Faculty of Engineering, Alexandria University
Computers and Systems Engineering Department

Software Engineering
Milestone 1

Music Zone: An Online Virtual Music Listening Platform

Prepared By

Bishoy Nader Fathy	**[21]**
Amr Mohamed Nasreldin	**[47]**
Marc Magdi Kamel	**[55]**
Michael Raafat Mikhail	**[57]**

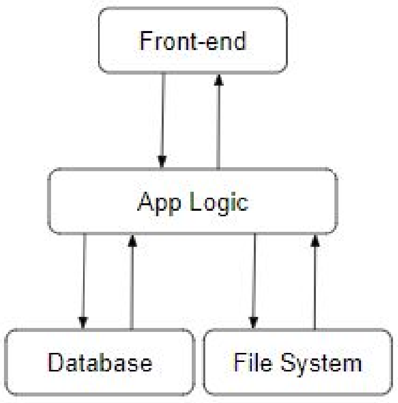
Framework : Ruby on Rails + Postgres

Decisions We proposed two frameworks to work with, Ruby on Rails and Spring on Java. We decided to go with Rails as its a more agile framework that can adapt easily to the change in requirements in our project. We have made an analysis for the familiarity of the team members with the two frameworks. We found that out of four members there exist one member who is familiar with rails and another who is familiar with spring however we decided to work with rails seizing this learning opportunity to learn new technology through this application.

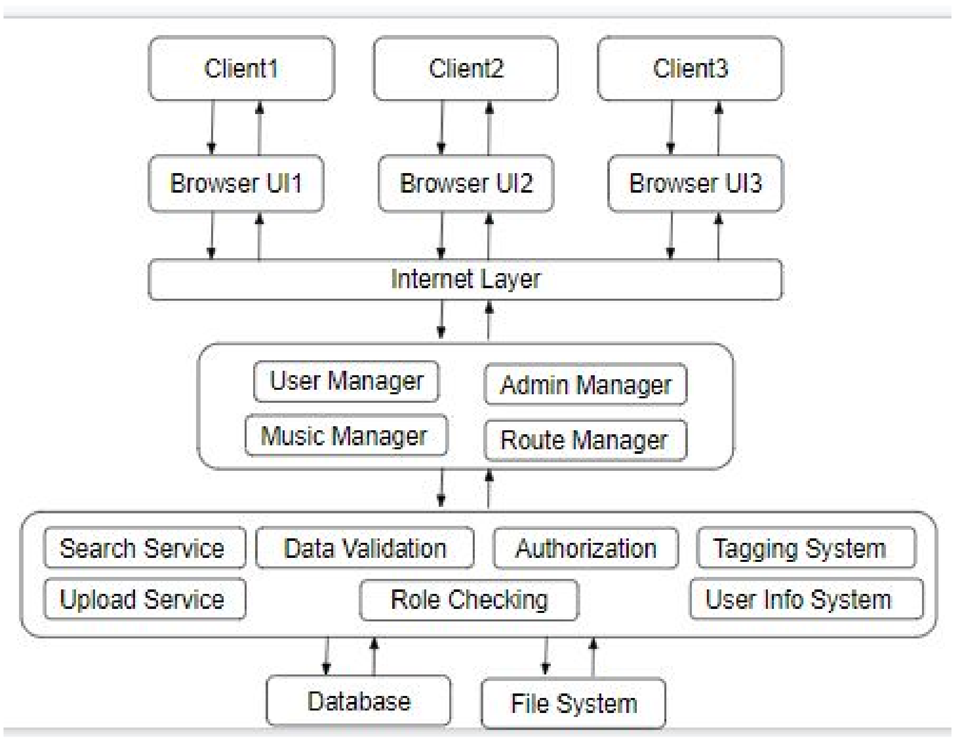
For the database engine, we decided to go with postgres as its more popular now, it handles the transactions better than MySql and we already have some experience with it.

Architecture Design

For our Architecture, As an overview of our architecture:



Going in details :



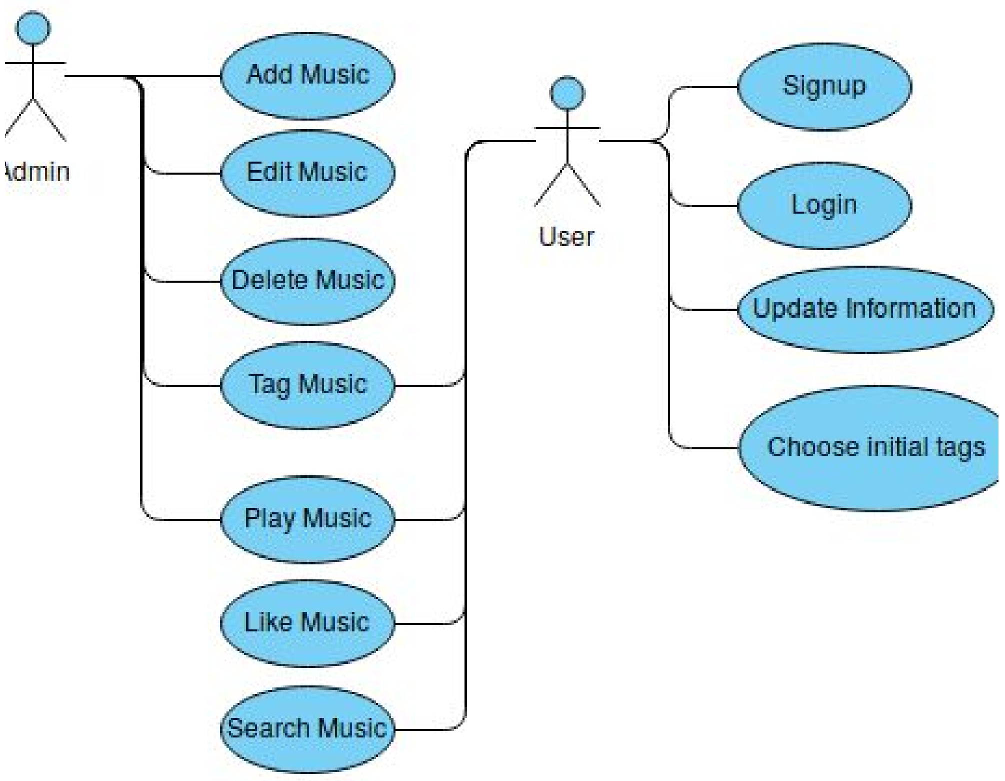
We decided to use MVC for front-end as we have Model of user's data and his/her requests, View in which our data is shown to user and Controller that controls any action done be used to get data from Model.

Mixed with Layered architectural pattern for the front end, we have three layers of our system:

- The first layer is the Front-end which is responsible for showing the user the data he/she wants
- The second layer is the UI management layer that controls authorization checks and other checks.
- The third layer is the core logic layer which has all the functionality the user wants.
- The fourth layer is the Database and file system of our system.

Also, repository Architecture for File System and finally client-server architecture, to enable clients to communicate through the internet with the server.

Use Case Diagrams:



The admin can:

1. **Add Music:** the admin can add upload new music to the system and add its main metadata.
2. **Edit Music:** the admin can edit the metadata of the uploaded music in the system.
3. **Delete Music:** the admin can delete any of the uploaded music to the system.

Both admin & user:

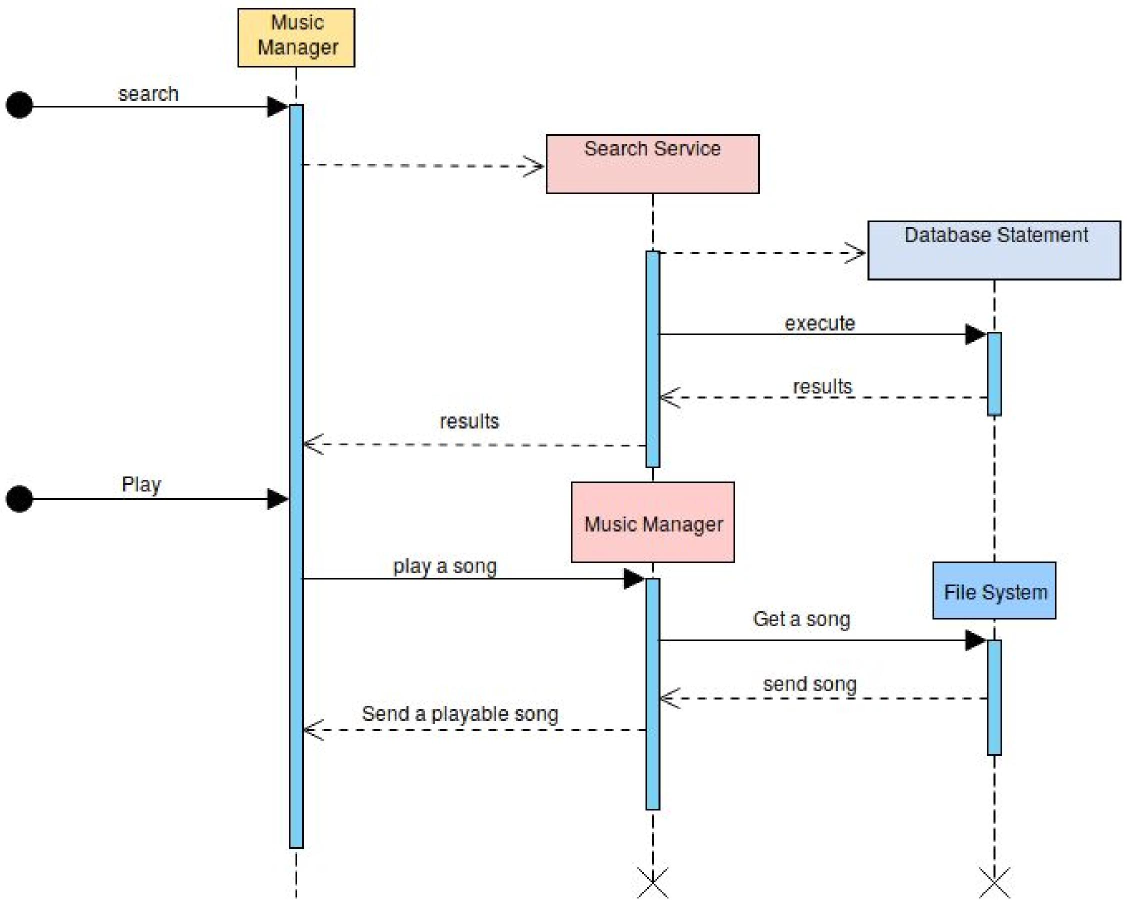
1. **Tag Music:** both admin and user can tag the uploaded music in the system.
2. **Play Music:** both admin and user can play the music uploaded to the system.

The user can:

1. **Like Music:** like any music in the system and add the music to its favourite list.
2. **Search Music:** search for any specific music in the system.
3. **Signup:** the user can sign up to the system and fill his main information.
4. **Login:** the user can his credentials to login to the system
5. **Update info:** the user can update his information in the system and update his interests.

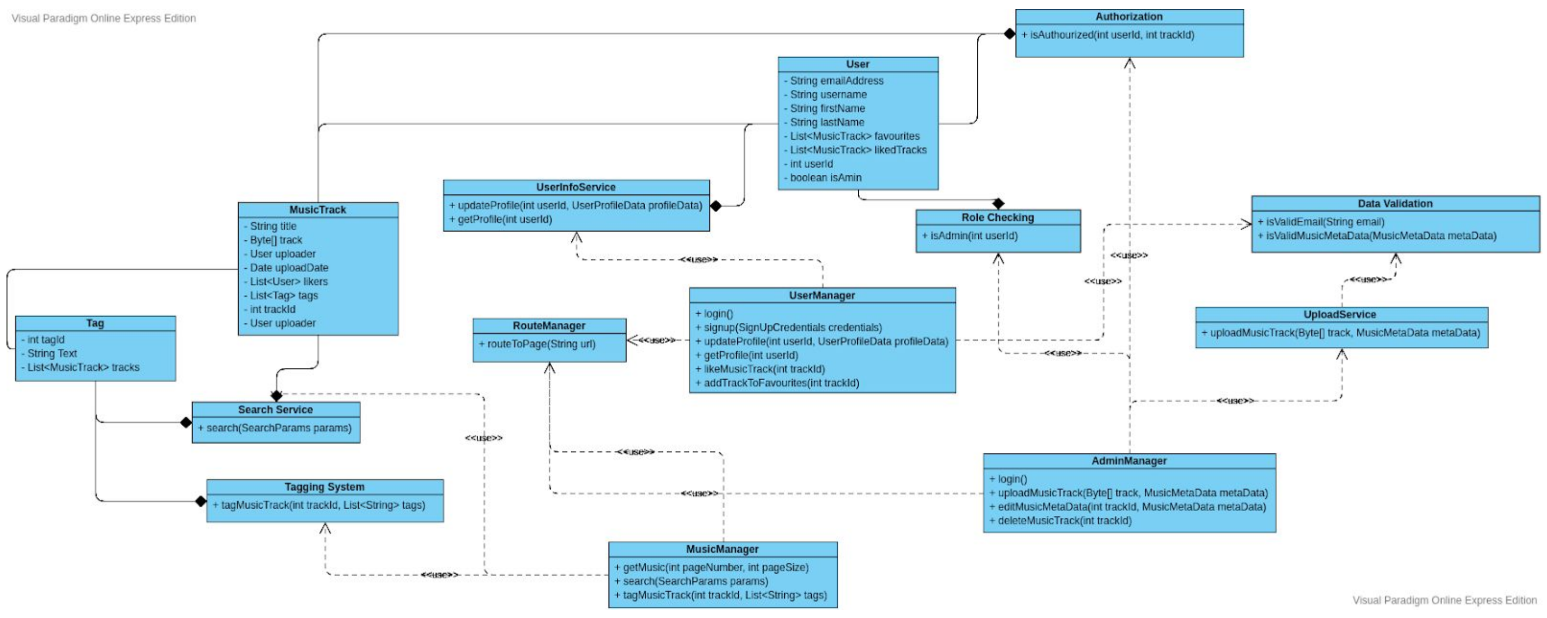
6. **Choose initial tags:** after the first login the user can choose his favourites tags so he can have a customized homepage.

Sequence Diagrams:



The user starts by searching for a specific music, he uses the search service to get his response. After that he ask the system to play a specific music, the system asks the file system for the required music file and start playing it.

Detailed Class Diagram


Visual Paradigm Online Express Edition
Authorization
+ isAuthourized(int userId, int trackId)
User
- String emailAddress
- String username
- String firstName
- String lastName
- List<MusicTrack> favourites
- List<MusicTrack> likedTracks
- int userId
- boolean isAmin
UserInfoService
+ updateProfile(int userId, UserProfileData profileData)
+ getProfile(int userId)
MusicTrack
- String title
- Byte[] track
- User uploader
- Date uploadDate
- List<User> likers
- List<Tag> tags
- int trackId
- User uploader
Role Checking
+ isAdmin(int userId)
Data Validation
+ isValidEmail(String email)
+ isValidMusicMetaData(MusicMetaData metaData)
Tag
- int tagId
- String Text
- List<MusicTrack> tracks
RouteManager
+ routeToPage(String url)
UserManager
+ login()
+ signup(SignUpCredentials credentials)
+ updateProfile(int userId, UserProfileData profileData)
+ getProfile(int userId)
+ likeMusicTrack(int trackId)
+ addTrackToFavourites(int trackId)
UploadService
+ uploadMusicTrack(Byte[] track, MusicMetaData metaData)
Search Service
+ search(SearchParams params)
Tagging System
+ tagMusicTrack(int trackId, List<String> tags)
AdminManager
+ login()
+ uploadMusicTrack(Byte[] track, MusicMetaData metaData)
+ editMusicMetaData(int trackId, MusicMetaData metaData)
+ deleteMusicTrack(int trackId)
MusicManager
+ getMusic(int pageNumber, int pageSize)
+ search(SearchParams params)
+ tagMusicTrack(int trackId, List<String> tags)
<<use>>
Visual Paradigm Online Express Edition

CRC

Interface **AdminManager**	
• Handle login requests of admin accounts • Handles music tracks addition requests • Handles music tracks metadata editing requests • Handles music tracks deletion requests	• RouteManager • Role Checking • Authorization • UploadService

Interface **UploadService**	
• Add a music track to the file system	• DataValidation

Interface **DataValidation**	
• Checks that user data is valid • Checks that music track metadata is valid	

Interface **Authorization**	
• Checks that the user is authorized to upload a certain track	• User • MusicTrack

Interface **Role Checking**	
• Checks if a user is an admin or a normal client	• User

Interface **UserManager**	
• Handles login requests for normal clients. • Handles registration requests from clients. • Handles update requests from clients. • Handles profile preview requests from clients. • Handles music like requests from clients. • Handles music addition to the favorites of a client requests from clients.	• UserInfoService • RouteManager • DataValidation

Interface **UserInfoService**	
• Updates the user data in the database. • Queries the database for a certain user.	• User

Interface **RouteManager**	
• The web server logic that is responsible for the routing of users to different pages/urls	

Interface **MusicManager**	
• Handles music queries from clients. • Handles music retrieval requests from clients. • Handles music tagging requests from clients.	• RouteManager • SearchService • TaggingSystem

Interface **SearchService**	
• Queries the music database for tracks with specific parameters.	• MusicTrack

Interface **TaggingSystem**	
• Manipulate the tags database. • Query the tags database.	• Tag

Abstract **Tag**	
• Encapsulate the tag entity.	• MusicTrack

Abstract **User**	
• Encapsulate the user object/entity.	• Music Track

Abstract **MusicTrack**	
• Encapsulate the music track entity and its metadata.	• User • Tag