SOFTWARE ARCHITECTURE PORTOFOLIO

MATA KULIAH ARSITEKTUR PIRANTI LUNAK

RICHARD DAVID TEDJA - 01082180003

INTRODUCTION

PROJECT TITLE

- The software is **Mu-Seek Audio Streaming System**
- It is an Audio Streaming System for record publishers at a record publishers association
- This system is designed to allow record publishers to publish records from their signed artists to a public cloud audio library
- The system will be composed of server-side components and client-side components
- The server-side component will manage the database operations and algorithms that produce recommendation results. The client-side components will be graphical interfaces that are integrated into the corresponding system
- The system will not facilitate direct communication between customers and the record label, but customers have the option to give feedback and reviews on a specific record.

PROJECT DESCRIPTION

- Mu-Seek is a freemium, open-source software under the GNU General Public License with the option of premium subscription
- It is a web-based application that runs on a Java-enabled mobile device
- The software implements client-server architecture and provides a system to legally share, acquire, and download audio tracks
- The client-side shall stream material from the music streaming service on the server-side through an active internet and play it through the built-in audio player
- The special feature of this product is that it has a built-in intelligent music recommender system which operates based on user behavior recorded on the client-side system and will recommend tracks or playlists based on a user's listening history, skipped songs, saved songs, playlists created, and also device location

FUNCTIONAL REQUIREMENTS

ENUMERATED FUNCTIONAL REQUIREMENTS

Identifier	Description
REQ-01	System shall allow an unregistered Streamer to provide personal information and create an account
REQ-02	System shall allow a registered Streamer subscribe as a Premium Streamer account
REQ-03	System shall allow a Premium Streamer to change payment for subscription
REQ-04	System shall allow a Premium Streamer subscribe to stop subscription
REQ-05	System shall allow a registered Streamer to view their personal information
REQ-06	System shall allow a registered Streamer to modify their personal information
REQ-07	System shall check if a Streamer is trying to claim a Streamer ID that has already been claimed by another Streamer in database
REQ-08	System shall identify a Streamer based on their Streamer ID and password
REQ-09	System shall allow a registered Streamer to subscribe to specific contents of an artist
REQ-10	System shall allow a registered Streamer to unsubscribe to specific contents of an artist

ENUMERATED FUNCTIONAL REQUIREMENTS (CONT.)

Identifier	Description
REQ-11	System shall allow a registered Streamer to close their account
REQ-12	System shall allow a registered Streamer to follow another registered Streamer
REQ-13	System shall allow a registered Streamer to unfollow another registered Streamer
REQ-14	System shall allow a registered Streamer to browse the database of tracks
REQ-15	System shall allow a registered Streamer to search for a specific track or artist in the database
REQ-16	System shall allow a registered Streamer to create a playlist containing Streamer-selected tracks
REQ-17	System shall allow a registered Streamer to modify a playlist containing Streamer-selected tracks
REQ-18	System shall allow a registered Streamer to delete a playlist containing Streamer-selected tracks
REQ-19	System shall allow a registered Streamer to view a specific playlist created by another Streamer whom they follow
REQ-20	System shall allow a registered Streamer to subscribe to a specific playlist created by another Streamer whom they follow

ENUMERATED FUNCTIONAL REQUIREMENTS (CONT.)

Identifier	Description
REQ-21	System shall allow a registered Streamer to select and play a specific track from the database or playlist
REQ-22	System shall allow a registered Streamer to give feedback to the current played track
REQ-23	System shall allow the Database Manager to add tracks to the database upon request from the record publisher
REQ-24	System shall allow the Database Manager to remove tracks from the database upon request from the record publisher
REQ-25	System shall allow a Premium Streamer to download tracks
REQ-26	System shall allow a Premium Streamer to listen downloaded tracks offline
REQ-27	System shall allow a Premium Streamers to have no advertisements unlike Non-Premium Streamers who gets advertisements every five tracks played
REQ-28	System shall allow a registered Streamer to operate the application using voice recognition
REQ-29	System shall allow a registered Streamer to browse the frequently asked questions (FAQ) page for a resolution to a problem faced
REQ-30	System shall allow a registered Streamer and Customer Support to start chat if not resolved by FAQ page

ENUMERATED FUNCTIONAL REQUIREMENTS (CONT.)

Identifier	Description
REQ-31	System shall allow a registered Streamer to file a support ticket if FAQ page has not resolved the problem faced
REQ-32	System shall allow a registered Streamer and Customer Support to close (resolve) a support ticket
REQ-33	System shall allow a registered Streamer or Customer Support to view resolved or on-going support tickets
REQ-34	System shall allow Customer Support to add frequently asked questions to FAQ page
REQ-35	System shall allow Customer Support to edit frequently asked questions at FAQ page
REQ-36	System shall allow Customer Support to remove frequently asked questions from FAQ page
REQ-37	System shall allow Customer Support to give a refund to a Streamer if needed
REQ-38	System shall allow a registered Streamer and Customer Support to comment on support ticket
REQ-39	System shall allow a registered Streamer to login using registered Streamer ID and Password
REQ-40	System shall allow a registered Streamer to logout from an active session

ENUMERATED NON-FUNCTIONAL REQUIREMENTS

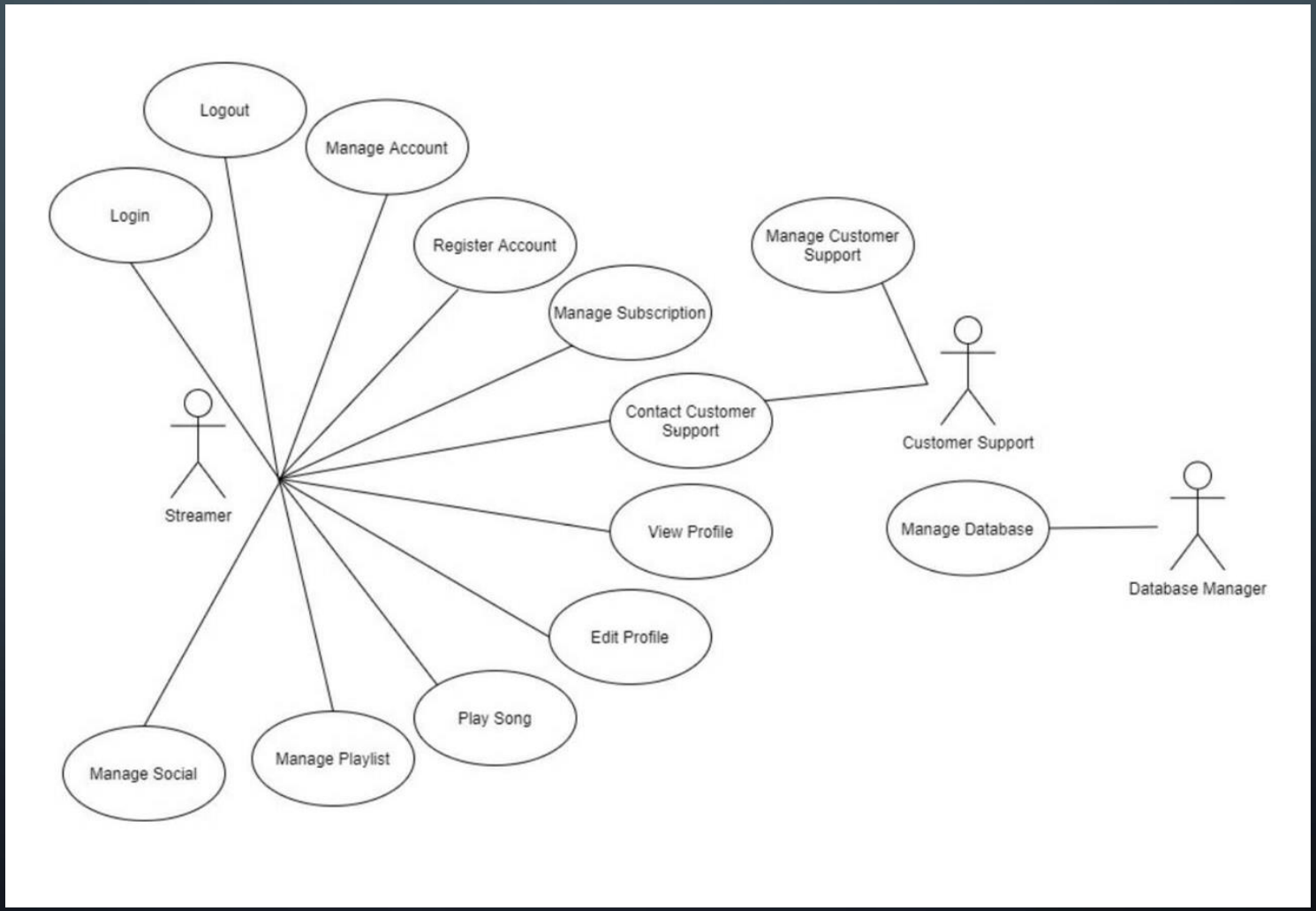
Identifier	Description
REQ-41	User is defined as Streamer, Database Manager, or Customer Support
REQ-42	User is allowed to change their Streamer ID and is notified when they attempt to do so
REQ-43	User is allowed to change their password and is notified when they attempt to do so
REQ-44	System shall allow a Registered User to access premium features through subscription
REQ-45	System shall allow the application to function through internet connection speeds of at least 128 kbps
REQ-46	System shall intelligently recommend tracks or playlists based on a user's listening history, skipped songs, saved songs, playlists created, and also device location

ON-SCREEN ENUMERATED REQUIREMENTS

Identifier	Description
REQ-47	System is designed to function on Android and iOS operating systems.
REQ-48	System must have a consistent interface across different operating system versions and screen resolutions.
REQ-49	System shall use English as the application interface language
REQ-50	System shall have an intuitive interface that supports finger gesture operations
REQ-51	System shall have a dark themed interface to protect User eyesight
REQ-52	Playback control buttons shall be light-coloured to put in contrast with the interface background

USE CASE DIAGRAM AND SCENARIO

USE CASE DIAGRAM



USE CASE SCENARIO

ID	UC-1
Title	Register Account
Description	Streamer access the System and submit an account creation request. System displays registration form. Streamer provide details required to create an account.
Scope	System
Level	Streamer goal
Primary Actor	Streamer
Stakeholder Interests	Streamer – wants to create an account
Preconditions	Streamer has a valid email address and phone number
Postconditions	Streamer is provided with an account registered on the System database

USE CASE SCENARIO (CONT.)

ID	UC-1
Main Success Scenario	1. Streamer selects “Sign Up” from menu 2. System requests Streamer details 3. Streamer provides details 4. System requests valid payment method 5. Streamer provides valid payment method 6. System sends verification to email address provided by Streamer 7. Streamer verified email 8. System records profile into database
Extensions	5a. Streamer does not want to subscribe for premium service - 5a1. System sends verification to email address provided by Streamer - 5a2. Streamer verifies email - 5a3. System records profile into database

USE CASE SCENARIO (CONT.)

ID	UC-2
Title	Login
Description:	Streamer provides valid Streamer ID and password. System authenticates Streamer based on data stored on the database. After successful authentication, Streamer is redirected to application home screen.
Scope	System
Level	User goal
Primary Actor	Streamer
Stakeholder Interests	Streamer – wants to login to their registered account
Preconditions	UC-1
Postconditions	Streamer is logged in to an active session

USE CASE SCENARIO (CONT.)

ID	UC-2
Main Success Scenario	1. Streamer selects “Sign In” from menu 2. System requests Streamer ID and password 3. Streamer provides Streamer ID and password 4. System authenticates Streamer 5. Streamer provided with an active application session
Extensions	4a. Streamer ID and/or password does not match with database record - 4a1. System displays “Authentication error, please try again” - 4a2. System requests Streamer ID and password - 4a3. Streamer either re-inputs Streamer ID and password or close the Sign In menu

USE CASE SCENARIO (CONT.)

ID	UC-3
Title	View Profile
Description:	Streamer requests to view their profile details. System displays Streamer profile details which includes their Streamer ID, subscription status, playlist library, etc.
Scope	System
Level	User goal
Primary Actor	User
Stakeholder Interests	Streamer – wants to view their profile details
Preconditions	UC-2
Postconditions	Streamer viewed their profile details

USE CASE SCENARIO (CONT.)

ID	UC-3
Main Success Scenario	1. Streamer selects “View Profile” menu 2. System displays Streamer details (name, Streamer ID, subscription status, playlist library, etc.)
Extensions	-

USE CASE SCENARIO (CONT.)

ID	UC-4
Title	Edit Profile
Description:	Streamer requests to edit their profile details. System displays profile menu. Streamer has an option to change their name, bio, and profile picture. Changes are automatically saved upon leaving the profile menu.
Scope	System
Level	User goal
Primary Actor	Streamer
Stakeholder Interests	Streamer – wants to edit their profile details
Preconditions	UC-2
Postconditions	Streamer successfully edited their profile

USE CASE SCENARIO (CONT.)

ID	UC-4
Main Success Scenario	1. Streamer selects “Edit Profile” from menu 2. System displays profile menu 3. Streamer makes changes to their profile details 4. System records changes to database
Extensions	-

USE CASE SCENARIO (CONT.)

ID	UC-5
Title	Manage Playlist
Description:	Streamer requests to create, edit or delete their playlist. Streamer could access their playlist from their library. System displays a list of playlist that the Streamer has. Streamer selects a playlist to add or remove songs from the playlist. The system displays the new playlist that has been created or modified.
Scope	System
Level	User goal
Primary Actor	Streamer
Stakeholder Interests	Streamer – wants to manage their playlist
Preconditions	UC-2
Postconditions	Streamer provided with a new and modified playlist

USE CASE SCENARIO (CONT.)

ID	UC-5
Main Success Scenario	1. Streamers selects “Library” 2. System displays Streamers playlist 3. Streamer selects a playlist 4. Streamer select add a song 5. System display suggested song 6. Streamer adds song 7. System display updated playlist
Extensions	3a. Streamer selects “Create playlist” - System request playlist name - Streamer provides playlist name - System display newly created playlist - Streamer select add a song - System display suggested song - Streamer adds song - System display updated playlist list 4a. Streamer selects “Delete playlist” - System display confirmation to delete the playlist - Streamer select delete - System display updated playlist list

USE CASE SCENARIO (CONT.)

ID	UC-6
Title	Contact Customer Support
Description:	Streamer requested for customer support. System displays frequently asked questions, asks the ask if they want to start a chat with the operator, and displays them. Streamer then interact with the operator to ask questions based on their problems. If they are unsatisfied with the solutions given, they can file a complaint ticket.
Scope	System
Level	User goal
Primary Actor	Streamer
Stakeholder Interests	Streamer – wants to troubleshoot their problem
Preconditions	UC-2
Postconditions	Passenger files a support ticket

USE CASE SCENARIO (CONT.)

ID	UC-6
Main Success Scenario	1. Passenger selects “Help and Support” from menu 2. System displays frequently asked question (FAQ) 3. System displays chatbot service 4. Passenger interacts with chatbot service 5. Passenger selects “File Complaint” 6. System displays complaint menu 7. System requests complaint information 8. Passenger provides complaint information 9. System records complaint and notifies Customer Support 10. System displays “Complaint has been filed” message
Extensions	3a. Passenger does not want to chat with chatbot - 3a1. Process is terminated 5a. Passenger is satisfied with the chatbot service - 5a1. Process is terminated

USE CASE SCENARIO (CONT.)

ID	UC-7
Title	Play Song
Description:	Streamers can stream the song they want to listen to. Streamers can choose the song from either the menu “Library”, “Browse”, or “Discover”. The “Library” contains songs compiled from created playlists, followed artists playlists, and followed friends playlists. The “Browse” menu allows the Streamer to browse Mu-Seek’s repository of songs, organized in genres. The “Discover” menu recommends songs compiled by Streamer behaviour.
Scope	System
Level	User goal
Primary Actor	Streamer
Stakeholder Interests	Streamer - wants to play song
Preconditions	UC-2
Postconditions	System plays the track selected by the Streamer

USE CASE SCENARIO (CONT.)

ID	UC-7
Main Success Scenario	1. Streamer selects “Library” from the menu 2. System displays Streamer previously saved songs 3. Streamer can choose to shuffle play or play each individual song 4. System displays the player interface 5. System plays the selected track 6. Streamer selects “Save to playlist” on the player interface 7. System displays list of Streamer created playlist 8. Streamer selects one playlist to save the song
Extensions	1a. Streamer selects “Playlists” from the menu - 1a1. This option is only available when Streamer had already created at least one playlist - 1a2. System displays list of playlists - 1a3. Streamer selects one playlist - 1a4. Streamer can choose to shuffle play or play each individual song from the selected playlist - 1a5. System displays the player interface - 1a6. System plays the selected track

USE CASE SCENARIO (CONT.)

ID	UC-7
Extensions	1b. Streamer selects “Browse” from the menu - 1b1. System displays search bar - 1b2. Streamer inputs desired artist or song title - 1b3. System displays search result - 1b4. Streamer selects a specific song to play - 1b5. Execute step 4 until 8 of Main Success Scenario 1c. Streamer selects “Discover” from the menu - 1c1. System starts intelligent music recommender module - 1c2. System displays recommended songs based on past Streamer activity - 1c3. Streamer selects a specific song to play - 1c4. Execute step 4 until 8 of Main Success Scenario

USE CASE SCENARIO (CONT.)

ID	UC-7
Extensions	6a. Streamer does not want to save song to playlist - 6a1. System displays player interface and awaits next Streamer action input 6b. Streamer with an active premium subscription requests to download song - 6b1. Streamer selects “Listen offline” on the interface - 6b2. System displays bitrate quality options - 6b3. Streamer chooses desired bitrate - 6b4. System saves currently playing song to device memory 7a. No playlist had been created - 7a1. Streamer selects “Create playlist” option - 7a2. System requests playlist name - 7a3. Streamer provides playlist name - 7a4. System saves current played song to the newly created playlist

USE CASE SCENARIO (CONT.)

ID	UC-8
Title	Manage Subscription
Description:	Streamer can view their subscription details, unsubscribe from their membership, and also view or edit their subscription payment method. Non-premium Streamers also have the option to start a premium subscription.
Scope	System
Level	User goal
Primary Actor	Streamer
Stakeholder Interests	Streamer – wants to manage their subscription
Preconditions	UC-2
Postconditions	Streamer modifies their membership details

USE CASE SCENARIO (CONT.)

ID	UC-8
Main Success Scenario	1. Streamer selects “Manage Subscription” from the menu. 2. System displays membership details 3. Streamer selects the “Change payment method” from the menu. 4. System displays list of accepted payments 5. Streamer selects preferred payment method 6. System displays confirmation dialog 7. Streamer selects “Yes” on the dialog 8. System requests payment verification token 9. Streamer provides verification token 10. System records transaction in database
Extensions	2a. Streamer is using a free account and wants to have premium subscription - 2a1. Streamer selects “Manage Subscription” from the menu. - 2a2. System displays “Start subscription?” dialog - 2a3. Streamer selects “Yes” on the dialog - 2a4. Execute steps 4 until 10 of the Main Success Scenario

USE CASE SCENARIO (CONT.)

ID	UC-8
Extensions	2b. Streamer wants to stop their premium subscription - 2b1. Streamer selects “Manage Subscription” from the menu. - 2b2. System displays membership details - 2b3. Streamer selects “Unsubscribe” from the menu. - 2b4. System displays confirmation dialog - 2b5. Streamer selects “Yes” on the dialog 9a. Unable to verify payment - 9a1. System displays “Verification failed, please try again” - 9a2. Streamer either retries verification or select another payment method

USE CASE SCENARIO (CONT.)

ID	UC-9
Title	Manage Database
Description:	In order to provide Streamer with the latest music and genre trends, a Database Manager is required to perform add and/or remove actions on the server-side song database. In addition to that, the Database Manager is responsible to analyze trending music from around the world, and the data will then be fed to the Intelligent Music Recommender System.
Scope	System
Level	User goal
Primary Actor	Database Manager
Stakeholder Interests	Database Manager - wants to add and/or remove items in database
Preconditions	UC-2
Postconditions	Database is modified

USE CASE SCENARIO (CONT.)

ID	UC-9
Main Success Scenario	1. Database Manager selects “Manage Database” from menu 2. System displays database menu 3. Database Manager selects the “Add artist” from menu. 4. System requests artist information 5. Database Manager provides artist information 6. System records artist information to database
Extensions	3a. Database Manager wants to remove artist from database - 3a1. Database Manager selects “View artists” from menu. - 3a2. System displays list of artists recorded in database - 3a3. Database Manager selects one specific artist - 3a4. System displays action menu - 3a5. Database Manager selects “Remove artist” from menu - 3a6. System displays confirmation dialog - 3a7. Database Manager selects “Yes” on the dialog - 3a8. System removes specified artist from database record

USE CASE SCENARIO (CONT.)

ID	UC-9
Extensions	3b. Database Manager wants to modify artist details - 3b1. Database Manager selects “View artists” from menu. - 3b2. System displays list of artists recorded in database - 3b3. Database Manager selects one specific artist - 3b4. System displays action menu - 3b5. Database Manager selects “Modify artist” from menu - 3b6. System requests modified artist details - 3b7. Database Manager provides modified artist details - 3b8. System records changes to database 3c. Database Manager wants to add song from a specific artist - 3c1. Database Manager selects “View artists” from menu. - 3c2. System displays list of artists recorded in database - 3c3. Database Manager selects one specific artist - 3c4. System displays action menu - 3c5. Database Manager selects “Add song” from menu - 3c6. System requests song upload - 3c7. Database Manager uploads song to database - 3c8. System records changes to database

USE CASE SCENARIO (CONT.)

ID	UC-9
Extensions	3d. Database Manager wants to remove song from database - 3d1. Database Manager selects “View artists” from menu. - 3d2. System displays list of artists recorded in database - 3d3. Database Manager selects one specific artist - 3d4. System displays action menu - 3d5. Database Manager selects “View songs” from menu - 3d6. System displays list of songs from a specified artist - 3d7. Database Manager selects one specific song - 3d8. System displays action menu - 3d5. Database Manager selects “Remove song” from menu - 3d8. System displays confirmation dialog - 3d9. Database Manager selects “Yes” on the dialog - 3d10. System removes specified song from database record

USE CASE SCENARIO (CONT.)

ID	UC-9
Extensions	3e. Database Manager wants to update music trends - 3e1. Database Manager selects “Update Trends” from menu. - 3e2. System requests music trend calculation - 3e3. Database Manager provides music trend calculation from the last 24 hours - 3e4. System analyzes trend data - 3e5. System generates automated smart music chart - 3e6. System records generated chart to database - 3e7. System inputs trend data to Intelligent Music Recommender module

USE CASE SCENARIO (CONT.)

ID	UC-10
Title	Manage Customer Support
Description:	The presence of Customer Support agents is necessary to guarantee Streamer satisfaction towards the application. The aforementioned agents are responsible for troubleshooting Streamer complaint tickets, generating and updating FAQ repositories, processing Streamer refund requests.
Scope	System
Level	User-goal
Primary Actor	Customer Support
Stakeholder Interests	Customer Support - wants to help Streamer troubleshoot problems related to the application
Preconditions	UC-2
Postconditions	Issues raised by the Streamer are addressed

USE CASE SCENARIO (CONT.)

ID	UC-10
Main Success Scenario	1. Customer Support selects “Support Action” from menu 2. System displays action menu 3. Customer Support selects the “Complaint Tickets” from menu. 4. System displays list of complaint tickets 5. Customer Support selects one specific ticket 6. System displays ticket details 7. System requests ticket feedback 8. Customer Support provides ticket feedback 9. System marked ticket as resolved 10. System notifies Streamer
Extensions	3a. Customer Support wants to update FAQ repository - 3a1. Customer Support selects “Update FAQ” from menu. - 3a2. System displays FAQ repository data - 3a3. System requests repository modification - 3a4. Customer Support modifies repository - 3a5. System records changes to database

USE CASE SCENARIO (CONT.)

ID	UC-10
Extensions	8a. Transaction refund is required - 8a1. Customer Support selects “Process Refund” on the ticket interface - 8a2. System requests refund amount - 8a3. Customer Support provides refund amount - 8a4. System refunds specified amount to Streamer account - 8a5. System records transaction to database

USE CASE SCENARIO (CONT.)

ID	UC-11
Title	Manage Social
Description:	Ease of social interactions is a key for building a healthy community within the application. Streamers are able to follow or unfollow artists, add or remove other Streamers as friends, and keep updated with the latest music trends by viewing the daily smart music chart
Scope	System
Level	User-goal
Primary Actor	Streamer
Stakeholder Interests	Streamer - wants to manage their social interactions
Preconditions	UC-2
Postconditions	Streamer controls whom they interact with

USE CASE SCENARIO (CONT.)

ID	UC-11
Main Success Scenario	1. Streamer selects “Manage Social” from menu 2. System displays action menu 3. Streamer selects “Search artists” from menu. 4. System displays search bar 5. System requests search term 6. Streamer provides search term 7. System displays list of search results 8. Streamer selects “Follow Artist” on one specific artist
Extensions	3a. Streamer wants to unfollow a specific artist - 3a1. Streamer selects “Followings” from menu - 3a2. System displays list of followed artists - 3a3. Streamer selects one specific artist - 3a4. System displays action menu - 3a5. Streamer can either unfollow or block the specified artist

USE CASE SCENARIO (CONT.)

ID	UC-11
Extensions	3b. Streamer wants to add another Streamer as a friend - 3b1. Streamer selects “Search Friend” from menu - 3b2. System displays search bar - 3b3. System requests search term - 3b4. Streamer provides search term - 3b5. System displays list of search results - 3b6. Streamer selects “Add as Friend” on one specific Streamer 3c. Streamer wants to unfriend a specific Streamer - 3c1. Streamer selects “Friends” from menu - 3c2. System displays list of friends - 3c3. Streamer selects one specific Streamer - 3c4. System displays action menu - 3c5. Streamer can either unfollow or block the specified Streamer

USE CASE SCENARIO (CONT.)

ID	UC-11
Extensions	3d. Streamer wants send a message to another Streamer on their friend list - 3d1. Streamer selects “Friends” from menu - 3d2. System displays list of friends - 3d3. Streamer selects one specific Streamer - 3d4. System displays action menu - 3d5. Streamer selects “Message” from menu - 3d6. System displays messaging interface - 3d7. System requests message text - 3d8. Streamer provides message text - 3d9. System sends message text to specified Streamer 3e. Streamer wants to view trending music chart - 3e1. Streamer selects “Charts” from menu - 3e2. System requests date - 3e3. Streamer either inputs date or selects “Today” - 3e4. System displays smart music chart for the corresponding date

USE CASE SCENARIO (CONT.)

ID	UC-12
Title	Manage Account
Description:	As required by law, Streamers of an application have full control over their privacy and account status. Streamers are able to temporarily deactivate their account, or close their account completely. Streamers also have an option to change their Streamer ID, password, and account privacy status.
Scope	System
Level	User-goal
Primary Actor	Streamer
Stakeholder Interests	Streamer - wants to control their account privacy
Preconditions	UC-2
Postconditions	Streamer modified their account

USE CASE SCENARIO (CONT.)

ID	UC-12
Main Success Scenario	1. Streamer selects “Manage Account” from menu 2. System displays action menu 3. Streamer selects “Remove Account” from menu. 4. System requests password verification 5. Streamer provides password verification 6. System displays confirmation dialog 7. Streamer selects “Yes” on the dialog 8. System displays “We’re sad to see you go!” message 9. System removes Streamer account from database after 30 days
Extensions	3a. Streamer wants to temporarily deactivate their account - 3a1. Streamer selects “Deactivate Account” from menu - 3a2. System displays duration menu (24 hours, 7 days, 30 days, 1 year) - 3a3. Streamer selects desired duration - 3a4. System displays confirmation dialog - 3a5. Streamer selects “Yes” on the dialog - 3a6. System disables online features for the length of the specified duration

USE CASE SCENARIO (CONT.)

ID	UC-12
Extensions	3b. Streamer wants to change their Streamer ID - 3b1. Streamer selects “Change Streamer ID” from menu - 3b2. System requests desired Streamer ID - 3b3. Streamer provides desired Streamer ID - 3b4. System checks database for conflicts - 3b5. System records changes to database - 3b6. If database conflict is detected, execute step 3b2 until 3b4 3c. Streamer wants to change their password - 3b1. Streamer selects “Change Password” from menu - 3b2. System requests modified password - 3b3. Streamer provides modified password - 3b4. System requests password confirmation (re-input) - 3b5. Streamer re-input modified password - 3b6. System records changes to database - 3b7. If password verification fails, execute step 3b2 until 3b5

USE CASE SCENARIO (CONT.)

ID	UC-12
Extensions	3d. Streamer wants to change their account privacy status - 3b1. Streamer selects “Privacy Settings” from menu - 3b2. System displays privacy options (public account, private account) - 3b3. Streamer selects desired privacy setting - 3b4. System records changes to database

USE CASE SCENARIO (CONT.)

ID	UC-13
Title	Logout
Description:	Streamers have an option to logout from an active session, in case they want to switch accounts. The next time they want to use the application, they will need to login again.
Scope	System
Level	User-goal
Primary Actor	Streamer
Stakeholder Interests	Streamer - wants to logout from an active session
Preconditions	UC-2
Postconditions	Streamer is logged out

USE CASE SCENARIO (CONT.)

ID	UC-13
Main Success Scenario	1. Streamer selects “Logout” from menu 2. System displays confirmation dialog 3. Streamer selects “Yes” on the dialog 4. System ends Streamer active session 5. System disconnects Streamer from server 6. System displays login interface
Extensions	-

QUALITY ATTRIBUTE REQUIREMENTS

AVAILABILITY GENERAL SCENARIO

Scenario Portion	Possible Values
Source of Stimulus	Internal to system: • Hardware access protocols • System resource External to system: • Human errors • Network communication
Stimulus	• Client connection to Server cannot be established • Permission to access hardware audio modules are denied • No audio output device detected • Insufficient device memory • Insufficient device storage
Environment	• Startup • Normal operation

AVAILABILITY GENERAL SCENARIO (CONT.)

Scenario Portion	Possible Values
Artifact	• Streamer interface • Network communication modules • Audio signal processing modules • Media player interface
Response	• Log faults • Notify user • Temporary unavailable • Retry access with a defined interval period • Continue to operate in degraded mode
Response Measure	• Number of attempts to restore availability with every 10 seconds interval in degraded mode • Proportion of certain module that the system handles without failing

INTEROPERABILITY GENERAL SCENARIO

Scenario Portion	Possible Values
Source of Stimulus	• A system
Stimulus	• A request to exchange information among Server-side system and the record publisher system
Environment	• Systems wishing to interoperate are discovered at run time

INTEROPERABILITY GENERAL SCENARIO (CONT.)

Scenario Portion	Possible Values
Artifact	• Server-side system • Record publisher system
Response	• Request is appropriately rejected and database manager is notified • Request is appropriately accepted and information is exchanged successfully • Request is logged by Server-side database
Response Measure	• Percentage of information exchanges correctly processed • Percentage of information exchanges correctly rejected

MODIFIABILITY GENERAL SCENARIO

Scenario Portion	Possible Values
Source of Stimulus	• Streamer • Database Manager • Developer
Stimulus	• Perform add, delete, and/or modify operations
Environment	• Design time • Run time

MODIFIABILITY GENERAL SCENARIO (CONT.)

Scenario Portion	Possible Values
Artifact	• Application code • User data • Database
Response	• Locate components within the architecture to be modified • Modification is made without affecting other functionalities (no side effect) • Modification is tested before deployment • Deploys modification
Response Measure	• Number of components affected • Cost of modification • Time required to perform modification operations • Number of functionalities affected

PERFORMANCE GENERAL SCENARIO

Scenario Portion	Possible Values
Source of Stimulus	• Streamer • Database Manager
Stimulus	• Initiate data transfer • Access Server-side resources
Environment	• Normal operations • Server-side overloaded operations

PERFORMANCE GENERAL SCENARIO (CONT.)

Scenario Portion	Possible Values
Artifact	• System • Network communication modules
Response	• Processes stimuli • Provide requested services
Response Measure	• Latency • Data transfer rate • Data transfer duration • Data loss percentage

SECURITY GENERAL SCENARIO

Scenario Portion	Possible Values
Source of Stimulus	• Correctly identified individual or service • Incorrectly identified individual or service • Unidentified individual or service
Stimulus	• Attempts to perform add, delete, and/or modify operations • Attempts to display information • Attempts to access system services • Attempts to reduce availability to system services
Environment	• Normal operations

SECURITY GENERAL SCENARIO (CONT.)

Scenario Portion	Possible Values
Artifact	• Data stored on Server-side system
Response	• Log events • Maintain audit trail • Authenticates the user • Grants or withdraws permission to access data and/or services • Allow or disallow attempts to access or modify data and/or services
Response Measure	• Resources to prevent unauthorized security operations • Success rate of implemented resources • Percentage of data and/or services still available during a security attack • Percentage of data and/or services unavailable after a security attack • Time required to restore data and/or services

TESTABILITY GENERAL SCENARIO

Scenario Portion	Possible Values
Source of Stimulus	• Unit testers • Integration testers • System testers • Users
Stimulus	• Performs test on a system unit / module • Performs test on a completed integration of a subsystem • Performs test on a completed implementation of a system • Problems and bugs found after complete software deployment
Environment	• Design time • Integration time • Completion of a component • Normal user operations

TESTABILITY GENERAL SCENARIO (CONT.)

Scenario Portion	Possible Values
Artifact	• Specific system component • Complete system
Response	• Component has behavior control interface • Execute tests and observe results • Monitor the state of the system • Capture activities that resulted in faults / errors
Response Measure	• Time and effort to detect faults / errors • Time to perform tests • Length of time to prepare tests and test environments • Probability of reduction in risk exposure

USABILITY GENERAL SCENARIO

Scenario Portion	Possible Values
Source of Stimulus	• End user
Stimulus	• Learn system features • Use system efficiently and effectively • Minimize impact of errors • Adapt and feel comfortable with the system
Environment	• At runtime on normal operations

USABILITY GENERAL SCENARIO (CONT.)

Scenario Portion	Possible Values
Artifact	• System
Response	**Learn system features** • Provide comprehensive FAQ repository • Interface is familiar end user. • Interface that is usable in an unfamiliar context **Use system efficiently and effectively** • Reuse already entered data • Aggregate frequently used commands • Allow multiple simultaneous activities **Minimize the impact of errors** • Recognize user error • Utilize undo or cancel action • Verify system resources **Adapt and feel comfortable with the system** • Provide customizability
Response Measure	• Number of tasks accomplished • Percentage of user satisfaction • Ratio of successful operations to total operations • Amount of time and/or data lost due to errors

PORTABILITY GENERAL SCENARIO

Scenario Portion	Possible Values
Source of Stimulus	• End user
Stimulus	• Application is run on multiple platforms
Environment	• At runtime on normal operations

PORTABILITY GENERAL SCENARIO (CONT.)

Scenario Portion	Possible Values
Artifact	• Client application
Response	• Install application on multiple platforms • Run application on each platform • Configure system according to host environment
Response Measure	• Number of faults during runtime • Ratio of successful operations to total operations • Amount of time and/or data lost due to errors

MOBILITY GENERAL SCENARIO

Scenario Portion	Possible Values
Source of Stimulus	• End user
Stimulus	• Application is run on multiple devices
Environment	• At runtime on normal operations

MOBILITY GENERAL SCENARIO (CONT.)

Scenario Portion	Possible Values
Artifact	• Client application
Response	• Install application on multiple devices • Run application on each device • Configure system according to host environment
Response Measure	• Number of faults during runtime • Number of adaptation errors during runtime • Ratio of successful operations to total operations • Amount of time and/or data lost due to errors

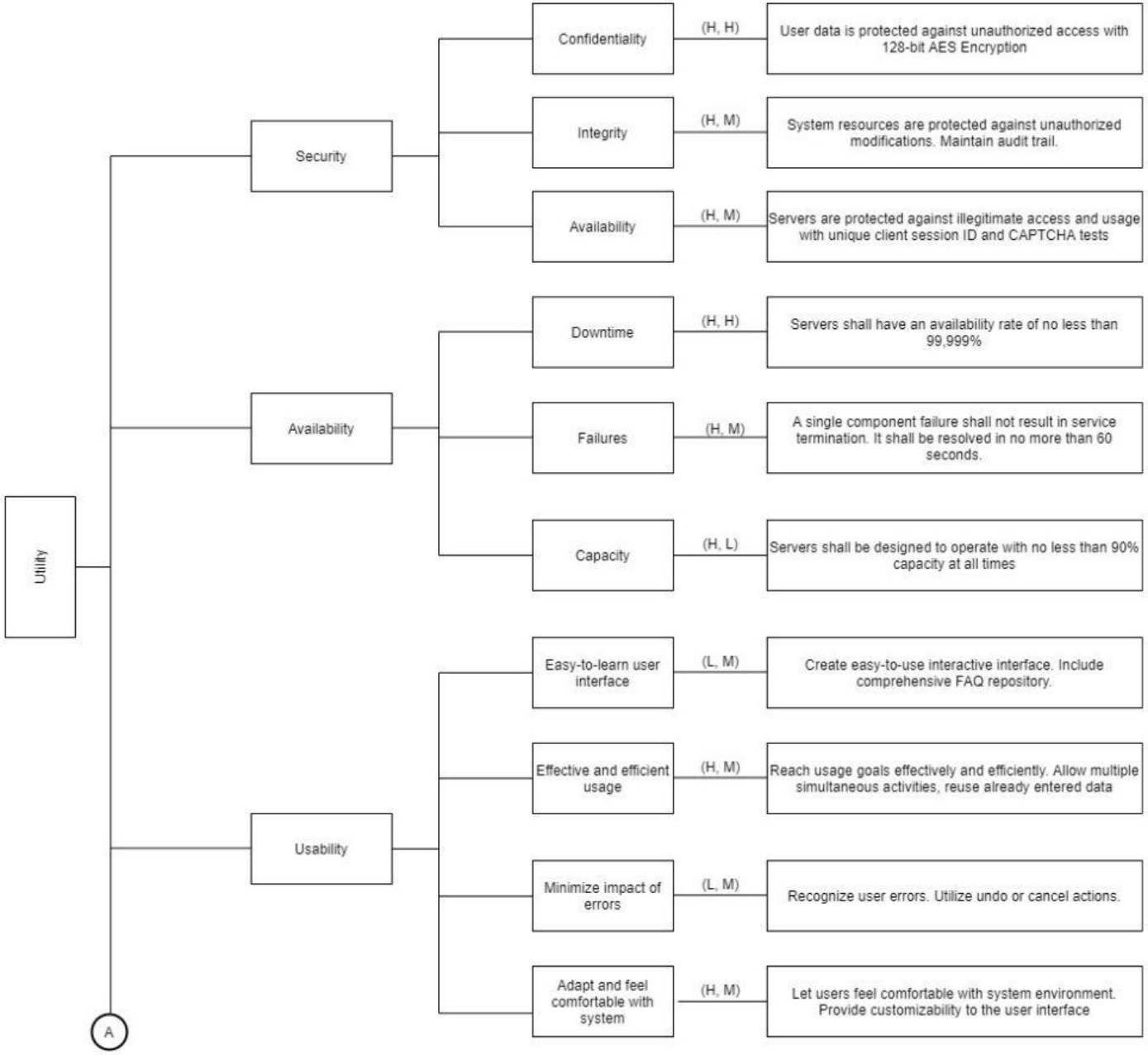
GENERAL CONSTRAINTS

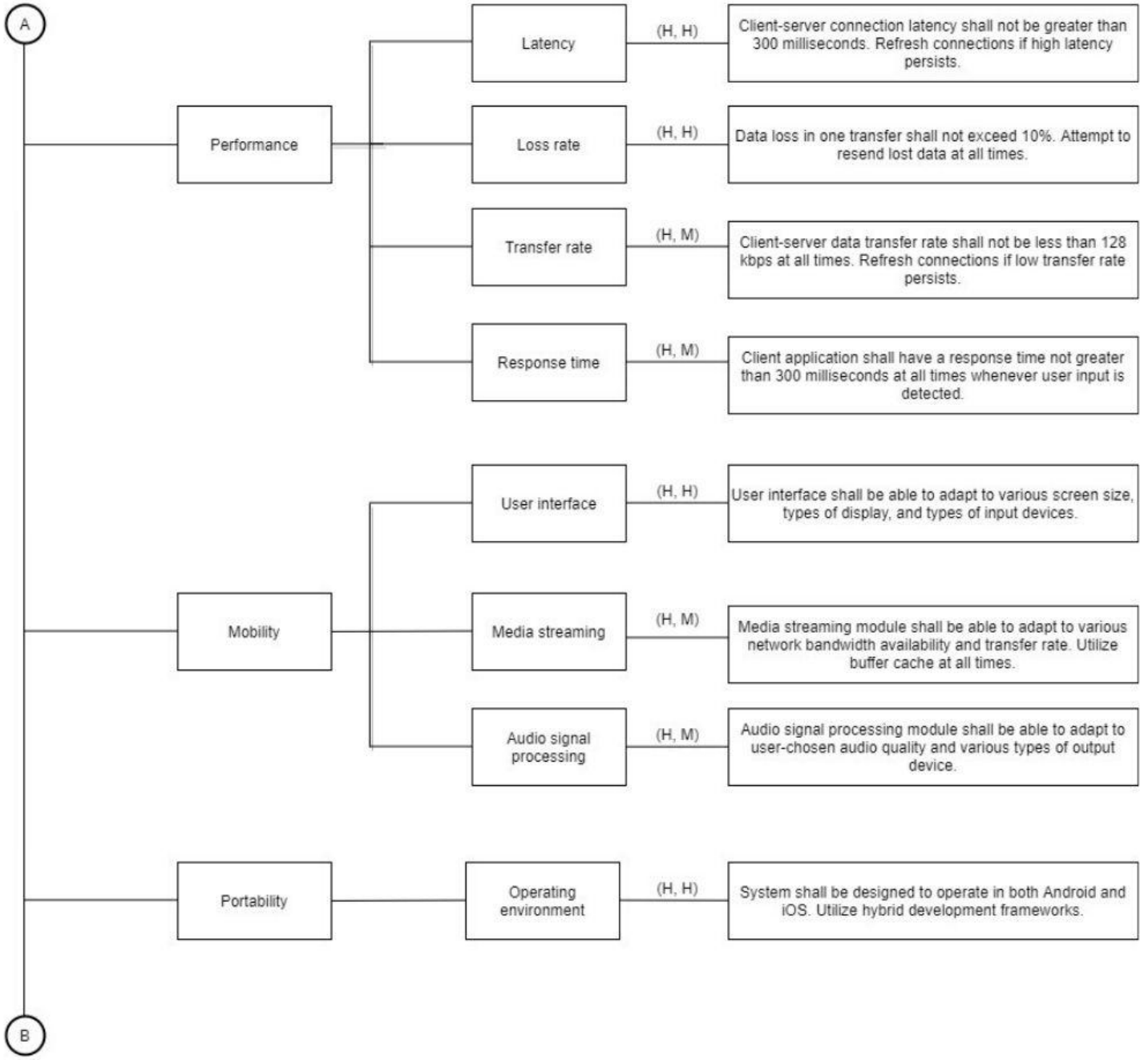
- User profile data usage during development requires developers to pay attention to regulatory policies which leads to difficulties in real time and sufficient data
- Millions of data are required for testing which requires huge amounts of disk space and clusters
- Gathering user data in real time requires the application to be reliable and safe to carry out this task. The system must be able to produce new data depending on user behavior and provide security towards user data
- Providing accurate recommendations. It is critical that expected accuracy and handling with sparse and huge data at the same time is possible to provide accurate recommendations

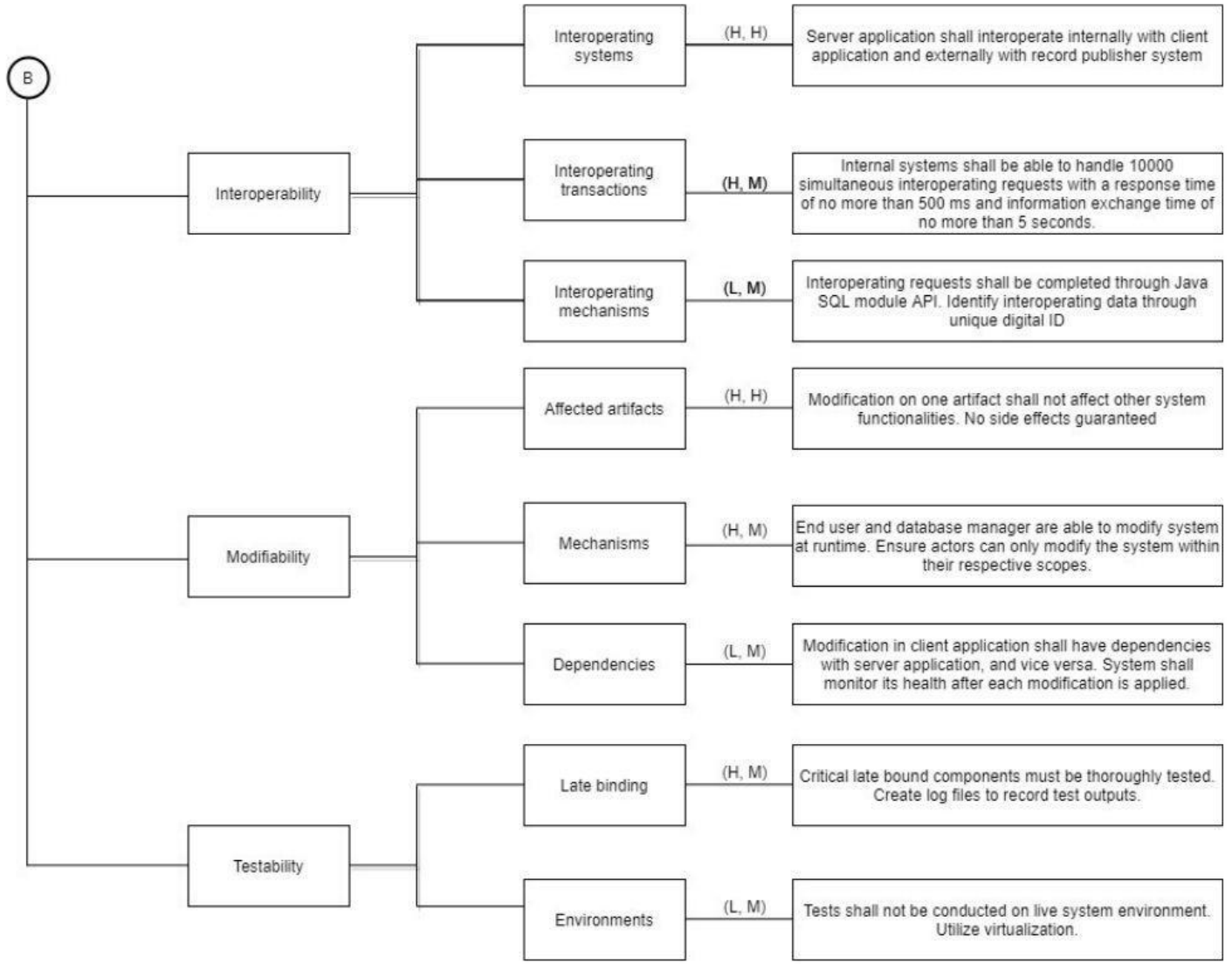
TECHNICAL AND OTHER CONSTRAINTS

- System shall have database which consists of track data and user data, well-maintained by approved database management tools
- System shall strongly depend on internet connection to run key features
- System shall have a central repository on the server-side for storing tracks provided by record publishers
- System shall provide a way to recommend music that caters to each user's music preference by tracking user's listening history, skipped songs, saved songs, playlists created, and also device location
- System shall provide a way to let record publishers know the trend in music by tracking user's feedback, listening history, saved songs, and skipped songs
- System shall be a freemium, open-source software under the GNU General Public License with the option of premium subscription
- System shall be a web-based application that runs on a Java-enabled mobile device
- System shall implement client-server architecture

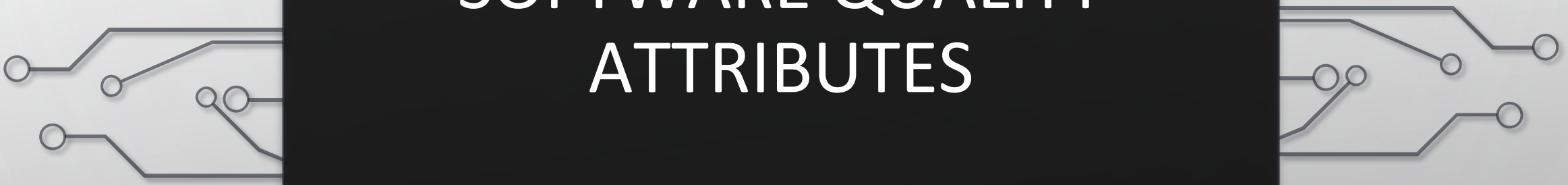
UTILITY TREE


Utility
Security
Availability
Usability
A
Confidentiality
(H, H)
User data is protected against unauthorized access with 128-bit AES Encryption
Integrity
(H, M)
System resources are protected against unauthorized modifications. Maintain audit trail.
Availability
(H, M)
Servers are protected against illegitimate access and usage with unique client session ID and CAPTCHA tests
Downtime
(H, H)
Servers shall have an availability rate of no less than 99,999%
Failures
(H, M)
A single component failure shall not result in service termination. It shall be resolved in no more than 60 seconds.
Capacity
(H, L)
Servers shall be designed to operate with no less than 90% capacity at all times
Easy-to-learn user interface
(L, M)
Create easy-to-use interactive interface. Include comprehensive FAQ repository.
Effective and efficient usage
(H, M)
Reach usage goals effectively and efficiently. Allow multiple simultaneous activities, reuse already entered data
Minimize impact of errors
(L, M)
Recognize user errors. Utilize undo or cancel actions.
Adapt and feel comfortable with system
(H, M)
Let users feel comfortable with system environment. Provide customizability to the user interface


A
Performance
Latency
(H, H)
Client-server connection latency shall not be greater than 300 milliseconds. Refresh connections if high latency persists.
Loss rate
(H, H)
Data loss in one transfer shall not exceed 10%. Attempt to resend lost data at all times.
Transfer rate
(H, M)
Client-server data transfer rate shall not be less than 128 kbps at all times. Refresh connections if low transfer rate persists.
Response time
(H, M)
Client application shall have a response time not greater than 300 milliseconds at all times whenever user input is detected.
Mobility
User interface
(H, H)
User interface shall be able to adapt to various screen size, types of display, and types of input devices.
Media streaming
(H, M)
Media streaming module shall be able to adapt to various network bandwidth availability and transfer rate. Utilize buffer cache at all times.
Audio signal processing
(H, M)
Audio signal processing module shall be able to adapt to user-chosen audio quality and various types of output device.
Portability
Operating environment
(H, H)
System shall be designed to operate in both Android and iOS. Utilize hybrid development frameworks.
B


B
Interoperability
Interoperating systems
(H, H)
Server application shall interoperate internally with client application and externally with record publisher system
Interoperating transactions
(H, M)
Internal systems shall be able to handle 10000 simultaneous interoperating requests with a response time of no more than 500 ms and information exchange time of no more than 5 seconds.
Interoperating mechanisms
(L, M)
Interoperating requests shall be completed through Java SQL module API. Identify interoperating data through unique digital ID
Modifiability
Affected artifacts
(H, H)
Modification on one artifact shall not affect other system functionalities. No side effects guaranteed
Mechanisms
(H, M)
End user and database manager are able to modify system at runtime. Ensure actors can only modify the system within their respective scopes.
Dependencies
(L, M)
Modification in client application shall have dependencies with server application, and vice versa. System shall monitor its health after each modification is applied.
Testability
Late binding
(H, M)
Critical late bound components must be thoroughly tested. Create log files to record test outputs.
Environments
(L, M)
Tests shall not be conducted on live system environment. Utilize virtualization.

SOFTWARE QUALITY ATTRIBUTES



AVAILABILITY

Design Checklist	Action
Allocation of Responsibilities	• Following components need to be highly available: - Database - Network communication channels - Media capture and streams API - Essential client authentication components • If fault is detected on one of the above components: - Log the fault - Notify server administrator - Disable services to the affected module - Display error message to client - In some cases, operate in degraded (offline) mode
Coordination Model	• Before attempting to submit request, ping the server • Retry ping attempt when no response is detected • Resend requests when incorrect response is received • Server monitors the health of database components • Rollback if faults are detected within database structures • Store received stream in buffer file if internet connection is unreliable

AVAILABILITY (CONT.)

Design Checklist	Action
Data Model	• Database containing user data and track data needs to be highly available • Incorrect request from client may trigger incorrect server response • Incorrect database configuration may lead to a crash within client environment • In an event of a fault: - Store submitted database write requests in cache, push to database when restored - Temporary disable new incoming write requests - Automatically switch to backup database - Operate in degraded (offline) mode
Mapping Among Architectural Elements	• If fault in server processor is detected, allocate processing to next available machine • If fault in main database is detected, switch to backup database and notify gateways • If fault in gateway is detected, re-assign communication protocols to another gateway • Prioritize masking the fault i.e. switching to backup artifacts, before temporarily disabling processes altogether

AVAILABILITY (CONT.)

Design Checklist	Action
Resource Management	• Fault occurred in client application, operations cannot continue, restart application • Fault occurred in server application: - Omission: attempt to mask the fault, continue operations in degraded mode until fix is applied - Crash: server halts operation and be temporarily unavailable - Server shall have buffer cache large enough to accommodate pending client write requests in an event of a failure
Binding Time	• Fault detection and recovery mechanisms are designed to work with components that are bound later than compile time • Recovery mechanisms are triggered at the exact moment a fault is flagged • If late binding of a component fails, flag that component as temporarily unavailable. System may operate at a degraded mode.

AVAILABILITY (CONT.)

Design Checklist	Action
Choice of Technology	• Database Management Tools are able to detect faults on database structures • Cloudflare is able to detect faults in network communication channels • Client application and server application maintains separate event log

INTEROPERABILITY

Design Checklist	Action
Allocation of Responsibilities	• Detect a correct request to exchange information and accept the request • Reject request if appropriate conditions are not met • Exchange required information • Record information in database • Log interoperating events
Coordination Model	• System is able to handle 10000 simultaneous interoperating requests • Systems shall interoperate using consistent communication protocols • Response time is under 0,5 ms • Transaction shall be completed in under 5 seconds • Three-way handshake is utilized • No information loss shall be tolerated

INTEROPERABILITY (CONT.)

Design Checklist	Action
Data Model	• Interoperating data shall only be made known by sending and receiving modules respectively • Data shall be treated as confidential • 128-bit encryption is utilized • Systems shall interoperate using information directly interpretable by both parties
Mapping Among Architectural Elements	• Interoperating components are mapped to machines specifically allocated for interoperating operations • Interoperating systems shall utilize the same API for data exchange processes through a specific network gateway

INTEROPERABILITY (CONT.)

Design Checklist	Action
Resource Management	• Interoperating request limit has been calculated to prevent denial-of-service for legitimate users trying to access the database • Backup machines are available to assist in information exchange in case interoperating requests exceeds the pre-defined limit
Binding Time	• Interoperating systems became known to each other when a request is sent to the external interoperating system • System shall only interoperate with requests coming from within the application • System shall reject untrusted interoperating requests i.e. not coming from the application • System shall log all incoming requests, whether accepted or rejected

INTEROPERABILITY (CONT.)

Design Checklist	Action
Choice of Technology	• Interoperability in this software is supported using Java SQL Module • Java.SQL provides the API for accessing and processing data stored in a data source (usually a relational database) using the Java programming language. • Expose data and data model using API • Identify interoperating data using unique digital ID

MODIFIABILITY

Design Checklist	Action
Allocation of Responsibilities	• Types of changes: - Adding a new feature - Modifying existing feature - Removing existing feature • Regression testing is required when existing source code is modified • Do not make modification on a live server environment. Use virtualization instead
Coordination Model	• Network communication protocols are likely to change at runtime: - Source code shall be able to accommodate protocols defined at runtime - Such changes only affects communication features between client and server • Information being exchanged shall not be modified under any conditions

MODIFIABILITY (CONT.)

Design Checklist	Action
Data Model	• End user may modify the following settings: - Audio quality - Screen size and resolution - Input and output devices • Database Manager may modify the following settings: - Manual database entry modification
Mapping Among Architectural Elements	• Changes should not be made directly to server environment • Modification execution in client application shall depend on server application, and vice versa • Modification in server application may affect assignment of data to database

MODIFIABILITY (CONT.)

Design Checklist	Action
Resource Management	• Modification of interoperability modules may affect performance and resource usage • Test modifications to meet system requirements • Encapsulate resource policies with resource manager
Binding Time	• Defer binding tactic is used • Resource files binding is done at startup i.e. when the registration module is called • Negotiation of communication protocols is done at runtime

MODIFIABILITY (CONT.)

Design Checklist	Action
Choice of Technology	• Database Management Tools assist in testing changes made to database structure, supports high modifiability • Middleware Tools are used to bridge application to database, supports high modifiability • Java SQL API is used to exchange information between User and System, it does not support high modifiability

PERFORMANCE

Design Checklist	Action
Allocation of Responsibilities	• Network communication channels shall be heavily-used • Communication gateways shall be heavily-loaded • Database operations are time-critical • Client module shall have a response time of under 0,5 seconds • Information exchange requests shall be either accepted or rejected within 0,3 seconds • Information exchange between client and server shall be done in 0,5 seconds • Database mapping should be done in under 3 seconds
Coordination Model	• Network gateway must periodically listen and be ready to receive incoming requests • Supports high concurrency • Client-server communications are synchronous • Since all requests are equal, utilize FIFO scheduling method

PERFORMANCE (CONT.)

Design Checklist	Action
Data Model	• Information exchange between client and server is time-critical • Additional processing resources to support parallel information exchange request processing can be added to server to ease bottlenecks • Partitioning data to smaller sizes would benefit performance
Mapping Among Architectural Elements	• Applying concurrency is feasible and have significant positive effect on reducing response time • Database operations are assigned to processors with the most processing capacity • Co-locating database middleware with the database itself would benefit performance

PERFORMANCE (CONT.)

Design Checklist	Action
Resource Management	• In addition to concurrency, utilize a load balancer to divide workload between servers • Treat all requests for resources as equal, utilize FIFO scheduling method • Lock access to resources that will be modified • Deploy additional resources to ease bottlenecks depending on current workload
Binding Time	• Network communication protocols should be bind in 0,5 seconds after successful negotiation

PERFORMANCE (CONT.)

Design Checklist	Action
Choice of Technology	• Java programming language used supports the following performance requirements: - Response deadlines when client submits a registration request - FIFO scheduling policies, earliest client request gets to be processed first - Concurrency and load balancing mechanisms, a maximum of 10000 simultaneous requests can be processed - Call additional resources when system is under heavy load

SECURITY

Design Checklist	Action
Allocation of Responsibilities	• Data exchange process must be secure and treated as confidential - End-to-end encryption shall be utilized - Server shall be able to identify request origin, deny requests not originating from client application - Client shall not send connection attempt if three-way handshake is unable to verify the encryption of communication channels used • Passwords must be hashed - Server shall verify hash values
Coordination Model	• Network communication protocols are used to communicate between client and server - It is used to exchange data - Client encapsulates request using unique session ID - Server authenticates session ID to verify that the request originates from legitimate client application - Three-way handshake is used to prevent data loss - 128-bit encryption is used during exchange process - Server shall flag and deny suspicious multiple requests incoming from the same origin

SECURITY (CONT.)

Design Checklist	Action
Data Model	• Client data exchanged is end-to-end encrypted • Encryption key is transmitted separately • Only database middleware module is able to decrypt information received • No third parties of any kind shall be able to access exchanged data • All information exchange process shall be logged • Multiple database backups shall exist, in event of a security failure and data restoration is required
Mapping Among Architectural Elements	• Modules shall be mapped to the following elements, in which each element has its own responsibilities: - Generate unique session ID - Encrypt information - Encapsulate data before sending request - Authenticate request origin, then accept or deny request - Decrypt information exchanged - Record or log exchange processes - Detect potential security intrusion and execute necessary recovery mechanisms

SECURITY (CONT.)

Design Checklist	Action
Resource Management	• Identification, authentication, and authorization is entirely done within database middleware • A separate module within the server shall be responsible to monitor and log resource activities • Multiple independent database middleware are running simultaneously to serve as backups when main middleware is compromised
Binding Time	• Qualified late bound components must satisfy the following requirements and abilities: - Unique session ID to verify request origin - Keys to encrypt and decrypt exchanged information - Log all requests, including origin IP address and location - Flag denied requests, assign temporary cooldown to origin IP address - Block suspicious requests, assign permanent cooldown to origin IP address

SECURITY (CONT.)

Design Checklist	Action
Choice of Technology	• 128-bit AES Encryption is utilized • 160-bit SHA-1 unique PHP session ID • Utilization of database encryption middleware • SSL / Start TLS

TESTABILITY

Design Checklist	Action
Allocation of Responsibilities	• Critical responsibilities: - Encryption and decryption mechanisms - Session ID generation and authentication - Database read / write components - Media read and stream components • Less critical responsibilities: - Network protocol negotiation - Network communication channels and gateway - Three-way handshake mechanisms - Client account registration form - User interface modules
Coordination Model	• Test suites shall include the following conditions: - Decryption using a mismatched key - Authentication using incorrect session ID - Read / write operations to main and backup databases - Calculate response time to meet performance requirements - Handling incoming requests originating from flagged or blocked IP addresses - Overloading communication channels and gateways - Stressing database operations

TESTABILITY (CONT.)

Design Checklist	Action
Data Model	• Incoming data is temporarily stored in server cache before written to database • It is possible to re-create a system condition leading to a fault using data abstractions from cache
Mapping Among Architectural Elements	• Call module from within the client application. This shall trigger the specific component • Send request to server from component. This shall trigger the network protocol negotiator component • If scheduling and load balancing are configured correctly, the exchange process on the servers shall be mapped to the correct processor • If concurrency is configured correctly, then multiple threads within the processor will be utilized to handle multiple requests simultaneously • If database is configured correctly, decrypted data shall be written correctly from cache

TESTABILITY (CONT.)

Design Checklist	Action
Resource Management	• Tests shall be executed identically representing the environment in which system will run • Virtualization may be utilized to simulate multiple client account creation requests • Log files are utilized to capture test results for further analysis
Binding Time	• Late binding components need to be tested: - Encryption and decryption functions - Authentication functions - Network communication functions - Database read / write functions • Log files are used to determine which components are causing a fault so that system conditions leading to a fault can be re-created

TESTABILITY (CONT.)

Design Checklist	Action
Choice of Technology	• Fault-injection is utilized to simulate system behavior under stressed conditions • Regression testing is utilized to simulate a change in source code • Specialized interfaces, such as set, get, report, and reset methods to control variable values

USABILITY

Design Checklist	Action
Allocation of Responsibilities	• Simple instructions in interface module • Straight-forward and interactive buttons • Warning messages when human error is detected • Error messages when server is unable to process request
Coordination Model	• Client application shall be intuitive, responding to user interactions in real-time • Have an option to cancel an ongoing process altogether • Display the interface correctly, adapting to various screen sizes and resolutions

USABILITY (CONT.)

Design Checklist	Action
Data Model	• Support undo, cancel, copy, paste, and autofill operations • When a user performs such operations, it shall be done within 0,1 ms • User interface shall support the ability to zoom in on the screen
Mapping Among Architectural Elements	• User shall be made to understand that this is a client-server architecture application • Data input within the client application will be sent to server for processing and storing • Client application does not store any data apart from cache and cookies • Client application is required to have an active internet connection in order to communicate to server

USABILITY (CONT.)

Design Checklist	Action
Resource Management	• System shall have a pre-defined user requirements • Devices that does not meet these requirements will not be able to run this application • Prevent connection-critical module from starting if client application does not have internet connection • Information exchange shall be kept small and concise, to support client applications with slow and unreliable internet connections
Binding Time	• User can choose the following configurations: - Use autofill when providing various types of information within the form - Choose how to receive one-time password i.e. via text messages, voice call, email, etc. - Choose stream quality - Choose user interface theme

USABILITY (CONT.)

Design Checklist	Action
Choice of Technology	• Online customer support is available through a specific module • User feedback is managed through a component within the user interface module

OTHER QUALITY ATTRIBUTES: PORTABILITY

Design Checklist	Action
Allocation of Responsibilities	• Application shall be designed to run on Android and iOS operating systems
Coordination Model	• Application source code is coded in reference to J2EE client-server architectures • Source code elements shall reference Cordova.js and/or Flutter frameworks • Cordova and/or Flutter will wrap the codebase depending on the operating system the application will be installed
Data Model	• A single codebase is utilized and must support high portability
Mapping Among Architectural Elements	• Native application source code is wrapped to run on multiple platforms

OTHER QUALITY ATTRIBUTE: PORTABILITY (CONT.)

Design Checklist	Action
Resource Management	• Source code is written with respect to the Flutter widget-oriented framework • Source codes must reference cordova.js to provide API bindings • Configure Cordova through a plugin interface for native application components to communicate with each other
Binding Time	• Code framework is bind at compile time • Code plugins, application parameters, and API bindings are done at runtime
Choice of Technology	• To create applications that can run on multiple platforms from a single codebase, the following technologies are considered: - Flutter - Cordova

OTHER QUALITY ATTRIBUTES: MOBILITY

Design Checklist	Action
Allocation of Responsibilities	• User interface module shall be able to adapt to various screen size, type of display, type of input devices, etc. • Media streaming module shall be able to adapt to various bandwidth availability and network speed • Audio signal processing module shall be able to adapt to various types of output device
Coordination Model	• User interface module shall coordinate with host system to get host screen and input parameters • Media streaming module shall do constant network tests to check available bandwidth • Audio signal processing module shall recognize defer binding parameters, process audio signal with user-chosen settings

OTHER QUALITY ATTRIBUTES: MOBILITY

Design Checklist	Action
Data Model	• User interface module shall manipulate screen parameters based on interface parameters from client system • Audio processing module shall initialize audio quality parameters based on user selection
Mapping Among Architectural Elements	• User interface runtime elements are mapped based on set parameters • Map graphical elements to GPU processing • Other mobile runtime elements are mapped to CPU processing

OTHER QUALITY ATTRIBUTES: MOBILITY

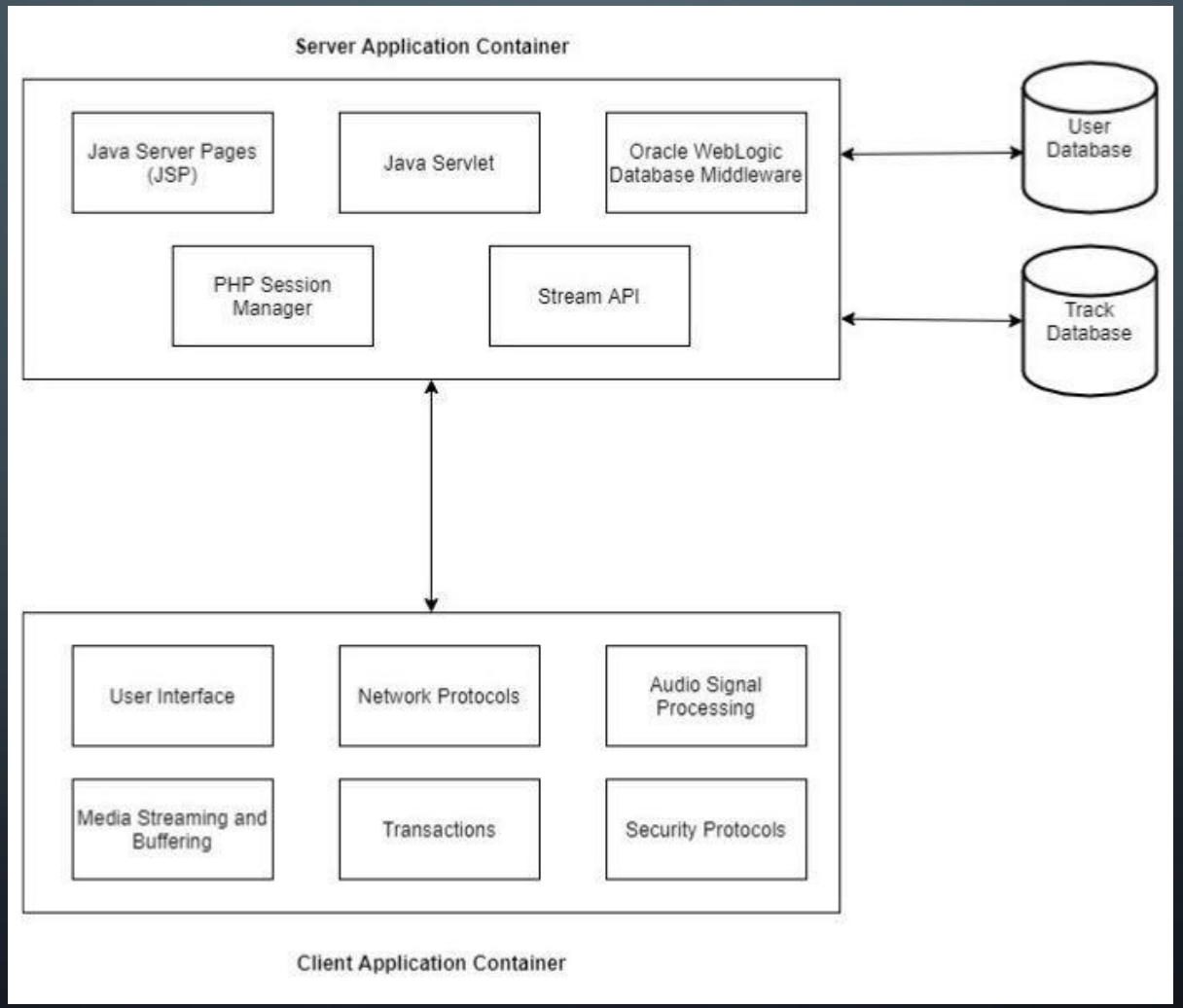
Design Checklist	Action
Resource Management	• Critical components that affect mobility: - User interface module - Media streaming module - Audio signal processing module • Those critical components adapt to parameters set by client environment
Binding Time	• Critical components that affect mobility are bind at runtime • Utilize defer binding tactic, let end user choose system configuration at startup
Choice of Technology	• Firebase Test Lab, to test if mobility is configured correctly


APPLIED PATTERN SOLUTION

CLIENT-SERVER PATTERN

Attribute	Analysis
Context	• Music tracks are shared resources that large numbers of distributed streamers wish to access • We wish to control access to the shared resources and quality of application services • Client application can access shared resources, but only server application can modify them
Problem	• By managing shared resources and services on a single location, we can promote usability, mobility, portability, and interoperability because of centralized control of those shared resources • Maintaining availability, security, and performance is a challenge while distributing the resources across multiple clients
Solution	• Clients interact by requesting access to resources within servers, which will grant access and provide required set or services. There will be one central server environment distributed towards multiple machines to improve performance and ease workload

CLIENT-SERVER ARCHITECTURE



CLIENT-SERVER SOLUTION

Attribute	Solution
Overview	• Clients initiate interactions with servers, requesting access to database resources. System invokes middleware services as needed and sends requested resources back to the client
Elements	• Client: a component that requests server resources and invokes services of a server component (e.g. Streamer) • Server: a component that provides resources and services to clients (e.g. Stream Music Tracks) • Request/reply connector: a data connector used by a client to invoke services on a server. Calls are remote and data exchange shall be encrypted (e.g. Network Communication Channels)
Relations	• The attachment relation associates clients with servers

CLIENT-SERVER SOLUTION

Attribute	Solution
Constraints	• Clients are connected to servers through request/reply connectors • Components may be arranged in tiers • Client and servers may not be modified independently
Weaknesses	• Server can be a single point of failure • Server can be a performance bottleneck • If server fails for any reason, then none of client requests can be fulfilled • Complex client-server systems may require high maintenance costs

SOFTWARE ARCHITECTURE SUMMARY

ARCHITECTURE OVERVIEW

- This system is designed to allow record publishers to publish records from their signed artists to a public cloud audio library
- The system will be composed of server-side components and client-side components
- The server-side component will manage the database operations and algorithms that produce recommendation results. The client-side components will be graphical interfaces that are integrated into the corresponding system
- The software implements client-server architecture and provides a system to legally share, acquire, and download audio tracks

ARCHITECTURE OVERVIEW (CONT.)

- The client-side shall stream material from the music streaming service on the server-side through an active internet and play it through the built-in audio player
- System shall have a central repository on the server-side for storing tracks provided by record publishers
- System shall have database which consists of track data and user data, well-maintained by approved database management tools

ARCHITECTURE SIGNIFICANT REQUIREMENTS

Quality Attribute	Attribute Refinement	ASR
Security	• Confidentiality • Integrity • Availability	• User data is protected against unauthorized access with 128-bit AES Encryption • System resources are protected against unauthorized modifications. Maintain audit trail • Servers are protected against illegitimate access and usage with unique client session ID and CAPTCHA tests
Availability	• Downtime • Failures • Capacity	• Servers shall have an availability rate of no less than 99,999% • A single component failure shall not result in service termination. It shall be resolved in no more than 60 seconds. • Servers shall be designed to operate with no less than 90% capacity at all times

ARCHITECTURE SIGNIFICANT REQUIREMENTS (CONT.)

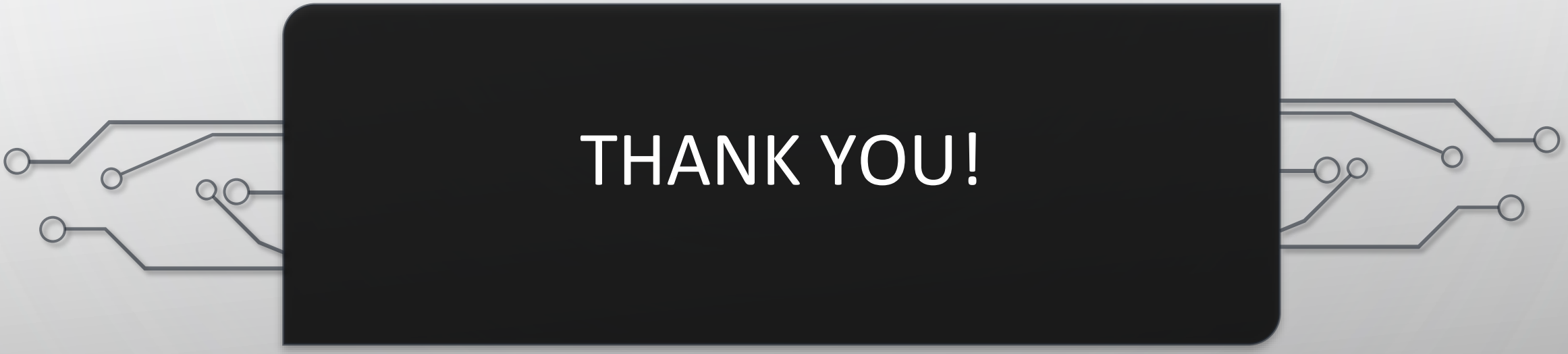
Quality Attribute	Attribute Refinement	ASR
Usability	• Easy-to-learn user interface • Effective and efficient usage • Minimize impact of errors • Adapt and feel comfortable with system	• Create easy-to-use interactive interface. Include comprehensive FAQ repository. • Reach usage goals effectively and efficiently. Allow multiple simultaneous activities, reuse already entered data • Recognize user errors. Utilize undo or cancel actions. • Let users feel comfortable with system environment. Provide customizability to the user interface
Performance	• Latency • Loss rate • Transfer rate • Response time	• Client-server connection latency shall not be greater than 300 milliseconds. Refresh connections if high latency persists. • Data loss in one transfer shall not exceed 10%. Attempt to resend lost data at all times. • Client-server data transfer rate shall not be less than 128 kbps at all times. Refresh connections if low transfer rate persists. • Client application shall have a response time not greater than 300 milliseconds at all times whenever user input is detected.

ARCHITECTURE SIGNIFICANT REQUIREMENTS (CONT.)

Quality Attribute	Attribute Refinement	ASR
Mobility	• User interface • Media streaming • Audio signal processing	• User interface shall be able to adapt to various screen size, types of display, and types of input devices. • Media streaming module shall be able to adapt to various network bandwidth availability and transfer rate. Utilize buffer cache at all times. • Audio signal processing module shall be able to adapt to user-chosen audio quality and various types of output device.
Portability	• Operating environment	• System shall be designed to operate in both Android and iOS. Utilize hybrid development frameworks.
Interoperability	• Interoperating systems • Interoperating transactions • Interoperating mechanisms	• Server application shall interoperate internally with client application and externally with record publisher system • Internal systems shall be able to handle 10000 simultaneous interoperating requests with a response time of no more than 500 ms and information exchange time of no more than 5 seconds. • Interoperating requests shall be completed through Java SQL module API. Identify interoperating data through unique digital ID

ARCHITECTURE SIGNIFICANT REQUIREMENTS (CONT.)

Quality Attribute	Attribute Refinement	ASR
Modifiability	• Affected artifacts • Mechanisms • Dependencies	• Modification on one artifact shall not affect other system functionalities. No side effects guaranteed • End user and database manager are able to modify system at runtime. Ensure actors can only modify the system within their respective scopes. • Modification in client application shall have dependencies with server application, and vice versa. System shall monitor its health after each modification is applied.
Testability	• Late binding • Environments	• Critical late bound components must be thoroughly tested. Create log files to record test outputs. • Tests shall not be conducted on live system environment. Utilize virtualization.


THANK YOU!