Design Document

Philip DiGiorgio, Samuel Fritz, Brandon Harvey

Project Pitch

Ultimate frisbee is a sport that began in the late 1960s. In the 50 years since, the sport has grown in popularity and is now played worldwide, including in multiple North American professional leagues. However, the recent growth of the sport has not allowed it to have the same in-depth analysis that popular sports (football, baseball, etc.) have available to them. As a way to combat this, we will be building a prediction app for the AUDL (American Ultimate Disc League) that takes weather factors into account. Ultimate frisbee is a sport often decided by how well a team can use weather conditions to their advantage. One of the largest of these influences in a game is the wind, simply due to the way it affects the aerodynamics of a moving disc. It changes the lift and drag of the disc in the air, and a team that cannot adapt well enough to it will often lose for this reason alone. As a result, weather conditions must be taken into account when predicting the outcome of a game.

Our app will be built around a machine learning prediction algorithm that takes into account all of the weather conditions during games as well as individual player and team-wide performance to make the best prediction of the outcome of the game. By specifically looking at past games for teams and players and looking at the weather conditions during their games, we can find trends in how they perform in those conditions and use the projected weather for a game to predict their performance in the near future. The AUDL is already separated into many different teams, but we can build our representations of teams player by player instead, opening the door to hypothetical matchup predictions between pre-existing teams and players, but also between customized teams. Though not a full fantasy sports app, elements of the like will exist to help drive user engagement and enable fans to examine situations of their own creation.

While sports prediction engines and fantasy matchups are not new, nothing is available for ultimate frisbee, and existing engines do not take weather data into account. By creating a new engine from scratch, we can target ultimate frisbee fans as our users with high specificity. The lack of a prediction engine for ultimate frisbee enables us to be the only ones in this user space. As a subset of fans of ultimate frisbee, coaches and players can use it to their advantage to gain insight into forecasted stats for their upcoming games and identify possible areas of issue. Our prediction app is highly specified for ultimate frisbee, and we can focus our user base to take advantage of that fact.

Broadly speaking, we wish to create something that is not available in the ultimate frisbee space. Creating such a prediction engine allows us to provide fans with a greater experience in the sport, allowing them to use our predictions for their own gain, on the field or off the field. For fans, it can be understanding the sport more and the different ways that weather conditions and each team member contributes to the overall success of the player. For coaches and players, it can be understanding where their team is and what they need to work on to

ensure success in the future. All of this contributes to the desire to allow ultimate frisbee fans to experience an aspect that fans of other sports get access to - predictions.

Technical Summary

The Ultimate Prediction Engine (UPE) is an innovative kind of sports prediction algorithm that takes weather data into account as well as player and team statistics to predict the outcome of ultimate frisbee games in the American Ultimate Disc League (AUDL). Machine learning, through the use of a support vector machine, will combine different factors about players, teams, locations, and weather to predict the outcome of games. To build the support vector machine, we will be using scikit-learn. The data for the model is being scraped from the AUDL website and the Visual Crossing weather API. After scraping and cleaning, data is stored into a MySQL database. To automate the retrieval, cleaning, and storage process for the data, we are using Node.js to run the back end. Besides the library components, we will develop the back end with an API that allows for data to be easily retrieved. This will be deployed onto an AWS EC2 server to make it publicly available. The front end for this application will use React to allow for custom inputs into the prediction algorithm as well as provide live updates of changing predictions and a breakdown of each prediction.

Ultimate frisbee is a game that is often decided by how well teams can use weather conditions to their advantage, so weather factors in predictions is a mandatory consideration. Key objectives for the engine include being able to predict with a high level of accuracy the outcome of games, being able to accurately update predictions as data is streamed from ongoing games, and being able to ensure that many predictions can be active at the same time.

We are confident that we can accomplish our goals because we have a support vector machine in development, which will provide us with the opportunity to produce the best predictions for our users. Additionally, all the data is available to us and updates are automated for the future, so there is no issue with getting data for predictions.

Direct costs are low, as we are using little pre-existing resources. There is a cost for sub-hourly historical weather data, but the cost is negligible ($2.50 for nearly 10 years of data). The only other direct cost is for the AWS server. Key milestones relate to the ability of predictions, including the ability to predict based on a few factors, the ability to update predictions as games are being played, the ability to predict based on all the factors, and the ability to predict custom match-ups. Each of these is broken out into segments of about a month and a half, with everything culminating in a fully functional prediction engine.

Project Description

User Stories

As an Ultimate Frisbee Fan, I would like to predict possible outcomes of matches between different AUDL teams so I can learn more about the sport and which teams are more likely to win the league.

As an Ultimate Frisbee Fan, I would like to customize match settings such as temperature, windspeed, precipitation, and humidity parameters, in order to see how different teams would perform against each other under different weather conditions.

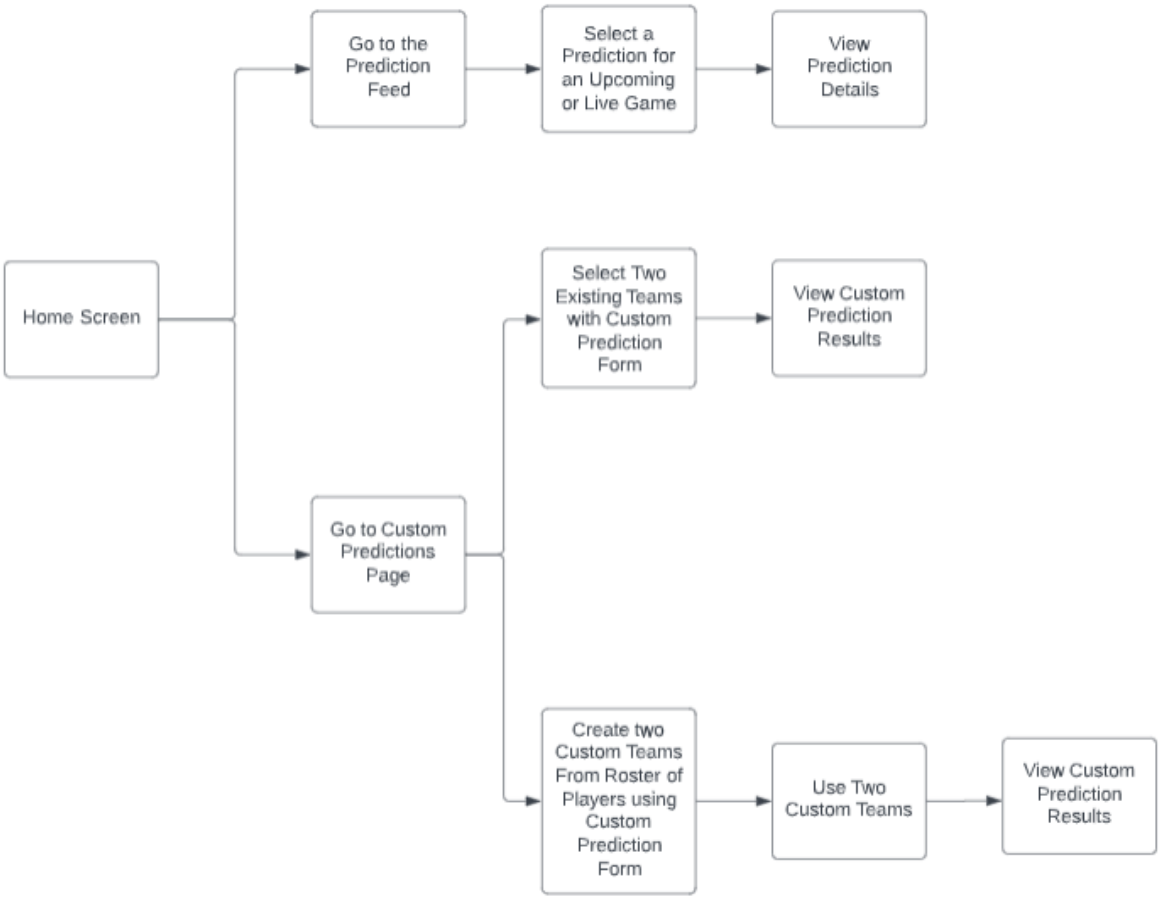
As an Ultimate Frisbee Fan, I would like to view a prediction feed for current and upcoming games so I can get an idea of who is most likely to win in the given real-time conditions.

As an Ultimate Frisbee Fan, I would like to view in depth details about how each prediction is being calculated, so I can understand the factors that determine the prediction.

As an Ultimate Frisbee Fan, I would like to create my own custom Ultimate Frisbee Predictions using real teams or made-up teams, so I can see how different players and teams perform against each other.

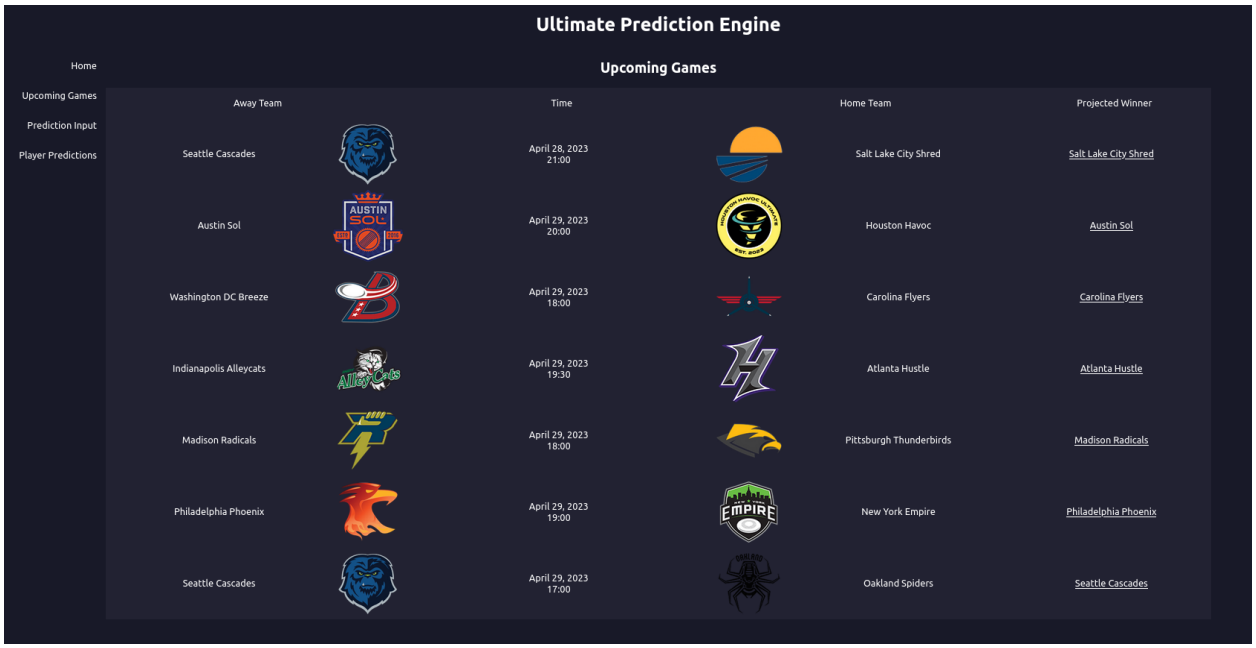
As an Ultimate Frisbee Fan, I would like to create my own custom teams from a selection of all past and current players when creating a custom prediction, so I can create impossible matchup scenarios.

Flow Diagram



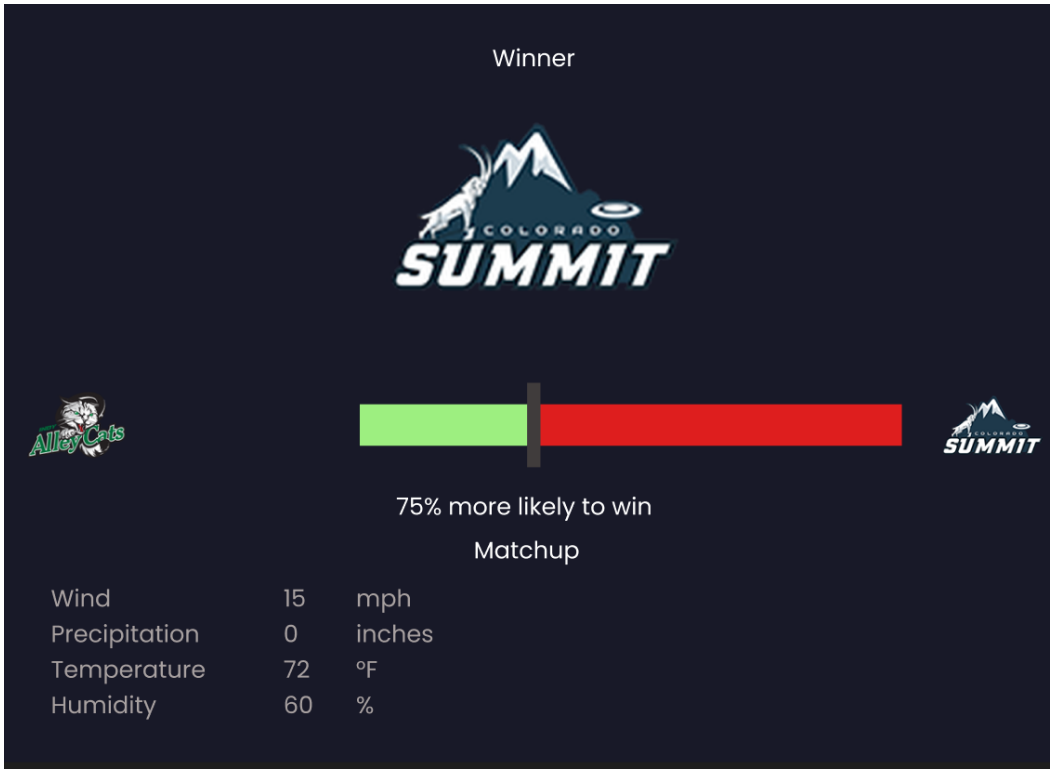
Mockups and Wireframes

Landing page:



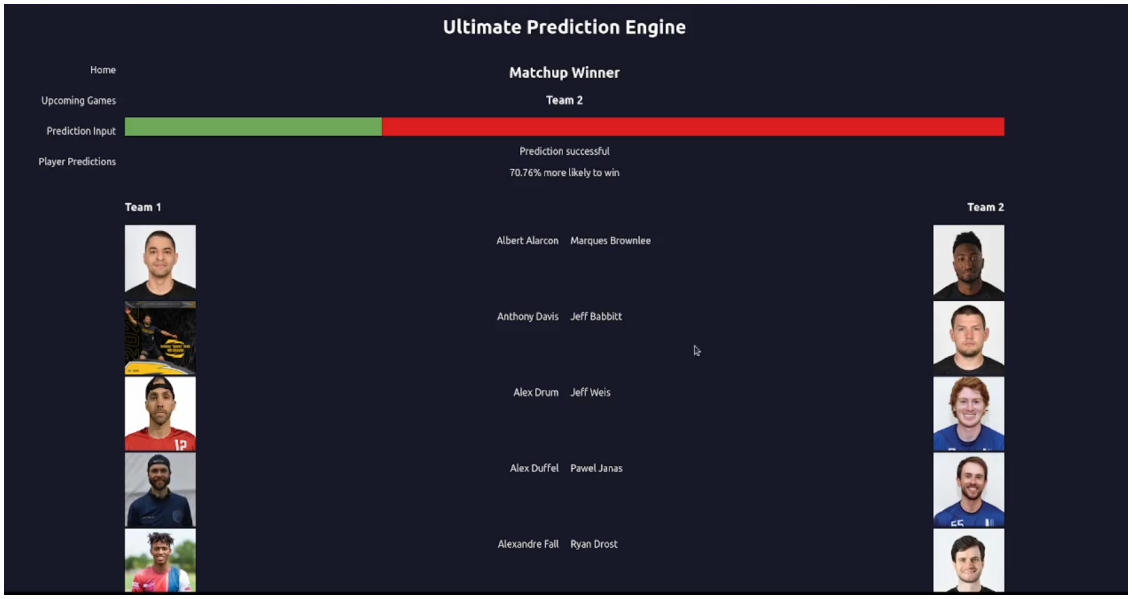
Input form (User creating a Team prediction):

Prediction Output for Team Prediction (Prediction generated based on user input):

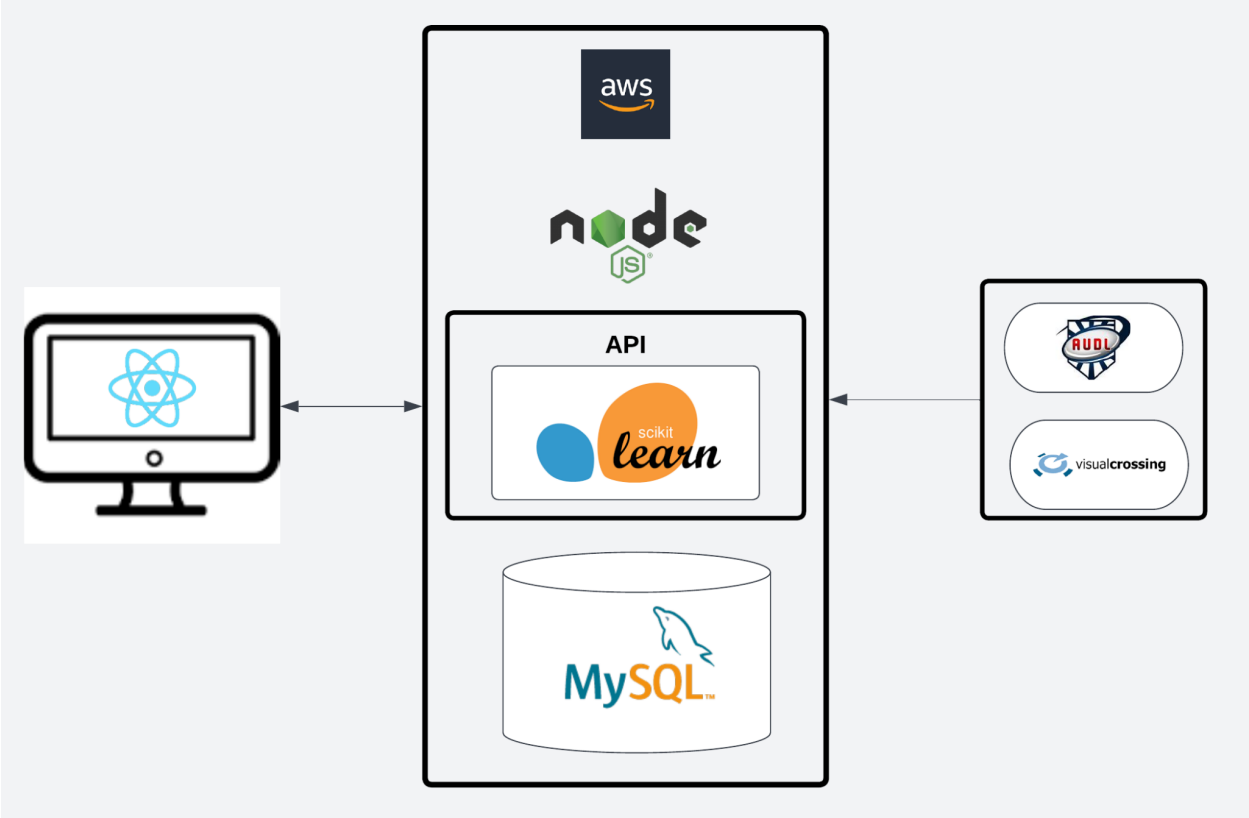


Input Form (User creating their own Teams):

Prediction Output for Fantasy Teams:



System Diagrams



When a user first tries to enter the site, a DNS lookup is performed, after which a simple GET request is made to the HTTP server that serves the React application. The application is sent to the client's browser, after which point it is used to request data from the server. When a

prediction is requested, the cached predictions are checked to see if the prediction was already performed recently. If so, the previous result is returned. If not, the backend requests the prediction API for the new matchup. This result is then returned to the front end and displayed to the user, getting cached along the way.

The data parser retrieves new data from the external data sources on a daily basis and updates the database accordingly. This data is used directly in the prediction engine and in the application to display data.

External Frameworks and APIs

Our project uses a few pieces of external technology. These are all used to build up either the core of the application or to simply the development of another piece. Each is listed below:

AWS EC2:
Our project will be deployed onto an AWS EC2 instance. This enables us to have the project available for public use in the future, and it gives us a centralized database that we can use during development. Specifically, we will use this instance to have a MySQL database that each developer can connect to and get the most up to date data during development. When it comes to deploy our application, this will serve as the central point of availability, with the specific deployment broken up into different containers.

Node.js:
Node.js is our back end for our application. This will serve as the connective piece between the database and public endpoints. It will expose routes that enable the front end to retrieve the necessary data for display on the interface

React:
React will be used to render the front end for our application. It will allow for dynamic inputs and updates in an easy to use manner, allowing the application display to be easily updated.

Visual Crossing API:

Goal:

- Get temperature, windspeed, precipitation, and humidity data from all past AUDL games as well as forecast data for upcoming games.

Description:

This API is used in the weather folder in the code. There is a separate script used to collect and store data from all past AUDL games. The parameters location and startTime for this script are gathered from game data files stored on a local system while the parameter endTime is computed automatically. This API will also be used to gather and update forecast data for upcoming games every 24 hours (To be implemented in the future).

Endpoints Used:

- GET /VisualCrossingWebServices/rest/services/weatherdata/history
- GET /VisualCrossingWebServices/rest/services/weatherdata/forecast

Scikit-learn:
This will be used in order to implement a Support Vector Machine with reliability in Python 3 allowing for easier implementation and testing of various features available based on previous game data to give a reliable outcome as to which team is predicted to win.

All of these external frameworks and APIs are necessary in order to facilitate the development of the application. Of note, only the AWS EC2 instance incurs a cost for development. However, this cost is very low (estimated around $250 with AWS Pricing Calculator).

Algorithms

Support Vector Machine (SVM) ML Algorithm (Using Sci-kit Python Library):
The main algorithm used for our prediction engine is an SVM model structured using the scikit-learn python library that takes in data from our database in the form of a 2-D array containing a list of samples with a certain number of features along with a 1-D array of targets which signify the team that each of the games in the 2-D array refers to. After the model is fitted, it uses the built-in predict() function to output which team will be predicted to win based on the parameterized features compared to the SVM.

This algorithm is the key part of the application, as it is how the predictions are made for matchups. Success for the algorithm can be measured by ensuring that a prediction can be made for two inputted teams and a selection of weather factors, ultimately giving a projected winner.