E-Gaming Store

The E-Gaming Store is a console-based application that simulates a gaming store where users can purchase games, and administrators can manage the store's inventory. The project is implemented in C++ and uses a linked list, stack, and queue for data management.

Features

Admin

- **View All Games**: Display all available games and their prices.
- **Add Game**: Add a new game to the store's inventory.
- **Remove Game**: Remove an existing game from the inventory.
- **Set Game Price**: Update the price of an existing game.
- **View Sold Games**: Display a record of all sold games.

Customer

- **View Available Games**: Display all available games in the store.
- **Add Game to Cart**: Add a game to the customer's shopping cart.
- **Checkout**: Purchase all games in the cart and update the customer's wallet balance.
- **View Purchased Games**: Display all games purchased by the customer.
- **Add Funds**: Add funds to the customer's wallet.

Classes

Game

Represents a game with a name and price.

- **Attributes**: name, price
- **Constructors**: Default and parameterized constructors.

Node

A node in a linked list, holding a Game object and a pointer to the next node.

- **Attributes**: data, next
- **Constructor**: Initializes the node with a Game object.

LinkedList

A linked list to manage the store's game inventory.

- **Attributes**: head
- **Methods**: addGame, removeGame, setPrice, viewGames, gameExists, getGamePrice

GQueue

A queue to manage sold games for record-keeping.

- **Attributes**: front, rear
- **Methods**: enqueue, dequeue, frontValue, isEmpty, writeToFile

GStack

A stack to manage a customer's cart and purchased games.

- **Attributes**: top
- **Methods**: push, pop, topValue, isEmpty

Customer

Manages the customer's interaction with the store, including the cart and purchased games.

- **Attributes**: counter, sum, username, walletBalance, cart, gameList, purchasedGames, soldItemsQueue
- **Methods**: addToCart, checkout, addFunds, viewGames, viewPurchasedGames, saveRecord

Admin

Manages the store's game inventory.

- **Attributes**: gameList
- **Methods**: addGame, removeGame, setPrice, viewGames, viewSoldRecords

Main Menu

Options:

1. **Login as Admin**: Requires admin credentials to access the admin menu.
2. **Continue as Customer**: Access the customer menu to view games, add to cart, and checkout.
3. **Exit**: Exit the program.

Admin Menu

Options:

1. **View All Games**: Display all games with their prices.
2. **Add Game**: Add a new game to the store.
3. **Remove Game**: Remove a game from the store.
4. **Set Game Price**: Update the price of an existing game.
5. **View Sold Games**: View the record of sold games.
6. **Log Out**: Return to the main menu.

Customer Menu

Options:

1. **View Available Games**: Display all games in the store.
2. **Add Game to Cart**: Add a game to the cart.
3. **Checkout**: Purchase all games in the cart.
4. **View Purchased Games**: Display all purchased games.
5. **Add Funds**: Add funds to the wallet.
6. **Sign Out**: Return to the main menu.

Usage

To run the application, compile and execute the main.cpp file. Follow the on-screen prompts to navigate through the menus and perform actions as either an admin or a customer.

Notes

- **Admin credentials are hardcoded:**
    - **Email:** sarmadhassan27@gmail.com or admin@gmail.com
    - **Pin Code:** 2022
- The sold games record is saved to **sold_items.txt.**