People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

IBN KHALDOUN UNIVERSITY OF TIARET

Dissertation

Presented to:

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
DEPARTEMENT OF COMPUTER SCIENCE

in order to obtain the degree of :

MASTER

Specialty:
Software Engineering

Presented by:
Hosni Issam

On the theme:

Design and Development of a Crowdfunding Platform with Expert Validation for Startups

---

Defended publicly on .. / .. /2025 in Tiaret in front the jury composed of: :

Mr BELARBI Mostefa	Prof. Tiaret University	Chairman
Mr TALBI Omar	MCA Tiaret University	Supervisor
Mr ADDA Boualem	MCA Tiaret University	Examiner

2024-2025

Contents

List of Figures

List of Tables

Acknowledgments

All praises to Allah, the Almighty and the Exalted, for granting me success in completing this work and for His guidance in that regard. I ask Allah to bless this work so that it bears the existing fruit and is useful for knowledge.

I express my sincerest gratitude to the jury members for honoring us by attending this thesis defense. It is a privilege for me that **Mr. Mostefa Belarbi** has kindly accepted to preside over this jury, and that **Mr. Boualem Adda** has agreed to examine my work. I am truly grateful for their presence and willingness to share their expertise and insights, which greatly enrich this academic endeavor.

I also would like to express my deep appreciation to my supervisor **Mr. Omar Talbi** for his support and assistance throughout this project. His unwavering guidance, helpful advice, and commitment have always been key and remarkable, not only for the effective realization of this work, but since the beginning of my journey into college.

Last but not least, I would like to express my deepest and greatest appreciation to my beautiful family, my friends, and all of whom I admire and respect. I am immensely grateful for your unconditional support and patience throughout this journey. Your encouragement has significantly strengthened me, and I would not have been able to achieve this without you.

Abstract

Since the financial crisis of 2008, the focus on the mediums of funding has been shifted from the traditional ways to new ways that could have more robust mechanisms, and one of these modern ways is crowdfunding. Crowdfunding provided a vital and remarkable impact, linking project owners and people who want to pledge to them. Thus, it revolutionizes the funding process for startups and creative projects. However, persistent trust issues and information asymmetry between project creators and backers hinder their potential. These problems attracted researchers from both academia and the industry to resolve or at least mitigate the impact of them on the crowdfunding process. However, these *mechanisms* still don't provide a robust and objective way to augment the level of trust needed or acceptable. One of such mechanisms is third-party endorsements, more specifically the adaptation of experts' validation, which is applied through approval/rejection voting, which represents a less-expressible voting mechanism for the experts to evaluate projects based on. This thesis proposes a novel trust-enhancement mechanism, double-score approval voting, to improve the objectivity and robustness of expert validation systems in crowdfunding. By integrating weighted expertise scores with nuanced voting, this approach addresses the limitations of traditional binary voting models. This proposed solution is integrated through CertiFund, a crowdfunding platform that builds on existing advancements applied by other platforms, with the guides and key findings provided by researchers in the field. This work provides both the theoretical and the practical frameworks to present this solution, contributing to the context of trust enhancement in the crowdfunding context, and provides insights into future improvements and extensions that could build upon the established work to mitigate the impact of the identified gaps. Practical challenges and integration strategies are also discussed, providing a blueprint for academic and industrial researchers with the aim of advancing trust

in digital funding ecosystems.

Keywords: Crowdfunding, Trust, Signaling theory, Expert validation, Approval voting, Information asymmetry.

Part I

Theoretical Side

Chapter 1

Introduction

Since the rising of crowdfunding as an alternative and vital funding source alternative to banks, it's been used worldwide by major platforms like Kickstarter and IndieGoGo to link between creators of projects and the backers who are willing to *trust* and engage in funding those projects they believed in, and as we mentioned it first, trust is the crucial element that drives the incentives of backers towards creators [55]. Thus, creators and their projects, and therefore the platform itself [73]. In addition to that the information asymmetry represent a critical problem given that the amount of reliable and pertinent data that the backer has is predominantly inferior to what the creator have because he's the owner of the project, and despite the effort of platform to close the gap between the parties, it still presents a real challenge to deal with.

In this chapter, we will focus first on giving a general context about crowdfunding and the challenges it faces, along with the aim of our proposed solution to the matter. Next we will define clearly the problem that we are trying to address in this thesis, after that we will generalize the problem into research questions that will drive the research, followed by the objectives or the goals of the study and finally the significance value this study provides to the academic and industrial masses addressing the same or a closely related theme.

1.1 Background and Motivation

Banks have been the dominant and central institutions when it comes to loan provision and venture capital, standing behind the largest businesses and initiatives across micro- and macroeconomics, both for profitable and non-profitable causes. But since the occurrence of extreme *tail* events that showed how such a centralized system could be fragile to unexpected events, namely the great recession that occurred from late 2007 till mid-2009, causing a massive downfall to businesses, especially startups [25], considering that the lack of capital plays a major role in startups failure [54]. Since then, the focus has shifted from the traditional mediums of funding (banks) to new ways of raising funds that are susceptible to being more robust, therefore 'decentralizing' the 'centralized' mechanism of raising funds, and one of the modern ways that has achieved great success regarding the subject is crowdfunding. The startup ecosystem has quickly and flexibly, as its nature, emerged around crowdfunding, likewise all the different types of businesses, initiatives, charity works, and creative projects. Thereby, the global market size of crowdfunding is expected to reach USD 4.45 billion by 2032 [30]. Globally, a multitude of crowdfunding platforms are embracing the concept with minimal modifications. However, crowdfunding still faces challenges that represent tangible impediments to this growth, and some of the major ones are the trust issues and the information asymmetry between the creators and the crowd who funds them (backers), and as these problems have existed since the early days of the internet bubble and the rise of e-commerce [74] [12], they continue to exist and evolve in this field. And as platforms adopted several solutions like project preparedness, fundraiser engagement, and third-party endorsement [47]. And while current solutions have a remarkable impact in addressing the mentioned issues, they still lack a more robust and objective methodology. Expert validation, where qualified individuals assess and evaluate projects, offers a solution to bridge this gap between the parties and attain a reasonable level of trust that ensures the continuity of the crowdfunding process.

1.2 Problem Statement

Trust has consistently remained a crucial factor in the interactions and transactions between people, and since the rise of the digital world and the transformation that happened

to such transactions to be completely online, it remains an important aspect to the success and continuity of such interactions, in fact more important digitally than in reality [46]. [55]. Therefore, the need for trust on crowdfunding platforms is highly required due to the absence of expert investors, opaque or misleading information, and limited professional gatekeepers [55]; in addition to that, there is another issue that arises in such online interactions, which is the contrast between the information in terms of quality, or, to be more precise, the pertinence that the creator's side has versus the backer's side, designated as the information asymmetry. And given that one party (backers) is more concerned about the other party's behavior or intentions [9] [19]. And that information asymmetry between the two parties is higher in crowdfunding than in traditional funding [19], which exacerbates the issue, significantly impacting the decisions of backers to either hesitate, take unknown risks, or not engage in the funding process at all. The current mechanisms noted by researchers and applied by crowdfunding platforms, such as institutional mechanisms, particularly monitoring systems or signaling strategies like creators' prior experience as backers, project description and videos and third-party endorsements, though proved to have a remarkable impact in enhancing the experience of crowdfunding in platforms and addressing trust issues and information asymmetry [47] [84], still lacks more robust and objective ways to increase the credibility of creators and their competence [47] [84], pushing backers to rely on limited or biased signals and leaving them vulnerable to fraud or failure. This underscores the necessity for another party to be present in order to boost the confidence of backers in fundraising, forming a robust system for the assessment of projects on platforms, from critical aspects that are often not mentioned or noticed by backers, thus the necessity for a structured, expert-driven validation system. Although the use of an expert-validation system in crowdfunding platforms is not a novel concept and in fact it's used widely across many platforms, but there is a catch here, the academic study on the matter is still small given that only three recent papers had directly broached the subject, works from Petit and Wirtz (2021), Chen et al. (2024) and Hu et al. (2024), and it resembles on the impact of the expert validation applied on crowdfunding platforms worldwide. But despite the the good results the system gives in addressing the stated issues we mentioned, we assume that it doesn't give an objective way to the experts to assess better crowdfunding projects, mainly using the method of vote by **yes** or **no**, which does not provide a solid foundation for expert

validation that the backer may rely on with greater assurance., Thus, the need for an object expert-driven system to assess projects by experts remains a challenge that warrants more attention in both the academic and industrial societies.

1.3 Research Questions

The present thesis intends to address the following research questions (RQ):

- **RQ1:** How can expert validation systems in crowdfunding platforms be designed to ensure greater objectivity and robustness in project assessment?

  In a closer look, the current used method of expert validation in crowdfunding platforms is approval/rejection evaluations by simply voting with yes or no, and the way to address this is to shift focus towards the voting itself, therefore posing an additional question that will bring us closer to the answer: Which voting system is better than the yes/no method? And efforts to answer that have guided us to search for an even wider-range problem of which voting system is **best**, then turning back to which voting system is **best** and **applicable** to the crowdfunding context, and we found the one that belongs to the family of ranked voting systems that is called the approval voting system. Approval voting provides an excellent solution to achieve a remarkable *rational* voting system that guarantees maximum disclosure of different opinions that will be aggregated to shape a unified decision from parties that voted that represents well their opinion about the candidate(s) of voting, and that's what's needed to achieve more credibility in the assessment of projects.

- **RQ2:** What is the impact of a structured, more rigorous expert validation system on mitigating information asymmetry and fostering trust between creators and backers?

  As it's found on the recent study of *The Impact of Experts' Voting on the Fundraising Performance of Crowdfunding Projects (2024)* [39], the used expert validation systems in crowdfunding platforms are yielding favorable outcomes, from getting creators to disclose more information about their projects thus mitigating the gap of information asymmetry to trigger backers to align with their recommendations, originated from the herding effect which is the tending to follow the actions of

previous people that experienced that engagement before (in our case the funding of the previous backers) instead of following private self-acquired information [87]. Therefore, given all these results achieved with approval/rejection voting, what are the consequences of using a more rigorous voting system that not only reaches more rationality but offers more freedom for experts to express their decision about the level of credibility, competence, and preparedness of project locating in the axis of their expertise.

- **RQ3:** What are the practical challenges that face the integration of a robust, transparent voting mechanism within existing expert validation frameworks on crowdfunding platforms?

  To address this question, we planned, designed, and implemented CertiFund, a crowdfunding platform that implements a variant of approval voting that leverages the use of scores, offering experts more choice and a transparent process of selecting the recommendation level they believe best reflects the initiatives that are being offered for their evaluation. In the next chapters, we will explain more about the variant and talk about what caused the problem of integration in the platform and what we added, and changed to integrate it in a method we called in this thesis **double-score approval voting**.

1.4 Objectives

The present thesis tries to reach the following goals and objectives (OBJ):

- **OBJ1:** Provide a complete literature review about crowdfunding, from its origins to the rise of demand for it after the financial crisis of 2008 to the issues it faces in modern times, about trust and the crucial role of it in digital interactions and the rise of the information asymmetry problem, signaling theory as a major factor that influences and attracts backers to fund more and trust creators, and finally the existing trust enhancement mechanisms applied in existing platforms and discussed in the contemporary advancements in crowdfunding.
- **OBJ2:** Define the criteria and points of effort that the proposed solution of double-score approval voting compared to the approval/rejection model using existing ex-

pert validation systems applied within platforms, and note the implications and the benefits of applying such a solution in real-world scenarios by delivering it in a crowdfunding platform 'CertiFund' that builds upon existing trust enhancement mechanisms and adds as a main feature the solution we discussed.

- **OBJ3:** Design and implementation of a new and robust approach to expert validation that integrates with crowdfunding platforms, ensuring a wider range of decision-making offered to experts to better evaluate and assess projects on the different aspects and points of view that the project could be viewed from.
- **OBJ4:** Provide different guides and suggestions for crowdfunding platform owners to better reach wider audiences and convey more trust and credibility to backers, creators, and experts. In addition to academic researchers interested in the matter by filling and contributing to the filling of the stated research gaps, thus contributing to both industry and academia.

1.5 Methodology

The research methodology that has been utilized to conduct this work could be considered a hybrid one, starting with pure or fundamental research to seek and understand existing knowledge about the pillars that form the context of this thesis, which are crowdfunding, trust, expert validation, signaling theory, and voting. Therefore, a disciplined study has been performed to acquire as much knowledge as possible about the subject, reading and reviewing several scientific articles, journals, books, blogs, websites, and any source of information. Thereafter a complete analyze has carried out to identify all the flows and the limitations followed by an applied research in attempt to find practical solutions to the identified challenges, used in a form of a research and development mimicking the real world scenarios of such branches working at companies to provide practical solutions to fulfill the needs of costumers, which led to the creating of CertiFund which is a new approve of how an expert validation system should work, and we included the previous works and establishments noted by academic researchers and applied by existing platforms to make the platform evolve as quickly as possible in the market, in addition to the core feature of it. And furthermore the process of development of the platform was led by an empirical discipline, experimenting several hypothesizes and methods in a trial and

error to get the best possible result, which led to taking a multidisciplinary approach of looking at the identified problems that caused a research journey through engineering, economics, philosophy and psychology in order to formulate a good practical solution, and thus avoiding the falling into the trap of the hammer-nail syndrome (*"To the man with only a hammer, every problem looks like a nail"*), that could lead us to view or approach the problem from one point of view or aspect and missing others which are potential sources of information.

1.6 Conclusion

In this chapter we established a good basis, which we will build upon along the next chapters of this thesis; we clarified the motivations behind the work on this thesis, and we stated clearly the problems that we are trying to address, originating from the trust issues of the online interactions and the effects of them, like information asymmetry, followed by the objectives of this work that needs to be met and aligned with the research questions identified; and finally, we explained the methodology, or, to be specific, the methodologies that carried out what has been done to establish this work. Now, what we need is to dive deeper into the knowledge that forms the context of this work, which is deemed academically as the literature review, in the following chapter.

Chapter 2

Literature Review and Solution

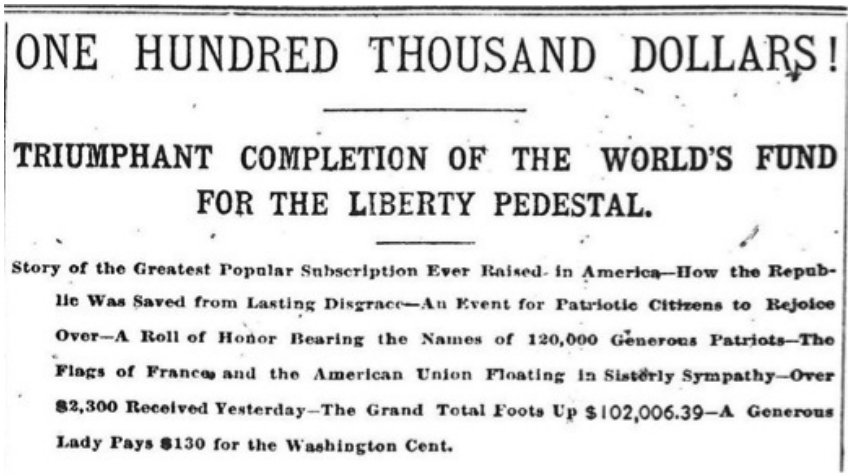
Whether it's startups or other businesses or forms of projects that provide something like products, services, or satisfaction of needs. Raising money has been indispensable in the creation of such initiatives, and while these projects, with their variety of types, have been introduced mainly with the brighter side of them, from success and achievement, there is a dark side that is rarely discussed and tackled: failure. And while the study of failure is as important as the study of success, as proven many times through history, like the work of the Statistical Research Group or SRG on shifting focus from the parts of the planes that made them safe to the parts of the planes that were known to be felled on the lands of Axis countries during WW2 [83], following a *via negativa* approach. Considering that if we focus on startups, we will see that in a recent study by TechCrunch (2025), data indicated that there is a 25.6% year-over-year increase in shutdowns tied to post-boom funding droughts, so to recapitulate, funding is a crucial problem that faces startups worldwide, forming what is called the cash flow death spiral. And though the rise of the alternative remedy in the form of crowdfunding, which marks and still does a great result, albeit that it still has crucial problems that its roots originated since day one of the internet, which are trust issues, platforms have provided mechanisms to resolve and address those problems, but still the problems continue to exist. One of the solutions that has potential is expert validation systems, centered around the idea of votes from different experts on projects to form a consensus around whether to approve or reject them. But since it's simply a binary choice of yes or no, that requires more attention to be given to it. In this chapter, we will synthesize the existing knowledge that forms

the pillars of our thesis, also known as the state of the art, first by crowdfunding, its definition, origins, types, and models of crowdfunding. Following that, we will look at trust by defining it and mentioning the work that has been done on it, along with its flaws that appear after the rise of the digital era. After that, we will look at signaling theory, which resembles the branch where most of the trust enhancement mechanisms lie, along with our proposed solution that has it part of approval voting, which we will end this chapter with by navigating through a brief history of it and how it works and what its variants are and an introduction of the solution proposed by this thesis, followed by a deep explanation of it, which is the double-score approval voting.

2.1 Crowdfunding

Crowdfunding is defined by Gerber and Hui (2013) as "the online request for resources from a distributed audience, often in exchange for a reward—provides a new way for individuals and teams to solicit financial support from a distributed audience"; hence, it's a public callout for different unknown persons that are considered outsiders to fund or *back* projects exhibited by creators in exchange for a reward or just for the sake of it, in other words, a donation. It allows the creators and project owners to seek and get funds for their projects without the traditional *centralized* way of financial fundraising through banks by utilizing intuitive platforms that make the process easier and also simplify the search for potential backers. Kickstarter, Indiegogo, and GoFundMe are some of the widely used and admired crowdfunding platforms, merely to cite a few examples [8]. The first recorded use of the term "crowdfunding" was in 2006 by the American writer Michael Sullivan when he needed a short word to describe the collective donations *funding* from people "crowd", thus crowdfunding [32]. Yet by a deep examination of history, we will determine that the **meaning** of the word existed way before its descriptive term had been coined, in a well-known phenomenon of the existence and practice of some action before its term even existed, better described in the book *through the language glass* by the linguist Guy Deutscher, where he reported that many primitive populations, without being color-blind, have verbal designations for only two or three colors. But with a simple test, they can effectively differentiate between given colors. Having stated that, by looking at history, there were many acts and initiatives that matched the definition

of crowdfunding; for example, the Statue of Liberty, one of the largest sculptures that exists in the world, was conceived in 1865 by Édouard de Laboulaye as a gift of friendship between France and the United States, and its process of moving it and establishing it on its pedestal was done by 1885. The two sides agreed that France would finance and build the statue, whereas the United States would provide the site and fund the pedestal. The construction of the statue by the French was financed mainly from fundraising, coming from public donations and contributions [53], and it was also funded by the Americans' pedestal part through fundraising. But there was a significant shortfall and slow progress in raising the needed funds, leading to a suspension of construction [68], which caused the launch of a major fundraising campaign led by the publisher of *The New York World*, Joseph Pulitzer (the one who's the Pulitzer Prize named after him). This campaign attracted over 120,000 contributors, with donations ranging from small amounts to large sums. Figure 2.1 shows the news coverage for the remarkable result of the campaign. This important event clearly shows an early instance of crowdfunding, where collective



ONE HUNDRED THOUSAND DOLLARS!

TRIUMPHANT COMPLETION OF THE WORLD'S FUND FOR THE LIBERTY PEDESTAL.

Story of the Greatest Popular Subscription Ever Raised in America—How the Republic Was Saved from Lasting Disgrace—An Event for Patriotic Citizens to Rejoice Over—A Roll of Honor Bearing the Names of 120,000 Generous Patriots—The Flags of France and the American Union Floating in Sisterly Sympathy—Over $2,300 Received Yesterday—The Grand Total Foots Up $102,006.39—A Generous Lady Pays $130 for the Washington Cent.

Figure 2.1: Article published on The New York World announcing the completion of Pulitzer's fundraising campaign

contributions of many individuals enabled the completion of a significant cultural landmark [68] [52].

Based on Agrawal et al. (2011), crowdfunding platforms could be divided into two main categories: community-based crowdfunding and financial-return crowdfunding. The former represents the action of funding often for the sake of support for the project or in exchange for compensation in the form of rewards; thus, we can subdivide this category into two sub-categories that are donation-based crowdfunding and reward-based crowd-

funding, respectively. [73] [43]. Moving to the latter (equity-based crowdfunding), we'll conclude that there is a difference in the process of funding compared to the former, for the reason that it concerns the gain of monetary profits as a consequence of funding, deemed as investment, presented in the guise of taking loans from backers similarly to banks or offering backers to own shares in the project (generally this is specific to startups) in return for the investment. Therefore, we can also subdivide this category into two sub-categories that are loan-based crowdfunding and equity-based crowdfunding, respectively. [73] [43], figure 4.31 presents a visual presentation of the crowdfunding categories hierarchy.

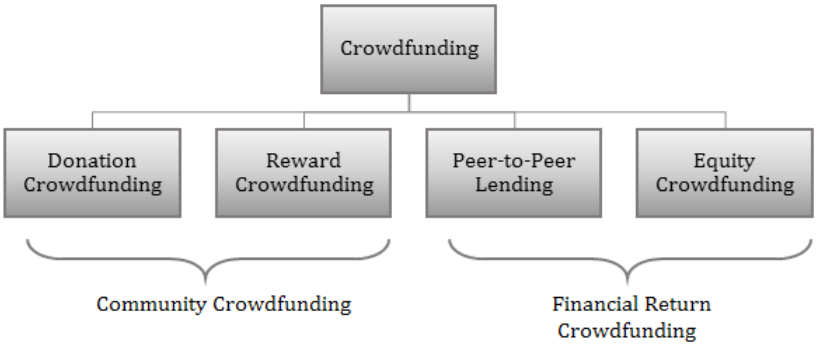


Figure 2.2: Crowdfunding categories and sub-categories, available here

Accordingly, this will lead to different stimuli that will attract different kinds of backers, as in community-based crowdfunding, the majority that rules will be unprofessional backers, and it will require more trust due to the fact that it's led mostly by psychological factors like emotions, ideology, the appeal of the creators' trustworthiness to them, etc., resembled in a descriptive term of *community logic* [29], thus leaving the other side to be professional backers that often commit funding in the platforms that adhere to financial return crowdfunding, which includes similar procedures of investment as any other type of investment in the market. More granular differences are illustrated in Figure 2.3. The traditional technique of fundraising for a project consists of publishing a campaign on a crowdfunding platform, setting a funding goal and deadline, detailing all information concerning the project and the use of different types of media to convey and persuade potential backers, which includes text descriptions, images, and videos [45]. Creators often focus on the formation of campaigns around certain key factors to achieve the maximum success possible and reach their goals, incorporating compelling storytelling, the excellent design of the campaign, the active promotion of the campaign through social

Features	Crowdfunding models (different names)			
	Donation-based Donation Crowddonation	**Reward-based Reward**	**Equity-based Equity Crowdinvesting**	**Lending-based Credit-based Crowdlending**
Motivation	Intrinsic, social	Intrinsic, social, extrinsic	Financial gain	Social or financial
Type of contribution	Donation	Pre-order	Investment	Loan
Expected return	Intangible benefits	Tangible and intangible benefits	Return on investment, profit sharing	Return on investment
Main focus	Philanthropy	Products for first adopters	Start-ups, SMEs	Short-term borrower
Complexity of the process	Very low	Low	High	Medium
Type of contract	A contracts without tangible reward	Purchase contract	Shareholding contract	Lending contract

Source: own study based on: Hossain, Oparaocha 2015, p. 40.

Figure 2.3: Key differences between crowdfunding categories, available here

media accounts owned by them (creators), and the strong networking of the entity behind the project formed by previous experiences. [45] [18]. While the campaign is ongoing, backers start to engage and fund projects, driven by forms of signal theory (which will be explained later on in this chapter) or by social proof or by following the past experiences of other backers that funded before on that project, following a *herd effect* (again, more to come on that later on in this chapter). The process of fundraising continues till the project reaches its deadline; after that, projects diverge into either failed projects, which didn't reach their funding goal, or successful ones, which did the inverse. But here a question poses itself, which is, in the case of failed projects, if, let's say, the funds raised reached 90% of the funding goal, is it irrecoverably gone? Here, crowdfunding platforms again get divided into two models, **Keep-It-All (KIA)** and **All-Or-Nothing (AON)**. As a recapitulation of what is detailed in table 2.1 about the differences between the two models, in the KIA model (used in platforms like IndieGoGo), it's permissible for creators to keep all raised money regardless of goal achievement, and although it looks like a good deal for creators and also backers, it's less successful than the other model, AON, where if you didn't reach your funding goal, then all the money raised is gone. As proved by Cumming et al. (2019), AON campaigns have higher completion ratios and attract more backers than KIA campaigns, and they involve larger capital goals than the latter. In addition, the AON model alleviates constraints on fundraising goals, leading to higher success rates. Nevertheless, the choice between the two models remains a decision that depends on project characteristics, funding goals, and team size.

Feature	All-Or-Nothing (AON)	Keep-It-All (KIA)
Funding Goal Requirement	Must meet the specified funding goal to receive any funds	Keeps all funds raised regardless of goal achievement
Risk Allocation	Shifts risk to the entrepreneur; if the goal is not met, no funds are received	Allocates risk to the crowd; Funds are used even if the project is underfunded
Success Rate	Higher success rates and completion ratios	Less successful in meeting fundraising goals
Project Type Preference	More suitable for larger, non-scalable projects	Preferred by small, scalable projects with lower fixed costs
Backer Insurance	Provides assurance to backers that the project will not proceed without sufficient funding	Less assurance for backers, as projects may proceed with inadequate funding

Table 2.1: Key differences between AON and KIA crowdfunding models

2.2 Trust

As defined by Moysidou and Hausberg (2020), "trust can be viewed as a measure of confidence that the trustee will behave in an expected manner and will refrain from opportunistic behavior"; therefore, trust is an important aspect that keeps relationships and interactions and, to a large extent, an essential pillar that forms civil society. The concept of trust has deep historical roots, with early forms emerging in various civilizations. In Islamic civilization, the concept of trust is closely linked to the concept of "Al Amanah", which resembles meanings of safety, confidence, integrity, and responsibility, and it extends to reach spiritual dimensions—fulfilling trust is seen as an act of worship and obedience to God (Allah) and sets among the highest and noblest virtues that form society [38]. The waqf, or Islamic trust, was a key legal entity that reflected confidence in the Islamic society. The treatises of the waqf have covered a wide range of good deeds that contributed to the welfare of society and incorporated aspects of charitable giving, inheritance, and bequest. The legitimization of the waqf was done through the traditions of Prophet Muhammad (PBUH) and his companions and became a foundational institution for social welfare and charitable endowments in Islamic civilization [37]. The structure and function of the waqf have been compared to the later Western legal concept of the trust, or the common law trust, that was used during medieval England. The chancellor

that administered the rules during that time developed the legal system of "equity" that provided remedies that were unavailable in prior rules, which led to the need for "use", a legal instrument that allowed one person to hold property for the benefit of another, which eventually evolved into the modern trust [61].

1995, the peak period of the internet or the dot-com bubble, with the rise of the digital world and its applications like the growth of e-commerce, therefore the lack of face-to-face interactions and the anonymity of the medium created a sense of uncertainty among users, which led to severe trust issues between them [46]. Furthermore, given the fact that in digital connections, trust is more crucial and significant than in real-world interactions. [55], that made the basis for the manifestation of many problems like the "lemons problem" or the "cyber lemons" problem, which refers to the situation where low-quality products (lemons) exile high-quality products in a market [60], due to the lack of transparency and quality guarantees and, above all, information asymmetry. Information asymmetry refers to the imbalance of information between two sides that form the agents of that transaction or interaction; it could be buyers and sellers in e-commerce or, in our case of crowdfunding, backers and creators. This results in imbalances in decision-making and potential market inefficiencies, and the anonymity of the internet exacerbates the problem even further, because without a physical inspection of products, interactions with the other party, or a direct assessment of the quality and credibility of products or services, the customers are more likely to experience uncertainty and distrust [79]. Crowdfunding platforms, in particular, are identified by high levels of information asymmetry compared to others forms of financial interactions or traditional funding [21], due to the early-stage nature of projects and the limited availability of information, making it difficult for backers to accurately evaluate project outcomes and creator's reliability [4], and consequently leading to market failures, where high-quality projects remains unfunded whereas low-quality ones attract funding, leading to distrust from backers, that are often trust creators initially and later verifies and adjusts trust beliefs accordingly, or the concept of swift trust [55], and that will leads to distrust in the projects, the platforms and potentially the concept of crowdfunding itself, as a form of trust transference, in a ascending or descending order [55] [73].

In order to respond to the trust issues and information asymmetry, various mechanisms were developed and applied over time. Considering the crowdfunding context, the earliest

forms of trust building applied by platforms were growing platform credibility and the quality of information provided by backers, which proved to be more influential than trust in the creators themselves, highlighting the central role of the platform as an intermediary in early crowdfunding models [55]. Another mechanism was institutional mechanisms, such as platform rules, monitoring, and funding security were also introduced to address these problems, providing protection for backers and also influencing their trust and distrust in both the platform and creators [73]. Another important mechanism was the implementation of signaling theory (more in the next section), which suggests the necessity of platforms and creators to employ a wider array of trust signals, including mainly project preparedness, consisting of all the information, images, and videos, etc.; creator's engagement through interactions with potential backers and active social media presence; and third-party endorsement, which includes all the possible ways to commit marketing for the project, from comments to shares on social media platforms, endorsement from online celebrities, or evaluations of experts [47]. [39], contributing to signaling trustworthiness and the reduction of information asymmetry. Even the design and features of the crowdfunding platform, like the credential information, interactive communication, and the use of narrative style and branding, play a critical role in fostering both cognitive and effective trust, particularly attracting younger generations [64]. Trust building in crowdfunding has taken a multidimensional form, broadened to include trust in the platform, project, other users, and the overall concept of crowdfunding [28]. Furthermore, trust merges in two forms: calculus trust, forming a competence-based dimension initiated from rational evaluations of information such as videos and endorsements, and relationship trust, forming a benevolence-based dimension rooted in interactions between creators and backers [47]. The interplay between the two forms shapes the funding intentions because they complement each other; thus, both forms are essential to reduce relational risks and enhance credibility.

2.3 Signaling Theory

Signaling theory is a concept that addresses the issue of information asymmetry between two parties, typically referred to as senders and receivers. It explains when one party, which is the sender, must decide how to communicate pertinent information to the other

party of receivers, indicating that by transmitting visible signals with reliable information about unobservable attributes, receivers with insufficient information can make decisions in the face of ambiguity [19]. A significant aspect of signaling theory is that the strength of signals differentiates between high-quality and low-quality senders; therefore, it's always tied to a cost, which helps to differentiate between high-quality and low-quality senders. Thus, in order for the signal to be credible, it must incur some cost to the sender, because if the signal was free and available to all, anyone could mimic high-quality parties and send the same signals, which makes it harder for receivers to know the difference between them [19]. As an example, let's consider the obtaining of ISO9000 certification for manufacturers. It's indeed a costly process that high-quality parties could afford compared to low-quality ones, who would need to implement significant changes to meet the certification standards. The theory also distinguishes between two types of equilibria, or states of balance, namely separating equilibrium, which we explained just before, where high-quality and low-quality are differentiated based on their signals, and the other type, pooling equilibrium, where both types may send the same signals, making it hard to distinguish them apart [19].

Signaling theory has been applied in various fields, including economics, anthropology, and marketing, which indicates its importance in analyzing behaviors across different contexts. In crowdfunding, creators use various signals to convey project quality, credibility, and trustworthiness to attract potential backers, who often lack direct means to verify these attributes themselves [19] [82]. Signals in crowdfunding can be categorized into several types (Figure 2.4), Project preparedness consisted of the quality and completeness of project materials like project descriptions, narrative quality, images, and videos [47]. It includes the proper choices of funding goal and campaign duration [11]. Another type is fundraiser (creator) engagement, which is the active participation of the creator in communicating with backers throughout the campaign, providing them with consistent updates about the progress of the campaign, responding to comments, and preserving an active involvement in the crowdfunding community [47]. Third-party endorsement, a *third* type of the ones mentioned already, refers to external validation of the project, which can come from endorsements from other backers in the form of comments or shares on social media platforms like Facebook or from professional investors or platform intermediaries (such as the platform's own equity stake) [21]. Or from expert evaluations

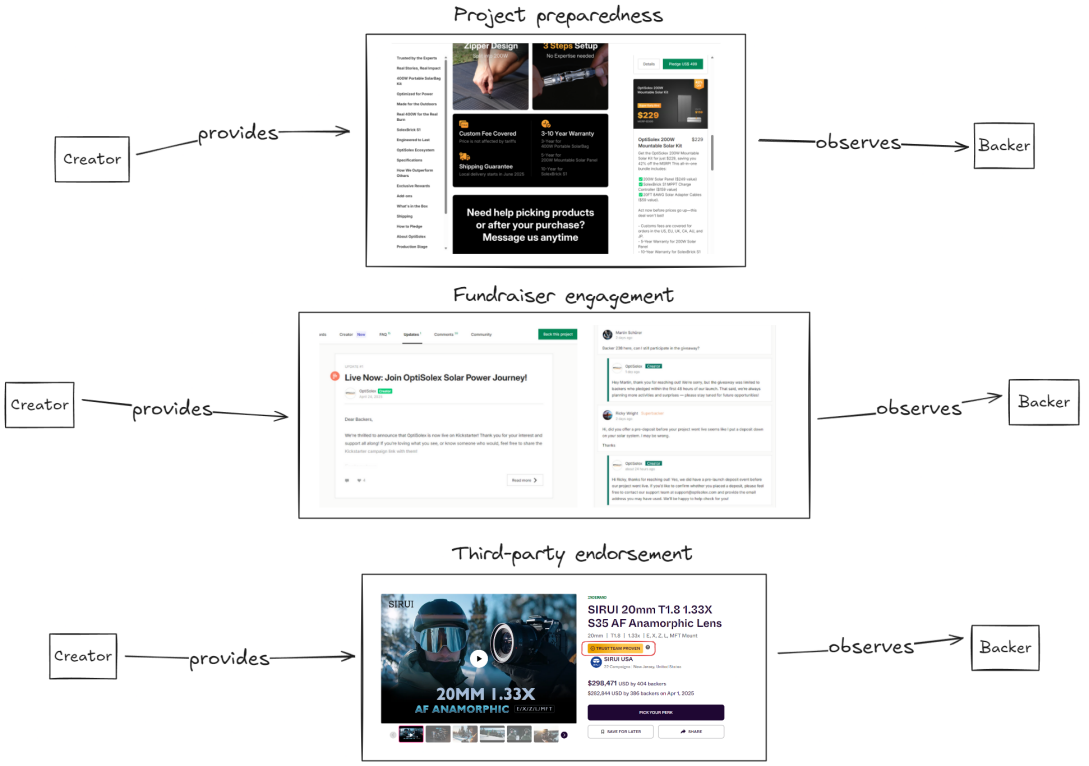


Figure 2.4: Signaling in crowdfunding - How creators use various signals to convey project quality to potential backers (pictures were taken from Kickstarter and IndieGoGo).

of projects through voting (more in the next section), which evaluate projects based on several factors determined mainly by the expert's specialization. These main signals and others impact the crowdfunding performance positively when worked in combination [47], and not only complement each other but can also substitute weaker signals like project preparedness [47]. Therefore, not all signals have equal effectiveness; for example, high funding goals or long campaign durations result in a non-linear *convex* relationship with success, and signals like reputation may even hinder outcomes if perceived as overreaching or insincere [14], thus the effectiveness of signals can depend on the combination of different signals and the context of the platform (reward, equity, donation) [47]. [21].

2.4 Critical Analysis and Evolution of Trust-Enhancement Mechanisms

Trust is fundamental to the success of crowdfunding platforms, as it directly influences backers' willingness to fund and perceive a probable and expected return or to fund without a substantiated need, as a donation. The unique risks related to crowdfunding,

namely fraud, misappropriation of funds, and performance opacity, make trust-building mechanisms essential for both platform sustainability and project(s) success [55] [28]. To recapitulate what is explained in the previous sections, existing trust-enhancement mechanisms play a reasonable role in mitigating the impact of distrust, but they still lack an objective basis to stand upon. Institutional protective mechanisms—mechanisms implemented by the platform to protect backers from opportunistic behavior—include transparent information disclosure, monitoring systems, platform rules, and recent leverage of blockchain technology. All have a significant influence on funding behavior [73] [80], and it guarantees a certain level of transparency and accountability. However, because these mechanisms still could be bypassed by certain techniques of fraud, even blockchain-based systems may lack robust fraud detection without additional oversight [85]; also, many existing platforms lack credible rule-enforcement mechanisms, which leaves backers exposed to risks of misappropriation of funds and misrepresentation of project details [3]. Additionally, things like platform rules have some impact but not so significant, which represents a problem that could affect the performance of crowdfunding on the platform [73].

Signaling strategies also drive success: creators' experience as backers increases participation, while creator history boosts total funds. Moving to images, videos, and project descriptions, they have a primary goal of aiding attainment rather than post-success funding [84]. But the overreliance on visual cues may not fully address deeper trust issues if not supported by verifiable information, which proves the higher priority of information quality, and although the latter (information quality) is critical, if not independently verified, it may still be manipulated [55]. Moving to the psychological and design features, established to have an important role in building both cognitive and effective trust (calculus-based and benevolence-based), especially among young backers, which defines a challenge to design interactive, convivial interfaces and clear credential information, preventing erosion of trust [64]. Third-party endorsements, paired with project preparedness and fundraiser engagement, synergistically enhance performance but still lack a more robust and objective way to increase the credibility of the fundraisers (creators) and their competence [47] [84]. In the case of third-party endorsements, it's noticeable to indicate the unreliability of it since it's mainly focused on the previous backers' comments and sharing throughout social media, triggering effects of social proof, where individuals look

to the actions of others to guide their own behavior [69] [47], a herding behavior where people imitate the actions of others [77]. And incentive biases of backers to engage in funding for merely the reason of helping projects reach their funding goals, without purely economic considerations, especially as campaigns near their target. Creating a prosocial motivation by the perceived impact of their contribution on the campaign success [23], therefore we can notice a level of subjectivity ongoing here, which necessitates reliance on different factors, objective professional *gatekeepers* that help backers decide better, contributing to fostering more *calculus trust* [55].

Advancements in the field of third-party endorsements have led to expert validation systems, which are systems composed of experts offering objective and reliable evaluations of projects, giving signals about the competence, credibility, and trustworthiness of them based on their (experts') fields of expertise. These systems typically involve experts voting on or commenting on projects before or during the fundraising process [39] [62]. The process of voting (which is what this thesis focuses on) is typically used through approval/rejection voting, as in voting with *yes* or *no* [39], and although the current performance of the validation systems is satisfactory, it still has some flaws, given that binary voting will restrict the decisions of experts to either yes or no, 1 or 0, and that will not give a full picture for rational voting to be, and this represents in its depth a loss of information. Moreover, it's not in the best interest of the platform to limit the specialization of experts to just one field, given that many experts are multi-disciplinary and projects belong to different categories, which form multi-dimensional aspects to consider in order to better evaluate projects.

Voting is an interesting topic; people from centuries ago have been trying to develop the best possible way to select, or precisely *elect*, things, people, places to visit, food to eat, decisions to make, etc., which led to a virtuous circle of developing and enhancing voting methods. Early works of Jean-Charles de Borda and Marquis de Condorcet were pivotal milestones that shaped voting methods around the era of the French Revolution. De Borda's method, called the Borda count, was to rank candidates by preference, which would give them points in a descending order, but it had a problem in that, the number of points given to each candidate is dependent on the total number of candidates, so adding extra candidates that have zero chance of winning will affect the winner [78]. Condorcet's work was a response to what de Borda had done in his Borda count method, where voters

rank their preferences and then count how many voters rank each candidate higher than each other candidate, but it has a flaw: it creates loops of preferences (A preferred to B preferred to C preferred to A), or what is known as Condorcet's paradox [59].
Sometimes in order to advance on the subject, you have to invert the thinking to know the borders, the limits, instead of how you get better on the subject. That's how brilliant minds like Claude Shannon made the digital age, and that's what the economist and mathematician Kenneth Arrow followed. In 1951, Arrow published his PhD thesis, and in it, he outlined five conditions that a rational voting system should have: unanimity, no dictators, unrestricted domain, transitivity, and independence of irrelevant alternatives. And above that, he proved that it's impossible to satisfy all five conditions in a ranked voting system [51]. Nevertheless, the mathematician Duncan Black found a much more optimistic way to vote, which leverages the effect of the median voter [5], avoiding the paradoxes and inconsistencies highlighted by Arrow. Moreover, Arrow's impossibility theorem applies only to typical ranked voting systems, but there are other systems called rated voting systems; among these, approval voting stands out for its simplicity and theoretical advantages. In approval voting, voters can approve as many candidates as they wish, and the candidate(s) with the most approvals win (Table 2.2). This method has been shown to be more fair and sincere than other voting systems where voters simply approve or reject each candidate [7] [6]. Approval voting has seen many variants, but we

	Candidate A	**Candidate B**	**Candidate C**
Voter 1	✗	✗	
Voter 2		✗	✗
Voter 3	✗	✗	

Table 2.2: Simple explanation of how approval voting works (winner here is candidate B).

will focus on one, which is range or score voting. In this system, voters assign a numerical score to each candidate within a specific range (for example, 0 to 10), and the candidate with the highest average score wins [71]. This variant has proven to outperform other voting methods in terms of minimizing collective regret, and through scoring, it does a great job in representing a more granular view of voters' preferences [70].

2.5 Double-score Voting: A new Approach

Returning to the example provided in table 2.2, and to apply our method of voting in the context of crowdfunding, we will make the voters experts, which will vote or particularly *evaluate* projects and decide whether to recommend them or not, so it's logical to set the candidates as levels of recommendation (Highly Not Recommended **HNR**, Not Recommended **NR**, Recommended **R**, Highly Recommended **HR**) and demand from the experts to choose between them, let's see table 2.3:

	HNR	**NR**	**R**	**HR**
Expert 1	✗		✗	
Expert 2			✗	✗
Expert 3			✗	✗

Table 2.3: Attempt to apply approval voting to the context of crowdfunding.

Obviously, the winner in this case will be recommended choice, which makes the experts' decision about the target project to be **Recommended**, but here we notice something illogical or irrational, to be precise. The principle of approval voting is to check on the candidate(s) they approve, but given this right to experts, they will go and check on highly not recommended and recommended at the same time! (expert 1's selections), And this doesn't seem to be logical. Thus, we moved to the variant of approval voting, which is score voting that is based on giving distributed scores to the candidates. This approach satisfies the condition to give experts a wider range of choices that can fairly and **logically** be represented in the vote, let's check table 2.4:

	HNR	**NR**	**R**	**HR**
Expert 1	0.0	0.2	0.5	0.3
Expert 2	0.1	0.6	0.2	0.1
Expert 3	0.0	0.4	0.4	0.1

Table 2.4: Applying score voting to the context of crowdfunding (winner here is Not recommended).

Now the perspective has changed, instead of "approve the candidates you like" it's "distribute the percentage of how likely that candidate represents the final decision about the given target". In our case, the percentage of how likely that recommendation level represents the opinion of the expert towards the target project (smaller values from 0 to 1 are used here to simplify calculations). This is better, like that the expert has more

freedom to evaluate the project, and all of the doubts or hesitancy will not be lost but present in the vote (for example if the expert didn't know if the level of recommendation of a project is either not recommended or recommended he can simply distribute 50% between the two, and leaving 0% for the rest). Now, to integrate this method into our platform, it would be simple to assign experts to projects in their field of expertise, and they will evaluate them, and the winner is the one who gets the highest score.

Now, in light of what we stated before that projects could belong to one or many categories, say a project belongs to technology and art, in the platform, it will be assigned to technology or art experts, right? But as we said before, that projects represents a complex system that needs a multidisciplinary approach, thus we also need experts in both fields to have their opinion, which will be very important. Now given that there is also another problem, not all experts have the same level of expertise in a certain field, and this represents a missing information, and to the nature of vote the opinion of a *lesser* expert would be equal to the opinion of a *more* expert. Which is a problem rooted back to the notion of democracy itself that got criticized by big philosophers like Socrates [76] who said in a famous quote: "Foolish leaders of Democracy, which is a charming form of government, full of variety and disorder, and dispensing a sort of equality to equals and unequaled alike.". Considering that we proposed to add another *score* which will balance the process a bit, that is a score of experts, measured by number of factors like experience, credentials, h-index for the case of an academic expert, certifications etc., Therefore we have **double-score voting**, consisted of the score assigned to the candidates (or the recommendation levels) and the score of the expert himself. To formalize more, here is the algorithm of the double-score voting:

For each recommendation level $l \in \{HNR, NR, R, HR\}$, the score S_l in a double-weighted recommendation voting system is computed as:

$$S_l = \sum_{e \in E} \lambda_e \cdot w_e^l$$

The final recommendation is determined by:

$$\text{Final Recommendation} = \arg \max_{l \in \{HNR, NR, R, HR\}} S_l$$

Where:

- E: Set of experts evaluating the project.
- λ_e: Expertise weight of expert e, where $\lambda_e \geq 0$.
- $w_e^{HNR}, w_e^{NR}, w_e^R, w_e^{HR}$: Weights assigned by expert e to "highly not recommended" (HNR), "not recommended" (NR), "recommended" (R), and "highly recommended" (HR), with $w_e^{HNR} + w_e^{NR} + w_e^R + w_e^{HR} = 1$.

This method will take both scores into consideration and merge them into one coherent way to achieve more fairness and flexibility in the evaluation of projects by experts.

2.6 Conclusion

The literature review highlights a rapid expansion of crowdfunding, with proven to be a vital funding mechanism, attracting several types of backers to engage with innovative and creative projects that merit. With the different types and models of it, it thrives to encompass a wide area of initiatives that have the potential to provide value, to do great in the market, and to succeed. However, despite the bright side of crowdfunding, it still has some issues to address, which tackle the smoothness of the process. Trust issues and information asymmetry have been a great challenge to address in the face of not only crowdfunding but also e-commerce and other forms of digital interactions that involves risk. Crowdfunding platforms didn't stand still but took the initiative to apply different strategies and mechanisms to mitigate the impact of these problems. These mechanisms reached a certain level of efficiency, but as we found out, it still doesn't provide more reliable factors to stand upon, which represents an obstacle in the way, which fuels the drive to solve it. Experts validation systems was a part of third-party endorsements that was needed to be provided, given the fact of absence for expertise was obvious with the situation of uncertainty and the lack of objectivity signals facing the backers, and on the other hand, the fraud and information misleading susceptible from the side of creators, and though the application of it but we saw that it still not robust enough, especially in the part of decisions from the experts. And throughout a theoretical journey in the methods of voting, we found that rated voting systems, especially approval voting with its variants, had proved to be a better and more rational way to represent reality and

the decision of who to elect. Therefore, contributing to the wisdom of decision making is vastly more important than knowledge.

"In theory there is no difference between theory and practice; in practice there is."

- Yogi Berra

In the next part we will be looking at the concrete, the practical, the interactive. CertiFund, an expert-driven crowdfunding platform that applies the best fruits of the theory of voting methods, driven by the motivation of providing a robust and interactive platform that people can easily use and rely on, and building on the mechanisms used before, following an approach of familiarity with the new. Next.

Part II

Practical Side

Chapter 3

Platform Blueprint

Following the theoretical foundations established in the previous chapters, this practical part, beginning with this chapter, seeks to apply the core concepts and frameworks to a real-world context. Based on what has been done before in existing crowdfunding platforms and adding what this thesis contributes, we try to move from the abstract to the tangible implementation, aiming to test the validity and applicability of what we suggested as a solution, which is the double-score approval voting. This chapter will encompass the core workflows ongoing on the platform with the functionalities provided by the system and the higher-level system design choices and details that forms CertiFund platform, including the high-level design choices, database and API design, security strategies, and finally the chosen technical stack. This chapter represents an important and pivotal role that links the theoretical side that we provided before, with the real world implementation methods and techniques in the upcoming chapter (chapter 4).

3.1 Functionalities and Workflows

3.1.1 Functionalities

One of the early and important phases in the software development life cycle is the definition of requirements, which establishes the basis for what will become the features and functionalities of the system [33]. As modern software engineering evolved and advanced more, It is observable that requirements phase is a vague and less likely complete initial state, which removes the used sentence to describe the phase as “requirements gather-

ing", given the fact that the requirements are rarely clear and well-defined and overall *gathered* all. This led to rather describe it as "exploration", done in an iterative feedback-looped way [41]. Considering also that the requirements don't need to be too specific, which paves the way for a more detailed view provided by other coming phases, like design. The needs that will shape CertiFund platform will not but so distinctive from any other major crowdfunding platform out there, namely Kickstarter and IndieGoGo, with of course additional needs that matches with the core *new* feature this platform provides. The main participants identified for the platform will be of two sides: privileged and non-privileged participants, For the non-privileged participants, there are creators, backers, and non-users which the latter is distinctive for strictly performing public actions on the platform, like searching for projects or accessing them. But despite that, the three participants share mostly the same use cases on the platform with little divergence between them, especially between creator and backer, given that a creator for a project could be a backer for another project, and so on. Moving to privileged participants, we identified administrators, reviewers, and experts, each are privileged than the other to perform certain tasks or use cases which will not be included in the others' responsibilities. The following use case diagram (figure 3.1) explains more.

Figure 3.1: Use case diagram representing CertiFund platform's functionalities and system's participants.

The Noticeable thing in the diagram is the use of two systems, which are public and private, and that's because, like we said in the functionalities provided by the platform, there are actions that gets done publicly without any form of registration or authentication. On the other hand, there are another actions that needs authentication due to it's sensitivity or to the duty of the platform to gain users.

3.1.2 Workflows

CertiFund system consists of several key workflows that we necessarily need to examine in order to understand the system in depth, the choice to demonstrate these workflows was activity diagram, due to it's excellence in illustrating workflows, simplifying processes, and provides clarity and straightforward understanding of the flow of control within the system, and most importantly the effectiveness of it in representing concurrent flows. Which made it a great choice then other modeling mediums like BPMN[1], which is necessary for representing more complex processes, or sequence diagrams, which are more suitable for nuanced interactions and message flows. Here are the key workflows of the system:

- **Project Creation:**

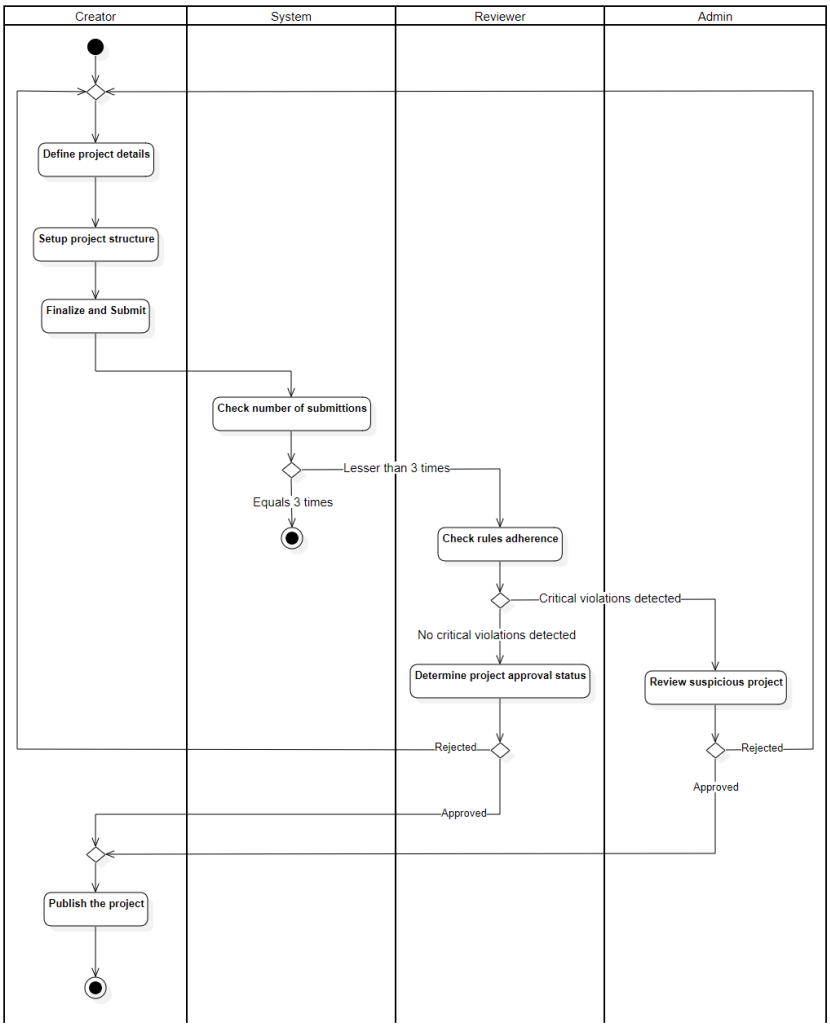


Figure 3.2: Activity diagram representing the steps of creating and reviewing projects.

[1]Business Process Model and Notation

- **Project Backing:**

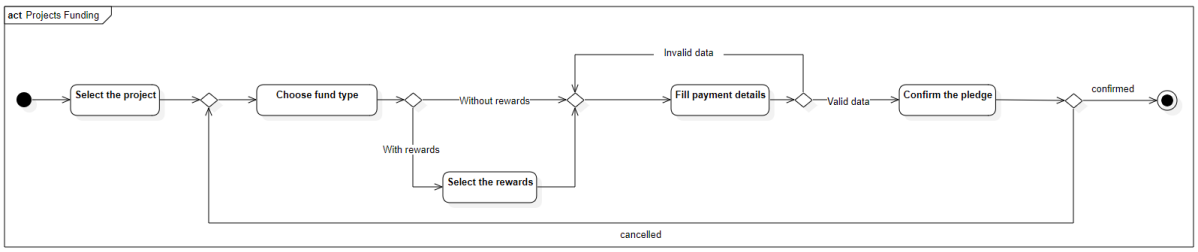


Figure 3.3: Activity diagram representing the steps for funding projects (horizontal swimlines because the process is simple and involves one actor (backer)).

- **Experts Validation:**

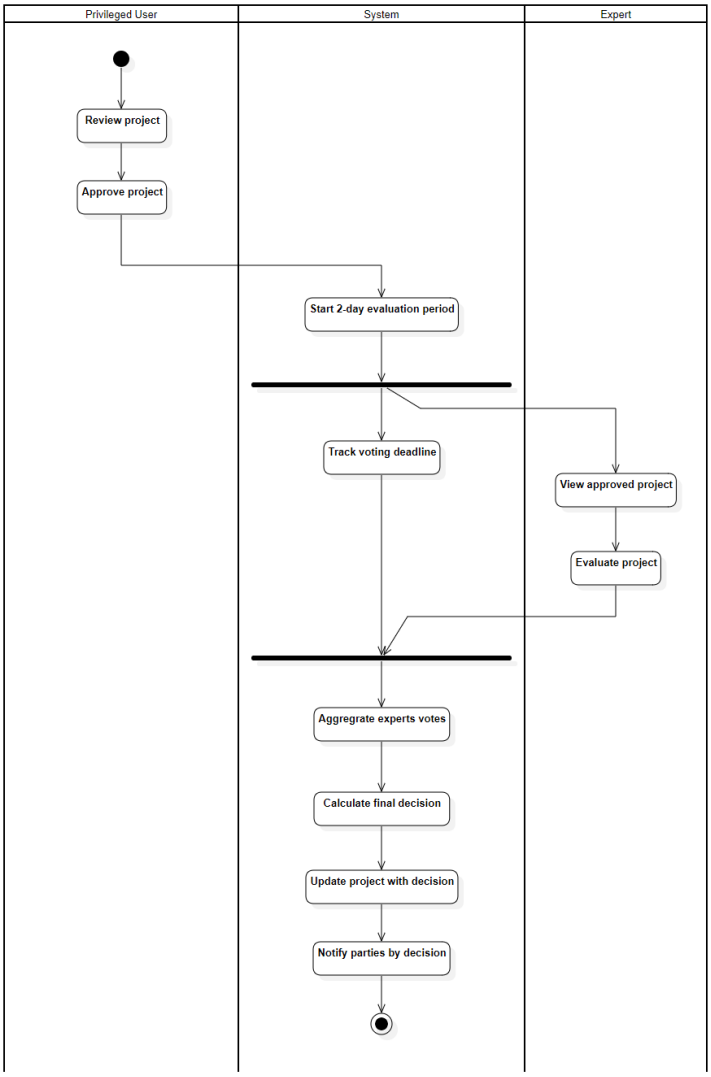


Figure 3.4: Activity diagram that represents the experts validation process (privileged user here is either an admin or a reviewer).

- **Dispute Handling:**

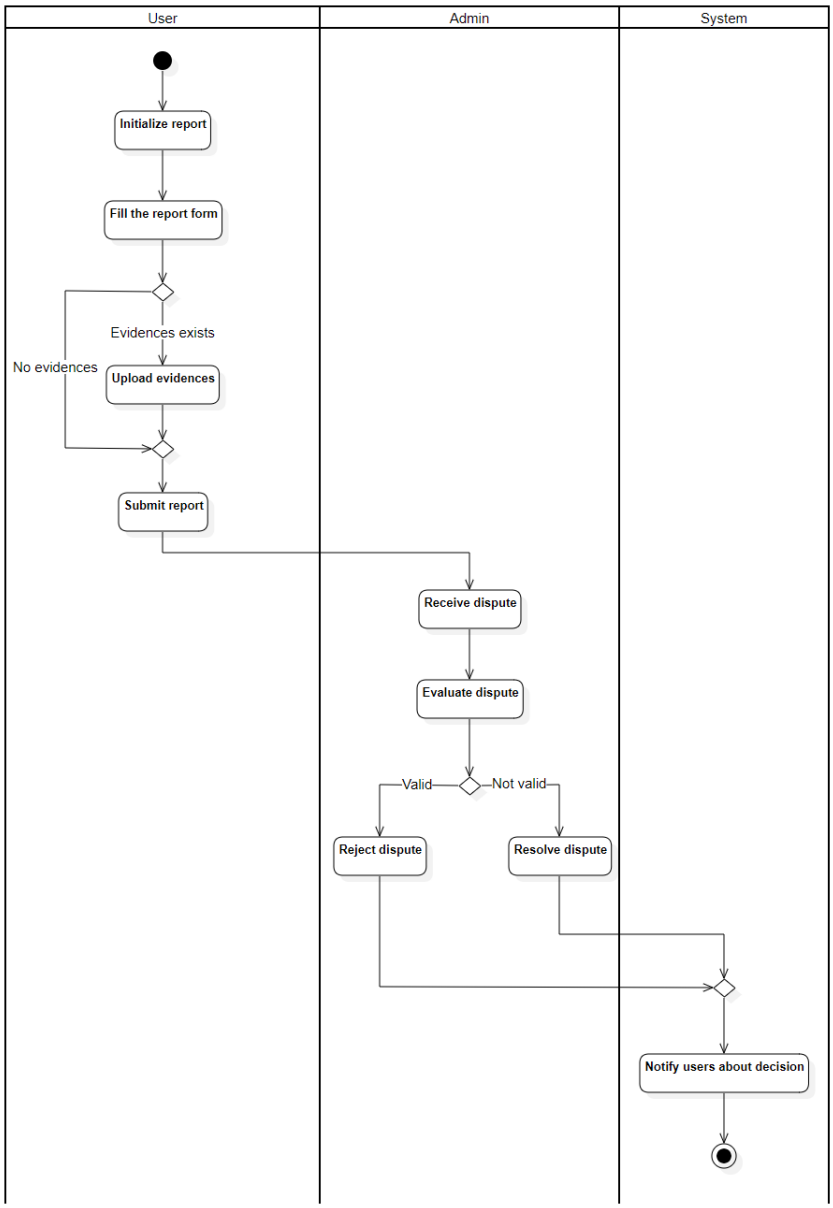


Figure 3.5: Activity diagram that represents dispute handling process.

3.2 High-level Design

High-level design (HLD) provides a wider look at the overall architecture and the main components that comprise the system. Thus, its subject of focus is to demonstrate what the system will do, and how the elements of it integrate with each other [24]. The process of conducting HLD plays a crucial role given that it has significant trade-offs to apply, which it's impact doesn't often seems clear in the beginning but important and critical in the late stages of developing software, which makes it a high-priority decision to

follow on for the rest of the SDLC[1]. CertiFund platform operates as a centralized web application that connects creators with backers through a structured process involving expert validation. The system follows a monolithic, layered architecture, one of the most common architecture styles out there, due to its simplicity and low cost, and familiarity among developers. It also follows Conway's law, which is the tendency to design systems in such a way that is similar to the communication structures of these teams. Given that the typical roles forming the teams that work on web-application systems are Frontend developers, Backend developers, and Database experts [67]. This architecture (layered) is organized into three architectural layers or main components, called presentation, application, and data (it's also often called presentation, domain, and data layers). Each layer is responsible for a specific aspects and roles that forms the platform:

3.2.1 Presentation Layer

The presentation layer represents the layer that directly faces the users when they access the platform. Consisted of the aesthetic choices based on UI/UX principles in order to conduct a better presentation for the users, combining aspects of interactivity, responsiveness, and intuitivism to create a better experience for users. The process of functioning for this layer is to present static data or the browser-side data to user, takes data from the users and deliver it to the proper application side or server side to deal with, and retrieve information back from the server side back to present it to user [16].

3.2.2 Application Layer

The application layer has the pivotal role in the application, that is, it manages the business logic, processes user requests, coordinates interactions between the other layers (presentation and data), and also with external services. The ordinary functioning this layer performs is to get data from the presentation layer (without the need to know where it's coming from)., it performs and execute specific business logic and rules based on the request and the given data, and either return the response to the presentation layer or pass it to data layer for persistence. For pre-existed data in the persistence layer, it can also retrieve that data and pass it to the presentation layer to be displayed in a properly formatted way [63].

[1]Software Development Life Cycle

3.2.3 Data Layer

The purpose of data layer is to give *memory* for the application, imagine a platform where you have to sign up every time you access it, or to redo every action you did before since it has a lost of memory, here comes the role of data layer, it's role is to store and manage all *persistent* data. It ensures data integrity, security, availability, and efficient retrieval. The common interfaces used to access this layer are SQL or NoSQL, based on the functionalities and needs of the application [63].

The following figure 3.6 summarizes the layered architecture used for the development of CertiFund platform.

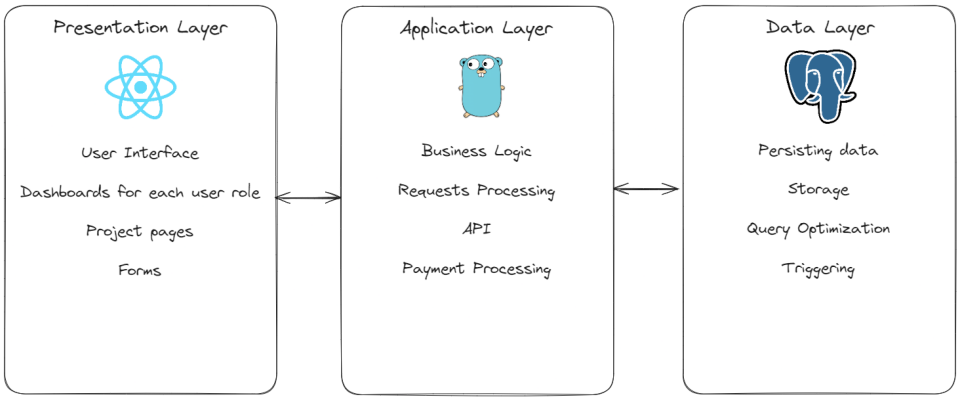


Figure 3.6: CertiFund's layered architecture, with examples of each layer's role.

3.3 Database Design

After we had a general overview of the high-level design and how everything works, we'll move on to a more detailed view of the data layer, starting from right to left. Examining more closely the core component of the layer and the whole application in general, the database.

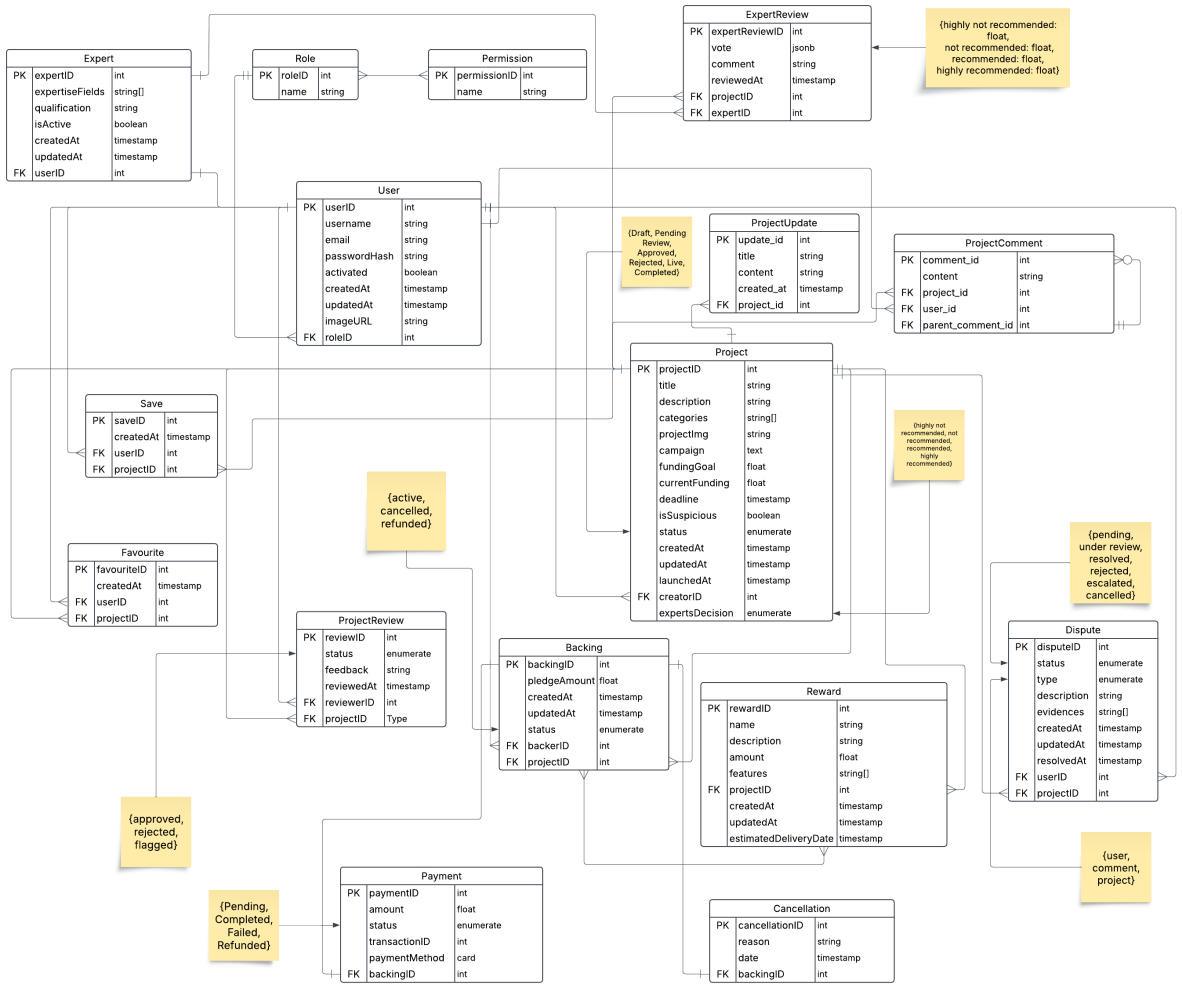


Figure 3.7: ERD of the architecture, using crow's foot notation.

The above figure represents the architecture of the database, built to be as modular and extensible as possible. The chosen approach to design and represent the architecture was the Entity Relationship Diagram or ERD, as it has proven excellence in better representing the structural organization and relationships between objects (or entities, to be precise). Why choose ERD instead of the more popular relational schema? Well, it's behind several motivations like conceptual clarity and the visual representation, which makes it easier for non-programmers to understand, compared to RS (relational schema). Moreover the transition from ER model to the ERD which is the graphical representation of the ER model makes it easier for the developer to grasp better the architecture from the domain of knowledge down to the lower-level of database structure, that's why it's easier to go back and forth between the abstracted ER or the visualized ERD depends on the party to showcase the architecture to [86]. Additionally, ERD is easier to convert into an

actual database than it is the case with RS. And personally, ERD helped a lot to rapidly establish a plan and design for the database, which was an advantage due to the iterative approach that led the process of development (more in section 4.1). ERD could be designed using several notations, namely Arrow notation, Barker's notation, Chen notation, IDEF1X notation, and Crow's foot notation. There is even UML notation that follows a representation of cardinality similar to what is used in diagrams like the class diagram. The difference between all these notations is the way each notation represents entities, attributes, and relationships, which are the main components of ERD. The choice for crow's foot notation was due to it's simplicity and the wider use of it across developers, since been introduced in the 1970s, it's been a great choice, leading the developers to be more clear and precise about the design of entities and how each entity relate to other entity, thus leading to a more detailed representation of the database [57].

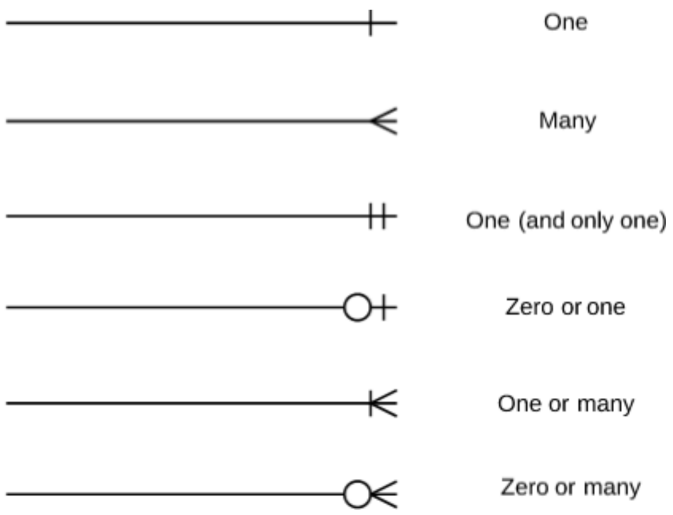


Figure 3.8: Summary of crow's foot notation.

The architecture could be grouped into the following key themes: (1) expert validation, (2) project management, (3) user management, (4) backing, (5) dispute management, and (6) project reviewing:

- **Expert validation**: This is the core feature of the platform, it focuses on incorporating subject(s)-matter experts into the evaluation of projects:
  - **Entities involved:** *Expert*, *ExpertReview*, *User*, *Role*, *Permission*
  - **Purpose:** Experts entity in particular is a special entity separated from the

user entity, why?, because the non-intersection difference is remarkable, given that number of attributes are specific to experts, which doesn't work good if we add them all to User entity (resulting many *nullable* values in the table on the database), so the choice were to separate it and link it to user entity to gain also the basic information shared by all users. Therefore, experts are assigned to evaluate projects based on their qualifications. Their reviews (*ExpertReview*) inform the platform about the decision along with their feedback.

  - **Key Role:** Adds credibility and quality control to the crowdfunding process.

- **Project management:** This theme handles the lifecycle and interactions of crowdfunding projects:
  - **Entities involved:** *Project*, *ProjectUpdate*, *ProjectComment*, *Favourite*, *Save*
  - **Purpose:** Users can create, update, and comment on projects. Projects pass through multiple statuses that represent their stages (e.g., Draft, Pending Review, Approved, etc.)
  - **Key Role:** Organizes project creation, visibility, engagement, and moderation
- **User management:** This theme handles the management of user accounts and role assignment:
  - **Entities involved:** *User*, *Role*, *Permission*
  - **Purpose:** Manages user accounts, provides resources for authentication and authorization, and role-based access control (more in security section).
  - **Key Role:** Controls access, maintains user data integrity, and supports various user roles and responsibilities within the platform
- **Backing:** This theme governs the two types of backers' funding that are donation or reward-based:
  - **Entities involved:** *Backing*, *Reward*, *Payment*, *Cancellation*
  - **Purpose:** Users back projects through funds, and can and can't receive funds, depending on the type of funding they choose. Payments are tracked, cancellations and refunds are handled systematically
  - **Key Role:** Manages financial transactions and user incentives in the platform

- **Dispute management:** This theme supports conflict resolution between platform parties (creators and backers):
  - **Entities involved:** *Dispute*
  - **Purpose:** Users can file disputes on projects, users, or comments with supporting evidence. These passes through an examination by administrators to decide whether to approve and deal with the report, or reject it.
  - **Key Role:** Ensures transparency and trust by offering resolution mechanisms
- **Project reviewing:** This theme provides administrative feedback on projects:
  - **Entities involved:** *ProjectReview*
  - **Purpose:** Allows reviewers or administrators to review projects' adherence to platform rules, ensuring internal assessments and iterative improvements.
  - **Key Role:** Ensures the platform's integrity and prevents malicious or inappropriate content.

It's also worth to mention that management of the database was done through the concept of schema migration or database migration, which is essentially controlled set of changes to the database schema (creating table, adding column, removing index) done in sequential order, where you can *rollback* to previous migrations or move *up* to new migrations [65]. This approach gives a big advantages, given that instead of a big large file where the database scripts are there (suspectable for lost ?) there are an incremental files that represents the evolution and changes made to the database, all described with *down* or *up* indicating backward and forward migrations respectively, and because these are regular files just like any other file, then they are easily trackable alongside the code using a version control system.

3.4 API Design

Rather than going with an end-to-end integration of frontend/backend sides, and for the purpose of future scalability and modularity, the chosen way to expose the backend side is through the choice of an API. API or Application Programming Interface is basically all the exposed information to another side, whether it's a service, library, frontend,

another API, etc. [26]. Now the next step will be to choose the API architecture, which is the patterns and structure that defines and organizes our API, ensuring consistency, scalability, and maintainability, and the decision was to go with REST architecture (or more precisely, architectural style). REST is the most popular and commonly used API architecture, It's easy to use and lightweight, making it a great choice to build APIs that scale up. It's built around a set of six principles, which are:

- A **client-server** architecture, that separates user interface concerns from data storage and processing, and requests are floating between them using the HTTP protocol.
- It's **stateless**, which means all the information necessary to fulfill a given request should be there in order to perform that processing, ensuring no session state is stored on the server between requests.
- It's **cacheable**, which means requests could be marked as cacheable, which allows the client to reuse those requests, which enhances latency (typically static information, static pages, etc.).
- It's a **layered system**, giving the possibility for the architecture to have multiple layers (layered architecture), giving a seamless interaction between the client and the server side, which could consist of several layers.
- **Code-on-demand** means the server side could respond with executable code when requested, thus enhancing the client's functionality.
- **Uniform interface**, that is, the utilization of a uniform, standardized interface, which simplifies the interactions and ensures decoupling, which makes the system flexible and easier to evolve.

REST also leverages and guides the use of several aspects to build the API which are the protocols (specifying to use HTTP/HTTPS), the use of versioning to track the evolution of the API, the leverage of HTTP methods (GET, POST, PATCH/PUT, DELETE), filtering and pagination for requesting efficiency and lower latency, and the endpoints which are the nouns describing the resources to request for [10].

3.5 Security

"Good fences make good neighbors."

- Robert Frost, Mending Wall

Security has been and will always be a crucial and critical aspect to consider when building any system. Any system that lacks security is destined to fail. Security in web applications consists of all the practices to protect the application from attacks, ensuring a smooth and safe functioning, and preventing any possible threats from data theft, leakage, digital vandalism, or any other forms of danger. And considering that saying "it works!" without effort to robustly protect the application won't work and will cause big troubles, security through obscurity just doesn't work. The following sections are the security mechanisms chosen to protect CertiFund platform, ensuring a better experience without issues (as much as possible):

3.5.1 Authentication/Authorization

Authentication

Authentication is the process of verifying and confirming who a user is by verifying given credentials (typically email and password for web applications). Choosing the approach of authentication can be tricky, considering the many options there are with their pros and cons. The considered approach to establish authentication for CertiFund is stateful token authentication. In the stateful token approach, a high-entropy cryptographically-secure string will be generated, called the token. This token will be hashed and stored in the database alongside the user ID and expiry time. The client needs to be given this token in order to be sent back in subsequent requests, which will be checked on the server side to determine the identity of the client sending the request and granting access to them. The biggest advantage of this approach is that the server has complete control over the token, which is very beneficial. Moreover, it provides options to store tokens like in sessions (a way to maintain state information about users' interactions with the application), which adds more flexibility, taking into account that sessions could be stored in the database, in memory, or in distributed caches (e.g., Redis works this way). Additionally, this approach is a simple and robust conceptually [44]. The following figure

describes the process of how stateful authentication works.

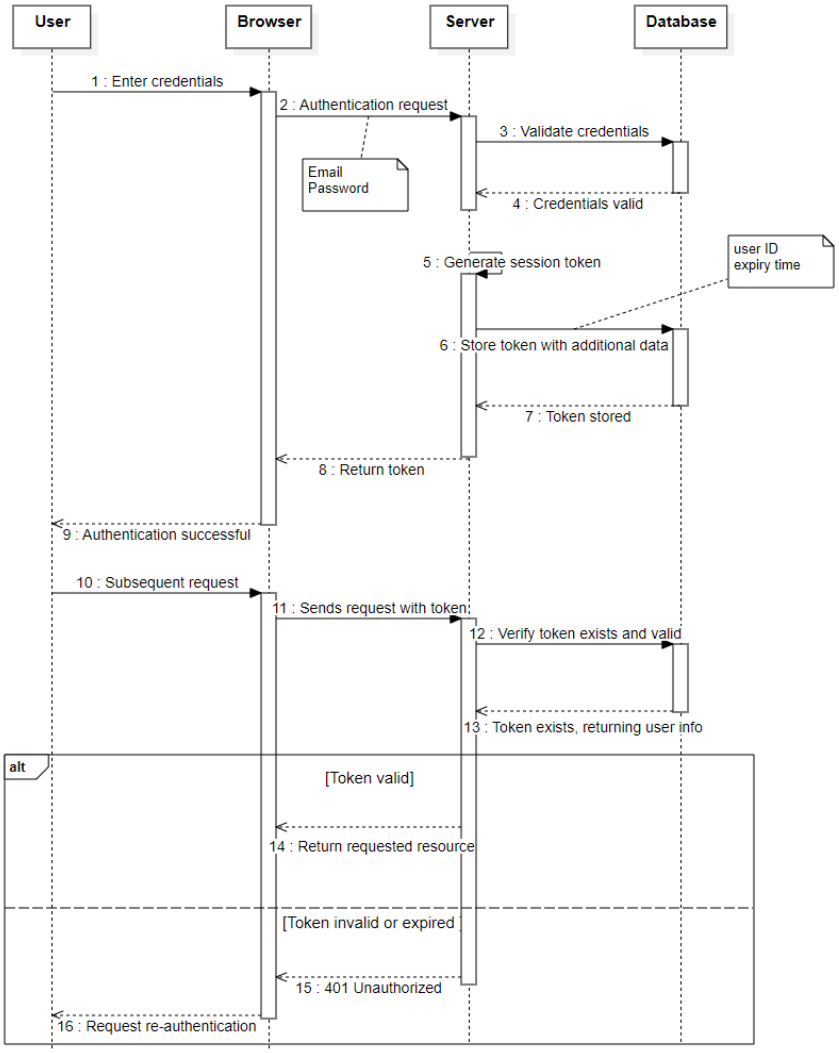


Figure 3.9: Sequence diagram that explains how stateful-token authentication works.

Authorization

Authorization is the process of giving someone the ability to access a resource, therefore restricting certain users from accessing certain resources and minimizing the scope to only those with proper permissions. The approach to establish authorization in CertiFund was Role-Based Access Control or RBAC. In an RBAC system, each user is assigned to a specific role which have a set of specific permissions to perform specific operations. This approach gives a more granular tracking of access for resources, eliminating the assignments of a customized set of permissions to users and moving instead to assigning roles that provide those permissions, which decreases complexity and makes it easier for administrators to assign operations based on roles. Additionally, this provides a higher

protection to sensitive data by enforcing the principle of least privilege, which is the minimum assigned operations given to the specific user to do their role effectively [48]. The following figure summarizes what has been said.

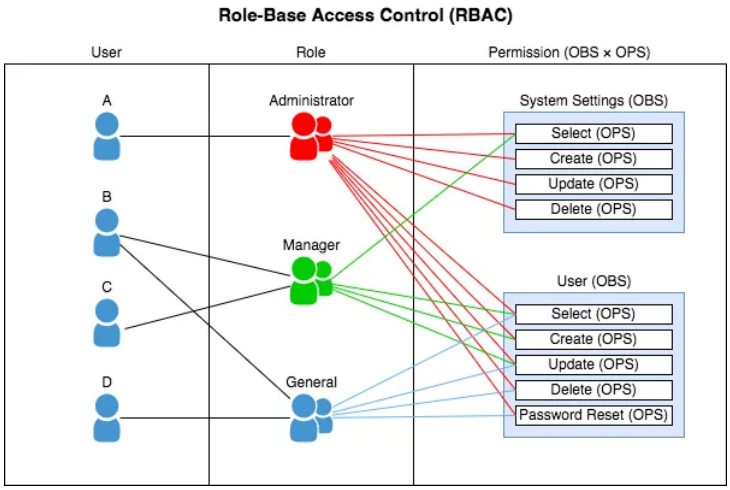


Figure 3.10: Role-Based Access Control conception. Available here

3.5.2 Encryption and Hashing

Encryption has been one of the early-adopted and most widely used mechanisms of security in systems, protecting data and assets from malicious intent. The process of encryption consists of applying an algorithm to certain data and converting it into unreadable, unguessable data. The other mechanisms that works along with encryption is hashing which is the generating of a *hash* value that represents a certain data, and then *rehashing* again for a given data and compare it with the existing *hash* to determine changes or corruption [40]. Considering that *tokenization* will be used widely in the platform (in user activation, authentication, and resets). This requires the unpredictability of the token, making it harder to be guessed or brute-forced. In the light of this, tokens need to be generated by a *cryptographically secure random number generator* (CSPRNG) and have a reasonable entropy or randomness factor to make it impossible to be guessed [56]. Another problem happens when the database itself, which holds all the sensitive data (including tokens and passwords), could be compromised. Which requires following a *fail-safe* design that involves designing around a *pessimistic* mindset that expects failure to happen. Thus, the safe option would be to hash the most sensitive data in the platform, which makes it impossible to know the content of that data (not so impossible,

there are techniques of hash collisions, but that is beyond immediate relevance). This led to choosing two algorithms to hash tokens and passwords, respectively: SHA256, bcrypt. SHA or Secure Hashing Algorithm is a family of cryptographic hash functions, and SHA256 is part of it, and it is one of the most popular hashing algorithms. However, it has some vulnerabilities that make it less preferable nowadays. The choice to hash tokens was considered due to the higher-entropy random strings that form the tokens, compared to lower-entropy ones like passwords, which led to choosing it for reasons of performance compared to slower algorithms like bcrypt. In contrast, Bcrypt is a more secure and reliable hashing algorithm. In addition to standard hashing, it adds a random piece of data, called salt (or technically cost), which enables it to generate a distinct hash that is *nearly* difficult to crack using hash dictionary and brute-force attack techniques [36]. Here is an example of the string "hello world!" hashed with bcrypt (cost factor used here is 12 by default):

```
$2a$12$RV.sfZvDeewb.5t5lkVwl.3ebzAHVlSOuOdKYWY45uxsCmryeevTS
```

3.5.3 CORS

Cross-Origin Resource Sharing or CORS is a security mechanism that indicates the permittable origins (domain, scheme, or port) are allowed to be served by request resources, outside the original domain which got served first (usually localhost). The default behavior for most browsers is the utilization of the "Same-Origin" Policy, which, in short, prevents a (potentially malicious) website or another origin from reading (possibly confidential) information from your website. Say loading a script like a button handler from another sub-domain that the script can only make requests from. This mechanism works good and provides an excellent safeguard, but in many other circumstances we need to *break* a little bit from this policy, this is the point at which CORS becomes relevant. CORS allows to do cross-origin requests between resources of different domains, say you have an API at `api.example.com` and a trusted frontend application running on `www.example.com`, and you want to exchange requests between them. This additional behavior of cross-origin is controlled through preflight requests that exchange a set of HTTP headers, which allows them to continue, referred to as "CORS headers". mainly we'll mention `Access-Control-Allow-Origin` header, which accepts values of the accepted domains to do cross-origin requests with. Using the wildcard character `*` will

allow *any* origin [72].

3.5.4 Rate Limiting

Imagine this scenario: a public API or a web application on the internet. Where it's freely available for any users (clients) to make too many requests in an extremely short period. This will form a heavy load on the system and potentially lead it to fall. The typical solution to such scenarios of too many requests will be rate limiting. Rate limiting is a mechanism that checks how many requests the server has received in the last N seconds, and then when it detects too many requests it sends the error `429 Too Many Requests` to stop receiving requests for some time, preventing what is known as *resource starvation*. There are several types, or specifically implementations of rate limiting; the chosen technique for this platform is token bucket rate limiting. Token bucket is one of the widely used approaches, which regulates the flow of requests using the concept of a token bucket. Each request consumes a token from the bucket, and once the bucket is empty, no more requests are allowed until the bucket is refilled again. This approach could be used in two ways: global, which will be applied to any request, and IP-based, which will be applied for a specific client each time, so that one bad client making too many requests cannot affect the others. Thus, this latter use case was used for this platform due to its flexibility, which allows more concurrency in the system [66] [17]. Figure 3.11 explains visually how rate limiting works.

3.6 Technical stack

The following section represents the technical stack of tools and technologies in order to develop CertiFund platform, and generally the whole thesis, organized by categories:

3.6.1 Frontend

- **Typescript:**[1] Typescript is a modern programming language that builds on JavaScript, adding safety nets. It adds typing, which means catching errors early at runtime. These features made it a robust choice to build the platform with it, to create an application that scales up with fewer bugs possible.

[1] https://www.typescriptlang.org/

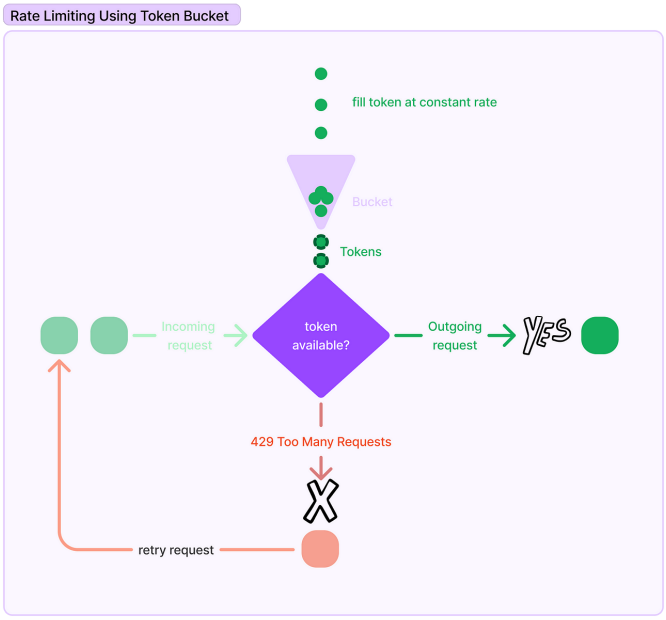


Figure 3.11: Explanation of rate-limiting applying the token bucket algorithm.

- **React:**[1] React is one of the most popular and widely used JavaScript frameworks (although it's nuancedly described as a library) for building user interfaces. Since its introduction by the meta team this it has been the first choice for many web applications to build interactive and intuitive web apps, leveraging the concept of reusable and divided pieces of the application, that is called components.
- **Next.js:**[2] a react framework, charging react-built applications with several features like caching, routing, server-side rendering, and better utilization of server actions (a react concept based on peace of code executable on the server part instead of the client or browser part), pre-rendering and streaming. It's ideal for creating not only UI interfaces but also the backend side, which makes it an end-to-end full-stack framework for building web applications.
- **Tailwind CSS:**[3] famous CSS library for styling web apps leveraging the feature of CSS classes, it's currently considered a standard to style web apps instead of regular CSS files, due to it's rich features of styling and the developer experience it provided (especially in terms of rapid developement).

[1]https://react.dev/
[2]https://nextjs.org/docs
[3]https://tailwindcss.com/

- **ShadcnUI:**[1] it's a collection of customizable and accessible React components built with Radix UI and tailwindCSS, it's designed to provide easy-to-copy, reusable components to quickly build web applications with minimal clutter.

3.6.2 Backend

- **Golang (Go):**[2] an open-source programming language, developed by Google. optimized for simplicity, performance, and concurrency, widely used to build scalable and efficient systems and servers, adapted by the big tech companies and used in the core development of big projects like Docker and Kubernetes.
- **Echo (Golang Framework):**[3] Echo is an extensible, minimal, highly performant web framework for Go, offering a wide range of features like a fast HTTP router, middleware, and data binding for building robust RESTful APIs

3.6.3 Database and Backend services

- **PostgreSQL:**[4] A Swiss-army knife open-source object-relational database, known for its reliability, and advanced features like custom types, JSONB support (which allows it to be used as a NoSQL database !), it also supports PL/SQL (called here PL/pgSQL). These features and others made it suitable for complex data workflows. It's also worth mentioning that PostgreSQL powers the majority of BaaS or backend as a service (Supabase, Pocketbase, etc.) due to its robustness and flexibility.
- **Supabase:**[5] an open-source backend-as-a-service (BaaS) platform built on PostgreSQL, providing several features of storage, authentication, real-time APIs, and edge functions. It was used for the storage of files of CertiFund platform.
- **Cloudinary:**[6] Cloud-based media management platform that enables uploading and storing images and videos for mobile and web applications. with APIs for dynamic manipulation.

[1] https://ui.shadcn.com/
[2] https://go.dev/
[3] https://echo.labstack.com/
[4] https://www.postgresql.org/
[5] https://supabase.com/
[6] https://cloudinary.com/

3.6.4 Payment Processing

- **Stripe:**[1] API-driven online payment processing platform that simplifies the integration of secure payment systems, supporting transactions, monetization, and testing mode for payment testing without billing. Suitable for web and mobile applications.

3.6.5 CI/CD

- **Docker:**[2] containerization platform that packages applications and their dependencies into portable containers, ensuring consistent execution across various environments (no more it works on my machine !).
- **Git:**[3] git is a distributed version control system built by Linus Torvalds. It tracks system changes, enables branching, merging, and collaboration across distributed development environments.
- **GitHub:**[4] A cloud-based platform made for hosting git repositories, providing collaboration tools, issue tracking, better pull-requests management, workflows, and also GitHub pages for hosting static pages.

3.6.6 Design, modeling and UI tools

- **Figma:**[5] A cloud-based design and prototyping software that enables collaborative creation of user interfaces, wireframes, and design systems, with real-time editing and integration capabilities.
- **StarUML:**[6] Software modeling tool supporting UML and other diagramming standards, with capabilities for code generation and cross-platform compatibility for system design.
- **Exalicdraw:**[7] An open source, collaborative diagramming platform with a hand-

[1] https://stripe.com/
[2] https://www.docker.com/
[3] https://git-scm.com/
[4] https://github.com/
[5] https://www.figma.com/
[6] https://staruml.io/
[7] https://excalidraw.com/

drawn aesthetic, established for creating and sharing visual representations of ideas and workflows, with great support of mermaid syntax for text-to-diagram approach.

- **tinyMCE:**[1] A customizable, open-source rich text editor for web applications, providing WYSIWYG functionality to create and edit formatted content, with plugins for advanced features like image uploads and spell-checking.

3.6.7 Other tools

- **Visual Studio Code:**[2] A free, open-source code editor developed by Microsoft, supporting multiple programming languages, with extensive customization through extensions and integrated AI features.
- **Postman:**[3] an API development and test platform that provides a variety of tools to design, document, and collaborate on building APIs, supporting workflows from development to monitoring.
- **Makefile:**[4] A build automation tool that uses scripts to define dependencies and compilation tasks, automating the process of building, testing, and running projects or other context-related scripts that need an automated execution.
- **Obsidian:**[5] A markdown-based note-taking and knowledge management application is extensible using extensions. It provides a variety of features to collect, organize, write, and build on markdown features.
- **Notion:**[6] An all-in-one productivity platform that combines note-taking, project management, and database functionalities, enabling collaborative organization of tasks, documents, and knowledge bases.
- **Overleaf:**[7] Online LaTeX editor for collaborative document creation, particularly suited for academic and technical writing, with real-time editing and version control features.

[1] https://www.tiny.cloud/
[2] https://code.visualstudio.com/
[3] https://www.postman.com/
[4] https://makefiletutorial.com/
[5] https://obsidian.md/
[6] https://www.overleaf.com/
[7] https://www.notion.so/

3.7 Conclusion

In this chapter, we demonstrated the “theoretical side of implementation”, outlining the conceptual foundation that powers CertiFund platform, beginning with requirements exploration to determine the core features this platform provides, along with a detailed explanation of the system’s interactions with various actors. The higher-level design gives a higher overview of what the system is made of and organized, database design with well-defined entities and relationships, supporting the platform’s core functionalities while maintaining data integrity and extensibility. The security strategies were selected to protect the platform from malicious intent. And finally, the technical choices of tools and technologies to actually build and maintain the system. Together, these elements form a robust blueprint for the platform, which helps clarify the next steps in the process of implementation. In chapter 4, we will build upon this foundation, detailing the technical implementation, development processes, and deployment strategy to bring the system to life, in what we call the “practical side of implementation”.

Chapter 4

Implementation and Deployment

The previous chapter provided a comprehensive conceptual framework for CertiFund platform. It outlined the higher-level technical decision to build a robust system. In this chapter, we shift focus to the practical realization of this design, detailing the technical implementation and deployment of the system. This chapter covers the development methodology adopted to guide the process, the project structure organizing the codebase, and the implementation of the frontend/backend components. It also highlights the implementation of the double-score voting algorithm, along with a description of the deployment strategy to make the platform operational. By addressing these aspects, this chapter demonstrates how the conceptual design is transformed into a functional system, advancing the research objectives of enhancing expert validation in crowdfunding.

4.1 Development Methodology

The adopted methodology to build CertiFund was the iterative approach, which means, in essence, a repetition of phases that yields results successively, which brings you closer to the desired goal. This approach was ideal to build this platform, given that it gives the opportunity to learn, to adapt to change, to react quickly, and to go in cycles of continuous improvement. It also provides a great power given that many of the details of the software could be vague in the beginning which requires an advancement that actually benefits the previous phase and so on, for example during the development of CertiFund the requirements identified weren't complete and clear, but the approach

was to identify good-enough amount of requirements and conceptualize the software, which provided insights for *hidden* requirements, and then moving to implementation the missing part of both the previous phases (requirements and conception) were identified and during test another missing parts starts to showing and then goes the first iteration, you fix what identified before and find new missing things and so on till you reach an acceptable result. The iterative approach is about discovery and learning, it allows you to narrow your focus on smaller parts and advance towards the big picture, thus building on modularity. Moreover working iteratively doesn't bound you, it doesn't apply to work on small features but always deliver them *complete* which could be unpredictable especially during early phases, instead it gives you choice to iterate and work on the product and enhance it by time based on good feedbacks from customers (provided by the supervisor). In general, the iterative approach applies way beyond software engineering, and could be applied and enhance the way to build products [27].

4.2 Project Structure

Another important step before actually writing software is the step of setting up the project structure, which will organize the codebase and make it easier to handle and work with, something generally named "separation of concerns". The structure architecture used to establish CertiFund's codebase is a monorepo architecture, which is the organization of the project in one repository (Git repository in our case), where each service or application is separated within this repository. This enables the overall project to be modular, and the applications inside of it (frontend and backend in our case) to be decoupled and interacted with clear defined interface between them (API), and enables also these applications to be separated in deployment, deploying each application in different container (more in deployment) [42].

4.3 Frontend

Following the principles of human-centered design (HCD) established by Don Norman, the UI/UX of this platform is designed and implemented to put the users' needs, behavior, and capabilities first and then design the platform to fulfill those needs. Following a simple

rule of creating the platform as if we were the users of it. This leads to a moderated understanding of psychology and technology [58]. The selected color palette (one of the first signals which the user notices) to style the platform has been a palette consisted from variants of blue, following the branch of color psychology which dictates that the blue color is the suitable choice for signaling trust and confidence towards the product, which is highly needed for a crowdfunding platform [15].

Figure 4.1: CertiFund's logo along with the used color palette.

The other intuitive aspect that is first noticeable is font choice, the font pairing selected for CertiFund is Inter and Montserrat, which is a powerful pairing that gives the product a distinctive look, which is required in a competitive web of applications [20]. Given that the platform has many pages and interfaces implemented and running, this section will narrow focus on only five key themes to demonstrate their UIs which are: (1) project creation, (2) the reviewing process, (3) expert validation, (4) project backing and (5) administrative management.

4.3.1 Project Creation

We assume that the creator has already signed up to the platform is activated his account, and also entered all his necessary information (phone number, bio, social media links, etc.). Now, here is the first page he will encounter, which is the landing page:

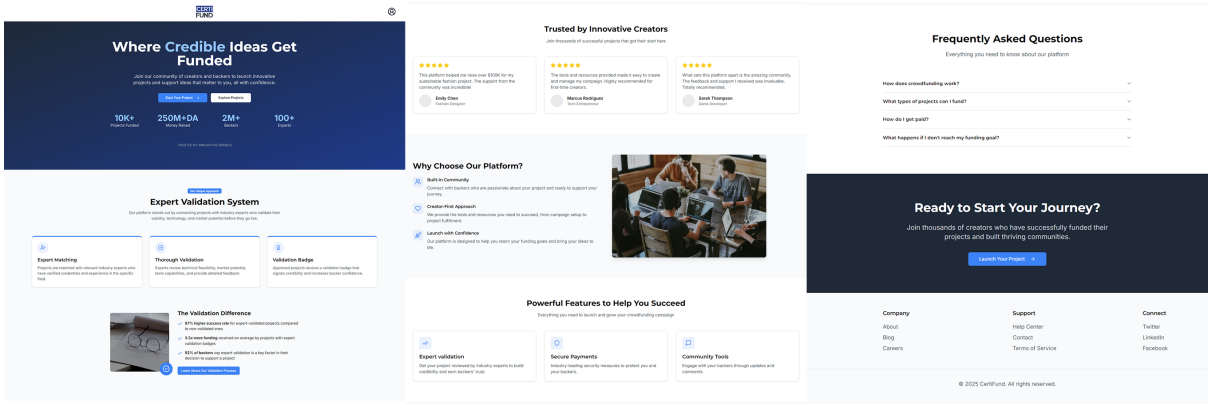


Figure 4.2: Portions from the landing page of CertiFund platform.

After that the creator naturally will click on `Start your project` or `Launch your project` buttons to be redirected to the project creation page, where they he will be demanded to fill the necessary information to start the project as a draft, leaving the complete detailed information for another page which is project overview page:

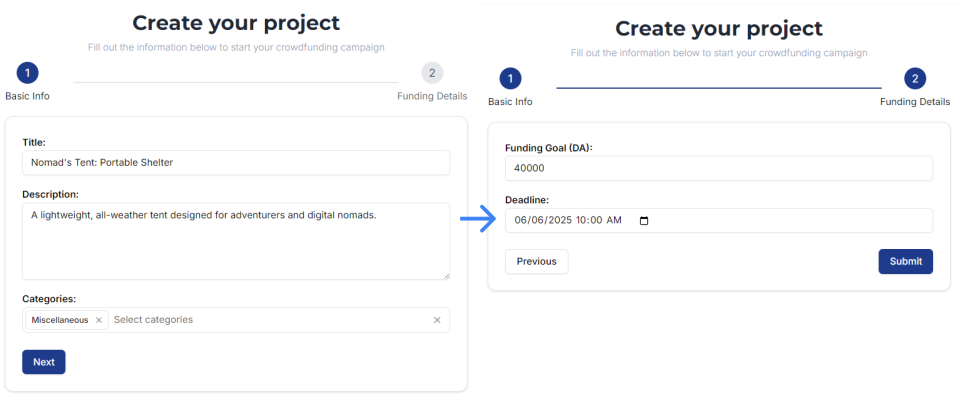


Figure 4.3: Project creation's forms that enable the creator to enter their project's basic information.

Next, after submitting, the system will validate the provided data to check for any invalid data, if it passes the check, the user will be redirected to project overview page, displaying a time-line where it will hold all the changes and updates for the project, starting by filling the rest of the needed information in order to *completely* submit his project, covering things like project image that will be used as a banner for the project, project story, and rewards if available. Moving to tracking creators' updates, which are necessary to keep the backers updated about the project status, and it will also show the project statuses over time (accepted, rejected, under review, etc.). The important part

that it's UI needs to be included here is the project story, which represents a marketing article for the project, detailed the description of the project, providing insights about how the projects does work, how the funded money will be spent, and it uses also medias like images, videos, links to convey the backers even more. For that, the followed approach is to use a WYSIWYG[1] rich-text editor to establish this part, which was TinyMCE (see section 3.6.6 for more). Here is the story part of the page:

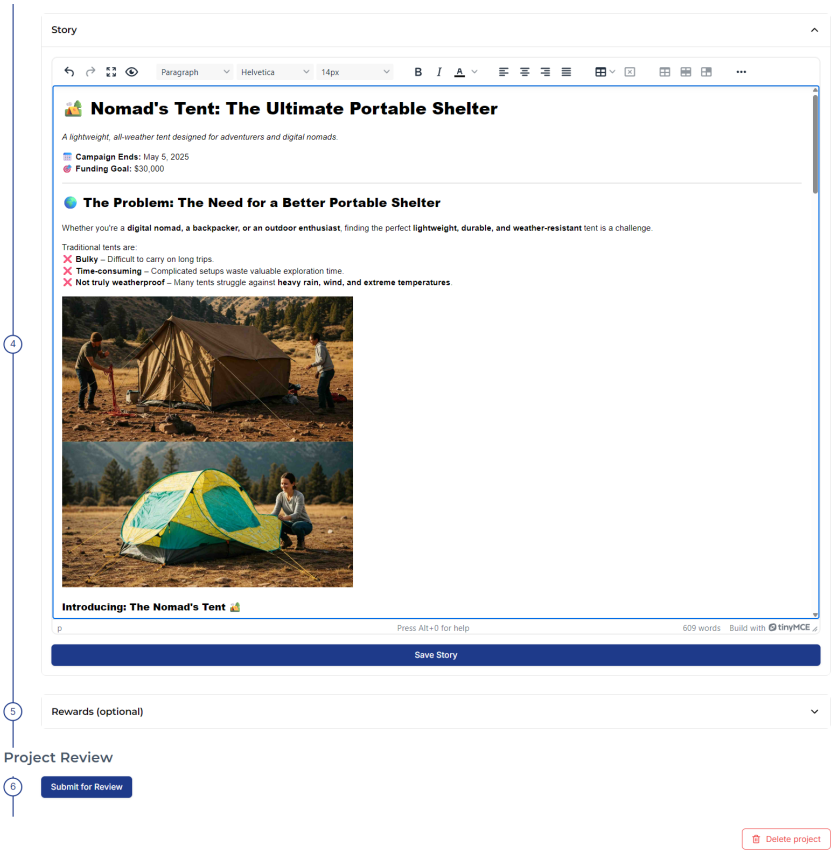


Figure 4.4: Project story's part in the project overview page.

When the user completes all the information and also accept the rules, he can submit the project for reviewing, after reviewing (more in the next section) the creator can launch the project as he wish, and after the launch he we be able only to update the backers (or any user in general) about the progress of the project:

[1]What You See Is What You Get

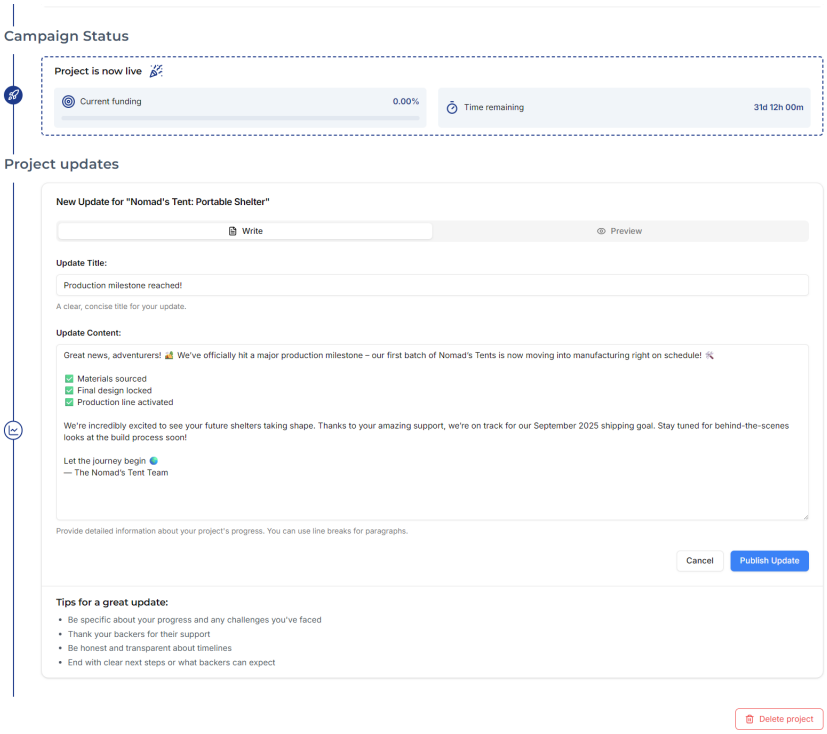


Figure 4.5: An example of a project update title: "Production milestone reached!"

4.3.2 Project Review

The process of reviewing starts when the creator submits their project, then the reviewer will find the project on their dashboard in the tab of pending reviews:

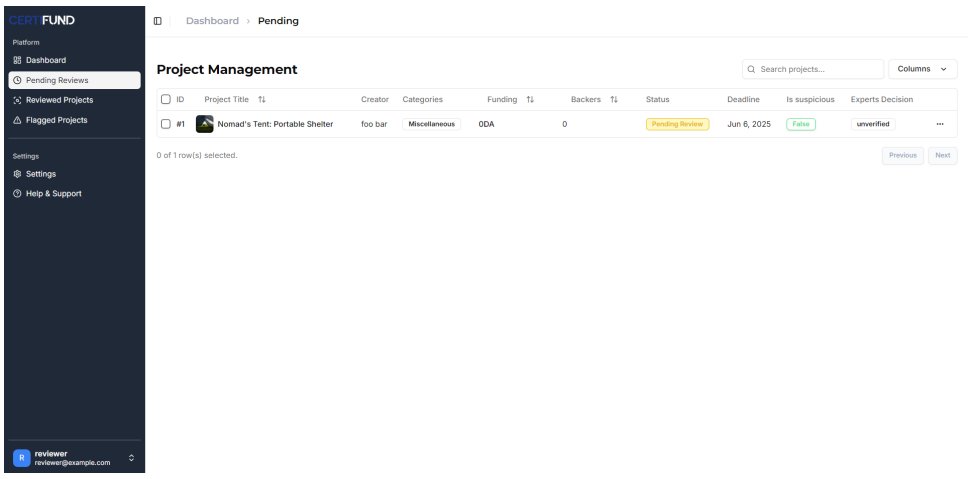


Figure 4.6: Pending Reviews page in the reviewer's dashboard.

After a brief look at the project, he can click on the project to get more details about it:

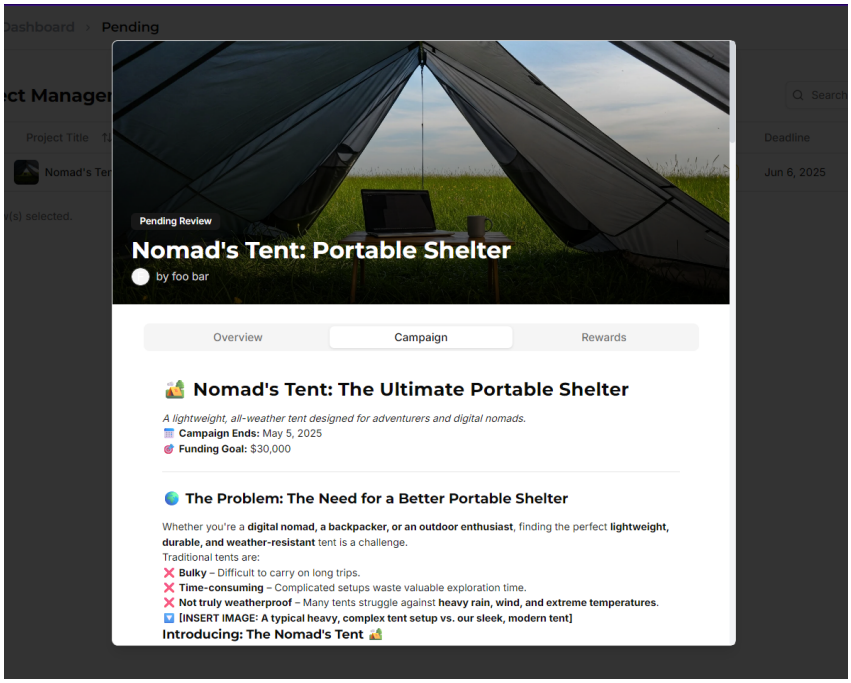


Figure 4.7: Project's details, contains an overview of details, the campaign (story), and the rewards tier.

Following this, when the reviewer has a well-defined knowledge about the project, it's time to decide about it, by clicking on the three dots in the extreme right of the row that has the project on the table, then selecting review the project, displaying:

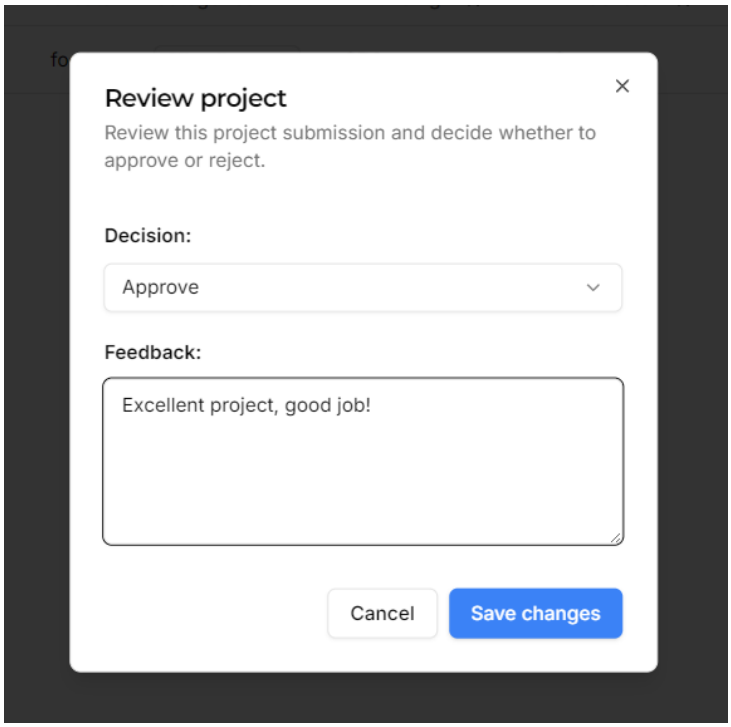


Figure 4.8: Review project form, the example here shows an approved decision about the project.

Thereafter, the creator will be updated by the decision in the project overview page, and the reviewer will find the projects they reviewed in the Reviewed Projects page in the dashboard:

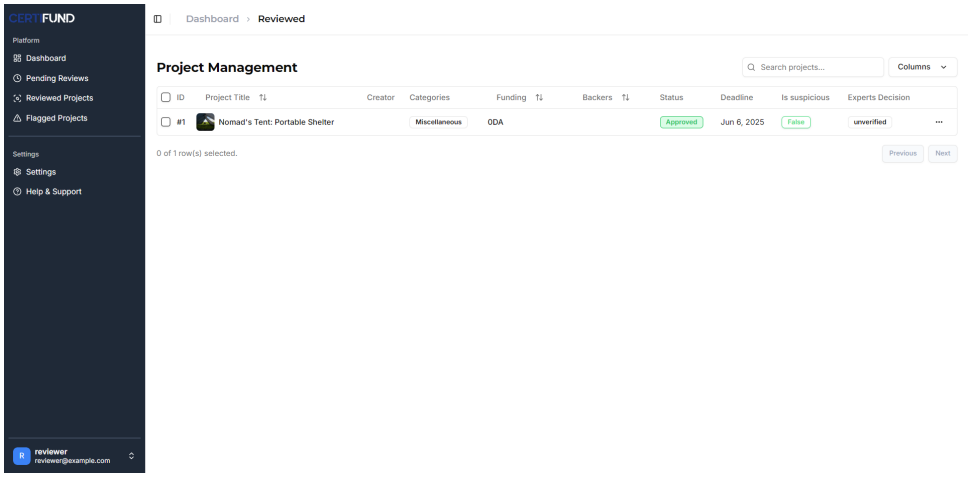


Figure 4.9: Reviewed projects page, displaying the reviewed projects by the current reviewer.

4.3.3 Backing

The backing process is another process on the platform, now concerns backers who are the other part of the interactions. After the registration of the backer will naturally want to explore the landing page and then explore projects on the platform, through the discover projects page:

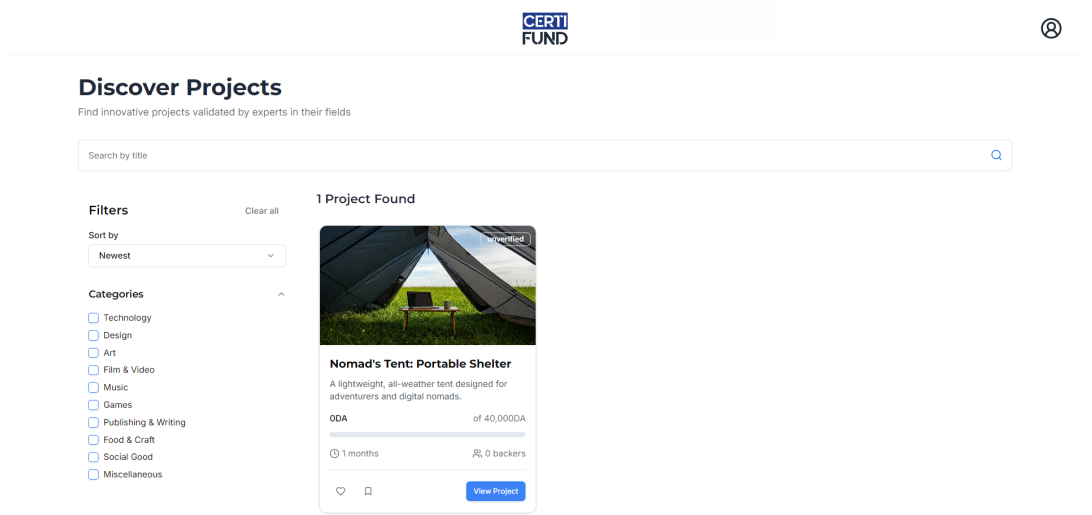


Figure 4.10: Discover projects page, showing the project created before.

This page provides all the means to search and explore projects, filters, or search by query. After searching for a project, a certain project will attract his attention, so he wants to view more, leading to the project details page:

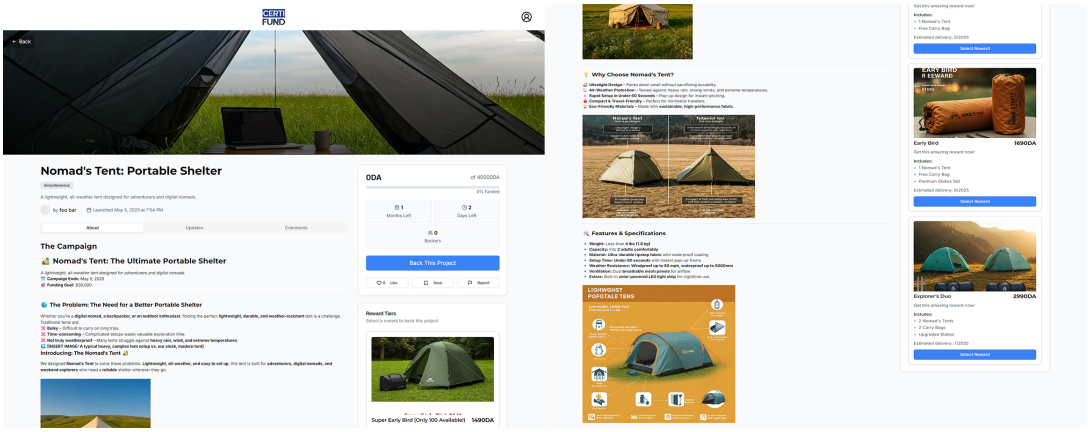


Figure 4.11: Project details page for the nomad's tent project.

After a reasonable examination of the project, the backer decides to fund this project (which we hope the most for all projects!). At this point, there are two options the backer needs to choose from: either back without selecting any reward, which means a donation, or fund in exchange for some selected rewards, depending on the way of conveying from the team behind the project and the nature of the project itself. Let's say this backer gets the first and second rewards, so he will fund them by selecting the two. After that, this form will be displayed:

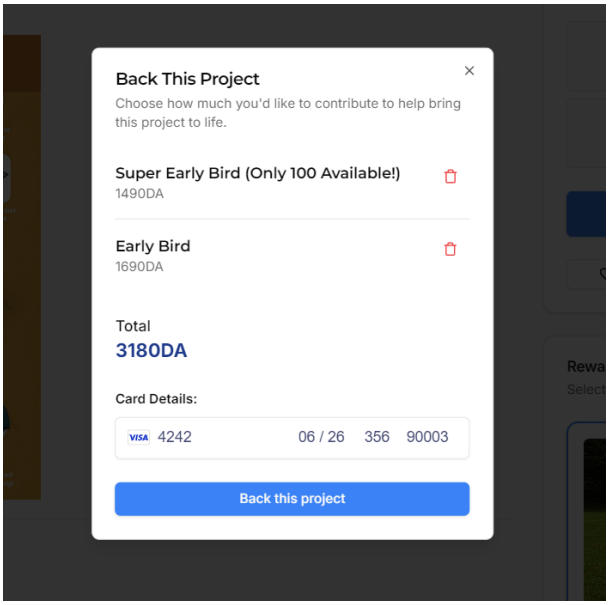


Figure 4.12: Backing form that shows the selected rewards along with payment details (fake details).

This form displays what are the selected rewards by the backer with their total amount, along with card details where the backer is asked to enter their credit card details. After that, he clicks `Back this project` button and the funding is successfully done, where the platform will display the following UI to convey that the funds got added and the backer is provided with the option of refund in case he reversed his decision (allowed only when the project is live). Additionally, he will receive a payment receipt via email confirming his funding (more on that in the next section):

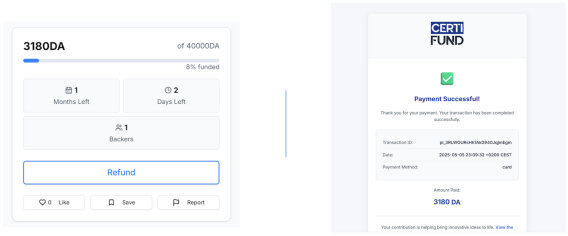


Figure 4.13: Consequences after a successful backing, showing the updated UI (left) and the payment receipt (right).

4.3.4 Expert validation

The process of expert validation starts once the project is approved by the reviewers, and lasts 2 days, giving experts enough time to evaluate projects without much clutter, and also so that it won't take long to provide the decision to the backers. The administrators of the platform are responsible for the addition of experts, filling in their needed information, and files that prove their experience:

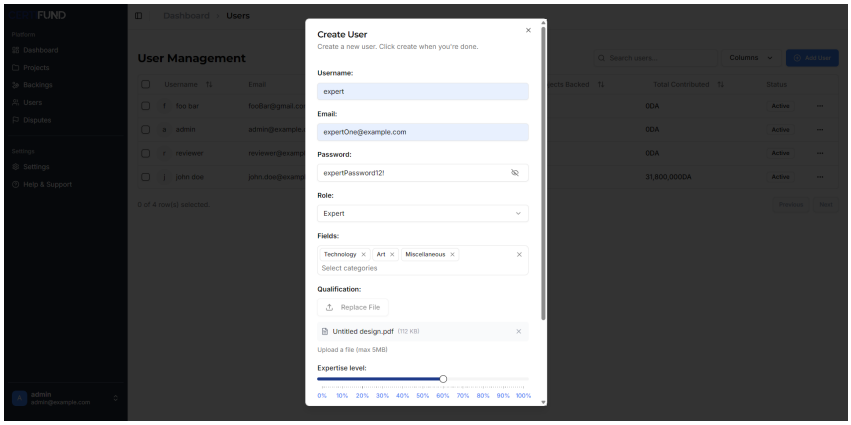


Figure 4.14: The process of adding a new expert to the platform (notice the expertise level).

After that, when the expert is given his credentials, he can sign in to the platform and access his dashboard, where he will be provided with the projects to be evaluated by him, assigned to him by his fields of expertise:

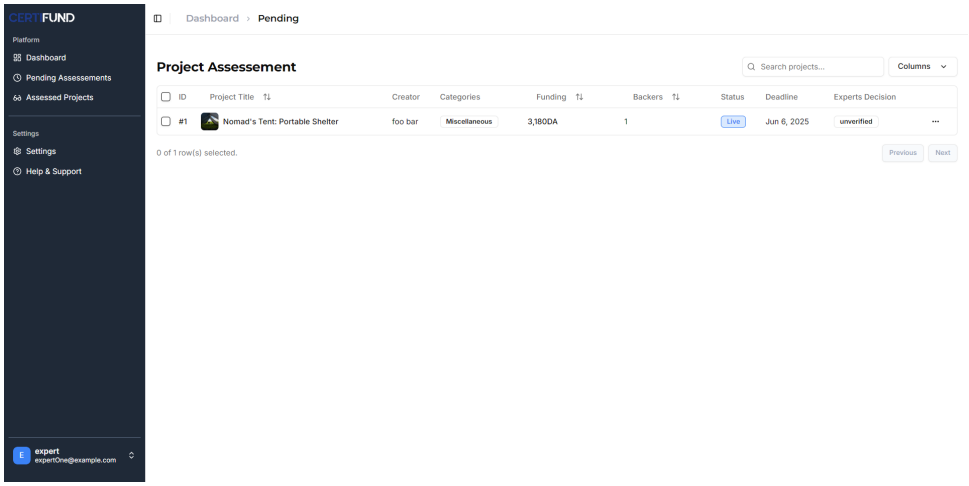


Figure 4.15: Pending assessments page in the experts dashboard.

After an examination of the project details (same as the reviewer, but with different perceptions), and possible contacts with the creators. The experts express their opinion about the project by clicking the three dots on the extreme right of the row that has the project on the table, then selecting Assess project, displaying:

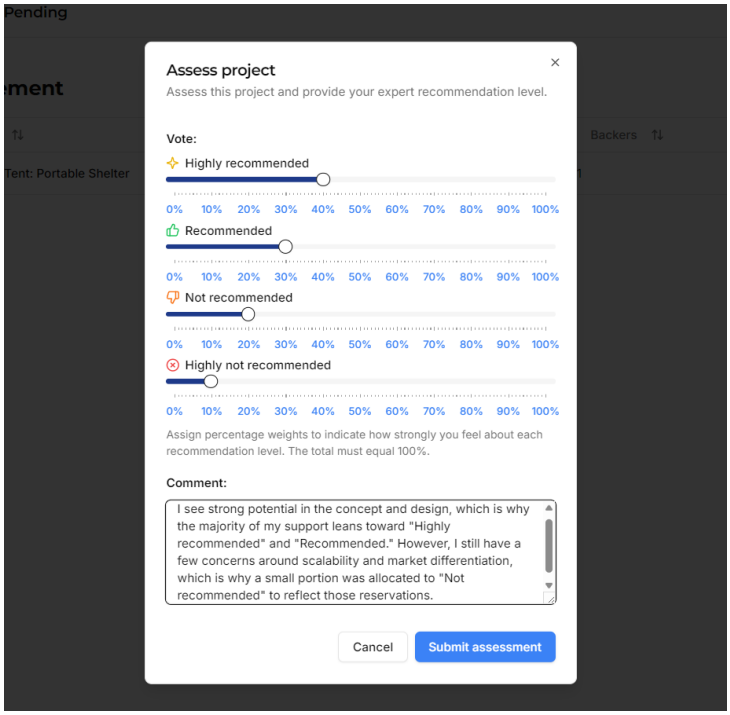


Figure 4.16: Assess project form where the experts can vote and express their opinion.

Here, as we explain in the double-score voting algorithm (see section 2.5), the expert will distribute the percentages of how likely that recommendation level represents his opinion about the project. After submitting the evaluation, the evaluated (assessed) project will be shown in the assessed projects for that expert:

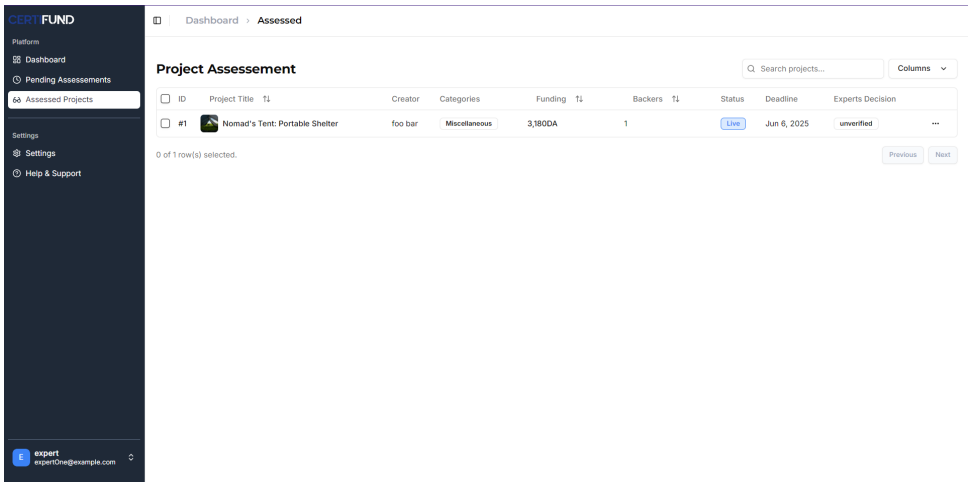


Figure 4.17: Assessed projects page that displays already-evaluated projects from that expert.

After 2 days, all the votes cast on the target project will be collected and calculated applying double-score voting, and the result will be shown for the backers:

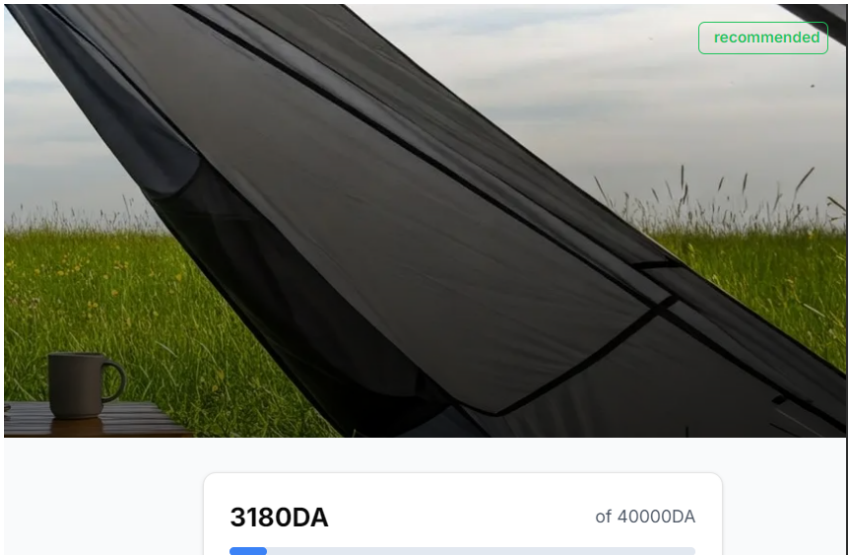


Figure 4.18: Project's experts decision shown in the top right corner of the project details page.

4.3.5 Administrative management

Once the administrator signs in to his account, he will be directed to his dashboard, where he will get an overview of monitoring statistics that help him understand the platform better and make decisions about it. Which are deemed KPIs or Key Performance Indicators:

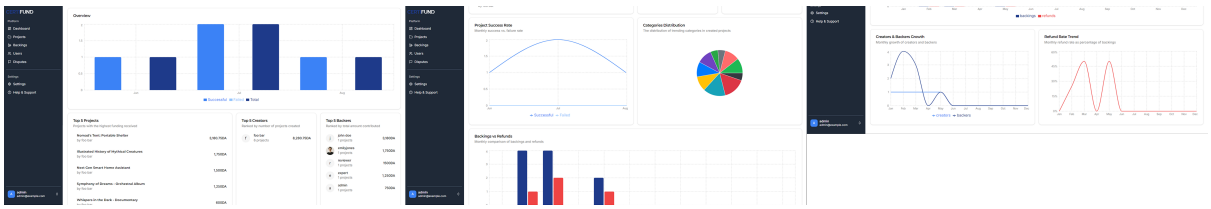


Figure 4.19: Monitoring statistics in admin's dashboard.

From his dashboard, he is responsible for projects management (checking current projects and also review projects tagged as suspicious by the reviewer), backings management (checking the current backings for trackability), users management (adding new users, removing, editing roles ...etc.,) and disputes management (resolving disputes between users), here are few screenshots from the dashboard:

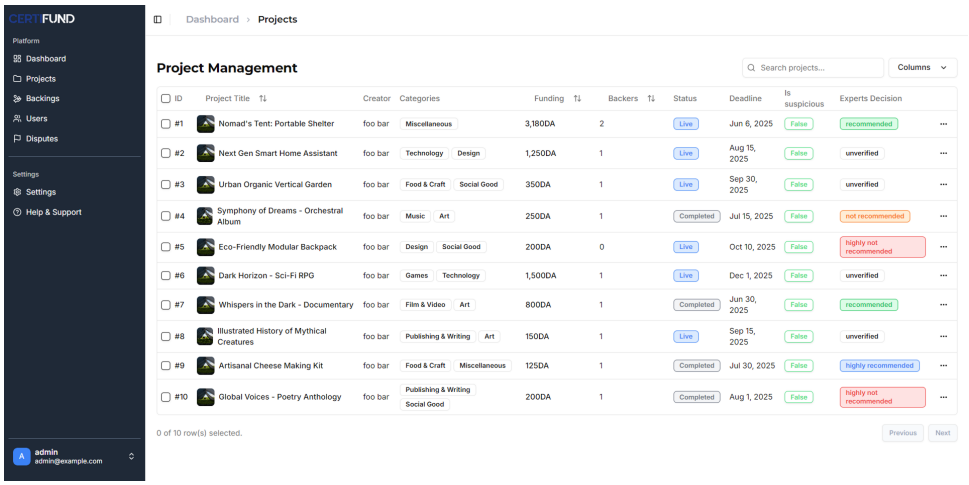


Figure 4.20: Projects management page in the admin's dashboard.

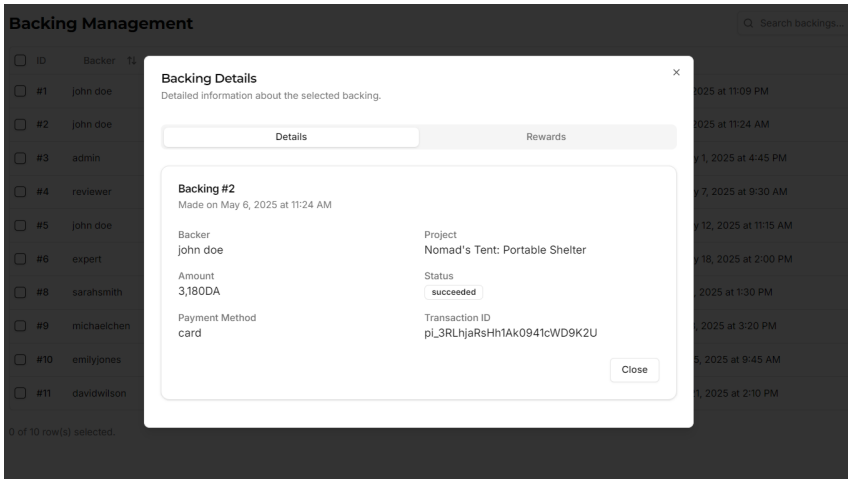


Figure 4.21: Examining backing details for the nomad's tent project.

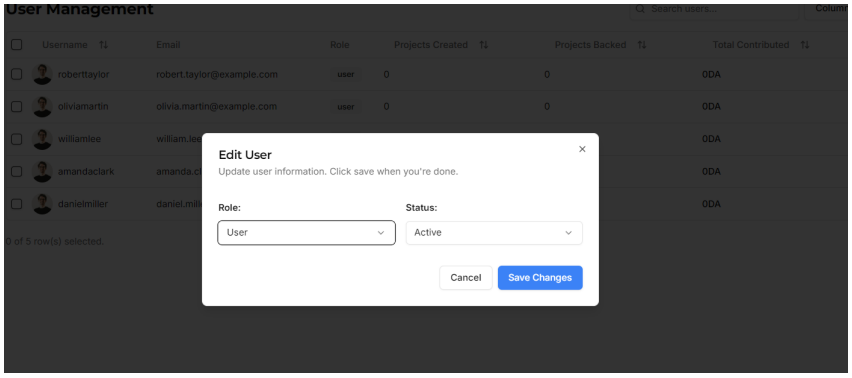


Figure 4.22: Change user's role and status form in users management page, the selected user here is the last added one.

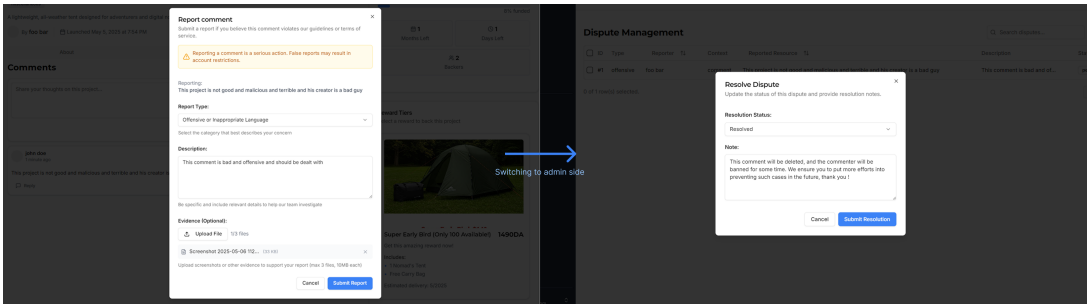


Figure 4.23: Reporting a certain content (comment in this case), and resolving it by the administrator.

4.4 Backend

As we explained before, the backend was built to be a RESTful API for more flexibility and modularity. Given that the API consists of more than 80 endpoints, varying to

handle the aspect all aspects of the platform, we will focus only on explaining one of them, which represents a missing part considering what we explained so far, that is, how the payment processing is handled? When the backer confirms his funding and clicks on submit (Figure 4.12), behind the scenes, a process starts called payment intent, which represents an *intent* of payment committed by the backer for that specific project. Technically, this is a request for this endpoint on the backend side:

```
authGroup.POST("/backing/backIntent/:id", app.
    createPaymentIntentHandler, app.RequirePermission("backing:create"),
     app.VerifyProjectNonOwnership())
```

Listing 4.1: Creating payment intent endpoint.

where:

- **authGroup:** group endpoints by mutual mechanisms (in this case, a middleware to check for authentication).
- **createPaymentIntentHandler:** handler of the endpoint (function that accepts the request coming in that endpoint, does some processing, and returns a response).
- **RequirePermission():** middleware to check for the permission of `backing:create` (see 3.5.1).
- **VerifyProjectNonOwnership():** middleware to prevent the owner of the project (creator) from backing his own project

Now, let's take a look at `createPaymentIntentHandler` to see in depth how the payment intent is created:

```
func (app *application) createPaymentIntentHandler(c echo.Context)
    error {
  // Extract and validate the project ID from the request parameters
  projectId, err := app.readIDParam(c)
  if err != nil {
    return echo.NewHTTPError(http.StatusNotFound, err.Error())
  }
  // Retrieve the project from the database using the project ID
  project, err := app.models.Projects.Get(projectId)
  if err != nil {
    switch {
    case errors.Is(err, data.ErrNoRecordFound):
      return echo.NewHTTPError(http.StatusNotFound, "Project not found"
    )
    default:
      return err
    }
```

```
  }
  // Check if the project's funding deadline has passed
  if time.Now().After(project.Deadline) {
    return echo.NewHTTPError(http.StatusNotFound, "Project funding
     duration is closed")
  }
  // Define and bind the request payload to extract the backing amount
  var input struct {
    Amount float64 `json:"amount"`
  }
  if err := c.Bind(&input); err != nil {
    return echo.NewHTTPError(http.StatusBadRequest, "Error while
     processing data")
  }
  // Get the authenticated user (backer) from the context
  backer := c.Get("user").(*data.User)
  // Validate the amount using a custom validator
  v := validator.New()
  if data.ValidateAmount(v, input.Amount); !v.Valid() {
    return echo.NewHTTPError(http.StatusUnprocessableEntity, v.Errors)
  }
  // Prepare Stripe payment intent parameters
  // Convert amount to cents/smallest currency unit for Stripe
  params := &stripe.PaymentIntentParams{
    Amount:   stripe.Int64(int64(input.Amount)),
    Currency: stripe.String(string(stripe.CurrencyDZD)), // Using
     Algerian Dinar
  }
  // Convert IDs to strings for Stripe metadata
  projectID := strconv.Itoa(projectId)
  backerID := strconv.Itoa(backer.ID)
  // Add metadata to the payment intent for tracking purposes
  params.AddMetadata("project_id", projectID)
  params.AddMetadata("backer_id", backerID)
  // Create the payment intent with Stripe
  pi, err := paymentintent.New(params)
  if err != nil {
    return err
  }
  // Return a successful response with the client secret for the
   frontend
  return c.JSON(http.StatusCreated, envelope{
    "message":       "Backing intent is done successfully",
    "client_secret": pi.ClientSecret,
  })
}
```

Listing 4.2: Create payment intent handler.

What this handler does is the following:

1. Extracts and validates the project ID from the request's parameters
2. Retrieves the project details and checks if it exists
3. Verifies that the project's funding deadline hasn't passed
4. Binds and validates the payment amount from the request body
5. Creates a stripe payment intent with the specified amount and currency (using stripe package in golang)
6. Attaches project and backer metadata (IDs) to the payment intent
7. Returns, if successful, the response in JSON format that is the client secret.

This client secret in particular is what we need from the intent creation, on the frontend side, this will be passed to a function that belongs to the stripe.js library to confirm the payment (funding) of the backer. If that process went successfully, the frontend side will request to record this backing as successfully done, exploiting this endpoint:

```
authGroup.POST("/backing/backProject/:id", app.recordBackingHandler,
    app.RequirePermission("backing:create"), app.
    VerifyProjectNonOwnership())
```

Listing 4.3: Record backing endpoint.

Behind the scenes `recordBackingHandler` will look like this:

```
func (app *application) recordBackingHandler(c echo.Context) error {
    // Extract and validate the project ID from the request parameters
    projectId, err := app.readIDParam(c)
    if err != nil {
     return echo.NewHTTPError(http.StatusNotFound, err.Error())
    }
    // Define and bind the request payload structure
    var input struct {
     PaymentIntentID string `json:"payment_intent_id"`
     PaymentMethod   string `json:"payment_method"`
     Rewards         []int  `json:"rewards"` // IDs of rewards selected
    by the backer
    }
    if err := c.Bind(&input); err != nil {
     return echo.NewHTTPError(http.StatusBadRequest, "Error while
    processing data")
    }
    // Verify the payment intent with Stripe to confirm it exists and
    retrieve its details
```

```
17    pi, err := paymentintent.Get(input.PaymentIntentID, nil)
18    if err != nil {
19     return err
20    }
21    // Get the authenticated user (backer) from the context
22    backer := c.Get("user").(*data.User)
23    // Create backing record to link the backer to the project
24    backing := data.Backing{
25     BackerID:  backer.ID,
26     ProjectID: projectId,
27    }
28    // Create payment record with details from the Stripe payment intent
29    payment := data.Payment{
30     Amount:        float64(pi.Amount),
31     Status:        string(pi.Status),
32     TransactionID: input.PaymentIntentID,
33     PaymentMethod: input.PaymentMethod,
34    }
35    // Insert both backing and payment records to the database
36    err = app.models.Backing.Insert(&backing, &payment)
37    if err != nil {
38     return err
39    }
40    // Retrieve the project to update its funding amount
41    project, err := app.models.Projects.Get(projectId)
42    if err != nil {
43     switch {
44     case errors.Is(err, data.ErrNoRecordFound):
45       return echo.NewHTTPError(http.StatusNotFound, "Project not found"
      )
46     default:
47       return err
48     }
49    }
50    // Verify project funding deadline hasn't passed
51    if time.Now().After(project.Deadline) {
52     return echo.NewHTTPError(http.StatusNotFound, "Project funding
      duration is closed")
53    }
54    // Update the project's current funding
55    // Divide by 100 to convert from cents (Stripe's format) to actual
      currency units
56    project.CurrentFunding = project.CurrentFunding + (payment.Amount /
      100)
57    // Save the updated project funding to database
58    err = app.models.Projects.Update(project)
59    if err != nil {
60     switch {
```

```
61      case errors.Is(err, data.ErrEditConflict):
62        return echo.NewHTTPError(http.StatusConflict, data.
    ErrEditConflict.Error())
63      default:
64        return err
65      }
66    }
67    // Process any rewards selected by the backer
68    if input.Rewards != nil {
69      for i, rewardID := range input.Rewards {
70        // Verify each reward exists
71        _, err := app.models.Rewards.Get(rewardID)
72        if err != nil {
73          switch {
74          case errors.Is(err, data.ErrNoRecordFound):
75            return echo.NewHTTPError(http.StatusNotFound, fmt.Sprintf("
    Reward %d not found", i))
76          default:
77            return err
78          }
79        }
80        // Link the reward to this backing
81        err = app.models.Rewards.InsertBackingReward(backing.BackingID,
    rewardID)
82        if err != nil {
83          return err
84        }
85      }
86    }
87    // Send receipt email in background to not block the response
88    app.background(func() {
89      data := map[string]interface{}{
90        "TransactionID":   payment.TransactionID,
91        "TransactionDate": payment.CreatedAt,
92        "PaymentMethod":   "card",
93        "Amount":          payment.Amount / 100, // Convert from cents to
     currency units
94      }
95      err = app.mailer.Send(backer.Email, "fund_receipt.tmpl", data)
96      if err != nil {
97        c.Logger().Error(err)
98      }
99    })
100   // Return success response with transaction details
101   return c.JSON(http.StatusCreated, envelope{
102     "message":        "Project is backed successfully",
103     "backing_id":     backing.BackingID,
104     "payment_id":     payment.PaymentID,
```

```
    "status":          payment.Status,
    "transaction_id": payment.TransactionID,
   })
}
```

Listing 4.4: Record backing handler.

This handler will do the following:

1. Validates the payment intent with Stripe
2. Creates a backing record linking the backer to the project
3. Records the payment details
4. Updates the project's current funding amount
5. Associates any rewards with the backing
6. Sends payment receipt to the backer's email (using SMTP server)

Thereafter, the funding will be successfully recorded in the platform's database and also in the Stripe transactions dashboard for trackability:

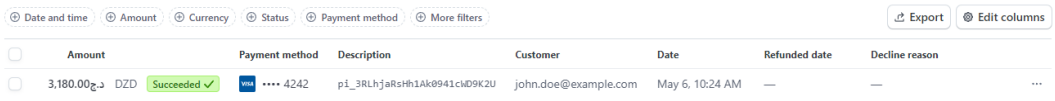


Figure 4.24: Funding transaction recorded in Stripe dashboard.

4.5 Double-score Voting Integration

At first impression, the double-score voting algorithm seems to be a backend solution and should be included in the backend section. However, the way it got implemented for CertiFund platform was by leveraging the features of PostgreSQL's PL/SQL, using functions and triggers. And, with the additional help of cron jobs in Linux (the hosting environment, more in the next section). There are three SQL functions and one trigger that power the double-score voting implementation, like this:

```
CREATE OR REPLACE FUNCTION calculate_expert_decision(project_id_param
    BIGINT)
RETURNS expert_review_decision AS $$
DECLARE
    result expert_review_decision;
BEGIN
    WITH weighted_votes AS (
```

```
7         -- Calculate weighted sum of votes using expertise level as a
  multiplier
8         SELECT
9             SUM(e.expertise_level * (er.vote->>'highly_not_recommended'
  )::float) AS weighted_highly_not_recommended,
10            SUM(e.expertise_level * (er.vote->>'not_recommended')::
  float) AS weighted_not_recommended,
11            SUM(e.expertise_level * (er.vote->>'recommended')::float)
  AS weighted_recommended,
12            SUM(e.expertise_level * (er.vote->>'highly_recommended')::
  float) AS weighted_highly_recommended
13        FROM expert_review er
14        JOIN expert e ON er.expert_id = e.expert_id
15        WHERE er.project_id = project_id_param
16    )
17    SELECT
18        -- Determine the final decision based on which weighted vote
  category has the highest sum
19        CASE
20            -- If all weighted votes are equal, return 'neutral'
21            WHEN weighted_highly_recommended =
  weighted_highly_not_recommended AND
22                weighted_highly_recommended = weighted_not_recommended
  AND
23                weighted_highly_recommended = weighted_recommended
24            THEN 'neutral'
25
26            -- If 'highly recommended' has the highest weighted sum
27            WHEN weighted_highly_recommended >=
  weighted_highly_not_recommended AND
28                weighted_highly_recommended >= weighted_not_recommended
   AND
29                weighted_highly_recommended >= weighted_recommended
30            THEN 'highly recommended'
31
32            -- If 'recommended' has the highest weighted sum
33            WHEN weighted_recommended >=
  weighted_highly_not_recommended AND
34                weighted_recommended >= weighted_not_recommended AND
35                weighted_recommended >= weighted_highly_recommended
36            THEN 'recommended'
37
38            -- If 'not recommended' has the highest weighted sum
39            WHEN weighted_not_recommended >=
  weighted_highly_not_recommended AND
40                weighted_not_recommended >= weighted_recommended AND
41                weighted_not_recommended >= weighted_highly_recommended
42            THEN 'not recommended'
```

```
43
44              -- If 'highly not recommended' has the highest weighted sum
45              WHEN weighted_highly_not_recommended >=
    weighted_not_recommended AND
46                  weighted_highly_not_recommended >= weighted_recommended
     AND
47                  weighted_highly_not_recommended >=
    weighted_highly_recommended
48              THEN 'highly not recommended'
49
50              -- Default fallback
51              ELSE 'neutral'
52          END
53      INTO result
54      FROM weighted_votes;
55
56      RETURN result;
57 END;
58 $$ LANGUAGE plpgsql;
59
60 CREATE OR REPLACE FUNCTION set_project_approved_timestamp()
61 RETURNS TRIGGER AS $$
62 BEGIN
63      IF NEW.status = 'Approved' AND (OLD.status IS NULL OR OLD.status !=
     'Approved') THEN
64          NEW.approved_at = NOW();
65      END IF;
66      RETURN NEW;
67 END;
68 $$ LANGUAGE plpgsql;
69
70 CREATE TRIGGER track_project_approval
71 BEFORE UPDATE OF status ON project
72 FOR EACH ROW
73 EXECUTE FUNCTION set_project_approved_timestamp();
74
75 CREATE OR REPLACE FUNCTION update_experts_decisions()
76 RETURNS INTEGER AS $$
77 DECLARE
78      updated_count INTEGER := 0;
79 BEGIN
80      -- Find projects approved exactly 2 days ago and update their
    experts_decision
81      UPDATE project
82      SET experts_decision = calculate_expert_decision(project_id)
83      WHERE status = 'Approved'
84        OR status = 'Live'
85        AND experts_decision = 'unverified'
```

```
86        AND approved_at IS NOT NULL
87        AND approved_at <= NOW() - INTERVAL '2 days'
88        AND approved_at > NOW() - INTERVAL '2 days 5 minutes';
89
90    -- Send the number of changes rows to stdout
91    GET DIAGNOSTICS updated_count = ROW_COUNT;
92    RETURN updated_count;
93 END;
94 $$ LANGUAGE plpgsql;
```

Listing 4.5: Functions and triggers used to implement double-score voting.

Let's break down the functions/triggers used here:

- **calculate_expert_decision:** This function calculates the final decision for a given project using a double-score voting algorithm. The final step compares recommendation levels to determine the winner (returns "neutral" in case of a tie or all zeros).
- **set_project_approved_timestamp:** This function sets the `approved_at` column value to the current timestamp when the project's status changes to "Approved" (useful for later tracking).
- **track_project_approval:** This trigger is very important, it activates the `set_project_approved_timestamp` function when the `status` field is updated on the project table, for each row.
- **update_experts_decisions:** The most important function, it identifies projects with status "Approved" from exactly two days ago (within a 5-minute window, using `AND approved_at > NOW() - INTERVAL '2 days 5 minutes'`, for reasons explained later) and calls the `calculate_expert_decision` function to determine the final decision. It also returns the count of updated rows for logging.

Now, this `update_experts_decisions` function is designed to be called by a scheduler, where at this point, cron jobs become pertinent. Cron jobs are a Linux program that runs specific tasks or scripts periodically, It is typically used when you have task that needs to be run at a periodic time (every minute, hour, day, week, month, etc.) [75]. In our case, we need to run `update_experts_decisions` at a periodic rate to check and apply double-score voting on projects, which always check projects that their *voting period* is done, and this decision needs to be provided. Here is how it should be done:

1. First in the hosting environment of the platform, we create a script named for example `update_expert_decisions.sh` with the content:

```
#!/bin/bash

psql postgres://[username]:[password]@localhost/[database] -c "SELECT update_experts_decisions();"
```

Listing 4.6: Cron job to be scheduled.

Replace username, password, and database with the actual credentials of the database (more in the next section).

2. now we run the following command:

```
{ crontab -l; echo "*/5 * * * * /path/to/update_expert_decisions.sh >> /path/to/logfile.log 2>&1"; } | crontab -
```

Listing 4.7: One-liner command to start the cron job.

This one-liner command will open crontab and sequentially add a new cron job. Here is an explanation of its syntax:

Figure 4.25: explanation of the cron job running the score-voting algorithm.

Notice the additional condition `AND approved_at > NOW() - INTERVAL '2 days 5 minutes'`. This ensures the cron job matches its periodic execution rate.

4.6 Deployment

The deployment phase is the final stage to go live to the actual customers, and it's one of the most important software development phases. Deployment of the software involves making the software available for use. This process can be done using the definitive deployment to the actually VPS (Virtual Private Server) to be hosted there executing a specific version of the software, or throughout the establishment of CI/CD pipeline, which avoids integration-hell (a common challenge faced by organizations that rely on multiple software applications and systems that need to exchange data and work together) by implementing a continuous integration, which allows the users to constantly get the latest working version of the software, and also helps the developers work on the same codebase without much trouble. Additionally, through continuous delivery, which keeps the process of deployment automatic, saves time and enhances the developer experience, avoiding the manual deployment [50].

To explain the process of deployment, we selected render.com, which is a cloud-based hosting platform that has a generous free plan to test the deployment process. Deployment to the cloud has been and still is widely popular among many teams, given that it saves the developer time and effort and provides the necessary hardware (a movement called Infrastructure as a Service, IaaS). After creating an account and project inside the platform, let's start step by step, with the bottom layer, the database:

4.6.1 Database

We start on the dashboard, then click on the button `New` above in the header. After that, we select the database option (postgres):

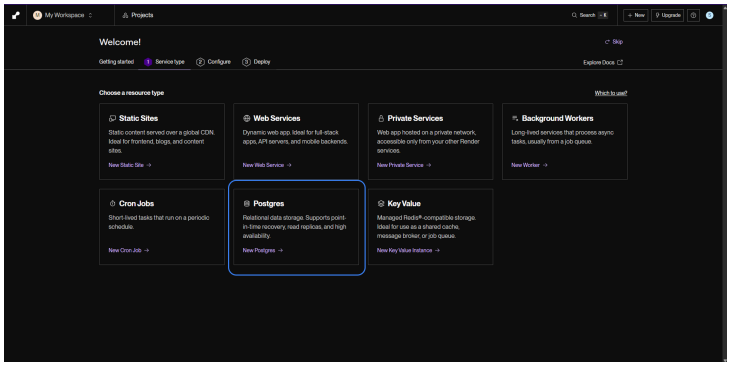


Figure 4.26: Create a new deployable instance from render dashboard.

After that, we go to the page where we fill out information about the database instance (username, password, database name, etc.). After that, we confirm, and now the database is successfully deployed:

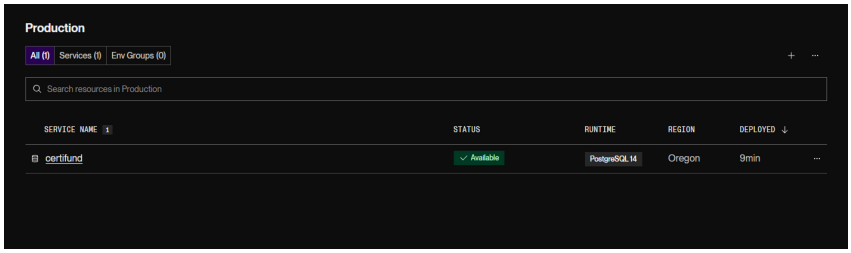


Figure 4.27: Currently deployed instances, showing the database `certifund` instance deployed.

Next, we need to copy two important data from the certifund database page, which are the external database URL (to access the database) and the psql command (to execute SQL commands remotely).

Figure 4.28: External database URL and PSQL command, two important data to be copied.

After finishing with the database, let's move on to the backend.

4.6.2 Backend

From this part and so on, Docker containerization will play a crucial role. Let's go back to the same new service page to create a new 'web service' instance (see figure 4.25). After that, this page appears to fill in the necessary information to create the instance, we'll focus on two parts of it:

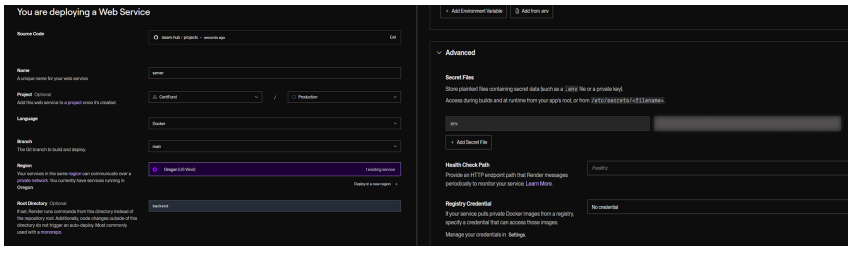


Figure 4.29: Screenshot for the two most important parts of the web service page (backend side).

The first part (left) is where we need to link the instance with our project repository in Github, then we need to define the root directory where the project to be deployed is located, render then will look on that root directory for backend Dockerfile which is a set of commended to be executed for deployment, this is the Dockerfile for the backend:

```
FROM golang:1.23.5-alpine

# Install make and required tools for downloading migrate
RUN apk add --no-cache make curl

# Download and install migrate tool
RUN curl -L https://github.com/golang-migrate/migrate/releases/download
   /v4.14.1/migrate.linux-amd64.tar.gz | tar xvz \
      && mv migrate.linux-amd64 /go/bin/migrate

WORKDIR /app

COPY . .

RUN make build

EXPOSE 9000

CMD [ "./bin/api" ]
```

Listing 4.8: Dockerfile for the backend deployment.

This file will do the following:

1. establish a new Docker container (with Alpine, which is a Linux distro)
2. install the tools: make, curl
3. install golang-migrate tool, necessary for the database migrations
4. create a folder named app, copy all the files into it, and then start the build process

Here, make tool proves instrumental. which allows us to define the commands and scripts to be ran, in this example, build. let's look at the build command along with its dependencies in the makefile:

```
.PHONY: build
build: migrateRun
  @echo "-> Building..."
  @go build -ldflags=${linker_flags} -o ./bin/api ./cmd/api/
  GOOS=linux GOARCH=amd64 go build -ldflags=${linker_flags} -o=./bin/
   linux_amd64/api ./cmd/api/

.PHONY: migrateRun
migrateRun:
```

```
9   @echo "-> Running up migrations..."
10  @$(MIGRATE_PATH) up
```

Listing 4.9: Two important makefile scripts to automate things.

The build command depends on the `migrateRun` command, which will run first, starting the migration process, then the build command starts to build the Go application and specify the architecture of the final output (one binary file !), in this case, it's linux/amd64 OS.

Moving to the second part (right), under the advanced section, we will add a new file which we will name `.env` that will hold, of course, our environment variables, which are:

```
1   PORT=[specified port]
2   DSN=[here we past the external database URL we copied before]
3   RPS=[rps rate, for the rate limiting]
4   BURST=[burst rate, for the rate limiting]
5   DISABLED=[whether to disable the rate limiting or not, the default
    value is false]
6   CLOUD_NAME=[cloud name, from cloudinary platform]
7   API_KEY=[api key, from cloudinary]
8   API_SECRET=[api secret, from cloudinary]
9   SMTP_HOST=[smtp host value]
10  SMTP_PORT=[smtp port]
11  SMTP_USERNAME=[smtp username]
12  SMTP_PASSWORD=[smtp password]
13  SMTP_SENDER=[smtp sender value]
14  STRIPE_SECRET_KEY=[stripe secret key]
```

Listing 4.10: The file .env for backend project.

notice: the smtp server used for the application is mailtrap.io.

After that, we create the instance, and once the building process is finished, we will get our backend instance ready:

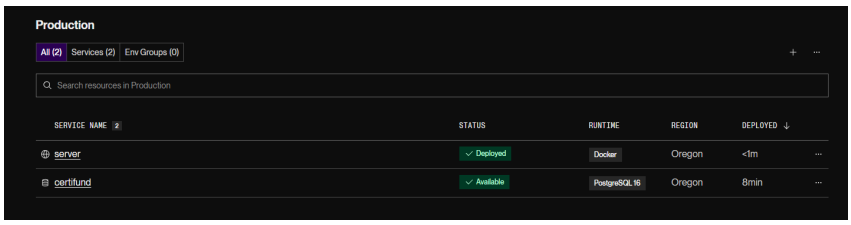


Figure 4.30: Currently deployed instances, showing the database 'certifund' instance deployed along with the 'server' backend instance.

4.6.3 Frontend

The frontend instance will follow the same trajectory as the backend service, create a new web service then specifying the frontend project instead where the Dockerfile of the frontend project will be found, here in particular, all the nextjs applications have the same deployment commands, so instead of writing our own custom Dockerfile, we retrieved one that is available on this URL. Thereafter, we add the `.env` file which will be:

```
1  API_URL="[the backend instance URL, retrieved when clicking the
   instance]/v1"
2  NEXT_PUBLIC_API_KEY="[tinyMCE public api key]"
3  NEXT_PUBLIC_STRIPE_PUBLIC_KEY="[stripe public key]"
4  NEXT_PUBLIC_SUPABASE_URL="[supabase public url, from supabase
   dashboard]"
5  NEXT_PUBLIC_SUPABASE_SERVICE_ROLE_KEY="[supabase service role key,
   from supabase dashboard]"
```

Listing 4.11: The file .env for frontend project.

It's also important to add the line `output:'standalone'` to the `next.config.ts` to create a standalone folder for the deployment (missing that will cause errors). After building the three main instances of CertiFund platform should be ready and we can access our deployed application!

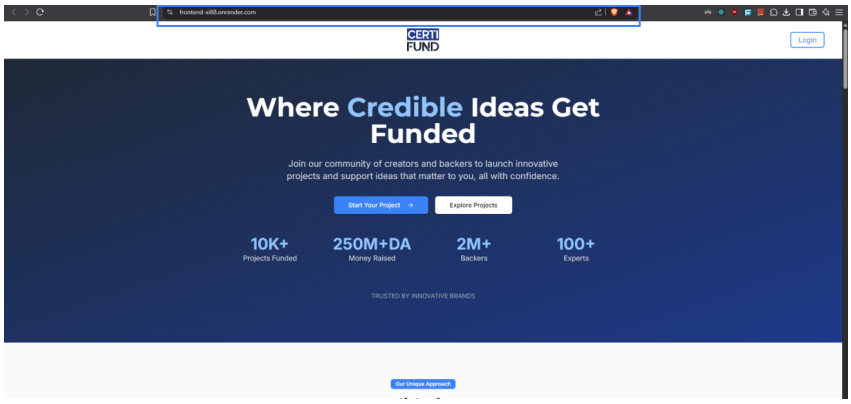


Figure 4.31: CertiFund platform deployed.

Next, there is just one missing thing, which is the cron job.

4.6.4 Double-score's cron job

The deployment of the cron job couldn't happen due to the reason that you need to subscribe to the premium version of render in order to have it, however, this is the process to create one. Again, like the previous steps, we go to the new instance page and

click on the cron job; from there, a new page appears where we will fill in the necessary information. In the root directory part, we will leave it by default, which will take the root directory of the whole project. There, we add a new Dockerfile for the cron job process:

```
    FROM ubuntu:latest

RUN apt-get update && \

      apt-get install -y make curl

COPY ./update_experts_decisions.sh /

CMD ./update_experts_decisions.sh
```

Listing 4.12: Dockerfile for the cron job deployment.

This container (runs on Ubuntu) will install the right tools, then copy the `update_experts_decisions.s` and run it afterwards. This file is added from the additional files under the advanced tab (where we added the `.env` files):

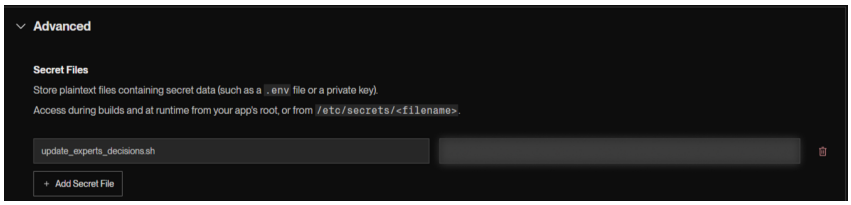


Figure 4.32: Secret files sections where we added update_experts_decisions.sh file.

The content of this file will be the same as it was before (see section 4.5). Except for a small change, we will instead of the used psql DSN (Data Source Name), we will use the PSQL command copied from the database creation part (figure 4.28).

4.7 Conclusion

In this chapter, we explained in plain detail how this system works in depth, examining all the parts of modern software engineering that power the applications that are made to be good products. By the iterative methodology, CertiFund were in continuous development and enhancement to ensure the best-possible outcome, Carefully built from the breaks of the UI which attract the users to use this platform, to the server side of the system that controls effectively the system and maintain it's powering, to the establishment of it's core

feature that this whole thesis propose and provides. And, finishing with the deployment process to bring this system to life. This chapter was a projection of the work that has been done in chapter before and and the entire theoretical side part, building upon the best practices and works established by both academia and the industry.

Part III

General Conclusion and Perspectives

Chapter 5

Conclusion

Although crowdfunding has become a game-changing financial alternative, providing an excellent choice for creators, startup owners, artisans, and much more, the opportunity to seek funding and gain market share from people at the same time. Despite that, the lack of objective competence evaluation, questionable Reliability of the third-party endorsements, and procedures to counteract inherent biases in project signaling are just a few of the serious flaws in the present frameworks of establishing trust and mitigating the information asymmetry. This thesis addresses this portion of third-party endorsements, enhancing the mechanisms of evaluation for experts. It started with the theoretical framework, beginning with the introduction to the subject of this thesis and what drives it to achieve, and addresses the research problems it states (chapter 1). Next, it moves on to the objectives of this thesis and the research methodology followed to conduct this work. Thereafter, it provides the complete context of this thesis, which is centered around crowdfunding, trust, signaling theory, and the existing trust-enhancement mechanisms used by the major platforms and discussed in academia. Covering both the state of the art of each part, the historical framework, and a critical examination to extract the flows to be addressed. Subsequently, it provides the proposed solution that represents the essence of this work (chapter 2). Following this, we move to the representational concrete practical framework, which begins with the higher-level design and technical mechanisms established to orchestrate the technical representation of the proposed solution, throughout CertiFund platform (chapter 3). Afterward, it starts to detail and demonstrate the technical work done on this thesis from the first steps to the deployment part, providing

the technical essence of the thesis (chapter 4).

This thesis's work can be seen as a double-trajectory endeavor, contributing to both the academic and industrial practical sides. We contributed to the continuous research to enhance the crowdfunding process in the digital world. Particularly, on the part of the credibility of the projects presented. Additionally, we contributed by proposing a better voting mechanism to better represent the opinions of experts within an expert-evaluation-driven crowdfunding platform, which, despite being built to provide a proof of concept to the theoretical work in this thesis, also provides a competitive system that could be easily launched and maintained in the market. Which could be insightful to business owners interested in the crowdfunding context, to apply what could be a remarkable advancement for this world of *decentralized* funding.

Chapter 6

Perspectives

Nothing is perfect, this is a crucial point that we need to start with in order to accomplish and reach a just-enough satisfiable level of the product. Considering that innovation and the true impact of it are accumulated, step by step, small movements by a person followed by another and another, and so on, to achieve the big outcomes. This thesis proposed a solution to define flaws and problems faced during the crowdfunding process, and represents a great basis to extend this work to other *better* forms by researchers in both academia and the industry. In light of the previous, we suggest a few guidelines for further research that will assist the researchers in following this work:

- **Real-world evaluation:** As we stated before, the work conducted on this thesis concerns the theoretical addressing of the problems identified along with a proof of concept, MVP[2] that is CertiFund platform, in order to represent the integration and execution of the proposed double-score algorithm. This means that this work needs more evaluation and real-world use cases to conduct a data analysis and evaluation to prove the impact of this work.
- **Robust experts' scoring:** The second crucial score used in the double-score voting algorithm is the expert score. Which is defined by the administrators of the platform during the registration phase for the experts. Although, the score is defined based on several aspects like the experience, number of publications, contribution to the industry etc., but there aren't a more objective and sort of *automatic* way to score or evaluate the expert based on, which opens the doors to

[2]Minimum Valuable Product

more work considering the matter in order to provide a robust objective basis to evaluate the expert based on it.

- **Blockchain-integration:** Blockchain subject in particular represents an interesting feature personally, and to CertiFund platform. Considering what's demonstrated before that the blockchain decentralized mechanism could be invaluable if not followed up by additional mechanisms (see section 2.4). It's interesting for future work to examine the interconnection between the decentralization, immutable transaction-recording mechanism, along with the double-score-driven expert validation systems, which assumably will provide *double-trust* enhancement, which will render the systems that adopt the two more robust.
- **AI-tools:** With the rise of the LLM[1] models, and as the majority of the platforms nowadays are *gold-rushing* towards adopting and integrating these systems with their platforms, as the statistics prove [81] [63]. As many platforms have more hardware and software restrictions to build their own LLM, they consider using the major LLMs available by subscription. But with the obstacle of context of the business logic and the necessary data, LLMs' benefit could be bound, which is currently resolved by the invention of MCP (Model Context Protocol) which simplifies the access of LLMs to tools, software, services and data, regardless of the format and the way they built, think of it as an API for LLMs [49]. With that said it's an interesting point to implement an MCP server as an extension to the crowdfunding platforms, providing context and data necessary to leverage the use of them (LLMs) in reviewing projects' rules adherence, detection of spam, help the administrators monitor better the platform, and all the possible use cases resulted from this merge between LLMs and the crowdfunding platform.

[1]Large Language Model

Bibliography

[1] Agrawal, A.K., Catalini, C., Goldfarb, A., 2011. The Geography of Crowdfunding. National Bureau of Economic Research Tech. rep.

[2] Azevedo, M. A., 2025. 2025 will likely be another brutal year of failed startups, data suggests. TechCrunch. https://techcrunch.com/2025/01/26/2025-will-likely-be-another-brutal-year-of-failed-startups-data-suggests/.

[3] Belavina, E., Marinesi, S., Tsoukalas, G., 2018. Rethinking Crowdfunding Platform Design: Mechanisms to Deter Misconduct and Improve Efficiency. ERPN: Start-Up & Small Business Finance (Sub-Topic). https://doi.org/10.2139/ssrn.3093437.

[4] Belleflamme, P., Omrani, N., Peitz, M., 2015. The Economics of Crowdfunding Platforms. ERN: Information Asymmetry Models (Topic). https://doi.org/10.2139/ssrn.2585611.

[5] Black, D., 1948. On the rationale of group decision-making. Journal of Political Economy, 56(1), 23–34.

[6] Brandl, F., Peters, D., 2022. Approval voting under dichotomous preferences: A catalogue of characterizations. Journal of Economic Theory, 205, 105532. https://doi.org/10.1016/j.jet.2022.105532.

[7] Brams, S., Fishburn, P., 1978. Approval Voting. American Political Science Review, 72, 831–847. https://doi.org/10.2307/1955105.

[8] Brown, T., Boon, E., Pitt, L., 2017. Seeking funding in order to sell: Crowdfunding as a marketing tool. Business Horizons, 60, 189–195. https://doi.org/10.1016/J.BUSHOR.2016.11.004.

[9] Buysere, K. D., Gajda, O., Kleverlaan, R., Marom, D., 2012. A Framework for European Crowdfunding. European Crowdfunding Network (ECN).

[10] ByteByteGo, 2025. The art of REST API design: Idempotency, pagination, and security. https://blog.bytebytego.com/p/the-art-of-rest-api-design-idempotency.

[11] Chakraborty, S., Swinney, R., 2019. Signaling to the Crowd: Private Quality Information and Rewards-Based Crowdfunding. Corporate Finance: Capital Structure & Payout Policies eJournal. https://doi.org/10.2139/ssrn.2885457.

[12] Chawdhry, P. k., Masera, M., Wilikens, M. andre, 2002. Strategies for Trust and Confidence in B2C e-Commerce. http://www.digiworld.org/fic/revue_telech/436/CS45_CHAWDHRY_MASERA_%20WILIKENS.pdf.

[13] Chen, D., Huang, C., Liu, D., Lai, F., 2024. The role of expertise in herding behaviors: evidence from a crowdfunding market. Electronic Commerce Research, 24(1), 155–203.

[14] Chen, Y., Zhou, S., Jin, W., Chen, S., 2022. Investigating the determinants of medical crowdfunding performance: a signaling theory perspective. Internet Research, 33, 1134–1156. https://doi.org/10.1108/intr-09-2021-0652.

[15] Cherry, K., 2024. Color psychology: Does it affect how you feel? Verywell Mind. https://www.verywellmind.com/color-psychology-2795824.

[16] Chiaramonte, M., 2024. The benefits of a three-layered application architecture. vFunction. https://vfunction.com/blog/the-benefits-of-a-three-layered-application-architecture/.

[17] Cloudflare, n.d. What is rate limiting? https://www.cloudflare.com/learning/bots/what-is-rate-limiting/.

[18] Colombo, M., Franzoni, C., Rossi-Lamastra, C., 2015. Internal Social Capital and the Attraction of Early Contributions in Crowdfunding. Entrepreneurship Theory and Practice, 39, 100–175. https://doi.org/10.1111/etap.12118.

[19] Connelly, B.L., Certo, S.T., Ireland, R.D., Reutzel, C.R., 2011. Signaling Theory: A Review and Assessment. Journal of Management, 37, 39–67.

[20] Connective Web Design, 2025. Best font combinations for Inter. https://connectivewebdesign.com/blog/best-font-combinations-for-inter.

[21] Courtney, C., Dutta, S., Li, Y., 2016. Resolving Information Asymmetry: Signaling, Endorsement, and Crowdfunding Success. Entrepreneurship Theory and Practice, 41,

265–290.

[22] Cumming, D.J., Leboeuf, G., Schwienbacher, A., 2019. Crowdfunding Models: Keep-It-All vs. All-Or-Nothing. Corporate Finance: Governance.

[23] Dai, H., Zhang, D., 2019. Prosocial Goal Pursuit in Crowdfunding: Evidence from Kickstarter. Journal of Marketing Research, 56, 498–517. https://doi.org/10.1177/0022243718821697.

[24] Deivaganapathy, G., 2025. System Design 75: Day 1 — Introduction to High-Level Design, Latency, RTT, Scaling & CAP Theorem. Medium. https://medium.com/@gunalan.d.official/system-design-75-day-1-introduction-to-high-level-design-latency-rtt-scaling-cap-t

[25] Dey, R., n.d. Startup failure during economic downturns: Historical stats. WinSavvy. https://www.winsavvy.com/startup-failure-during-economic-downturns-historical-stats/.

[26] Farley, D., 2022. What is an API? Modern Software Engineering, 148–149. Addison-Wesley Professional.

[27] Farley, D., 2022. Working Iteratively. Modern Software Engineering, 43–55. Addison-Wesley Professional.

[28] Ferreira, V., Papaoikonomou, E., Terceño, A., 2021. Unpeel the layers of trust! A comparative analysis of crowdfunding platforms and what they do to generate trust. Business Horizons.

[29] Fisher, G., Kuratko, D.F., Bloodgood, J.M., Hornsby, J., 2017. Legitimate to whom? The challenge of audience diversity and new venture legitimacy. Journal of Business Venturing, 32, 52–71.

[30] Fortune Business Insights, 2025. Crowdfunding market size, share & growth report [2032]. https://www.fortunebusinessinsights.com/crowdfunding-market-107129.

[31] Fowler, M., 2022. Conway's Law. MartinFowler.com. https://martinfowler.com/bliki/ConwaysLaw.html.

[32] Fronea, C., 2021. History of crowdfunding. Smallbrooks. https://smallbrooks.com/history-of-crowdfunding/.

[33] GeeksforGeeks, 2025. Software Development Life Cycle (SDLC). https://www.geeksforgeeks.org/software-development-life-cycle-sdlc/.

[34] Gerber, E., Hui, J., 2013. Crowdfunding. ACM Transactions on Computer-Human Interaction (TOCHI), 20, 1–32.

[35] Grabner-Kräuter, S., Kaluscha, E.A., 2003. Empirical research in on-line trust: a review and critical assessment. International Journal of Human-Computer Studies, 58, 783–812.

[36] Grigutytė, M., 2023. What is bcrypt and how does it work? NordVPN. https://nordvpn.com/blog/what-is-bcrypt/.

[37] Hennigan, P., 2004. The Birth of a Legal Institution: The Formation of the Waqf in Third-Century A.H. Hanafī Legal Discourse.

[38] Herijanto, H., 2022. Al amanah in al qur'an vs trust: a comparative study. International Journal of Ethics and Systems. https://doi.org/10.1108/ijoes-03-2021-0064.

[39] Hu, J., You, W., Chen, B., 2024. The Impact of Experts' Voting on the Fundraising Performance of Crowdfunding Projects. Wuhan International Conference on E-Business.

[40] Hughes, A., 2024. Encryption vs. hashing vs. salting – What's the difference? Ping Identity. https://www.pingidentity.com/en/resources/blog/post/encryption-vs-hashing-vs-salting.html.

[41] Hunt, A., Thomas, D., 2020. Before the Project. The Pragmatic Programmer, 405–416. Addison-Wesley Professional.

[42] Julakadar, M., 2025. Monorepos: A comprehensive guide with examples. Medium. https://medium.com/@julakadaredrishi/monorepos-a-comprehensive-guide-with-examples-63202cfab711.

[43] Kirby, E., Worner, S., 2014. Crowd-funding: An infant industry growing fast. IOSCO Research Department, 2014, 1–63.

[44] Kshirsagar, A., 2024. Stateless vs. stateful authentication: Understanding the key differences. Medium. https://medium.com/@abhikshirsagar1999/stateless-vs-stateful-authentication-understanding-the-key-differences-e33eaa569b23

[45] Lagazio, C., Querci, F., 2018. Exploring the multi-sided nature of crowdfunding campaign success. Journal of Business Research. https://doi.org/10.1016/J.JBUSRES.2018.05.031.

[46] Lee, B., Ang, L., Dubelaar, C., 2005. Lemons on the Web: A signalling approach to the problem of trust in Internet commerce. Journal of Economic Psychology, 26, 607–623.

[47] Li, Q., Wang, N., 2024. Fundraiser engagement, third-party endorsement and crowdfunding performance: A configurational theory approach. PLOS ONE, 19.

[48] Lindemulder, G., Kosinski, M., 2024. What is role-based access control (RBAC)? IBM. https://www.ibm.com/think/topics/rbac.

[49] Lisowski, E., 2025. MCP explained: The new standard connecting AI to everything. Medium. https://medium.com/@elisowski/mcp-explained-the-new-standard-connecting-ai-to-everything-79c5a1c98288.

[50] Mad Devs, n.d. Deployment stage in software development: Mad Devs' approach. https://maddevs.io/development-process/deployment-stage/.

[51] Maskin, E., Sen, A., 2014. The Arrow Impossibility Theorem.

[52] McNamara, R. J., 2013. The Statue of Liberty and America's crowdfunding pioneer. BBC News. https://www.bbc.com/news/magazine-21932675.

[53] McNamara, R. J., 2019. Who paid for the Statue of Liberty? ThoughtCo. https://www.thoughtco.com/who-paid-for-the-statue-of-liberty-1773828.

[54] Meng, X., 2022. Why startups fail: new roadmap for entrepreneurial success. Journal of Commercial Biotechnology, 27(2), 7–11. https://doi.org/10.5912/jcb1268.

[55] Moysidou, K., Hausberg, J.P., 2020. In crowdfunding we trust: A trust-building model in lending crowdfunding. Journal of Small Business Management, 58, 511–543.

[56] Nakov, S., n.d. Secure random generators (CSPRNG). Practical Cryptography for Developers. https://cryptobook.nakov.com/secure-random-generators/secure-random-generators-csprng.

[57] Nalimov, C., 2020. Crow's foot notation in entity-relationship diagrams. Gleek. https://www.gleek.io/blog/crows-foot-notation.

[58] Norman, D., 2013. The Psychopathology of Everyday Things. The Design of Everyday Things: Revised and Expanded Edition, 8–9. Basic Books; Revised edition.

[59] Olken, B., 2012. Lecture 12: Sometimes it gets complicated: Condorcet's paradox and Arrow's impossibility theorem. MIT OpenCourseWare. https://ocw.mit.edu/courses/14-75-political-economy-and-economic-development-fall-2012/a9fd8e5ab75a325016094e6bbe625b2a_MIT14_75F12_Lec12.pdf.

[60] Pan, Y., 2008. Improving the Cyber 'Lemons' Market with a Trust-Intermediary in E-Commerce: Model and Case Study. 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing, 1–6.

[61] Penner, J., 2019. The historical origins of the trust. The Law of Trusts. https://doi.org/10.1093/HE/9780198747598.003.0001.

[62] Petit, A., Wirtz, P., 2021. Experts in the crowd and their influence on herding in reward-based crowdfunding of cultural projects. Small Business Economics, 58, 419–449. https://doi.org/10.1007/s11187-020-00424-x.

[63] Pragma Market Research, 2024. Global Large Language Model (LLM) market size, share, growth drivers, competitive analysis, recent trends & developments, and demand forecast to 2030. https://www.pragmamarketresearch.com/reports/121032/large-language-model-llm-market-size.

[64] Prashar, A., Gupta, P., 2023. How to build trust in Gen Y in online donation crowdfunding: an experimental study. Behaviour & Information Technology, 43, 677–694. https://doi.org/10.1080/0144929X.2023.2183061.

[65] Prisma, n.d. What are database migrations? Prisma's Data Guide. https://www.prisma.io/dataguide/types/relational/what-are-database-migrations.

[66] Redis, n.d. Rate limiting. https://redis.io/glossary/rate-limiting.

[67] Richards, M., Ford, N., 2020. Fundamentals of Software Architecture: An Engineering Approach. O'Reilly Media.

[68] Rocholl, J., 2016. An Introduction to Crowdfunding, 14, 03–05.

[69] Roy, S., 2021. Theory of Social Proof and Legal Compliance: A Socio-Cognitive Explanation for Regulatory (Non) Compliance. German Law Journal, 22(2), 238–255.

[70] Smith, W., 2023. The case for score voting. Constitutional Political Economy, 34, 297–309. https://doi.org/10.1007/s10602-023-09403-2.

[71] Smith, W., Quintal, J., Greene, D., 2005. What if the 2004 US presidential election had been held using Range or Approval voting.

[72] StackHawk, n.d. Demystifying CORS: Best practices & common pitfalls. https://www.stackhawk.com/blog/what-is-cors/.

[73] Strohmaier, D., Zeng, J., Hafeez, M., 2019. Trust, distrust, and crowdfunding: A study on perceptions of institutional mechanisms. Telematics Informatics, 43.

[74] Tang, Z., Hu, Y. J., Smith, M. D., 2007. Gaining Trust Through Online Privacy Protection: Self-Regulation, Mandatory Standards, or Caveat Emptor. Social Science Research Network. https://doi.org/10.2139/SSRN.555878.

[75] The Kubernetes Authors, n.d. CronJob. Kubernetes. https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/.

[76] TheCollector, n.d. How did Socrates view democracy? https://www.thecollector.com/how-did-socrates-view-democracy/.

[77] Tian, X., Song, Y., Luo, C., Zhou, X., Lev, B., 2021. Herding behavior in supplier innovation crowdfunding: Evidence from Kickstarter. International Journal of Production Economics, 239, 108184. https://doi.org/10.1016/J.IJPE.2021.108184.

[78] Tullock, G., 2005. Problems of Voting. Public Choice, 123, 49–58. https://doi.org/10.1007/S11127-005-7905-3.

[79] Turan, M., 2007. Transparency in Electronic Markets: A Unified Framework.

[80] Unsal, M.S., 2025. Blockchain and Stablecoin Integration for Crowdfunding: A framework for enhanced efficiency, security, and liquidity. ArXiv, abs/2501.11145.

[81] Uspenskyi, S., 2025. Large language model statistics and numbers (2025). Springs. https://springsapps.com/knowledge/large-language-model-statistics-and-numbers-2024.

[82] Vismara, S., 2017. Signaling to Overcome Inefficiencies in Crowdfunding Markets. IRPN: Innovation & Finance (Topic). https://doi.org/10.1007/978-3-319-66119-3_3.

[83] Wallis, W. A., 1980. The Statistical Research Group, 1942–1945. Journal of the American Statistical Association, 75(370), 320–330. https://doi.org/10.2307/2287451.

[84] Wang, N., Liang, H., Ge, S., Xue, Y., 2015. How to Crowdfund More: A Signaling Perspective.

[85] Xu, Y., Li, Q., Zhang, C., Tan, Y., Zhang, P., Wang, G., Zhang, Y., 2023. A decentralized trust management mechanism for crowdfunding. Information Sciences, 638, 118969. https://doi.org/10.1016/j.ins.2023.118969.

[86] Zanini, A., 2023. Comparing ER diagrams, ER models and relational schemas. DbVisualizer. https://www.dbvis.com/thetable/er-diagrams-vs-er-models-vs-relational-schemas/.

[87] Zhang, Y., Zheng, X., 2016. A Study of Herd Behavior Based on the Chinese Stock Market. Journal of Applied Management and Investments, 5(2), 131–135. https://ideas.repec.org/a/ods/journl/v5y2016i2p131-135.html.

[88] Zheng, H., Hung, J., Qi, Z., Xu, B., 2016. The role of trust management in reward-based crowdfunding. Online Information Review, 40, 97–118. https://doi.org/10.1108/OIR-04-2015-0099.