Final Project Documentation

* 1. **Project Overview, Introduction and Requirements Specification**

“The Scaredy Cats Podcast has built up an impressive catalog of podcast episodes, and its listening audience deserves a central hub for everything the podcast entails. Right now people have many locations to stream but some are paid services, and there is also no location to provide extra content relating to the episodes.

I will be changing that with this web-hub for the podcast. This will be the one-stop shop for all things Scaredy Cats Podcast.

* 1. **Introduction**

My name is Bryan Cozean, and I will be the developer of this application. I will be mainly pulling from:

Intro to Web Programming

Business Architecture

Programming Fundamentals II

Business Applications

Data Systems: Programming & Admin

Database Systems: Concept & Design

**1.2/1.3 Project Overview**

This project will allow users to listen to podcasts on the site. It will also allow users to download individual episodes. If users create a profile they will receive an email alert upon the upload of a new episode. Along with the podcast content, users will also be able to view text and picture content that relate to individual episodes.

This iteration of the project will not allow users to favorite certain episodes, and it will not keep track of which episodes the user has watched on the site.

**1.4 Project Requirements**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **User Story** | **Priority** | **Acceptance Criteria** | **Status** |
| **1** | As a user, I would like to view additional content related to the episode I just watched | **A** | The additional content will be readily available on the site | **BackL** |
| **2** | As a user, I would like to listen to the podcast on the website so that I do not have to pay for any apps that would carry it | **A** | There will be functionality to listen on the website away from any app | **BackL** |
| **3** | As a user, I would like to register an account so that I can be alerted when new things are added to the site | **A** | Users will be able to register and receive notifications when new pods are added | **BackL** |
| **4** | As an administrator, I would like to be able to add podcasts to the site | **A** | The admin center for the site will allow admins to add and delete content at will | **BackL** |
| **5** | As an administrator, I would like to send emails to users when new things are added to the site | **B** | When things are added to the site, logic will be in place to send emails to users | **BackL** |
| **6** | As a user, I would like to see episodes related to certain topics | **B** | Each episode will be allocated a certain category that users can scan | **BackL** |
| **7** | As a user, I would like to download episodes so that I can listen to them at my leisure | **B** | Each episode will be downloadable from the data source where held | **BackL** |

**1.5** **Definitions, Acronyms, and Abbreviations used in documentation**

Ongoing…

**2.1 Description of Problem**

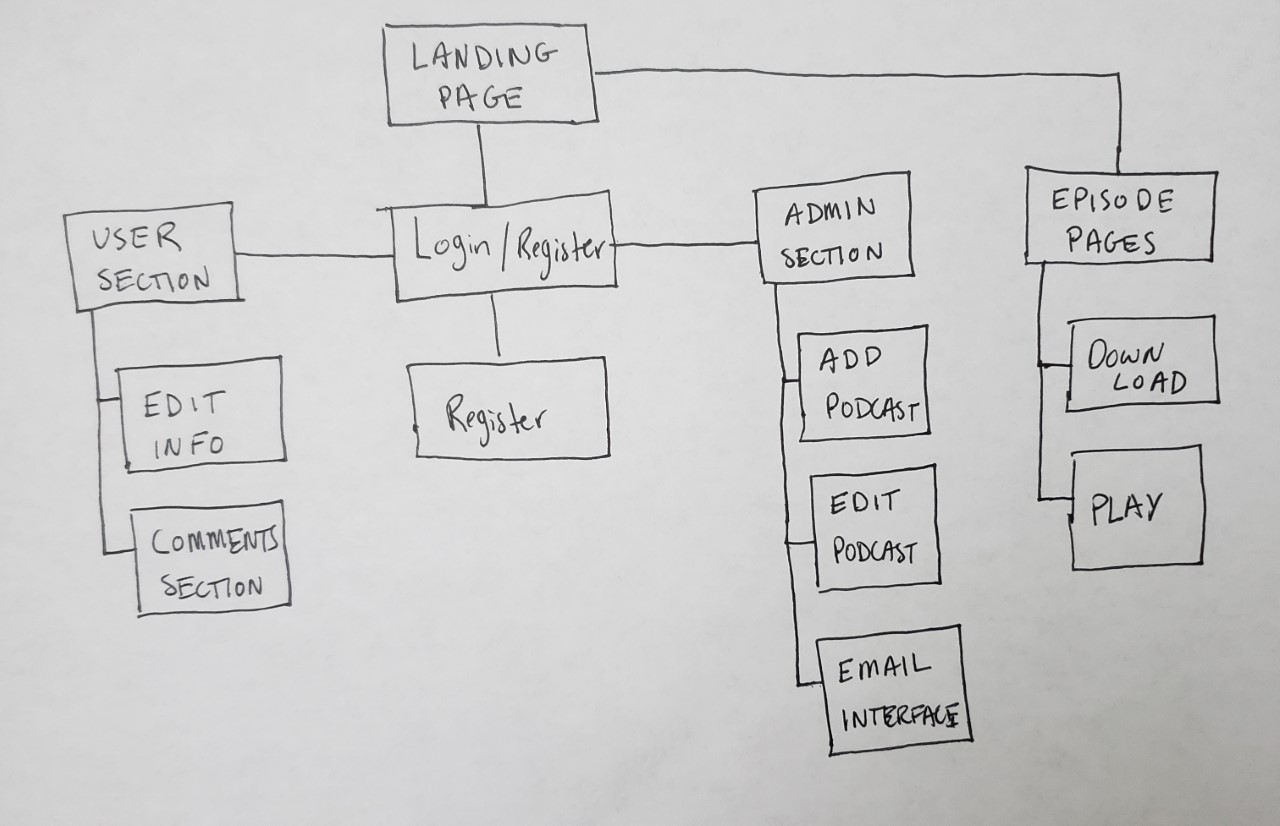
Many things come along with running a podcast. One problem many people have is creating a home for the listeners of their podcast – a place where people can interact and find out more about what they listen to. This is the problem I am aiming to address with this application. When this application is complete and hosted it will serve as a one stop shop for Scaredy Cats everywhere.

Social media does present a space to connect with listeners, but it has its own set of problems. I believe the tandem of social media and our own web platform is the best approach for our listeners.

**2.2 Technologies Used**

|  |  |
| --- | --- |
| **Title** | **Explanation** |
| ASP.NET MVC 5 | I will be utilizing the power of the model-view-controller framework presented with ASP.NET MVC 5. |
| SQL/MYSQL | I will be using a SQL database as part of my data store for my backend. At this point, I believe I will be using the built in SQL development tool in Visual Studio. |
| C# | I will be using C# to connect my backend to my front end. |
| Bootstrap | I will not exclusively be using Bootstrap for my front end, but it is hard for me to see not using it for at least some parts. |
| HTML/CSS | I will also be using html to build part of my views in the application, and I will use CSS to style some of said HTML. |
| Razor Views | I am not sure this warrants its own section or if it is grouped in with the MVC framework, but I will be using Razor syntax, combined with the HTML and CSS, in my Razor Views to make for a dynamic web experience. |
| GitHub | I will be using GitHub for all version control pertaining to this application. |

**2.3 System Architecture**

****

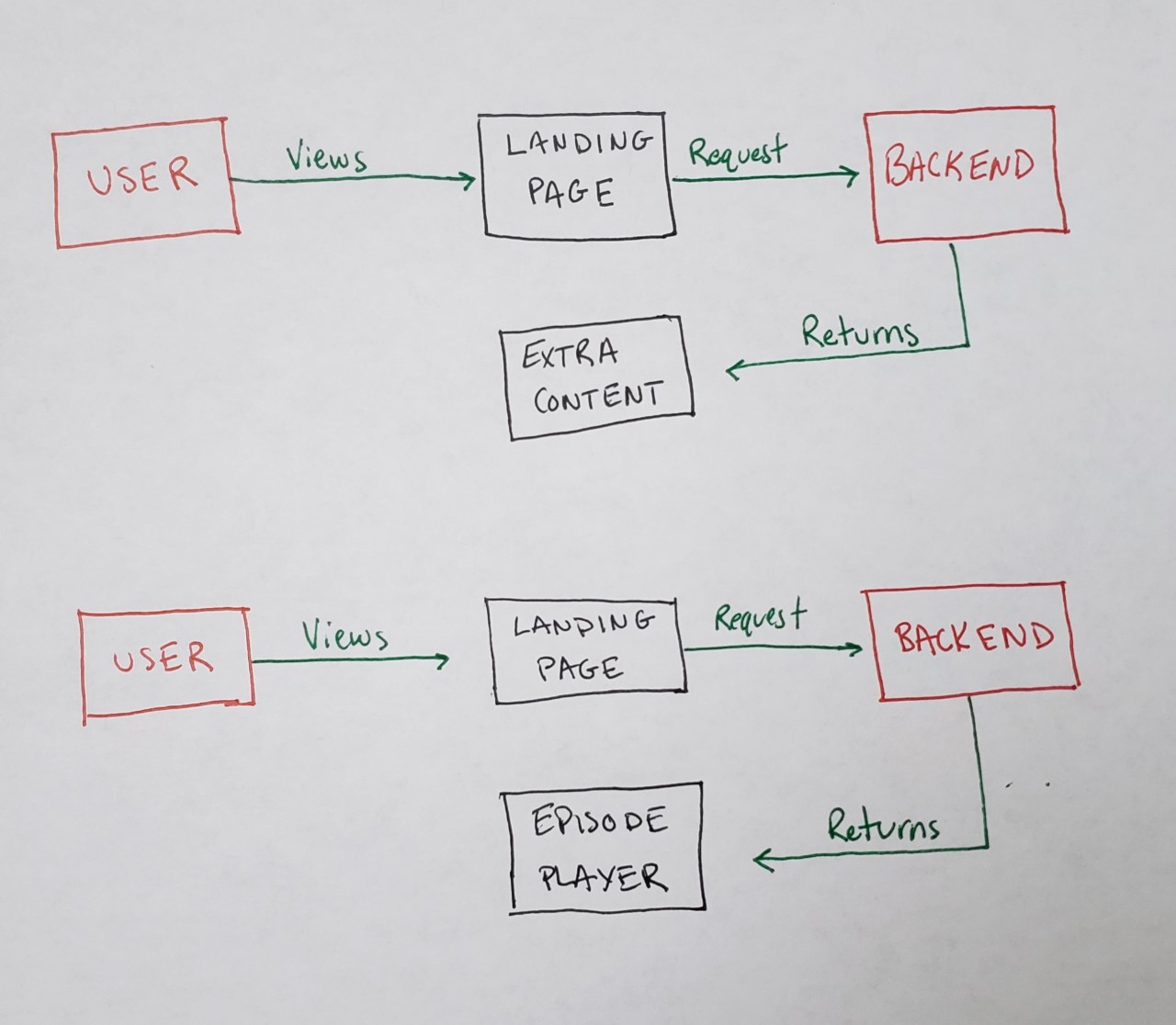
This is my rough concept for my system architecture. You can see that the website will be viewable if you are not logged in and if you are. Whilst doing it I realized this is a pretty basic setup, and the real power behind it will be the ability to add over time. I am imagining only doing 5 or so episodes for THIS project, but as we get all our current and future content up the catalog will become quite robust.

* Correction 5/11/2021
  + The episode page and user section have not been developed at the time of presentation. The landing page, login/register, and admin section are the main branches available to present at this time.

**2.4 Use Case Diagrams and Descriptions**

**USER VIEWING EXTRA CONTENT/**

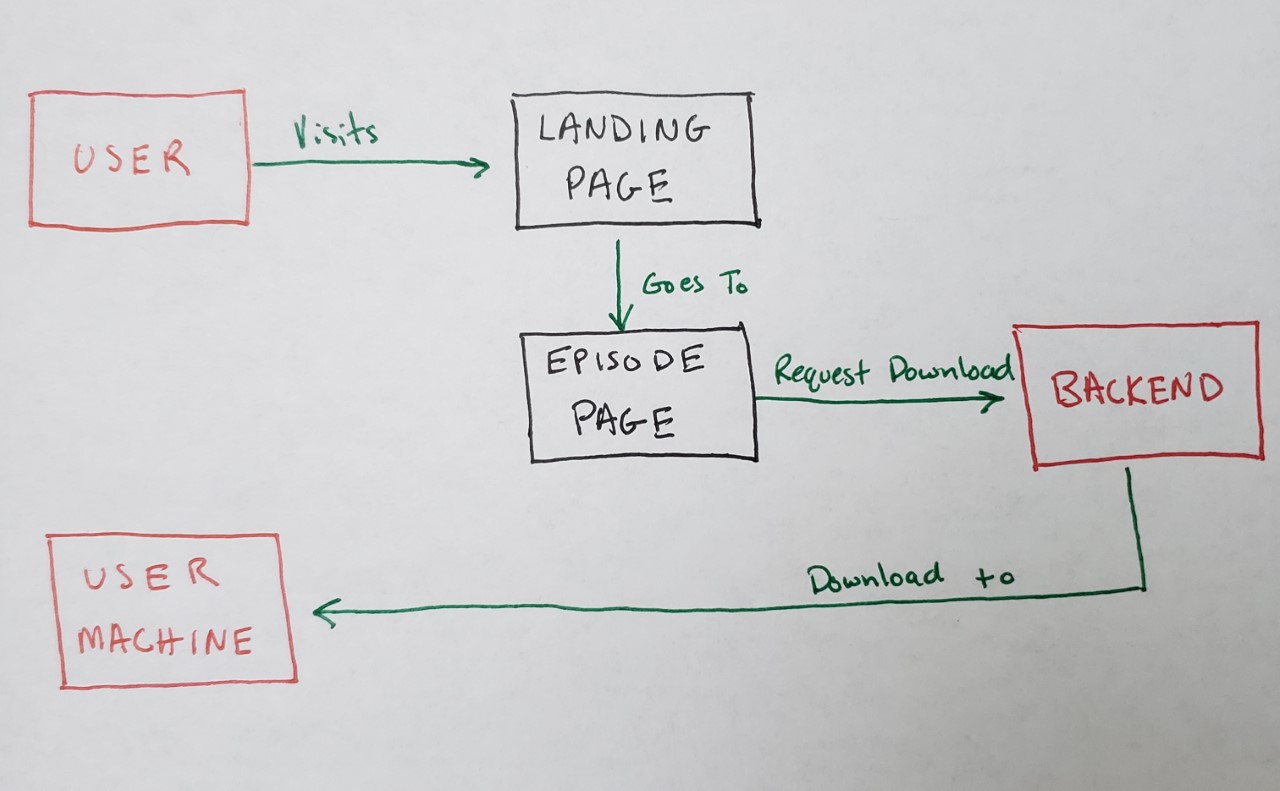
**PLAYING EPISODE**

****

I chose to put these together because they were so similar. This would describe your casual viewer coming to the website and interacting without registering or signing in. It will be simple for people to come to the site’s landing page and play the episode right there or view extra content based on the episode.

* Correction 5/11/2021
  + The extra content page will not be available at the time of presentation

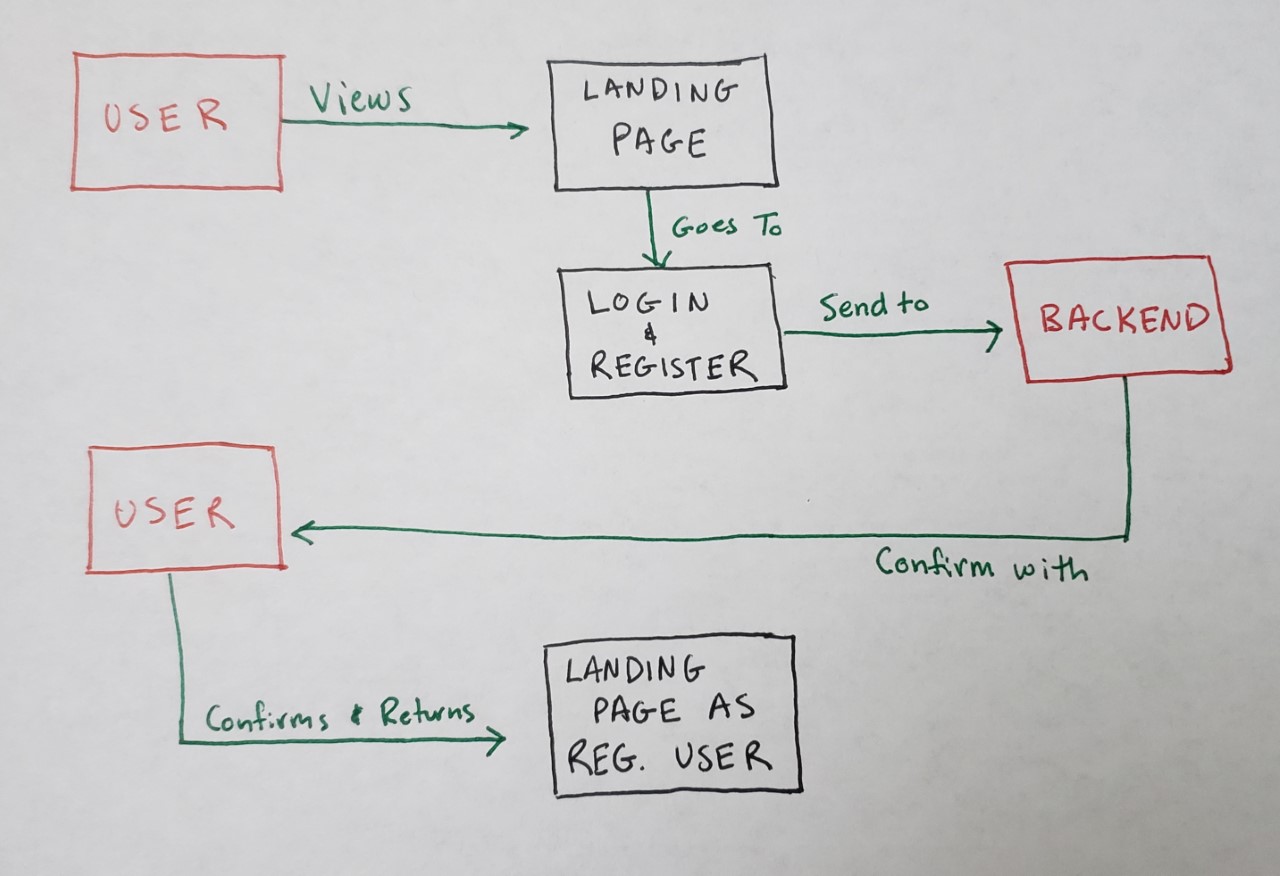
**USER DOWNLOADING EPISODE**



This shows a visitor to the site clicking a download link, triggering the backend, and the backend downloading the episode to the user’s machine.

* Correction 5/11/2021
  + The episode page of this diagram has been rendered not necessary within the development phase. The users can download the file straight from the landing page.

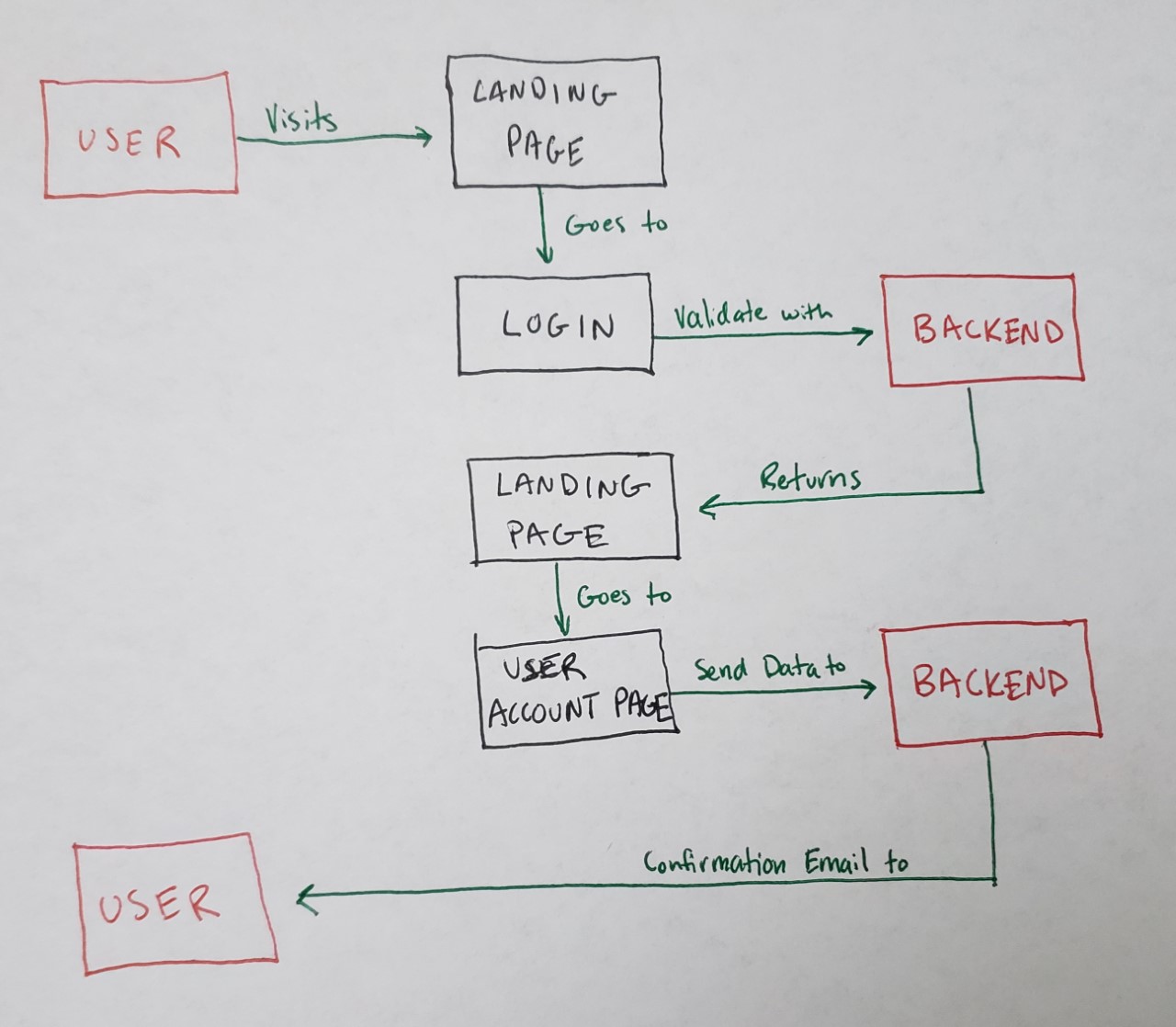
**ADMIN REGISTERING AN ADMIN**



This shows a user registering with the site and the backend sending a confirmation email to them.

* Correction 5/11/2021
  + Throughout the development process the decision was made to only allow admins to create accounts. The users can interact with the website via the email subscription facet on the landing page. The story board above still describes the functionality of the registering correctly. The user is simply swapped with an admin.

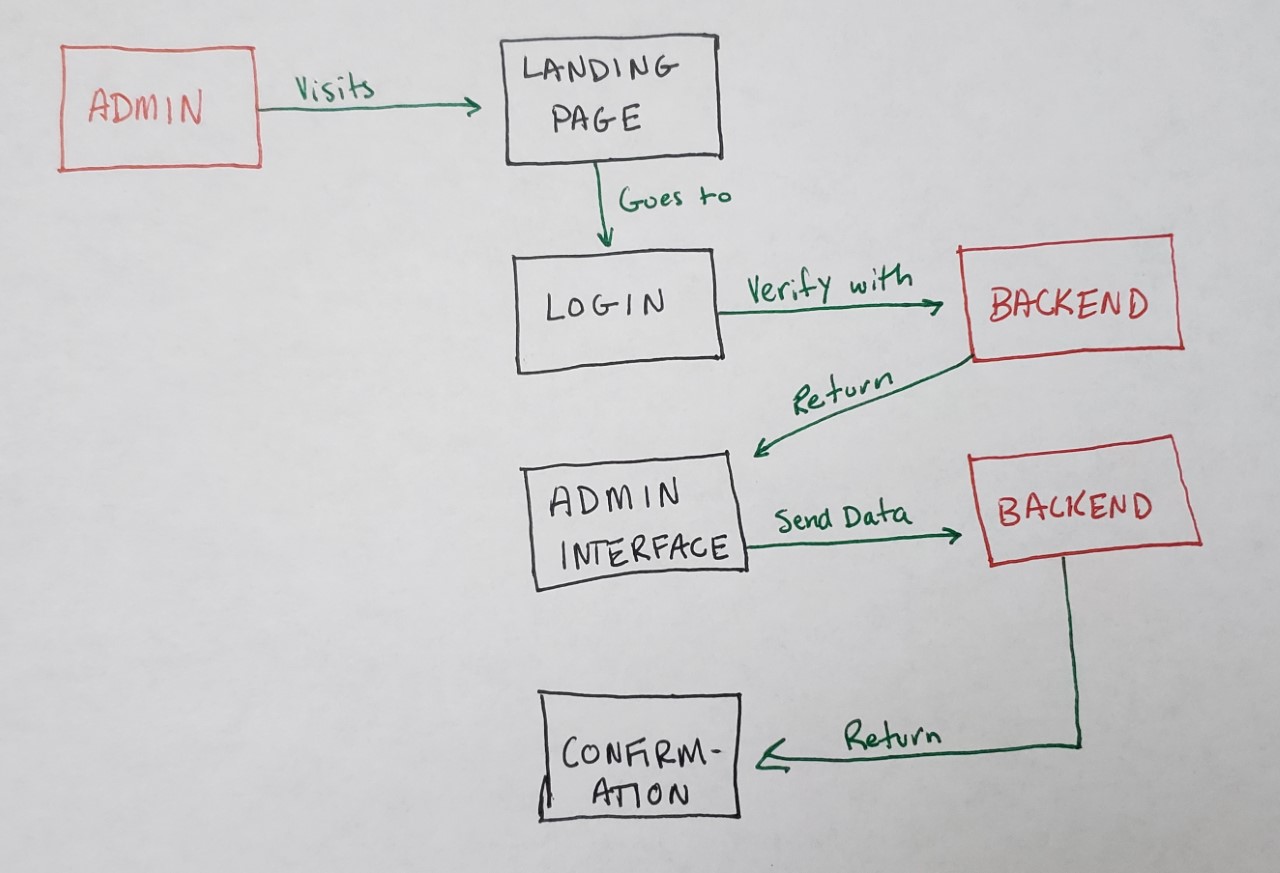
**ADMIN EDITING ACCOUNT DETAILS**

****

This shows a user who wants to edit something in their account settings.

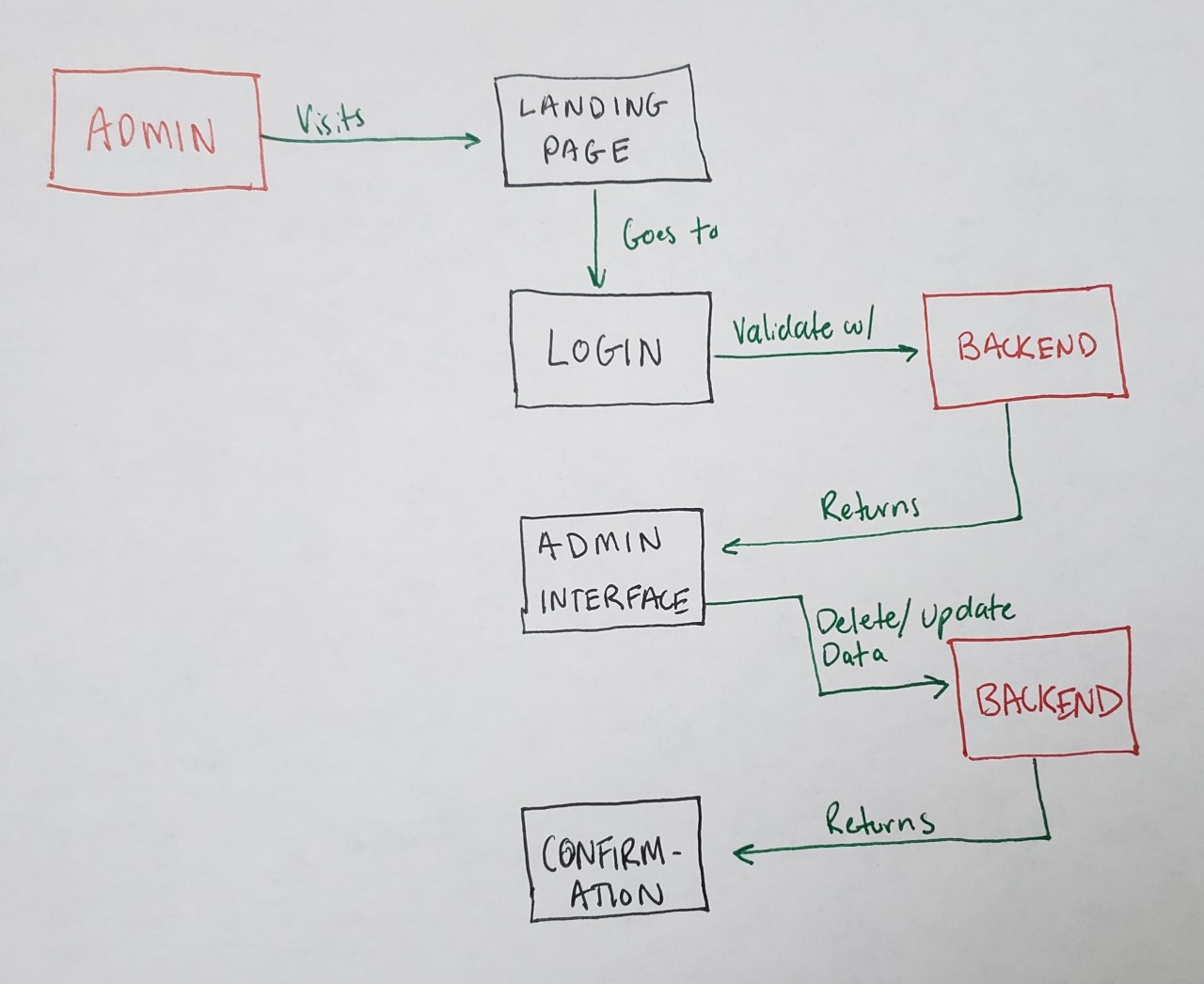
* Correction 5/11/2021
  + This now would describe the admin editing the account details. This function may or may not be available at the time of presentation.

**ADMIN ADDING PODCAST**



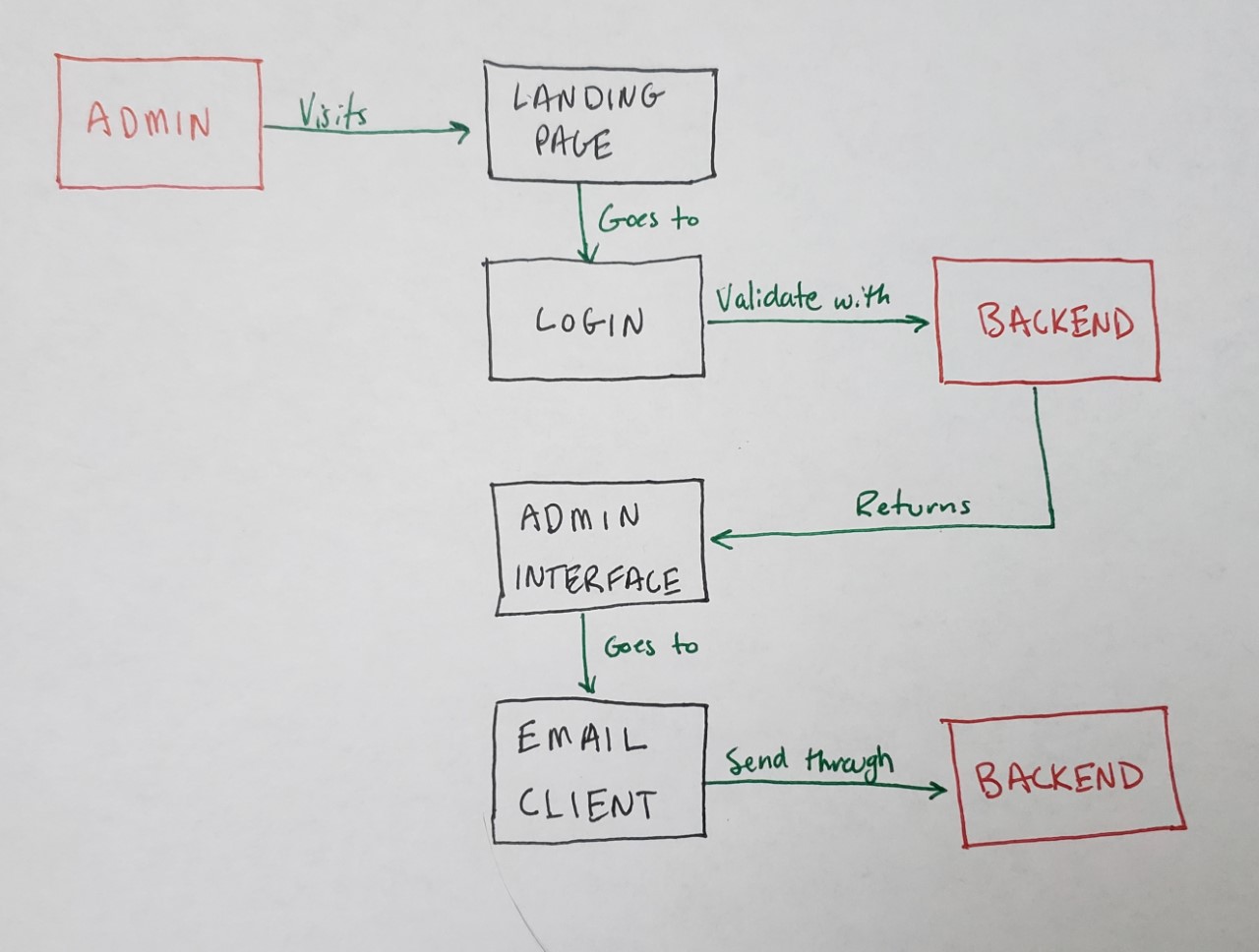
This shows an administrator adding a podcast to the site’s memory stores and returning a confirmation message that it was successful. Now that I am looking at it, this could also describe the admin adding anything to the site – a user for example.

**ADMIN EDITING/DELETING PODCAST**



This is describing an administrator changing the details or deleting a podcast episode, and as mentioned before this could work with any update or deletion process.

**ADMIN EMAILING USERS**



This describes the admin sending out an email when there is a new upload to the site. While I am not entirely sure how I am going to implement this just yet, I believe this is a rough estimate as to what the process will look like. I will obviously have to dig into the details or potentially emailing certain groups of users depending on their wanting to be alerted.

* Correction 5/11/2021
  + The email functionality will not be available at the time of presentation.

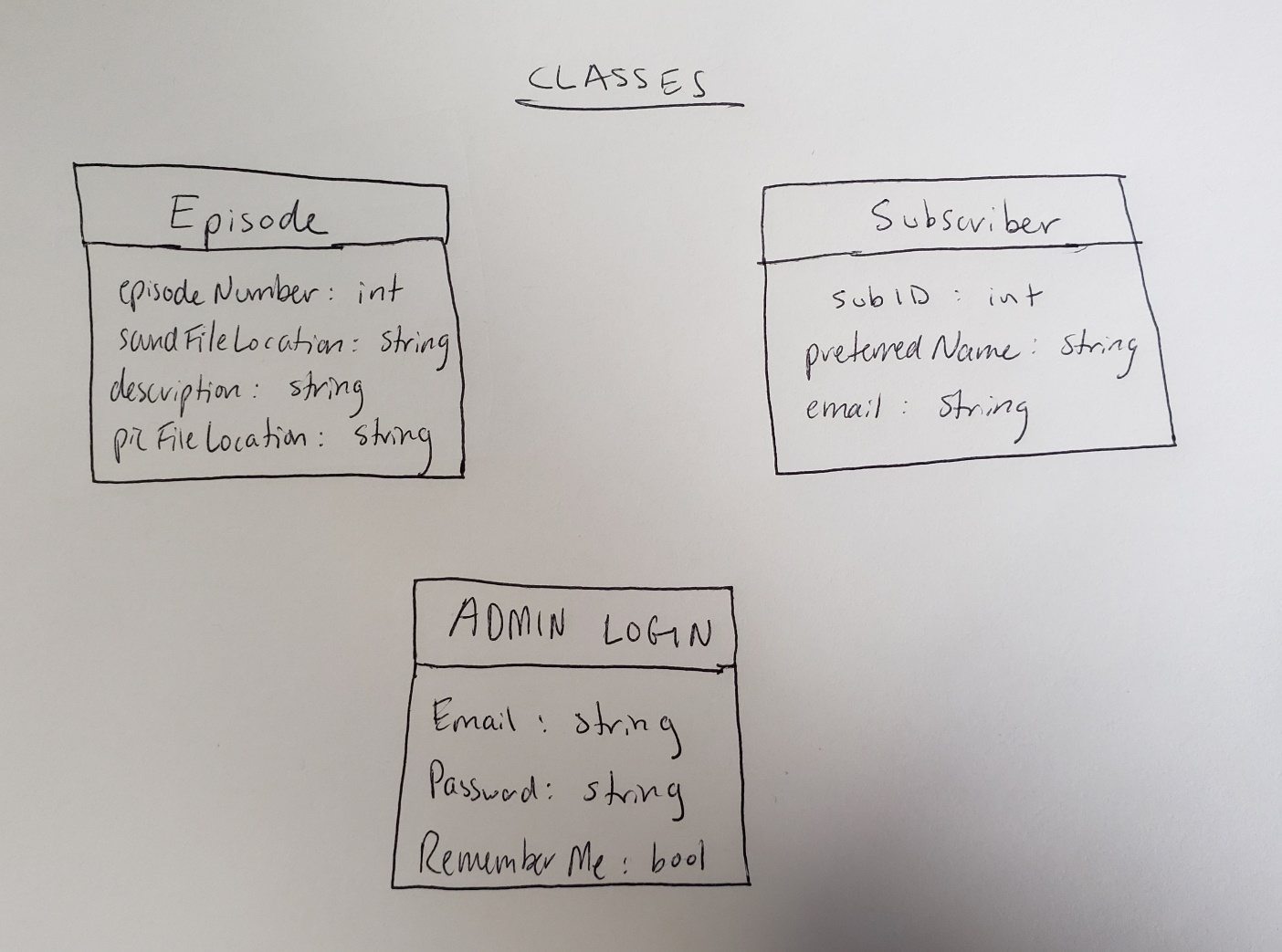
**2.5 Other Diagrams**

To be added when necessary.

**3. Design Section**

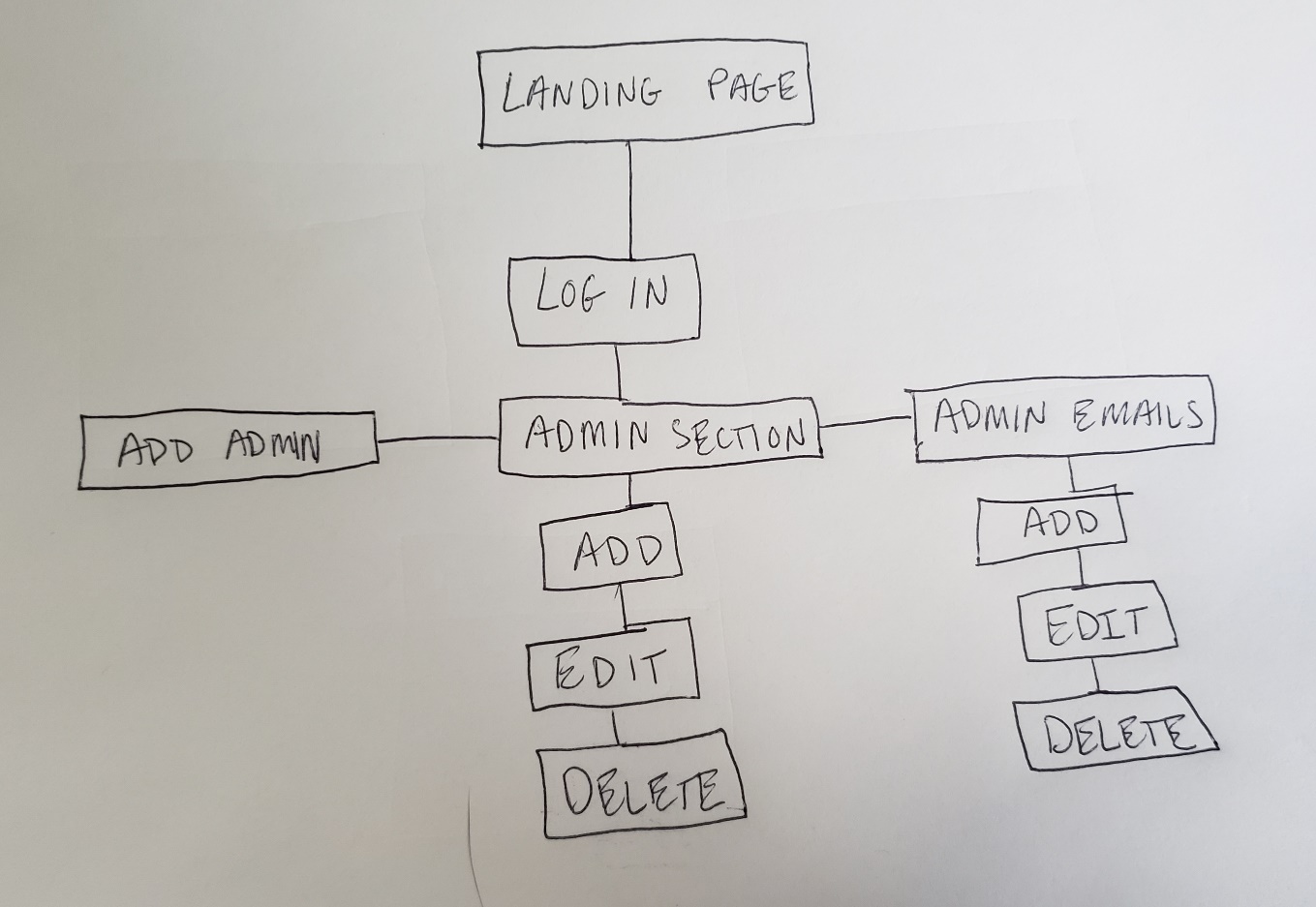
**3.1 Architecture Design**

**CLASS DIAGRAMS**

****

This describes the episode, subscriber, and admin login class. The admin login is the class used for the login functionality and is different than the user table in the database.

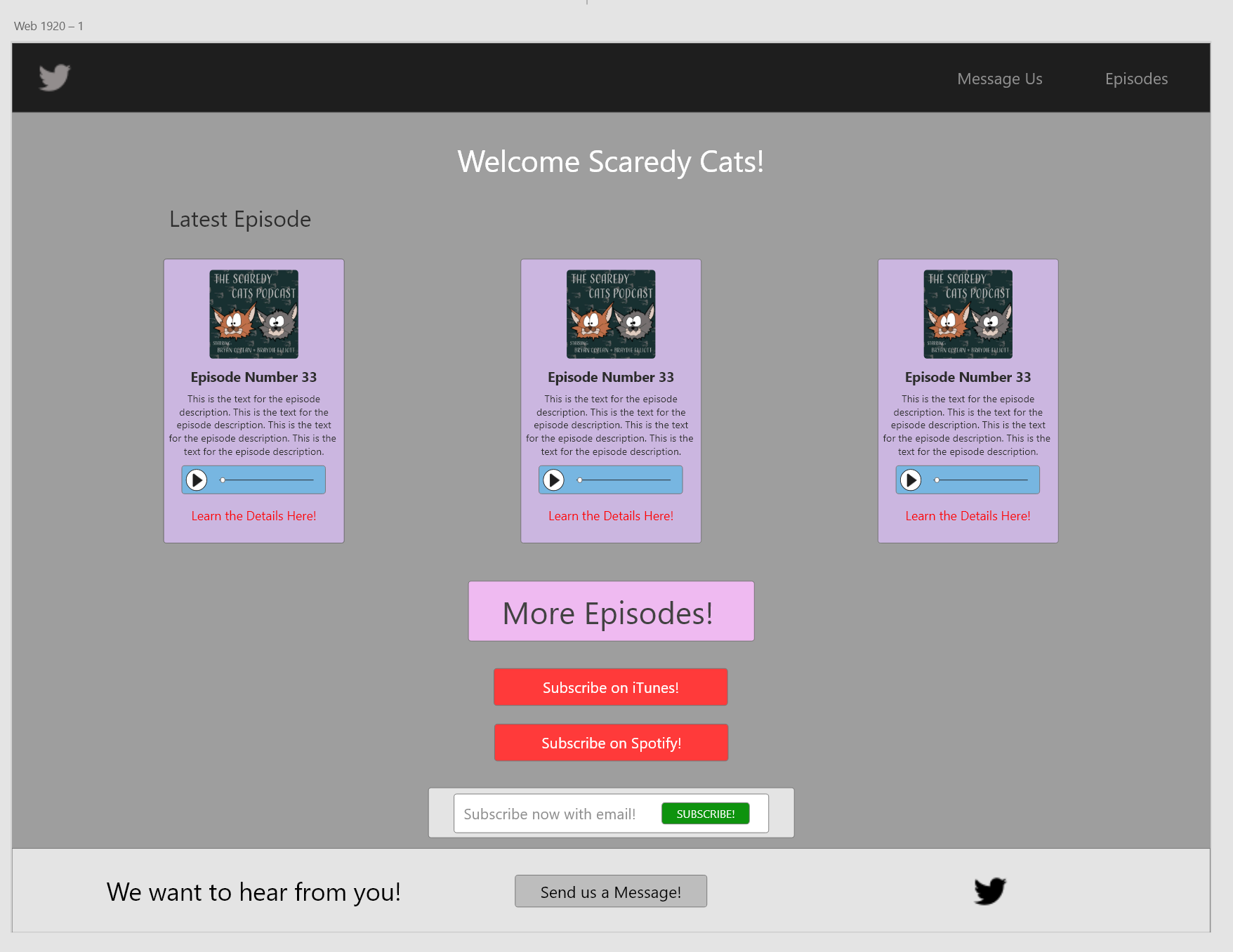
**WEBSITE HIERARCHY**



This is the updated website hierarchy as of 5/11/2021.

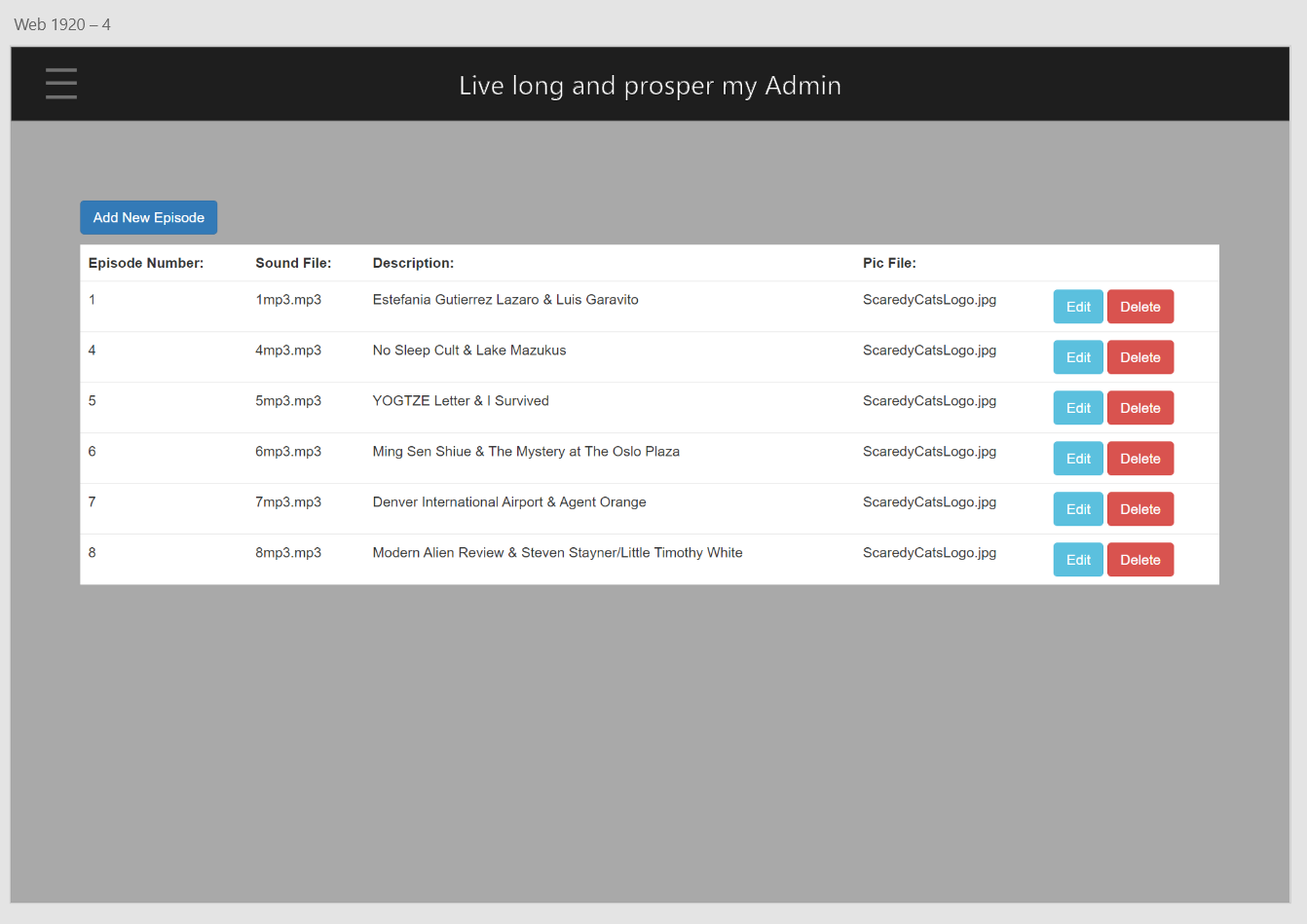
**3.2 User Interface Design**

**LANDING PAGE**

****

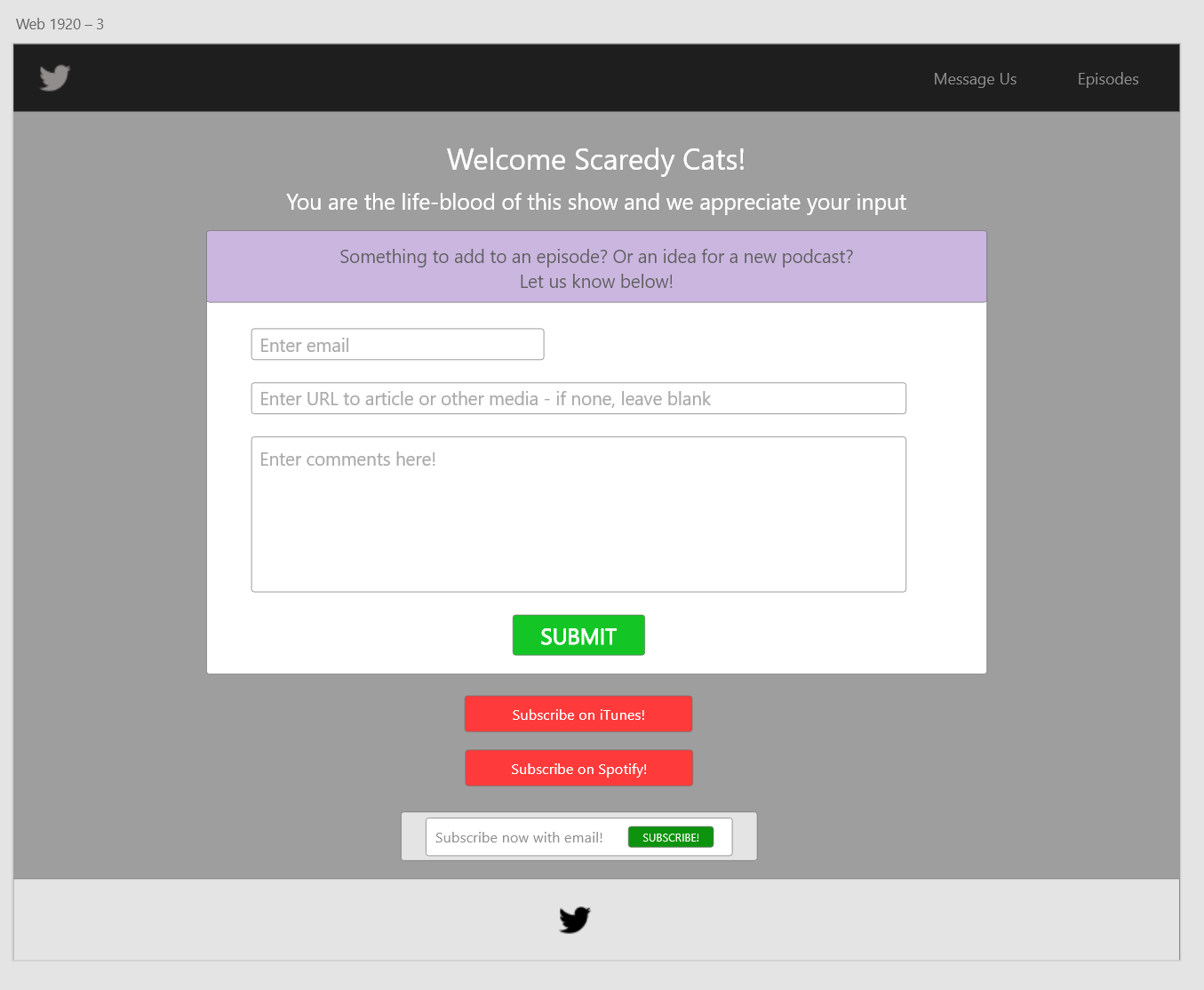
This is the wireframe of the landing page designed in Adobe XD

**ADMIN SECTION**

****

This is the wireframe of the admin section designed in Adobe XD

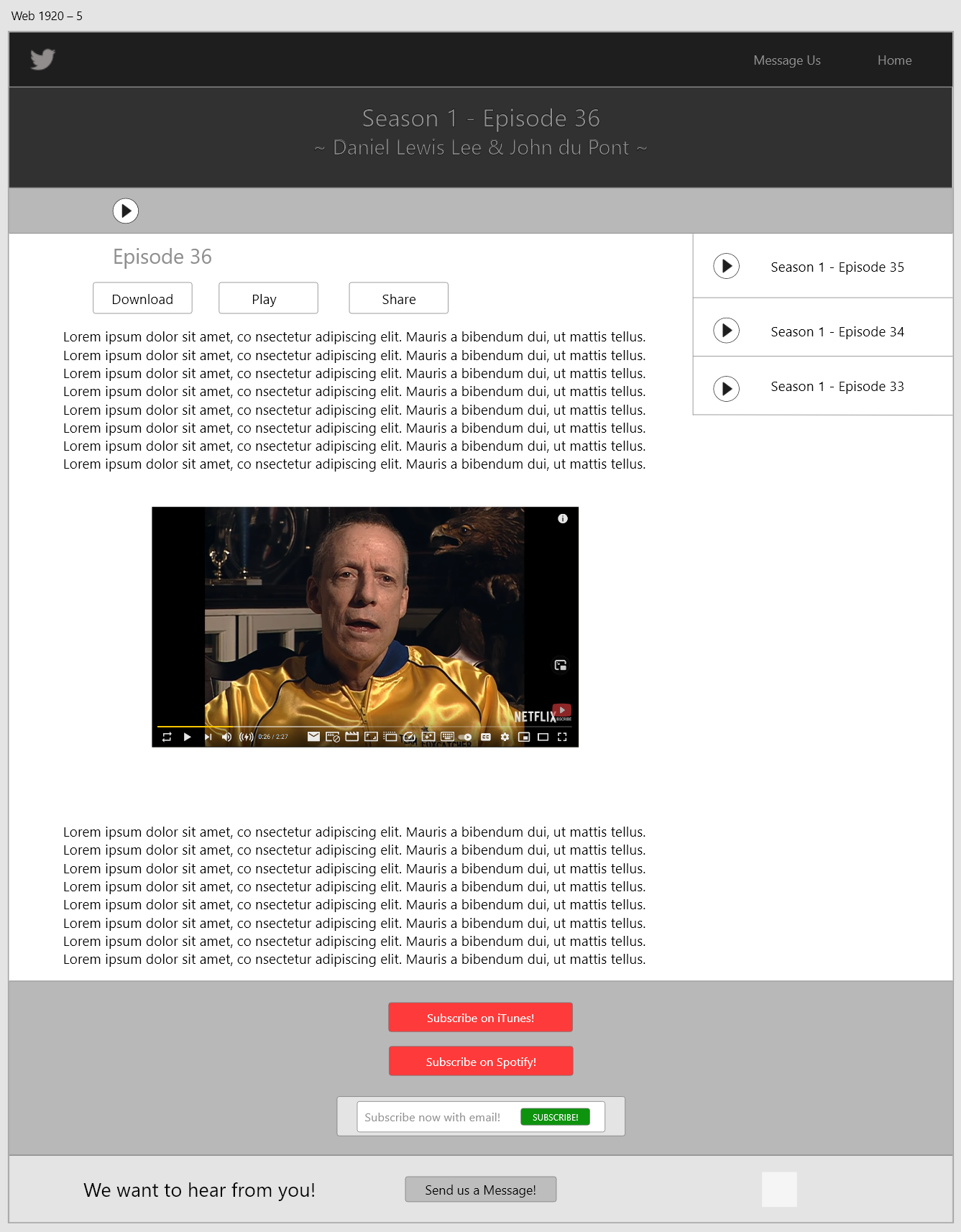
**MESSAGE US**

****

This is the wireframe for the message us section designed in Adobe XD.

This section will not be reflected in the presentation, but serves as a representation of what the website will be.

**EPISODE EXTRA CONTENT**

****

This is the wireframe for the extra content section designed in Adobe XD.

This section will not be reflected in the presentation, but serves as a representation of what the website will be.

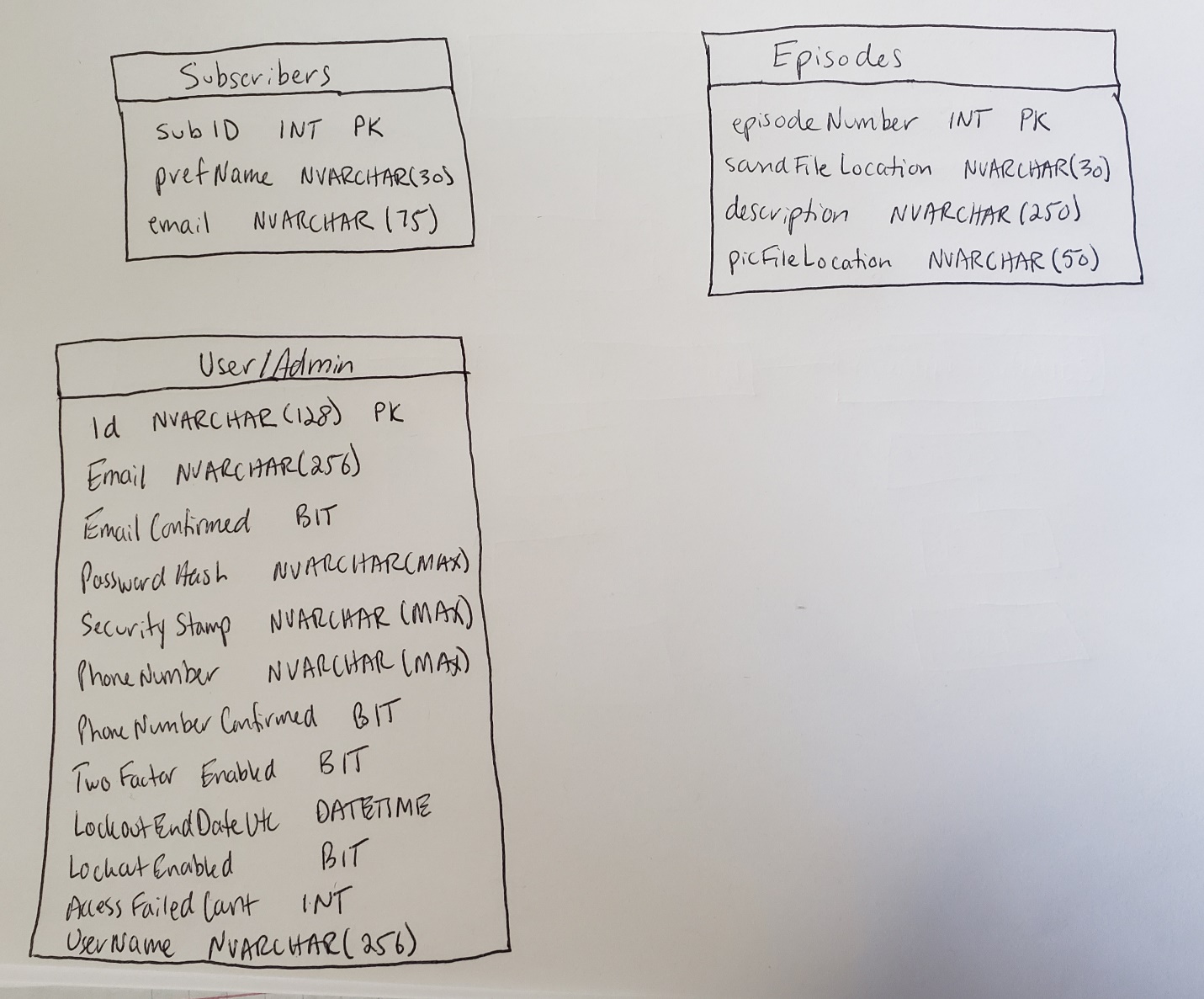
**3.3 Source Control and Security Layout**

My source code is in a github repository. There are no external API connections, or code snippets that are valid for this section

https://github.com/Dunwoody-CWEB-Final-Project-Repository/final-project-repository-cozean

**3.4 Data Models and Storage**

**ERD**

****

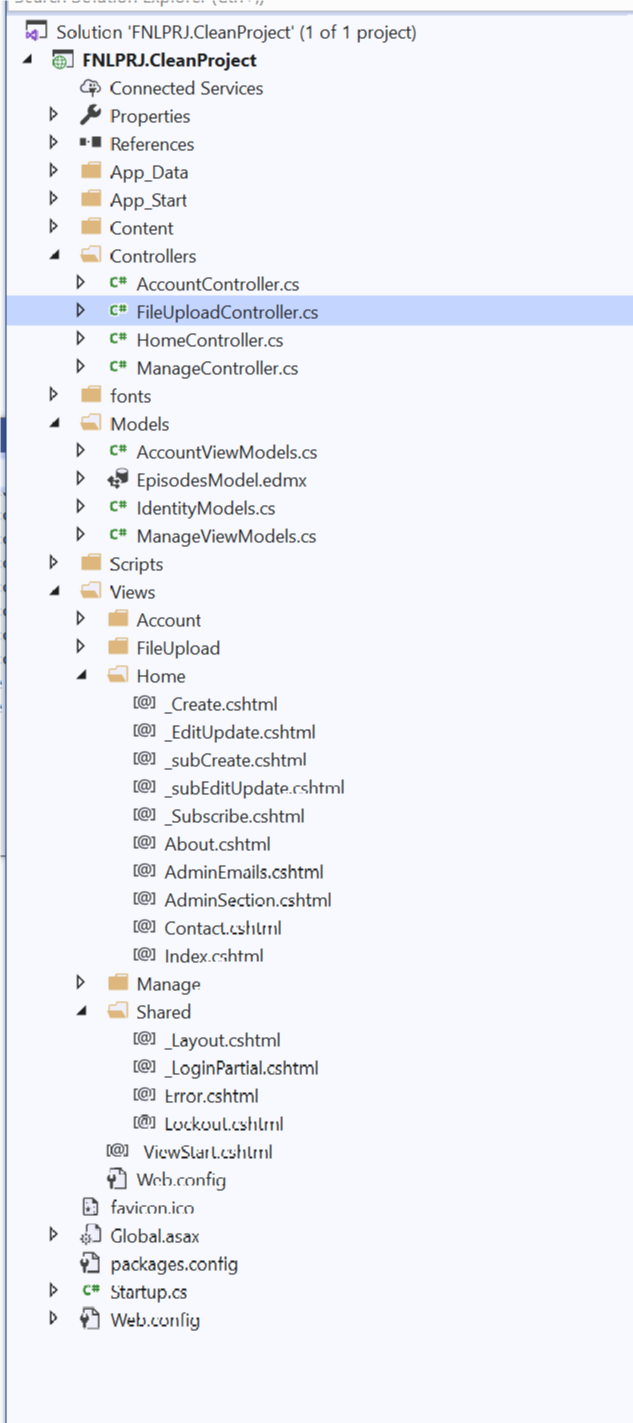
This is the updated ERD Diagram as of 5/11/2021. It shows only the data tables in use at the time of presentation. They are each separate entities that do not interact.

**3.5 Security Considerations**

I will be securing my code on many fronts. The sensitive password data for the admin users, and eventually the user users, will be hashed with up to date encryption through the ASP.NET MVC Framework. There will be no string queries throughout my code – a lambda function and parameters will be used when interacting with the database. The administration section and all the services related to the admin will be secured behind an authorization wall. All of the inputs have validation connected with them as well so that the right type of data will be entered into the database.

**4. Implementation**

**4.1 Map of Framework**

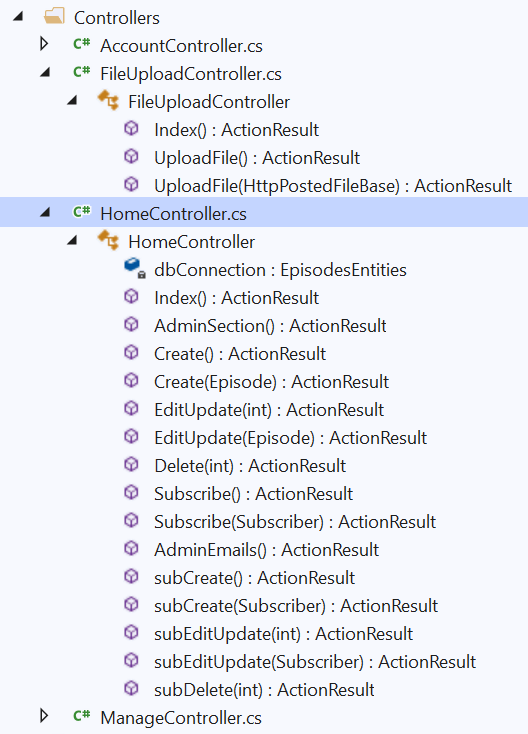
****

Above is a screenshot of the file structure in my MVC application. The majority of the code I generated is located in the HomeController. The AccountController is also used in relation to creating admins. I have created, but as of right now not fully figured out, a FileUploadController that controls the upload of the episode mp3s. The EpisodesModel contains the class structures and related code for both the episode and subscriber aspects of the project. Almost all views, except those linked with the account controller, are located within the Home folder of the Views section. There are both full and partial views depending what the view is doing. The \_Layout.cshtml page is the default layout section used by all views in the project. Finally, the connection strings and other related data is held within the Web.config file at the bottom of the structure.

**4.2 Description of major component files**

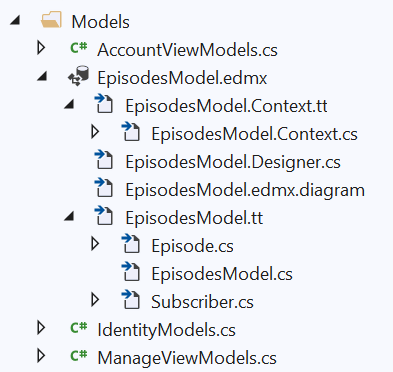
The major files in the project are obviously the Model, View, and Controller sections. Almost all functionality of the project is contained within those files. However, there is also the Content file which holds the bootstrap/CSS files, mp3 files, and picture files. All these files are used to build the landing page, with their locations stored in the database.

**4.3 Controllers**

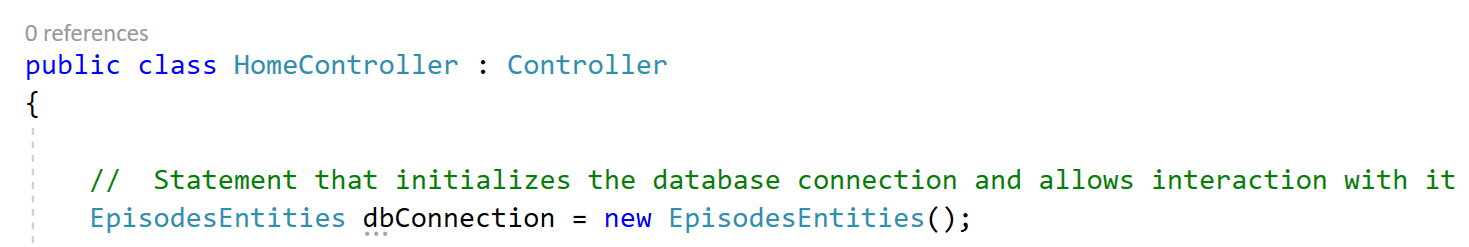


This is a more in depth view of the controller files. While there are a few methods being utilized in the AccountController folder, the vast majority of code created by myself comes from the FileUploadController and HomeController. All of the methods that facilitate interaction with the database are shown above.

**4.4 Models**

****

This is the overall model setup for my project. The EpisodesModel contains the information needed to connect to the database and facilitate movement of data based on their blueprint.

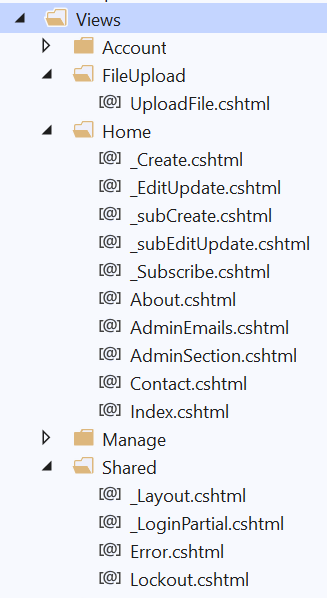


This is the code that actually connects to the database and gives access to the data blueprinted by the model.



This is an example of how I am safely querying and updating the database with parameterized queries.

**4.5 Views**

****

Here is the structure of the views in my MVC project. All of the views and partial views that facilitate interaction with the database are located in the Home and FileUpload folders within the main Views folder. The \_Layout view is used by all the full views as the blueprint of what it will look like – rendered in the body section of the view code.

**5. Testing and On-Going Maintenance**

**5.1/5.2 Overview of Site Test Plan/Methods Used for Testing**

In developing this application there was no fancy testing involved. I did not use any special testing packages within the MVC framework. Throughout the development process I would run my application, test it to the point I was looking for, and then change the code and repeat. At times I did change code within the browser’s inspect feature. Then these changes made in the browser were implemented into the code of my project. I was the tester.

**5.2.2 Interactive Tests**

For the purposes of this project I did develop an interactive test carried out with my girlfriend, whom is the other half of the podcasting team. She will possibly be using the platform’s admin features to update the site, so periodically I allow her to use the interfaces and give me feedback. She is quite picky in the way she wants things to run, so I felt this was necessary to give her the best experience possible. The tests simply consisted of her running through all the functionality of the site and taking notes as she verbally described her experiences to me. Many of the pop-up modals on the site had her fingerprints all over them. She is not too big of a fan of the color scheme, but that is something we will be tackling in the future.

**5.3 Methods Used for Maintaining Application**

I plan on maintaining the project into the future. There are not any APIs or extra data sources, so being that we control all of the data it should be easy to keep the website going. Then the only thing to do will be for me to code the rest of the features that we have planned for the site.

**5.3.1 Overview of Maintenance Plan**

I think for the most part this will be a private project. I am not sure I like the whole GitHub thing for a single developer project. I like keeping my own versions on my local machine, but if this was to grow into a multi-developer job I could see maintaining the GitHub repository. Currently I think I will copy the private repository and make a public one for employers and possible clients to be able to see the project I built.