

Music Use for People with Dementia : System Analysis and Design Report

KS1908

Alexander MacKay
Caitlin Colin-Thome
Jordan Su



Executive Summary

This document's main objective is to provide an in-depth analysis of the current development of the mobile application, including its requirements, risks, constraints and design of both logic and aesthetics.

The document begins with presenting information regarding our client, Christopher Millington, and his vision for the mobile application to achieve for the MARCS Institute. This is followed by the problem statement, which goes more in-depth with the objectives themselves and what potential obstacles that need to be addressed.

Analysing the problem has led to the identification of the functional and non-functional requirements needed in order for the mobile application to fulfil the objectives dealt in the previous section. Such requirements range from scalability, security to account management and database systems, and are important in not only the maintenance of the mobile application, but the potential growth that will occur in the future . It also identifies any risks and considerations that can affect the development and use of the proposed system, including the design of the administrative site, the application's clarity in its aesthetic and the system's software architecture.

By taking these results into consideration, our team was able to design both the aesthetics and the logical system structure of the mobile application as well as the administration interface site; such design choices have been found to best fulfil the objectives whilst counteracting the risks and constraints. These designs are explained extensively in the Detailed System Design section, along with contingency courses in case the main courses of action are unavailable. Test Plans have also been included in this report, outlining features that will be in the mobile application and the admin site, and the pass-fail criteria that they will be critiqued against.

In this report, our team has conducted a comprehensive investigation that provides insight on the logical design of the system and the features that will be implemented to achieve the objectives of MARC Institute.

Table of Contents

Executive Summary	2
Introduction	4
1 - Client Statement	5
2 - Problem Statement	5
3 - System Requirements	6
3.1 - Functional Requirements	6
3.1.1 - FR01 Account Management	6
3.1.2 - FR02 View Information/Guides on Dementia	6
3.1.3 - FR03 User Analysis	6
3.1.4 - FR04 Song Analysis	6
3.1.5 - FR05 Database System	6
3.1.6 - FR06 Playlist System	7
3.1.7 - FR07 Questionnaire Management	7
3.1.8 - FR08 Song Management	7
3.2 - Non-Functional Requirements	8
3.2.1 - NFR01 Performance	8
3.2.2 - NFR02 Response Time	8
3.2.3 - NFR03 Usability	8
3.2.4 - NFR04 Security	8
3.2.5 - NFR05 Platform Compatibility	8
3.2.6 - NFR06 Privacy	9
3.2.7 - NFR07 Scalability	9
4 - Risks and Constraints	10
4.1 Design Considerations	11
4.1.1 App Design	11
4.1.2 Server / Database Design	11
4.2 System Software Architecture	12
5 - Detailed System Design	13
5.1 Use Case Diagram	13
5.2 Expanded Use Cases	15
5.3 Entity Relationship Diagram (ERD)	28
5.4 Database Schema	29
5.5 Screen Designs	30
6 - Test Plan	36
6.1 Features/Use Cases	36
6.2 Candidate Test Cases	37
Conclusion	43

Introduction

This document will take the contents of the project plan and project proposal into consideration in order to provide detailed plan on testing of the project and display tables and diagrams of the systems structure. The following pages provide a detailed analysis of all functional and nonfunctional requirements based on the high level business functions in the proposal. Any identified risk and constraint, both internally and externally have also been listed and resolutions identified. Our design consideration before initiating the project are provided which provide insight into the structural diagram of the application and website functions and how they interact with the different users, the server and the database. Use case diagrams further describe the actions and permissions available to the different user types and the functions of the server application.

A descriptive Entity Relationship Diagram is provided to demonstrate the relationships inside our database and a Database Schema provides more detail about the database tables and fields. Screenshots of all the applications current pages have been provided along with a table detailing the functionality of each page. Finally, our future testing plans for the application and website functionality have been recorded, all situations and actions having test scenarios which will be carried out and the results recorded in the handover report. This document will provide an in depth insight into the structure of the systems and applications being developed and show our plans for ensuring the usability and quality of the product through testing.

1 - Client Statement

Our client Christopher Millington is a Research Officer at The MARCS Institute for Brain, Behaviour and Development. The MARCS Institute for Brain, Behaviour and Development is an interdisciplinary research institute of Western Sydney University who study the scientific bases of human communication. Currently the brain science department are working on multisensory communication research to investigate how information from our different senses contributes to communication, speech and language abilities and how they underpin much of human communication and also using behavioural and neurophysiological approaches to investigate the temporal dynamics in individual and group performance, perception, and creation of music and dance. Christopher has contributed to Inter-University Neuroscience and mental Health projects and has been working with a colleague to provide us the insight we need into dementia and how music therapy assists with its symptoms. He has guided our team with his personal experience and knowledge, assisting in crucial development decisions to best benefit the patient users of the application and the overall experience for all.

2 - Problem Statement

The problem our client is trying to solve with the development of this application is a lack of technology to develop personalised music therapy for users. Currently there isn't a service that takes patients or their carers inputs through questionnaires, analyses their responses and develops a personalised playlist of songs for patients with dementia to listen to. The benefits of this technology will not only make the automation and tailored creation of a playlist quicker and easier for carers but dementia patients will more easily be able to access and benefit from music therapy. Another problem that existed was the fact that other technologies that could meet some of the clients needs, other playlist creations apps such as spotify, aren't widely available, free and customisable. The application and administration website we are creating will all for all content (songs) to be tailored, playlists for patients can be edited and customisation in general will allow the client full control.

3 - System Requirements

3.1 - Functional Requirements

3.1.1 - FR01 Account Management

The user should be able to sign up and sign in to their account and view their account details such as name and institution. A carer user type should be able to add a patient user type under their account. An admin should also be able to view and edit all users.

3.1.2 - FR02 View Information/Guides on Dementia

The user should be able to view information about dementia and guides about how to treat it. This information will be presented in a usable and clean way so that patients and carers can navigate the information easily.

3.1.3 - FR03 User Analysis

Upon signing up, the user will have to complete a series of questionnaires and surveys. These can include questions about the user's mental health history, songs they enjoy or any instruments they play. The results will be used to identify a list of songs that may help the user when they're distressed or anxious.

3.1.4 - FR04 Song Analysis

The tempo and mode of a song needs to be analysed to decide whether or not it will be suitable for treating the user. Typically, songs with a low tempo (80BPM - 120BPM) and major mode are preferred. The genre of the song is also analysed to determine whether the song will be added to the patient playlist.

3.1.5 - FR05 Database System

The database system is used to store user details, the results and details of the questionnaires and the song details. This will allow our client to analyse the data and identify trends.

3.1.6 - FR06 Playlist System

The server should automatically create a playlist depending on the survey answers the user supplied. The playlist should also take into account the time of day and how familiar the user is with the songs. If the user is typically sleepy in the mornings, a playlist should be suggested so that it helps energise them. More familiar songs should be played less frequently to ensure the user doesn't get bored of it.

3.1.7 - FR07 Questionnaire Management

An admin should be able to easily add, edit and delete questionnaires. This can include changing the choices a question has, changing what song features (tempo, mode, instrumental/vocal, etc.) a choice maps to and changing the choice type (radio buttons, text, rating, etc.). This will allow the client to fix any typos in the questions and add more questionnaires in the future without needing KS1908 to recompile the Android app.

3.1.8 - FR08 Song Management

An admin should be able to add, edit and delete songs to be used for treatment. This will allow the client to slowly build up treatment songs and have a wider variety of playlists so that patients don't get the same playlists.

3.2 - Non-Functional Requirements

3.2.1 - NFR01 Performance

The application will be performance tested on multiple different devices, both through the android emulator and on a physical phone. The performance of all application actions that will be used by the users will be tested to ensure peak responsiveness, useability and accuracy of our systems. Any flaws exposed by these performance tests will be amended. The admin website will also be tested to ensure it can perform all capabilities correctly. This requirement will be met if the app and website perform all required actions correctly and timely.

3.2.2 - NFR02 Response Time

This application will require internet access for immediate updating and generation of playlist as it will need to communicate with the database. We are also looking into offline options of storing data/changes locally and making these changes instantly once the device connects to the internet. All in-application actions will be instantaneous when connected to the internet, with the exception of the generation of playlist, this may take a few seconds to ruin the scripts to gather the songs, so the user will be redirected while they wait.

3.2.3 - NFR03 Usability

Simple Interface - allows users to utilise the features provided

Guides - integrated to help users (patients and carers) to use the app and its features

Error-handling - helps with identifying any issues that the app may have throughout its use, and thus allow us to develop any updates to combat these vulnerabilities

The Admin site - easy to navigate and all management options successfully update the database

3.2.4 - NFR04 Security

Dealing with the sensitive data, information must be encrypted and access to server must be restricted to authorized personnel only. The administration website will be secured with login credentials as will the account of any user of the app. The server itself and the database will be secured by the university security already existing.

3.2.5 - NFR05 Platform Compatibility

This application was requested as an android app and will be compatible across all android devices. The administration website to manage users and questionnaires will be available on any web browser making it widely accessible for any administrative users.

3.2.6 - NFR06 Privacy

Carer and patient information (Name, symptoms, patients, etc.) will be collected through the application and stored in the database. Results from in-app surveys will also collect personal information and be stored in the database. It is important that this information is kept private and secure. Patients will have to be informed that their carer or a higher body will be able to view the results of surveys taken, but only the necessary people with the correct privileges will be able to view private details and personal information.

3.2.7 - NFR07 Scalability

Scalability refers to the adequate handling of change in workload. A scalable application will be able to cater for a significant increase or decrease in the amount of users accessing the app. The database will need to be able to store the credentials of new users, as well as their responses to the appendix questions and playlist information. The database will also need to have removal/deletion functionality to remove any data no longer needing to be stored (carers that leave their job, patients that no longer use the application, ect.). Scalability will be measured by the maximum number of users able to access the application at once, as well as the maximum number of queries, updates or creations the database can handle.

4 - Risks and Constraints

Risk	Resolution	Type of Risk
Team members are out of practice on mobile application programming	Revise previous mobile applications unit content. Research mobile applications programming to refresh.	Internal
Caitlin and Alexander are not familiar with Github	Jordan can teach them how to use Github. Video on VUWS.	Internal
Client responsiveness is too slow	If responses are too delayed discuss with supervisor	External
A project scope increase beyond possible completion (Scope creep)	Understand our capabilities and be prepared to say no to additional tasks	External
Team disputes and conflicts	Attempt to find a compromise, contact supervisor individually or as a team for assistance	Internal
Development halts due to lack of understanding of how to implement an aspect of a system	Contact supervisor who will provide relevant resources, address the issue at the next meeting	Internal
Lack of accountability / No one responsible for certain project aspects	Assign main responsibility for project aspects, refer to that member for progress updates and if any issues occur (see Group Milestone Plan)	Internal
Lack of relevant resources (Server, database, ect.)	Contact supervisor for guidance, these resources can be obtained through provided contacts	Internal
Project stops/falls behind due to unforeseen circumstances (software being used has downtime for maintenance, client is unresponsive about critical inquiry, etc.)	Ensure a log is kept to detail all records of project progression and any temporary pauses/stops and the cause. This may need to be reviewed is project completion is affected.	Internal
Hardware changes at university preventing the project from being completed properly	Work with university staff to help diagnose issues and resolve to a solution	External
Server temporarily shuts down, preventing the team from working on the project	Work with the technical support staff to resolve the issue, Keep in contact for future issues, progress on other project aspects	External

4.1 Design Considerations

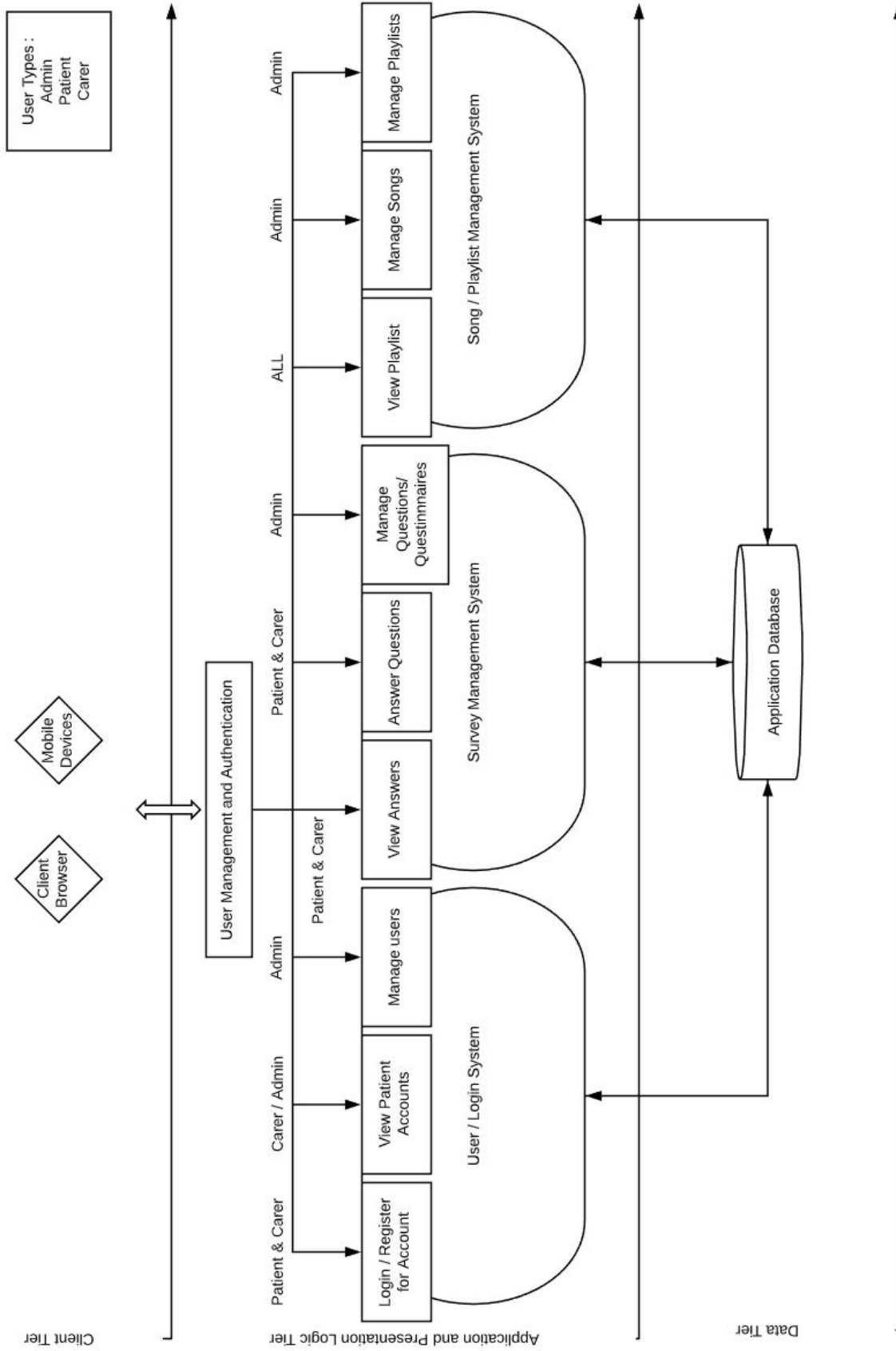
4.1.1 App Design

For our Android app, we considered developing a mobile web application so that it can be cross platform, but we had no experience with the tools to do that. As we have experience with Android Studio, we decided to develop a native Android app instead so that we spend more time developing the app rather than wasting time learning how to use the tools to develop a mobile web application. This also means we will be familiar with the ecosystem and any quirks that may appear so that we're able to fix any issues quickly.

4.1.2 Server / Database Design

For our server, we decided to use NodeJS as we did have experience with JavaScript so learning Node wouldn't be too difficult. There are also many libraries and frameworks to use in Node which make it easy to start developing our server. We chose to use the Express framework to handle the HTTP requests and responses as it's popular so it should be reliable and mostly bug-free. Our API only consists of GET and POST requests for retrieving information and adding/updating information respectively. As for the response, the server sends back a JSON response as many languages support parsing JSON. This makes our API easy to use and easy to develop an application for. The user's passwords are hashed with bcrypt as it is arguably the most resistant to brute force attacks due to how long it takes to compute the hash. It is also easily scalable so that as computers get more powerful, bcrypt's strength can also increase. Our database uses MySQL as that is compatible with the university's server.

4.2 System Software Architecture



5 - Detailed System Design

5.1 Use Case Diagram

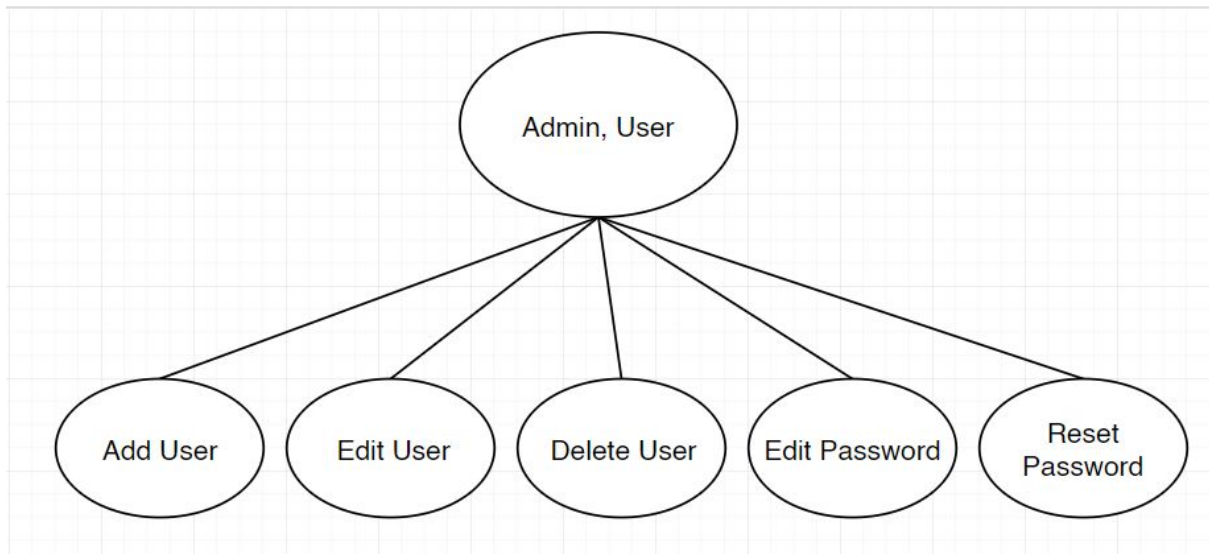


Figure 1 - Manage Users Use Case

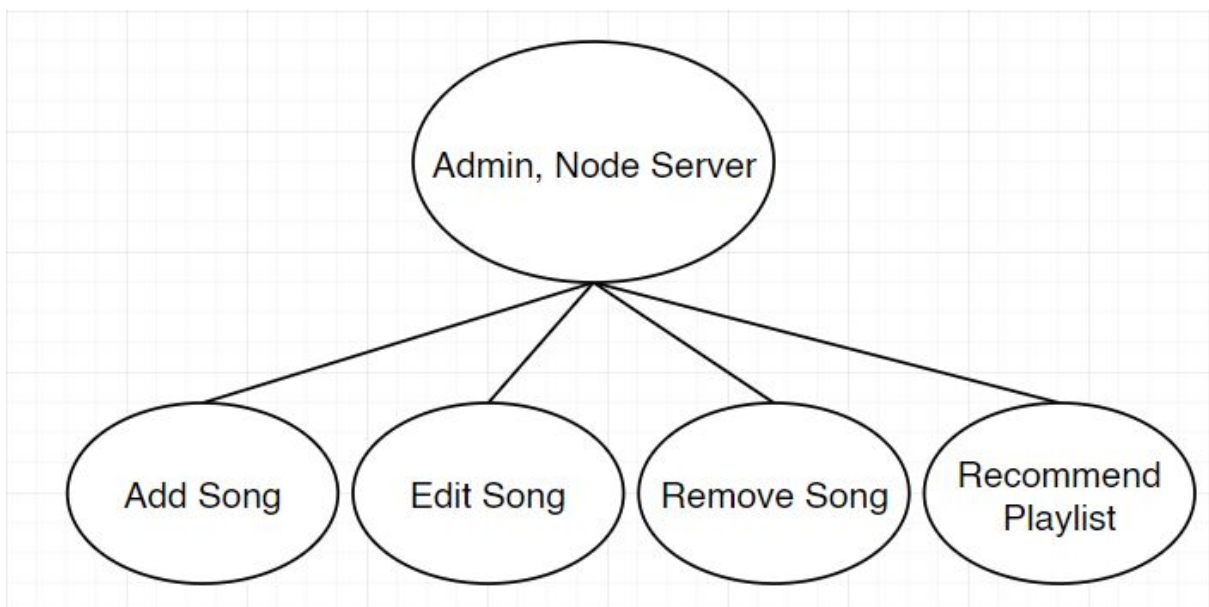


Figure 2 - Manage Songs Use Case

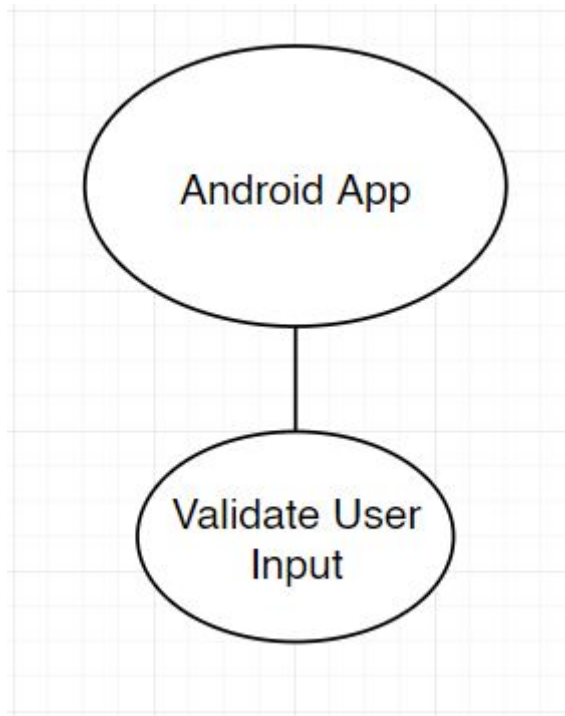


Figure 3 - Collect Answers to Questionnaires/Surveys Use Case

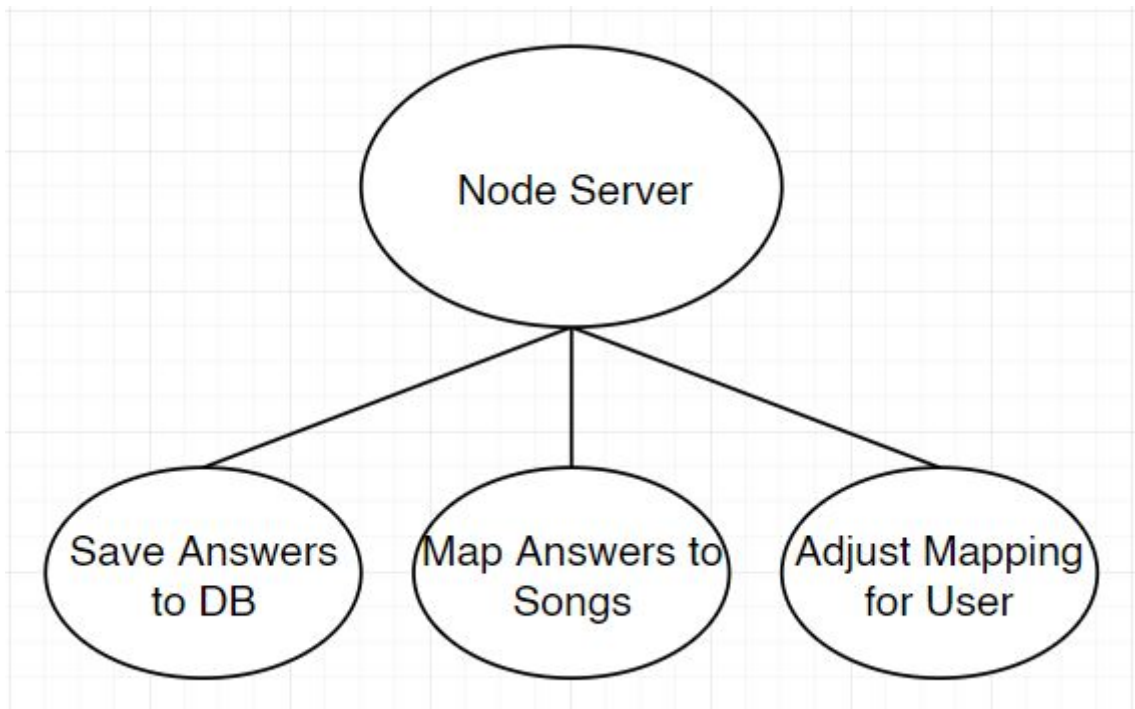


Figure 4 - Process Answers to Questionnaires/Surveys Use Case

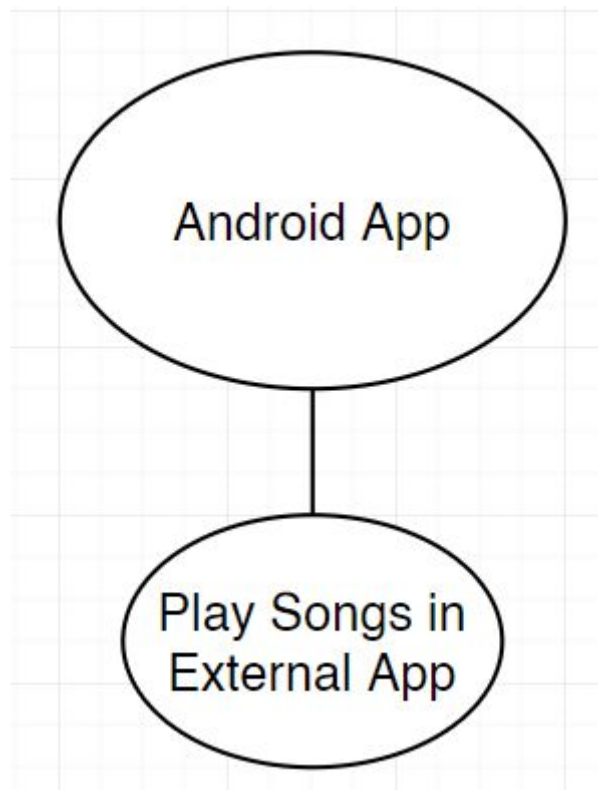


Figure 5 - Play Songs Use Case

5.2 Expanded Use Cases

Use Case:	Add User / Sign Up (UC01)
Category:	Core
Actor(s):	Admin, User
Trace:	BF01
Description:	Sends email, password, type, institution, first name and last name to UC02. If successfully validated, the user will be created. Otherwise, an error message will be returned.
Purpose:	Allows an admin/user to create an account to answer questionnaires/surveys and listen to songs.
Notes:	
Pre-condition(s):	User information is valid.
Post-condition(s):	User will be created and added to the database.

Typical Course of Events

#	Actor Action	System Event	#	System Response	Trace
1	Enter email, password, type, institution, first name and last name	Successfully validates input	2	Output "Signed up" message	

Alternate Course of Events A

#	Actor Action	System Event	#	System Response	Trace
1	Enter email, password, type, institution, first name and last name	Validation finds an account with the email	2	Output "Email in use" message	

Alternate Course of Events B

#	Actor Action	System Event	#	System Response	Trace
1	Enter email, password, type, institution, first name and last name	Fails to validate input or server error	2	Output "Sign up failed" message	

Use Case:	Edit User (UC02)
Category:	Core
Actor(s):	Admin, User
Trace:	BF01
Description:	Sends email, password, type, institution, first name and last name to UC02. If successfully validated, the user information will be updated. Otherwise, an error message will be returned.
Purpose:	Allows an admin/user to update user information.
Notes:	
Pre-condition(s):	User must be signed in to update their information.
Post-condition(s):	User's information will be updated

Typical Course of Events

#	Actor Action	System Event	#	System Response	Trace
1	Enter email, password, type, institution, first name and last name	Successfully validates input	2		
3		Email, hash of password, type, institution, first name and last name is stored in the database	4	Output "Successfully updated account info" message	

Alternate Course of Events A

#	Actor Action	System Event	#	System Response	Trace
1	Enter email, password, type, institution, first name and last name	Server error	2	Output "Failed to update account info" message	

Use Case:	Delete User (UC03)
Category:	Core
Actor(s):	Admin
Trace:	BF01
Description:	Sends the user ID to be deleted and if found, user will be marked as deleted.
Purpose:	Allows an admin/user to delete a user.
Notes:	
Pre-condition(s):	User ID should point to an existing user.
Post-condition(s):	User will be marked as deleted.

Typical Course of Events

#	Actor Action	System Event	#	System Response	Trace
1	Send user ID	Checks for a user with the specified ID and finds a user	2		
3		Marks the specified user as deleted in the database	4	Output "Successfully deleted user" message	

Alternate Course of Events A

#	Actor Action	System Event	#	System Response	Trace
1	Send user ID	No user found with the specified ID	2	Output "No user found" message	

Alternate Course of Events B

#	Actor Action	System Event	#	System Response	Trace
1	Send user ID	Server error	2	Output "Failed to delete user" message	

Use Case:	Edit Password (UC04)
Category:	Core
Actor(s):	Admin, User
Trace:	BF01
Description:	Sends the current user password and the new password. If the current password matches, the user's password will be updated to the new password. Otherwise, an error message will be returned.
Purpose:	Allows an admin to change a user's password if the user forgot it. Allows a user to change their password.
Notes:	
Pre-condition(s):	User should be signed in to change their password.
Post-condition(s):	User's password will be changed.

Typical Course of Events

#	Actor Action	System Event	#	System Response	Trace
1	Send current password and new password	Checks that the current password matches the password stored in the database for the specified user	2		
3		User's password will be changed	4	Output "Password updated" message	

Alternate Course of Events A

#	Actor Action	System Event	#	System Response	Trace
1	Send current password and new password	Current password doesn't match the password stored in the database	2	Output "Current password mismatch" message	

Alternate Course of Events B

#	Actor Action	System Event	#	System Response	Trace
1	Send current password and new password	Server error	2	Output "Failed to update password" message	

Use Case:	Reset Password (UC05)
Category:	Core
Actor(s):	Admin, User
Trace:	BF01
Description:	A temporary password will be sent to the specified email which should allow the user to sign in and change their password.
Purpose:	Allows an admin to reset a user's password if they forgot their email. Allows a user to reset their password if they've forgotten their password.
Notes:	
Pre-condition(s):	User should be signed in to change their password.
Post-condition(s):	User's will have a temporary password.

Typical Course of Events

#	Actor Action	System Event	#	System Response	Trace
1	Send email address	Checks that the email address matches belongs to a user	2		
3		Temporary password will be sent to the email address	4	Output "Email sent" message	

Alternate Course of Events A

#	Actor Action	System Event	#	System Response	Trace
1	Send email address	Email address doesn't belong to a user	2	Output "No account was found with the specified email" message	

Alternate Course of Events B

#	Actor Action	System Event	#	System Response	Trace
1	Send email address	Server error	2	Output "Failed to reset password" message	

Use Case:	Add Song (UC06)
Category:	Core
Actor(s):	Admin, Node Server
Trace:	BF02
Description:	Sends song information (name, artist, link, tempo, mode, genre, length, year and lyric type) to the server. If successfully validated, the song will be added to the database. Otherwise, an error message will be returned.
Purpose:	Allows an admin to add a song to be used for playlists.
Notes:	
Pre-condition(s):	Song information should be valid.
Post-condition(s):	Song will be added to the database.

Typical Course of Events

#	Actor Action	System Event	#	System Response	Trace
1	Send song information	Successfully validates song information	2	Output "Song added" message	

Alternate Course of Events A

#	Actor Action	System Event	#	System Response	Trace
1	Send song information	Fail to validate song information	2	Output "Invalid song information" message	

Alternate Course of Events B

#	Actor Action	System Event	#	System Response	Trace
1	Send song information	Server error	2	Output "Failed to add song" message	

Use Case:	Edit Song (UC07)
Category:	Core
Actor(s):	Admin, Node Server
Trace:	BF02
Description:	Sends song information (name, artist, link, tempo, mode, genre, length, year and lyric type) to the server. If successfully validated, the song information will be updated. Otherwise, an error message will be returned.
Purpose:	Allows an admin to edit a song.
Notes:	
Pre-condition(s):	Song should exist in the database and be valid.
Post-condition(s):	Song information will be updated.

Typical Course of Events

#	Actor Action	System Event	#	System Response	Trace
1	Send song information	Successfully validates song information	2	Output "Song updated" message	

Alternate Course of Events A

#	Actor Action	System Event	#	System Response	Trace
1	Send song information	Fail to validate song information	2	Output "Invalid song information" message	

Alternate Course of Events B

#	Actor Action	System Event	#	System Response	Trace
1	Send song information	Server error	2	Output "Failed to update song" message	

Use Case:	Remove Song (UC08)
Category:	Core
Actor(s):	Admin, Node Server
Trace:	BF02
Description:	Send song ID to the server. If the song ID matches an existing song, the song will be marked as removed.
Purpose:	Allows an admin to remove a song.
Notes:	
Pre-condition(s):	Song should exist in the database.
Post-condition(s):	Song will be marked as deleted.

Typical Course of Events

#	Actor Action	System Event	#	System Response	Trace
1	Send song	Successfully validates song information	2	Output "Song removed" message	

Alternate Course of Events A

#	Actor Action	System Event	#	System Response	Trace
1	Send song ID	Song ID doesn't match any of the existing songs in the database.	2	Output "No song found" message	

Alternate Course of Events B

#	Actor Action	System Event	#	System Response	Trace
1	Send song ID	Server error	2	Output "Failed to remove song" message	

Use Case:	Recommend Playlist (UC09)
Category:	Core
Actor(s):	Node Server
Trace:	BF02, BF04
Description:	After completing all the required questionnaires, a playlist will be sent back to user.
Purpose:	Playlists can help the user.
Notes:	
Pre-condition(s):	User should have completed all the required questionnaires.
Post-condition(s):	Playlist will be sent to the user.

Typical Course of Events

#	Actor Action	System Event	#	System Response	Trace
1	Request playlist.	Checks that all the required questionnaires are completed.	2	Return a playlist that may help the user.	

Alternate Course of Events A

#	Actor Action	System Event	#	System Response	Trace
1	Request playlist.	Not all of the required questionnaires are completed.	2	Output "Some of the required questionnaires haven't been completed yet" message.	

Alternate Course of Events B

#	Actor Action	System Event	#	System Response	Trace
1	Request playlist	Server error	2	Output "Failed to request playlist" message	

Use Case:	Validate User Input (UC10)
Category:	Core
Actor(s):	Android App
Trace:	BF03
Description:	After completing a questionnaire, the input should be validated before sending it to the server.
Purpose:	To ensure all the necessary fields are completed and valid.
Notes:	
Pre-condition(s):	User should be signed in to access the questionnaires.
Post-condition(s):	Questionnaire will be ready to be sent to the server.

Typical Course of Events

#	Actor Action	System Event	#	System Response	Trace
1	Answer a question	Check that input is valid.	2	Continue to next question.	

Alternate Course of Events A

#	Actor Action	System Event	#	System Response	Trace
1	Answer a question	Input is invalid.	2	Output message that states why the input is invalid. For example, "Please input an answer"	

Use Case:	Save Answers to Database (UC11)
Category:	Core
Actor(s):	Node Server
Trace:	BF03, BF04
Description:	After completing a questionnaire, the input is validated (UC10). If successfully validated, the answers will be sent to the server which will save them to the database. Otherwise, an error message will be returned.
Purpose:	To store the answers for analysis by the client.
Notes:	
Pre-condition(s):	Answers should be valid.
Post-condition(s):	Answers will be saved in the database.

Typical Course of Events

#	Actor Action	System Event	#	System Response	Trace
1	Answer the last question	Check that input is valid.	2	Send answers to server and output "Successfully saved answers".	UC10

Alternate Course of Events A

#	Actor Action	System Event	#	System Response	Trace
1	Answer the last question	Input is invalid.	2	Output message that states why the input is invalid. For example, "Please input an answer"	UC10

Alternate Course of Events B

#	Actor Action	System Event	#	System Response	Trace
1	Answer the last question	Server error	2	Output "Failed to save answers" message	UC10

Use Case:	Map Answers to Songs (UC12)
Category:	Core
Actor(s):	Node Server
Trace:	BF04
Description:	Depending on the answers to a questionnaire, the recommended playlist will differ.
Purpose:	To build a suitable playlist for users who answered similarly.
Notes:	
Pre-condition(s):	Questionnaire, questions, answers and songs should all exist and be valid.
Post-condition(s):	Answers will be mapped to a song.

Typical Course of Events

#	Actor Action	System Event	#	System Response	Trace
1	Receives answers from user	Save answers to database	2		UC11
3		Go through all the songs in the database and find the ones that satisfy a criteria	4	Send generated playlist to user	

Use Case:	Adjust Mapping for User (UC13)
Category:	Core
Actor(s):	Node Server
Trace:	BF04
Description:	As the user answers questionnaires, the generated playlist will slowly update to suit their answers.
Purpose:	To build a suitable playlist for the user.
Notes:	
Pre-condition(s):	Questionnaire, questions, answers and songs should all exist and be valid.
Post-condition(s):	Answers will be mapped to a song.

Typical Course of Events

#	Actor Action	System Event	#	System Response	Trace
1	Receives answers from user	Save answers to database	2		UC11
3		Update song mapping for user	4	Send user updated playlist	UC12

Use Case:	Play Songs in External App (UC14)
Category:	Core
Actor(s):	Android App
Trace:	BF08
Description:	The user should be able to play a song in an external app like YouTube.
Purpose:	To make it more convenient for the user to play a song.
Notes:	
Pre-condition(s):	User should be signed in and answered all of the required questionnaires to be able to play a suitable song.
Post-condition(s):	Song will be played in external app.

Typical Course of Events

#	Actor Action	System Event	#	System Response	Trace
1	User clicks "Play Song"	Checks if the song has a link	2	Open link in YouTube	

Alternate Course of Events A

#	Actor Action	System Event	#	System Response	Trace
1	User clicks "Play Song"	Song doesn't have a link	2	Output "No link for song" message	

5.3 Entity Relationship Diagram (ERD)

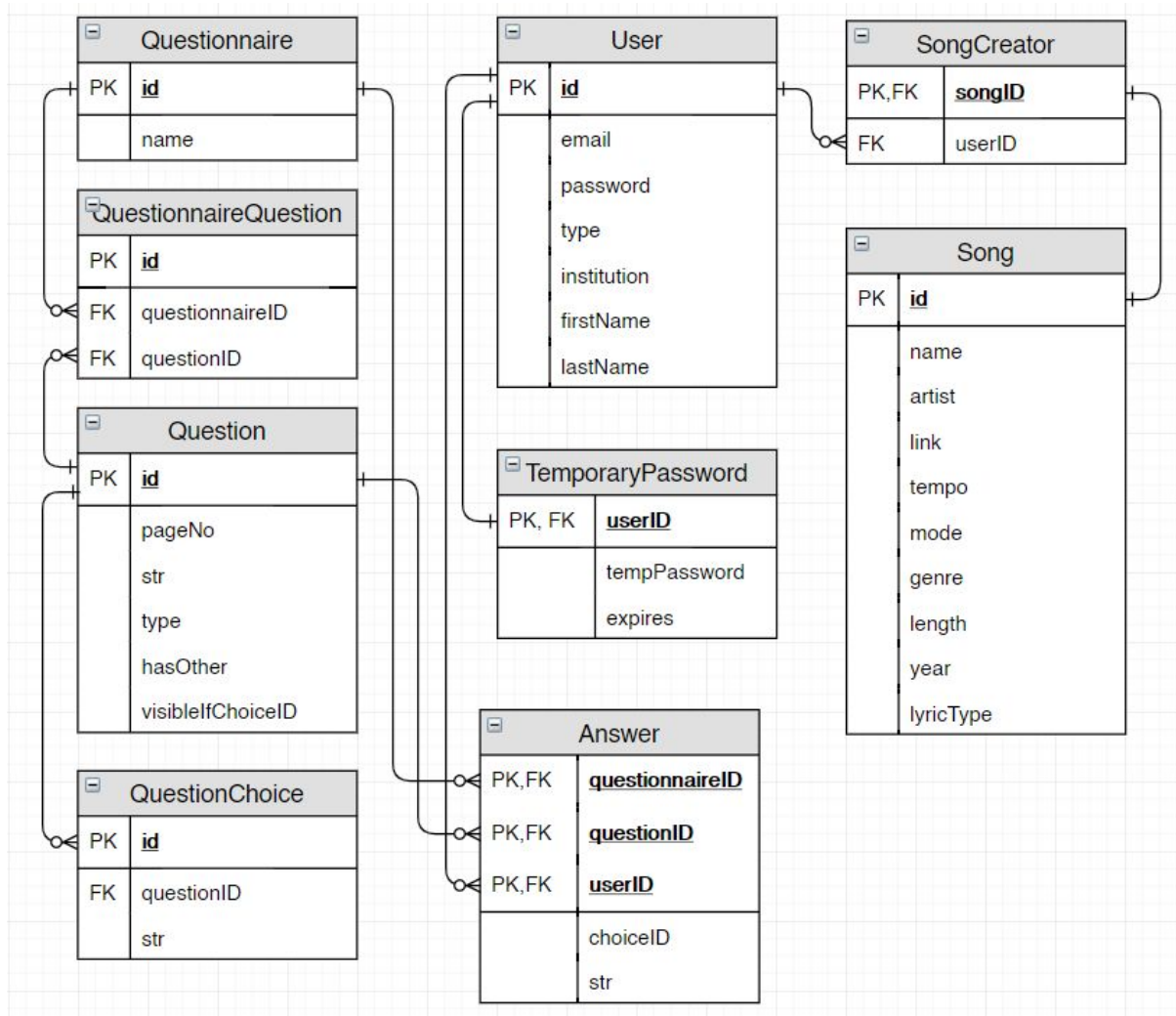


Figure 6 - Entity Relationship Diagram

5.4 Database Schema

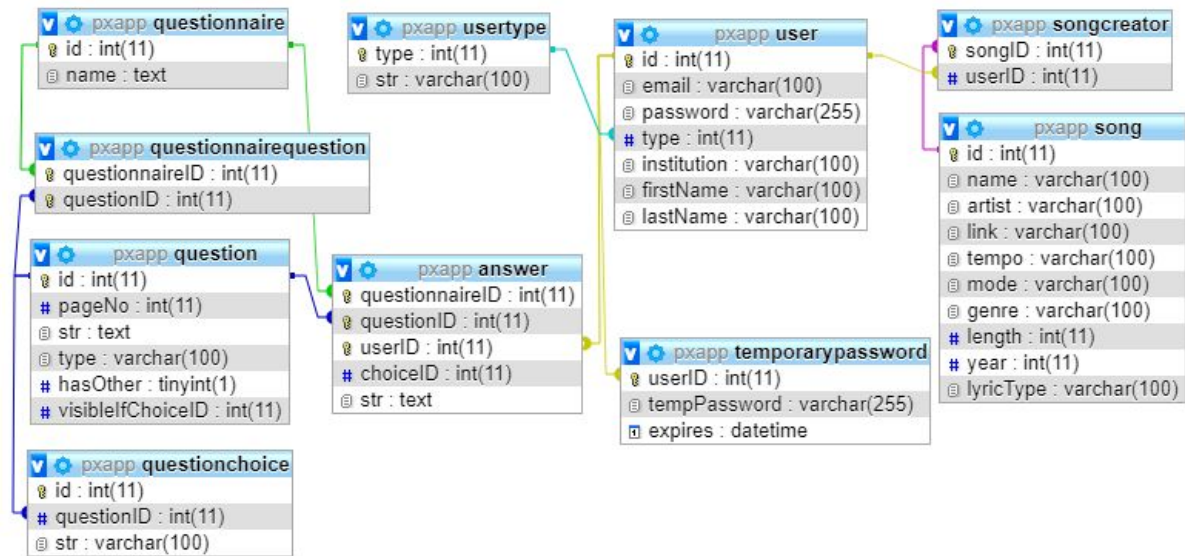
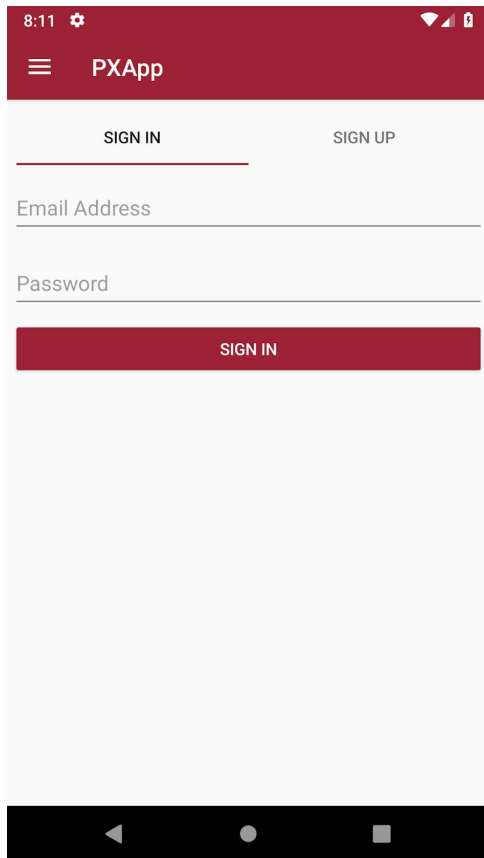


Figure 7 - Database Schema

5.5 Screen Designs

The following images display the screen designs of the mobile application in different sections of the app.



8:11 PXApp

SIGN IN SIGN UP

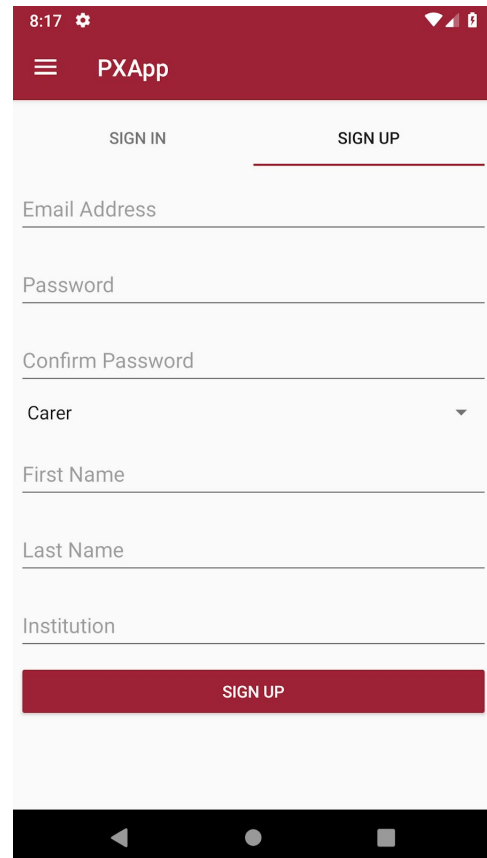
Email Address

Password

SIGN IN

This is a mobile application screen for signing in. It features a dark red header with the time '8:11' and the app name 'PXApp'. Below the header, there are two tabs: 'SIGN IN' (active) and 'SIGN UP'. The main content area has two text input fields labeled 'Email Address' and 'Password'. At the bottom of the form is a dark red button labeled 'SIGN IN'. The screen is framed by a black Android navigation bar at the bottom.

Screen 1 - Sign In



8:17 PXApp

SIGN IN SIGN UP

Email Address

Password

Confirm Password

Carer

First Name

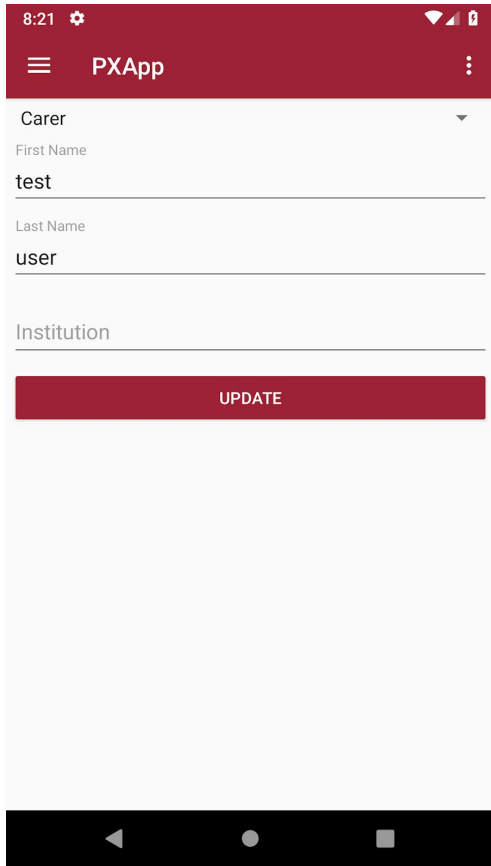
Last Name

Institution

SIGN UP

This is a mobile application screen for signing up. It features a dark red header with the time '8:17' and the app name 'PXApp'. Below the header, there are two tabs: 'SIGN IN' and 'SIGN UP' (active). The main content area has six text input fields labeled 'Email Address', 'Password', 'Confirm Password', 'Carer' (with a dropdown arrow), 'First Name', and 'Last Name'. Below these is another text input field labeled 'Institution'. At the bottom of the form is a dark red button labeled 'SIGN UP'. The screen is framed by a black Android navigation bar at the bottom.

Screen 2 - Sign Up



8:21 PXApp

Carer

First Name

test

Last Name

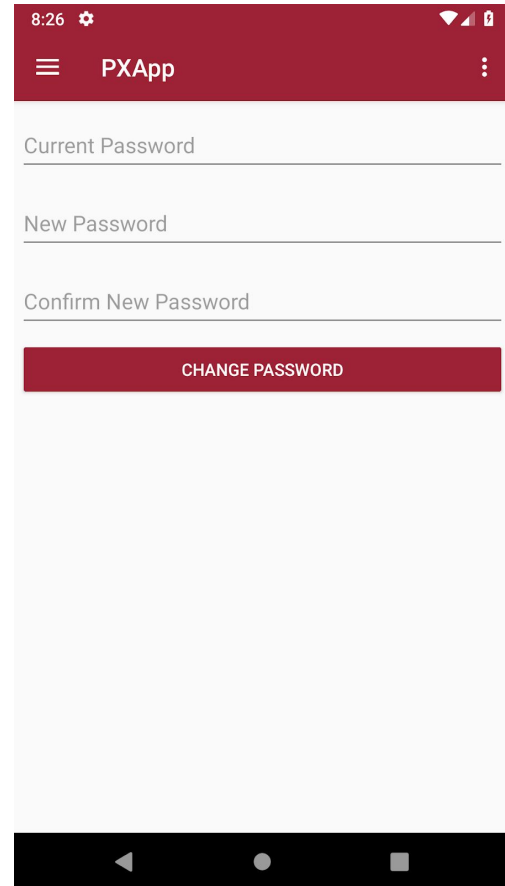
user

Institution

UPDATE

This screenshot shows the 'Update Account Info' screen. It features a red header bar with the time '8:21', a settings icon, the app name 'PXApp', and a menu icon. Below the header, there are three input fields: 'First Name' with the value 'test', 'Last Name' with the value 'user', and 'Institution'. A red 'UPDATE' button is positioned below these fields. The bottom of the screen shows the Android navigation bar.

Screen 3 - Update Account Info



8:26 PXApp

Current Password

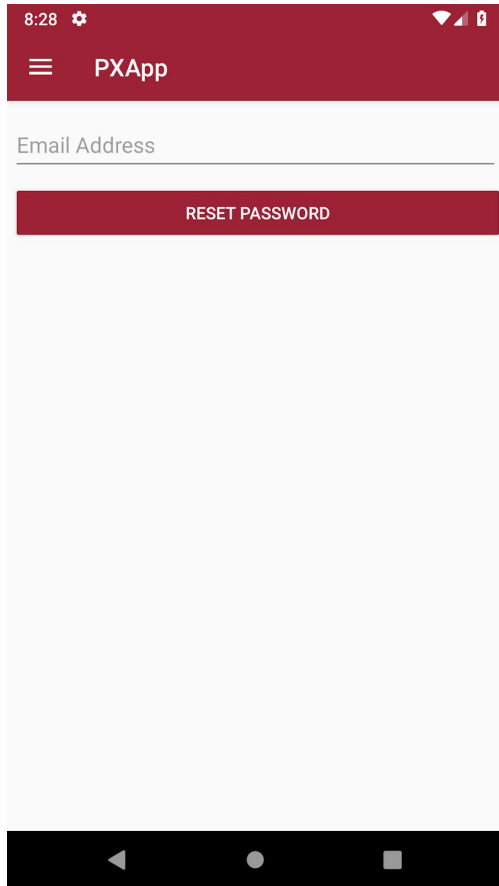
New Password

Confirm New Password

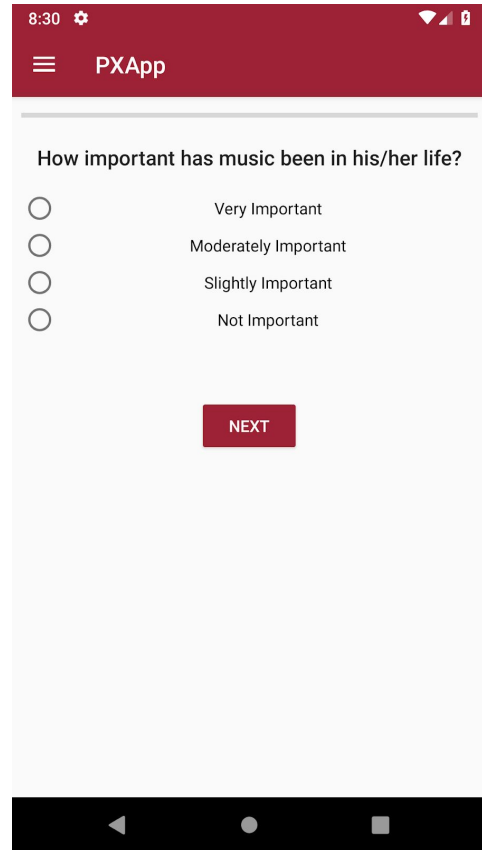
CHANGE PASSWORD

This screenshot shows the 'Change Password' screen. It features a red header bar with the time '8:26', a settings icon, the app name 'PXApp', and a menu icon. Below the header, there are three input fields: 'Current Password', 'New Password', and 'Confirm New Password'. A red 'CHANGE PASSWORD' button is positioned below these fields. The bottom of the screen shows the Android navigation bar.

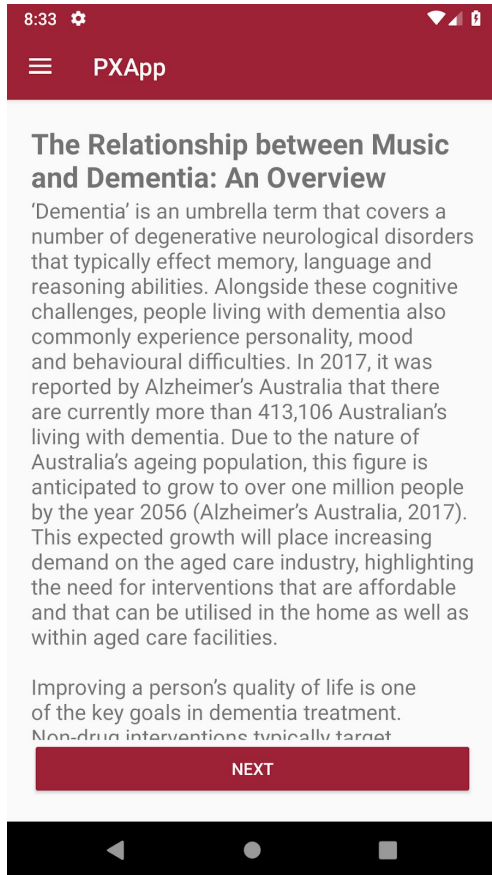
Screen 4 - Change Password



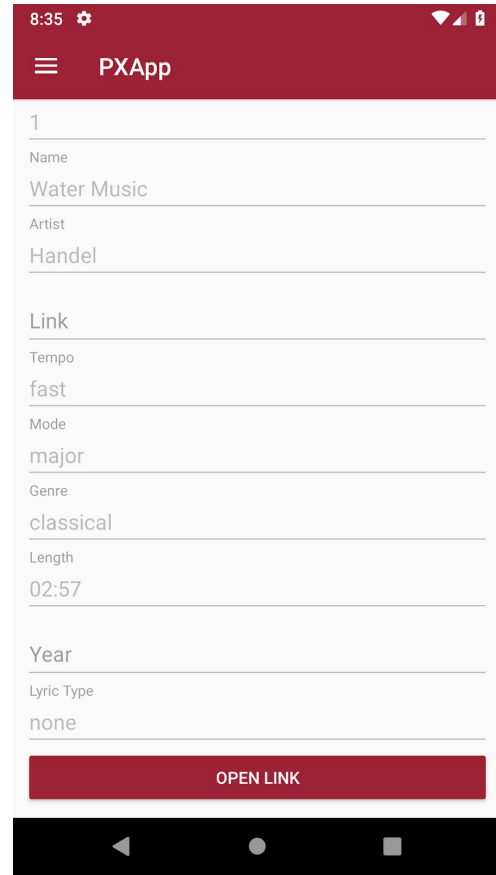
Screen 5 - Reset Password



Screen 6 - Surveys



Screen 7 - Information



Screen 8 - Songs

Message No.	Message Detail	Associated Use Case	Associated Screens
Msg_01	Email can not be empty		Screen 1
Msg_02	Invalid email		Screen 1
Msg_03	Password can not be empty		Screen 1
Msg_04	Invalid email or password		Screen 1
Msg_05	Signed in		Screen 1
Msg_06	Email can not be empty	UC01	Screen 2
Msg_07	Invalid email	UC01	Screen 2
Msg_08	Password can not be empty	UC01	Screen 2
Msg_09	Please confirm your password	UC01	Screen 2
Msg_10	Passwords don't match	UC01	Screen 2
Msg_11	First name can not be empty	UC01	Screen 2
Msg_12	Last name can not be empty	UC01	Screen 2
Msg_13	Email in use	UC01	Screen 2
Msg_14	Signed up	UC01	Screen 2
Msg_15	First name can not be empty	UC02	Screen 3
Msg_16	Last name can not be empty	UC02	Screen 3
Msg_17	Successfully updated account info	UC02	Screen 3
Msg_18	Password can not be empty	UC04	Screen 4
Msg_19	Please confirm your password	UC04	Screen 4
Msg_20	Passwords don't match	UC04	Screen 4
Msg_21	Current password mismatch	UC04	Screen 4
Msg_22	Password updated	UC04	Screen 4
Msg_23	Email can not be empty	UC05	Screen 5
Msg_24	Invalid email	UC05	Screen 5
Msg_25	No account was found with the specified email	UC05	Screen 5

Msg_26	Email sent	UC05	Screen 5
Msg_27	Please make a selection	UC10	Screen 6
Msg_28	Answers submitted	UC11	Screen 6
Msg_29	No link found	UC14	Screen 8

6 - Test Plan

6.1 Features/Use Cases

Types of Testing	Pass Fail Criteria	Personnel	When / Where	Risks
App Interface - Design Testing	No agreed upon interface specifications	Team member - Caitlin	Prior to week 10	Find any design flaws
App Interface - Client Acceptance	Client accepts design, change if not aligned	Client - Chris	Prior to week 10	Too many changes requested
App - Register Testing	User is added to the database	Team member - Jordan	Week 10	Users can't register
App - Log in Testing	User is able to login and view their personal data	Team member - Jordan	Week 10	Users can't login or personal data isn't displayed
App - Password Reset Option Functionality Testing	User is able to reset their password and login with new password	Team member - Jordan	Week 10	User cant reset their password or login using new password
Questionnaire - Validation Testing	User input match supplied parameters	Team member - Alex	Week 10	Validation fails, needs to be changes
Questionnaire - Server Connection Testing (Submitting Answers)	Submitted answers by user are added to database	Team member - Jordan	Week 10	Answers not submitted, server communication is wrong
Provided Text - Access and Readability Testing	User is able to navigate to and read text provided in app tabs	External Test User	Week 11	Unable to navigate to text
Admin Site - Managing Users Testing	Admin user can manage content and application can view updates	Team member - Alex	Week 11	Unable to action CRUD operations due to server/app communication
Admin Site - Managing Questionnaire Testing	Admin user can manage content and application can view updates	Team member - Caitlin	Week 11	Unable to action CRUD operations due to server/app communication
Admin Site - Managing Songs Testing	Admin user can manage content and application can view updates	Team member - Caitlin	Week 11	Unable to action CRUD operations due to server/app communication

6.2 Candidate Test Cases

Use Case / Feature		Registering a New User/ Login after Registration	
Test Purpose		To ensure users are able to create an account and login using the supplied credentials	
Expected Result		Users can register and login using their provided email address and password	
Success/Failure		<<To be entered after testing, results included in Handover Report>>	
Test Data			
Column Name	Set 1	Set 2	Set 3
Email Address	test1@test.com	test2@test.test	test3@test.com
Password	test1	test2	test3
Expected Result	Success	Failure	Success
Success/Failure			
Date Tested			
Actions			

Use Case / Feature		Reset User Password	
Test Purpose		To ensure users are able to reset their password using their email address if they forget their password or wish to change it	
Expected Result		Users can enter their accounts email address and receive a link to reset their password	
Success/Failure		<<To be entered after testing, results included in Handover Report>>	
Test Data			
Column Name	Set 1	Set 2	Set 3
Email Address	test1@test.com	test2@test.com	test3@test.com
Reset Email Entered	test1@test.com	test222@test.com	test3@test.com
Old Password	test1	test2	test3

New Password	test11	test22	test33
Expected Result	Success	Failure	Success
Success/Failure			
Date Tested			
Actions			

Use Case / Feature	Questionnaire Answers Sent to Database
Test Purpose	To ensure entered answers to the surveys are saved in the database and associated with the relevant user
Expected Result	The database is populated with the correct inputs and when loading the surveys in the app any previous answers are loaded to the user
Success/Failure	<<To be entered after testing, results included in Handover Report>>

Test Data			
Column Name	Set 1	Set 2	Set 3
Email Address	test1@test.com	test2@test.com	test3@test.com
Password	test1	test2	test3
Questionnaire 1, Question 1 Answer	Agree	Neither Agree Nor Disagree	Disagree
Expected Result	Success	Success	Success
Success/Failure			
Date Tested			
Actions			

Use Case / Feature	Questionnaire Answers Sent to Database
Test Purpose	To ensure entered answers to the surveys are saved in the database and associated with the relevant user
Expected Result	The database is populated with the correct inputs

	and when loading the surveys in the app any previous answers are loaded to the user		
Success/Failure	<<To be entered after testing, results included in Handover Report>>		
Test Data			
Column Name	Set 1	Set 2	Set 3
Email Address	test1@test.com	test2@test.com	test3@test.com
Password	test1	test2	test3
Questionnaire 1, Question 1 Answer	Agree	Neither Agree Nor Disagree	Disagree
Expected Result	Success	Success	Success
Success/Failure			
Date Tested			
Actions			

Use Case / Feature		Manage Questionnaires from Admin Site	
Test Purpose		Ensure admin users can manage (edit, add, delete) questionnaires and their questions from the admin website	
Expected Result		Any changes (edits, additions or removals) are represented in the android application	
Success/Failure		<<To be entered after testing, results included in Handover Report>>	
Test Data			
Column Name	Set 1	Set 2	Set 3
Questionnaire 1, Question 1 (Before text changed)	‘In the last 2 weeks I have had little interest or pleasure in doing things’		
Questionnaire 1, Question 1 (After test change)	‘Testing Set 1’		
Questionnaire 2, Page		‘Agitation or anxiety’	

1 (Before removal of question 1)			
Questionnaire 2, Page 1 (After removal of question 1)		'Withdrawal or Apathy'	
Listview Displaying Questionnaires (Before addition)			5 questionnaire options displayed in listview
Listview Displaying Questionnaires (After addition)			6 questionnaire options displayed in listview
Expected Result	Success (Display question 1 text as normal before change, display 'Testing Set 1' after change)	Success (On page 1, display Question 1 before removal, display question 2 after removal)	Success (On listview of questionnaires there should be 5 before the addition, and 6 after the addition)
Success/Failure			
Date Tested			
Actions			

Use Case / Feature		Manage Users from Admin Site	
Test Purpose		Ensure admin users can manage (edit, add, delete) users from the admin website	
Expected Result		Any changes (edits, additions or removals) are represented in the android application	
Success/Failure		<<To be entered after testing, results included in Handover Report>>	
Test Data			
Column Name	Set 1	Set 2	Set 3
Email Address	test1@test.com	test2@test.com	test3@test.com
Password	test1	test2	test3
First Name		John	
Institution			WSU

New Password (Edited on Admin site)	test11		
New First Name (Edited on Admin site)		TESTING	
New Institution (Edited on Admin site)			UWS
Expected Result	Success (Display new password in database and user can ONLY login with new password)	Success (first name changes in database and for user when they are logged in, changed from 'John' to 'TESTING')	Success (institution changes in database and for user when they are logged in, changed from 'WSU' to 'UWS')
Success/Failure			
Date Tested			
Actions			

Use Case / Feature		Manage Songs from Admin Site	
Test Purpose		Ensure admin users can manage (edit, add, delete) songs from the admin website	
Expected Result		Any changes (edits, additions or removals) are represented in the android application	
Success/Failure		<<To be entered after testing, results included in Handover Report>>	
Test Data			
Column Name	Set 1	Set 2	Set 3
Song Name	Water Music		
Song BPM		177	
Song Genre			Classical
New Song Name (Edited on Admin site)	TESTING Music		
New Song BPM (Edited on Admin site)		80	
New Song Genre			TESTING

(Edited on Admin site)			
Expected Result	Success (Display new song name in database, 'TESTING Music' replacing 'Water Music')	Success (Display new song BPM in database, 80 replacing 177)	Success (Display new song genre in database, 'Classical' replacing 'TESTING')
Success/Failure			
Date Tested			
Actions			

Conclusion

This document will provide an in depth insight into the structure of the systems and applications being developed and show our plans for ensuring the usability and quality of the product through testing. It provides a detailed analysis of all functional and nonfunctional requirements based on the high level business functions in the proposal. Any identified risk and constraint, our design consideration and structural diagrams of the application and website functions are provided with all the necessary description. Use case diagrams further describe the actions and permissions available to the different user types and the functions of the server application. An Entity Relationship Diagram, a Database Schema and screenshots of all the applications detail the inner workings of the systems we have developed and the functionality of the application. Our testing plans are also detailed at the end and demonstrate our desire and plan to deliver a functional and reliable project.