**CPS707 - Fall2021**

**Group 6**

**Phase 2:**

Suhail Al-Hakimi (500796337)

Avneet Jaswal (500838517)

Tusaif Azmat (500660278)

Course Project Assignment #2 - Front End Rapid Prototype

Application Name: Event Ticketing System (tix)

**1: Design Document**

The Front End, a point of purchase terminal for ticket selling and buying transactions.

It runs a Console application, that is, all the functions of application are to be invoked from

a command line and use text and text file input/output only.

Following table provides the overall structure of the application in the form of a table with all the classes and the methods.

| **Main method/ function** |
| --- |
| As this is a console application that runs as a command line action. The main method calls all other classes and their methods in sequential order depending on user actions.   * Displays welcome message and prompt users with the functions of the application i-e login, logout, buy, sell etc. * It also loads the user accounts file to authenticate user login function. * Main method provides all the functionalities of the application but it calls methods of the helper classes that maintain the different functions of the application. |

| **Class name** | **Method name** | **description** |
| --- | --- | --- |
| **Class login:** |  | Main method calls this class methods at the beginning of application. |
|  | Def login(): | This method initializes the login sequence by asking for a username as an input and returning the current user as an object. |
|  | Def get\_user\_account(user): | This method will search the user account file to find the user and return the user account as an object. |
| **Class user:** |  | This class prompts users for account details such as username, account type and credit amount. This class interacts with the backend where user account files and daily transaction files are stored and maintain logs. |
|  | Def user\_logout(self): | This method performs a logout function and logs the user action to the daily transaction log file. |
|  | Def create\_user(self): | This method will create a new user of the system with required information such as username, account type and credit amount. It also updates the user account files. |
|  | Def delete\_user(self): | This method deletes the user account and updates the user account file.Can only be done from a privileged account type |
|  | Def refund\_credit(): | This method takes a user object as an account and processes transactions such as buyer/ seller name to refund/ transfer amount between the accounts. |
| **Class ticket:** |  | A class to model a Ticket object and its associated properties such as event title, sale price of ticket, number of tickets, and seller username. |
|  | Def sell\_ticket(user): | User is adding tickets to sell and is being stored in the account transaction file. |
|  | Def buy\_ticket(user): | User is buying tickets to an event. |
| **Class add\_credit:** |  | This class helps to add or update the credit amount of a user and also updates the daily transaction file. Can only be done from a privileged account type. |
|  | Def add\_amount(user): | Add the amount of credit credit to the account and by what user and which account. All the information is stored in the daily transactions file and user account file. |
|  | Def request\_amount(user): | Takes the request from the user to add amount and stores that information in daily transactions file. |

**2: Source Code:**

The Front End of Ticket Selling Service done using python and tested using blackbox testing. Only one python file contains all the methods and classes that are required for the functionality of the front end. The system performs the point of purchase terminal for ticket selling and buying transactions.

Following is the source code of the python filename **event\_ticketing\_system.py**.

# Main method calls class methods at the beginning of application.

**def main\_method():**

print("Welcome to Event Ticketing System")

current\_user = Login.login()

user\_action = input(‘Enter a command from the list(logout, addcredit, create, delete, buy, sell, refund)’)

while user\_action != "EXIT":

user\_action = user\_action.lower()

user\_action = user\_action.strip()

if user\_action == "logout":

current\_user.logout(current\_user)

current\_user = Login.login()

elif user\_action == "buy":

if current\_user.role != "SS":

Ticket.buy(current\_user)

else:

print("Access denied.")

elif user\_action == "sell":

if current\_user.role != "BS":

Ticket.sell(current\_user)

else:

print("Access denied.")

break

elif user\_action == "create":

if current\_user.role == "AA":

User.create(current\_user)

else:

print("Access denied.")

elif user\_action == "delete":

if current\_user.role == "AA":

User.delete(current\_user)

else:

print("Access denied.")

elif user\_action == "refund":

if current\_user.role == "AA":

User.refund(current\_user)

else:

print("Access denied.")

elif user\_action == "addcredit":

second\_action = uIn.userIn("Enter add or request: ")

if second\_action == "add" and current\_user.role == "AA":

addcredit.add\_amount(current\_user)

elif second\_action == "request":

addcredit.request\_amount(current\_user)

else:

print("Access denied.")

else:

print("Invalid option, try again.")

user\_action = input("Enter a command from the list (logout, addcredit, create, delete, buy, sell, refund)")

**class Login:**

# The methods of this class below

def login():

user\_input = input("For log in, please type login: ")

while user\_input.lower() != "login":

user\_input = input("For log in, please type login: ")

user\_name = input("Enter Username: ")

current\_user = Login.get\_user\_account(user\_name)

while current\_user == 0:

print("Invalid username, please try again")

user = input("Enter Username: ")

current\_user = Login.get\_user\_account(user\_name)

print("Login successful...")

return current\_user

def get\_user\_account(user):

with open('user\_account\_file.txt', 'r') as reader:

user\_file = reader.readlines()

# reads user\_account\_file line by line and gets user names

for x in user\_file:

if user == x[:15].strip():

role = x[16:18]

credit = x[19:]

user\_name = User(user, role, float(credit))

reader.close()

return user\_name

print("User not found. ")

reader.close()

return 0

**class Ticket:**

# The methods of this class below

def sell\_ticket(user):

# this will be implemented soon

print("Ticket not found")

def buy\_ticket(user):

# this will be implemented soon

print("Ticket not found")

**class addcredit:**

# The methods of this class below

def add\_amount(user):

# this will be implemented soon

print("Credit added")

def request\_amount(user):

# this will be implemented soon

print("Request made")

**class User():**

# The methods of this class below

def user\_logout(self):

# this will be implemented soon

print("Logout successful")

def create\_user(self):

# this will be implemented soon

print("Account created")

def delete\_user(self):

# this will be implemented soon

print("Deletion successful")

def refund\_credit(user):

# this will be implemented soon

print("Refund complete")

**if \_\_name\_\_ == '\_\_main\_\_':**

**main\_method()**