Code Snippets

Main Class-

```
package sample;

import com.mongodb.DB;
import javafx.animation.PauseTransition;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.util.Duration;
import java.util.Timer;
import java.util.concurrent.TimeUnit;
import com.mongodb.MongoClient;

public class Main extends Application {
    public static Stage Primarystage;
    public static Stage splashStage=new Stage();


    @Override
    public void start(Stage primaryStage) throws Exception{
        Primarystage=primaryStage;
        Parent root = FXMLLoader.load(getClass().getResource("sample.fxml")) ;
        primaryStage.setTitle("Inventory Control");
        primaryStage.setScene(new Scene(root, 956, 638));
        primaryStage.show();

    }


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

Controller for the start page-

```
package sample;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;

import java.awt.*;

public class Controller {
    @FXML
    Button btnNewAcc;

    @FXML
    Button btnLogin;

    public static Stage createAccStage=new Stage();
    public static Stage loginStage=new Stage();

    @FXML
    public void accCreate() throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("createAcc.fxml"));
        createAccStage.setTitle("Create Account");
        createAccStage.setScene(new Scene(root, 956, 638));
        createAccStage.show();
        Main.Primarystage.close();
    }

    @FXML
    public void signUp() throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("signUp.fxml"));
        loginStage.setTitle("Log into your account");
        loginStage.setScene(new Scene(root, 956, 638));
        loginStage.show();
        Main.Primarystage.close();
    }


}
```

Controller for create account window –

```
package sample;

import com.mongodb.BasicDBObject;
import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
import com.mongodb.client.FindIterable;
import com.sun.org.apache.xpath.internal.operations.And;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.stage.Stage;
import org.bson.Document;



public class CreateAcc {

    @FXML
    Button btnBack;

    @FXML
    Button btnCreate;

    @FXML
    TextField txtUsername;

    @FXML
    TextField txtEmail;

    @FXML
    PasswordField txtPassword;

    @FXML
    PasswordField txtConfirmPassword;

    public String username;
    public String password;
    public String email;
    public String confirmPassword;

    public boolean found=false;
    public boolean spaceFound;
    public boolean foundUsername;


    @FXML
    public void proceedWithCreation() throws Exception {
        foundUsername=false;
        username = txtUsername.getText();
        email = txtEmail.getText();
        password = txtPassword.getText();
        confirmPassword = txtConfirmPassword.getText();
        if (!password.equals(confirmPassword)) {
            Alert wrongPassword = new Alert(Alert.AlertType.NONE);
            wrongPassword.setAlertType(Alert.AlertType.WARNING);
            wrongPassword.setContentText("Passwords do not match. Please Re-enter");
            wrongPassword.showAndWait();
        } else {
            if (!username.equals("") && !password.equals("") && !email.equals("")) {
                try {
                    DBSetup.init();
                    DBCollection usernameCheck = DBSetup.database.getCollection("LoginDetails");
                    DBCursor findIterable=usernameCheck.find();
                    for (DBObject count:findIterable) {
                        if( username.equals(count.get("Username"))) {
                            foundUsername = true;
                        }

                    }
                    if (foundUsername) {
                            Alert usernameExists = new Alert(Alert.AlertType.NONE);
                            usernameExists.setAlertType(Alert.AlertType.WARNING);
                            usernameExists.setContentText("Username already exists. Please enter another username");
                            usernameExists.showAndWait();

                    } else {
                        BasicDBObject basicDBObject1 = new BasicDBObject();
                        basicDBObject1.put("Username", username);
                        basicDBObject1.put("Password", password);
                        DBSetup.init();
                        DBCollection collection = DBSetup.database.getCollection("LoginDetails");
                        collection.insert(basicDBObject1);
                        Alert a = new Alert(Alert.AlertType.NONE);
                        a.setAlertType(Alert.AlertType.INFORMATION);
                        a.setContentText("Account created successfully. You may now login");
                        a.showAndWait().ifPresent(response -> {
                            if (response == ButtonType.OK) {
                                Controller.createAccStage.close();
                                Main.Primarystage.show();
                            }
                        });

                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            } else {
                Alert emptyDetails = new Alert(Alert.AlertType.NONE);
                emptyDetails.setAlertType(Alert.AlertType.WARNING);
                emptyDetails.setContentText("All details have not been entered. Please fill them");
                emptyDetails.showAndWait();
            }
        }

    }
    @FXML
    public void goBacktoMain() throws Exception{
        Controller.createAccStage.close();
        Main.Primarystage.show();
    }
}
```

Controller for login window –

```
package sample;

import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.stage.Stage;

public class SignUp {

    @FXML
    Button btnBack;

    @FXML
    Button btnLogin;

    @FXML
    TextField txtUsername;

    @FXML
    PasswordField txtPassword;

    public static String Username;
    public static String Password;
    public static boolean LoginDetailsFound;
    public static Stage HomePage =new Stage();


    @FXML
    public void login() throws Exception{
            LoginDetailsFound=false;
            DBSetup.init();
            DBCollection loginCheck = DBSetup.database.getCollection("LoginDetails");
            DBCursor findIterable=loginCheck.find();
            for (DBObject counter:findIterable) {
                Username= (String) counter.get("Username");
                Password= (String) counter.get("Password");
                System.out.println(Username);
                System.out.println(Password);
                if (((txtUsername.getText()).equals(Username)) &&
((txtPassword.getText()).equals(Password))) {
                    LoginDetailsFound = true;
                    Parent root =
FXMLLoader.load(getClass().getResource("HomePage.fxml"));
                    HomePage.setTitle("Home");
                    HomePage.setScene(new Scene(root, 956, 638));
                    HomePage.show();
                    Controller.LoginStage.close();
                    txtPassword.setText("");
                    txtUsername.setText("");
                    break;

                }
            }

            if (LoginDetailsFound==false){
                Alert wrongLoginDetails = new Alert(Alert.AlertType.NONE);
                wrongLoginDetails.setAlertType(Alert.AlertType.WARNING);
                wrongLoginDetails.setContentText("Wrong Username or Password. Please re-
enter");
                wrongLoginDetails.showAndWait();

            }
    }

    @FXML
    public void goBacktoMain() throws Exception{
        Controller.LoginStage.close();
        Main.Primarystage.show();
    }
}
```

Code for the database initialization–

```
package sample;
import com.mongodb.DB;
import com.mongodb.MongoClient;
public class DBSetup {
    public static DB database;
    public static MongoClient mongoClient = new MongoClient("localhost", 27017);

    public static void init() {
        database = mongoClient.getDB("InventoryControl");
        database.createCollection("LoginDetails", null);
    }
    public static void initCategory() {
        database = mongoClient.getDB("InventoryControl");
        database.createCollection("Categories", null);
    }
    public static void initProductsAndStocks() {
        database = mongoClient.getDB("InventoryControl");
        database.createCollection("Product Details", null);
    }
}
```

Controller for Add category window –

```
package sample;

import com.mongodb.BasicDBObject;
import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.ButtonType;
import javafx.scene.control.TextField;

public class AddCategory {
    @FXML
    TextField txtCategory;

    public static String catName;
    public static boolean foundCategory;
    public static String DBCategoryName;

    @FXML
    public void addCategory(){
        foundCategory = false;   //Variable to indicate the finding of the relevant
category during the database search process
        catName=txtCategory.getText();
        if (!catName.equals("")) {  //Checking whether textfield is not left empty
            if (catName.equals(catName.toLowerCase())) { //To prevent the entering of
uppercase letters
                try {
                    DBSetup.initCategory();
                    DBCollection categoryCheck =
DBSetup.database.getCollection("Categories");
                    DBCursor findIterable = categoryCheck.find();
                    for (DBObject count : findIterable) {
                        DBCategoryName = (String) count.get("CategoryName");
                        if (catName.equals(DBCategoryName)) {
                            foundCategory = true;
                        }

                    }
                    if (foundCategory) {
                        Alert CategoryExists = new Alert(Alert.AlertType.NONE);
                        CategoryExists.setAlertType(Alert.AlertType.WARNING);
                        CategoryExists.setContentText("Category already exists");
                        CategoryExists.showAndWait();

                    } else {
                        BasicDBObject basicDBObjectForCat = new BasicDBObject();
                        basicDBObjectForCat.put("CategoryName", catName);
                        DBSetup.initCategory();
                        DBCollection collection =
DBSetup.database.getCollection("Categories");
                        collection.insert(basicDBObjectForCat);
                        Alert a = new Alert(Alert.AlertType.NONE);
                        a.setAlertType(Alert.AlertType.INFORMATION);
                        a.setContentText("Category entered successfully");
                        a.showAndWait();
                        txtCategory.setText("");
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }else{
                Alert lowercaseCategory = new Alert(Alert.AlertType.NONE);
                lowercaseCategory.setAlertType(Alert.AlertType.WARNING);
                lowercaseCategory.setContentText("Data entered must be lowercase");
                lowercaseCategory.showAndWait();
            }
        } else {
            Alert emptyCategory = new Alert(Alert.AlertType.NONE);
            emptyCategory.setAlertType(Alert.AlertType.WARNING);
            emptyCategory.setContentText("Category field is empty");
            emptyCategory.showAndWait();
        }
    }


}
```

Controller for Edit Category window –

```
package sample;

import com.mongodb.BasicDBObject;
import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.TextField;

public class EditCategory {
    @FXML
    TextField txtexistingCat;

    @FXML
    TextField txtmodifiedCat;

    public static String existingCategory;
    public static String modifiedCategory;
    public static boolean foundCategory;
    public static boolean modifiedCategoryExists;
    public static String DBCategoryName;
    public static String DBProductCategory;

    @FXML
    public void clickUpdate(){
        modifiedCategoryExists=false;
        foundCategory=false;
        existingCategory=txtexistingCat.getText();
        modifiedCategory=txtmodifiedCat.getText();
        if (!existingCategory.equals("") && !modifiedCategory.equals("")) {
            try {
                DBSetup.initCategory();
                DBCollection categoryCheck =
DBSetup.database.getCollection("Categories");
                DBCursor findIterable=categoryCheck.find();
                for (DBObject count:findIterable) {
                    DBCategoryName= (String) count.get("CategoryName");
                    if( existingCategory.equals(DBCategoryName)) {
                        foundCategory = true;
                    }
                    if(modifiedCategory.equals(DBCategoryName)) {
                        modifiedCategoryExists=true;
                    }
                }
                if (foundCategory==true && modifiedCategoryExists==false){  //If the
category to be deleted exists and the category which we're trying to change into doesn't
exist
                    BasicDBObject query = new BasicDBObject();
                    query.put("CategoryName", existingCategory);
                    BasicDBObject newValue = new BasicDBObject();
                    newValue.put("CategoryName", modifiedCategory);
                    BasicDBObject updateObject = new BasicDBObject();
                    updateObject.put("$set", newValue);
DBSetup.database.getCollection("Categories").update(query,updateObject);
                    Alert a = new Alert(Alert.AlertType.NONE);
                    a.setAlertType(Alert.AlertType.INFORMATION);
                    a.setContentText("Category updated successfully");
                    a.showAndWait();
                    txtexistingCat.setText("");
                    txtmodifiedCat.setText("");
                    DBSetup.initProductsAndStocks();
                    DBCollection productCheck = DBSetup.database.getCollection("Product 
Details");
                    DBCursor findIterable1 = productCheck.find();
                    for (DBObject count : findIterable1) {
                        DBProductCategory = (String) count.get("Category");
                        if (existingCategory.equals(DBProductCategory)) {
                            BasicDBObject queryForProductCategory = new BasicDBObject();
                            queryForProductCategory.put("Category", existingCategory);
                            BasicDBObject newValue1 = new BasicDBObject();
                            newValue1.put("Category", modifiedCategory);
                            BasicDBObject updateObject1 = new BasicDBObject();
                            updateObject1.put("$set", newValue1);
                            DBSetup.database.getCollection("Product 
Details").update(queryForProductCategory,updateObject1);
                        }

                    }
                }else if (foundCategory==false) {
                    Alert CategoryExists = new Alert(Alert.AlertType.NONE);
                    CategoryExists.setAlertType(Alert.AlertType.WARNING);
                    CategoryExists.setContentText("Category doesn't exist to update it");
                    CategoryExists.showAndWait();
                }else if (modifiedCategoryExists==true){
                    Alert ModCategoryExists = new Alert(Alert.AlertType.NONE);
                    ModCategoryExists.setAlertType(Alert.AlertType.WARNING);
                    ModCategoryExists.setContentText("The category which you're trying to 
change into already exists");
                    ModCategoryExists.showAndWait();
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        } else {
            Alert emptyCategory = new Alert(Alert.AlertType.NONE);
            emptyCategory.setAlertType(Alert.AlertType.WARNING);
            emptyCategory.setContentText("One of the two fields are empty");
            emptyCategory.showAndWait();
        }
    }
}
```

Code for Category View Model–

```
//Some parts of the following lines of code have been referenced from the following
website, https://medium.com/@keeptoo/adding-data-to-javafx-tableview-stepwise-
df582acbae4f
package sample;

import javafx.beans.property.SimpleStringProperty;

public class CategoryModel {

    private SimpleStringProperty category;

    public CategoryModel(String category) {
        this.category = new SimpleStringProperty(category);
    }

    public String getCategory() {
        return category.get();
    }

    public void setCategory(String category) {
        this.category = new SimpleStringProperty(category);
    }
}
```

Controller for View Category window –

```
//Some parts of the following lines of code have been referenced from the following
website, https://medium.com/@keeptoo/adding-data-to-javafx-tableview-stepwise-
df582acbae4f
package sample;


import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import java.net.URL;
import java.util.Locale;
import java.util.ResourceBundle;


public class CategoryViewController<findIterable> implements Initializable {

    @FXML
    private TableView<CategoryModel> tblView;

    @FXML
    public TableColumn<CategoryModel, String> tblColumn;


    @Override
    public void initialize(URL location, ResourceBundle resources) {

        tblColumn.setCellValueFactory(new PropertyValueFactory<>("category"));
        display();

    }
    public void display(){
            ObservableList<CategoryModel> mainList = FXCollections.observableArrayList();
            DBSetup.initCategory();
            DBCollection categoryCheck = DBSetup.database.getCollection("Categories");
            DBCursor findIterable = categoryCheck.find();
             for (DBObject count : findIterable) {
                CategoryModel category = new CategoryModel((String)
count.get("CategoryName"));
                category.setCategory((String) count.get("CategoryName"));
                //new CategoryModel((String) count.get("CategoryName"));
            mainList.add(category);

        }
        tblView.setItems(mainList);
    }
}
```

Controller for Delete Category window –

```
package sample;

import com.mongodb.BasicDBObject;
import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.TextField;

import java.awt.*;

public class DeleteCategory {
    @FXML
    TextField txtDelete;

    public static String deleteValue;
    public static boolean foundCategory;
    public static boolean categoryInProductTable;
    public static String DBCategoryName;
    public static String DBProductCategory;

    public void clickDelete(){
        foundCategory = false;  //Variable to indicate whether the item to delete exists
in the DB while searching it
        categoryInProductTable=false;  // Variable to indicate whether the category that
we're trying to delete has been used in the product table
        deleteValue=txtDelete.getText();
        if (!deleteValue.equals("")) {
            try {
                DBSetup.initCategory();
                DBCollection categoryCheck =
DBSetup.database.getCollection("Categories");
                DBCursor findIterable=categoryCheck.find();
                for (DBObject count:findIterable) {
                    DBCategoryName= (String) count.get("CategoryName");
                    if( deleteValue.equals(DBCategoryName)) {
                        foundCategory = true;
                    }
                }
                if (foundCategory) {
                    BasicDBObject basicDBObjectForCat = new BasicDBObject();
                    basicDBObjectForCat.put("CategoryName", deleteValue);
                    categoryCheck.findAndRemove(basicDBObjectForCat);
                    txtDelete.setText("");
                    DBSetup.initProductsAndStocks();
                    DBCollection productCheck = DBSetup.database.getCollection("Product 
Details");
                    DBCursor findIterable2 = productCheck.find();
                    for (DBObject counter : findIterable2) {
                        DBProductCategory= (String) counter.get("Category");
                        if (deleteValue.equals(DBProductCategory)) {
                            BasicDBObject basicDBObjectForProduct = new BasicDBObject();
                            basicDBObjectForProduct.put("Category", deleteValue);
                            productCheck.findAndRemove(basicDBObjectForProduct);
                            categoryInProductTable=true;
                        }
                    }
                    if (categoryInProductTable==true) { // Code to be executed if the 
deleted category has been used in the product table
                        Alert a1 = new Alert(Alert.AlertType.NONE);
                        a1.setAlertType(Alert.AlertType.WARNING);
                        a1.setContentText("The category that you have deleted had been 
used in the product details sector, as a result all corresponding records which consisted 
of that category have been deleted as well");
                        a1.showAndWait();
                    }else{
                        Alert a = new Alert(Alert.AlertType.NONE);
                        a.setAlertType(Alert.AlertType.INFORMATION);
                        a.setContentText("Category deleted successfully");
                        a.showAndWait();
                    }

                } else {
                    Alert CategoryExists = new Alert(Alert.AlertType.NONE);
                    CategoryExists.setAlertType(Alert.AlertType.WARNING);
                    CategoryExists.setContentText("Category doesn't exist to delete");
                    CategoryExists.showAndWait();
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        } else {
            Alert emptyCategory = new Alert(Alert.AlertType.NONE);
            emptyCategory.setAlertType(Alert.AlertType.WARNING);
            emptyCategory.setContentText("Nothing has been entered to delete");
            emptyCategory.showAndWait();
        }
    }
}
```

Controller for Add product window –

```
package sample;
```

```
import com.mongodb.BasicDBObject;
import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.ComboBox;
import javafx.scene.control.TextField;

public class AddProduct {
    @FXML
    TextField txtProductName;

    @FXML
    TextField txtID;

    @FXML
    ComboBox cmbCategory;

    public static String DBCategoryName;
    public static String DBID;
    public static String ID;
    public static String ProductName;
    public static String CategorySelected;
    public static boolean foundID;

    @FXML
    public void initialize(){  //Initialization of the dropdown box
        DBSetup.initCategory();
        DBCollection categoryCheck = DBSetup.database.getCollection("Categories");
        DBCursor findIterable=categoryCheck.find();
        ObservableList dropDownList = FXCollections.observableArrayList();
        for (DBObject count:findIterable) {
            DBCategoryName= (String) count.get("CategoryName");
            dropDownList.add(DBCategoryName);
        }
        cmbCategory.setItems(dropDownList);
    }

    @FXML
    public void clickAdd(){
        foundID=false;
        ID=txtID.getText();
        ProductName=txtProductName.getText();
        CategorySelected= (String) cmbCategory.getValue();
        System.out.println(CategorySelected);
        if (!ID.equals("") && !ProductName.equals("") && !(CategorySelected ==null)) {
            if (ID.length()<=10) { //Validation of the id length
```

```
                try {
                    DBSetup.initProductsAndStocks();
                    DBCollection productCheck = DBSetup.database.getCollection("Product
Details");
                    DBCursor findIterable = productCheck.find();
                    for (DBObject count : findIterable) {
                        DBID = (String) count.get("ProductID");
                        if (ID.equals(DBID)) {
                            foundID = true;
                        }

                    }
                    if (foundID) {
                        Alert ProductExists = new Alert(Alert.AlertType.NONE);
                        ProductExists.setAlertType(Alert.AlertType.WARNING);
                        ProductExists.setContentText("Product already exists");
                        ProductExists.showAndWait();

                    } else {
                        BasicDBObject basicDBObjectForProduct = new BasicDBObject();
                        basicDBObjectForProduct.put("ProductID", ID);
                        basicDBObjectForProduct.put("Product Name", ProductName);
                        basicDBObjectForProduct.put("Category", CategorySelected);
                        DBCollection collection = DBSetup.database.getCollection("Product
Details");
                        collection.insert(basicDBObjectForProduct);
                        Alert a = new Alert(Alert.AlertType.NONE);
                        a.setAlertType(Alert.AlertType.INFORMATION);
                        a.setContentText("Product details entered successfully");
                        a.showAndWait();
                        txtID.setText("");
                        txtProductName.setText("");
                        cmbCategory.setValue("");
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }else{
                Alert IDLengthAlert = new Alert(Alert.AlertType.NONE);
                IDLengthAlert.setAlertType(Alert.AlertType.WARNING);
                IDLengthAlert.setContentText("Product ID must be less than 10
characters");
                IDLengthAlert.showAndWait();
            }
        } else {
            Alert emptyDetails = new Alert(Alert.AlertType.NONE);
            emptyDetails.setAlertType(Alert.AlertType.WARNING);
            emptyDetails.setContentText("All details have not been entered");
            emptyDetails.showAndWait();
        }
```

```
    }


}
```

Code for view products model –

```
//Some parts of the following lines of code have been referenced from the following
website, https://medium.com/@keeptoo/adding-data-to-javafx-tableview-stepwise-
df582acbae4f
package sample;

import javafx.beans.property.SimpleStringProperty;

public class ProductModel {

    private SimpleStringProperty product;
    private SimpleStringProperty productID;
    private SimpleStringProperty productCategory;


    public ProductModel(String product,String productID,String productCategory) {
        this.product = new SimpleStringProperty(product);
        this.productID = new SimpleStringProperty(productID);
        this.productCategory = new SimpleStringProperty(productCategory);
    }

    public String getProduct() {
        return product.get();
    }

    public void setProduct(String product) {
        this.product = new SimpleStringProperty(product);
    }
    public String getProductID() {
        return productID.get();
    }

    public void setProductID(String productID) {

        this.productID = new SimpleStringProperty(productID);
    }
    public String getProductCategory() {

        return productCategory.get();
    }
    public void setProductCategory(String productCategory) {

        this.productCategory = new SimpleStringProperty(productCategory);
    }

}
```

Controller for view product window –

```
//Some parts of the following lines of code have been referenced from the following
website, https://medium.com/@keeptoo/adding-data-to-javafx-tableview-stepwise-
df582acbae4f
package sample;

import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;

import java.net.URL;
import java.util.Locale;
import java.util.ResourceBundle;


public class ViewProductsController<findIterable> implements Initializable {

    @FXML
    private TableView<ProductModel> tblView;

    @FXML
    public TableColumn<ProductModel, String> tblColumn;

    @FXML
    public TableColumn<ProductModel, String> tblColumnID;

    @FXML
    public TableColumn<ProductModel, String> tblColumnCategory;
    @Override
    public void initialize(URL location, ResourceBundle resources) {

        tblColumn.setCellValueFactory(new PropertyValueFactory<>("product"));
        tblColumnID.setCellValueFactory(new PropertyValueFactory<>("productID"));
        tblColumnCategory.setCellValueFactory(new
PropertyValueFactory<>("productCategory"));
        display();

    }
    public void display(){
        ObservableList<ProductModel> mainList = FXCollections.observableArrayList();
        DBSetup.initProductsAndStocks();
        DBCollection productCheck = DBSetup.database.getCollection("Product Details");
        DBCursor findIterable = productCheck.find();
        for (DBObject count : findIterable) {
            ProductModel product = new ProductModel((String) count.get("Product Name"),
(String) count.get("ProductID"), (String) count.get("Category"));
            mainList.add(product);

        }
        tblView.setItems(mainList);
    }
}
```

Controller for select id to modify window –

```
package sample;

import com.mongodb.BasicDBObject;
import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

public class EditProduct {
    @FXML
    TextField txtIDtoModify;

    public static String ProductIDtoModify;
    public static String DBID;
    public static boolean foundID;
    public static Stage editProductStage=new Stage();


    @FXML
    public void clickGO() throws Exception{
        foundID=false;
        ProductIDtoModify=txtIDtoModify.getText();
        if (!ProductIDtoModify.equals("")) {
            if (ProductIDtoModify.length()<=10) {
                try {
                    DBSetup.initProductsAndStocks();
                    DBCollection productCheck = DBSetup.database.getCollection("Product
Details");
                    DBCursor findIterable = productCheck.find();
                    for (DBObject count : findIterable) {
                        DBID = (String) count.get("ProductID");
                        if (ProductIDtoModify.equals(DBID)) {
                            foundID = true;
                        }

                    }
                    if (foundID) {
                        Parent root =
FXMLLoader.load(getClass().getResource("/UpdateProduct.fxml"));
                        editProductStage.setTitle("Enter modification details");
                        editProductStage.setScene(new Scene(root, 705, 439));
                        editProductStage.show();
                        HomePageController.EditProduct.close();
                        txtIDtoModify.setText("");

                    } else {
                        Alert IDdoesntExist = new Alert(Alert.AlertType.NONE);
                        IDdoesntExist.setAlertType(Alert.AlertType.WARNING);
                        IDdoesntExist.setContentText("Product ID does not exist");
                        IDdoesntExist.showAndWait();
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }else{
                Alert IDLengthAlert = new Alert(Alert.AlertType.NONE);
                IDLengthAlert.setAlertType(Alert.AlertType.WARNING);
                IDLengthAlert.setContentText("Product ID must be less than 10
characters");
                IDLengthAlert.showAndWait();
            }
        } else {
            Alert emptyID = new Alert(Alert.AlertType.NONE);
            emptyID.setAlertType(Alert.AlertType.WARNING);
            emptyID.setContentText("Product ID has not been entered");
            emptyID.showAndWait();
        }


    }
}
```

Controller for update product window –

```
package sample;

import com.mongodb.BasicDBObject;
import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.ComboBox;
import javafx.scene.control.TextField;

public class UpdateProduct {
    @FXML
    TextField txtProductName;

    @FXML
    TextField txtID;

    @FXML
    ComboBox cmbCategory;

    public static String DBCategoryName;
    public static String DBID;
    public static String ID;
    public static String ProductName;
    public static String CategorySelected;
    public static boolean foundID;

    @FXML
    public void initialize(){
        DBSetup.initCategory();
        DBCollection categoryCheck = DBSetup.database.getCollection("Categories");
        DBCursor findIterable=categoryCheck.find();
        ObservableList dropDownList = FXCollections.observableArrayList();
        for (DBObject count:findIterable) {
            DBCategoryName= (String) count.get("CategoryName");
            dropDownList.add(DBCategoryName);
        }
        cmbCategory.setItems(dropDownList);
    }

    @FXML
    public void clickUpdate(){
        foundID=false;
        ID=txtID.getText();
        ProductName=txtProductName.getText();
        CategorySelected= (String) cmbCategory.getValue();
        System.out.println(CategorySelected);
        if (!ID.equals("") && !ProductName.equals("") && !(CategorySelected ==null)) {
            if (ID.length()<=10) {
                try {
                    DBSetup.initProductsAndStocks();
                    DBCollection productCheck = DBSetup.database.getCollection("Product
Details");
                    DBCursor findIterable = productCheck.find();
                    for (DBObject count : findIterable) {
                        DBID = (String) count.get("ProductID");
                        if (ID.equals(DBID)) {
                            foundID = true;
                        }

                    }
                    if (foundID) {
                        Alert ProductExists = new Alert(Alert.AlertType.NONE);
                        ProductExists.setAlertType(Alert.AlertType.WARNING);
                        ProductExists.setContentText("The product ID that you're trying
to change into already exists");
                        ProductExists.showAndWait();

                    } else {
                        BasicDBObject query = new BasicDBObject();
                        query.put("ProductID", EditProduct.ProductIDtoModify);
                        BasicDBObject newValue = new BasicDBObject();
                        newValue.put("ProductID", ID);
                        newValue.put("Product Name", ProductName);
                        newValue.put("Category", CategorySelected);
                        BasicDBObject updateObject = new BasicDBObject();
                        updateObject.put("$set", newValue);
                        DBSetup.database.getCollection("Product
Details").update(query,updateObject);
                        Alert a = new Alert(Alert.AlertType.NONE);
                        a.setAlertType(Alert.AlertType.INFORMATION);
                        a.setContentText("Product details updated successfully");
                        a.showAndWait();
                        txtID.setText("");
                        txtProductName.setText("");
                        cmbCategory.setValue("");
                        EditProduct.editProductStage.close();
                        HomePageController.EditProduct.show();
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }else{
                Alert IDLengthAlert = new Alert(Alert.AlertType.NONE);
                IDLengthAlert.setAlertType(Alert.AlertType.WARNING);
                IDLengthAlert.setContentText("Product ID must be less than 10 
characters");
                IDLengthAlert.showAndWait();
            }
        } else {
            Alert emptyDetails = new Alert(Alert.AlertType.NONE);
            emptyDetails.setAlertType(Alert.AlertType.WARNING);
            emptyDetails.setContentText("All details have not been entered");
            emptyDetails.showAndWait();
        }


    }
}
```

Controller for delete product window –

```
package sample;

import com.mongodb.BasicDBObject;
import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.TextField;

public class DeleteProduct {
    @FXML
    TextField txtDelete;

    public static boolean foundProduct;
    public static String DBProductID;
    public static String deleteValue;

    public void clickDelete(){
        foundProduct = false;
        deleteValue=txtDelete.getText();
        if (!deleteValue.equals("")) {
            if (deleteValue.length()<=10) {
                try {
                    DBSetup.initProductsAndStocks();
                    DBCollection productCheck = DBSetup.database.getCollection("Product
Details");
                    DBCursor findIterable = productCheck.find();
                    for (DBObject count : findIterable) {
                        DBProductID = (String) count.get("ProductID");
                        if (deleteValue.equals(DBProductID)) {
                            foundProduct = true;
                        }
                    }
                    if (foundProduct) {
                        BasicDBObject basicDBObjectForProduct = new BasicDBObject();
                        basicDBObjectForProduct.put("ProductID", deleteValue);
                        productCheck.findAndRemove(basicDBObjectForProduct);
                        Alert a = new Alert(Alert.AlertType.NONE);
                        a.setAlertType(Alert.AlertType.INFORMATION);
                        a.setContentText("Product deleted successfully");
                        a.showAndWait();
                        txtDelete.setText("");

                    } else {
                        Alert ProductExists = new Alert(Alert.AlertType.NONE);
                        ProductExists.setAlertType(Alert.AlertType.WARNING);
                        ProductExists.setContentText("Product doesn't exist to delete");
                        ProductExists.showAndWait();
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }else{
                Alert IDLengthAlert = new Alert(Alert.AlertType.NONE);
                IDLengthAlert.setAlertType(Alert.AlertType.WARNING);
                IDLengthAlert.setContentText("Product ID must be less than 10
characters");
                IDLengthAlert.showAndWait();
            }
        } else {
            Alert emptyField = new Alert(Alert.AlertType.NONE);
            emptyField.setAlertType(Alert.AlertType.WARNING);
            emptyField.setContentText("Nothing has been entered to delete");
            emptyField.showAndWait();
        }
    }
}
```

Controller for update stocks window –

```
package sample;

import com.mongodb.BasicDBObject;
import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.TextField;

public class UpdateStocks {
    @FXML
    TextField txtID;

    @FXML
    TextField txtQuantity;

    public static String DBID;
    public static String ID;
    public static String Quantity;
    public static boolean foundID;

    @FXML
    public void clickUpdate(){
        foundID=false;
        ID=txtID.getText();
        Quantity=txtQuantity.getText();
        if (!ID.equals("") && !Quantity.equals("")) {
            if (ID.length()<=10) {
                try {
                    DBSetup.initProductsAndStocks();
                    DBCollection productCheck = DBSetup.database.getCollection("Product
Details");
                    DBCursor findIterable = productCheck.find();
                    for (DBObject count : findIterable) {
                        DBID = (String) count.get("ProductID");
                        if (ID.equals(DBID)) {
                            foundID = true;
                        }
                    }
                    if (foundID) {
                        BasicDBObject query = new BasicDBObject();
                        query.put("ProductID", ID);
                        BasicDBObject newValue = new BasicDBObject();
                        newValue.put("Stocks Available", Quantity);
                        BasicDBObject updateObject = new BasicDBObject();
                        updateObject.put("$set", newValue);
                        DBSetup.database.getCollection("Product 
Details").update(query,updateObject);
                        Alert a = new Alert(Alert.AlertType.NONE);
                        a.setAlertType(Alert.AlertType.INFORMATION);
                        a.setContentText("Stock details updated successfully");
                        a.showAndWait();
                        txtID.setText("");
                        txtQuantity.setText("");
                    } else {
                        Alert ProductdoesntExist = new Alert(Alert.AlertType.NONE);
                        ProductdoesntExist.setAlertType(Alert.AlertType.WARNING);
                        ProductdoesntExist.setContentText("Product ID doesn't exist");
                        ProductdoesntExist.showAndWait();
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }else{
                Alert IDLengthAlert = new Alert(Alert.AlertType.NONE);
                IDLengthAlert.setAlertType(Alert.AlertType.WARNING);
                IDLengthAlert.setContentText("Product ID must be less than 10 
characters");
                IDLengthAlert.showAndWait();
            }
        } else {
            Alert emptyDetails = new Alert(Alert.AlertType.NONE);
            emptyDetails.setAlertType(Alert.AlertType.WARNING);
            emptyDetails.setContentText("All details have not been entered");
            emptyDetails.showAndWait();
        }


    }
}
```

Code for view stocks model –

```
//Some parts of the following lines of code have been referenced from the following 
website, https://medium.com/@keeptoo/adding-data-to-javafx-tableview-stepwise-
df582acbae4f
package sample;
```

```
import javafx.beans.property.SimpleStringProperty;

public class StocksModel {

    private SimpleStringProperty product;
    private SimpleStringProperty stock;


    public StocksModel(String product, String stock) {
        this.product = new SimpleStringProperty(product);
        this.stock = new SimpleStringProperty(stock);
    }

    public String getProduct() {
        return product.get();
    }

    public void setProduct(String product) {
        this.product = new SimpleStringProperty(product);
    }

    public String getStock() {
        return stock.get();
    }

    public void setStock(String stock) {
        this.stock = new SimpleStringProperty(stock);
    }

}
```

Controller for view stocks(all) window –

```
//Some parts of the following lines of code have been referenced from the following
website, https://medium.com/@keeptoo/adding-data-to-javafx-tableview-stepwise-
df582acbae4f
package sample;

import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;

import java.net.URL;
import java.util.Locale;
import java.util.ResourceBundle;


public class ViewStockDetails<findIterable> implements Initializable {

    @FXML
    private TableView<StocksModel> tblView;

    @FXML
    public TableColumn<StocksModel, String> tblColumnProduct;

    @FXML
    public TableColumn<StocksModel, String> tblColumnStocks;

    public static String stocks;



    @Override
    public void initialize(URL location, ResourceBundle resources) {

        tblColumnProduct.setCellValueFactory(new PropertyValueFactory<>("product"));
        tblColumnStocks.setCellValueFactory(new PropertyValueFactory<>("stock"));
        display2();

    }
    public void display2(){
        ObservableList<StocksModel> mainList = FXCollections.observableArrayList();
        DBSetup.initProductsAndStocks();
        DBCollection productCheck = DBSetup.database.getCollection("Product Details");
        DBCursor findIterable = productCheck.find();
        for (DBObject count : findIterable) {
            stocks= (String) count.get("Stocks Available");
            System.out.println(stocks);
            if(stocks==null){
                stocks="Not entered";
            }
            StocksModel tabledata = new StocksModel((String) count.get("Product Name"),
stocks);
            mainList.add(tabledata);
        }
        tblView.setItems(mainList);
    }
}
```

Controller for view stocks(single) window –

```
package sample;

import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;

public class ViewStocksSingle {
    @FXML
    TextField txtSearch;

    @FXML
    Label lblStock;

    public static String searchText;
    public static String DBProductID;
    public static String DBStocks;
    public static boolean productFound;


    @FXML
    public void clickSearch(){
    productFound=false;
        searchText=txtSearch.getText();
        if (!searchText.equals("")) {
            if (searchText.length()<=10) {
                try {
                    DBSetup.initProductsAndStocks();
                    DBCollection Products = DBSetup.database.getCollection("Product
Details");
                    DBCursor findIterable = Products.find();
                    for (DBObject count : findIterable) {
                        DBProductID = (String) count.get("ProductID");
                        DBStocks = (String) count.get("Stocks Available");
                        if (DBStocks==null){ //If stock details haven't been entered yet
for a product
                            DBStocks="Not Entered";
                        }
                        if (searchText.equals(DBProductID)) {
                            productFound = true;
                            break;
                        }
                    }
                    if (productFound) {
                        lblStock.setText(DBStocks);  //Displaying stock number in a label

                    } else {
                        Alert a = new Alert(Alert.AlertType.NONE);
                        a.setAlertType(Alert.AlertType.WARNING);
                        a.setContentText("The product you are searching for is not
available");
                        a.showAndWait();
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }else{
                Alert IDLengthAlert = new Alert(Alert.AlertType.NONE);
                IDLengthAlert.setAlertType(Alert.AlertType.WARNING);
                IDLengthAlert.setContentText("Product ID must be less than 10
characters");
                IDLengthAlert.showAndWait();
            }
        } else {
            Alert emptyCategory = new Alert(Alert.AlertType.NONE);
            emptyCategory.setAlertType(Alert.AlertType.WARNING);
            emptyCategory.setContentText("Search details have not been entered");
            emptyCategory.showAndWait();
        }
    }







}
```