FACULTY OF TECHNOLOGY AND ENGINEERING

DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

FS BE-II CSE

OBJECT ORIENTED PROGRAMMING WITH JAVA
(CSE1304CS)

SurSagar

SUBMITTED BY:-

Sumit Khapre(386246) - 8022055725
Rudra Manger(386248) - 8025064039
Pratham Pandya(386253) - 8025064048
Krish Patel(386255) - 8022054486
Khushi Solanki(386261) - 8025064118
Niyati Thavrani(386263) - 8025064119

Date Of Submission:- 28th November’ 2025

ACKNOWLEDGEMENT

We wholeheartedly thank our guide Prof. Snehal Patel, faculty of Computer Science and Engineering. For being source of motivation, constant supervision and for guiding us with perfection in doing all the work. We are also very thankful to all our Teachers and Friends who have given Valuable suggestions and help in all stages of the development of the project. And I would like to mention here about all the online resources that we used for learning and error solving, which helps a lot in developing the project. I would like to express my sincere gratitude to everyone who supported me throughout this journey. Special thanks to my family and friends for their unwavering encouragement, and to my mentors and peers for their valuable guidance and insights. Your support has been instrumental in helping me reach this milestone. Finally, we would like to dedicate this work to our teachers who have provided support and encouragement during every part of our life.

ABSTRACT

The **SurSagar** is a lightweight desktop application developed using Java and the Swing framework to provide personalized and efficient music discovery for users.
The system enables users to explore and play music based on their preferred **artists**
and **genres**, while also maintaining a seamless and interactive interface. It integrates recommendation logic with user-specific features such as **Favourites** and **Searching History**, ensuring that the application adapts to user behaviour and preferences over time.

Users can search for music by entering an artist name or genre, upon which the system generates relevant suggestions through its recommendation engine. When a
song is selected for playback, the application redirects the user to **YouTube**, allowing instant access to online streaming without requiring large local media storage.

Additionally, the system manages user-added favourite songs and automatically records playback history, enabling quick access to frequently played tracks. The project demonstrates practical implementation of Java Swing for GUI development, efficient data handling, and the integration of external platforms for media play-
back. Overall, the application provides a simple, user-friendly solution for personalized music browsing and recommendation.

FUNCTIONALITY BREAKDOWN

1. 🔍 Login Panel (MainApp.java)

The Login Panel is the application's entry point, designed to restrict access and transition the user to the main application view.

- Input Username/Password:

The login form uses a standard Swing setup with a JTextField for the username and a JPasswordField for secure password entry. The panel employs GridBagLayout to organize fields and labels neatly, ensuring a professional appearance.

- Attempt Login:-

Clicking the "Login" button triggers an ActionListener that retrieves the input strings from the fields. The button itself is styled using the centralized Utils.styleButton() method for visual consistency.

- Authentication Logic (Simulation):-

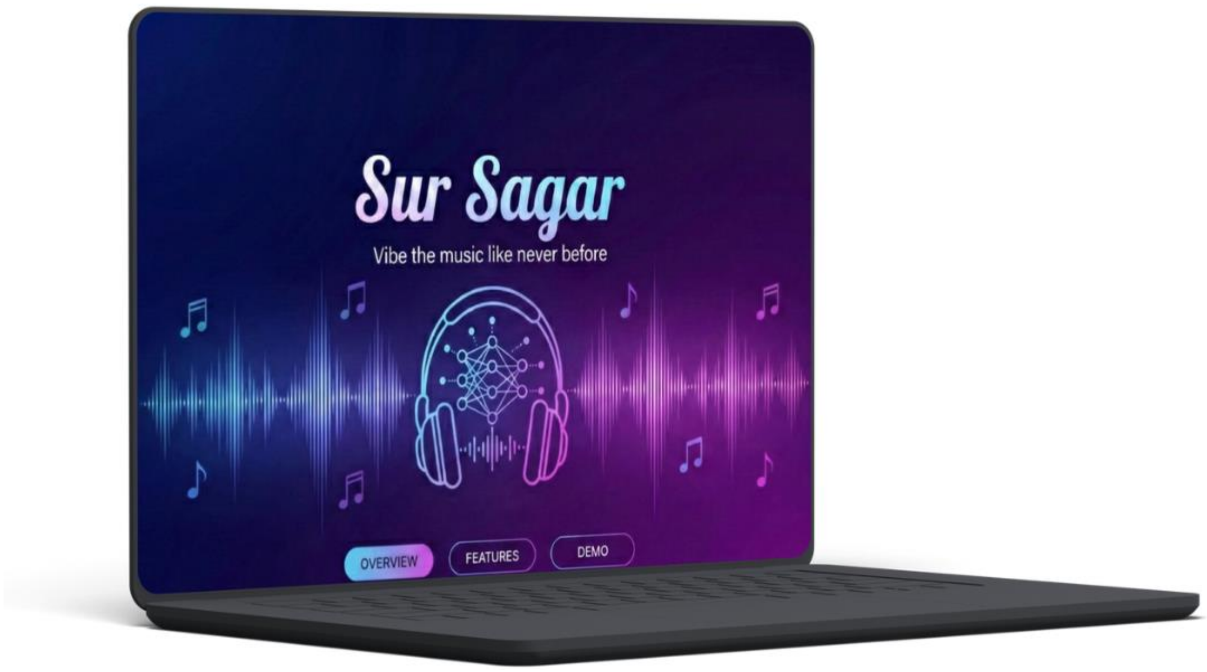
The current implementation simulates a successful authentication if both fields are non-empty. If credentials fail this basic check, a JOptionPane error message is displayed to the user.

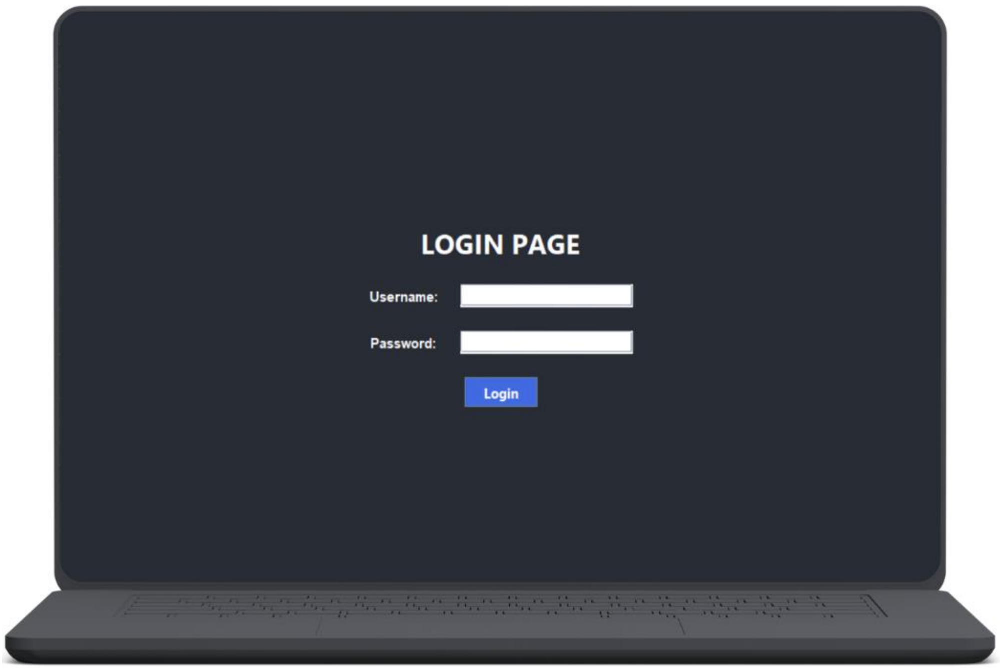
- Set Current User:-

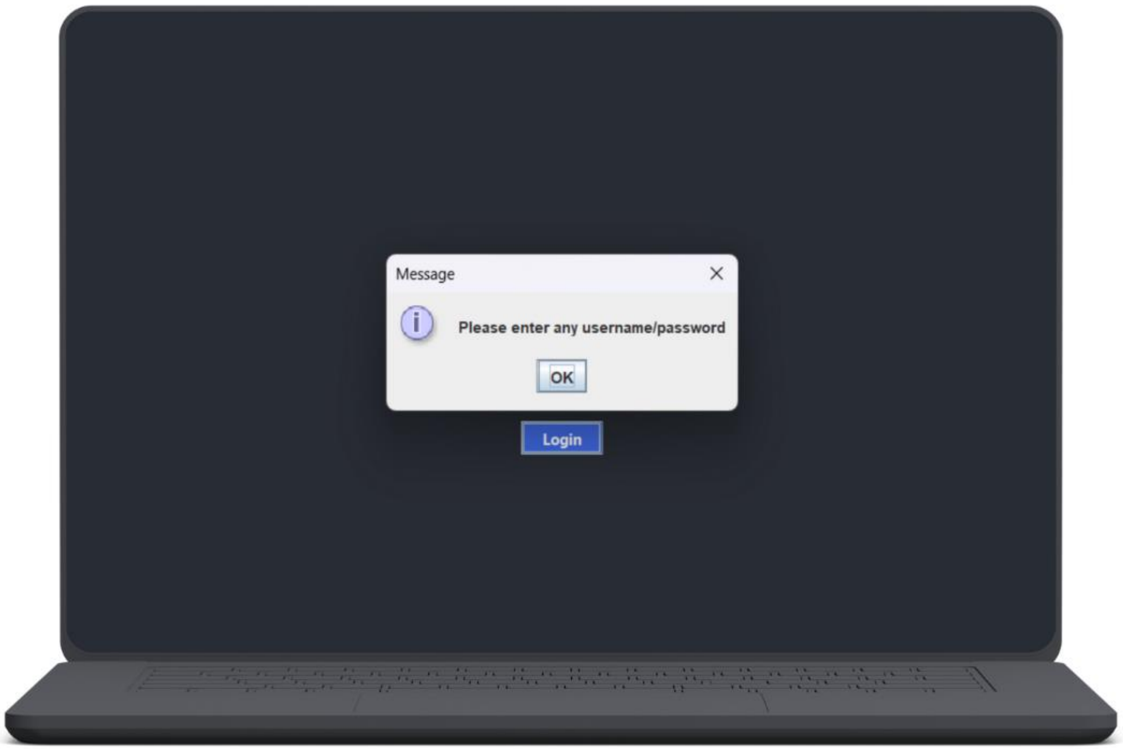
Upon successful login, the entered username is captured and passed to the dashboard.setCurrentUser(user) method. This username is crucial for logging specific user activities in the History feature.

- Screen Transition:-

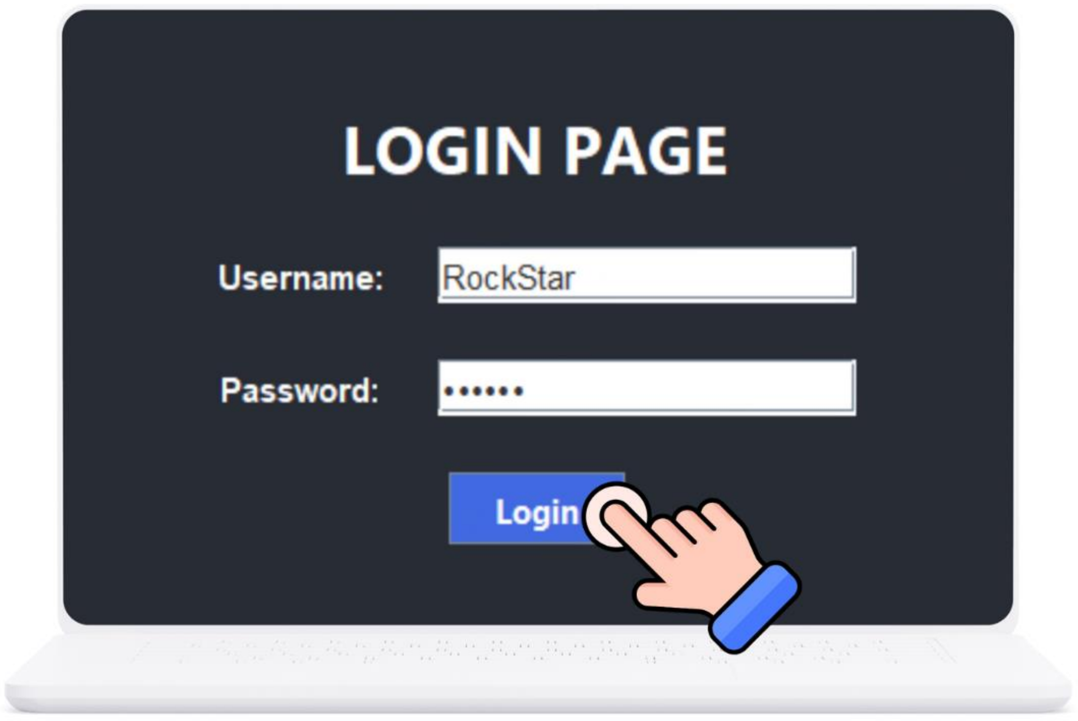
The CardLayout.show(mainPanel, "DASHBOARD") method is called. This is a core Swing technique for switching between large, distinct application views efficiently without reloading the entire window.


Sur Sagar
Vibe the music like never before
OVERVIEW
FEATURES
DEMO


LOGIN PAGE
Username:
Password:
Login


Message

Please enter any username/password


LOGIN PAGE

Username: RockStar

Password: ••••••

2. 🎵 Dashboard - Search Panel (DashboardUI.java)

This panel encapsulates the core business logic of fetching and displaying AI-driven music recommendations.

-Specify Search Criteria:-

Users can input a text string for the Artist and select a specific Mood (e.g., "Happy," "Study," "Workout") from a predefined list in the JComboBox. This dual input refines the prompt sent to the AI model.

-Initiate Search:-

Clicking the "Get Recommendations" button executes the performSearch() method. A simple validation check ensures the Artist field is not empty before proceeding.

-Update History:-

As soon as the search starts, a detailed log string (e.g., "user searched: artist (mood)") is created and saved to the history.txt file using the static DataManager.saveHistory() method.

-Asynchronous API Call (SwingWorker):-

The application utilizes a SwingWorker to perform the musicService.fetchRecommendations() call off the main UI thread. The doInBackground() method handles the lengthy network operation.

-External API Communication:-

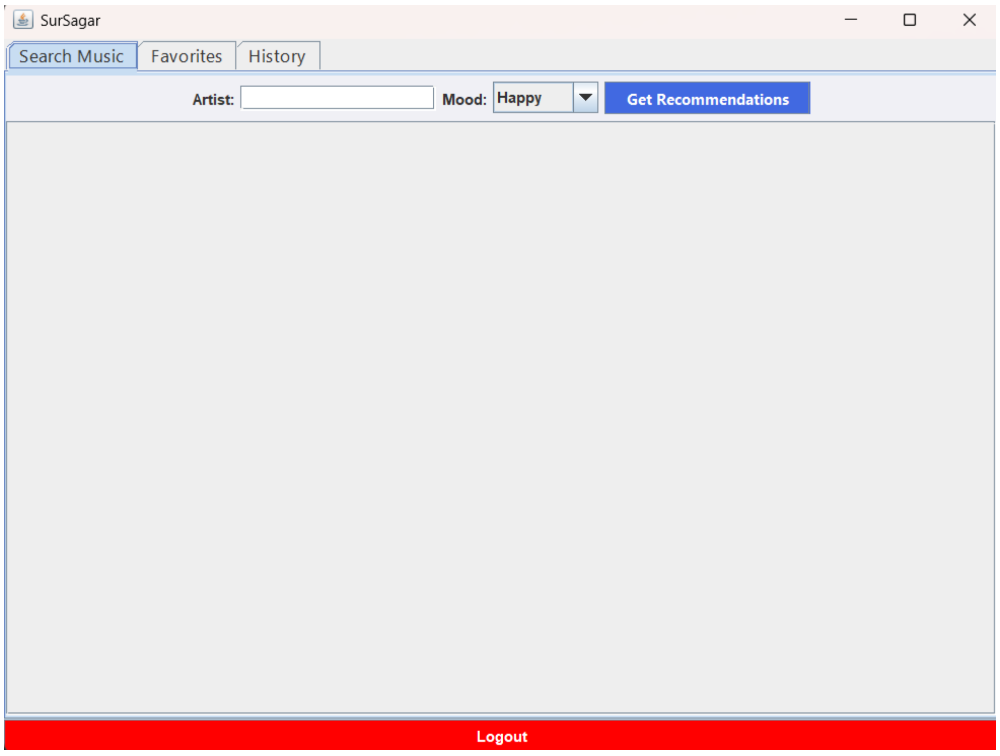
The MusicAPIService constructs a JSON payload containing the prompt and sends it to the OpenRouter/Grok API via an HttpURLConnection. It includes an API key for authorization.

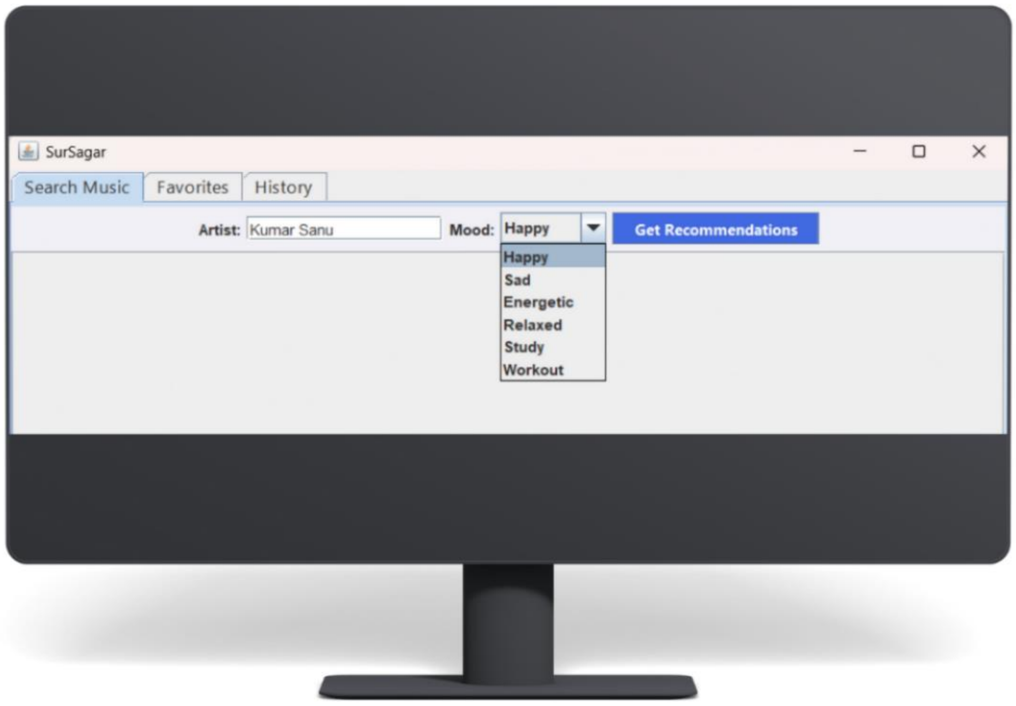
-Response Handling and Parsing:-

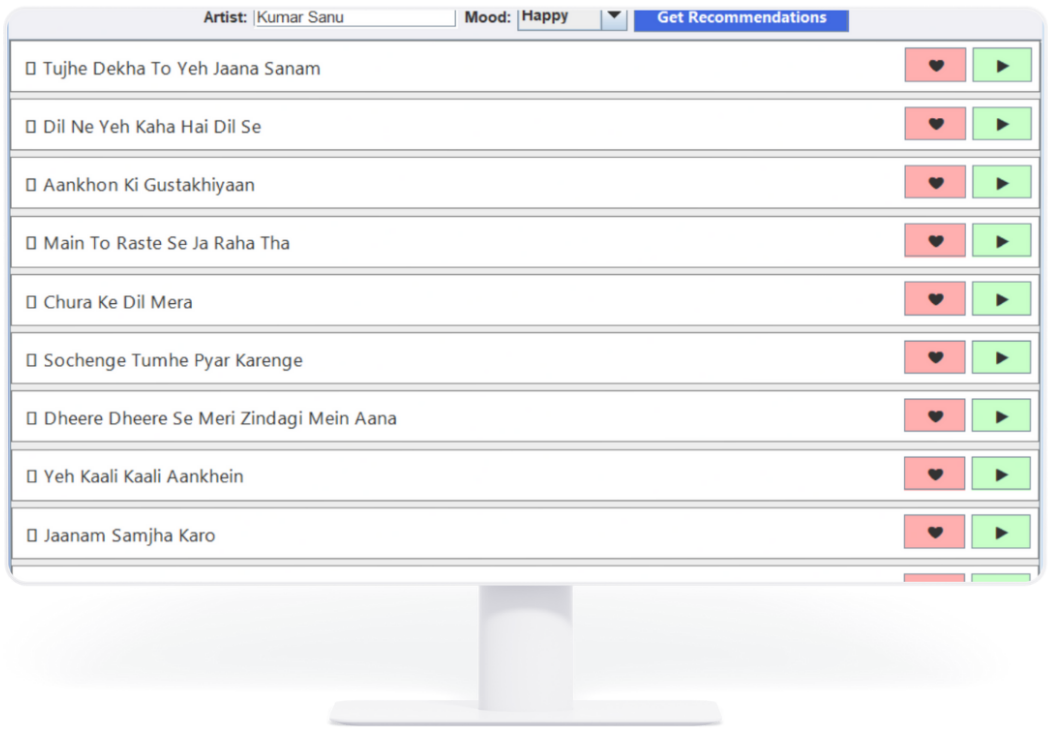
Upon receiving the response (HTTP status 200), the service manually extracts the song list from the deeply nested JSON structure. The output is processed to clean up brackets, quotes, and commas, returning a clean List<String> of song titles.

-Render Results (Custom Component):-

The SwingWorker.done() method ensures the UI update occurs on the EDT. It iterates through the received songs, creating and adding a custom ResultPanel component for each entry.

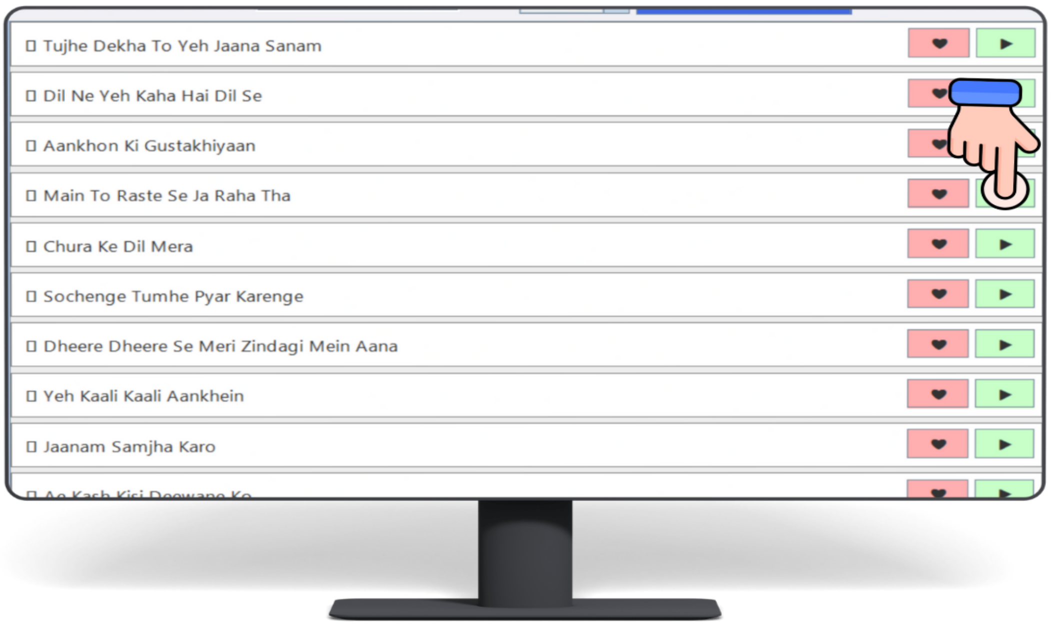




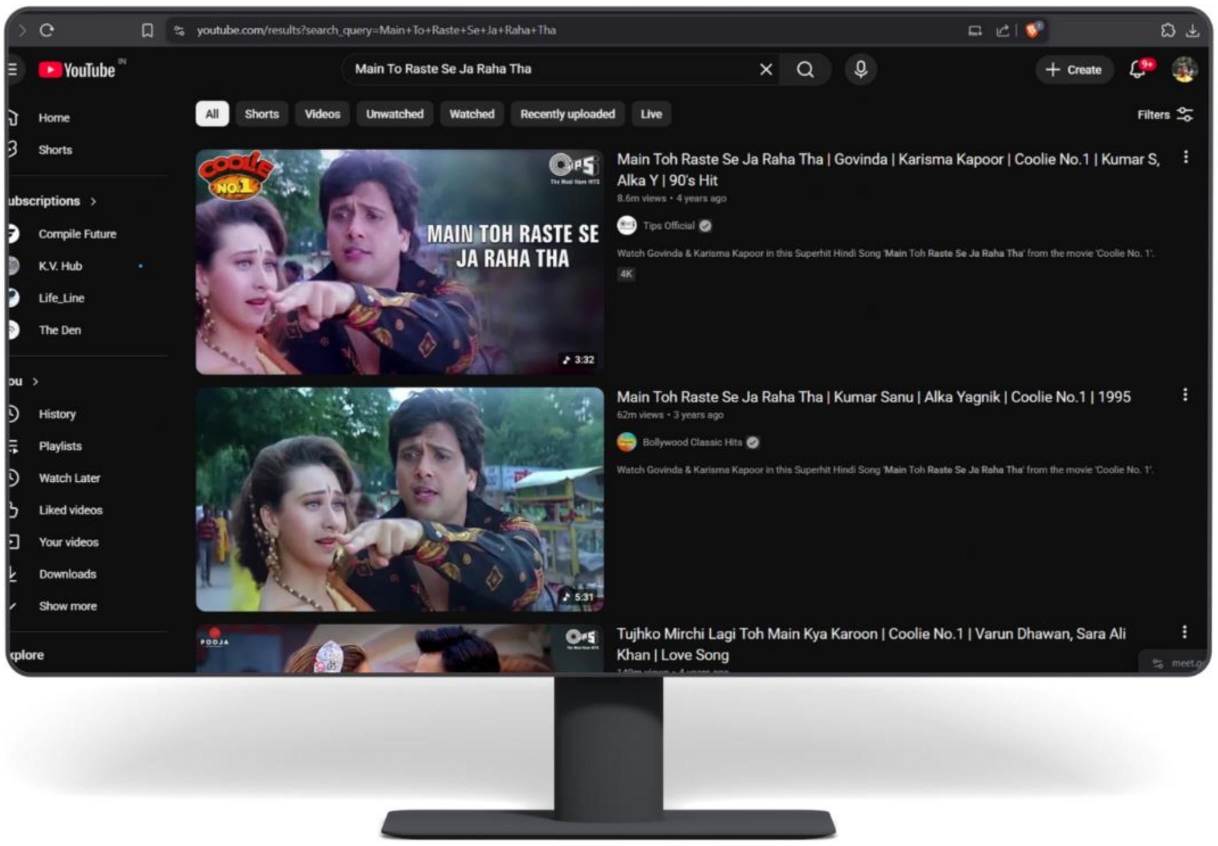


Artist: Kumar Sanu Mood: Happy Get Recommendations

▯ Tujhe Dekha To Yeh Jaana Sanam
▯ Dil Ne Yeh Kaha Hai Dil Se
▯ Aankhon Ki Gustakhiyaan
▯ Main To Raste Se Ja Raha Tha
▯ Chura Ke Dil Mera
▯ Sochenge Tumhe Pyar Karenge
▯ Dheere Dheere Se Meri Zindagi Mein Aana
▯ Yeh Kaali Kaali Aankhein
▯ Jaanam Samjha Karo



▯ Tujhe Dekha To Yeh Jaana Sanam
▯ Dil Ne Yeh Kaha Hai Dil Se
▯ Aankhon Ki Gustakhiyaan
▯ Main To Raste Se Ja Raha Tha
▯ Chura Ke Dil Mera
▯ Sochenge Tumhe Pyar Karenge
▯ Dheere Dheere Se Meri Zindagi Mein Aana
▯ Yeh Kaali Kaali Aankhein
▯ Jaanam Samjha Karo



Main Toh Raste Se Ja Raha Tha | Govinda | Karisma Kapoor | Coolie No.1 | Kumar S, Alka Y | 90's Hit
8.6m views • 4 years ago
Tips Official
Watch Govinda & Karisma Kapoor in this Superhit Hindi Song 'Main Toh Raste Se Ja Raha Tha' from the movie 'Coolie No. 1'.

Main Toh Raste Se Ja Raha Tha | Kumar Sanu | Alka Yagnik | Coolie No.1 | 1995
62m views • 3 years ago
Bollywood Classic Hits
Watch Govinda & Karisma Kapoor in this Superhit Hindi Song 'Main Toh Raste Se Ja Raha Tha' from the movie 'Coolie No. 1'.

Tujhko Mirchi Lagi Toh Main Kya Karoon | Coolie No.1 | Varun Dhawan, Sara Ali Khan | Love Song

3. ❤️ Favorites Panel (DashboardUI.java & ResultPanel.java)

This feature enables users to curate a personal list of recommended songs, integrating the search and data layers.

A. Saving a Favorite (Originates in Search Tab)

-Click Favorite Button:-

The user interacts with the custom ResultPanel (Student 5's work), clicking the "♡" button.

- Persistence Call:-

This action listener calls DataManager.saveFavorite(songName). This writes the song name to the favorites.txt file in append mode.

- Duplicate Handling:-

Before writing, the DataManager loads the existing favorites and uses the List.contains() method to check for and prevent adding a duplicate song entry, optimizing file integrity.

B. Viewing Favorites (In Favorites Tab)

-Refresh Action:-

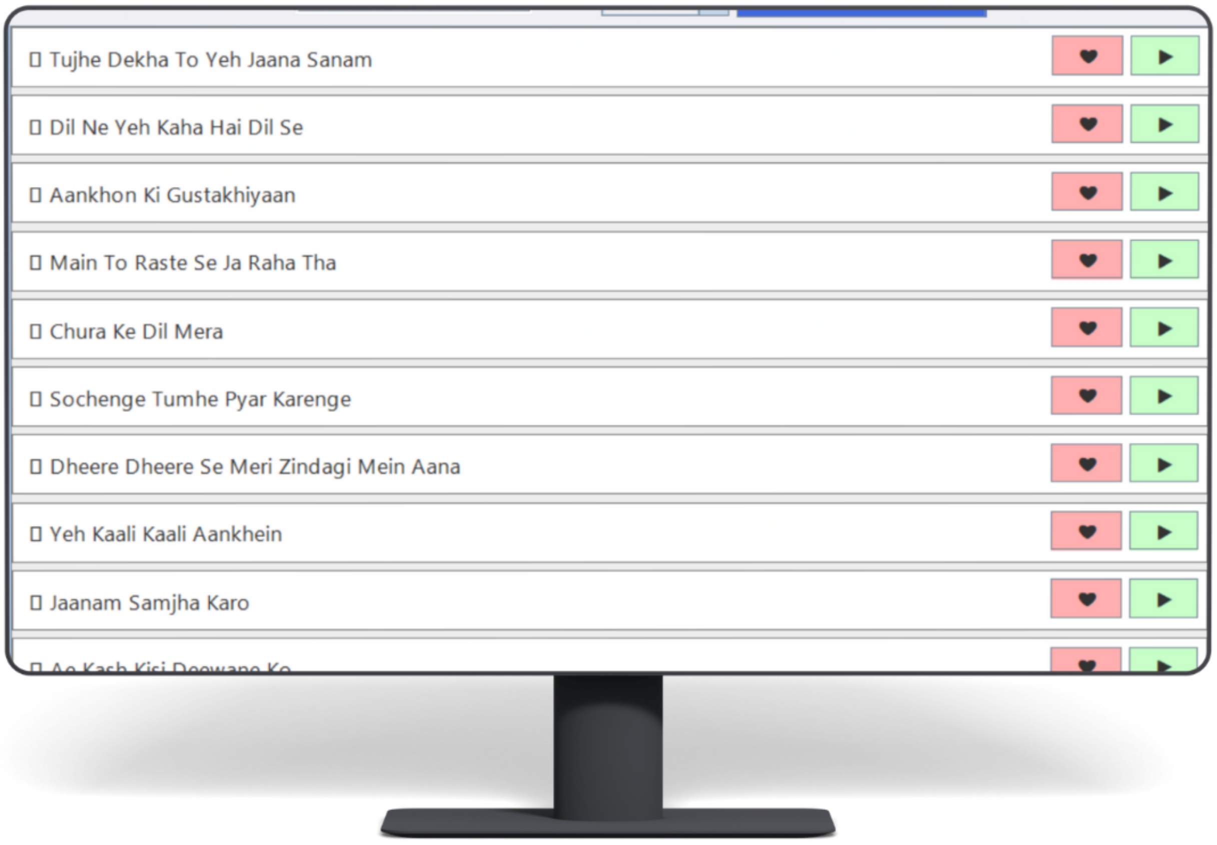
The user clicks the "Refresh Favorites" button, triggering the loadFavorites() method in DashboardUI.

-Data Retrieval:-

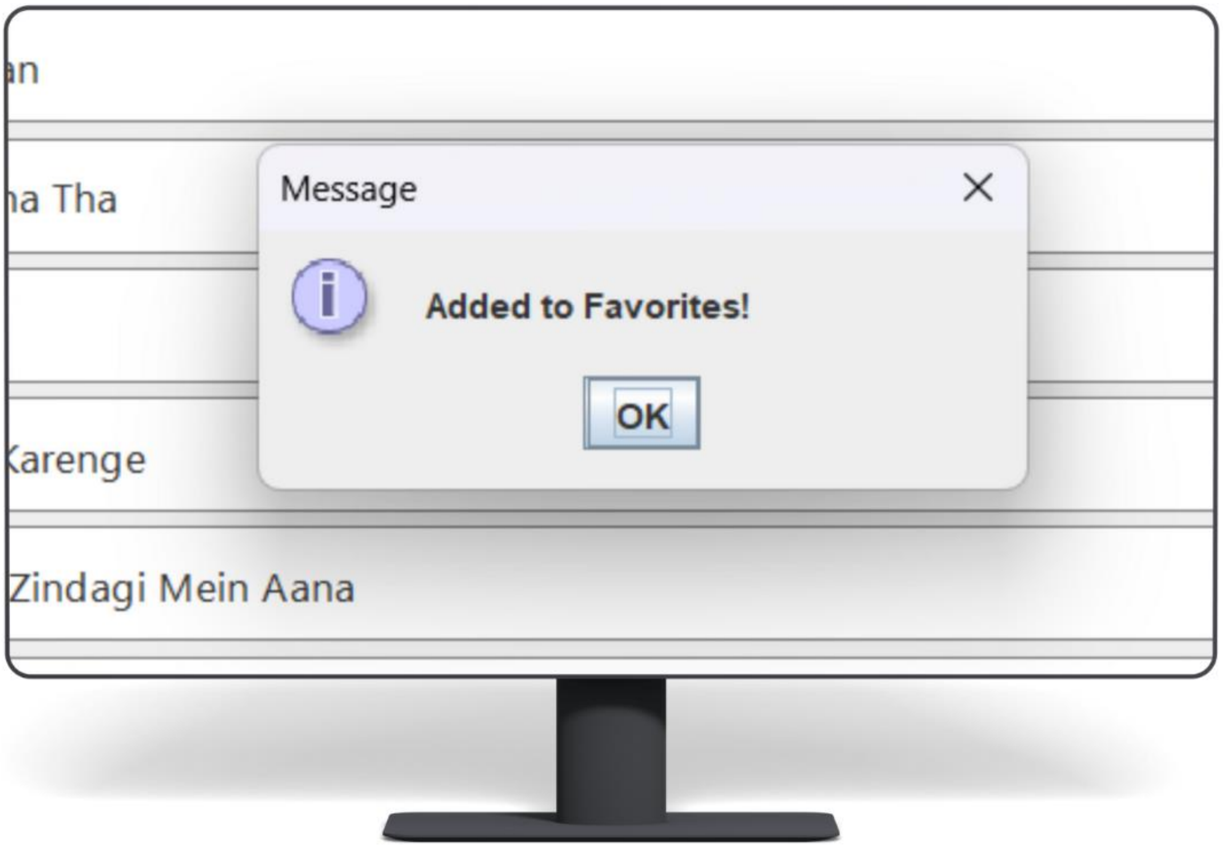
DataManager.loadFavorites() reads all lines from the favorites.txt file using a BufferedReader. This demonstrates simple file I/O for data retrieval.

-UI Population:-

The list model DefaultListModel<String> favModel is cleared and then populated with the retrieved songs, which are then displayed to the user in the JList component.

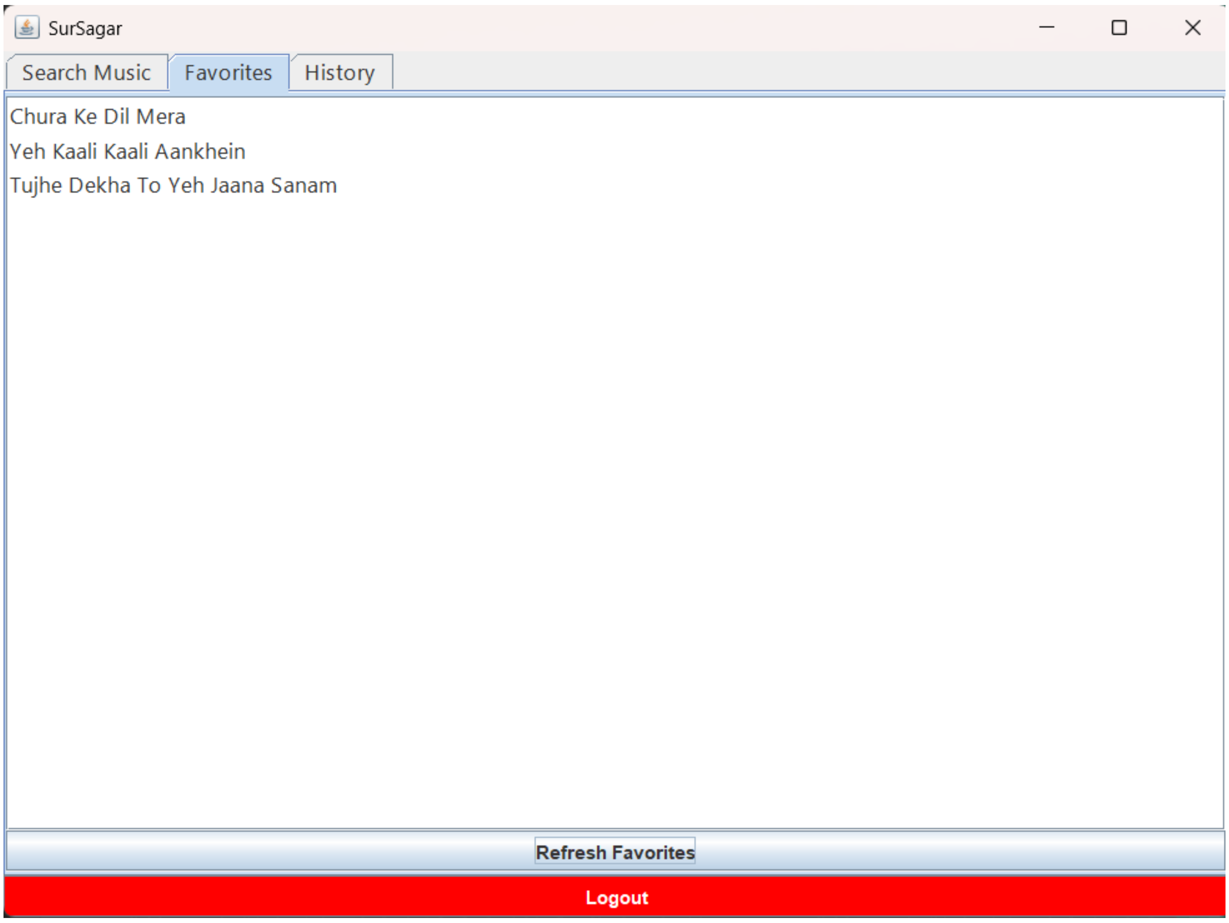


▯ Tujhe Dekha To Yeh Jaana Sanam
▯ Dil Ne Yeh Kaha Hai Dil Se
▯ Aankhon Ki Gustakhiyaan
▯ Main To Raste Se Ja Raha Tha
▯ Chura Ke Dil Mera
▯ Sochenge Tumhe Pyar Karenge
▯ Dheere Dheere Se Meri Zindagi Mein Aana
▯ Yeh Kaali Kaali Aankhein
▯ Jaanam Samjha Karo



Message

Added to Favorites!



Chura Ke Dil Mera
Yeh Kaali Kaali Aankhein
Tujhe Dekha To Yeh Jaana Sanam

3. 🕒 History Panel (DashboardUI.java)

This panel provides an auditable record of the user's activities within the application, demonstrating data logging capabilities.

- Refresh Action:-

The user navigates to the History tab and clicks "Refresh History," executing the loadHistory() method.

- Data Retrieval:-

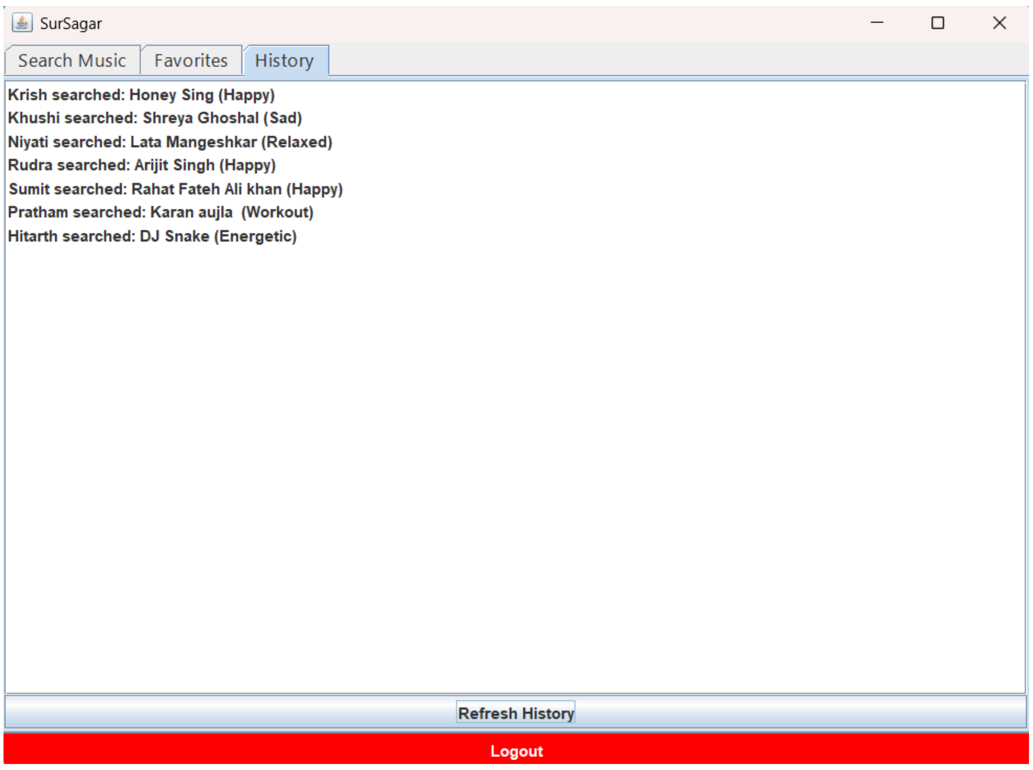
DataManager.loadHistory() is called, which reads line by line from the persistent log file, history.txt.

-UI Population:-

The DefaultListModel<String> historyModel is cleared and updated with the chronologically ordered search logs.

-Log Format Display:-

Each entry displayed in the JList shows the exact log string created during the search, including the user, artist, and mood.



```
Krish searched: Honey Sing (Happy)
Khushi searched: Shreya Ghoshal (Sad)
Niyati searched: Lata Mangeshkar (Relaxed)
Rudra searched: Arijit Singh (Happy)
Sumit searched: Rahat Fateh Ali khan (Happy)
Pratham searched: Karan aujla  (Workout)
Hitarth searched: DJ Snake (Energetic)
```

SUMMARY

The SurSagar project is a desktop application developed in Java Swing, designed to provide users with personalized music recommendations by integrating with a modern Large Language Model (LLM) API. The application simulates a full-featured music discovery service, demonstrating key concepts in UI design, asynchronous networking, and local data persistence.

Key Architectural Highlights:

- Front-End (UI): The user interface is built using Java Swing, utilizing JFrame as the main container and CardLayout for navigating between the Login screen and the Main Dashboard. The Dashboard employs JTabbedPane to organize the Search, Favorites, and History views.
- External Service Layer: The core recommendation logic resides in the MusicAPIService. This service establishes a connection to the OpenRouter/Grok AI API via HttpURLConnection and sends a custom prompt (based on artist and mood) to generate a list of 10 songs.
- Asynchronous Processing: To ensure the user interface remains responsive, the network-intensive API call is executed in a background thread using the SwingWorker class, preventing the Event Dispatch Thread (EDT) from blocking.
- Data Persistence Layer: The DataManager handles data persistence by saving user history and favorited songs to local text files (history.txt and favorites.txt). This simulates a simple database connection, allowing data to persist across sessions.

Core Functionalities Achieved:

1. AI Search: Users input an artist and select a mood to receive AI-generated song recommendations, displayed in custom ResultPanel components.
2. Interactive Results: Each recommendation provides buttons to "Add to Favorites" and to "Find on YouTube", the latter utilizing the Java Desktop API to open a web browser.
3. Logging: Every search action is logged with user details to the History panel for review.

This project successfully integrates UI design principles with advanced networking and threading concepts in Java, resulting in a functional, modern recommendation application.