## Approach for legacy

The team has already begun to learn the legacy code. After learning how to deploy the application, we have reviewed the code thoroughly and all of us have a general understanding of the current functionality. First, we looked at the current login system and determined that we need the login information for the previous team's firebase account. Then we took an in depth look at the current frontend and discovered that the application is utilizing React to connect to the backend in Rails. Another thing that we looked at is the APIs that the previous team created. We learned how the APIs interact with the application and we have some ideas for how the APIs will work with our improvements to the application.

The first step to continuing to learn the prior code is to understand the tests that the previous team was running on their code. The first step to learn about the tests is to read the documentation and work on the tests individually. In the documentation, the previous team has broken down the tests that they are running by category. We will continue to understand the tests by not only analyzing their test case code but also by modifying and fixing their valid test cases. We will go through their previous tests by category and find the code that matches each case and ensure that we understand the code and test case by breaking the tests and breaking the code individually before repairing it. Through understanding and modifying what is being tested and how it is being tested, we will be able to write our own test cases for the code. This will also reveal any potentially problematic areas. The next step that we will take is to continue to review the documentation. Through the documentation we can learn where the code came from and ensure that we do not introduce any undesired behavior. A key step to learning code is to refactor the code separately from making changes. This will ensure that we know how the code works before attempting to add functionality that could break it.

After we finish learning about our legacy project, we are going to make a few changes and refactor a few items to improve this project. The first thing that we are going to improve is to refactor the old React components to rely primarily on Ruby and Javascript instead of leaning on the React code. This will allow for future programming to be more streamlined and take out unnecessary complexity.

Before implementing any major changes, we must consult the client. The priority of the team is to maintain consistent communication with the client to ensure that the product that he desires is delivered. The client has expressed that there are changes that they would like in order to create the product that they desire.

An example of a change that we will make to the legacy code is to rework the metadata for the video uploads. Currently, once a video has been uploaded, the automatically generated metadata is not as helpful as the client would like. We plan on allowing the user to input metadata for each video and change some of the current metadata to be relevant for coaches and players. Another thing that we will update from the legacy code is the database. The current database model is not organized in a way where the data can be filtered, and there are additional data points and tables that need to be stored. Additionally, the current placeholder API produces different results than the client would like so we will change this to fit our data model. One last change that the client requested is an overhaul of the login system. This is because the current system is not very secure and uses a database system where access is not provided to us. The client would like an administrator account to manage the accounts that have access to the project. Outside of these changes, we will be able to keep the majority of the backend code from the previous iteration of this project.