PRACTICAL – 6

AIM: 6 Project: SplayList – A group recommendation Music webapp

- **Purpose**

  The purpose of Splaylist is to facilitate seamless music collaboration among groups by automatically generating a shared playlist that reflects the combined taste of all group members. By integrating with Spotify, Splaylist allows each user to authenticate, fetch their top tracks, and collectively analyze group preferences to create a unique playlist that everyone will enjoy. The project aims to leverage machine learning and data visualization to provide insights about group music compatibility and trends, making group music sessions more engaging and tailored.

- **Scope**

  Splaylist is a web-based application that enables users to log in with their Spotify accounts, add or remove others from a collaborative group, and generate playlists that represent the musical taste of the group. The system collects and analyzes users' top tracks using the Spotify API and applies clustering techniques to identify group-level preferences. Besides playlist generation, Splaylist offers visual analytics (charts, statistics, PDF reports) to help users understand the diversity and overlap in their group's listening habits. The project is scoped for educational purposes as a mini project and is intended for users with existing Spotify accounts who want to automate and enhance group playlist creation without manual song selection.

- **Technologies Used**
  - **Programming Language:**

    Python 3.x – The primary language for application logic, data processing, APIs, and machine learning.

  - **Web Framework:**

    Flask – A lightweight Python web framework used to build the web-based user interface and manage HTTP requests.

  - **Third-Party APIs:**

    Spotify Web API (accessed via Spotipy) – Used for authenticating users, fetching user data, and managing playlists.

  - **Data Science and Machine Learning Libraries:**
    - Pandas – For data cleaning and analysis.
    - NumPy – For numerical computing and data manipulation.
    - scikit-learn – For clustering music data and building recommendation logic.

  - **Data Visualization & Reporting:**
    - Matplotlib & Seaborn – For generating analytical charts and graphs.
    - ReportLab – For automating the creation of PDF reports.

  - **Other Libraries:**
    - Requests – For HTTP requests (used internally by APIs).
    - python-dotenv – For securely managing environment variables (e.g., API keys).

- **Operating Environment:**
  - Web browser (for user interface)
  - Runs on standard desktop operating systems (Windows/Linux/Mac)
- **Installation & Dependency Management:**
  - requirements.txt – Specifies all Python dependencies for quick setup.

- **Functional Requirements**

FR1: Authentication Requirements

- **FR1.1**: The system SHALL authenticate users through Spotify OAuth2 protocol
- **FR1.2**: The system SHALL maintain secure session management with token refresh capabilities
- **FR1.3**: The system SHALL provide logout functionality with complete session and cache cleanup
- **FR1.4**: The system SHALL handle authentication errors gracefully with user-friendly messages
- **FR1.5**: The system SHALL force fresh login dialogs when adding different users

FR2: Group Management Requirements

- **FR2.1**: The system SHALL allow users to add multiple group members to a collaborative session
- **FR2.2**: The system SHALL provide individual user removal functionality from active sessions
- **FR2.3**: The system SHALL display current group members with management controls
- **FR2.4**: The system SHALL prevent duplicate user additions with appropriate warnings
- **FR2.5**: The system SHALL provide "Clear All Users" functionality to reset sessions

FR3: Data Collection and Processing Requirements

- **FR3.1**: The system SHALL collect user's top 50 tracks data from Spotify API
- **FR3.2**: The system SHALL process and clean music data for analysis
- **FR3.3**: The system SHALL generate pseudo-genres for tracks without genre information
- **FR3.4**: The system SHALL save processed data in CSV format for future analysis
- **FR3.5**: The system SHALL handle missing or incomplete data gracefully

FR4: Playlist Generation Requirements

- **FR4.1**: The system SHALL create unified playlists combining all group members' preferences
- **FR4.2**: The system SHALL remove duplicate tracks while preserving order
- **FR4.3**: The system SHALL save generated playlists directly to authenticated user's Spotify account
- **FR4.4**: The system SHALL handle playlist creation errors with fallback mechanisms
- **FR4.5**: The system SHALL provide Spotify links to created playlists

FR5: Analytics and Visualization Requirements

- **FR5.1**: The system SHALL generate genre distribution charts
- **FR5.2**: The system SHALL create popularity analysis visualizations
- **FR5.3**: The system SHALL provide user diversity comparison charts
- **FR5.4**: The system SHALL perform statistical analysis (descriptive and inferential)
- **FR5.5**: The system SHALL generate comprehensive PDF reports with embedded charts
- **FR5.6**: The system SHALL allow users to download analysis results

FR6: Machine Learning Requirements

- **FR6.1**: The system SHALL apply K-Means clustering to group similar music preferences
- **FR6.2**: The system SHALL automatically determine optimal number of clusters
- **FR6.3**: The system SHALL provide cluster-based recommendations
- **FR6.4**: The system SHALL save and load trained models for consistency

- **Non – Functional Requirements**

NFR1: Performance Requirements

- **NFR1.1**: Playlist generation SHALL complete within 30 seconds for groups up to 10 users
- **NFR1.2**: Chart generation SHALL complete within 15 seconds for datasets up to 500 tracks
- **NFR1.3**: User authentication SHALL complete within 10 seconds
- **NFR1.4**: PDF report generation SHALL complete within 60 seconds including all visualizations
- **NFR1.5**: The system SHALL handle concurrent user sessions without performance degradation

NFR2: Usability Requirements

- **NFR2.1**: The web interface SHALL be intuitive and require minimal training
- **NFR2.2**: The system SHALL provide clear navigation with consistent design patterns
- **NFR2.3**: Response messages SHALL be informative and guide users through processes
- **NFR2.4**: The interface SHALL be responsive and work across different screen sizes
- **NFR2.5**: Error messages SHALL be user-friendly and provide actionable guidance

NFR3: Reliability Requirements

- **NFR3.1**: The system SHALL handle Spotify API failures gracefully without crashing
- **NFR3.2**: Authentication token refresh SHALL occur automatically without user intervention
- **NFR3.3**: Data processing SHALL continue even with partial data availability
- **NFR3.4**: Error recovery SHALL maintain session state whenever possible
- **NFR3.5**: The system SHALL have 95% uptime during active development/demonstration

NFR4: Security Requirements

- **NFR4.1**: User authentication SHALL use OAuth2 standard security protocols
- **NFR4.2**: API keys and sensitive data SHALL be stored securely using environment variables
- **NFR4.3**: Session data SHALL be properly cleared upon logout or session expiration
- **NFR4.4**: The system SHALL not store user passwords or permanent personal data
- **NFR4.5**: All HTTP communications SHALL be conducted over secure connections

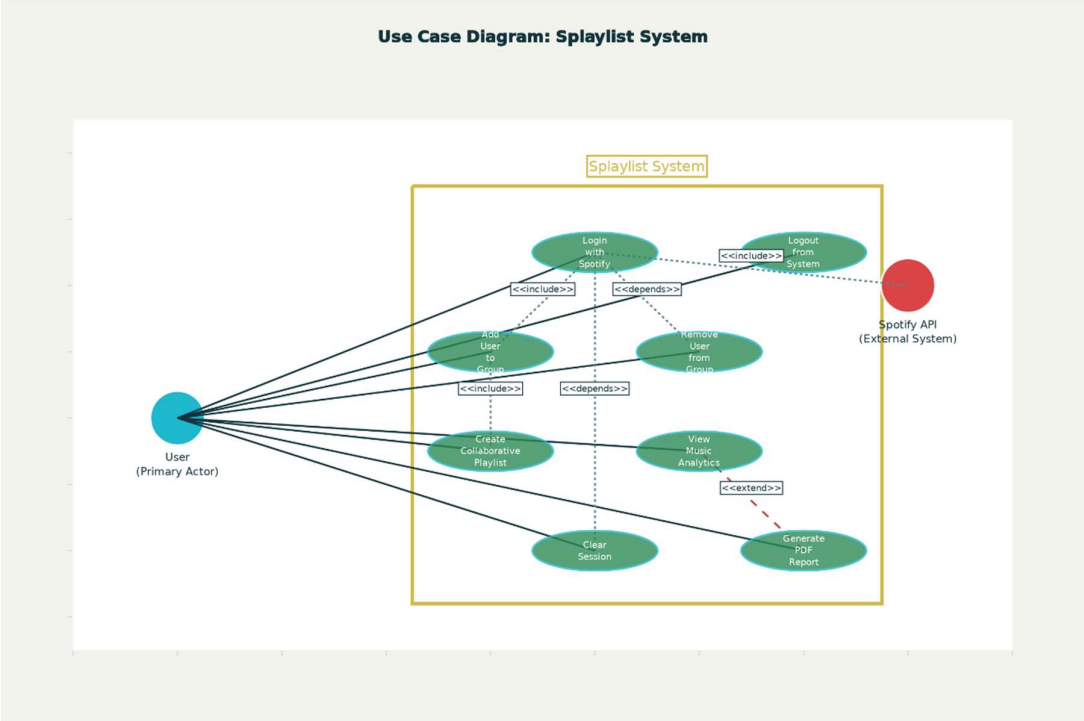
NFR5: Compatibility Requirements

- **NFR5.1**: The system SHALL be compatible with modern web browsers (Chrome, Firefox, Safari, Edge)
- **NFR5.2**: The application SHALL run on Python 3.8+ environments
- **NFR5.3**: Generated files SHALL be compatible with standard PDF viewers and CSV applications
- **NFR5.4**: The system SHALL work across Windows, macOS, and Linux operating systems
- **NFR5.5**: The interface SHALL be mobile-responsive for basic functionality

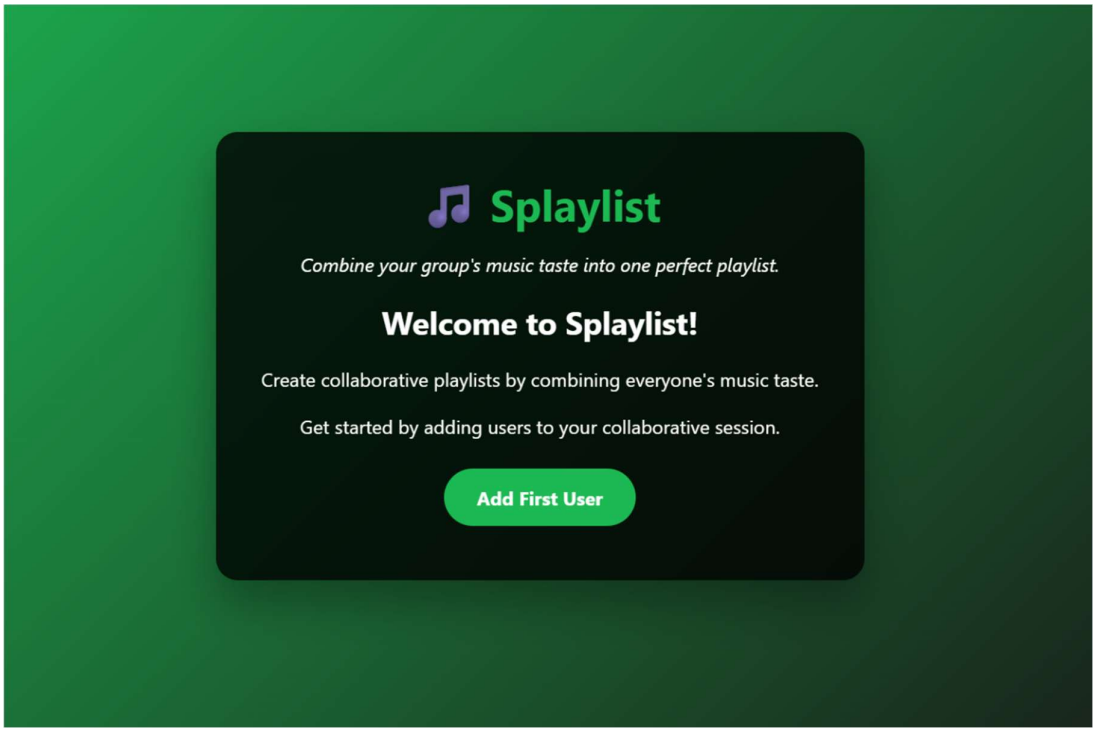
NFR6: Scalability Requirements

- **NFR6.1**: The system SHALL support groups of up to 10 users efficiently
- **NFR6.2**: The system SHALL handle datasets of up to 1000 tracks per analysis
- **NFR6.3**: Memory usage SHALL not exceed 512MB during normal operation
- **NFR6.4**: The system SHALL process multiple visualization requests concurrently

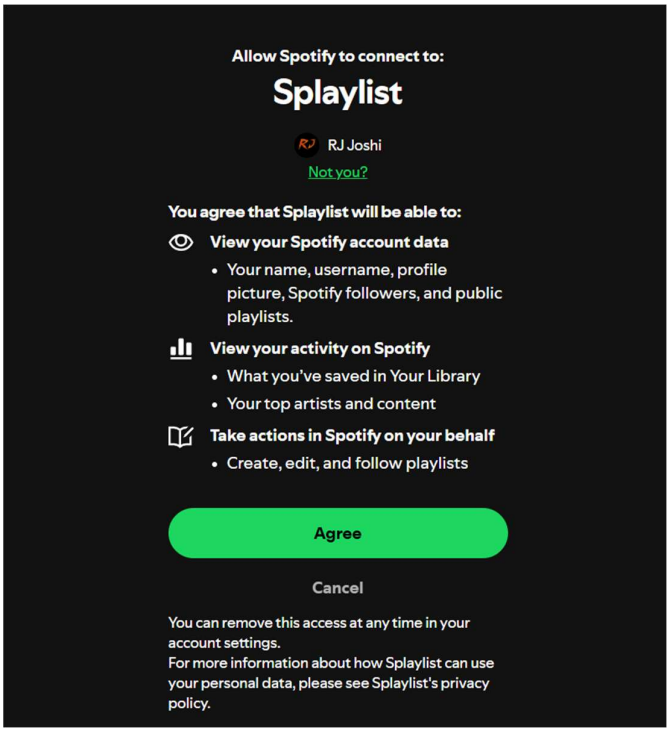
- **System Design of the SplayList**
  - Use Case Diagram

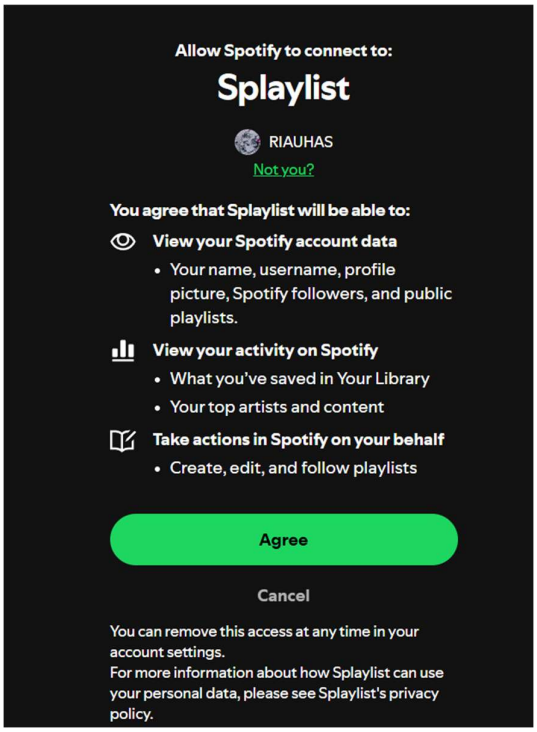


  - Home Page

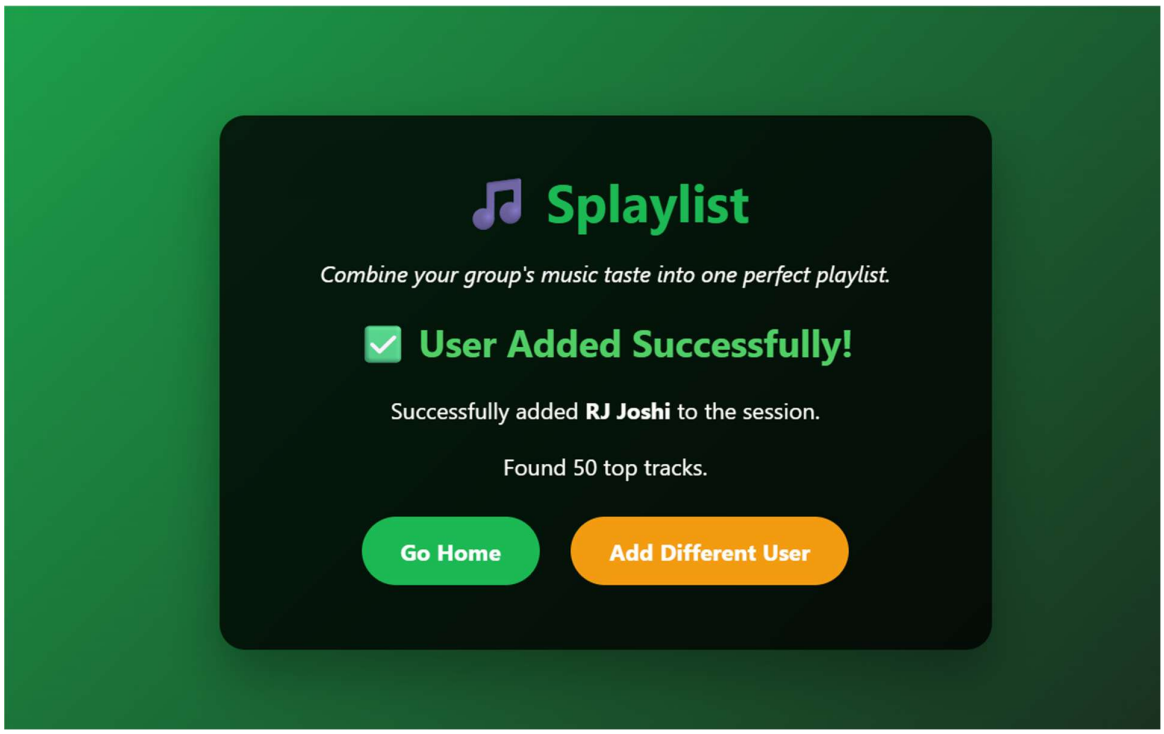


- Auth Dialog (For Ex. Adding two different users)

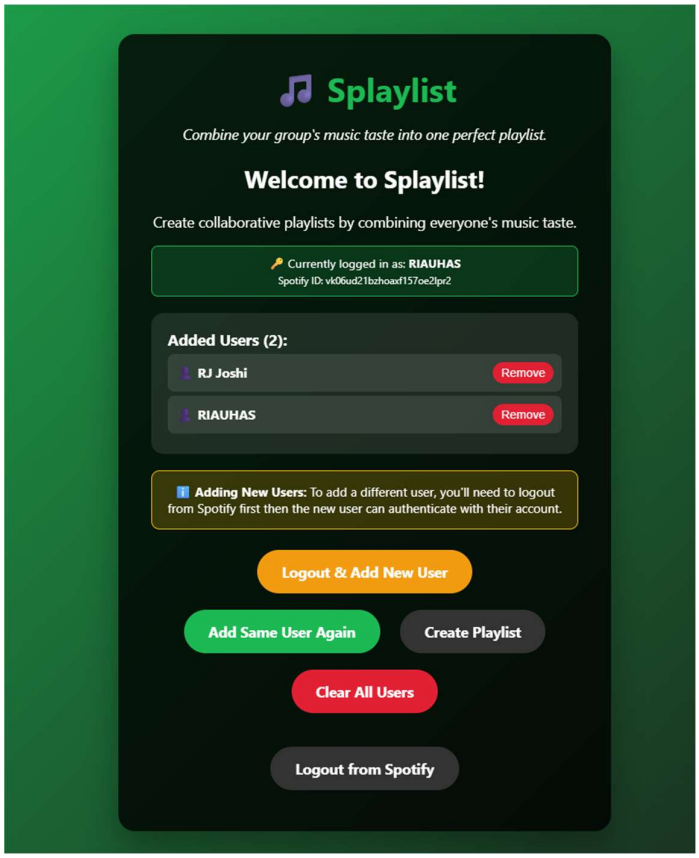




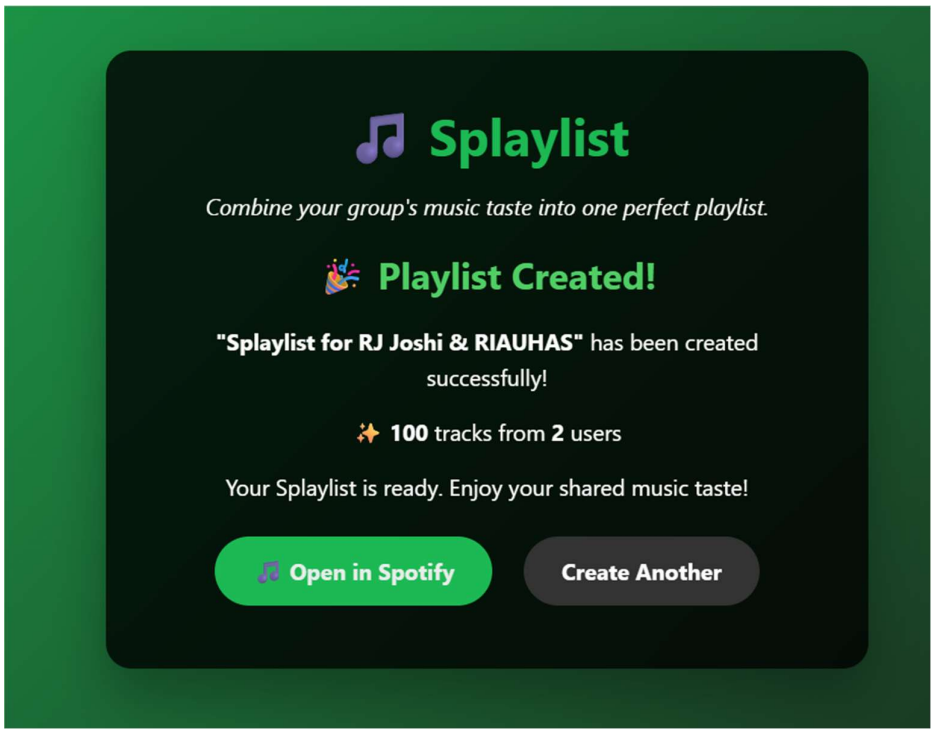
- User added Page



- User added Page



- Play-list created



- Play list in Spotify