Implemented
Partially Implemented

Function Name	Function Description
Utilities::splitString	Splits a string into a vector of strings based on a specified delimiter.
Utilities::trim	Trims leading and trailing whitespace from a string.
Utilities::isNumber	Checks if a given string represents a valid number by ensuring all characters are digits.
Utilities::stringToFloat	Converts a string to a floating-point number and handles possible exceptions.
UserSession::UserSession()	Default constructor initializes the session state as not logged in and sets the user type to None.
UserSession::login	Prompts the user for a username and attempts to authenticate them by checking against stored user accounts.
UserSession::logout	Logs out the current user, clearing their session data, and returns a logout message.
UserSession::isLoggedIn	Returns a boolean indicating whether a user is currently logged in.
UserSession::getCurrentUser	Returns the username of the currently logged-in user.
UserSession::getCurrentUserType	Returns the type of the currently logged-in user.
UserAccounts::UserAccounts()	Default constructor that initializes the file path to a default value for user accounts.
UserAccounts::UserAccounts(const std::string&)	Constructor with a file path parameter for specifying a custom path to the user accounts file.
UserAccounts::loadAccounts	Loads user account data from the specified file.

UserAccounts::getAllAccountsInfo	Retrieves information for all user accounts in a formatted string vector.
UserAccounts::createUser	Adds a new user to the system with the specified username, user type, and initial credit.
UserAccounts::deleteUser	Deletes a user from the system based on the username.
UserAccounts::saveAccounts	Saves the current list of user accounts to the file specified by accountsFilePath.
UserAccounts::userTypeToString	Converts a UserType enum to its corresponding string representation.
UserAccounts::userExists	Checks if a user exists in the system based on the username.
UserAccounts::addCredit	Adds credit to a user's account, identified by username, in the amount specified.
TransactionProcessing::TransactionProcessing	Constructor initializes the class with references to the UserAccounts and GameInventory.
TransactionProcessing::processTransaction	Main method to process transactions based on a transaction code and arguments.
TransactionProcessing::processSellTransaction	Processes a sell transaction, allowing a user to list a game for sale.
TransactionProcessing::processBuyTransaction	Processes a buy transaction, allowing a user to purchase a game.
TransactionProcessing::processRefundTransaction	Processes a refund transaction between two users.
TransactionProcessing::processAddCreditTransaction	Processes a transaction to add credit to a user's account.
TransactionProcessing::processRefund	Specific method for processing refunds not triggered by the main transaction processing method.
GameInventory::GameInventory()	Default constructor that initializes the inventory file path to an empty string.

GameInventory::GameInventory(const std::string&)	Constructor that takes a file path as an argument and loads the game inventory from this file.
GameInventory::loadInventory	Loads the game inventory from the specified file.
GameInventory::addGame	Adds a new game to the inventory, checking for duplicates and validating the game name length and price.
GameInventory::saveInventory	Saves the current game inventory to the file.
FileIO::readUserAccounts	Reads user account data from a specified file and stores it in a vector of UserAccount structures.
FileIO::readGameInventory	Reads game inventory data from a specified file and stores it in a vector of Game structures.
FileIO::parseUserAccountLine	Parses a line from the user accounts file and fills a UserAccount structure with the data.
FileIO::parseGameLine	Parses a line from the game inventory file and fills a Game structure with the data.
FileIO::writeTransactionLog	Writes transaction log data to a specified file, appending each transaction to the file.
AdminActions::AdminActions	Constructor for the AdminActions class, taking references to UserAccounts and TransactionProcessing.
AdminActions::createUser	Creates a new user with specified username, userType, and initial credit, checking for duplicates.
AdminActions::deleteUser	Deletes an existing user from the system by their username, checking if the user exists before attempting deletion.
AdminActions::issueRefund	Issues a refund from one user's account to another by their usernames and the specified amount.
AdminActions::addCredit	Adds credit to a user's account by their username and the specified amount.

AdminActions::displayAllAccounts	Displays all user accounts in the system, retrieving user account information and displaying it.
displayMenu	Displays the main menu to the console, listing all available actions to the user.
main	The entry point of the application, initializing components and handling the main loop for menu selection and action execution.