import datamodel.Player;

import datamodel.PlayerData;

import dbUtil.dbConnection;

import javafx.application.Platform;

import javafx.beans.value.ChangeListener;

import javafx.beans.value.ObservableValue;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.collections.transformation.FilteredList;

import javafx.collections.transformation.SortedList;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.input.KeyCode;

import javafx.scene.input.KeyEvent;

import javafx.scene.layout.BorderPane;

import javafx.scene.paint.Color;

import javafx.util.Callback;

import java.io.IOException;

import java.nio.FloatBuffer;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.util.\*;

import java.util.function.Predicate;

public class AuctionController {

private List<Player> players;

@FXML

private BorderPane mainBorderPane;

@FXML

private Label FplayerName;

@FXML

private Label FplayerDOB;

@FXML

private Label FbasePrice;

@FXML

private Label Fage;

@FXML

private Label Fcountry;

@FXML

private Label Frole;

@FXML

private Label FpreviousTeam;

@FXML

private ListView listViewPlayers;

@FXML

private ContextMenu listContextMenu;

@FXML

private Label timerLabel;

@FXML

private Button cskID;

@FXML

private Button ddID;

@FXML

private Button kkrID;

@FXML

private Button kxipID;

@FXML

private Button rcbID;

@FXML

private Button rrID;

@FXML

private Button miID;

@FXML

private Button srhID;

@FXML

private Label bidWithText;

@FXML

private Label bidWithTeam;

@FXML

private Label takenTeam;

@FXML

private Label playerSoldTo;

@FXML

private Label lakhsText;

@FXML

private Label bidRateText;

@FXML

private Label bidTemp;

@FXML

private Label FcskPurse;

@FXML

private Label FddPurse;

@FXML

private Label FkkrPurse;

@FXML

private Label FkxipPurse;

@FXML

private Label FmiPurse;

@FXML

private Label FrrPurse;

@FXML

private Label FrcbPurse;

@FXML

private Label FsrhPurse;

private static int tempBidRate;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* THIS IS AUCTION OVERVIEW DATABASE TABLE\*/

@FXML

private TableView<PlayerAuctionData> playerauctiontable;

@FXML

private TableColumn<PlayerAuctionData, String> playernamecolumn;

@FXML

private TableColumn<PlayerAuctionData, Integer> playeragecolumn;

@FXML

private TableColumn<PlayerAuctionData, String> playerteamcolumn;

@FXML

private TableColumn<PlayerAuctionData, Integer> playerpricecolumn;

@FXML

private TableColumn<PlayerAuctionData, String> playerrolecolumn;

@FXML

private TableColumn<PlayerAuctionData, String> playercountrycolumn;

private dbConnection dc;

public ObservableList<PlayerAuctionData> data;

private ToggleButton filterToggleButton;

private FilteredList<Player> filteredList;

private Predicate<Player> wantAllItems;

private Predicate<Player> wantTodaysItems;

private int interval = 16;

private int count = 0;

Purse purse = new Purse();

public void initialize(){

this.dc = new dbConnection();

disableEverything();

bidWithText.setVisible(false);

playerSoldTo.setVisible(false);

lakhsText.setVisible(false);

bidRateText.setVisible(false);

listViewPlayers.setStyle("-fx-font-size : 20.0; -fx-control-inner-background: #333847;");

playerauctiontable.setStyle("-fx-font-size : 13.0; ");

listContextMenu = new ContextMenu();

MenuItem deleteMenuItem = new MenuItem("Delete"); // the one that comes when we right click with mouse

deleteMenuItem.setOnAction(new EventHandler<ActionEvent>() {

@Override

public void handle(ActionEvent event) {

Player player = (Player) listViewPlayers.getSelectionModel().getSelectedItem();

deleteItem(player);

}

});

listContextMenu.getItems().addAll(deleteMenuItem);

listViewPlayers.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<Player>() {

@Override

public void changed(ObservableValue<? extends Player> observable, Player oldValue, Player newValue) {

if(newValue != null) {

Player player = (Player) listViewPlayers.getSelectionModel().getSelectedItem();

// itemDetailsTextArea.setText(item.getDetails());

// DateTimeFormatter df = DateTimeFormatter.ofPattern("MMMM d, yyyy"); // "d M yy");

// deadlineLabel.setText(df.format(item.getDeadline()));

FplayerName.setText(player.getPlayerName());

DateTimeFormatter df = DateTimeFormatter.ofPattern("MMMM d, yyyy");

FplayerDOB.setText(df.format(player.getPlayerDOB()));

FbasePrice.setText(player.getBasePrice());

Fage.setText(player.getAge());

Fcountry.setText(player.getCountry());

Frole.setText(player.getRole());

FpreviousTeam.setText(player.getPreviousTeam());

// playerSoldTo.setVisible(false);

// takenTeam.setText("");

// bidWithText.setVisible(false);

// disableEverything();

bidWithTeam.setVisible(true);

bidWithTeam.setText(" ");

playerSoldTo.setVisible(false);

takenTeam.setText(" ");

timerLabel.setText("");

lakhsText.setVisible(false);

bidRateText.setVisible(false);

bidTemp.setText("");

setPrice();

}

}

});

wantAllItems = new Predicate<Player>() {

@Override

public boolean test(Player player) {

return true;

}

};

wantTodaysItems = new Predicate<Player>() {

@Override

public boolean test(Player player) {

return (player.getPlayerDOB().equals(LocalDate.now()));

}

};

filteredList = new FilteredList<Player>(PlayerData.getInstance().getPlayers(), wantAllItems);

SortedList<Player> sortedList = new SortedList<Player>(filteredList,

new Comparator<Player>() {

@Override

public int compare(Player o1, Player o2) {

return o2.getAge().compareTo(o1.getAge());

}

});

// todoListView.setItems(TodoData.getInstance().getTodoItems());

listViewPlayers.setItems(sortedList);

listViewPlayers.getSelectionModel().setSelectionMode(SelectionMode.SINGLE);

listViewPlayers.getSelectionModel().selectFirst();

listViewPlayers.setCellFactory(new Callback<ListView<Player>, ListCell<Player>>() {

@Override

public ListCell<Player> call(ListView<Player> param) {

ListCell<Player> cell = new ListCell<Player>() {

@Override

protected void updateItem(Player player, boolean empty) {

super.updateItem(player, empty);

if(empty) {

setText(null);

} else {

setText(player.getPlayerID());

if(player.getPlayerDOB().isBefore(LocalDate.now().plusDays(1))) {

setTextFill(Color.WHITE);

} else if(player.getPlayerDOB().equals(LocalDate.now().plusDays(1))) {

setTextFill(Color.WHITE);

}

}

}

};

cell.emptyProperty().addListener(

(obs, wasEmpty, isNowEmpty) -> {

if(isNowEmpty) {

cell.setContextMenu(null);

} else {

cell.setContextMenu(listContextMenu);

}

});

return cell;

}

});

}

@FXML

public void showNewItemDialog(){

Dialog<ButtonType> dialog = new Dialog<>();

dialog.initOwner(mainBorderPane.getScene().getWindow());

dialog.setTitle("Add IPL Player");

dialog.setHeaderText("Enter the details of the Player Below !");

FXMLLoader fxmlLoader = new FXMLLoader();

fxmlLoader.setLocation(getClass().getResource("playerDialog.fxml"));

try{

dialog.getDialogPane().setContent(fxmlLoader.load());

}catch (IOException e){

System.out.println("couldnt load the dialog");

e.printStackTrace();

return;

}

dialog.getDialogPane().getButtonTypes().add(ButtonType.OK);

dialog.getDialogPane().getButtonTypes().add(ButtonType.CANCEL);

Optional<ButtonType> result = dialog.showAndWait();

if(result.isPresent() && result.get() == ButtonType.OK){

DialogController controller = fxmlLoader.getController();

Player newItem = controller.processResults();

listViewPlayers.getSelectionModel().select(newItem);

}

}

@FXML

public void startAuction(){

if(count==0){

setTimer();

count++;

bidWithTeam.setText("");

}else{

interval=16;

bidWithTeam.setText("");

}

nonDisableEverything();

bidWithText.setVisible(false);

tempBidRate = Integer.parseInt(FbasePrice.getText());

int tempBaseBid = Integer.parseInt(FbasePrice.getText());

outOfPurse(tempBaseBid);

}

@FXML

public void cskClicked(){

interval = 16;

nonDisableEverything();

cskID.setDisable(true);

bidWithTeam.setVisible(true);

bidWithTeam.setText("CSK");

lakhsText.setVisible(true);

bidRateText.setVisible(true);

incrementBid();

outOfPurse(tempBidRate);

}

@FXML

public void ddClicked(){

interval = 16;

nonDisableEverything();

ddID.setDisable(true);

bidWithTeam.setText("DD");

lakhsText.setVisible(true);

bidRateText.setVisible(true);

incrementBid();

outOfPurse(tempBidRate);

}

@FXML

public void kkrClicked(){

interval = 16;

nonDisableEverything();

kkrID.setDisable(true);

bidWithTeam.setText("KKR");

lakhsText.setVisible(true);

bidRateText.setVisible(true);

incrementBid();

outOfPurse(tempBidRate);

}

@FXML

public void kxipClicked(){

interval = 16;

nonDisableEverything();

kxipID.setDisable(true);

bidWithTeam.setText("KXIP");

lakhsText.setVisible(true);

bidRateText.setVisible(true);

incrementBid();

outOfPurse(tempBidRate);

}

@FXML

public void rcbClicked(){

interval = 16;

nonDisableEverything();

rcbID.setDisable(true);

bidWithTeam.setText("RCB");

lakhsText.setVisible(true);

bidRateText.setVisible(true);

incrementBid();

outOfPurse(tempBidRate);

}

@FXML

public void rrClicked(){

interval = 16;

nonDisableEverything();

rrID.setDisable(true);

bidWithTeam.setText("RR");

lakhsText.setVisible(true);

bidRateText.setVisible(true);

incrementBid();

outOfPurse(tempBidRate);

}

@FXML

public void miClicked(){

interval = 16;

nonDisableEverything();

miID.setDisable(true);

bidWithTeam.setText("MI");

lakhsText.setVisible(true);

bidRateText.setVisible(true);

incrementBid();

outOfPurse(tempBidRate);

}

@FXML

public void srhClicked(){

interval = 16;

nonDisableEverything();

srhID.setDisable(true);

bidWithTeam.setText("SRH");

lakhsText.setVisible(true);

bidRateText.setVisible(true);

incrementBid();

outOfPurse(tempBidRate);

}

@FXML

public void handleKeyPressed(KeyEvent keyEvent) {

Player selectedItem = (Player) listViewPlayers.getSelectionModel().getSelectedItem();

if(selectedItem != null) {

if(keyEvent.getCode().equals(KeyCode.DELETE)) {

deleteItem(selectedItem);

}

}

}

@FXML

public void handleClickListView() {

Player player = (Player) listViewPlayers.getSelectionModel().getSelectedItem();

// itemDetailsTextArea.setText(item.getDetails());

// deadlineLabel.setText(item.getDeadline().toString());

FplayerName.setText(player.getPlayerName());

FplayerDOB.setText(player.getPlayerDOB().toString());

FbasePrice.setText(player.getBasePrice());

Fage.setText(player.getAge());

Fcountry.setText(player.getCountry());

Frole.setText(player.getRole());

FpreviousTeam.setText(player.getPreviousTeam());

}

public void deleteItem(Player player) {

Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

alert.setTitle("Delete Todo Item");

alert.setHeaderText("Delete item: " + player.getPlayerName());

alert.setContentText("Are you sure? Press OK to confirm, or cancel to Back out.");

Optional<ButtonType> result = alert.showAndWait();

if(result.isPresent() && (result.get() == ButtonType.OK)) {

PlayerData.getInstance().deletePlayer(player);

}

}

@FXML

public void handleFilterButton() {

Player selectedItem = (Player) listViewPlayers.getSelectionModel().getSelectedItem();

if(filterToggleButton.isSelected()) {

filteredList.setPredicate(wantTodaysItems);

if(filteredList.isEmpty()) {

FplayerDOB.setText("");

FbasePrice.setText("");

Fage.setText("");

Fcountry.setText("");

Frole.setText("");

FpreviousTeam.setText("");

} else if(filteredList.contains(selectedItem)) {

listViewPlayers.getSelectionModel().select(selectedItem);

} else {

listViewPlayers.getSelectionModel().selectFirst();

}

} else {

filteredList.setPredicate(wantAllItems);

listViewPlayers.getSelectionModel().select(selectedItem);

}

}

@FXML

public void handleExit() {

Platform.exit();

}

public void setTimer() {

Timer timer = new Timer();

timer.scheduleAtFixedRate(new TimerTask() {

public void run() {

if(interval > 0)

{

Platform.runLater(() -> timerLabel.setText("00 : "+interval));

//System.out.println(interval);

interval--;

if(interval==0){

Platform.runLater(() -> timerLabel.setText("00 : 00"));

//System.out.println("Player is taken");

}

}

else if(interval == 0){

String isTaken = bidWithTeam.getText();

if(isTaken==""){

Platform.runLater(() -> takenTeam.setText("UNSOLD"));

Platform.runLater(() -> takenTeam.setTextFill(Color.RED));

bidWithText.setVisible(false);

bidWithTeam.setVisible(false);

disableEverything();

}else{

if(isTaken!=" "){

playerSoldTo.setVisible(true);

Platform.runLater(() -> takenTeam.setText(isTaken));

bidWithTeam.setVisible(false);

bidWithText.setVisible(false);

disableEverything();

if(isTaken.equals("CSK")){

purse.purseCSK -= tempBidRate;

}

if(isTaken.equals("DD")){

purse.purseDD -= tempBidRate;

}

if(isTaken.equals("KKR")){

purse.purseKKR -= tempBidRate;

}

if(isTaken.equals("KXIP")){

purse.purseKXIP -= tempBidRate;

}

if(isTaken.equals("RCB")){

purse.purseRCB -= tempBidRate;

}

if(isTaken.equals("RR")){

purse.purseRR -= tempBidRate;

}

if(isTaken.equals("MI")){

purse.purseMI -= tempBidRate;

}

if(isTaken.equals("SRH")){

purse.purseSRH -= tempBidRate;

}

Platform.runLater(() -> setPrice());

}

}

interval--;

String name = FplayerName.getText();

int age = Integer.parseInt(Fage.getText());

String team;

String role = Frole.getText();

String country = Fcountry.getText();

if(takenTeam.getText()==" "){

team = "UNSOLD";

}

else{

team = takenTeam.getText();

}

int rate = tempBidRate;

System.out.println(name);

System.out.println(age);

System.out.println(team);

System.out.println(rate);

AuctionDatabase a = new AuctionDatabase();

try {

a.registerPlayer(name,age,role,country,team,rate);

Player selectedItem = (Player) listViewPlayers.getSelectionModel().getSelectedItem();

Thread.sleep(5000);

Platform.runLater(() ->PlayerData.getInstance().deletePlayer(selectedItem));

} catch (Exception e) {

e.printStackTrace();

}

//String name = FplayerName.getText();

// System.out.println(name);

// System.out.println(takenByTeam);

// timer.cancel();

}

}

}, 1000,1000);

}

public void nonDisableEverything(){

bidWithText.setVisible(true);

cskID.setDisable(false);

ddID.setDisable(false);

kkrID.setDisable(false);

kxipID.setDisable(false);

miID.setDisable(false);

rrID.setDisable(false);

rcbID.setDisable(false);

srhID.setDisable(false);

}

public void disableEverything(){

cskID.setDisable(true);

ddID.setDisable(true);

kkrID.setDisable(true);

kxipID.setDisable(true);

miID.setDisable(true);

rrID.setDisable(true);

rcbID.setDisable(true);

srhID.setDisable(true);

}

public void incrementBid(){

if(tempBidRate < 100){

tempBidRate+=5;

}else if(tempBidRate >= 100 && tempBidRate < 200){

tempBidRate+=10;

}else if(tempBidRate >=200 && tempBidRate < 500){

tempBidRate+=20;

}else{

tempBidRate+=25;

}

String mediumTwo = Integer.toString(tempBidRate);

bidTemp.setText(mediumTwo);

}

public void outOfPurse(int inputBidRate){

if(purse.purseCSK < inputBidRate){

cskID.setDisable(true);

}

if(purse.purseDD < inputBidRate){

ddID.setDisable(true);

}

if(purse.purseKKR < inputBidRate){

kkrID.setDisable(true);

}

if(purse.purseKXIP < inputBidRate){

kxipID.setDisable(true);

}

if(purse.purseMI < inputBidRate){

miID.setDisable(true);

}

if(purse.purseRR < inputBidRate){

rrID.setDisable(true);

}

if(purse.purseRCB < inputBidRate){

rcbID.setDisable(true);

}

if(purse.purseSRH < inputBidRate){

srhID.setDisable(true);

}

}

public void setPrice(){

FcskPurse.setText(Double.toString((double)purse.purseCSK/100)+" Cr");

FddPurse.setText(Double.toString((double)purse.purseDD/100)+" Cr");

FkxipPurse.setText(Double.toString((double)purse.purseKXIP/100)+" Cr");

FkkrPurse.setText(Double.toString((double)purse.purseKKR/100)+" Cr");

FmiPurse.setText(Double.toString((double)purse.purseMI/100)+" Cr");

FrcbPurse.setText(Double.toString((double)purse.purseRCB/100)+" Cr");

FrrPurse.setText(Double.toString((double)purse.purseRR/100)+" Cr");

FsrhPurse.setText(Double.toString((double)purse.purseSRH/100)+" Cr");

}

private String sql;

public void loadPlayerData() {

try{

Connection conn = dbConnection.getConnection();

this.data = FXCollections.observableArrayList();

ResultSet rs = conn.createStatement().executeQuery(sql);

while(rs.next()){

this.data.add(new PlayerAuctionData(rs.getString(1),rs.getInt(2),rs.getString(3),rs.getInt(4),rs.getString(5),rs.getString(6)));

}

}catch (SQLException e1){

System.out.println(e1);

}

this.playernamecolumn.setCellValueFactory(new PropertyValueFactory<PlayerAuctionData,String>("playerNameDB"));

this.playeragecolumn.setCellValueFactory(new PropertyValueFactory<PlayerAuctionData,Integer>("playerAgeDB"));

this.playerteamcolumn.setCellValueFactory(new PropertyValueFactory<PlayerAuctionData,String>("playerTeamDB"));

this.playerpricecolumn.setCellValueFactory(new PropertyValueFactory<PlayerAuctionData,Integer>("playerPriceDB"));

this.playerrolecolumn.setCellValueFactory(new PropertyValueFactory<PlayerAuctionData,String>("playerRoleDB"));

this.playercountrycolumn.setCellValueFactory(new PropertyValueFactory<PlayerAuctionData,String>("playerCountryDB"));

this.playerauctiontable.setItems(null);

this.playerauctiontable.setItems(this.data);

}

@FXML

public void pressedQuery() throws SQLException{

sql = "SELECT \* FROM AuctionData ORDER BY price DESC";

loadPlayerData();

}

@FXML

public void pressedCSKQuery() throws SQLException{

sql = "SELECT \* FROM AuctionData WHERE team='CSK'";

loadPlayerData();

}

@FXML

public void pressedDDQuery() throws SQLException{

sql = "SELECT \* FROM AuctionData WHERE team='DD'";

loadPlayerData();

}

@FXML

public void pressedKKRQuery() throws SQLException{

sql = "SELECT \* FROM AuctionData WHERE team='KKR'";

loadPlayerData();

}

@FXML

public void pressedKXIPQuery() throws SQLException{

sql = "SELECT \* FROM AuctionData WHERE team='KXIP'";

loadPlayerData();

}

@FXML

public void pressedMIQuery() throws SQLException{

sql = "SELECT \* FROM AuctionData WHERE team='MI'";

loadPlayerData();

}

@FXML

public void pressedRCBQuery() throws SQLException{

sql = "SELECT \* FROM AuctionData WHERE team='RCB'";

loadPlayerData();

}

@FXML

public void pressedRRQuery() throws SQLException{

sql = "SELECT \* FROM AuctionData WHERE team='RR'";

loadPlayerData();

}

@FXML

public void pressedSRHQuery() throws SQLException{

sql = "SELECT \* FROM AuctionData WHERE team='SRH'";

loadPlayerData();

}

@FXML

public void pressedIndianQuery() throws SQLException{

sql = "SELECT \* FROM AuctionData WHERE country='India' OR country = 'INDIA' OR country='india'";

loadPlayerData();

}

@FXML

public void pressedOverseasQuery() throws SQLException{

sql = "SELECT \* FROM AuctionData WHERE country NOT LIKE '%India%'";

loadPlayerData();

}

@FXML

public void pressedUnsoldQuery() throws SQLException{

sql = "SELECT \* FROM AuctionData WHERE team='UNSOLD'";

loadPlayerData();

}

}