Technical Innovation (Joe Hallal):

Building fantasy football lineups is a complicated NP-complete problem. It would take over 3,000 years to weigh every possible lineup combination in a naive solution. In order to overcome this, we aim to apply available technologies in new, creative ways. We will be able to do this in a reasonable time-frame by using a solution strategy for the “knapsack problem”. By grouping players into subunits and building combinations of these groups, we will cut out redundant comparisons.This approach allows us to incorporate constraints such as a maximum salary, because we can easily eliminate lineups that do not meet the needed thresholds. We will use a Python script to execute our algorithm that will work with imported data in CSV format. We know this approach is valid because there are existing technologies, such as RotoWire, which use this approach for NBA lineup generation. Their optimizer reports a generation time of around .5 seconds per lineup. While NFL lineups may have a slightly more complex combination output, we firmly believe we will be able to approximate this efficiency and generate lineups much more quickly than the competition.

A second way our algorithm will be unique is in its implementation of correlation factors between players. Maximizing positive correlation in lineups is a great way to optimize for successful combinations. We will leverage a data-mining solution using a modified A-priori implementation to find these correlations and apply them to our algorithm. No other optimizer does this automatically and Take the Lead’s users will likely end up with unique combinations of players that yield higher collective performance ceilings.

Another way we are going to make this process more efficient is by using Amazon Web Services (AWS) to store and compute lineup combinations. Even with our algorithmic optimization, this process will still take a tremendous amount of computational power and storage. A goal of Take the Lead Fantasy is to provide lineups to the user nearly instantly. By using the power of Amazon’s cloud computing service we will be able to generate our high quality lineups in a fraction of a second. Our competitors, like FantasyCruncher, may take more than ten minutes to generate valid lineup combinations whereas we believe we can dramatically cut this down.

An additional way we are ensuring the happiness of the user is by including a data visualization aspect to our product to inform players about our lineup generation process. We will use the widely implemented D3js visualization package to create custom aesthetic tools. D3 contains a plethora of UI components as well as ample documentation and flexibility. Data Visualization is a component unique to Take the Lead and there are no other industry implementations of this type of service. A distinguishing factor at take the lead is our ability to teach the user about the generation process and give them the knowledge they need to get successful lineups on their own.

Finally, we will deploy an Angular Web-App with a simple and seamless UI. Contrary to our competitors, Take the Lead will feature a simple and easy to use interface that will not need dozens of hours to learn. We want new users to be able to jump right in and generate lineups that they understand, so we will use tools such as D3js and Bootstrap to ensure their experience is seamless on the frontend.

Technical Feasibility(Mihir Mankad)

There are a multitude of different software tools that will go into the individual sections of Take the Lead Fantasy. The first tool being used in this project is the python programming language, so that data can be acquired seamlessly without much overhead. We are using a bottom-up recursive approach to find lineup combinations. Our method focuses on removing redundant comparisons by pairing off players of the same position. The foundation of this project will be comprised of our python algorithm in conjunction with java and the Spring framework. The Spring framework allows for high performing, easily testable, and reusable code in the java language. This framework will help in connecting our components together smoothly, with dependency injection being useful because data will be supplied to the objects. A database storing the player statistics as well as calculated correlations will be in SQL. SQL is the optimal choice because of the nature of the data being used in this project. SQL also allows for easy connection to our website application. The website application component will be created with Angular, a structural framework for dynamic web apps, using HTML as the template language. Angular was the choice because of the ability for form validation, routing, deep-linking, and dependency injection being built into the . It also allows for a higher level of abstraction, which is beneficial for this project. Moving onto to the visualization technology, these visualizations will be built in D3Js. D3 is very useful for creating unique and interactive visualizations that can convey statistics in interesting ways. This will be used to create fantasy visualizations such as a decision tree based on correlation and a lineup profile. This concludes the tools and technologies that already exist, and that will be used to build this project.

There a plenty of reasons for why this project is do-able. The first reason is that similar websites already exist in some format. There are optimizers that are popular, that perform similar operations to what this project aims to do. The difference, is that this project is innovating with respect to the backend technology, as the core algorithm is something completely unique. The algorithm in mind has already been applied to other situations, which is also another reason this project is do-able. The tools mentioned previously fit the project well, and will not cause issues with completion of the project. With respect to the front end aspect, visualization for almost every situation can be done in D3Js, so there are no issues with the implementation of the visualization elements. There is confidence that each of these components can be completed due to our background and extensive research into these technologies. This team will also work tirelessly to make sure the project is completed as designed.

***Cost***

The greatest cost-prohibitive constraint to Take the Lead is computation time. Generating fantasy lineups takes a large amount of time because of the algorithmic nature of the problem. The core of the generator is an algorithmic paradigm called the Knapsack problem. The Knapsack problem takes a few inputs: a salary cap and list of players. Each player in the list has a projected performance value and a salary. Part of our algorithm finds the highest projected lineup of players with a combined salary less than the salary cap. As the number of players participating in Fantasy Football grows, the combinations of these plays grows exponentially. The most basic strategy of generation takes at least 20 minutes, which is clearly not suited for a quick and responsive user interaction. This computational complexity forces our software to use more effective strategies.

The first strategy Take the Lead can use is cloud computing. Generating highest projected lineups under a salary cap can be done with relative ease. This operation can be performed with under 16 gigabytes of memory and the processing power of a quad core cpu. However, generating a wide variety of correlations between players and lineups in Fantasy Football promises to require both a significant increase in required memory and processing power. The easiest way to generate this information is to use CUDA technology, which leverages CUDA cores in Nvidia Graphics Processing Units to provide computational parallelization. Most of the computations correlations use are independent of one another and require a minimal amount of shared memory and resources to run. This allows each CUDA core to run separately for the most optimal divide-and-conquer strategy. By using cloud computing with AWS, we can leverage a vast GPU infrastructure that is already built to provide a faster competitive experience to the user. A reasonable estimate of the cost to use cloud computing is $3.06/hr with Amazon EC2 P3. This helps to mitigate the computational cost of our product, but still leaves the issue of web hosting to be solved.

Another core component of our project is a website. Take the Lead uses the Java Spring Boot framework to handle backend server elements, but this needs to be publicly hosted for internet access. Building the website does not incur and additional cost other than time, but hosting it will require a variable fee. With the provider Bluehost, our website can be hosted for as little as $3.95/month, giving us unlimited bandwidth, storage, and a free domain. These paid options certainly seem obtainable, but as with any new startup, a certain amount of risk is present.

***Risk***

One of the risks to our project is a secure pricing model. When dealing with queries that require a large computational task, it is important to be absolutely certain that users are paying for the services they use. If our pricing model is off even slightly, a hoard of users could flood our website incurring massive fines. This risk can be mitigated simply by comparing our AWS pricing model to our traffic rates and query types.

Another risk to this product is time wasting. As we develop our website and algorithm, we may come to find that many of the intense computations we projected to need are actually feasible with less hardware. For example, if our cloud computing module can actually run on a standard quad-core laptop, the time spent developing the cloud software infrastructure will be permanently lost. The only way to help mitigate this issue is to carefully plan each step of the development process and test code frequently.