ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

Object Oriented Development with Java (CT038-3-2-OODJ)

Students Name	**: Ching Cheng Kang (TP051436)**
	Ashween Yogeswaran (TP051262)
Lecturer	**: Dr.Kadhar Batcha Nowshath**
Intake Code	**: UC2F2008SE**
Submission Date	**: 22nd November 2020**

Table of Contents

1. Introduction

The courier system is a system created to make the working environment work efficiently without having any drawbacks. There are 2 different users in this system one of it is the managing staff and the other user is the delivery staff. All the staff are registered with email and password to login into this system. Besides, the managing staff will be able to conduct multiple tasks through this system meanwhile the delivery staff will be able to conduct 2 tasks like profile edit and delivery management. Moreover, the managing staff will be to able manage order, manage report, manage feedback and setup staff directory. As an overview, this system would be the right system to use it as a courier service system.

2. UML Diagram

2.1. Class Diagram

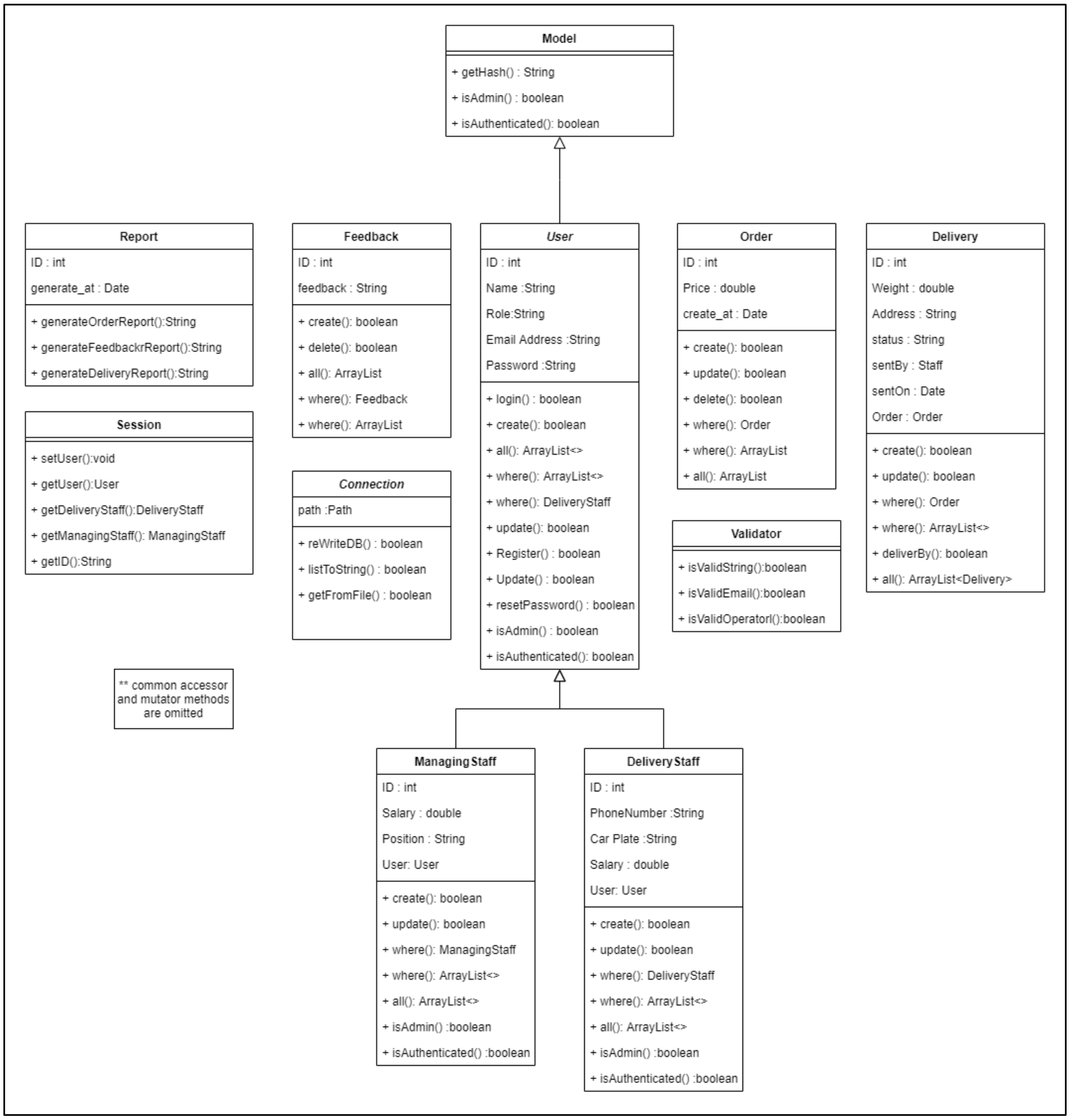


Figure 1 Class diagram of the project

2.2. Use Case Diagram

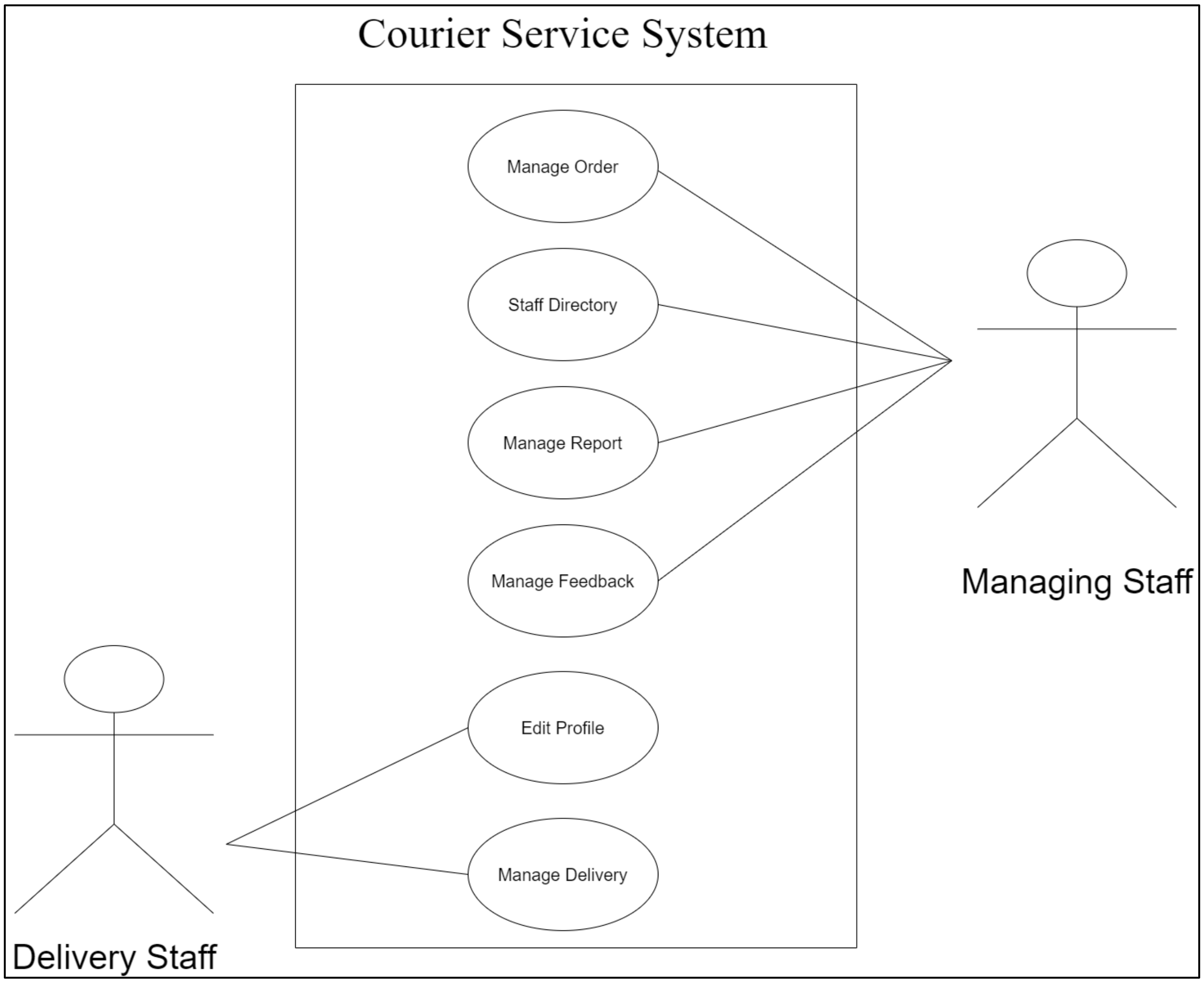


Figure 2 Class diagram of Courier Service System

2.3. Use case specification

Use case name	Generate Report
Scenario	Generating report based for order, delivery, and feedback.
Triggering event	Admin wants to analyse the order, delivery and feedback workflow.
Actors	Managing Staff Admin
Pre-Requirement	• Must be registered as admin • Order data, Delivery data and Feedback data will be shown
Post Condition	• System will display any changes made • Report will pop out once button selected for specific report to generate
Flow of activities	Actor • Login the system. • Able to go through order data, delivery data and feedback data. • Make changes in data if needed. • Update once done with changes. System • Verify the admin • Display all data in table form • Pop up message when changes made.
Validation	Display report once select button
Exception Conditions	If the username or password does not exist, this use case will be aborted.

Use case name	Login
Scenario	The system provides different privileges to different users.
Triggering event	Login with different identity will provide different task for each user
Brief Description	Logging to the system with right credentials
Actors	Managing staff and delivery staff
Pre-Requirement	• Must be registered as managing staff or delivery
Post Condition	• Able to go through the system once login
Flow of activities	**Actor:** • Login the system with the right password and id as registered. • Once the id and password us verified, user will be able to access the system. **System:** • Verify the user • System displays "login successful"
Validation	The system displays message "login successful" once user logged in with the correct id and password
Exception Conditions	If the id or password is incorrect this use case will be aborted.

Flow of activities:

Actor	System
• Login the system with the right password and id as registered. • Once the id and password us verified, user will be able to access the system.	• Verify the user • System displays "login successful"

Use case name	Manage Delivery
Scenario	Manage delivery of every order using the system
Triggering event	Once new orders have been place, they are ready to be delivered
Brief Description	The manage order will be viewed to identify the delivery staff and other information
Actors	Managing staff
Pre-Requirement	• Must be registered as managing staff
Post Condition	• New delivery will be updated
Flow of activities	**Actor:** • Login the system with the right password and id as registered. • Once the id and password us verified, user will be able to access the system. **System:** • Verify the user • System displays “login successful”
Validation	The system displays message “login successful” once user logged in with the correct id and password
Exception Conditions	If the id or password is incorrect this use case will be aborted.

Flow of activities:

Actor	System
• Login the system with the right password and id as registered. • Once the id and password us verified, user will be able to access the system.	• Verify the user • System displays “login successful”

3. OOP Concept

Object oriented programming is a paradise of programming fundamental in the model development where is used by almost every developer today. Up to oriented programming is the most popular programming paradigm today any is most preferred by most of the programmer. There is multiple principle that should be follow where will be discussed here.

https://www.educative.io/blog/object-oriented-programming

3.1. Abstraction

Abstraction is one of the most important principle hey after enter programming where the user only interacts with the selected attribute in method off the object. The man advantages of attraction are use the simple concept to represent a complex idea and it also had complexity deal from the user in order to increase the usability.

3.1.1. Abstract Class

```
abstract public class Model implements Queryable, Creatable, Updatable, Validable {
    abstract public String getHash(byte[] inputBytes);
        …omitted…
}
```

Figure 3 Abstract class Model

In the above code snippet, it is an abstract class call Model, and it has also had an abstract method name as getHash an require a byte array as its argument and return a string. This will limit and standard the class that inherit from this abstract class must have a method with the same signature in order to achieve the purpose of abstraction.

```java
public String getHash(byte[] inputBytes) {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(inputBytes);
            byte[] bytes = md.digest();
            StringBuilder passwordInString = new StringBuilder();
            for (int i = 0; i < bytes.length; i++) {
                passwordInString.append(Integer.toString((bytes[i] & 0xff) + 0x100, 16).sub
string(1));
            }
            //Get complete hashed password in hex format
            return passwordInString.toString();
        } catch (NoSuchAlgorithmException e) {
            System.out.println(e.getMessage());
        }
        System.out.println("STH wrong la brop");
        return null;
    }
```

Figure 4 abstraction method of getHash()

In the above code snippet that exits on the User class which inherit from the Model class.

3.1.2. Interface

```
public interface Creatable {
    public boolean create();
}
```

Figure 6 Creatable Interface

```
public interface Updatable {
    public boolean update();
}
```

Figure 7 Updatable Interface

```
public interface Queryable {
    public Object where(String type, String queryOperator, String queryString);
    public Object where(String type, String queryString);
    public Object all();
}
```

Figure 5 Queryable Interface

Above code snippet shows few of the interface in the project, which is Creatable interface, Updatable interface and Queryable interface respectively. Therefore, any class or model that inplements the interface must have the method that matches the signature and override otherwise an error will occur. In the Queryable interface, polymorphism concept is applied to increase the functionality and increase the abstraction for developer to use the system.

3.2. Encapsulation

Encapsulation is one of the concepts of object-oriented programming where it means it will content all of the information within the object and expose selected attribute to be open. Encapsulation is very important in order to prevent illegal access. Therefore, the behavior and attribute or declare within the class template. After that, When the object is initiated for a class, the methods in the data will be encapsulate in the object and all of the internal operations will be hided.

```
private int ID;
private String phoneNumber;
private String carPlate;
private Double salary;
private User user;
```

Figure 8 private attribute of Delivery Staff

By using public, private and protected keyword, we can change different behaviour of the attribute and the accessibility of the attribute or method. In the above code snippet, we use private key word to ensure it can only access within the class only.

3.2.1. Getter (Accessor)

```
public int getID() {
    return ID;
}

public Double getPrice() {
    return price;
}
```

Figure 9 Example of getter of Order

As id and price is private attribute, in order to access or get the data from this class, a getter must be set in order to access the attribute from the other classes

3.2.2. Setter (Mutator)

```
public void setPrice(Double price) {
    this.price = price;
}
public void setPayAt(LocalDateTime payAt) {
        this.payAt = payAt;
    }
```

Figure 10 Example of setter in Order Class

In the above code snippet, it shows 2 setters for price and pay at in Order class. These 2 setters allow the developer to set the respective attribute as an operation such as creation of a model as shown below.

```
Order order = new Order();
order.setPrice(Double.valueOf(price));
order.setPayAt((jCheckBox1.isSelected()) ? LocalDateTime.now() : LocalDateTime.MIN);

if (order.create()) {
    JOptionPane.showMessageDialog(null, "Saved");
    new ManageOrder().setVisible(true);
    this.dispose();
}
```

Figure 11 Example of creation of Order in ManageOrder.java

3.3. Inheritance

Another concept of object-oriented programming is inheritance concept where evil enable the classes to inherit feature and information from other classes as well as extended the attribute the behavior to the children classes. The biggest advantages of this concern are the usability of the code snippet it from the parent class in order to achieve the practice of DRY (Don't repeat yourself) and reusability.

Several types of inheritance is implemented in this project including single level inheritance, multi-level inheritance and hierarchical inheritance. (SOFTWARETESTINGHELP, 2020)

3.3.1. Multi-level

```
abstract public class Model implements Queryable, Creatable, Updatable, Validable {
…omitted…
}

public class User extends Model {
…omitted…
}

public class ManagingStaff extends User {
…omitted…
}
```

Figure 12 Example of Multi level inheritance

In the above code snippet, it demonstrates 3 classes that inherit from each other. In the above example, Model is the Super Class where it uses 4 Interface to achieve abstraction concept which is Queryable, Creatable, Updatable, Valdiatable. After that User class is inherited the Model Class where it will store common attribute and behaviour of User. Last but not lease, ManagingStaff Class extends User class where it will be a multi-level behaviour inheritance.

3.3.2. Hierarchical Inheritance

```
public class User extends Model {
…omitted…
}

public class ManagingStaff extends User {
…omitted…
}

public class DeliveryStaff extends User {
…omitted…
}
```

Beside multi-level inheritance, hierarchical inheritance also implemented in this project. In the above code snippet, User class is the parent class of Managing Staff and Delivery and it is a type of hierarchical inheritance.

3.4. Polymorphism

Polymorphism is one of the Object-oriented Programming concept where design the object to share same behaviour. Technique such as overloading or overriding allowed it to share the parent behaviour with specific behaviour.

3.4.1. Overloading

```
public User where(String type, String queryString) {

…omitted…
}

public ArrayList<User> where(String type, String queryOperator, String queryString) {
…omitted…
}
```

Figure 13 Example of Overloading

In the above code snippet, it displays a technique of overloading of same method name call as "where" to find a specific record. The return type and the argument of each method is different in order to change the behaviour of the method.

In the First method , the first argument can accept the type of query such as "id", "name", "email" and etc where the second argument can accept the string that need to be query such as "1", "staff Name", "admin@admin.com" and etc. This method is only query one single record from the model, so it only returns the User object itself and its use.

```
String userId = model.getValueAt(i, 0).toString();
User u = new User().where("id", userId);
u.setEmail(email);
u.setName(name);
u.setRole(role);
if (!u.update()) {
    status = false;
}
```

Figure 14 Example of use case in StaffDirectory.java

In the second method, the first argument and the last argument are same as the same with the first method, and the second argument accept any comparison operator such as “>”, “<”, “=”, “>=”, “<=” to match the condition of the query type. This method is used to query one or more then one model which is used at the used at bottom.

```
@Override
    public ArrayList<User> all() {
        return this.where("id", ">=", "1");
    }
```

Figure 15 Example method that use the second method in User.java

In the above method, it will query the user because it will query user that user id is equal or more than 1which is every user.

3.4.2. Override

```
abstract public class Model implements Queryable, Creatable, Updatable, Validable {
    public boolean isAdmin() {
        return false;
    }
…omitted…
}
public class User extends Model {
    public final String MANAGING_ROLE = "admin";
    @Override
    public boolean isAdmin() {
         return this.MANAGING_ROLE.equals(this.role);
    }
…omitted…
}
```

Figure 16 Example of Override

In the above code snippet, there is one method in the abstract class call as isAdmin() where it return will Boolean. On the other hand, User class that inherit Model also have same method with override tag having a different behaviour from the original class and it will check if the object's role is equal to the variable of MANAGING_ROLE.

4. Conclusion

To conclude, we were able to develop a fully functional Courier Service System. This system will make workflow of the courier service company way better and faster to complete task. Besides, they will be able to coordinator their workers in a better manner and efficiently Throughout this project we have learn many things and gain extra knowledges where it will be able to apply in the future. Is has been one of the most important this semester because it will give us a lot of experience completing this project and java is also one of the top programming language in the world so this project had been a great opportunity to under object oriented with java. We will be able to apply all this knowledge in our future working environment if we will assign to complete java projects. We have learn the concept of OOP and done a project based on it as well based on this we have sharpen of skill related to java with objected oriented programming which will be a great advantage in the IT world.

5. User Manual

5.1. Login Form

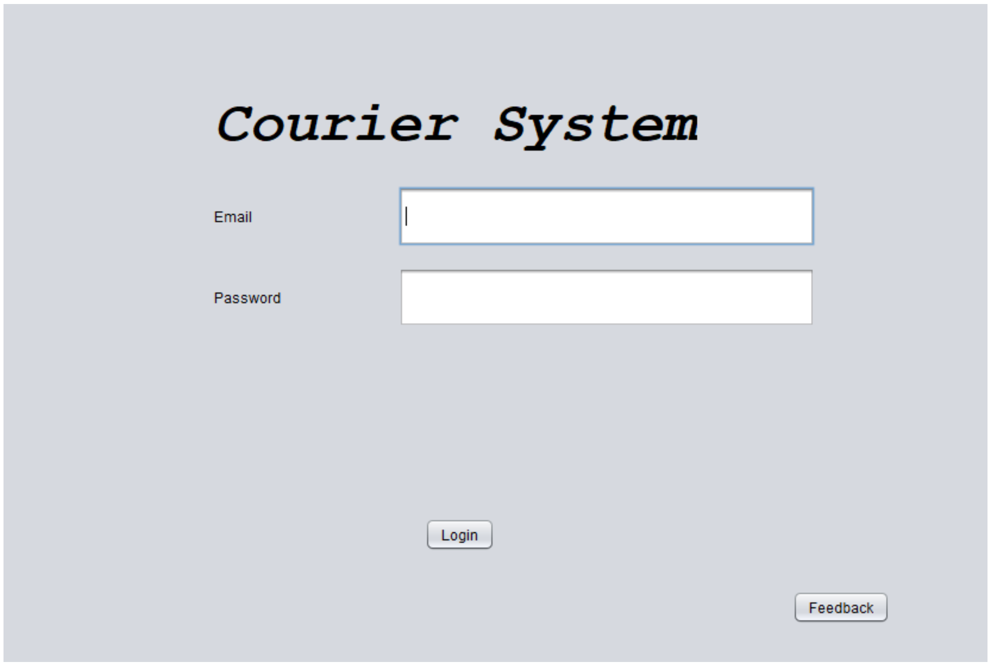


Figure 17 Screenshot of Login Form

This is the login section of the Courier Service System. Users must fill your email and password to login to the system and then select the login button. There is a feedback button at the right bottom where when it is selected it will direct the user to the feedback section.

5.2 Add Feedback Form

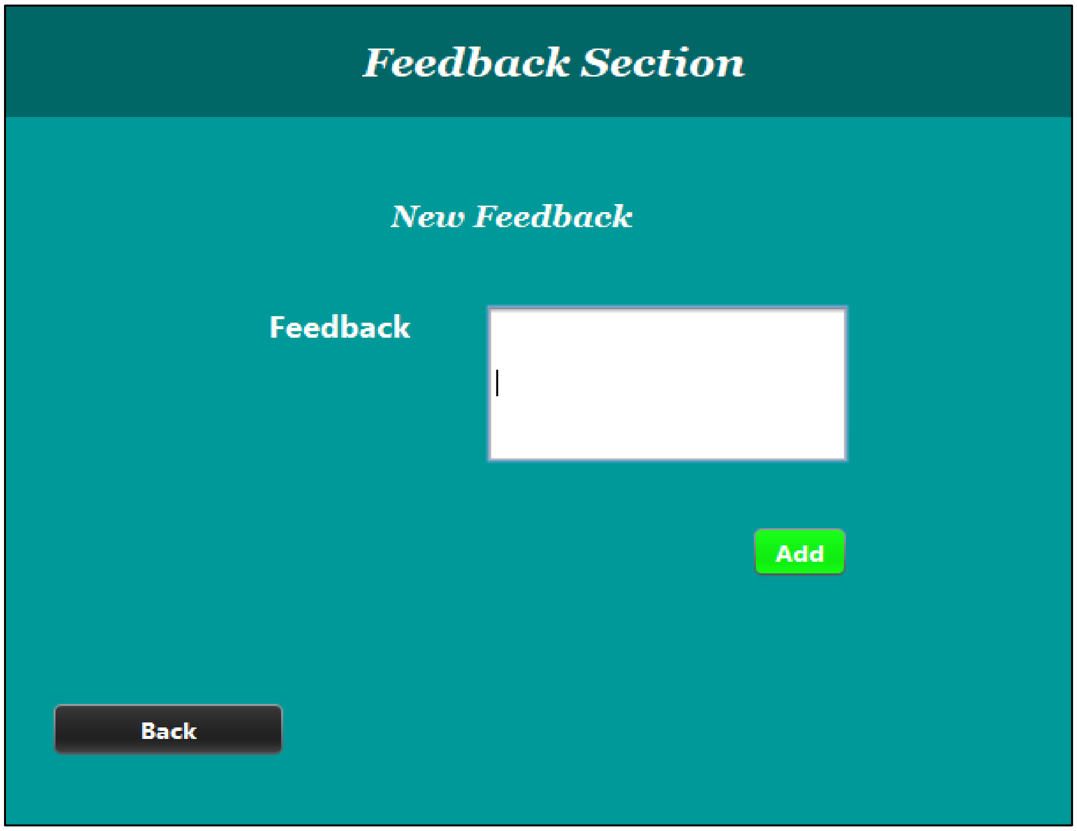


Figure 18 Screenshot of add Feedback Form

The Feed Section above for users to add their feedback regarding the workflow. Once done filling feedback users have to select the add button where their feedback will be added to the data. Back button is to head back to the login form.

5.3 Managing Menu

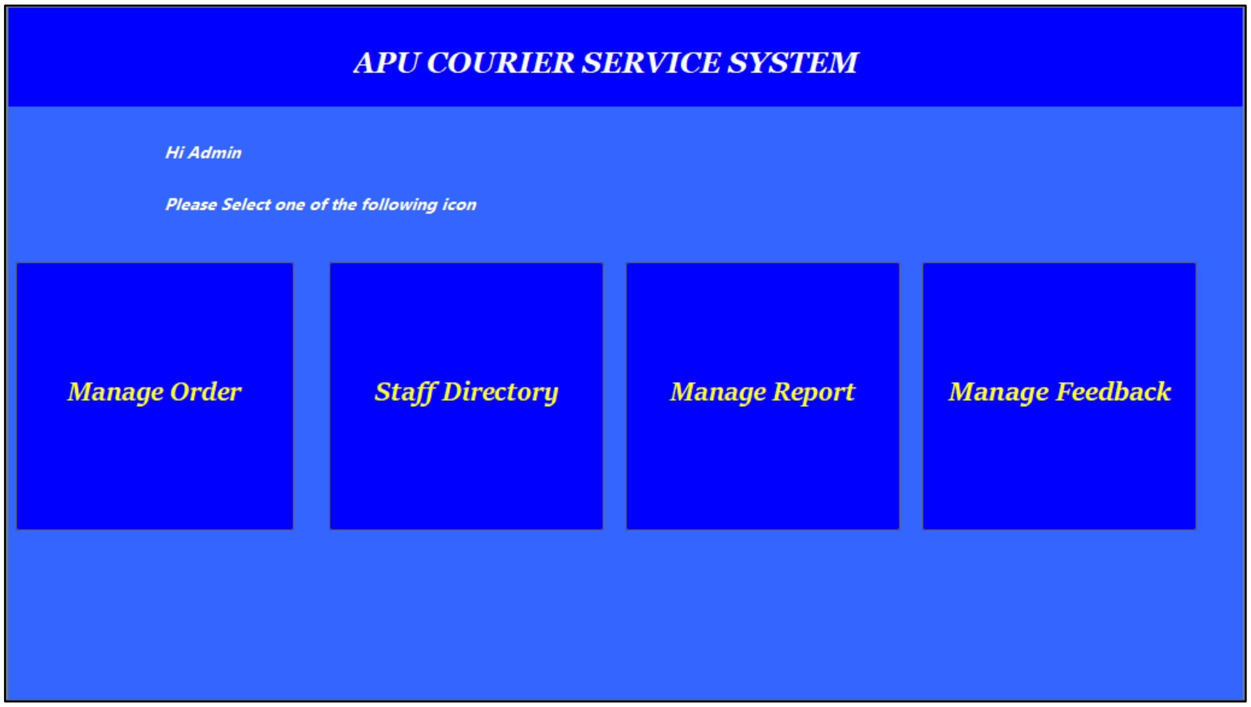
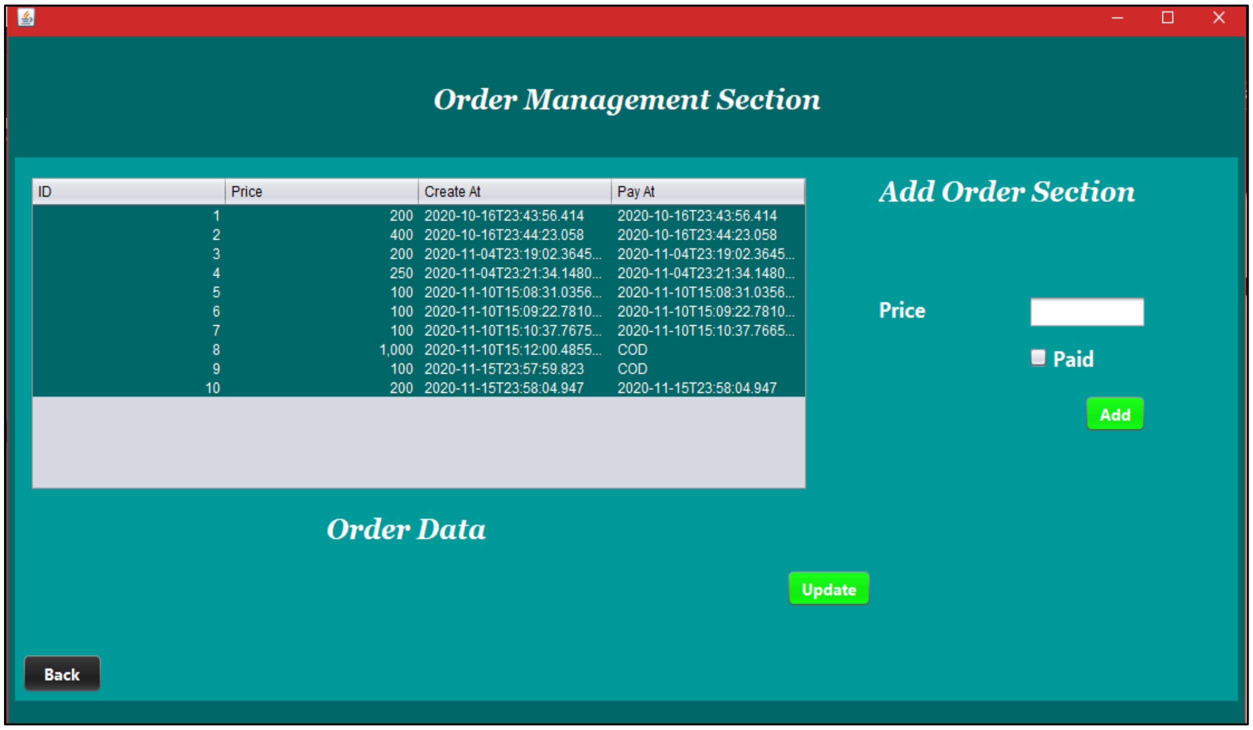

APU COURIER SERVICE SYSTEM
Hi Admin
Please Select one of the following icon
Manage Order
Staff Directory
Manage Report
Manage Feedback

Figure 19 Screenshot of Admin Navigation Page

When the managing staff successfully login to the system, they will face this section of the system. This section has 4 different type of buttons like manage order button, staff directory button, manage report button and manage feedback button. The managing staff will be directed to respective form when any button is selected.

5.4 Order Management Section

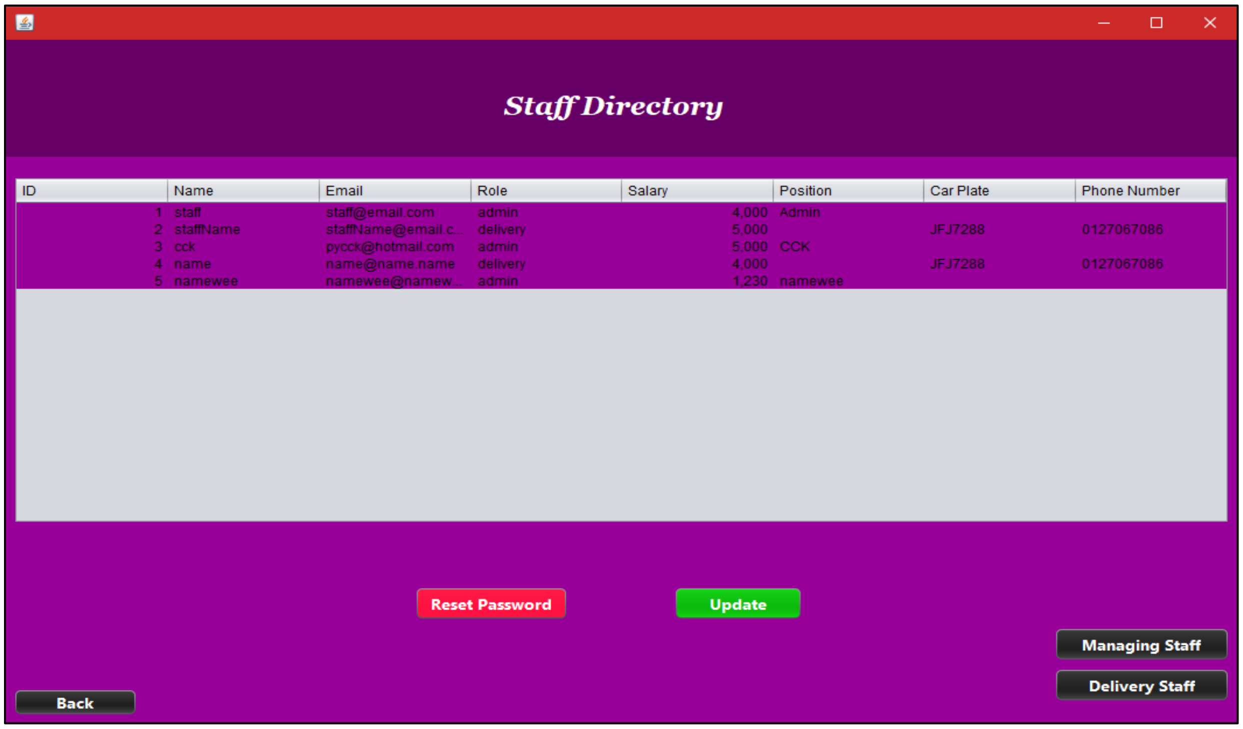


ID	Price	Create At	Pay At
1	200	2020-10-16T23:43:56.414	2020-10-16T23:43:56.414
2	400	2020-10-16T23:44:23.058	2020-10-16T23:44:23.058
3	200	2020-11-04T23:19:02.3645...	2020-11-04T23:19:02.3645...
4	250	2020-11-04T23:21:34.1480...	2020-11-04T23:21:34.1480...
5	100	2020-11-10T15:08:31.0356...	2020-11-10T15:08:31.0356...
6	100	2020-11-10T15:09:22.7810...	2020-11-10T15:09:22.7810...
7	100	2020-11-10T15:10:37.7675...	2020-11-10T15:10:37.7665...
8	1,000	2020-11-10T15:12:00.4855...	COD
9	100	2020-11-15T23:57:59.823	COD
10	200	2020-11-15T23:58:04.947	2020-11-15T23:58:04.947

Figure 20 Screenshot of Order Management Page

Therefore, when manage order button is selected, the user will be directed to this form. This form is the order management section where the order data will be displayed on table form. The managing staff will be able to add new order at this form. The add order section is provided with a price column and paid radio button so the staff must fill in the price and select whether the payment had been completed or still pending. Once done with that process, just select the add button new order will be successfully added.

5.5 Staff Directory



ID	Name	Email	Role	Salary	Position	Car Plate	Phone Number
1	staff	staff@email.com	admin	4,000	Admin		
2	staffName	staffName@email.c...	delivery	5,000		JFJ7288	0127067086
3	cck	pycck@hotmail.com	admin	5,000	CCK		
4	name	name@name.name	delivery	4,000		JFJ7288	0127067086
5	namewee	namewee@namew...	admin	1,230	namewee		

Figure 21 Screenshot of Staff Directory

This staff directory section is to view staff personal details besides this section it also to update any detail and reset password as well. The bottom right of this section has 2 different buttons where the managing staff and delivery, when users select the button, they will be directed to respective form to add staff.

5.6 Create Managing Staff

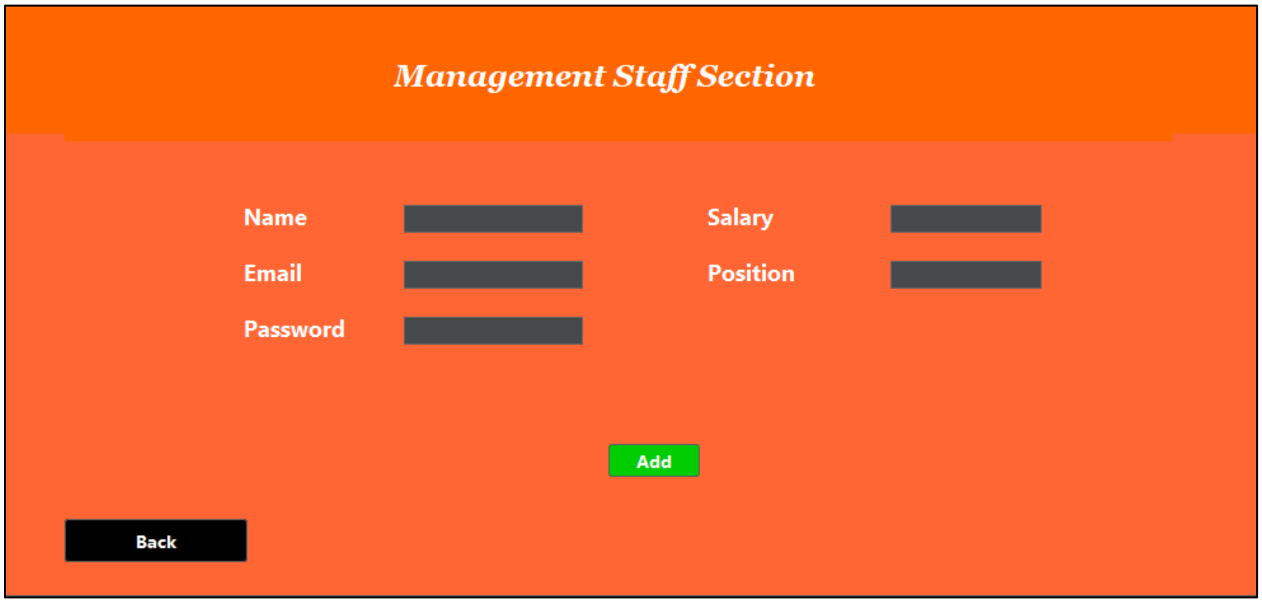


Figure 22 Screenshot of Add Managing Staff

This form will be displayed when user select the managing staff button at the staff directory form. This section is to add managing staff into the data by filling the details needed. There is a back button at the bottom of the form when user select that, they will be directed to the staff profile section.

5.7 Create Delivery Staff

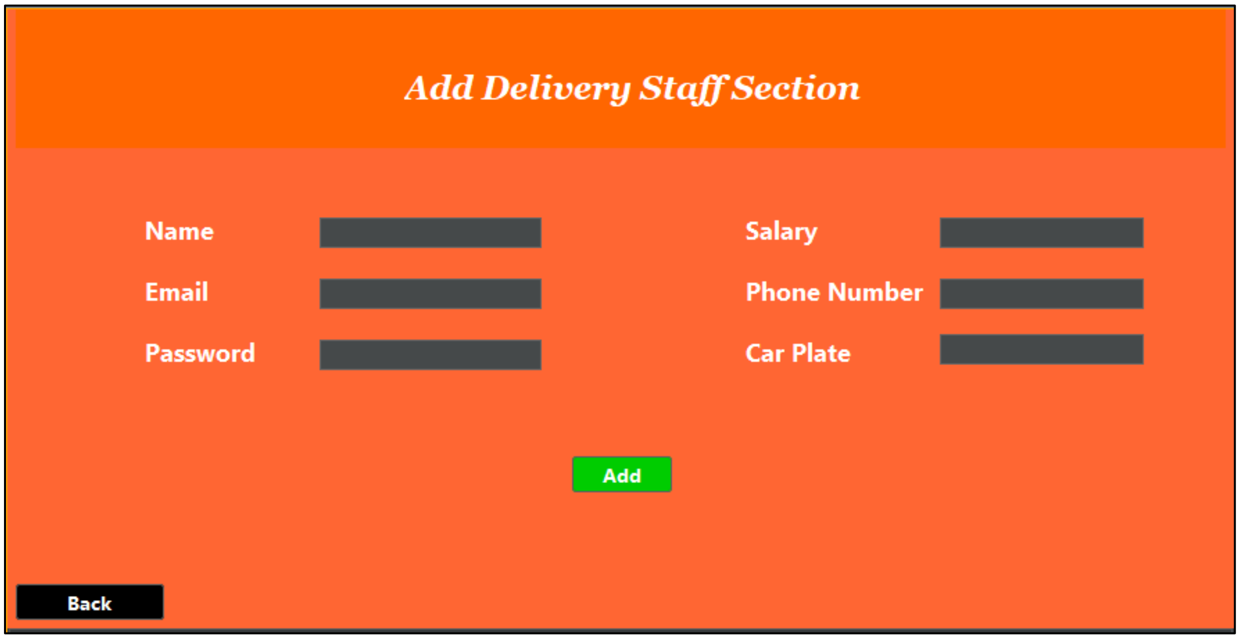
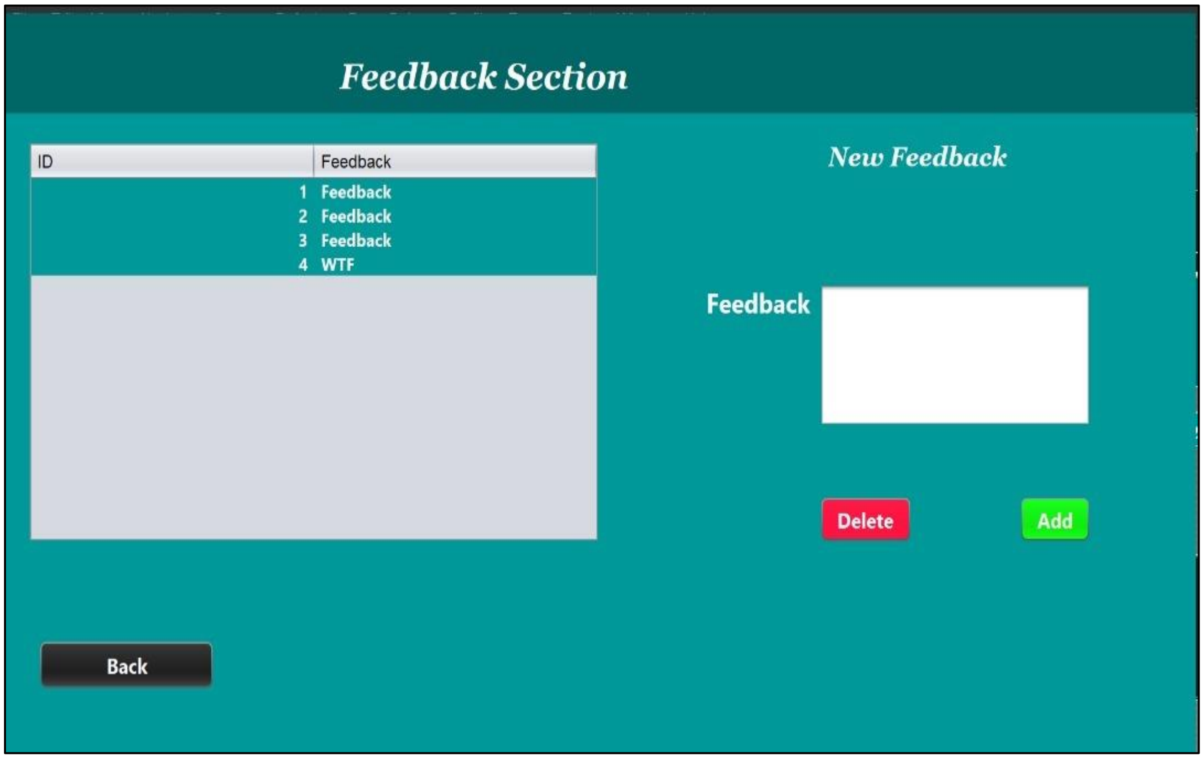


Figure 23 Screenshot of Add Delivery Staff Form

Meanwhile, this form will be displayed when user select the delivery staff button at the staff directory form. This section is to add delivery staff into the data by stating all the required details as well. The back button is to head back to the staff profile section.

5.8 Manage Feedback Section

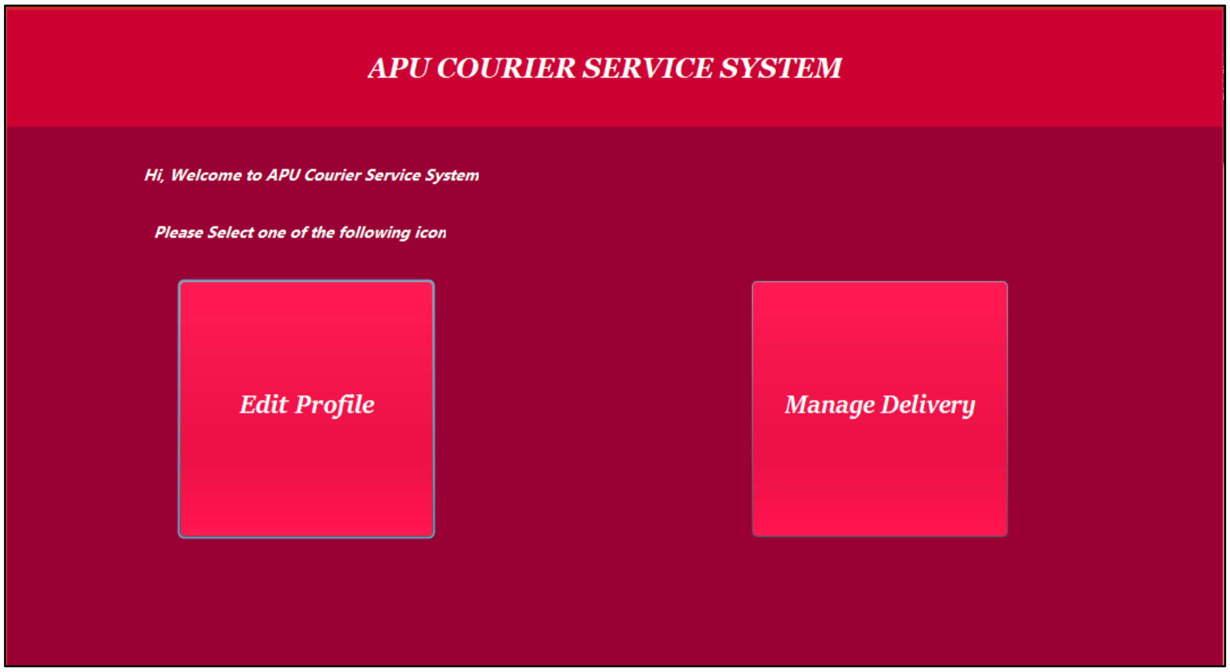


ID	Feedback
1	Feedback
2	Feedback
3	Feedback
4	WTF

Figure 24 Screenshot of Managing Feedback

The feedback section data will be displayed as table form and the right-side user will be able to add feedback and delete feedback as well.

5.9 Staff Login


APU COURIER SERVICE SYSTEM

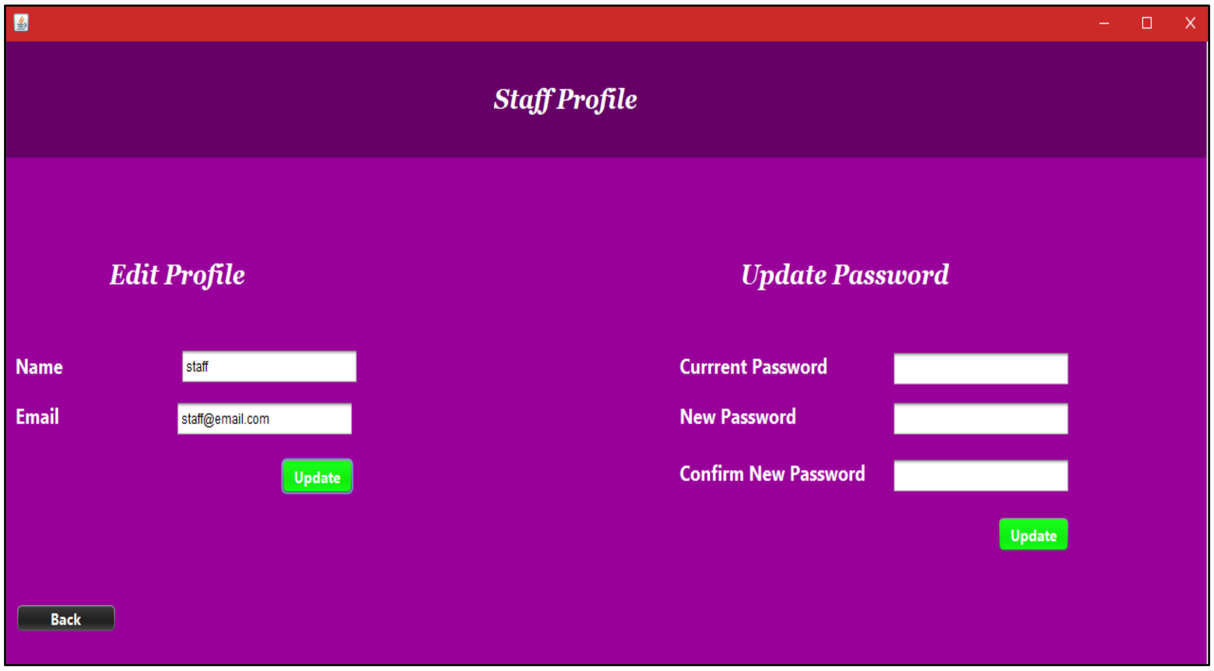
Hi, Welcome to APU Courier Service System

Please Select one of the following icon

Figure 25 Screenshot of Delivery Staff Navigation Page

This section will be directed from the login section when delivery staff successfully able to login to the system. There are 2 buttons on this section for the delivery staff to select. One of the button is the edit profile button and another one is the manage delivery button.

5.10 Staff Profile



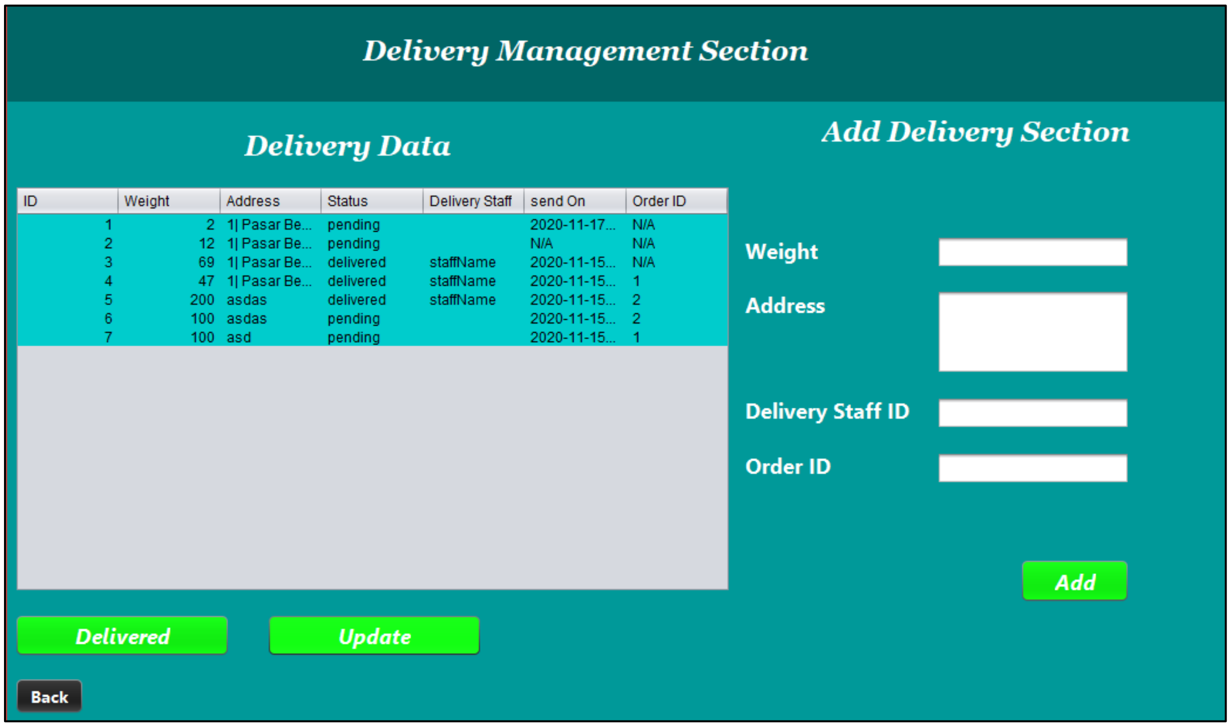
Name: staff
Email: staff@email.com

Currrent Password
New Password
Confirm New Password

Figure 26 Screenshot of Personal Profile change page

The staff profile form will be displayed when the delivery staff select the edit profile button. This section the delivery will be able to complete two tasks on a single form like edit profile and update password. Once done filling the just select the update button it will be updated to the data.

5.11 Delivery Management Section



ID	Weight	Address	Status	Delivery Staff	send On	Order ID
1	2	1| Pasar Be...	pending		2020-11-17...	N/A
2	12	1| Pasar Be...	pending		N/A	N/A
3	69	1| Pasar Be...	delivered	staffName	2020-11-15...	N/A
4	47	1| Pasar Be...	delivered	staffName	2020-11-15...	1
5	200	asdas	delivered	staffName	2020-11-15...	2
6	100	asdas	pending		2020-11-15...	2
7	100	asd	pending		2020-11-15...	1

Figure 27 Screenshot of Delivery Management Page

The delivery management section will be directed when delivery staff select the manage delivery button. This section will display delivery data in table form where the table can be updated, and the delivery can be marked as delivered once it is completed.

References

baeldung. (29 September, 2020). *Creating PDF Files in Java*. Retrieved from Baeldung: https://www.baeldung.com/java-pdf-creation

Gupta, L. (20 September, 2020). *Compare LocalDateTime instances*. Retrieved from HowToDoInJava: https://howtodoinjava.com/java/date-time/compare-localdatetime/

Gupta, L. (25 September, 2020). *Java Secure Hashing – MD5, SHA256, SHA512, PBKDF2, BCrypt, SCrypt*. Retrieved from HowToDoInJava: https://howtodoinjava.com/java/java-security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/

Rollbar. (5 November, 2020). *Throwing exceptions in Java*. Retrieved from Rollbar : https://rollbar.com/guides/java-throwing-exceptions/

SOFTWARETESTINGHELP. (1 November, 2020). *Types Of Inheritance In Java – Single Vs Multiple Inheritance*. Retrieved from SoftwareTestingHelp: https://www.softwaretestinghelp.com/types-of-inheritance-in-java/