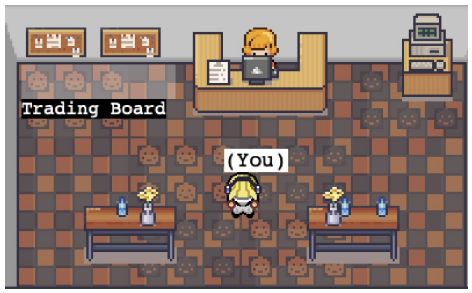
CS4530: Final Project “Commercial Areas”

Group members: Jiaying Zheng, Manh Tran, Yuezhu Lang, Krishna Singh

Our Features: Grocery Store, Inventory, and Trading Board:

After some brainstorming, we recognized the trend of people multitasking by doing online shopping while attending online classes. This sparked the idea: "Why not integrate an e-commerce platform directly into the virtual classroom?"

In our discussions, we also considered the practicality of incorporating a grocery store within the virtual environment, along with a trading board feature. This trading board would enable players to post personalized offers, fostering interactive exchanges. Additionally, implementing a player inventory feature became essential to allow users to easily monitor their possessions within the virtual space.



The screenshot on the left shows our grocery store area where players can pick grocery items, add them to cart, and check out; while the one on the right shows our trading area where players can freely interact with other players by offering and accepting trades.

Future Work:

Now that we have ve set up our e-commerce platform for groceries, we can easily adapt it for various purposes by reusing and refining the code. For example, we can quickly integrate features like in-game rewards or pet purchases by making minor adjustments to the content and naming conventions. This flexibility allows us to efficiently expand the platform's capabilities while maintaining consistency and scalability across different aspects of the virtual environment.

Links:
https://spring24-project-team-509-d2ln.onrender.com/
https://github.com/neu-cs4530/spring24-project-team-509.git

Tech Stack and Design:

To enhance our existing codebase, we integrated database support using Supabase with PostgreSQL. This decision was driven by the need to manage player inventories and balances efficiently.

Our backend architecture consists of three main areas: the grocery store, inventory, and trading board. These areas communicate with the Supabase database and push data models to the frontend controllers using the observer pattern. For security reasons, only the backend has access to the Supabase client, ensuring that the frontend is responsible only for sending valid commands.

The frontend controllers store data for rendering and send commands to the backend. They also listen for change events from the backend using the observer pattern, updating the frontend accordingly for re-rendering. This architecture ensures seamless communication between the frontend and backend while maintaining data integrity and security.

Grocery Store

ITEM NAME	PRICE	QUANTITY	
apple	3	100	Add
bacon	5	100	Add
banana	3	100	Add

Inventory

ITEM NAME	PRICE	QUANTITY
banana	3	1
bread	4	1
carrot	1	1

Potential Improvements:

To enhance our backend architecture, we can implement a microservice dedicated to communicating with Supabase. This microservice would centralize access to the database and handle requests from different backend areas. By decoupling database communication from individual backend components, we can improve scalability, maintainability, and security. Additionally, this approach allows for better management of database interactions and ensures consistency across the application.