Frontend Design Document

CSCI 3060U

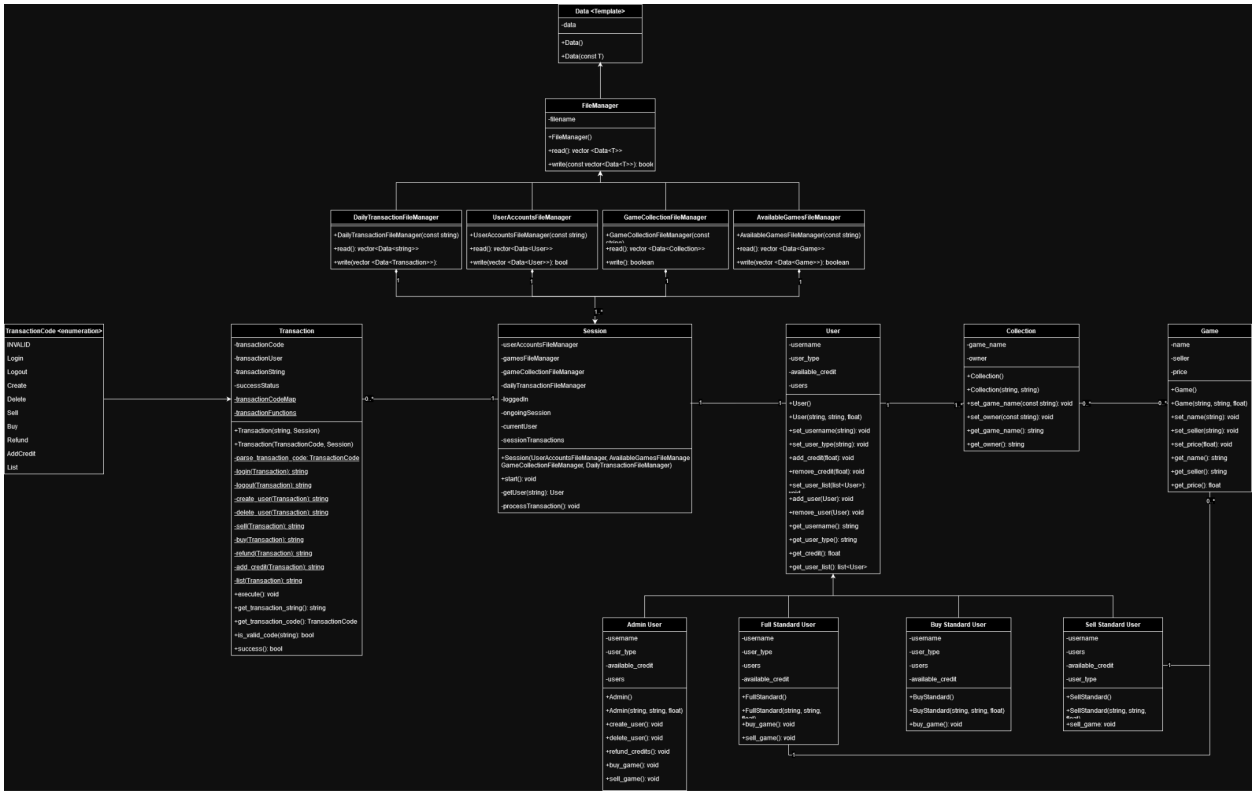
Dr. Michael Miljanovic

Levi Willms, Daniel Hinbest, Syed Rizvi, Raje Singh

Purpose of this document:
The purpose of this document is to show the internal and external designs of the frontend architecture. Details regarding the specific coding practices and standards can be found in the GitHub repository in the README.md file.

In this document will be found a class diagram and a table explaining functionality and purpose of each class. Some notable considerations for the frontend rapid prototype are that a focus has been made on completing the architectural requirements and framework. Thus, specific working functionality of some transactions are left out to allow for the rapid development. As testing in the next phase begins the remaining features will be completed as well as additional bugs found.

Class Diagram - Clearer version provided on GitHub repository



Class: Collection
Description: This class is designed to represent a collection of games that would be available to a user.

Method	**Description**
Collection()	**Parameters:** none **Return type:** N/A A default constructor to create a new Collection object with default attributes
Collection(string, string)	**Parameters:** string gameName, string owner **Return type:** N/A A constructor creating a new Collection object with specified values
get_game_name()	**Parameters:** none **Return type:** string An accessor method to return the game name
get_owner()	**Parameters:** none **Return type:** string An accessor method that returns the name of the game's owner

set_game_name(string)	**Parameters:** string gameName **Return type:** void A mutator method that assigns a value to the gameName attribute
set_owner(string)	**Parameters:** string owner **Return type:** void A mutator method that assigns a value to the owner attribute

Class: Data
Description: A generic class that represents data types in a file. It is used to store data read or written to/from a file

Method	**Description**
Data()	**Parameters**: none **Return type:** N/A A default constructor to create a new Data object with default attributes
Data(string, string)	**Parameters:** string gameName, string owner **Return type:** N/A A constructor creating a new Data object with specified values
getData()	**Parameters:** none **Return type:** T A mutator method that returns the data type from a template

Class: File Manager
Description: An abstract class for file management with virtual functions to read or write from a file

Method	**Description**
FileManager()	**Parameters**: none **Return type:** N/A A destructor to destroy a FileManager object.
read()	**Parameters:** none **Return type:** vector<Data<T>> data A read method that reads data from a file and returns data objects with the data read from the file
write()	**Parameters:** vector<Data<T>> data **Return type:** bool Writes data to the file and returns true if successful, or false if unsuccessful

Class: AvailableGamesFileManager (Base: FileManager)

Description: Inherited from FileManager, AvailableGamesFileManager is a file manager that is designed for file management for the available games

Method	**Description**
AvailableGamesFileManager()	**Parameters**: none **Return type:** N/A A constructor to create a new AvailableGamesFileManager object.
read()	**Parameters:** none **Return type:** vector<Data<Game>> data A read method that reads Game data from a file and returns Game data objects with the Game data read from the file
write()	**Parameters:** vector<Data<Game>> data **Return type:** bool Writes Game data to the file and returns true if successful, or false if unsuccessful

Class: DailyTransactionFileManager (Base: FileManager)
Description: Inherited from FileManager, DailyTransactionFileManager is a file manager that is designed for file management for the daily transaction file

Method	**Description**
DailyTransactionFileManager ()	**Parameters**: none **Return type:** N/A A constructor to create a new DailyTransactionFileManager object.
read()	**Parameters:** none **Return type:** vector<Data<string>> data A read method that reads transaction data from a file and returns transaction data objects with the transaction data read from the file
write()	**Parameters:** vector<Data<string>> data **Return type:** bool Writes transaction data to the file and returns true if successful, or false if unsuccessful

Class: GameCollectionFileManager (Base: FileManager)
Description: Inherited from FileManager, GameCollectionFileManager is a file manager that is designed for file management for the game collection file

Method	**Description**
GameCollectionFileManager ()	**Parameters**: none **Return type:** N/A A constructor to create a new GameCollectionFileManager object.

read()	**Parameters:** none **Return type:** vector<Data<Collection>> data A read method that reads game collection data from a file and returns game collection data objects with the game collection data read from the file
write()	**Parameters:** vector<Data<Collection>> data **Return type:** bool Writes game collection data to the file and returns true if successful, or false if unsuccessful

Class: UserAccountsFileManager (Base: FileManager)
Description: Inherited from FileManager, GameCollectionFileManager is a file manager that is designed for file management for the game collection file

Method	**Description**
UserAccountsFileManager()	**Parameters**: none **Return type:** N/A A constructor to create a new UserAccountsFileManager object.
read()	**Parameters:** none **Return type:** vector<Data<User>> data A read method that reads User data from a file and returns User data objects with the User data read from the file
write()	**Parameters:** vector<Data<User>> data **Return type:** bool Writes User data to the file and returns true if successful, or false if unsuccessful

Class: Game
Description: A class that represents a single game object in the program

Method	**Description**
Game()	**Parameters:** none **Return type:** N/A A default constructor that creates a new Game object with default attribute values
Game(string, string, float)	**Parameters:** string name, string seller, float price **Return type:** N/A A constructor that creates a new Game object with assigned values for the Game attributes
set_name(string)	**Parameters:** string name **Return type:** void A mutator method that assigns a value to the name attribute in a

	Game object
set_seller(string)	**Parameters:** string seller **Return type:** void A mutator method that assigns a value to the seller attribute in a Game object
set_price(float)	**Parameters:** float price **Return type:** void A mutator method that assigns a value to the price attribute in a Game object
get_name()	**Parameters:** none **Return type:** string An accessor method that returns the value of the name attribute from the Game object
get_seller()	**Parameters:** none **Return type:** string An accessor method that returns the value of the seller attribute from the Game object
get_price()	**Parameters:** none **Return type:** float An accessor method that returns the value of the price attribute from the Game object

Class: Session
Description: The session class represents a user session activity, from when a user logs in to when the logs out

Method	**Description**
Session(UserAccountsFileManager, AvailableGamesFileManager, DailyTransactionFileManager)	**Parameters:** UserAccountsFileManager userAccountsFileManager, AvailableGamesFileManager gamesFileManager, DailyTransactionFileManager **Return type:** N/A A constructor that creates a new Session that accesses the user accounts file manager and the available games file manager
start()	**Parameters:** none **Return type:** void A method that starts a new session when a user is signing in
processTransaction(string)	**Parameters:** string transactionCode **Return type:** void A method that processes a user transaction (e.g., refunds, sales, and purchases) in the session
getUser(string)	**Parameters:** string username

	Return type: User Returns a User from the session

Class: TransactionCode <enum>
Description: An enum class storing a list of the transaction code names in the program

Method	**Description**
Transaction codes (INVALID, Login, Logout, Create, Delete, Sell, Buy, Refund, AddCredit, List)	These are the list of all the transactions and transaction codes that will be permitted in our system

Class: Transaction
Description: A class that handles the transaction execution and its related functionality

Method	**Description**
Transaction(string, User)	**Parameters:** string codeStr, User user **Return type:** N/A A constructor to create a new transaction with a code string and a user object
Transaction(Transaction Code, User)	**Parameters:** TransactionCode code, User user **Return type:** N/A A constructor to create a new transaction with a Transaction Code and a user object
login(User)	**Parameters:** User user **Return type:** string A method that handles the transaction for a user login
logout(User)	**Parameters:** User user **Return type:** string A method that handles the transaction for the user logout
create_user(User)	**Parameters:** User user **Return type:** string A method that handles a user creation transaction
delete_user(User)	**Parameters:** User user **Return type:** string A method that handles a user deletion transaction
sell(User)	**Parameters:** User user **Return type:** string A method that handles a transaction to sell a game
buy(User)	**Parameters:** User user

	Return type: string A method that handles a transaction to buy a game
refund(User)	**Parameters:** User user **Return type:** string A method that handles a refund for a user
add_credit(User)	**Parameters:** User user **Return type:** string A method that handles a transaction for a user to add credits
list(User)	**Parameters:** User user **Return type:** string A method that returns a list of user transactions
parse_transaction_code (string)	**Parameters:** string codeStr **Return type:** TransactionCode A method that parses a string for a valid transaction code
execute()	**Parameters:** none **Return type:** void Executes a transaction with the operation depending on the transaction code that was associated with it
get_transaction_string()	**Parameters:** none **Return type:** string Get the transaction string for the daily transaction file
get_transaction_code	**Parameters:** none **Return type:** TransactionCode Get the transaction code for the daily transaction file
is_valid_code(string)	**Parameters:** string codeStr **Return type:** bool Checks if the string is a valid Transaction Code, and returns true if it is valid or false if it is invalid
success()	**Parameters:** none **Return type:** bool Returns true if transaction was successful, false if not

Class: User

Description: This class is a representation of the users in the system

Method	**Description**
User()	**Parameters:** none **Return type:** N/A This is a default constructor that creates a new User object with default values
User(string, string, float)	**Parameters:** string username, string user_type, float

	available_credits **Return type:** N/A This is a constructor that creates a new User object with specified values
set_username(string)	**Parameters:** string username **Return type:** void A mutator method that assigns a value to the username attribute
set_user_type(string)	**Parameters:** string user_type **Return type:** void A mutator method that assigns a value to the username attribute
add_credit(float)	**Parameters:** float added_credits **Return type:** void A mutator method that assigns adds credits to a User
remove_credit(float)	**Parameters:** float removed_credits **Return type:** void A mutator method that removes credits from a User
set_user_list(list<User>)	**Parameters:** list<User> users **Return type:** void A method that creates a list of all Users in the system
add_user(User)	**Parameters:** User user **Return type:** void A method that adds a new User to the Users list
remove_user(User)	**Parameters:** User user **Return type:** void A method that removes a User from the Users list
get_username()	**Parameters:** none **Return type:** string An accessor method that returns a username of a User
get_user_type()	**Parameters:** none **Return type:** string An accessor method that returns a User's type
get_credit()	**Parameters:** none **Return type:** float An accessor method that returns a User's total credits
get_user_list()	**Parameters:** none **Return type:** list<User> An accessor method that returns the list of Users

Class: Admin (Base: User)
Description: Inherited from the User class, Admin represents a User with admin privileges

Method	Description
Admin()	**Parameters:** none **Return type:** N/A Default constructor to create a new Admin User with default values
Admin(string, string, float)	**Parameters:** string username, string userType, float availableCredits **Return type:** N/A Admin constructor to create a new Admin User with specified values
create_user()	**Parameters:** none **Return type:** void Admin method that allows an Admin-level user to create a new user in the system
delete_user()	**Parameters:** none **Return type:** void Admin method that allows an Admin-level user to delete a user in the system
refund_credits()	**Parameters:** none **Return type:** void Admin method that allows an Admin-level user to process a credit refund to a user
buy_game()	**Parameters:** none **Return type:** void Method that allows an Admin-level user to buy a game available in the system
sell_game()	**Parameters:** none **Return type:** void Method that allows an Admin-level user to sell a game to be available in the system

Class: FullStandard (Base: User)
Description: Inherited from the User class, FullStandard represents a User with buying and selling privileges

Method	Description
FullStandard ()	**Parameters:** none **Return type:** N/A Default constructor to create a new FullStandard User with default values
FullStandard (string, string, float)	**Parameters:** string username, string userType, float availableCredits

	Return type: N/A FullStandard constructor to create a new FullStandard User with specified values
buy_game()	**Parameters:** none **Return type:** void Method that allows an FullStandard-level user to buy a game available in the system
sell_game()	**Parameters:** none **Return type:** void Method that allows an FullStandard-level user to sell a game to be available in the system

Class: BuyStandard (Base: User)
Description: Inherited from the User class, BuyStandard represents a User with buying privileges

Method	**Description**
BuyStandard()	**Parameters:** none **Return type:** N/A Default constructor to create a new BuyStandard User with default values
BuyStandard(string, string, float)	**Parameters:** string username, string userType, float availableCredits **Return type:** N/A BuyStandard constructor to create a new BuyStandard User with specified values
buy_game()	**Parameters:** none **Return type:** void Method that allows a BuyStandard user to buy an available game in the system

Class: SellStandard (Base: User)
Description: Inherited from the User class, SellStandard represents a User with selling privileges

Method	**Description**
SellStandard()	**Parameters:** none **Return type:** N/A Default constructor to create a new SellStandard User with default values
SellStandard(string,	**Parameters:** string username, string userType, float

string, float)	availableCredits **Return type:** N/A BuyStandard constructor to create a new SellStandard User with specified values
buy_game()	**Parameters:** none **Return type:** void Method that allows a SellStandard user to sell a game to be available in the system