buttonText: {
    color: 'white',
    textAlign: 'center',
  },
  flatText: {
    color: GlobalStyles.colors.primary200,
  },
  pressed: {
    opacity: 0.75,
    backgroundColor: GlobalStyles.colors.primary100,
    borderRadius: 4,
  },
});
```

Output:



12.managing-app-wide-state

## In manageexpense.js

```
import { useEffect } from 'react';
import { StyleSheet, View } from 'react-native';

import Button from '../components/UI/Button';
import IconButton from '../components/UI/IconButton';
import { GlobalStyles } from '../constants/styles';

function ManageExpense({ route, navigation }) {
  const editedExpenseId = route.params?.expenseId;
  const isEditing = !!editedExpenseId;

  useEffect(() => {
    navigation.setOptions({
      title: isEditing ? 'Edit Expense' : 'Add Expense',
    });
  }, [navigation, isEditing]);

  function deleteExpenseHandler() {
    navigation.goBack();
  }

  function cancelHandler() {
    navigation.goBack();
  }

  function confirmHandler() {
    navigation.goBack();
  }

  return (
    <View style={styles.container}>
      <View style={styles.buttons}>
        <Button style={styles.button} mode="flat" onPress={cancelHandler}>
          Cancel
        </Button>
        <Button style={styles.button} onPress={confirmHandler}>
          {isEditing ? 'Update' : 'Add'}
        </Button>
      </View>
    </View>
  );
}
```

```

    {isEditing && (
      <View style={styles.deleteContainer}>
        <IconButton
          icon="trash"
          color={GlobalStyles.colors.error500}
          size={36}
          onPress={deleteExpenseHandler}
        />
      </View>
    )}
  </View>
);
}

export default ManageExpense;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 24,
    backgroundColor: GlobalStyles.colors.primary800,
  },
  buttons: {
    flexDirection: 'row',
    justifyContent: 'center',
    alignItems: 'center',
  },
  button: {
    minWidth: 120,
    marginHorizontal: 8,
  },
  deleteContainer: {
    marginTop: 16,
    paddingTop: 8,
    borderTopWidth: 2,
    borderTopColor: GlobalStyles.colors.primary200,
    alignItems: 'center',
  },
});

```

## 2)in expenses-context.js

```
import { createContext, useReducer } from 'react';

const DUMMY_EXPENSES = [
  {
    id: 'e1',
    description: 'A pair of shoes',
    amount: 59.99,
    date: new Date('2021-12-19'),
  },
  {
    id: 'e2',
    description: 'A pair of trousers',
    amount: 89.29,
    date: new Date('2022-01-05'),
  },
  {
    id: 'e3',
    description: 'Some bananas',
    amount: 5.99,
    date: new Date('2021-12-01'),
  },
  {
    id: 'e4',
    description: 'A book',
    amount: 14.99,
    date: new Date('2022-02-19'),
  },
  {
    id: 'e5',
    description: 'Another book',
    amount: 18.59,
    date: new Date('2022-02-18'),
  },
  {
    id: 'e6',
    description: 'A pair of trousers',
    amount: 89.29,
    date: new Date('2022-01-05'),
  },
],
```



```

    {
      id: 'e7',
      description: 'Some bananas',
      amount: 5.99,
      date: new Date('2021-12-01'),
    },
    {
      id: 'e8',
      description: 'A book',
      amount: 14.99,
      date: new Date('2022-02-19'),
    },
    {
      id: 'e9',
      description: 'Another book',
      amount: 18.59,
      date: new Date('2022-02-18'),
    },
  ],
];

export const ExpensesContext = createContext({
  expenses: [],
  addExpense: ({ description, amount, date }) => {},
  deleteExpense: (id) => {},
  updateExpense: (id, { description, amount, date }) => {},
});

function expensesReducer(state, action) {
  switch (action.type) {
    case 'ADD':
      const id = new Date().toString() + Math.random().toString();
      return [{ ...action.payload, id: id }, ...state];
    case 'UPDATE':
      const updatableExpenseIndex = state.findIndex(
        (expense) => expense.id === action.payload.id
      );
      const updatableExpense = state[updatableExpenseIndex];
      const updatedItem = { ...updatableExpense, ...action.payload.data };
      const updatedExpenses = [...state];
      updatedExpenses[updatableExpenseIndex] = updatedItem;

```

```

        return updatedExpenses;
      case 'DELETE':
        return state.filter((expense) => expense.id !== action.payload);
      default:
        return state;
    }
  }
}

function ExpensesContextProvider({ children }) {
  const [expensesState, dispatch] = useReducer(expensesReducer,
DUMMY_EXPENSES);

  function addExpense(expenseData) {
    dispatch({ type: 'ADD', payload: expenseData });
  }

  function deleteExpense(id) {
    dispatch({ type: 'DELETE', payload: id });
  }

  function updateExpense(id, expenseData) {
    dispatch({ type: 'UPDATE', payload: { id: id, data: expenseData } });
  }

  return <ExpensesContext.Provider>{children}</ExpensesContext.Provider>;
}

export default ExpensesContextProvider;

```

## 13.using context

### In expenses-context.js(store)

```

import { createContext, useReducer } from 'react';

const DUMMY_EXPENSES = [
  {
    id: 'e1',
    description: 'A pair of shoes',
    amount: 59.99,
    date: new Date('2021-12-19'),
  }
]

```

```
},
{
  id: 'e2',
  description: 'A pair of trousers',
  amount: 89.29,
  date: new Date('2022-01-05'),
},
{
  id: 'e3',
  description: 'Some bananas',
  amount: 5.99,
  date: new Date('2021-12-01'),
},
{
  id: 'e4',
  description: 'A book',
  amount: 14.99,
  date: new Date('2022-02-19'),
},
{
  id: 'e5',
  description: 'Another book',
  amount: 18.59,
  date: new Date('2022-02-18'),
},
{
  id: 'e6',
  description: 'A pair of trousers',
  amount: 89.29,
  date: new Date('2022-01-05'),
},
{
  id: 'e7',
  description: 'Some bananas',
  amount: 5.99,
  date: new Date('2021-12-01'),
},
{
  id: 'e8',
  description: 'A book',
```

```

    amount: 14.99,
    date: new Date('2022-02-19'),
  },
  {
    id: 'e9',
    description: 'Another book',
    amount: 18.59,
    date: new Date('2022-02-18'),
  },
];

export const ExpensesContext = createContext({
  expenses: [],
  addExpense: ({ description, amount, date }) => {},
  deleteExpense: (id) => {},
  updateExpense: (id, { description, amount, date }) => {},
});

function expensesReducer(state, action) {
  switch (action.type) {
    case 'ADD':
      const id = new Date().toString() + Math.random().toString();
      return [{ ...action.payload, id: id }, ...state];
    case 'UPDATE':
      const updatableExpenseIndex = state.findIndex(
        (expense) => expense.id === action.payload.id
      );
      const updatableExpense = state[updatableExpenseIndex];
      const updatedItem = { ...updatableExpense, ...action.payload.data };
      const updatedExpenses = [...state];
      updatedExpenses[updatableExpenseIndex] = updatedItem;
      return updatedExpenses;
    case 'DELETE':
      return state.filter((expense) => expense.id !== action.payload);
    default:
      return state;
  }
}

function ExpensesContextProvider({ children }) {

```

```

    const [expensesState, dispatch] = useReducer(expensesReducer,
DUMMY_EXPENSES);

    function addExpense(expenseData) {
        dispatch({ type: 'ADD', payload: expenseData });
    }

    function deleteExpense(id) {
        dispatch({ type: 'DELETE', payload: id });
    }

    function updateExpense(id, expenseData) {
        dispatch({ type: 'UPDATE', payload: { id: id, data: expenseData } });
    }

    const value = {
        expenses: expensesState,
        addExpense: addExpense,
        deleteExpense: deleteExpense,
        updateExpense: updateExpense,
    };

    return (
        <ExpensesContext.Provider value={value}>
            {children}
        </ExpensesContext.Provider>
    );
}

export default ExpensesContextProvider;

```

## 2)in app.js

```

import { StatusBar } from 'expo-status-bar';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from
 '@react-navigation/native-stack';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { Ionicons } from '@expo/vector-icons';

import ManageExpense from '../screens/ManageExpense';

```

```

import RecentExpenses from '../screens/RecentExpenses';
import AllExpenses from '../screens/AllExpenses';
import { GlobalStyles } from '../constants/styles';
import IconButton from '../components/UI/IconButton';
import ExpensesContextProvider from '../store/expenses-context';

const Stack = createNativeStackNavigator();
const BottomTabs = createBottomTabNavigator();

function ExpensesOverview() {
  return (
    <BottomTabs.Navigator
      screenOptions={({ navigation }) => ({
        headerStyle: { backgroundColor: GlobalStyles.colors.primary500 },
        headerTintColor: 'white',
        tabBarStyle: { backgroundColor: GlobalStyles.colors.primary500 },
        tabBarActiveTintColor: GlobalStyles.colors.accent500,
        headerRight: ({ tintColor }) => (
          <IconButton
            icon="add"
            size={24}
            color={tintColor}
            onPress={() => {
              navigation.navigate('ManageExpense');
            }}
          />
        ),
      })}
    >
    <BottomTabs.Screen
      name="RecentExpenses"
      component={RecentExpenses}
      options={{
        title: 'Recent Expenses',
        tabBarLabel: 'Recent',
        tabBarIcon: ({ color, size }) => (
          <Ionicons name="hourglass" size={size} color={color} />
        ),
      }}
    />

```

```

    <BottomTabs.Screen
      name="AllExpenses"
      component={AllExpenses}
      options={{
        title: 'All Expenses',
        tabBarLabel: 'All Expenses',
        tabBarIcon: ({ color, size }) => (
          <Ionicons name="calendar" size={size} color={color} />
        ),
      }}
    />
  </BottomTabs.Navigator>
);
}

export default function App() {
  return (
    <>
      <StatusBar style="auto" />
      <ExpensesContextProvider>
        <NavigationContainer>
          <Stack.Navigator
            screenOptions={{
              headerStyle: { backgroundColor:
GlobalStyles.colors.primary500 },
              headerTintColor: 'white',
            }}
          >
            <Stack.Screen
              name="ExpensesOverview"
              component={ExpensesOverview}
              options={{ headerShown: false }}
            />
            <Stack.Screen
              name="ManageExpense"
              component={ManageExpense}
              options={{
                presentation: 'modal',
              }}
            />
          </Stack.Navigator>
        </NavigationContainer>
      </ExpensesContextProvider>
    </>
  );
}

```

```

        </Stack.Navigator>
      </NavigationContainer>
    </ExpensesContextProvider>
  </>
);
}

```

### 3)in allexpenses.js

```

import { useContext } from 'react';

import ExpensesOutput from '../components/ExpensesOutput/ExpensesOutput';
import { ExpensesContext } from '../store/expenses-context';

function AllExpenses() {
  const expensesCtx = useContext(ExpensesContext);

  return (
    <ExpensesOutput expenses={expensesCtx.expenses} expensesPeriod="Total"
  />
  );
}

export default AllExpenses;

```

### 4)in recentexpenses.js

```

import { useContext } from 'react';

import ExpensesOutput from '../components/ExpensesOutput/ExpensesOutput';
import { ExpensesContext } from '../store/expenses-context';
import { getDateMinusDays } from '../util/date';

function RecentExpenses() {
  const expensesCtx = useContext(ExpensesContext);

  const recentExpenses = expensesCtx.expenses.filter((expense) => {
    const today = new Date();
    const date7DaysAgo = getDateMinusDays(today, 7);

    return (expense.date >= date7DaysAgo) && (expense.date <= today);
  });

  return (
    <ExpensesOutput expenses={recentExpenses} expensesPeriod="Recent"
  />
  );
}

export default RecentExpenses;

```



```

    });

    return (
      <ExpensesOutput expenses={recentExpenses} expensesPeriod="Last 7 Days"
    />
    );
  }
}

export default RecentExpenses;

```

## 5)in date.js(util)

```

export function getFormattedDate(date) {
  return `${date.getFullYear()}-${date.getMonth() +
1}-${date.getDate()}`;
}

export function getDateMinusDays(date, days) {
  return new Date(date.getFullYear(), date.getMonth(), date.getDate() -
days);
}

```

## 6)in manageexpenses.js

```

import { useContext, useEffect } from 'react';
import { StyleSheet, View } from 'react-native';

import Button from '../components/UI/Button';
import IconButton from '../components/UI/IconButton';
import { GlobalStyles } from '../constants/styles';
import { ExpensesContext } from '../store/expenses-context';

function ManageExpense({ route, navigation }) {
  const expensesCtx = useContext(ExpensesContext);

  const editedExpenseId = route.params?.expenseId;
  const isEditing = !!editedExpenseId;

  useEffect(() => {
    navigation.setOptions({
      title: isEditing ? 'Edit Expense' : 'Add Expense',

```

```

    });
}, [navigation, isEditing]);

function deleteExpenseHandler() {
    expensesCtx.deleteExpense(editedExpenseId);
    navigation.goBack();
}

function cancelHandler() {
    navigation.goBack();
}

function confirmHandler() {
    if (isEditing) {
        expensesCtx.updateExpense(
            editedExpenseId,
            {
                description: 'Test!!!!',
                amount: 29.99,
                date: new Date('2022-05-20'),
            }
        );
    } else {
        expensesCtx.addExpense({
            description: 'Test',
            amount: 19.99,
            date: new Date('2022-05-19'),
        });
    }
    navigation.goBack();
}

return (
    <View style={styles.container}>
        <View style={styles.buttons}>
            <Button style={styles.button} mode="flat" onPress={cancelHandler}>
                Cancel
            </Button>
            <Button style={styles.button} onPress={confirmHandler}>
                {isEditing ? 'Update' : 'Add'}
            </Button>
        </View>
    </View>
);

```

```

        </Button>
      </View>
      {isEditing && (
        <View style={styles.deleteContainer}>
          <IconButton
            icon="trash"
            color={GlobalStyles.colors.error500}
            size={36}
            onPress={deleteExpenseHandler}
          />
        </View>
      )}
    </View>
  );
}

export default ManageExpense;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 24,
    backgroundColor: GlobalStyles.colors.primary800,
  },
  buttons: {
    flexDirection: 'row',
    justifyContent: 'center',
    alignItems: 'center',
  },
  button: {
    minWidth: 120,
    marginHorizontal: 8,
  },
  deleteContainer: {
    marginTop: 16,
    paddingTop: 8,
    borderTopWidth: 2,
    borderTopColor: GlobalStyles.colors.primary200,
    alignItems: 'center',
  },
},

```

```
});
```

## 14.finished app

### In app.js

```
import { StatusBar } from 'expo-status-bar';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from
 '@react-navigation/native-stack';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { Ionicons } from '@expo/vector-icons';

import ManageExpense from '../screens/ManageExpense';
import RecentExpenses from '../screens/RecentExpenses';
import AllExpenses from '../screens/AllExpenses';
import { GlobalStyles } from '../constants/styles';
import IconButton from '../components/UI/IconButton';
import ExpensesContextProvider from '../store/expenses-context';

const Stack = createNativeStackNavigator();
const BottomTabs = createBottomTabNavigator();

function ExpensesOverview() {
  return (
    <BottomTabs.Navigator
      screenOptions={({ navigation }) => ({
        headerStyle: { backgroundColor: GlobalStyles.colors.primary500 },
        headerTintColor: 'white',
        tabBarStyle: { backgroundColor: GlobalStyles.colors.primary500 },
        tabBarActiveTintColor: GlobalStyles.colors.accent500,
        headerRight: ({ tintColor }) => (
          <IconButton
            icon="add"
            size={24}
            color={tintColor}
            onPress={() => {
              navigation.navigate('ManageExpense');
            }}
          />
        ),
      })
    />
  ),
```

```

    )))
  >
  <BottomTabs.Screen
    name="RecentExpenses"
    component={RecentExpenses}
    options={{
      title: 'Recent Expenses',
      tabBarLabel: 'Recent',
      tabBarIcon: ({ color, size }) => (
        <Ionicons name="hourglass" size={size} color={color} />
      ),
    }}
  />
  <BottomTabs.Screen
    name="AllExpenses"
    component={AllExpenses}
    options={{
      title: 'All Expenses',
      tabBarLabel: 'All Expenses',
      tabBarIcon: ({ color, size }) => (
        <Ionicons name="calendar" size={size} color={color} />
      ),
    }}
  />
</BottomTabs.Navigator>
);
}

export default function App() {
  return (
    <>
      <StatusBar style="light" />
      <ExpensesContextProvider>
        <NavigationContainer>
          <Stack.Navigator
            screenOptions={{
              headerStyle: { backgroundColor:
GlobalStyles.colors.primary500 },
              headerTintColor: 'white',
            }}

```

```

    >
    <Stack.Screen
      name="ExpensesOverview"
      component={ExpensesOverview}
      options={{ headerShown: false }}
    />
    <Stack.Screen
      name="ManageExpense"
      component={ManageExpense}
      options={{
        presentation: 'modal',
      }}
    />
  </Stack.Navigator>
</NavigationContainer>
</ExpensesContextProvider>
</>
);
}

```

## 2)in all-expenses.js

```

import { useContext } from 'react';

import ExpensesOutput from '../components/ExpensesOutput/ExpensesOutput';
import { ExpensesContext } from '../store/expenses-context';

function AllExpenses() {
  const expensesCtx = useContext(ExpensesContext);

  return (
    <ExpensesOutput
      expenses={expensesCtx.expenses}
      expensesPeriod="Total"
      fallbackText="No registered expenses found!"
    />
  );
}

export default AllExpenses;

```

### 3)in recent-expenses.js

```
import { useContext } from 'react';

import ExpensesOutput from '../components/ExpensesOutput/ExpensesOutput';
import { ExpensesContext } from '../store/expenses-context';
import { getDateMinusDays } from '../util/date';

function RecentExpenses() {
  const expensesCtx = useContext(ExpensesContext);

  const recentExpenses = expensesCtx.expenses.filter((expense) => {
    const today = new Date();
    const date7DaysAgo = getDateMinusDays(today, 7);

    return expense.date >= date7DaysAgo && expense.date <= today;
  });

  return (
    <ExpensesOutput
      expenses={recentExpenses}
      expensesPeriod="Last 7 Days"
      fallbackText="No expenses registered for the last 7 days."
    />
  );
}

export default RecentExpenses;
```

### 3)in expenseoutput.js

```
import { StyleSheet, Text, View } from 'react-native';

import { GlobalStyles } from '../../constants/styles';
import ExpensesList from './ExpensesList';
import ExpensesSummary from './ExpensesSummary';

function ExpensesOutput({ expenses, expensesPeriod, fallbackText }) {
  let content = <Text style={styles.infoText}>{fallbackText}</Text>;

  if (expenses.length > 0) {
```

```
        content = <ExpensesList expenses={expenses} />;
    }

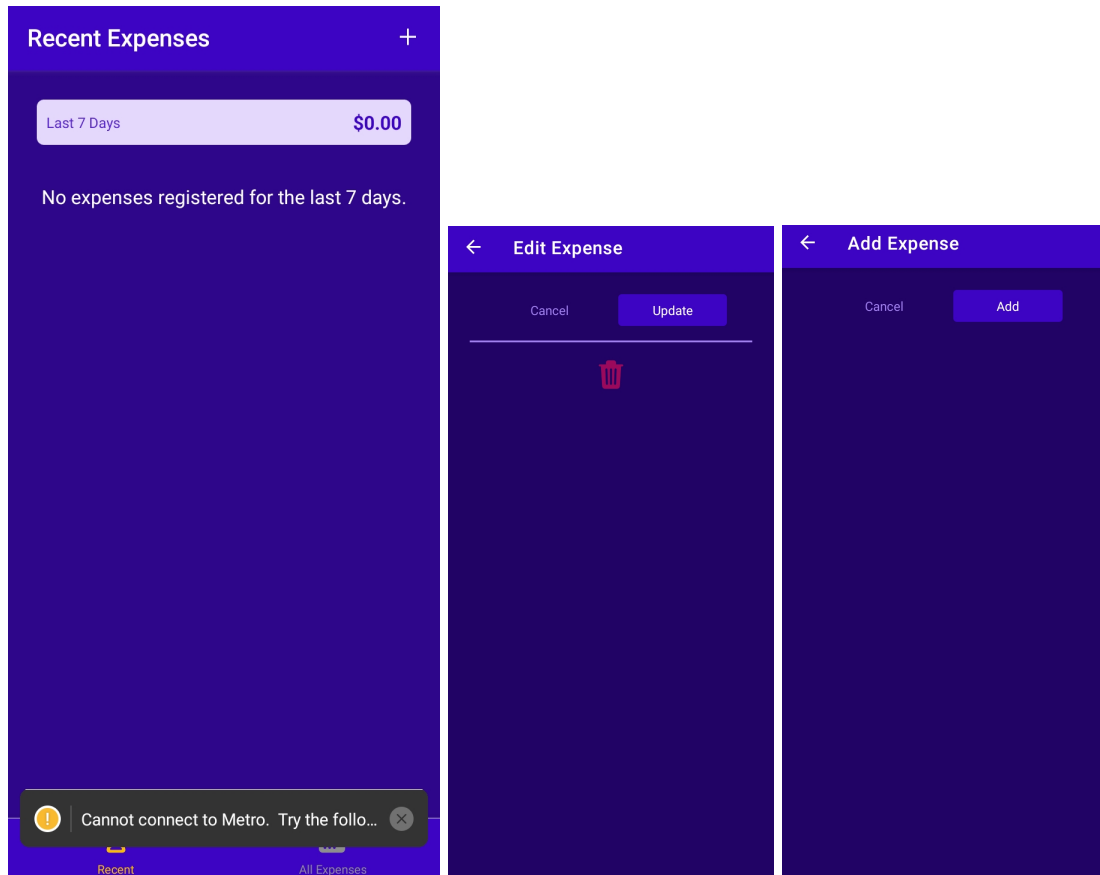
    return (
      <View style={styles.container}>
        <ExpensesSummary expenses={expenses} periodName={expensesPeriod} />
        {content}
      </View>
    );
  }
}

export default ExpensesOutput;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    paddingHorizontal: 24,
    paddingTop: 24,
    paddingBottom: 0,
    backgroundColor: GlobalStyles.colors.primary700,
  },
  infoText: {
    color: 'white',
    fontSize: 16,
    textAlign: 'center',
    marginTop: 32,
  },
});
```

Output:





## Handling user input

1) in `expenseform.js` (components/manageexpense folder)

```
import { View } from 'react-native';

import Input from '../Input';

function ExpenseForm() {
  function amountChangedHandler() {}

  return (
    <View>
      <Input
        label="Amount"
        textInputConfig={{
          keyboardType: 'decimal-pad',
          onChangeText: amountChangedHandler,
        }}
      />
```

```

    <Input
      label="Date"
      textInputConfig={{
        placeholder: 'YYYY-MM-DD',
        maxLength: 10,
        onChangeText: () => {},
      }}
    />
    <Input
      label="Description"
      textInputConfig={{
        multiline: true,
        // autoCapitalize: 'none'
        // autoComplete: false // default is true
      }}
    />
  </View>
);
}

export default ExpenseForm;

```

## 2)in input.js(components/manageexpense folder)

```

import { Text, TextInput, View } from 'react-native';

function Input({ label, textInputConfig }) {
  return (
    <View>
      <Text>{label}</Text>
      <TextInput {...textInputConfig} />
    </View>
  );
}

export default Input;

```

## 3)in manageexpense(screen folder)

```

import { useContext, useEffect } from 'react';
import { StyleSheet, TextInput, View } from 'react-native';

```

```
import ExpenseForm from '../components/ManageExpense/ExpenseForm';
import Button from '../components/UI/Button';
import IconButton from '../components/UI/IconButton';
import { GlobalStyles } from '../constants/styles';
import { ExpensesContext } from '../store/expenses-context';

function ManageExpense({ route, navigation }) {
  const expensesCtx = useContext(ExpensesContext);

  const editedExpenseId = route.params?.expenseId;
  const isEditing = !!editedExpenseId;

  useEffect(() => {
    navigation.setOptions({
      title: isEditing ? 'Edit Expense' : 'Add Expense',
    });
  }, [navigation, isEditing]);

  function deleteExpenseHandler() {
    expensesCtx.deleteExpense(editedExpenseId);
    navigation.goBack();
  }

  function cancelHandler() {
    navigation.goBack();
  }

  function confirmHandler() {
    if (isEditing) {
      expensesCtx.updateExpense(
        editedExpenseId,
        {
          description: 'Test!!!!',
          amount: 29.99,
          date: new Date('2022-05-20'),
        }
      );
    } else {
      expensesCtx.addExpense({
```

```

        description: 'Test',
        amount: 19.99,
        date: new Date('2022-05-19'),
    });
}
navigation.goBack();
}

return (
    <View style={styles.container}>
        <ExpenseForm />
        <View style={styles.buttons}>
            <Button style={styles.button} mode="flat" onPress={cancelHandler}>
                Cancel
            </Button>
            <Button style={styles.button} onPress={confirmHandler}>
                {isEditing ? 'Update' : 'Add'}
            </Button>
        </View>
        {isEditing && (
            <View style={styles.deleteContainer}>
                <IconButton
                    icon="trash"
                    color={GlobalStyles.colors.error500}
                    size={36}
                    onPress={deleteExpenseHandler}
                />
            </View>
        )}
    </View>
);
}

export default ManageExpense;

const styles = StyleSheet.create({
    container: {
        flex: 1,
        padding: 24,
        backgroundColor: GlobalStyles.colors.primary800,

```

```

    },
    buttons: {
      flexDirection: 'row',
      justifyContent: 'center',
      alignItems: 'center',
    },
    button: {
      minWidth: 120,
      marginHorizontal: 8,
    },
    deleteContainer: {
      marginTop: 16,
      paddingTop: 8,
      borderTopWidth: 2,
      borderTopColor: GlobalStyles.colors.primary200,
      alignItems: 'center',
    },
  },
});

```

## Output:

The screenshot shows a mobile application interface for adding an expense. The top bar is blue with a white back arrow and the text 'Add Expense'. The main content area is white and contains three input fields: 'Amount', 'Date' (with a placeholder 'YYYY-MM-DD'), and 'Description'. At the bottom, there are two buttons: 'Cancel' and 'Add'.

## 2)adding styling

### In input.js(manageexpense folder)

```

import { StyleSheet, Text, TextInput, View } from 'react-native';

import { GlobalStyles } from '../../constants/styles';

function Input({ label, textInputConfig }) {

```

```

const inputStyles = [styles.input];

if (textInputConfig && textInputConfig.multiline) {
  inputStyles.push(styles.inputMultiline)
}

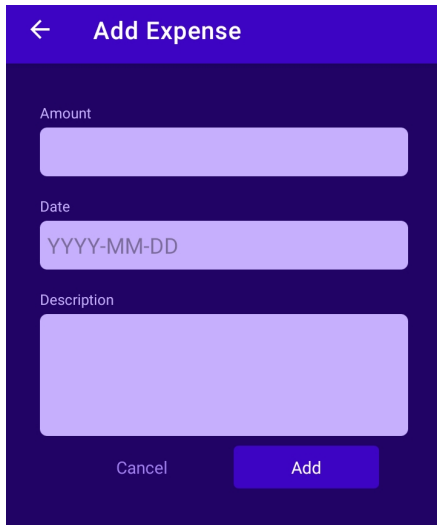
return (
  <View style={styles.inputContainer}>
    <Text style={styles.label}>{label}</Text>
    <TextInput style={inputStyles} {...textInputConfig} />
  </View>
);
}

export default Input;

const styles = StyleSheet.create({
  inputContainer: {
    marginHorizontal: 4,
    marginVertical: 8,
  },
  label: {
    fontSize: 12,
    color: GlobalStyles.colors.primary100,
    marginBottom: 4,
  },
  input: {
    backgroundColor: GlobalStyles.colors.primary100,
    color: GlobalStyles.colors.primary700,
    padding: 6,
    borderRadius: 6,
    fontSize: 18,
  },
  inputMultiline: {
    minHeight: 100,
    textAlignVertical: 'top'
  }
});

```

## Output:



## 5)handling user input generic

### In expenseform.js(manageexpense folder)

```
import { useState } from 'react';
import { StyleSheet, Text, View } from 'react-native';

import Input from './Input';

function ExpenseForm() {
  const [inputValues, setInputValues] = useState({
    amount: '',
    date: '',
    description: '',
  });

  function inputChangedHandler(inputIdentifier, enteredValue) {
    setInputValues((curInputValues) => {
      return {
        ...curInputValues,
        [inputIdentifier]: enteredValue,
      };
    });
  }

  return (
    <View style={styles.form}>
      <Text style={styles.title}>Your Expense</Text>
```

```

    <View style={styles.inputsRow}>
      <Input
        style={styles.rowInput}
        label="Amount"
        textInputConfig={{
          keyboardType: 'decimal-pad',
          onChangeText: inputChangedHandler.bind(this, 'amount'),
          value: inputValues.amount,
        }}
      />
      <Input
        style={styles.rowInput}
        label="Date"
        textInputConfig={{
          placeholder: 'YYYY-MM-DD',
          maxLength: 10,
          onChangeText: inputChangedHandler.bind(this, 'date'),
          value: inputValues.date,
        }}
      />
    </View>
    <Input
      label="Description"
      textInputConfig={{
        multiline: true,
        // autoCapitalize: 'none'
        // autoCorrect: false // default is true
        onChangeText: inputChangedHandler.bind(this, 'description'),
        value: inputValues.description,
      }}
    />
  </View>
);
}

export default ExpenseForm;

const styles = StyleSheet.create({
  form: {
    marginTop: 40,
  },
});

```



```

    },
    title: {
      fontSize: 24,
      fontWeight: 'bold',
      color: 'white',
      marginVertical: 24,
      textAlign: 'center',
    },
    inputsRow: {
      flexDirection: 'row',
      justifyContent: 'space-between',
    },
    rowInput: {
      flex: 1,
    },
  });

```

## Output:

## 6)managing form state and submission

### In manageexpense.js(screens folder)

```

import { useContext, useEffect } from 'react';
import { StyleSheet, TextInput, View } from 'react-native';

import ExpenseForm from '../components/ManageExpense/ExpenseForm';
import Button from '../components/UI/Button';
import IconButton from '../components/UI/IconButton';

```

```

import { GlobalStyles } from '../constants/styles';
import { ExpensesContext } from '../store/expenses-context';

function ManageExpense({ route, navigation }) {
  const expensesCtx = useContext(ExpensesContext);

  const editedExpenseId = route.params?.expenseId;
  const isEditing = !!editedExpenseId;

  useEffect(() => {
    navigation.setOptions({
      title: isEditing ? 'Edit Expense' : 'Add Expense',
    });
  }, [navigation, isEditing]);

  function deleteExpenseHandler() {
    expensesCtx.deleteExpense(editedExpenseId);
    navigation.goBack();
  }

  function cancelHandler() {
    navigation.goBack();
  }

  function confirmHandler(expenseData) {
    if (isEditing) {
      expensesCtx.updateExpense(editedExpenseId, expenseData);
    } else {
      expensesCtx.addExpense(expenseData);
    }
    navigation.goBack();
  }

  return (
    <View style={styles.container}>
      <ExpenseForm
        submitButtonLabel={isEditing ? 'Update' : 'Add'}
        onSubmit={confirmHandler}
        onCancel={cancelHandler}
      />
    </View>
  );
}

```

```

    {isEditing && (
      <View style={styles.deleteContainer}>
        <IconButton
          icon="trash"
          color={GlobalStyles.colors.error500}
          size={36}
          onPress={deleteExpenseHandler}
        />
      </View>
    )}
  </View>
);
}

export default ManageExpense;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 24,
    backgroundColor: GlobalStyles.colors.primary800,
  },
  deleteContainer: {
    marginTop: 16,
    paddingTop: 8,
    borderTopWidth: 2,
    borderTopColor: GlobalStyles.colors.primary200,
    alignItems: 'center',
  },
});

```

## In expenseform.js

```

import { useState } from 'react';
import { StyleSheet, Text, View } from 'react-native';

import Input from '../Input';
import Button from '../UI/Button';

function ExpenseForm({ submitButtonLabel, onCancel, onSubmit }) {
  const [inputValues, setInputValues] = useState({

```

```

    amount: '',
    date: '',
    description: '',
  });

function inputChangedHandler(inputIdentifier, enteredValue) {
  setInputValues((curInputValues) => {
    return {
      ...curInputValues,
      [inputIdentifier]: enteredValue,
    };
  });
}

function submitHandler() {
  const expenseData = {
    amount: +inputValues.amount,
    date: new Date(inputValues.date),
    description: inputValues.description
  };

  onSubmit(expenseData);
}

return (
  <View style={styles.form}>
    <Text style={styles.title}>Your Expense</Text>
    <View style={styles.inputsRow}>
      <Input
        style={styles.rowInput}
        label="Amount"
        textInputConfig={{
          keyboardType: 'decimal-pad',
          onChangeText: inputChangedHandler.bind(this, 'amount'),
          value: inputValues.amount,
        }}
      />
      <Input
        style={styles.rowInput}
        label="Date"

```

```

        textInputConfig={{
          placeholder: 'YYYY-MM-DD',
          maxLength: 10,
          onChangeText: inputChangedHandler.bind(this, 'date'),
          value: inputValues.date,
        }}
      />
    </View>
    <Input
      label="Description"
      textInputConfig={{
        multiline: true,
        // autoCapitalize: 'none'
        // autoComplete: false // default is true
        onChangeText: inputChangedHandler.bind(this, 'description'),
        value: inputValues.description,
      }}
    />
    <View style={styles.buttons}>
      <Button style={styles.button} mode="flat" onPress={onCancel}>
        Cancel
      </Button>
      <Button style={styles.button} onPress={submitHandler}>
        {submitButtonLabel}
      </Button>
    </View>
  </View>
);
}

```

```
export default ExpenseForm;
```

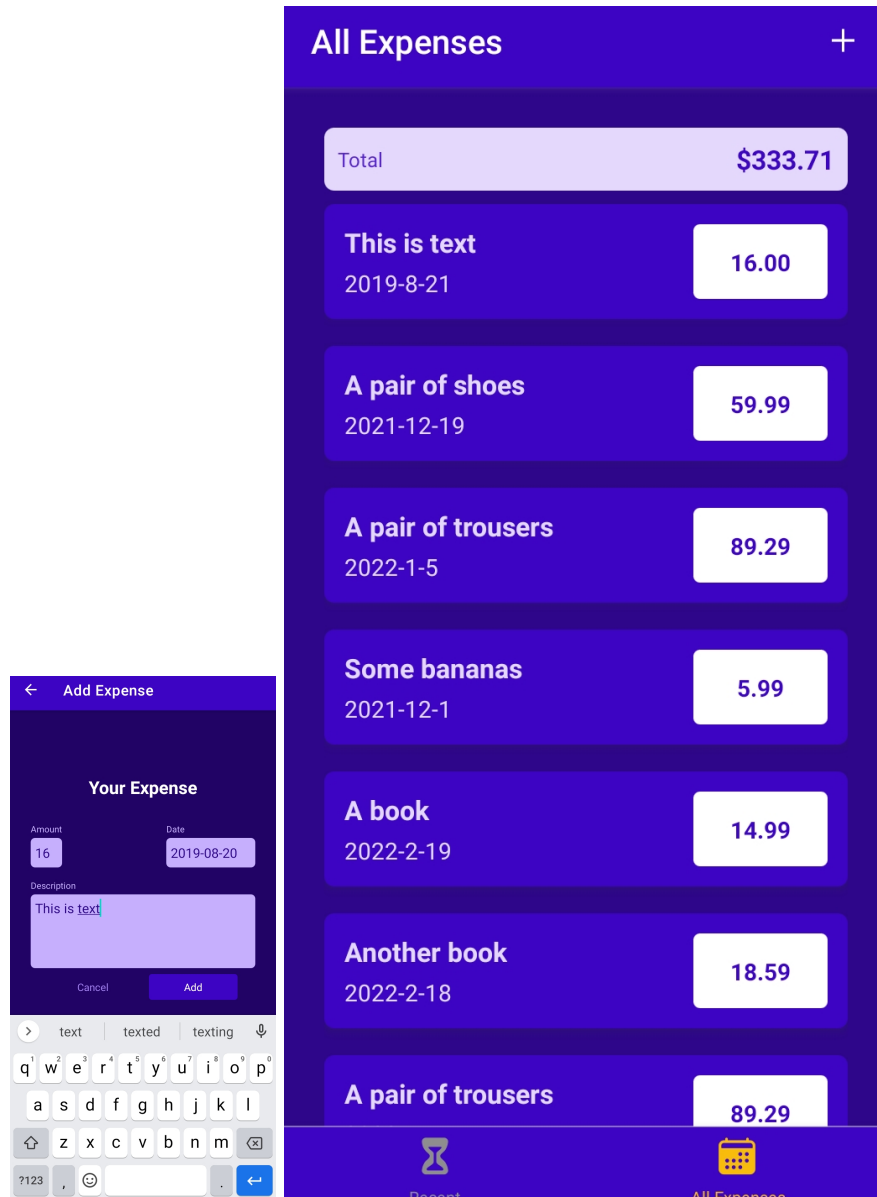
```

const styles = StyleSheet.create({
  form: {
    marginTop: 40,
  },
  title: {
    fontSize: 24,
    fontWeight: 'bold',
    color: 'white',
  },
});

```

```
        marginVertical: 24,  
        textAlign: 'center',  
      },  
      inputsRow: {  
        flexDirection: 'row',  
        justifyContent: 'space-between',  
      },  
      rowInput: {  
        flex: 1,  
      },  
      buttons: {  
        flexDirection: 'row',  
        justifyContent: 'center',  
        alignItems: 'center',  
      },  
      button: {  
        minWidth: 120,  
        marginHorizontal: 8,  
      },  
    }  
  ));
```

Output:



## 7)setting using default values(for editing checking)

### In managexpense.js

```
import { useContext, useEffect } from 'react';
import { StyleSheet, TextInput, View } from 'react-native';

import ExpenseForm from '../components/ManageExpense/ExpenseForm';
import Button from '../components/UI/Button';
import IconButton from '../components/UI/IconButton';
import { GlobalStyles } from '../constants/styles';
import { ExpensesContext } from '../store/expenses-context';
```

```

function ManageExpense({ route, navigation }) {
  const expensesCtx = useContext(ExpensesContext);

  const editedExpenseId = route.params?.expenseId;
  const isEditing = !!editedExpenseId;

  const selectedExpense = expensesCtx.expenses.find(
    (expense) => expense.id === editedExpenseId
  );

  useEffect(() => {
    navigation.setOptions({
      title: isEditing ? 'Edit Expense' : 'Add Expense',
    });
  }, [navigation, isEditing]);

  function deleteExpenseHandler() {
    expensesCtx.deleteExpense(editedExpenseId);
    navigation.goBack();
  }

  function cancelHandler() {
    navigation.goBack();
  }

  function confirmHandler(expenseData) {
    if (isEditing) {
      expensesCtx.updateExpense(editedExpenseId, expenseData);
    } else {
      expensesCtx.addExpense(expenseData);
    }
    navigation.goBack();
  }

  return (
    <View style={styles.container}>
      <ExpenseForm
        submitButtonLabel={isEditing ? 'Update' : 'Add'}
        onSubmit={confirmHandler}
      />
    </View>
  );
}

```



```

        onCancel={cancelHandler}
        defaultValues={selectedExpense}
      />
    {isEditing && (
      <View style={styles.deleteContainer}>
        <IconButton
          icon="trash"
          color={GlobalStyles.colors.error500}
          size={36}
          onPress={deleteExpenseHandler}
        />
      </View>
    )}
  </View>
);
}

export default ManageExpense;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 24,
    backgroundColor: GlobalStyles.colors.primary800,
  },
  deleteContainer: {
    marginTop: 16,
    padding: 8,
    borderTopWidth: 2,
    borderTopColor: GlobalStyles.colors.primary200,
    alignItems: 'center',
  },
});

```

## In date.js

```

export function getFormattedDate(date) {
  return date.toISOString().slice(0, 10);
}

export function getDateMinusDays(date, days) {

```

```
    return new Date(date.getFullYear(), date.getMonth(), date.getDate() -
days);
}
```

## In expenseform.js

```
import { useState } from 'react';
import { StyleSheet, Text, View } from 'react-native';

import Input from '../Input';
import Button from '../UI/Button';
import { getFormattedDate } from '../../util/date';

function ExpenseForm({ submitButtonLabel, onCancel, onSubmit,
defaultValues }) {
  const [inputValues, setInputValues] = useState({
    amount: defaultValues ? defaultValues.amount.toString() : '',
    date: defaultValues ? getFormattedDate(defaultValues.date) : '',
    description: defaultValues ? defaultValues.description : '',
  });

  function inputChangedHandler(inputIdentifier, enteredValue) {
    setInputValues((curInputValues) => {
      return {
        ...curInputValues,
        [inputIdentifier]: enteredValue,
      };
    });
  }

  function submitHandler() {
    const expenseData = {
      amount: +inputValues.amount,
      date: new Date(inputValues.date),
      description: inputValues.description,
    };

    onSubmit(expenseData);
  }

  return (
```

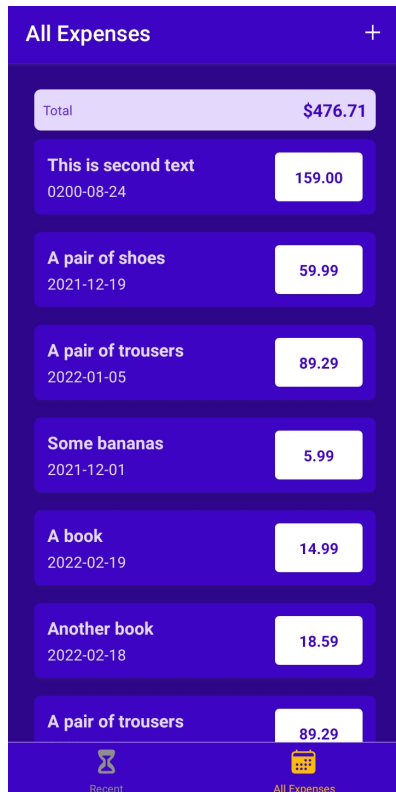
```
<View style={styles.form}>
  <Text style={styles.title}>Your Expense</Text>
  <View style={styles.inputsRow}>
    <Input
      style={styles.rowInput}
      label="Amount"
      textInputConfig={{
        keyboardType: 'decimal-pad',
        onChangeText: inputChangedHandler.bind(this, 'amount'),
        value: inputValues.amount,
      }}
    />
    <Input
      style={styles.rowInput}
      label="Date"
      textInputConfig={{
        placeholder: 'YYYY-MM-DD',
        maxLength: 10,
        onChangeText: inputChangedHandler.bind(this, 'date'),
        value: inputValues.date,
      }}
    />
  </View>
  <Input
    label="Description"
    textInputConfig={{
      multiline: true,
      // autoCapitalize: 'none'
      // autoCorrect: false // default is true
      onChangeText: inputChangedHandler.bind(this, 'description'),
      value: inputValues.description,
    }}
  />
  <View style={styles.buttons}>
    <Button style={styles.button} mode="flat" onPress={onCancel}>
      Cancel
    </Button>
    <Button style={styles.button} onPress={submitHandler}>
      {submitButtonLabel}
    </Button>
  </View>
</View>
```

```
        </View>
      </View>
    );
  }

export default ExpenseForm;

const styles = StyleSheet.create({
  form: {
    marginTop: 40,
  },
  title: {
    fontSize: 24,
    fontWeight: 'bold',
    color: 'white',
    marginVertical: 24,
    textAlign: 'center',
  },
  inputsRow: {
    flexDirection: 'row',
    justifyContent: 'space-between',
  },
  rowInput: {
    flex: 1,
  },
  buttons: {
    flexDirection: 'row',
    justifyContent: 'center',
    alignItems: 'center',
  },
  button: {
    minWidth: 120,
    marginHorizontal: 8,
  },
});
```

Output:



## 8)adding validation In expenseform.js

```
import { useState } from 'react';
import { StyleSheet, Text, View, Alert } from 'react-native';

import Input from '../Input';
import Button from '../UI/Button';
import { getFormattedDate } from '../../util/date';

function ExpenseForm({ submitButtonLabel, onCancel, onSubmit,
defaultValues }) {
  const [inputs, setInputs] = useState({
    amount: {
      value: defaultValues ? defaultValues.amount.toString() : '',
      isValid: true,
    },
    date: {
      value: defaultValues ? getFormattedDate(defaultValues.date) : '',
      isValid: true,
    },
    description: {
```

```

        value: defaultValues ? defaultValues.description : '',
        isValid: true,
    },
});

function inputChangedHandler(inputIdentifier, enteredValue) {
    setInputs((curInputs) => {
        return {
            ...curInputs,
            [inputIdentifier]: { value: enteredValue, isValid: true },
        };
    });
}

function submitHandler() {
    const expenseData = {
        amount: +inputs.amount.value,
        date: new Date(inputs.date.value),
        description: inputs.description.value,
    };

    const amountIsValid = !isNaN(expenseData.amount) && expenseData.amount > 0;
    const dateIsValid = expenseData.date.toString() !== 'Invalid Date';
    const descriptionIsValid = expenseData.description.trim().length > 0;

    if (!amountIsValid || !dateIsValid || !descriptionIsValid) {
        // Alert.alert('Invalid input', 'Please check your input values');
        setInputs((curInputs) => {
            return {
                amount: { value: curInputs.amount.value, isValid: amountIsValid },
                date: { value: curInputs.date.value, isValid: dateIsValid },
                description: {
                    value: curInputs.description.value,
                    isValid: descriptionIsValid,
                },
            };
        });
        return;
    }

```

```

    }

    onSubmit(expenseData);
  }

  const formIsValid =
    !inputs.amount.isValid ||
    !inputs.date.isValid ||
    !inputs.description.isValid;

  return (
    <View style={styles.form}>
      <Text style={styles.title}>Your Expense</Text>
      <View style={styles.inputsRow}>
        <Input
          style={styles.rowInput}
          label="Amount"
          textInputConfig={{
            keyboardType: 'decimal-pad',
            onChangeText: inputChangedHandler.bind(this, 'amount'),
            value: inputs.amount.value,
          }}
        />
        <Input
          style={styles.rowInput}
          label="Date"
          textInputConfig={{
            placeholder: 'YYYY-MM-DD',
            maxLength: 10,
            onChangeText: inputChangedHandler.bind(this, 'date'),
            value: inputs.date.value,
          }}
        />
      </View>
      <Input
        label="Description"
        textInputConfig={{
          multiline: true,
          // autoCapitalize: 'none'
          // autoComplete: false // default is true

```

```

        onChangeText: inputChangedHandler.bind(this, 'description'),
        value: inputs.description.value,
      }}
    />
    {formIsValid && (
      <Text>Invalid input values - please check your entered
data!</Text>
    )}
    <View style={styles.buttons}>
      <Button style={styles.button} mode="flat" onPress={onCancel}>
        Cancel
      </Button>
      <Button style={styles.button} onPress={submitHandler}>
        {submitButtonLabel}
      </Button>
    </View>
  </View>
);
}

```

```
export default ExpenseForm;
```

```

const styles = StyleSheet.create({
  form: {
    marginTop: 40,
  },
  title: {
    fontSize: 24,
    fontWeight: 'bold',
    color: 'white',
    marginVertical: 24,
    textAlign: 'center',
  },
  inputsRow: {
    flexDirection: 'row',
    justifyContent: 'space-between',
  },
  rowInput: {
    flex: 1,
  },
},

```



```

    buttons: {
      flexDirection: 'row',
      justifyContent: 'center',
      alignItems: 'center',
    },
    button: {
      minWidth: 120,
      marginHorizontal: 8,
    },
  },
});

```

## Output:

The screenshot shows a mobile app interface for adding an expense. The form is titled 'Your Expense' and contains three input fields: 'Amount' with the value '26', 'Date' with the value 'Fub', and a 'Description' field. Below these fields is a red error message: 'Invalid input values - please check your entered data!'. At the bottom of the form are two buttons: 'Cancel' and 'Add'.

## 9)adding-error-styling In expenseform.js

```

import { useState } from 'react';
import { StyleSheet, Text, View, Alert } from 'react-native';
import Input from './Input';
import Button from './UI/Button';
import { getFormattedDate } from '../util/date';
import { GlobalStyles } from '../constants/styles';
function ExpenseForm({ submitButtonLabel, onCancel, onSubmit,
  defaultValues }) {
  const [inputs, setInputs] = useState({
    amount: {
      value: defaultValues ? defaultValues.amount.toString() : '',
      isValid: true,
    },
    date: {
      value: defaultValues ? getFormattedDate(defaultValues.date) : '',

```

```

        isValid: true,
    },
    description: {
        value: defaultValues ? defaultValues.description : '',
        isValid: true,
    },
});

function inputChangedHandler(inputIdentifier, enteredValue) {
    setInputs((curInputs) => {
        return {
            ...curInputs,
            [inputIdentifier]: { value: enteredValue, isValid: true },
        };
    });
}

function submitHandler() {
    const expenseData = {
        amount: +inputs.amount.value,
        date: new Date(inputs.date.value),
        description: inputs.description.value,
    };

    const amountIsValid = !isNaN(expenseData.amount) && expenseData.amount > 0;
    const dateIsValid = expenseData.date.toString() !== 'Invalid Date';
    const descriptionIsValid = expenseData.description.trim().length > 0;

    if (!amountIsValid || !dateIsValid || !descriptionIsValid) {
        // Alert.alert('Invalid input', 'Please check your input values');
        setInputs((curInputs) => {
            return {
                amount: { value: curInputs.amount.value, isValid: amountIsValid },
                date: { value: curInputs.date.value, isValid: dateIsValid },
                description: {
                    value: curInputs.description.value,
                    isValid: descriptionIsValid,
                },
            },
        });
    }
}

```

```

    };
  });
  return;
}

onSubmit(expenseData);
}

const formIsValid =
  !inputs.amount.isValid ||
  !inputs.date.isValid ||
  !inputs.description.isValid;

return (
  <View style={styles.form}>
    <Text style={styles.title}>Your Expense</Text>
    <View style={styles.inputsRow}>
      <Input
        style={styles.rowInput}
        label="Amount"
        invalid={!inputs.amount.isValid}
        textInputConfig={{
          keyboardType: 'decimal-pad',
          onChangeText: inputChangedHandler.bind(this, 'amount'),
          value: inputs.amount.value,
        }}
      />
      <Input
        style={styles.rowInput}
        label="Date"
        invalid={!inputs.date.isValid}
        textInputConfig={{
          placeholder: 'YYYY-MM-DD',
          maxLength: 10,
          onChangeText: inputChangedHandler.bind(this, 'date'),
          value: inputs.date.value,
        }}
      />
    </View>
    <Input

```

```

        label="Description"
        invalid={!inputs.description.isValid}
        textInputConfig={{
            multiline: true,
            // autoCapitalize: 'none'
            // autoCorrect: false // default is true
            onChangeText: inputChangedHandler.bind(this, 'description'),
            value: inputs.description.value,
        }}
    />
    {formIsInvalid && (
        <Text style={styles.errorText}>
            Invalid input values - please check your entered data!
        </Text>
    )}
    <View style={styles.buttons}>
        <Button style={styles.button} mode="flat" onPress={onCancel}>
            Cancel
        </Button>
        <Button style={styles.button} onPress={submitHandler}>
            {submitButtonLabel}
        </Button>
    </View>
</View>
);
}

export default ExpenseForm;

const styles = StyleSheet.create({
    form: {
        marginTop: 40,
    },
    title: {
        fontSize: 24,
        fontWeight: 'bold',
        color: 'white',
        marginVertical: 24,
        textAlign: 'center',
    },
},

```

```

inputsRow: {
  flexDirection: 'row',
  justifyContent: 'space-between',
},
rowInput: {
  flex: 1,
},
errorText: {
  textAlign: 'center',
  color: GlobalStyles.colors.error500,
  margin: 8,
},
buttons: {
  flexDirection: 'row',
  justifyContent: 'center',
  alignItems: 'center',
},
button: {
  minWidth: 120,
  marginHorizontal: 8,
},
});

```

## In input.js

```

import { StyleSheet, Text, TextInput, View } from 'react-native';

import { GlobalStyles } from '../../constants/styles';

function Input({ label, invalid, style, textInputConfig }) {

  const inputStyles = [styles.input];

  if (textInputConfig && textInputConfig.multiline) {
    inputStyles.push(styles.inputMultiline)
  }

  if (invalid) {
    inputStyles.push(styles.invalidInput);
  }

```

```

    }

    return (
      <View style={[styles.inputContainer, style]}>
        <Text style={[styles.label, invalid &&
styles.invalidLabel]}>{label}</Text>
        <TextInput style={inputStyles} {...textInputConfig} />
      </View>
    );
  }
}

export default Input;

const styles = StyleSheet.create({
  inputContainer: {
    marginHorizontal: 4,
    marginVertical: 8
  },
  label: {
    fontSize: 12,
    color: GlobalStyles.colors.primary100,
    marginBottom: 4,
  },
  input: {
    backgroundColor: GlobalStyles.colors.primary100,
    color: GlobalStyles.colors.primary700,
    padding: 6,
    borderRadius: 6,
    fontSize: 18,
  },
  inputMultiline: {
    minHeight: 100,
    textAlignVertical: 'top'
  },
  invalidLabel: {
    color: GlobalStyles.colors.error500
  },
  invalidInput: {
    backgroundColor: GlobalStyles.colors.error50
  }
});

```

```
});
```

## Output:

← Add Expense

**Your Expense**

Amount: 26 Date: Fub

Description:

Invalid input values - please check your entered data!

Cancel Add

### 3)sending http requests(firebase)

#### 1)firebase gives a rest-api

a)Create a project in firebase

b)click on Real time database and start in test mode

#### 2)installing axios(for sending http request)

npm i axios

### 3)sending http requests

In http.js(util folder)

```
import axios from 'axios';

export function storeExpense(expenseData) {
  axios.post(

'https://react-native-firebase-default-rtdb.firebaseio.com/expenses.json'

  );
```

```
}
```

## In manageexpense.js(screens)

```
import { useContext, useEffect } from 'react';
import { StyleSheet, View } from 'react-native';

import ExpenseForm from '../components/ManageExpense/ExpenseForm';
import IconButton from '../components/UI/IconButton';
import { GlobalStyles } from '../constants/styles';
import { ExpensesContext } from '../store/expenses-context';
import { storeExpense } from '../util/http';

function ManageExpense({ route, navigation }) {
  const expensesCtx = useContext(ExpensesContext);

  const editedExpenseId = route.params?.expenseId;
  const isEditing = !!editedExpenseId;

  const selectedExpense = expensesCtx.expenses.find(
    (expense) => expense.id === editedExpenseId
  );

  useEffect(() => {
    navigation.setOptions({
      title: isEditing ? 'Edit Expense' : 'Add Expense',
    });
  }, [navigation, isEditing]);

  function deleteExpenseHandler() {
    expensesCtx.deleteExpense(editedExpenseId);
    navigation.goBack();
  }

  function cancelHandler() {
    navigation.goBack();
  }

  function confirmHandler(expenseData) {
    if (isEditing) {
      expensesCtx.updateExpense(editedExpenseId, expenseData);
    } else {

```



```

        storeExpense(expenseData);
        expensesCtx.addExpense(expenseData);
    }
    navigation.goBack();
}

return (
    <View style={styles.container}>
        <ExpenseForm
            submitButtonLabel={isEditing ? 'Update' : 'Add'}
            onSubmit={confirmHandler}
            onCancel={cancelHandler}
            defaultValues={selectedExpense}
        />
        {isEditing && (
            <View style={styles.deleteContainer}>
                <IconButton
                    icon="trash"
                    color={GlobalStyles.colors.error500}
                    size={36}
                    onPress={deleteExpenseHandler}
                />
            </View>
        )}
    </View>
);
}

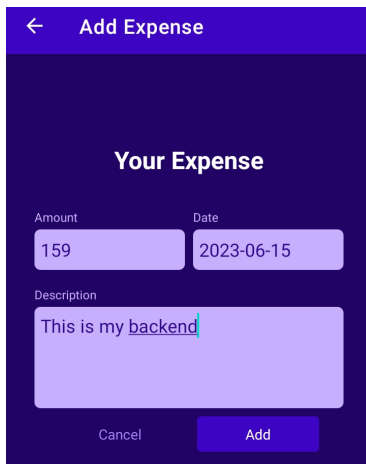
export default ManageExpense;

const styles = StyleSheet.create({
    container: {
        flex: 1,
        padding: 24,
        backgroundColor: GlobalStyles.colors.primary800,
    },
    deleteContainer: {
        marginTop: 16,
        padding: 8,
        borderTopWidth: 2,

```

```
borderTopColor: GlobalStyles.colors.primary200,
alignItems: 'center',
},
});
```

## Output:



## 3)transforming-using -fetched data In http.js

```
import axios from 'axios';

const BACKEND_URL =
'https://react-native-firebaee-default-rtdb.firebaseio.com';

export function storeExpense(expenseData) {
  axios.post(BACKEND_URL + '/expenses.json', expenseData);
}

export async function fetchExpenses() {
  const response = await axios.get(BACKEND_URL + '/expenses.json');
```

```

const expenses = [];

for (const key in response.data) {
  const expenseObj = {
    id: key,
    amount: response.data[key].amount,
    date: new Date(response.data[key].date),
    description: response.data[key].description
  };
  expenses.push(expenseObj);
}

return expenses;
}

```

### In recentexpenses.js(screens)

```

import { useContext, useEffect } from 'react';

import ExpensesOutput from '../components/ExpensesOutput/ExpensesOutput';
import { ExpensesContext } from '../store/expenses-context';
import { getDateMinusDays } from '../util/date';
import { fetchExpenses } from '../util/http';

function RecentExpenses() {
  const expensesCtx = useContext(ExpensesContext);

  useEffect(() => {
    async function getExpenses() {
      const expenses = await fetchExpenses();
      expensesCtx.setExpenses(expenses);
    }

    getExpenses();
  }, []);

  const recentExpenses = expensesCtx.expenses.filter((expense) => {
    const today = new Date();
    const date7DaysAgo = getDateMinusDays(today, 7);

```

```

    return expense.date >= date7DaysAgo && expense.date <= today;
  });

  return (
    <ExpensesOutput
      expenses={recentExpenses}
      expensesPeriod="Last 7 Days"
      fallbackText="No expenses registered for the last 7 days."
    />
  );
}

export default RecentExpenses;

```

## In expenses-context.js(store folder)

```

import { createContext, useReducer } from 'react';

export const ExpensesContext = createContext({
  expenses: [],
  addExpense: ({ description, amount, date }) => {},
  setExpenses: (expenses) => {},
  deleteExpense: (id) => {},
  updateExpense: (id, { description, amount, date }) => {},
});

function expensesReducer(state, action) {
  switch (action.type) {
    case 'ADD':
      const id = new Date().toString() + Math.random().toString();
      return [{ ...action.payload, id: id }, ...state];
    case 'SET':
      return action.payload;
    case 'UPDATE':
      const updatableExpenseIndex = state.findIndex(
        (expense) => expense.id === action.payload.id
      );
      const updatableExpense = state[updatableExpenseIndex];
      const updatedItem = { ...updatableExpense, ...action.payload.data };
      const updatedExpenses = [...state];
      updatedExpenses[updatableExpenseIndex] = updatedItem;

```

```

        return updatedExpenses;
      case 'DELETE':
        return state.filter((expense) => expense.id !== action.payload);
      default:
        return state;
    }
  }
}

function ExpensesContextProvider({ children }) {
  const [expensesState, dispatch] = useReducer(expensesReducer, []);

  function addExpense(expenseData) {
    dispatch({ type: 'ADD', payload: expenseData });
  }

  function setExpenses(expenses) {
    dispatch({ type: 'SET', payload: expenses });
  }

  function deleteExpense(id) {
    dispatch({ type: 'DELETE', payload: id });
  }

  function updateExpense(id, expenseData) {
    dispatch({ type: 'UPDATE', payload: { id: id, data: expenseData } });
  }

  const value = {
    expenses: expensesState,
    setExpenses: setExpenses,
    addExpense: addExpense,
    deleteExpense: deleteExpense,
    updateExpense: updateExpense,
  };

  return (
    <ExpensesContext.Provider value={value}>
      {children}
    </ExpensesContext.Provider>
  );
}

```

```
}  
  
export default ExpensesContextProvider;
```

#### 4)using response data from post In manageexpense.js

```
import { useContext, useEffect } from 'react';  
import { StyleSheet, View } from 'react-native';  
  
import ExpenseForm from '../components/ManageExpense/ExpenseForm';  
import IconButton from '../components/UI/IconButton';  
import { GlobalStyles } from '../constants/styles';  
import { ExpensesContext } from '../store/expenses-context';  
import { storeExpense } from '../util/http';  
  
function ManageExpense({ route, navigation }) {  
  const expensesCtx = useContext(ExpensesContext);  
  
  const editedExpenseId = route.params?.expenseId;  
  const isEditing = !!editedExpenseId;  
  
  const selectedExpense = expensesCtx.expenses.find(  
    (expense) => expense.id === editedExpenseId  
  );  
  
  useEffect(() => {  
    navigation.setOptions({  
      title: isEditing ? 'Edit Expense' : 'Add Expense',  
    });  
  }, [navigation, isEditing]);  
  
  function deleteExpenseHandler() {  
    expensesCtx.deleteExpense(editedExpenseId);  
    navigation.goBack();  
  }  
  
  function cancelHandler() {  
    navigation.goBack();  
  }  
}
```

```

    }

    async function confirmHandler(expenseData) {
      if (isEditing) {
        expensesCtx.updateExpense(editedExpenseId, expenseData);
      } else {
        const id = await storeExpense(expenseData);
        expensesCtx.addExpense({ ...expenseData, id: id });
      }
      navigation.goBack();
    }

    return (
      <View style={styles.container}>
        <ExpenseForm
          submitButtonLabel={isEditing ? 'Update' : 'Add'}
          onSubmit={confirmHandler}
          onCancel={cancelHandler}
          defaultValues={selectedExpense}
        />
        {isEditing && (
          <View style={styles.deleteContainer}>
            <IconButton
              icon="trash"
              color={GlobalStyles.colors.error500}
              size={36}
              onPress={deleteExpenseHandler}
            />
          </View>
        )}
      </View>
    );
  }
}

export default ManageExpense;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 24,
  },
});

```

```

        backgroundColor: GlobalStyles.colors.primary800,
      },
      deleteContainer: {
        marginTop: 16,
        paddingTop: 8,
        borderTopWidth: 2,
        borderTopColor: GlobalStyles.colors.primary200,
        alignItems: 'center',
      },
    },
  ));

```

## In http.js

```

import axios from 'axios';

const BACKEND_URL =
  'https://react-native-firebase-default-rtdb.firebaseio.com';

export async function storeExpense(expenseData) {
  const response = await axios.post(BACKEND_URL + '/expenses.json',
expenseData);
  const id = response.data.name;
  return id;
}

export async function fetchExpenses() {
  const response = await axios.get(BACKEND_URL + '/expenses.json');

  const expenses = [];

  for (const key in response.data) {
    const expenseObj = {
      id: key,
      amount: response.data[key].amount,
      date: new Date(response.data[key].date),
      description: response.data[key].description
    };
    expenses.push(expenseObj);
  }

  return expenses;
}

```



```
}
```

## In expense-context.js

```
import { createContext, useReducer } from 'react';

export const ExpensesContext = createContext({
  expenses: [],
  addExpense: ({ description, amount, date }) => {},
  setExpenses: (expenses) => {},
  deleteExpense: (id) => {},
  updateExpense: (id, { description, amount, date }) => {},
});

function expensesReducer(state, action) {
  switch (action.type) {
    case 'ADD':
      return [action.payload, ...state];
    case 'SET':
      const inverted = action.payload.reverse();
      return inverted;
    case 'UPDATE':
      const updatableExpenseIndex = state.findIndex(
        (expense) => expense.id === action.payload.id
      );
      const updatableExpense = state[updatableExpenseIndex];
      const updatedItem = { ...updatableExpense, ...action.payload.data };
      const updatedExpenses = [...state];
      updatedExpenses[updatableExpenseIndex] = updatedItem;
      return updatedExpenses;
    case 'DELETE':
      return state.filter((expense) => expense.id !== action.payload);
    default:
      return state;
  }
}

function ExpensesContextProvider({ children }) {
  const [expensesState, dispatch] = useReducer(expensesReducer, []);

  function addExpense(expenseData) {
    dispatch({ type: 'ADD', payload: expenseData });
  }
}
```

```

    }

    function setExpenses(expenses) {
      dispatch({ type: 'SET', payload: expenses });
    }

    function deleteExpense(id) {
      dispatch({ type: 'DELETE', payload: id });
    }

    function updateExpense(id, expenseData) {
      dispatch({ type: 'UPDATE', payload: { id: id, data: expenseData } });
    }

    const value = {
      expenses: expensesState,
      setExpenses: setExpenses,
      addExpense: addExpense,
      deleteExpense: deleteExpense,
      updateExpense: updateExpense,
    };

    return (
      <ExpensesContext.Provider value={value}>
        {children}
      </ExpensesContext.Provider>
    );
  }
}

export default ExpensesContextProvider;

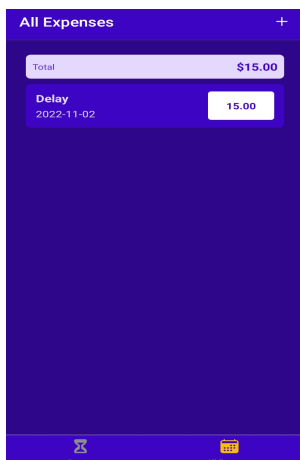
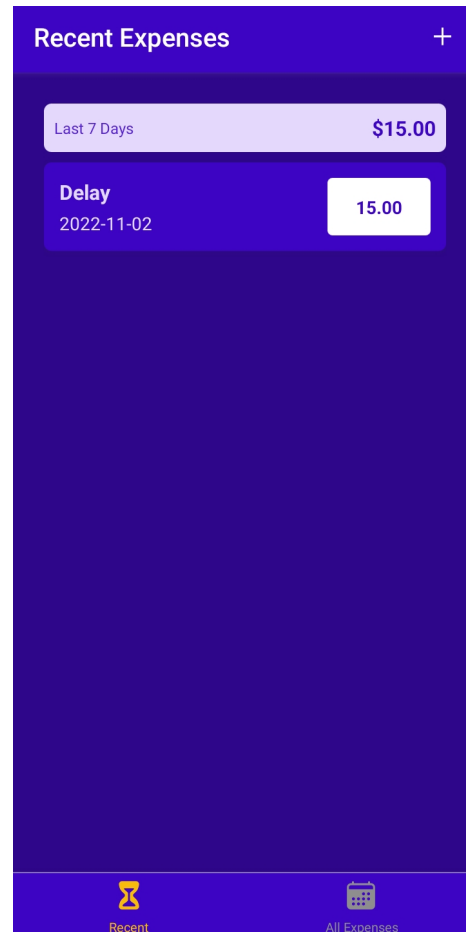
```

**Output:**

```

https://react-native-firebaee-default-rtdb.firebaseio.com/
└─ expenses
  └─ -NFrK0Q6iD1n8D-awk_b
    ├── amount: 15
    ├── date: "2022-11-02T00:00:00.000Z"
    └── description: "Delay"

```



## 5)updating and deleting In http.js

```

import axios from 'axios';

const BACKEND_URL =
  'https://react-native-firebaee-default-rtdb.firebaseio.com';

```

```

export async function storeExpense(expenseData) {
  const response = await axios.post(BACKEND_URL + '/expenses.json',
expenseData);
  const id = response.data.name;
  return id;
}

export async function fetchExpenses() {
  const response = await axios.get(BACKEND_URL + '/expenses.json');

  const expenses = [];

  for (const key in response.data) {
    const expenseObj = {
      id: key,
      amount: response.data[key].amount,
      date: new Date(response.data[key].date),
      description: response.data[key].description
    };
    expenses.push(expenseObj);
  }

  return expenses;
}

export function updateExpense(id, expenseData) {
  return axios.put(BACKEND_URL + `/expenses/${id}.json`, expenseData);
}

export function deleteExpense(id) {
  return axios.delete(BACKEND_URL + `/expenses/${id}.json`);
}

```

## In manageexpense.js

```

import { useContext, useEffect } from 'react';
import { StyleSheet, View } from 'react-native';

import ExpenseForm from '../components/ManageExpense/ExpenseForm';
import IconButton from '../components/UI/IconButton';
import { GlobalStyles } from '../constants/styles';

```

```
import { ExpensesContext } from '../store/expenses-context';
import { storeExpense, updateExpense, deleteExpense } from '../util/http';

function ManageExpense({ route, navigation }) {
  const expensesCtx = useContext(ExpensesContext);

  const editedExpenseId = route.params?.expenseId;
  const isEditing = !!editedExpenseId;

  const selectedExpense = expensesCtx.expenses.find(
    (expense) => expense.id === editedExpenseId
  );

  useEffect(() => {
    navigation.setOptions({
      title: isEditing ? 'Edit Expense' : 'Add Expense',
    });
  }, [navigation, isEditing]);

  async function deleteExpenseHandler() {
    await deleteExpense(editedExpenseId);
    expensesCtx.deleteExpense(editedExpenseId);
    navigation.goBack();
  }

  function cancelHandler() {
    navigation.goBack();
  }

  async function confirmHandler(expenseData) {
    if (isEditing) {
      expensesCtx.updateExpense(editedExpenseId, expenseData);
      await updateExpense(editedExpenseId, expenseData);
    } else {
      const id = await storeExpense(expenseData);
      expensesCtx.addExpense({ ...expenseData, id: id });
    }
    navigation.goBack();
  }
}
```

```

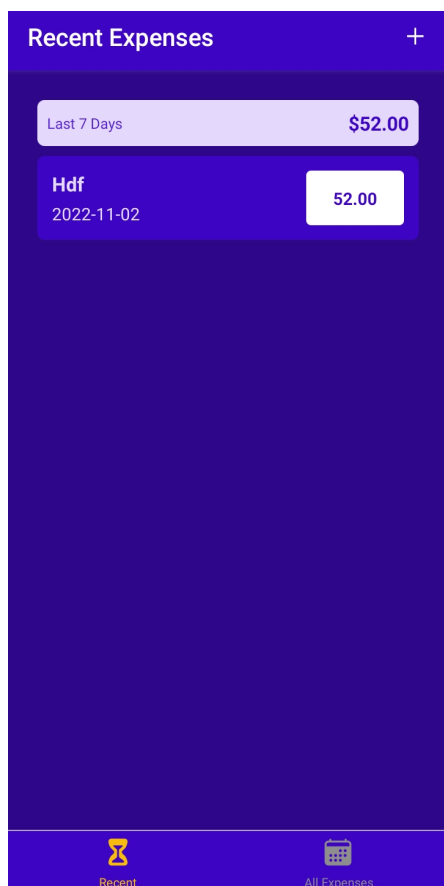
return (
  <View style={styles.container}>
    <ExpenseForm
      submitButtonLabel={isEditing ? 'Update' : 'Add'}
      onSubmit={confirmHandler}
      onCancel={cancelHandler}
      defaultValues={selectedExpense}
    />
    {isEditing && (
      <View style={styles.deleteContainer}>
        <IconButton
          icon="trash"
          color={GlobalStyles.colors.error500}
          size={36}
          onPress={deleteExpenseHandler}
        />
      </View>
    )}
  </View>
);
}

export default ManageExpense;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 24,
    backgroundColor: GlobalStyles.colors.primary800,
  },
  deleteContainer: {
    marginTop: 16,
    paddingTop: 8,
    borderTopWidth: 2,
    borderTopColor: GlobalStyles.colors.primary200,
    alignItems: 'center',
  },
});

```

## Output:



## 6.mannaging-loading-state In loading-overlay.js(ui)

```
import { View, ActivityIndicator, StyleSheet } from 'react-native';

import { GlobalStyles } from '../../constants/styles';

function LoadingOverlay() {
  return (
```

```

    <View style={styles.container}>
      <ActivityIndicator size="large" color="white" />
    </View>
  );
}

export default LoadingOverlay;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    padding: 24,
    backgroundColor: GlobalStyles.colors.primary700,
  },
});

```

## In recentexpenses.js

```

import { useContext, useEffect, useState } from 'react';

import ExpensesOutput from '../components/ExpensesOutput/ExpensesOutput';
import LoadingOverlay from '../components/UI/LoadingOverlay';
import { ExpensesContext } from '../store/expenses-context';
import { getDateMinusDays } from '../util/date';
import { fetchExpenses } from '../util/http';

function RecentExpenses() {
  const [isFetching, setIsFetching] = useState(true);
  const expensesCtx = useContext(ExpensesContext);

  useEffect(() => {
    async function getExpenses() {
      setIsFetching(true);
      const expenses = await fetchExpenses();
      setIsFetching(false);
      expensesCtx.setExpenses(expenses);
    }

    getExpenses();
  });

```



```

    }, []));

    if (isFetching) {
      return <LoadingOverlay />
    }

    const recentExpenses = expensesCtx.expenses.filter((expense) => {
      const today = new Date();
      const date7DaysAgo = getDateMinusDays(today, 7);

      return expense.date >= date7DaysAgo && expense.date <= today;
    });

    return (
      <ExpensesOutput
        expenses={recentExpenses}
        expensesPeriod="Last 7 Days"
        fallbackText="No expenses registered for the last 7 days."
      />
    );
  }
}

export default RecentExpenses;

```

## In managexpense.js

```

import { useContext, useLayoutEffect, useState } from 'react';
import { StyleSheet, View } from 'react-native';

import ExpenseForm from '../components/ManageExpense/ExpenseForm';
import IconButton from '../components/UI/IconButton';
import LoadingOverlay from '../components/UI/LoadingOverlay';
import { GlobalStyles } from '../constants/styles';
import { ExpensesContext } from '../store/expenses-context';
import { storeExpense, updateExpense, deleteExpense } from '../util/http';

function ManageExpense({ route, navigation }) {
  const [isSubmitting, setIsSubmitting] = useState(false);

```

```
const expensesCtx = useContext(ExpensesContext);

const editedExpenseId = route.params?.expenseId;
const isEditing = !!editedExpenseId;

const selectedExpense = expensesCtx.expenses.find(
  (expense) => expense.id === editedExpenseId
);

useLayoutEffect(() => {
  navigation.setOptions({
    title: isEditing ? 'Edit Expense' : 'Add Expense',
  });
}, [navigation, isEditing]);

async function deleteExpenseHandler() {
  setIsSubmitting(true);
  await deleteExpense(editedExpenseId);
  // setIsSubmitting(false);
  expensesCtx.deleteExpense(editedExpenseId);
  navigation.goBack();
}

function cancelHandler() {
  navigation.goBack();
}

async function confirmHandler(expenseData) {
  setIsSubmitting(true);
  if (isEditing) {
    expensesCtx.updateExpense(editedExpenseId, expenseData);
    await updateExpense(editedExpenseId, expenseData);
  } else {
    const id = await storeExpense(expenseData);
    expensesCtx.addExpense({ ...expenseData, id: id });
  }
  navigation.goBack();
}

if (isSubmitting) {
```

```

        return <LoadingOverlay />;
    }

    return (
        <View style={styles.container}>
            <ExpenseForm
                submitButtonLabel={isEditing ? 'Update' : 'Add'}
                onSubmit={confirmHandler}
                onCancel={cancelHandler}
                defaultValues={selectedExpense}
            />
            {isEditing && (
                <View style={styles.deleteContainer}>
                    <IconButton
                        icon="trash"
                        color={GlobalStyles.colors.error500}
                        size={36}
                        onPress={deleteExpenseHandler}
                    />
                </View>
            )}
        </View>
    );
}

```

```
export default ManageExpense;
```

```

const styles = StyleSheet.create({
    container: {
        flex: 1,
        padding: 24,
        backgroundColor: GlobalStyles.colors.primary800,
    },
    deleteContainer: {
        marginTop: 16,
        padding: 8,
        borderTopWidth: 2,
        borderTopColor: GlobalStyles.colors.primary200,
        alignItems: 'center',
    },
});

```

```
});
```

## 7)handling -request-errors

### In recentexpenses.js

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

import ExpensesOutput from '../components/ExpensesOutput/ExpensesOutput';
import ErrorOverlay from '../components/UI/ErrorOverlay';
import LoadingOverlay from '../components/UI/LoadingOverlay';
import { ExpensesContext } from '../store/expenses-context';
import { getDateMinusDays } from '../util/date';
import { fetchExpenses } from '../util/http';

function RecentExpenses() {
  const [isFetching, setIsFetching] = useState(true);
  const [error, setError] = useState();

  const expensesCtx = useContext(ExpensesContext);

  useEffect(() => {
    async function getExpenses() {
      setIsFetching(true);
      try {
        const expenses = await fetchExpenses();
        expensesCtx.setExpenses(expenses);
      } catch (error) {
        setError('Could not fetch expenses!');
      }
      setIsFetching(false);
    }

    getExpenses();
  }, []);

  if (error && !isFetching) {
    return <ErrorOverlay message={error} />;
  }

  if (isFetching) {
```

```

    return <LoadingOverlay />;
  }

  const recentExpenses = expensesCtx.expenses.filter((expense) => {
    const today = new Date();
    const date7DaysAgo = getDateMinusDays(today, 7);

    return expense.date >= date7DaysAgo && expense.date <= today;
  });

  return (
    <ExpensesOutput
      expenses={recentExpenses}
      expensesPeriod="Last 7 Days"
      fallbackText="No expenses registered for the last 7 days."
    />
  );
}

export default RecentExpenses;

```

## 2)in manageexpense.js

```

import { useContext, useLayoutEffect, useState } from 'react';
import { StyleSheet, View } from 'react-native';

import ExpenseForm from '../components/ManageExpense/ExpenseForm';
import ErrorOverlay from '../components/UI/ErrorOverlay';
import IconButton from '../components/UI/IconButton';
import LoadingOverlay from '../components/UI/LoadingOverlay';
import { GlobalStyles } from '../constants/styles';
import { ExpensesContext } from '../store/expenses-context';
import { storeExpense, updateExpense, deleteExpense } from '../util/http';

function ManageExpense({ route, navigation }) {
  const [isSubmitting, setIsSubmitting] = useState(false);
  const [error, setError] = useState();

  const expensesCtx = useContext(ExpensesContext);

  const editedExpenseId = route.params?.expenseId;

```

```
const isEditing = !!editedExpenseId;

const selectedExpense = expensesCtx.expenses.find(
  (expense) => expense.id === editedExpenseId
);

useLayoutEffect(() => {
  navigation.setOptions({
    title: isEditing ? 'Edit Expense' : 'Add Expense',
  });
}, [navigation, isEditing]);

async function deleteExpenseHandler() {
  setIsSubmitting(true);
  try {
    await deleteExpense(editedExpenseId);
    expensesCtx.deleteExpense(editedExpenseId);
    navigation.goBack();
  } catch (error) {
    setError('Could not delete expense - please try again later!');
    setIsSubmitting(false);
  }
}

function cancelHandler() {
  navigation.goBack();
}

async function confirmHandler(expenseData) {
  setIsSubmitting(true);
  try {
    if (isEditing) {
      expensesCtx.updateExpense(editedExpenseId, expenseData);
      await updateExpense(editedExpenseId, expenseData);
    } else {
      const id = await storeExpense(expenseData);
      expensesCtx.addExpense({ ...expenseData, id: id });
    }
    navigation.goBack();
  } catch (error) {
```

```

        setError('Could not save data - please try again later!');
        setIsSubmitting(false);
    }
}

if (error && !isSubmitting) {
    return <ErrorOverlay message={error} />;
}

if (isSubmitting) {
    return <LoadingOverlay />;
}

return (
    <View style={styles.container}>
        <ExpenseForm
            submitButtonLabel={isEditing ? 'Update' : 'Add'}
            onSubmit={confirmHandler}
            onCancel={cancelHandler}
            defaultValues={selectedExpense}
        />
        {isEditing && (
            <View style={styles.deleteContainer}>
                <IconButton
                    icon="trash"
                    color={GlobalStyles.colors.error500}
                    size={36}
                    onPress={deleteExpenseHandler}
                />
            </View>
        )}
    </View>
);
}

export default ManageExpense;

const styles = StyleSheet.create({
    container: {
        flex: 1,

```

```

padding: 24,
backgroundColor: GlobalStyles.colors.primary800,
},
deleteContainer: {
marginTop: 16,
paddingTop: 8,
borderTopWidth: 2,
borderTopColor: GlobalStyles.colors.primary200,
alignItems: 'center',
},
});

```

### 3)in erroroverlay.js(ui folder)

```

import { View, StyleSheet, Text } from 'react-native';

import { GlobalStyles } from '../../constants/styles';

function ErrorOverlay({ message }) {
  return (
    <View style={styles.container}>
      <Text style={[styles.text, styles.title]}>An error occurred!</Text>
      <Text style={styles.text}>{message}</Text>
    </View>
  );
}

export default ErrorOverlay;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    padding: 24,
    backgroundColor: GlobalStyles.colors.primary700,
  },
  text: {
    color: 'white',
    textAlign: 'center',
    marginBottom: 8,
  },
});

```



```
},  
title: {  
  fontSize: 20,  
  fontWeight: 'bold',  
},  
});
```

## Output:

