LAUNDROVALLEY

Table of Contents

1. Introduction

In large college campuses like BITS Pilani ,there is no central laundry management. There are local laundry facilities but getting information about the pickup and delivery of washed clothes requires the individual to call the laundry services, also the other information like - when will the laundry service man visit the hostel for picking up soiled clothes etc.
LaundroValley is a central laundry management system specially designed for students and laundry service providers. It is a central platform where the laundry service provider meets the college students.

1.1 Scope

This software will be a Laundry Management System for a Vendor of Laundry Services. This software will be designed to maximize the Vendor's productivity by providing tools to assist in automating various processes which would otherwise have to be performed manually. By maximizing the vendor's work efficiency and production the system will meet the vendor's needs while remaining easy to understand and use.
More specifically, this system is designed to allow a Vendor to manage Customers. The system also contains a relational database containing a list of Customers (students in our case), Vendor's Staff and Subscription Plans.
This System also provides a way to provide feedback so that the vendor can improve its services.

1.2 Intended Audience

- Laundry Service Vendor
- Students

1.3 Product Flow

- The Student will login via login page.
- After Logging in, if the student hasn't subscribed a plan then he/she can subscribe a plan.
- After Subscribing a plan, the student can schedule a wash by clicking on the schedule wash button.
- On the Schedule wash page, Student needs to enter the details about number of clothes, Date etc.
- Once the wash is scheduled, a staff will receive a notification about this newly scheduled wash.
- Alternatively, the student can check the status of the wash, for this student needs to enter the wash id.
- After the wash is completed, student can provide a feedback or even write a complaint for the service.

1.4 Product Functions

Functionalities:

- *Student/staff Registration*
- *User Authentication*
- *Get Subscription plans*
- *Schedule Wash*
- *Enquire Wash Status*
- *Plan Expiry notification*
- *Feedback Provision*
- *Manage User*
- *Complaint Registration*

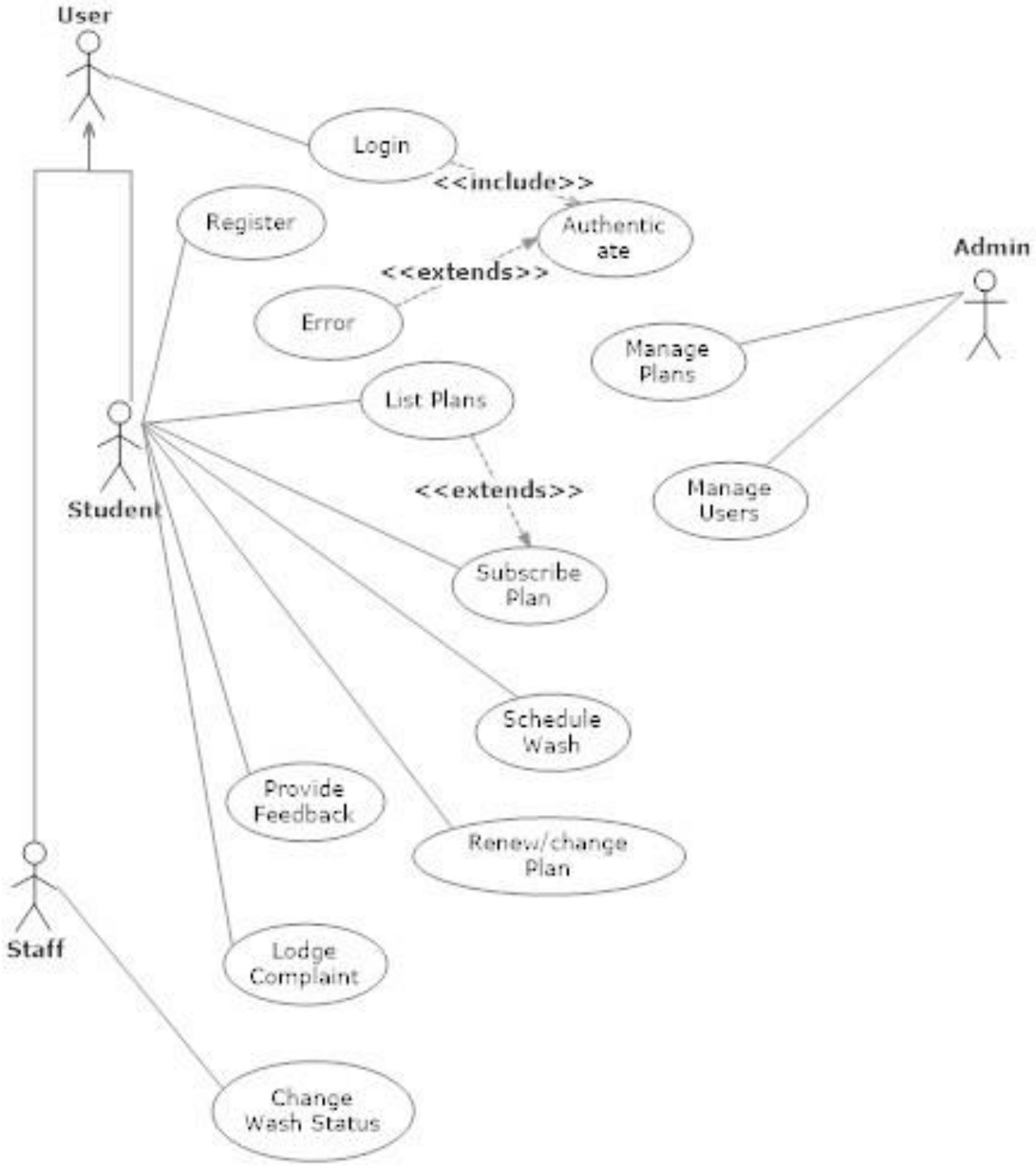
1.5 Technologies Used

Following are the technologies/frameworks that we have used in this project: -

- **Java**
- **JavaScript/jQuery**
- **HTML5/CSS**
- **MySQL**
- **Spring MVC**
- **Maven**
- **REST API**
- **Hibernate**
- **Bootstrap**

2. UML Modeling

2.1 Use Case Diagram


User
Login
<<include>>
Authentic
ate
Register
<<extends>>
Error
Admin
Manage
Plans
List Plans
Student
<<extends>>
Manage
Users
Subscribe
Plan
Schedule
Wash
Provide
Feedback
Renew/change
Plan
Staff
Lodge
Complaint
Change
Wash Status

2.2 Use Case Template

Following are the fully dressed use case templates of the main use cases of our application.

2.2.1 Schedule Wash Use Case

Use Case Name:	Schedule Wash
Scope:	LaundroValley
Level:	User Goal
Primary Actor:	User (Student)
Stakeholders:	• Student - Wants quick scheduling of wash according to his subscription plan without any failure. • Staff - Wants to be notified about new washes without delay. • Vendor-Wants smooth functioning of schedule wash service.
Preconditions:	User should be logged in and must have a plan subscribed.
Success Guarantee:	Wash is scheduled. Staff is notified about the new wash. Pickup time is scheduled.
Main Success Scenario:	1. Initiate the wash. 2. Provide the number of clothes etc. 3. Once submitted the details, user will get expected pickup time. 4. Staff will be notified about the wash as soon as student schedules the wash.
Alternative Scenario Extensions:	2.a Student provides custom information for clothes that needs to be ironed if he/she do not have (wash + iron) subscription plan. Amount for custom iron will be added to his expenses. 3.a Student can also provide custom pickup time according to him.
Special Requirements	N/A
Frequency	Very Often

2.2.2 Renew/New Plan Subscription Use Case

Use Case Name:	Renew/New Plan Subscription
Scope:	LaundroValley
Level:	User Goal
Primary Actor:	Student
Stakeholders:	• Student – Wants to subscribe a plan which suits him/her best according to his/her needs. • Vendor – Wants to know which plan is being subscribed and which is not so that plans can be modified.
Preconditions:	User is logged into the system.
Success Guarantee:	Plan is subscribed by the student. Amount is updated into his expenses.
Main Success Scenario:	1. Student logs into the system. 2. Student navigate to plan renewal. 3. Student selects renew old plan and provides duration. 4. System provides the amount for the plan. Student confirms the payment. 5. System provides payment receipt to the student. 6. Student downloads the receipt if he wishes to. He can also retrieve his payment receipt anytime in the future saved into his account,
Extensions:	1.a If user enters invalid credentials, display appropriate error message to the user. 3.a Student can select new plan. In this case all plans will be listed by the system. User selects any plan of his choice and provides duration.
Special Requirements:	N/A
Frequency:	Often.

Following is the general flow of all the other non-critical use cases.

2.2.3 Register User

1. Student Visits Login page
2. Click to register as new user

3. Enter identifying information: username, email, Name, password (twice)
4. Submit form
5. Verify email

2.2.4 Log in

1. Student Visits Login page
2. Provide Credentials
3. Login

2.2.5 List Subscription Plans

1. Student Visit LaundroValley's page
2. Navigate to Plans.
3. Click on List Subscription Plans

2.2.6 Change/Forgot Password

1. Student Visits the Home/login Page.
2. Login and click on change password in account settings tab.
3. Click on Chang Password.
4. Provide Username, Old Password, New Password(twice).
5. Request for New Password in case of Forgot password.
6. Provide OTP Received via Email.

2.2.7 Enquiry of Wash

1. Student Navigate to My Washes
2. Select the ongoing/live wash.
3. Check the status of the wash.

2.2.8 Changing Wash Status

1. Staff Navigates to Washes.
2. Provide wash id.
3. Change the status of wash. For example, scheduled(default), picked, ongoing, completed etc.

2.2.9 Expiry Notification

1. Notification will be sent to the user (student in this case) just before the plan is about to expire.

2.2.10 Provide Feedback

1. Once the wash is completed (wash status changed to delivered), user can provide feedback.
2. User Navigates to feedback and provide feedback.

3. User can rate the wash, provide answer to some basic questions and provide comments if any.
4. These feedbacks can be viewed globally by the users.

2.2.11 Manage User Profile

1. User Navigates to his profile.
2. He can add/remove/update his profile details.

2.2.12 Payment

1. The Student will navigate to Plans using list plans feature.
2. Click on the subscribe button associated with a particular plan.
3. The user will be redirected to Payment page.
4. Complete payment process
5. Show the expiry date
6. Send Receipt via Email

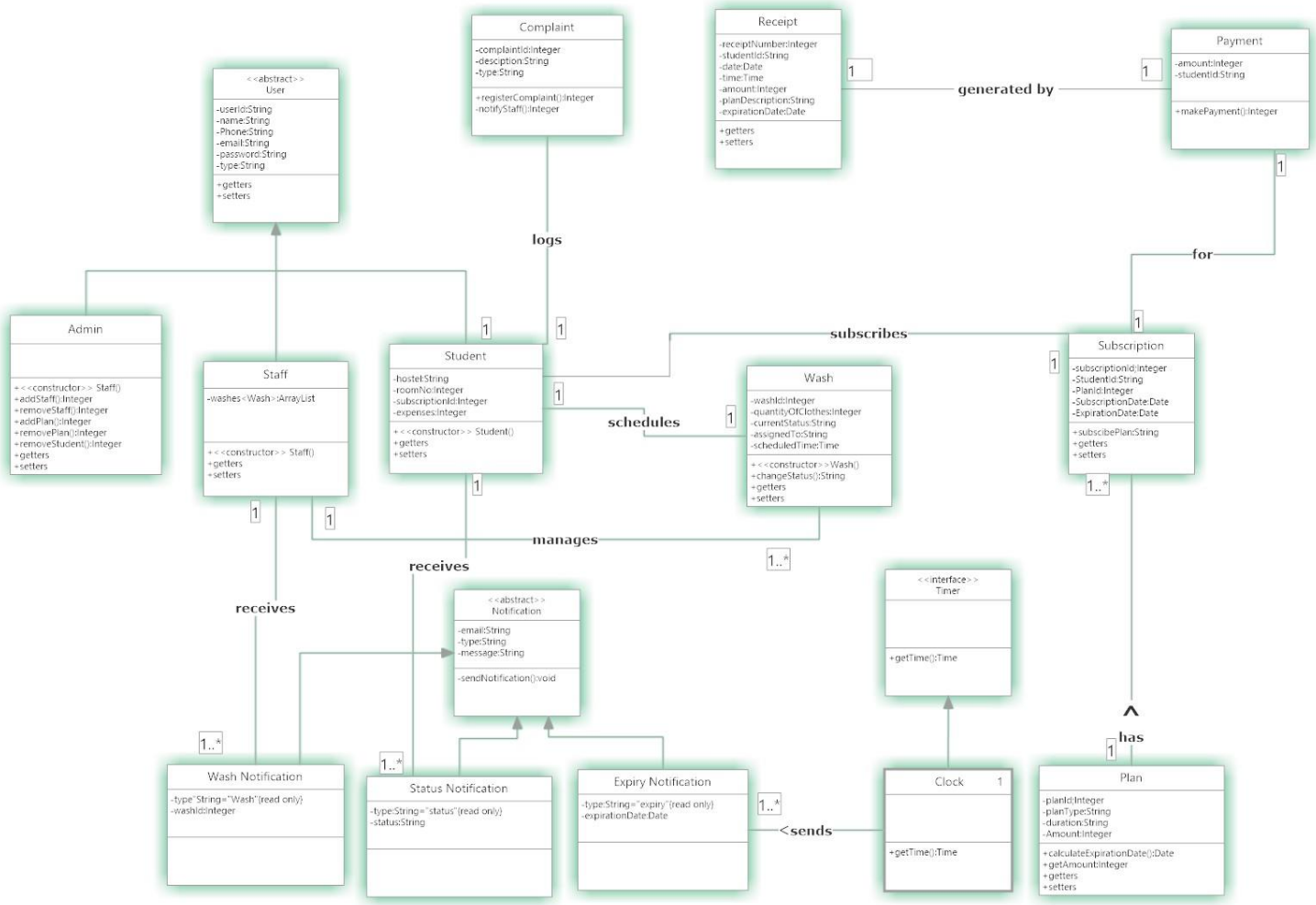
2.2.13 Complaint

1. Student navigates to complaint section.
2. He can lodge complaint.
3. The complaints will be addressed by staff.
4. Complaint status will also be managed by the staff which can be viewed by the user.

2.2.14 Complaint

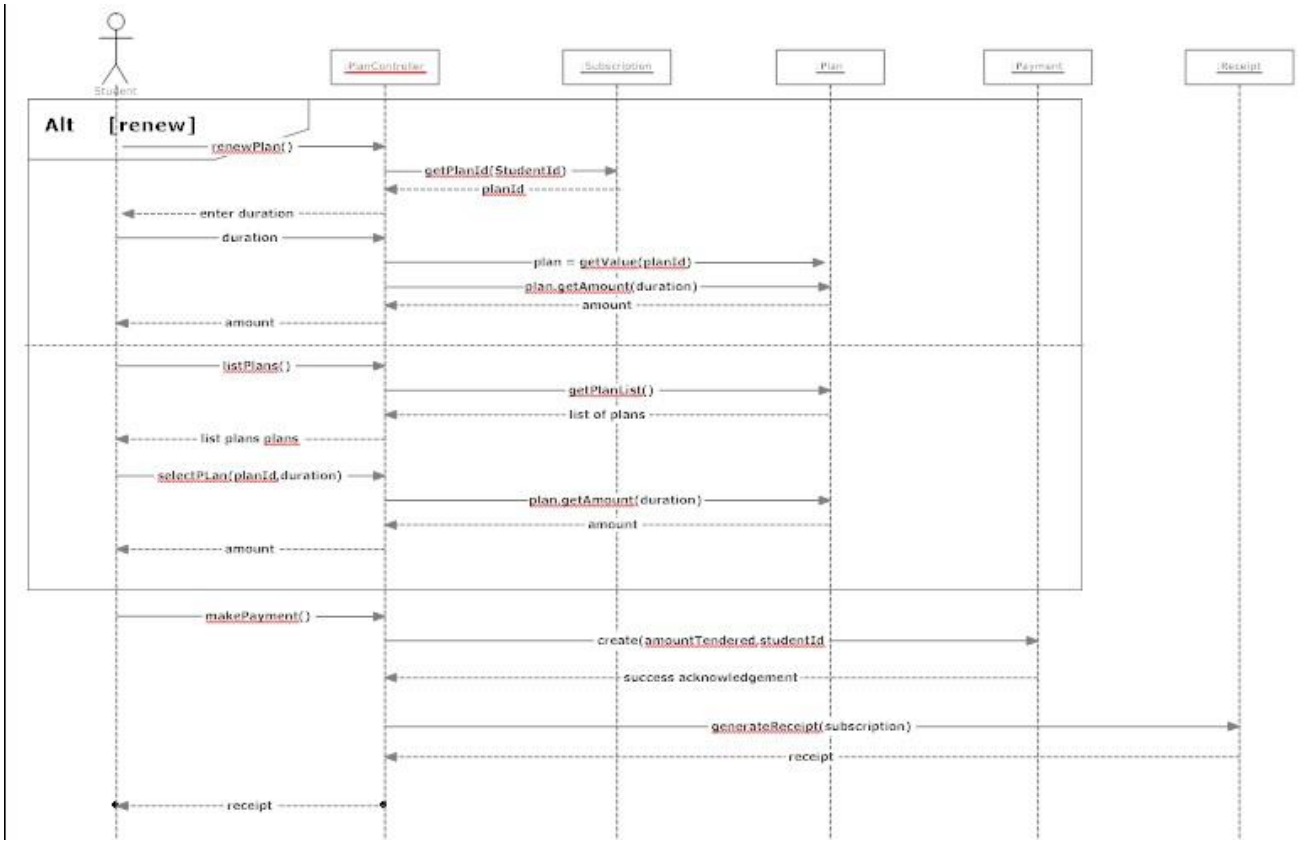
1. Staff Navigates to complaints.
2. List all the active complaints.
3. Address the complaints and change the status.
4. After the issue has been resolved, it will be verified by the user(student).

2.3 Design Class Diagram

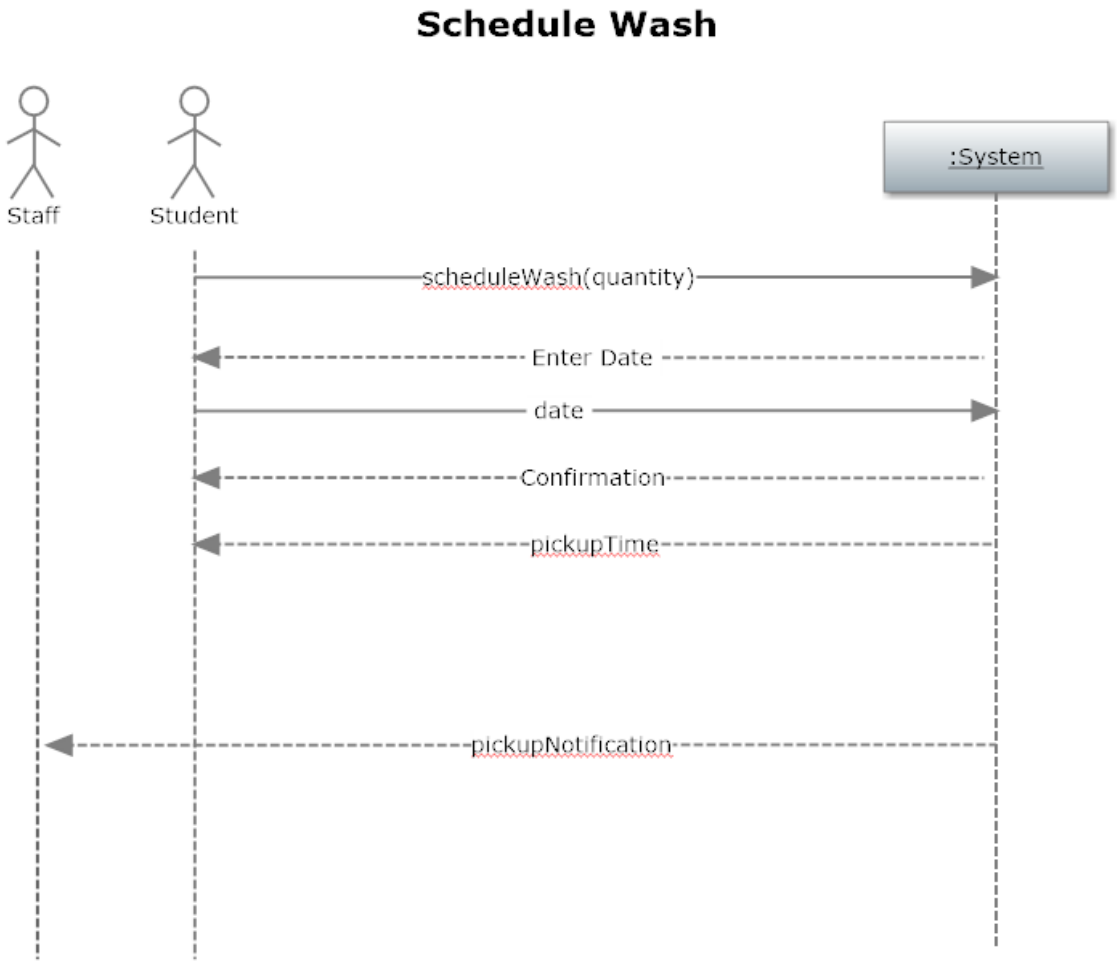

Complaint
-complaintId:Integer
-description:String
-type:String
+registerComplaint():Integer
-notifyStaff():Integer
Receipt
-receiptNumber:Integer
-studentId:String
-date:Date
-time:Time
-amount:Integer
-planDescription:String
-expirationDate:Date
+getters
+setters
generated by
Payment
-amount:Integer
-studentId:String
+makePayment():Integer
<<abstract>>
User
-userId:String
-name:String
-Phone:String
-email:String
-password:String
-type:String
+getters
+setters
logs
for
Admin
+<<constructor>> Staff()
+addStaff():Integer
+removeStaff():Integer
+addPlan():Integer
+removePlan():Integer
+removeStudent():Integer
+getters
+setters
Staff
-washes<Wash>:ArrayList
+<<constructor>> Staff()
+getters
+setters
Student
-hostel:String
-roomNo:Integer
-subscriptionId:Integer
-expenses:Integer
+<<constructor>> Student()
+getters
+setters
subscribes
schedules
Wash
-washId:Integer
-quantityOfClothes:Integer
-currentStatus:String
-assignedTo:String
-scheduledTime:Time
+<<constructor>> Wash()
+changeStatus():String
+getters
+setters
Subscription
-subscriptionId:Integer
-StudentId:String
-PlanId:Integer
-SubscriptionDate:Date
-ExpirationDate:Date
+subscribePlan:String
+getters
+setters
manages
receives
receives
<<abstract>>
Notification
-email:String
-type:String
-message:String
-sendNotification():void
<<interface>>
Timer
+getTime():Time
has
Wash Notification
-type"String="Wash"{read only}
-washId:Integer
Status Notification
-type:String="status"{read only}
-status:String
Expiry Notification
-type:String="expiry"{read only}
-expirationDate:Date
sends
Clock
+getTime():Time
Plan
-planId:Integer
-planType:String
-duration:String
-Amount:Integer
+calculateExpirationDate():Date
+getAmount:Integer
+getters
+setters

2.4 Sequence Diagram

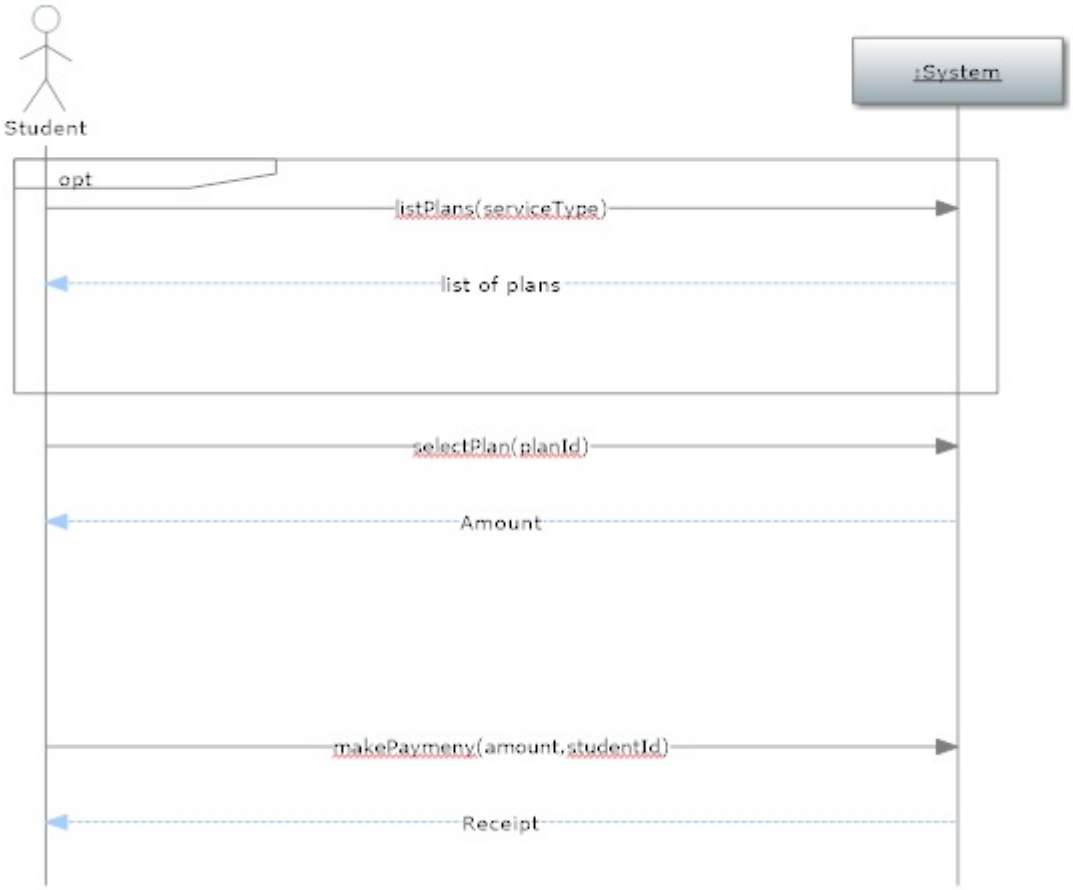
Following is the sequence diagram for **Renew/New Subscription plan** .



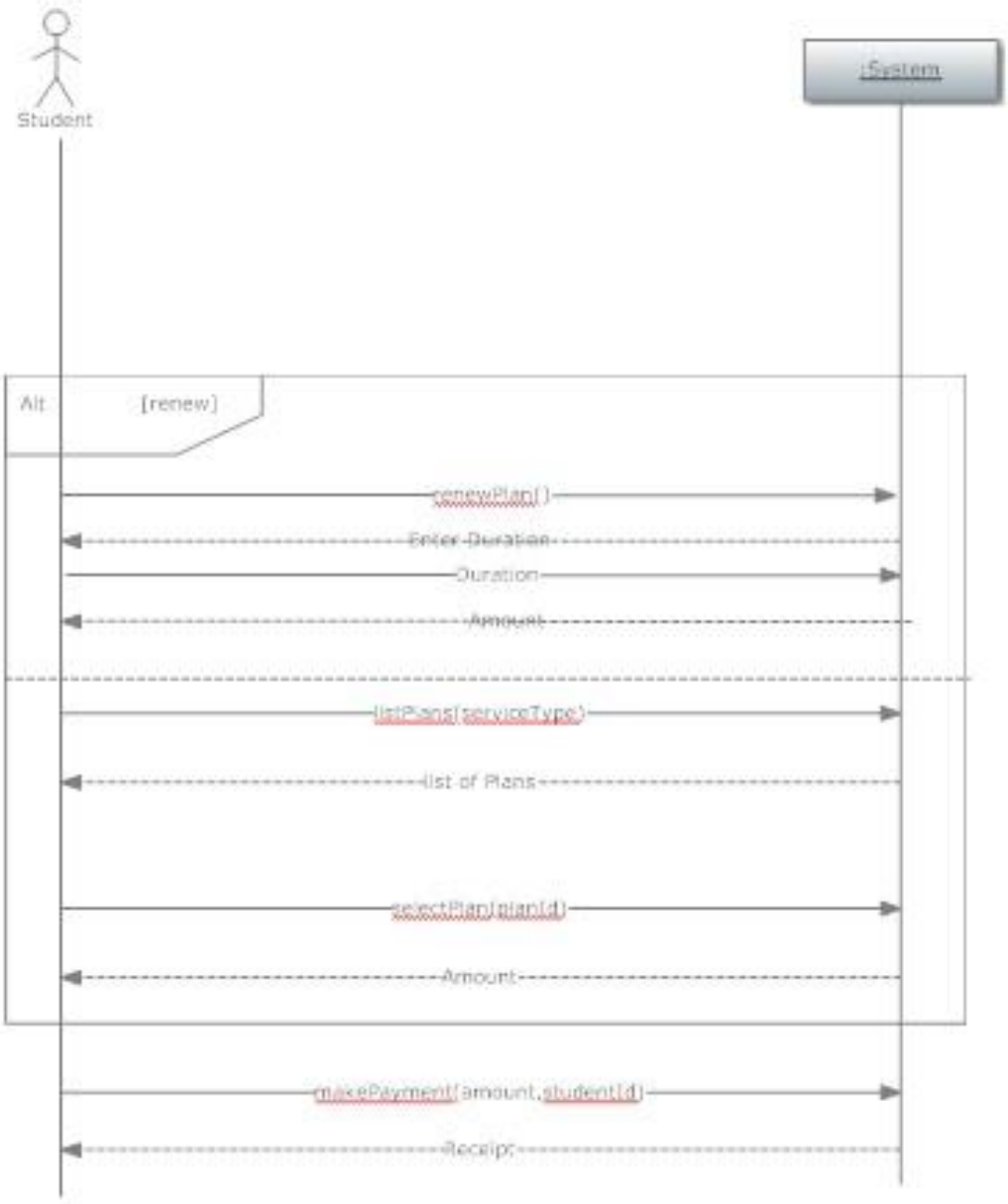
2.5 System Sequence Diagrams

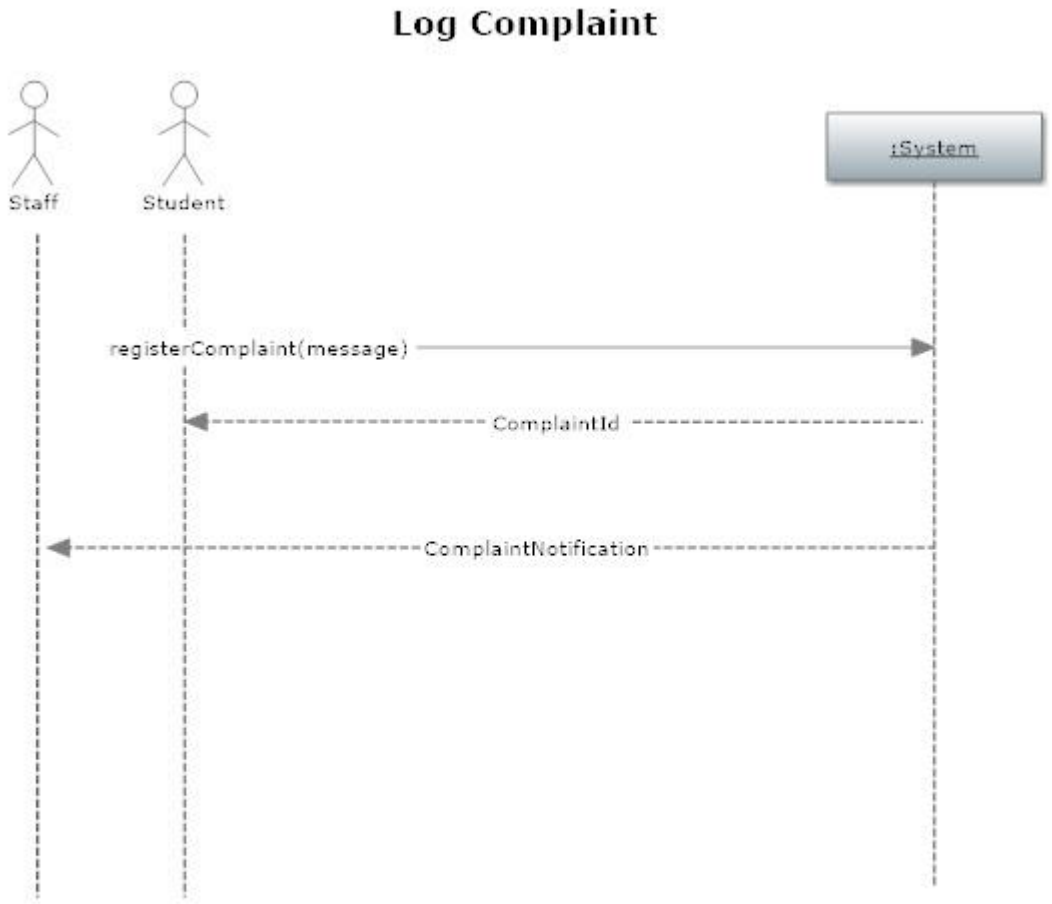


Subsribe Plan

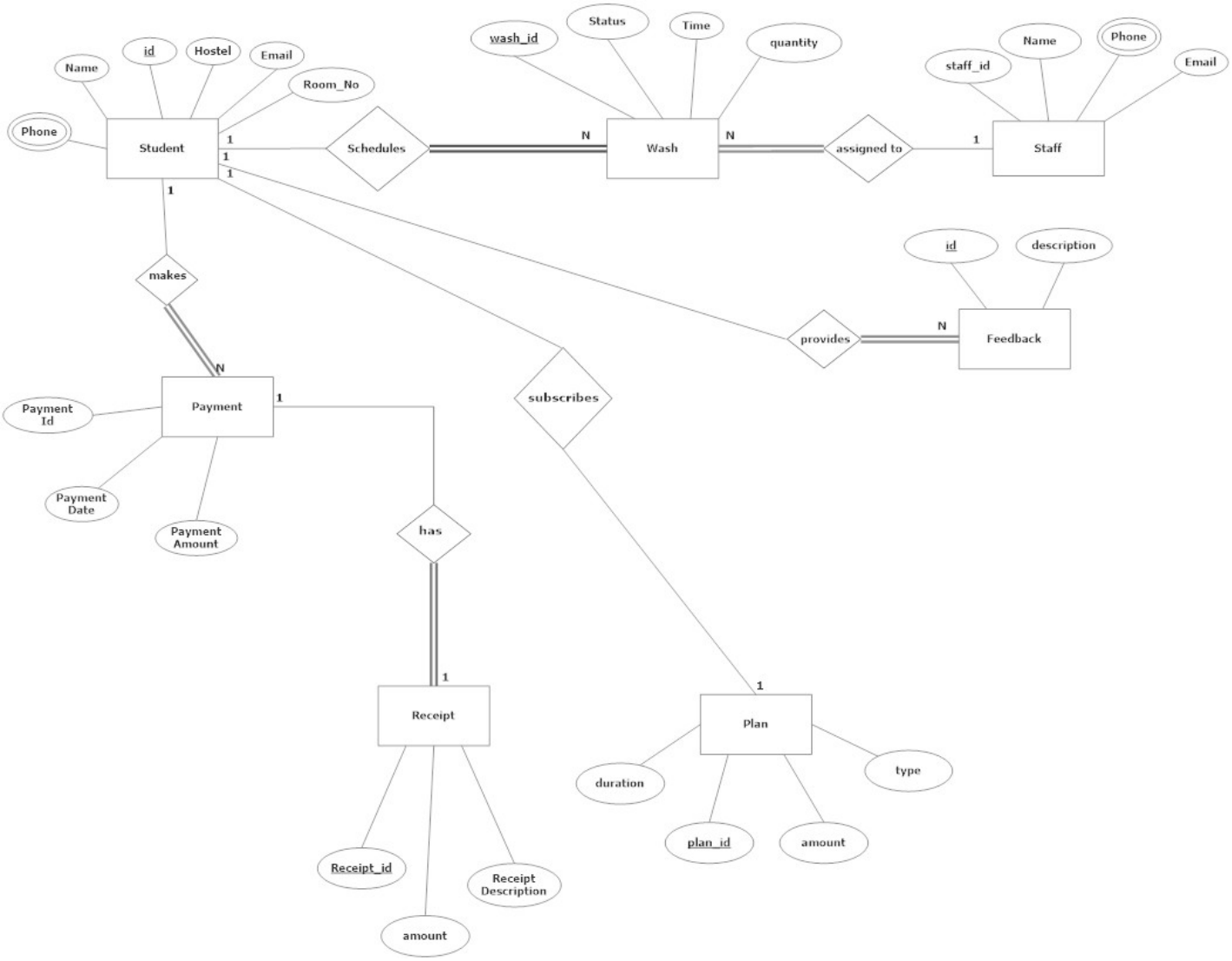


Plan Renewal/Change




Log Complaint
Staff
Student
:System
registerComplaint(message)
ComplaintId
ComplaintNotification

2.6 ER Diagram

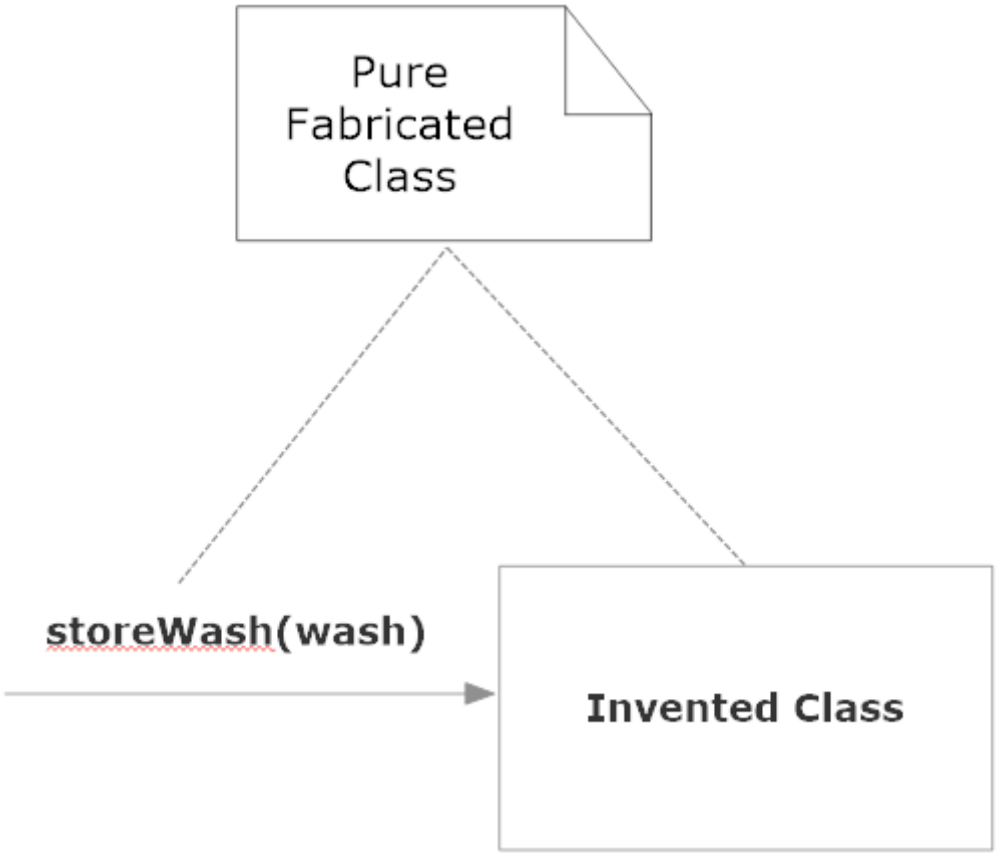


3. Design Patterns Used

Following are the design pattern that we have identified so far. Their explanation based on our projects are as:

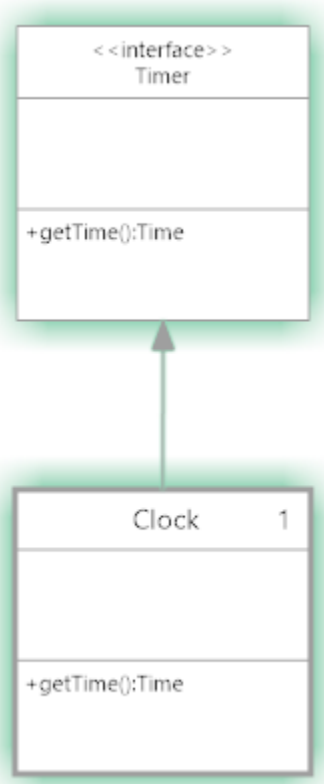
3.1 Pure Fabrication

Introduction of Persistant Storage Class for Storing and retreiving information to and from the Database which will reduce Coupling and Increase Cohesion.
For example :
In order to store information about the newly scheduled Wash instead of assigning the responsibility of storing this information in the database to the Wash Class, we can deligate this responsibility to a made up Persistant Storage Class.



3.2 Singleton Design Pattern

Each payment mode like credit & debit card will have their different implementation of makepayment().



3.3 Observer Pattern

Every time a new wash is scheduled by a student, a staff should be notified about this newly scheduled wash. To achieve this, we will be using Observer design pattern in which our subject is Wash and observer will be staff.