MP3er Software Design Specification

Group 5

Tanner Kocher, Baraa Erras, Juan Garcia, Mohamed Mohamed, Abdiwali Ahmed

1. System Overview

In an era where information and entertainment have become predominantly digital, access to diverse and affordable audio content is paramount. The realm of audio files, particularly mp3 files, encompasses music, podcasts, lectures, and more, enriching our lives, educating us, and connecting us across cultural and geographical boundaries. However, not all audiences across the globe have the luxury of easy access to such content due to regional, economic, or technical restrictions.

1.1. Purpose of the System:

The proposed system aims to bridge the access gap by providing a platform for individuals to upload, download, and search mp3 files directly via a dedicated web application. It will serve as a centralized repository of audio content, ensuring that diverse voices, music, and stories can be shared and heard globally. By offering a space where anyone, regardless of their location or economic status, can access a vast collection of audio files.

2. Development Tasks and Timeline

2.1. Task and Task Distribution

2.1.1. System Architecture

2.1.1.1. Tasks:

- Determine the server infrastructure and cloud services to use.
- Design the database schema
- Design the system's security measures, especially for data storage

2.1.1.2. Team Members:

- System Architect: Makes high-level design choices and dictates technical standards.
- Database Engineer: Focuses on how data will be stored, accessed, and updated.

2.1.2. Frontend Development

2.1.2.1. Tasks:

- Design a user-friendly interface for the application.
- Implement search, edit, upload, and download functionalities.
- Ensure the site is responsive and mobile-friendly.
- Implement user account management features.

2.1.2.2. Team Members:

- UI/UX Designer: Works on the look and feel of the platform, ensuring it's user-friendly.
- Frontend Developers: Implement the designs and ensure functionalities work as intended.

2.1.3. Backend Development

2.1.3.1. Tasks:

- Develop Restful APIs for the frontend to interact with.
- Implement upload, storage, and retrieval mechanisms for mp3 files.
- Design a system for categorizing and tagging content.
- Implement user account management on the backend side.
- Ensure security measures are in place.

2.1.3.2. Team Members:

- Backend Developers: Develop and maintain the server-side logic of the application.

2.1.4. Quality Assurance and Testing:

2.1.4.1. Tasks:

- Develop test cases based on system requirements.
- Perform various tests including unit, integration, system, and acceptance testing.
- Identify, report, and help fix bugs and issues.
- Ensure the system works as intended across different devices and browsers.

2.1.4.2. Team Members:

- QA Engineers: Conduct tests and ensure that the software is of high quality.

2.1.5. Deployment:

2.1.5.1. Tasks:

- Set up the live server environment.
- Migrate the application from the development environment to the live environment.
- Monitor the system's performance and address any immediate issues.

2.1.5.2. Team Members:

- DevOps Engineers: Handle deployment, monitoring, and maintaining the system in a live environment.

2.2. Development Timeline

Week 1-2: Project planning and requirement gathering.

Week 3-5: System design and architecture.

Week 6-10: Frontend and Backend development.

Week 11-13: Initial QA and testing.

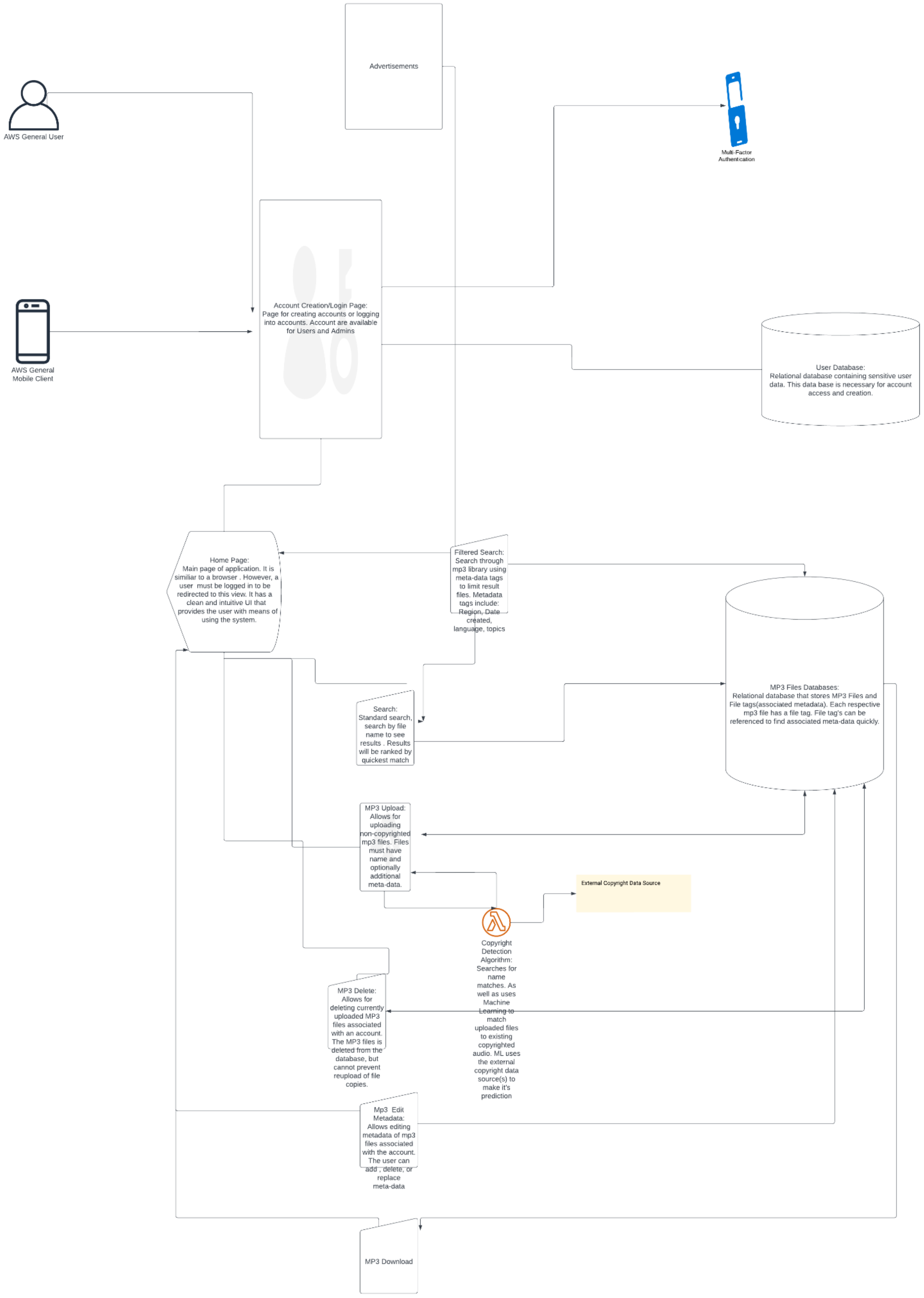
Week 14: Address issues found in testing.

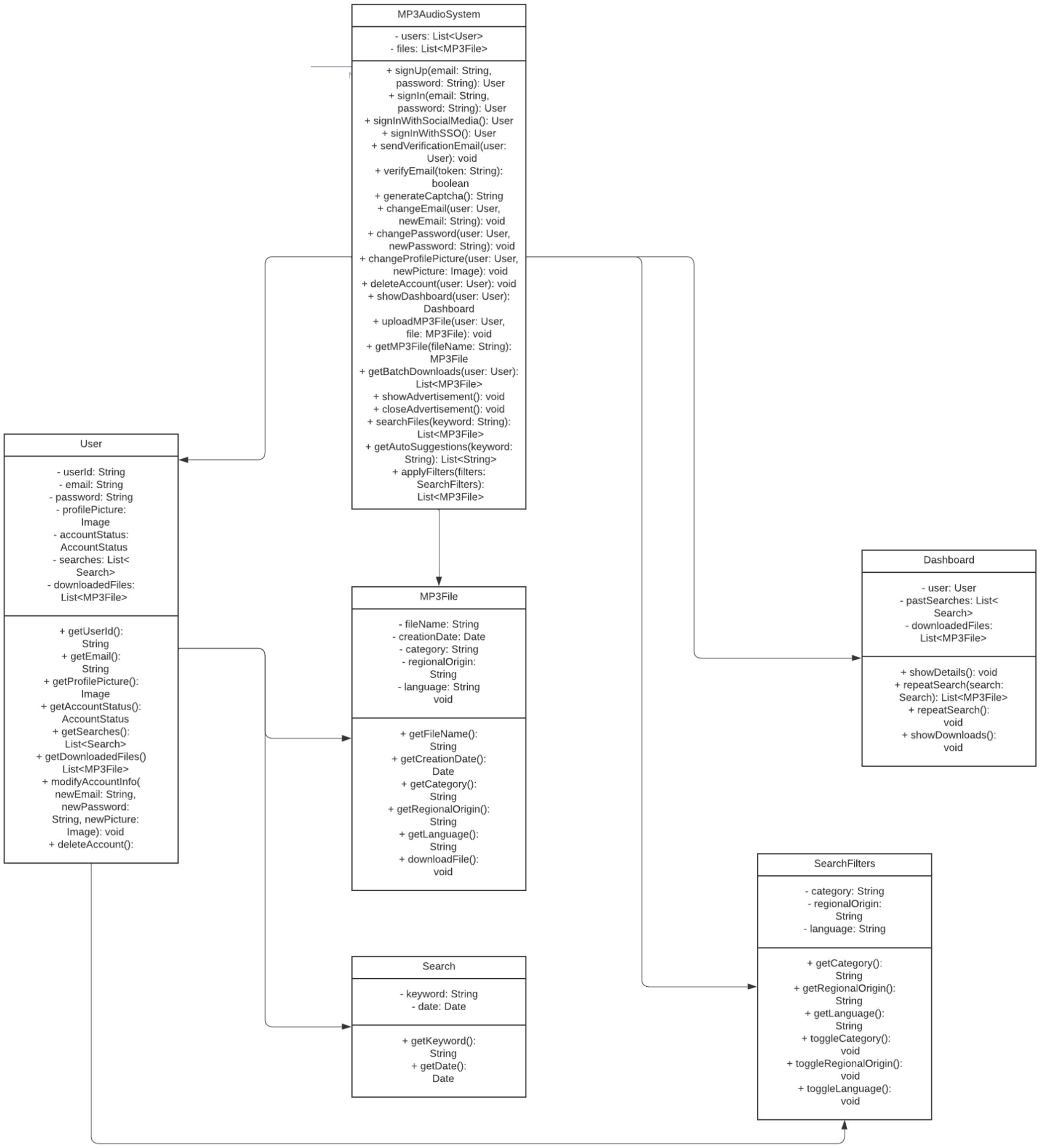
Week 15: Deployment to a staging environment.

Week 16: Final QA and any last-minute bug fixes.

Week 17: Official launch and marketing kickoff.

Week 18+: Continuous monitoring, feedback gathering, and iterative improvements.


Advertisements
AWS General User
Multi-Factor Authentication
AWS General Mobile Client
Account Creation/Login Page:
Page for creating accounts or logging into accounts. Account are available for Users and Admins
User Database:
Relational database containing sensitive user data. This data base is necessary for account access and creation.
Home Page:
Main page of application. It is similiar to a browser . However, a user must be logged in to be redirected to this view. It has a clean and intuitive UI that provides the user with means of using the system.
Filtered Search:
Search through mp3 library using meta-data tags to limit result files. Metadata tags include: Region, Date created, language, topics
MP3 Files Databases:
Relational database that stores MP3 Files and File tags(associated metadata). Each respective mp3 file has a file tag. File tag's can be referenced to find associated meta-data quickly.
Search:
Standard search, search by file name to see results . Results will be ranked by quickest match
MP3 Upload:
Allows for uploading non-copyrighted mp3 files. Files must have name and optionally additional meta-data.
External Copyright Data Source
Copyright Detection Algorithm:
Searches for name matches. As well as uses Machine Learning to match uploaded files to existing copyrighted audio. ML uses the external copyright data source(s) to make it's prediction
MP3 Delete:
Allows for deleting currently uploaded MP3 files associated with an account. The MP3 files is deleted from the database, but cannot prevent reupload of file copies.
Mp3 Edit Metadata:
Allows editing metadata of mp3 files associated with the account. The user can add , delete, or replace meta-data
MP3 Download



MP3AudioSystem

- users: List<User>
- files: List<MP3File>

+ signUp(email: String, password: String): User
+ signIn(email: String, password: String): User
+ signInWithSocialMedia(): User
+ signInWithSSO(): User
+ sendVerificationEmail(user: User): void
+ verifyEmail(token: String): boolean
+ generateCaptcha(): String
+ changeEmail(user: User, newEmail: String): void
+ changePassword(user: User, newPassword: String): void
+ changeProfilePicture(user: User, newPicture: Image): void
+ deleteAccount(user: User): void
+ showDashboard(user: User): Dashboard
+ uploadMP3File(user: User, file: MP3File): void
+ getMP3File(fileName: String): MP3File
+ getBatchDownloads(user: User): List<MP3File>
+ showAdvertisement(): void
+ closeAdvertisement(): void
+ searchFiles(keyword: String): List<MP3File>
+ getAutoSuggestions(keyword: String): List<String>
+ applyFilters(filters: SearchFilters): List<MP3File>

User

- userId: String
- email: String
- password: String
- profilePicture: Image
- accountStatus: AccountStatus
- searches: List< Search>
- downloadedFiles: List<MP3File>

+ getUserId(): String
+ getEmail(): String
+ getProfilePicture(): Image
+ getAccountStatus(): AccountStatus
+ getSearches(): List<Search>
+ getDownloadedFiles() List<MP3File>
+ modifyAccountInfo(newEmail: String, newPassword: String, newPicture: Image): void
+ deleteAccount():

MP3File

- fileName: String
- creationDate: Date
- category: String
- regionalOrigin: String
- language: String void

+ getFileName(): String
+ getCreationDate(): Date
+ getCategory(): String
+ getRegionalOrigin(): String
+ getLanguage(): String
+ downloadFile(): void

Search

- keyword: String
- date: Date

+ getKeyword(): String
+ getDate(): Date

Dashboard

- user: User
- pastSearches: List< Search>
- downloadedFiles: List<MP3File>

+ showDetails(): void
+ repeatSearch(search: Search): List<MP3File>
+ repeatSearch(): void
+ showDownloads(): void

SearchFilters

- category: String
- regionalOrigin: String
- language: String

+ getCategory(): String
+ getRegionalOrigin(): String
+ getLanguage(): String
+ toggleCategory(): void
+ toggleRegionalOrigin(): void
+ toggleLanguage(): void