Leveraging the Wisdom of the Crowds

Bachelor End Project - 2020

Leveraging the Wisdom of the Crowds

Erwin Dam
Marc Droogh
Jeroen van Steijn

in partial fulfillment of the requirements for the degree of

Bachelor of Science
in Computer Science and Engineering

at the Delft University of Technology,
to be defended publicly on 3 July 2020 at 16:00

made possible by

Client	Prof. Christian Doerr
Coach	Dr. Przemysław Pawełczak
Bachelor Project Coordinator	Ir. Otto Visser
	Thomas Overklift

TU Delft

Preface

This bachelor's thesis was created by Erwin Dam, Marc Droogh, and Jeroen van Steijn as part of the requirements of the Bachelor Computer Science and Engineering at the Delft University of Technology. In a period of one academic quarter, we have examined the potential and problems of prediction markets and developed a prototype which aims to prevent the aforementioned problems. This project was commissioned by the Cybersecurity Group at the Delft University of Technology.

We would like to thank Dr. Przemysław Pawełczak for being our project coach and for providing clear feedback, and to Otto Visser and Thomas Overklift for coordinating the Bachelor End Project and approving this group.

In addition, our thanks go to everyone at the Delft University of Technology and its Cybersecurity group who helped to user test the product.

Lastly, a special thanks goes out to Prof. Christian Doerr, who acted as our client, for his guidance, support, and enthusiasm over the course of our project, as well as his feedback on our progression in our weekly meetings.

Erwin Dam
Marc Droogh
Jeroen van Steijn
Delft, June 2020

Summary

Wisdom of the crowds is the idea that groups of people can collectively make wise decisions. Research suggests that these crowds can even outsmart experts. To gather the wisdom of the crowds, this project utilizes a prediction market. To successfully gather the wisdom of the crowds, a prediction market has to overcome serious challenges, such as gathering a large and active user base, and deciding on a fair initial market value. The main goal of the project is to create a prediction market that can overcome these challenges and successfully gather the wisdom of the crowds.

Research has been done in the field of prediction markets. This process started with researching the theory behind prediction markets, the wisdom of the crowds. After that evaluating existing prediction markets and reviewing literature related to those markets was useful. Before and during the research phase, clear goals were set for the project, together with a clear set of requirements. These goals can be divided into: leveraging the wisdom of the crowd, solving problems associated with prediction markets and developing a product that is easily maintainable.

The final product reaches the goals of the project and meets the requirements. The prediction market correctly aggregates the estimations of users on the market, and provides probabilities on real-world events. These probabilities are contained in the values on the market. The prediction markets solves the problems encountered on other prediction markets. The project makes use of gamification, an automated market maker and a reward system to correctly initialise market values. The system was thoroughly tested and developed with maintainability in mind.

Contents

Preface i

Summary ii

1 Introduction 1
1.1 Document Structure . . . 1

2 Problem Definition and Research 2
2.1 Problem . . . 2
2.1.1 Problem Definition . . . 2
2.1.2 Problem Analysis . . . 2
2.2 Existing Systems . . . 3
2.2.1 Different Applications of Prediction Markets . . . 3
2.2.2 Existing Prediction Markets . . . 4
2.3 Leveraging the wisdom of the crowds . . . 5
2.3.1 Applications . . . 5
2.3.2 Quantity . . . 5
2.3.3 Quality . . . 6
2.4 Problems with prediction markets . . . 7
2.4.1 Reaching critical mass . . . 7
2.4.2 Market liquidity problems . . . 7
2.4.3 Market initialisation . . . 8
2.4.4 Market Manipulation . . . 8
2.5 Market Makers . . . 9
2.5.1 Continuous Double Auction . . . 9
2.5.2 Automated Market Maker . . . 9
2.6 Gamification . . . 10
2.7 Goals . . . 10
2.7.1 Leverage the Wisdom of the Crowds . . . 10
2.7.2 Solve Problems Associated with Prediction Markets . . . 11
2.7.3 High Maintainability . . . 11
2.8 Requirements Analysis . . . 12
2.8.1 Requirements . . . 12
2.8.2 Success Criteria . . . 13

3 System Design and Specification 14
3.1 Back End . . . 14
3.1.1 Application Programming Interface . . . 14
3.1.2 Database Migrations . . . 15
3.1.3 Authentication . . . 15
3.1.4 Transactional Emailing . . . 15
3.1.5 File Structure . . . 16
3.2 Front End . . . 17
3.2.1 Front End Framework . . . 17
3.2.2 Atomic Design . . . 17
3.2.3 State . . . 17
3.2.4 Browser Support . . . 18
3.2.5 Styling . . . 18
3.2.6 Dependencies . . . 18
3.2.7 File Structure . . . 19

3.3 Database . . . 19
3.4 Infrastructure . . . 20
3.4.1 Continuous Integration and Continuous Delivery . . . 21

4 Implementation 22
4.1 Primary Features . . . 22
4.1.1 Market Maker . . . 22
4.1.2 Trading Screen . . . 23
4.1.3 Visualization . . . 24
4.1.4 Dashboard . . . 24
4.1.5 Rewards System & Kick-Starting Markets . . . 25
4.1.6 Administrator Capabilities . . . 25
4.1.7 iFrame Embedding and Sharing . . . 26
4.2 Secondary Features . . . 26
4.2.1 Email Confirmation . . . 26
4.2.2 Password Forgot . . . 26
4.2.3 Statistics . . . 27
4.2.4 Privacy . . . 27
4.2.5 Middleware Options . . . 27

5 Software Quality 28
5.1 Static Analysis . . . 28
5.2 Testing Approach . . . 28
5.2.1 Back End . . . 28
5.2.2 Front End . . . 29
5.3 SIG Analysis . . . 29
5.3.1 First Submission and Metric Definitions . . . 29
5.3.2 Second Submission and Achieved Improvements . . . 31

6 Process 33
6.1 Development and Planning Methodology . . . 33
6.2 Communication . . . 33
6.2.1 Daily Stand-Up . . . 33
6.2.2 Weekly Sprint Review . . . 34
6.3 Unexpected Challenges . . . 34

7 Discussion 35
7.1 Usage Recommendations . . . 35
7.1.1 Creating Questions . . . 35
7.1.2 Branding and Marketing . . . 36
7.2 Ethical Implications . . . 36
7.2.1 Gamification and Gambling Addiction . . . 36
7.2.2 Predictive Data and Influence . . . 36
7.3 Further Research Suggestions . . . 36
7.4 Further Development Suggestions . . . 37

8 Conclusion 38

Bibliography 39

Appendices 41

A Project Info Sheet 42

B Project Description 43

C Requirements Document 44

1 Introduction

Wisdom of the crowds is the idea that groups of people can collectively make wise decisions. To do so, the group does not have to consist of experts on a certain topic. Quite the contrary: a large group of uninformed people can collectively outsmart experts on a topic [35]. By combining the opinions of individuals in a crowd, the wisdom of this crowd can be leveraged. An effective way to leverage the wisdom of the crowd on real-world probabilities is to make use of prediction markets.

Prediction markets are futures markets, which are designed to aggregate information and produce predictions on future events [8]. Information content in market values can give insights into real-world probabilities on future events or aggregate information for decision making [38]. Despite the promising results found in research, prediction markets have to overcome serious challenges to produce reliable predictions [3, 38].

The purpose of this project is to create a prediction market which more optimally leverages the wisdom of the crowds. In order to do so, the created prediction market has to overcome challenges which were experienced on existing prediction markets. The problems referred to can be found in section 2.1.1. Solutions which were implemented in this project to solve these problems are gamification, market kick-starting mechanisms and a market maker.

1.1 Document Structure

Before the development of the prediction market could start, research regarding the wisdom of the crowds and prediction markets had to be done. In Chapter 2 the Research Phase will be discussed and summarised. This consists of literature research towards the theory behind prediction markets as well as research on existing prediction markets. The research that was conducted, combined with our personal experiences and the experiences of our client on existing markets, lead to the design of the developed prediction market. The chapter which follows, Chapter 3 will expand on how the system is designed. After this, the implementation of the project will be discussed in Chapter 4. In Chapter 5 it will be explained how the software quality of the developed product is assessed. Next, the development process will be discussed in Chapter 6. After this follows Chapter 7 which discusses the project's findings, as well as usage recommendations and suggestions for further research. Lastly, a reflection on the final product can be found in Chapter 8.

2

Problem Definition and Research

The first two weeks of the project were dedicated to research. The first step in this process was defining the problem and analysing it accordingly. The concrete problem, as well as the process of defining and analyzing it, will be described in Section 2.1. After evaluating the problem, existing prediction markets will be evaluated in Section 2.2. Section 2.3 elaborates on the theory behind prediction markets, the principle of the wisdom of the crowds. After that, problems related to prediction markets, which were found in the research process, will be defined in Section 2.4. The section after that, Section 2.5, describes how market makers work, and how this can solve the liquidity problem described in Section 2.4. In Section 2.6 gamification is discussed, as gamification plays a role in solving the critical mass problem described in Section 2.4. Section 2.7 will define the goals of the project, after which the last section of this chapter, Section 2.8, lists the specific requirements which were formulated and which the product needs to meet.

2.1 Problem

As mentioned in previous section, the first step in the research process consisted of defining the problem and analyzing the defined problem. In Section 2.1.1 the defined problem is described to provide a clear view of the purpose of the project. After that, in Section 2.1.2, an analysis of the defined project will be given. Together, these sections motivate the existence of the project and explain the context of the project.

2.1.1 Problem Definition

Markets that produce predictions on future events have been available for quite some time and have been studied closely. Despite the promising results found in research, there are currently no prediction markets which produce accurate results on many different topics. Whilst the Iowa Electronic Markets and similar systems do perform great on some markets, they do lack engagement on others.

The goal of this project is to develop a prediction market which is able to gather insights in probabilities of real-world events. In order to do so, this project has to overcome challenges that caused other markets to fail. At the end of the project, the Cybersecurity Group at the TU Delft can determine whether the prediction market developed should be maintained and developed further.

2.1.2 Problem Analysis

Prediction markets have to overcome serious challenges to produce reliable predictions [38]. For a prediction market to produce reliable predictions, in other words, to correctly leverage the 'Wisdom of the Crowds', it is essential for a market to have a big, motivated and diverse community of participants [29, 38]. Creating a well-performing prediction market can, therefore, partly be interpreted as building a prediction market which could attract a large number of traders.

Furthermore, the product should not be as prone to problems which prevented other prediction markets from optimally leveraging the wisdom of the crowds. These problems are related to liquidity, market initialisation and market manipulation.

2.2 Existing Systems

Multiple prediction markets have been developed in the past, and analyzing these could provide insights into the performance of prediction markets in practice. Different prediction markets have been developed for different purposes. Multi-billion-dollar corporations and the United States Department of Defense have used prediction markets for decision making purposes [3]. However, this project focuses mainly on prediction markets which give insights in the probability of real-life events. Therefore we will only look at this kind of prediction markets in this section. On most of the prediction markets, there is an all-or-nothing payout structure, which means there is only one correct answer to a market and only that answer gets a reward. Other examples are prediction markets where the payout is based on an index or a spread. This project will focus on the all-or-nothing payout structure. In this section, we will first distinguish various kinds of prediction markets. After which specific prediction markets will be discussed.

2.2.1 Different Applications of Prediction Markets

Many prediction markets which differ in implementation have been developed, but these markets share some design principles. First, the accessibility of prediction markets will be touched upon, after which the kind of currency used on markets is discussed. These properties have a strong influence on the prediction market.

Accessibility

Prediction markets differ in their degree of accessibility: where some prediction markets are open to the public, other markets are only accessible to a selected group of individuals [3]. This choice of accessibility is mostly motivated by the goal of the prediction market: If, for example, the prediction market is developed to improve decision making in a company, most of the time only the informed perspectives of the employees are relevant, and therefore only the employees of this company are part of the market community. For prediction markets designed to get insight into real-world probabilities however, also uninformed perspectives are relevant, thus a market should gather as many perspectives as possible to optimally leverage the wisdom of the crowds [39]. As a result, these prediction markets are most often openly accessible.

Currency

A distinction can be made between real-money markets and fictional-money markets. There are motivations for both of these implementation choices: some theories suggest that using real-money would motivate information discovery since individuals are taking a financial risk when investing in the markets [8]. This would then increase the accuracy of the markets [29]. Using play-money markets has two benefits compared to real-money. Firstly, since it is play-money, individuals do not have to bear financial risk. This could then result in a bigger trading community since the barrier of entry of the market platform is lower [29]. Secondly, in a play-money market where everyone starts with the same financial resources, people who made successful trades in the past have more financial resources than others and have more impact on the market in the future. If this weighing of perspectives has a positive impact on predictive ability still remains to be seen.

2.2.2 Existing Prediction Markets

Existing prediction markets which are able to give insights in real-life probabilities have already shown potential. Multiple markets have been able to produce remarkably accurate predictions [3].

The following subsections will describe prediction markets in various fields and with different characteristics. In the first subsection, the prediction market which is considered to be the first successful prediction market, the Iowa Electronic Markets[1], will be discussed [38]. The subsection after that focuses on a prediction market which was created specifically for betting on sports, called Tradesports[2]. NewsFutures was one of the most popular trading websites a decade ago and will be looked at in the next subsection. To show prediction markets can be successful in less common areas for betting, the last subsection will discuss the Hollywood Stock Exchange[3].

Iowa Electronic Markets

The Iowa Electronic Markets is considered to be the first prediction market [38]. This prediction market runs different kinds of markets designed to predict the outcomes of elections, earnings reports and the value development of securities. Researches found that the Iowa Electronic Market outperformed opinion polls in predicting results of political elections systematically [6, 7, 10].

Tradesports.com

Tradesports.com [4] is a prediction market build by a company which tries to make a profit from operating it. The market can be seen as a gambling website, which is targeted at sports events. Users on the market trade with real money, and there is an all-or-nothing payout structure, a contract is valued at a hundred dollars if it is correct and zero otherwise. The traders trade with each other with no intermediary. Tradesports makes money by charging a small fee on each transaction [38]. The results of this market are very accurate, an example of this is the prediction of the National Football League (NFL) game outcomes. In a sample of 208 NFL games, between 4 September 2003 and 8 December 2993, 135 times the team favoured on Tradesports won. The average price for the favourite team on Tradesports was 65.1 for this sample, which is a very good estimation for the 66.8% win rate [29].

NewsFutures.com (no longer available)

NewsFutures was not only a prediction market but also a provider of prediction market software and services. The company was founded in 2000 but seized to exist in 2010. In this period of time, the prediction market maintained contracts on over 120,000 real-life events in a wide variety of topics. The prediction market consisted of contracts on politics, finance and sports. The payout structure on contracts was the same as it is for Tradesports, but instead of using real money, the users traded with play money on the market. Because of this, the traders could not gain or lose any money on this platform. The users received a fixed amount of play money on registration and received a small amount once the trader fell below a certain level of net worth. The most successful traders on this platform could bid on rewards worth a few hundred dollars at the end of each month using their aggregated worth on the platform. The product has been compared to Tradesports. When the same sample of 208 NFL games as taken for Tradesports was taken, it did slightly better with 139 out of 208 (66.8%) favourite team victories, and an average trading price of 65.5 for the favourable team [29].

Hollywood Stock Exchange

The Hollywood Stock Exchange (HSX) trades fictitious shares based on films and film stars. The currency on the market is known as "Hollywood dollars" and is obviously play-money, thus does not carry any value outside of the platform [20]. The platform uses an automated market maker to facilitate trades [31]. The payout structure is different from the other mentioned markets, players receive one Hollywood Dollar for every million real dollars of domestic revenues that are reported [20]. The players on the platform are limited to ten-thousand shares at a time. On the HSX users can also trade options which have the same payout and behaviour as the ordinary stock market. Apart from this, next to these basic products the HSX contains all-or-nothing payout contracts which are based on real-life awards. An example of the accuracy is how the HSX managed to appoint all eight of the eight winners of the Academy Awards, whilst the Wall Street Journal was only able to predict six of them by polling all voters [20].

[1] Iowa Electronic Markets: https://iemweb.biz.uiowa.edu
[2] TradeSports: https://www.tradesports.com
[3] Hollywood stock exchange: http://www.hsx.com
[4] Tradesports: https://www.tradesports.com

2.3 Leveraging the wisdom of the crowds

Wisdom of the crowds is the idea that, under the right circumstances, groups of people can judge situations more intelligently than any of their parts individually, and often more accurately than any given expert. Even if most people in the group are not well informed, collectively they can still make wise decisions. This is the underlying principle of prediction markets.

Wisdom of the crowds relies on a mathematical truism: when a large group of diverse and independent people estimate something, these estimations contain an error. When these estimates get aggregated, however, the errors of these estimates cancel out (total variance is divided by the number of data points) and only the expected value remains. This expected value can be extremely accurate in some situations but may not be so accurate in others [35]. The effective performance of the wisdom of the crowds is heavily dependent on the number of entities that are part of these crowds, and the quality of the collective they form [18]. Sections 2.3.2 and 2.3.3 will elaborate on when a process is suitable to leverage the wisdom of the crowds.

2.3.1 Applications

Systems based on the wisdom of the crowds can provide insights in probabilities for closed real-life questions. It may, for example, prove to be useful for the task of preventing biases which are present in most large-scale data sets that are sourced from socially structured platforms such as social media [4, 16].

Prediction Markets

Prediction markets use the principle of the wisdom of the crowds to provide insights into real-world probabilities. On all events of which a clear payout structure can be defined, market values can convey information about the possible outcomes. To define a clear payout structure for the market an event needs to have a measurable future outcome. For example: "Who will win the presidential election?" has a measurable outcome, but "Will it be nice weather tomorrow?" leaves a lot of room for interpretation.

Deplhi Technique

Another application of the wisdom of the crowds is the Delphi Technique. The Delphi Technique is a structured method which was developed as a systematic forecasting method. This method relies on a panel of experts. The theory behind the method is based on the idea that forecasts of a structured group of individuals produce forecast results with higher accuracy than unstructured groups. In this technique, the experts in the panel receive an anonymised summary of the forecasts after each round of forecasting. In between the rounds, the experts should re-evaluate their answers with the new knowledge of the anonymised summary. The process should reduce the difference between the answers and converge to the correct answer [21].

Who wants to be a millionaire

A real-world example of wisdom of the crowds was observed at the popular TV show, "Who wants to be a millionaire?", where experts gave the right answer 65% of the time but the audience outperformed them and managed to give the correct answer 91% of the time [35].

2.3.2 Quantity

There exist many ways of gathering insights into real-world probabilities, a prediction market is only one of them. Other examples are queries to professionals, predictive algorithms, surveys and the mining of public data. Some of these methods make use of the wisdom of the crowds too, but many of the methods suffer from systematic and random errors [12, 30, 34]. The majority of this is caused by the fact that each individual is subject to errors, thereby causing small samples to have considerable differing prediction accuracies [18]. Within a large group of people, random errors can be accounted for more robustly, due to the law of large numbers causing convergence to the expected value [18]. Therefore, the wisdom of the crowds also only produces reliable outcomes once the size of the crowd becomes considerably large.

Since the questions that prediction markets seek to provide insights on are probabilistic, often relatively dependent on perspectives, and suffer from the random errors as described above, methodologies that depend on small sample sizes will most likely not provide sufficient accuracy to be credible. Therefore, the main challenge for prediction markets to optimally leverage the wisdom of the crowds, is to reach numbers great enough to cause truly usable results [17].

2.3.3 Quality

The correctness of the outcomes produced by leveraging the wisdom of the crowds also depends on the quality of this crowd. The quality of the crowd does not refer to the expertise of the individuals on a certain topic, but rather to how the crowd is constituted and how its individuals interact with each other. Correct composition of a crowd is important to be able to overcome problems caused by collective behaviours and problems from crowd psychology, as well as the biases that are a result of these behaviours and problems.

Groupthink

A problem from group psychology, which can negatively impact the performance of wisdom of the crowds is Groupthink. Groupthink is a phenomenon which degrades the quantity and quality of perspectives within a community by aligning them to a common vector [37]. It prevents the raising of controversial issues within a community and may result in a flawed or even dysfunctional decision-making process [37]. Therefore groupthink is highly undesirable in processes which make use of the wisdom of the crowd principles. To prevent groupthink, it is important for members of the community to not be influenced by others in the community in making their decision [22].

A prediction market which is completely open to the public will most likely not be very sensitive to biases induced by data aggregation or control of the community, since there is very limited control over the community when the market is openly accessible. On top of that, a prediction market uses values of a market system to aggregate data and is therefore unlikely to induce bias [39]. The competitive nature of markets, combined with the independent behaviour resulting from individual profit maximisation, complement the system's ability to satisfy the conditions that allow the concept of wisdom of the crowds to function. To maximise the profit gained by trading in a market system, individuals have to be independent. This is important because it is in the individuals best interest not to influence other individuals into choosing their perceived potentially 'successful' trade as this would at the very least not provide them with an increase in worth.

Diversity

One of the predominant features of the wisdom of the crowds is the fact that a collective consisting of many individuals evaluates a problem from more perspectives than an individual or organised group feasibly can and is able to weigh them proportionally. Diversity in perspectives and thus backgrounds of individuals within the collective is, therefore, an important factor in making the wisdom of the crowds a feasibly usable evaluation method [28, 35].

Bias

Using the aforementioned conditions, it should be noted that a system which is meant to use the wisdom of the crowds ought to limit bias induced by the system itself. Important matters regarding this topic are the control of the community and the aggregation of their data: aggregation should not induce any biases, and there should be no, or as little as possible, centralised control over the community.

2.4 Problems with prediction markets

Problems described in this section are related to prediction markets in theory or to ones which have occurred in existing prediction markets. The theory of the first problem, reaching critical mass, was explained in the previous chapter. In Subsection 2.4.1, the problem of reaching critical mass will be evaluated further. After that, liquidity problems will be explained in Subsection 2.4.2. Not only will it be explained what liquidity problems are in general, but also how it influences the accuracy of a prediction market. In Subsection 2.4.3 a description of the market initialisation problem is given. In the last subsection, Subsection 2.4.4, the focus is on market manipulation. Whilst this problem has not occurred in many prediction markets in practice, it is a problem which is heavily associated with prediction markets.

2.4.1 Reaching critical mass

An important aspect of the wisdom of the crowds is to gather the needed amount of perspectives and to process them correctly to get an overall opinion as described in section 2.3. The biggest challenge for prediction markets is to reach critical mass, quantified as the active trading community and thus the number of perspectives. Other challenges prediction markets face are often related to the challenge of reaching critical mass.

Size

It is hard to tell what the exact critical mass for a prediction market is or how it should be reached. However, prediction markets that have been developed and maintained before, give indications at what size a prediction market can produce accurate results. First, we will take a look at the performance of the most popular prediction market in research, the Iowa Electronic Markets [5]. This market outperformed polls by 71% over sixteen years. The market did this with having at most 790 active traders but outperformed polls 73% of the time with only 592 active traders [7]. Taking in consideration, however, that the active trading group on this market consisted mostly of people inside the state of Iowa, according to section 2.3.3, markets could be even more accurate when the active trading community is more diverse. STOCCER [6] which uses play-money to predict the outcome of soccer games, performed as well as predictions based on betting odds [23]. However, it must be noted that the best 100 traders on the STOCCER market were able to earn some real-world rewards in a lottery, therefore there was still a financial incentive for traders to use the market. STOCCER managed to achieve these results with around 1260 traders [23].

Incentive

Reaching critical mass is a problem for all public prediction markets. Conventional stock markets have clear incentives to participate in the market: they can be used to increase the value or longevity of one's capital trough trading or dividends and may be used for hedging. Betting, such as sports betting, also has clear incentives: entertainment, competition, and often the chance to improve ones capital [19]. Prediction markets that do not provide the opportunity to exchange virtual currency for real money or vice versa, do not benefit from the incentives based on monetary gains. Companies which develop prediction markets for internal usage can tell their employees to trade on the markets and therefore can easily reach critical mass. Public markets, however, do not have this possibility and have to create additional incentives for users to trade.

2.4.2 Market liquidity problems

Market liquidity at a traditional stock market can be loosely defined as how easy it is to acquire and sell a certain stock [5]. This also indicates that in an illiquid market there will most likely be fewer trades than in a market that is more liquid. In prediction markets, a trade in the market can be seen as an individual putting their perspective on a situation into the market. While it is relatively easy to aggregate the results of a survey, it is much harder to process the results of a prediction market. In prediction markets, the price of the different stocks in a question market represents the wisdom of the crowd. However, when there is low liquidity, new information is not correctly reflected in trading prices and as a consequence, the market does not reflect the correct combination of perspectives. Not only could it reflect the perspectives poorly, but it also could take away the incentive for users to participate in the market, as they might feel like the markets are abandoned, decreasing their incentive to trade [23]. Research has already shown that prediction markets which have high liquidity produce outcomes with a higher accuracy [31].

[5] Iowa Electronic Markets: https://iemweb.biz.uiowa.edu
[6] STOCCER: www.STOCCER.com

2.4.3 Market initialisation

Prediction markets contain different question markets, all of these question markets also have a moment at which they open for trading. There are different ways to determine the starting value of answers in a new market. On the conventional stock market, this is known as an initial public offering (IPO). In case of an IPO, a company gets help from investment banks and other institutions to guide the process and come up with a reasonable price. For a prediction process, this process would cost too many resources in addition to the fact that there are probably no institutions which could reasonably provide these services for prediction markets [36]. Therefore a prediction market cannot adopt the method from the ordinary stock market and has to come up with another initialisation method.

Equal Starting Prices

A naive method to start a market is to make all the answers equal in starting price, and therefore equal in the predicted chance of being the ultimate answer to a question. In most cases, this is not a very good indication of reality. A result of this method is that the price on many question markets is likely to heavily fluctuate after new market open since it is obvious for many traders that the price is incorrect. To keep traders from maximising profits by predominantly only buying stocks in the starting phase of a market, the initial values of the shares should be as realistic as possible. Thereby reducing the chance to realise unrealistic and practically risk-free profits at the start of a market. Unrealistic starting prices prevent the market from functioning correctly since traders will try to find the largest discrepancies between starting prices and market values. If the difference between the initial price and the market value is too big, it would be in the best interests of traders to wait for a new market to open and maximise their profits in the first moments after market conception. This drains available resources and disincentivises trading on markets when unrealistic conditions are not present, thus preventing normal market functionality.

Administrator Determined Price

A different way to start a market is to let the administrator who opens the market decide what the starting values of the answers should be. This way the chance of arbitrage at the beginning of the market could be lower. However, this is in contradiction with Section 2.3, since the judgement of the expert is not very trustworthy. This would also mean a market on specific topics can only be opened by experts on this topic. This is undesirable since it ncites the variety of question markets down significantly. As a consequence, the prediction market will most likely receive attention from a less varied and smaller community.

2.4.4 Market Manipulation

A common problem associated with prediction markets is the idea that prediction markets could be easily manipulated. The incentive to manipulate the market could either be to gain capital or to steer the market in a specific direction of personal preference. In order to incentivise fair trading practices on the question markets and to prevent manipulation of insights gathered from the data, it is important to discourage and/or prevent market manipulation.

Although several attempts to manipulate prediction markets are known, none of these attempts had a big impact on the price of the respective markets, except during a short transition phase. An example is how Strumpf tried to manipulate a prediction market in 2004 by placing random $500 bets on the Iowa Electronic Markets and traced their effect. This showed that the effects on the price were only temporary [33]. In the end, the thinness of the market determines the impact an individual can have on a market, and therefore how sensitive the market is for manipulation [38].

A market can be made less manipulable by providing free entry in the market and limiting the maximal amount of shares an individual can possess in the market. Providing free entry in a market offsets effects of manipulators since it lowers the threshold for people to participate. This way, the market gets thicker. Limiting the maximum amount of shares an individual can possess in a market limits the influence of an individual on the market on the short term. In addition, several implementation choices can be made to provide additional prevention methods, such as limiting the amount of tradable currency [39].

2.5 Market Makers

Given a market system, there are multiple ways to facilitate the trade of shares between parties. The most common system is keeping a market order book, this is also known as the continuous double auction [32]. In this case, the market operator maintains a list of buy and sell orders. When there is a buy order which agrees on the price with a sell order, the trade goes through. All of the existing markets discussed in section 2.2 make use of this system. Market makers on such markets can either be other traders or a company which has an agreement with the stock exchange to provide liquidity for a stock [25]. A different way of organising trade on a market is to make use of an automated market maker which handles all the trades. In such a market, traders do not directly trade with each other. Instead, the traders trade with a market maker which is operated by the market operator. In this case, the stock price is determined by a market scoring rule [9, 14].

2.5.1 Continuous Double Auction

On conventional stock markets, when a trader wants to buy or sell a stock but there is no-one who wants to do the opposite, a market maker steps in and facilitates the trade [25]. This so-called, designated market maker receives a fee from the market operator to provide liquidity. Users, however, can act as a market maker as well by narrowing down the difference between the lowest sell price and the highest buy price of a certain stock. This benefits the users who act as market makers since they can make a small profit doing this. Having market makers on the market makes trade easier and therefore provides liquidity to the market. It should be noted, however, that acting as market maker requires knowledge in this field and most importantly requires a lot of resources as well. Therefore, on prediction markets that use play-money, there is no incentive for companies and users to act as a market maker. As a result of the lack of incentive to operate as a market maker, prediction markets which operate on play-money may have to face the liquidity problems described in section 2.4.2.

2.5.2 Automated Market Maker

Prediction markets that suffer from a lack of market makers which provide liquidity, can make use of an Automated Market Maker (AMM). This market maker facilitates all trade on the market [9] and completely substitutes the market order book. When a trader wants to buy or sell a stock, the AMM takes the opposite side of the transaction. The price of the transaction is determined by the AMM, to do this the market maker uses a market scoring rule [14]. Scoring rules have been used in other fields for forecasting purposes, an example of this is weather forecasting [26]. Multiple scoring rules have been developed, of which the quadratic scoring rule and the logarithmic scoring rule are the most popular. Only the latter can be used to determine the price of a stock on prediction markets [14].

Studies also examined the potential of other AMM systems: Dynamic Parmutuel Market (DPM), dynamic price adjustments (DPA) and a special method used by the Hollywood Stock Exchange (HSX). Due to the confidentiality of the latter, it is hard to compare its performance with the other ones. The confidentiality and patents on the AMM of the HSX make this solution impossible to implement in this project. Of the other three, only DPM uses continuous price functions and is therefore the only one of these three which is not sensitive to arbitrage. Comparing the LMSR and DPM, LMSR seems to deliver the best overall forecasting results, therefore, we will elaborate on this AMM more below [31].

Logarithmic Market Scoring Rule

An automated market maker can use the logarithmic marketing scoring rule (LMSR) to determine the values of the stocks in a market [14]. Prediction markets using a market maker with an LMSR have shown reliable forecasting results [31]. In practice this process works as follows:

Given a question market with n possible ultimate answers:
Market contains n share options.
Each share option i has a number of shares in the market: q_i. Stock value v_i is determined by an LMSR (Logarithmic Market Scoring Rule) [14]

$$v_j = \frac{e^{\frac{q_j}{c}}}{\sum_{i=1}^{n}(e^{\frac{q_i}{c}})} \tag{2.1}$$

Where c is an arbitrary constant which influences the rate of change within the domain $< 0, \infty >$.
Consider for example a question market where: $n = 3$, $q = [12, 30, 58]$

As can be deduced from formula 2.1, c influences how quickly the value of the stocks are able to diverge from $\frac{1}{n}$. Infinite values of c make it impossible to diverge from n, whereas low values cause the option with the most shares to converge to 100 almost immediately (others to 0 respectively).

Cost of Trades

In order for users to be able to buy and sell shares, the market market should be able to calculate the cost for any trade. This value does not equal v_i times the quantity of share option i bought, as the cost of trade has to represent the total amount of change in the value of the market. As such, the cost of all shares, and how they are affected by the trade, have to be accounted for. To do this for a system using the LMSR [14], one can use the formula:

$$\text{cost of trade} = \Delta\text{cost} = (c * \ln \sum_{i=1}^{n} (e^{\frac{q_i}{c}})) - (c * \ln \sum_{i=1}^{n} (e^{\frac{q_{n,i}}{c}})) \tag{2.2}$$

where $q_{n,i}$ is the new quantity of shares in the market for share option i. Again, this quantity is dependent on the arbitrary value c, which influences the rate of change. A higher value for this constant results in slower convergence, and thus a cost of trade which more closely represents the current value of the share.
Determining the value for c to use for the market maker must be done with the expected size of the market in mind: setting c to a great value for small markets may cause the market to change too little, thus making it quite useless for predictions, as shareholders cannot influence the share values sufficiently. Setting c to a small value for large markets may conversely cause markets to converge unreasonably fast, and possibly to unrealistic values due to the spread between buying and selling prices being too large.

2.6 Gamification

Gamification is the process of applying game-like concepts, processes, principles, and rules to situations that are not games themselves. This process allows platforms to change the perception of users on task-like events into positive, possibly fun events. It allows platforms to engage and motivate users more and further using psychological rewards [13], and can be used to activate sleeping resources [24]. While some prediction markets have some gamification elements such as a leader board, there is no relevant research related to gamification on prediction markets. However, there is strong evidence that gamification can be used to extrinsically and intrinsically motivate potential users on a stock trading market [24]. The gamification elements that are responsible for this motivation could be applied to prediction markets to reach a larger user base.

2.7 Goals

This section describes the goals of the project. These goals are a guide for the development process of the product and are used to define clear requirements in Section 2.8. The first goal of the prediction market is to leverage the wisdom of the crowds, described in Section 2.3, correctly. After this, goals that are related to challenges other prediction markets faced are described. Besides these functional goals, the developed product should be easily maintainable for the client after completion of the project. These maintainability goals are described in Section 2.7.3

2.7.1 Leverage the Wisdom of the Crowds

The main goal of the project is to develop a prediction market which is able to leverage the wisdom of the crowds. The prediction market should be able to aggregate the estimations of individuals correctly and give insights into probabilities in real-life events. The product should adhere to the principles described in Section 2.3, in order to produce accurate prediction results. The next two subsections will define sub-goals which are related to Sections 2.3.2 and 2.3.3, which describe the requirements to correctly leverage the wisdom of the crowds.

Gather a Great Number of Perspectives

The product should be able to aggregate many different opinions and combine their wisdom in market values. The market system itself must be able to do this, and the prediction market as a whole must be available to a wide audience of potential traders. The market should be able to a big load of people simultaneously interacting with the prediction market, this way the needed quantity of opinions can be realized.

Aggregate a High Quality Collection of Perspectives

The prediction market should reduce groupthink among its users, as the goal is to enable users to think as an individual. As described in Section 2.3.3, groupthink reduces the accuracy of the product and should, therefore, be minimized. On top of that, the market should allow for different kinds of users to make trades. Increasing the diversity of individuals within the crowd is an important factor to increase the accuracy of predictions, this is an important goal. Lastly, to further increase the prediction accuracy, the system itself should not induce biases, nor have centralised control over the community.

2.7.2 Solve Problems Associated with Prediction Markets

Another important goal of this project is to overcome challenges which were encountered by other prediction markets. These problems, which were described in Section 2.4, frustrated the prediction process on existing markets. Overcoming these challenges should enable the product to produce more accurate prediction results than existing prediction markets. The following subsections define sub-goals related to the aforementioned challenges.

Reach Critical Mass

The challenge of reaching critical mass has been mentioned a few times now. The goal is quite apparent: the prediction market should reach a number of users great enough to overcome critical mass. However, since this project has a limited time span, reaching a high number of users while developing the product is not realistic. Therefore the prediction market should be engaging to interact with and engage users to participate in trading in order to ensure future growth.

Solve Market Liquidity Problems

Low liquidity on prediction markets can decrease the accuracy of the predictions [31]. Therefore the product should have an efficient system to provide liquidity to the prediction market. This system should enable users to make trades at all times, without paying out or charging unrealistic prices.

Correctly Initialize Markets

When creating a market, the values of answers in a market should be realistic. Therefore, it is a goal to develop a method to assign realistic starting values to answers. This method should be suitable for a prediction market, thus also in accordance with the theory of Section 2.3.

Prevent Market Manipulation

While market manipulation has not been a huge problem on existing prediction markets, the product should discourage users to try it. Apart from that, the product should reduce the chance of market manipulation as much as possible.

2.7.3 High Maintainability

The last set of goals concerns the maintainability of the product. After the development process, the product should be maintainable. The product will not be maintained by the developers and therefore the developed product should also be maintainable for others. This maintainability is heavily influenced by the code quality, testability, ease of deployment, and modular independence. These sub-goals will be discussed below.

High Code Quality

Code quality is essential for the maintainability of a software product. A project with high-quality code is easier to understand and therefore also easier to improve. Therefore the project code must comply with the code quality standards of the Bachelor Computer Science and Engineering at the University of Technology Delft.

Low Complexity and High Testability

Code modules which have a low complexity are much easier to test and to understand. When modules are easy to test, it is easy to verify if code behaves as expected. Therefore, code which is easy to test has a lower chance of containing bugs. Complexity and testability are highly connected to modular independence since high coupling between modules increases complexity.

High Modularity

Highly independent modules are less complex and much easier to test since the expected behaviour is clear and there are no large initialisation processes needed for modules to function. Modular independence does not only improve the complexity and testability but also influences the traceability of errors. On top of that, in systems with high modular independence, it is easy to improve the product, since changes to a specific module are less likely to influence or break other modules.

Easy to Deploy

After the development phase, it should be easy to deploy the product. Whenever the product needs to be set up on a server, the deployment should be smooth and should not introduce any problems for the maintainers. Therefore, the product should not rely on exceptional infrastructure, and therefore be easy to deploy.

2.8 Requirements Analysis

At the beginning of the project, clear objectives and requirements were set. These requirements were set with the goals of the project in mind. These requirements reflect on how to reach the goals of the project during the development phase. First, in Section 2.8.1 the requirements are given together with the goal(s) the requirements are related to. In the section after that, Section 2.8.2, it is expanded upon the requirements.

2.8.1 Requirements

The goals in Section 2.7 allow for a list of requirements for the final product to be formulated. In Table 2.1 the primary requirements of the project are given, with the goal they are related to. These requirements are the basis for the main design choices made throughout the duration of the project. After that, in Table 2.2 the secondary requirements are given, which are of lower priority than the primary requirements for the final product, yet help in the development process and were aimed to be implemented.

Primary Requirements

Requirement:	**Goal:**
The product must have an open market system where users can trade on	Leverage the Wisdom of the Crowds
The product must have an automated market maker to facilitate trades	Solve Market Liquidity Problems
The product must have a system to initialise markets with realistic starting values	Solve Market Liquidity Problems - Correctly Initialize Markets
The product must have an engaging trading interface and gamification elements	Reach Critical Mass
Users must be able to register and participate freely online	Reach Critical Mass
Users must receive a small amount of currency on registration	Leverage the wisdom of the crowds - Prevent Market Manipulation
Users must be able to view personal trade history and owned assets	Reach Critical Mass
Administrators must be able to create and resolve questions markets	Leverage the Wisdom of the Crowds
Administrators must be able assign markets to topics	Reach Critical Mass
The code must have high maintainability	High Maintainability

Table 2.1: Primary requirements and related goals.

In Table 2.8.1, the goal to 'Reach Critical Mass' is highly correlated to the 'Gather a Great Number of Perspectives' and 'Aggregate a High Quality Collection of Perspectives' sub-goals of Section 2.7.1. Therefore only the former goal is mentioned in the table. All the sub-goals of section 2.7.3 are also not stated explicitly as these are all part of the 'High Maintainability' goal.

Secondary Requirements

Requirement:	**Goal:**
Users should be able see the most active question markets over a time period	Reach Critical Mass
Users should be able to share markets on social media platforms	Reach Critical Mass
Users should be able to embed the current stock price of a question market using an iFrame	Reach Critical Mass
Users should receive rewards based on activity	Reach Critical Mass
Users should be able to reset their password by email	
Users should be confirmed by email	Prevent Market Manipulation
Administrators should be able to create add, edit and delete topics	Reach Critical Mass
Users should be able to earn badges for certain achievements	Reach Critical Mass
Administrators should be able to open a discussion forum under question markets	Reach Critical Mass
Administrators should be able add links and documents to enrich question markets	Reach Critical Mass
Active users should be able become moderators and propose questions to Administrators	Reach Critical Mass
Administrators should be able to assign moderators and accept question market proposals	Reach Critical Mass
The users could always have a view of the current share price updated in real-time	
The Registered Participants could place orders for buying and selling a share at a certain price	
The users could be able to create an account and login using third party single sign-on	Reach Critical Mass
Administrators and moderators could be able to make a certain question market only open to a subset of users	
Administrators and moderators could open a survey or vote on question markets that are not open yet so that the initial value of question markets can be gathered	Correctly Initialize Markets

Table 2.2: Secondary requirements and related goals, ordered by priority.

In the secondary requirement table, Table 2.2, the sub-goals of Section 2.7.1 are again not explicitly mentioned but are part of the goal to reach 'Reach Critical Mass'. In the list of secondary requirements, some requirements are not directly connected to the goal of 'Reaching Critical Mass' but do improve the overall platform. Improving the overall platform improves the user experience and the likelihood of interacting with the market. Therefore requirements which improve the platform are related to the goal of 'Reaching critical mass'.

2.8.2 Success Criteria

Success criteria are an important aspect of software projects. They define when the project is completed and successful. The formulated goals and requirements are a good foundation to build the success criteria upon. The main design goals of this project defined in section 2.7 give a good idea what the project should look like. In order for the project to be a success, all of the primary requirements defined in Table 2.1 must be met. These requirements are essential to enable the Cybersecurity Group at the TU Delft to build upon and maintain the product. The secondary requirements in Table 2.2 are less important for the successfulness of the project. However, these secondary requirements contribute to a more complete and therefore more successful product.

3

System Design and Specification

In this chapter, an overview of the system design will be provided. The goal of this overview is to enable further development of the application by providing insight into the system design philosophy as well as the infrastructure.

3.1 Back End

The back end for the project is written in Golang[1] and provides an Application programming interface (API) server using the Gin Web Framework[2].

The back-end is responsible for connecting to the database, serving data through an API and calculation and manipulation in-between. Since a lot of financial calculations are required in the form of a market maker, language features around mathematics, testing utilities and good type safety will be relevant in language selection. Two languages that can provide this are Java and Golang, both being considered as options since the client is already familiar with both of these languages specifically, and they can both be used for these types of operations, there are no large trade-offs between these two languages. We decided on Golang as it is a relatively newer language that provides nice options on database migrations we were already familiar with. Another convenient feature is Go marshalling, which makes it possible to represent Go objects between database representation, JSON strings and Go structs. This makes operating our API with Go more convenient.

3.1.1 Application Programming Interface

The communication between the front end and back end is implemented using an Application programming interface (API) which follows the Representational state transfer (Rest) software architecture style. This style requires five architectural constraints which are all met by our application.

Client-server architecture There is a clear separation of concerns between the client and the server.

stateless The API is stateless. No information is stored about the current session. All required information for a response has to be provided with the request.

cacheability Several API endpoints could be cached. In theory, all endpoints are cacheable. Only the transactional endpoints should never be cached, as they are required for correct price calculation which can update at any time.

layered system The system could be parallelised, as long as database concurrency issues were to be solved regarding transactional endpoints were to be solved. This can be done without the client observing any difference except performance.

uniform interface The uniformity of the interface is mostly handled by following best-practices in naming and following the HTTP standard. By sending informative status codes and using the correct HTTP operations, the uniformity of the interface is achieved.

[1] Golang: https://golang.org
[2] Gin Web Framework: https://gin-gonic.com

The Rest API is documented using the OpenAPI 3.0 specification, which can be found within a convenient user interface on Swaggerhub [3] and in the repository's */docs* folder. The endpoint structure is based around permissions, specified by the first part of the URL path. For example, getting markets is a public action, and thus available as a *GET* action on the */public/markets* endpoint, whilst the administrator action for deletion of market answers is the *DELETE* action on the */admin/markets/answer* endpoint.

3.1.2 Database Migrations

The back end contains command line interface options to execute database migrations. Database migrations are version control on the relational database schema. A schema migration is performed on a database when it is necessary to update or revert the schema of that database to a newer or older version. By executing *make migrateup* or *make migratedown* the migration files will be executed on the database in-order (either descending or ascending, in order of number). The largest advantage of this is that updated database schemas do not have to be shared between developers, and updating the live database can be done by simply running an 'up' migration after each deployment. This saves time, automatizes the database sharing process and thus makes it more reliable.

3.1.3 Authentication

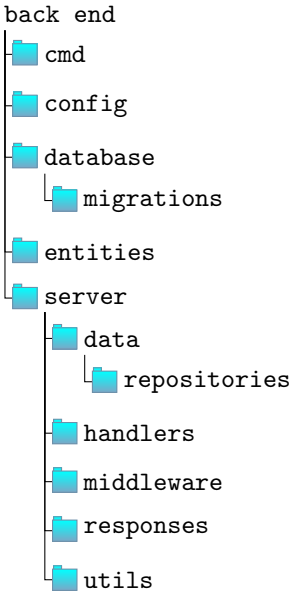
Authentication is handled using JSON Web Tokens (JWTs) [4], which is a reliable open industry standard that is secure and relatively easy to implement. Furthermore, the option to expire JWTs was implemented to accommodate for password resets, as to require the user to log in with the new password before gaining access to their account. Accounts are also verified for email confirmation before releasing a JWT. An alternative to JWT that was considered is OAuth [5]. However, the implementation of OAuth did not seem realistic due to time constraints.

3.1.4 Transactional Emailing

Transactional emailing is handled through a connection with Sendgrid [6]. Sendgrid exposes it's interface through an SDK for Go which is included in the project dependencies. Email templates can be called with variables such as emails for forgotten passwords or the initial email confirmation, which contain a unique token to verify the user.

[3] Swaggerhub API documentation: https://app.swaggerhub.com/apis/JeroenVanSteijn/PredictionMarket/0.1.0
[4] JSON Web Tokens: https://jwt.io
[5] OAuth: https://oauth.net/2
[6] Sendgrid: https://sendgrid.com

3.1.5 File Structure



cmd contains everything which handles the command line interface of the application. Note that for interaction with these commands, a Makefile is included in the repository.

config handles the .env file and provides an object for getting its configuration.

database/migrations contains the .sql migration files

entities contains entity definitions

server contains all code relating to the API server logic

server/data/repositories contains objects to interact with from outside (i.e. the database).

server/handlers the first entry-point for an endpoint is the handler. It validates, sends the relevant data to a utility file and then returns a response.

server/middleware contains request logging and authentication validation middleware.

server/resposes abstraction of the possible API responses.

server/utils utility functions that can be used as stand-alone functions

3.2 Front End

The front end is a standalone application created using the React framework [7]. For type safety the TypeScript [8] super set of JavaScript [9] is used.

3.2.1 Front End Framework

At the start of the project, several front end frameworks were considered. The first and most important requirement that lead our search was that the front end should be a standalone application, such that it connects to an API and does not concern itself with back end operations. Furthermore, it should be very maintainable for the client in the future, and it should be well maintained. Lastly, the framework should have a large community behind it so that the implementation of standardized dependencies should be relatively easy and largely available.

After carefully considering and comparing the 3 most popular frameworks: React[7], Angular [10] and Vue [11], as well as Vaadin [12], a Java framework that was proposed by the client, we have opted for React. This choice was made because it seems to have the best options in terms of separation of concerns and testability. Furthermore, it has been the most popular and fastest growing framework. It additionally is supported by Facebook[13], one of the largest tech companies in the world, and thus it is expected to be well maintained. Another welcome advantage is the fact that the application could be easily ported to React Native [14] to create a native app later on.

3.2.2 Atomic Design

React is a library for building composable user interfaces. It encourages the creation of reusable UI components which present data that changes over time [1]. To complement this component-centred approach, a good file, component and import structure is a must. Our structure is guided by the atomic design principles as described in the book "Atomic Design" by Brad Frost [11].

Figure 3.1: the Atomic Design component structure

3.2.3 State

To make sure that the front-end is a standalone application which abstracts away the API, state management is a must. It allows us to test interaction with other environments, such as the server connection, but also test components in isolation of the state, and keeps open the option to port the full state management and connection to the API implementation to React Native later on. Redux [15] was chosen for state management as it is the default state management tool in the React community. Alternatives such as using the React context API would be too lightweight for our heavy interaction with the API and would clutter components, making

[7] React framework: https://reactjs.org
[8] TypeScript: https://www.typescriptlang.org
[9] JavaScript: https://developer.mozilla.org/nl/docs/Web/JavaScript
[10] Angular: https://angular.io
[11] Vue: https://vuejs.org
[12] Vaadin: https://vaadin.com
[13] Facebook: https://facebook.com
[14] React Native: https://reactnative.dev
[15] Redux: https://redux.js.org

testing a lot harder. Redux' biggest competitor, Mobx only has around 60k usage on GitHub [16], compared to the 1.1 million for Redux (as of May 2020) [17], making it a safer option as it is more likely to be maintained well for a longer time.

3.2.4 Browser Support

Backwards compatibility and browser support are created using Babel [18], a JavaScript compiler that compiles the newer ES6 JavaScript standard to ES5 and below such that older browsers will also be supported. CSS transpiling is handled by Babel as well. However, as the modern CSS-grid functionality is utilised, and there is no polyfill available to transpile this to older browsers, the application will not work in older browsers such as Internet Explorer [19]. This trade-off was made knowing that Internet Explorer and other old browsers have a relatively small market share, and that the development speed, code quality and flexibility that CSS-grid brings are very valuable to the platform.

3.2.5 Styling

Elements are styled using CSS-in-JS. This functionality is provided through the use of Styled Components. Styled components allow us to extend components such as the 'Card' styling that is used throughout the project. Styling specific to a certain component, container or section is provided in a file next to the element called element.styles.tsx. This further separates styling from other front-end logic.

3.2.6 Dependencies

The front end uses a few dependencies for specific purposes that are not related to the overall system design. The dependencies are managed through node package manager and can be installed using Yarn[20].

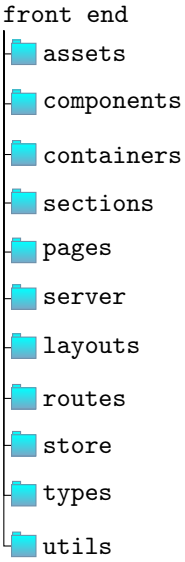
Apex Charts provides the graphing tools for viewing historical prices of the market in a React component.

Font Awesome provides a tweak-able icon set in *.svg* format as React components

[16] Mobx - GitHub: https://github.com/mobxjs/mobx
[17] Redux - GitHub: https://github.com/reduxjs/redux
[18] Babel: https://babeljs.io/
[19] Internet Explorer: https://www.microsoft.com/nl-nl/download/internet-explorer.aspx
[20] Yarn: https://yarnpkg.com

3.2.7 File Structure

The main application is written in the *src* subfolder. Besides this folder, the root of the repository contains configuration files for the development environment (Testing, Linting, CI/CD, TypeScript, Git), as well as the required *public* folder for the publication of the application.



assets provides images, fonts and other static files

components The smallest unit, has a single purpose and is re-used. Not connected to state.

containers Connects to state and generates a set of components.

sections Connects to state and generates a set of containers and/or components.

pages Does not connect to state and generates a set of sections

layouts Does not connect to state and provides a generalized page layout.

routes Defines the Router as well as a Routes enumeration. New pages should be added here

store Contains all files related to the Redux store/state. These are divided into multiple sets of sub-states that go logically together

types Contains type definitions that are used throughout the project.

utils Contains stand-alone utility functions

3.3 Database

The database is a PostgreSQL [21] relational database system.

Early on in the project a relational database system was chosen, as this makes it possible to enforce ACID principles. Furthermore, the data structure will not variate too much, which would be one of the main benefits of a non-relational database. As this main benefit is not utilized, but the drawback of not being able to enforce ACID principles is there, non-relation databases are not the most viable option for this project.

After the choice of relational versus non-relational database was made, several database management systems were considered. In the end, PostgreSQL was decided on as it is used and maintained a lot, has good integration with Golang, and above all has great performance. Alternatives such as MySQL [22] could be just as viable but would not make much difference in performance or options for this project.

[21] PostgreSQL: https://www.postgresql.org
[22] MySQL: https://www.mysql.com/

The database schema follows the principles of third normal form database design. This makes the database very well maintainable. An overview of the database schema can be found in figure 3.2

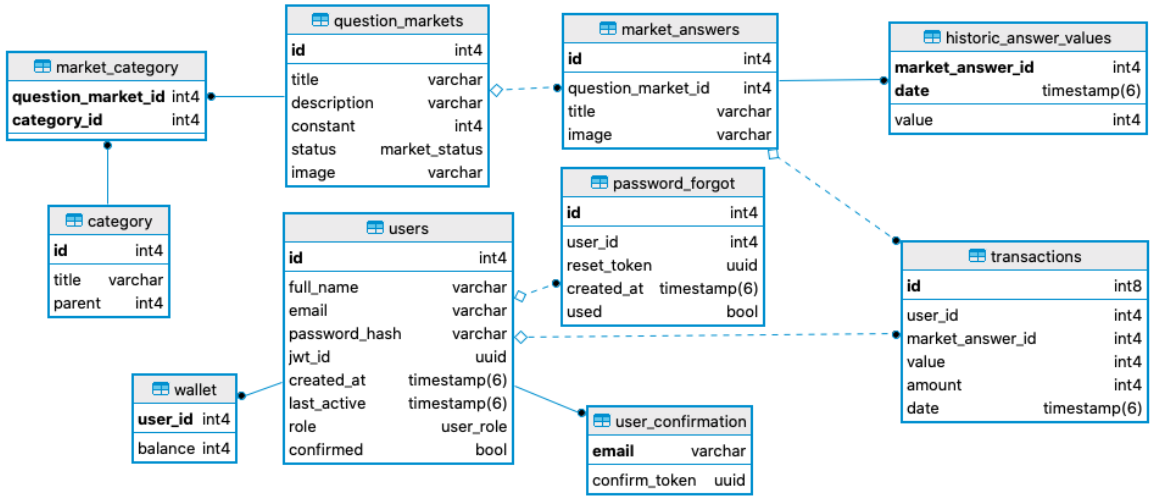


Figure 3.2: The database schema

3.4 Infrastructure

The current infrastructure of the application assumes an Ubuntu server setup. The front end of the application is served using Nginx [23], which routes to the /build folder which is populated using the build command. The '/api' path is configured to route to the Go server, which is running as a system service using a systemd file. Authorization between the front end and back end happens using JSON Web Tokens, which are sent using HTTP headers. A PostgreSQL database is running on the same server over port 5432, which is not open in the firewall.

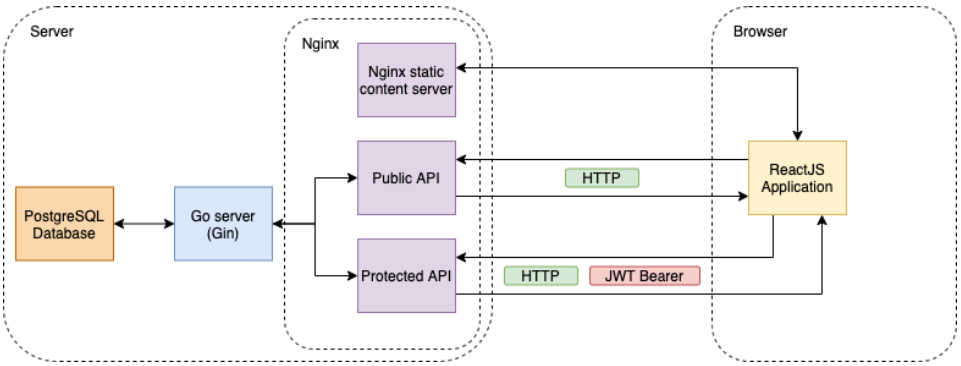


Figure 3.3: The application infrastructure

[23] Nginx: https://www.nginx.com/

3.4.1 Continuous Integration and Continuous Delivery

Continuous delivery saves development time and allows for an always up-to-date working version to be demonstrated, which fits well with the SCRUM methodology discussed in Chapter 6. Furthermore, automation of deployments removes the possibility of human error in the deployment process. When combined with continuous integration, which checks code quality by performing a test build and running the system tests, this not only saves time but also improves the reliability of the system overall.

Continuous Integration (CI) was set up using GitLab CI [24]. It first runs a build of the respective software element (front-end or back-end). If this is successful, it checks if the code style is correct by running a linter. Then it runs the testing suite. If all tests are passed, the CI will be successful and show the test coverage on GitLab.

Continuous Delivery (CD) was set up using GitLab CD. After the CI has finished successfully, it triggers the CD to connect to the server through ssh and run a deployment script. This connection is made with GitLab environment variables. Note that when changing servers a new SSH key pair should be generated and added to both the server and the GitLab environment variables for the CD to keep working.

Continuous Delivery could be improved by deploying using software such as Ansible [25]. This would allow for the full environment configuration to be declaratively defined, and not rely on the server environment being correctly set up as the current setup does.

[24] GitLab - About: https://about.gitlab.com
[25] Ansible: https://www.ansible.com

4

Implementation

After the design of the system and the infrastructure had been determined, the implementation of the system followed.

In Section 4.1 a detailed overview of implemented primary features can be found. The client and the development team had additionally come to the agreement that if time permitted it, secondary features would be added to the product, with many of these features being prioritised in the weekly meetings with the client. In addition, some features were added to the product which were not part of the initial project description, yet were requested by the client as additional prioritised secondary features. Information on these and aforementioned implemented secondary features can be found in Section 4.2.

4.1 Primary Features

In Sections 4.1.1 to 4.1.7 the primary features which were implemented into the final prototype back end and front end are described.

4.1.1 Market Maker

As described in Section 2.5.2 of the research, a market maker using a LMSR had to be implemented. The implementation of the market maker is one that follows Hanson's [14, 15], insights into market makers and what he calls "sequential shared scoring rules." Market makers were implemented in both the front end and the back end, both using Equations 2.1 and 2.2.

Front End

In the front end, a market maker is implemented which uses information on the market in question, and the amounts of shares in the options of that market to provide the user with prices which are indicative of the true values of the shares on the market at the moment in time of which the quantitative information was requested. It should be noted that due to the fact that the LSMR market maker is translation invariant, these values normally sum to 1, which allows for easily implementable intuitive visualisation of indications as percentiles or percentile-like values. As such, the value is brought into the $[0, 100]$ range through multiplication. This makes the fact that the values of the shares can be compared to respective indicative probabilities more parsable.

Values returned by the front end market maker are rounded for user convenience, as unrounded values may complicate the user interface, or could lead to worse aesthetics. This in addition to the fact that rounded values have practical advantages: through use of rounding, user experience is improved by reducing the chance of the back end market maker rejecting a proposed trade as a result of a change in value since the last time quantitative data has been received. This problem arises especially in more active markets with high market constants. These markets tend to have relatively small changes on small timescales, which can be caught by the margin introduced by rounding. Markets with less activity overall have a smaller chance of these value changes occurring, reducing net user experience gained through the use of rounding. Share values are rounded to two decimal points, leading to a resolution of 10,000 distinct possible values, where buying and selling prices are rounded to 1 decimal point (up and down respectively), leading to a resolution of 1,000 distinct

possible values. The share values are rounded less in order to reduce the chance of sudden large changes, and a more accurate representation of the value of the middle of the buying/selling price spread, allowing users to more easily view a value closer to the true value of the option as determined by the market maker.

Back End

In the back end, a market maker is implemented which uses database information directly on each incoming transaction request. This allows for the accurate acquisition of actual share worth as long as no concurrent request happen within the time the transaction is processed, else the value may differ, with an impact proportional to the amount traded and inversely proportional the constant used in the transaction which has started processing first. In most active markets with high market maker constant values this effect is rather small, yet can still be exploited at this point in time when using API calls. This is currently regarded as breaching the terms of service as agreed upon by the client.

The primary job of the market maker in the back end is to validate incoming requests by comparing the buy or sell value provided through the request with the values calculated from the database data. If the request is to sell for less than the market maker would accept a selling request for, the selling request is approved. Conversely, if the request is to buy shares, this can be done for all values above the minimum the market maker would accept. It should be noted that for the protection of the users (primarily ones not using the provided front end) buying values higher than 100 are not considered valid, as they always result in a loss, thus proving that a user has either has made a mistake or is attempting to do a transaction with goals other than (virtual) wealth acquisition in mind. For selling transactions, similar restrictions apply to attempted trades with values less than zero, as the maximum loss one should be able to take is the price of share acquisition. Illegitimising selling trades with values less than zero primarily protects users, as selling for zero is always possible in the back end.

4.1.2 Trading Screen

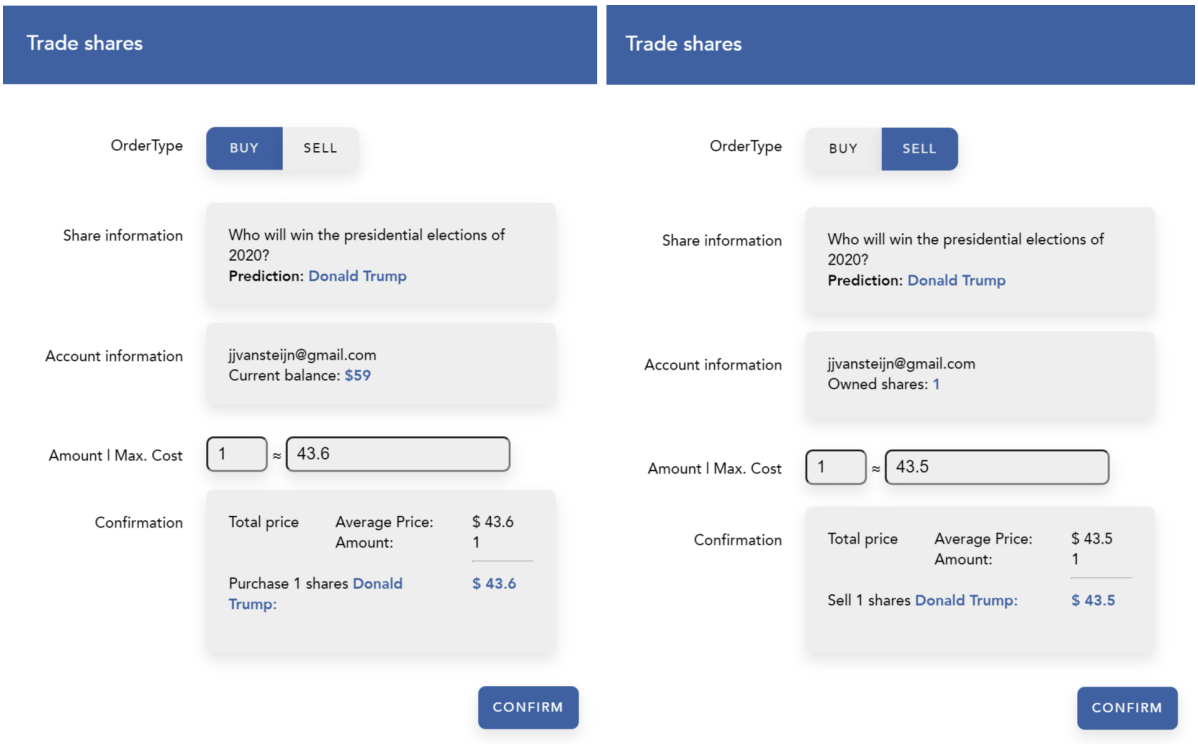


Figure 4.1: The buy and sell trading modals

The trading screen handles one of the main interactions a user will frequently have with the platform: trading. It is thus important that it is intuitive, insightful, and flexible. To enhance the user experience the trading screen provides an overview of the information relevant to the user at the time of transaction. When selling

shares it provides an overview of the current portfolio of this share. When purchasing new shares it gives insight into your current balance. To create flexibility it can handle both an amount-of-shares input and a maximum-cost input. As the market maker is generally a one-way operation, the maximum cost is calculated using a divide and conquer algorithm which was created to search for the optimal result recursively. This is done within the limits of the browser's stack, as there is a maximum amount of options given the maximum amount of shares per transaction (1000).

From a usability standpoint, it was important that key listeners were added that allow the user to exit the modal by pressing the Escape key, as well as confirming their purchase using the Enter key. The modal also allows the user to exit by clicking outside of it on the backdrop.

4.1.3 Visualization

The visualization of market pricing has a large impact on multiple fronts. One aspect of this is gamification. Seeing the market move can give an incentive for users to trade. Without this data visualization, the market could feel inactive and the incentive for users to trade on the market would be absent. Another effect of the visualization is that the ultimate goal of the project, which is to gather insight into the chances of different possible outcomes, is made available as a timeline. The graph is limited on the y-axis which rounds to 10 points above and below the respective highest and lowest points of the data, with an absolute limit of [0, 100], which is reasonable given the translation invariance of the scoring rule used. The data is available on a per-minute basis, where the latest transaction in the minute determines the value of that minute.

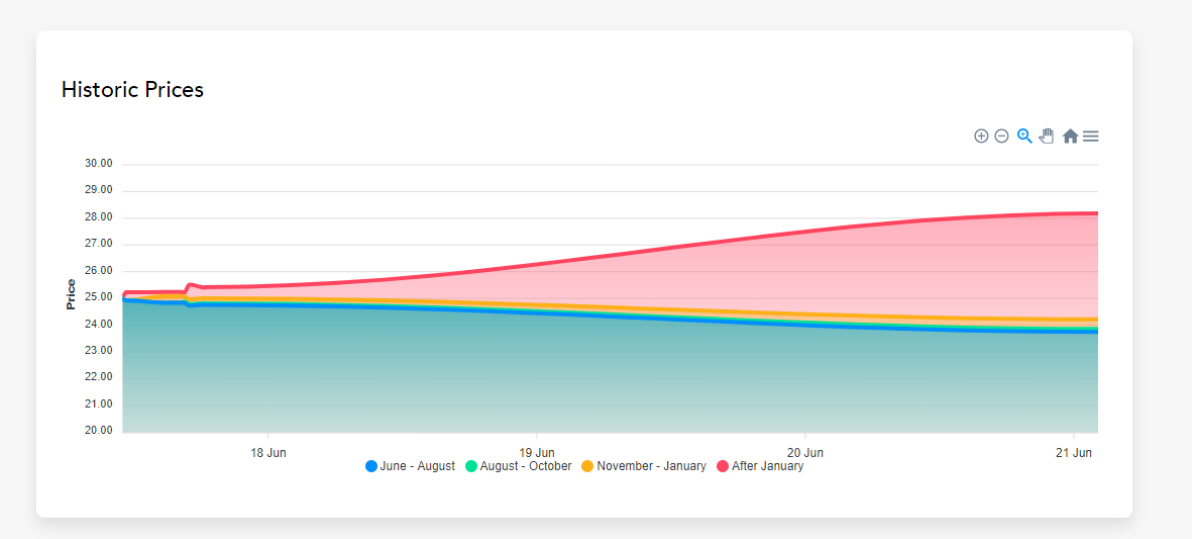


Figure 4.2: The historic pricing graph

4.1.4 Dashboard

The dashboard is the central place for a logged-in user. It gives feedback on the performance by showing the balance and value of the assets owned by the user. It displays all the information on their recent activity. It displays the assets in the portfolio with the associated market and the number of profitable portfolio items in the form of a donut chart. On top of that, the user is able to monitor recent transactions on this dashboard. To motivate the user to make more successful trades, the place on the leaderboard is displayed on the dashboard.

Portfolio

On the dashboard is also an overview of the user portfolio. Here, the user can see all of their shares, together with their average value, average investment, and profit/loss ratio. This overview enables the user to monitor its performance and give the user an idea of how well it has been doing over time.

Leaderboard

As a user your rank is available on the dashboard directly. The top 100 users are available on the user dashboard by clicking through to 'see leaderboard'. These users are ranked in order of highest balance. A leaderboard

can motivate users to perform more and better trades. Climbing in the leaderboard can feel like a real accomplishment. Thus, this feature adds to the gamification of the platform.

Transactions

At the bottom of the page is the users' transaction history. This overview includes both regular transactions and transactions as part of the resolvement of a market. Received rewards will show up in this history as well. This way, users cannot only see their historic transactions, but also get feedback when their estimation was correct.

4.1.5 Rewards System & Kick-Starting Markets

Administrators have the ability to set the status of a market to 'initialization'. This signifies to the platform that the market is finalised and is going to be made public soon. In the time between being set to 'initialized' and being made public, the platform can gather useful information for the starting values of the market. To this end, users are presented questions on their dashboard, as part of a daily reward system. The reward for answering questions is that they receive shares in the answer they selected as being the answer. This way, the starting values of the market can already start to converge before the market actually opens for trading, which in turn allows for the selection of greater market constant values, as the market can already start to get closer to equilibrium at an earlier stage.

In order to assure that the market starts to converge to logical values for its answers, the expected value for convergence can be derived:
If users choose share/answer i with probability p_i, we want the market to converge to that value. If users answer the question presented to them, they cumulatively build up the shares owned for that answer to some q_i. Given Equation 2.1 for the value of a share:

$$\begin{aligned} E[v(q_i)]_{n\to\infty} &= E[\frac{e^{\frac{q_i}{c}}}{\sum_{j=1}^{n}(e^{\frac{q_j}{c}})}]_{n\to\infty} \\ &= lim_{n\to\infty}(\frac{e^{\frac{np_i}{c}}}{\sum_{j=1}^{n}(e^{\frac{np_j}{c}})}) \\ &= p_i \end{aligned}$$

As such, values will converge to the same values as present in the distribution of the answers to the questions posed to the users on the dashboard. This distribution is determined by how likely the collective of users receiving the rewards think an outcome is to happen.

As an additional benefit, this implementation of a reward system has the added benefit of spreading interest and engagement for new markets to the whole user base, by incentivising evaluating the status of ones reward shares, causing users to engage in markets they would normally not engage with.

4.1.6 Administrator Capabilities

Administrators have the ability to manage categories as well as markets. Both can be edited and added. As markets require images, the administrator can select an image from their files and upload it. After uploading, they are provided with a preview. Furthermore, administrators can choose from a predefined range of market volatilities, which have been put in place to help administrators who are not experienced with determining the right value for the LMSR. Finally, when a market outcome has become clear, the market can be resolved to the correct outcome, and users will be paid automatically for their shares. In the future, the possibility of user management for administrators could be a quick addition, as endpoints for requesting all users and deleting a specific user are already available for admin accounts.

4.1.7 iFrame Embedding and Sharing

On a single market page, 4 share options can be found. The first three: Facebook, Twitter and LinkedIn are all direct links to social media. The fourth option is embedding. By selecting this option, an <iframe /> code will be added to the user's clipboard. The user can then share this iFrame code on any website. This can create engagement on sites where market questions are very relevant. Whilst the current implementation does lack a great visual design, it is fully functional and can easily be built upon as a way of reaching critical mass as described in section 2.4.1

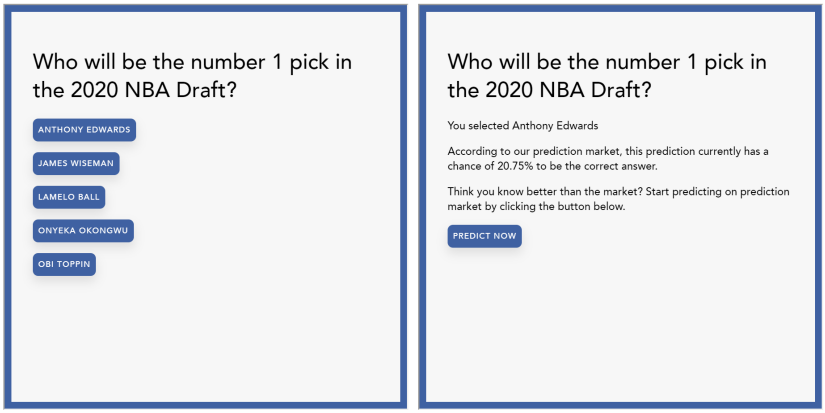


Figure 4.3: The iframe embedding functionality

4.2 Secondary Features

In Sections 4.2.1 to 4.2.5 the secondary features which were implemented into the final prototype back end and front end are described.

4.2.1 Email Confirmation

As soon as user testing was started, it became clear that having a small barrier of entry would be necessary. By allowing users to sign up without confirming any identity at all, the users quickly generated more than one account. Whilst this is not a problem in small amounts, if no limit were to be added, some users could potentially ruin the platform in seconds by spamming the site and manipulating values. Thus, email confirmation was put into place which needs to be followed through before the user can finalise their account. This is a small one-time confirmation that removes most of the issue of spam accounts. If this problem would become apparent in the future, despite the confirmation email, the requirement of a phone number which could be texted a one-time code would be easily implementable and even more robust, as acquiring access to many different phone numbers is not very scaleable for malicious users. However, the drawback here is that this barrier to entry could be too high for users who do not have their phone around when starting the application or simply do not want to provide their phone number, in addition to the fact that texting fees would apply per user sign-up.

4.2.2 Password Forgot

Users tend to forget their password. Even though a password forgot option was not put in the requirements, during the building of the project it seemed like a must for any mature platform, as providing user support as soon as a user forgets their password is not very scaleable. This also becomes apparent as, during one week of user testing with a small group of around twenty people, this feature was utilized at least 5 times.

4.2.3 Statistics

In order to present the user with quick access to the most relevant markets at any point in time they access the platform, a statistical ranking endpoint is available. It allows for the client to request any time frame for which to calculate the top ranking markets in the categories: 'fastest-growing market',' market with the most trades', 'market with the largest amount of unique users' and 'the market with the biggest traded monetary volume'. Currently, the web client requests this information for the interval between now and one week ago, but by using denominations such as *1_DAY*, *2_WEEKS*, *3_MONTHS* or *1_YEAR* for the same endpoint, this information can also fetched for other time periods.

4.2.4 Privacy

To account for GDPR compliance, as well as provide users control over their data, the platform accommodates for a 'delete account' action that is available for any user. This action is first confirmed, as it can not be undone. After this, the user data is no strictly deleted. Instead, all identifying data is obfuscated. This ensures that the active markets are not influenced by deletion of user accounts that are large shareholders.

4.2.5 Middleware Options

A small middleware module for Gin is provided that allows for easy verification of permissions for the routing. By simply adding another subroute and providing the required role, any role such as the proposed 'Moderator' role can be added by simply adding it to the 'Role' enumeration using a migration file, and calling the *validateRole* function from *middleware/auth*.

5

Software Quality

As this project is the foundation on which the platform will be further developed, software quality is very important. In this chapter, we will explain the software quality approach that was decided on at the beginning of the project, and assess the end product based on this approach. Furthermore, during the project, the Software Improvement Group (SIG) provided us with two opportunities to use their extensive static analysis tools to assess the quality of the codebase. These will be analysed and reflected upon in section 5.3

5.1 Static Analysis

Static analysis is the assessment of code without execution of the code. It ensures that all of the code adheres to the same code style as much as possible, but more importantly it catches a big portion of mistakes. The back-end codebase includes golangci-lint[1], which in turn includes multiple linters that find issues such as unused struct fields and unchecked errors. The front-end codebase includes the TypeScript compiler in strict mode and Eslint, which check for unused variables and correct typing among other things.

5.2 Testing Approach

Automated testing is a big part of software quality. Having a reliable test suite provides greater certainty that the current product will continue to work when adding new functionality, increasing the overall speed of further development. As this project contains two separate code bases that have very different concerns, the testing approach will be described separately for the front end and back end.

5.2.1 Back End

Correctly testing the back end code base is an intensive process. It is important that the utility methods, which are small modules of code that are re-used in the code base, are extensively unit tested. This also includes the files that interact with the database. The aim is to get the statement coverage of these utilities to at least 80%. On top of that, since the back end provides an API that has a large number of important interactions, this part of the codebase must be correctly integration-tested. Since the handlers are responsible for processing API requests, these should have a statement coverage of approximately 100%. This way the API requests can be verified and are specified fully. Any impactful change to the specification will thus result in the tests failing. The test coverage is calculated in the CI described in section 3.4.1, with the standard Go cover tool [2]. Both coverage goals are met and the overall statement coverage of the back end at the moment of writing is 92.7%

The test files use the standard Go testing packages[3] to run the tests. To test the utility files which interact with the database and to integration test the handlers, the database had to be mocked. Go-sqlmock [4] was used to mock the PostgreSQL database.

[1] golangci-lint: https://golangci-lint.run/usage/linters
[2] Go cover tool: https://golang.org/cmd/cover
[3] Go testing package: https://golang.org/pkg/testing
[4] Go-sqlmock: github.com/DATA-DOG/go-sqlmock

To correctly test the handlers, a test router of the Gin Web Framework [5] was used in combination with the Go httptest [6] package. With this combination of tools, it is possible to send test requests to the back end and record the result of the code. This tests the complete API and confirms the back end code behaves as expected.

5.2.2 Front End

The testing for the front end code base includes unit and integration tests. Both types of tests are run using Jest for JavaScript testing. The overall coverage of the front end is 70% statement coverage. Specific coverage was aimed at least 90% statement coverage for the state management. Fully visual UI components did not have a required amount of coverage as line coverage on UI components is not very meaningful. Instead, *onChange* and other interaction functions are covered.
To test interaction with the user interface (UI), Enzyme is used. The UI tests are mainly integration tests, as unit level UI tests are not very meaningful. The most important part of a UI is interaction with the user, which is ultimately always integration-based.
The other big part of the front end is the state / store. It handles interaction with the API in .action files, as well as updates to the state in .reducer files. Both of these parts of state management are unit tested. The actions are tested by mocking the *apiRequest* function. State updates are tested by mocking the previous state and executing an action through the reducer.

5.3 SIG Analysis

In the sixth and ninth week of the project, the codebase was submitted to the Software Improvement Group (SIG)[7], an independent code evaluation company which performs static analysis on code and compares it to other projects submitted to them. The submitted code represents the project code base as it were around ~70% and ~100% progression towards the presented prototype respectively. Feedback from the first code submission was integrated into the code of the second submission in order to further improve code quality. It should be noted that the SIG presents quality metrics with ratings between 1 and 5 stars, where more stars represent better performance [2].

5.3.1 First Submission and Metric Definitions

The code of the system scored 3.9 stars for maintainability overall, with a score of 3.7 and 3.9 for the back end and the front end respectively. This means that the submitted code was of above-average maintainability [2]. The provided system fact sheet can be found in Table 5.1 below.

Metric	Front end (stars)	Back end (stars)	Overall (stars)
Volume	5.5	5.5	5.5
Duplication	4.2	4.5	4.4
Unit size	2.0	2.2	2.1
Unit complexity	2.6	2.4	2.5
Unit interfacing	2.1	3.9	3.2
Module coupling	5.4	3.9	4.5
Component balance	N/A	N/A	5.5
Component independence	N/A	N/A	N/A
Total maintainability	3.9	3.7	3.9

Table 5.1: Software Improvement Group System fact sheet (first submission)

As can be found in Table 5.1 above, unit size, unit complexity, and unit interfacing received the lowest ratings, with the rating for duplication being acceptable yet not quite satisfactory. As such, the maintainability of the system could be improved if these points were improved upon.

The SIG defines the notion of units as the smallest executable parts of source code, such as functions [2].

[5] Gin Web Framework: https://gin-gonic.com
[6] Go httptest package: https://golang.org/pkg/net/http/httptest
[7] Software Improvement Group: https://www.softwareimprovementgroup.com

The SIG defines unit size as the size of the source code units in terms of the number of their source code lines [2]. To assess the size of units, the following Table 5.2 is used [2]:

Unit Size	Associated Risk
1-15	Easy to understand, no risk
16-30	Slightly long unit, low risk
31-60	Long unit, moderate risk
>60	Very long unit, high risk

Table 5.2: Unit size metrics assessment table

In order to improve the maintainability from a unit size perspective, the unit size should ideally be reduced to 15 or lower wherever possible. There should be no unit sizes above 30 if not strictly necessary. In the first submission, 45 violations were found, the worst being a unit size of 104.

The SIG defines unit complexity as the degree of complexity in the units of the source code as defined by the MacCabe Index [2]. To assess the complexity of units, the following Table 5.3 is used [2]:

McCabe Index	Associated Risk
0-5	Easy to understand, no risk
6-10	Complex, moderate risk
11-25	Very complex, high risk
>25	Untestable code, very high risk

Table 5.3: Unit complexity metrics assessment table

In order to improve the maintainability from a unit complexity perspective, the McCabe index of the units should ideally be reduced to 5 or lower wherever possible. There should be no unit sizes above 5 if not strictly necessary. In the first submission, 17 violations were found, the worst having a McCabe complexity of 19.

The SIG defines unit interfacing as the size of the interfaces of the units of the source code in terms of the number of interface parameter declarations [2]. To assess the unit interfacing quality of units, the following Table 5.4 is used [2]:

No. of Parameters	Associated Risk
0-2	No risk
3-4	Low risk
5-6	Medium risk
>6	High risk

Table 5.4: Unit interfacing metrics assessment table

In order to improve the maintainability from a unit interfacing perspective, the number of parameters of the units should ideally be reduced to 2 or lower wherever possible. Where this is not realistic, a maximum of 4 parameters per unit should be regarded as the upper limit. In the first submission, 23 violations were found, the worst having 5 parameters.

The feedback of the first SIG submission demonstrated to the development team that the maintainability of the code should be improved significantly, as the maintenance and further development of the system could be hindered if this would not be done. This also leads to the realisation that making the code more human-readable could further improve the ability of the maintainers and future developers of the system. As such, line length and complexity of individual lines was limited further, and the code was formatted in a manner which would be easier to parse.

5.3.2 Second Submission and Achieved Improvements

Taking the lessons of the first submission into account, the development team aimed to improve the unit complexity first and foremost. This was done to ensure that the individual units could be easier to understand and thus maintain. The second priority for adjustments to the code base can be found in the unit size; measures were taken to ensure that units did not become too large to be reasonably easy to understand. The second to last priority for the second submission was to keep duplication low, as duplicate code could make maintaining or expanding the system harder. The last priority was to limit the number of parameters to four or less wherever possible.
The feedback of the second SIG submission can be found in Table 5.5 below.

Metric	Front end (stars)	Back end (stars)	Overall (stars)
Volume	5.5 (=)	5.5 (=)	5.5 (=)
Duplication	3.3 (-0.87)	3.7 (-0.78)	3.6 (-0.8)
Unit size	2.1 (+0.07)	4.1 (+1.94)	3.4 (+1.26)
Unit complexity	3.3 (+0.74)	5.3 (+2.90)	4.6 (+2.10)
Unit interfacing	1.7 (-0.42)	2.2 (-1.69)	2.0 (-1.13)
Module coupling	5.5 (=)	4.1 (+0.18)	4.6 (+0.05)
Component balance	N/A	N/A	5.4 (-0.05)
Component independence	N/A	N/A	N/A
Total maintainability	3.9 (=)	4.3 (+0.59)	4.3 (+0.34)

Table 5.5: Software Improvement Group System fact sheet (second submission)

As can be seen in Table 5.5, the task of first priority of the changes for the second submission, to reduce unit complexity, has been fulfilled. This was achieved by rewriting most functions in the back end and reworking the worst offenders in the front end. This reduction in unit complexity has significant value for the product, as this was one of the primary reasons our evaluation concluded that the product at the first submission was not maintainable enough. The maintainability of the front end remained roughly the same, which was expected as no large changes to its structure were made, given the time constraints as well as prioritizing the back end improvements. The largest issues in the first review were resolved, but a lot of additional features were integrated. For these new issues, the same code quality was achieved.

The task of second priority, to reduce the unit size, was primarily achieved in the back end. This was primarily done through the same means as the improvements in unit complexity. The smaller increase in the calculated metric can be explained by one of the takeaways prioritised from the first submission, which was not directly related to the metrics provided: the codebase should be formatted to be easier to read. After having reworked most functions and having rewritten them to be easier to read, some units had to be compromised on, as reducing the unit size would have meant that the code had become less readable, whilst further reducing complexity was often not reasonable. Another example of this is logging of errors to the server. This creates additional lines but is still worth more than having a smaller unit size. Reduced unit size could have been achieved by placing brackets or even what now are multiple lines of code on a single line, or by omitting logging, yet this would have resulted in a reduction of maintainability, not an increase.

The task of second to following, to reduce code duplication, has been achieved with a greater amount of success than can be deduced from the provided metrics, where a decrease in quality is visible. Whilst a lot of code duplication has been removed from the code base, a lot has been introduced to the code base by the same method as described for the task of unit size reduction: one of the actions taken to ensure that the code would become more readable was to move the PostgreSQL queries, which had formerly been represented in single-line, very long strings, to the declaration of constants at the top of their respective files. In order to make these query strings easier to understand, multi-line strings were utilised in combination with the reformatted and indented SQL query string representations. SIG seems to treat single-line and multi-line strings differently, and as such, many parts of the query strings were flagged as duplicates. Whilst this may technically be correct, one has to realise that query strings often utilise the same recurring patterns. Whilst it is true that this could be reduced through string concatenation, this would defeat the purpose of the reformatted query strings as easy to read alternatives for the previous single-line strings.

The task of the following priority, to limit the number of parameters per unit to 4 or less than possible, has mostly succeeded. At the second submission, there were 2 units remaining with a greater number of parameters. Apart from these, the limit of 4 parameters has been enforced. The primary objective of reducing the amount of parameters is to increase ease of testing. As reducing the metric to be 2 parameters or lower for the entire project would be rather unreasonable, partly due to the increased complexity of the needed parameter types/structs. The primary reason that the provided metric has decreased significantly is that a lot code has been added which depends on multiple types, in addition to the reworked functions which introduced more units with numbers which had more than two parameters.

6 Process

This chapter describes the process of the project. The development and planning methodology is discussed in Section 6.1. The internal and external communication is discussed in section 6.2. Finally, unexpected challenges during the process are described in Section 6.3

6.1 Development and Planning Methodology

The development and planning methodology was an implementation of SCRUM. Small adaptations were made at the beginning of the project to accommodate for the relatively small team. The requirements were prioritized using a list of must-haves, should-haves, could-haves and will-not-haves.

The product progressed using weekly sprints, which had a clearly defined goal to be reached at the end of the week. Progress was tracked on a day-by-day basis using the GitLab issue board feature. Issues were tracked in five categories: open, to-do, doing, in-review and closed.

Quality assurance took place in the form of GitLab merge request, which had to be tested and reviewed by at least one other developer before a certain feature could be merged. Any questions or issues would be mentioned through the GitLab comment system. Besides quality, this also helps other developers from staying up-to-date with changes to the code base, which in turn will provide a faster and more robust development cycle.

6.2 Communication

Good communication is key for any project of this size and timeline. To stay up-to-date internally, a daily stand up meeting was put in place. For communication with the client and coach, Mattermost was used for ad-hoc questions and feedback. Demonstration of the latest working product and planning took place during the weekly sprint review meetings.

6.2.1 Daily Stand-Up

To stay coordinated as a team, a daily stand-up meeting of about 15 to 45 minutes. Here, everyone described their progress of the day and their goals for the upcoming day. Any blocking issues or problems were addressed as well. Most problems were tackled using Visual Studio Code live-share[1] sessions to execute remote pair programming.

[1] Visual Studio Code live-share: https://visualstudio.microsoft.com/services/live-share

6.2.2 Weekly Sprint Review

The weekly sprint review meeting took place over video call and was attended by the team and client. At times, the coach also attended this meeting to stay up to date on the project.

The meeting followed roughly the same structure each week:

- Progress summary of the past week
- Demonstration of the current working version
- Planning and prioritization for the next week
- High-level discussion on gamification, the wisdom of the crowds concept, market-making and/or market initialization
- Feedback on progress and planned features from the client

6.3 Unexpected Challenges

The research document reflected too much research in the directions of gathering insights and user interface design, which diverted from why prediction markets fail and how to kick-start them. This was resolved by using this research in the final report and creating a new timeline to rewrite the research document. The team also worked extra hours during multiple weekends and evenings to catch up with the timeline.

Furthermore, the 2019-2020 Coronavirus outbreak posed a large unexpected challenge. At the start of the project, a lockdown prevented us from meeting physically. For this reason, no physical meetings have taken place during the project. Problems did arise in the form of communication issues. Misunderstandings are more frequent in (video) calls than in real life. Realizing this helped prevent the build-up of frustrations. As a result, more meetings were planned than usual. This helped with aligning the team and staying in touch with the project and each other. Online tools such as VS Code Live share further helped this remote set-up [2].

[2] Visual Studio Code live-share: https://visualstudio.microsoft.com/services/live-share

7 Discussion

In this chapter, we will provide our recommendations on the further use and development of the project, as well as discuss the ethical implications of the growth of the platform. Lastly, we will suggest further research that could be done using this product or that came up during its development.

7.1 Usage Recommendations

The platform includes all the necessary features to start gathering predictions. However, administrating the platform and creating a brand around the platform is essential to make it a success. We will discuss our recommendations for how this could be approached.

7.1.1 Creating Questions

As described in section 4.1.6, administrators will be responsible for creating new questions on the platform. This has great influence over the success of the platform in terms of predictive power as well as user experience. To this end, we have a few pointers on things that should be kept in mind when creating a new question market.
First of all, to help with the goals this project aims for as mentioned in Chapter 2, the question should be a good question that can be gathered from the wisdom of the crowds. This means that it is a multiple-choice question with a clearly defined answer that lies in the future.
Secondly, the question should not be ambiguous in any way, and should not induce any bias.
Lastly, it should be (almost) impossible for more than one answer to be correct, and the answers should be exhaustive as much as possible, as resolving a market with no, two, or more correct answers is not supported by the platform.

Selecting the Correct Market Maker Parameter

As described in Section 2.5, market makers with a LMSR have one parameter which needs to be provided before the start of the market. This parameter influences the market convergence speed, volatility, and liquidity. High values for this parameter increase liquidity and reduce volatility, yet decrease convergence speed. This parameter should be set to be as high as reasonable for a given expected amount of users on a market: if not many users trade on a market, it may never converge to a value on which equilibrium occurs if the parameter is set too high, but on markets which are able to reach this equilibrium, the parameter influences the width of the spread between the buying and selling prices of the market maker. Maximum liquidity and thus maximum accuracy in this situation is reached if this width is as small as possible. Determining a good value for this parameter is more of an art than science [27], and is thus inherently hard to pinpoint for inexperienced administrators.

In order to help inexperienced administrators with determining this parameter, a discrete set of options is provided in the front-end market creation screen. This approach does however have drawbacks such as limiting the range of possible values which could be useful for experienced administrators. In Section 7.4, this fact will be discussed further.

7.1.2 Branding and Marketing

The platform currently has not yet reached critical mass in terms of users, meaning that its predictions will not be very accurate (as described in section 2.3.2). To gather enough users to make meaningful predictions, branding and marketing for the platform would be a must. One idea would be to create a snowball effect, for example using referral marketing. However, a creative incentive for users to refer would still need to be explored.

7.2 Ethical Implications

It is important to be aware of possible ethical implications a created system has in order to prevent unintended negative consequences. Two large implications will be looked at: gamification and gambling addiction, and the influence of predictive data.

7.2.1 Gamification and Gambling Addiction

As the product has been created with gamification in mind, there is the risk of addiction to the platform. This risk is very minor as the platform currently has too little activity for one to be playing for larger amounts at a time or at short intervals. However, if the platform were to become larger, to the extent that it is big enough that one could spend several hours at a time trading, prevention and assistance of addiction could be valuable for vulnerable players.

7.2.2 Predictive Data and Influence

The prediction market acts as a predictor. When a prediction appears accurate enough that it is relied upon for decision making, it is important for the responsible party making this decision to be aware of the limitations of the prediction. Problems of groupthink arise, as well as cognitive biases such as expecting the highest probability to always occur. The prediction market's outcomes could also become self-fulfilling prophecies if their influence becomes large enough. It should also be noted that not every person might be able to accurately interpret any question posed, or any market situation. A misinterpretation on the user side can cause people to make real-life decisions which they might not want to take if they had understood the situation correctly. A primary aim should, therefore, be to actively communicate to resolve any misunderstandings, and to prevent misunderstandings by making questions and answers very clear, in addition to only using markets for appropriate problems.

7.3 Further Research Suggestions

This project provides the foundation of a prediction market platform. Once the hurdle of critical mass has been overcome, this opens up a variety of options for further research, such as research comparing the predictions of the crowds to those of experts in several different ways and for several different fields of expertise. This way, the limits of the wisdom of the crowds can be further explored.

Another interesting research question building upon this thesis is the question if the weighing of the perspectives of the crowds through mimicking financial resources is effective, given that the market uses 'play money'. The consequence of the market is that successful traders have greater influence over the market. Although 'real money' markets have this property as well, 'play money' markets have no other ways of gathering large amounts of resources, thus giving relatively more power to successful players. A possible way to assess the impact of this effect would be to implement a simple per-user share limit per market.

An additional research question would be on the effect of the variety in users and thus perspectives on the predictive capabilities of this system, and thus the wisdom of the crowds. This could be achieved by restricting market access for some parallel markets to groups of users grouped by some criterion such as overall success in related markets, if the platform has acquired enough data on similar markets and their payouts and above all has reached a user base large enough to sustain all these parallel markets, even when divided.

7.4 Further Development Suggestions

As the system at the moment of writing has been developed by a team of which no member has a background in design, a logical development step would be to improve the visual design, user interface, and thus user experience of the front end by working with a party which has experience in the field of (user interface) design. This could improve user retention and could be a distinguishing factor when combined by the relatively simple trading process. It could also prove to help the branding of the platform, helping it to reach and retain critical mass.

Another useful feature for use in further research could be the calculation of variance or errors in the predictions made by the market maker. These results could provide further insights into the behaviour and usability of systems like this one, yet was beyond the scope of this project.

A final recommended development step is to omit the discrete market constant selector values from the front-end development interface with a continuous input field. This may not be as intuitive for new administrators, yet could provide experienced administrators more control over the volatility and liquidity of markets they initialise. This feature had already been implemented, yet was omitted in favour of the more intuitive discrete set of choices, as these simplify the selection of this constant to a question of predicted user amount and wanted convergence time.

8

Conclusion

Three months ago at the time of writing, the Delft University of Technology Cybersecurity Group requested for a platform to be developed which could utilise the wisdom of the crowds. To this end, a client-server system had to be developed. Many systems utilising market systems which have been developed before had shown problems regarding the lack of liquidity on their markets, the lack of competing perspectives, the kick-starting of markets, and the incentivising of users (and thus user retention). A potential solution for these problems had to be found and implemented to achieve the goal of creating a prediction market that can successfully gather the wisdom of the crowds.

The platform which has been developed during this project uses a combination of logarithmic market scoring rules, gamification, and a reward system in order to address problems with market liquidity, user/perspective gathering and retention, and the kick-starting of markets respectively.

In nine weeks, research was done to base the system design on, the system was designed, implemented, and tested. This prototype has been continuously deployed and can be found at its website [1] at the moment of writing. This system has been developed to be functional, yet at the moment of writing still lacks the user base which is needed to reach critical mass. The system includes all of the primary requirements specified by the client (see Section 4.1), in addition to various secondary requirements (see Section 4.2) and additional requirements specified by the client during the project. This system can be used as a prototype for a more fully market-ready implementation which would include more detailed design, branding, and marketing. As such, this system has been implemented and documented in a way which enables the addition and maintenance of features to be quite easy (see Chapter 3).

In the last weeks of the project, users were invited to test the platform. First signs indicate that there has been somewhat consistent activity on the platform, despite the lack of great amounts of available market content and the fact that the critical mass has not yet been reached. Great feedback was provided by the test users, and most of the feedback provided has been applied. We have thoroughly enjoyed to hear that some of the test users have had fun interacting with the platform.

This system could provide useful insights into future events through the use of the wisdom of the crowds, as well as into the principle of 'wisdom of the crowds' itself. These insights will become especially insightful when the system is developed further and expanded upon (see Section 7.4). This, and the fact that all of the primary requirements and a substantial amount of the secondary requirements have been met, leads to the conclusion that this project can be seen as successful.

[1] Deployment Website: https://predictionmarket.ewi.tudelft.nl

Bibliography

[1] Why React. URL https://reactjs.org/blog/2013/06/05/why-react.html.

[2] Sigrid user manual, Dec 2019. URL https://sigrid-says.com/assets/sigrid_user_manual_20191224.pdf.

[3] Kenneth J Arrow, Robert Forsythe, Michael Gorham, Robert Hahn, Robin Hanson, John O Ledyard, Saul Levmore, Robert Litan, Paul Milgrom, Forrest D Nelson, et al. The promise of prediction markets. *Science-new york then washington-*, 320(5878):877, 2008.

[4] Ricardo Baeza-Yates. Bias on the web. *Communications of the ACM*, 61(6):54–61, 2018. doi: 10.1145/3209581.

[5] Malcolm Baker and Jeremy C Stein. Market liquidity as a sentiment indicator. *Journal of Financial Markets*, 7(3):271–299, 2004.

[6] Joyce Berg, Forrest Nelson, and Thomas Rietz. Accuracy and forecast standard error of prediction markets. *Tippie College of Business Administration, University of Iowa*, 2003.

[7] Joyce Berg, Robert Forsythe, Forrest Nelson, and Thomas Rietz. Results from a dozen years of election futures markets research. *Handbook of experimental economics results*, 1:742–751, 2008.

[8] Joyce E Berg and Thomas A Rietz. Prediction markets as decision support systems. *Information systems frontiers*, 5(1):79–93, 2003.

[9] Yiling Chen, Lance Fortnow, Nicolas Lambert, David M Pennock, and Jennifer Wortman. Complexity of combinatorial market makers. In *Proceedings of the 9th ACM conference on Electronic commerce*, pages 190–199, 2008.

[10] Robert Forsythe, Thomas A Rietz, and Thomas W Ross. Wishes, expectations and actions: a survey on price formation in election stock markets. *Journal of Economic Behavior & Organization*, 39(1):83–110, 1999.

[11] Brad Frost. *Atomic design*. Brad Frost Pittsburgh, 2016.

[12] R. M. Groves. *Survey errors and survey costs*. Wiley-Interscience.

[13] Juho Hamari, Jonna Koivisto, and Harri Sarsa. Does gamification work?–a literature review of empirical studies on gamification. In *2014 47th Hawaii international conference on system sciences*, pages 3025–3034. Ieee, 2014.

[14] Robin Hanson. Combinatorial information market design. *Information Systems Frontiers*, 5(1):107–119, 2003.

[15] Robin Hanson. Logarithmic markets coring rules for modular combinatorial information aggregation. *The Journal of Prediction Markets*, 1(1):3–15, 2007.

[16] Eszter Hargittai. Is bigger always better? potential biases of big data derived from social network sites. *The ANNALS of the American Academy of Political and Social Science*, 659(1):63–76, Sep 2015. doi: 10.1177/0002716215570866.

[17] Robin Hill. What sample size is “enough” in internet survey research. *Interpersonal Computing and Technology: An electronic journal for the 21st century*, 6(3-4):1–12, 1998.

[18] Ziva Kunda and Richard E Nisbett. Prediction and the partial understanding of the law of large numbers. *Journal of Experimental Social Psychology*, 22(4):339–354, 1986.

[19] Choong-Ki Lee, Namho Chung, and Bo J Bernhard. Examining the structural relationships among gambling motivation, passion, and consequences of internet sports betting. *Journal of Gambling Studies*, 30(4):845–858, 2014.

[20] Saul Levmore. Simply efficient markets and the role of regulation: Lessons from the iowa electronic markets and the hollywood stock exchange. *J. Corp. L.*, 28:589, 2002.

[21] Harold A Linstone, Murray Turoff, et al. *The delphi method*. Addison-Wesley Reading, MA, 1975.

[22] J. Lorenz, H. Rauhut, F. Schweitzer, and D. Helbing. How social influence can undermine the wisdom of crowd effect. *Proceedings of the National Academy of Sciences*, 108(22):9020–9025, 2011. doi: 10.1073/pnas.1008636108.

[23] Stefan Luckner and Christof Weinhardt. Arbitrage opportunities and market-making traders in prediction markets. In *2008 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services*, pages 53–59. IEEE, 2008.

[24] David Moberg and Carl Moller. Gamification of stock trading: Activating sleeping resources, 2019.

[25] Ulrich A Müller and Richard B Olsen. Method for market making, December 16 2008. US Patent 7,467,110.

[26] Allan H Murphy and Robert L Winkler. Probability forecasting in meteorology. *Journal of the American Statistical Association*, 79(387):489–500, 1984.

[27] Abraham Othman, David M Pennock, Daniel M Reeves, and Tuomas Sandholm. A practical liquidity-sensitive automated market maker. *ACM Transactions on Economics and Computation (TEAC)*, 1(3):1–25, 2013.

[28] Scott E Page. *The difference: how the power of diversity creates better groups, firms, schools, and societies*. Princeton University Press, 2007.

[29] Emile Servan-Schreiber, Justin Wolfers, David M Pennock, and Brian Galebach. Prediction markets: Does money matter? *Electronic markets*, 14(3):243–251, 2004.

[30] Shunrong Shen, Haomiao Jiang, and Tongda Zhang. Stock market forecasting using machine learning algorithms. *Department of Electrical Engineering, Stanford University, Stanford, CA*, pages 1–5, 2012.

[31] C. Slamka, B. Skiera, and M. Spann. Prediction market performance and market liquidity: A comparison of automated market makers. *IEEE Transactions on Engineering Management*, 60(1):169–185, 2013.

[32] Eric Smith, J Doyne Farmer, L szl Gillemot, Supriya Krishnamurthy, et al. Statistical theory of the continuous double auction. *Quantitative finance*, 3(6):481–514, 2003.

[33] Erik Snowberg, Justin Wolfers, and Eric Zitzewitz. Prediction markets for economic forecasting. In *Handbook of Economic Forecasting*, volume 2, pages 657–687. Elsevier, 2013.

[34] Thomas Stewart. Uncertainty, judgment, and error in prediction. *Prediction: Science, Decision Making, and the Future of Nature*, 01 2000.

[35] J. Surowiecki. *The wisdom of crowds*. Anchor Books, 2004.

[36] Derek Thomson. Considering an ipo to fuel your company's future? insight into the costs of going public and being public. *PwC, November*, 2017.

[37] Marlene E Turner and Anthony R Pratkanis. Twenty-five years of groupthink theory and research: Lessons from the evaluation of a theory. *Organizational Behavior and Human Decision Processes*, 73(2-3):105–115, 1998. doi: 10.1006/obhd.1998.2756.

[38] Justin Wolfers and Eric Zitzewitz. Prediction markets. *Journal of economic perspectives*, 18(2):107–126, 2004.

[39] Justin Wolfers and Eric Zitzewitz. Five open questions about prediction markets. Technical report, National Bureau of Economic Research, 2006.

Appendices

A

Project Info Sheet

General Information

Title Leveraging the wisdom of the crowds
Client Cybersecurity Group at the TU Delft
Date and time of presentation July 3, 2020, 16:00

Description

The challenge of this project is to create a play-money prediction market that can overcome the challenges faced by earlier prediction market. Research was done into gamification, market makers, market initialization, and the problems of other prediction markets. By iteratively developing the product, the team and client could prioritize features on a week-by-week basis and continue discussion on the possibilities of market makers. As the product is now ready for the public, it still remains to be seen if a critical mass of users will be gathered. Additional design and branding could further improve the chances of the product becoming successful.

Project Members

All members contributed to developing and preparing for the meetings, documentation, report and project presentation.

Erwin Dam

Student at the Delft University of Technology. Besides programming, primary interests are (modern) physics, science, history, politics, data visualisation, philosophy, mathematics, and more generally computer science- and hardware-related topics. Erwin contributed most on back-end development, most prominently on the parts related to the markets directly.

Marc Droogh

Student at the Delft University of Technology, with interests in finance and sports, besides programming. Likes to spend time at a student association and related activities. Marc contributed most on the back-end code base, of which most was related to user and handler functions.

Jeroen van Steijn

Student at the Delft University of Technology and owner of a business that develops web applications. Besides programming likes to play soccer and League of Legends in his spare time. Jeroen contributed most of the infrastructure, most of the front end and the architecture of the application.

Additional Information

Client

Name: Prof. Christian Doerr
Affiliation: Professor in Cyber Security and Enterprise Security at Hasso Plattner Institute / University of Potsdam and Associate Professor at TU Delft

Coach

Name: Dr. Przemysław Pawełczak
Affiliation: Assistant Professor at the Embedded Software Group at TU Delft

Contact

Jeroen van Steijn, jjvansteijn@gmail.com
Marc Droogh, marcdroogh@gmail.com
Erwin Dam, erwin.s.dam+predictionmarket@gmail.com

The final report for this project can be found at: http://repository.tudelft.nl

B

Project Description

When predictions and estimations given incomplete data need to be made, one potential option is to leverage the wisdom of the crowds. A classic example of this is the famous 1906 experiment where at a country fair visitors could win a cow if they guessed its weight. The average of all guesses matched the animal's weight exactly, in contrast to those guesses of "experts", such as farmers and butchers. Since then, prediction markets have implemented this principle through web sites, where participants bet with virtual money on the likelihood of an event to occur, and the average price of the "stock" approximates the accumulated wisdom of the market.

Prediction markets can however only work when enough participants have signed up and actively trade, and until now only very few of these have worked in practice. In 2006, an economist named Hanson published an algorithm called "Logarithmic Market Scoring Rule". The LMSR is a virtual buyer and trades with participants whenever no real counterpart is available. While on the stock market, there is always a buyer around when you are trying to sell some stock, the LMSR market maker fulfills this role in small markets and the resulting liquidity can thus also let small markets work.

In this project, you are implementing an LMSR prediction market. Users can pose questions and participants trade based on their opinion, with the LMSR stepping in if necessary to ensure that trades can always happen. We will investigate gamification strategies from the academic literature and implement an incentive mechanism and community features so that participants are enticed to engage in trade even on topics they are not directly involved in.

The system will be split into a front-end / back-end part, the technology stack is chosen together with the project team.

C

Requirements Document

Entity definitions

To clarify the requirements we have defined a number of entities. We suggest to look these up while reading the requirements.

Question Market

A question market is a system in which various answers to a given question are represented by shares and their respective values. Each question market has exactly one question to be answered, together with a definition that is used for deciding which answer is correct. This definition must be as objectively measure-able as possible. Must have an implicit or explicit defined end date.

Potential Answers

Each question market has two or more potential answers. These represent all possible answers to the question posed by the question market. Potential answers have two prices attached to them, namely a 'selling price' and a 'buying price', that are calculated by the market maker.

Resolving a question market

Once the correct answer of a question market becomes known, the market is resolved. The value of the correct answer will then be 100%, whilst all of the other answers will have a value of 0

Share

Shares represent a portion of the value of an option which represents a potential answer to a particular question market.

Hierarchically organized

Organised in order of rank, based on topic areas. [e.g. Sports -> Soccer -> Champions League]

Virtual Wallet

Quantifier which quantifies the amount of virtual currency funds a Registered Participant has at their disposal (to trade with).

Market maker

A market maker handles transactions (selling and buying) of stocks, and adjusts the price based on these transactions. The price is calculated and updated based on a logarithmic market scoring rule (LMSR).

Users

Users are all individuals interacting with the trading platform

Registered Participant
Registered Participants are all users that have an account on the trading platform

Moderator
A trusted user that has special rights on the platform to propose new question markets.

Administrator
An employee that creates questions, closes questions and resolves questions.

Periodical Login/Action Award
An award provided periodically on fulfillment of a task, an example being a reward on every day the Registered Participant logs in, or a reward on every day the Registered Participant makes a trade.

Third Party Single Sign-On
Third party authentication methodologies such as the ones provided by Facebook, Google, Apple, OpenID and LDAP

Must-haves
The must-haves of this project are divided into multiple categories to provide a better overview.

Interaction with the platform
The users must be able to interact with a web-based front end interface
Trading must only be accessible to registered participants
The interface for prices must be accessible for non registered participants
The users must be able to view a dashboard
The registered participants must be able to view personal recent trades on their dashboard
The registered participants must be able to view the performance of their trades in the past X days compared to the day before, or since they entered the market, on their dashboard

Question Markets
Administrators must be able to create question markets
Administrators must be able to assign one or more topic(s) to a question market
Administrators must be able to resolve a question market
Administrators must be able to close transactions of a question market
Topics of question markets must be hierarchically organized
The users should be able to view the historic price of a question market
The application must have a mechanism for determining a fair initial value for a question market.

Currency and transactions
he Registered Participants must each have one virtual wallet available to them
The Registered Participants must receive a quantity of the virtual currency in their virtual wallet on signup
The Registered Participants must be able to use virtual wallet funds for buying shares
The Registered Participants must be able to receive virtual wallet funds for selling shares (at value)
The Registered Participants must be rewarded for successful trades on question markets of which the outcome is known by means of the proceeds being added to their virtual wallet.
Question markets must use a market maker for transactions
The user must be informed when the price in the back end has changed from the price on their screen

Implementation
The front end must be connected to the back end with an API
The front end must use a library or external theme for styling the platform
The front end must use a library for generating graphs

Should-haves

The users must be able to view the most active markets in the last X days, based on a metric
The Registered Participants should receive a quantity of the virtual currency in their virtual wallet on intervals or as a periodical login/action award
The users should be able to become moderators
The users should be able to share a question market on social media platforms
The users should be able to embed the current stock price of a question market on their website using an iFrame
Users and moderators should be able to propose new question markets
The moderators should be able to close transactions of the question market they proposed
Administrators should be able to look at proposed question markets and finalize them into open question markets
Administrators should be able to upgrade particular users to moderators
Administrators should be able to add, edit and delete topics and their hierarchical structure
Administrators and moderators should be able to enrich question markets with additional material such as links and documents
The Registered Participants should be rewarded for proposing a question
The users should be able to earn badges/flairs for certain achievements
Administrators and moderators could be able to open a discussion forum under question markets

Could-haves

The users should always have a view of the current share price updated in real-time
The Registered Participants could place orders for buying and selling a share at a certain price
The users could be able to create an account and login using third party single sign-on
Administrators and moderators could be able to make a certain question market only open to a subset of users
Administrators and moderators could open a survey or vote on question markets that are not open yet so that the initial value of question markets can be gathered

Will-not-haves

The users will not be able to short-sell shares
The users will not be able to purchase derivatives of shares
The users will not be able to lend shares
There will not be any trading bots introduced to manipulate share value