CS4222/CS5052 Project Submission Documentation

# Group Number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Project Number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Please use this checklist to ensure that you have all the items for your submission. You should deliver these items in accordance with the Delivery Instructions outlined on this page.

* **Signed percentages of the work done by each student (equal work = 25% each, assuming 4 persons in a group) (cf. Page 3 of this document)**
* **Completed Project Methods/Person Contribution Table (cf. Page 4 of this document)**
* **Printed Documented Source Code (printed in landscape mode)**
* **Completed Project Group Meetings Deliverable (cf. Page 2 of this document)**

**Delivery Instructions**

* **Email Documented Source Code to annette.mcelligott@ul.ie**

The group's documented source code, data files, Java documentation and so forth should be emailed to me by the project leader, by the deadline for this project. Each of the other group members should be copied on this email. The subject of this email should be CS4222/CS5052 Project X Group Y where X is the Project Number and Y is the Group's Number.

* **By the deadline, the Project Leader should put the following items in a see-through plastic pocket organiser and then place them in the relevant box underneath the stairs in the CS Building.**
  + Printed Documented Source Code (printed in landscape mode)
  + Signed percentages of the work done by each student (cf. Page 3 of this document)
  + Completed Project Methods/Person Contribution Table (also signed by each person in the group) (cf. Page 4 of this document)
  + Completed Project Group Meetings Deliverable (cf. Page 2 of this document)

**CS4222/CS5052: Project Group Meetings Deliverable**

# Group Number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Project Number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**You should also use this document to detail any problems encountered by the group.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Meeting Number** | **In Attendance** | **Assumptions Made and Decisions Taken/Agreed** |
|  |  |  |  |
|  |  |  |  |

|  |  |
| --- | --- |
| **Student ID** | **Student Name (Signature)** |
| **17241499** |  |
|  |  |
|  |  |
|  |  |

**CS4222/CS5052: Signed percentages of the work done by each student**

# Group Number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Project Number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| **Student ID** | **Student Name and Student Signature** | **Percentage of marks** |
| **17241499** | *Donal O’Connell* |  |
| **17238838** | *Wiktor Luczak* |  |
| **17206413** | Eoin Gohery |  |

**CS4222/CS5052: Project Methods/Person Contribution**

**Group Number \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**TM1:** *Eoin Gohery*

**TM2:** *Donal O’Connell*

**TM3:** *Wiktor Luczak*

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **TM1** | **TM2** | **TM3** |
| main |  |  |  |
| ReadUser |  |  |  |
| readBooking |  |  |  |
| readFacility |  |  |  |
| createUser |  |  |  |
| login |  |  |  |
| createFacility |  |  |  |
| viewAvailability |  |  |  |
| shiftDate |  |  |  |
| addBooking |  |  |  |
| decomissionFacility |  |  |  |
| removeFacility |  |  |  |
| updateFacilities |  |  |  |
| updateBooking |  |  |  |
| recordPayment |  |  |  |
| viewStatement |  |  |  |
| ChooseUser |  |  |  |
| ChooseFacility |  |  |  |
| View Bookings |  |  |  |
| isDateValid |  |  |  |

**Signatures of Group Members**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**public class User**

**{**

**public int userID;**

**public String email;**

**public int userType;**

**public String password;**

**public User(int userID, String email, String password, int userType)**

**{**

**this.userID = userID;**

**this.email = email;**

**this.password = password;**

**this.userType = userType;**

**}**

**public int getUserID()**

**{**

**return userID;**

**}**

**public void setUserID(int userID)**

**{**

**this.userID = userID;**

**}**

**public String getEmail()**

**{**

**return email;**

**}**

**public void setEmail(String email)**

**{**

**this.email = email;**

**}**

**public String getPassword()**

**{**

**return password;**

**}**

**public void setPassword(String password)**

**{**

**this.password = password;**

**}**

**public int getUserType()**

**{**

**return userType;**

**}**

**public void setUserType(int userType)**

**{**

**this.userType = userType;**

**}**

**}**

**public class Facility**

**{**

**public int facilityID;**

**public String facilityName;**

**public double pricePerHour;**

**public String decommissionedUntilDate;**

**public Facility(int facilityID, String facilityName, double pricePerHour, String decommissionedUntilDate)**

**{**

**this.facilityName = facilityName;**

**this.decommissionedUntilDate = decommissionedUntilDate;**

**this.facilityID = facilityID;**

**this.pricePerHour = pricePerHour;**

**}**

**public String getFacilityName()**

**{**

**return facilityName;**

**}**

**public void setFacilityName(String facilityName)**

**{**

**this.facilityName = facilityName;**

**}**

**public String getDecommissionedUntilDate()**

**{**

**return decommissionedUntilDate;**

**}**

**public void setDecommissionedUntilDate(String decommissionedUntilDate)**

**{**

**this.decommissionedUntilDate = decommissionedUntilDate;**

**}**

**public int getFacilityID()**

**{**

**return facilityID;**

**}**

**public void setFacilityID(int facilityID)**

**{**

**this.facilityID = facilityID;**

**}**

**public double getPricePerHour()**

**{**

**return pricePerHour;**

**}**

**public void setPricePerHour(double pricePerHour)**

**{**

**this.pricePerHour = pricePerHour;**

**}**

**}**

**public class Booking**

**{**

**public int bookingID;**

**public int facilityID;**

**public int userID;**

**public int slot;**

**public String date;**

**public String paymentStatus;**

**public Booking(int bookingID, int facilityID, int userID, int slot, String date, String paymentStatus)**

**{**

**this.bookingID = bookingID;**

**this.facilityID = facilityID;**

**this.userID = userID;**

**this.slot = slot;**

**this.date = date;**

**this.paymentStatus = paymentStatus;**

**}**

**public String getPaymentStatus()**

**{**

**return paymentStatus;**

**}**

**public void setPaymentStatus(String paymentStatus)**

**{**

**this.paymentStatus = paymentStatus;**

**}**

**public String getDate()**

**{**

**return date;**

**}**

**public void setDate(String date)**

**{**

**this.date = date;**

**}**

**public int getFacilityID()**

**{**

**return facilityID;**

**}**

**public void setFacilityID(int facilityID)**

**{**

**this.facilityID = facilityID;**

**}**

**public int getSlot()**

**{**

**return slot;**

**}**

**public void setSlot(int slot)**

**{**

**this.slot = slot;**

**}**

**public int getBookingID()**

**{**

**return bookingID;**

**}**

**public void setBookingID(int bookingID)**

**{**

**this.bookingID = bookingID;**

**}**

**public int getUserID()**

**{**

**return userID;**

**}**

**public void setUserID(int userID)**

**{**

**this.userID = userID;**

**}**

**}**

**import java.util.\*;**

**import java.io.\*;**

**import javax.swing.\*;**

**import java.text.\*;**

**public class Driver**

**{**

**private static final String[] mainAdmin = { "Register a New User", "Add Facility", "View Facilities","View Bookings", "Record Payment", "View User Statements", "Log Out" };**

**private static final String[] subAdmin = { "View Availabilty", "Add Booking", "Decommission Facility", "Remove Facility"};**

**private static final String[] mainUser = { "View Bookings","View Statement" };**

**private static final String[] slots = { " 9:00 - 10:00","10:00 - 11:00", "11:00 - 12:00", "12:00 - 13:00", "13:00 - 14:00", "14:00 - 15:00", "15:00 - 16:00", "16:00 - 17:00", "17:00 - 18:00" };**

**private static SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");**

**public static Date now = new Date();**

**private static boolean loggedIn = false;**

**private static boolean admin = false;**

**public static int currentFacilityNum, loginAttempts;**

**public static int currentUserNum;**

**private static File userInfo = new File ("users.txt");**

**private static File bookingInfo = new File ("bookings.txt");**

**private static File facilityInfo = new File ("facilities.txt");**

**public static List<Facility> facilities = new ArrayList<Facility>();**

**public static List<User> users = new ArrayList<User>();**

**public static List<Booking> bookings = new ArrayList<Booking>();**

**public static Facility nullFacility = new Facility(0, null, 0.0, null);**

**public static User nullUser = new User(0, null, null, 0);**

**public static Booking nullBooking = new Booking(0, 0, 0, 0, null, null);**

**public static void main(String[] args) throws IOException**

**{**

**/\*\***

**\* The main menu is a JOptionPane drop down menu which compares the chosen option with Strings**

**\* Before the menu, the program will load the file into ArrayLists with the readUser, readBooking and readFacility methods.**

**\* There are two seperate menus, one for the user and one for the admin.**

**\*/**

**readUser();**

**readBooking();**

**readFacility();**

**while (!loggedIn)**

**{**

**login();**

**}**

**boolean main = true;**

**if (admin)**

**{**

**while (main) {**

**String section = (String) JOptionPane.showInputDialog(null, "Menu","",JOptionPane.QUESTION\_MESSAGE, null, mainAdmin, mainAdmin[0]);**

**if(section=="Register a New User") {**

**createUser();**

**}else if(section=="Add Facility") {**

**createFacility();**

**} else if(section=="View Bookings") {**

**viewBookings();**

**} else if(section =="Record Payment") {**

**recordPayment();**

**} else if(section=="View Facilities") {**

**chooseFacility();**

**boolean sub = true;**

**while (sub) {**

**String subSection = (String) JOptionPane.showInputDialog(null, "Menu","",JOptionPane.QUESTION\_MESSAGE, null, subAdmin, subAdmin[0]);**

**if(subSection=="View Availabilty")**

**{**

**viewAvailabilty();**

**} else if(subSection=="Add Booking")**

**{**

**chooseUser();**

**addBooking();**

**} else if(subSection =="Decommission Facility")**

**{**

**decommissionFacility();**

**sub = false;**

**} else if(subSection =="Remove Facility")**

**{**

**removeFacility();**

**} else {**

**sub = false;**

**}**

**}**

**} else if(section=="View User Statements")**

**{**

**chooseUser();**

**viewStatement();**

**} else**

**main = false;**

**}**

**} else**

**{**

**while (main)**

**{**

**String section = (String) JOptionPane.showInputDialog(null, "Menu","",JOptionPane.QUESTION\_MESSAGE, null, mainUser, mainUser[0]);**

**if(section=="View Bookings") {**

**viewBookings();**

**}else if(section=="View Statement") {**

**viewStatement();**

**}else**

**main = false;**

**}**

**}**

**JOptionPane.showMessageDialog(null, "You have been logged out");**

**System.exit(0);**

**}**

**public static void readUser() throws IOException**

**{**

**/\*\***

**\* The method reads through the file and if there is a gap in the user number(eg userID 1 then userID 3)**

**\* a nullUser will be placed to ensure that the userID will always be equal to the position of the user in the users ArrayList.**

**\* A nullUser is placed at the start of the Array for the same reason.**

**\*/**

**PrintWriter pw = new PrintWriter(new FileWriter(userInfo,true));**

**String lineFromFile;**

**int i = 0;**

**String fileElements[];**

**int userID;**

**String email;**

**int userType;**

**String password;**

**Scanner read = new Scanner(userInfo);**

**users.add(nullUser);**

**while(read.hasNext())**

**{**

**i++;**

**lineFromFile = read.nextLine();**

**fileElements = lineFromFile.split(",");**

**userID = (Integer.parseInt(fileElements[0]));**

**email = (fileElements[1]);**

**password = (fileElements[2]);**

**userType = (Integer.parseInt(fileElements[3]));**

**while (i<userID)**

**{**

**i++;**

**users.add(nullUser);**

**}**

**User tempUser = new User(userID, email, password, userType);**

**users.add(tempUser);**

**}**

**read.close();**

**}**

**public static void readBooking() throws IOException**

**{**

**/\*\***

**\* The method reads through the file and if there is a gap in the booking number(eg bookingID 1 then booingID 3)**

**\* a nullBooking will be placed to ensure that the bookingID will always be equal to the position of the booking in the bookings ArrayList.**

**\* A nullBooking is placed at the start of the Array for the same reason.**

**\*/**

**PrintWriter pw = new PrintWriter(new FileWriter(bookingInfo,true));**

**String lineFromFile;**

**String fileElements[];**

**int bookingID;**

**int facilityID;**

**int userID;**

**int slot;**

**String date;**

**String paymentStatus;**

**int i = 0;**

**bookings.add(nullBooking);**

**Scanner read = new Scanner(bookingInfo);**

**while(read.hasNext())**

**{**

**i++;**

**lineFromFile = read.nextLine();**

**fileElements = lineFromFile.split(",");**

**bookingID = (Integer.parseInt(fileElements[0]));**

**facilityID = (Integer.parseInt(fileElements[1]));**

**userID = (Integer.parseInt(fileElements[2]));**

**slot = (Integer.parseInt(fileElements[3]));**

**date = (fileElements[4]);**

**paymentStatus = (fileElements[5]);**

**while (i<bookingID)**

**{**

**i++;**

**bookings.add(nullBooking);**

**}**

**Booking tempBooking = new Booking(bookingID, facilityID, userID, slot, date, paymentStatus);**

**bookings.add(tempBooking);**

**}**

**read.close();**

**}**

**public static void readFacility() throws IOException**

**{**

**/\*\***

**\* The method reads through the file and if there is a gap in the facility number(eg facilityID 1 then facilityID 3)**

**\* a nullFacility will be placed to ensure that the facilityID will always be equal to the position of the facility in the facilities ArrayList.**

**\* A nullFacility is placed at the start of the Array for the same reason.**

**\* This method will also check if and of the decommisioned facilities can be recomisioned**

**\* by checking if the decommissionedUntilDate is equal to or before the current date.**

**\* The method then runs the updateFacilities mehtod which will delete the facilities file and print a new one based on the ArrayList.**

**\*/**

**PrintWriter pw = new PrintWriter(new FileWriter(facilityInfo,true));**

**String lineFromFile, date;**

**String fileElements[];**

**int facilityID;**

**String facilityName;**

**double pricePerHour;**

**String decommissionedUntilDate;**

**int i = 0;**

**Scanner read = new Scanner(facilityInfo);**

**facilities.add(nullFacility);**

**while(read.hasNext())**

**{**

**i++;**

**lineFromFile = read.nextLine();**

**fileElements = lineFromFile.split(",");**

**facilityID = (Integer.parseInt(fileElements[0]));**

**facilityName = (fileElements[1]);**

**pricePerHour = (Double.parseDouble(fileElements[2]));**

**if (fileElements.length == 4)**

**{**

**decommissionedUntilDate = (fileElements[3]);**

**try**

**{**

**Date tempDate = formatter.parse(decommissionedUntilDate);**

**if ((tempDate.before(now)) || (tempDate.equals(now)))**

**{**

**decommissionedUntilDate = null;**

**}**

**} catch (ParseException e) {}**

**} else**

**{**

**decommissionedUntilDate = null;**

**}**

**while (i<facilityID)**

**{**

**i++;**

**facilities.add(nullFacility);**

**}**

**Facility tempFacility = new Facility(facilityID, facilityName, pricePerHour, decommissionedUntilDate);**

**facilities.add(tempFacility);**

**}**

**read.close();**

**updateFacilities();**

**}**

**public static void createUser() throws IOException**

**{**

**/\*\***

**\* This method requsts a user email and will automatically generate a 6 character pawword of leter(higher and lower) anf numbers**

**\* The details of the user are then added to the array list and printed to file.**

**\*/**

**int userID = users.size();**

**String email = JOptionPane.showInputDialog(null,"Please enter the email");**

**String generatedPass = "";**

**int userType = 1;**

**String AB = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";**

**Random rnd = new Random();**

**StringBuilder sb = new StringBuilder(6);**

**for( int i = 0; i < 6; i++ )**

**{**

**sb.append( AB.charAt( rnd.nextInt(AB.length()) ) );**

**}**

**generatedPass = sb.toString();**

**JOptionPane.showMessageDialog(null,"your password is " + generatedPass); User tempUser = new User(userID,email,generatedPass,userType);**

**users.add(tempUser);**

**PrintWriter pw = new PrintWriter(new FileWriter(userInfo,true));**

**pw.println(userID + "," + email + "," + generatedPass + "," + userType);**

**pw.close();**

**}**

**public static void login() throws IOException**

**{**

**/\*\***

**\* This method requests the user's email and password,it will then check if the user name and password exists in the file**

**\* if it does exists the user will be logged in,if not the user will be given 2 more chances (3 in total)**

**\* if the user fails too many times the program will close**

**\* if the file is empty the user will be requested to set up an admin account.**

**\*/**

**String Email,Pass;**

**Scanner in;**

**boolean found = false;**

**PrintWriter pw = new PrintWriter(new FileWriter(userInfo,true));**

**if (userInfo.length() == 0)**

**{**

**Email = JOptionPane.showInputDialog(null,"Please enter your new admin acount name");**

**Pass = JOptionPane.showInputDialog(null,"Please enter your password");**

**pw.println("1," + Email + "," + Pass + ",0");**

**User tempUser = new User(1,Email,Pass,0);**

**users.add(tempUser);**

**admin = true;**

**loggedIn =true;**

**pw.close();**

**} else if (userInfo.length() > 0)**

**{**

**Email = JOptionPane.showInputDialog(null,"Please enter your email");**

**Pass = JOptionPane.showInputDialog(null,"Please enter your password");**

**in = new Scanner(userInfo);**

**int x,y;**

**String tempEmail,tempPass,line;**

**String lineElements[];**

**while (in.hasNext() && !found)**

**{**

**line = in.nextLine();**

**lineElements = line.split(",");**

**x = Integer.parseInt(lineElements[0]);**

**tempEmail = lineElements[1];**

**tempPass = lineElements[2];;**

**y = Integer.parseInt(lineElements[3]);**

**if (Email.equals(tempEmail.trim()) && Pass.equals(tempPass.trim()))**

**{**

**found = true;**

**currentUserNum = x;**

**JOptionPane.showMessageDialog(null,"Login Successful");**

**loggedIn = true;**

**if (y == 0)**

**{**

**admin = true;**

**} else**

**{**

**admin = false;**

**}**

**}**

**}**

**if (!found && loginAttempts < 3)**

**{**

**JOptionPane.showMessageDialog(null,"Login failed");**

**loginAttempts++;**

**}**

**else if (!found && loginAttempts == 3)**

**{**

**JOptionPane.showMessageDialog(null,"Too many failed login attempts");**

**System.exit(0);**

**}**

**}**

**}**

**public static void createFacility() throws IOException**

**{**

**/\*\***

**\* This mehtod cretes a new facility by requesting a facility name and price.**

**\* If the facility name is a;ready in use the program will request a differnet name.**

**\* The facility is then added to the ArrayList and printed to the end of the file.**

**\* decommissionedUntilDate is set as null to start as it does not make sense for a new facility to be immediatly decommisioned.**

**\*/**

**String facilityName;**

**double pricePerHour;**

**boolean available = true;**

**PrintWriter pw = new PrintWriter(new FileWriter(facilityInfo,true));**

**facilityName = (String) JOptionPane.showInputDialog(null,"Please enter the name of the new facility","");**

**for (int i=0; i<facilities.size(); i++)**

**{**

**if (facilityName.equals(facilities.get(i).getFacilityName()))**

**{**

**JOptionPane.showMessageDialog(null,"That facility name is already in use.");**

**available = false;**

**break;**

**} else**

**{**

**available = true;**

**}**

**}**

**if (available == true)**

**{**

**pricePerHour = Double.parseDouble(JOptionPane.showInputDialog(null,"Please enter the price per hour of the new facility","00.00"));**

**Facility tempFacility = new Facility(facilities.size(), facilityName, pricePerHour, null);**

**facilities.add(tempFacility);**

**pw.println(facilities.size()-1 + "," + facilityName + "," + pricePerHour + ",");**

**}**

**pw.close();**

**}**

**public static void viewAvailabilty() throws IOException**

**{**

**/\*\***

**\* View Availabilty will display all time slots for a particular date range for a single facilty.**

**\* Next to each of the slots will be status that idicates wheter the facility is available or Unavailable.**

**\* To ensure the dates are valid the dates inputed are run through the isDateValid mehtod.**

**\* The Availabilty is checked by checking the date slot and faciltyID.**

**\* If all three are in use ny a single booking then tat slot is Unavailable.**

**\* The start date is incremented by one day until the date is equal to the endDate using the shiftDate method**

**\*/**

**String start, end;**

**start = (String) JOptionPane.showInputDialog(null,"On what date would you like to start viewing? Format: dd/mm/yyyy","//");**

**while (!isDateValid(start))**

**{**

**start = (String) JOptionPane.showInputDialog(null,"On what date would you like to start viewing? Format: dd/mm/yyyy","//");**

**}**

**end = (String) JOptionPane.showInputDialog(null,"On what date would you like to finish viewing? Format: dd/mm/yyyy","//");**

**while (!isDateValid(end))**

**{**

**end = (String) JOptionPane.showInputDialog(null,"On what date would you like to finish viewing? Format: dd/mm/yyyy","//");**

**}**

**try**

**{**

**Date startDate = formatter.parse(start);**

**Date endDate = formatter.parse(end);**

**shiftDate(endDate);**

**while (startDate.before(endDate))**

**{**

**System.out.println(" " + formatter.format(startDate));**

**for (int slot=0; slot<slots.length; slot++)**

**{**

**for (int i=0; i<bookings.size(); i++)**

**{**

**if (bookings.get(i).getDate() != null)**

**{**

**Date checkDate = formatter.parse(bookings.get(i).getDate());**

**if (bookings.get(i).getSlot()==slot+1 && bookings.get(i).getFacilityID()==currentFacilityNum && startDate.equals(checkDate))**

**{**

**System.out.println(slots[slot] + ": Unavailable");**

**break;**

**} else if (i==bookings.size()-1)**

**{**

**System.out.println(slots[slot] + ": Available");**

**}**

**}**

**}**

**}**

**shiftDate(startDate);**

**System.out.println();**

**}**

**} catch (ParseException e) {}**

**}**

**public static void shiftDate(Date d)**

**{**

**/\*\***

**\* This method cretes a date one day after the inputted date.**

**\* the inputted date is then set to equal the new date and is returned.**

**\*/**

**Calendar c = Calendar.getInstance();**

**c.setTime(d);**

**c.add(Calendar.DATE, 1);**

**d.setTime(c.getTimeInMillis());**

**}**

**public static void addBooking() throws IOException**

**{**

**/\*\***

**\* This method adds a booking for the facilty and user chosen in the chooseUser and chooseFacility methods.**

**\* The date chosen by the user is ran through the isDateValid method.**

**\* The user is then shown a drop down menu of all timeslots.**

**\* the method checks to ensure that the booking is not a nullBooking**

**\* The date slot and facilityID are all compared to check if the facilty is available at that time.**

**\*/**

**String chosenDate;**

**int slot = 0;**

**boolean valid = false;**

**chosenDate = (String) JOptionPane.showInputDialog(null,"On what date do you wish to make the booking. Format: dd/mm/yyyy","//");**

**while (!isDateValid(chosenDate))**

**{**

**chosenDate = (String) JOptionPane.showInputDialog(null,"On what date do you wish to make the booking. Format: dd/mm/yyyy","//");**

**}**

**while (!valid)**

**{**

**slot = (Arrays.asList(slots).indexOf((String) JOptionPane.showInputDialog(null, "Choose a time.","",JOptionPane.QUESTION\_MESSAGE, null, slots, slots[0]))) +1;**

**for (int i=0; i<bookings.size(); i++)**

**{**

**if (bookings.get(i).getDate() != null)**

**{**

**try**

**{**

**Date existingDate = formatter.parse(bookings.get(i).getDate());**

**if (bookings.get(i).getSlot()==slot && bookings.get(i).getFacilityID()==currentFacilityNum && existingDate.equals(chosenDate))**

**{**

**JOptionPane.showMessageDialog(null,"The Facility is currently booked for this date and time.");**

**valid = false;**

**break;**

**}**

**valid = true;**

**} catch (ParseException e) {}**

**}**

**}**

**}**

**Booking tempBooking = new Booking(bookings.size(), currentFacilityNum, currentUserNum, slot, chosenDate, "N");**

**bookings.add(tempBooking);**

**updateBookings();**

**}**

**public static void decommissionFacility() throws IOException**

**{**

**/\*\***

**\* The facilty chosen through the choose facilty method will be decommisioned until the date inputted by the user.**

**\* the date is run through the isDateValid mehtod.**

**\* The decommissionedUntilDate of that particular facility is changed from null to the date set.**

**\* The update facility method is then run which will rewrite the file with the new decommissionedUntilDate**

**\*/**

**String facilityLine, facilityName, decommissionedUntilDate;**

**int facilityID;**

**double pricePerHour;**

**decommissionedUntilDate = (String) JOptionPane.showInputDialog(null,"On what date do you wish the facility to be recommissioned. Format: dd/mm/yyyy","//");**

**while (!isDateValid(decommissionedUntilDate))**

**{**

**decommissionedUntilDate = (String) JOptionPane.showInputDialog(null,"On what date do you wish the facility to be recommissioned. Format: dd/mm/yyyy","//");**

**}**

**facilities.get(currentFacilityNum).setDecommissionedUntilDate(decommissionedUntilDate);**

**PrintWriter pw = new PrintWriter(new FileWriter(facilityInfo));**

**for (int i=0; i<facilities.size(); i++)**

**{**

**facilityName = facilities.get(i).getFacilityName();**

**if ((facilityName != null) && (facilityName.length() > 0))**

**{**

**facilityName = facilities.get(i).getFacilityName();**

**facilityID = facilities.get(i).getFacilityID();**

**pricePerHour = facilities.get(i).getPricePerHour();**

**decommissionedUntilDate = facilities.get(i).getDecommissionedUntilDate();**

**if (decommissionedUntilDate == null)**

**{**

**pw.println(facilityID + "," + facilityName + "," + pricePerHour + ",");**

**} else**

**{**

**pw.println(facilityID + "," + facilityName + "," + pricePerHour + "," + decommissionedUntilDate);**

**}**

**}**

**}**

**pw.close();**

**}**

**public static void removeFacility() throws IOException**

**{**

**/\*\***

**\* this will immediatly change the values of the facilities so that the facility is made into a nullFacility.**

**\* The updateFacilities method is then run which will rewrite the new facilties file without that facility.**

**\*/**

**facilities.get(currentFacilityNum).setFacilityID(0);**

**facilities.get(currentFacilityNum).setFacilityName(null);**

**facilities.get(currentFacilityNum).setPricePerHour(0.0);**

**facilities.get(currentFacilityNum).setDecommissionedUntilDate(null);**

**updateFacilities();**

**}**

**public static void updateFacilities() throws IOException**

**{**

**/\*\***

**\* The update facilities file will delete the current facilities file and create a new one.**

**\* the method will print the details of any non nullFacility facilty.**

**\*/**

**String facilityLine;**

**String facilityName, decommissionedUntilDate;**

**int facilityID;**

**double pricePerHour;**

**PrintWriter pw = new PrintWriter(new FileWriter(facilityInfo));**

**for (int i=0; i<facilities.size(); i++)**

**{**

**facilityName = facilities.get(i).getFacilityName();**

**if ((facilityName != null) && (facilityName.length() > 0))**

**{**

**facilityID = facilities.get(i).getFacilityID();**

**pricePerHour = facilities.get(i).getPricePerHour();**

**decommissionedUntilDate = facilities.get(i).getDecommissionedUntilDate();**

**if (decommissionedUntilDate == null)**

**{**

**pw.println(facilityID + "," + facilityName + "," + pricePerHour + ",");**

**} else**

**{**

**pw.println(facilityID + "," + facilityName + "," + pricePerHour + "," + decommissionedUntilDate);**

**}**

**}**

**}**

**pw.close();**

**}**

**public static void updateBookings() throws IOException**

**{**

**/\*\***

**\* The update bookings file will delete the current bookings file and create a new one.**

**\* the method will print the details of any non nullBooking booking.**

**\*/**

**int bookingID;**

**int facilityID;**

**int userID;**

**int slot;**

**String date;**

**String paymentStatus;**

**PrintWriter pw = new PrintWriter(new FileWriter(bookingInfo));**

**for (int i=0; i<bookings.size(); i++)**

**{**

**bookingID = bookings.get(i).getBookingID();**

**if (bookingID > 0)**

**{**

**facilityID = bookings.get(i).getFacilityID();**

**userID = bookings.get(i).getUserID();**

**slot = bookings.get(i).getSlot();**

**date = bookings.get(i).getDate();**

**paymentStatus = bookings.get(i).getPaymentStatus();**

**if (admin || userID==currentUserNum);**

**{**

**pw.println(bookingID + "," + facilityID + "," + userID + "," + slot + "," + date + "," + paymentStatus);**

**}**

**}**

**}**

**pw.close();**

**}**

**public static void recordPayment() throws IOException**

**{**

**/\*\***

**\* this method displays all unpaid bookings and requests the user to pick one.**

**\* the Payment status is then cghanged from N to Y**

**\* The update booking method is then run the update the file.**

**\*/**

**int bookingNum;**

**int bookingID, facilityID, userID, slot;**

**String date, paymentStatus, facilityName;**

**for (int i=0; i<bookings.size(); i++)**

**{**

**bookingID = bookings.get(i).getBookingID();**

**facilityID = bookings.get(i).getFacilityID();**

**userID = bookings.get(i).getUserID();**

**slot = bookings.get(i).getSlot();**

**date = bookings.get(i).getDate();**

**paymentStatus = bookings.get(i).getPaymentStatus();**

**facilityName = facilities.get(facilityID).getFacilityName();**

**if (paymentStatus != "Y" && paymentStatus != null)**

**{**

**System.out.println("Number " + bookingID + ": " + facilityName);**

**System.out.println("Date: " + date + " Time: " + slots[slot-1]);**

**System.out.println();**

**}**

**}**

**bookingNum = Integer.parseInt(JOptionPane.showInputDialog(null,"What is the number of the booking that you would like to mark as paid?",""));**

**bookings.get(bookingNum).setPaymentStatus("Y");**

**updateBookings();**

**}**

**public static void viewStatement()**

**{**

**/\*\***

**\* all bookings by a chosen user are displayed with the faciltyID replaced with the facility name.**

**\* there are two money counters.**

**\* one for the total cost of all the bookings and another for the amount left to be paid.**

**\* the status of whether a booking is paid or not is dosplayed.**

**\*/**

**int x,y,z,q;**

**double g;**

**double toBePaid = 0.0;**

**double total = 0.0;**

**String k,n,m;**

**for (int i = 0;i<bookings.size();i++)**

**{**

**x = bookings.get(i).getUserID();**

**y = bookings.get(i).getFacilityID();**

**k = facilities.get(y).getFacilityName();**

**g = facilities.get(y).getPricePerHour();**

**q = bookings.get(i).getBookingID();**

**if (q > 0)**

**{**

**if (x == currentUserNum)**

**{**

**n = bookings.get(i).getDate();**

**z = bookings.get(i).getSlot();**

**m = bookings.get(i).getPaymentStatus();**

**System.out.println("Facility name: " + k + " Price : " + g + " Date: " + n + " Time: " + slots[z-1] + " Paid: " + m);**

**total += g;**

**if (m != "Y" && m != null)**

**{**

**toBePaid += g;**

**}**

**}**

**}**

**}**

**System.out.println(" Total: " + total);**

**System.out.println(" To Be Paid: " + toBePaid);**

**}**

**public static void chooseUser()**

**{**

**/\*\***

**\* The choose user will read through all the user and add the emails to an arraylist.**

**\* the arraylsit is then turned into an array and displayed in a drop down menus**

**\* when a user is chosen the method will then read through the user objects again until the correct user email is found.**

**\* the userID of that email is then set as the global variable currentUserNum**

**\*/**

**String email;**

**ArrayList<String> userList = new ArrayList<String>();**

**for (int i=0; i<users.size(); i++)**

**{**

**email = users.get(i).getEmail();**

**if (email != null)**

**{**

**userList.add(email);**

**}**

**}**

**if (userList.size() == 0)**

**{**

**JOptionPane.showMessageDialog(null, "No users have been registered yet.");**

**} else**

**{**

**Object[] userArray = userList.toArray();**

**email = (String) JOptionPane.showInputDialog(null, "Users:","",JOptionPane.QUESTION\_MESSAGE, null, userArray, userArray[0]);**

**if ((email != null) && (email.length() > 0))**

**{**

**for (int i=0; i<users.size(); i++)**

**{**

**if (email.equals(users.get(i).getEmail()))**

**{**

**currentUserNum = i;**

**break;**

**}**

**}**

**} else**

**{**

**return;**

**}**

**}**

**}**

**public static void chooseFacility() throws IOException**

**{**

**/\*\***

**\* The choose facilty will read through all the facilties and add the facilityName to an arraylist.**

**\* the arraylsit is then turned into an array and displayed in a drop down menu**

**\* when a facilty is chosen the method will then read through the facilty objects again until the correct facilityName is found.**

**\* the facilityID of that facilty is then set as the global variable currentFacilityNum**

**\*/**

**String facilityName, decommissionedUntilDate;**

**ArrayList<String> facilityList = new ArrayList<String>();**

**for (int i=0; i<facilities.size(); i++)**

**{**

**facilityName = facilities.get(i).getFacilityName();**

**decommissionedUntilDate = facilities.get(i).getDecommissionedUntilDate();**

**if ((facilityName != null) && (facilityName.length() > 0) && (decommissionedUntilDate == null))**

**{**

**facilityList.add(facilityName);**

**}**

**}**

**if (facilityList.size() == 0)**

**{**

**JOptionPane.showMessageDialog(null, "No facilities have been created yet.");**

**} else**

**{**

**Object[] facilityArray = facilityList.toArray();**

**facilityName = (String) JOptionPane.showInputDialog(null, "Menu","",JOptionPane.QUESTION\_MESSAGE, null, facilityArray, facilityArray[0]);**

**if ((facilityName != null) && (facilityName.length() > 0))**

**{**

**for (int i=0; i<facilities.size(); i++)**

**{**

**if (facilityName.equals(facilities.get(i).getFacilityName()))**

**{**

**currentFacilityNum = i;**

**break;**

**}**

**}**

**} else**

**{**

**return;**

**}**

**}**

**}**

**public static void viewBookings()**

**{**

**/\*\***

**\* This method simply displays the details of all bookings.**

**\* the facility id and user id are replaced with the facility name and user email**

**\*/**

**int x,c,m;**

**String y,z,l;**

**for (int i = 0;i <bookings.size();i++)**

**{**

**x = bookings.get(i).getFacilityID();**

**m = bookings.get(i).getUserID();**

**if (x > 0)**

**{**

**l = users.get(m).getEmail();**

**y = bookings.get(i).getDate();**

**z = facilities.get(x).getFacilityName();**

**c = bookings.get(i).getSlot();**

**System.out.println("Facility name: " + z + " User: "+ l + " Date: " + y + " Time: " + slots[c-1]);**

**}**

**}**

**}**

**public static boolean isDateValid(String date)**

**{**

**/\*\***

**\* This method when inputed with a string will check if the string is in the correct format.**

**\* the method also sanity checks the date to ensure that is a real dates**

**\* eg to prevent 32/13/2018**

**\* The method then returns a boolean**

**\*/**

**Calendar cal = Calendar.getInstance();**

**try**

**{**

**Date checkDate = formatter.parse(date);**

**formatter.setLenient(false);**

**formatter.parse(date);**

**cal.setTime(checkDate);**

**return true;**

**}**

**catch (ParseException e)**

**{**

**JOptionPane.showMessageDialog(null, "Invalid date");**

**return false;**

**}**

**}**

**}**