RV COLLEGE OF ENGINEERING®
BENGALURU – 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# "Farm Equipment Retail System using JavaFx"

**MINI-PROJECT REPORT**
**OBJECT ORIENTED PROGRAMMING USING JAVA (18CS45)**
**IV SEMESTER**

**2020-21**

**Submitted by**

| | |
|---|---|
| ROHITH NAIR | 1RV19CS131 |
| SATVIK PATIL | 1RV19CS141 |

**Under the Guidance of**

**Soumya A**
**Department of CSE, RVCE,**
**Bengaluru - 560059**

RV COLLEGE OF ENGINEERING®, BENGALURU - 560059

*(Autonomous Institution Affiliated to VTU, Belagavi)*

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# CERTIFICATE

Certified that the **Mini-Project** work titled "Farm Equipment Retail System using JavaFx" has been carried out by **Rohith Nair(1RV19CS131) and Satvik Patil(1RV19CS141),** bonafide students of RV College of Engineering, Bengaluru, have submitted in partial fulfillment for the **Assessment of Course: OBJECT ORIENTED PROGRAMMING USING JAVA (18SC45) – Open-Ended Experiments** during the year 2020-2021. It is certified that all corrections/suggestions indicated for the internal assessment have been incorporated in the report.

**Faculty Incharge**
Department of CSE,
RVCE., Bengaluru –59

**Head of Department**
Department of CSE,
RVCE, Bengaluru–59

RV COLLEGE OF ENGINEERING®, BENGALURU - 560059

*(Autonomous Institution Affiliated to VTU)*

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# DECLARATION

We, **Rohith Nair (1RV19CS131) and Satvik Patil (1RV19CS141),** the students of 4th Semester B.E., Department of Computer Science and Engineering, RV College of Engineering, Bengaluru hereby declare that the Mini-Project titled **"Farm Equipment Retail System using JavaFx"** has been carried out by us and submitted in partial fulfillment for the **Assessment of Course: OBJECT ORIENTED PROGRAMMING USING JAVA (18CS44) - Open-Ended Experiment** during the year 2020-2021.

| | |
|---|---|
| **Place: Bengaluru** | **Rohith Nair** |
| **Date:** | **Satvik Patil** |

## ABSTRACT

The main objective of this project is to help the farmers in increasing their productivity and efficiency in farming through assisting them with the issue of expensive farm machinery. The issue raised here is inadequate and minimal usage of latest technology by Indian farmers due to the high cost and unavailability of latest farm machinery.

Our team’s approach towards solving this problem is through connectivity and collaboration between farmers and farm equipment merchants. By that we mean a countrywide network that will help the farmers to buy the advanced farm machinery, thus helping the merchants too who are selling them. This network helps increase productivity and employability in the field of agriculture.

In conclusion, usage of adequate machinery and availability of sufficient resources plays a pivotal role in the productivity and income a farmer can generate. And this platform that interlinks the required individuals will help us take a step further in eradicating the issue.

# 1.Introduction

## 1.1Object Oriented Concepts

### 1.1.1 Traditional versus Object Oriented Approach

Traditional Programming can be defined as a programming model which is derived from structured programming, which is based upon the concept of calling procedure. Procedures, which are  also known as routines, subroutines or functions, consist of a series of computational steps that need to be carried out. During a program’s execution, any given procedure might be called at any point, including by other procedures or itself.
FORTRAN, ALGOL, COBOL, BASIC, Pascal and C are some of the languages which use Procedural/Traditional Programming.

Object Oriented Programming can be defined as a programming model which is based upon the concept of objects. Objects contain data in the form of attributes and code in the form of methods. In this, computer programs are designed using the concept of objects that can interact with the real world. Object oriented programming languages are various but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types. Java, C++, C#, Python, PHP, JavaScript, Ruby, Perl, Objective-C, Dart, Swift, Scala are some of the languages which use Object Oriented Programming.

### 1.1.2 OOA, OOD & OOP and their Relationship

i. Object-Oriented Analysis(OOA)

It emphasizes the building of real-world models, using
an object-oriented view of the world.
It is a method of analysis that examines requirements from the perspective of the classes and objects of the problem domain.In the analysis of the task, from the objective existence of things and things The relationship between the related objects (including the attributes and behaviors of the objects) and the relationship between the objects are summarized, and the Objects with the same attributes and behaviors are represented by a class.

ii. Object-Oriented Design(OOD)

It is a method of design encompassing the process of object oriented decomposition
and a notation for depicting both logical and physical as well as static and dynamic models of the system under design. Object-oriented design:
(1) leads to an object-oriented decomposition and
(2) uses different notations to express different models of the logical (class and object structure) and physical (module and process architecture) design of a system, in addition to the static and dynamic aspects of the system.

iii. Object-Oriented Programming(OOP)

It is a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships.

How are OOP, OOD and OOA related?
The products of object-oriented analysis serve as the models from which we may start an object-oriented design. The products of object-oriented design can then be used as blueprints for completely implementing a system using object-oriented programming methods.

### 1.1.3 Features of OOP approach

OOP has the following features:

- Encapsulation

Encapsulation is a process of information hiding. It is simply the combination of process and data into a single entity. Data of an object is hidden from the rest of the system and available only through the services of the class. It allows improvement or modification of methods used by objects without affecting other parts of a system.

- Abstraction

It is a process of taking or selecting necessary methods and attributes to specify the object. It focuses on essential characteristics of an object relative to the perspective of the user.

- Inheritance

Inheritance is a great feature that allows to create sub-classes from an existing class by inheriting the attributes and/or operations of existing classes.

- Polymorphism and Dynamic Binding

Polymorphism is the ability to take on many different forms. It applies to both objects and operations. A polymorphic object is one whose true type hides within a super or parent class.

In polymorphic operation, the operation may be carried out differently by different classes of objects. It allows us to manipulate objects of different classes by knowing only their common properties.

## 1.2 Overview of Java Programming Language

### 1.2.1 Features of Java

Java has the following features:

(a) Simple - Java is very easy to learn, and its syntax is simple, clean and easy to understand. Java language is a simple programming language because Java syntax is based on C++. Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc. There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

(b) Object-Oriented - Java is an OO programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporate both data and behavior.Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.

(c) Portable - Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

(d) Platform-Independent - Java code can be executed on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms, i.e., Write Once and Run Anywhere (WORA).

(e) Multi-Threaded - A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multimedia, Web applications, etc.

(f) Dynamic - Java is a dynamic language. It supports the dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++. Java supports dynamic compilation and automatic memory management (garbage collection).

### 1.2.2 Inheritance

It is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOP. The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also. Inheritance represents the IS-A relationship which is also known as a *parent-child* relationship.

Example:

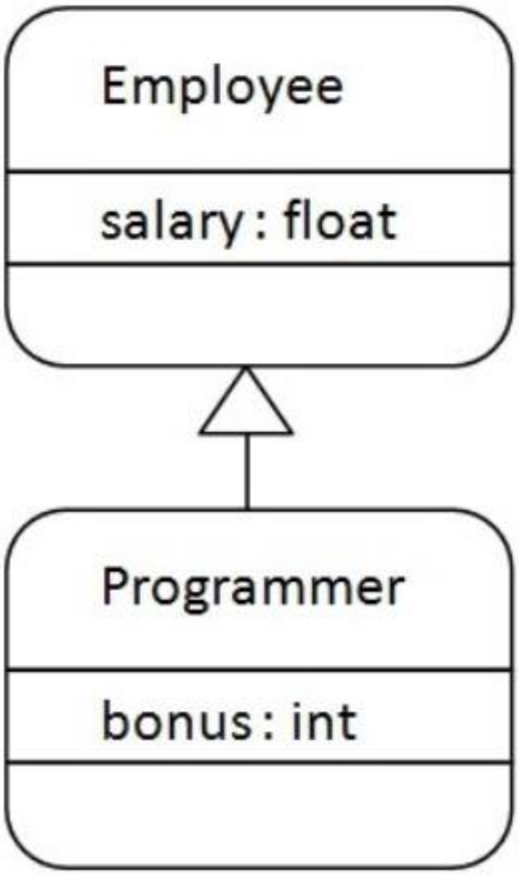
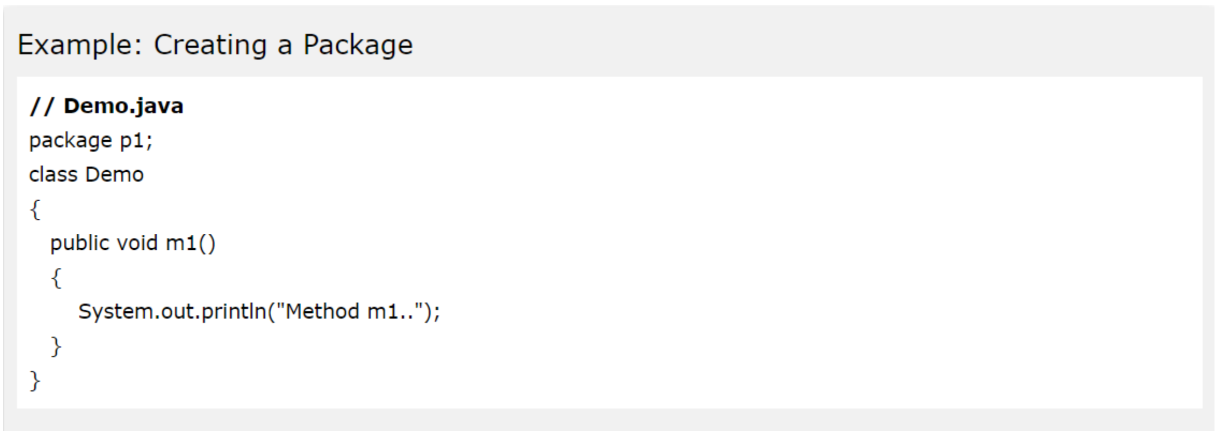


fig1: Showing an example of inheritance using class diagram

## 1.2.2 Interfaces & Packages

A package is a mechanism to group the similar type of classes, interfaces and sub-packages and provide access control. It organizes classes into a single unit.
In Java already many predefined packages are available, used while programming.
For example**:** java.lang, java.io, java.util etc.

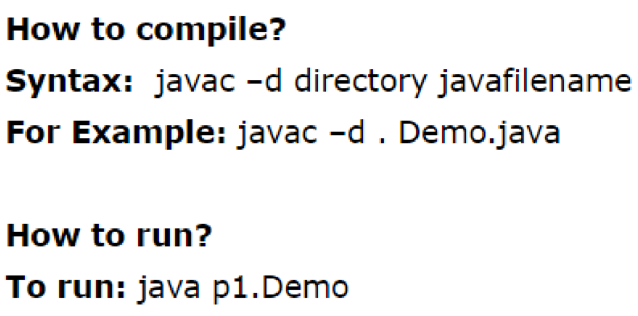
i. Packages provide code reusability, because a package has a group of classes.
ii. It helps in resolving naming collisions when multiple packages have classes with the same name.
iii. Package also provides the hiding of class facilities. Thus other programs cannot use the classes from hidden packages.
iv. Access limitation can be applied with the help of packages.
v. One package can be defined in another package.



Example: Creating a Package

```
// Demo.java
package p1;
class Demo
{
  public void m1()
  {
      System.out.println("Method m1..");
  }
}
```

fig2: We are here creating a package

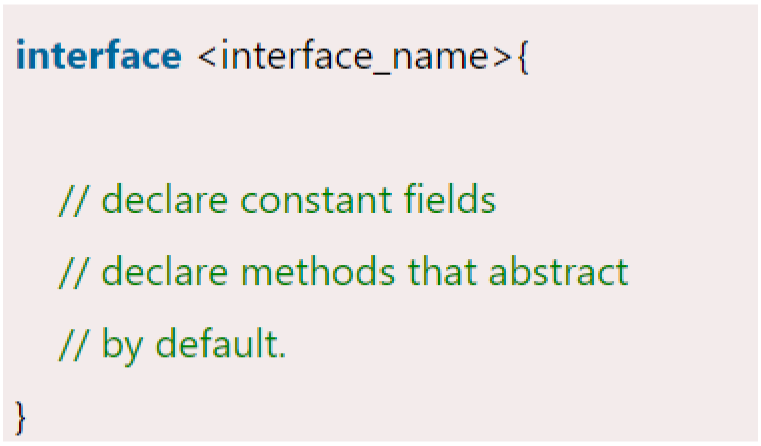


**How to compile?**
**Syntax:** javac -d directory javafilename
**For Example:** javac -d . Demo.java

**How to run?**
**To run:** java p1.Demo

fig3: how to compile and run a java program

An **interface** is a blueprint of a class. It has static constants and abstract methods. The interface in Java is *a mechanism to achieve abstraction*. There can be only abstract methods in the Java interface, not method bodies. It is used to achieve abstraction and multiple inheritance in Java. In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body. Java Interface also represents the IS-A relationship. It cannot be instantiated just like the abstract class.



```
interface <interface_name>{

    // declare constant fields
    // declare methods that abstract
    // by default.
}
```

`

fig4: Syntax for interface

### 1.2.4 Exception Handling

Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.
The main advantage of exception handling is to maintain the normal flow of the application or code. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions.

<u>Hierarchy of Java Exception Classes</u>

The java.lang.Throwable class is the root class of Java Exception hierarchy inherited by two subclasses: Exception and Error. The hierarchy of Java Exception classes is given below:

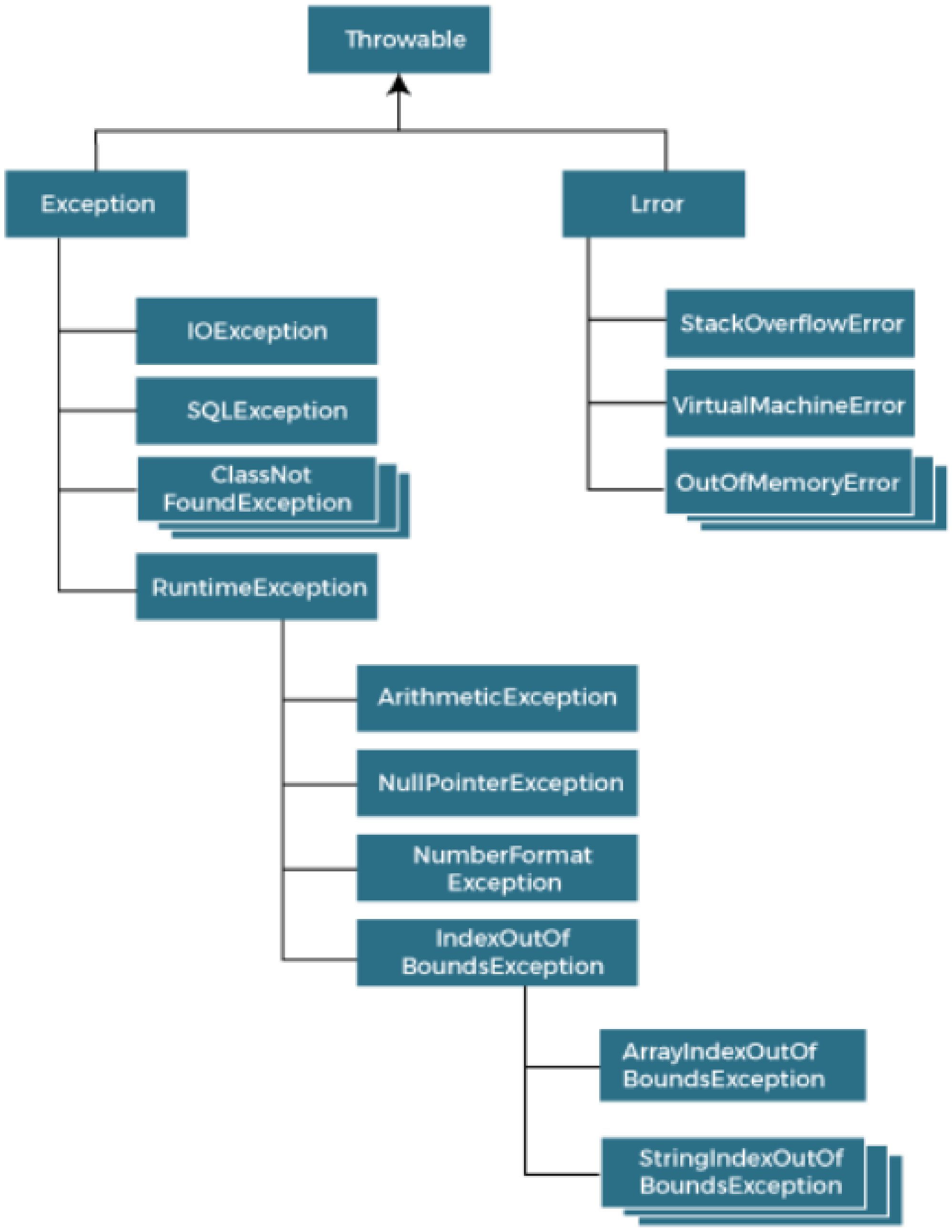


fig5: Hierarchy of Java Exception Class

### 1.2.5 Multithreaded Programming

Multithreading is a concept wherein a process is divided into two or more subprocesses, to parallelize the processing by running concurrently. Two or more such parts of the process are called threads, with each thread defining a separate path of execution.
A thread cannot exist independently and must be attached to a process. Multithreading programming is used in order to make the process faster by achieving concurrent execution. This has a benefit over creating multiple processes since threads are relatively lightweight and do not require a separate address space unlike processes.

The life-cycle of a thread mainly comprises five stages:

a) Newborn state-when the thread is created
b) Runnable state- when the thread is ready to execute
c) Running state- when the thread occupies the CPU

d) Blocked state- when the thread is blocked from entering the runnable state
e) Dead state- the thread has either been killed or has finished executing

Threads can go between these states using certain functions such as suspend, sleep, wait (running/runnable to blocked); resume, notify (blocked to running/runnable); yield (running to runnable) and stop (to dead state).

To illustrate threads and their lifecycle, we employ the below diagram:

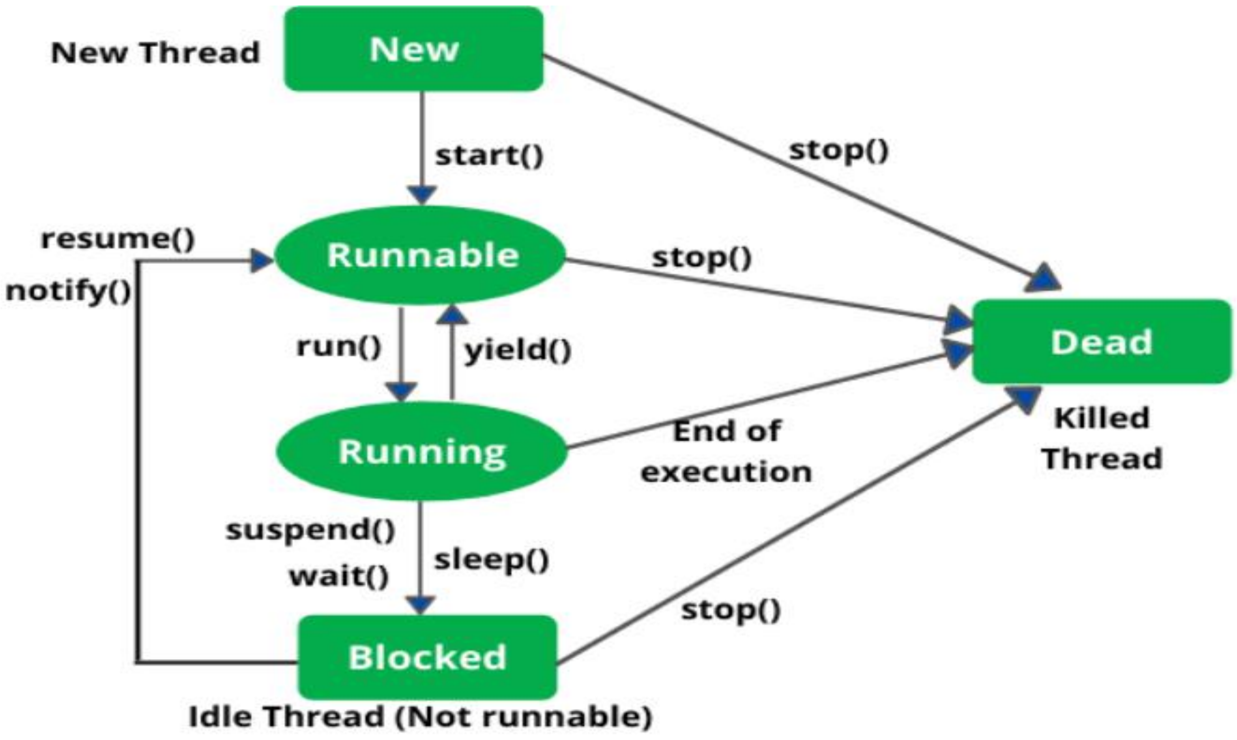


fig6: Lifecycle of a thread

In Java, all threads branch from the Main thread and can be named, given a particular priority or even put into groups using constructors. To create a thread, we can either extend the Thread class or implement the Runnable interface. Further, threads require the run method for execution and need to be set into motion using the start function.

To obtain the complete use of threads, we need to synchronize them. This enables inter-thread communication and allows us to let multiple threads access common parameters without the risk of a deadlock.

### 1.2.6 Lambda Expressions

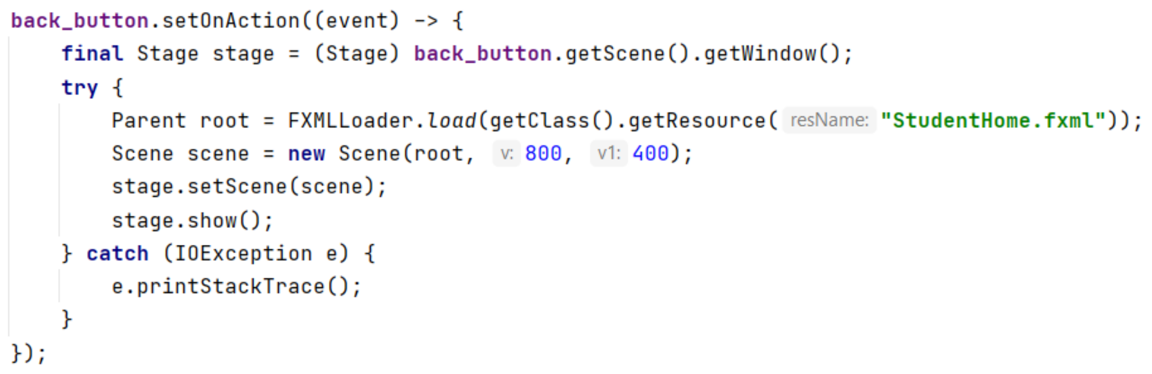
Lambda expressions are a concise way to represent one method interface with the help of an expression. Since it can help in iterating, filtering and extracting data from collections, it is useful in this library.

In a lambda expression, we provide the implementation of a functional interface. An important term to define here is functional interface. This is an interface that consists of only one abstract method. This abstract method is later elucidated in the lambda expression which is treated as a function, thereby allowing us to forgo method definition. Thus, it reduces the amount of code required.

The syntax of a lambda expression has three components including the argument list, followed by an arrow (->) and finally succeeded by the body. We can define this in multiple ways, by varying the number of parameters in the argument list or the number of lines in the body of the function.

A concept of interest is the generic functional interfaces, which accept any data type and this further reduces our code while giving us a skeleton to define our lambda expression, which is now not limited by any data type.

In this project, we have used lambda expressions to handle various events, for example the press of a button. In such cases, the EventHandler is taken and further action upon clicking the button is described by the lambda function. An excerpt of our code illustrating the same is given below:



```
back_button.setOnAction((event) -> {
    final Stage stage = (Stage) back_button.getScene().getWindow();
    try {
        Parent root = FXMLLoader.load(getClass().getResource( resName: "StudentHome.fxml"));
        Scene scene = new Scene(root,  v: 800,  v1: 400);
        stage.setScene(scene);
        stage.show();
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

fig7: Example of lambda expression

### 1.2.7Regular Expressions

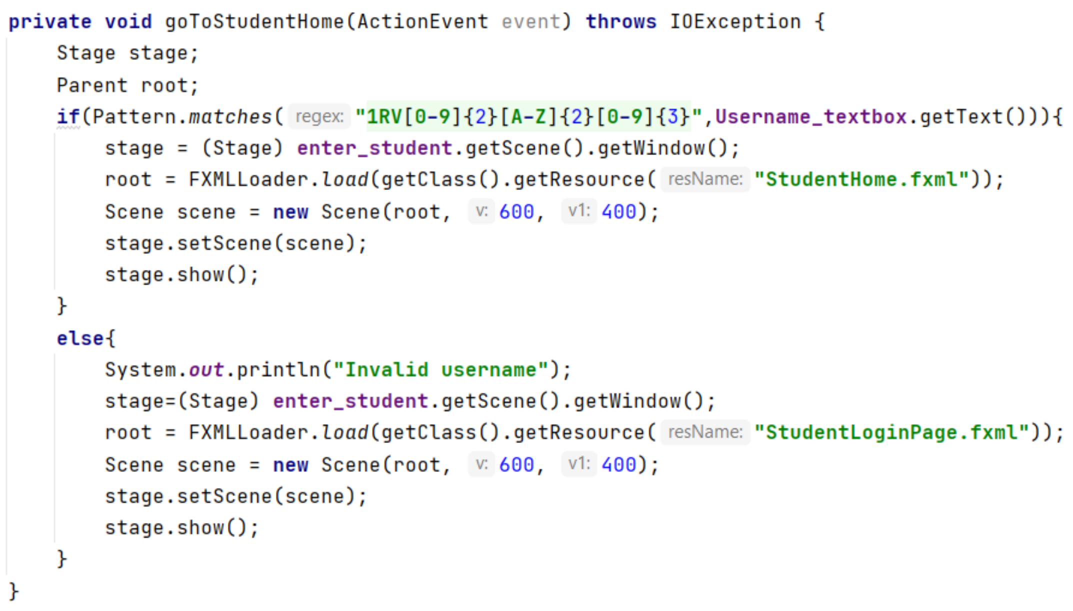
A regular expression is a character string describing a sequence, called a pattern. This pattern in turn, can be used in finding matches in a given input of character sequences. Regular expressions can be used to match a given string exactly, or identify a string fulfilling a set of conditions defined using wildcard characters, character sets and several quantifiers.

Two classes come into play while using regular expression processing, i.e., Pattern and Matcher. These classes work in tandem for sequence matching. While the Pattern class defines a regular expression, the Matcher class matches such a pattern with another sequence. The Pattern class defines no objects, and uses the compile method, which can be used in combination with the methods from Matcher class such as matches and find.

Apart from the classes, three quantifiers are pivotal to regular expression processing. These are:

| | |
|---|---|
| + | : Matches if one or more occurrences |
| * | : Matches if zero or more occurrences |
| ? | : Matches if zero or one occurrence |

Additionally, wildcards are used to denote any symbol, which when combined with quantifiers make a powerful tool that aids in data pre-processing, natural language processing, web scraping and data validation. The latter has been used in our project, wherein the credentials of the users are checked when the login using regular expressions processing. The illustration is given below:



```java
private void goToStudentHome(ActionEvent event) throws IOException {
    Stage stage;
    Parent root;
    if(Pattern.matches( regex: "1RV[0-9]{2}[A-Z]{2}[0-9]{3}",Username_textbox.getText())){
        stage = (Stage) enter_student.getScene().getWindow();
        root = FXMLLoader.load(getClass().getResource( resName: "StudentHome.fxml"));
        Scene scene = new Scene(root,  v: 600,  v1: 400);
        stage.setScene(scene);
        stage.show();
    }
    else{
        System.out.println("Invalid username");
        stage=(Stage) enter_student.getScene().getWindow();
        root = FXMLLoader.load(getClass().getResource( resName: "StudentLoginPage.fxml"));
        Scene scene = new Scene(root,  v: 600,  v1: 400);
        stage.setScene(scene);
        stage.show();
    }
}
```

fig8: Example of Regular expression for finding if usn is valid or not

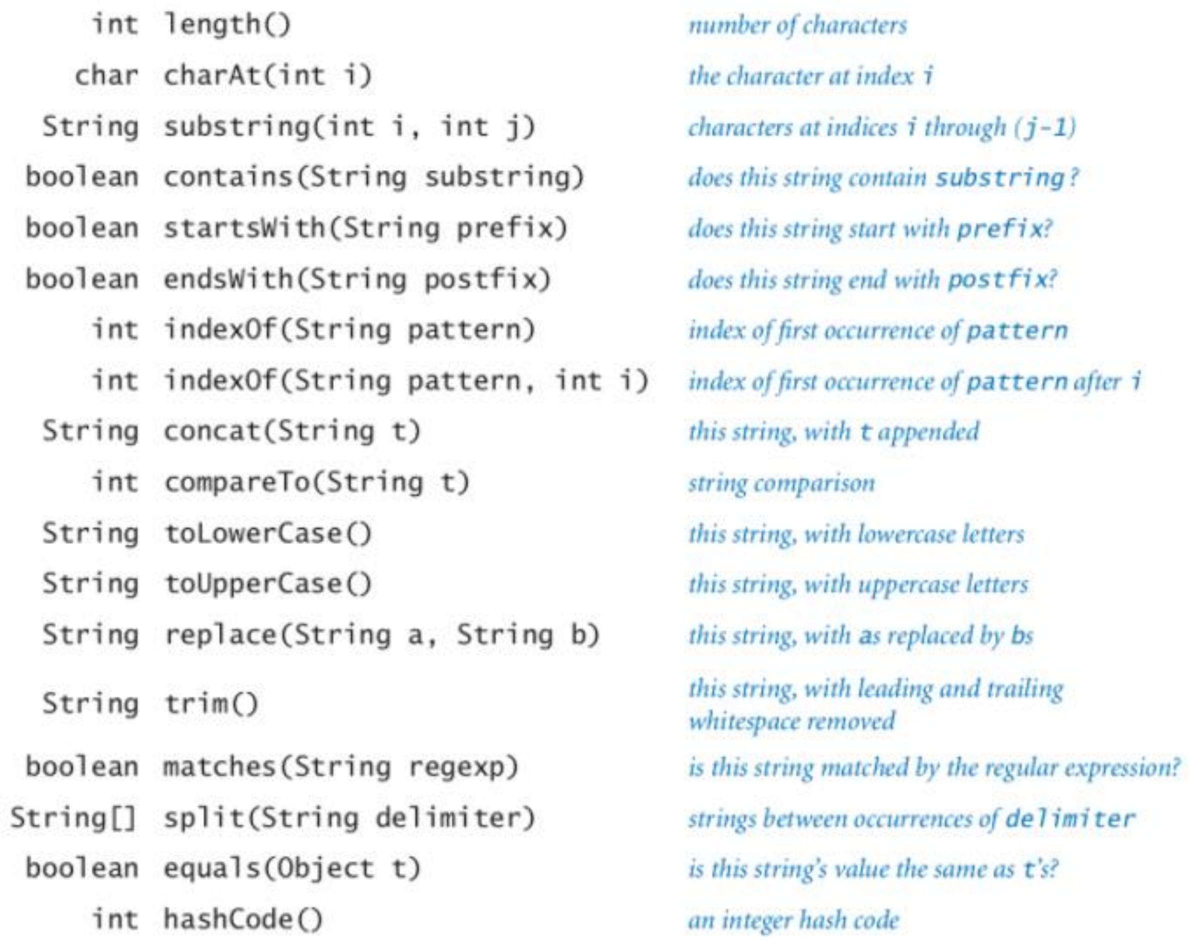
### 1.2.8 Strings

Strings are inherently instrumental to programming, since they are used extensively in a range of applications. From taking inputs from the user to processing of expressions, we require strings and Java provides a range of functions defined on the Strings to aid in programming.

Defined by an array of characters or string literals, Strings can be equated, concatenated, and various substrings or components such as characters and bytes can be extracted from them.

Furthermore, we can also convert various data types to Strings. Additionally, modifications such as trimming whitespace, changing the case of characters in a string can be performed on Strings. Finally, we have searching, which as described above, is similar to regular expression searching and helps finding the exact string or a substring of required characters.

A comprehensive list of methods on Strings, if not exhaustive is described below:



| | | |
|---|---|---|
| int | length() | *number of characters* |
| char | charAt(int i) | *the character at index* i |
| String | substring(int i, int j) | *characters at indices* i *through* (j-1) |
| boolean | contains(String substring) | *does this string contain* substring? |
| boolean | startsWith(String prefix) | *does this string start with* prefix? |
| boolean | endsWith(String postfix) | *does this string end with* postfix? |
| int | indexOf(String pattern) | *index of first occurrence of* pattern |
| int | indexOf(String pattern, int i) | *index of first occurrence of* pattern *after* i |
| String | concat(String t) | *this string, with* t *appended* |
| int | compareTo(String t) | *string comparison* |
| String | toLowerCase() | *this string, with lowercase letters* |
| String | toUpperCase() | *this string, with uppercase letters* |
| String | replace(String a, String b) | *this string, with* as *replaced by* bs |
| String | trim() | *this string, with leading and trailing whitespace removed* |
| boolean | matches(String regexp) | *is this string matched by the regular expression?* |
| String[] | split(String delimiter) | *strings between occurrences of* delimiter |
| boolean | equals(Object t) | *is this string's value the same as* t*'s?* |
| int | hashCode() | *an integer hash code* |

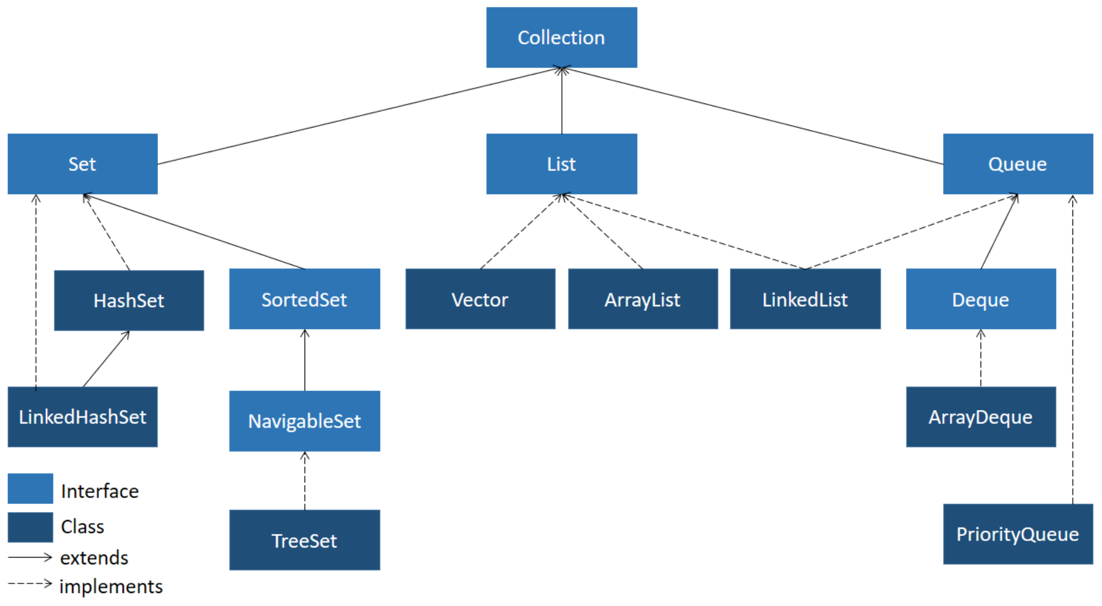
fig9: All the in-built methods of String handling

### 1.2.9 Collections Framework

A collection is an object that groups multiple elements into a single unit. Collections are used to store, retrieve and manipulate data. Collections framework in Java helps us in representing and manipulating collections.

It is a collection of interfaces and classes which helps in storing and processing the data efficiently. Collections Framework standardizes the way we store and access the data from collections and is a part of the java.util package.

The diagram given below represents the hierarchy of the Java Collections framework.



The following list describes the various interfaces of the collection framework:

- **Collection** - the root of the collection hierarchy

A collection represents a group of objects known as its elements.
Some types of collections allow duplicate elements and others do not. Some are ordered and others are unordered.

- **Set** - a collection that cannot contain duplicate elements

- **List** - an ordered collection

Lists can contain duplicate elements.

A programmer can access elements by their index(position) and has control over where each element is inserted in the list.

**Queue and Deque** (pronounced as "Deck") - collections used to hold multiple elements prior to processing. Besides basic collection operations, a Queue and a Deque provides additional insertion, extraction, and inspection operations.

- Queue

Queues typically, but do not necessarily, order elements in a FIFO (first-in-first-out) manner. Among the exceptions are priority queues.
In a FIFO queue, all new elements are inserted at the tail of the queue. Other kinds of queues may use different placement rules.

- Deque

Deques can be used both as FIFO (first-in-first-out) and LIFO (last-in-first-out).
In a Deque, elements can be inserted, retrieved and removed from both ends.

### 1.2.10 JavaFX Framework

JavaFX is an open source Java-based framework for developing rich client applications. It contains several features that make it a preferred choice for developing rich client applications:

- JavaFX is written in Java, which enables you to take advantage of all Java features such as multithreading, generics, and lambda expressions.
- JavaFX supports data binding through its libraries.
- JavaFX code can be written using any Java virtual machine (JVM)-supported scripting languages such as Visage, Groovy, and Scala.
- JavaFX offers two ways to build a user interface (UI): using Java code and using FXML. FXML is an XML-based scriptable markup language to define a UI declaratively. Oracle provides a tool called Scene Builder, which is a visual editor for FXML.
- JavaFX provides a rich set of multimedia support such as playing back audios and vid-eos. It takes advantage of available codecs on the platform.
- JavaFX lets you embed web content in the application.
- JavaFX provides out-of-the-box support for applying effects and animations, which are important for developing gaming applications. You can achieve sophisticated animations by writing a few lines of code.

## 1.3 Proposed System

### 1.3.1 Objectives

We have utilized this project to understand the benefits of JavaFx and the full scope of its applications. Further, implementation of object oriented programming concepts have helped us understand the benefits of using OOP over traditional methods of programming.
Through the Farm Equipment Retail Portal, we aim to achieve the following:

a) Efficient communication between the farmers and farm equipment merchants.
b) Centralized approach towards Farm Equipment Retail.
c) Effective management of information, and availability of products.
d) Faster communication and a more organized approach for purchase.

### 1.3.2 Methodology

To concisely describe the development of our project, we have divided it into three parts, as follows.

- Designing

In order to fully understand the different scenes required and their functionalities, we developed a class diagram with the basic requirements. Furthermore, the user interface was designed and developed using SceneBuilder, which is integrated into Intellij to easily develop a seamless application interface. Thus we created several frames for login, home pages, registration form and query table.

- Integrating JavaFx

Building upon the frames developed in the designing phase, we allocated functionalities to them with reference to the class diagram. JavaFx frameworks, and various nodes such as buttons and editable tables were applied while integrating the scenes and making them interactive. The different scenes were thus layered and provided most of the functionality of the program.

- Integrating the Database

We have used java classes here to represent as database in our project to store values and retrieve data from this and display in the javafx tableview and treeview. It has been used to store details about the equipments, users.

**1.3.3 Scope**

This project can be used in real-life. Many times farmers find it difficult to buy adequate products for efficient farming and they don’t know where to find them, whom to contact and how much it costs.

Thus this project helps overcome that problem and farmers will have a one stop platform for all their farm equipment needs

# 2. Requirement Specifications

## 2.1 Hardware specifications

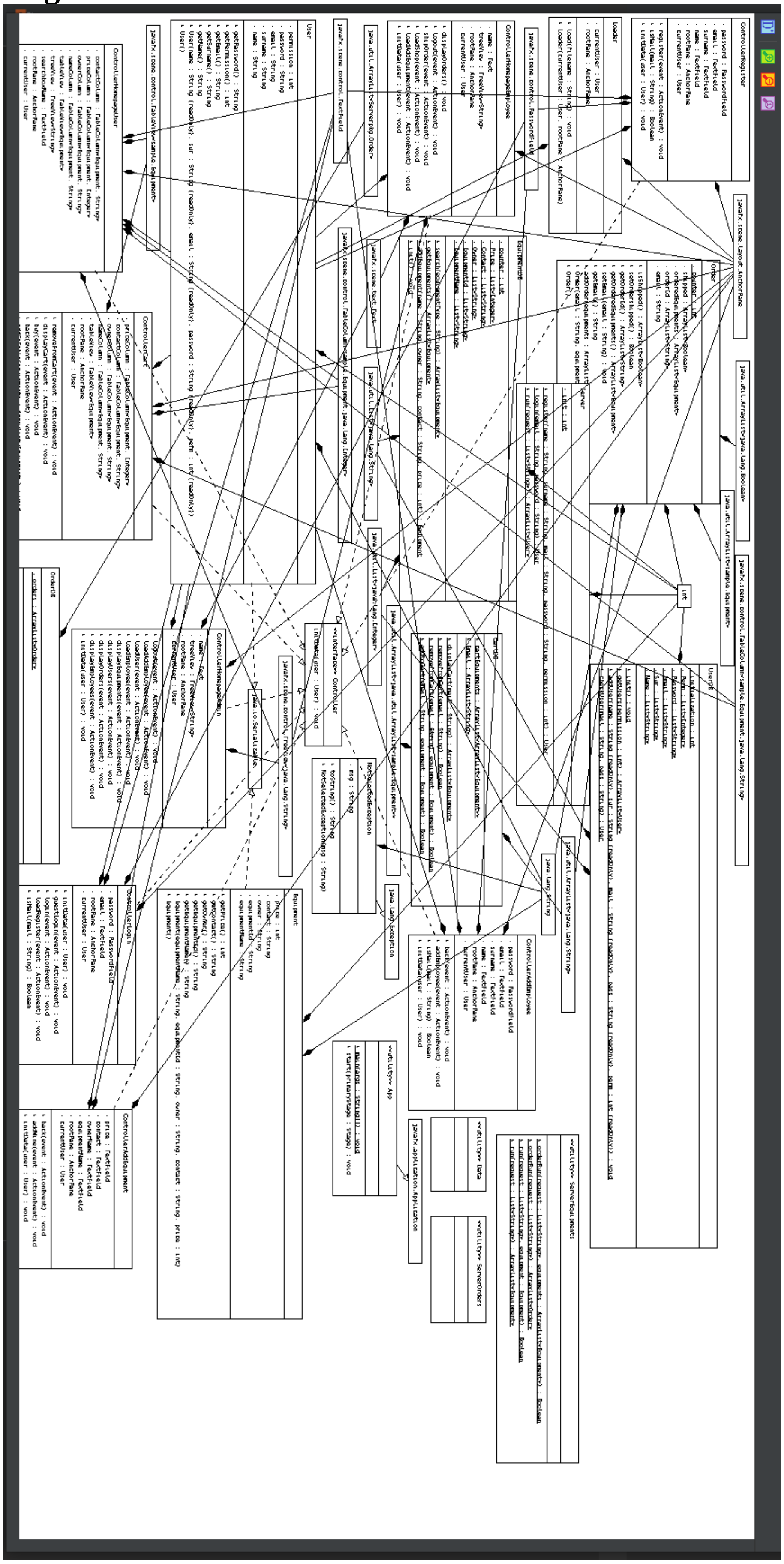
Our program does not have any hardware requirements as such and can be used from any system capable of running Intellij Idea platform, or any similar IDE that allows JavaFx.

## 2.2 Software specifications

Along with the Java platform, we need the JavaFx package to be installed to run the farm equipment rental system. We also used Scenebuilder to create the GUI part of the application.

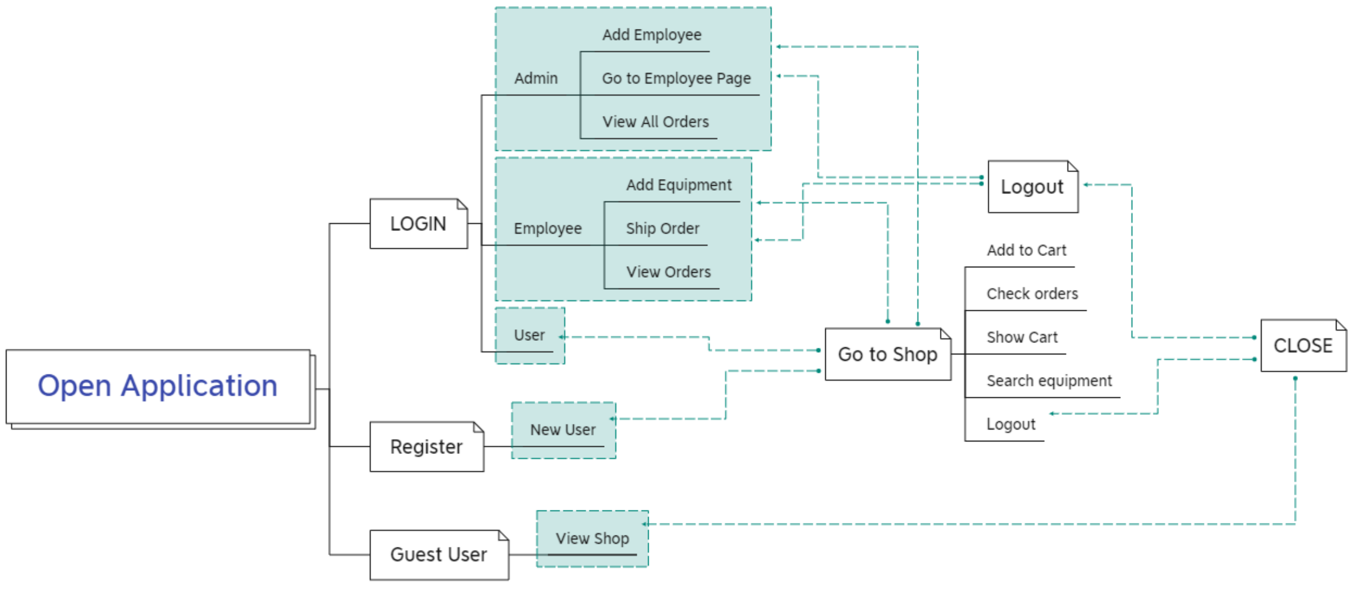
# 3. System Design and Implementation

## 3.1 Class Diagram



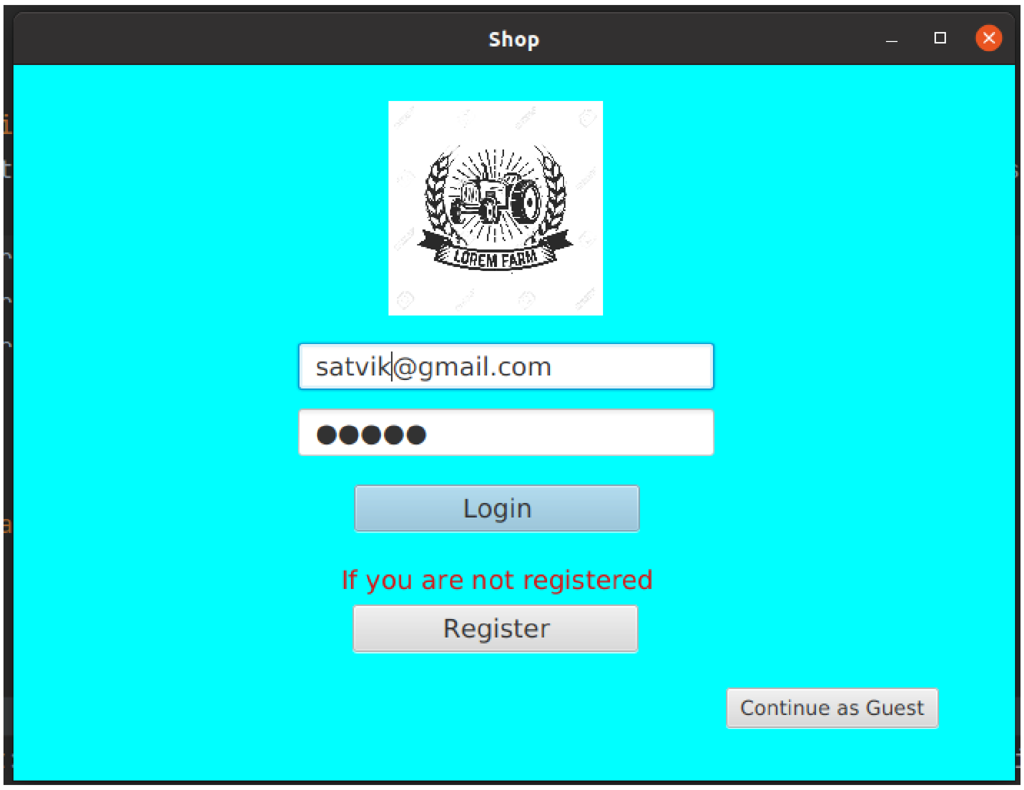
## 3.2 Modular Description

The portal can be divided into 2 parts, the teacher interface and the student interface. Users can select if they want to login as a student or a teacher before they are guided to the login page, which verifies their credentials. Upon authorization, they are redirected to their respective home pages. Students canview their doubts and register new doubts, while teachers can answer doubts according to the subject.

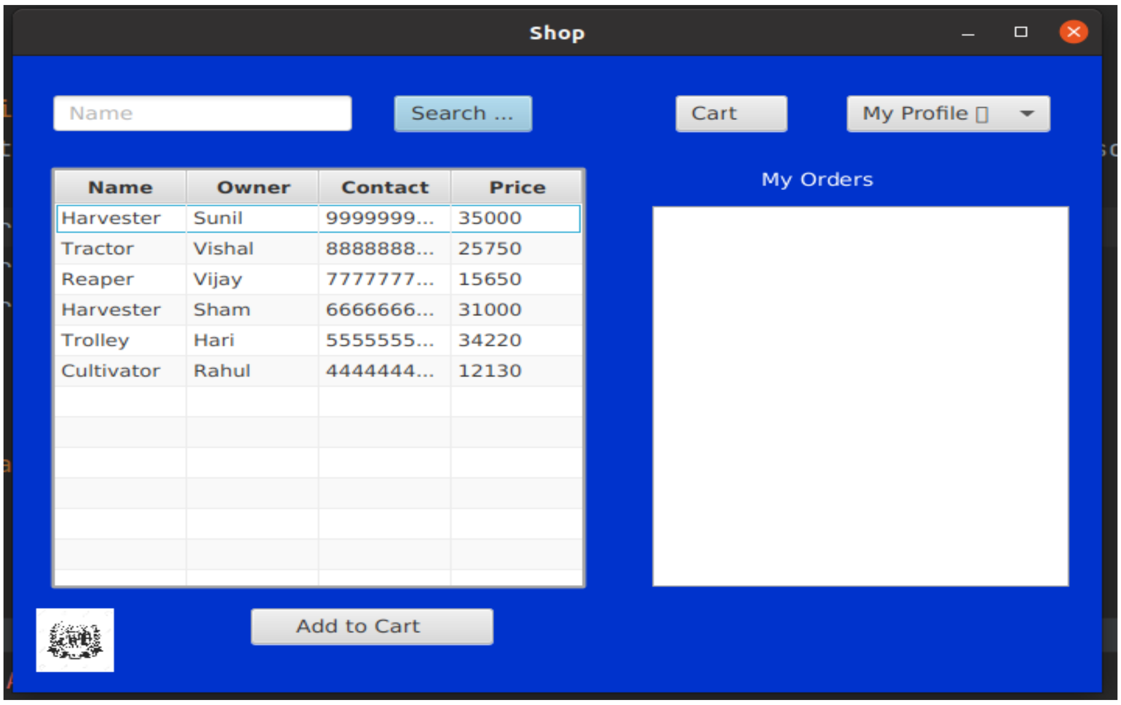


## 4. Results and Snapshots

I. Login Page

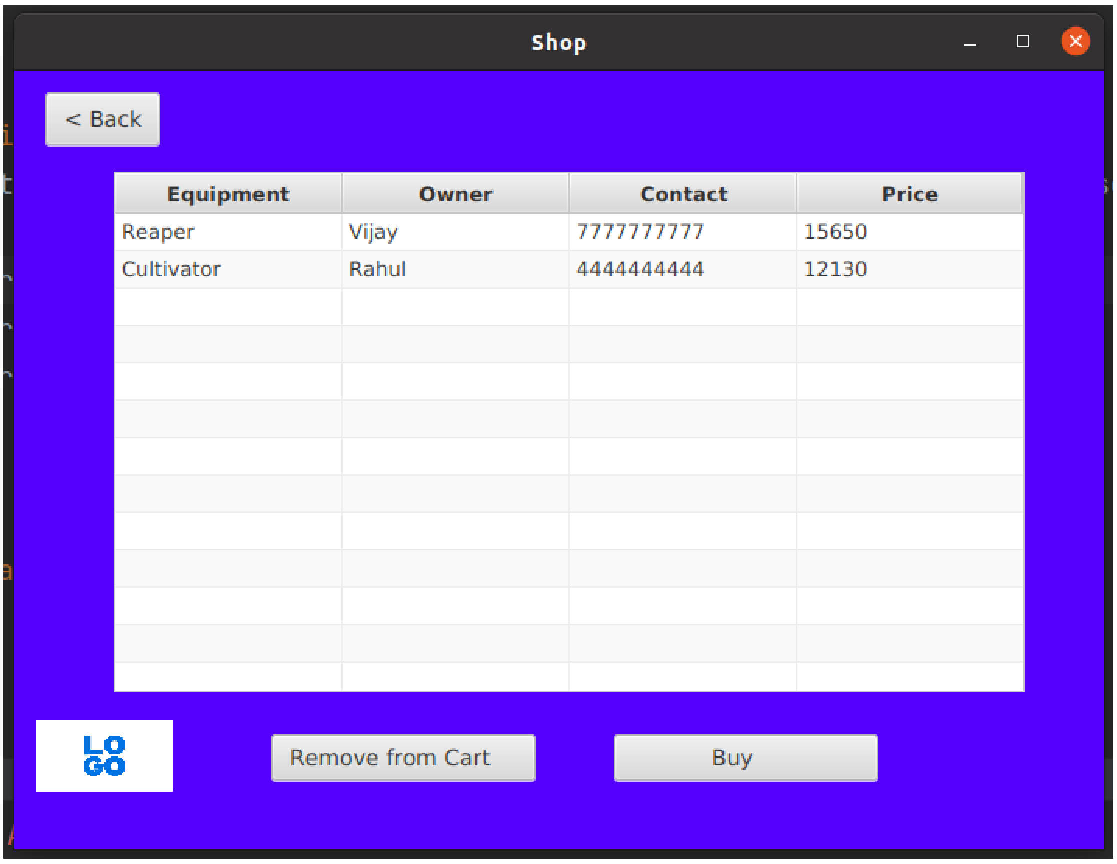


II. Shop Page



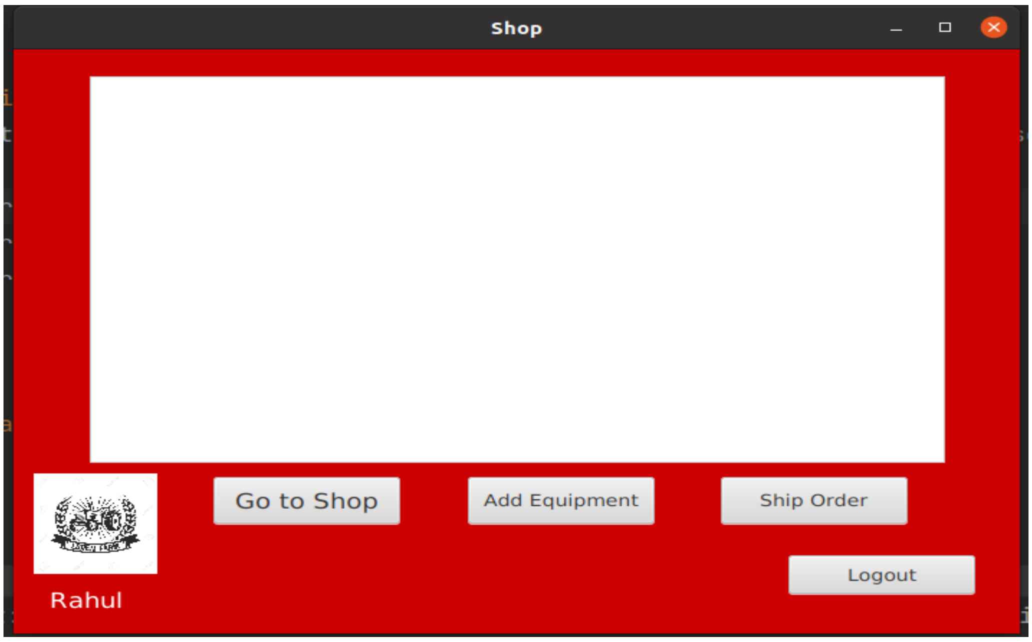
| Name | Owner | Contact | Price |
|---|---|---|---|
| Harvester | Sunil | 9999999... | 35000 |
| Tractor | Vishal | 8888888... | 25750 |
| Reaper | Vijay | 7777777... | 15650 |
| Harvester | Sham | 6666666... | 31000 |
| Trolley | Hari | 5555555... | 34220 |
| Cultivator | Rahul | 4444444... | 12130 |

III. Cart Page

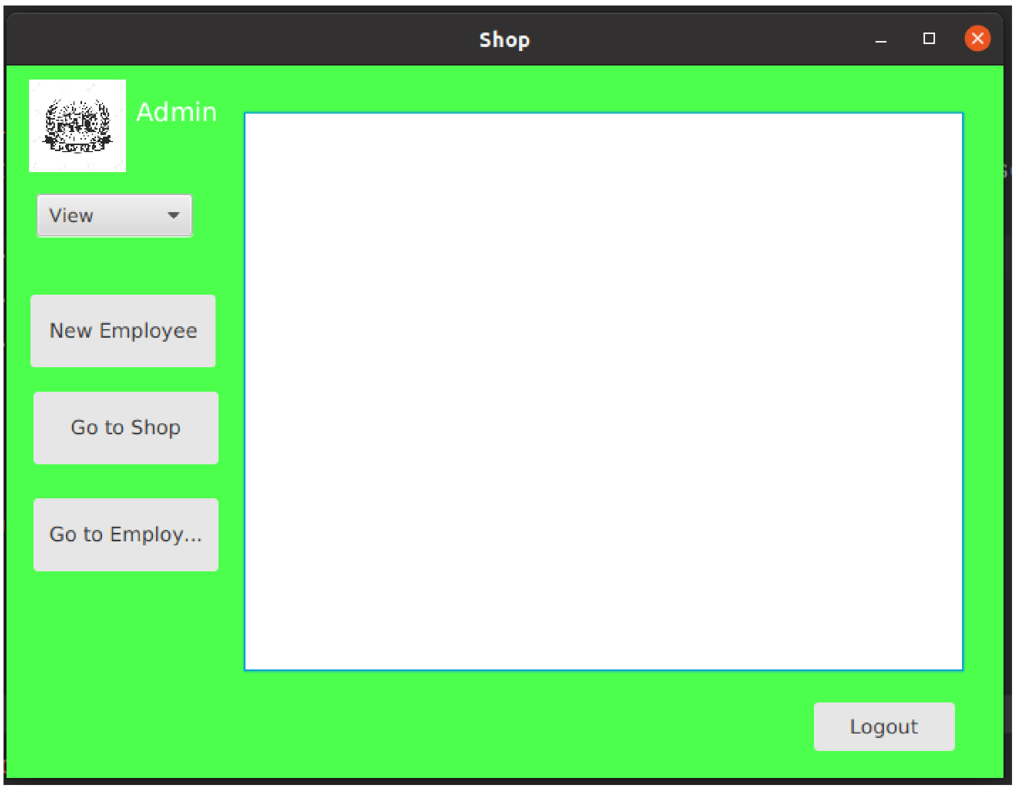


| Equipment | Owner | Contact | Price |
|---|---|---|---|
| Reaper | Vijay | 7777777777 | 15650 |
| Cultivator | Rahul | 4444444444 | 12130 |

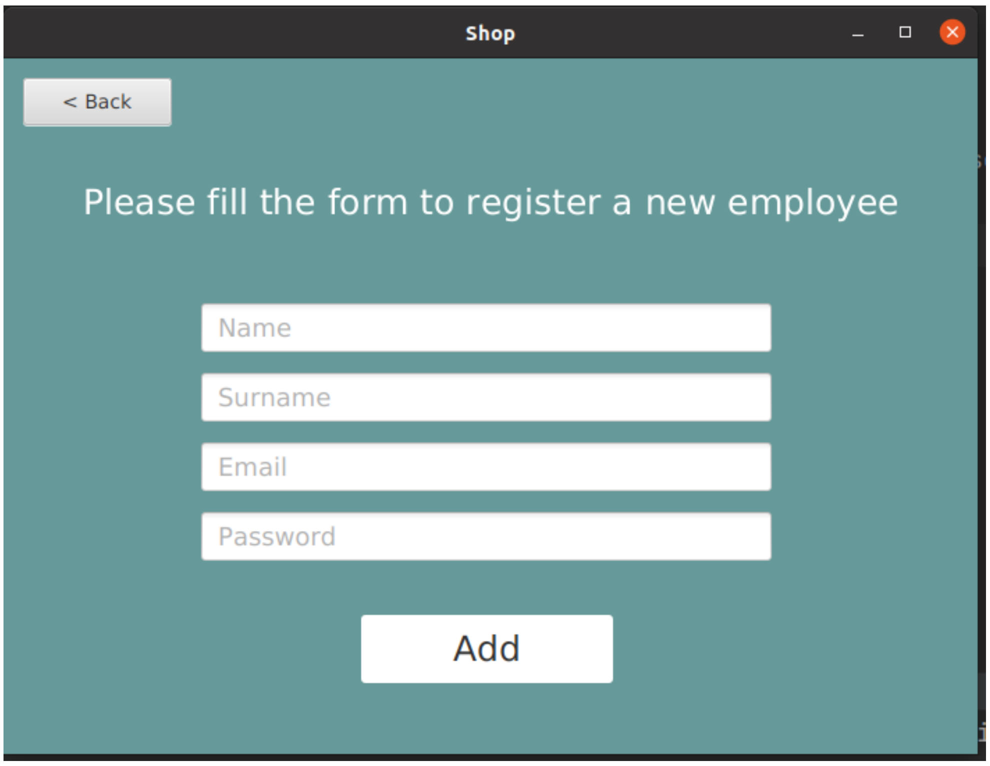
IV. Employee Page



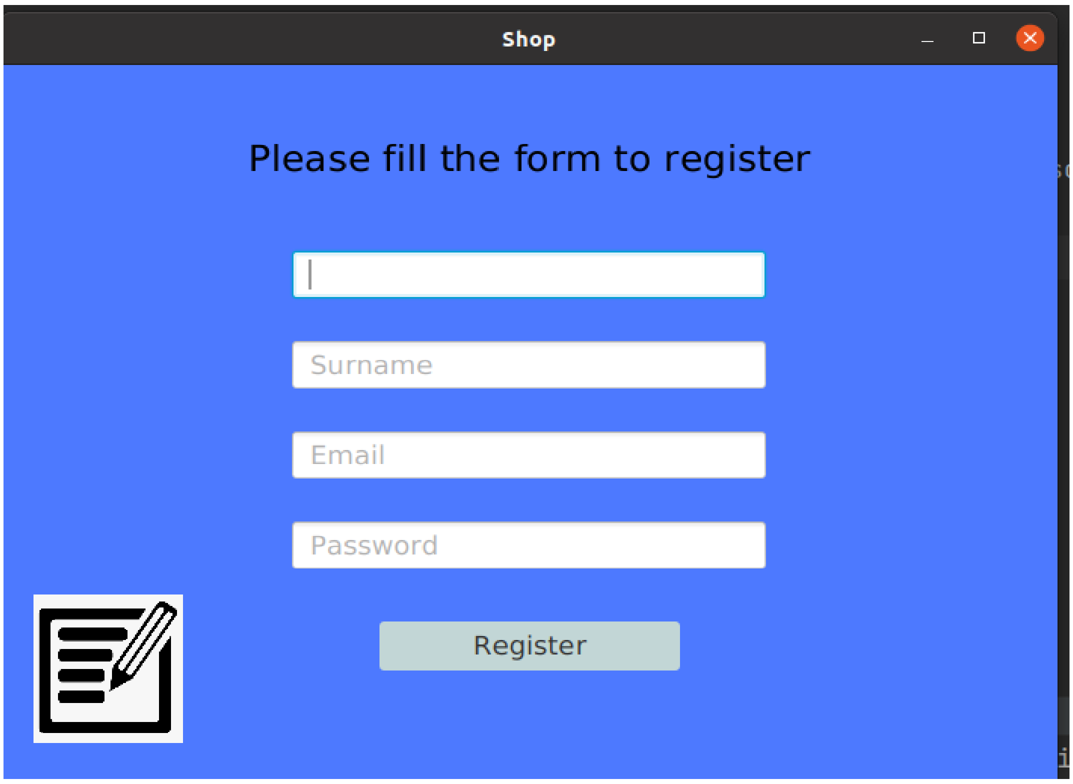
V. Admin Page



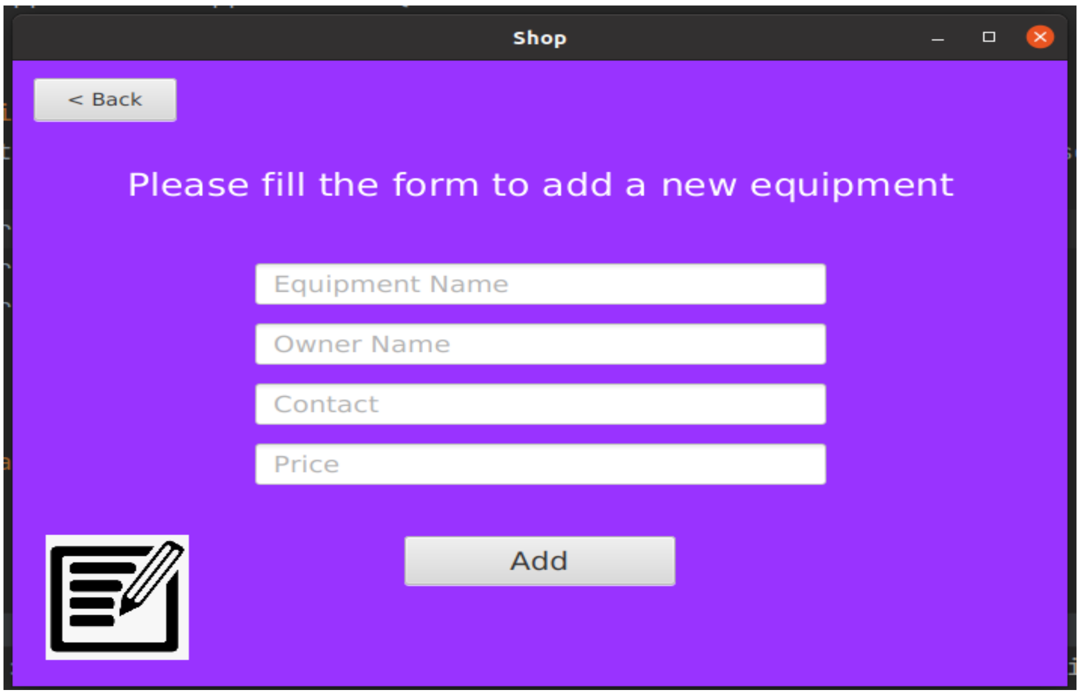
VI. Add Employee Page



VII. New User Register Page



VIII. Add Equipment page



## 5. Conclusion

JAVAFX is a very useful and easy to learn software platform for creating and delivering desktop applications, as well as rich web applications that can run across a wide variety of devices. JavaFX has support for desktop computers and web browsers on Microsoft Windows, Linux, and macOS, as well as mobile devices running iOS and Android.
The JavaFX transition classes greatly simplify the creation of animations for various use cases. You can work with the basic fade, fill, path, pause, rotate, scale, stroke, and translate transitions; and you can leverage the compound transitions to create arbitrarily complex hierarchies of basic and compound transitions that run in parallel or sequentially. You can even create your own transition classes without too much effort.
And our product made can be very useful in simplifying the sophisticated process of farm equipment retail, thus eradicating the struggles caused to a farmer in finding adequate and appropriate farm equipment.

## 6. References

- https://openjfx.io/highlights/16/
- "OpenJFX Project". Oracle Corporation. Retrieved 2011-12-07.
- "JavaFX ComboBox not responding on Windows 10". stackoverflow.com. Retrieved 2018-05-01.
- "JavaFX Supported Configurations". Oracle.com. 2014-04-08. Retrieved 2016-08-01.
- "Oracle Technology Network for Java Developers | Oracle Technology Network | Oracle". Java.sun.com. Retrieved 2016-08-01.
- "Java Client Roadmap Update"(PDF). Oracle. March 2018. Retrieved March 23, 2021.
- "Java Client Roadmap Update" (PDF). Oracle. May 11, 2020. Retrieved March 23,2021.
- "JavaFX Developer Home". www.oracle.com. Retrieved 2019-06-14.
- Smith, Donald (March 7, 2018). "The Future of JavaFX and Other Java Client Roadmap Updates".
- "Oracle Java SE Support Roadmap". Oracle Technology Network. Oracle. 2020-05-13. Retrieved 2020-05-31.
- "JavaFXPorts - Gluon". Gluon. Retrieved 2018-05-01.
- "Rapid Enterprise Mobile Apps: Build, Connect, Manage with Gluon". 2017-12-16. Retrieved 2017-12-16.
- "Google Code Archive - Long-term storage for Google Code Project Hosting". Code.google.com. Retrieved 2016-08-01.
- "Archived copy". Archived from the original on 2012-12-01. Retrieved 2012-11-24.
- "F3 ( Chris Oliver's Weblog)". 2012-01-06. Archived from the original on 2012-01-06. Retrieved 2016-08-20.
- "Repositories and Releases". 2013-06-29. Retrieved 2013-10-18.
- Marinacci, Joshua (2009-06-09). "Top 5 Most Important Features in JavaFX 1.2". Archived from the original on 2009-06-13. Retrieved 2009-06-12.
- "JavaFX 1.3 Released, Improves User Experiences". 2010-04-22. Archived from the original on 2011-04-30. Retrieved 2010-04-25.
- Meyer, David (2011-10-06). "JavaFX 2.0 arrives and heads for open source". ZDNet. Retrieved 2011-10-09.
- Brown, Greg (2011-08-15). "Introducing FXML" (PDF). Retrieved 2011-10-09.
- "JDK 7u4 and JavaFX 2.1 released, now also including Mac OS X | Across the Universe". Terrencebarr.wordpress.com. 2012-04-27. Retrieved 2016-08-01.
- "JavaFX 2.1 Release Notes". Oracle Corporation. Retrieved 2012-05-05.

# APPENDIX

## Source Code:

### a. App.java

```
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.util.Objects;

public class App extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root =
FXMLLoader.load(Objects.requireNonNull(getClass().getResource("login.fxml")));
        Scene scene = new Scene(root);
        primaryStage.setTitle("Shop");
        primaryStage.setScene(scene);
        primaryStage.show();
    }


    public static void main(String[] args) {
        launch(args);
    }
}
```

### b. Login.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.Cursor?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>

<AnchorPane fx:id="rootPane" maxHeight="-Infinity" maxWidth="-Infinity"
minHeight="500.0" minWidth="700.0" prefHeight="500.0" prefWidth="700.0" style="-fx-
background-color: cyan;" xmlns="http://javafx.com/javafx/16"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.ControllerLogin">
```

```
    <children>
        <VBox prefHeight="200.0" prefWidth="100.0" />
        <Button defaultButton="true" layoutX="238.0" layoutY="293.0"
mnemonicParsing="false" onAction="#login" prefHeight="28.0" prefWidth="100.0"
text="Login" AnchorPane.leftAnchor="238.0" AnchorPane.rightAnchor="262.0">
            <cursor>
                <Cursor fx:constant="HAND" />
            </cursor>
            <font>
                <Font size="18.0" />
            </font>
        </Button>
        <Button layoutX="237.0" layoutY="377.0" maxWidth="200.0" minWidth="50.0"
mnemonicParsing="false" onAction="#loadRegister" prefHeight="33.0" prefWidth="200.0"
text="Register" AnchorPane.bottomAnchor="90.0" AnchorPane.leftAnchor="237.0"
AnchorPane.rightAnchor="263.0">
            <cursor>
                <Cursor fx:constant="HAND" />
            </cursor>
            <font>
                <Font size="18.0" />
            </font>
        </Button>
        <TextField fx:id="email" layoutX="199.0" layoutY="194.0" prefHeight="33.0"
prefWidth="291.0" promptText="Email" AnchorPane.leftAnchor="199.0"
AnchorPane.rightAnchor="210.0">
            <font>
                <Font size="18.0" />
            </font>
        </TextField>

        <PasswordField fx:id="password" layoutX="199.0" layoutY="240.0" prefHeight="33.0"
prefWidth="291.0" promptText="Password" AnchorPane.leftAnchor="199.0"
AnchorPane.rightAnchor="210.0">
            <font>
                <Font size="18.0" />
            </font>
        </PasswordField>
        <Text fill="#da1515" layoutX="229.0" layoutY="366.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="If you are not registered" textAlignment="CENTER"
AnchorPane.bottomAnchor="129.7548828125" AnchorPane.leftAnchor="229.0"
AnchorPane.rightAnchor="252.099609375">
            <font>
                <Font size="18.0" />
            </font>
        </Text>
        <Button layoutX="498.0" layoutY="435.0" mnemonicParsing="false"
onAction="#guestLogin" text="Continue as Guest">
            <font>
                <Font name="Calibri" size="14.0" />
            </font></Button>
      <ImageView fitHeight="150.0" fitWidth="200.0" layoutX="262.0" layoutY="25.0"
pickOnBounds="true" preserveRatio="true">
         <image>
```

```
          <Image url="@image/logo.jpg" />
        </image>
      </ImageView>
    </children>
</AnchorPane>
```

**c. ControllerLogin.java**

```
package sample;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import Serverpkg.Server;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.AnchorPane;

public class ControllerLogin implements Controller{
    private User currentUser;

    @FXML
    private AnchorPane rootPane;

    @FXML
    private TextField email;

    @FXML
    private PasswordField password;


    public Boolean isMail(String mail) {
        String mailRegex = "\\w+@\\w+\\.\\w+";
        Pattern mailValidator = Pattern.compile(mailRegex);
        Matcher mailMatcher = mailValidator.matcher(mail);
        return mailMatcher.matches();
    }



    @FXML
    public void loadRegister(ActionEvent event) throws IOException {
        Loader loader = new Loader(new User(),this.rootPane);
        loader.load("register");
    }


    @FXML
```

```
public void login(ActionEvent event) throws IOException {
    String mail = email.getText();
    String pass = password.getText();

    try {
        // checks if the email and the password have been inserted
        if (mail.length() > 0 && pass.length() > 0) {
            // checks if the email is valid
            if (isMail(mail)) {
                List<String> request = new ArrayList<>();
                request.add("login");
                request.add(mail);
                request.add(pass);

                ArrayList<User> response = Server.run(request);
                User user = response.get(0);
                int permission = user.getPermission();
                this.currentUser = user;

                Loader loader = new Loader(this.currentUser, this.rootPane);
                // different cases are distinguished based on the User's permission
                switch (permission) {
                    // permission = 1 ->it's a user and the user's homepage is loaded
                    case 1:
                        loader.load("homepage_user");
                        break;

                    // permission = 2 ->it's an employee and the employee's homepage is loaded
                    case 2:
                        loader.load("homepage_employee");
                        break;

                    // permission = 3 ->it's an admin and the admin's homepage is loaded
                    case 3:
                        loader.load("homepage_admin");
                        break;

                    default:
                        // notifies the user if a wrong email or password are inserted
                        Alert alert = new Alert(AlertType.WARNING);
                        alert.setTitle("Wrong login");
                        alert.setHeaderText("Email or password are wrong. Please retry.");
                        alert.showAndWait();
                        password.clear();
                        break;
                }
            } else {
                // notifies the user if the email is not valid
                myThread mt = new myThread();
                mt.run();
            }
        } else {
            // notifies if some fields are not filled
            Alert alert = new Alert(AlertType.WARNING);
```

```
                alert.setTitle("All fields must be filled");
                alert.setHeaderText("Please fill all the fields");
                alert.showAndWait();
            }
        } catch (IOException e) {
            System.out.println(e);
            Alert alert = new Alert(AlertType.ERROR);
            alert.setTitle("Page Missing");
            alert.setHeaderText("Page is unreachable. Try again later");
            alert.showAndWait();

        }catch (Exception e) {
            System.out.println(e);
            Alert alert = new Alert(AlertType.ERROR);
            alert.setTitle("Cannot connect");
            alert.setHeaderText("Try again later.");
            alert.showAndWait();
        }
    }

    @FXML
    public void guestLogin(ActionEvent event) throws IOException {
        try {
            List<String> request = new ArrayList<>();
            request.add("guest");
            ArrayList<User> response = Server.run(request);
            response.get(0);

            Loader loader = new Loader(this.currentUser, this.rootPane);
            loader.load("homepage_user");

        } catch (IOException e) {
            Alert alert = new Alert(AlertType.ERROR);
            alert.setTitle("Page Missing");
            alert.setHeaderText("Page is unreachable. Try again later");
            alert.showAndWait();
        }catch(Exception e){
            System.out.println(e);
            Alert alert = new Alert(AlertType.ERROR);
            alert.setTitle("Cannot connect");
            alert.setHeaderText("Try again later.");
            alert.showAndWait();
        }

    }

    @Override
    public void initData(User user) {
        this.currentUser = user;
    }

}

class myThread extends Thread{
    public void run(){
        Alert alert = new Alert(AlertType.WARNING);
        alert.setTitle("Email not valid");
        alert.setHeaderText("The provided email is not valid, please retry.");
        alert.showAndWait();
    }
}
```

**d. homepage_admin.fxml**

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.MenuButton?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.control.TreeView?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>

<AnchorPane fx:id="rootPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-
Infinity" minWidth="-Infinity" prefHeight="500.0" prefWidth="700.0" style="-fx-background-
color: #4dff4d;" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sample.ControllerHomepageAdmin">
    <children>
        <TreeView fx:id="treeView" layoutX="167.0" layoutY="33.0" prefHeight="392.0"
prefWidth="505.0" AnchorPane.bottomAnchor="75.0" AnchorPane.leftAnchor="167.0"
AnchorPane.rightAnchor="28.0" />
        <ImageView fitHeight="65.0" fitWidth="68.0" layoutX="16.0" layoutY="10.0">
            <image>
                <Image url="@image/logo.jpg" />
            </image>
        </ImageView>
        <Text fx:id="name" fill="WHITE" layoutX="91.0" layoutY="39.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="FMACH">
            <font>
                <Font name="Calibri" size="18.0" />
            </font>
        </Text>
        <Button layoutX="567.0" layoutY="447.0" mnemonicParsing="false" onAction="#logout"
prefHeight="33.0" prefWidth="99.0" style="-fx-background-color: #e6e6e6;" text="Logout">
            <font>
                <Font name="Calibri" size="14.0" />
            </font></Button>
        <MenuButton layoutX="21.0" layoutY="90.0" mnemonicParsing="false"
prefHeight="30.0" prefWidth="110.0" text="View">
            <items>
                <MenuItem mnemonicParsing="false" onAction="#displayOrders" text="Orders" />
                <MenuItem mnemonicParsing="false" onAction="#displayEmployees"
text="Employees" />
                <MenuItem mnemonicParsing="false" onAction="#displayEquipments"
text="Equipments" />
```

```
            <MenuItem mnemonicParsing="false" onAction="#displayUsers" text="Users" />
          </items>
          <font>
            <Font name="Calibri" size="13.0" />
          </font>
      </MenuButton>
        <Button layoutX="17.0" layoutY="161.0" mnemonicParsing="false"
onAction="#loadAddEmployee" prefHeight="50.0" prefWidth="130.0" style="-fx-background-
color: #e6e6e6;" text="New Employee">
           <font>
              <Font name="Calibri" size="14.0" />
           </font>
     </Button>
        <Button layoutX="19.0" layoutY="229.0" mnemonicParsing="false"
onAction="#loadUser" prefHeight="50.0" prefWidth="130.0" style="-fx-background-color:
#e6e6e6;" text="Go to Shop">
           <font>
              <Font name="Calibri" size="14.0" />
           </font>
     </Button>
        <Button layoutX="19.0" layoutY="304.0" mnemonicParsing="false"
onAction="#loadEmployee" prefHeight="50.0" prefWidth="130.0" style="-fx-background-
color: #e6e6e6;" text="Go to Employee">
           <font>
              <Font name="Calibri" size="14.0" />
           </font>
     </Button>
   </children>
</AnchorPane>
```

**e. ControllerHomePageAdmin**

```
package sample;

import java.io.IOException;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.List;

import Serverpkg.Order;
import Serverpkg.Server;
import Serverpkg.ServerEquipments;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.TreeItem;
import javafx.scene.control.TreeView;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.AnchorPane;
import javafx.scene.text.Text;

public class ControllerHomepageAdmin implements Controller {
   private User currentUser;
```

```
    @FXML
    private AnchorPane rootPane;

    @FXML
    private TreeView<String> treeView;

    @FXML
    private Text name;

    public void initData(User user) {
        this.currentUser = user;
        this.name.setText(this.currentUser.getName());
    }

    @FXML
    @SuppressWarnings("unchecked")
    public void displayEmployees(ActionEvent event) throws IOException {
        if (this.currentUser.getPermission() > 2) {

            List<String> request = new ArrayList<>();
            request.add("get_employees");

            try {
                ArrayList<User> employees = Server.run(request);
                TreeItem<String> rootItem = new TreeItem<String>("Employees");

                for (User employee : employees) {
                    TreeItem<String> rootEmployee = new TreeItem<String>(employee.getEmail());
                    TreeItem<String> name = new TreeItem<String>("Name: " +
employee.getName());
                    TreeItem<String> surname = new TreeItem<String>("Surname: " +
employee.getSurname());
                    rootEmployee.getChildren().addAll(name, surname);
                    rootItem.getChildren().add(rootEmployee);
                }
                treeView.setRoot(rootItem);
                treeView.setShowRoot(false);
            } catch (Exception e) {
                System.out.println(e);
            }
        } else {
            // user is not authorized to perform the action
            Alert alert = new Alert(AlertType.ERROR);
            alert.setTitle("Not authorized");
            alert.setHeaderText("You are not allowed to perform this action.");
            alert.showAndWait();
        }
    }

    @FXML
    public void displayOrders(ActionEvent event) throws IOException {
        if (this.currentUser.getPermission() > 2) {

            try {
```

```
            ArrayList<Order> orders = new ArrayList<>();
            List<String> request = new ArrayList<>();
            request.add("get_orders");
            orders = ServerEquipments.orderRun(request);
            TreeItem<String> rootItem = new TreeItem<String>("Orders");

            for (Order order : orders) {
                TreeItem<String> rootOrder = new TreeItem<String>(order.getEmail());

                ArrayList<String> orderId = order.getOrderId();
                ArrayList<Equipment> orderEquipments = order.getOrderedEquipments();
                ArrayList<Boolean> shipped = order.isShipped();

                for (int i=0;i<orderId.size();i++) {
                    TreeItem<String> rootProduct = new TreeItem<String>(orderId.get(i));
                    TreeItem<String> equipment = new TreeItem<String>(
                            String.format("%s(%s) - %s",orderEquipments.get(i).getEquipmentName(),
                                    orderEquipments.get(i).getEquipmentId(),
orderEquipments.get(i).getOwner()));
                    TreeItem<String> contact = new
TreeItem<>(String.format("Contact->%s",orderEquipments.get(i).getContact()));
                    TreeItem<String> price = new
TreeItem<>(String.format("price->%s",orderEquipments.get(i).getPrice()));
                    TreeItem<String> ship = new
TreeItem<String>(String.format("Shipped->%s",shipped.get(i)));
                    rootProduct.getChildren().addAll(equipment,contact,price,ship);
                    rootOrder.getChildren().add(rootProduct);
                }
                rootItem.getChildren().add(rootOrder);
            }
            treeView.setRoot(rootItem);
            treeView.setShowRoot(false);
        } catch (Exception e) {
            System.out.println(e);
        }
    } else {
        // user is not authorized to perform the action
        Alert alert = new Alert(AlertType.ERROR);
        alert.setTitle("Not authorized");
        alert.setHeaderText("You are not allowed to perform this action.");
        alert.showAndWait();
    }
}


@FXML
public void displayUsers(ActionEvent event) throws UnknownHostException, IOException {
    if (this.currentUser.getPermission() > 2) {

        List<String> request = new ArrayList<>();
        request.add("get_users");
        try {
            ArrayList<User> users = Server.run(request);
            TreeItem<String> rootItem = new TreeItem<String>("Users");
            for (User user : users) {
                TreeItem<String> rootUser = new TreeItem<String>(user.getEmail());
                TreeItem<String> name = new TreeItem<String>("Name: " + user.getName());
                TreeItem<String> surname = new TreeItem<String>("Surname: " +
user.getSurname());
                rootUser.getChildren().addAll(name, surname);
                rootItem.getChildren().add(rootUser);
            }
            treeView.setRoot(rootItem);
            treeView.setShowRoot(false);
        } catch (Exception e) {
            System.out.println(e);
        }
    } else {
        // user is not authorized to perform the action
        Alert alert = new Alert(AlertType.ERROR);
        alert.setTitle("Not authorized");
        alert.setHeaderText("You are not allowed to perform this action.");
        alert.showAndWait();
    }
}

@FXML
public void displayEquipments(ActionEvent event) throws IOException {
    if (this.currentUser.getPermission() > 2) {
        // user is authorized to perform the action
        List<String> request = new ArrayList<>();
        request.add("get_equipments");

        try {
            ArrayList<Equipment> equipments = ServerEquipments.run(request);
            TreeItem<String> rootItem = new TreeItem<String>("Equipments");

            for (Equipment equipment : equipments) {
                TreeItem<String> equipmentName = new
TreeItem<String>(equipment.getEquipmentName());
                TreeItem<String> equipmentId = new TreeItem<String>(
                        "Equipment ID: " + equipment.getEquipmentId());
                TreeItem<String> owner = new TreeItem<String>("Owner: " +
equipment.getOwner());
                TreeItem<String> contact = new TreeItem<String>("Contact: " +
equipment.getContact());
                TreeItem<String> price = new TreeItem<String>(
                        "Price: " + Integer.toString(equipment.getPrice()));
                equipmentName.getChildren().addAll(equipmentId,owner,contact,price);
                rootItem.getChildren().add(equipmentName);
            }
            treeView.setRoot(rootItem);
            treeView.setShowRoot(false);
        } catch (Exception e) {
            System.out.println(e);
        }
    } else {
        // user is not authorized to perform the action
        Alert alert = new Alert(AlertType.ERROR);
        alert.setTitle("Not authorized");
        alert.setHeaderText("You are not allowed to perform this action.");
        alert.showAndWait();
    }
}

@FXML
public void loadEmployee(ActionEvent event) throws IOException {
    Loader loader = new Loader(this.currentUser, this.rootPane);
    loader.load("homepage_employee");
}

@FXML
public void loadUser(ActionEvent event) throws IOException {
    Loader loader = new Loader(this.currentUser, this.rootPane);
    loader.load("homepage_user");
}

@FXML
public void loadAddEmployee(ActionEvent event) throws IOException {
    Loader loader = new Loader(this.currentUser, this.rootPane);
    loader.load("add_employee");
}


@FXML
public void logout(ActionEvent event) throws IOException {
//        AnchorPane pane = FXMLLoader.load(getClass().getResource("./login.fxml"));
//        rootPane.getChildren().setAll(pane);
    Loader loader = new Loader(new User(),this.rootPane);
    loader.load("login");
}
}
```

**f. homepage_employee.fxml**

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.TreeView?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>

<AnchorPane fx:id="rootPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-
Infinity" minWidth="-Infinity" prefHeight="500.0" prefWidth="700.0" style="-fx-background-
color: #cc0000;" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sample.ControllerHomepageEmployee">
    <children>
```

```
        <HBox alignment="TOP_CENTER" layoutX="6.0" layoutY="23.0" prefHeight="331.0"
prefWidth="689.0" spacing="20.0">
            <children>
                <TreeView fx:id="treeView" prefHeight="331.0" prefWidth="595.0" />
            </children>
        </HBox>
        <Button layoutX="539.0" layoutY="433.0" mnemonicParsing="false" onAction="#logout"
prefHeight="33.0" prefWidth="130.0" text="Logout">
            <font>
                <Font name="Calibri" size="14.0" />
            </font></Button>
        <Text fx:id="name" fill="WHITE" layoutX="25.0" layoutY="478.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="FMACH">
            <font>
                <Font name="Calibri" size="18.0" />
            </font>
        </Text>
      <ImageView fitHeight="103.0" fitWidth="86.0" layoutX="14.0" layoutY="363.0"
pickOnBounds="true" preserveRatio="true">
         <image>
            <Image url="@image/logo.jpg" />
         </image>
      </ImageView>
            <Button layoutX="316.0" layoutY="366.0" mnemonicParsing="false"
onAction="#loadAddEquipment" prefHeight="40.0" prefWidth="130.0" text="Add
Equipment">
                <font>
                    <Font name="Calibri" size="14.0" />
                </font>
            </Button>
            <Button layoutX="492.0" layoutY="366.0" mnemonicParsing="false"
onAction="#shipOrder" prefHeight="40.0" prefWidth="130.0" text="Ship Order">
                <font>
                    <Font name="Calibri" size="14.0" />
                </font>
            </Button>
            <Button layoutX="139.0" layoutY="366.0" mnemonicParsing="false"
onAction="#loadShop" prefHeight="40.0" prefWidth="130.0" text="Go to Shop">
                <font>
                    <Font size="18.0" />
                </font>
            </Button>
    </children>
</AnchorPane>
```

**g. ControllerHomePageEmployee.java**

```
package sample;

import java.io.IOException;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.List;
import Serverpkg.Order;
import Serverpkg.ServerEquipments;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.TreeItem;
import javafx.scene.control.TreeView;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.AnchorPane;
import javafx.scene.text.Text;

public class ControllerHomepageEmployee implements Controller {

    private User currentUser;

    @FXML
    private AnchorPane rootPane;

    @FXML
    private TreeView<String> treeView;

    @FXML
    private Text name;


    public void initData(User user) {
        this.currentUser = user;
        name.setText(this.currentUser.getName());

        try {
            displayOrders();
        } catch (IOException e) {
            System.out.println(e);
        }
    }


    @FXML
    public void loadAddEquipment(ActionEvent event) throws IOException {
        Loader loader = new Loader(this.currentUser, this.rootPane);
        loader.load("add_equipment");
    }


    @FXML
    public void loadShop(ActionEvent event) throws IOException {
        Loader loader = new Loader(this.currentUser, this.rootPane);
        loader.load("homepage_user");
    }


    @FXML
    public void shipOrder(ActionEvent event) throws UnknownHostException, IOException {
        if (this.currentUser.getPermission() > 1) {
            // user is not authorized to perform the action
            TreeItem<String> selectedItem = treeView.getSelectionModel().getSelectedItem();
            // the item is selected
            if (selectedItem != null) {

                while (selectedItem.getParent() != treeView.getRoot()) {
                    selectedItem = selectedItem.getParent();
                }
                // warning message if the item is not selected
            } else {
                Alert alert = new Alert(AlertType.WARNING);
                alert.setTitle("Select an order");
                alert.setHeaderText("Please select an order.");
                alert.showAndWait();
                return;
            }
            Boolean shipped=false;
            try {
                List<String> request = new ArrayList<>();
                request.add("shipOrder");
                request.add(selectedItem.getValue());
                System.out.println(selectedItem.getValue());
                shipped = ServerEquipments.run(request, new Equipment());
                // server's answer is true -> the order has been shipped correctly
                if (shipped) {
                    displayOrders();
                    Alert alert = new Alert(AlertType.INFORMATION);
                    alert.setTitle("Shipping successfull");
                    alert.setHeaderText(
                            String.format("Email %s orders has been shipped", selectedItem.getValue()));
                    alert.showAndWait();
                    // server's answer is false -> the order has not been shipped correctly
                } else {
                    Alert alert = new Alert(AlertType.ERROR);
                    alert.setTitle("Shipping failed.");
                    alert.setHeaderText("Unable to ship order. Try again later.");
                    alert.showAndWait();
                }
            } catch (Exception e) {
                System.out.println(e);
            }
            // the page is refreshed and the orders are updated (only not shipped
            // orders can be visualized by the employee)
            initData(this.currentUser);
        } else {

            // user is not authorized to perform the action
            Alert alert = new Alert(AlertType.ERROR);
            alert.setTitle("Not authorized");
            alert.setHeaderText("You are not allowed to perform this action.");
            alert.showAndWait();
        }
    }
```

```
    @FXML
    public void logout(ActionEvent event) throws IOException {
        Loader loader = new Loader(new User(),rootPane);
        loader.load("login");
    }

    public void displayOrders() throws IOException {
        if (this.currentUser.getPermission() > 1) {
            try {
                ArrayList<Order> orders = new ArrayList<>();
                List<String> request = new ArrayList<>();
                request.add("get_orders_employee");
                orders = ServerEquipments.orderRun(request);
                TreeItem<String> rootItem = new TreeItem<String>("Orders");

                for (Order order : orders) {
                    TreeItem<String> rootOrder = new TreeItem<String>(order.getEmail());

                    ArrayList<String> orderId = order.getOrderId();
                    ArrayList<Equipment> orderEquipments = order.getOrderedEquipments();
                    ArrayList<Boolean> shipped = order.isShipped();

                    for (int i=0;i<orderId.size();i++) {
                        TreeItem<String> rootProduct = new TreeItem<String>(orderId.get(i));
                        TreeItem<String> equipment = new TreeItem<String>(
                                String.format("%s(%s) - %s",orderEquipments.get(i).getEquipmentName(),
                                        orderEquipments.get(i).getEquipmentId(),
orderEquipments.get(i).getOwner()));
                        TreeItem<String> contact = new
TreeItem<>(String.format("Contact->%s",orderEquipments.get(i).getContact()));
                        TreeItem<String> price = new
TreeItem<>(String.format("price->%s",orderEquipments.get(i).getPrice()));
                        TreeItem<String> ship = new
TreeItem<String>(String.format("Shipped->%s",shipped.get(i)));
                        rootProduct.getChildren().addAll(equipment,contact,price,ship);
                        rootOrder.getChildren().add(rootProduct);
                    }
                    rootItem.getChildren().add(rootOrder);
                }
                treeView.setRoot(rootItem);
                treeView.setShowRoot(false);
            } catch (Exception e) {
                System.out.println(e);
            }
        } else {
            // user is not authorized to perform the action
            Alert alert = new Alert(AlertType.ERROR);
            alert.setTitle("Not authorized");
            alert.setHeaderText("You are not allowed to perform this action.");
            alert.showAndWait();
        }
    }
}
```

**h. homepage_user.fxml**

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.Cursor?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.MenuButton?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.TreeView?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>

<AnchorPane fx:id="rootPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-
Infinity" minWidth="-Infinity" prefHeight="500.0" prefWidth="700.0" style="-fx-background-
color: #0033cc;" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sample.ControllerHomepageUser">
   <children>
      <TableView fx:id="tableView" layoutX="26.0" layoutY="89.0" prefHeight="328.0"
prefWidth="341.0" style="-fx-background-color: #a6a6a6;">
         <columns>
            <TableColumn fx:id="nameColumn" editable="false" prefWidth="99.0"
sortable="false" text="Name" />
            <TableColumn fx:id="ownerColumn" editable="false" prefWidth="115.0"
sortable="false" text="Owner" />
            <TableColumn fx:id="contactColumn" editable="false" prefWidth="55.0"
sortable="false" text="Contact" />
            <TableColumn fx:id="priceColumn" editable="false" prefWidth="65.0"
sortable="false" text="Price" />
         </columns>
         <columnResizePolicy>
            <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
         </columnResizePolicy>
      </TableView>
      <Button layoutX="153.0" layoutY="434.0" mnemonicParsing="false"
onAction="#addToCart" prefHeight="28.0" prefWidth="156.0" text="Add to Cart🛒">
         <font>
            <Font name="Calibri" size="14.0" />
         </font></Button>
<!--      <TextField fx:id="quantity" layoutX="249.0" layoutY="346.0" prefHeight="27.0"
prefWidth="69.0" promptText="Quantity">-->
<!--         <font>-->
<!--            <Font name="Calibri" size="14.0" />-->
<!--         </font></TextField>-->
      <TreeView fx:id="treeView" layoutX="411.0" layoutY="118.0" prefHeight="299.0"
prefWidth="268.0" style="-fx-background-color: #a6a6a6;" />
      <Text fill="WHITE" layoutX="481.0" layoutY="102.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="My Orders">
         <font>
```

```
              <Font name="Calibri Bold" size="14.0" />
           </font>
        </Text>
          <MenuButton layoutX="536.0" layoutY="31.0" mnemonicParsing="false" text="My
Profile 👤  ">
             <items>
                <MenuItem fx:id="menuItemLogout" mnemonicParsing="false"
onAction="#logout" text="Back to Login" />
             </items>
             <font>
                <Font name="Calibri" size="14.0" />
             </font>
          </MenuButton>
          <Button layoutX="426.0" layoutY="31.0" mnemonicParsing="false"
onAction="#showCart" text="Cart 🛒">
             <cursor>
                <Cursor fx:constant="HAND" />
             </cursor>
             <font>
                <Font name="Calibri" size="14.0" />
             </font>
          </Button>
          <Button defaultButton="true" layoutX="245.0" layoutY="31.0"
mnemonicParsing="false" onAction="#search" prefHeight="28.0" prefWidth="89.0"
text="Search 🔍">
             <cursor>
                <Cursor fx:constant="HAND" />
             </cursor>
             <font>
                <Font name="Calibri" size="14.0" />
             </font>
          </Button>
      <ImageView fitHeight="50.0" fitWidth="90.0" layoutX="15.0" layoutY="434.0"
pickOnBounds="true" preserveRatio="true">
         <image>
            <Image url="@image/logo.jpg" />
         </image>
      </ImageView>
          <TextField fx:id="searchboxName" layoutX="26.0" layoutY="31.0" prefHeight="28.0"
prefWidth="192.0" promptText="Name">
             <font>
                <Font name="Calibri" size="14.0" />
             </font>
      </TextField>
    </children>
</AnchorPane>
```

**i. ControllerHomePageUser.java**

```
package sample;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import Serverpkg.EquipmentDB;
import Serverpkg.Order;
import Serverpkg.ServerEquipments;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TreeItem;
import javafx.scene.control.TreeView;
import javafx.scene.control.TextField;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.AnchorPane;

public class ControllerHomepageUser implements Controller {

    private User currentUser;

    @FXML
    private AnchorPane rootPane;

    @FXML
    private TextField searchboxName;

    @FXML
    private TreeView<String> treeView;

    @FXML
    private TableView<Equipment> tableView;

    @FXML
    private TableColumn<Equipment, String> nameColumn;

    @FXML
    private TableColumn<Equipment, String> ownerColumn;

    @FXML
    private TableColumn<Equipment, Integer> priceColumn;

    @FXML
    private TableColumn<Equipment, String> contactColumn;


    public void initData(User user) {
        this.currentUser = user;

        try {
            ArrayList<Equipment> equipments = new ArrayList<>();
            List<String> request = new ArrayList<>();
            request.add("get_equipments");
```

```
//          equipments = ServerEquipments.run(request,equipments);
          equipments = EquipmentDB.getEquipments();
          // adds the equipments of the shop to the TableView to be displayed
          addToTable(equipments);

          // displays the orders made by the User
          displayOrders();

      } catch (Exception e) {
          System.out.println(e);
      }
  }

  public void addToTable(ArrayList<Equipment> equipments) {
      // set up the columns in the table
      nameColumn.setCellValueFactory(new PropertyValueFactory<Equipment,
String>("equipmentName"));
      ownerColumn.setCellValueFactory(new
PropertyValueFactory<Equipment,String>("Owner"));
      contactColumn.setCellValueFactory(new
PropertyValueFactory<Equipment,String>("Contact"));
      priceColumn.setCellValueFactory(new
PropertyValueFactory<Equipment,Integer>("Price"));
      System.out.println(equipments);
      ObservableList<Equipment> oListequipments =
FXCollections.observableArrayList(equipments);
      // load data
      tableView.setItems(oListequipments);
  }



  @FXML
  public void addToCart(ActionEvent event) throws IOException {
      // permission check, guests can't add to cart
      if (this.currentUser!=null && this.currentUser.getPermission() > 0) {

          try {

              Equipment equipment = tableView.getSelectionModel().getSelectedItem();
              if(equipment == null){
                  throw new NotSelectedException("Didn't choose an equipment");
              }
              List<String> request = new ArrayList<>();
              request.add("add_to_cart");
              request.add(this.currentUser.getEmail());

              Boolean response = ServerEquipments.run(request,equipment);

              if (response){
                  System.out.println(equipment.getEquipmentId() + " " +
equipment.getEquipmentName());
                  // operation addToCart was successful
                  Alert alert = new Alert(AlertType.INFORMATION);
```

```
                alert.setTitle(String.format("Added to cart"));
                alert.setHeaderText(String.format("Added %s to cart.",
equipment.getEquipmentName()));
                alert.showAndWait();
            } else {
                // operation addToCart was not successful
                Alert alert = new Alert(AlertType.WARNING);
                alert.setTitle(String.format("Select an Equipment"));
                alert.setHeaderText("You have to click on a Equipment, enter the quantity and then
Add.");
                alert.showAndWait();
            }

        }catch (NotSelectedException e){
            System.out.println(e);
            Alert alert = new Alert(AlertType.WARNING);
            alert.setTitle(String.format("Select an Equipment"));
            alert.setHeaderText("You have to click on the equipment to add it to the cart.");
            alert.showAndWait();
        }
        catch (Exception e) {
            System.out.println(e);
        }

    } else {
        // the User is a guest, so he needs to login first
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Please login");
        alert.setHeaderText("You need to login to perform this action.");
        alert.showAndWait();
    }
}


@FXML
public void search(ActionEvent event) throws IOException, ClassNotFoundException {

//      // displays the result in the tableview
    ArrayList<Equipment> Equipments = new ArrayList<>();
    ArrayList<String> request = new ArrayList<>();
    request.add("search");
    request.add(searchboxName.getText());
    Equipments = ServerEquipments.run(request);
    addToTable(Equipments);
}


@FXML
public void showCart(ActionEvent event) throws IOException {
    if (this.currentUser!=null && this.currentUser.getPermission() >0) {
        Loader loader = new Loader(this.currentUser, this.rootPane);
        loader.load("cart");
    } else {
        Alert alert = new Alert(AlertType.INFORMATION);
```

```
            alert.setTitle("Please login");
            alert.setHeaderText("You need to login to perform this action.");
            alert.showAndWait();
        }
    }


    public void displayOrders() throws IOException {
        if(this.currentUser!=null && this.currentUser.getPermission() >0) {
            try {
                ArrayList<Order> orders = new ArrayList<>();
                List<String> request = new ArrayList<>();
                request.add("get_order_user");
                request.add(this.currentUser.getEmail());
                orders = ServerEquipments.orderRun(request);
                Order order = orders.get(0);
                TreeItem<String> rootItem = new TreeItem<String>("Orders");

                TreeItem<String> rootOrder = new TreeItem<String>(order.getEmail());

                ArrayList<String> orderId = order.getOrderId();
                ArrayList<Equipment> orderEquipments = order.getOrderedEquipments();
                ArrayList<Boolean> shipped = order.isShipped();

                for (int i = 0; i < orderId.size(); i++) {
                    TreeItem<String> rootProduct = new TreeItem<String>(orderId.get(i));
                    TreeItem<String> equipment = new TreeItem<String>(
                            String.format("%s(%s) - %s", orderEquipments.get(i).getEquipmentName(),
                                    orderEquipments.get(i).getEquipmentId(),
orderEquipments.get(i).getOwner()));
                    TreeItem<String> contact = new
TreeItem<>(String.format("Contact->%s",orderEquipments.get(i).getContact()));
                    TreeItem<String> price = new
TreeItem<>(String.format("price->%s",orderEquipments.get(i).getPrice()));
                    TreeItem<String> ship = new TreeItem<String>(String.format("Shipped-%s",
shipped.get(i)));
                    rootProduct.getChildren().addAll(equipment, contact,price,ship);
                    rootOrder.getChildren().add(rootProduct);
                }
                rootItem.getChildren().add(rootOrder);

                treeView.setRoot(rootItem);
                treeView.setShowRoot(false);
            } catch (Exception e) {
                System.out.println(e);;
            }
        }
    }


    @FXML
    public void logout(ActionEvent event) throws IOException {
        try{
            Loader loader = new Loader(new User(),rootPane);
            loader.load("login");
        }catch (NullPointerException e){
            System.out.println(e);
        }catch(Exception e){
            System.out.println(e);
        }
    }
}
```

**j. register.fxml**

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.Cursor?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>

<AnchorPane fx:id="rootPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-
Infinity" minWidth="-Infinity" prefHeight="500.0" prefWidth="700.0" style="-fx-background-
color: #4d79ff;" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sample.ControllerRegister">
    <children>
        <VBox layoutX="193.0" layoutY="130.0" prefHeight="240.0" prefWidth="314.0"
spacing="30.0">
            <children>
                <TextField fx:id="name" prefHeight="28.0" prefWidth="181.0"
promptText="Name">
                    <font>
                        <Font size="18.0" />
                    </font>
                </TextField>
                <TextField fx:id="surname" promptText="Surname">
                    <font>
                        <Font size="18.0" />
                    </font>
                </TextField>
                <TextField fx:id="email" promptText="Email">
                    <font>
                        <Font size="18.0" />
                    </font>
                </TextField>
                <PasswordField fx:id="password" promptText="Password">
                    <font>
                        <Font size="18.0" />
                    </font>
                </PasswordField>
            </children>
        </VBox>
        <Button defaultButton="true" layoutX="251.0" layoutY="389.0"
mnemonicParsing="false" onAction="#register" prefHeight="33.0" prefWidth="199.0" style="-
fx-background-color: #c2d6d6;" text="Register">
            <cursor>
               <Cursor fx:constant="HAND" />
            </cursor>
            <font>
               <Font size="18.0" />
            </font>
        </Button>
        <Text layoutX="164.0" layoutY="74.0" strokeType="OUTSIDE" strokeWidth="0.0"
text="Please fill the form to register">
            <font>
               <Font name="Calibri" size="25.0" />
            </font>
        </Text>
      <ImageView fitHeight="104.0" fitWidth="116.0" layoutX="22.0" layoutY="370.0"
pickOnBounds="true" preserveRatio="true">
         <image>
            <Image url="@image/register.png" />
         </image>
      </ImageView>
   </children>
</AnchorPane>
```

**k. ControllerRegister.java**

```
package sample;

import java.io.IOException;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.List;
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import Serverpkg.Server;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.layout.AnchorPane;
import javafx.scene.control.Alert;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.control.Alert.AlertType;


public class ControllerRegister implements Controller {

   private User currentUser;

   @FXML
   private AnchorPane rootPane;

   @FXML
```

```
private TextField name;

@FXML
private TextField surname;

@FXML
private TextField email;

@FXML
private PasswordField password;


public void initData(User user) {
    this.currentUser = user;
}


public Boolean isMail(String mail) {
    String mailRegex = "\\w+@\\w+\\.\\w+";
    Pattern mailValidator = Pattern.compile(mailRegex);
    Matcher mailMatcher = mailValidator.matcher(mail);
    return mailMatcher.matches();
}


@FXML
public void register(ActionEvent event) throws UnknownHostException, IOException {

    // gets data from the page
    String nam = name.getText();
    String sur = surname.getText();
    String mail = email.getText();
    String pass = password.getText();

    if (nam.length() == 0 || sur.length() == 0 || mail.length() == 0 || pass.length() == 0) {
        // all fields are required
        Alert alert = new Alert(AlertType.WARNING);
        alert.setTitle("All fields must be filled");
        alert.setHeaderText("Please fill all the fields");
        alert.showAndWait();
    } else if (!isMail(mail)) {
        // checks if the email is valid
        Alert alert = new Alert(AlertType.WARNING);
        alert.setTitle("Email not valid");
        alert.setHeaderText("The provided email is not valid, please retry.");
        alert.showAndWait();
    } else {
        List<String> request = new ArrayList<>();
        request.add("register_user");
        request.add(nam);
        request.add(sur);
        request.add(mail);
        request.add(pass);
        ArrayList<User> response = Server.run(request);
```

```
            this.currentUser = response.get(0);
            // loads the homepage
            try {
                Loader loader = new Loader(this.currentUser, this.rootPane);
                loader.load("homepage_user");
            } catch (Exception e) {
                System.out.println(e);
                Alert alert = new Alert(AlertType.ERROR);
                alert.setTitle("Page Missing");
                alert.setHeaderText("Page is unreachable. Try again later");
                alert.showAndWait();
            }
        }
    }
}
```

**l. Equipment.java**

```
package sample;


import javafx.scene.image.Image;

import java.io.Serializable;

public class Equipment implements Serializable {

    private String equipmentName;
    private String equipmentId;
    private String owner;
    private String contact;
    private int price;


    public Equipment() {
        this.equipmentName="";
        this.equipmentId="";
        this.contact="";
        this.owner="";
        this.price=0;
    }

    public Equipment(String equipmentName,String equipmentId,String owner,String contact,int
price) {
        this.equipmentName = equipmentName;
        this.equipmentId = equipmentId;
        this.owner = owner;
        this.contact = contact;
        this.price = price;
    }


    public String getEquipmentName() {
        return equipmentName;
    }

    public String getEquipmentId() {
        return equipmentId;
    }

    public String getOwner() {
        return owner;
    }

    public String getContact() {
        return contact;
    }

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

**m. Loader.java**

```
package sample;

import java.io.IOException;
import javafx.fxml.FXMLLoader;
import javafx.scene.layout.AnchorPane;



public class Loader {

    private AnchorPane rootPane;
    private User currentUser;

    public Loader(User currentUser, AnchorPane rootPane) {
        this.rootPane = rootPane;
        this.currentUser = currentUser;
    }

    public void load(String filename) throws IOException {
        FXMLLoader loader = new FXMLLoader();
        loader.setLocation(getClass().getResource(filename + ".fxml"));
        AnchorPane parent = loader.load();

        Controller controller = loader.getController();
        controller.initData(this.currentUser);
        this.rootPane.getChildren().setAll(parent);

    }
}
```

**n. User.java**

```
package sample;

import java.io.Serializable;

public class User implements Serializable {

    private String name;
    private String surname;
    private String email;
    private String password;
    private int permission;

    public User() {
        this.name = "";
        this.surname = "";
        this.email = "";
        this.password = "";
        this.permission = 0;
    }

    public User(final String name, final String sur, final String email, final String password ,final
int perm) {
        this.name = name;
        this.surname = sur;
        this.email = email;
        this.password = password;
        this.permission = perm;
    }

    public String getName() {
        return this.name;
    }

    public String getSurname() {
        return this.surname;
    }


    public String getEmail() {
        return this.email;
    }

    public int getPermission() {
        return this.permission;
    }

    public String getPassword() {
        return this.password;
    }
}
```

**o. add_employee.fxml**

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>

<AnchorPane fx:id="rootPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-
Infinity" minWidth="-Infinity" prefHeight="500.0" prefWidth="700.0" style="-fx-background-
color: #669999;" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sample.ControllerAddEmployee">
   <children>
      <VBox alignment="CENTER" layoutX="142.0" layoutY="159.0" prefHeight="219.0"
prefWidth="410.0" spacing="15.0">
         <children>
            <TextField fx:id="name" prefHeight="35.0" prefWidth="410.0"
promptText="Name">
               <font>
                  <Font size="18.0" />
               </font></TextField>
            <TextField fx:id="surname" prefHeight="35.0" prefWidth="410.0"
promptText="Surname">
               <font>
                  <Font size="18.0" />
               </font></TextField>
            <TextField fx:id="email" prefHeight="35.0" prefWidth="410.0"
promptText="Email">
               <font>
                  <Font size="18.0" />
               </font></TextField>
            <PasswordField fx:id="password" prefHeight="35.0" prefWidth="410.0"
promptText="Password">
               <font>
                  <Font size="18.0" />
               </font></PasswordField>
         </children>
      </VBox>
      <Text fill="WHITE" layoutX="57.0" layoutY="112.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Please fill the form to register a new employee">
         <font>
            <Font name="Calibri" size="25.0" />
         </font>
      </Text>
      <Button layoutX="14.0" layoutY="14.0" mnemonicParsing="false" onAction="#back"
prefHeight="34.0" prefWidth="107.0" text="&lt; Back">
         <font>
            <Font name="Calibri" size="14.0" />
```

```
          </font></Button>
         <Button defaultButton="true" layoutX="257.0" layoutY="400.0"
mnemonicParsing="false" onAction="#addEmployee" prefHeight="46.0" prefWidth="181.0"
style="-fx-background-color: white;" text="Add">
            <font>
               <Font size="25.0" />
            </font>
      </Button>
   </children>
</AnchorPane>
```

**p. add_equipment.fxml**

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>

<AnchorPane fx:id="rootPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-
Infinity" minWidth="-Infinity" prefHeight="500.0" prefWidth="700.0" style="-fx-background-
color: #9933ff;" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sample.ControllerAddEquipment">
   <children>
      <VBox alignment="CENTER" layoutX="161.0" layoutY="132.0" prefHeight="236.0"
prefWidth="379.0" spacing="15.0">
         <children>
            <TextField fx:id="equipmentName" promptText="Equipment Name">
               <font>
                  <Font size="18.0" />
               </font></TextField>
            <TextField fx:id="ownerName" promptText="Owner Name">
               <font>
                  <Font size="18.0" />
               </font></TextField>
            <TextField fx:id="contact" promptText="Contact">
               <font>
                  <Font size="18.0" />
               </font></TextField>
            <TextField fx:id="price" promptText="Price">
               <font>
                  <Font size="18.0" />
               </font></TextField>
         </children>
      </VBox>
      <Button layoutX="14.0" layoutY="14.0" mnemonicParsing="false" onAction="#back"
prefHeight="34.0" prefWidth="95.0" text="&lt; Back">
         <font>
            <Font name="Calibri" size="14.0" />
```

```
        </font></Button>
       <Button layoutX="260.0" layoutY="380.0" mnemonicParsing="false"
onAction="#addEquipment" prefHeight="39.0" prefWidth="180.0" text="Add">
          <font>
             <Font size="20.0" />
          </font>
    </Button>
      <Text fill="WHITE" layoutX="76.0" layoutY="108.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Please fill the form to add a new equipment">
         <font>
            <Font name="Calibri" size="25.0" />
         </font>
      </Text>
    <ImageView fitHeight="107.0" fitWidth="95.0" layoutX="22.0" layoutY="379.0"
pickOnBounds="true" preserveRatio="true">
        <image>
          <Image url="@image/register.png" />
        </image>
    </ImageView>
  </children>
</AnchorPane>
```

**q. cart.fxml**

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>

<AnchorPane fx:id="rootPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-
Infinity" minWidth="-Infinity" prefHeight="500.0" prefWidth="700.0" style="-fx-background-
color: #5500ff;" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sample.ControllerCart">
   <children>
      <Button layoutX="20.0" layoutY="14.0" mnemonicParsing="false" onAction="#back"
prefHeight="34.0" prefWidth="74.0" text="&lt; Back">
         <font>
            <Font name="Calibri" size="14.0" />
         </font></Button>
      <TableView fx:id="tableView" layoutX="64.0" layoutY="65.0" prefHeight="334.0"
prefWidth="585.0">
         <columns>
            <TableColumn fx:id="nameColumn" editable="false" prefWidth="75.0"
sortable="false" text="Equipment" />
            <TableColumn fx:id="ownerColumn" editable="false" prefWidth="75.0"
sortable="false" text="Owner" />
            <TableColumn fx:id="contactColumn" editable="false" prefWidth="75.0"
sortable="false" text="Contact" />
```

```
            <TableColumn fx:id="priceColumn" editable="false" prefWidth="75.0"
sortable="false" text="Price" />
         </columns>
         <columnResizePolicy>
            <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
         </columnResizePolicy>
      </TableView>
       <Button layoutX="165.0" layoutY="426.0" mnemonicParsing="false"
onAction="#removeFromCart" prefHeight="30.0" prefWidth="170.0" text="Remove from
Cart🛒">
          <font>
             <Font name="Calibri" size="14.0" />
          </font>
    </Button>
       <Button layoutX="385.0" layoutY="426.0" lineSpacing="20.0" mnemonicParsing="false"
onAction="#buy" prefHeight="30.0" prefWidth="170.0" text="Buy $ ">
          <font>
             <Font name="Calibri" size="14.0" />
          </font>
    </Button>
    <ImageView fitHeight="57.0" fitWidth="88.0" layoutX="14.0" layoutY="417.0"
pickOnBounds="true" preserveRatio="true">
       <image>
          <Image url="@image/logo.png" />
       </image>
    </ImageView>
   </children>
</AnchorPane>
```

**r. ControllerAddEquipment.java**

```
package sample;
import java.io.IOException;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.List;

import Serverpkg.ServerEquipments;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.TextField;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.AnchorPane;

public class ControllerAddEquipment implements Controller{

    private User currentUser;

    @FXML
    private AnchorPane rootPane;

    @FXML
    private TextField equipmentName;
    @FXML
    private TextField ownerName;

    @FXML
    private TextField contact;

    @FXML
    private TextField price;

    public void initData(User user) {
        this.currentUser = user;
    }


   @FXML
    public void addEquipment(ActionEvent event) throws UnknownHostException, IOException
{
        int amount=0;
        String nam = equipmentName.getText();
        String own = ownerName.getText();
        String con = contact.getText();
        String pri = price.getText();


        if (nam.length() == 0 || own.length() == 0 || con.length() == 0 || pri.length() == 0) {
            Alert alert = new Alert(AlertType.WARNING);
            alert.setTitle("All fields must be filled");
            alert.setHeaderText("Please fill all the fields");
            alert.showAndWait();
        } else {
            try {
                amount = Integer.parseInt(pri);
            } catch (NumberFormatException e) {
                Alert alert = new Alert(AlertType.ERROR);
                alert.setTitle("Invalid Price");
                alert.setHeaderText("Please insert a valid price.");
                alert.showAndWait();
            } finally {
                if (this.currentUser.getPermission() > 1) {
                   List<String> request = new ArrayList<>();
                   request.add("add_equipment");
                   request.add(nam);
                   request.add(own);
                   request.add(con);
                   request.add(pri);
                   ArrayList<Equipment> equipments = new ArrayList<>();


                    try {
                        equipments = ServerEquipments.run(request);
                        Alert alert = new Alert(AlertType.INFORMATION);
                        if (equipments.size() !=0) {
                            alert.setTitle("Equipment added!");
                    alert.setHeaderText(
                            String.format("Equipment %s by %s has been added.\nID: %s",
                                    equipments.get(0).getEquipmentName(),
equipments.get(0).getOwner(),
                                    equipments.get(0).getEquipmentId()));
                } else {
                    alert.setTitle("Equipment already inserted");
                    alert.setHeaderText("Equipment is already present in the database.");
                }
                alert.showAndWait();
            } catch (Exception e) {
                System.out.println(e);;
            }
        } else {
            // user is not authorized to perform the action
            Alert alert = new Alert(AlertType.ERROR);
            alert.setTitle("Not authorized");
            alert.setHeaderText("You are not allowed to perform this action.");
            alert.showAndWait();
        }
    }
  }
}

@FXML
public void back(ActionEvent event) throws IOException {
    Loader loader = new Loader(this.currentUser, this.rootPane);
    loader.load("homepage_employee");
}

}
```

**s. ControllerAddEmployee.java**

```
package sample;
import Serverpkg.Server;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.AnchorPane;

import java.util.ArrayList;
import java.util.List;
import java.util.regex.Pattern;
import java.io.IOException;
import java.net.UnknownHostException;
import java.util.regex.Matcher;

public class ControllerAddEmployee implements Controller {

    private User currentUser;
    @FXML
    private AnchorPane rootPane;

    @FXML
    private TextField name;

    @FXML
    private TextField surname;

    @FXML
    private TextField email;

    @FXML
    private PasswordField password;

    public void initData(User user) {
        this.currentUser = user;
    }

    public Boolean isMail(String mail) {
        String mailRegex = "\\w+@\\w+\\.\\w+";
        Pattern mailValidator = Pattern.compile(mailRegex);
        Matcher mailMatcher = mailValidator.matcher(mail);
        return mailMatcher.matches();
    }

    @FXML
    public void addEmployee(ActionEvent event) throws UnknownHostException, IOException
{
        // gets data
        String nam = name.getText();
        String sur = surname.getText();
        String mail = email.getText();
        String pass = password.getText();

        if (nam.length() == 0 || sur.length() == 0 || mail.length() == 0 || pass.length() == 0) {
            // all fields are required
            Alert alert = new Alert(AlertType.WARNING);
            alert.setTitle("All fields must be filled");
            alert.setHeaderText("Please fill all the fields.");
            alert.showAndWait();
        } else if (!isMail(mail)) {
            // email is not valid
            Alert alert = new Alert(AlertType.WARNING);
            alert.setTitle("Email not valid");
            alert.setHeaderText("The provided email is not valid, please retry.");
            alert.showAndWait();
        } else {
            if (this.currentUser.getPermission() > 2) {
                // user is authorized to perform the action
                List<String> request = new ArrayList<>();
                request.add("register_employee");
                request.add(nam);
                request.add(sur);
                request.add(mail);
                request.add(pass);
                ArrayList<User> response = Server.run(request);

                try {
                    User newEmployee = response.get(0);
                    int permission = newEmployee.getPermission();

                    if (permission < 1) {
                        // permission < 1 = nullUser => user is already registered.
                        Alert alert = new Alert(AlertType.WARNING);
                        alert.setTitle("Already registered");
                        alert.setHeaderText("There is already an account with this email.");
                        alert.showAndWait();
                    } else {
                        // employee registered successfully
                        Alert alert = new Alert(AlertType.INFORMATION);
                        alert.setTitle("Success");
                        alert.setHeaderText("Employee registered successfully.");
                        alert.showAndWait();
                    }
                } catch (Exception e) {
                    System.out.println(e);;
                }
            } else {
                // user is not authorized to perform the action
                Alert alert = new Alert(AlertType.ERROR);
                alert.setTitle("Not authorized");
                alert.setHeaderText("You are not allowed to perform this action.");
                alert.showAndWait();
            }

            Loader loader = new Loader(this.currentUser, this.rootPane);
            loader.load("homepage_admin");
        }
    }

    @FXML
    public void back(ActionEvent event) throws IOException {
        Loader loader = new Loader(this.currentUser, this.rootPane);
        loader.load("homepage_admin");
    }
}
```

**t. ControllerCart.java**

```
package sample;

import java.io.IOException;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.List;
import Serverpkg.ServerEquipments;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.AnchorPane;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;

class NotSelectedException extends Exception{
    private String msg;
    public NotSelectedException(String msg){
        this.msg = msg;
    }

    public String toString(){
        return msg;
    }
}

public class ControllerCart implements Controller {

    private User currentUser;

    @FXML
    private AnchorPane rootPane;

    @FXML
    private TableView<Equipment> tableView;

    @FXML
    private TableColumn<Equipment, String> nameColumn;

    @FXML
    private TableColumn<Equipment, String> ownerColumn;

    @FXML
    private TableColumn<Equipment, String> contactColumn;

    @FXML
    private TableColumn<Equipment, Integer> priceColumn;


    public void initData(User user) {
        this.currentUser = user;

        try {
            ArrayList<Equipment> equipments = new ArrayList<>();
            List<String> request = new ArrayList<>();
            request.add("display_cart");
            request.add(this.currentUser.getEmail());
            equipments = ServerEquipments.run(request);
            addToTable(equipments);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }


    public void addToTable(ArrayList<Equipment> equipments) {
        // set up the columns in the table
        this.nameColumn.setCellValueFactory(new PropertyValueFactory<Equipment, 
String>("equipmentName"));
        this.ownerColumn.setCellValueFactory(new PropertyValueFactory<Equipment, 
String>("owner"));
        this.contactColumn.setCellValueFactory(new PropertyValueFactory<Equipment, 
String>("contact"));
        this.priceColumn.setCellValueFactory(new 
PropertyValueFactory<Equipment,Integer>("price"));
        ObservableList<Equipment> oListEquipment = 
FXCollections.observableArrayList(equipments);
        // load data
        tableView.setItems(oListEquipment);
    }


    @FXML
    public void back(ActionEvent event) throws IOException {
        Loader loader = new Loader(this.currentUser, this.rootPane);
        loader.load("homepage_user");
    }


    @FXML
    public void buy(ActionEvent event) throws IOException, UnknownHostException {
        if (this.currentUser.getPermission() > 0) {

            try {
                List<String> request = new ArrayList<>();
                request.add("display_cart");
                request.add(this.currentUser.getEmail());
                ArrayList<Equipment> equipments = ServerEquipments.run(request);
                request.clear();
                request.add(this.currentUser.getEmail());
                Boolean response = ServerEquipments.orderRun(request,equipments);
                if (!response) {
                    Alert alert = new Alert(AlertType.WARNING);
                    alert.setTitle("Order failed!");
                    alert.showAndWait();
                } else {
                    request.clear();
                    request.add("remove_all_from_cart");
                    request.add(this.currentUser.getEmail());
                    response = ServerEquipments.run(request, new Equipment());
                    request.clear();
                request.add("display_cart");
                request.add(this.currentUser.getEmail());
                equipments = ServerEquipments.run(request);
                addToTable(equipments);

                Alert alert = new Alert(AlertType.INFORMATION);
                alert.setTitle("Order submitted!");
                alert.setHeaderText(String.format("Your order has been placed!"));
                alert.showAndWait();
              }
            } catch (Exception e) {
              System.out.println(e);
            }
        } else {
            // user is not authorized to perform the action
            Alert alert = new Alert(AlertType.ERROR);
            alert.setTitle("Not authorized");
            alert.setHeaderText("You are not allowed to perform this action.");
            alert.showAndWait();
        }
    }


    @FXML
    @SuppressWarnings("unchecked")
    public void displayCart(ActionEvent event) throws IOException {
        try {
            ArrayList<Equipment> equipments = new ArrayList<>();
            List<String> request = new ArrayList<>();
            request.add("display_cart");
            equipments = ServerEquipments.run(request);

            addToTable(equipments);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @FXML
    public void removeFromCart(ActionEvent event) throws UnknownHostException,
IOException {

        try {
            // getting selection of the tableview
            Equipment equipment = tableView.getSelectionModel().getSelectedItem();

            if(equipment==null)
              throw new NotSelectedException("Didn't choose an equipment");
            Boolean response= false;
            List<String> request = new ArrayList<>();
            request.add("remove_from_cart");
            request.add(this.currentUser.getEmail());
            response = ServerEquipments.run(request,equipment);
            if (response) {
                initData(this.currentUser);
                Alert alert = new Alert(AlertType.INFORMATION);
                alert.setTitle(String.format("Removed from cart"));
                alert.setHeaderText(String.format("Removed %s from cart.",
equipment.getEquipmentId()));
                alert.showAndWait();
            }
        } catch(NotSelectedException e) {
            System.out.println(e);
            Alert alert = new Alert(AlertType.WARNING);
            alert.setTitle(String.format("Select an Equipment"));
            alert.setHeaderText("You have to click on the equipment and then Remove.");
            alert.showAndWait();
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

**u. Controller.java (Interface)**

```
package sample;

public interface Controller {
    void initData(User user);

}
```

**v. cartDB.java**

```
package Serverpkg;

import sample.Equipment;

import java.util.ArrayList;

public class CartDB {
    public static ArrayList<String> Email = new ArrayList<>();
    public static ArrayList<ArrayList<Equipment>> cartEquipments = new ArrayList<>();

    public static Boolean addtoCart(String email, Equipment equipment){
        try{
            for(int i=0;i<Email.size();i++){
                if(Email.get(i).equals(email)){
                    for(int j=0;j<cartEquipments.get(i).size();j++){
                        if(cartEquipments.get(i).get(j).equals(equipment)){
                            return true;
                        }
                    }
                    cartEquipments.get(i).add(equipment);
                    return true;
                }
            }
        }catch(NullPointerException e){
            System.out.println(e);
        }catch(Exception e){
            System.out.println(e);
        }
        Email.add(email);
        ArrayList<Equipment> equipments = new ArrayList<>();
        equipments.add(equipment);
        cartEquipments.add(equipments);
        return true;
    }

    public static Boolean removefromCart(String email, Equipment equipment){
        int i=-1,j=-1,index=-1;
        for(i=0;i<Email.size();i++){
            if(Email.get(i).equals(email)){
                for(j=0;j<cartEquipments.get(i).size();j++){

if(cartEquipments.get(i).get(j).getEquipmentId().equals(equipment.getEquipmentId())){
                            break;
                        }
                    }
                cartEquipments.get(i).remove(j);
                break;
                }
            }

        return true;
    }

    public static Boolean removefromCart(String email){
        int i=-1;
        for(i=0;i<Email.size();i++){
            if(Email.get(i).equals(email))
                break;
        }
        if(i!=Email.size()){
            Email.remove(i);
            cartEquipments.remove(i);
            return true;
        }
        return false;
    }


    public static ArrayList<Equipment> displayCart(String mail){
        ArrayList<Equipment> equipments = new ArrayList<>();
        for(int i=0;i<Email.size();i++){
            if(Email.get(i).equals(mail)){
                equipments =  cartEquipments.get(i);
            }
        }
        return equipments;
    }
}
```

**w. equipmentDB.java**

```
package Serverpkg;

import sample.Equipment;
import sample.User;

import java.util.ArrayList;
import java.util.List;

public class EquipmentDB {
    private static List<String> EquipmentName = new ArrayList<>();
    private static List<String> EquipmentId = new ArrayList<>();
    private static List<String> Owner = new ArrayList<>();
    private static List<String> Contact = new ArrayList<>();
    private static List<Integer> Price = new ArrayList<>();
    private static int counter;

    static{
        counter = 1000;
    }


    public static void init(){

    }

    public static Equipment addEquipment(String name,String owner,String contact,int price){

        EquipmentName.add(name);
        EquipmentId.add("E" + counter++);
        Owner.add(owner);
        Contact.add(contact);
        Price.add(price);
        int i = EquipmentName.size()-1;
        return new
Equipment(EquipmentName.get(i),EquipmentId.get(i),Owner.get(i),Contact.get(i),Price.get(i));
    }

    public static ArrayList<Equipment> getEquipments(){
        EquipmentDB.addEquipment("Harvester","Sunil","9999999999",35000);
        EquipmentDB.addEquipment("Tractor","Vishal","8888888888",25750);
        EquipmentDB.addEquipment("Reaper","Vijay","7777777777",15650);
        EquipmentDB.addEquipment("Harvester","Sham","6666666666",31000);
        EquipmentDB.addEquipment("Trolley","Hari","5555555555",34220);
        EquipmentDB.addEquipment("Cultivator","Rahul","4444444444",12130);
        ArrayList<Equipment> equipmentList = new ArrayList<>();
        for(int i=0;i<EquipmentName.size();i++){
            equipmentList.add(new 
Equipment(EquipmentName.get(i),EquipmentId.get(i),Owner.get(i),Contact.get(i),
                    Price.get(i)));
        }
        System.out.println(equipmentList);
        return equipmentList;
    }

    public static ArrayList<Equipment> search(String equipmentType){
        ArrayList<Equipment> equipments = new ArrayList<>();
        for(int i=0;i<EquipmentName.size();i++){
            System.out.println(EquipmentName.get(i));
            if(EquipmentName.get(i).equalsIgnoreCase(equipmentType))
            {
                equipments.add(new 
Equipment(EquipmentName.get(i),EquipmentId.get(i),Owner.get(i),Contact.get(i),
                        Price.get(i)));
            }
        }
        return equipments;
    }
}
```

**x. order.java**

```
package Serverpkg;

import sample.Equipment;

import java.lang.reflect.Array;
import java.util.ArrayList;

public class Order {
    private String email;
    private ArrayList<String> orderId = new ArrayList<>();
    private ArrayList<Equipment> orderedEquipments = new ArrayList<>();
    private ArrayList<Boolean> shipped = new ArrayList<>();
    private static int counter;
    static{
        counter=100;
    }

    public Order(){

    }

    public Order(String email,ArrayList<Equipment> equipments){
        this.email = email;
        for(int i=0;i<equipments.size();i++){
            this.orderId.add("O-"+counter++);
            this.orderedEquipments.add(equipments.get(i));
            this.shipped.add(false);
        }
    }
    public void addOrder(ArrayList<Equipment> equipments){
        for(int i=0;i<equipments.size();i++){
            this.orderId.add("O-" + counter++);
            this.orderedEquipments.add(equipments.get(i));
            this.shipped.add(false);
        }
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }


    public ArrayList<Equipment> getOrderedEquipments() {
        return orderedEquipments;
    }

    public ArrayList<String> getOrderId(){
        return orderId;
    }

    public Boolean setOrdershipped(){
        for(int i=0;i<shipped.size();i++){
            shipped.set(i,true);
            //shipped.remove(i+1);
        }
        return true;
    }
    public ArrayList<Boolean> isShipped(){
        return shipped;
    }

}
```

**y. orderDB.java**

```
package Serverpkg;

import sample.Equipment;

import java.util.ArrayList;

public class OrderDB {
    private static ArrayList<Order> orders = new ArrayList<>();



    public static Boolean addOrder(String email, ArrayList<Equipment> equipments){
        for(int i=0;i<orders.size();i++){
            if(orders.get(i).getEmail().equals(email)){
                orders.get(i).addOrder(equipments);
                return true;
            }
        }
        Order order = new Order(email,equipments);
        orders.add(order);
        return true;
    }

    public static Order getOrders(String email){
        for(int i=0;i<orders.size();i++){
            if(orders.get(i).getEmail().equals(email)){
                return orders.get(i);
            }
        }
        return new Order();
    }

    public static ArrayList<Order> getOrders(){
        return orders;
    }
    public static ArrayList<Order> getOrdersEmployees(){
        ArrayList<Order> ord = new ArrayList<>();
        for(int i=0;i<orders.size();i++){
            if(!orders.get(i).isShipped().get(i)){
                ord.add(orders.get(i));
            }
        }
        return ord;
    }
    public static Boolean shipOrder(String email){
        Boolean response = false;
        for(int i=0;i<orders.size();i++){
            if(orders.get(i).getEmail().equals(email)){
                response = orders.get(i).setOrdershipped();
            }
        }
        return response;
    }


}
```

**y. Server.java**

```
package Serverpkg;

import sample.User;


import java.util.ArrayList;
import java.util.List;
```

```
public class Server {
    static int init =0;
    public static ArrayList<User> run(List<String> request){
        if(init==0){
            UserDB.init();
            init++;
        }
        User user = new User();
        ArrayList<User> response = new ArrayList<>();
        try{
            switch(request.get(0)){
                case "login":
                    user = login(request.get(1),request.get(2));
                    break;
                case "guest":
                    user = new User();
                    break;
                case "register_user":
                    user = register(request.get(1),request.get(2),request.get(3),request.get(4),1);
                    break;

                case "register_employee":
                    user = register(request.get(1),request.get(2),request.get(3),request.get(4), 2);
                    break;

                case "get_employees":
                    ArrayList<User> employees = UserDB.getUsers(2);
                    return employees;

                case "get_users":
                    ArrayList<User> users = UserDB.getUsers(1);
                    return users;


                default:
                    break;
            }
            response.add(user);
            return response;
        }catch(Exception e){
            e.printStackTrace();
        }
        response.add(user);
        return response;
    }

    public static User login(String email, String password) {
        User nullUser = new User();
        try {
            nullUser = UserDB.checkUser(email,password);
            return nullUser;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return nullUser;
    }

    public static User register(String name, String surname, String mail, String password, int
permission) {
        User nullUser = new User();

        try {
            nullUser = UserDB.checkUser(mail,password);
            if(nullUser.getName().equals(name)){
                //already present
                return new User();
            }
            UserDB.addUser(name,surname,mail,password,permission);

            User newUser = new User(name, surname, mail, password, permission);
            // registration successful, the User object is returned
            return newUser;

        } catch (Exception e) {
            e.printStackTrace();
        }
        // registration not successful, the nullUser object is returned
        return nullUser;
    }
}
```

**z. ServerEquipment.java**

```
package Serverpkg;

import sample.Equipment;

import java.util.ArrayList;
import java.util.List;

public class ServerEquipments {


    public static ArrayList<Equipment> run(List<String> request) {
        EquipmentDB.init();
        ArrayList<Equipment> response = new ArrayList<>();
        try {
            switch (request.get(0)) {

                case "add_equipment":
                    Equipment equipment = EquipmentDB.addEquipment(request.get(1),
request.get(2), request.get(3),
                        Integer.parseInt(request.get(4)));
                    response.add(equipment);
                    return response;

                case "search":
                    response = EquipmentDB.search(request.get(1));
                    return response;

                case "get_equipments":
                    response = EquipmentDB.getEquipments();
                    return response;

                case "display_cart":
                    response = CartDB.displayCart(request.get(1));
                    return response;


                default:
                    break;
            }
            return response;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return response;
    }

    public static Boolean run(List<String> request,Equipment equipment) {
        EquipmentDB.init();
         boolean response = false;
        try {
            switch (request.get(0)) {

                case "add_to_cart":
                    response = CartDB.addtoCart(request.get(1),equipment);
                    return response;

                case "remove_from_cart":
                    response = CartDB.removefromCart(request.get(1),equipment);
                    return response;

                case "remove_all_from_cart":
                    response = CartDB.removefromCart(request.get(1));
                    return response;

                case "shipOrder":
                    response = OrderDB.shipOrder(request.get(1));
                    return response;

                default:
                    break;
            }
            return response;
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
        return response;
    }

    public static ArrayList<Order> orderRun(List<String> request) {
        ArrayList<Order> response = new ArrayList<>();
        try {
            switch (request.get(0)) {

                case "get_order_user":
                    Order order = OrderDB.getOrders(request.get(1));
                    response.add(order);
                    return response;

                case "get_orders":
                    response = OrderDB.getOrders();
                    return response;
                case "get_orders_employee":
                    response = OrderDB.getOrdersEmployees();
                    return response;

                default:
                    break;
            }
            return response;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return response;
    }

    public static Boolean orderRun(List<String> request,ArrayList<Equipment> equipments) {
        return OrderDB.addOrder(request.get(0),equipments);

    }
}
```

**A. UserDB.java**

```
package Serverpkg;

import sample.User;

import java.util.ArrayList;
import java.util.List;

public class UserDB {
    private static List<String> Name = new ArrayList<>();
    private static List<String> Sur = new ArrayList<>();
    private static List<String> Email = new ArrayList<>();
    private static List<String> Password = new ArrayList<>();
    private static List<Integer> Perm = new ArrayList<>();
    public static User checkUser(String mail, String pass){
        User user = new User();

        for(int i=0;i<Name.size();i++){
            if(Email.get(i).equals(mail) && Password.get(i).equals(pass)){
                user = new User(Name.get(i),Sur.get(i),Email.get(i),Password.get(i),Perm.get(i));
                return user;
            }
        }
        return user;
    }

    public static void addUser(final String name,final String sur,final String mail,final String pass,
                               final int perm){
        Name.add(name);
        Sur.add(sur);
        Email.add(mail);
        Password.add(pass);
        Perm.add(perm);
    }

    public static ArrayList<User> getUsers(int permission){
        ArrayList<User> userList = new ArrayList<>();
        for(int i=0;i<Name.size();i++){
            if(Perm.get(i)==permission){
                userList.add(new User(Name.get(i),Sur.get(i),Email.get(i),"",Perm.get(i)));
            }
        }
        return userList;
    }

    private static int initialization;
    static{
        initialization=100;
    }
    public static void init(){
        if(initialization==100){
            Name.add("Admin");
            Sur.add("Admin");
            Email.add("admin@a.c");
            Password.add("a");
            Perm.add(3);
            UserDB.addUser("Rahul","Pawar","rahul@gmail.com","12345",2);
            UserDB.addUser("Cierra","Vega","ciera@gmail.com","12345",2);
            UserDB.addUser("Brock","Lesnar","brock@gmail.com","12345",2);
            UserDB.addUser("Samoan","joe","samoan@gmail.com","12345",1);
            UserDB.addUser("Ramesh","Thakur","ramesh@gmail.com","12345",1);
            UserDB.addUser("Vinay","Kumar","vinay@gmail.com","12345",1);
        }
        initialization++;
    }
}
```