



UE22CS352B - Object Oriented Analysis & Design

Mini Project Report

Title

Submitted by:

Srijan Aditya: PES1UG22CS612

Shubhangi Vats: PES1UG22CS589

Shrujana I S: PES1UG22CS586

Sneha V Singanal: PES1UG22CS599

6th Semester J section

Faculty Name: Bhargavi Mokashi

January - May 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013) 100ft
Ring Road, Bengaluru – 560 085, Karnataka, India

Problem Statement:

Modern music enthusiasts need a simple yet effective way to manage their music collections across devices without depending solely on commercial streaming platforms. Current solutions often come with limitations such as subscription costs, internet connectivity requirements, or limited customization options.

This project aims to develop a web-based music player application using the Spring Boot framework with an MVC architecture and MongoDB for data persistence. The application allows users to upload, store, manage, and play music files while providing personalized queue management.

Key Features:

User Management

- **User Registration and Authentication:** Secure signup and login functionality
- **Personalized User Profiles:** Individual accounts with customized preferences
- **Session Management:** Persistent user sessions with secure authentication

Music Library Management

- **Song Upload Capability:** Users can upload their own music files to the system
- **Metadata Management:** Store and display song information including title and artist
- **Music Search Functionality:** Find songs by title in the music library
- **Duplicate Detection:** Prevent duplicate uploads of the same song

Playback Features

- **Browser-Based Audio Playback:** Play music directly in the web browser
- **Advanced Queue Management:** Add, remove, and clear songs from the playback queue
- **Persistent Queues:** User queues are saved between sessions
- **Queue Synchronization:** Queue changes are reflected in both the database and active session

Queue System

- **Personal Queue Creation:** Each user maintains their own personalized song queue
- **Queue Manipulation:** Add songs to queue, remove specific songs, and clear entire queue

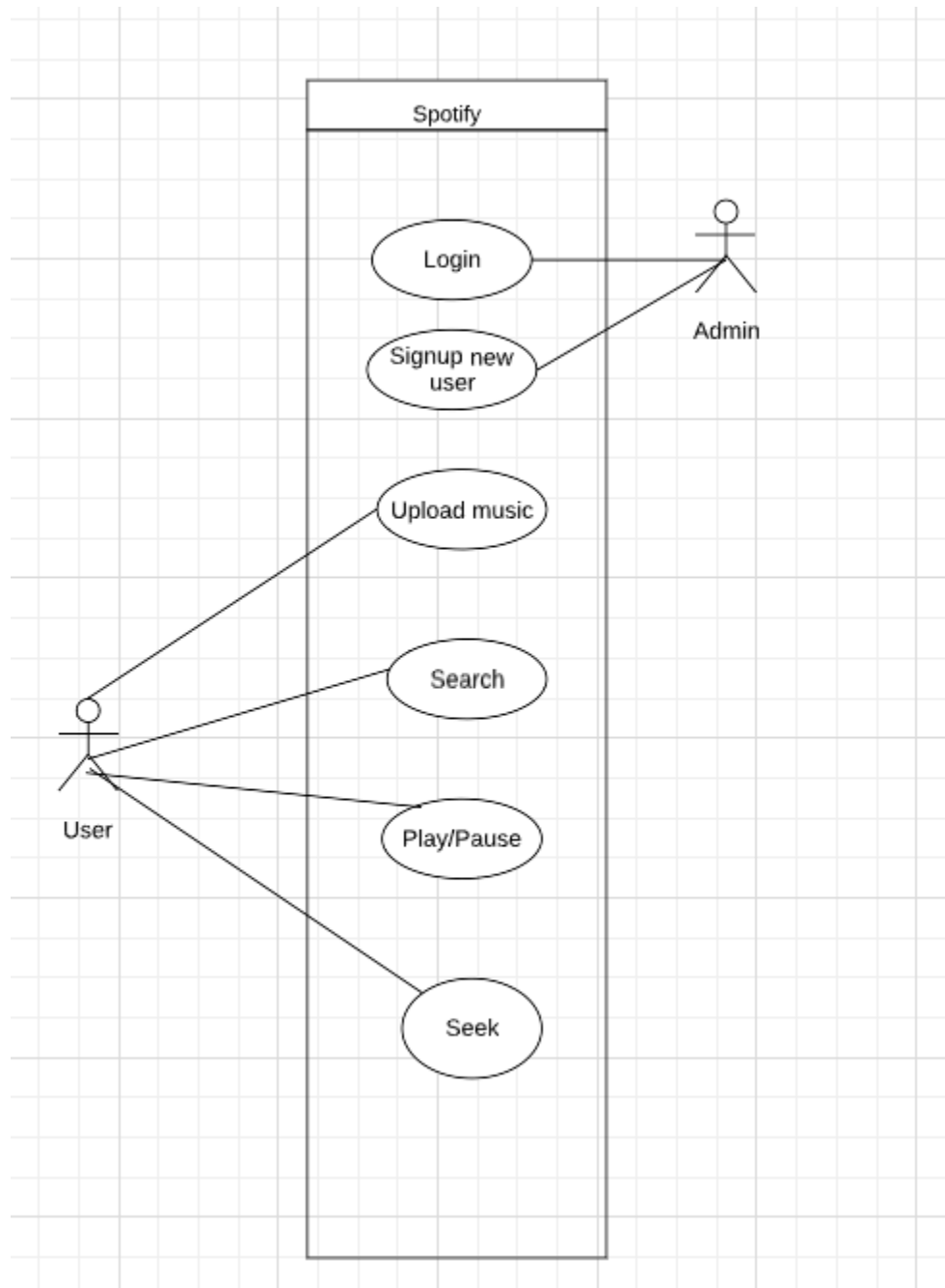
- **Cross-Session Queue Persistence:** Queues are stored in the database and retrieved upon login
- **Real-Time Queue Updates:** Changes to the queue are immediately reflected in the user interface
- **Queue-Session Synchronization:** User's queue is synchronized between the database and session storage
- **Default Queue Population:** New users receive a default queue with sample songs

Storage Architecture

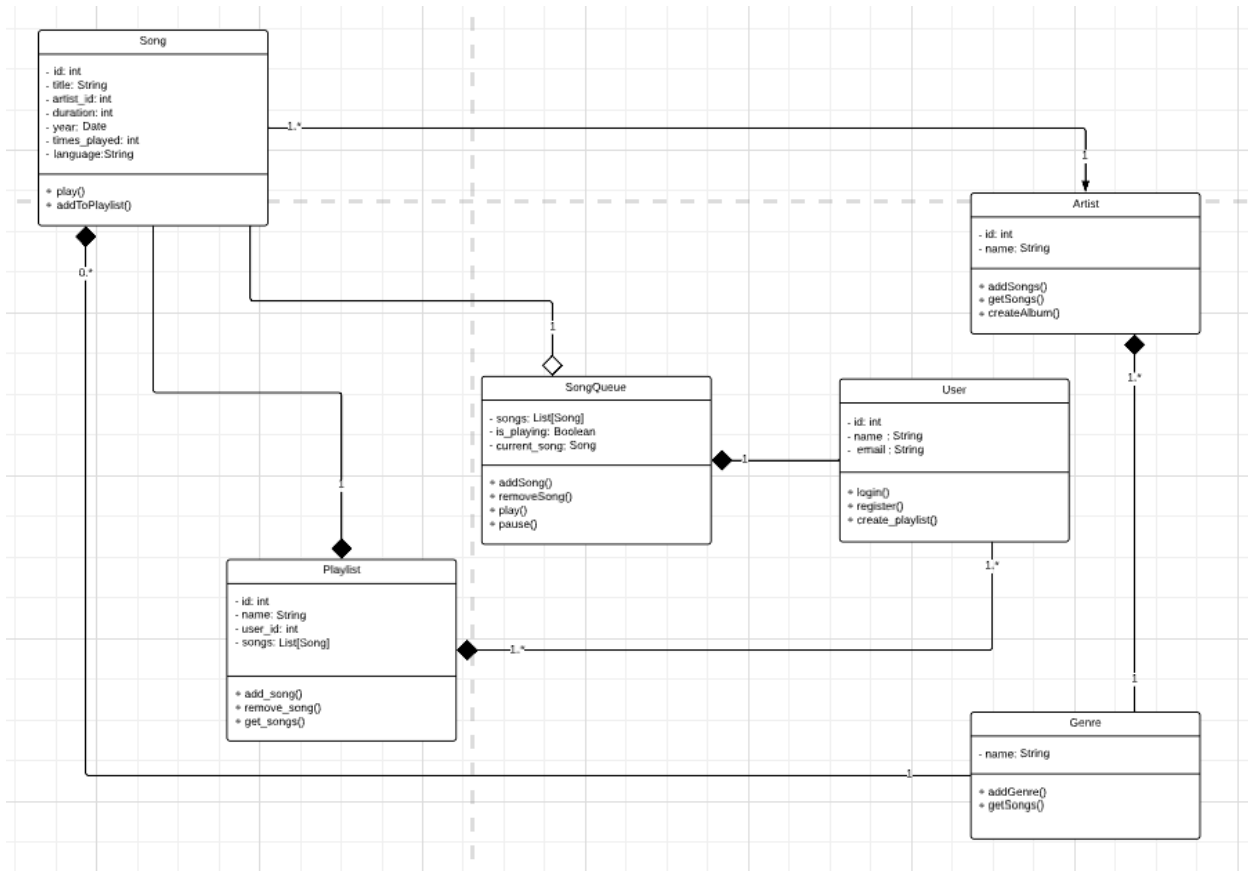
- **Cloud Storage Integration:** Amazon S3 integration for scalable music file storage
- **Graceful Fallback Mechanism:** Automatic switch to local storage when cloud storage is unavailable
- **Hybrid Storage Strategy:** Ability to function with either cloud or local storage systems
- **Database-Storage Synchronization:** Music metadata in database stays in sync with storage system

Models:

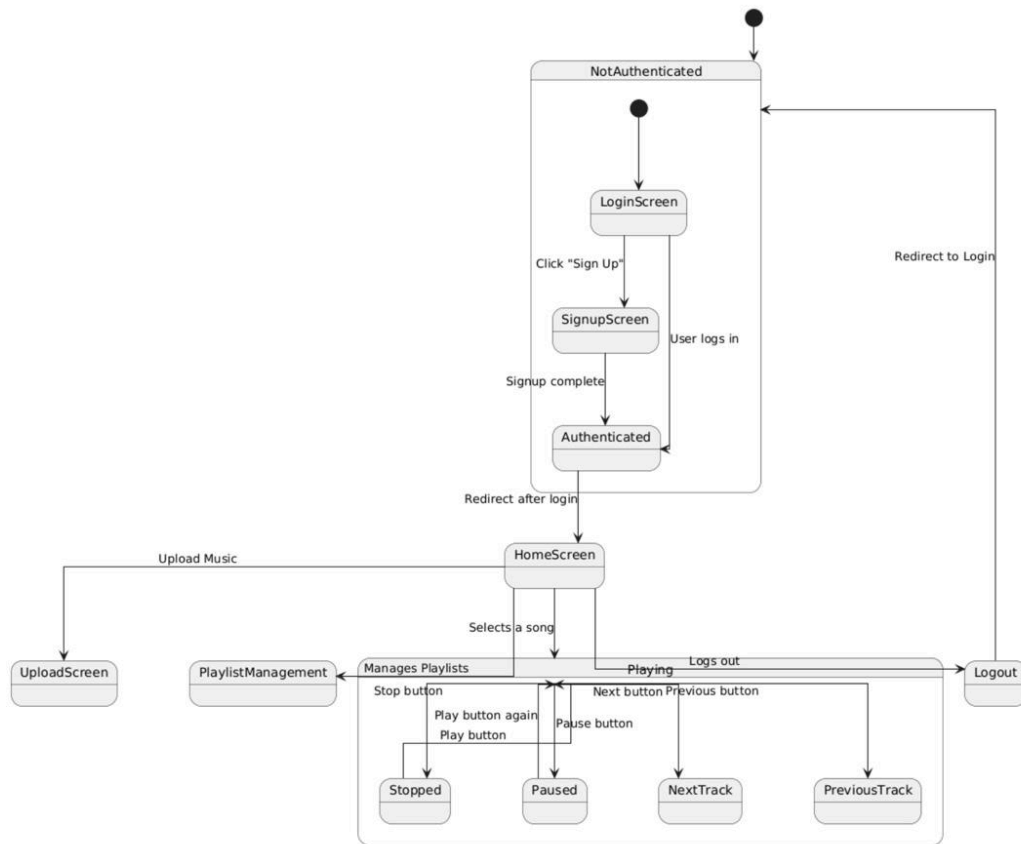
Use Case Diagram:



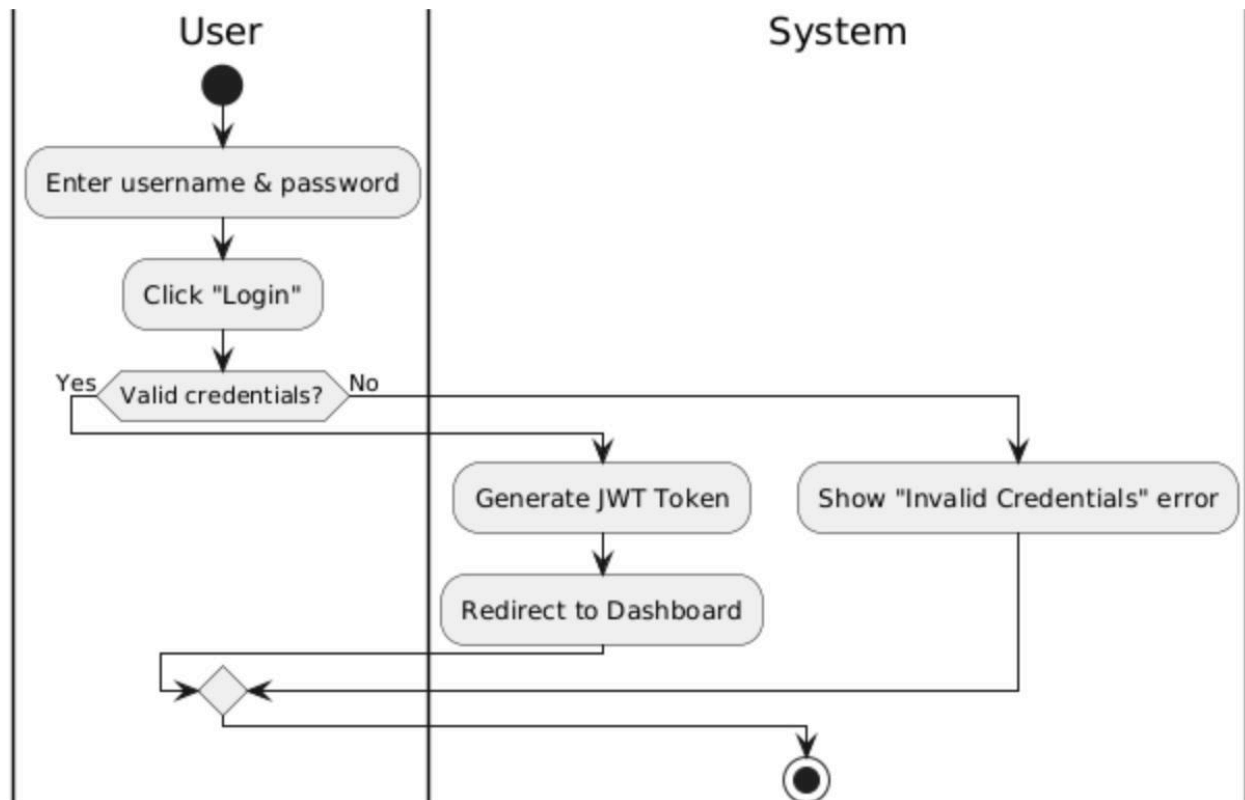
Class Diagram:

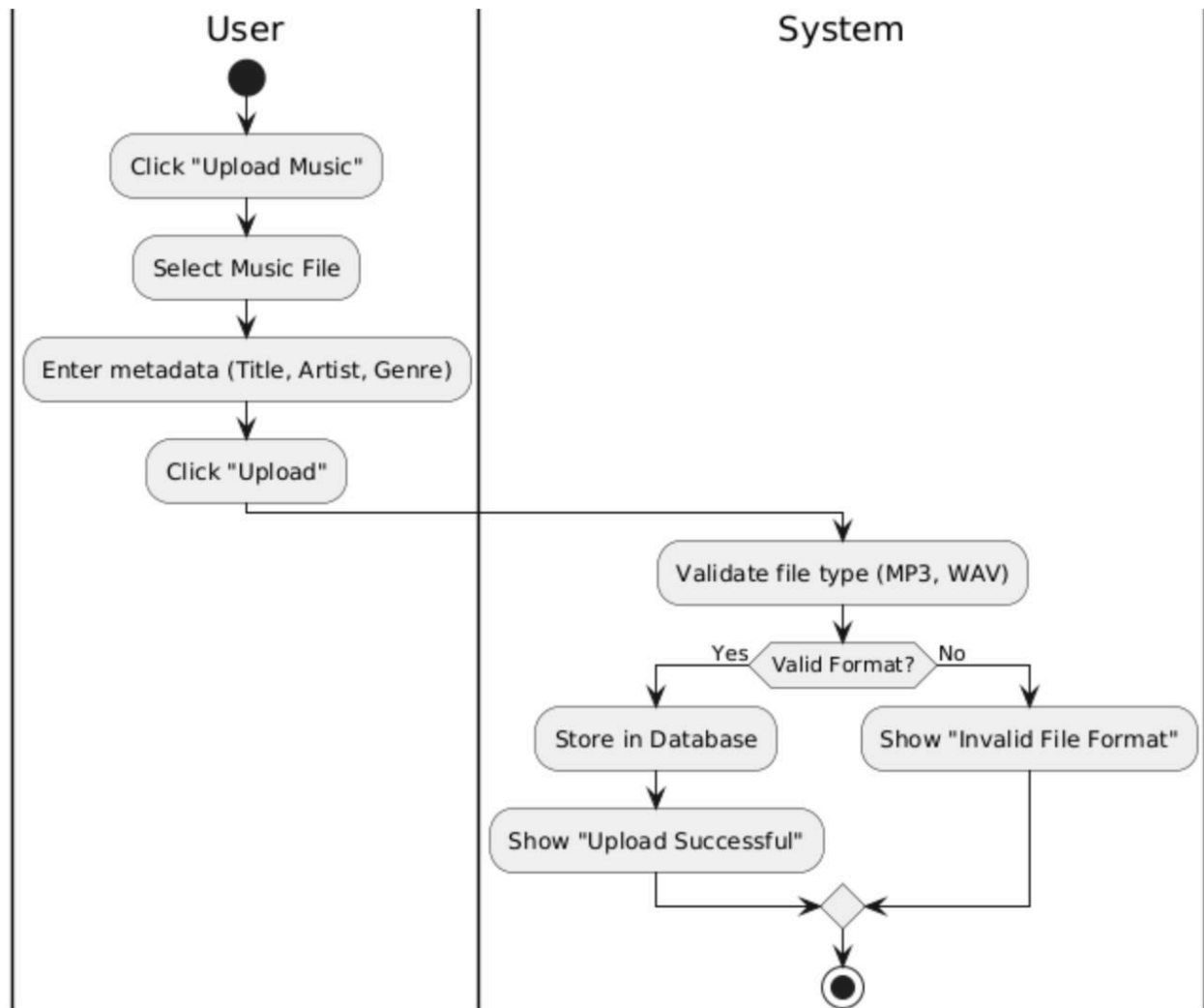


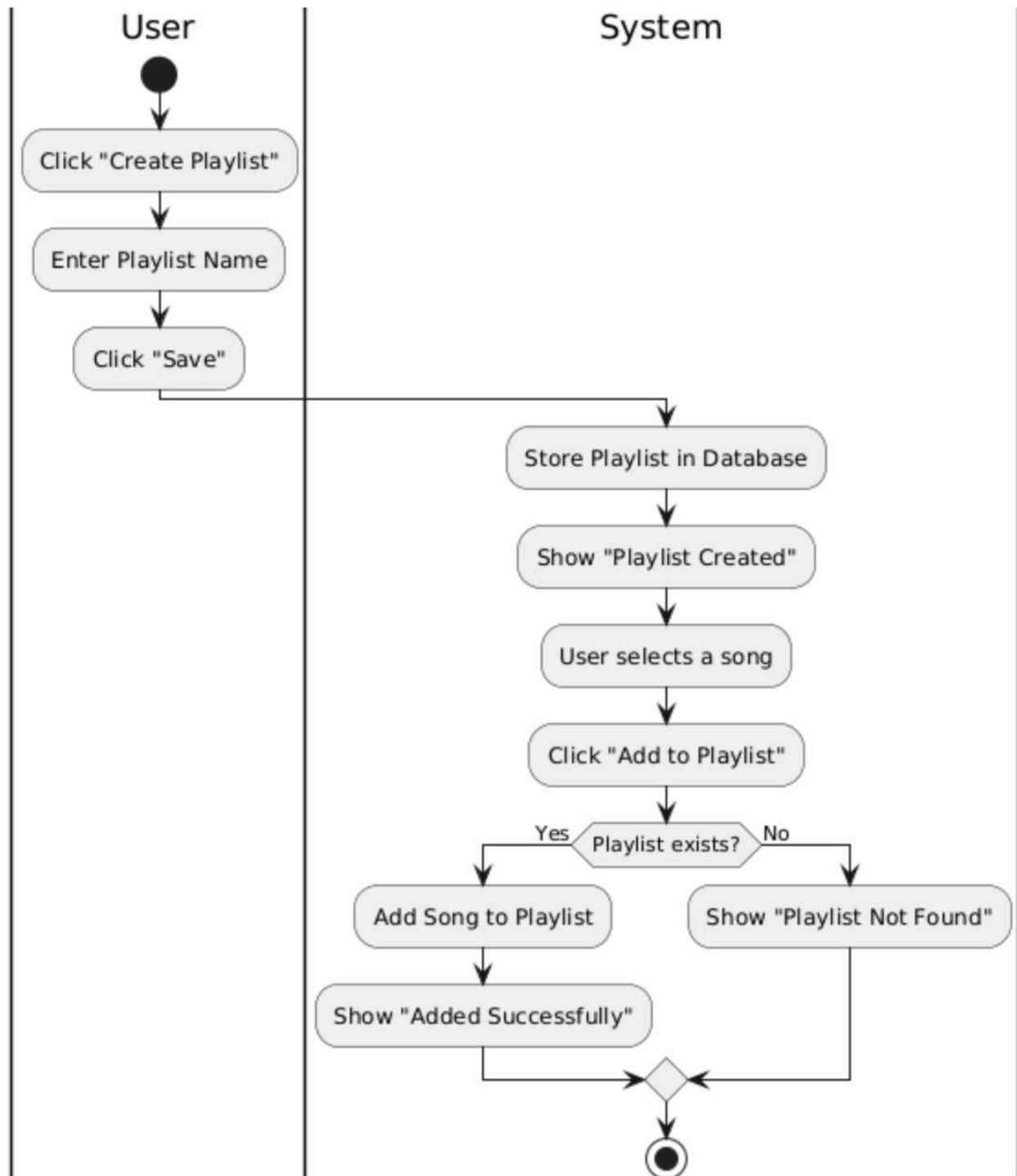
State Diagram:

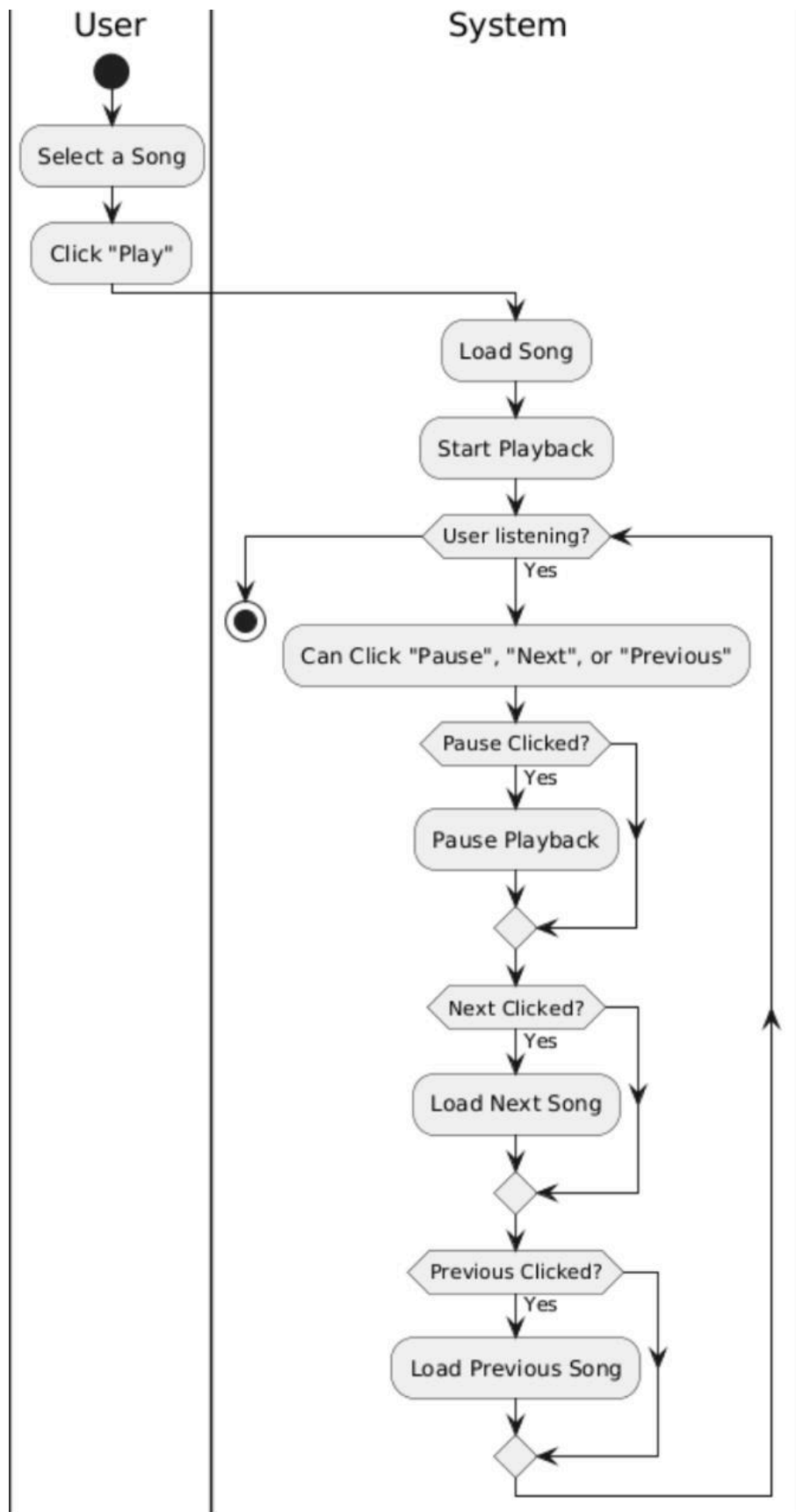


Activity Diagrams:









Architecture Patterns, Design Principles, and Design Patterns:

Architecture Patterns

1. MVC (Model-View-Controller) Pattern

This is the primary architectural pattern used throughout the application:

- Models: Song, User, and Role classes
- Views: Templates like index.html, myQueue.html, etc.
- Controllers: IndexController, SongController, and UserController

2. Microservice Potential

- Separate databases for users and songs
- Configuration supports multiple MongoDB connections

3. Layered Architecture

- Presentation layer (Controllers)
- Business logic layer (Services)
- Data access layer (Repositories)
- Domain layer (Models)

4. Multiple Database Configuration

- Uses multiple MongoDB databases through configuration
- Separate databases for user data and song data

5. Repository Pattern

Used to separate data access logic from business logic:

- SongRepository and UserRepository interfaces extend MongoRepository
- This provides an abstraction layer over data persistence operations

6. Service Layer Pattern

Business logic is encapsulated in service classes:

- SongService and StorageService handle business operations
- Controllers delegate to these services rather than implementing logic directly

Design Principles

1. Separation of Concerns

- Clear separation between controllers, services, and repositories
- Each class has a focused responsibility

2. Single Responsibility Principle

- Each class has a defined purpose (User handling, Song handling, storage, etc.)

3. Interface Segregation Principle

- Repository interfaces define specific methods needed by the application

4. Dependency Inversion Principle

- High-level components (controllers) depend on abstractions (interfaces)
- Low-level details implemented by concrete classes

5. Don't Repeat Yourself (DRY)

- Common functionality extracted to service classes
- Reuse of MongoDB operations through repositories

Design Patterns

1. Dependency Injection

Spring's DI is used extensively:

- Constructor injection in controllers (e.g., IndexController, SongController)
- `@Autowired` annotation for injecting dependencies

2. Singleton Pattern

Spring beans are singletons by default:

- All services, repositories, and controllers are created as singleton beans
- `@Service`, `@Controller`, `@Repository` annotations mark classes for singleton creation

3. Factory Pattern

Used implicitly through Spring's configuration:

- MongoTemplate beans are created via factory methods in MultipleMongoConfig

4. Strategy Pattern

Used in StorageService:

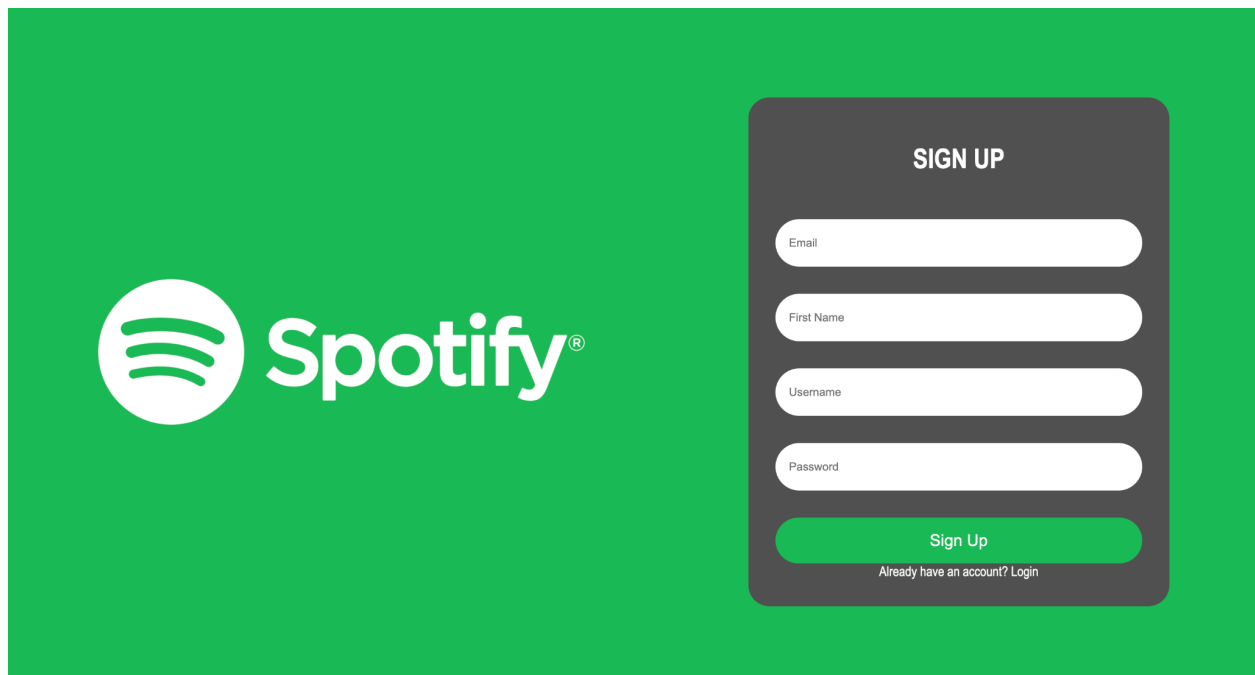
- Different storage strategies (S3 vs. local mock storage) based on configuration
- The client code doesn't need to know which strategy is being used

Github link to the Codebase:

https://github.com/SrijanAditya/MUSIC_SPRINGBOOT

Screenshots

UI:





LOGIN

Log In

Don't have an account? Sign Up



Welcome,

hehe

Go To Queue

Add Songs

All Time Hits

Search

test.mp3





naseeb-indian-pop-music-267951.mp3



gorila-315977.mp3





Welcome,
hehe
All Songs

Current Queue

Clear Queue

naseeb-indian-pop-music-267951.mp3	▶ -
gorila-315977.mp3	▶ -
naseeb-indian-pop-music-267951.mp3	▶ -
gorila-315977.mp3	▶ -
test.mp3	▶ -

gorila-315977.mp3

◀

▶ 0:00 / 0:00

🔊 ⋮

▶

Uploads

Title

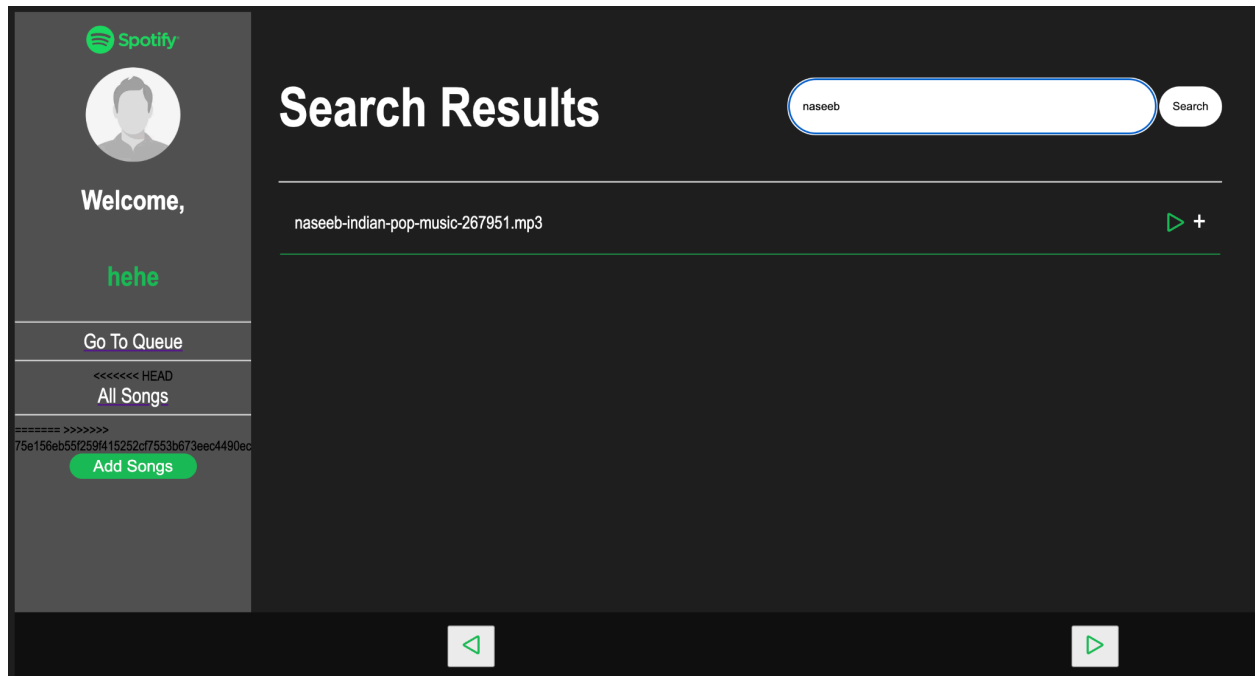
Artist

Choose file

No file chosen

Save

Cancel



Individual contributions of the team members:

Name	Module worked on
Srijan Aditya	Queue Feature + Database(MongoDB)
Shubhangi Vats	Search Feature + Role
Sneha V Singanal	Login and Signup + Song
Shrujana I S	Search feature + User