MAST5112 POE PART 1 AND 2

ST10450589

ATETE UMUSU

**Reflection On POE Part 1:**

Apologies for the late submission, I have chosen to restart my project, attached below will be both parts 1 and 2 of my POE with the designs included.

**Overview**

The application will have two key interfaces:

**Chef Interface:** For Christoffel to input new menu items, manage existing items, and view an overview of the menu (with average price per course).

**Guest Interface:** For clients to filter and view menu items based on the course (e.g., starters, mains, desserts).

**Project Structure:**

**React Native CLI or Expo:** Since we are building a cross-platform app, we will use React Native, which allows us to deploy to both iOS and Android. Expo simplifies the process for quicker iteration during development.

**TypeScript:** We'll use TypeScript for static type checking and a safer, more maintainable codebase.

**React Navigation**: For handling screen navigation.

**State Management:** We'll use React's useState and useEffect for state management in the first iteration (can later be optimized using context or Redux).

***Features Breakdown***

*Chef Menu Management Interface (Home screen):*

Allows the chef to enter:

**Dish Name**

**Description**

**Course Selection: Starter, Main, Dessert, etc.**

**Price**

Displays the average price of menu items, broken down by course.

Ability to add and remove menu items.

**Guest Menu Filtering Interface:**

Allows clients to filter the menu items by course (e.g., only starters).

Displays filtered items with dish name, description, and price.

**Screen Designs**

Screen 1: Home Screen (Chef’s Dashboard)

**Input Fields:** Fields for entering dish details (name, description, price, course).

**Add Menu Item Button:** Adds a dish to the menu array.

**Menu Summary:** Displays a breakdown of average price by course.

**Menu List:** Shows added items with the ability to remove an item.

Screen 2: Filter Menu (Guest View)

**A dropdown filter for courses** (e.g., starters, mains, desserts).

**Filtered Menu List:** Displays items based on the selected filter.

**CODE:**

npx create-expo-app chef-menu-app

cd chef-menu-app

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

npm install react-native-gesture-handler react-native-reanimated react-native-screens react-native-safe-area-context

npm install typescript @types/react @types/react-native --save-dev

// App.tsx

import React, { useState } from 'react';

import { NavigationContainer } from '@react-navigation/native';

import { createStackNavigator } from '@react-navigation/stack';

import { HomeScreen } from './screens/HomeScreen';

import { FilterMenuScreen } from './screens/FilterMenuScreen';

const Stack = createStackNavigator();

export default function App() {

return (

<NavigationContainer>

<Stack.Navigator>

<Stack.Screen name="Home" component={HomeScreen} />

<Stack.Screen name="FilterMenu" component={FilterMenuScreen} />

</Stack.Navigator>

</NavigationContainer>

);

}

// screens/HomeScreen.tsx

import React, { useState } from 'react';

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

interface MenuItem {

name: string;

description: string;

course: string;

price: number;

}

export const HomeScreen: React.FC = () => {

const [menuItems, setMenuItems] = useState<MenuItem[]>([]);

const [name, setName] = useState('');

const [description, setDescription] = useState('');

const [course, setCourse] = useState('');

const [price, setPrice] = useState<number>(0);

const addMenuItem = () => {

const newItem: MenuItem = { name, description, course, price };

setMenuItems([...menuItems, newItem]);

setName('');

setDescription('');

setCourse('');

setPrice(0);

};

const removeMenuItem = (index: number) => {

const updatedMenu = [...menuItems];

updatedMenu.splice(index, 1);

setMenuItems(updatedMenu);

};

const averagePriceByCourse = (course: string) => {

const filteredItems = menuItems.filter(item => item.course === course);

const total = filteredItems.reduce((sum, item) => sum + item.price, 0);

return filteredItems.length ? total / filteredItems.length : 0;

};

return (

<View>

<Text>Dish Name</Text>

<TextInput value={name} onChangeText={setName} placeholder="Enter dish name" />

<Text>Description</Text>

<TextInput value={description} onChangeText={setDescription} placeholder="Enter description" />

<Text>Course</Text>

<TextInput value={course} onChangeText={setCourse} placeholder="Enter course (e.g., starter)" />

<Text>Price</Text>

<TextInput value={price.toString()} onChangeText={(val) => setPrice(parseFloat(val))} placeholder="Enter price" keyboardType="numeric" />

<Button title="Add Menu Item" onPress={addMenuItem} />

<Text>Average Price (Starters): {averagePriceByCourse('starter')}</Text>

<Text>Average Price (Mains): {averagePriceByCourse('main')}</Text>

<FlatList

data={menuItems}

keyExtractor={(\_, index) => index.toString()}

renderItem={({ item, index }) => (

<View>

<Text>{item.name} - {item.course} - ${item.price}</Text>

<Button title="Remove" onPress={() => removeMenuItem(index)} />

</View>

)}

/>

</View>

);

};

// screens/HomeScreen.tsx

import React, { useState } from 'react';

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

interface MenuItem {

name: string;

description: string;

course: string;

price: number;

}

export const HomeScreen: React.FC = () => {

const [menuItems, setMenuItems] = useState<MenuItem[]>([]);

const [name, setName] = useState('');

const [description, setDescription] = useState('');

const [course, setCourse] = useState('');

const [price, setPrice] = useState<number>(0);

const addMenuItem = () => {

const newItem: MenuItem = { name, description, course, price };

setMenuItems([...menuItems, newItem]);

setName('');

setDescription('');

setCourse('');

setPrice(0);

};

const removeMenuItem = (index: number) => {

const updatedMenu = [...menuItems];

updatedMenu.splice(index, 1);

setMenuItems(updatedMenu);

};

const averagePriceByCourse = (course: string) => {

const filteredItems = menuItems.filter(item => item.course === course);

const total = filteredItems.reduce((sum, item) => sum + item.price, 0);

return filteredItems.length ? total / filteredItems.length : 0;

};

return (

<View>

<Text>Dish Name</Text>

<TextInput value={name} onChangeText={setName} placeholder="Enter dish name" />

<Text>Description</Text>

<TextInput value={description} onChangeText={setDescription} placeholder="Enter description" />

<Text>Course</Text>

<TextInput value={course} onChangeText={setCourse} placeholder="Enter course (e.g., starter)" />

<Text>Price</Text>

<TextInput value={price.toString()} onChangeText={(val) => setPrice(parseFloat(val))} placeholder="Enter price" keyboardType="numeric" />

<Button title="Add Menu Item" onPress={addMenuItem} />

<Text>Average Price (Starters): {averagePriceByCourse('starter')}</Text>

<Text>Average Price (Mains): {averagePriceByCourse('main')}</Text>

<FlatList

data={menuItems}

keyExtractor={(\_, index) => index.toString()}

renderItem={({ item, index }) => (

<View>

<Text>{item.name} - {item.course} - ${item.price}</Text>

<Button title="Remove" onPress={() => removeMenuItem(index)} />

</View>

)}

/>

</View>

);

};

// screens/FilterMenuScreen.tsx

import React, { useState } from 'react';

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

interface MenuItem {

name: string;

description: string;

course: string;

price: number;

}

interface Props {

route: { params: { menuItems: MenuItem[] } };

}

export const FilterMenuScreen: React.FC<Props> = ({ route }) => {

const { menuItems } = route.params;

const [selectedCourse, setSelectedCourse] = useState<string>('starter');

const filteredMenu = menuItems.filter(item => item.course === selectedCourse);

return (

<View>

<Button title="Show Starters" onPress={() => setSelectedCourse('starter')} />

<Button title="Show Mains" onPress={() => setSelectedCourse('main')} />

<FlatList

data={filteredMenu}

keyExtractor={(\_, index) => index.toString()}

renderItem={({ item }) => (

<View>

<Text>{item.name} - ${item.price}</Text>

</View>

)}

/>

</View>

**Reporting:**

UI Design:

The Home Screen has input fields for adding menu items, a button to submit a dish, and a list displaying the menu with an option to remove items.

The Filter Screen allows clients to filter items by course.

We ensure the average price is dynamically calculated for each course on the Home Screen.

State Management: useState is used to manage the list of menu items. We save these items in an array, with functionality to add or remove them dynamically.

**Navigation:** React Navigation handles screen transitions, allowing the chef to navigate between managing the menu and viewing it as the guest would.

**Mobile Application Development for Christoffel's Personalized Culinary Experience**

*Introduction*

Christoffel, a renowned private chef, requested a cross-platform mobile application to manage his dynamic menus for clients. Using React Native with TypeScript, we developed a solution that enables him to efficiently update and manage his menu, tailored to each dining experience. This report covers the overall project design, feature breakdown, and implementation details. The application consists of two primary user interfaces: one for the chef to manage the menu and one for guests to view and filter the menu items by course.

Objectives

**Chef Interface:**

Allow Christoffel to add new menu items, specifying the dish name, description, course, and price.

Dynamically calculate and display the average price of items by course (e.g., starters, mains).

Provide the ability to delete menu items.

**Guest Interface:**

Allow guests to filter and view menu items based on the course (e.g., only starters or desserts).

Project Structure

The app consists of two main screens:

Home Screen (Chef's Dashboard): This allows Christoffel to manage the menu items and view a summary.

Filter Menu Screen (Guest Interface): This provides guests the option to filter and view the menu items by course.

Application Features

1. Home Screen (Chef’s Menu Management)

Input Fields:

Chef can enter details for each dish:

Dish Name: Text input.

Description: Brief description of the dish.

Course Selection: Dropdown or text input (e.g., Starter, Main, Dessert).

Price: Numeric input.

Add Button: After filling in the inputs, the chef can click this button to add the dish to the menu array.

Menu List: Once a dish is added, it appears in the list with the option to remove any item. Each dish includes:

Dish Name

Course

Price

Remove button

Average Price by Course: Displays the average price for items categorized by course (e.g., Starters, Mains).

2. Filter Menu Screen (Guest View)

Filter by Course: Guests can select a specific course (e.g., Starters, Mains, Desserts) to view the relevant menu items.

Menu List: Filtered items are displayed with the dish name, description, and price.

State Management

Menu Array: A useState hook is used to store the menu items, and each item consists of the following properties: dish name, description, course, and price.

Adding/Removing Items: The chef can add new items to the array and remove items using the provided button.

Calculating Average Price: A helper function calculates the average price of items per course, dynamically updating whenever a new dish is added or removed.

Technology Stack

React Native: Cross-platform development for iOS and Android.

TypeScript: Ensures type safety and maintainability.

React Navigation: Handles navigation between the home screen and the filter menu screen.

Expo: Simplifies setup and development for cross-platform deployment.

Code Implementation Recap

Home Screen (Chef's Dashboard)

Input fields for entering dish details (name, description, price, course).

Button to add items to the menu array.

Dynamic display of the average price per course.

List of added items with the ability to remove them.

Filter Menu (Guest Interface)

Dropdown or buttons to filter items by course.

List view to show filtered menu items.

User Flow

Chef's Experience:

Opens the app and navigates to the home screen.

Inputs new dishes by specifying the dish name, description, course, and price.

Adds the dish to the menu, which is stored in the app's state and displayed in a list.

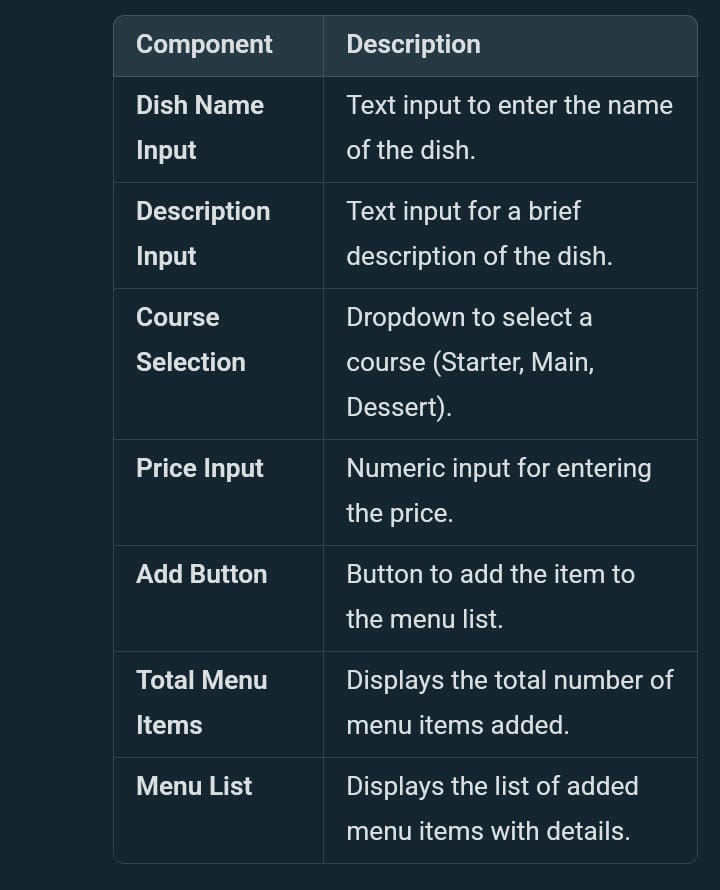
Can view average prices by course and remove any menu items as needed.

Guest's Experience:

Navigates to the filter menu screen.

Selects a course to filter the dishes (e.g., starters).

Views the filtered dishes with all relevant details (name, description, price).

**DESIGNS: (MOCKUP)**

A screenshot of a computer program

Description automatically generated

A screenshot of a menu

Description automatically generated

**Conclusion**

The mobile app is designed to streamline Christoffel’s operations, allowing him to efficiently manage his dynamic menus and enhance client experiences. The use of React Native ensures cross-platform compatibility, and TypeScript ensures that the app remains maintainable and scalable. Further iterations can introduce features such as data persistence (using local storage or a backend), user authentication, or even client feedback for improving future menus.