CPS707 - Fall2021

Group 6

Phase 4:

Suhail Al-Hakimi (500796337)

Avneet Jaswal (500838517)

Tusaif Azmat (500660278)

Course Project Assignment #4 - Back End Rapid Prototype

Application Name: Event Ticketing System (tix)

1: Design Document

The Back End, an overnight batch processor to maintain and update a master ticket file (written in Java)

It also runs a Console application, that means, all the functions of the application are to be invoked from a command line and use text and text file input/output only.

The Back End reads in the previous day's User Accounts File and Available Tickets File and then applies all of the daily transactions from a merged set of daily transaction files to these files to produce a new Current User Accounts File and new Available Tickets File for tomorrow's front End runs.

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

Class name	Method name	description
Class BackEnd		The BackEnd class runs the backend functions.
	public static void main(String[] args)	This is the main method that reads through the daily transaction file(dtf.txt) to apply changes to the user account file(uaf.txt) and available ticket file (atf.txt).
	public static void mergeFiles()	This method mergeFiles merges multiple daily transaction files into one
Class UpdateBackEnd		This class has methods to update each backend file according to the correct action
	public void addCredit(String input)	This method updates the User Account File (uaf) file with the new amount of credit for that user

	public void buy(String input)	This method updates the Available Ticket File (atf) file with the new amount of available tickets information and it also updates the User Account File (uaf) with buyers new credit amount information.
	public void sell(String input)	This method update the Available ticket file (atf) file with the new ticket available for sale
	public void create(String input)	This method updates the User Account File (uaf) file with the newly created user information.
	public void delete(String input)	This method deletes the account from User account File (uaf) and any related tickets deleted from Available Ticket file (atf).
	public void refund(String input)	This method updates the buyer and seller's credit amount in the User Account File (uaf).

	static void modifyFiles(String filePath, String oldString, String newString)	This method modifies the files. Reading all the lines of input text file into oldContent and Replacing oldString with newString in the oldContent. Lastly Rewriting the input text file with newContent

2: Source Code:

```
public class BackEnd
 {
        public static void main(String[] args) throws IOException
        {
                // This is where back end reads through daily transaction file to apply
                changes to User Account File (uaf) and Available Ticket File(atf).
                        mergeFiles();
                        Scanner scanner = new Scanner(new
                                                File("daily_transaction_file.txt"));
                        while (scanner.hasNextLine())
                          {
                           String action = scanner.nextLine();
                           UpdateBackEnd update = new UpdateBackEnd();
                           if (action.length() > 0) {
                              if (action.substring(0, 2).equals("01")) {
                                 update.create(action);
                              } else if (action.substring(0, 2).equals("02")) {
                                 update.delete(action);
                              } else if (action.substring(0, 2).equals("03")) {
                                 update.sell(action);
                              } else if (action.substring(0, 2).equals("04")) {
                                 update.buy(action);
                              } else if (action.substring(0, 2).equals("05")) {
                                 update.refund(action);
                              } else if (action.substring(0, 2).equals("06")) {
```

```
                        update.addC(action);
                    } else if (action.substring(0, 2).equals("00")) {

                    } else {
                            //do nothing
                        }
                    }
                }
            scanner.close();
        }

    public static void mergeFiles() throws IOException
    {
        // This method merges multiple daily transaction files into one and stores
    them in one directory.
        File createDir = new File("src/output");
        PrintWriter prWriter = new PrintWriter("daily_transactions_file.txt");
        String[] fileNames = createDir.list();
         for (String fileName : fileNames) {
            if (fileName.contains("daily_transactions_file")) {
                File createFile = new File(createDir, fileName);
                // create object of BufferedReader
                BufferedReader buffReader = new BufferedReader
                                                (new FileReader(createFile));
                // Read from current file
                String line = buffReader.readLine();
                while (line != null) {
                    // write to the output file
                     prWriter.println(line);
                    line = buffReader.readLine();
                }
            prWriter.flush();
            buffReader.close();
            }
          prWriter.close();
        }
}

class UpdateBackEnd
{
    public void addCredit(String input) throws IOException
```

```
{
        // this will update the user account File (uaf) file with
        //the new amount of credit for that user
         String userName = input.substring(3, 19).strip();
         String userRole = input.substring(19, 22).strip();
         String userAmount = input.substring(22).strip();
         Scanner scan = new Scanner((new File("src/user_account_file.txt")));
         while (scan.hasNextLine()) {
                String output = scan.nextLine();
                if (output.substring(0, 16).strip().equals(userName)) {
                        Float total = Float.parseFloat(output.substring(20)) +
                                        Float.parseFloat(userAmount);
                        String news = String.format("%-15s", userName) + " "
                                + userRole + " " + (String.format("%09.2f", total));
                modifyFiles("src/user_account_file.txt", output, addNewLine);
                break;
                        }
                 }
                 scan.close();
        }

public void buy(String input) throws IOException
{
        // still to add code here
}

public void sell(String input) throws IOException
{
        // the method will update the available ticket file (atf)  file with the
        //new ticket available for sale
         UpdateBackEnd.modifyFiles("src/available_ticket_file.txt", "END",
                input.substring(3) + "\nEND");
}

public void create(String input) throws IOException
{
        // this method will update user account file (uaf) with
        //the newly created user
        BufferedWriter buffWriter = new BufferedWriter(
            new FileWriter("src/uaf.txt", true)   );
```

```
        buffWriter.write(input.substring(3));
        buffWriter.close();
}

public void delete(String input) throws IOException
{
          // delete account from user account file (uaf) and
         // any related tickets deleted from available ticket file (atf)
        List<String> linesRead = FileUtils.readLines(new
                                    File("src/user_account_file.txt"));
        List<String> updatedLines = linesRead.stream().filter(s ->
           !s.contains(input.substring(3))).collect(Collectors.toList());
        FileUtils.writeLines(new File("src/user_account_file.txt"),
                                          updatedLines, false);

        linesR = FileUtils.readLines(new File("src/available_ticket_file.txt"));
        updatedLines = linesR.stream().filter(s ->
          !s.contains(input.substring(3, 19).strip())).collect(Collectors.toList());
        FileUtils.writeLines(new File("src/available_ticket_file.txt"),
                                                        updatedLines, false);
}

public void refund(String input) throws IOException
{
          //still to add code
}

static void modifyFiles(String filePath, String oldString, String newString)
 throws IOException
{
       // modify the file as specified as per the requirements
        File modifyFile = new File(filePath);
        String readOldContent = "";
        BufferedReader fileReader = null;
        FileWriter newFileWriter= null;
        fileReader = new BufferedReader(new FileReader(modifyFile));
        //Reading all the lines of input text file into read Old Content
        String line = fileReader.readLine();
        while (line != null) {
           readOldContent = readOldContent + line + System.lineSeparator();
```

```
            line = fileReader.readLine();
        }
        //Replacing oldString with newString in the read old Content
        String newContent = readOldContent.replaceAll(oldString, newString);
        //Rewriting the input text file with newContent
        newFileWriter = new FileWriter(modifyFile);
        newFileWriter.write(newContent);

        fileReader.close();
        newFileWriter.close();
}
```