Options Trading Simulator

Team 1 --- The Big Team (Tom Busby, Blake Sweet, Justin Luttrell, Josh Luttrell, Tej Patel)

University of Kentucky -- CS499 -- Fall 2019

September 24, 2019

Word Count for the entire document: 3161

Authors:

- **Tej Patel**
  - **Introduction, Project Overview, & Functional Requirements**
- **Justin Luttrell**
  - **Project Overview, Functional Requirements, & Nonfunctional Requirements**
- **Josh Luttrell**
  - **Functional Requirements, Nonfunctional Requirements, User Interaction**
- **Tom Busby**
  - **Development and Target Environments & System Model**
- **Blake Sweet**
  - **Feasibility & Conclusion**

Table of Contents

Introduction

The following document talks about the specifications of our project. It talks about the basic project overview, development environment, target environment, project model, user interaction, function & non functional requirements and feasibility. The document ends with a short conclusion with some basic guidelines about the use of the software and its future use. The software holds great value for the company, as it can provide its customers with an opportunity to measure whether or not their respective investment strategy would have performed well. Hence providing great value to anyone who wishes to invest in options, and wants to know if they have a good portfolio-strategy combination.

The project our team is working on is an "Options Trading Simulator", a prototype for the ODDS Online platform whose basic function is to take in a set of data and evaluate a portfolio based on the selected investment strategy by the user.

The options simulator is going to enable the user to select an investment strategy, comprised of multiple options, with a principle amount and an investment-percentage selected by the user themself. This would enable the user to make deductions if the strategy is profitable or not based on the past record. The user will also be able to see the daily performance of the options selected along with the interest gained from the money within the portfolio. The issue that still stands with this is that past data can help evaluate the profitability of a strategy, but cannot guarantee success in the future.

Project Overview

This section of the document gives an overview of the problem we plan to tackle with this software, customers, stakeholders and the intended users. It also explores the current solution in place and why is it necessary that our solution be implemented.

This web-application, in its prototype stage, is intended for internal use by ODDS Online employees to see how their various investment strategies would perform on past data. In production this software system would be made available to paid ODDS Online members to run their own simulations. In this way, the stakeholders would be Don Fishback who is the owner of ODDS online along with James Hughsback the Director of Information Technology, along with his fellow colleagues.

An individual would not like to lose money which is why the simulator provides users with an opportunity to test their respective investment strategy. A person's profits and losses can also be determined by the principle amount and the percent they invest hence why the software allows users to pick and choose the strategy and the principle amount based on the past data. This way, based on a thorough analysis, a potential investor can gauge how much money they may have made in the past over a certain period of time. This way an investor can make an informed decision about the profitability of a strategy.

As of now, someone who wishes to predict the profitability of the options needs tons of calculations and high microsoft-excel skills. And even then, it is difficult to monitor daily progress and the final profit or loss. This method is inefficient since it is time consuming and a lot of manual work. Hence with our automated options simulator, the user will be able to visualize the daily progress data as well as their gains.

This way there is no need for the user to utilize a lot of their time behind figuring out the math, gathering the past data and visualizing the results; since the software does it all. All the user needs to do now, is to choose the principle and the investment strategy. Saving them hours that they would have rather wasted in coming up with predictions.

Because the company uses its own calculation models and formulas, there is no other software that currently implements the data visualization. Also since the investment strategies are proprietary, the calculation methods cannot incorporated into an already existing software; which is why it needs to be developed.

Development and Target Environments

The first section is about software that can be run on any platform. The second section is about specific software required for development environments. The third section is about software required for the production server. The fourth section talks about hardware requirements to run the software we will be writing.

The language that will be used is PHP 7.0 - as required by the company. The back end framework that will be used is Laravel 5.5, since that is the latest version of Laravel to support PHP 7.0. Bootstrap will be used for front end styling. We would use mySQL for querying database. Composer will be used to download any packages required for the project. Git will be used for software version control.

The development environments will be run on virtual machines on the students personal laptops. The virtual machines will be setup using Vagrant. The local virtual machine will run the pre-packaged Vagrant box Homestead. Homestead contains a ready to go development environment running Ubuntu 18.04 for Laravel. The virtual machines will run Nginx to serve the web pages. MySQL will be the database software used.

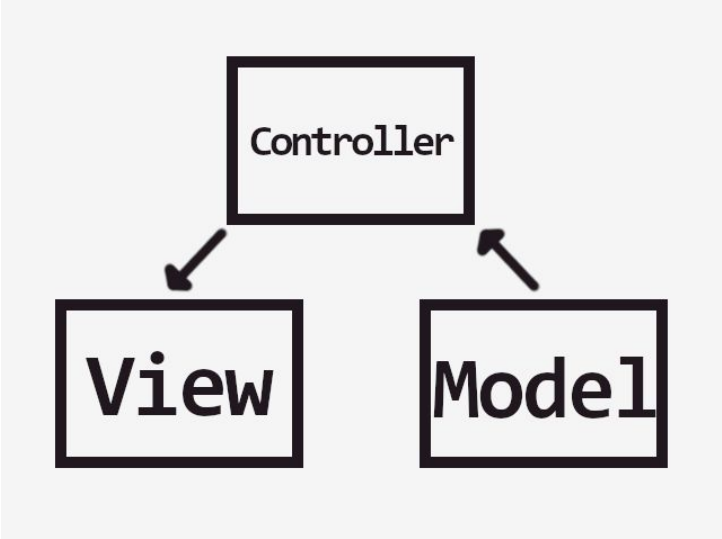
The production server will run on Debian Jesse 8. Apache will be used to serve web pages. Percona will be the database software used.

The hardware requirements are up to the company. The company could run our software on anything from a raspberry pi to an MIT supercomputer. But for the sake of being useful the web server will need at least a CPU with 2 cores and will need at least 8 gigabytes of ram. The storage required for their information will be several terabytes hard drive space.

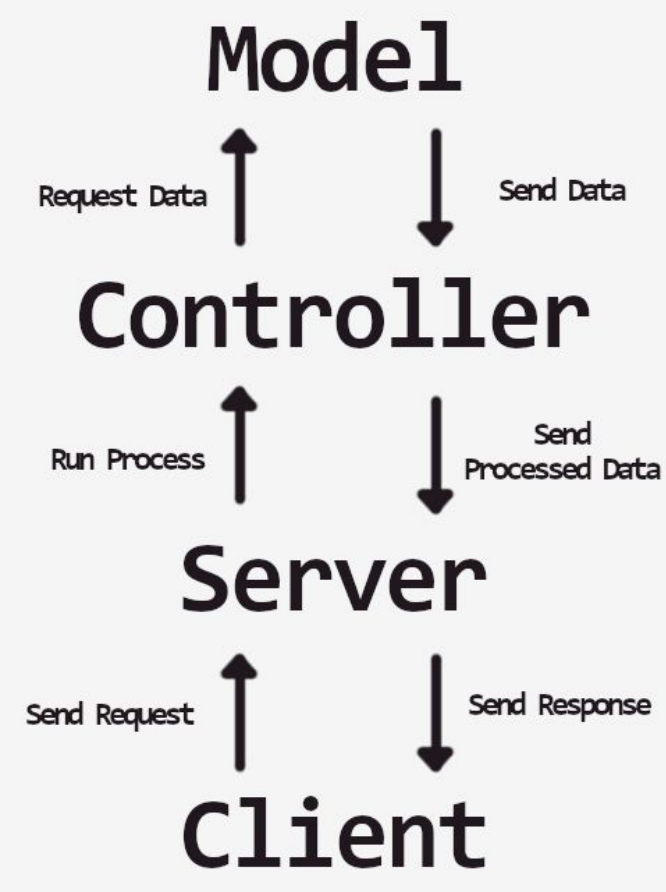
System Model

The first section will describe the existing technology that we are using and the system that the existing model is based off of. The second section will show diagram and explain how the systems work.

Laravel uses an MVC framework. The model that we are using is an SQL database that is made from a schema. The controllers are classes and methods written in PHP that processes data and sends that processed data to the view. The view in this case are blade templates. Blade templates take data sent from a controller and writes the data to the view using a templating engine.



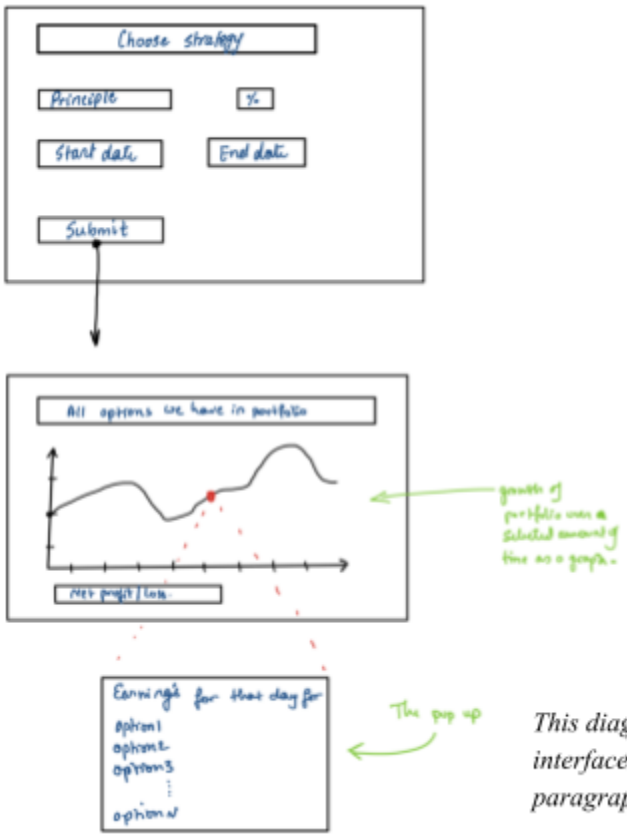
To handle requests from the browser, we will AJAX. AJAX is a combination of technologies that allows for asynchronous web applications. Laravel is made to be able to handle AJAX requests.



To start, a client's web browser will send a request to the server. Then the web server will read the request and start the Laravel kernel. Laravel will find look at the request and match that request with the necessary programmed controller. The controller, if necessary, will request data from the Model. The Model will then send back data to the Controller. The Controller will process the data and then send it back to the web server to send it to the client.

User Interaction

From the user's point of view, they will first be prompted to set up their portfolio by specifying an investment strategy, a principle amount, investment percentage, along with a start and end date. After this information is submitted, users will then see a graph showing how the option is trending over time. The user can select a point on the trend line, as a pop-up box, to see specific information on how the option is performing during that day. Along with the graph, the net gain or loss will also be shown at the bottom of the page with a color code of "green" for profit and "red" for loss.



This diagram shows the user interface described in the paragraph above.

Functional Requirements

This section deals with the functional requirements of this project. It details out the three functional requirements that our program intends to perform. We will discuss each requirement with a set format. The format is as follows - priority, details and verification. Each requirement is verifiable and we will discuss how.

Note - Priority 1 - highest amongst all

Net Profit/Loss

Priority - 1

Details - The main goal of this project is to show whether the selected strategy and inputs yields a net profit or loss. The calculations can be performed through an excel sheet, but we wish to reduce the human effort. Also, the algorithm for the calculations used are developed by Don Fishback and are not be shared. Hence through the means of this app, they can be masked and still yield the right kind of result. The expected gain or loss is made visible to the user, who wishes to test their investment strategy, without them knowing the back end calculations or the input data that is used for the calculations.

It also implements a color scheme, where if the user made a profit, then the number would be displayed in “green”, if no profit then the number will be displayed in “red” and if no profit or loss then in “grey”.

This feature holds paramount importance as at the end of the day, the main job of this webapp is to tell the user if he/she would have made money and how much.

Verification - The web app is first going to be used by Don as a prototype. He already has an excel sheet where the calculations have been made for certain set of inputs. Hence all we need to do for verification is top compare the final number he has, with what we have. If they match and the correct color scheme is applied, based on profit, loss or no profit; we know that this functionality performs correctly.

Graphical Performance of Portfolio

Priority - 2

Details - This feature deals with the pictorial representation of the net change in the portfolio value with each day. This line graph would show the user how the portfolio performed over time. It also helps see when was a sudden loss or gain occurred, hence helping them relate the trend to the specific event that occurred on that day. This feature will therefore help the user to make informed relations and draw conclusions about how the recurrence of such a future event may impact the performance of those options.

Also, it serves as a base for the pop-up window feature that we will discuss next. The user can click on the graph and choose a date. A pop-up window will open to show the growth/decline of each option for that day, as well as the value of portfolio for that day.

Verification - Since it is going to be used as a prototype first and we already have a set of calculations ready. We can just have those portfolio values and create a graph through a few different platforms. If each of the graphs (i.e. the test graphs that we built on different platforms and the graph that are webapp builds) are identical, then it is verified that the webapp is able to create the correct graph for the data points we have.

Display Daily Performance

Priority - 3

Details - The user may not only wish to see the net growth or loss, but would also like to explore what led to that specific gain or loss. Therefore, this feature is necessary. The user can look up the performance for each option for a particular day. When the user clicks on the graph, a pop-up window will open. This window will consist of the net profit or loss that each option made for that day. It will also show the net growth or loss till that day.

This feature becomes necessary to pinpoint days when the major changes in the trend occurs. Also it helps to notice how the options performed specifically for a day, when some major event may have occurred in the past.

Verification - The webapp will be first used as a prototype for the data. Since there already exists examples of strategies and their net profit/loss; we can just compare the daily change in portfolio-value, to them and verify if the correct data is being displayed in the pop-up.

Non-Functional Requirements

The following section deals with the non-functional requirements for our webapp. It will talk in detail about the traits that will discuss the operational side of the app and not the behavioral. Therefore, we will discuss cross-platform compatibility, response time, and maintainability as our non-functional requirements.

Maintainability

Priority - 1

Details - The app for now is going to be used for 2 sets of strategies, namely
“Buy call” & “Buy put” which is something that the customer would like to expand upon so more strategies can be added. This makes more testing of option strategies possible. Therefore it needs to be built in a way that there is future scope of additions.

Verification - We can build the app in a way such that it first becomes functional with just 1 strategy. Then we ourselves need to be able to add 1 more strategy. This would complete our part for the project as well as give us proof that strategy-addition is possible. Finally, the customer can go ahead and add one more strategy to verify that it is maintainable.

Cross-browser compatibility

Priority - 2

Details - The app is going to eventually be published for multiple customers to be used. This would mean that the app would need to accessible through various browsers such as Google Chrome, Safari, etc.

Verification - We can write unit tests using Selenium to test whether the webapp is functioning properly on each browser.

Response time

Priority - 3

Details - When the user inputs the set of information for the portfolio they wish to test for, the webapp response time should be less than 10 seconds. If the time exceeds this limit, then our application may not be seen as user friendly.

Verification - We can have a simple timer built that would calculate the time it takes to register the responses. Hence that can be used to verify the response time we hope to achieve.

Feasibility

In order to ensure that our program is not only complete as well as exceeding specifications, our team has agreed to have the project completed early allowing time for adjustment or improvements if needed. We plan to focus on receiving data into our program in the early stages of the project.

The high level overview of our timeline for this bare-bones version described is to create a product which can gather input from the given database, then spend more time manipulating the input to the specifications. Since our client has requested that we implement our program to use the values given in their database as well as functions they are already using in their own products, we have agreed to ensure the accuracy of the program by starting with an accurate foundation of database input. After this is confirmed to be correct, we will work with the client further in order to manipulate this data for picking the correct options in the user's portfolio according to the given algorithm. This algorithm relies heavily on gathering the correct information from the client specified method of the given "calendar" function as well as database input. After having all the correct information to display, we will spend the rest of the time on the appearance of the project and making the interface easy to use. We will also follow up with any clarifications at this point from the client to make last minute adjustments.

Requirements and environment setup (9/23 - 27)
Data Input Verification and Calculation (9/27 - 10/15)
    Portfolio Setup implementation (9-27 - 10/3)
    Date Range Selection implementation (10/3 - 10/9)
    Strategy Selection implementation (10/9 - 10/15)
Net Gain/Loss Result (10/15 - 10/21)
Data-Graph Development (10/21 - 11/14)
    Graphical Portfolio Trend (10/21 - 11/1)
    Daily Portfolio Performance (11/1 - 11/14)
Buffer - Extra Time to Clarify With Client if our product needs advancements (11/14 - 11/22)
Expected Completion date (11/22)

If we had an enhanced version, we would want to extend it to add more strategies. We also would like to have different types of graph options, the ability to download the results as an pdf file and also have comparare-strategy function in there. These are not of paramount importance for our current project, but it would be something that would enhance the app in the future and add more functionalities to it.

Conclusion

The above mentioned information talks about how we plan to execute the development of our project. It gives a basic overview, talks about our target & development environments, a system model, the user interaction plan, the functional requirements, the non-functional requirements and the feasibility for this project.

This product will take an input of a time period, strategy option, principle amount and allow them to create a portfolio of options in which they should choose to invest. Given an amount of money, the user will be able to view the previous net gains/loss and will choose if our algorithm shows an option they agree they should invest in.

We hope that this app can be enhance the user's experience of visualizing and analyzing the past data. Enabling them to make an informed investment decision in the future.

APPENDICES:

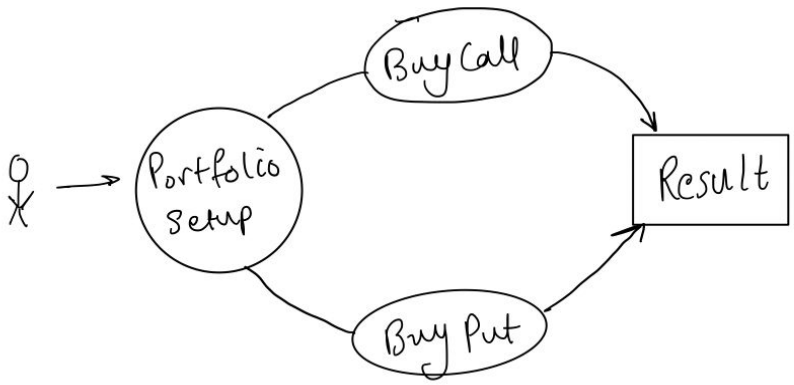
Github Link

Project Link: https://github.com/TejPatel98/cs_499_project

Logs:

- Justin Luttrell:
  - https://github.com/TejPatel98/cs_499_project/wiki/Developer-Logs:-Justin-Luttrell
- Josh Luttrell:
  - https://github.com/TejPatel98/cs_499_project/wiki/Developer-Logs:-Josh-Luttrell
- Tej Patel
  - https://github.com/TejPatel98/cs_499_project/wiki/Developer-Log-:-Tej-Patel
- Tom Busby
  - https://github.com/TejPatel98/cs_499_project/wiki/Developer-Log-:-Tom-Busby
- Blake Sweet
  - https://github.com/TejPatel98/cs_499_project/wiki/Developer-Log-:-Blake-Sweet

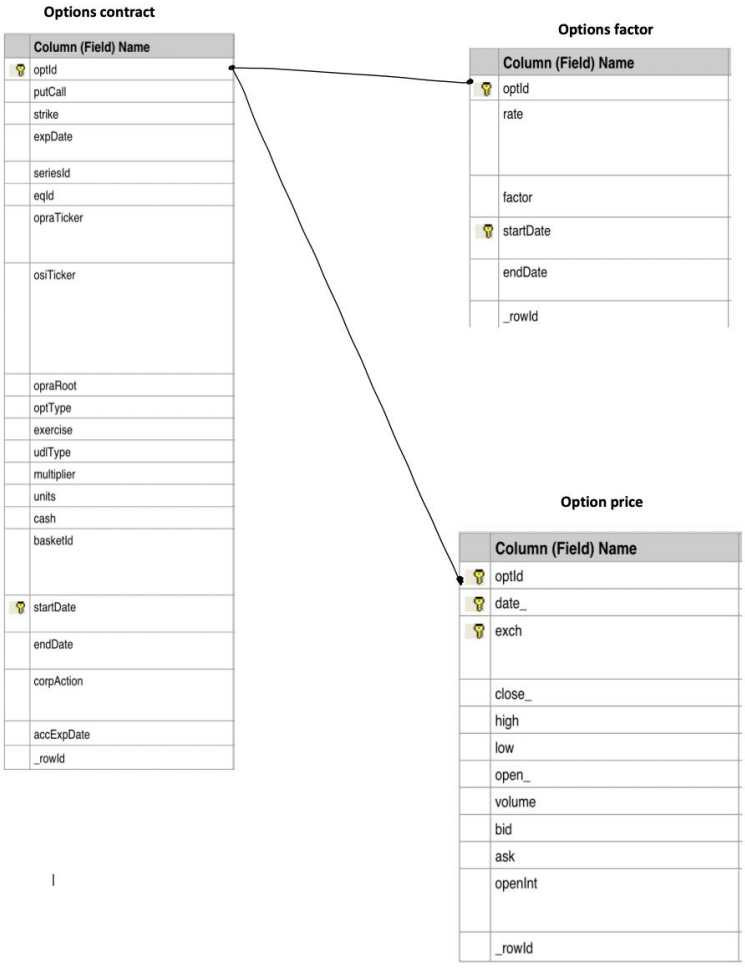
Use Case Diagram



This diagram shows the use cases for our webapp. The use case presents itself as simple since it only consists of a user inputting information into a form and then submitting to get the results back in a graphical format. The user

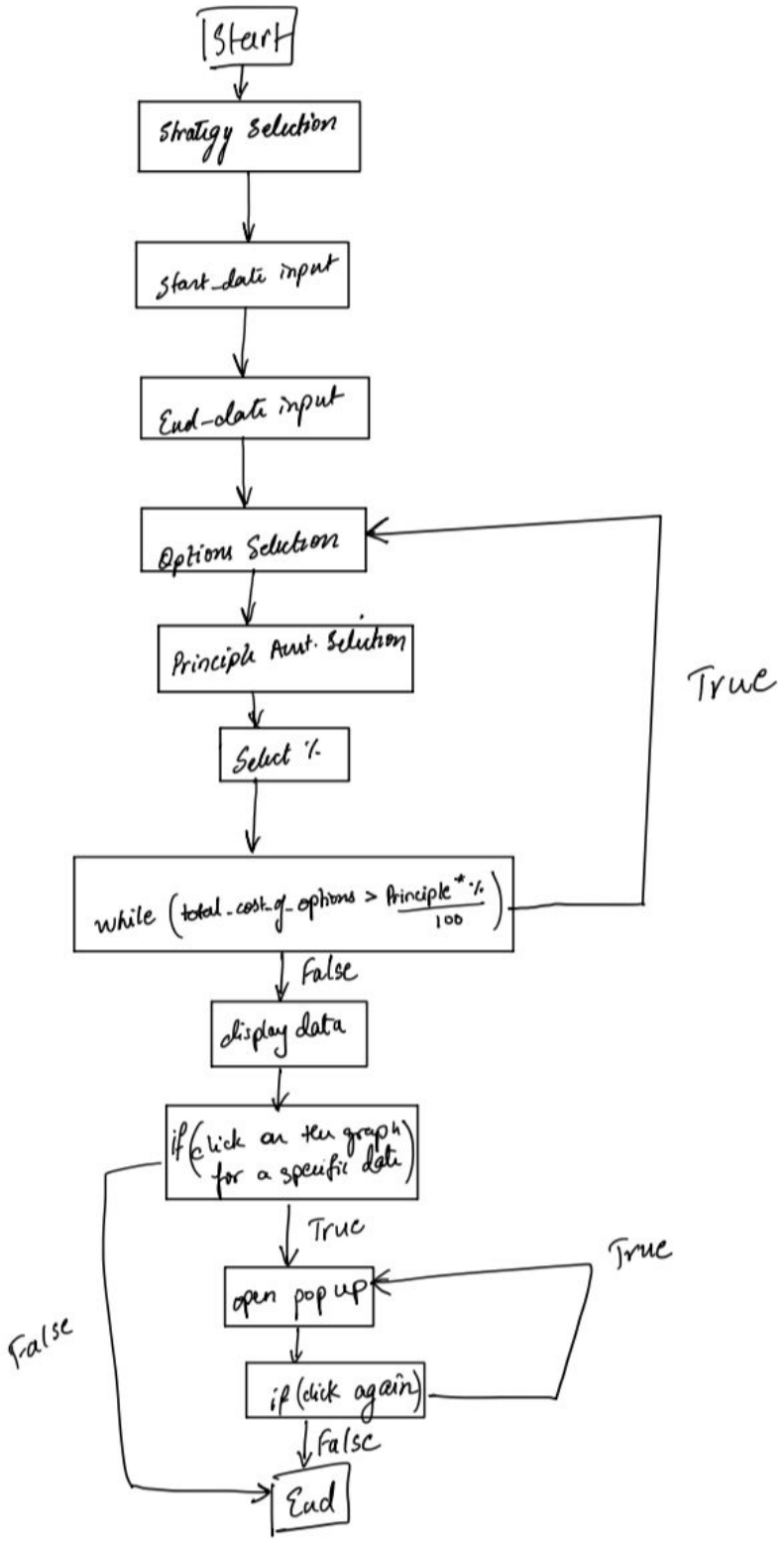
does have the option to choose either the "buy call" or "buy put" strategy but will still see the results in this same format.

ER Diagram



The above diagram shows the relevant tables in the existing OVS database that we will be using. Our customer has licensed the use of the database that will allow us to gather the relevant data to make calculations of the simulation to present to the user.

Logic Flow Diagram



This diagram shows the logical flow of the webapp.