Milestone 4 Report for:

Middletown Radio Application

Development Team:
Rachel Harvey
Nick Torres
Kristen Weber
Seth Winslow

CS 495: Capstone
Ball State University
April 27, 2018

Table of Contents

1.0: Project Information

1.1: Client Introduction

Dr. Robert Willey is an Associate Professor at Ball State University in the School of Music's Music Media Production and Industry department. He has worked with many Ball State Computer Science Capstone groups before for many different projects.

Most contact between the capstone group and the client is going to be done through email. However, for meetings to demonstrate big updates, we will schedule meetings with him. His contact information is as follows:

Robert Willey | rkwilley@bsu.edu | Music Instruction Building (MI) 211 | 765-285-5537

1.2: Team Contact Information

Rachel Harvey: raharvey@bsu.edu | 812-736-2817
Nick Torres: njtorres@bsu.edu | 630-470-7896
Kristen Weber: kmweber@bsu.edu | 812-363-4870
Seth Winslow: swwinslow@bsu.edu | 317-498-7558

1.3: Statement of Task

We are creating two mobile applications that allows listeners of the application to listen to radio stations located in the Midwest as an alternative to larger commercial based national radio stations. The client defines the "Midwest" as the states of Minnesota, Iowa, Missouri, Wisconsin, Illinois, Indiana, Michigan, Ohio, and Ontario.

More specifically, listeners of the application will be able to listen to radio stations that have been approved by the admin. Listeners will have the option to save sets of six stations as preset banks, submit suggested stations for approval, scan all radio stations, and customize the radio stations by geographical location, style, ownership, popular, and list to be scanned.

We will also be creating an admin panel for our client to use. He will use this to manage the stations that are on the application. He will be able to create, edit, review, and delete stations that are stored in the database as well as edit and review categories in the database.

2.0: Preliminary Requirements Analysis

2.1: Overview of the Application:

Middletown Radio Application is going to be a mobile application for iOS and Android. The application allows you to listen to radio stations that are stored in the database and that have been approved by the administrator of the application, Dr. Willey. There are five main screens: Home, Load, Save, Setup, and List. Below you can see the functional requirements for each one of these pages.

2.2: Functional Requirements:

2.2.1: Menu - Requirements

2.2.1.1: When the listener of the application clicks on the mobile menu, a screen will slide out to cover ¾ of the screen width with the menu options: Radio, Load, Save, Setup, Middletown Radio, and Suggest A Station.

2.2.1.1.1: When the listener clicks on Radio, they shall be redirected to the radio screen.

2.2.1.1.2: When the listener clicks on ‘Load’, they shall be redirected to the load screen.

2.2.1.1.3: When the listener clicks on ‘Save’, they shall be redirected to the save screen.

2.2.1.1.4: When the listener clicks on ‘Setup’, they shall be redirected to the setup screen.

2.2.1.1.5: When the listener clicks on ‘Middletown Radio’, they shall be redirected to the Middletown Radio website.

2.2.1.1.6: When the listener clicks on ‘Suggest A Station’, they shall be redirected to the Middletown Radio website to the page where they can suggest a station.

2.2.2: Home (Radio) - Requirements

2.2.2.1: When the listener opens the app, a pop up will come up that says, ‘You can continue using the app in portrait view, or you may flip your phone to see the radio in landscape mode.’

2.2.2.2: When the listener clicks on one of the double arrow scan buttons, the radio will scan through stations that currently match their setup requirements. The scan button will then become red to indicate they are currently scanning.

2.2.2.2.1: The radio station will play for 5 seconds before changing to the next station.

2.2.2.2.2: The radio will stop scanning when the listener clicks on the double arrow again and the button color will change back to black.

2.2.2.3: When the listener clicks on one of the single arrow scan buttons, the station will change to the next station that currently matches their setup requirements, and the button will flash green.

2.2.2.4: When the listener clicks on 'Play', if the current station showing in the display area isn't playing, then it will begin playing and the button will turn green. If it is already playing, then it will do nothing.

2.2.2.5: When the listener clicks on 'Stop', if the current station showing in the display area is playing, then it will stop playing and the button will turn green. If it isn't playing, then it will do nothing.

2.2.2.6: When the listener clicks on a preset that has a station saved to it, the radio will change to that station, add a heart next to the station title, and begin playing.

2.2.2.7: When the listener clicks on a preset that is a number, meaning a station is not saved to it, then it will do nothing and the radio will continue playing the station it currently has up.

2.2.2.8: When the listener long holds a preset button that currently has a number number displaying, then the station currently playing will be saved in that preset spot. The number in the preset area will change to the stations abbreviation and a heart will appear next to the station title in the display area.

2.2.2.9: When the listener long holds a preset button that currently has a station abbreviation displaying, then the station currently playing will be saved over the station currently in that preset spot. The station abbreviation would change to the current station abbreviation and a heart will appear next to the station title in the display area.

2.2.2.10: When the listener clicks on the question mark, a popup will display these instructions: 'You can use this screen to scan through radio stations and set presets. If you would like to filter what stations you have playing on your radio, please go to setup in the mobile menu'.

2.2.3: Load - Requirements

2.2.3.1: When the listener long holds on one of their preset banks, the listener will be redirected to the home screen and they will see their presets for that preset bank and the first preset will begin playing.

2.2.3.2: When the clicks holds on one of their preset banks, a pop up will display their list of presets for that preset bank.

2.2.3.2.1: When the listener clicks on 'Load', they shall be redirected to the radio with that preset bank loaded and it will have begun playing the first preset station.

2.2.3.3: When the listener clicks on 'Delete', small checkmarks will appear to the left of each preset bank where the listener can then select which preset bank(s) they would like to delete.

2.2.3.3.1: When the listener clicks on ‘Delete’ again, a pop up will display warning the listener if they delete these banks they will be removed.

2.2.3.3.1.1: When the listener clicks on ‘Delete’, they shall be redirected to the preset bank screen with their selected stations deleted.

2.2.3.3.1.2: When the listener clicks on ‘Cancel’, they shall be redirected to the preset bank screen.

2.2.3.4: When the listener clicks on the question mark, a popup will display these instructions: ‘You can use this screen to load in any of your currently saved preset banks by long holding on one or delete any preset banks you have saved by clicking ‘Delete’ and then selecting which ones you would like to delete.’

2.2.4: Save - Requirements

2.2.4.1: When the listener clicks on ‘Save’, it adds a number to the list of preset banks and allows listener to type in a new preset bank name.

2.2.4.2: When the listener clicks on ‘Delete’, small checkmarks will appear to the left of each preset bank where the listener can then select with preset bank(s) they would like to delete.

2.2.4.2.1: When the listener clicks on ‘Delete’ again, a pop up will display warning the listener if they delete these banks they will be removed.

2.2.4.2.1.1: When the listener clicks on ‘Delete’, they shall be redirected to the saved screen with their selected stations deleted.

2.2.4.2.1.2: When the listener clicks on ‘Cancel’, they shall be redirected to the save screen.

2.2.4.2: When the listener clicks on the question mark, a popup will display these instructions: ‘You can use this screen to save the current presets you have on your radio by clicking ‘New’ or delete any of your currently saved preset banks by clicking ‘Delete’ and then selecting which ones you would like to delete.’

2.2.5: Save - Optional

2.2.5.1: When the listener clicks on one of their currently saved preset banks, it will save the presets from their radio over the presets currently saved to that preset bank.

2.2.6: Setup - Requirements

2.2.6.1: When the listener clicks on ‘Geographical’, a pop up will display the geographical locations the listener can select.

2.2.6.1.1: When the listener clicks on ‘Cancel’, the pop up will disappear and nothing will happen.

2.2.6.1.2: When the listener clicks on ‘Done’, the locations will be saved and it will the pop up will disappear.

2.2.6.1.3: When the listener clicks on one of the checkboxes, the stations in that geographical area will be added to the stations that they will be scanning by on the radio screen.

2.2.6.2: When the listener clicks on ‘Style’, a pop up will display the style options the listener can select.

2.2.6.2.1: When the listener clicks on ‘Cancel’, the pop up will disappear and nothing will happen.

2.2.6.2.2: When the listener clicks on ‘Done’, the styles will be saved and it will the pop up will disappear.

2.2.6.2.3: When the listener clicks on one of the checkboxes, the stations of that style will be added to the stations that they will be scanning by on the radio screen.

2.2.6.3: When the listener clicks on ‘Ownership’, a pop up will display the ownership options the listener can select

2.2.6.3.1: When the listener clicks on ‘Cancel’, the pop up will disappear and nothing will happen.

2.2.6.3.2: When the listener clicks on ‘Done’, the ownerships will be saved and the pop up will disappear.

2.2.6.3.3: When the listener clicks on one of the checkboxes, the stations in that specific ownership category will be added to the stations that they will be scanning by on the radio screen.

2.2.6.4: When the listener clicks ‘Popular’, a pop up will display the popular stations the listener can select.

2.2.6.4.1: When the listener clicks on ‘Cancel’, the pop up will disappear and nothing will happen.

2.2.6.4.2: When the listener clicks on ‘Done’, the popular stations selected will be saved and the pop up will disappear.

2.2.6.4.3: When the listener clicks on one of the checkboxes, that popular station will be added to the stations that they will be scanning by on the radio screen.

2.2.6.5: When the listener clicks on ‘List’, they shall be redirected to the list screen.

2.2.6.6: When the listener clicks on the question mark, a popup will display these instructions: ‘You can use this screen to customize your radio stations by geographical location, style, ownership, popular, and list of all stations by clicking on any option and select which stations you would like to include or not include.’

2.2.7: List - Requirements

2.2.7.1: When the listener checks a station, it will be added to their selection of stations when scanning.

2.2.7.2: When the listener unchecks a station, it will be removed from their selection of stations when scanning.

2.2.7.3: When the clicks on a station name, it will display a pop up of the information for that station.

2.2.7.4: When the listener clicks on the question mark, a pop up will display these instructions: ‘You can use this screen to see what stations are currently going to be on your radio and what won’t be on your radio. You can then check or uncheck certain stations for your radio station selection.’

2.2.8: Admin Panel - Requirements

2.2.8.1: The admin will be prompted for login credentials to get into the admin panel.
2.2.8.2: If the admin has correctly put in their credentials, then they will be sent to the home screen where they can select ‘Midwest’.
2.2.8.3: When the admin clicks Stations, all station data will be displayed and editable in a table.
2.2.8.4: When the admin selects a filter for the table, such as popular or category, the table will display station data based on the filter criteria.
2.2.8.5: Within the station table, there will be a clickable checkbox that indicates which stations have been approved. If the admin clicks this box for a station row, it will toggle the station to be published.
2.2.8.6: If the admin clicks on the download button, a download will begin to export the current state of the database in csv format.

2.2.9: Public Website Portion - Requirements

2.2.9.1: On the public website, the web users will be able to view all the published radio station data in a table format.
2.2.9.2: On the public website, web users will be able to submit a form to suggest new radio station for admin approval.
2.2.9.3: When a web user is viewing the station table and selects a filter such as by popular or category, the table will display the appropriate date for each filter.

2.2.10: Backend Configurations - Requirements

2.2.10.1: All of the station information is going to be stored on a mySQL database.
2.2.10.1.1: That stations will be stored in a stations table. This will include, ID, Short Name, Long Name, City, State, Slogan, Genre ID, Ownership.
2.2.10.2 Both mobile applications, the admin panel, and the public website features will make requests to a PHP API.

2.3: Non-Functional Requirements:

2.3.1: The application must be compatible for Android 4.1 and iOS 9 and above.
2.3.2: The UI of both applications should have some characteristics of a physical radio such as preset buttons, scan functionality, etc.
2.3.3: The UI of the Android app and iOS app should be nearly identical in terms of the design and technical limitations.

2.3.4: The application will update stations on open.
2.3.5: The application will preserve data usage when a station is not currently playing.
2.3.6: The backend needs to be written in PHP 4.3.0 or higher.

2.4: GUI Mockups

Below are our UI designs for our application. They may be subject to change throughout development based on client feedback.

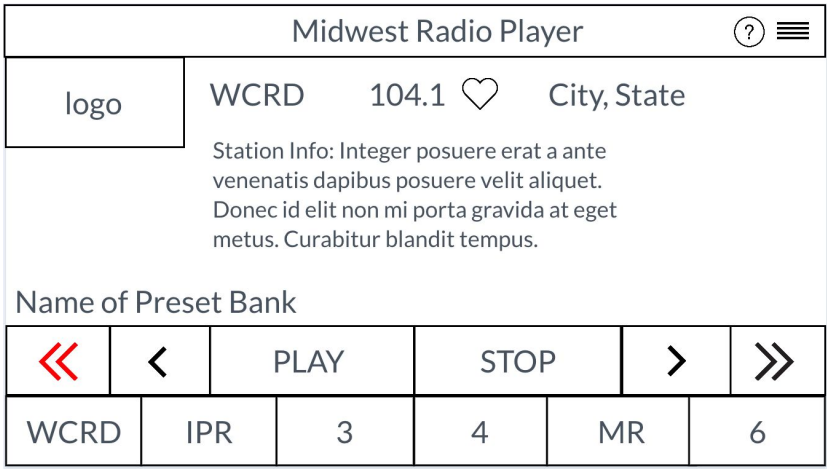


Figure 1: The home screen in landscape view.

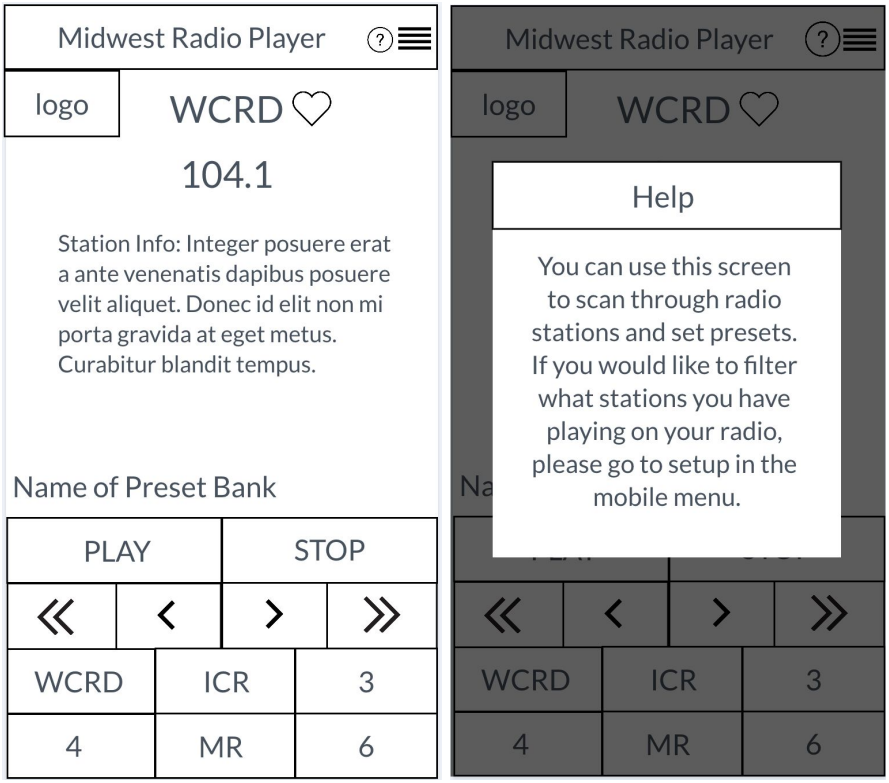


Figure 2: The home screen in portrait view (left) and the home screen showing the help pop up (right). This help pop up is how all help screens will look throughout the app.

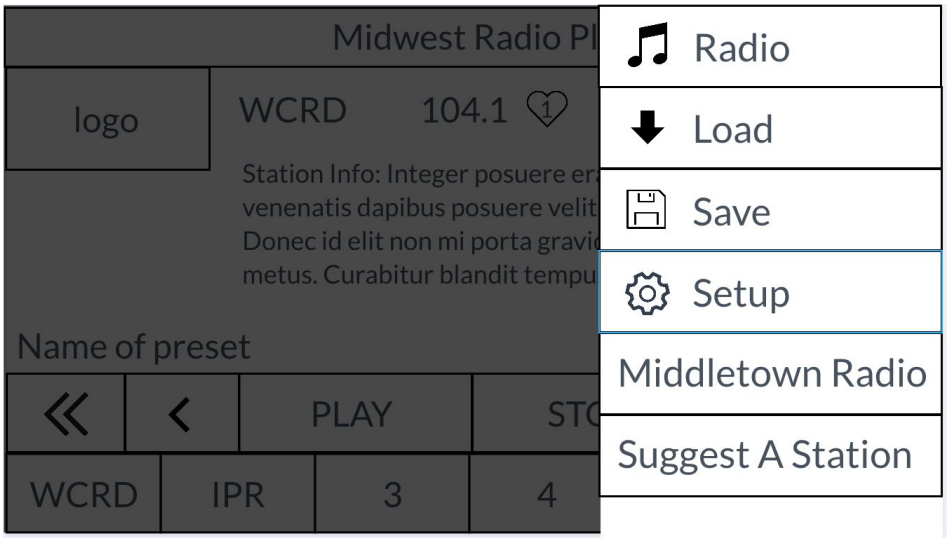


Figure 3: The mobile menu slide out.

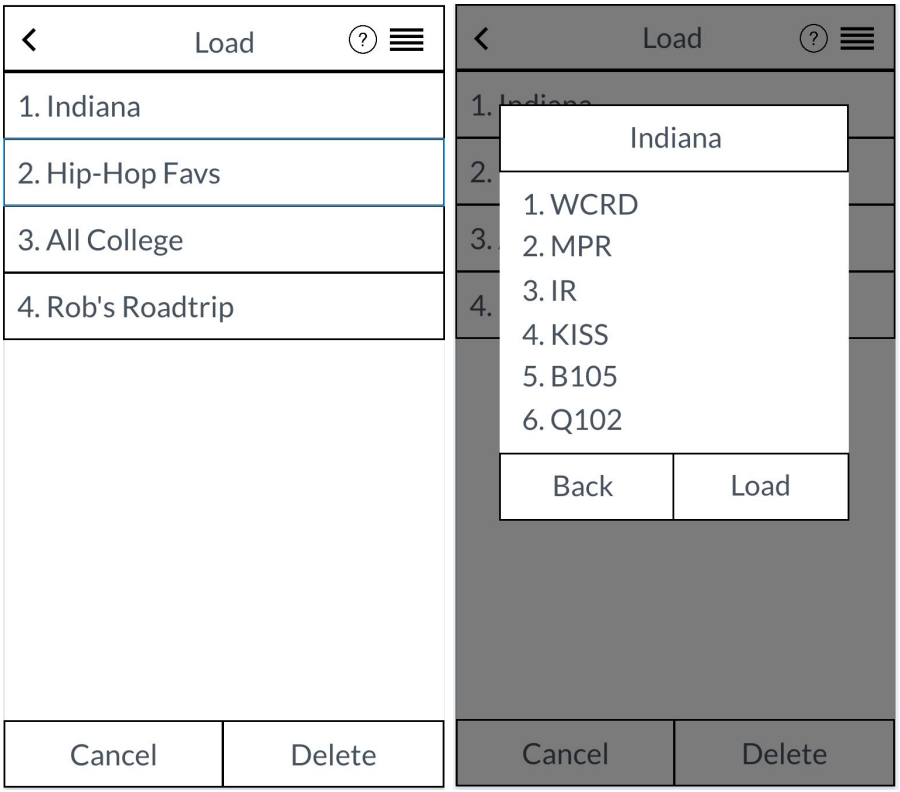


Figure 4: The load screen (left) and the presets for an individual preset bank pop up (right). The load screen on the left looks similar to the save screen.

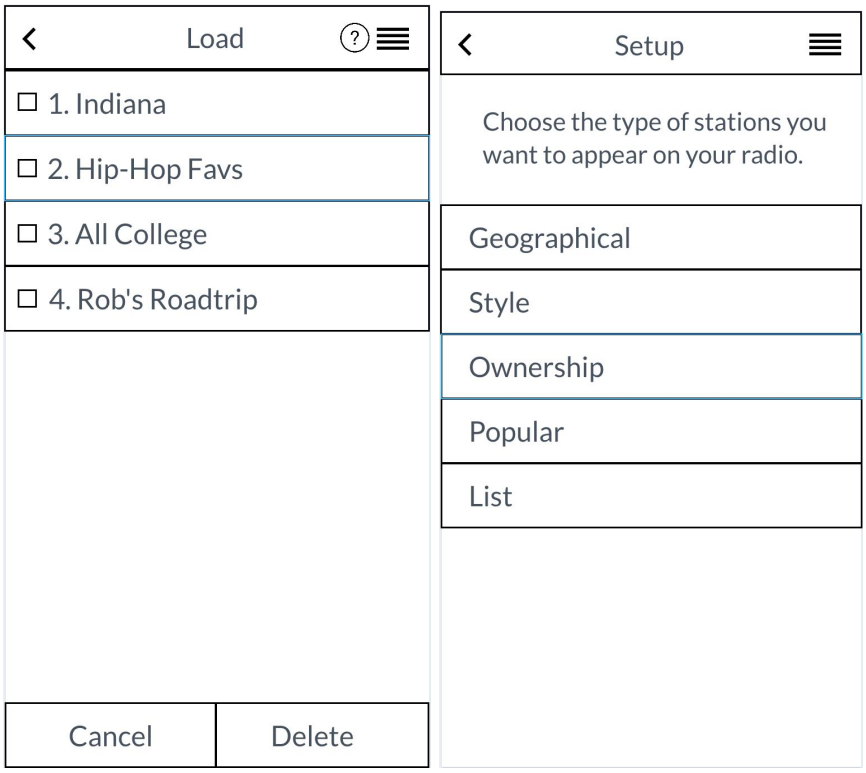


Figure 5: The load delete screen (left) and the setup screen (right). The load delete screen on the left looks similar to the save delete screen.

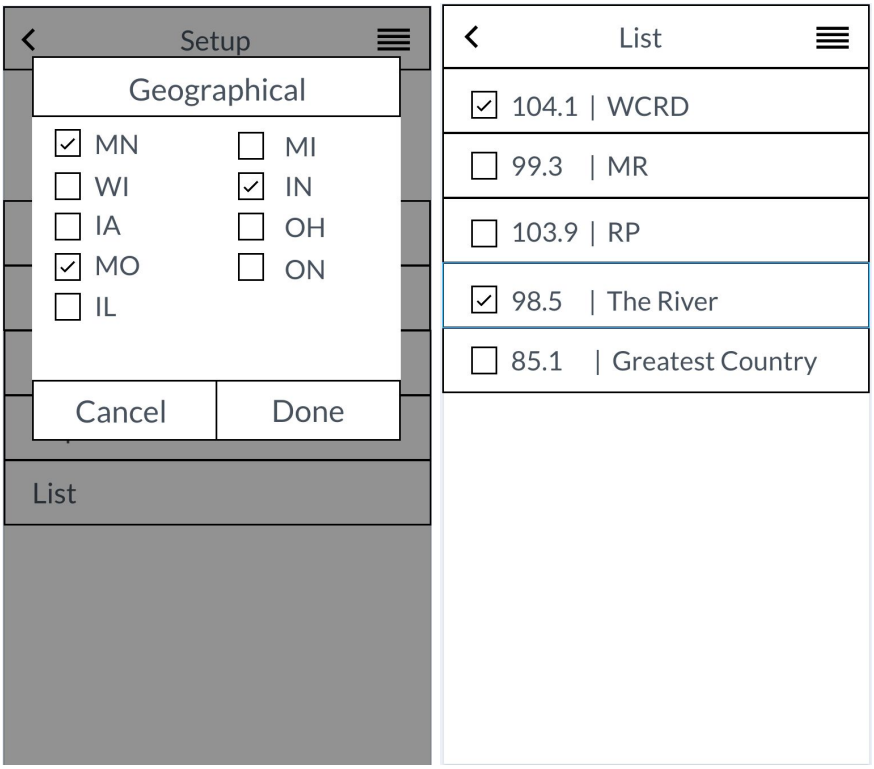


Figure 6: The setup geographical location pop up (left) and the list screen (right). The design of the pop up on the right is identical to the pop up for the other setup options.

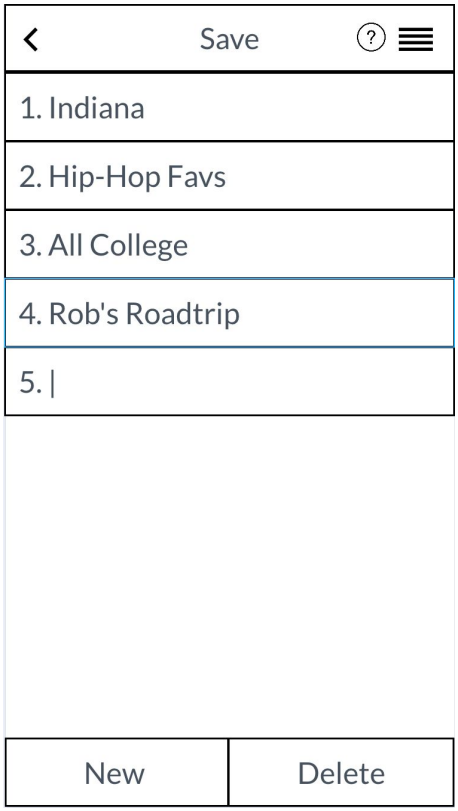


Figure 7: The save screen once the listener clicks ‘New’.

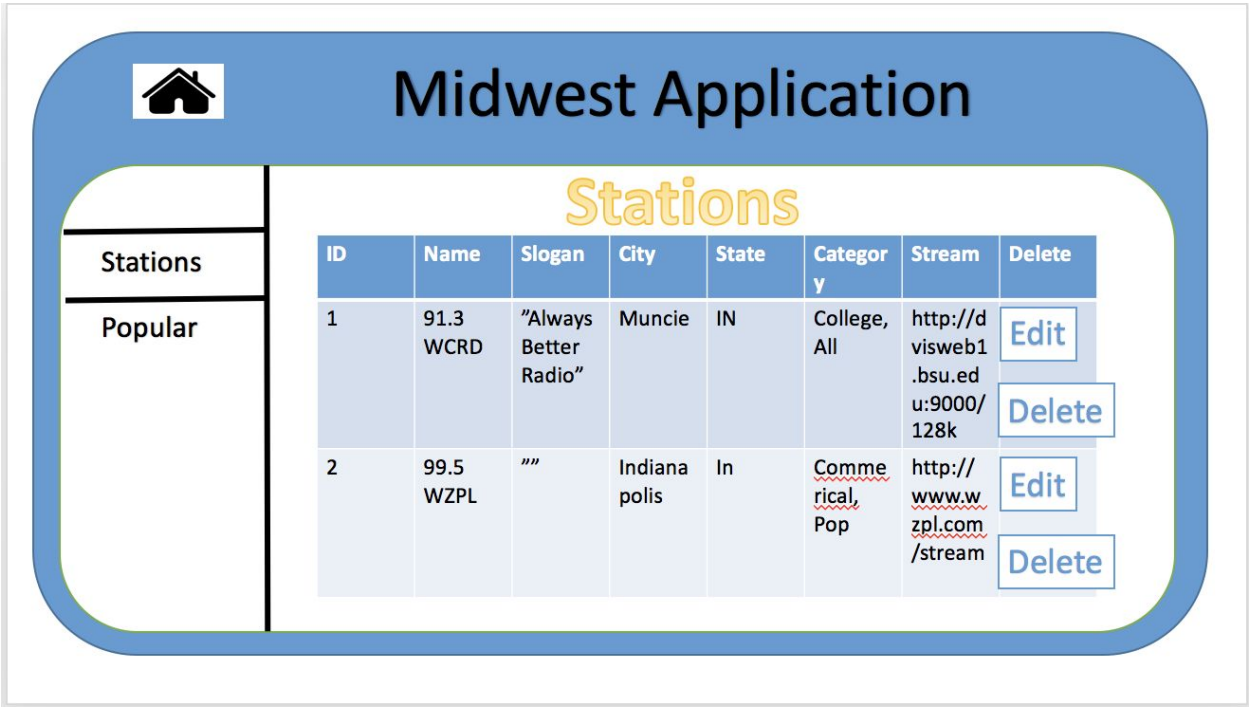


Figure 8: Admin Screen displaying list of stations inside the application.

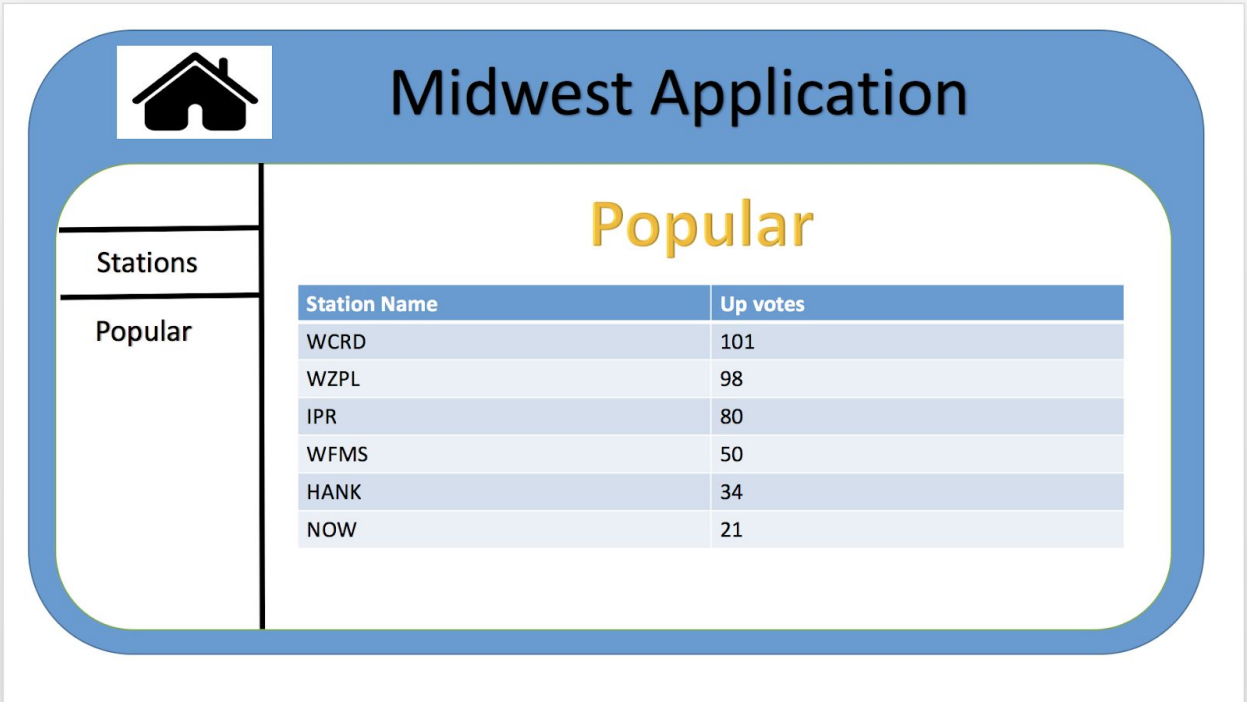


Figure 9: Admin view displaying the popular stations in the app with the popular count.

2.5: Suggested Deliverables

2.5.1: Management Deliverables

2.5.1.1: Gantt Chart

2.5.1.1.1 To keep our iterations and progress organized, a Gantt Chart will be used to plan out our iterations and track our progress. This chart will be kept in our Google drive folder so we all have access to update it when necessary.

2.5.1.2: Repository README

2.5.1.2.1: This file will contain the repository information for any future developers. This will include dependencies, versions, releases and any other important notes about the repo.

2.5.1.3: Admin README

2.5.1.3.1: This file will contain instructional information for the client and any future admins about the different sections of the admin panel. This will include how to create, approve, update, and delete current stations. Additionally, it will include instructions on how to add additional categories, and download the current state of the database to csv format.

2.5.2: Technical Deliverables

2.5.2.1: Admin Panel

2.5.2.1.1: In order to control the data, the client will need a panel where he can approve submissions, add submissions manually, and edit current stations should any information change. The current deadline is mid/late Fall 2017 semester.

2.5.2.2: Minimum Functioning Apps

2.5.2.2.1: By the end of the Fall 2017 semester, we will have two working apps, with the minimum functionality possible, the ability to see a list of stations and pick one to play.

2.5.2.3: Fully Functioning Apps

2.5.2.3.1: By the end of the Spring 2018 semester, we will have two fully functioning apps with the ability to scan stations, customize radio stations, and save presets.

2.5.2.4: Public Facing Website Side

2.5.2.4.1: By the end of the Spring 2018 semester, we will have the public information about all the stations that are currently being stored in the database.

3.0: Plan of Attack

3.1: Project Plan

Our project is going to be completed using fourteen sprints which is defined in our Gantt chart. For the first semester we will complete six sprints, and for the second semester we will complete eight. Each sprint will be approximately two week long. By the end of the Fall 2017 semester, we plan to have a functioning app that will display all of the stations that you can click on to listen to. Then, by the end of the Spring 2018 semester, we plan to have all of the features completed with a working app in the app store.

					Periods	1 (W 1:2)	2 (W 3:4)	3 (W5:6)	4 (W6:7)	5 (W8:9)	6 (W10 - 11)	7 (W 12 - 13)	8 (W 14 -15)	9(W16 - 17)	10 (W 18 -19)	11 (W 20 - 21)	12 (W 22 -23)	13 (W 24 - 25)	14 (W 26 - 27)
Activity	Plan Start (Sprints)	Plan Duration (Weeks)	Acutual Start (Weeks)	Actual Duration	Percent Completed	9/24 - 10/7	10/8 - 10/21	10/22 - 11/4	11/5 - 11/18	11/19 - 12/2	12/3 - 12 - 16	1/7 - 1/20	1/21 - 2/3	2/4 - 2/17	2/18 - 3/3	3/11 - 3/24	3/25 - 4/7	4/8 - 4/21	4/22 - 5/4
Requirements analysis	1	1	1	N/A	94%														
UI Mock Ups	1	2	1	N/A	85%														
Frontend Joint Admin Panel	1	3	1	N/A	25%														
Backend Database Creation	1	2	N/A	N/A	0%														
Front End Admin Panel	2	2	N/A	N/A	0%														
Backend Database Creation	2	1	N/A	N/A	0%														
Fontend Public Facing Site	2	1	N/A	N/A	0%														
Implemet Designs	1	6	1	N/A	5%														
Displaying Database information	4	2	N/A	N/A	0%														
Playing Radio Player	5	2	N/A	N/A	0%														
Bug Testing	6	2	N/A	N/A	0%														
Inital Push to App Stores	6	2	N/A	N/A	0%														
Bug Testing	7	2	N/A	N/A	0%														
Menu Navigation	8	1	N/A	N/A	0%														
Setup Scanning	8	4	N/A	N/A	0%														
- Picking Criteria	8	2	N/A	N/A	0%														
- Showing Stations	9	1	N/A	N/A	0%														
- Playing / Stoping Stations	9	1	N/A	N/A	0%														
- Prepare Scanning list for radio	10	1	N/A	N/A	0%														
Presets	10	4	N/A	N/A	0%														
-Edit Presets	10	2	N/A	N/A	0%														
-Saving Presets	10	2	N/A	N/A	0%														
- Loading Presets	11	2	N/A	N/A	0%														
- Toggle between Presets	11	2	N/A	N/A	0%														
- CRUD on banks	11	2	N/A	N/A	0%														
Bug Testing	12	2	N/A	N/A	0%														
Pushing to App Store	13	3	N/A	N/A	0%														

Figure 10: Gantt Chart

3.2: The Process

3.2.1: Development Methodology

For our project, we are going to be following an agile development process, comprised of two week iterations. Following our Gantt chart, we plan to have a set number of tasks per iteration that will be expected to be completed. In the event we do not complete a given task, we will move that task into the next iteration. In the event the unfinished task is not of immediate priority, it may be put off in favor of higher priority tasks.

Using this agile development process is in our best interest in order to adapt to client feedback, as well as divide our work into manageable durations. Because our client is unsure about some features and how they may interact with each other, we need to be prepared for change. By only developing a set of tasks during an iteration we will be able tailor future iterations to changing feedback. Additionally, some of our tasks include technologies we are not completely familiar with. Because of this, we will want to extend the usual agile iteration of one week to two so we have adequate time to complete these tasks.

3.2.2: Technical Details

Since we are developing for Android and iOS we will be exploring using React Native, but if need be we will also use Android Studio and Xcode for development and develop each app separate. For version control and issue tracking we will be using GitHub.

3.3: Visibility

3.3.1: Within the Team

We will communicate online via our group Slack team and share documents over Google Drive. We will meet a minimum of Tuesdays and Thursdays 11:00-12:30 (right before class) with additional times on Monday and Wednesday afternoon and evenings if necessary.

3.3.2: With the Client

Our client has indicated that he prefers to be “hands off,” so we will not be meeting regularly with him. Instead will be communicating via email if we have questions or want to give him a status update. We will reserve face to face meetings for larger concerns and for the review of our progress as we reach major milestones.

4.0: Business and Risks

4.1: Technical Requirements

Though this may change, the following lists required technologies, divided by our group members’ responsibilities:

4.1.1: Version Control and Issue Tracking

4.1.1.1: We will use GitHub account with permissions to read/write to our repo.

4.1.2: Backend

4.1.2.1: The required platform for backend is PHP version 5.3.

4.1.3: Admin Panel

4.1.3.1: The admin panel will consist of HTML/CSS/Javascript. We will also utilize AngularJS.

4.1.4: Mobile - General

4.1.4.1: In order to use React Native for both Android and iOS development, we will have to configure our individual development environments to utilize package managers and mobile emulators/simulators. Depending on the operating systems available, this may not be exactly the same for each developer's environment.

4.1.5: Mobile - iOS

4.1.5.1: In the event that we need to supplement React Native with native iOS development, we will need access to Xcode.

4.1.6: Mobile - Android

4.1.5.1: In the event that we need to supplement React Native with native Android development, we will need access to Android studio.

4.2: Business Considerations

4.2.1: Apple and Android Development Profiles

4.2.1.1: The client will provide access to his accounts.

4.2.2: Copyright ownership

4.2.2.1: The copyright will be owned by the client, but the development team will be allowed to include this project in our portfolios and resumes.

4.3: Risk Analysis

4.3.1: Use of React Native

4.3.1.1: React may need to be abandoned if it doesn't suit our needs or give us enough flexibility. This could result in a loss of time/work.
4.3.1.2: React only supports Android 4.1, might have functional limitations supporting a lower API (16).

4.3.2: Full app release

4.3.2.1: In order to publish our app to the Apple App Store and Android Play Store, we will need to go through their official approval processes. If our application is rejected for any reason we will need to have contingency time to resolve any issues.
4.3.2.2: If we are able to release a minimum functioning app in December, this will make May's release smoother as the approval for updates is less intensive.

4.4: Conclusion Milestone #1

Our client has worked with Capstone groups in the past, and so has mostly reasonable expectations of our finished product. However, he has been indecisive about some of the features and their implementation, so scope creep could be an issue as the year continues. We have agreed to deliver a minimum functioning app by the end of the Fall 2017 semester, and we feel confident that this is achievable. Our client then hopes to begin advertising for the application in order to gain an audience for the later app release at the end of the spring semester, which will incorporate all of our requirements. His end goal is to both allow his students to use the app for his classes, but on a larger scale attract more listeners to local midwest radio stations, and we think our product could help him achieve this.

5.0 Current Progress

5.1: Project Update

2.1.0: Functional and Non-Functional requirements have not changed.
2.1.1: We have dropped the use of React Native and will instead be developing two native apps. This does not affect us reaching our goals.

6.0 Object-Oriented Analysis

6.1: Actors

6.1.0: An actor called, the Listener, is a user using the radio application to listen to stations.
6.2.0: An actor called, the Phone, is the mobile application.
6.3.0: An actor called, the Admin, is a person with credentials to login the admin panel to edit stations on the application.
6.4.0: An actor, the Web Server, is the API and MySQL database located on a remote server.

6.2: Use Case Diagram

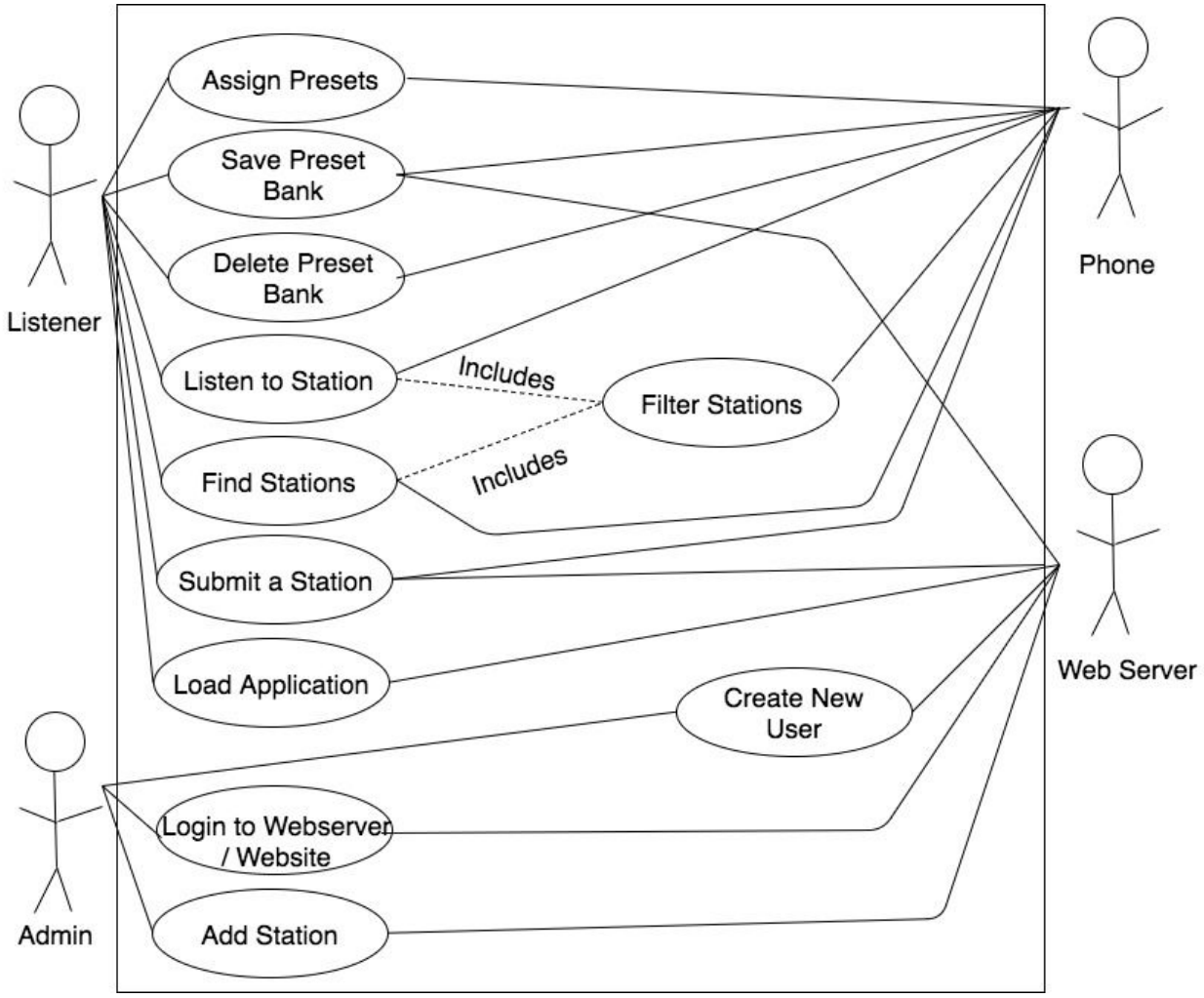


Figure 11: Use Case Diagram

Above, you can see the use case diagram for our use cases. Since our project has a great amount of use cases, this does not show all the use cases for our whole project. We have acknowledged these use cases, but they are being implemented in a later iteration to prevent our milestone project from being 100 pages.

Our diagram shows the actions our listener can perform and actions the admin can perform and how the web server interacts with some of these use cases. An example being, a listener can submit a station. When they do this, they get sent to the website to submit from a form. Once they click ‘Submit’, it goes to the web server to add it to the database, where the admin can approve it as a station.

6.3: Use Case Specifications

6.3.0: Use Case 1: Assign Presets

- Primary Actor: Listener
- Secondary Actor: Phone
- Precondition: Listener has installed and opened the application radio successfully or selected one of their saved preset banks
- Main Flow:
    1. Listener selects to a station
    2. Listener long presses one of the spots in the preset area to assign station
- Alternate Flow:
  *None

6.3.1: Use Case 2: Save Preset Bank

- Primary Actor: Listener
- Secondary Actor: Web Server, Phone
- Precondition: Listener has assigned one or more stations to preset area on play screen
- Main Flow:
    1. Listener clicks on menu button from play screen
    2. Listener then clicks on ‘Save’ in the menu
    3. Application redirects users to the save screen
    4. Listener clicks on ‘New’ at the bottom the screen to add preset bank name
    5. Application displays pop up for listener to type in the name they want for the preset bank
    6. Listener types in the name they want for their presets
    7. Listener clicks ‘Save’ to save the preset bank
    8. Application makes request to web server to set bank stations as favorites
- Alternate Flow:
  4a. Listener can click on a saved preset bank
  4b. Application will save current presets over old preset bank presets
  7a. Listener clicks ‘Cancel’ to not save the preset bank

6.3.2: Use Case 3: Delete Preset Bank

- Primary Actor: Listener
- Secondary Actor: Phone
- Precondition: Listener has saved a preset bank
- Main Flow:
    1. Listener clicks on menu button from play screen

2. Listener then clicks on ‘Save’ in the menu
3. Application redirects users to the save screen
4. Listener clicks on ‘Delete’ at the bottom the screen to delete a preset bank
5. List view adds checkboxes on the left side of preset bank name
6. Listener clicks desired stations to delete
7. Listener clicks ‘Delete’
8. Application displays delete warning box
9. Listener clicks ‘Delete’ to delete preset bank(s)

- Alternate Flow:
  7a. Listener clicks ‘Cancel’ to not delete any preset bank(s)
  9a. Listener clicks ‘Cancel’ to not delete any preset bank(s)

6.3.3: Use Case 4: Listen to Station

- Primary Actor: Listener
- Secondary Actor: Phone
- Precondition: Listener has installed and opened the application successfully
- Main Flow:
  1. Listener clicks the play button on the radio screen
  2. First station in the list will begin to playing
- Alternate Flow:
  2a. Listener lost connection to internet and station didn’t begin to play

6.3.4: Use Case 5: Find Stations

- Primary Actor: Listener
- Secondary Actor: Phone
- Precondition: Listener has installed and opened the application successfully
- Main Flow:
  1. Listener clicks on menu button from play screen
  2. Listener then clicks on ‘List’ in the menu
  3. Application redirects users to the list screen
  4. Listener finds station they want to view by name
  5. Listener clicks on station name
  6. Application displays station information
- Alternate Flow:
  *None

6.3.5: Use Case 6: Filter Stations in Application

- Primary Actor: Listener
- Secondary Actor: Phone
- Precondition: Listener has installed and opened the application successfully
- Main Flow:

1. Listener clicks on menu button from play screen
2. Listener then clicks on ‘Setup’ in the menu
3. Application redirects users to the setup screen
4. Listener clicks on any option to filter stations specifically by that type
5. Application display that specific type’s options to filter by
6. Listener selects filter options they want to filter by
7. Listener clicks ‘Done’ to have those filter options applied
8. Application loads stations to radio that were selected in filter

- Alternate Flow:
  7a. Listener clicks ‘Cancel’ to not have those filter options applied

6.3.6: Use Case 7: Direct to Submit a Station

- Primary Actor: Listener
- Secondary Actor: Phone
- Precondition: Listener has installed and opened the application successfully
- Main Flow:
  1. Listener clicks on menu button from play screen
  2. Listener clicks ‘Suggest a Station’
  3. Listener is redirected to mobile web browser
- Alternate Flow:
  *None

6.3.7: Use Case 8: Login to Web Server / Website

- Primary Actor: Admin
- Secondary Actor: Web server
- Precondition: Website is loaded
- Main Flow:
  1. Admin enters in email and password information
  2. Web server gets admin information
  3. Web server validates admin information
  4. Web server returns session key and id
  5. Admin is redirected to the select database screen
  6. Admin selects ‘Midwest Radio’
- Alternate Flow:
  1a. Admin clicks ‘Forget Password’
    1a.1. Admin puts in email
    1a.2. Admin opens email link
    1a.3. Admin types in password two times
    1a.4. Webserver validates admin information
    1a.5. Web Server returns session key an id
    1a.6. Admin is redirected to the select database screen
    1a.7. Admin Selects ‘Midwest Radio’

6.3.8: Use Case 9: Add a Station

- Primary Actor: Admin
- Secondary Actor: Web Server
- Precondition: Successfully logged in with credentials
- Main Flow:
    1. Admin clicks ‘Add Station’
    2. Application displays a table to add station information
    3. Admin enters station information
    4. Admin clicks ‘Activate’ to activate station
    5. Web server adds station
- Alternate Flow:
    *None

6.3.9: Use Case 10: Create New User for Backend

- Primary Actor: Web Server
- Secondary Actor: Admin
- Precondition: Admin goes to website and logins with credentials
- Main Flow:
    1. Admin clicks on ‘Manage Users’
    2. Admin clicks on ‘Create New User’
    3. Admin enters in email and password
    4. Web server compares hash password and email to database to see if there is a user that matches it
    5. Web server hashes password
    6. Web server stores credentials
    7. Web server redirects admin to admin panel
- Alternate Flow:
    2a. Web server returns error when hash password and email doesn’t match database

6.3.10: Use Case 11: Load Application

- Primary Actor: Listener
- Secondary Actor: Web server
- Precondition: Application is installed on the phone
- Main Flow:
    1. Listener opens application
    2. Phone makes load request
    3. Application loads all data from the web server
    4. Application will load the radio screen

5. Pop up will appear letting the listener know that they can flip to landscape mode.

- Alternate Flow:
  2a. Application fails to load data from the web server
    2a.1. Application displays a no internet message

6.3.2: Use Case Sequence Diagrams

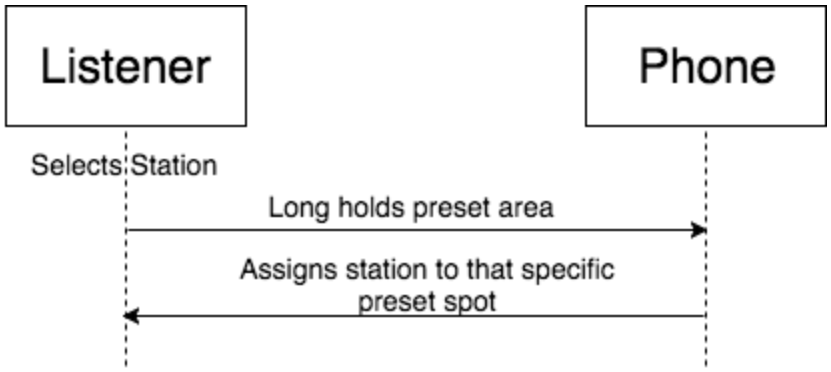


Figure 12: Sequence Diagram for 3.3.0: Assigning Presets

Assigning the presets happens after a listener has a station currently selected on their device. On the device, there are buttons 1 through 6 listed out. The user can long hold press any button and that will assign that current station to that specific preset spot.

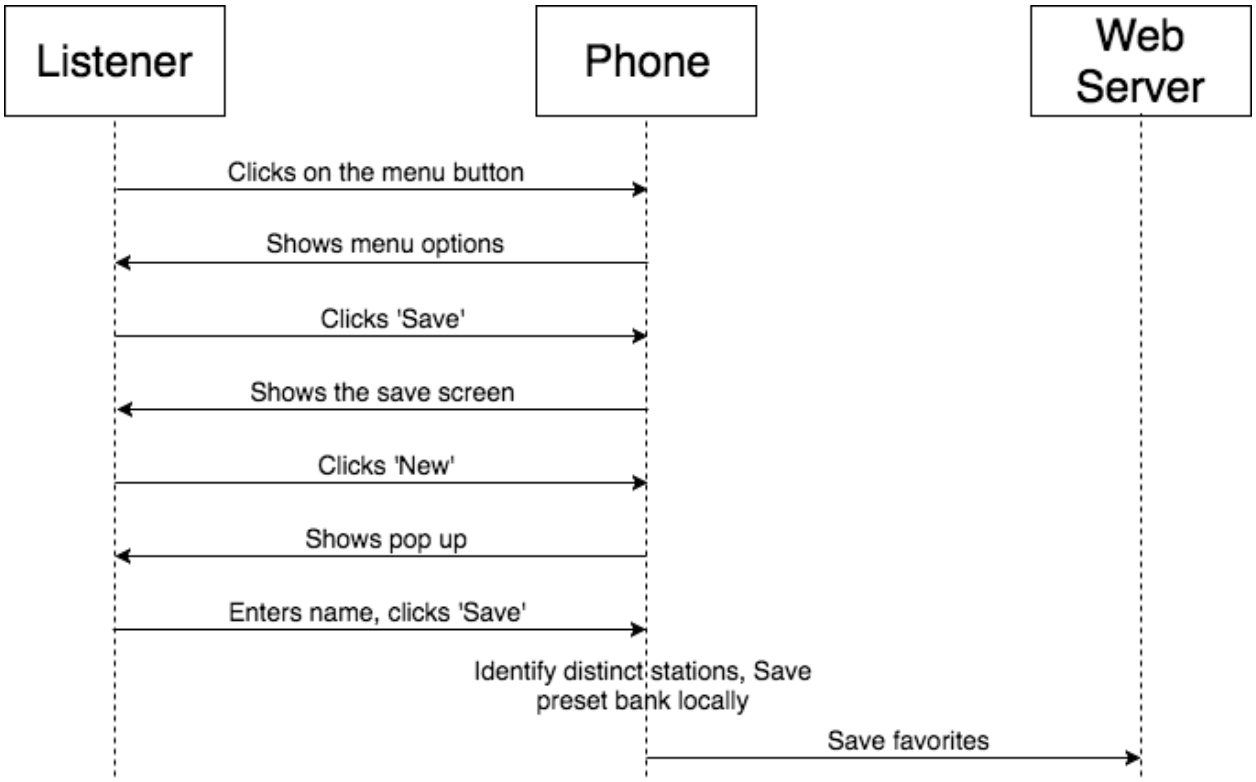


Figure 13: Sequence Diagram for 3.3.1: Save Preset Bank

Storing Preset Banks is storing the all the stations to the phone's memory to be pulled up later on. As previously mentioned above, users will have the ability to save a station to a preset spot. After a user has stored what they want, they can click the menu button. The listener will then click 'Save' and the lists of presets banks will appear. If the user clicks 'New', then a pop up will appear with a spot for entering a name. After they click save, the phone will save all the preset spots to that specific preset bank. The application will then determine which stations that were saved are new and are not in any other preset bank. After that, the application will make a network request to the webserver will those station ids.

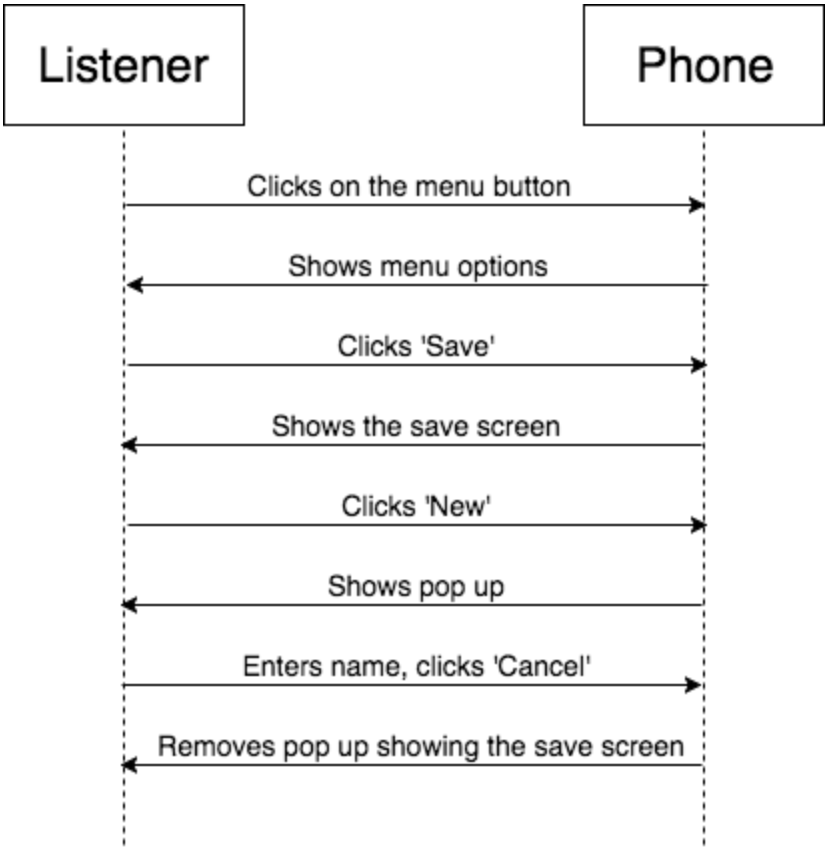


Figure 14: Sequence Diagram for 3.3.1: Save Preset Bank Alternate Flow

This is the alternative flow for saving a preset bank. All of the steps are the same, expect the user has the option to cancel their preset bank. The phone will cancel that process and go back to showing all of the preset banks.

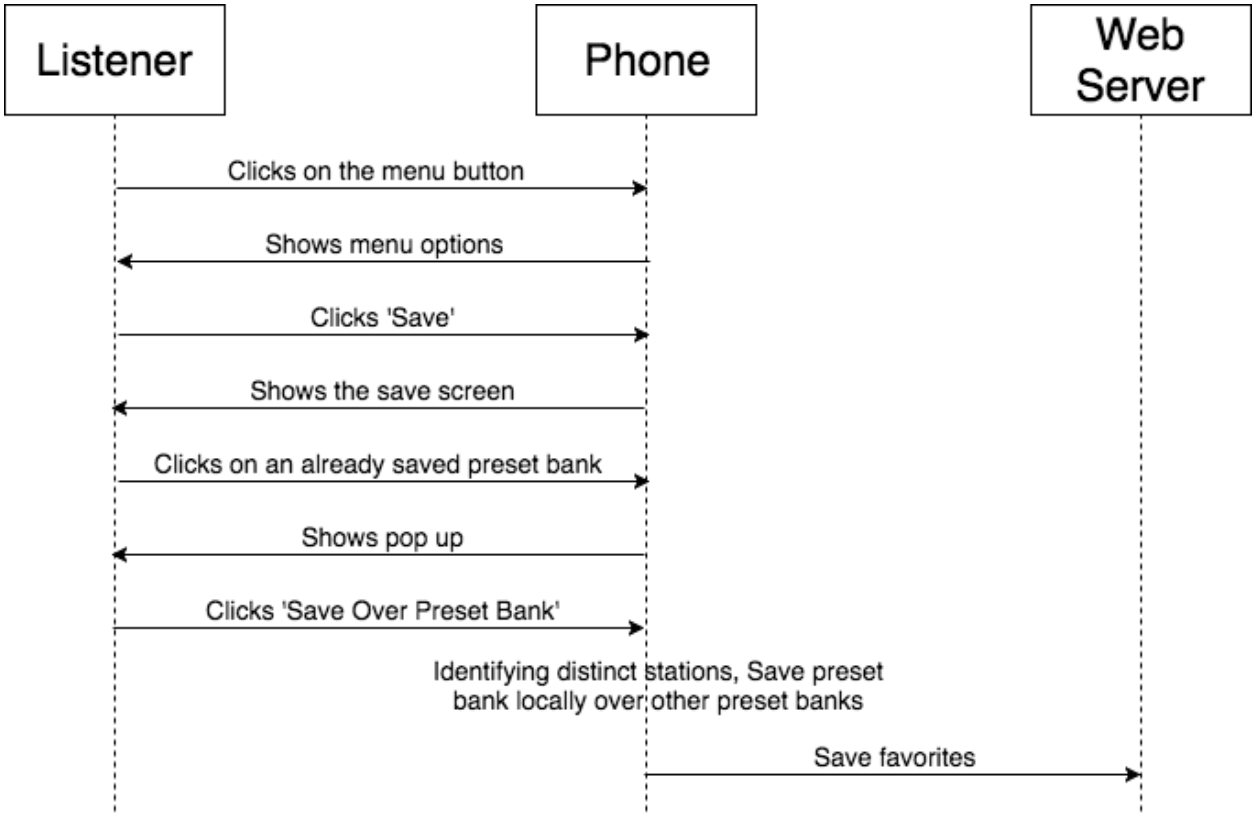


Figure 15: Sequence Diagram for 3.3.1: Save Preset Bank Alternate Flow

This is an alternative flow for saving the preset bank. Instead of a listener clicking on the ‘New’ button, the user can click any of preset banks that are listed to save over them. Again, the application will save that to the application’s memory plus will identify the unique stations and will send out the network request to the web server.

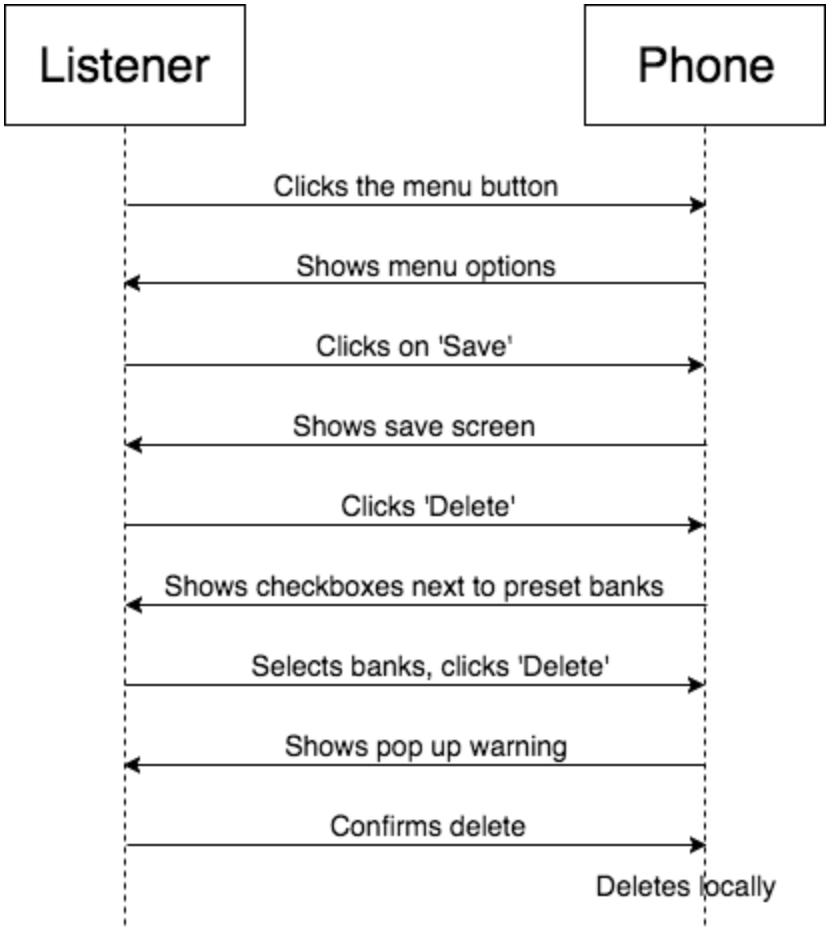


Figure 16: Sequence Diagram for 3.3.2: Delete Preset Bank

Once a user has stored a couple of preset banks to their phone, they will be able to delete them. The user will first click on the menu button, followed by the ‘save’ button to show all of the preset banks. The user can then press delete button which is located at the bottom of the screen. The application will then show checkboxes next to each of the preset banks. The listener can click on any or all of the preset banks and then press the ‘Delete’ button. The application will produce a warning message. Once the listener confirms again, the preset banks will be deleted from the Application’s memory.

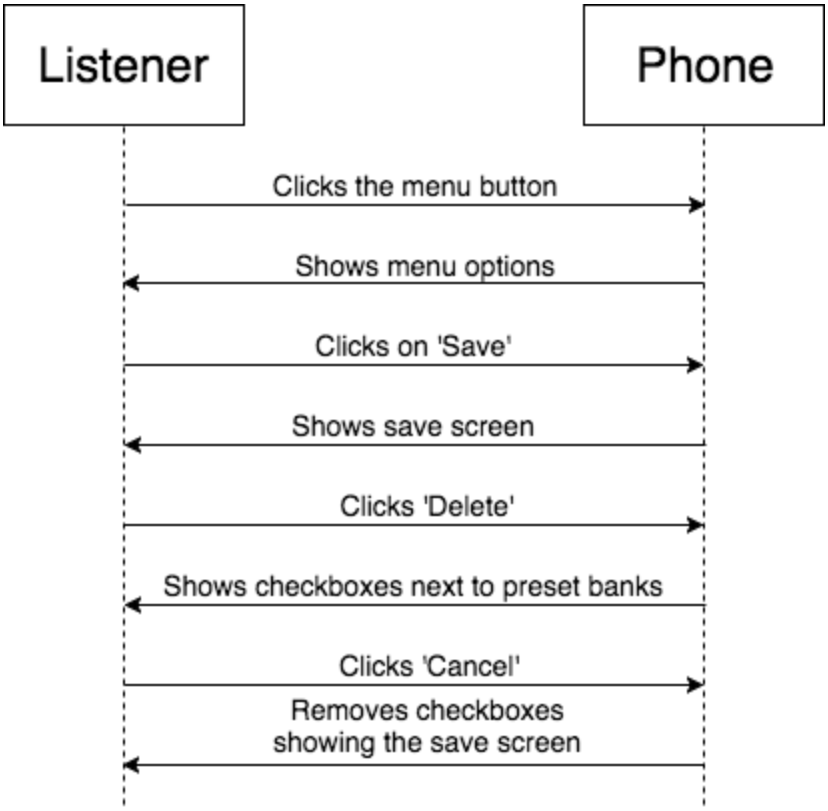


Figure 17: Sequence Diagram for 3.3.2: Delete Preset Bank Alternate Flow

Again, this will follow the same flow as the previous user diagram. After the check boxes are shown on the left hand side of the screen, and the user clicks on ‘Cancel’, then it will remove the checkboxes on the left side of the screen.

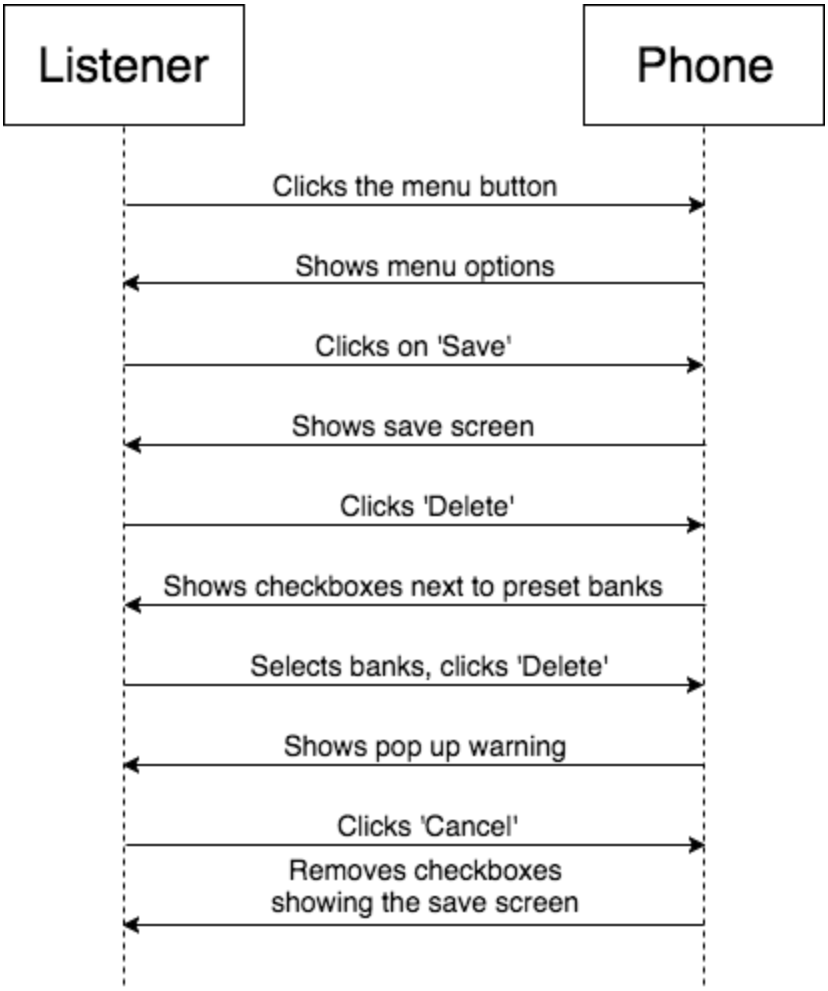


Figure 18: Sequence Diagram for 3.3.2: Delete Preset Bank Alternate Flow

Following the save steps as above, after the listener clicks on the delete button, the checkboxes will appear next to the preset banks. After the user has selected 1 or more, and then presses ‘Delete’, a warning message will appear. The user can click on cancel and it will remove the checkboxes showing on the save screen.

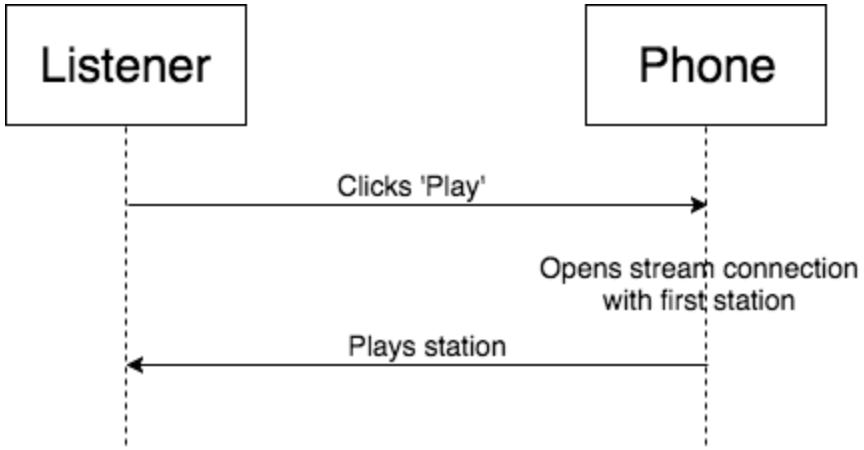


Figure 19: Sequence Diagram for 3.3.3: Listen to Station

Once the application has been loaded on the phone, the user can click on the ‘Play’ button on the selected station. The application will make many networks call out to that station’s respective stream and will try to connect. One the connection has been made, it will return the audio stream of the station.

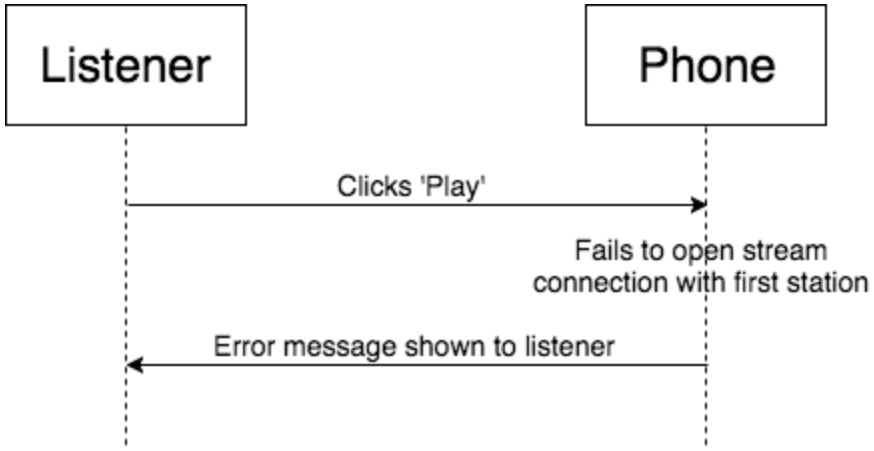


Figure 20: Sequence Diagram for 3.3.3: Listen to Station Alternate Flow

Same steps as the above user diagram. If there is an issue during the process of trying to connect to the station’s steam, the application will time out and display an error to the user explaining the issue.

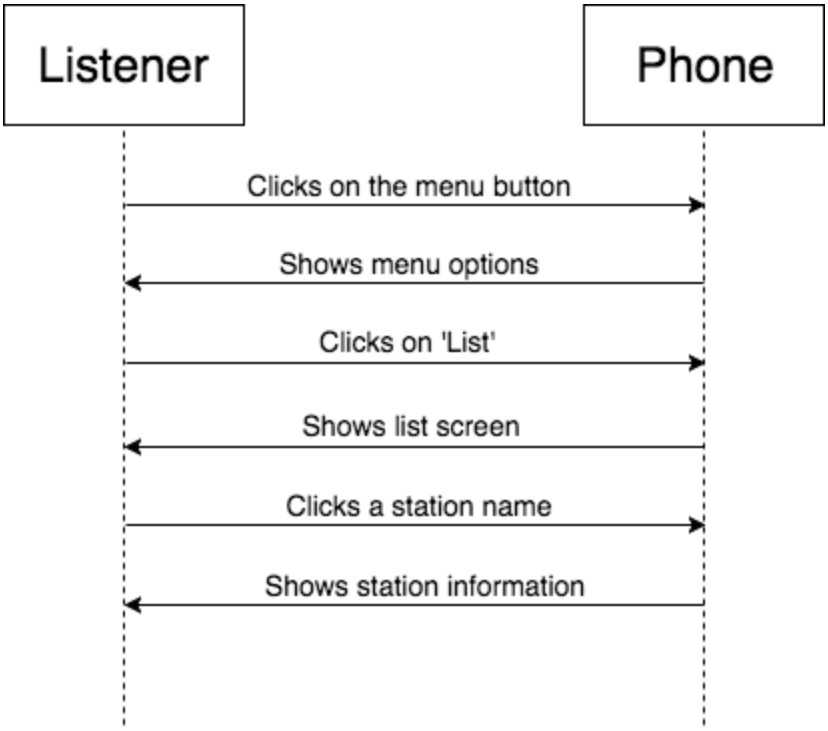


Figure 21: Sequence Diagram for 3.3.4: Find Stations

A user can find different stations that were loaded from the web server. The listener can click on the menu button, this will list out the menu options. Next, user will click on ‘List’. This will show the list of stations that have been loaded. The listener can then click on a specific station and that will show all the station information associated with it.

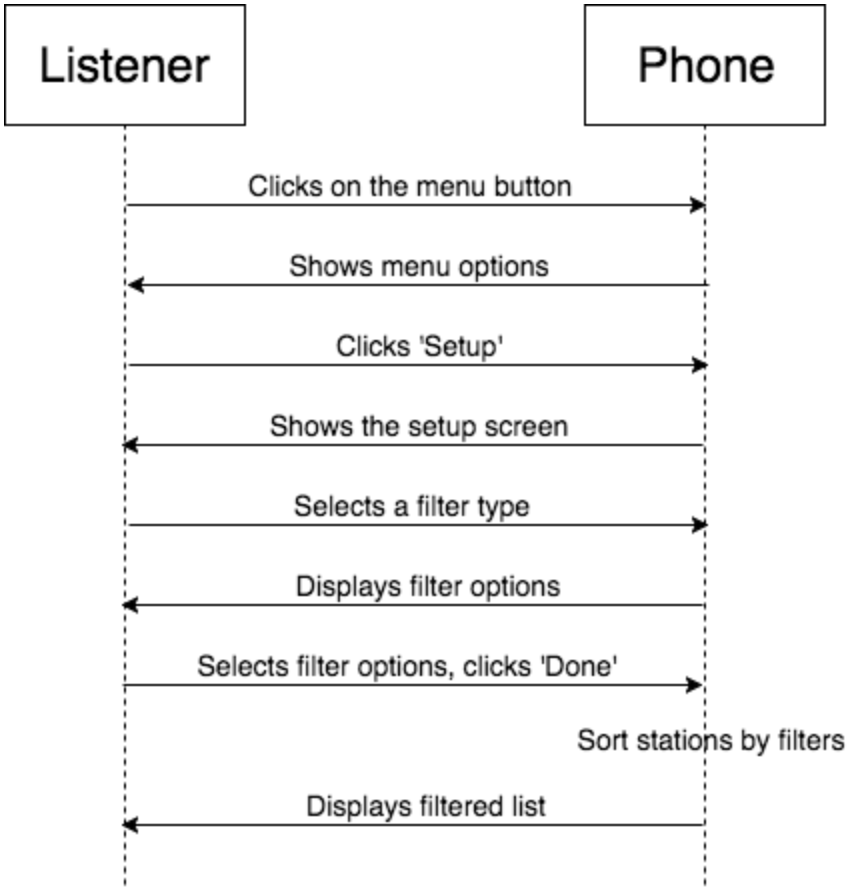


Figure 22: Sequence Diagram for 3.3.5: Filter Stations in Application

The listener can can create their own scan list that will go over the stations they they have an interest in. The user will first click on the menu button followed by the setup button from the menu options. The listener then click on the filter type of their choice and then it will show all the specific criteria associated with that filter. After the listener has chosen all they want, they can click done. The application will then sort all of the station that fit that criteria and display the list.

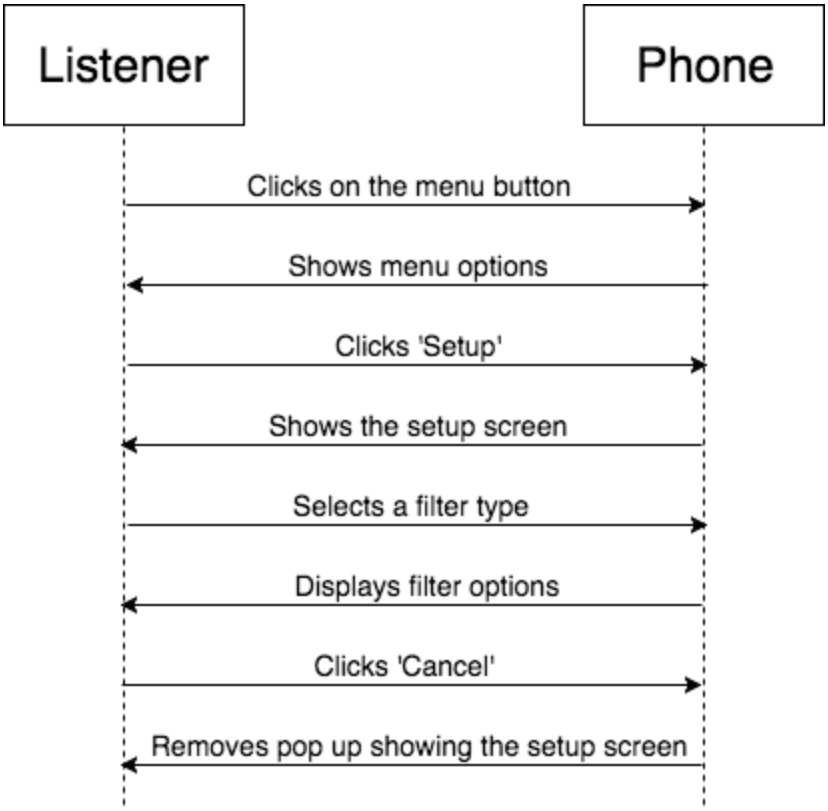


Figure 23: Sequence Diagram for 3.3.5: Filter Stations in Application Alternate Flow

This alternate flow will follow the same steps as the above use case. This diagram shows that during the picking of filters, the listener can click on the cancel button and that will remove the pop up showing the setup screen.

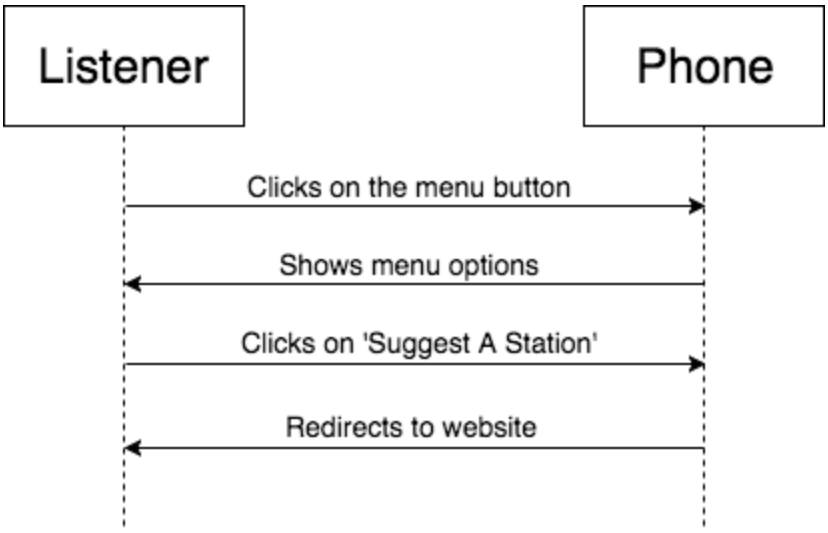


Figure 24: Sequence Diagram for 3.3.6: Direct to Submit a Station

One unique aspect to this application is that users can submit a station that could be added to the database in the near future. The user will first click on the menu button and will display all the menu options. The listener will then choose the ‘Suggest A Station.’ The application will redirect to a specific URL on the mobile web browser that the phone has.

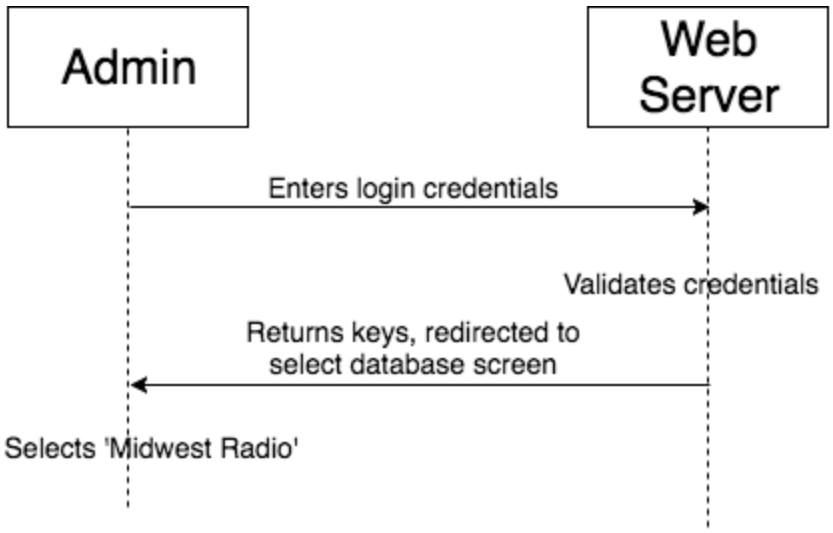


Figure 25: Sequence Diagram for 3.3.7: Login to Web Server / Website

The Admin is responsible for maintaining the database for these stations. It is critically that it is secure. The admin will go to a admin panel URL and put in their email address along with their password. This will make a network request to the web server. The server will validate this information and return the session keys and will be redirected to the select a database screen.

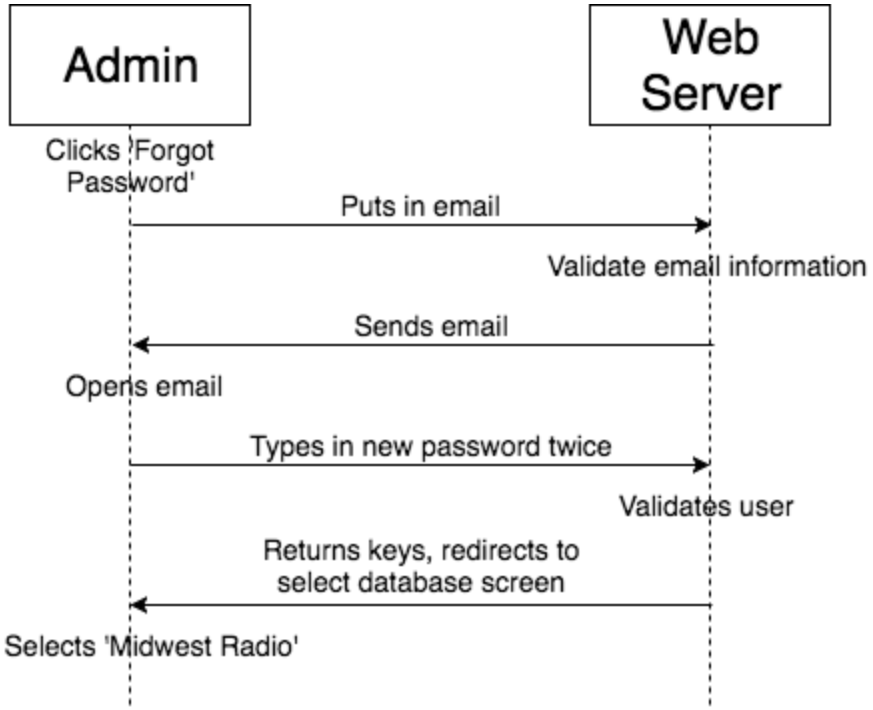


Figure 26: Sequence Diagram for 3.3.7: Login to Web Server / Website Alternate Flow

There is a Password request process for admin who may have forgot their password which is also an alternative flow for logging in. First, the admin will click on the “Forget Password” button and they will be redirected to a web page to insert their email. After the email has been entered, it is sent to the web server where it will verify that email address and will send an email to that person. The admin can then open it up and type their password twice to confirm it. After the user click ‘Submit’, it is sent to the web browser where the token is validated and the passwords will be updated in a hash form. This will also return session keys and ids to the admin. It will then redirect the user to the Database selection screen.

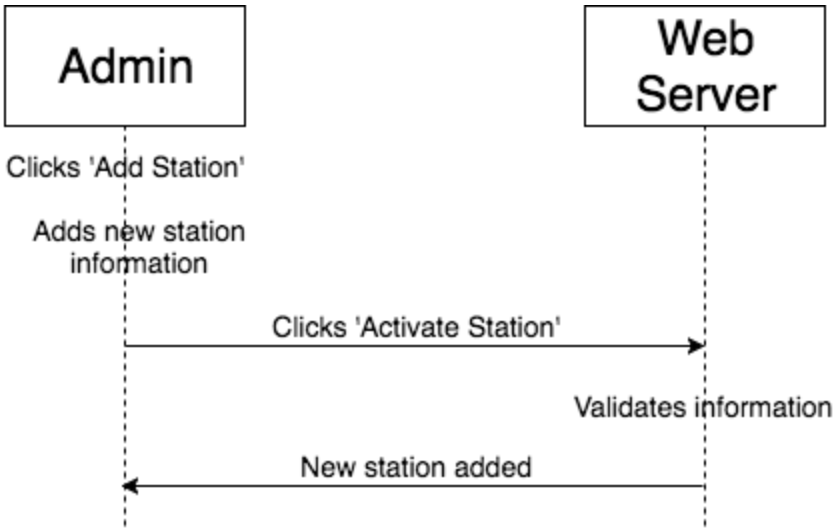


Figure 27: Sequence Diagram for 3.3.8: Add Station to Database

One of the features for an admin is adding stations to the database. After the admin has successfully logged in, they can click on ‘Add Station’ and a row in a table will appear. The admin then inserts all the stations information and clicks the activate station button. The web server will then validate that information and will add that station to the database.

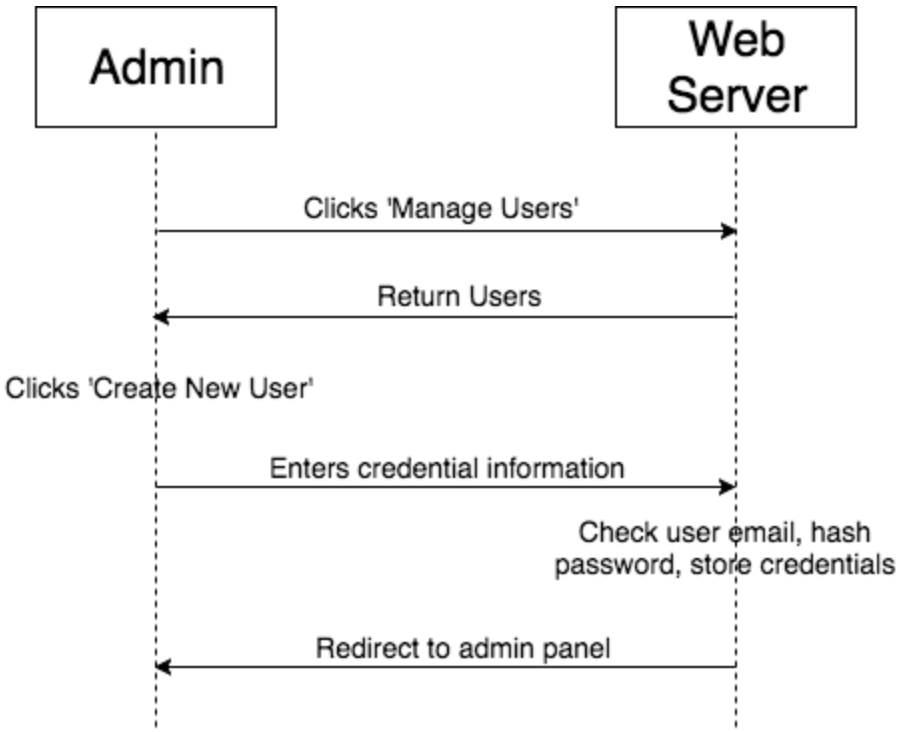


Figure 28: Sequence Diagram for 3.3.9: Create New User for Backend

A current admin will have the opportunity to add more admins to the site. First, the user will click on ‘Manage Users.’ This will return all the users that the database has. The admin can then click on ‘Create New User’ and a box will appear to add in their credential information. The server will collect this information and make sure that user has not already been created. It will then store a hash of the password and then store that information. After this has been completed, it will return the user back to the admin panel.

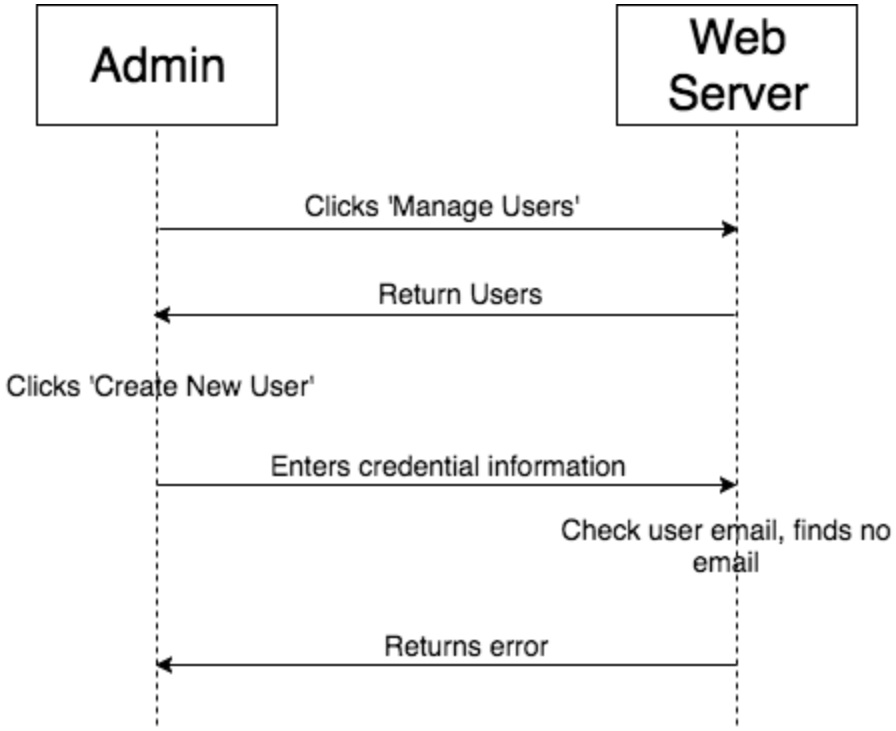


Figure 29: Sequence Diagram for 3.3.9: Create New User for Backend Alternate Flow

Following all the steps as the previous use case for 3.3.9. If the email that the current admin has entered matches an email that is already in the database, the web server will send back an error message to the user letting them know the user can not be inserted.

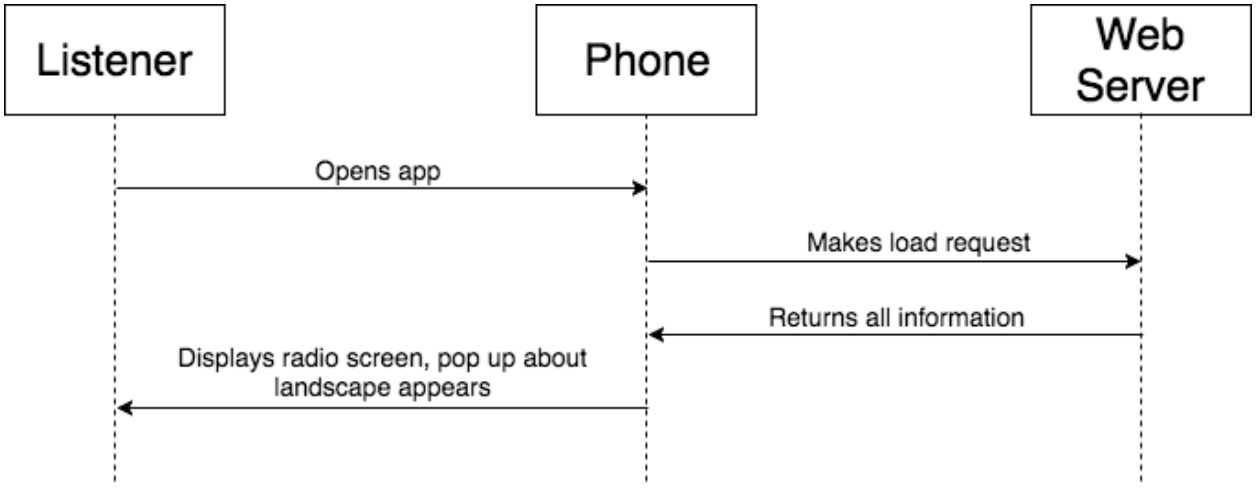


Figure 30: Sequence Diagram for 3.3.10: Load Application

The first step to the Listener completing any other use case first lies with getting the application running. Once the user has opened the application, the application will make a call to the web server to load in the Station Information. The web server will return a large object containing all

the information. The Application will process it and display the radio screen and will come with a pop up letting the user know that they can turn their device sideways to view in landscape.

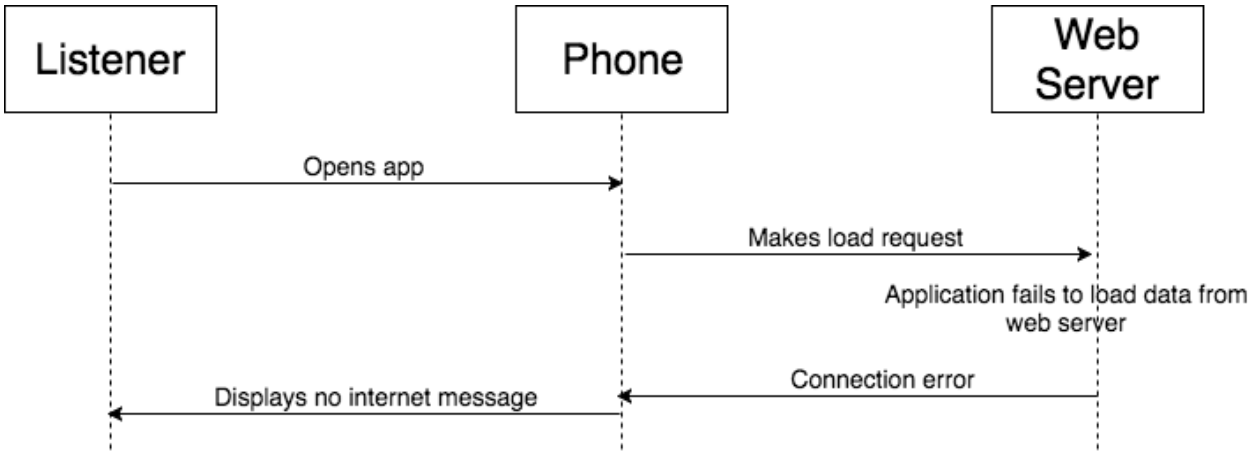


Figure 31: Sequence Diagram for 3.3.10: Load Application Alternate Flow

Following the same steps as above use case 3.3.10. If the application loads, it will make the request to the web server. The web server might be down or the phone might not be connected to Internet. Therefore, the application will still display an error message to the user letting them know the problem.

6.4: Domain Modeling

6.4.1: Entity Relationships

	Admin	Client	Station	Preset Banks	Categories
Admin			Manages		
Client			Listens to	Manages	selects
Station	Managed by	Listened by		Stored in	Filtered
Preset Banks		Managed by	Stores		
Categories		Selected by	Filtered By		

Figure 32: Entity Relationship Diagram

The above shows how the entities are connected to one another. It can be read “Row title” (is) “cell action” “column title”, e.g. *Station is managed by Admin, Preset Banks stores Station,* etc.

6.4.2: Class Diagram

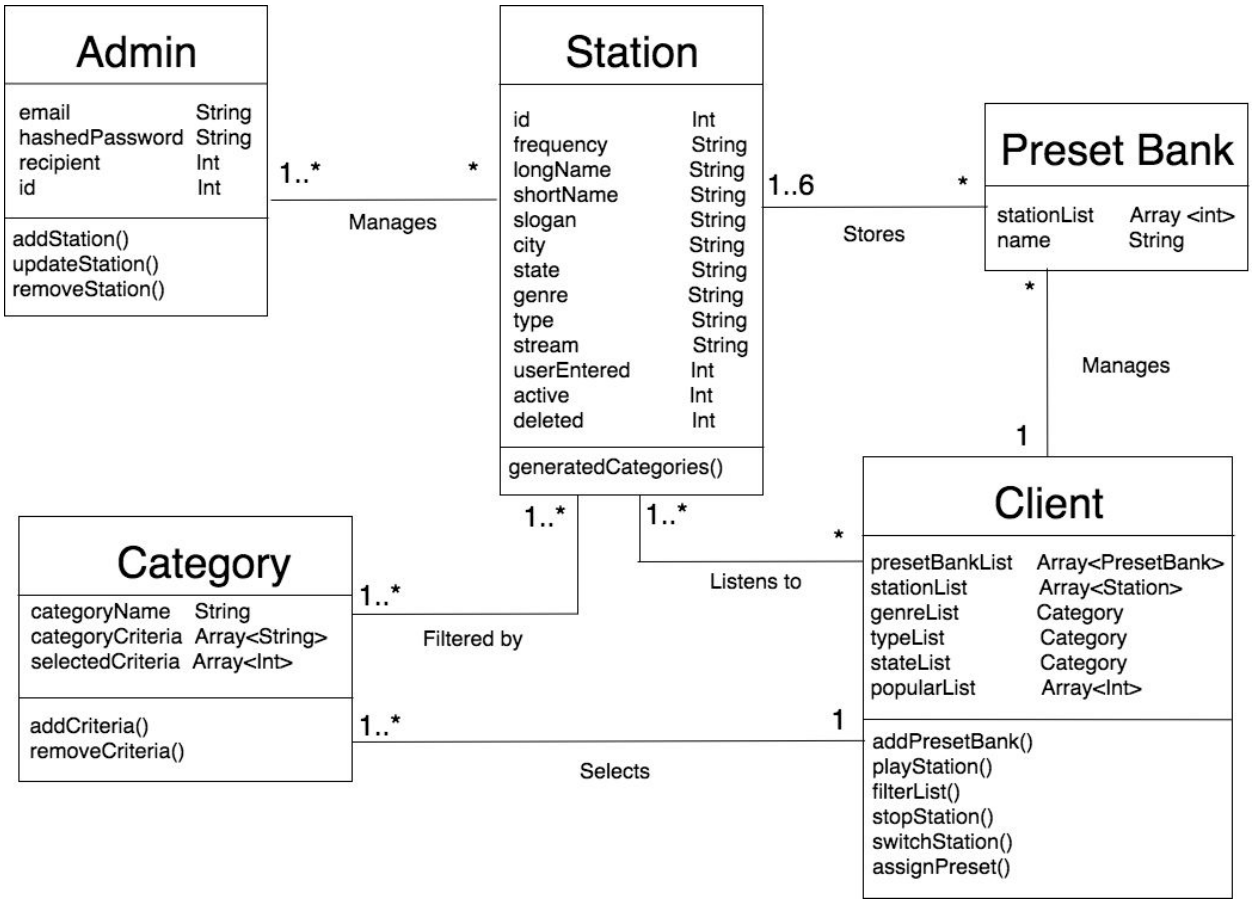


Figure 33: UML Class Diagram

This UML Class Diagram explains the relationships between each of the major roles in our application. For example the admin and stations relationship, you can see that one to many admin can manage many stations. Another example could be the relationship between category and station. One to many stations are filtered by one to many categories.

6.5: Traceability

6.5.1: Case Requirements Matrix

	1	2	3	4	5	6	7	8	9	10	11	Later Iteration
1.1.1.1		x										
1.1.1.1.1	x											
1.1.1.1.2												x
1.1.1.1.3		x										
1.1.1.1.4						x						
1.1.1.1.5												x
1.1.1.1.6					x							
1.1.2.1											x	
1.1.2.2												x
1.1.2.2.1												x
1.1.2.2.2												x
1.1.2.3												x
1.1.2.4												x
1.1.2.5												x
1.1.2.6												x
1.1.2.7												x
1.1.2.8	x											
1.1.2.9	x											
1.1.2.10												x
1.1.3.1												x
1.1.3.2												x
1.1.3.2.1												x
1.1.3.3			x									
1.1.3.3.1			x									
1.1.3.3.1.1			x									
1.1.3.3.1.2			x									

1.1.3.4												x
1.1.4.1		x										
1.1.4.2			x									
1.1.4.2.1			x									
1.1.4.2.1.1			x									
1.1.4.2.1.2			x									
1.1.4.2												x
1.1.5.1		x										
1.1.6.1						x						
1.1.6.1.1						x						
1.1.6.1.2						x						
1.1.6.1.3						x						
1.1.6.2						x						
1.1.6.2.1						x						
1.1.6.2.2						x						
1.1.6.2.3						x						
1.1.6.3						x						
1.1.6.3.1						x						
1.1.6.3.2						x						
1.1.6.3.3						x						
1.1.6.4						x						
1.1.6.4.1						x						
1.1.6.4.2						x						
1.1.6.4.3						x						
1.1.6.5						x						
1.1.6.6						x						
1.1.7.1												x
1.1.7.2												x
1.1.7.3												x
1.1.7.4												x

1.1.8.1							x				
1.1.8.2							x				
1.1.8.3											x
1.1.8.4											x
1.1.8.5											x
1.1.8.6											x
1.1.9.1											x
1.1.9.2							x				
1.1.9.3											x
1.1.10.1											x
1.1.10.1.1											x
1.1.10.1.2											x

Figure 34: Mapping of use cases to requirements

Above, you can see how the use cases and requirements match up. Some of the requirements match up with the column titled ‘later iteration’. This is because our project has a great amount of use cases, so in order to keep milestone to a reasonable amount we decided to show some of the more important use cases. The other use cases have been created, just not shown here. If we had all of our use cases wrote out here, then all of the requirements would be mapped up to a use case.

6.5.2: Requirements - Dependency Matrix

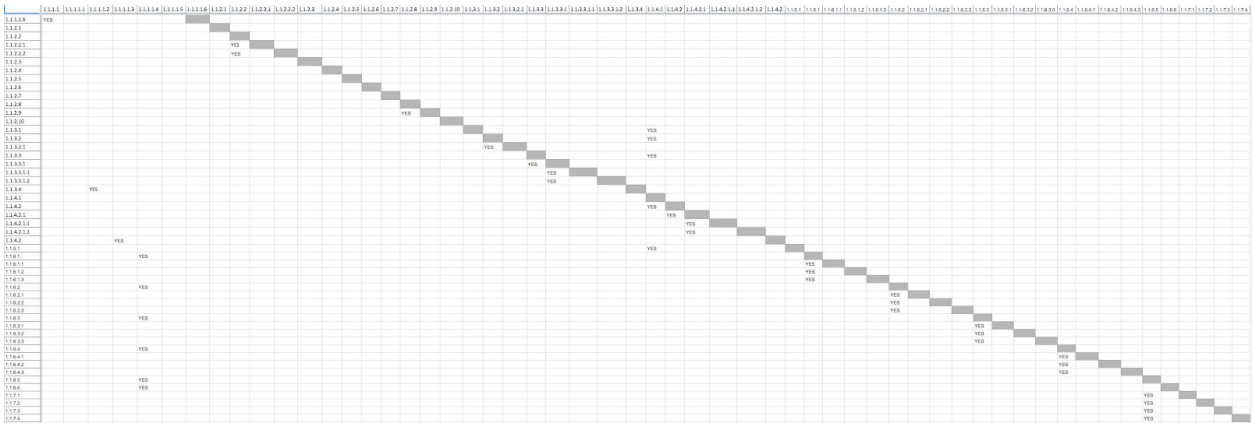


Figure 35: Requirements Dependency Matrix for Application

	1.1.8.1	1.1.8.2	1.1.8.3	1.1.8.4	1.1.8.5	1.1.8.6	1.1.9.1	1.1.9.2	1.1.9.3	1.1.10.1	1.1.10.1.1	1.1.10.1.2
1.1.8.1												
1.1.8.2	YES											
1.1.8.3												
1.1.8.4			YES									
1.1.8.5												
1.1.8.6												
1.1.9.1												
1.1.9.2												
1.1.9.3							YES					
1.1.10.1												
1.1.10.1.1										YES		
1.1.10.1.2										YES		

Figure 36: Requirements Dependency Matrix for Admin Panel

For Figure 35 and Figure 36, we show how each requirement depends on another requirement being completed first. For example in Figure 36, you can see 1.1.8.2 depends on 1.1.8.1 being completed first. That is because the admin will have to be prompted to login to get into the admin panel before they are sent to the home screen where they can select the Midwest database.

7.0: Technical Summary

For this project, our teams goal is to produce the admin panel, and two mobile applications for both iOS and Android.

7.1: iOS Considerations

According to our Non Functional Requirement of 1.2.1, iOS will be able to run at iOS 9. The current version of iOS is currently at 11.1.2. However, looking at the breakdown of current configurations on devices, iOS 9.0 is still used in nearly 8% of all current iPhones and iPads.

Currently, we are developing the iOS Application in the latest version of XCode. This IDE allows us to test the phones on different devices including iPhones and iPads as well as testing out different versions of the OS.

7.2: Android Considerations.

Similar to above, according to our Non Functional Requirement 1.2.1, Android will be able to run at Android 4.4. When looking at the breakdown for the Android system, there is currently 14.5% of users that are using this version.

The current IDE of Android also allows us to test our project on different phones as well as different operating system. It is important to note that we are using a specific software

packages that must meet API 16 and above in order for it to work. By staying at API 19 with Android 4.4, we have covered this requirement

7.2: Conclusion for Milestone #2

In conclusion, we have not made any major changes to our requirements, but we have changed our development language. We began developing in React Native, which would have allowed us to build apps for both platforms with only one code base. However, because of the trouble we had with testing capabilities and building proof of concept, we decided to instead build two separate native applications. Even so, we are still on track to meet or exceed all of our goals for the application, backend, admin panel, and public facing site for the project. For this milestone, we have completed a use case diagram, dependency matrix, traceability matrix, and many more diagrams. With all this work, we are confident in the background of our project and how it shall all fit together. This has helped escalate our project progress forward. In meeting with the client, he has expressed his excitement about the project and its possibilities.

8.0: User Interface Design

8.1 Mobile Applications Designs

Figure 37

Here you can see the navigation through the pages of the app. It opens to the radio screen. The menu can be opened from the icon in the top right hand corner. In the menu you can navigate to the list screen, which lists all the available stations, the radio screen, or be redirected to the Middletown Music website on the phones browser. The only difference between iOS and

Android is that Android lacks the back button in the top left hand corner of the screen since this functionality is already built in to the phone.

Figure 38

Here you see the same navigation as in Figure 2, but in the landscape view.

8.2 Admin Panel Designs

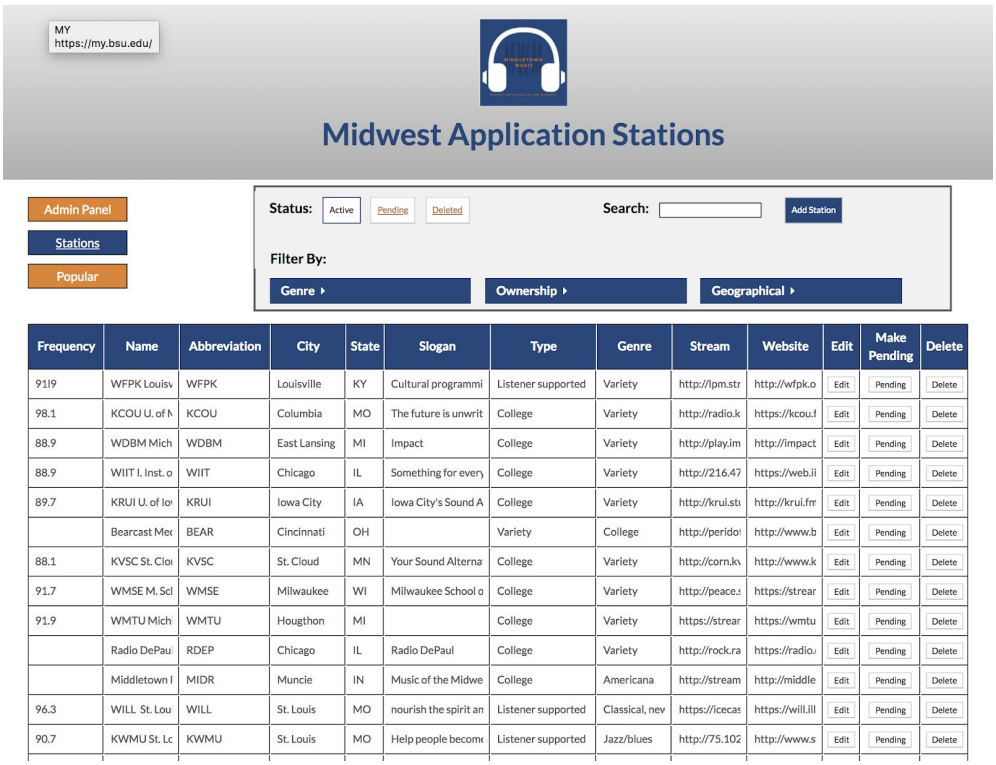
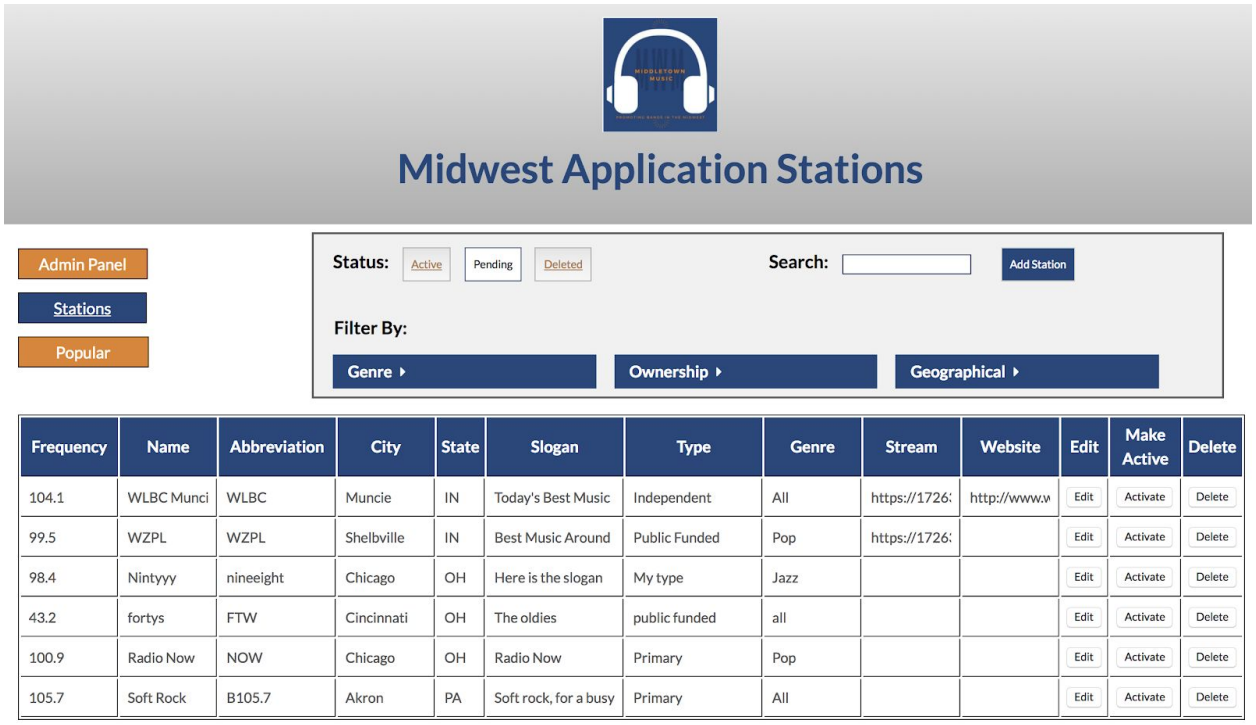


Figure 39

This is the admin view for all the current active stations on the site. If you click on any of the table headings, it will sort the table in numerical or alphabetical order by that column. You can also click edit to edit the row, click pending to make the station go to pending, or click delete to delete the station from the app.



Frequency	Name	Abbreviation	City	State	Slogan	Type	Genre	Stream	Website	Edit	Make Active	Delete
104.1	WLBC Munci	WLBC	Muncie	IN	Today's Best Music	Independent	All	https://1726:	http://www.w	Edit	Activate	Delete
99.5	WZPL	WZPL	Shelbville	IN	Best Music Around	Public Funded	Pop	https://1726:		Edit	Activate	Delete
98.4	Nintyyy	nineeight	Chicago	OH	Here is the slogan	My type	Jazz			Edit	Activate	Delete
43.2	fortys	FTW	Cincinnati	OH	The oldies	public funded	all			Edit	Activate	Delete
100.9	Radio Now	NOW	Chicago	OH	Radio Now	Primary	Pop			Edit	Activate	Delete
105.7	Soft Rock	B105.7	Akron	PA	Soft rock, for a busy	Primary	All			Edit	Activate	Delete

**If station has a blue background, that means a user submitted a station to be added to the radio.

Figure 40

Here is the admin view for all the pending stations on the site. If you click on any of the table headings, it will sort the table in numerical or alphabetical order by that column. You can also click edit to edit the row, click activate to make the station go to active, or click delete to delete the station from the app.

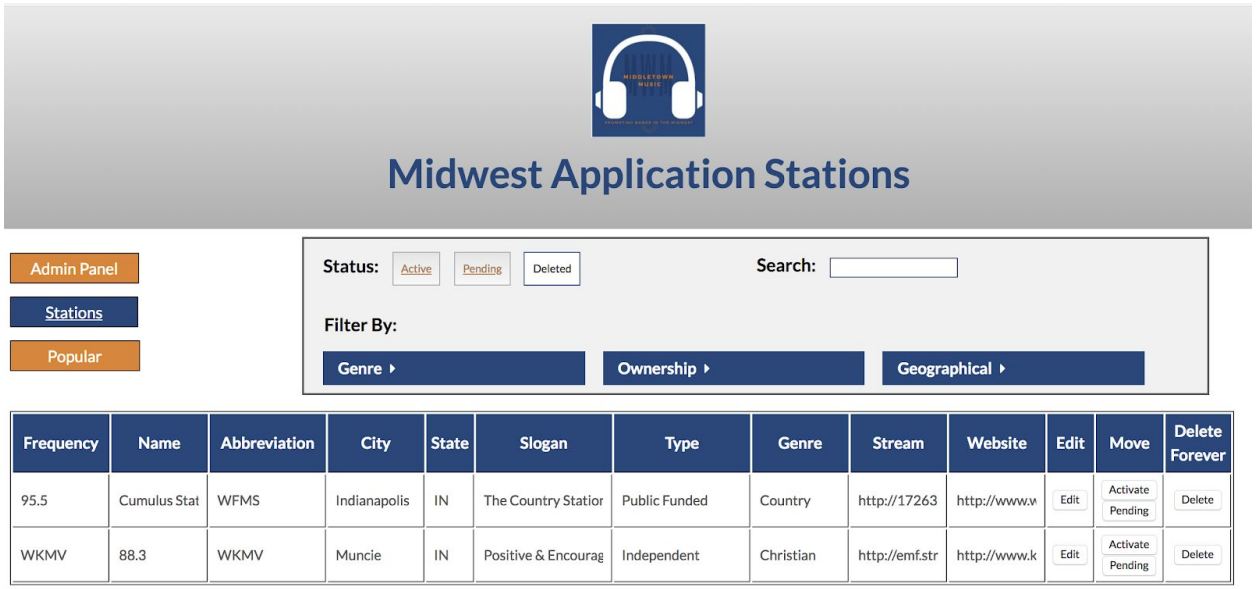


Figure 41

This shows the admin view for all the current deleted stations on the site. If you click on any of the table headings, it will sort the table in numerical or alphabetical order by that column. You can also click edit to edit the row, click pending or activate in the move column to make the station go to pending or active, or click delete to delete the station from the app forever.

9.0: Database Design

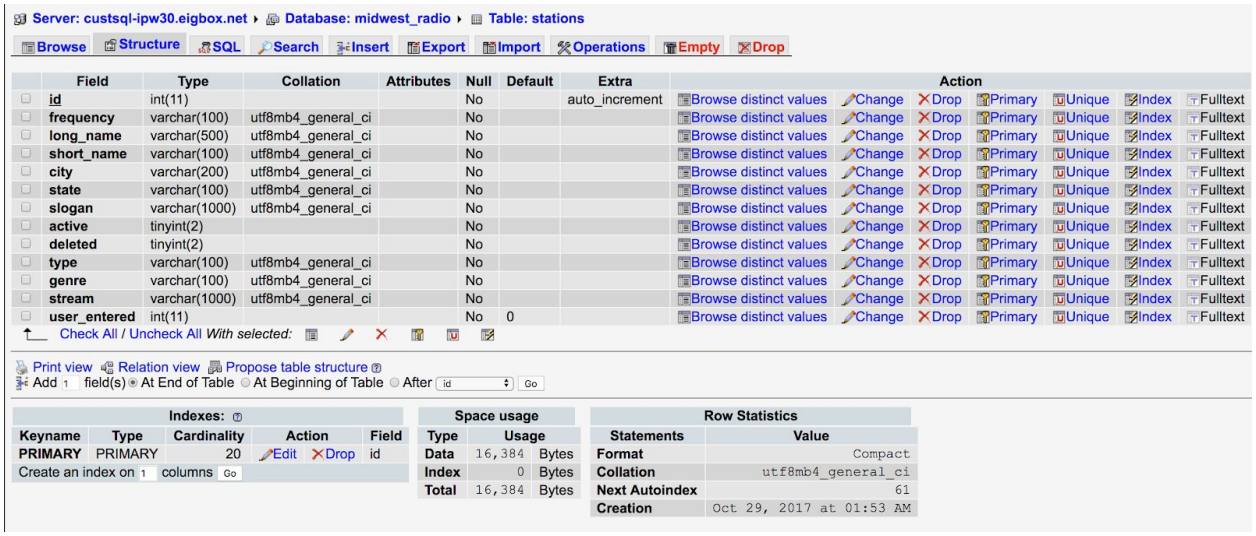


Figure 42

Structure View for the radio stations. It contains all the information on the stations, including their unique ids, radio frequency, full station name, abbreviated station name, location, slogan,

active/pending/deleted status, type of station (college, commercial, etc.), genre, streaming url, and whether it was user entered or not.

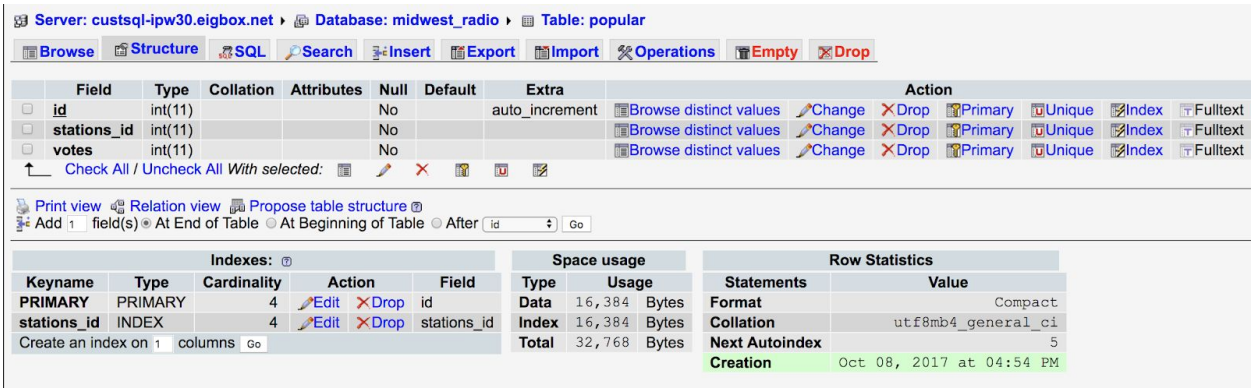


Figure 43

Structure View for the popular votes table. This stores the station id and the number of times it has been saved to a preset by unique users.

10.0: Software Architecture

10.1: Architecture Overview

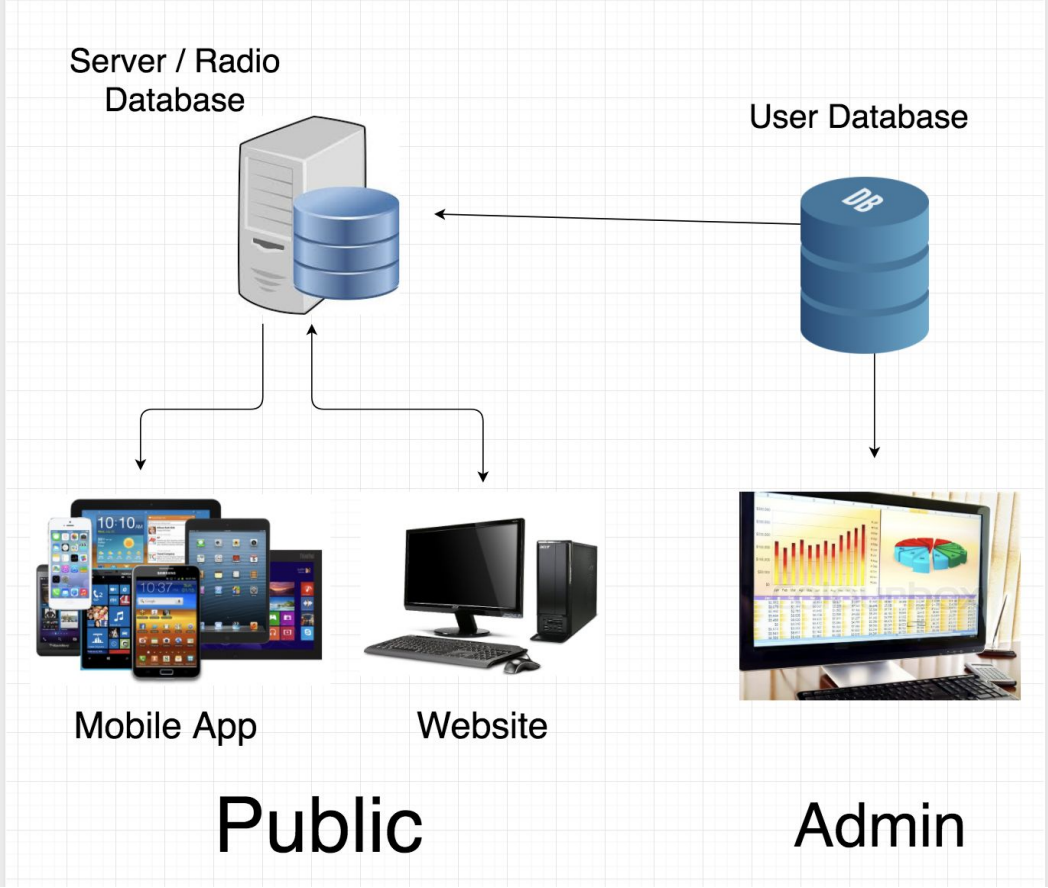


Figure 44

This figure shows a high level view of how each our components fit together. Below is an in depth explanation of each smaller component.

10.1.1: Mobile Applications

The iOS and Android apps are different code bases, but are basically the same in both appearance and functionality. The iOS app will function on iPhones operating on version 10.0 and above, and the Android app will function on phones operating on version 4.4 and above. They both interact with the web server through the backend by requesting all the information about the stations on launch. When a user saves a new station to their presets, they send a request to add a popular vote to that station. They will also redirect users to the submission and report online forms through the menu.

10.1.2: Web Server

The webserver is the backend to all three sides of our product: mobile applications, admin panel, and submission & report form. The web server handles all the functionality of adding, editing, and deleting stations from the database. There are specific Application Program Interface (API) for each specific product.

10.1.3: Admin Panel

The admin panel is a website where users with login credentials can add, edit, delete, and update the stations that are stored on the database for the applications. The data will then get sent to the web server where it will handle all actions. An admin user will also be able to see the popular stations that are stored in the database.

10.1.4: Submission & Report Forms

The submission and report forms are forms where users of the mobile apps can go to submit a station they would like to see on the mobile apps or report an issue with a station already on the applications. The information submitted through these forms gets sent to the admin panel where a user with login credentials can view the submissions.

10.2: Detailed Design

10.2.1: Backend

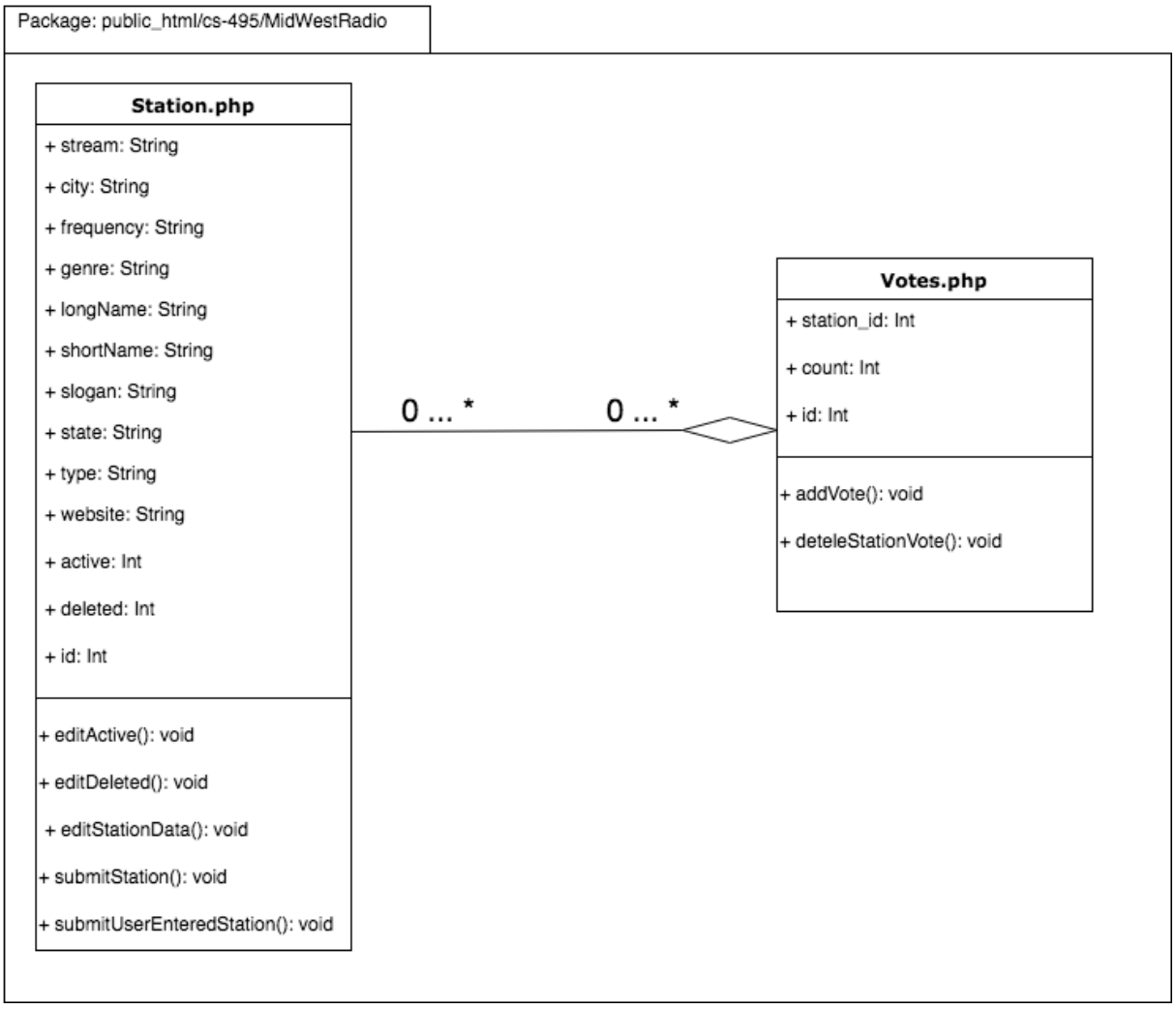


Figure 45

This figure is the backend class diagram. It is very simple with just a Station and Votes Object.

- Station: This class responsibility is to manage all the information for each station. There are alot of different attributes that are associated with including stream, frequency, city, etc.
- Votes: This class responsibility is to keep track of each stations amount of votes that the user saves as a preset on the phone.

10.2.2: iOS App

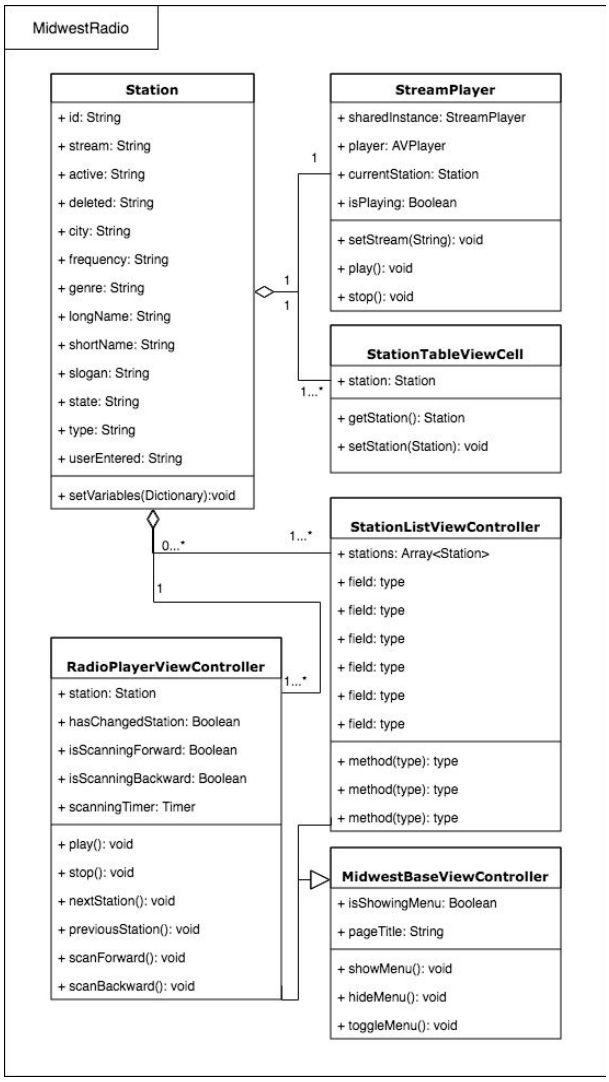


Figure 46

Here you see the class diagram within its containing package for the iOS application. In this case, because our apps are in different code bases, their class diagrams are different. Android is listed below.

- Station: This class is responsible for holding all of a station's data.
- StreamPlayer: This class is responsible for playing and controlling the radio stream.
- StationTableViewCell: This class is responsible for the display of each station in the database and for sending data about that station to the player.
- StaitonListViewController: This class is responsible for displaying the list of Stations as StationTableViewCells

- MidwestBaseViewController: This class is responsible for displaying and controlling the menu
- RadioPlayerViewController: This class is responsible for the GUI of the radio and sending the user's interactions to the player

10.2.3: Android App

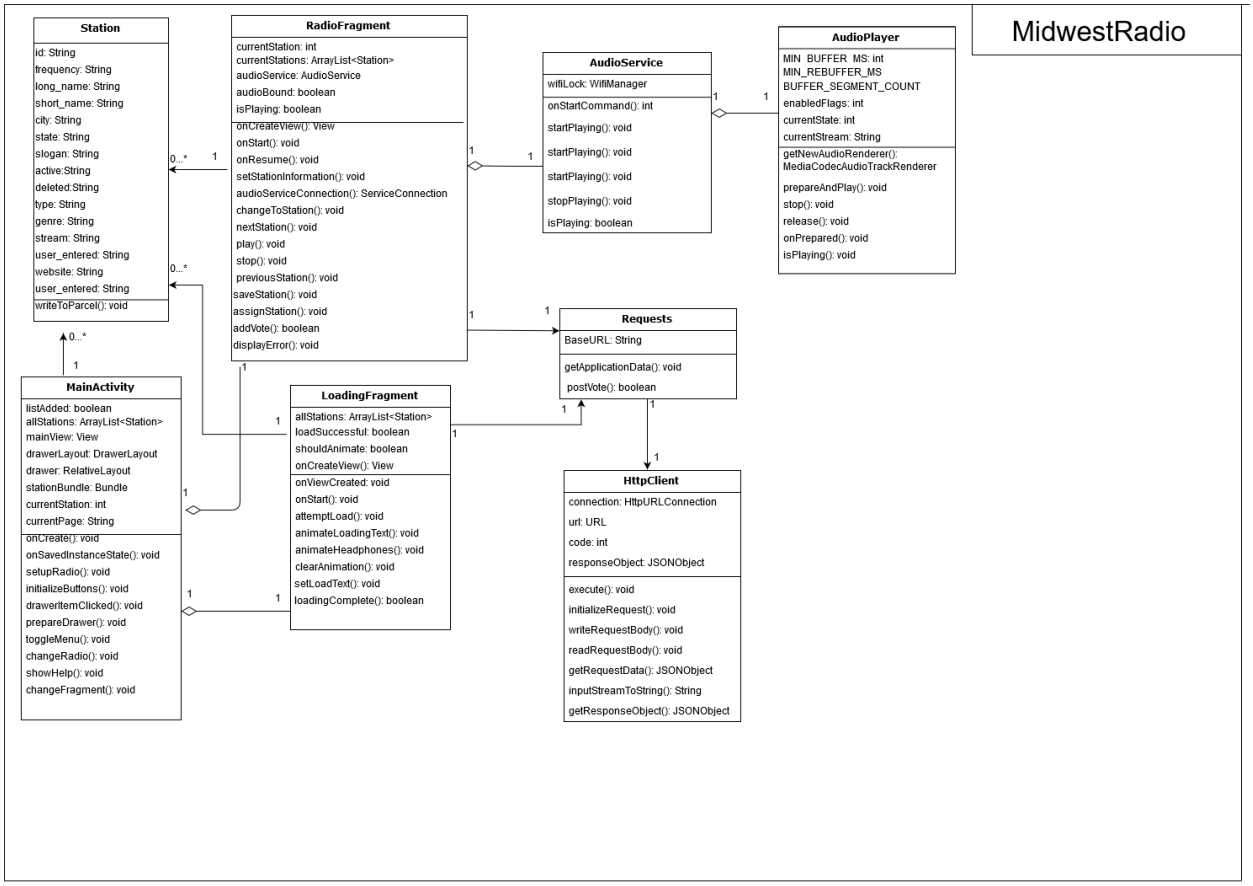


Figure 47

Here you see the class diagram within its containing package for the Android application, which, as noted above, is different from that of the iOS application because of the difference in code bases.

- Station: This Class is responsible for holding all the station data that is needed for the application.
- MainActivity: The MainActivity class is responsible for all the fragments, and facilitates the switching of them. It also contains the menu, which is a DrawerLayout.
- RadioFragment: This class facilitates the radio page, and all its associated functionality. It additionally binds an AudioService to play audio on the phone.

- LoadingFragment: The LoadingFragment is responsible for loading the application data on app load, and also animates the loading animation while waiting. It additionally gives the user feedback if no internet is available.
- HttpClient: The HttpClient is in charge of making all the various http requests that the application will need. It also contains a JSONObject to return with requested data.
- Requests: This object is used to make http requests for the application, utilizing the HTTPClient. It stores the url needed, and invokes the various HttpCilent methods as needed.
- AudioPlayer: The AudioPlayer class handles the various ExoMediaPlayer methods in order to control audio. It additionally will store the current Stream and the current state of the player.
- AudioService: This class acts as a control to the AudioPlayer, and creates a wifiLock to keep the stream alive.

10.2.4 Web Admin Front End

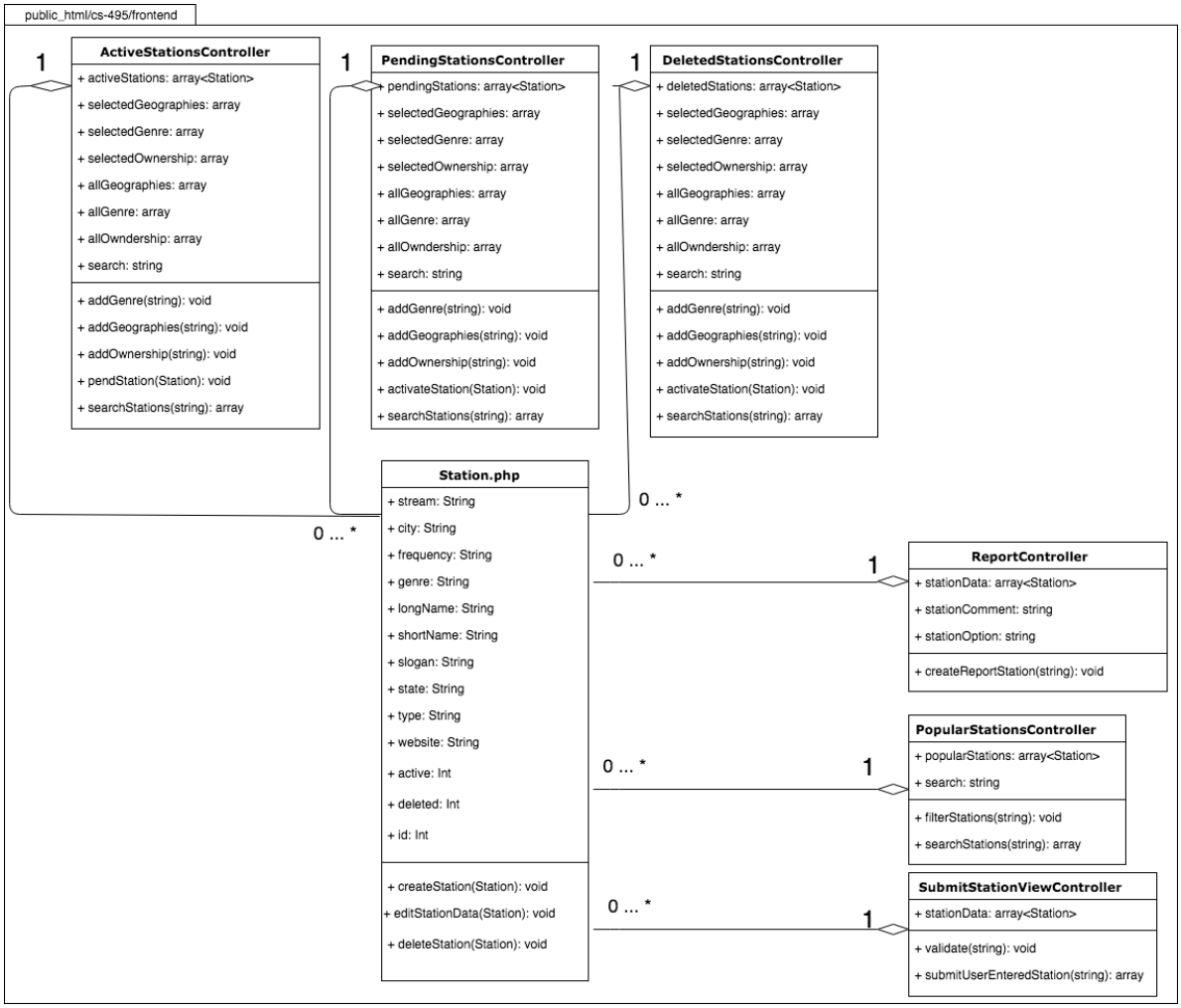


Figure 48

This is the class diagram for the front end of the admin panel. The main object is the Station and then it connects to each of them with the View Controllers.

- ActiveStationsController: This controller controls the displaying and filtering of active stations.
- PendingStationsController: This controller controls the functionality for pending stations, and also the filtering functionality. The user has the functionality to change to active or deleted.
- DeletedStationsController: This controller manages the displaying of deleted stations. It can control if he wants to move a station to pending or deleted.
- Station: This class is responsible for holding all the station information.
- ReportController: The ReportController handles the submission of report forms for stations that have issues.
- PopularStationsController: This class is responsible for keeping track of the popular count in the database. It also has additional validation checking to ensure valid streams are added.
- SubmitStationViewController: The SubmitStationViewController handles the form data for submitting a new station.

10.3: Detailed Major Use Cases

10.3.1: Mobile Application Use Cases

10.3.1.1: Use Case 1: Assign Presets

- Primary Actor: Listener
- Secondary Actor: Phone
- Precondition: Listener has installed and opened the application radio successfully or selected one of their saved preset banks
- Main Flow:
    1. Listener selects to a station
    2. Listener long presses one of the spots in the preset area to assign station
- Alternate Flow:
  *None

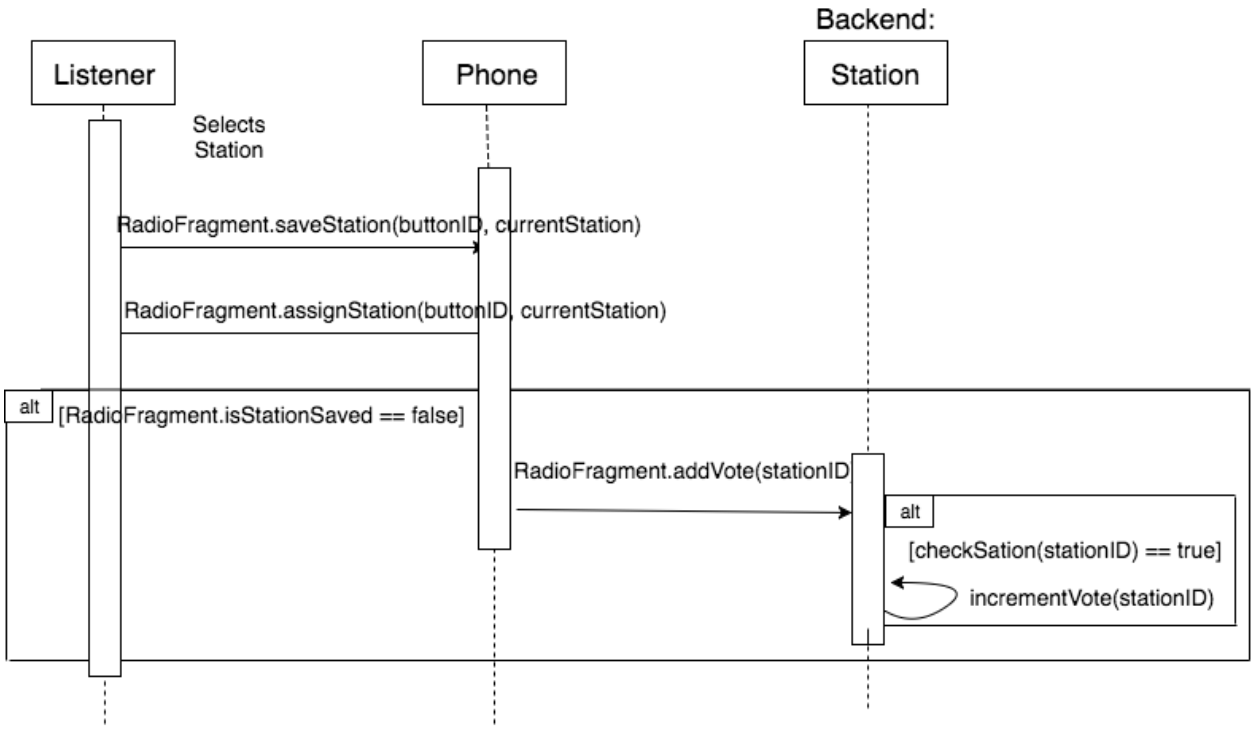


Figure 49

Use case diagram for 5.3.1.1: Use Case 1: Assign Presets

10.3.1.2: Use Case 2: Listen to Station

- Primary Actor: Listener
- Secondary Actor: Phone
- Precondition: Listener has installed and opened the application successfully
- Main Flow:
    1. Listener clicks the play button on the radio screen
    2. First station in the list will begin to playing
- Alternate Flow:
    2a. Listener lost connection to internet and station didn’t begin to play

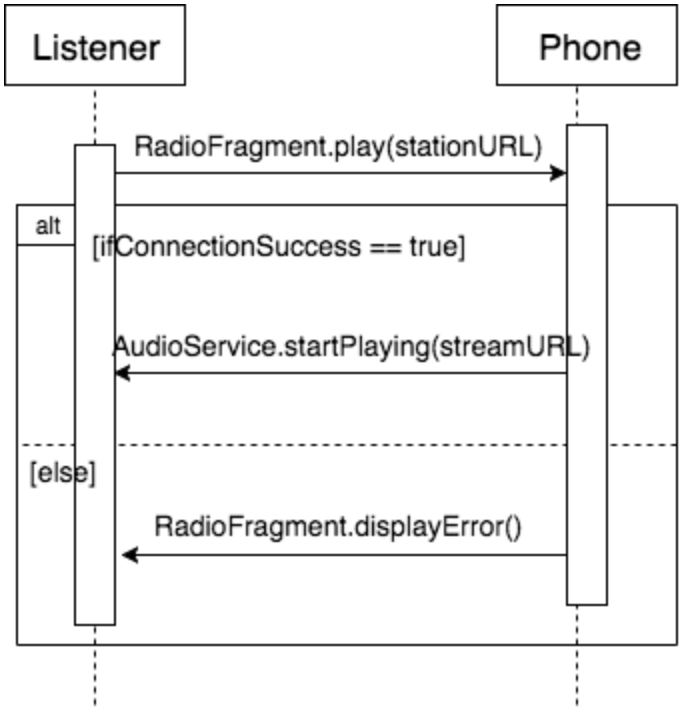


Figure 50

Use case diagram for 5.3.1.2: Use Case 2: Listen to Station

10.3.1.3: Use Case 3: Find Stations

- Primary Actor: Listener
- Secondary Actor: Phone
- Precondition: Listener has installed and opened the application successfully
- Main Flow:
    1. Listener clicks on menu button from play screen
    2. Listener then clicks on ‘List’ in the menu
    3. Application redirects users to the list screen
    4. Listener finds station they want to view by name
    5. Listener clicks on station name
    6. Application displays station information
- Alternate Flow:
    *None

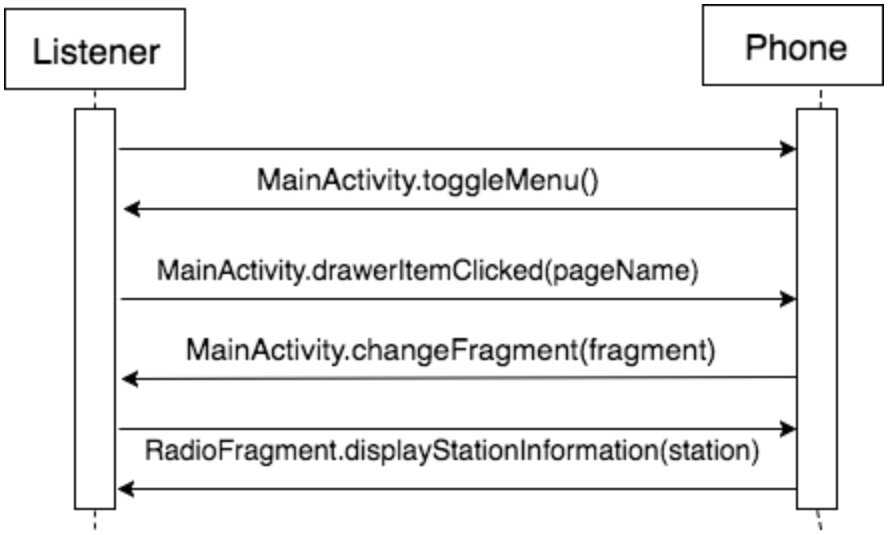


Figure 51

Use case diagram for 5.3.1.3: Use Case 3: Find Stations

10.3.1.4: Use Case 4: Direct to Submit a Station

- Primary Actor: Listener
- Secondary Actor: Phone
- Precondition: Listener has installed and opened the application successfully
- Main Flow:
  1. Listener clicks on menu button from play screen
  2. Listener clicks ‘Suggest a Station’
  3. Listener is redirected to mobile web browser
- Alternate Flow:
  *None

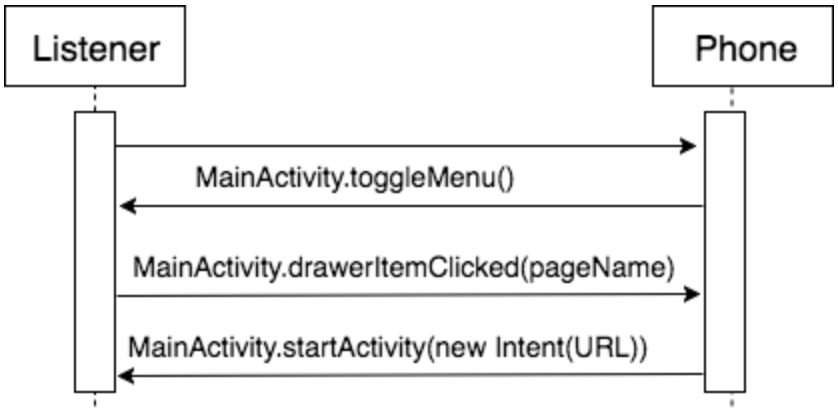


Figure 52

Use case diagram for 5.3.1.4: Use Case 4: Direct to Submit a Station

10.3.1.5: Use Case 5: Load Application

- Primary Actor: Listener
- Secondary Actor: Web server
- Precondition: Application is installed on the phone
- Main Flow:
    1. Listener opens application
    2. Phone makes load request
    3. Application loads all data from the web server
    4. Application will load the radio screen
    5. Pop up will appear letting the listener know that they can flip to landscape mode.
- Alternate Flow:

    2a. Application fails to load data from the web server and displays a no internet message

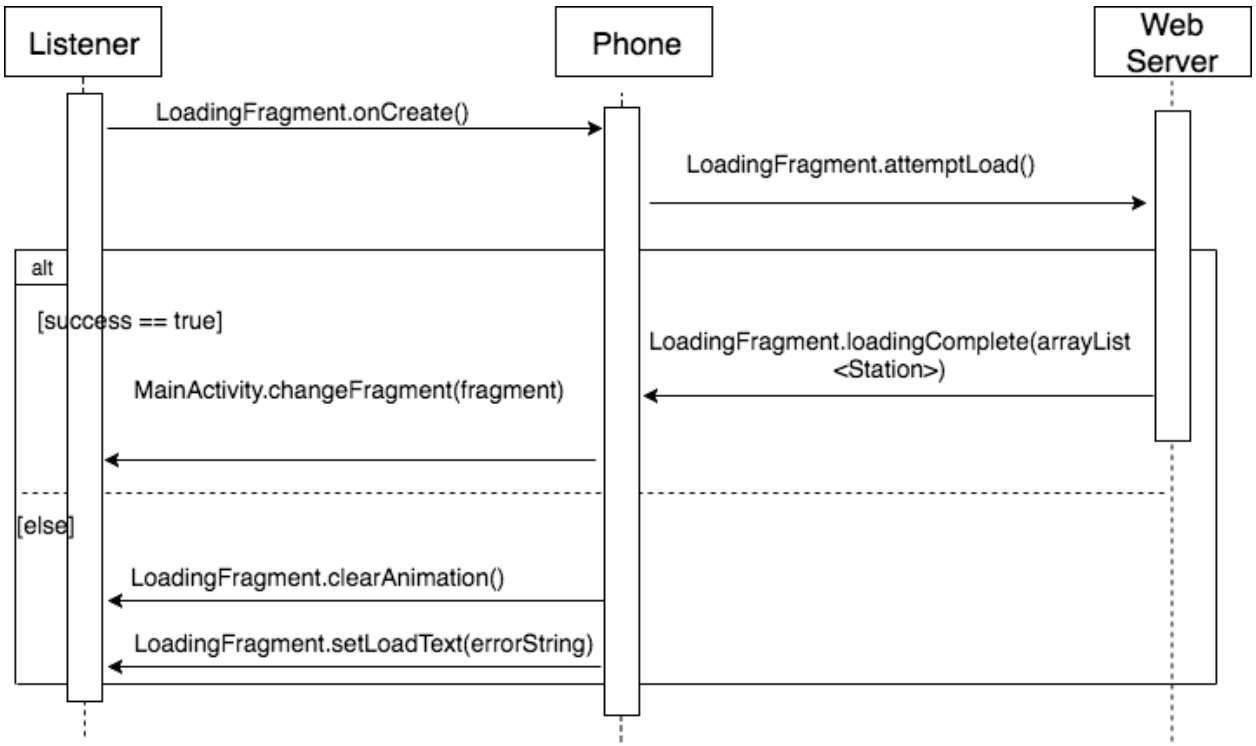


Figure 53

Use case diagram for 5.3.1.5: Use Case 5: Load Application

10.3.2: Admin Panel Use Cases

10.3.2.1: Use Case 6: Login to Web Server / Website

- Primary Actor: Admin
- Secondary Actor: Web server
- Precondition: Website is loaded
- Main Flow:
    1. Admin enters in email and password information
    2. Web server gets admin information
    3. Web server validates admin information
    4. Web server returns session key and id
    5. Admin is redirected to the select database screen
    6. Admin selects ‘Midwest Radio’
- Alternate Flow:

    3a. Application fails to validate admin information and displays error

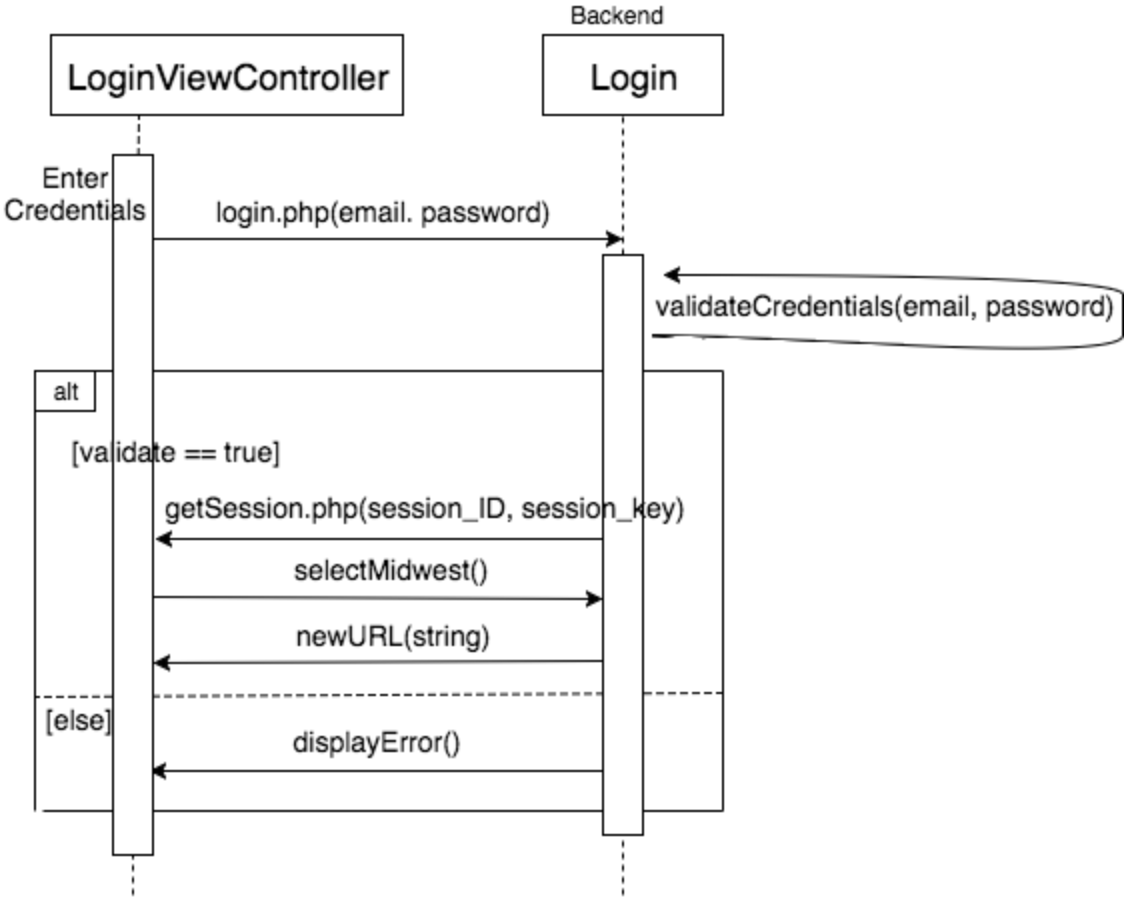


Figure 54

Use case diagram for 5.3.2.1: Use Case 6: Login to Web Server / Website

10.3.2.2: Use Case 7: Add a Station

- Primary Actor: Admin
- Secondary Actor: Web Server
- Precondition: Successfully logged in with credentials
- Main Flow:
    1. Admin clicks ‘Add Station’
    2. Application displays a table to add station information
    3. Admin enters station information
    4. Admin clicks ‘Activate’ to activate station
    5. Web server adds station
- Alternate Flow:
    *None

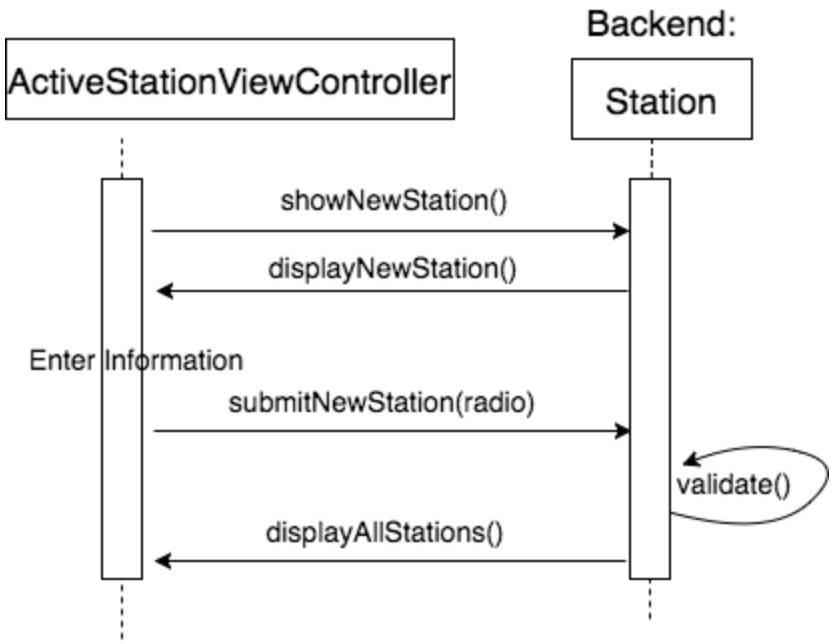


Figure 55

Use case diagram for 5.3.2.2: Use Case 7: Add a Station

10.3.2.3: Use Case 8: Create New User for Backend

- Primary Actor: Web Server
- Secondary Actor: Admin
- Precondition: Admin goes to website and logins with credentials
- Main Flow:
    1. Admin clicks on ‘Manage Users’
    2. Admin clicks on ‘Create New User’

3. Admin enters in email and password
4. Web server compares hash password and email to database to see if there is a user that matches it
5. Web server hashes password
6. Web server stores credentials
7. Web server redirects admin to admin panel

- Alternate Flow:

  2a. Web server returns error when hash password and email doesn't match database

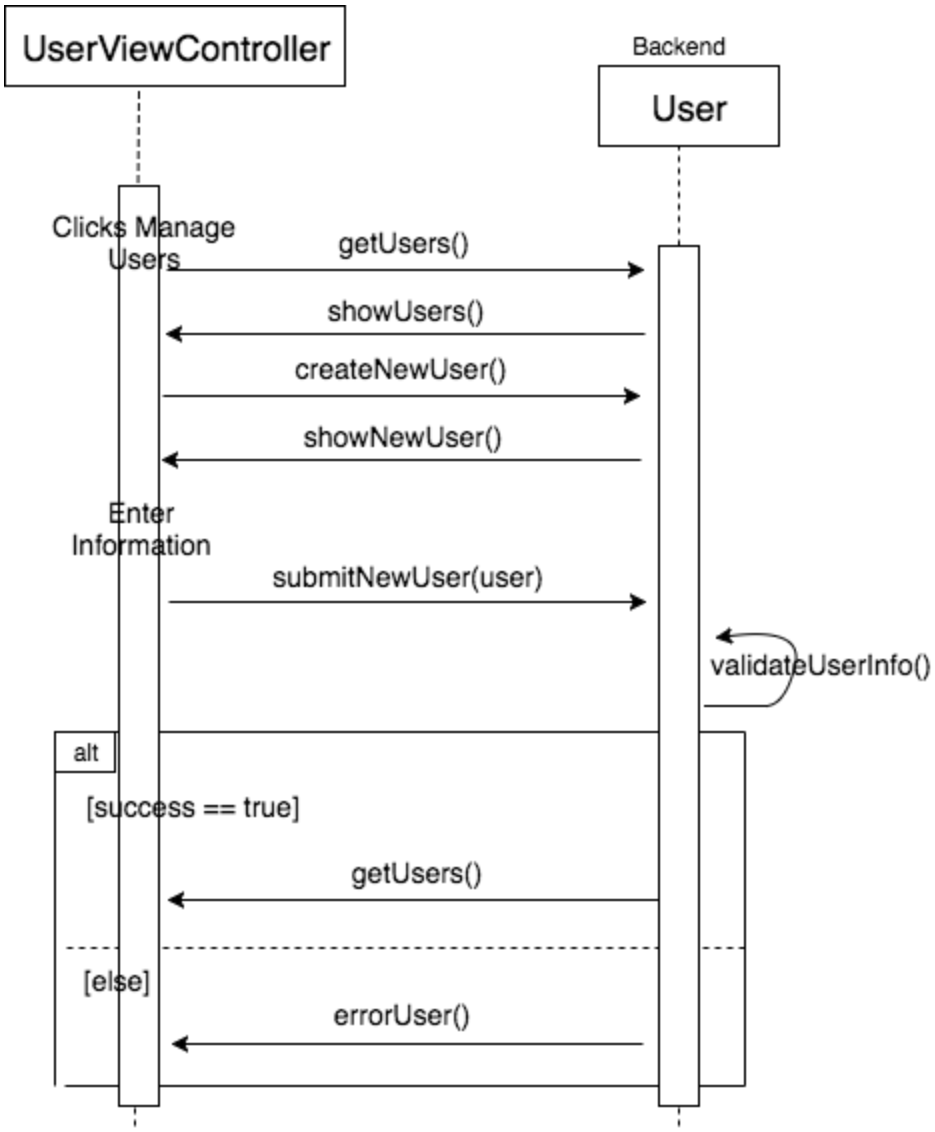


Figure 56

Use case diagram for 5.3.2.3: Use Case 8: Create New User for Backend

11.0: Future Considerations for Milestone #3

We have already released our first version of the app to the stores. So, moving forward, we plan to implement more features and release a second version at the very least. We will also spend time finding and fixing bugs so our app functions the best that it can. Lastly, we will be creating any documents/tutorials that our client and/or future developers will need for the apps and admin panel when we are gone. This seems very possible with the removal of what our client viewed as unnecessary features. We are proud of what we've achieved thus far, but will continue to push to the end and create the best possible products for our client and the application users.

12.0 Progress Update for Milestone #4

12.1 Progress Update

Our final milestone is complete. It was another seven weeks of hard work for the team. We are happy to announce that all work that was scheduled to be complete is complete.

12.2 GANTT Chart

Activity	Plan Start (Sprints)	Plan Duration (Weeks)	Acutual Start (Weeks)	Actual Duration (Weeks)	Percent Completed	Periods	1 (W 1:2) 9/24 - 10/7	2 (W 3:4) 10/8 - 10/21	3 (W5:6) 10/22 - 11/4	4 (W6:7) 11/5 - 11/18	5 (W8:9) 11/19 - 12/2	6 (W10 - 11) 12/3 - 12 - 16	7 (W 12 - 13) 1/7 - 1/20	8 (W 14 -15) 1/21 - 2/3	9(W16 - 17) 2/4 - 2/17	10 (W 18 -19) 2/18 - 3/3	11 (W 20 - 21) 3/11 - 3/24	12 (W 22 -23) 3/25 - 4/7	13 (W 24 - 25) 4/8 - 4/21	14 (W 26 - 27) 4/22 - 5/4
Requirements analysis	1	1	1	1	100%															
UI Mock Ups	1	2	1	2	100%															
Frontend Joint Admin Panel	1	7	4	7	100%															
Backend Database Creation	1	2	1	2	100%															
Front End Admin Panel	2	2	2	2	100%															
Backend Database Creation	1	3	4	3	100%															
Fontend Public Facing Site	7	2	13	2	100%															
Learning React	1	3	1	2	100%															
List Stations	2	2	4	2	100%															
Radio Station Player (App)	3	3	6	3	100%															
Playing Radio Player	6	2	8	2	100%															
Bug Testing	6	2	6	2	100%															
Inital Push to App Stores	7	2	6	2	100%															
Menu Navigation	8	1	12	1	100%															
Setup Scanning	8	4	12	1	100%															
- Picking Criteria	N/A	N/A	N/A	N/A	N/A															
- Showing Stations	9	1	12	1	100%															
- Playing / Stoping Stations	9	1	12	1	100%															
- Prepare Scanning list for radio	N/A	N/A	N/A	N/A	N/A															
Presets	10	2	10	2	100%															
-Edit Presets	10	2	10	2	100%															
-Saving Presets	10	2	10	2	100%															
- Loading Presets	N/A	N/A	N/A	N/A	N/A															
- Toggle between Presets	N/A	N/A	N/A	N/A	N/A															
- CRUD on banks	N/A	N/A	N/A	N/A	N/A															
Bug Testing	11	2	11	2	100%															
Pushing to App Store	13	1	13	1	100%															

Figure 57

Here is the latest version of our GANTT Chart. We were able to complete all assigned tasks within our timeline. The red lines represent features that are no longer valid.

13.0 Testing / Validation

13.1 iOS Development

13.1.1: Unit Testing

Below are the unit test cases for the iOS application. The following test cases ensure that critical components of the application that receive dynamic input are correct.

- Station list parsing
- Station Saving

Test #	**Test Case**	**Condition**	**Expected Results**	**Pass / Fail**
1	Make station list full	Active stations JSON Array supplied to StreamPlayer	StreamPlayer returns an ArrayList, where each	Pass

			object is of type Station.class	
2	Save new Station to preset	Button and station ID that is not in local storage is supplied to the changePresetFunction	Station ID is saved to local storage.	Pass

13.1.2: Functional Testing

Below is a table of the test cases completed for the iOS application. If it had a functional requirement(FR) that went with it, it is labeled under the test number. If there is a functional requirement that is not shown below, it is because that requirement has been removed.

Test #	**Test Case**	**Condition**	**Expected Results**	**Pass / Fail**
1 FR 2.2.1.1	View menu options	Click the hamburger button in the top hand of the screen	Displays menu options	Pass
2	Navigate to the 'All Stations' screen from the menu	Click the hamburger button in the top hand of the screen then click 'All Stations'	Displays the 'All Stations' screen, with the stations in alphabetical order	Pass
3 FR 2.2.1.1.5	Navigate to the 'Middletown Music' website from the menu	Click the hamburger button in the top hand of the screen then click 'Middletown Music'	Displays the 'Middletown Music' website in the web browser	Pass
4 FR 2.2.1.1.6	Navigate to the 'Suggest A Station' screen from the menu	Click the hamburger button in the top right hand of the screen then click 'Suggest A Station'	Displays 'Suggest A Station' in the web browser	Pass
5	Navigate to the 'Report an Issue' screen from the menu	Click the hamburger button in the top hand of the screen then click 'Report An Issue'	Displays 'Report An Issue' in the web browser	Pass
6 FR 2.2.2.3	Change to next station on the right	Click on the single arrow button to the right	Radio changes to the next station to the right	Pass
7 FR 2.2.2.3	Change to the next station on the left	Click on the single arrow button to the left	Radio changes to the next station to the left	Pass

8 FR 2.2.2.2, 2.2.2.2.1	Scan through stations to the right	Click on the double arrow button to the right	Radio scans to the next station to the right, stays for 5 seconds, and then scans again to the right continuously	Pass
9 FR 2.2.2.2, 2.2.2.2.1	Scan through stations to the left	Click on the double arrow button to the left	Radio scans to the next station to the left, stays for 5 seconds, and then scans again to the left continuously	Pass
10	Stop scanning through stations	Click any button on the radio while scanning	Scanning stops and current station continues playing	Pass
11 FR 2.2.2.4	Play a station	Click 'Play'	Plays the station currently in the radio	Pass
12 FR 2.2.2.5	Stop a station	Click 'Stop'	Stops the station currently in the radio	Pass
13 FR 2.2.2.6	Play a station from the preset area	Click on one of the saved preset stations	Begin playing the preset station clicked on	Pass
14 FR 2.2.2.9	Save a station to preset area	Long hold preset spot	Station currently in radio saves to that preset spot	Pass
15 FR 2.2.7.3	View station information	On 'All Stations' screen, click on one of the stations	Displays station information and "Play this Station" button in pop up	Pass
16	Make station information disappear	On 'All Stations' screen, click out of the pop up (anywhere else on the screen)	Remove station information pop up	Pass
17	Play station chosen from 'All Stations' screen	Click on one of the stations and then click 'Play this Station'	Redirects user to radio screen and begins playing that station	Pass
18 FR 2.2.2.10, 2.2.7.4	Display helpful information	Click on the question mark	Displays helpful information for that screen	Pass

13.1.3: Environment Testing

Device	**iOS version**	**App Build**	**Functional Tests**

iPhone 7 Plus	11	Success	All Pass
iPhone 7	11	Success	All Pass
iPhone 6s	11	Success	All Pass
iPhone 6s Plus	11	Success	All Pass
iPhone 6	11	Success	All Pass
iPhone 6 Plus	11	Success	All Pass
iPhone SE	11	Success	All Pass
iPhone 5s	11	Success	All Pass
iPhone 6s	10	Success	All Pass
iPhone 6s Plus	10	Success	All Pass
iPhone 6	10	Success	All Pass
iPhone 6 Plus	10	Success	All Pass
iPhone SE	10	Success	All Pass
iPhone 5s	10	Success	All Pass
iPhone 5c	10	Success	All Pass
iPhone 5	10	Success	All Pass

13.2 Android Development

13.2.1: Unit Testing

Below are the unit test cases for the Android application. The following test cases ensure that critical components of the application that receive dynamic input are correct. These cases have been integrated into the build cycle via Gradle.

- Station list parsing
- Station Saving

Test #	**Test Case**	**Condition**	**Expected Results**	**Pass / Fail**

1	Make station list full	Active stations JSON Array supplied to factory	Factory returns an ArrayList, where each object is of type Station.class	Pass
2	Make station list partial	Active stations JSON Array supplied to factory with empty values	Factory returns an ArrayList, where each object is of type Station.class	Pass
3	Save new Station to preset	Button and station ID that is not in local storage is supplied to the changePresetFunction	Station ID is saved to local storage.	Pass
4	Re-save station to preset	Button and station ID that is in local storage is supplied to the changePresetFunction	Station ID is not saved to local storage.	Pass

```
                "]}";

        JSONObject ob = new JSONObject(sampleInfoFull);
        JSONArray arr = ob.getJSONArray("active");
        this.input = arr;

        ob = new JSONObject(sampleInfoEmpty);
        arr = ob.getJSONArray("active");
        this.input2 = arr;

    }

    @Test
    public void makeStationListTest() throws Exception {
        StationListFactory test = new StationListFactory();
        for (Station s: test.makeObjects(input)) {
            assertEquals(Station.class, s.getClass() );
        }
    }

    @Test
    public void makeStationListEmptyTest() throws Exception {
        StationListFactory test = new StationListFactory();
        for (Station s: test.makeObjects(input2)) {
            assertEquals(Station.class, s.getClass() );
        }
    }
}
```

13.2.2: Functional Testing

Below is a table of the test cases completed for the Android application. If it had a functional requirement(FR) that went with it, it is labeled under the test number. If there is a functional requirement that is not shown below, it is because that requirement has been removed.

Test #	**Test Case**	**Condition**	**Expected Results**	**Pass / Fail**
1 FR 2.2.1.1	View menu options	Click the hamburger button in the top hand of the screen	Displays menu options	Pass

2	Navigate to the ‘All Stations’ screen from the menu	Click the hamburger button in the top hand of the screen then click ‘All Stations’	Displays the ‘All Stations’ screen in alphabetical order	Pass
3 FR 2.2.1.1.5	Navigate to the ‘Middletown Music’ website from the menu	Click the hamburger button in the top hand of the screen then click ‘Middletown Music’	Displays the ‘Middletown Music’ website in the web browser	Pass
4 FR 2.2.1.1.6	Navigate to the ‘Suggest A Station’ screen from the menu	Click the hamburger button in the top right hand of the screen then click ‘Suggest A Station’	Displays ‘Suggest A Station’ in the web browser	Pass
5	Navigate to the ‘Report an Issue’ screen from the menu	Click the hamburger button in the top hand of the screen then click ‘Report An Issue’	Displays ‘Report An Issue’ in the web browser	Pass
6 FR 2.2.2.3	Change to next station on the right	Click on the single arrow button to the right	Radio changes to the next station to the right	Pass
7 FR 2.2.2.3	Change to the next station on the left	Click on the single arrow button to the left	Radio changes to the next station to the left	Pass
8 FR 2.2.2.2, 2.2.2.2.1	Scan through stations to the right	Click on the double arrow button to the right	Radio scans to the next station to the right, stays for 5 seconds, and then scans again to the right continuously	Pass
9 FR 2.2.2.2, 2.2.2.2.1	Scan through stations to the left	Click on the double arrow button to the left	Radio scans to the next station to the left, stays for 5 seconds, and then scans again to the left continuously	Pass
10	Stop scanning through stations	Click ‘Stop’	Scanning stops	Pass
11 FR 2.2.2.4	Play a station	Click ‘Play’	Plays the station currently in the radio	Pass
12 FR 2.2.2.5	Stop a station	Click ‘Stop’	Stops the station currently in the radio	Pass
13 FR 2.2.2.6	Play a station from the preset area	Click on one of the saved preset stations	Begin playing the preset station clicked on	Pass
14	Save a station to	Long hold preset spot	Station currently in radio	Pass

FR 2.2.2.9	preset area		saves to that preset spot	
15 FR 2.2.7.3	View station information	On ‘All Stations’ screen, click on one of the stations	Displays station information in modal	Pass
16	Make station information disappear	On ‘All Stations’ screen, click the back button	Removed station information modal	Pass
17	Play station chosen from ‘All Stations’ screen	Click on one of the stations and then click the ‘Play’ button	Redirects user to radio screen and begins playing that station	Pass
19 FR 2.2.2.10, 2.2.7.4	Display helpful information	Click on the question mark	Displays helpful information for that screen	Pass

13.2.3: Environment Testing

Device	**API version**	**Gradle Build**	**Functional Tests**
Galaxy Nexus	16	Success	All Pass
Galaxy Nexus	19	Success	All Pass
Galaxy Nexus 4	19	Success	All Pass
Galaxy Nexus 5	23	Success	All Pass
Galaxy Nexus 5	24	Success	All Pass
Galaxy Nexus 6	25	Success	All Pass
Galaxy Nexus 6	23	Success	All Pass
Galaxy Nexus S	25	Success	All Pass
Pixel 2	24	Success	All Pass
Samsung Galaxy S6	25	Success	All Pass
Pixel 2 XL	27	Success	All Pass

13.3 Front End Development

13.3.1: Functional Testing for Admin Panel

Below is a table of the test cases completed for the admin panel. If it had a functional requirement(FR) that went with it, it was labeled under the test number. If there is a functional requirement that is not shown below, it is because that requirement has been removed.

Test #	**Test Case**	**Condition**	**Expected Results**	**Pass / Fail**
1 FR: 2.2.8.1, 2.2.8.2	Login to admin panel	Enter valid email and password and click 'Login'	Redirects user to database selection screen	Pass
2	New password sent for valid email	Click 'Forgot Password', type in valid email, and click 'Reset My Password'	Email sent to user with link to reset their password	Pass
3	Add user to admin panel	Click 'New User', type in user information, and click 'Save New User'	Adds user to user table	Pass
4	Delete user from admin panel	Click 'Delete' next to user	Deletes user from admin panel	Pass
5	Make user the admin	Click 'Make Recipient' next to user	Makes user current admin for the admin panel	Pass
6 FR: 2.2.8.3	Go to 'Midwest Radio' admin panel	Enter in valid login credentials, click 'Login', then click 'Midwest Radio'	Redirects user to Midwest Radio admin panel	Pass
7	Add a station	Click 'Add Station', enter station information, and click 'Activate'	Adds station to the application	Pass
8 FR: 2.2.8.4	Toggle filter options	Click 'Genre', 'Ownership', or 'Geographical' once to open and another time to close	Shows and unshows filter options	Pass
9	Edit a station	Click 'Edit', change the station information, and click 'Save'	Station information updated	Pass
10	Toggle edit and save buttons	Click 'Edit' to edit the station and click 'Save' to update the	Toggles the button from 'Edit' to 'Save'	Pass

		station		
11	Make station go from active to pending	Click ‘Pending’	Station goes to pending stations list	Pass
12	Make station go from active to deleted	Click ‘Delete’	Station goes to deleted stations list	Pass
13	Make station go from pending to active	Click ‘Activate’	Station goes to active stations list	Pass
14	Make station go from pending to deleted	Click ‘Delete’	Station goes to deleted stations list	Pass
15	Make station go from deleted to pending	Click ‘Pending’	Station goes to pending stations list	Pass
16	Make station go from deleted to active	Click ‘Activate’	Station goes to active stations list	Pass
17	Delete station forever	Click ‘Delete’	Station gets deleted from the database	Pass
18	Sort table	Click on any of the table headings	Table sorts stations alphabetically by that column	Pass
19	Change first station shown in app	Select station and click ‘Save First Station’	Station becomes first station loaded on applications	Pass
20 FR: 2.2.8.6	Download stations	Click ‘All Stations Download’	Excel sheet of all the stations downloads to computer	Pass

13.3.2: Functional Testing for Public Website

Below is a table of the test cases completed for the public website. If it had a functional requirement(FR) that went with it, it was labeled under the test number. If there is a functional requirement that is not shown below, it is because that requirement has been removed.

Test #	**Test Case**	**Condition**	**Expected Results**	**Pass**

				/ Fail
1 FR: 2.2.9.1	View stations in applications	Go to website	Redirects user to website showing all stations in applications	Pass
2 FR: 2.2.9.2	Submit station for application	Go to http://willshare.com/cs495/MidwestRadioPlayer/frontend/#/submit, type in the station information, and click 'Submit'	Station information is sent to the pending page in the admin panel	Pass
3	Report an error with a station	Go to http://willshare.com/cs495/MidwestRadioPlayer/frontend/#/report, type in the information wrong with a station, and click 'Report'	Email with information reported is sent to admin	Pass

13.4 Backend Development / Server

13.4.1 Information

For the testing of the backend, a variety of test happened over the course of the application mainly including API Unit test with PostMan.

Test #	**Test Case**	**Condition**	**Expected Results**	**Pass / Fail**
1	Login to Admin Panel	Enter correct email or password	Status: 200; Sessions Variables should be returned	Pass
2	Login to Admin Panel	Enter incorrect email or password	Status: 400; Explaining no user	Pass
3	Login to Admin Panel	Missing Data Fields	Status: 400; Explaining no user	Pass
4	Password Reset	Enter a valid email	Status: 200; Explaining that information has been sent. Email should come through	Pass
5	Password Reset	Enter a nonvalid email	Status: 200; Explaining that information has been sent.	Pass
6	Adding New User	Adding correct information	Status: 200; User Should be returned	Pass

7	Adding New User	Missing some required information	Status: 400; Explaining data fields not all there	Pass
8	Deleting User	Enter correct user ID	Status: 200; User should now be deleted	Pass
9	Deleting User	Enter an invalid user ID	Status: 400; Explaining data fields are not correct	Pass
10	Adding Stations	Entering all required information	Status 200; Station should be added	Pass
11	Adding Stations	Entering not all required information	Status 400; Explaining data fields not all there	Pass
12	Adding User Station	Entering all required information	Status: 200; Station should be added and the User Entered parameter should be set	Pass
12	Editing Stations	Entering all the required information	Status: 200; Station should be updated with the current information	Pass
13	Editing Stations	Enter not all the required information	Status: 400; Explaining data fields not all there	Pass
14	Add Votes	Enter Required information with current data	Status: 200; Stations should now have a plus one in the vote	Pass
15	Add Votes	Enter Required information but not a valid station DI	Status: 400; Explaining data fields not valid	Pass
16	Add Votes	Enter not all required information	Status: 400; Explaining data fields not all there	Pass
17	Reporting Station	Enter Required Information	Status: 200; Information has now been reported.	Pass
18	Reporting Station	Enter Required Information	Status: 400; Explaining data fields not all there	Pass
19	Getting Application Data	GET Request	Information should be returned in the correct JSON format	Pass
20	Getting Application Data	POST Request	Status: 400; Information should not be returned.	Pass

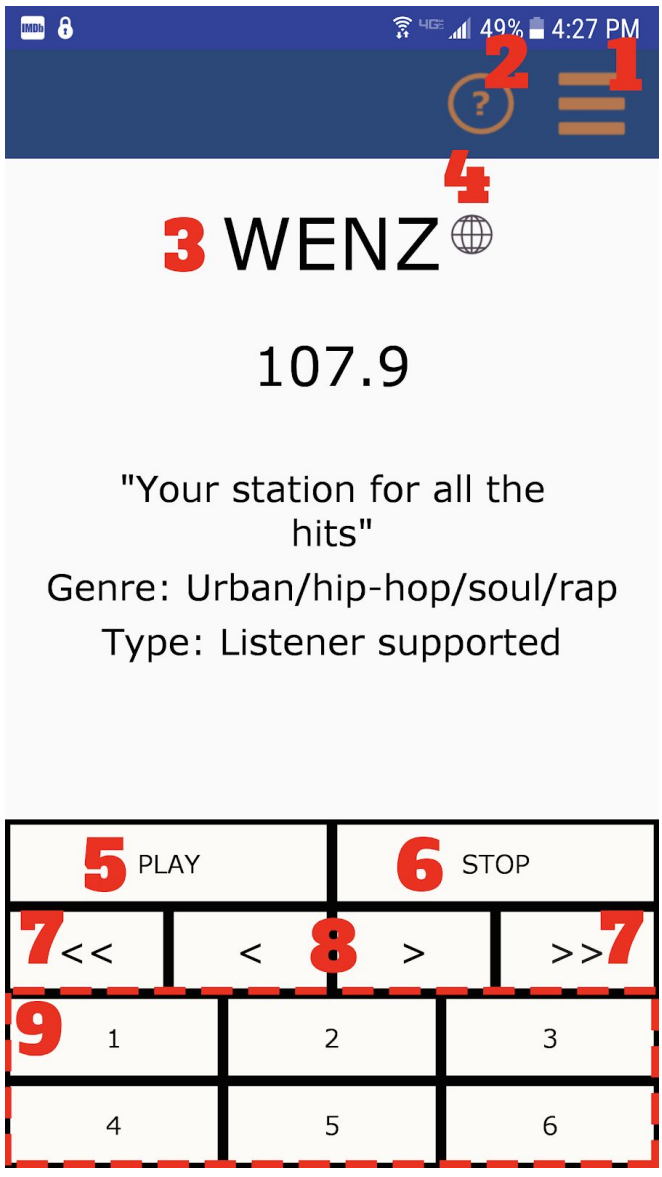
14.0 Documentation

14.1 iOS / Android User Manual

14.1.1 Tutorials / Documentation

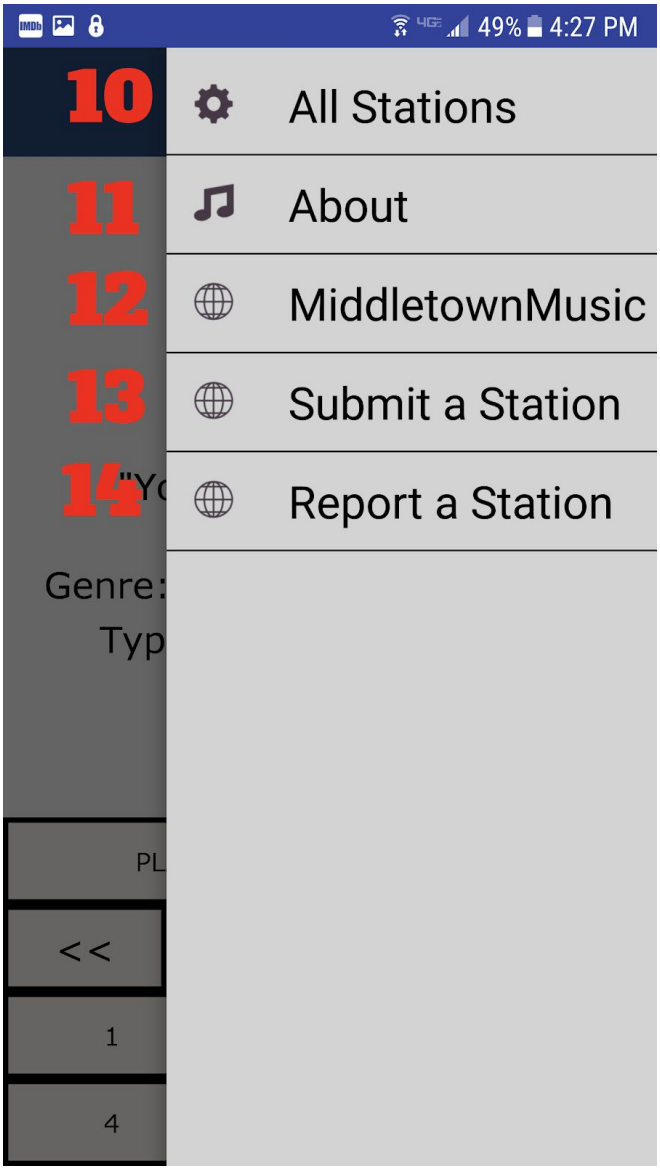
Play Screen

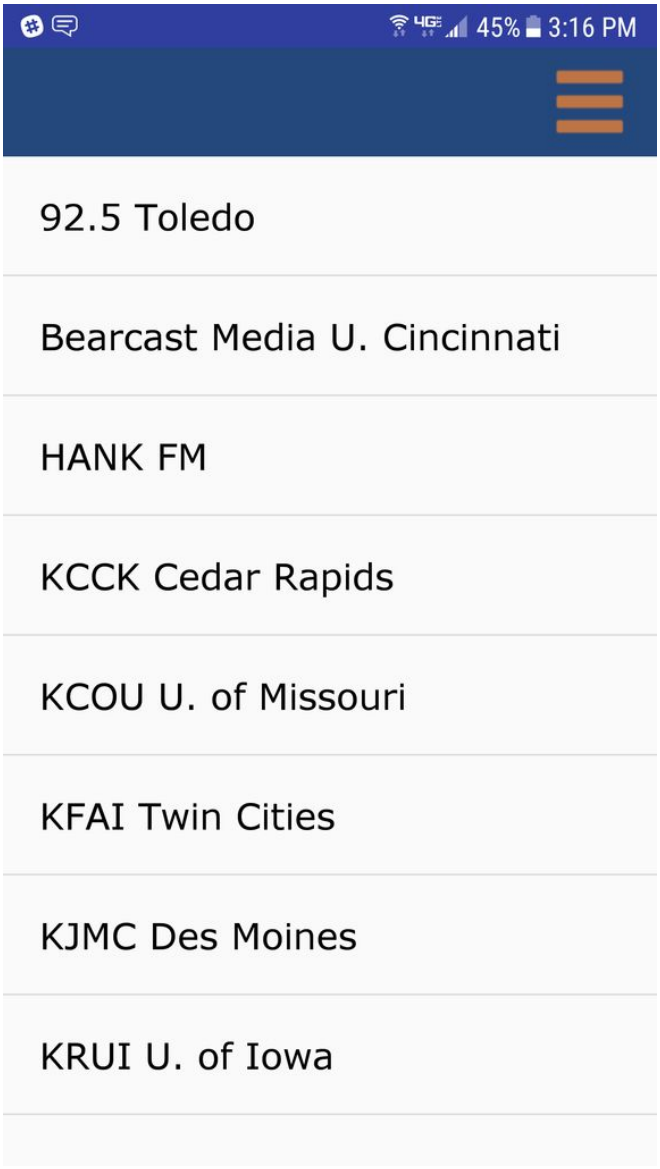
1. Click here for the Main Menu
2. Click here for the help text for that screen
3. Here is all the radio station information is stored.
4. Click here for the website of the current listed station.
5. Click on the “Play” button to play the current station that is listed.
6. Click on the “Stop” button to stop the radio.
7. These buttons control the scan feature. The radio will stay on the station for 5 seconds and then move on.
8. The forward and back buttons let the user browse, but does not play the station until the play button is hit
9. This boxed area is the presets. A user can press and hold to set the current radio station down.



Main Menu

10. Click here to see a list of stations
11. Click here to see general information about the app
12. Click here to go to the Middletown Music Website
13. Click here to submit a station through the webpage
14. Click here to report a station that the data is not correct or a bad stream



Radio Stations 15. Here is a list of stations that are currently loaded into the app, in alphabetical order.. Clicking on the station will have the Radio modal load this station into the radio, and begin playing.	

Radio Module 16. This modal will show all the station information. You can click on the play button and that will redirect to the radio screen.	

14.1.2 Read Me Android

Dependencies: MidwestRadio/app/build.gradle

/Audio:

The Audio files AudioService.java and AudioPlayer.java are used to facilitate the App's use of ExoPlayer, and utilize a service architecture.

The audio player is responsible for managing the AudioPlayer instance, and managing a wifilock for the service.

/fragments:

Each java file within the fragments folder contains an instance subclassed from android.support.v4.app.DialogFragment or

android.support.v4.app.Fragment. each of these files is used to control the corresponding xml view found in app/src/main/res/layout

or layout-land if applicable. These files depend upon distinct util classes, models, Audio, and the network.

All fragments are initialized in the MainActivity class, and managed with android.support.v4.app.FragmentManager;

/models:

This folder contains the crucial Station Object, along with it's factory to facilitate instantiation. The Station Object itself is parcelable to be used

within Android Bundles and contains redundant getter and setter methods that are required for Jackson json mapping. The StationList Factory itself

utilizes com.fasterxml.jackson.databind.ObjectMapper to map the json into an ArrayList of Station Objects. It is crucial that this Station object

contain every attribute received from the backend with the redundant getter and setter methods. Adding new attributes to the Station model

on the backend without updating the Station Object here could be detrimental to the application.

/network:

This folder manages the HTTP requests using a standard HTTP client, that is used in LoadingFragement and RadioFragment. Requests made by this

client expect an HTTP response code of 200 in order for the callback to be considered successful. Requests.java contains an example of the expected

JSON format for the station data.

/utils:

The Utils folder contains various front end components and callbacks that are necessary for the App, especially in the xml fragment files.

Deployment:

To redeploy for Android, the APK will need to be signed with the keystore file found on the google play developer console. The version name and version code will need to be incremented in /app/build.gradle before re-signing the apk.

14.1.3 Read Me iOS

LoadingScreenViewController.swift : contains the animation for the loading screen and makes the initial get request

MidwestBaseViewController.swift : parent class to control opening menu and help buttons\

NetworkRequestor.swift : contains all the network request code

RadioPlayerViewController.swift : the controller of the radio page

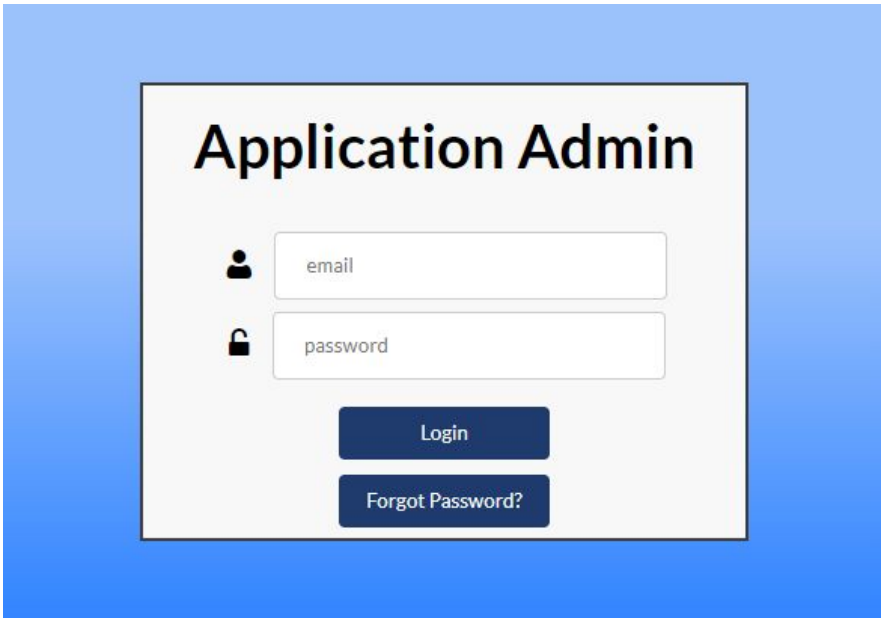
Station.swift : objectified station from the json

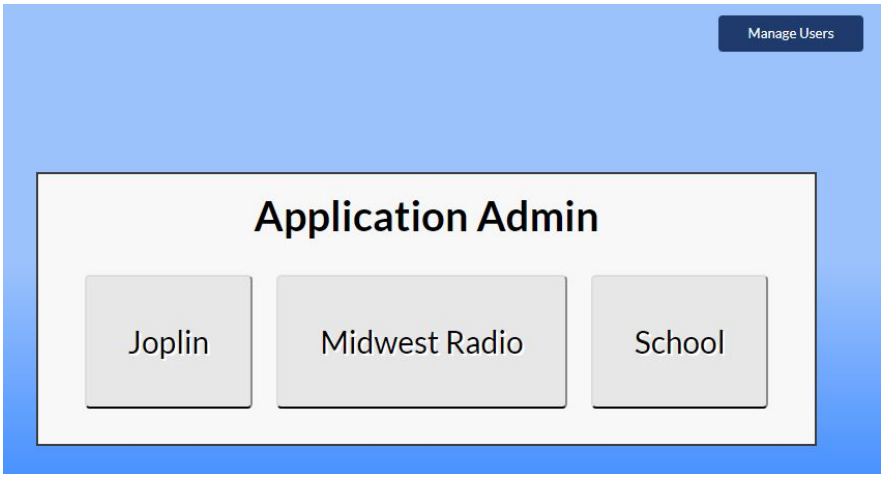
StationListViewController.swift : the controller of the station list page

StreamPlayer.swift : controls the audio streamer

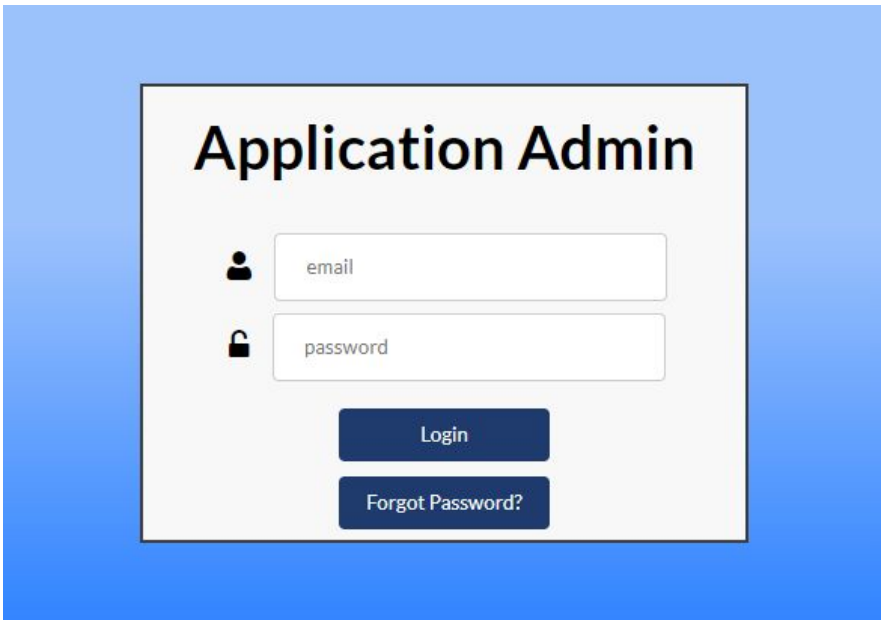
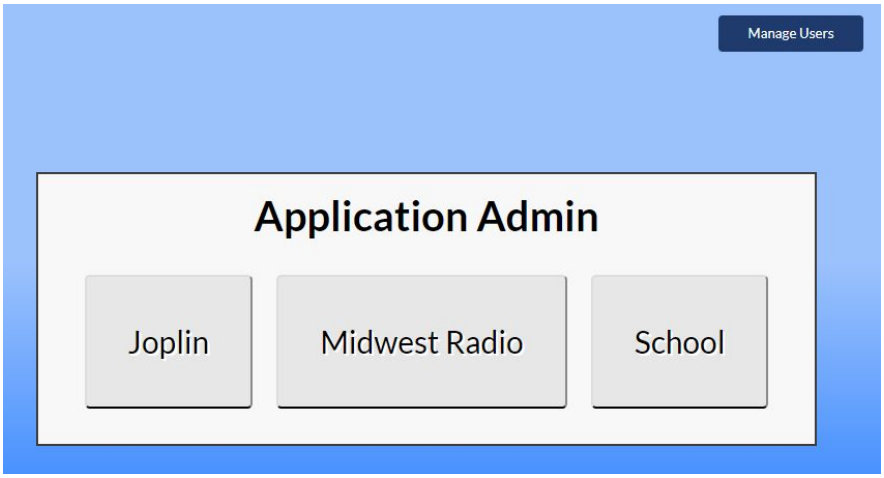
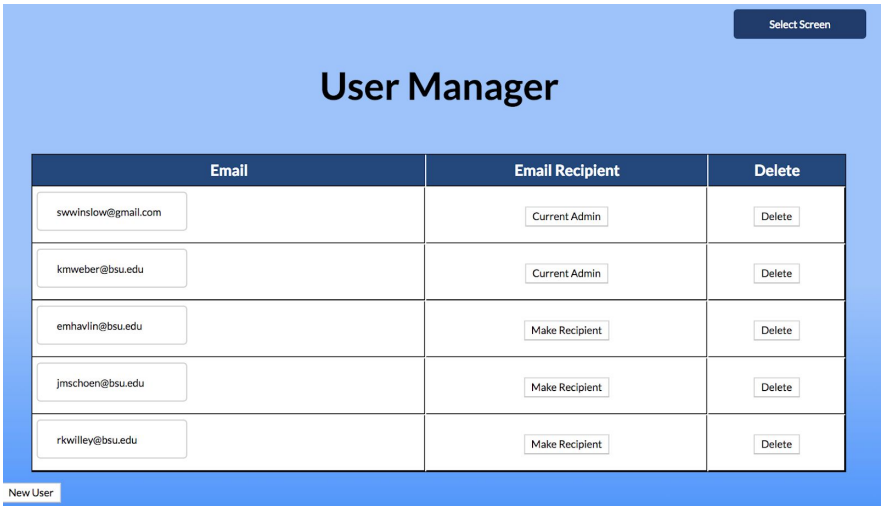
14.2 Front End Development User Manual

14.2.1 Login Tutorials

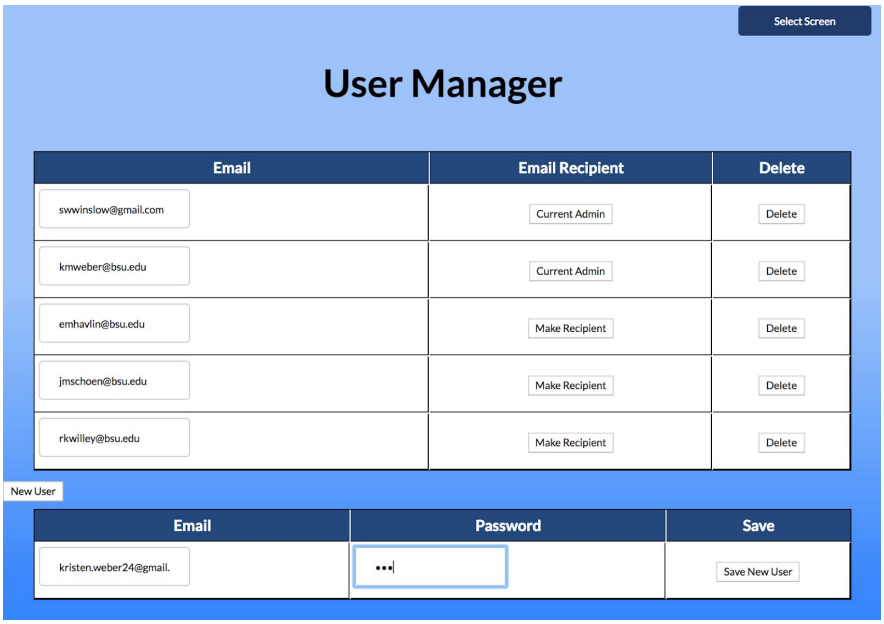
1. Go to http://willshare.com/cs495/admin/frontend/#/ and login with your email and password. Then click, 'Login'.	

2. If you have forgotten your password click ‘Forgot Password’ on the login page. It will redirect you to the page shown on the right. Type in your email and click ‘Reset My Password’. You will then receive an email with the new password for your account. And you can go back to the login screen and login with your email address and the new password.	
3. Once logged in, you can select ‘Manage Users’ in the upper right hand corner to manage users or select ‘Midwest Radio’ to go to the admin panel.	

14.3.2 Create New User Tutorials

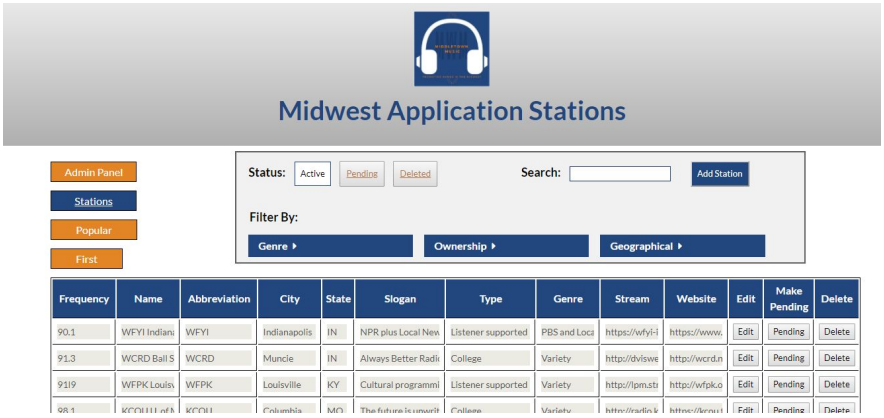
1. Go to http://willshare.com/cs495/admin/frontend/#/ and login with your email and password. Then click, 'Login'.	
2. Once logged in, select 'Manage Users' in the upper right hand corner.	
3. Next, click 'New User' in the bottom left hand corner of the screen.	

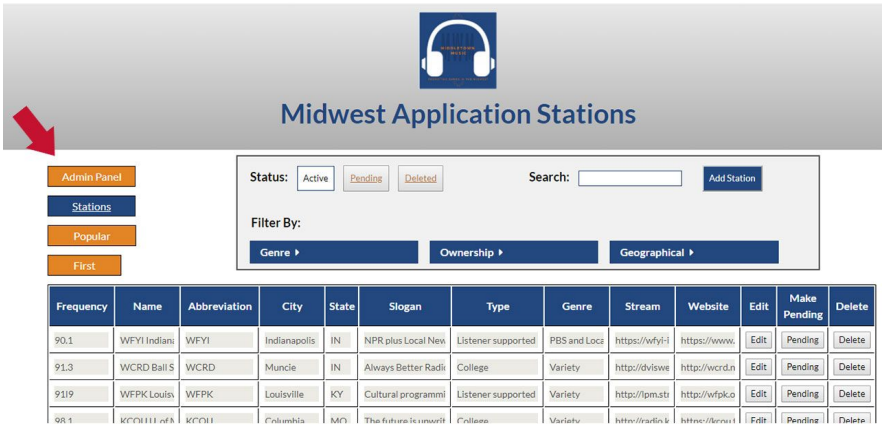
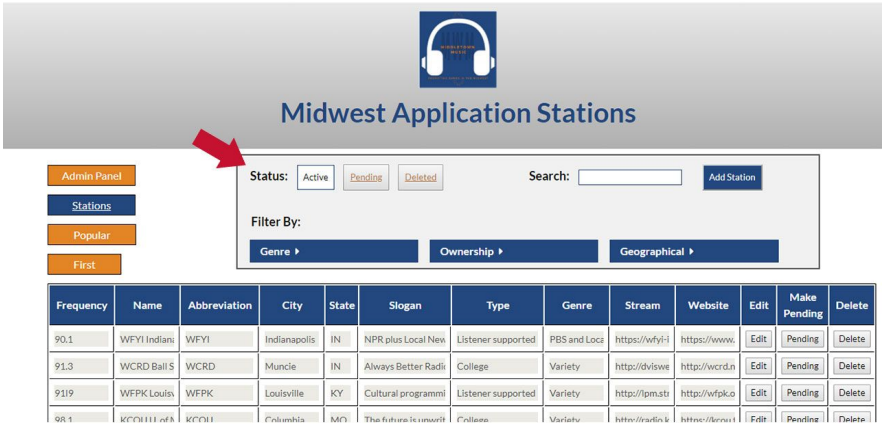
4. Then a table will pop up at the bottom where you can type in the user's email and password. Lastly, click 'Save New User' and that user will be added to the list.

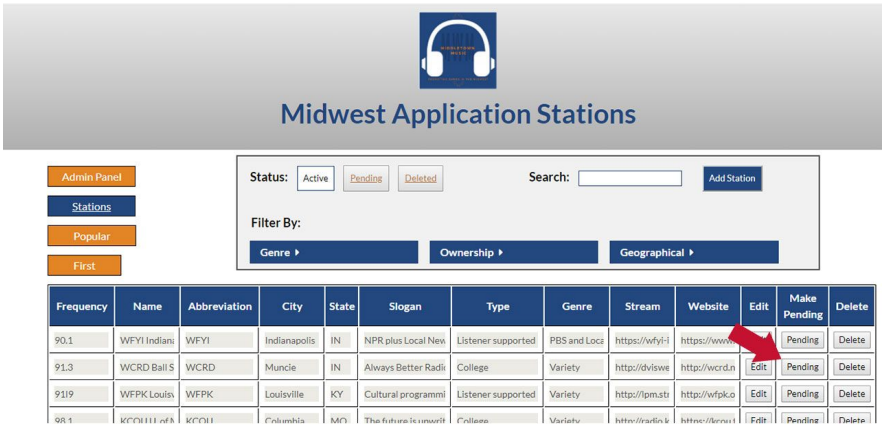
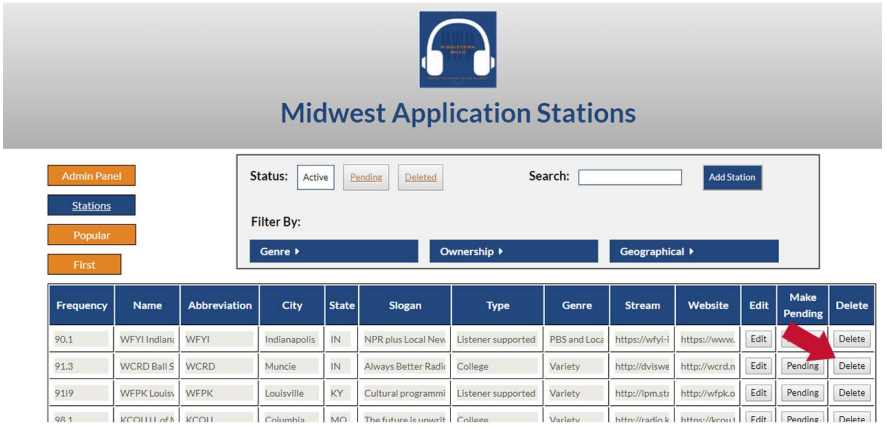


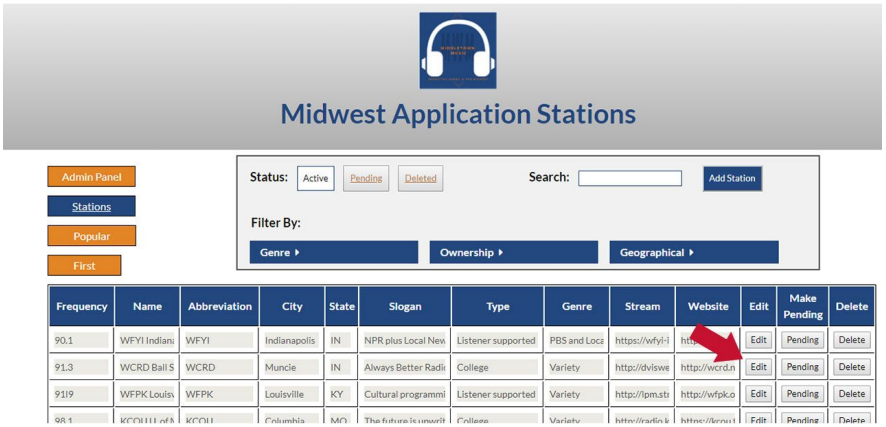
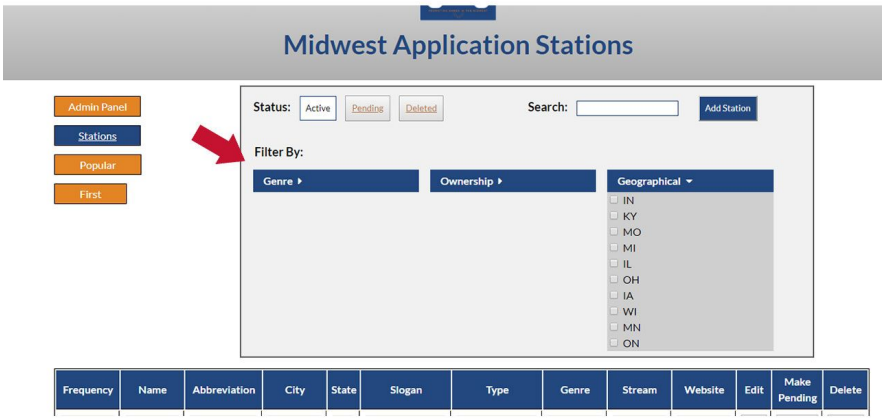
14.3.3 Admin Panel Tutorials

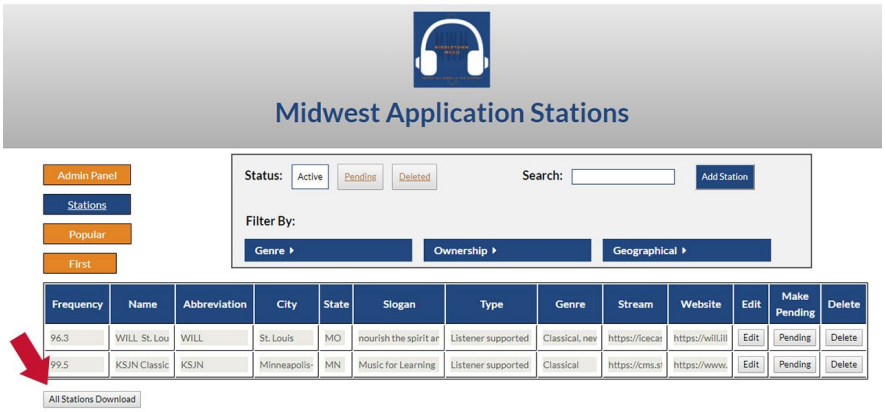
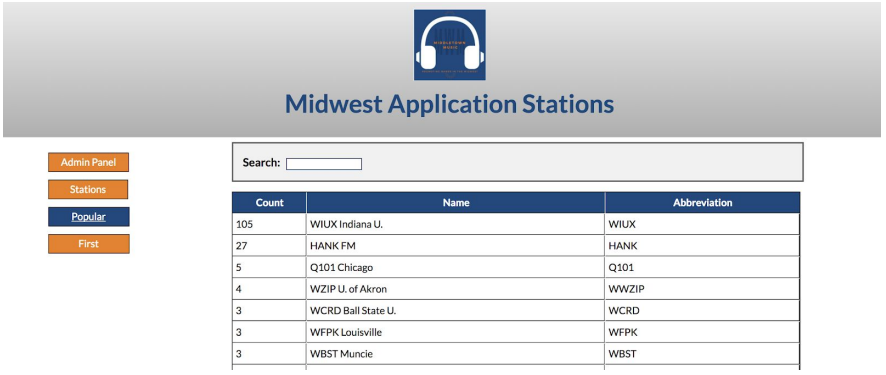
The admin panel is where you can **manage stations**. The photo to the right shows you the active stations and all their information.



The navigation on the left of the screen, is how you **navigate** the middletown admin panel. You can go back to the admin panel select screen, all stations screen, popular stations screen, and first station screen.	
The **status area** at the top of the screen tells you what stations you are currently viewing. The options are active, pending, and deleted. You can move stations between each status by looking at the last two columns and clicking the respective button. Example below.	

If you would like to **remove a station from the app** and make it pending, you go to the active station page and find that station in the table. Then select the 'pending' button. That station will then go to the pending stations and will be removed from the apps.	
If you would like to **remove a station from the app** and delete it, you go to the active station page and find the station in the table. Then select the 'delete' button. That station will then go to the deleted stations and will be removed from the apps.	

If you would like to **edit a stations information,** find the station in the table and then click the ‘Edit’ button. Once you have edited the information, click the ‘Save’ button. (The save button is in the same place the edit button was)	
In order to **filter stations**, you click on ‘Genre’, ‘Ownership’, or ‘Geography’ and select which criteria you would like to filter the below stations by.	

If you would like to **download all the stations**, you can scroll to the bottom of the active stations page. There you will see a button that says ‘All Stations Download’. Click that button and all stations will download into an excel sheet.	
If you would like to **add a station to the apps**, you first click the button on the right side of the screen that says ‘Add Station’. Then a table will pop up (like the screen to the right) where you can type in the station information. Once you have the information typed in, select the ‘Activate’ button and it will make the station active.	
If you would like to **view which stations are popular**, select ‘Popular’ from the navigation on the left.	

If you would like to change which **station shows up first when the app is loaded**, select ‘First’ from the navigation on the left. Then select which station you would like to show first and click ‘Save First Station’	

14.3 Backend Development / Server User Manual

The server is the heart of the application. It controls all three aspects to the software, the public website, the admin panel, and both applications. It is critical that the server stays up and running, but also no updates or change the the internal configs up the server. If changes are needed, take extreme caution. The changes that could happen, might break the server and therefore all parts of the application.

14.3.1 End Points

14.3.2.1 GetApplicationData.php

This GET endpoints gets all the information that is needed for the mobile applications.

14.3.2.2 AddVote.php

This is a POST endpoint that will submit a station to the voting database. Required field is the “station_id”.

14.3.2.3 AddStation.php

POST endpoint that lets the admin add stations to the database. It will take in all the different criteria for a station including: short_name, long_name, frequency, city, state, slogan, type, genre, stream, website, active.

14.3.2.4 GetPopular.php

GET endpoint that is used for the voting page on the admin page. It will return an array of all the stations that have votes.

14.3.2.5 UpdateStation.php

POST endpoint that will take in all the required fields from the station attribute and update them accordingly.

14.3.2.6 UEAddStation.php

POST endpoint that the user has to submit a station into the database. The required fields are the same as the normal station, expect the user can not insert the type, genre, and website. The station will be sent to the pending stations page.

14.3.2.7 ReportStation.php

POST endpoint that the users has to report a station has wrong information. The information that is collect is the long_name, broken, and comment. This information would then be emailed to the contacted users in the system.

15.0 Deployment / Handover plan

https://github.com/middletownradio is where all the cleaned out repo are stored. The real production copies are given directly to Dr. Willey through download.

15.1 iOS Development

15.1.1 Current Configuration

The iOS Application has been updated with the latest version on the Apple's App Store with Dr. Willey's account. Users have already started to download the application.

15.1.2 Reproduce

In order to reproduce for a another region, the items that need to change are the URLs to the database, the app logo, and adding an application to the App Store. It is critical that the backend and database same in similar structure in order for the application being able to work with new data.

15.2 Android Development

15.2.1 Current Configurations

The Android Application has been updated with the latest version on the Google Play Store with Dr. Willey's account. Users have already started to download the application. You will need the Android Key in order to submit it to the Google Play Store.

15.2.2 Reproduce

In order to reproduce this application for another region, the items that need to change are the URLs to the database, the app logo, and adding an application to the Google Play Store. It is critical that the backend and database use identical formats in order for the application to work correctly. Included in the source code are explicit examples on conforming to the application's format.

15.3 Front End Development

15.3.1 Current Configurations

The front end of the admin and the public websites are already live and on the server that Dr. Willey has supplied to us. All of the information is working and coming in correctly from the database that is also on the server.

15.3.2 Reproduce

In order to reproduce, the admin will need to create a new folder inside the server and load in all of the development files that make up the current admin panel. Then, the admin will need to go into the factories controller and change out the current backend URLs to the new backend URLs for the application. After that is complete, the front end should populate with the new data from another region.

15.4 Backend development

15.4.1 Current Configurations

The backend of the project is written all in PHP version 5.3. Therefore, the program might fail to work if that version is updated because of the now deprecated methods that are in its current verison. Everything has been uploaded to the iPower server.

15.4.2 Reproduce

In order to reproduce the backend code for other regions, the admin will need to create a new directory for all the backend files and connect it to the new database configs that have been set up.

15.5 Database

15.5.1 Current Configurations

Below is the most recent database structure for the stations table. The current version of mySQL is 5.6.32.

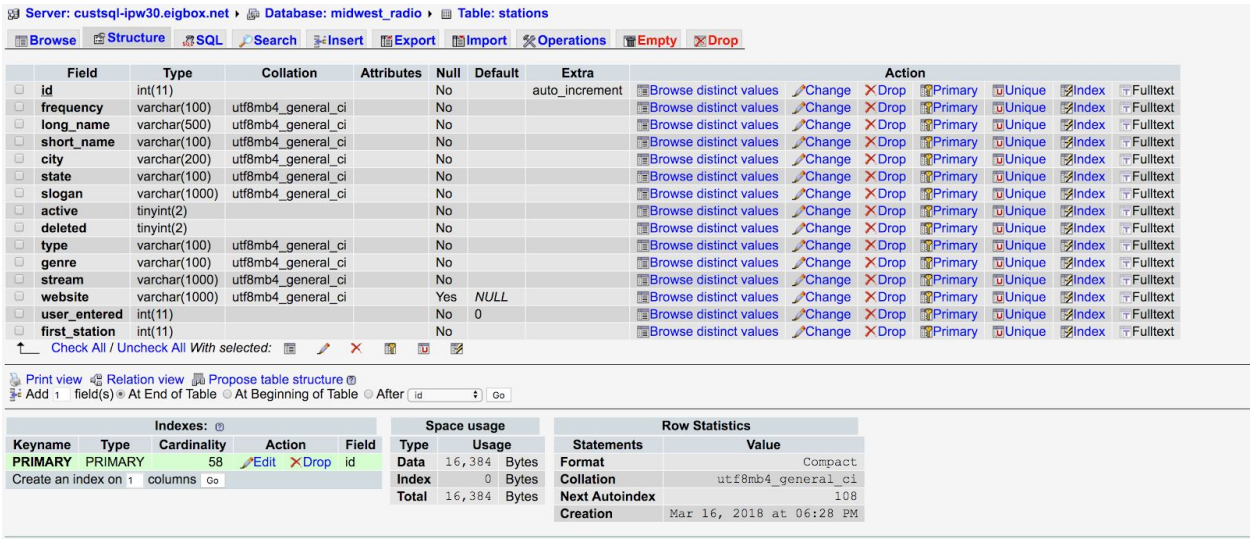


Figure 58

Here is the latest structure for the station

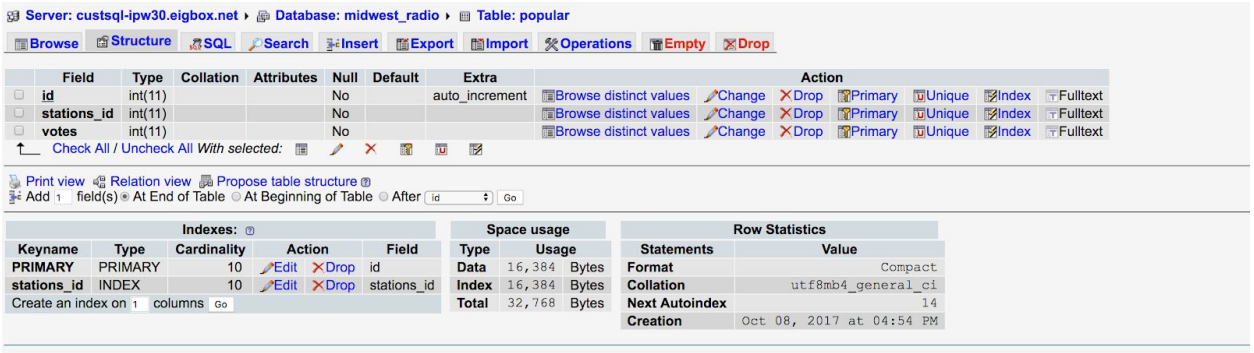


Figure 59

This is the latest structure for the votes in the database.

15.5.2 Reproduce

In order to reproduce this application into other regions with the same code base, the admin will need to copy the database tables and clear all the data from them. If it is on the same server, you will be able to use the same username and password. If not, you will need to the credentials to access the database.

16.0 Dependencies

16.1 iOS Development

Development & Testing dependencies:

- XCode, version 9.2

16.2 Android Development

Development dependencies:

- Android SDK 16-current
- Jackson Parser: com.fasterxml.jackson.core:jackson-databind:2.8.5
- com.fasterxml.jackson.core:jackson-core:2.8.5'
- 'com.fasterxml.jackson.core:jackson-annotations:2.8.5'
- Exo Player: 'com.google.android.exoplayer:exoplayer:r1.5.3'

Testing dependencies:

- Test runner: android.support.test.runner.AndroidJUnitRunner
- JUnit 4.12

16.3 Front End Development

Development dependencies

- AngularJS CDN version 1.5.9
- AngularJS Route CDN version 1.5.8

16.4 Backend Development / Server

Development dependencies

- PHP current version is 5.3
    - Must be greater than 4.3.0, but less than 5.3
- mySQL

    - Current version 5.6.32

17.0 Work Breakdown

Seth Winslow

- Database / Backend
- Primary contact with client
- Completed work on milestone reports

Kristen Weber

- Front End Admin and Public Web Site
- Completed work on milestone reports

Rachel Harvey

- iOS Development
- Completed work on milestone reports

Nick Torres

- Android Development
- Completed work on milestone reports

18.0 Conclusions

What a semester it has been. We hit a few bumps through the journey, but we are still thrilled with all the work that we were able to accomplish. We have built a well suited application to be able to last for months and years ahead.