package com.example.itsspotifylove;  
  
import com.google.gson.JsonArray;  
import com.google.gson.JsonElement;  
import com.google.gson.JsonObject;  
import com.google.gson.JsonParser;  
import javafx.application.Application;  
import javafx.beans.property.SimpleStringProperty;  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.concurrent.Task;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.image.Image;  
import javafx.scene.image.ImageView;  
import javafx.scene.input.KeyCode;  
import javafx.scene.layout.BorderPane;  
import javafx.scene.layout.GridPane;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.VBox;  
import javafx.scene.media.Media;  
import javafx.scene.media.MediaPlayer;  
import javafx.scene.media.MediaView;  
import javafx.scene.text.Font;  
import javafx.stage.Stage;  
  
import java.io.\*;  
import java.net.HttpURLConnection;  
import java.net.URL;  
import java.net.URLConnection;  
import java.net.URLEncoder;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Map;  
import java.util.Scanner;  
import java.util.stream.Collectors;  
  
public class MusicDownloaderApp extends Application {  
  
 private static final String *APIKEY* = "AIzaSyBzVE5k6xoxz6\_56f5Mj-t8PAnZL8Y5Q2Q";  
 private TableView<String> resultsTable;  
 private TextField searchField;  
 private List<String> videoIds;  
 private ObservableList<String> downloadedSongs;  
 private VBox sidebar;  
 private boolean sidebarVisible = true;  
 private ComboBox<String> formatComboBox;  
 // private static final String DOWNLOAD\_FOLDER = "downloads/%(title)s.%(ext)s";  
 private final ArrayList<String[]> credentials = new ArrayList<>();  
  
 private final File credentialsFile = new File("credentials.txt");  
  
 public void start(Stage primaryStage) {  
  
 loadCredentialsFromFile();  
 BorderPane bp = new BorderPane();  
  
 Image image = new Image("file:D:\\Dawood\\Books\\COMsats\\2nd sem (Temporary ab say )\\OOP\\GUI\\itsspotifylove22\\itsspotifylove\\image.png"); // Update path if necessary  
 ImageView imageView = new ImageView(image);  
 imageView.setFitWidth(370);  
 imageView.setFitHeight(364);  
 imageView.setPreserveRatio(false);  
  
 bp.setTop(imageView);  
  
 GridPane gridPane = new GridPane();  
 gridPane.setAlignment(Pos.*CENTER*);  
 gridPane.setHgap(10);  
 gridPane.setVgap(10);  
  
  
  
 Label usernameLabel = new Label("Username:");  
 usernameLabel.setFont(new Font("Arial", 25));  
 usernameLabel.setStyle(" -fx-text-fill:#FFD700 ; -fx-font-weight: bold;");  
 TextField usernameField = new TextField();  
  
 Label passwordLabel = new Label("Password:");  
 passwordLabel.setStyle(" -fx-text-fill:#FFD700 ; -fx-font-weight: bold;");  
 passwordLabel.setFont(new Font("Arial", 25));  
 PasswordField passwordField = new PasswordField();  
  
  
 Button loginButton = new Button("Login");  
 loginButton.setStyle("-fx-background-color: #FFD700; -fx-text-fill: black; -fx-font-weight: bold ;");  
 loginButton.setPrefHeight(300);  
 loginButton.setPrefWidth(150);  
  
 Button saveButton = new Button("Sign up");  
 saveButton.setStyle("-fx-background-color: #FFD700; -fx-text-fill: black; -fx-font-weight: bold;");  
 saveButton.setPrefHeight(300);  
 saveButton.setPrefWidth(100);  
  
 Button exitButton = new Button("Exit");  
 exitButton.setStyle("-fx-background-color: #FF0000; -fx-text-fill: black; -fx-font-weight: bold;");  
 exitButton.setPrefHeight(300);  
 exitButton.setPrefWidth(100);  
  
 Label notificationLabel = new Label();  
 gridPane.add(usernameLabel, 0, 0);  
 gridPane.add(usernameField, 1, 0);  
 gridPane.add(passwordLabel, 0, 1);  
 gridPane.add(passwordField, 1, 1);  
  
  
  
 GridPane buttonPane = new GridPane();  
 buttonPane.setHgap(30);  
 buttonPane.setVgap(50);  
 buttonPane.add(loginButton, 0, 2);  
 buttonPane.add(saveButton, 1, 2);  
 buttonPane.add(exitButton, 2, 2);  
 gridPane.add(buttonPane, 0, 2, 2, 1);  
  
  
 gridPane.add(notificationLabel, 0, 3, 2, 1);  
 notificationLabel.setAlignment(Pos.*CENTER*);  
  
  
  
 bp.setCenter(gridPane);  
 bp.setStyle("-fx-background-color: #000000;");  
  
  
  
  
 saveButton.setOnAction(e -> {  
 String username = usernameField.getText();  
 String password = passwordField.getText();  
 if (!username.isEmpty() && !password.isEmpty()) {  
 saveCredentialsToFile(username, password);  
 notificationLabel.setText("Credentials saved successfully!");  
 } else {  
 notificationLabel.setText("Username and password cannot be empty!");  
 }  
 });  
 loginButton.setOnAction(e -> {  
 String username = usernameField.getText();  
 String password = passwordField.getText();  
  
 if (validateCredentials(username, password)) {  
 notificationLabel.setText("Login successful!");  
  
 openMain(primaryStage);  
  
 } else {  
 notificationLabel.setText("Invalid username or password!");  
 }  
 });  
  
 exitButton.setOnAction(e -> primaryStage.close());  
  
 Scene scene = new Scene(bp, 800, 600);  
  
 primaryStage.setTitle("Login Windowww");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
  
 }  
 private void loadCredentialsFromFile() {  
 try {  
 if (!credentialsFile.exists()) {  
 credentialsFile.createNewFile();  
 }  
  
  
 Scanner scanner = new Scanner(credentialsFile);  
 while (scanner.hasNextLine()) {  
 String line = scanner.nextLine();  
 String[] parts = line.split(",");  
 if (parts.length == 2) {  
 credentials.add(parts);  
 }  
 }  
 scanner.close();  
 } catch (IOException e) {  
 System.*out*.println("Error loading credentials: " + e.getMessage());  
 }  
 }  
  
 private boolean validateCredentials(String username, String password) {  
  
 for (String[] pair : credentials) {  
 if (pair[0].equals(username) && pair[1].equals(password)) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 private void saveCredentialsToFile(String username, String password) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(credentialsFile, true))) {  
 writer.write(username + "," + password);  
 writer.newLine();  
 credentials.add(new String[]{username, password}); // Update in-memory list as well  
 } catch (IOException e) {  
 System.*out*.println("Error saving credentials: " + e.getMessage());  
 }  
 }  
  
  
  
  
 public void openMain(Stage stage) {  
 BorderPane root = new BorderPane();  
 root.setStyle("-fx-background-color: #121212;");  
 HBox topBar = new HBox(10);  
 topBar.setPadding(new Insets(15));  
 topBar.setStyle("-fx-background-color: #333333;");  
  
 searchField = new TextField();  
 searchField.setPromptText("Enter song name...");  
 searchField.setStyle("-fx-background-color: #222222; -fx-text-fill: white; -fx-prompt-text-fill: gray;");  
 searchField.setPrefWidth(450);  
  
  
 Button searchButton = new Button("Search");  
 searchButton.setStyle("-fx-background-color: #FFD700; -fx-text-fill: black; -fx-font-weight: bold;");  
 searchButton.setOnAction(e -> searchSongs());  
 searchField.setOnKeyPressed(e -> {  
 if (e.getCode() == KeyCode.*ENTER*) {  
 searchSongs();  
 }  
 });  
  
 topBar.getChildren().addAll(searchField, searchButton);  
  
  
 resultsTable = new TableView<>();  
 resultsTable.setPlaceholder(new Label("No results for now."));  
 resultsTable.setStyle("-fx-background-color: white; -fx-text-fill: white; -fx-font-size: 14px;");  
  
 TableColumn<String, String> titleColumn = new TableColumn<>("Searched Results:");  
 titleColumn.setCellValueFactory(data -> new SimpleStringProperty(data.getValue()));  
 titleColumn.setPrefWidth(550);  
  
 resultsTable.getColumns().add(titleColumn);  
  
  
 sidebar = new VBox(10);  
 sidebar.setPadding(new Insets(15));  
 sidebar.setStyle("-fx-background-color: #282828; -fx-border-color: #FFD700;");  
 sidebar.setPrefWidth(350);  
  
 Label sidebarTitle = new Label("Downloaded Songs");  
 sidebarTitle.setStyle("-fx-font-size: 16px; -fx-text-fill: #FFD700; -fx-font-weight: bold;");  
  
 ListView<String> downloadedList = new ListView<>();  
  
 downloadedSongs = FXCollections.*observableArrayList*(); // observabke list  
 downloadedList.setItems(downloadedSongs);  
  
 downloadedList.setOnMouseClicked(event -> {  
 if (event.getClickCount() == 2) {  
 String selectedFilePath = downloadedList.getSelectionModel().getSelectedItem();  
 if (selectedFilePath != null) {  
  
 File file = new File(selectedFilePath.replace("file:///", ""));  
 if (file.exists()) {  
  
 Media media = new Media(file.toURI().toString());  
 MediaPlayer mediaPlayer = new MediaPlayer(media);  
 MediaView mediaView = new MediaView(mediaPlayer);  
  
  
  
 Stage playerStage = new Stage();  
 playerStage.setTitle("Playing: " + file.getName());  
  
 BorderPane playerPane = new BorderPane();  
 playerPane.setCenter(mediaView);  
 playerPane.setStyle("-fx-background-color: black;");  
  
 Scene playerScene = new Scene(playerPane, 800, 500);  
 playerStage.setScene(playerScene);  
 playerStage.show();  
 mediaPlayer.play();  
  
 playerStage.setOnCloseRequest(e -> mediaPlayer.stop());  
 } else {  
 showAlert("Error", "Kuch Masla hai with this file " + file.getAbsolutePath(), Alert.AlertType.*ERROR*);  
 }  
 }  
 }  
 });  
  
  
  
 sidebar.getChildren().addAll(sidebarTitle, downloadedList);  
 sidebar.setVisible(false);  
  
  
 Button toggleSidebarButton = new Button("Downloads");  
 toggleSidebarButton.setStyle("-fx-background-color: #FFD700; -fx-text-fill: black; -fx-font-weight: bold;");  
 toggleSidebarButton.setPrefWidth(130);  
 toggleSidebarButton.setOnAction(e -> {  
 sidebarVisible = !sidebarVisible;  
 sidebar.setVisible(sidebarVisible);  
 });  
  
 formatComboBox = new ComboBox<>();  
 formatComboBox.getItems().addAll("MP3 (Audio)", "MP4 (Video)", "WAV");  
 formatComboBox.setValue("MP3 (Audio)");  
 formatComboBox.setStyle("-fx-background-color: #FFD700; -fx-text-fill: black; -fx-font-weight: bold;");  
 formatComboBox.setPrefWidth(150);  
  
 Button downloadButton = new Button("Download Selected");  
 downloadButton.setStyle("-fx-background-color: #FFD700; -fx-text-fill: black; -fx-font-weight: bold;");  
 downloadButton.setOnAction(e -> downloadSelectedSong());  
  
 Button exitButton = new Button("Exit");  
 exitButton.setStyle("-fx-background-color: #FF0000; -fx-text-fill: black; -fx-font-weight: bold;");  
 exitButton.setOnAction(e -> stage.close());  
  
  
  
  
 HBox bottomBar = new HBox(10, toggleSidebarButton, formatComboBox, downloadButton, exitButton);  
 bottomBar.setPadding(new Insets(15));  
 bottomBar.setStyle("-fx-background-color: #333333;");  
  
  
  
  
  
  
 // Set layout sections  
 root.setTop(topBar);  
 root.setCenter(resultsTable);  
 root.setBottom(bottomBar);  
 root.setLeft(sidebar);  
  
 loadDownloadedSongs();  
 // Scene and stage setup  
 Scene scene = new Scene(root, 1000, 700);  
 stage.setTitle("Music Downloader");  
 stage.setScene(scene);  
 stage.show();  
 }  
  
 private void loadDownloadedSongs() {  
 // Get all files in the download folder  
 File downloadDir = new File("downloads");  
  
 // Filter out non-media files  
 File[] files = downloadDir.listFiles((dir, name) -> name.endsWith(".mp3") || name.endsWith(".mp4") || name.endsWith(".wav"));  
 if (files != null) {  
 for (File file : files) {  
 downloadedSongs.add("file:///" + file.getAbsolutePath());  
 }  
 }  
 }  
  
 private void searchSongs() {  
 String search = searchField.getText().trim();  
 if (search.isEmpty()) {  
 showAlert("Error", "Search box cannot be empty.", Alert.AlertType.*ERROR*);  
 return;  
 }  
  
 try {  
 String searchUrl = "https://www.googleapis.com/youtube/v3/search?part=snippet&type=video&maxResults=50&q="  
 + URLEncoder.*encode*(search, "UTF-8") + "&key=" + *APIKEY*;  
  
 URL url = new URL(searchUrl);  
 URLConnection urlConnection = url.openConnection();  
  
 HttpURLConnection connection = (HttpURLConnection) urlConnection;  
 connection.setRequestMethod("GET");  
  
 InputStreamReader isr= new InputStreamReader(connection.getInputStream()); //getting the binary data  
 BufferedReader in = new BufferedReader(isr); //converting binary into characters  
 String response = in.lines().collect(Collectors.*joining*()); //ye collecting all character into a single string  
 //this response is actually wo wala JASON shapar, it looks like that screenshot  
 in.close();  
  
 List<String> results = parseYouTubeResponse(response);//wo jo items wali list return ho rhi hai, wo result wali list mn store ho rhi hai  
  
  
  
 resultsTable.setItems(FXCollections.*observableArrayList*(results));  
  
 } catch (Exception e) {  
 showAlert("Error", "Failed to fetch results: " + e.getMessage(), Alert.AlertType.*ERROR*);  
 }  
 }  
  
 private List<String> parseYouTubeResponse(String jsonResponse) { // method is returning a string type ki list.  
 videoIds = new ArrayList<>();  
 List<String> titles = new ArrayList<>();// yehi to hmara function return kray ga  
  
 JsonObject responseObject = JsonParser.*parseString*(jsonResponse).getAsJsonObject();  
 if (!responseObject.has("items")) {  
 return titles;  
 }  
  
// ab us jason object mn say we getting ITEMS wali array. jis mn objects of VIDEO ID and Title/discription etc sab kuchhhh!!.  
 JsonArray items = responseObject.getAsJsonArray("items"); //items has the JASON type ki array  
 for (JsonElement item : items) {//JasonEelement is generic, it can store jason type arrays objects etc  
  
  
 // this converts item array into item object  
 JsonObject itemObject = item.getAsJsonObject();  
  
 // now finally we can extract the string type data from the jason objects.  
  
 String videoId = itemObject.getAsJsonObject("id").get("videoId").getAsString();  
 String title = itemObject.getAsJsonObject("snippet").get("title").getAsString();  
 videoIds.add(videoId);  
 titles.add(title);  
  
 }  
  
 // we get raw JASON string  
 // we convert that raw JASON string into Jason object  
 // then we convert that JASON object to JASON type Array  
 // us JASON type array k andar hain mazeeeeeedd Jason type k objects i.e videoID and title  
 // we convert those JASON type k objects i.e videID and title into Strings.  
  
  
 return titles;  
 }  
  
  
  
  
  
 private void downloadSelectedSong() {  
 String selectedSong = resultsTable.getSelectionModel().getSelectedItem();  
 int selectedIndex = resultsTable.getSelectionModel().getSelectedIndex();  
  
 String videoId = videoIds.get(selectedIndex); // from YouTube parse function  
 String selectedFormat = formatComboBox.getValue(); // combo box mn say selected format so we can match  
  
 String formatwalichez; // -x is for audio file jesy mp3 and wav and -f is for mp4  
 String formatType;  
 if ("MP3 (Audio)".equals(selectedFormat)) {  
 formatwalichez = "-x"; // Extract audio  
 formatType = "--audio-format=mp3";  
 } else if ("MP4 (Video)".equals(selectedFormat)) {  
 formatwalichez = "-f"; // Download video  
 formatType = "mp4";  
 } else if ("WAV".equals(selectedFormat)) {  
 formatwalichez = "-x"; // Extract audio  
 formatType = "--audio-format=wav";  
 } else {  
 return;  
 }  
  
 Task<Void> downloadTask = new Task<>() {  
 @Override  
 protected Void call() throws Exception { // A Task is a class in JavaFX designed to perform operations on a background thread, avoiding blocking the JavaFX application thread (UI thread).  
 // Construct the output file path dynamically  
 String outputFilePath = "downloads/" + selectedSong + "." + selectedFormat.split(" ")[0].toLowerCase();  
  
 ProcessBuilder pb = new ProcessBuilder( //utility class to excecute cmd commands  
 "yt-dlp",  
 formatwalichez,  
 formatType,  
 "--ffmpeg-location", "C:\\Program Files (x86)\\ffmpeg-7.1-full\_build\\bin\\ffmpeg.exe",  
 "https://www.youtube.com/watch?v=" + videoId,  
 "-o", outputFilePath  
 );  
  
 Map<String, String> env = pb.environment();  
 String currentPath = env.get("PATH");  
 env.put("PATH", currentPath + ";C:\\Program Files (x86)\\yt-dlp");  
  
 pb.start().waitFor();  
 return null;  
 }  
 };  
  
 downloadTask.setOnSucceeded(e -> {  
  
 downloadedSongs.add("file:///" + new File("downloads/" + selectedSong + "." + selectedFormat.split(" ")[0].toLowerCase()).getAbsolutePath());  
 showAlert("Success", "Downloaded: " + selectedSong + " as " + selectedFormat, Alert.AlertType.*INFORMATION*);  
 });  
  
  
 new Thread(downloadTask).start();  
 }  
  
  
 private void showAlert(String title, String message, Alert.AlertType alertType) {  
 Alert alert = new Alert(alertType);  
 alert.setTitle(title);  
 alert.setHeaderText(null);  
 alert.setContentText(message);  
 alert.showAndWait();  
 }  
  
 public static void main(String[] args) {  
 *launch*();  
 }  
}