**Student Number: x14115034**

**Project name: Smart Retail** **Environment**

**Smart Retail application.**

The premise of this application is to enhance and simplify the users interaction with an online retail environment. Using 3 services of Stock Control, Customer Service and Checkout the idea is to automatically issue updates upon purchase that will let the user know that the purchase has been accepted and that stock has been bought and a list of items purchased, and inventory has been updated and with the aid of customer service the user can initiate intelligent conversations so that informed on the spot decisions can be made that will enhance their experience and shorten the time need and taken to complete a purchase. The types of RPC’s have been used unary RPC server side and bidirectional streaming RPC

The 3 services used in the application are Checkout, StockControl and CustomerService each with their own functionalities.

**Checkout**

**Service Definitions**

* The checkout service defined in the Checkout.proto uses RPC methods in the application such as AddToCart, the processCheckout and startChatSession

**Service Implementations**

* AddToCart function allows users to add products to their cart by calling the AddToCart RPC method.
* The function calculateTotal creates a bidirectional stream communication with the server using processCheckout by constructing a cart object with the products that will be purchased and sends the cart along with the card details to the server via the bidirectional stream. This allows for trust from the user as it shows secure payment processing methods.
* The startChatSession function will initialize a chat session with the server by calling the chat PRC method. The user can send a simple message to the server which acknowledges and the sever will reply.
* If the user does not enter a product to the cart but tries to checkout the client side registers an error and tells the user to enter a product.

**Naming Services**

* Simple naming has been used across the project and all names match in the proto files and the client and server files

**Error Handling**

* In checkout error handling is implemented using bidirectional communication where the errors are captured by the server sent to the client side which logs the errors and then sends the appropriate message to user. If the user enters incorrect card details the purchase will not continue and the user is asked to check their details.

The functionality of the checkout is that it serves as client side implementation which allows for interactions from the users, processing payments and communication with the server. The contribution of the checkout service to the overall application leads to a seamless friction free process that is streamlined allowing for safe and secure transactions and provides feedback for the user so they know whether the transaction has been approved or there are problems with the credit/debit card or if the user simply forgot to put products in the cart.

**Stock Control**

**Service Definitions**

* The StockControl Service defined in the Stockcontrol.proto uses RPC methods in the app. The RPC methods are addProduct which adds products and getProduct which retrieves products.
* ChatService is responsible for chat functionalities. The PRC method is chat is responsible for the bidirectional chat where messages are sent and received by the user.

**Service Implementations**

* The main function is the entry point for stock control and will loop throughout the program to ask users if they wish to continue regardless of outcomes of actions performed
* AddProduct function allows users to add product to stock by calling the addProduct RPC method.
* RetrieveProducts function allows users to retrieve the products they had added to the array that was sent to the server by calling the GetProducts RPC method.
* The startChat function is bidirectional and will initialize a chat session with the server by calling the chat PRC method. The chat also asks the suer of they wish to start a new chat with the server. It will send the message and also listen for messages from the server.
* The getUserInput function is bidirectional and will initiate a chat from the client to the server where it gets the users message and sends it to the server the server responds.
* The askForAnotherAction function will ask the users if they want to perform another actions.

**Naming Services**

* Simple naming has been used across the project and all names match in the proto files and the client and server files

**Error Handling**

* In stock control on the client side error handling is done when a user does not fill all the fields or inputs incorrect values user is asked if they would like to retry. gRPC call errors are handled within the callback functions in AddProduct and GetProduct. In the server side errors are returned by by gRPC method calls such as AddProduct and GetProduct that are managed within client side callback functions.

The functionality of stock control allows for adding and retrieving products, starting a chat with the server and handling errors. It also loops with will allow for user interaction until the user decides to leave

**CustomerService**

**Service Definitions**

The CustomerService service defined in the CustomerService.proto file uses RPC methods in the application. RPC Methods are

* AskQuestion which handles customer queries about specific aspects of the service such as delivery cost or opening times.
* Chat which facilitates bidirectional chat where messages are sent and received by the user and the server.

**Service Implementations**

1. Main Function. The main function is the entry point for customer service functionality.

It loops throughout the program to prompt users if they wish to continue regardless of the outcomes of actions performed.

* AskQuestion Function. This allows users to inquire about delivery cost or opening times by calling the AskQuestion RPC method and handles responses based on the question type in the request.
* startChat Function initializes a bidirectional chat session with the server by calling the Chat RPC method. It asks the user if they wish to start a new chat with the server and also sends messages to the server and listens for messages from the server.
* getUserInput Function. This initiates a bidirectional chat from the client to the server.

It gets the user's message, sends it to the server and waits for the server's response.

* askForAnotherAction Function. This prompts users if they want to perform another action after completing a task.

**Naming Services**

Simple and consistent naming conventions are maintained across the project, ensuring that all names match in the proto files, client and server files.

**Error handling**

It is implemented on the client-side error handling occurs when a user does not fill all fields or inputs incorrect values prompting the user if they would like to retry. gRPC call errors are managed within the callback functions such as AskQuestion and Chat.

Server-side errors are returned by gRPC method calls such as AskQuestion and Chat and are handled within client-side callback functions.

Overall Functionality:

The CustomerService functionality enables users to inquire about delivery details and opening times for customer service. It facilitates seamless communication between users and the customer service team. Error handling ensures smooth operation and user-friendly interactions throughout the application.

In summary the smart retail app integrates checkout stock control and customer service to offer users a streamlined user experience. With the use of bidirectional and unary RPC’s there is communication between the server and user to enable actions such as adding products, checkout queries and has error handling to address any issues that arise

https://github.com/01000110010/14115034\_Cullen\_Janni.git

references

rpc

van Steen, M. (2023). Distributed Systems (4th ed.). Pages 38-41, 190, 192, 205-208.