```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package view.studentView;

import java.io.IOException;
import java.net.URL;
import java.time.LocalDate;
import java.time.ZoneId;
import java.util.Date;
import java.util.List;
import java.util.ResourceBundle;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.RadioButton;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.ToggleGroup;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import model.ActiveUser;
import model.BorrowedEquipments;
import model.Equipments;
import presenter.ActiveUserPresenter;
import presenter.EquipmentPresenter;
import presenter.StudentPresenter;

/**
 * FXML Controller class
 *
 * @author Adarsha
 */
public class SearchEquipmentController implements Initializable {

    @FXML
    private Button homeadmin_fx;
    @FXML
    private Button equipmentSearch_fx;
    @FXML
    private Text home_fx;
    @FXML
    private TableView<Equipments> equipdettable_fx;
```

```
    @FXML
    private TableColumn<Equipments, Integer> id_column;
    @FXML
    private TableColumn<Equipments, String> name_column;
    @FXML
    private TableColumn<Equipments, String> brand_column;
    @FXML
    private TableColumn<Equipments, String> type_column;
    @FXML
    private TableColumn<Equipments, String> availableQuantities_column;
    private TableColumn<Equipments, String> campus_column;
    @FXML
    private RadioButton rbequipsearchbyid_fx;
    @FXML
    private RadioButton rbequipsearchbyname_fx;
    @FXML
    private TextField equipsearch_fx;
    @FXML
    private Button searchequipdet_fx;
    @FXML
    private Button notify_fx;
    @FXML
    private Button fullEquipmentListBtn;
    @FXML
    private Button bookEquipmentBtn;

    private EquipmentPresenter equipmentPresenter;

    private Equipments borrowEquipments;
    private int equipmentId;
    @FXML
    private DatePicker borrowDatePicker;
    @FXML
    private DatePicker returnDatePicker;

    private ActiveUserPresenter activeUserPresenter;
    private StudentPresenter studentPresenter;
    @FXML
    private Button searchequip_fx;
    @FXML
    private Button booking_fx;

    /**
     * Initializes the controller class.
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        activeUserPresenter = new ActiveUserPresenter();
        equipmentPresenter = new EquipmentPresenter();
        studentPresenter = new StudentPresenter();


        List<Equipments> equipmentList =
equipmentPresenter.getAllEquipmentListForStudents();
        getAllEquipmentList(equipmentList);


        ToggleGroup group = new ToggleGroup();
```

```
        rbequipsearchbyid_fx.setToggleGroup(group);
        rbequipsearchbyname_fx.setToggleGroup(group);
    }

    @FXML
    private void onHomeButtonClicked(ActionEvent event) throws
IOException {
        Parent root =
FXMLLoader.load(getClass().getResource("StudentDashboard.fxml"));
        Scene scene = new Scene(root);
        Stage window = (Stage) ((Node)
event.getSource()).getScene().getWindow();
        window.setScene(scene);
        window.show();
    }

    @FXML
    private void onSearchButtonClicked(ActionEvent event) {
         if(rbequipsearchbyname_fx.isSelected()==true){
            String keyword = equipsearch_fx.getText();
            System.out.println("Keyword is :"+ keyword);
            List<Equipments> searchedEquipments =
equipmentPresenter.findEquipmentsByName(keyword);
            getAllEquipmentList(searchedEquipments);
        }else if (rbequipsearchbyid_fx.isSelected()==true){
            String keyword = equipsearch_fx.getText();
            try{
                int id = Integer.parseInt(keyword);
                List<Equipments> searchedEquipments =
equipmentPresenter.findEquipmentsById(id);
                getAllEquipmentList(searchedEquipments);
            }catch(NumberFormatException ex){
                Alert alert = new Alert(Alert.AlertType.WARNING);
                alert.setTitle("Invalid Input");
                alert.setHeaderText("Please enter a numeric value for
Equipment ID.");
                alert.showAndWait();
            }
        }else{
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("No search type selected");
            alert.setHeaderText("Please select one search option
first.");
            alert.showAndWait();
        }

    }

    @FXML
    private void onFullEquipmentListBtnClicked(ActionEvent event) {
        List<Equipments> equipmentList =
equipmentPresenter.getAllEquipmentListForStudents();
        getAllEquipmentList(equipmentList);

    }

    @FXML
    private void onBookEquipmentBtnClicked(ActionEvent event) {
```

```
        borrowEquipments =
equipdettable_fx.getSelectionModel().getSelectedItem();
        if(borrowEquipments != null){
            equipmentId = borrowEquipments.getEquipment_id();
        }
        ZoneId defaultZoneId = ZoneId.systemDefault();
        equipdettable_fx.getSelectionModel().getSelectedItem();

        Date borrowDate =
Date.from(borrowDatePicker.getValue().atStartOfDay(defaultZoneId).toInsta
nt());
        Date returnDate =
Date.from(returnDatePicker.getValue().atStartOfDay(defaultZoneId).toInsta
nt());
        String status = "Borrowed";

         List<ActiveUser> myUsername =
activeUserPresenter.getMyUsername();

        BorrowedEquipments selectedEquipment = new
BorrowedEquipments(borrowEquipments, myUsername.get(0), borrowDate,
returnDate, status);
        System.out.println(" the returned equipment is : "+
selectedEquipment);
        Boolean b =
this.equipmentPresenter.addBorrowedEquipment(selectedEquipment);

        if(b==true){
            Date dateNow =
this.studentPresenter.getDateFromLocalDate(LocalDate.now());
            String notification = (myUsername.get(0).getUsername() + " "
+ dateNow +" - Booked an equipment named : "+
selectedEquipment.getEquipmentName());
            System.out.println("notification is : "+ notification);
            this.studentPresenter.writeToFile(notification);

            Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
            alert.setTitle("Equipment Booked");
            alert.setHeaderText("Equipment
"+selectedEquipment.getEquipmentName()+" has been Booked.");
            alert.showAndWait();


        }else{
            Date dateNow =
this.studentPresenter.getDateFromLocalDate(LocalDate.now());
            String notification = (myUsername.get(0).getUsername() + " "
+ dateNow +" - Failed to Book an equipment named : "+
selectedEquipment.getEquipmentName());
            System.out.println(notification);
            this.studentPresenter.writeToFile(notification);

            Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
            alert.setTitle("Failed to Book Equipment.");
            alert.setHeaderText("Failed to Book : "+
selectedEquipment.getEquipmentName());
            alert.showAndWait();
         }
```

```
    }

    private void getAllEquipmentList(List<Equipments> equipmentList) {
        id_column.setCellValueFactory(new
PropertyValueFactory<>("equipment_id"));
        name_column.setCellValueFactory(new
PropertyValueFactory<>("equipment_name"));
        brand_column.setCellValueFactory(new
PropertyValueFactory<>("equipment_brand"));
        type_column.setCellValueFactory(new
PropertyValueFactory<>("equipment_type"));
        availableQuantities_column.setCellValueFactory(new
PropertyValueFactory<>("availableQuantities"));

        ObservableList<Equipments> equipmentsList =
FXCollections.observableList(equipmentList);
        equipdettable_fx.setItems(equipmentsList);
        if (equipmentsList.isEmpty()) {
            equipdettable_fx.setPlaceholder(new Label("No records
found!"));
        }
    }

@FXML
    private void onSettingsButtonClicked(ActionEvent event) throws
IOException {
        Parent root =
FXMLLoader.load(getClass().getResource("StudentProfile.fxml"));
        Scene scene = new Scene(root);
        Stage window = (Stage) ((Node)
event.getSource()).getScene().getWindow();
        window.setScene(scene);
        window.show();
    }

    @FXML
    private void onNotificationButtonClicked(ActionEvent event) throws
IOException {
        Parent root =
FXMLLoader.load(getClass().getResource("NotificationStudent.fxml"));
        Scene scene = new Scene(root);
        Stage window = (Stage) ((Node)
event.getSource()).getScene().getWindow();
        window.setScene(scene);
        window.show();
    }

    @FXML
    private void onSearchEquipmentButtonClicked(ActionEvent event) throws
IOException {
        Parent root =
FXMLLoader.load(getClass().getResource("SearchEquipment.fxml"));
        Scene scene = new Scene(root);
        Stage window = (Stage) ((Node)
event.getSource()).getScene().getWindow();
        window.setScene(scene);
        window.show();
    }
```

```
    @FXML
    private void onBookingButtonClicked(ActionEvent event) throws
IOException {
        Parent root =
FXMLLoader.load(getClass().getResource("StudentHistory.fxml"));
        Scene scene = new Scene(root);
        Stage window = (Stage) ((Node)
event.getSource()).getScene().getWindow();
        window.setScene(scene);
        window.show();
    }
}
```