A PROJECT ON BITS LAUNDROMAT MANAGEMENT SYSTEM

CS F213 OBJECT ORIENTED PROGRAMMING

Members:

1.	Chinmay Anand	2020B3A70800P
2.	Devansh	2020B5A72001P
3.	Nachiketh S Shastry	2021A7PS2686P
4.	Rajat Payghan	2021A7PS2218P

Submitted To

Prof. Amit Dua

PLAGIARISM STATEMENT

I am aware that plagiarism refers to using someone else's thoughts, words, creations, or other works as one's own. I am aware that using another person's ideas extensively without giving due credit to them constitutes plagiarism (which includes the proper use of quotation marks). I am aware that using content from online and textual sources in this manner constitutes plagiarism.I acknowledge and understand that plagiarism is wrong. This assignment is my own work, or my group’s own unique group assignment. I acknowledge that copying someone else’s assignment, or part of it, is wrong, and that submitting identical work to others constitutes a form of plagiarism.I have not allowed, nor will I in the future allow, anyone to copy my work with the intention of passing it off as their own work.

NAME: CHINMAY ANAND

SIGNATURE: Chinmay Anand

NAME: DEVANSH

SIGNATURE: [illegible]

NAME: NACHIKETH S SHASTRY

SIGNATURE: Nachi

NAME: RAJAT PAYGHAN

SIGNATURE: Rajat

WORK DIVISION

Name	**Work Done**
Chinmay Anand	**Implemented all the methods in Student Activities and handled errors in the code.**
Devansh	**Implemented all the methods in the Admin Activities and handled the errors.**
Nachiketh S Shastry	**Worked towards SOLID principles and design pattern, also made all the UML diagrams.Worked with the documentation**
Rajat Payghan	**Implemented the GUI and worked towards documentation.Helped in exception handling**

INTRODUCTION

1. AIMS AND OBJECTIVES

A) Students can register for the laundromat system as a new user.
B) Provision for the student to select a wash plan from the menu.
C) Provision for the student to drop off laundry with weight checking only on the allotted day for the student which is pre-assigned to him/her according to his/her hostel.
D) Provision for admin to login and check the status of laundry of each student and obtain the data of each student, update the status for each student, schedule delivery time for return and obtain the total revenue collected from all students.
E) Provision for the student to receive laundry after the admin has updated the status of the order.

2. PROBLEMS ADDRESSED BY THE PROJECT

1. Chances of having a Damaged laundry bag or a missing laundry bag is very minimal as otherwise records can be lost due to human error.
2. Sorting and accessing records is easier and efficient compared to paperwork.
3. Transparency is improved as the user gets notified that laundry has been returned and the admin gets notified that the laundry has been dropped immediately.
4. Errors in tracking revenue can be minimized to great extents.

3. INSTRUCTIONS

1. Download the file(unzip it)
2. Open the file using Eclipse/IntelliJ IDE.
3. Make sure to add the pathname for the csv files in the code wherever the csv file is read in order for the code to compile and run.
4. Navigate to the package Driver and run the Driver.java file which has the main method in it.

<u>DESIGN PATTERN</u>

The source code used in the project is not in accordance with a particular design pattern. The segregation of class is done mainly on the basis of activities of the two users involved in the project. Since there are only two kinds of users (Student and Admin), this segregation does not affect the readability of the code too much. It also has better scope for extension and is not that difficult to modify as well.

Factory design pattern suits the best to laundromat management system project among the various patterns. It is a creational design pattern wherein it allows the subclasses to choose the type of objects to create. It promotes the Dependency Inversion Principle(DIP) as it promotes loose-coupling and allows the code to solely interact with an interface or an abstract class. First of all a Student object gets created after login using factory design . In our code we have a class 'Plans' which stores all the WashPlans in it. When a Student object chooses a particular WashPlan, a new WashPlan object will be created and linked to the Student object.

Similarly, we can create an interface Action which is then implemented by an interface by the name checkStatus and we can then create separate adminCheckStatus and studentCheckStatus classes which implement the checkStatus and create objects or run the overridden methods which follows Single Responsibility Principle and Implementation Segregation Principle.

SOLID PRINCIPLES

1. **The Single Responsibility Principle-**

   a) The SRP states that our class should only have a single reason to change, but in the AdminActivities class, we have used updatestatus() method to update status as well as RA_method() which is used to obtain total revenue.
   b) printRevenue() and checkPresent() are both present in the Customer class but both have different logic.
   c) addToTotalCost() and addWashOrder() are two functions with different logic present in the same class Student.
   d) C_method() and R_method() have different logic present in the class StudentActivities.

   The rest of the code which has classes like StudentLoginAndRegister, WashOrders, WashPlan etc have getters, setters and implementation of similar functions in them.

2. **Open Closed Principle-**

   Interfaces have not been used which otherwise could have been implemented in different classes. By doing this, the code would be open for extension. We cannot add a new feature without making modifications in the classes already existing. This problem is more profound in classes like AdminActivities, StudentActivities etc where extending these classes won’t be a plausible option as they have multiple and diverse functionality in them.
   Otherwise the actions are listed clearly linked with the use case actions classes making it easier for changes in activities for both the student and the admin.

3. **Liskov Substitution Principle-**

   The Liskov Substitution Principle states that subclasses should be substitutable for their base classes. This principle can be violated when we use inheritance and pass an object which is referenced by a superclass reference. But since our code is not extensive in terms of inheritance, there is no scope for the violation of Liskov Substitution Principle.

4. **Interface Segregation Principle-**

   We have segregated the classes based on the activities of each user and login,register for admin and student. We haven’t used interfaces in our code and hence there is no scope for Interface segregation. If Activities interface was created which had a method action() in it we could have created sub-interfaces like CheckStatus with overriding we would implement CheckStatus in classes wherever status check is happening ensuring interface segregation in our code.

5. **Dependency Inversion Principle**

   The Dependency Inversion principle states that our classes should depend upon interfaces or abstract classes instead of concrete classes and functions. The code does not link different classes directly using inheritance but they are linked by passing objects of different classes to a particular class for data storage and data retrieval. Since OCP and DIP are closely related flaws and benefits mentioned in OCP along with usage of abstract classes/ interfaces would make the code follow Dependency Inversion Principle to a greater extent.

GUI IMPLEMENTATION

Our team tried to implement Graphical User Interface for this project, but we were unsuccessful in implementing it. We have uploaded a video explaining the GUI interface which we tried to implement for our project, which was in the prototype stage.

IMPORTANT TASK TO PERFORM BEFORE RUNNING PROGRAM

Please replace the path name of various read file objects in the program before running into your own system. You need to do this replacement at a total of 5 places in the code, and in 3 classes.

These classes are “AdminLogin.java”, “Config.java”, “StudentLoginAndRegister.java” and finally “StudentActivities.java”.

UML DIAGRAMS

CLASS DIAGRAM

The public link for the UML class diagram is - https://drive.google.com/file/d/14OR9mODuZeFx9SShbrfgBjVimwZJ2nGP/view?usp=sharing

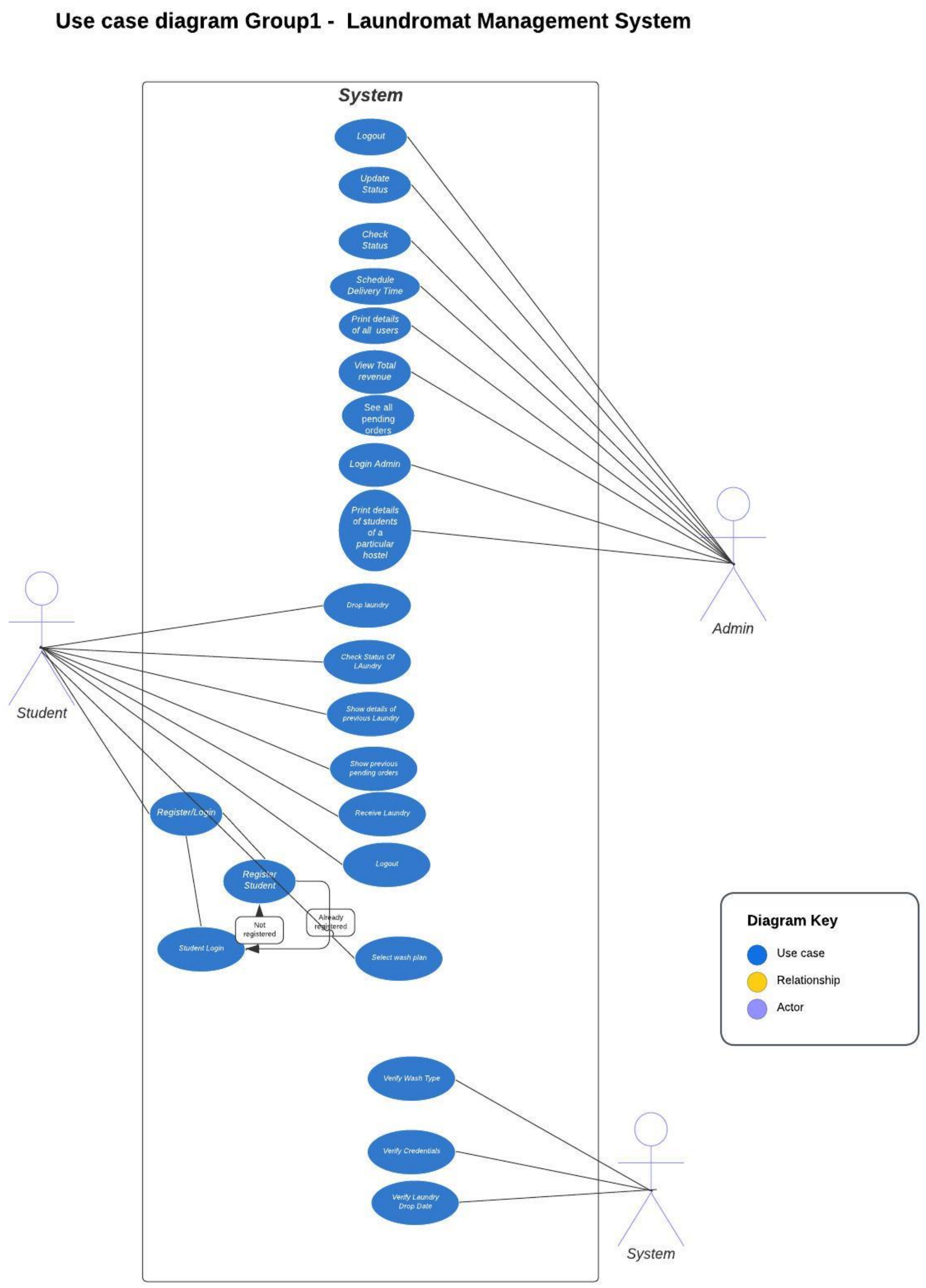
We could not paste the image here because of the very poor resolution of image which rendered it useless for understanding. This link gives high resolution Class diagram

SEQUENCE DIAGRAM

https://drive.google.com/file/d/1CrxXso5pN2HytvrHxu8qhwIcAHqSelpy/view?usp=sharing

Click on the link to access the sequence diagram.

<u>CASE DIAGRAM</u>



GOOGLE DRIVE LINK

Google Drive link for all explanation video of all out members and contains all the UML diagrams pdf which are used in this document
https://drive.google.com/drive/u/0/folders/1EsrggcUFb3wDXLudJiQdGpeE7g6AIqaH