*XP Course Project*

*Game Flow Software Design Document*

*Ammar Hatiya*

*Sammak Ahmed – 100747439*

*Fahad Fauzan*

*February 16, 2024*

# Table of Contents

[Table of Contents 2](#_Toc159114345)

[Introduction: 3](#_Toc159114346)

[**Architecture Overview:** 3](#_Toc159114347)

[**Front End Design:** 3](#_Toc159114348)

[**Class and Method Intentions:** 4](#_Toc159114349)

# **Introduction:**

The software designed in this project will resemble a Digital Game Distribution Service which will allow Users to purchase and sell games among other things. This document intends to outline the design of the system, illustrating the general architecture of the system and including a UML diagram to display the various methods/classes and their intended purpose. One thing to note, the Front End is where most of the user interaction will occur similar to a point of purchase terminal, while the Back End will just be an overnight processor that handles the transaction, account, and game files for the day.

# **Architecture Overview:**

There are two main focuses:

The Front End which is a point of purchase terminal written in C++.

The Back End which is an overnight processor written in Python.

They will interact with each other and users through command line text-based input/output streams.

# **Front End Design:**

Will consist of various functions in relation to each command:

**User Interface Module:** Focuses on handling user input and displaying appropriate output

**Authentication Module:** Will handle logging in and logging out and access control

**Account Management Module:** Will handle creating and deleting user accounts

**Game Management Module:** Handles the buying and selling of games

**Credit Module:** Handles interactions with credits, refund or add credit

**User Information Module:** Handles user information; list and/or profiles

**Transaction Records Module:** Handles the recording into transaction files

# **Class and Method Intentions:**

|  |  |  |
| --- | --- | --- |
| Class | Method | Intention |
| InputProcessor |  | Checking validities and shortcut functions. |
|  | inputProcessor(loginRef: Login&) | Constructor |
|  | getCommands | Get a string value of the list of commands available per access type |
|  | ignoreCase | Ignore casing for input and the such |
|  | isLogin | Checks to see if input is login commands |
|  | isValidInput | Checks to see if input is valid for user type |
|  | checkUserValid | Checks to see if user exists |
|  | updateUserCredit | Used to update user credit values |
|  | addCreditCommand | Used to add credit to user accounts |
|  | typeUser | Converts to user type based on the shorthands (e.g AA -> Admin) |
|  | listProfiles | Displays all the current user profiles |
| Buy |  | Class for functions related to buying a game |
|  | Buy(loginRef: Login&, listRef: List&) | Constructor |
|  | buyGame | Function to purchase a game |
| Login |  | Class for methods relating to logging in/out and user profile |
|  | isLoggedIn | Checks to see if user is logged in or not |
|  | getCurrentUser | Return current user logged in |
|  | getUserCredit | Returns current user credit amount |
|  | setUserCredit | Sets the current users credit amount |
|  | getUserLevel | Gets current user’s access type |
|  | typeUser | Converts user type to shorthand |
|  | loginUser | Logs in the user |
|  | logoutUser | Logs out user and updates DT file |
| List |  | Class to handle all methods pertaining to List |
|  | getGameName | Gets the game name |
|  | getSeller | Gets the seller of that game |
|  | getPrice | Gets the price of that game |
|  | collectList | Collects list of all games in the file |
|  | displayList | Displays all items on that list |
| AddCredit |  | Class to handle methods relating to Add Credit |
|  | AddCredit | Default constructor |
|  | addCreditCommand | Adds credit to the user and writes to the file |
| Profiles |  | Class relating to Profiles command |
|  | Profiles | Default constructor |
|  | listProfiles | Displays information of all current user accounts |
| Refund |  | Class that handles matters pertaining to refunding |
|  | Refund | Default constructor |
|  | refundTransaction | Method to refund user purchases as well as update relevant files |
| UserManagement |  | Handles orders relating to user information |
|  | loadExistingUser | Collects all the users that exist |
|  | addUser | Adds user to the list of active users |
|  | processCreateCommands | Deals with the create commands execution |
| User |  | Class that pertains to User information |
|  | User | Default constructor |
|  | User | Sets the values for the user |
| CreateAndDelete |  | Handles process of deleting and create accounts |
|  | stringToUserType | Converts the string type to a UserType |
|  | userTypeToString | Converts the UserType to a string value |
|  | isValidUsername | Checks to see if username is valid |