CS3714 iOS Mobile Software Development

Semester Project Report †

Fall 2016

TV Playlists

PATRICK E. GATEWOOD

Department of Computer Science
Virginia Tech
Blacksburg, VA 24061

Date: 10/10/16
Email: PG8wood@vt.edu
Submitted to: Prof. Osman Balci

† This report determines 5% of your course grade. Your semester project app is worth 20% of the course grade.

EXECUTIVE SUMMARY

The TVPlaylists app allows users to create, edit, and share multiple playlists of TV shows. Users may search the web for shows and add them, and search through the shows they have already saved.

This app solves a problem in the current structure of TV-related apps: users typically must conduct multiple searches in order to obtain a specific query. For the types of queries it was designed for, this app makes searching much easier.

TVPlaylists uses a variety of iOS software functionalities in order to accomplish the easiest user experience possible. Some of the technologies used include:

1. Getting data from the TMDB API in order to obtain information about TV shows, actors, and episodes
2. Storing the user's playlists locally after the user has chosen to add data found online to an existing playlist
3. Implementing an iMessage app that allows the user to share their favorite playlists without leaving iMessage
4. Using a System Action share button to allow the user to share playlists on their favorite social media, notes, and other platforms
5. Filtering search results for a fine-tuned search

The app makes queries to an online database and filters results, then allowing the user to browse their requested data or save it to their local device.

TABLE OF CONTENTS

EXECUTIVE SUMMARY **2**

1. SOFTWARE LIFE CYCLE **1**

2. MY OBJECTIVE **2**

3. PROBLEM SPECIFICATION **2**

3.1. WHAT IS THE PROBLEM? 2

3.2. WHY IS THE PROBLEM IMPORTANT TO SOLVE BY ENGINEERING AN iOS APPLICATION? 2

3.3. FUNCTIONALITIES TO BE IMPLEMENTED BY THE PROPOSED APP 3

3.4. NEW iOS FEATURES TO BE IMPLEMENTED BY THE PROPOSED APP 3

3.5. DESCRIPTION OF THE EXPECTED FUNCTIONALITIES OF THE PROPOSED APP 3

3.6. COMPLEXITY OF THE PROPOSED APP 5

4. REQUIREMENTS SPECIFICATION **6**

5. ARCHITECTURE SPECIFICATION **6**

6. DESIGN SPECIFICATION **6**

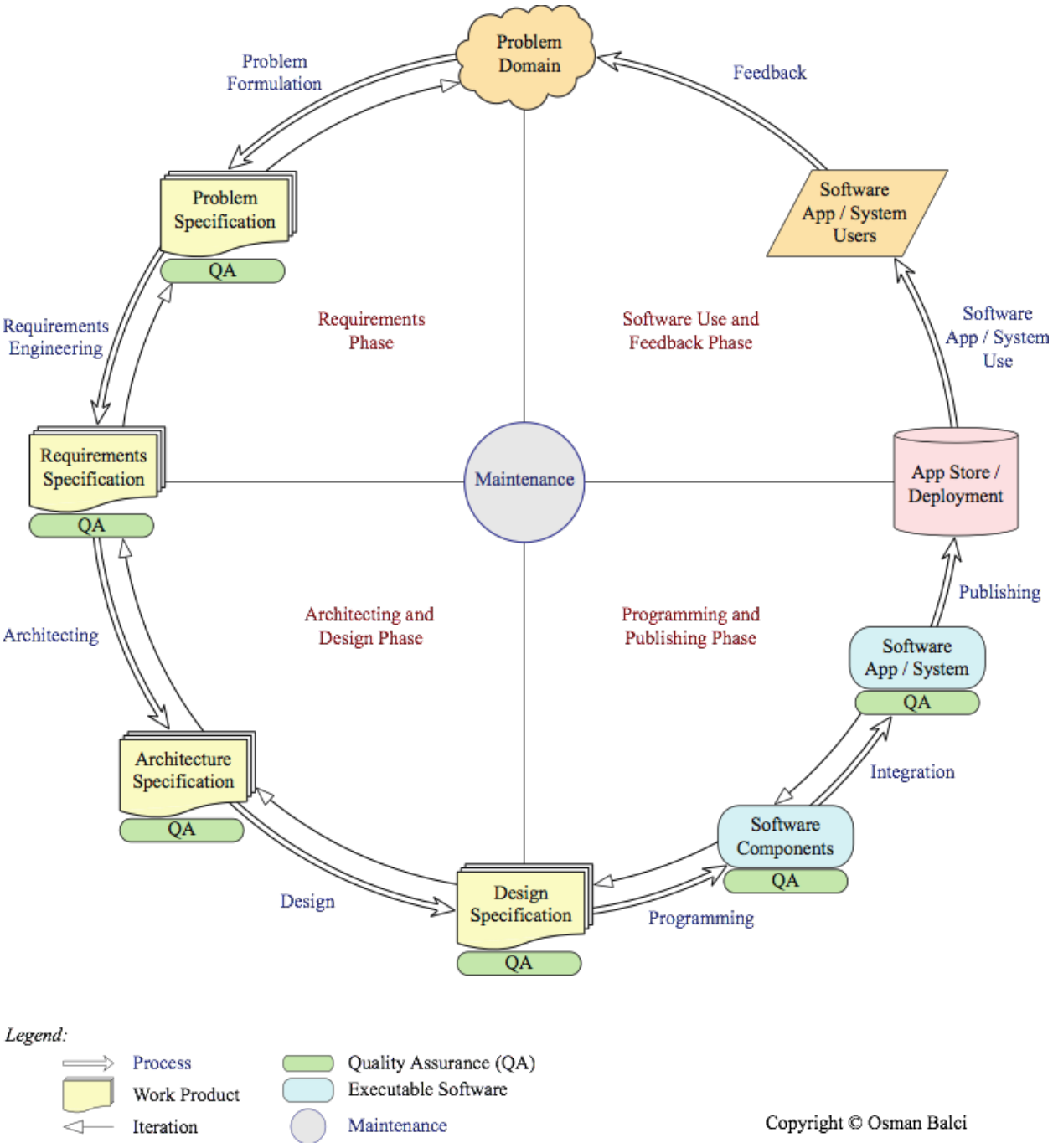
7. DELIVERED SOFTWARE **8**

8. CONCLUSIONS **18**

REFERENCES **18**

1. SOFTWARE LIFE CYCLE

A **good software engineer** develops software by following the software life cycle shown below.



A **programmer (hacker or ad-hoc developer)** develops software by looking at the problem and directly coding in an IDE. This approach is known as the **Build-and-Fix Approach**, which must never be used!

2. MY OBJECTIVE

The objective of my iOS semester project app is to demonstrate how capable I am in engineering an iOS app to solve a problem. The app is created for the purpose of showing how complex and different functionalities and iOS features I am capable of developing.

3. PROBLEM SPECIFICATION

This section provides an overview of why this app idea is relevant and what issues it solves.

3.1. What is the problem?

While there are many TV search apps available, none allow users to explicitly set favorite shows and search through those shows for specific episodes and create playlists from the returned data. For instance, in order to compile a playlist of one's favorite TV Christmas specials or all episodes of one's favorite shows that Neil Patrick Harris appears in, one must use a variety of services or manually comb through a lot of data.

3.2. Why is the problem important to solve by engineering an iOS application?

Engineering an iOS application to solve this problem will save users time, allow them to more easily find what they are looking for and share this information with their friends.

3.3. Functionalities to be implemented by the proposed app

My proposed app shall implement the following 5 functionalities:

Functionality Number	*Description of the Functionality to be Implemented*
1	Using Wikipedia or IMDB API to get lists of TV shows and episodes.
2	Storing users' favorite TV shows locally using plist files.
3	~~Enabling the user to take photos / choose from Camera Roll to provide cover images for the users' lists of shows.~~ This functionality was removed in favor of focusing development time on offering a more robust search filtering experience
4	Enabling the user to share their playlists to other users using the System Action (share) button.
5	Enabling the user to send their playlists to others in iMessage using an iMessage app extension.

3.4. New iOS features to be implemented by the proposed app

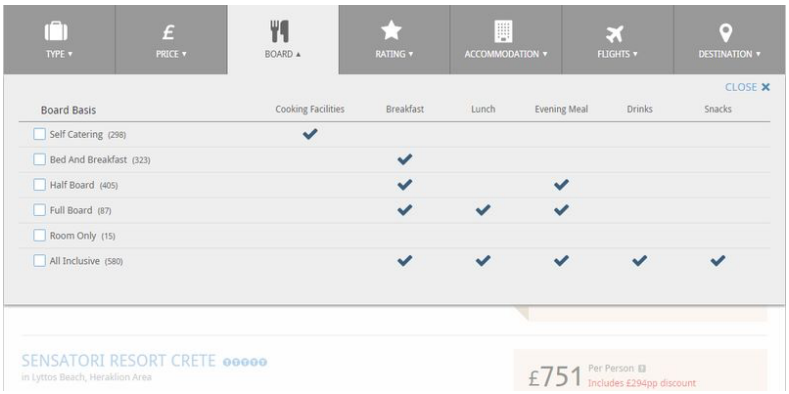
This app will include an iMessage app extension. This allows users to access features of the standalone app without having to leave Messages. Like other iMessage app extensions available in iOS 10, requires the recipient to download the iMessage app in order to view/interact with it.

3.5. Description of the expected functionalities of the proposed app

This app will require several different views and will pull data from multiple sources. The bulk of the app will be the processing required under the hood: finding the correct API to pull data from and arranging that data into a clean UI will be a complicated process.

The app will allow users to select their favorite TV shows by searching for them and saving them to lists (i.e. favorites), and then search all episodes of those shows with given criteria. ~~The shows will be listed in either a grid view or a list view depending on the user's preferences.~~ The shows will be listed in a horizontally scrollable view in order to allow the user to quickly navigate their playlists and shows.

The user will be able to add search criteria and filter search results via a search filtering UI. This UI will persist as the user scrolls through results. It will be similar to a mobile version of the filtering tabs shown in the image below, and it will be horizontally scrollable if more filters are available than screen space.



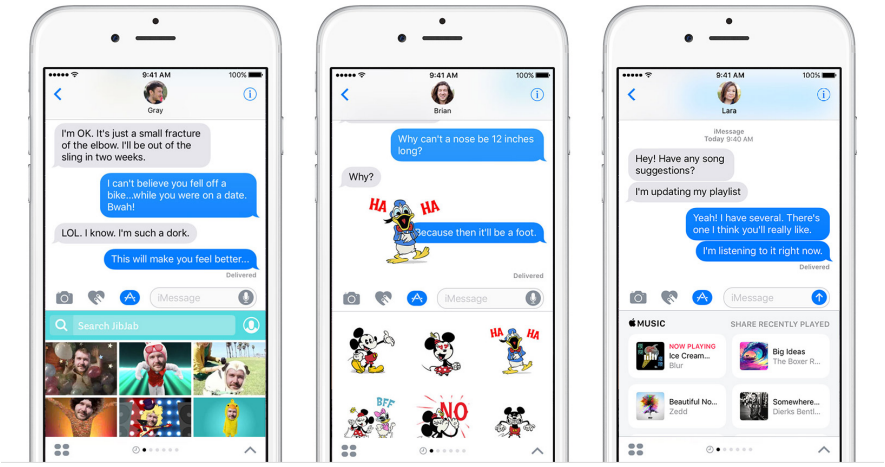
The search filtering will be where the bulk of the application lies, as the app filters through seasonal (or seasonal) episodes, episodes featuring selected actors, keywords in the episode description, and other filters that are currently TBD. The app will need to pull data from a variety of sources and comb through the data quickly.

The user's favorite shows (or other lists created) will be stored locally on the user's device using a plist file. Each list will contain a cover image as designated by the user, the shows, seasons, and episodes. Information for each episode will be loaded upon request by the user should they choose to explicitly view information regarding an episode by tapping it rather than searching. If available, Rotten Tomatoes ratings will be embedded in the episode information.

After a search, the user can elect to create a new playlist from the given search results. This list will be added to the current user's lists and will be displayed on the home screen of the app.

The user will also be able to share their playlists with other users using a system System Action (share) button. Sharing with other users will convert the TV playlist into a simple unordered list for sharing via Notes, email, etc.

Finally, for a more robust sharing option (note: not through the Share Action button) the app will include an iMessage extension that will allow the user to send playlists to their friends. The iMessage extension will require a considerable amount of development and will send playlist data from one device to another through the iMessage client. As with current (non-stock iOS) iMessage apps, the recipient will need to download the iMessage app in order to view the message attachment.



iMessage apps are new in iOS 10, so I will need to do a lot of research on how to implement the app extension, and I may run into problems that are either currently unsolved or that are widely unknown by the iOS development community due to iMessage apps being very new.

3.6. Complexity of the proposed app

This app will have more moving pieces than any tutorial or assignment we have done in class as of yet. There will be many screens and data will be pulled from more places, but most importantly, the data will be *manipulated* in a way that allows it to be shown to the user in a clean UI.

The iMessage app will need a way to communicate with the base app in a TBD fashion, and protocols such as this are as-of-yet uncovered in class.

Finally, the UI will have many more elements than that of any assignment done in class, and all of those elements will need various connections in order to work together.

Due to this complicated compilation and manipulation of data, this app is worth at least 25% of the course grade as compared to the other assignments and the midterm completed in the course.

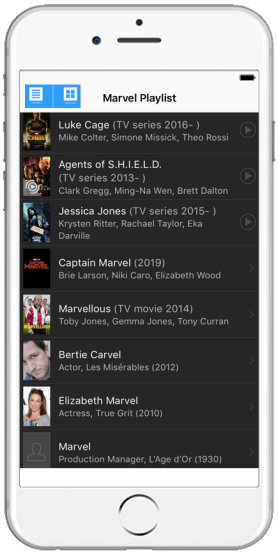
4. REQUIREMENTS SPECIFICATION

- ~~The app shall support both list and grid views of playlists.~~
- The app shall support horizontally-scrollable selections in playlists
- The app shall allow the user to create new playlists and add shows/movies to them.
- The app shall allow the deletion of playlists and shows.
- The app shall work in portrait mode only.
- The app shall allow users to tap the Share button to share playlists.
- The app shall include an iMessage app extension.
- The app shall allow searching and filtering by actors, holidays, and episode names.
- The app shall allow searching through both saved shows and all known shows online.
- ~~The app shall allow users to open the camera for cover art if desired.~~
- The app shall have a scrollable interface.

5. ARCHITECTURE SPECIFICATION

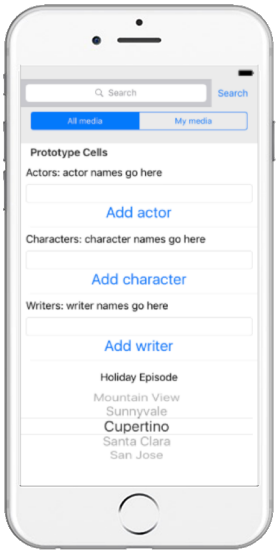
This app will use a Client-Server Architecture. The app will store a minimal amount of data about the user's playlists and shows locally and will load additional information on-demand. Basic searches will be conducted on the iOS device, but any search using filters will make a call to a server computer on which the actual searching/result filtering algorithm(s) will be run.

6. DESIGN SPECIFICATION



The playlist screen will display information about each title and minimal information about the title. If the user taps the row, they will see more information about the title they tapped.

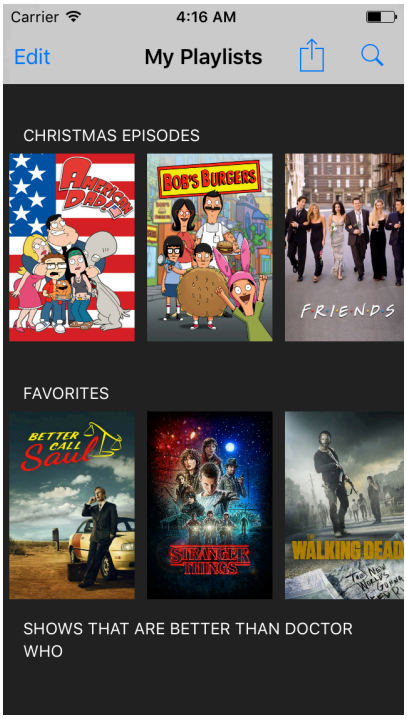
Tapping the search bar will reveal a scrollable table view with many filters available. The user can add as many or as few filters as they like: the search will be conducted with any number of filters when the search button is tapped.

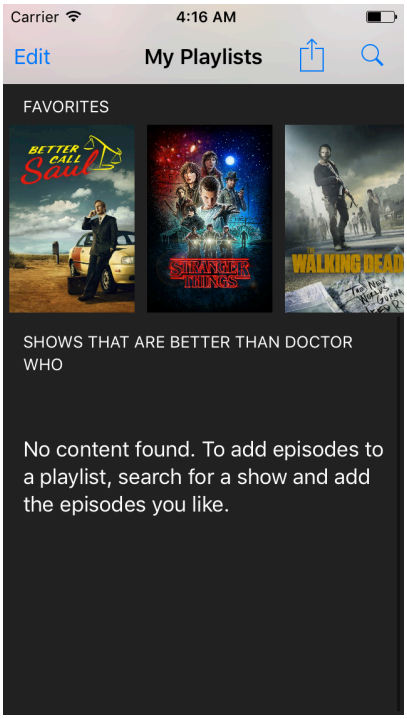


The iMessage app will allow users to share a playlist with another user who owns the app. If the user's friend does not own the app, they will need to download it in order to receive the playlist. Otherwise, the user can elect to send the playlist as plaintext.

7. DELIVERED SOFTWARE

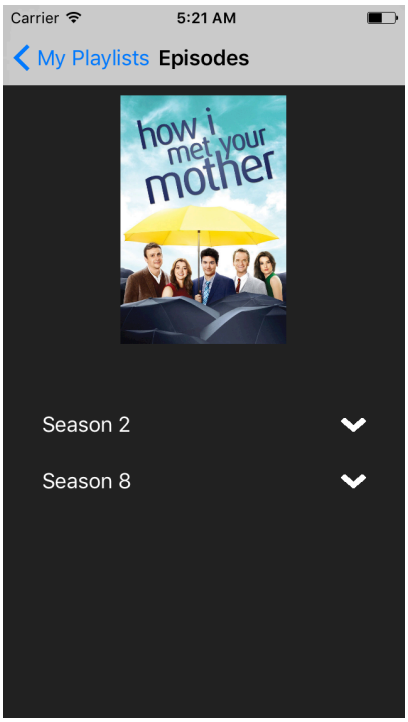
TVPlaylists allows the user to create, edit, and share playlists of TV shows. Upon first opening the app, the user is shown a default set of three playlists. The third playlist is empty, and contains a hint message to help the user figure out how to add content to a playlist.





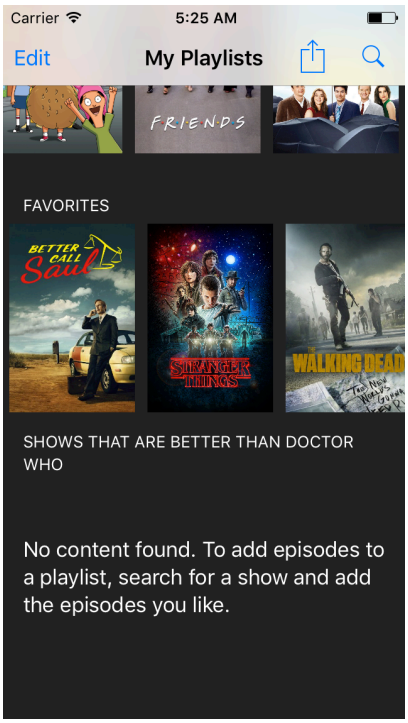
To view the contents of a playlist, simply swipe left or right on the episodes. All shows in a playlist will be shown on this main screen.

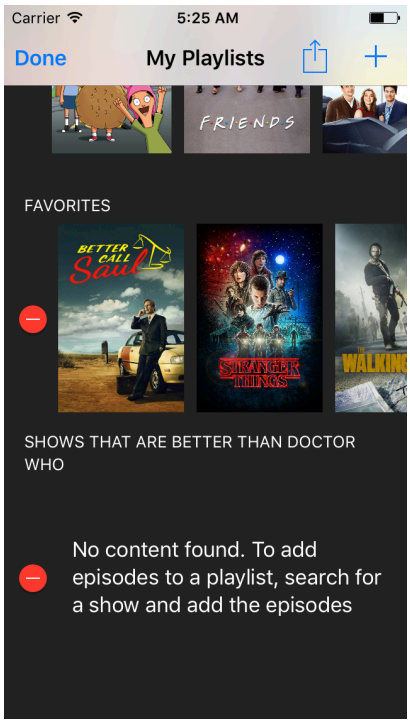
In order to view which episodes of a show are contained in a playlist, tap on the show's cover photo. This will reveal the Episodes view.



The Episodes view initially displays all seasons of a show and the show's cover photo. Tapping on a season reveals all the episodes of that season that are contained in the selected playlist. Each TableViewCell displays the episode name, description, guest stars (if any), and the episode's rating.

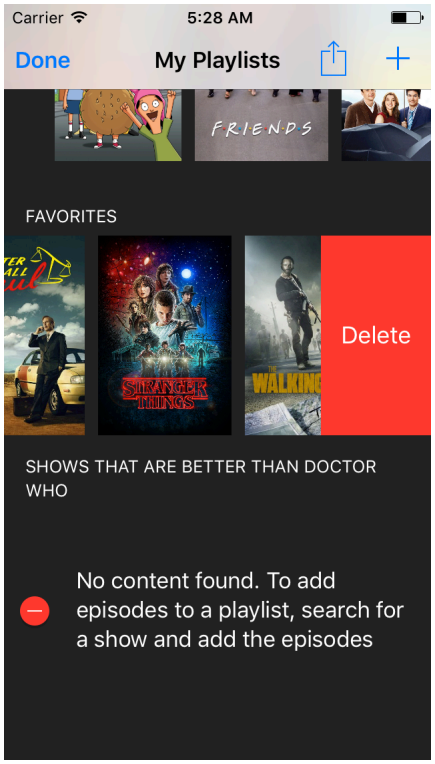
To edit or add a new playlist, the user returns to the home screen by hitting the "< My Playlists" button. Then, the user taps the "Edit" button in the top left corner of the Navigation Bar.

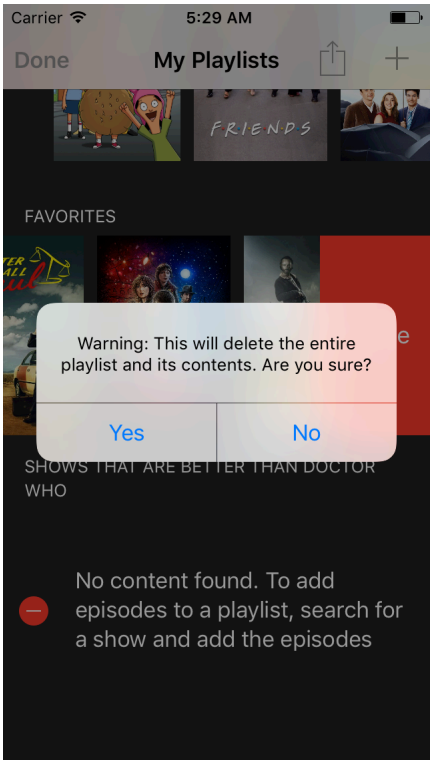




Tapping “Edit” Changes the bar button items in the Navigation Bar. The Edit button becomes a Done button, and the Search button becomes a “+” button. From here, there are a variety of operations the user can perform.

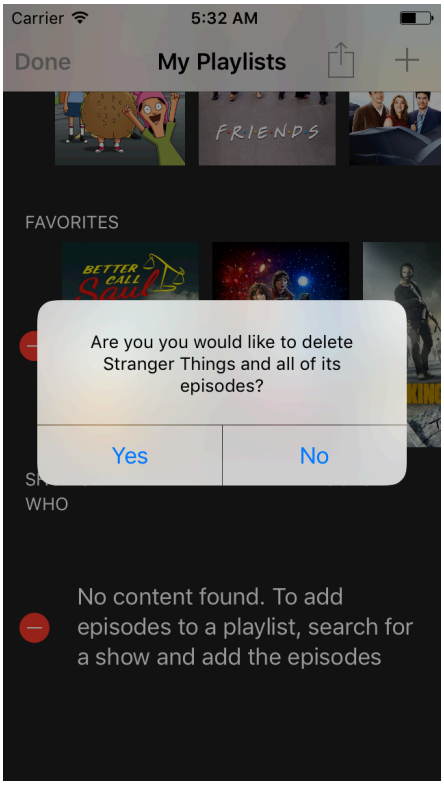
If the user would like to delete an entire playlist, the user taps the “-“ icon, and then hits the delete button. A warning message will display and ask the user to conform that they really want to delete the entire playlist.



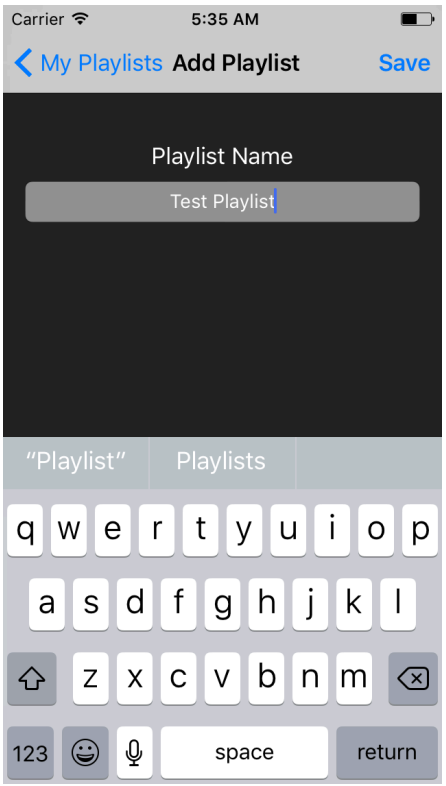


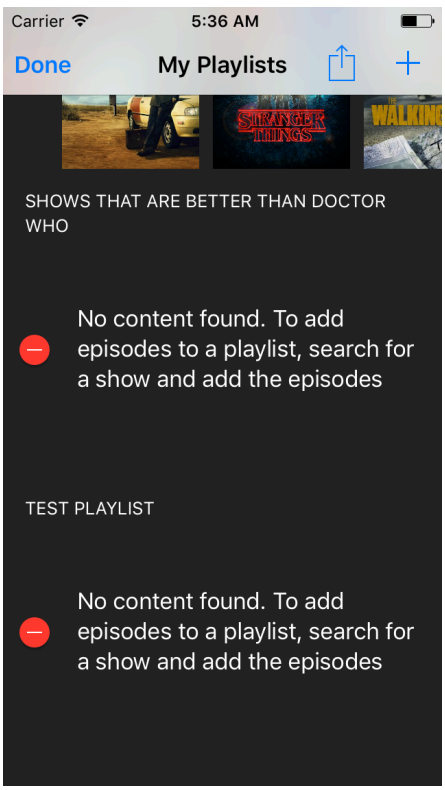
As expected, tapping “Yes” will delete the playlist, and tapping “No” will cancel the operation. The playlist will be deleted from the dictionary object containing playlist data (which is written to the device’s Documents directory when the app resigns its active state), so the playlist will be deleted forever and will not display if the app is terminated and relaunched.

If the user would like to delete a single show from a playlist instead, they can tap on a show’s cover image while the TableView is in edit mode. Upon confirmation from the user, the show will be removed from the playlist.

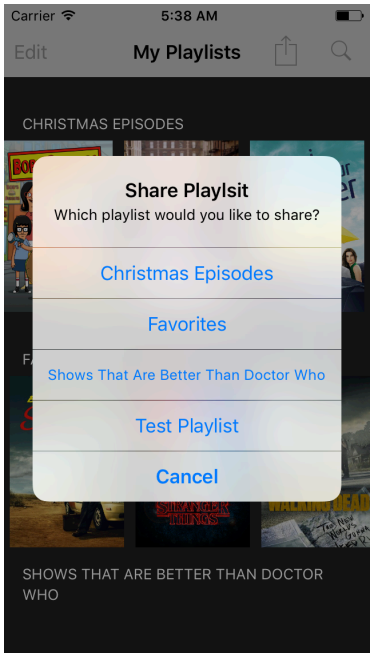


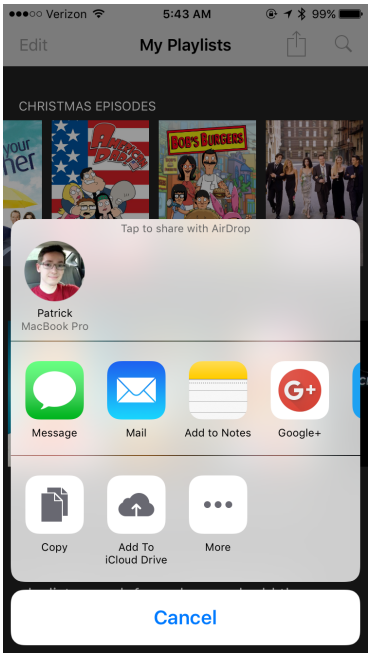
Edit mode is not just for deletion. If the user would like to create a new playlist, they'll tap the + icon in the upper right corner of the Navigation Bar to display the Add Playlist view. After the playlist name is entered (and at least one character must be entered), the playlist is saved to the device and will display in the home view.

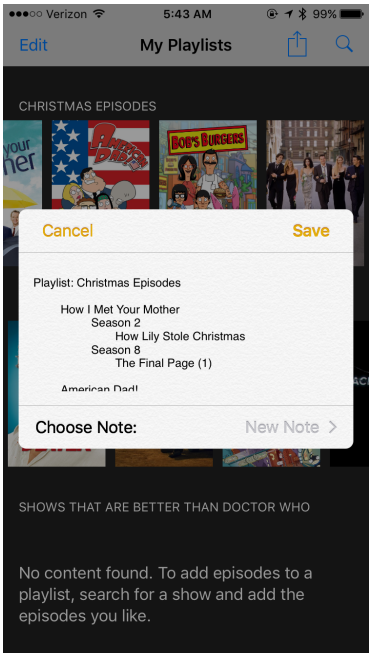




Another feature available on the home screen is the ability to share a representation of the user's playlist. If the user taps the System Action (Share) icon just to the left of the Search icon, the user will be asked which playlist they would like to share. Note that for any apps to display, the user must be signed in with a valid Apple ID and have social media accounts set up. **If the app is tested with the iOS simulator, no social options will display.** From here, the user can choose which app they would like to share a String representation of the playlist with, AirDrop it to a nearby friend, copy to their clipboard, etc.

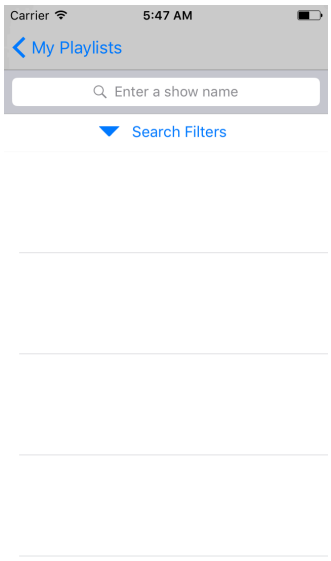


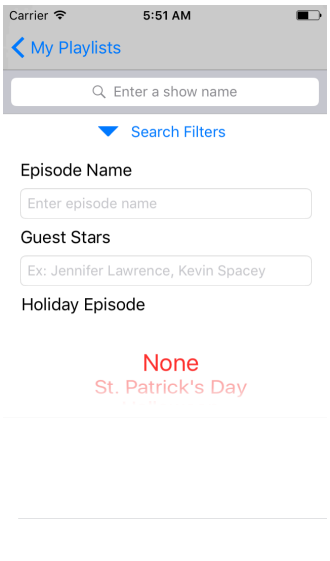


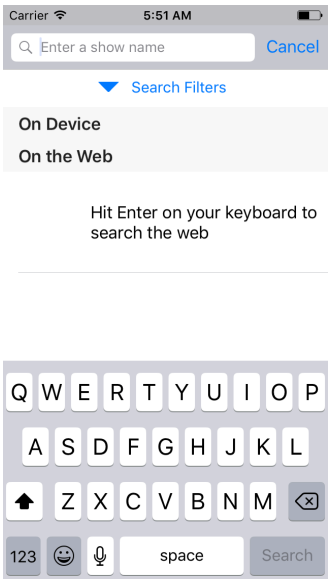


Once the user is done viewing the preset shows on their device, they may want to add additional shows to their playlists. This is accomplished by tapping the Search icon to open the Search view. In the search view, users can search episodes on their local device, or on the web. The first thing the user sees is a blank TableView with a Search Bar in the header, and a Search Filters button. Tapping the Search Filters button will expand the top TableViewCell into a view that allows the user to enter various filters when conducting a web search.

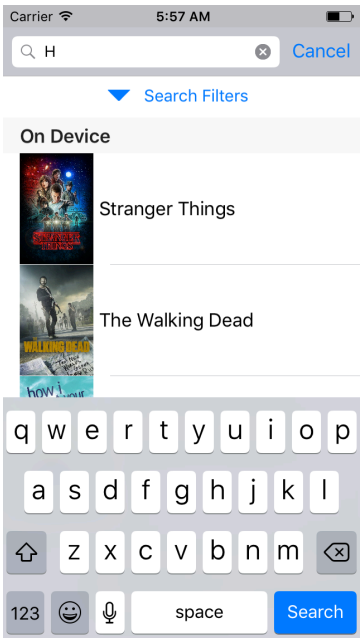
When the user taps in the Search Bar, the filters are hidden by default to save screen space. They can be reopened at any time during the search if the user wishes. Two section headers appear, denoting results found locally and results that will be found when conducting an online search. A hint message appears, telling the user they must tap "Search" or "Enter" on their keyboard to initiate the online search.

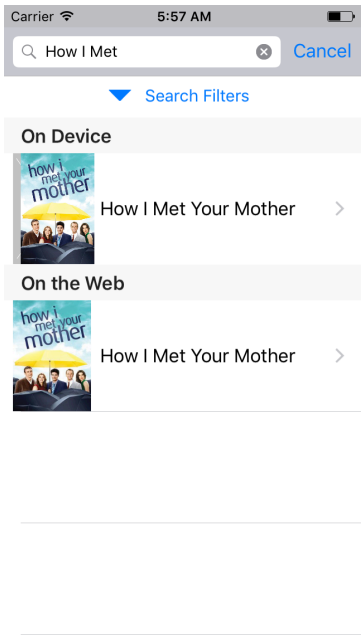


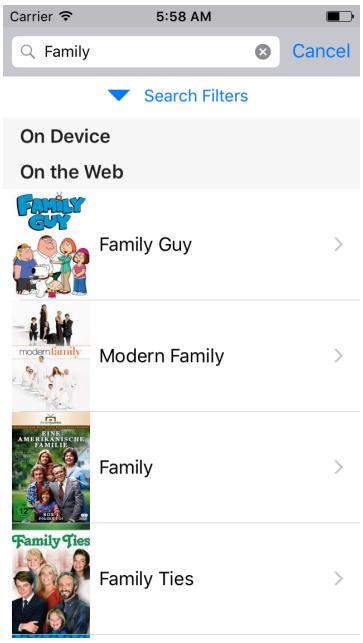




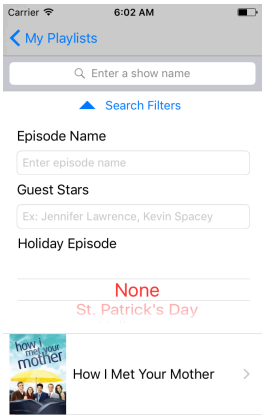
Local results will be updated in realtime with each character the user types, but online searches are not conducted until the user taps "Search" (note the differences in the search string in the screenshots below)

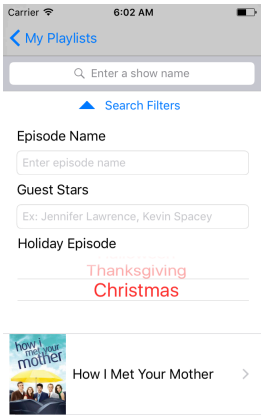


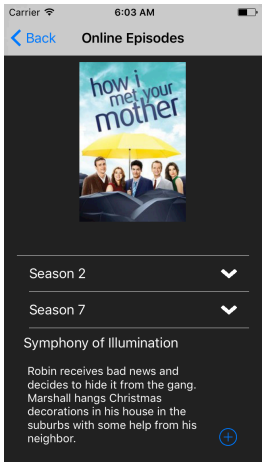




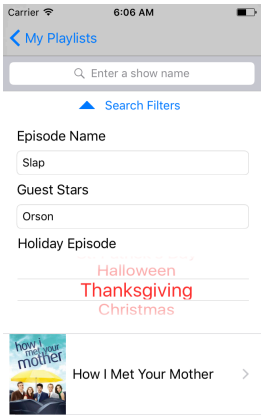
Once results are on the screen, the user can tap on the local result or the web result. The local result will display the episodes stored locally on the user's device, and the web result will display all seasons and episodes if no filters are passed, or only matches if filters are passed.

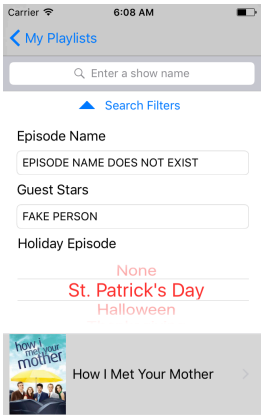


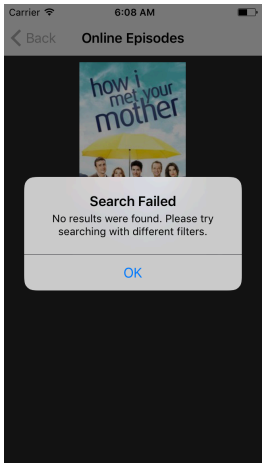




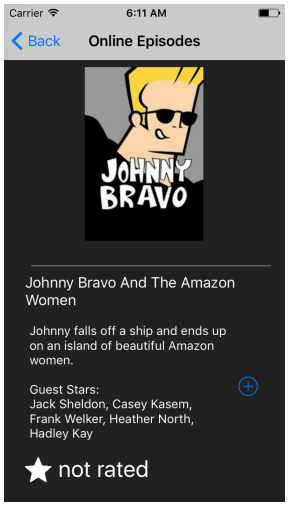
Note that any combination of filters can be selected. If no results were found, an error message displays.

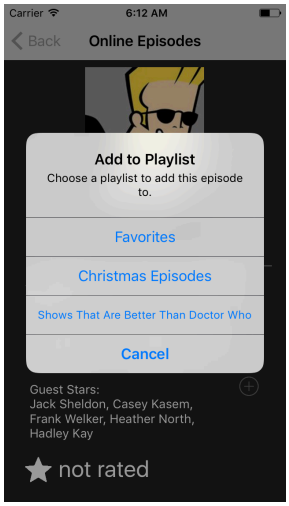


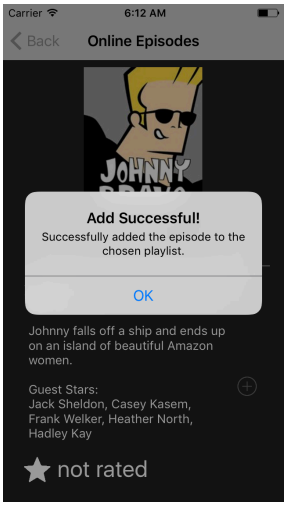


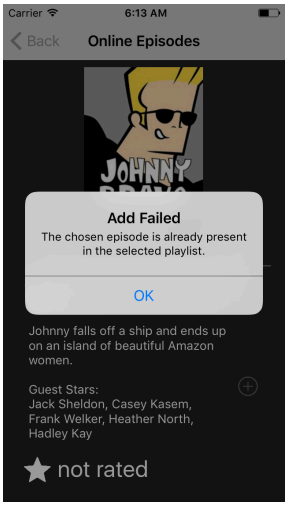


When viewing an episode, the user may attempt to add the episode to an existing playlist. This is accomplished by tapping the "+" icon on the right of the TableViewCell. A popup displays, asking the user to choose a playlist to add the episode to, or cancel the operation. Upon successful addition, a success message displays. If the selected episode is already in the selected playlist, an error displays to prevent the user from adding duplicate entries into their playlists.

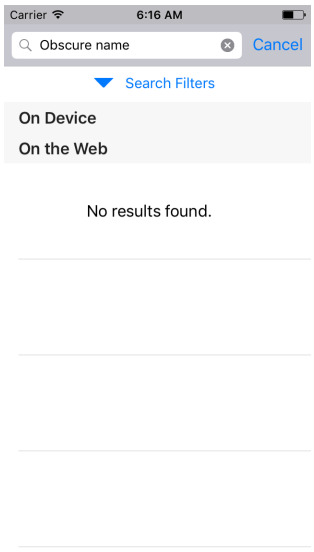


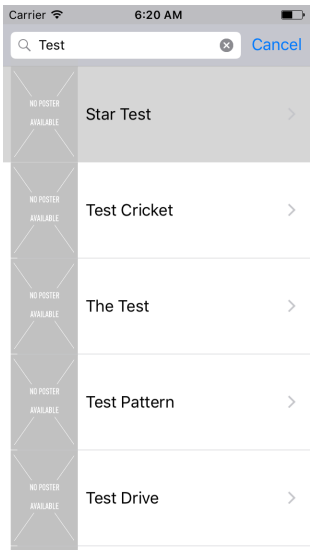


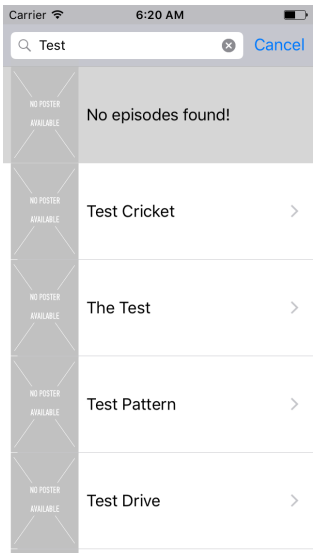


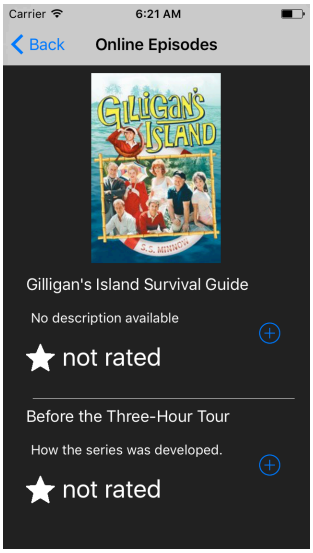


When working with APIs, sometimes data will not be present when expected. In these cases, the app responds by adding a set of helpful default labels where data was expected but not found. For instance, if no shows are found by the name searched, a message is displayed. If a show is found, but no episodes are returned, the app does not bother loading the episode view and instead displays that no episodes were found. If a season is empty or expected data is missing, the app states so.





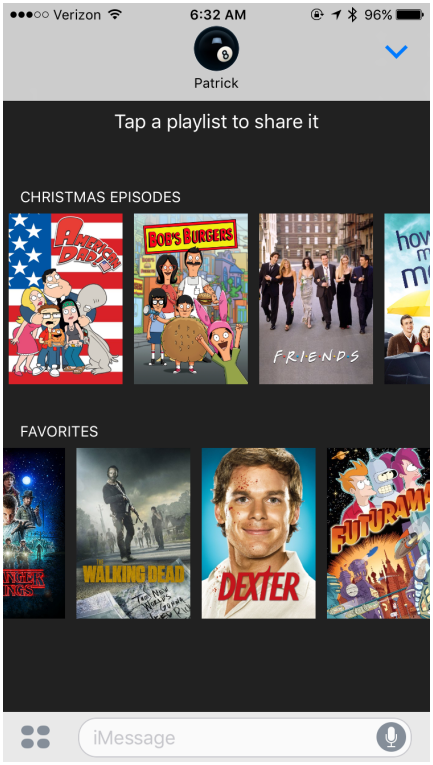


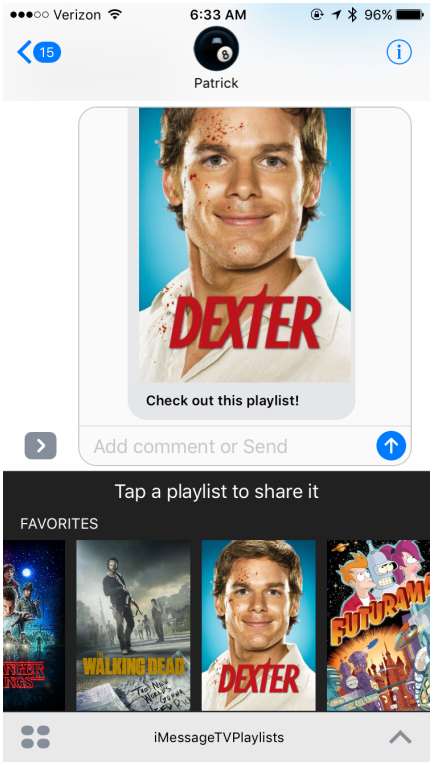


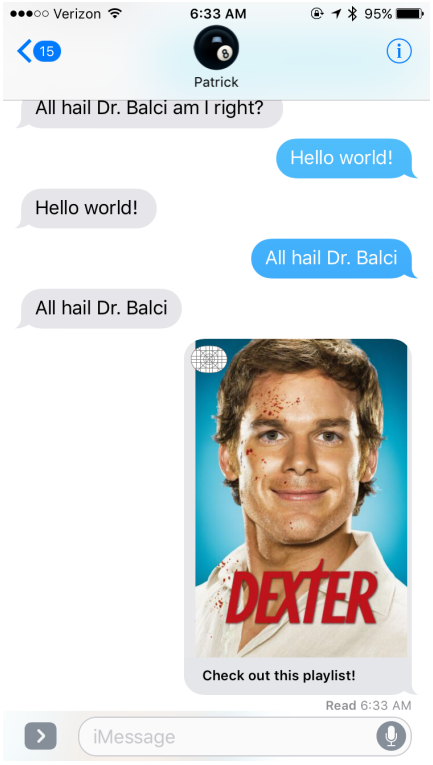
Finally, this app includes an iMessage app, however it is not formatted in the typical way iMessage apps are. The Virginia Tech iOS Developer Program does not support students adding apps to App Groups. Because of this, it is impossible to share data between apps, as this is the only Apple-sanctioned way to share data between apps. Because I was unable to add App Groups to my Apple ID, I made a standalone iOS application. Typically, an app like this would share its data with its parent app.

The iMessage app displays where the keyboard typically displays upon launch. From there, the user can choose a playlist to share directly, or expand the app into a "fullscreen" iMessage app. Upon tapping a playlist, the app creates a message with the contents of the playlist, chooses a photo to accompany the playlist, and sends it to the recipient.

Anyone can receive the message, however the user must have the iMessage app installed in order to receive the playlist. Otherwise, tapping the iMessage will bring them to the iMessage App Store.







8. CONCLUSIONS

This project was absolutely invaluable to my education as an iOS software solutions provider. I have never spent so long on a project before, but I loved every second of it. This project and this class as a whole have absolutely given me the skills I need to succeed at my iOS software engineering job that I have secured for the summer at WillowTree Apps.

I would like to thank Dr. Balci for providing me with an unbeatable iOS education this semester, and I look forward to taking his cloud class next semester as a result.

REFERENCES

Apple, Inc. (2016), "iOS Developer Library,"
https://developer.apple.com/library/ios/navigation/index.html

Balci, O. (2016), "CS3714 iOS Mobile Software Development Course Website,"
http://manta.cs.vt.edu/cs3714

<< Reference other sources used in your work. >>