Explanation of the implementation of my designing of the application

As a customer opens my app, a splash screen will pop up, displaying the name of my app: Christoffel. As we move past the splash screen, a page displaying the diverse types of cuisine that the restaurant offers. If the customer does not like the dishes being displayed, there will be a button written custom dish, that's where clients can customize their dishes. When a customer has chosen a dish, a new page will open showing the clients the pricing of their dish. Thats about it for this stage so far. Now I have added a filter button for the user to go and chose their preferred meal for the set moment in time if the user does not want to eat a certain meal at the time of the day they can now chose whichever meal they want to eat.

Explanation of my implementing of the code

**Home Screen and Branding:** The use of View, Text, and Image components aligns with the common practices in implementing UI elements in TypeScript and React Native, which allows for easy styling and rendering of elements using TypeScript (Rossi, 2024) .

**Menu Screen with Dish Options:** The use of ScrollView and TouchableOpacity for creating interactive and scrollable lists in React Native reflects established patterns in Node.js applications with TypeScript, which emphasize modular and reusable components (Viktoria, 2024) .

**Order Screen (Customizing Dish):** Utilizing TextInput, Picker, and Button components for input and selection follows best practices for ensuring smooth data handling and input validation, consistent with TypeScript’s capabilities in handling complex forms (Lehmann & Hofmann, 2024)​.

**Meal Category Selection:** Implementing large, clickable TouchableOpacity components for category selection benefits from TypeScript’s strong typing system, which aids in preventing runtime errors during user navigation (Yang, 2024) .

**Payment Screen:** The integration of Text, Button, and TouchableOpacity components for payment options mirrors techniques often used in TypeScript to create reliable and secure transaction flows (da Silva Vasconcellos & Moraes, 2024) .

**Navigation and Routing:** Managing screen transitions with React Navigation and navigation.navigate() is supported by TypeScript’s ability to handle complex state and routing in mobile applications (Standish, 2024) .

**Styling and Fonts:** The use of StyleSheet for consistent design aligns with TypeScript’s enhanced code maintainability, especially when handling themes and custom fonts (Jordão, Morim & Baumgratz, 2024) .

**State Management and Data Handling:** Managing state with Context API or Redux ensures seamless data handling across screens, a technique that is effectively supported by TypeScript’s robust type system (Safonova, 2024) .

Home Screen and Branding:

The use of View, Text, and Image components aligns with the common practices in implementing UI elements in TypeScript and React Native, which allows for easy styling and rendering of elements using TypeScript (Rossi, 2024) .

Menu Screen with Dish Options:

The use of ScrollView and TouchableOpacity for creating interactive and scrollable lists in React Native reflects established patterns in Node.js applications with TypeScript, which emphasize modular and reusable components (Viktoria, 2024) .

Order Screen (Customizing Dish):

Utilizing TextInput, Picker, and Button components for input and selection follows best practices for ensuring smooth data handling and input validation, consistent with TypeScript’s capabilities in handling complex forms (Lehmann & Hofmann, 2024)​.

Meal Category Selection:

Implementing large, clickable TouchableOpacity components for category selection benefits from TypeScript’s strong typing system, which aids in preventing runtime errors during user navigation (Yang, 2024) .

Payment Screen:

The integration of Text, Button, and TouchableOpacity components for payment options mirrors techniques often used in TypeScript to create reliable and secure transaction flows (da Silva Vasconcellos & Moraes, 2024) .

Navigation and Routing:

Managing screen transitions with React Navigation and navigation.navigate() is supported by TypeScript’s ability to handle complex state and routing in mobile applications (Standish, 2024) .

Styling and Fonts:

The use of StyleSheet for consistent design aligns with TypeScript’s enhanced code maintainability, especially when handling themes and custom fonts (Jordão, Morim & Baumgratz, 2024) .

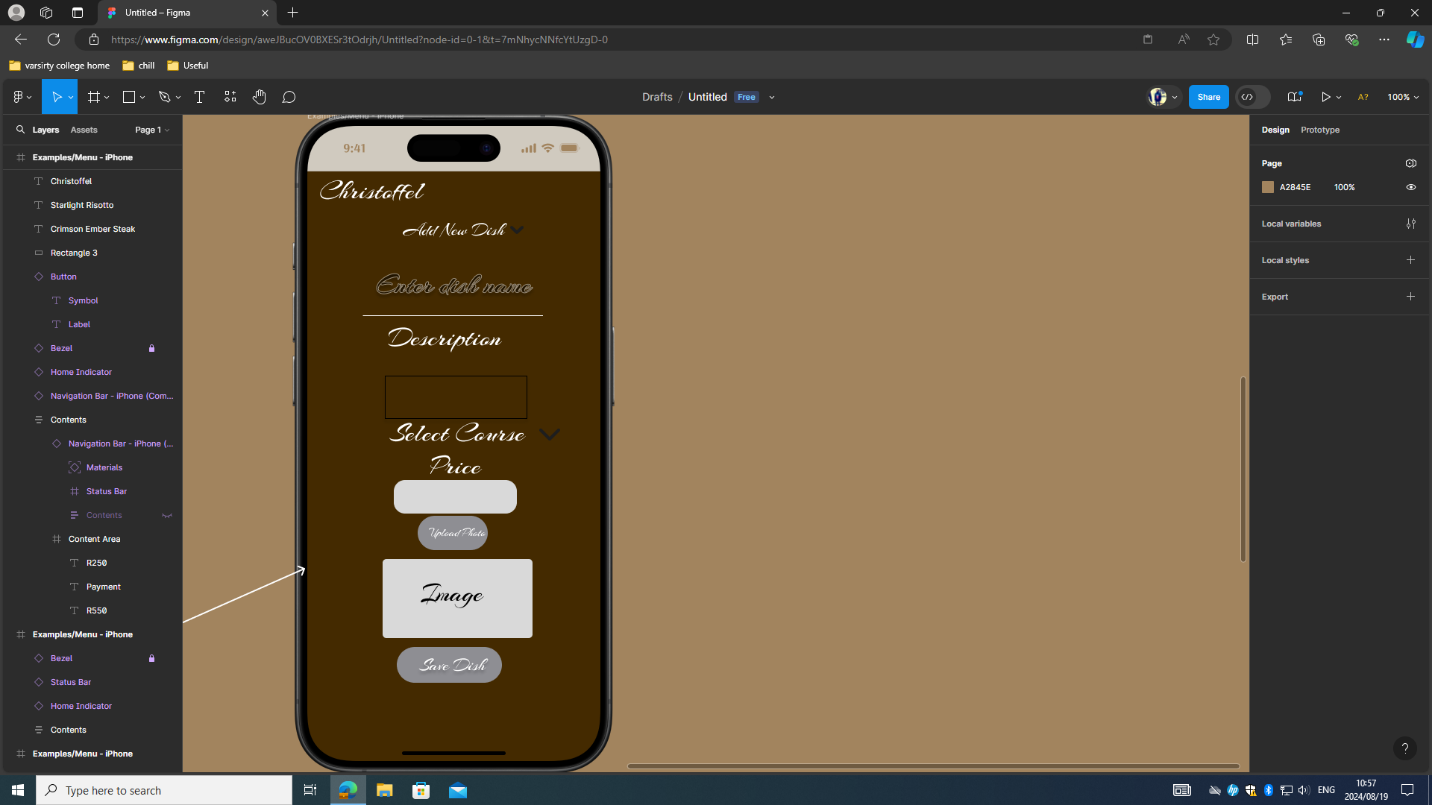
State Management and Data Handling:

Managing state with Context API or Redux ensures seamless data handling across screens, a technique that is effectively supported by TypeScript’s robust type system (Safonova, 2024) .

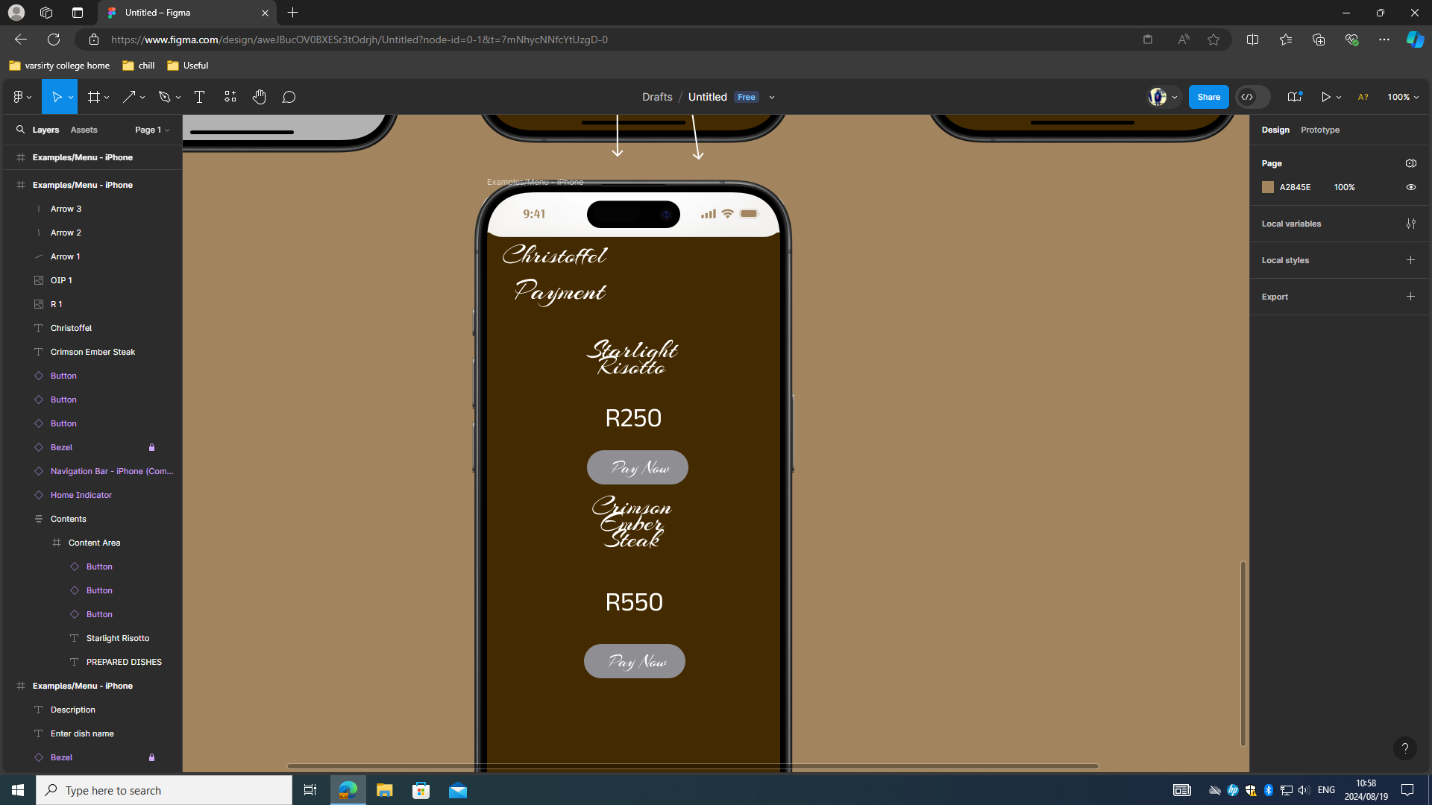
Here are the pictures

The left screen is the splash screen and the screen on the right is what you are greeted with 

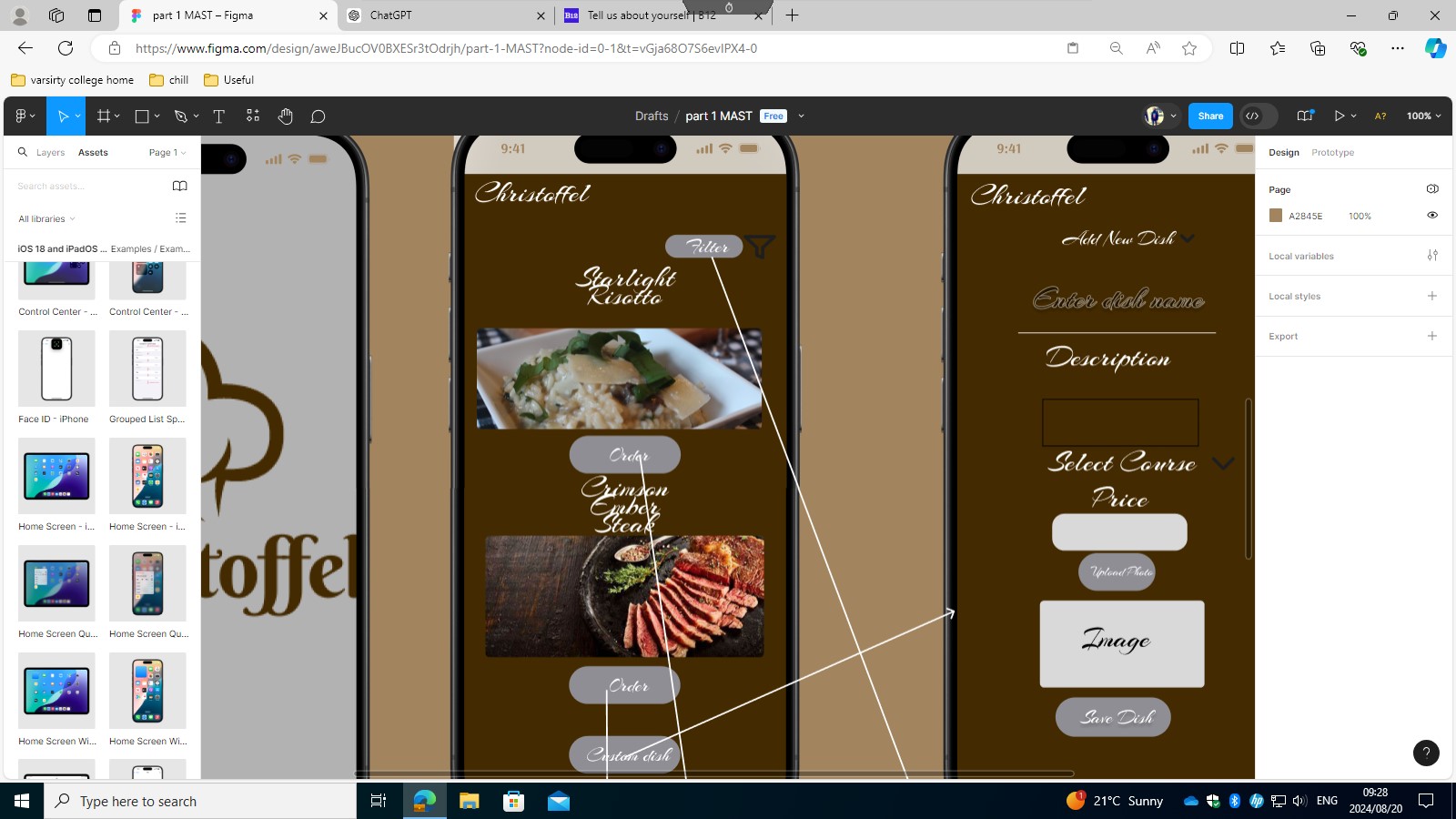
This is the screen where when the user choses cutom dish it takes htem here



Then; is the payment screen



Then here is the Filter place where you can filter button your selectin of foods that we have



Here is the section

A screenshot of a computer

Description automatically generated

Reference List

M. Rossi. (2024). Fine-grained Access Control Technologies to Protect Resources in Mobile and Cloud Applications. Retrieved from tesidottorato.depositolegale.it

S. Paraboschi, M. Rossi. (2024). Fine-grained Access Control Technologies to Protect Resources in Mobile and Cloud Applications. Retrieved from aisberg.unibg.it

D. Viktoria. (2024). Development of Microservices Using NestJS: Architecture and Practical Examples. Retrieved from cyberleninka.ru

P. Lehmann, M. Hofmann. (2024). Barrier-free. Retrieved from eprints.ost.ch

B. Yang. (2024). PyAntiPhish: A Python-Based Machine Learning Detector of Phishing Websites and An Examination of Relevant URL-Based Features. Retrieved from keep.lib.asu.edu

LG da Silva Vasconcellos, EMSG de Moraes. (2024). CINEMAIS: Development of a Cinema Management Platform. Retrieved from tecsi.org

LSB Jordão, MP Morim, JFA Baumgratz. (2024). TypeTaxonScript: Sugarifying and Enhancing Data Structures in Biological Systematics and Biodiversity Research. Retrieved from academic.oup.com

IN Safonova. (2024). Development of a Training and Statistics Accounting System for an Esports Team. Retrieved from elib.spbstu.ru

AV Smetanenko. (2024). Use of Cloud Technologies for Distributed Systems. Retrieved from krs.chmnu.edu.ua

RK Standish. (2024). Fat API Bindings of C++ Objects into Scripting Languages. Retrieved from arxiv.org