

جامعة ابن زهر  
+ⵓⵎⵓⵏⵉⵔ ⵉⵎⵓⵏⵉⵔ  
UNIVERSITÉ IBN ZOHR



UNIVERSITÉ IBN ZOHR  
FACULTÉ DES SCIENCES AGADIR

Département d'Informatique

FILIÈRE SCIENCES MATHÉMATIQUES ET INFORMATIQUE

PROJECT DE FIN D'ÉTUDE

Présenté Par: Taha Bouhsine

POUR L'OBTENTION DE LA  
Licence en Sciences Mathématiques et Informatique

---

# Design And Full Stack Development Of A Crowdfunding Platform « Sahem »

---

ENCADRÉ PAR : Dr. BELAQZIZ SALWA

*Année universitaire : 2019 - 2020*

This Page is Intentionally Left Blank



---

# Dedication

In memoriam Hanya Laaiba, my dear grandmother,  
her last words of encouragement are living with me to this day,  
feeding my motivation, and always urging me to pursue my dreams, and my studies,  
never to be forgotten, never to be erased,  
may your soul rest in peace,  
may your memory guide me through my future steps to the unknown

To your beautiful soul,  
I dedicate this work.

*Taha* BOUHSINE



---

# Acknowledgements

I wish to express my sincere appreciation to my supervisor, Dr. BELAQZIZ SALWA , she convinci -ngly guided and encouraged me to take the exact desisions and do the right things even when the road got tough. Without her persistent help, and profound belief in my abilities the goal of this project would not have been realized.

I wish to acknowledge the help provided by the administration staff in Ibn Zohr University, Faculty of Science, and would like to thank them for giving us a great studying environment. And to all my classmates and the professors of Computer Science Departement for having contributed in the formulation of our ideas and for providing a suitable working environment towards the completion of this project.

Finally, I must express my very profound gratitude to my parents for raising me, and to my friends for providing me with unfailing support and continuous encouragement throughout my years of study and throughout the process of researching and developing this project. This accomplishment would not have been possible without them. Thank you.



---

# Abstract

Crowdfunding refers to behavior where public individuals, rather than institutions, to use digital technologies to make financial contributions to people, projects, or businesses in response to either financial or developmental commitments from those people, projects, or businesses. Sahem Crowdfunding Platform is a project that aims to develop a system that will be a gateway to allow project funders to contribute towards a good project idea of a creator who lacks sufficient resources to implement the project idea. It is important to develop this project because it will seek to create a platform that will bridge the gap between viable project ideas and successful implementation of these projects so that good project ideas that can benefit individuals and the community at large cannot go into a waste. With the aim to successfully develop this project, we will employ the use of the Software development methodology, Iterations and Increments process, as well as we will be using the Kanban board Model to divide and organize our project into small tasks, and also will be carrying out reviews and analysis of existing solutions in an attempt to create a unique user experience with the use of Stripe API as a payment Service.



---

# Contents

|   |            |
|---|------------|
| <b>Dedication</b>                           | <b>ii</b>  |
| <b>Acknowledgements</b>                     | <b>iii</b> |
| <b>Abstract</b>                             | <b>iv</b>  |
| <b>Contents</b>                             | <b>vii</b> |
| <b>List of Figures</b>                      | <b>ix</b>  |
| <b>List of Tables</b>                       | <b>x</b>   |
| <b>Acronyms</b>                             | <b>xi</b>  |
| <b>Introduction</b>                         | <b>1</b>   |
| <b>1 Project General Context</b>            | <b>7</b>   |
| 1.1 Project Presentation . . . . .          | 7          |
| 1.1.1 Presentation . . . . .                | 7          |
| 1.1.2 Problem . . . . .                     | 8          |
| 1.1.3 Main Actors . . . . .                 | 8          |
| 1.1.4 Design Principles . . . . .           | 10         |
| 1.1.5 Objectives . . . . .                  | 11         |
| 1.2 Functionality . . . . .                 | 12         |
| 1.3 Feasibility . . . . .                   | 13         |
| 1.4 Project Management . . . . .            | 13         |
| 1.4.1 Project Development Process . . . . . | 13         |

|          |  |           |
|----------|--|-----------|
| 1.4.2    | Monitoring And Planning . . . . .                    | 15        |
| <b>2</b> | <b>Analysis And Design</b>                           | <b>17</b> |
| 2.1      | General description of the platform . . . . .        | 17        |
| 2.2      | Specification of requirements . . . . .              | 18        |
| 2.2.1    | Requirements . . . . .                               | 18        |
| 2.2.2    | Identification of actors . . . . .                   | 19        |
| 2.2.3    | Identification of objectives and use cases . . . . . | 19        |
| 2.3      | Functional Analysis . . . . .                        | 21        |
| 2.4      | Dynamic Analysis . . . . .                           | 23        |
| 2.5      | Structured Analysis . . . . .                        | 27        |
| <b>3</b> | <b>Technical And Environmental Study</b>             | <b>28</b> |
| 3.1      | Capture of technical needs . . . . .                 | 28        |
| 3.1.1    | Capturing Software Specifications . . . . .          | 29        |
| 3.2      | Adopted Software Architecture . . . . .              | 30        |
| 3.2.1    | Software Architecture Pattern . . . . .              | 31        |
| 3.2.2    | Codebase Architecture . . . . .                      | 31        |
| 3.2.3    | Controller Architecture . . . . .                    | 31        |
| 3.2.4    | Cloud Architecture . . . . .                         | 32        |
| 3.3      | Frameworks . . . . .                                 | 32        |
| 3.3.1    | Front-End . . . . .                                  | 33        |
| 3.3.2    | Back-End . . . . .                                   | 34        |
| 3.4      | Programming Languages . . . . .                      | 35        |
| 3.4.1    | Conception . . . . .                                 | 35        |
| 3.4.2    | General . . . . .                                    | 35        |
| 3.4.3    | Front-End . . . . .                                  | 35        |
| 3.4.4    | Back-End . . . . .                                   | 36        |
| <b>4</b> | <b>Realization, GUI And Tests</b>                    | <b>37</b> |
| 4.1      | Hardware Environments . . . . .                      | 37        |
| 4.1.1    | Development hardware . . . . .                       | 37        |
| 4.1.2    | Production hardware . . . . .                        | 38        |
| 4.2      | Development Environments . . . . .                   | 38        |
| 4.2.1    | Design And Planing . . . . .                         | 38        |
| 4.2.2    | User Interface And Product Design . . . . .          | 39        |
| 4.2.3    | Development . . . . .                                | 39        |

|                                   |                               |           |
|-----------------------------------|-------------------------------|-----------|
| 4.3                               | Platform security . . . . .   | 41        |
| 4.3.1                             | Security Principles . . . . . | 41        |
| 4.3.2                             | Physical level . . . . .      | 42        |
| 4.3.3                             | Logical level . . . . .       | 42        |
| 4.4                               | Project Tests . . . . .       | 45        |
| 4.5                               | Deployment . . . . .          | 45        |
| 4.6                               | Interfaces . . . . .          | 46        |
| <b>Conclusion And Perspective</b> |                               | <b>52</b> |
| <b>Webography</b>                 |                               | <b>54</b> |
| <b>Bibliography</b>               |                               | <b>54</b> |



# List of Figures

|    |  |    |
|----|--|----|
| 1  | Survey on crowdfunding in Morocco [1] . . . . .        | 6  |
| 2  | Iterative process project life cycle . . . . .         | 14 |
| 3  | « Sahem » Business Objectives . . . . .                | 15 |
| 4  | « Sahem » Platform's Actors . . . . .                  | 20 |
| 5  | « Sahem » System Global Use Case . . . . .             | 21 |
| 6  | User's Authentication and Account Management . . . . . | 22 |
| 7  | Funding Projects Use Cases . . . . .                   | 22 |
| 8  | User's interactions with projects Use Cases . . . . .  | 22 |
| 9  | « Sahem » Activity Diagram . . . . .                   | 23 |
| 10 | User Authenticating Sequence Diagram . . . . .         | 24 |
| 11 | Create a Project Sequence Diagram . . . . .            | 25 |
| 12 | Get Projects Sequence Diagram . . . . .                | 25 |
| 13 | Get And Fund Project Sequence Diagram . . . . .        | 26 |
| 14 | « Sahem » Class Diagram . . . . .                      | 27 |
| 15 | Software Architecture Factors . . . . .                | 30 |
| 16 | Model View Controller . . . . .                        | 31 |
| 17 | Monolithic Applications . . . . .                      | 32 |
| 18 | The Flow In Mean Stack . . . . .                       | 33 |
| 19 | Communication In Mean Stack . . . . .                  | 33 |
| 20 | « Sahem » Platform's Hardware Architecture . . . . .   | 38 |
| 21 | Responsive Design . . . . .                            | 46 |
| 22 | « Sahem » Main Page - Navbar Activated . . . . .       | 46 |
| 23 | « Sahem » Main Page - Themes . . . . .                 | 47 |

|    |  |    |
|----|--|----|
| 24 | « Sahem » Main Page - Testimonials . . . . . | 47 |
| 25 | « Sahem » Main Page - Footer . . . . .       | 47 |
| 27 | Login Page - Themes . . . . .                | 48 |
| 26 | Login Page . . . . .                         | 48 |
| 28 | Register Page . . . . .                      | 48 |
| 29 | Fundraiser Form . . . . .                    | 49 |
| 30 | Fundraisers List . . . . .                   | 49 |
| 31 | Fundraiser View . . . . .                    | 50 |
| 32 | Profile Form . . . . .                       | 50 |
| 34 | Profile Page - Themes . . . . .              | 51 |
| 33 | Profile Page . . . . .                       | 51 |



---

## List of Tables

|   |  |    |
|---|--|----|
| 1 | Identification of objectives and use cases . . . . .           | 21 |
| 2 | Identification of technical objectives and use cases . . . . . | 30 |



---

## Acronyms

**CORS** Cross-Origin Resource Sharing

**CSRF** Cross-Site Request Forgery

**CSS** Cascading Style Sheets

**HTML** Hypertext Markup Language

**HTTP** HyperText Transfer Protocol

**JSON** JavaScript Object Notation

**JWT** JSON Web Token

**MVC** Model View Controller

**NPM** Node Package Manager

**ODM** Object Data Modelin

**PaaS** Platform as a service

**REST** Representational state transfer

**SaaS** Software as a service

**SPA** Single Page Applications

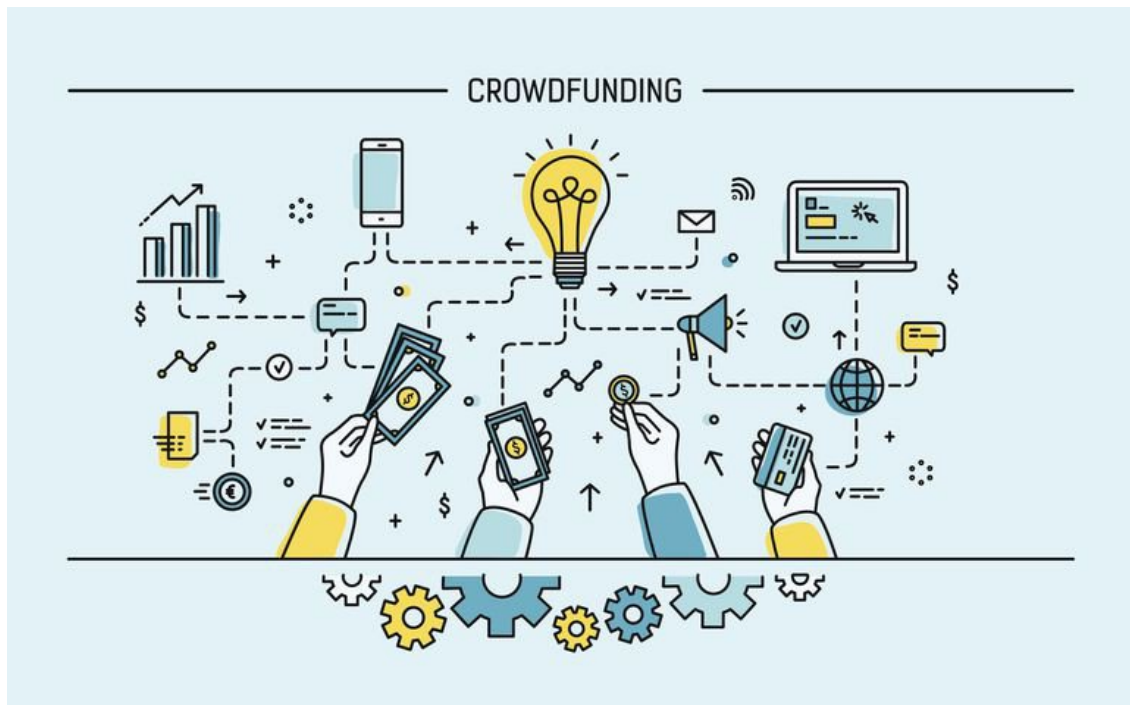
**UML** Unified Modeling Language

**VM** Virtual Machine

**XSS** Cross-Site Scripting

**XSSI** Cross-Site Script Inclusion

# Introduction



## Crowdfunding

In recent years, crowdfunding has emerged as a revolutionary financing model that allows small entrepreneurs to raise funding in the early stages of their projects, particularly those that may otherwise struggle to obtain capital (Kuppuswamy and Bayus 2013; Belleflamme et al. 2014). Today, there are approximately 1250 active crowdfunding platforms across the world, which together channeled \$16.2 billion in 2014, representing a 167 % increase from \$6.1 billion in 2013 (Massolution 2015). Having their project successfully funded is crucial to project creators as it provides not only initial funds for project development but also access to valuable future resources, and eventually turn their projects into successful entrepreneurial organizations. Previous research shows that only 45 % of the projects on these platforms are successfully funded. As a result, identifying general antecedents of funding success has been

of great interest to researchers because it can provide insights to project creators to maximize their funding success.

Which made Crowdfunding One of the biggest topics discussed around the business community for the past few years, the idea of crowdfunding using an online platform is a great tool for building capital for a startup, funding growth for your company, or development of services or products to further your business.

New crowdfunding platforms and websites are popping up regularly to meet the needs of the expanding market, with plenty of room to benefit from the advantages of this technology by creating a crowdfunding platform.

## Definition

Crowdfunding is the practice of raising money from a large number of individuals for the purposes of financing a project, venture, business, or cause. Traditionally, crowdfunding has been carried out via subscriptions, benefit events, and door-to-door fundraising. However, today the term is typically associated with raising money through website platforms, which allows crowdfunding to reach a larger pool of potential funders.

## Crowdfunding Models

Crowdfunding facilitates the raising of capital for a variety of purposes, using numerous variations of the model. Below is a typology of how the operators in the market can potentially be segregated. The majority of platforms can be categorized under these four types, but there are several variations, such as hybrid models and those platforms that define themselves in a sectoral vertical rather than by the type of finance they provide.

1. **Donation Model:** The donation model of crowdfunding is a means for charities, or those who raise money for social or charitable projects, to gather a community online and to enable them to donate to a project. While most established charities facilitate this through their own website, crowdfunding is popular for small organizations and people raising money for personal or specific charitable causes.
2. **Lending Model:** Crowdfunded lending is largely an evolution of the peer-to-peer model of lending. Projects or businesses seeking debt apply through the platform uploading their pitch, with members of the crowd taking small chunks of the overall loan.
3. **Reward Model:** The model allows people to contribute to projects and receive non-financial rewards in return, usually operating a tiered system where the more you donate the better the reward you receive. The model often closely resembles philanthropy with the donation far exceeding the monetary value of the reward or the reward costing the fundraiser little, such as experience or recognition-related rewards. In these cases, entrepreneurs or artists crowdfund the production cost of their record, movie, game or product and allow the donors to be the first recipients once the production is complete.
4. **Investment Model:** The final type is the application of crowdfunding to investing for equity, or profit/revenue sharing in businesses or projects. This form of the model has been the slowest to grow due to regulatory restrictions that relate to this type of activity.

## How big is crowdfunding?

In 2015, crowdfunding raised \$34 Billion worldwide and the total had doubled every year for the previous four years! Many crowdfunding campaigns are for quite small sums but some are very large. The model of crowdfunding which raised the most money is the lending model. In the UK crowdfunding in 2015 totaled £1,112 million and this figure is expected to continue to rise.

## Why is crowdfunding growing?

Making a public call to fundraise from the crowd is not a new idea but crowdfunding as we now understand it has grown very quickly for several reasons since its emergence in the 1990s. These include:

1. Technological developments : The emergence of wide and low-cost access to the internet and communication tools like social media means it is easier and cheaper for us to reach out to much more widely dispersed and larger groups of people.
2. Societal changes : These technical changes have also empowered us to take on new activities that were once controlled by gatekeepers. We can see this in the way people write and publish books, publish music, writing blogs, and the general sharing of our lives online. Crowdfunding is just the financial manifestation of this sense of empowerment and the ability to take “ownership” of a process. At the same time, we are also increasingly comfortable with transacting financially online be it shopping and e-commerce or checking our bank balances. This confidence to use money online is essential for the growth of crowdfunding.
3. Economic factors : The other key factor in the extraordinary growth of crowdfunding is that after the financial crisis of 2008 access to funding has been more problematic and so people are exploring alternatives to the traditional sources. At the same time interest rates have fallen to historically low levels and so “retail investors” are looking for better places to put their investments and some crowdfunding campaigns seem to offer better returns than would be available on the high street.

## Principe

The other important variation in crowdfunding is the distinction between what is known as the “Keep It All” and “All Or Nothing” models.

In a “Keep It All” campaign, you keep everything you raise regardless of whether you reach your target or not.

In an “All Or Nothing” campaign you only get to keep what you raise if you succeed in reaching your target.

Crowdfunding can be undertaken by both individuals and organizations.

## **Downsides of Crowdfunding**

1. It is not easy, to be a success takes a lot of time and effort to carry out the continued promotion of your campaign.
2. Many campaigns are unsuccessful and the successful ones take work, preparation, and effort to make them happen, there is no guarantee of success but work, preparation, and effort can increase the chances of success.
3. It is also a very public process and so you must be prepared to be open and honest in a public arena and expect brickbats and bouquets in equal measure, it is important to accept that there may be public scrutiny of your project and prepare for this.
4. Reputation considerations: there are many aspects to this, including your appearance on the platform in the first place. Might customers and others view it negatively? There are also significant expectations from investors if your crowdfunding campaign is successful. If it meets delays or runs into problems, the reputation of your company might become damaged. Finally, by putting your idea onto a crowdfunding platform before you're ready to bring it to the wider market (i.e. before it's fully developed and tested), you are leaving your company open to public criticism of your plans, idea, or business.

## **Benefits of Crowdfunding**

1. It is a very accessible process which is open to all and can be carried out on your terms
2. you decide on the amount you want to raise and the timescale to raise the funds.
3. You are in control, the promotion and selling of the project is the responsibility of your group.
4. It can bring much more than money, it can attract new people and support (nonfinancial) to your group.
5. What money it does bring can be very different from a traditional investment, funds raised through crowdfunding are unrestricted and can be used for all elements of your project.
6. It can be quick, as you are in control you can work quickly to start raising funds.
7. You will also produce a very valuable asset for you or your organization as you run a campaign and that "crowd asset" can be a very useful and enduring resource, people who support your project are also likely to support your group.
8. It's a quick and easy way to get a lot of exposure for your brand and for your new idea or initiative
9. You can engage directly with investors who could also become your customers
10. There is a low barrier to entry, often much lower than with other forms of investment
11. It's an easy way to get feedback on your idea

## **Existent Platforms**

### **Non-Profit:**

- Fundly: This is probably the best known of the non-profit crowdfunding sites. This platform charges a fee for all campaigns and a small transaction fee for processing. It



offers a customizable donation page, with an integrated mobile experience. You can use multiple media formats to promote your cause as well as blogging.

- Salsa: This p2p platform offers branded marketing materials for every supporter and customizes messages as well. This is a fully integrated system with Salsa CRM. So, if you already use that, software data is transferred easily.
- Soapbox Engage: This is a social change platform that offers not only the ability to create cause for donation but allows you to create forms, petitions, and events to further your cause.

#### Business:

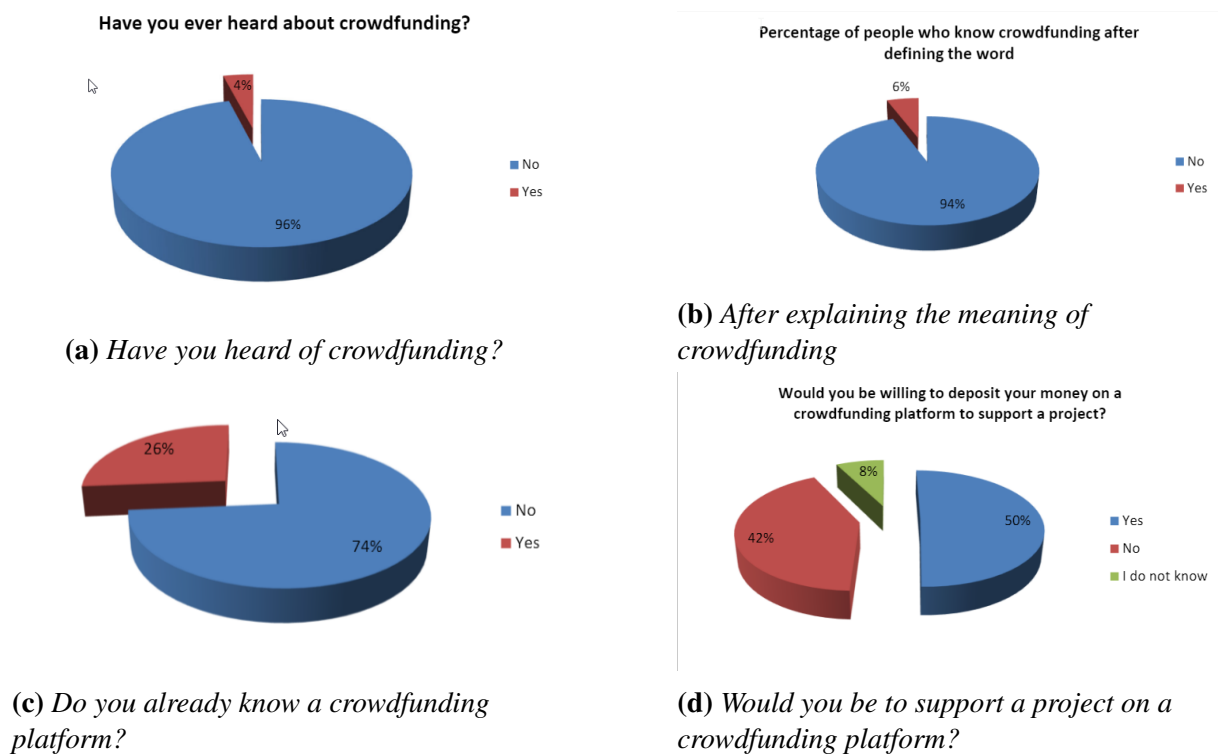
- Kickstarter: This is a crowdfunding platform that allows you to market potential products or businesses for investors to donate to. You create a custom page for your book, game, or any other creative project and then set a goal and start building funds. Each project will set up designated donations that have rewards attached to them.
- Indigogo: This is a great sight for startups and creatives that uses similar methods to Kickstarter. You set up a custom page and goals and market your campaign. They have integrated systems to help with the fulfillment of delivery, mobile management, perk options, etc.
- Seedrs: This crowdfunding platform uses equity investing to raise funds for small businesses and startups. It has three options to invest whether equity, funds, or convertible donations and allows other investors a discount in the future.

## Crowdfunding In Morocco

Expected since April 2018, the examination of the so-called «collaborative financing» bill is finally starting in Morocco. In the latter case, for management companies that want to propose capital investment, it will be necessary to obtain a visa from the Moroccan Capital Market Authority . The platforms must have a minimum capital of 310,000DH and the collaborative finance companies that will manage them must have a risk prevention and reduction policy.

A few researchers in the Economic field started a study [1], to understand the future of crowdfunding in Morocco, they started a survey with a total of 200 participants over the internet (Figure 1). Surprisingly, The majority of respondents 96% do not even know what the word crowdfunding means, and of 5% left most respondents that know about crowdfunding plan to have their projects with a percentage of 62%, while 5% answered I do not know against 33% do not want to have their projects. However, according to the distribution by the labor force, they found that this majority consists mainly of the unemployed and students, while more than half of the workers do not want to have their projects. For those who want to have their projects, it is mainly the lack of funding , and the lack of idea and the risk of failure that prevents them, so crowdfunding can fill this gap is becoming an alternative to financing projects or even a compliment.

Finally, they found that the majority who want to start their project and who have financing problems for a percentage of 55.5% are interested in crowdfunding to launch or develop their projects, however, the lack of information on this system is the first cause for those who are not interested in this new mode of financing, which is normal.



**Figure 1.** Survey on crowdfunding in Morocco [1]

## Chapter

**1****Project General Context****Contents**

|            |                             |           |
|------------|-----------------------------|-----------|
| <b>1.1</b> | <b>Project Presentation</b> | <b>7</b>  |
| 1.1.1      | Presentation                | 7         |
| 1.1.2      | Problem                     | 8         |
| 1.1.3      | Main Actors                 | 8         |
| 1.1.4      | Design Principles           | 10        |
| 1.1.5      | Objectives                  | 11        |
| <b>1.2</b> | <b>Functionality</b>        | <b>12</b> |
| <b>1.3</b> | <b>Feasibility</b>          | <b>13</b> |
| <b>1.4</b> | <b>Project Management</b>   | <b>13</b> |
| 1.4.1      | Project Development Process | 13        |
| 1.4.2      | Monitoring And Planning     | 15        |

**1.1 Project Presentation****1.1.1 Presentation**

Multiple projects and ideas all over the world go to waste and get canceled due to either lack or insufficient funds. This means the loss of a huge amount of new business chances and opportunities that would have been of great benefit for both individuals and communities.

The result is an extreme demand and concern to come up with a crowdfunding platform the necessary tools for the public to create, fund, and support causes and creative minds all over the world.

### 1.1.2 Problem

For the last few years, all over the world crowdfunding become one of the main tools and funding source for most of the new startups and creative minds all over the globe, so we sought to create a new crowdfunding application, that will bring and adopt all the creative minds and ambitious souls and provides them with a tool to seek funds from a global community, the creators should be able to coordinate multiple campaigns easier, and should be able to find people who are willing to invest with little equity involved. And help in gatekeeping that will monitor and create symbiotic relations with other algorithms and information online by using cloud-based solutions for better access. Our platform can help leap the hurdle of lack of experience in each field. Letting the barrier of entry be lessened for everyone who has a dream.

So for our project, there is a real challenge in creating a product that competes with all the other platforms out there, so we have to try and understand the human's motivation and behavior, what does urge him to help others, or seek the help of others, a lot of articles helped us along the way, studying the psychological side of human nature and the act of giving, and which factors that affect his decision to give.

### 1.1.3 Main Actors

Different players are involved in crowdfunding models. First, there are creators, innovators, and entrepreneurs that have ideas and projects to be funded. They want to use crowdfunding to gather financial support from interested supporters. Then there is the crowd of people that provides this financial support to these projects, bearing an investment risk and expecting a certain payoff. And finally, there is the crowdfunding platform, the intermediary that acts as a matchmaker between those who want to deliver the new initiatives using crowdfunding mechanisms and those who want to support such initiatives through their investment efforts [2].

1. Creator: Or capital seekers, are mainly concerned with their motivations to get involved in crowdfunding, the determinants of success, and the legal restrictions of equity-based crowdfunding [3].
2. Funder: Or capital providers, are essential to the success of a crowdfunding campaign. Interesting research can be performed on the motives of this unique group of investors for participating in crowdfunding. One thing that makes this group so special is that they are not exclusively motivated by earning money.

#### Creators

The reasons for people to fund their projects via crowdfunding are wider than just money. Gerber et al. [4] have, by interviewing people involved in crowdfunding, identified the following motivations:

1. Raise Funds: Almost trivial, one of the motivations is to raise funds. Also, platforms provide a way to collect payments online, and accept small payments from a large number of people. Therefore, capital seekers do not need to develop an infrastructure for it.
2. Establish Relationships and form Connections: In addition to raising funds, one other advantage is the opportunity for a direct connection between creators and funders

potentially extending beyond the campaign itself. The long term relationship stands in contrast to the short term relationship that occurs in many alternatives to crowdfunding.

3. Maintain Control.
4. Learn New Fundraising Skills.
5. Receive Validation and approval: Successful experiences and receiving public recognition of their success increase a person's confidence in his/herself and the project. According to the writers, this finding is consistent with social cognitive theory, which suggests that people build beliefs in their ability through social interactions. This finding is supported by prior research in online communities, which finds that people engage in these communities to build self-esteem.
6. Replicate Successful Experience of Others: According to the researchers, initial findings suggest that people participate in crowdfunding because they want to replicate the success of others. Creators that succeed in funding a project online provide social proof that motivates others to become creators as well.
7. Expands Awareness of Work through Social Media: Findings suggest that creators were motivated to participate in crowdfunding because it expanded their awareness through social media. In one of the interviews in the research, an anthropologist who used the crowdfunding platform RocketHub to fund her research on ancient Roman skeletons, described being motivated to not only share her work publicly but engage in a dialogue about her work. She gained a lot of followers on Twitter and has even started a blog as a result of her newfound fame.

Creators deterrents to Crowdfunding:

1. Inability to Attract Supporters
2. Fear of Public Failure and Exposure
3. Time and Resource Commitment

### **Factors For Successful Crowdfunding**

1. Orientation of the project: Whether the project or the organization behind it is a non-profit or not seems to influence the success of funding. Belleflamme et al. [5] performed an empirical analysis to investigate this. They found that non-profit organizations tend to attract larger amounts of money. According to them, this finding is in line with earlier research stating that non-profit organizations are better at attracting outside funds because of their stronger focus on the social outcome than on monetary gains.
2. Amount and duration: According to that same research [5], increasing goal size is negatively associated with success. Less expected was that an increased duration of a campaign decreases the chances of success. This might be explained as that longer durations are a sign of lack of confidence.
3. Social network: Research by Agrawal et al. [6] indicates the important role that friends and family may play in generating early investment in entrepreneurial ventures. They speculate that this early investment may serve as a signal of entrepreneurial commitment. Later investors may use this signal thereby increasing the likelihood of further funding by way of access to distant sources of capital.

## Funders

Funders are essential to the success of a crowdfunding campaign. Interesting research can be performed on the motives of this unique group of investors for participating in crowdfunding. One thing that makes this group so special is that they are not exclusively motivated by earning money. In crowdfunding, consumers have taken on the role of investors or capital providers. And they are diverse. Even on the same platform, the motives to make investments can greatly vary between consumers. Based on earlier research, Lin et al. [7] have identified a set of motivations that may drive a person's participation in crowdfunding:

1. **Seek Rewards:** At reward-based platforms, capital seekers can offer rewards linked to the size of the contribution by the investor. These rewards range from t-shirts and acknowledgment on the project page to pre-ordering the actual product. The latter sometimes leads to confusion situations for consumers. Although explicitly disclaimed by Kickstarter, many consumers are under the impression that the web site is essentially an online retail storefront in which project creators are pre-selling products [8].
2. **Help Others and support Creators and causes:** One of the interviewees from a study carried out by Gerner et al. [4] stated that he funds an idea that he thinks is neat, but he also really likes the idea of people being able to get off the ground without needing to buy into a big giant corporate structure.
3. **Engage and Contribute to a Trusting and Creative Community:** People generally feel like they are involved or engaged in the project throughout the duration. Crowdfunding allows a lot of people to be involved in something that they maybe otherwise would not have the opportunity to be involved in. Just to be a part of something is what motivates people in those cases [4].
4. **Reputation:** Another motivation for many participants of online and crowdsourcing communities is the reputation benefits and recognition that can be derived from active participation in the community [7]. Fellow investors on Kickstarter can see what projects you backed. This creates a sense of 'high-profile community members'.

One motivation of supporters in crowdfunding communities is the desire to collect external rewards such as an acknowledgment, a tangible artifact, or an experience. An acknowledgment may come in the form of a telephone call, while a tangible artifact could be a CD or gadget. An experience may involve, for instance, meeting with the creator. The creator's goal is to provide rewards that satisfy the supporters' desire to collect.

### 1.1.4 Design Principles

Based on these motivations, Gerber et al. [9] developed three design principles for crowdfunding intermediaries. These principles enhance motivation for these actors to individually decide to become and stay involved in crowdfunding.

## Support Resource Export

Crowdfunding actors should be able to exchange human, information, and financial resources before, during, and after the crowdfunding campaign. The human resources are persons that

can help to fulfill tasks associated with creative production, such as creation, manufacturing, implementation, marketing, planning, and fulfilling. Examples of information resources are information and explanations. Access to informational and human resources has been found to have a direct positive impact on persistence in ambiguous tasks. With financial resources, we refer to funding. Almost all platforms already provide the exchange of financial resources, as this is fundamental to crowdfunding. The exchange of the other types are often overlooked. Adding this to the design could potentially enhance a project's success. During the preparation stage, capital seekers are advised to search for example projects, read advice blogs, seek one-on-one advice, and outsource preparation tasks. However, on many intermediaries, users cannot see unsuccessful campaigns [10].

### **Support community before, during, and after**

Intermediaries could offer users the possibility to interact or meet up before, during, and after the campaign. There should be opportunities to meet up with potential capital providers to increase awareness of the upcoming project before the campaign starts. During the campaign, there should be tools and channels available to promote the project. And when the campaign is successful, there should be a way to keep supporters up-to-date about the execution of the project. According to earlier research, people are more likely to persist when they publically commit beforehand and then share small wins with others throughout the effort. Through the sharing process, they receive positive validation and are more likely to believe they can accomplish a task, are willing to take on more challenging work, have a greater intrinsic motivation to complete a task, persist in the face of challenges, and expend more effort in the task [9]. The community around crowdfunding developed tools to support after-campaign work. Intermediaries could offer a matchmaking service to bring together supply and demand [7].

### **1.1.5 Objectives**

So one of our main objectives is to create a platform that can improve our user's experience and boost the chance of our creators to reach the maximum of the audience as well as facilitate the access and the funding process for our funders, and provide a place where they can build there trust and look for promising talents and empowering entrepreneurs to help in creating new possibilities.

Firstly, we need to make sure that the code is maintainable for future alterations and additions by other programmers. For this, we will try to keep the files organized in a logical setup and we will comment our code where necessary, to understand its workings.

Secondly, « Sahem » platform will deal with sensitive data on several occasions. First of all, there is the user data, which includes personal information and passwords. Secondly, there is a payment system that involves sensitive financial information of the users. This is why it is very important to build our system to be as secure as possible.

Thirdly, the platform needs to have an attractive and clean design. It needs to be user-friendly, easily navigable, and look professional so that users will trust the platform is capable of performing well in the financial setting. So I came up with a unique design, and tried to create a new experience for the users, fast, smooth, mobile responsive, and easy interface to interact with.

1. **User Experience:** Making sure that our crowdfunding software solutions are easy to navigate for the end-user is a crucial part of the development process. If our customers are befuddled by how to navigate our application, then it is highly likely that they will leave and find another crowdfunding platform. Not only will we want a functional interface, but one that is eye-catching as well. One thing to take note of is the importance of giving them the terms and conditions within the first few steps, so they fully understand what to expect. This shows a dedication to transparency that many startups and entrepreneurs will appreciate.
2. **Account Management:** Our customers will want to know what is going on, and making it easy on them will help make our platform successful. That means setting up systems that make it very clear what's going on with their project. We will want them to be able to access who has been investing, how much money they have, how far they are from their goal, and any other metric they need to run a successful campaign. This could also include reports for recharges, withdrawals all available via a simple to navigate dashboard.
3. **Report Generation:** As the platform owner, you need a way to benefit from your time spent on creating this site that will help so many people. So, ensuring you also have access to backed reports like rewards, investors, and such can help you help them, as they say. This means creating a dashboard that, just like for the actual campaign creators, is easy to navigate and gives you access to reports you can use to course-correct and upgrade the systems.
4. **Payment Gateway & Marketing:** When starting a crowdfunding platform you will want to set it up with access to the right payment gateways. Each gateway has its features, and so doing some in-depth research into them will allow you to choose one or several that works for the largest number of potential customers.

## 1.2 Functionality

The platforms operate by allowing those seeking finance to make a pitch on the site outlining how much money they need, what they need the money for, if anything, you get in return for contributing. Potential funders can then view pitches on the platform, interact with both those looking for finance and other potential funders, and then decide whether or not they want to back the campaign. The majority of platforms operate the all-or-nothing model where, if the target amount is not raised within a given timeframe, contributions are returned to funders and no financing goes ahead, but we are using the Keep It All model, anything you raise is yours.

When a new user arrives at the landing page he will be greeted with the main interface of the application, that holds multiple pieces of information and presentations of our platform, it explains the basic functionalities and the different advantages of our application.

Our simple design, based on some principles of improving the user's experience found in Krug's book "Don't make me think, revisited: a common sense approach to Web usability", we have chosen to make the most important buttons into unique color from the others, on normal cases the user will click on one of them to navigate to the appropriate page, at first he will have the ability to navigate to the existing fundraiser campaigns of the platform, and if he chooses to fund the project, he will be redirected to the Registration form in order to authenticate, and fill his personal profile, and provide the platform with his credit card information.

After the authentication, he will have the ability to create and submit his own fundraiser.



In his profile, he can see his personal information and get to show all of his achievements, the projects he funded, and the fundraisers he created as well as the posts he published in the platform.

## 1.3 Feasibility

This project is technically and financially feasible. I will spend approximately nothing to host the app, and in the future, there is an option to pay as you grow, by generating income from cutting a % from the funds at the end of each fundraiser, we will get the costs covered. With regards to the technical feasibility, it will be built in well-established MEAN Stack technologies, that have a low cost with great scalability, and maintainability.

« Sahem » provides the following features:

1. Trust-building through the corroboration mechanism.
2. Opportunity for long term commitment between Creators and Funders.
3. Uses secure technologies on both the frontend and the backend, and any requests sent by the users are passed by multiple middlewares on the backend.
4. Easy Navigation through simple user interface design built using Angular, a framework that helps in creating fast and reliable Single Page Applications.
5. A headless backend, which allow us to create different frontend application afterward, and that are linked to the same backend.
6. Global crowdfunding interaction, no limits between countries, everyone can fund or create fundraisers.

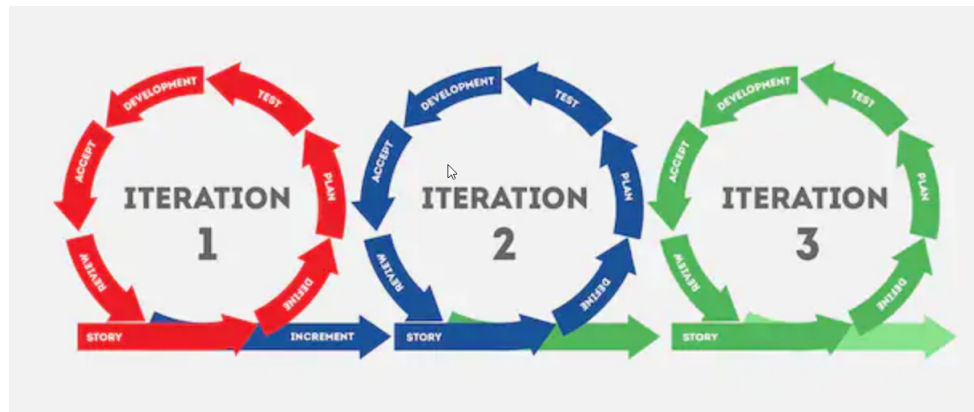
## 1.4 Project Management

Process-driven software development is based on rigorously defined activities and tasks that are also repeatable and measurable. Formal processes facilitate planning, analysis of requirements from multiple angles, design of high-quality software models by following standards and using team-based tools, and incorporation of quality through walkthroughs, inspections, and testing. As a result, such formal processes enhance quality and maximize user benefits.

### 1.4.1 Project Development Process

The process of discipline is complex. This is because a process considers myriad different hard and soft factors that impact development. Many software developers argue that processes restrict their creativity. Far from that, processes enable creativity with value. This is because processes ensure the effort made by architects, designers, developers, and testers will be well directed toward the commonly agreed goals (business objectives) of a project. Processes also facilitate measures and metrics that indicate individual and team productivity and quality. Metrics and measurements in software projects also enable the assignment of responsibilities and accountabilities. To develop a good product we need a good development method. One very popular agile framework for this is iterations and increments.

The iterations and increments as shown in Figure 2 are the basis of most modern-day approaches to developing good software. In this iterative and incremental approach, no deliverable is produced in a single attempt. Instead, at least three iterations (repetitions) are undertaken before producing a deliverable. This is followed by incrementally adding another package, which would have its own three or more iterations. The three terms iteration, incremental, and parallel are further discussed next.



**Figure 2.** *Iterative process project life cycle*

### Iterative

The iterative aspect of a process enables the repetition of tasks. As a result, the deliverables are produced gradually. For example, when a use case is iterated, additional material is added to the description of the use case—such as alternative flows within the use case. The iterative approach encourages a slow and steady philosophy rather than hurrying and finishing up a deliverable in the first attempt. Deliverables are gradually matured by undertaking at least three iterations across multiple other deliverables. For example, while following an iterative process one might move from an initial use case to another use case in another diagram, then identify classes and draw a sequence diagram before coming back to the original use case and completing it.

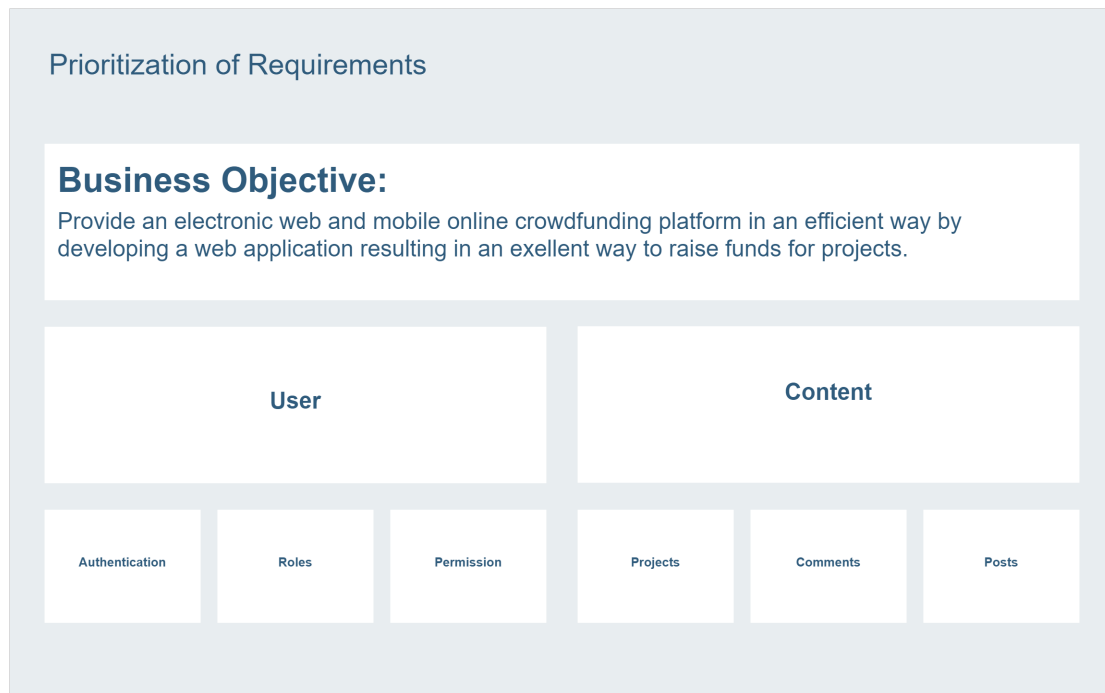
### Incremental

The incremental aspect of a process enables adding new elements and diagrams to an existing deliverable. An example is to add new packages to existing or developing packages. New requirements are thus discovered and modeled incrementally. This incremental aspect of the process enables the creation of parts of a system in as complete a manner as possible before proceeding with the development of additional parts of the system. The incremental aspect of a process often goes hand in hand with the iterative aspect. For example, while a new deliverable is incrementally added (a new use case), an existing deliverable is iteratively augmented during a later iteration (e.g., additional steps added to a use case).

## 1.4.2 Monitoring And Planning

### Monitoring

Breaking down a project into subparts does help a lot in enabling controlled the execution and monitoring of the project, it helps with keeping the project well displayed and well planned to avoid all the mishaps spaghetti code afterward. Figure 3 shows how we divided the business objective into subparts. To arrive at acceptable performance criteria of the system, the BO gets further divided into smaller parts or subject areas.



**Figure 3.** « *Sahem* » Business Objectives

### Planning

During the planning we should provide precise routing of one or more business processes with opportunities to optimize on time and costs associated with the processes. So we created a Gantt Chart in teamGantt to help us follow the progress of different tasks and the time left to complete them:

## Sahem

### Architectural Space Modeling

Project Design Pattern

MVC

Controller

Express

Model

MongoDB

View

Angular

### Crowdfunding

Crowdfunding: Why people are moti...

The Economics of Crowdfunding Plat...

Fighting for Funds: An Exploratory St...

## Sahem

Business Objectives

Deviding BO into smaller BOs

### Problem Space Modeling

Actors

Use Cases

Activity

### Solution Space Modeling

Class Diagrams

Sequence Diagrams

Data Modeling

## Backend

Authentication

Roles

Permissions

Projects CRUD

Comments CRUD

Votes

Stripe

## Frontend

UI Design

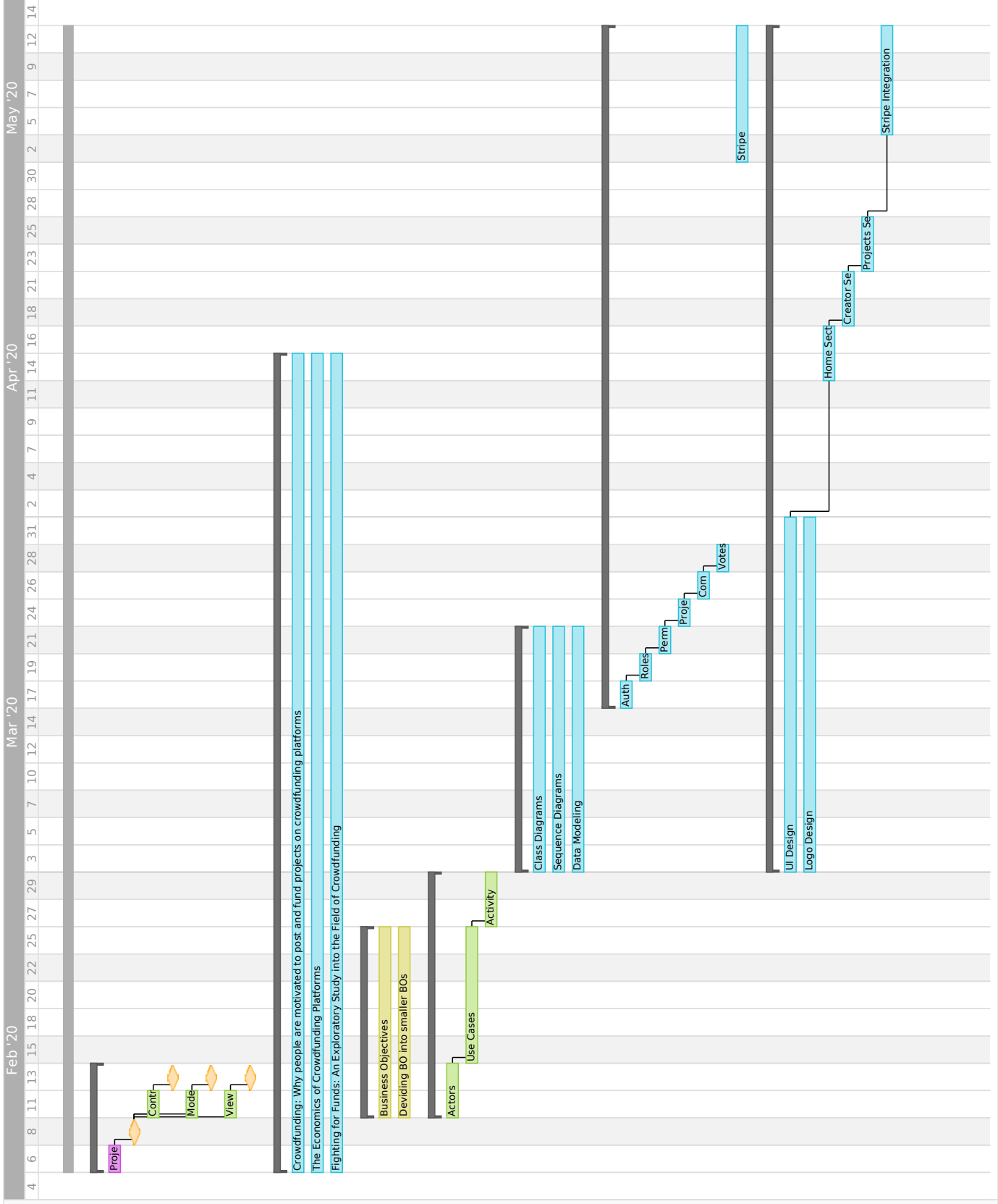
Logo Design

Home Section

Creator Section

Projects Section

Stripe Integration



## Chapter

**2****Analysis And Design****Contents**

|            |  |           |
|------------|--|-----------|
| <b>2.1</b> | <b>General description of the platform . . . . .</b> | <b>17</b> |
| <b>2.2</b> | <b>Specification of requirements . . . . .</b>       | <b>18</b> |
| 2.2.1      | Requirements . . . . .                               | 18        |
| 2.2.2      | Identification of actors . . . . .                   | 19        |
| 2.2.3      | Identification of objectives and use cases . . . . . | 19        |
| <b>2.3</b> | <b>Functional Analysis . . . . .</b>                 | <b>21</b> |
| <b>2.4</b> | <b>Dynamic Analysis . . . . .</b>                    | <b>23</b> |
| <b>2.5</b> | <b>Structured Analysis . . . . .</b>                 | <b>27</b> |

Among all the stages of the product's life cycle, the design stage contributes significantly to the improvement of competitiveness, because it permits reduction of costs and often, shortening of the time necessary to get the solutions to the production stage.

**2.1 General description of the platform**

« Sahem » is a unique crowdfunding app built primarily in MEAN Stack technologies. It has one type of user that can be two types at the same time: can be either a creator or a funder. A funder donates while a creator acquires the fund. This app has a trust-building mechanism called corroboration where funders can certify creators and allow creators to certify their claims through uploading documentary proofs in an image format. The final mode of crowdfunding takes place through the exchange of transactional information where Funders can pay an amount of money by a bank transfer or by card. We took the chance in providing easy accessibility to the main functionalities of the platform, as we tried to ask what is the most thing that a user wants to do, and we tried to optimize the number of clicks that a user would have to make in order to do the actions he wants, so we sought to make both of the creation of a fundraiser, and funding a project accessible by a maximum of 3 clicks and after the creator starts a fundraiser he can directly share a link that can lead to the funding page, bypassing the project description and content, to help in making the funders fund in a very short time, instead of filling unnecessary forms and passing by multiple pages.

## 2.2 Specification of requirements

We will proceed according to the UML method which consists of identifying and modeling the various business processes in order to easily migrate to an object architecture from a static and dynamic point of view. This analysis presents a total abstraction being independent of any technology or implementation.

The specification of the needs will allow us to have a better approach of the users, the functionalities, and the relationship between the two. It will be in the form of a use case. For this we will proceed as follows:

- Identification of requirements.
- Identification of the actors of the system.
- Identification of use cases.
- Description of use cases.
- Grouping of use cases into packages.

### 2.2.1 Requirements

The requirements that we have put on the end-product were divided into basic functionalities and optional functionalities. The User's basic functionalities are:

1. Account management: Create an account, login, profile, settings, personal info, personal dashboard.
2. Start a campaign: send in a proposal for a Fundraiser (webform).
3. Share function: functionality so that a campaign can be shared within social networks, such as Facebook, Instagram, LinkedIn, Twitter, and potentially Pinterest and Google.
4. Payment function: Funders should be able to pay their ticket through Credit Card.
5. Review function: Funders should have the opportunity to give reviews about and react to events that have taken place.
6. Countdown timer: a timer that shows how much time a campaign has left until it closes.
7. Current overview: each campaign should show for example the number of tickets sold/amount of investors, the amount of money raised so far and the goal amount to be raised.

The User's optional functionalities are:

1. Contribution invitation: functionality to send out concrete requests to contribute a certain amount within the personal social network.
2. Personalized recommendations (recommender system): the system will advise, suggest, or give concrete offers. Logged-in visitors see a personalized recommendation based on social profiles and the platform's web page content and actions.
3. Share option via Whatsapp
4. Online helpdesk: chat possibility with a helpdesk worker.

5. Communication tools: communication possibilities with other members and/or the organization of an event via chat.
6. Connection with social media: e.g. Facebook application for special opportunities and/or campaigns that enabled even more interaction with the target group.
7. Social login: login with a social account.
8. Wallet: deposit money in your wallet, which enables you to make several contributions by paying with the money in your wallet.
9. Recruit function: a tool to gather more contributors. A contributor can share a campaign within their own social networks.
10. Notifications: reminders and/or messages related to upcoming events and updates of events.
11. Mobile App: extra functionalities through the use of a camera and GPS on the actual event: e.g. picture/video upload to the website or live visitor numbers.
12. Agenda/calender: an overview that shows which events will take place on which dates (including events that have already taken place).
13. Newsletter: send an overview of the events to members of the platform community or a certain selection.
14. Promotion function: functionality to share and promote events on different levels: community (everyone), events (only participants of a specific event), social media (within social networks).
15. Campaign management: management tools for the event organization.
16. Reports/statistics: data analyses, tools for advanced analysis of the web page visits.

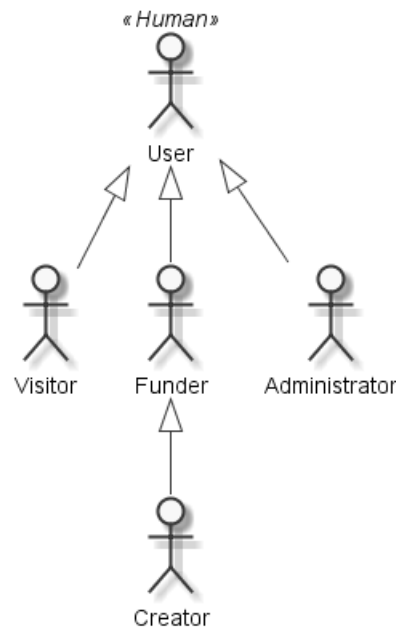
### 2.2.2 Identification of actors

In UML we don't use the term users but actors. A player in a system is an entity external to this system that interacts (data entry, reception of information, etc.) with it. The actors make it possible to define the interface that the system will offer to its environment. And also the actor groups together several users who have the same role. And to define actores it will be necessary to identify their roles. To facilitate identification, we can imagine this: everything that is outside and interacts with the system is an actor, everything that is inside is a functionality to be realized. The different actors of the studied system are as Figure 4 shows:

1. Creator.
2. Funder will be merged with Creator.
3. Administrator.
4. Visitor.

### 2.2.3 Identification of objectives and use cases

A use case described in the form of actions and reactions, the behavior of a system from a user point of view. It must add value to the actor concerned. Each use case contains a list of functionalities that will be detailed in the following section.



**Figure 4.** « Sahem » Platform's Actors

We will first determine the objectives that will allow us to deduce the use cases and all this in Table 1:

| Management     | Functionalities      | Sub-functionalities                         |
|----------------|----------------------|---|
| Administration | Authentication       | Authentication                              |
|                |                      | Registration                                |
|                | User Maintenance     | Add User                                    |
|                |                      | Modify User                                 |
|                |                      | Delete User                                 |
|                |                      | Consult the information of a user           |
|                |                      | Show the list of users                      |
|                | Roles                | Show users by roles<br>Modify a user's role |
|                | Statistics           | Access the statistics of the plateforme     |
| Fundraiser     | Creator's Monitoring | Personal information modification           |
|                |                      | Access a creator profile                    |
|                |                      | Desactivate an Account                      |
|                | Funds Monitoring     | Fund a project                              |
|                |                      | Confirm funding a project                   |
|                |                      | List of previous funds                      |
|                | Projects Monitoring  | Create a project                            |
|                |                      | Edit a project                              |
|                |                      | Delete a project                            |
|                |                      | Like a project<br>Share a project           |



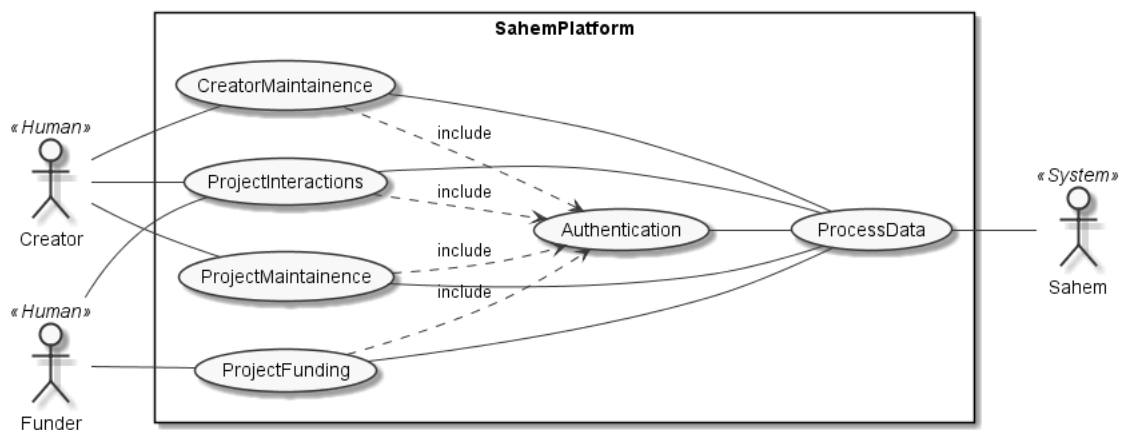
|         |                |                              |
|---------|----------------|------------------------------|
|         | Search         | Comment on a project         |
|         |                | Search a project by name     |
|         |                | Show Fundraisers by category |
| Payment | Billing        | Show the bill                |
|         | Online Payment | Choose payment method        |

**Table 1.** *Identification of objectives and use cases*

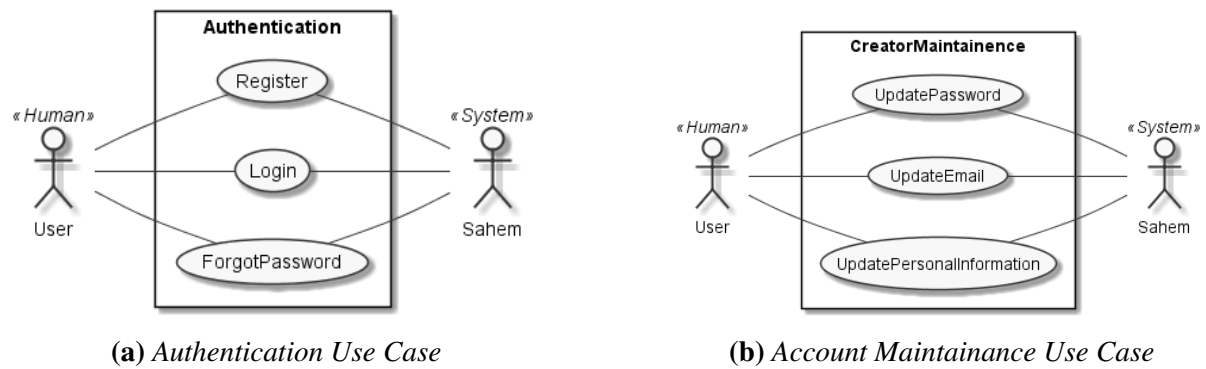
## 2.3 Functional Analysis

Functional analysis is a methodology for analyzing the mission and performance requirements of a system, and translating them into discrete activities or tasks which must be performed by the system. Determination of the system's functionality is the first step in obtaining a conceptual view of a system that is to be designed.

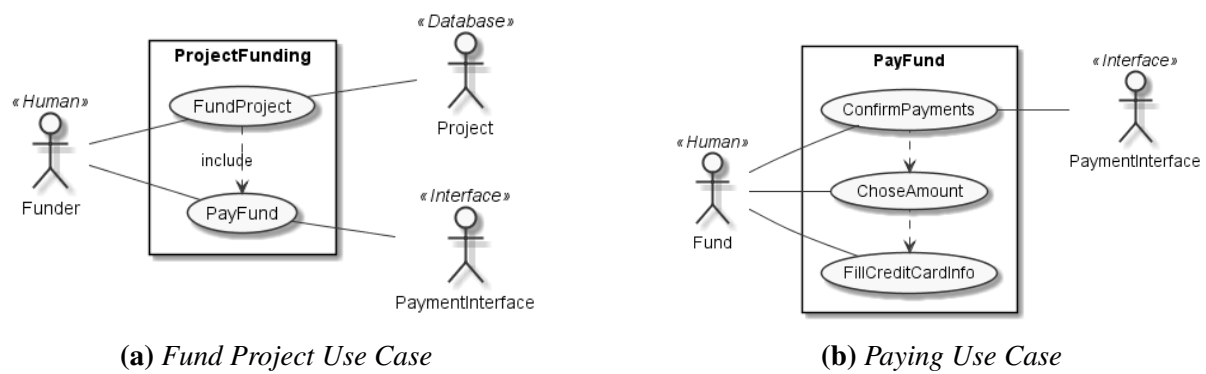
Figure 5 showcase the general use case of a user in « Sahem » System. And is more detailed in Figure 7, Figure 8, and Figure 6.



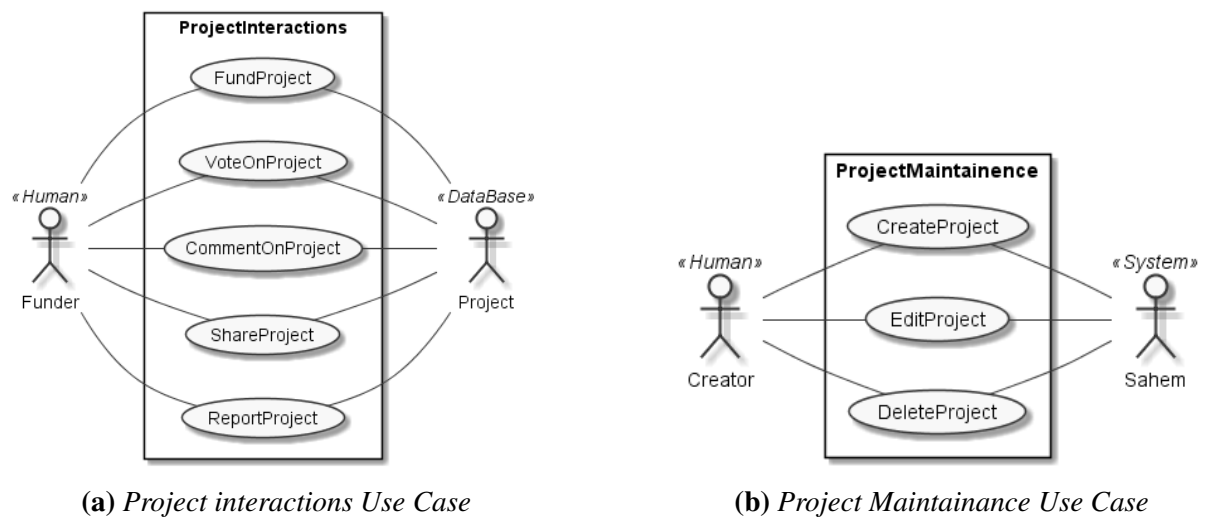
**Figure 5.** « Sahem » System Global Use Case



**Figure 6.** User's Authentication and Account Management



**Figure 7.** Funding Projects Use Cases

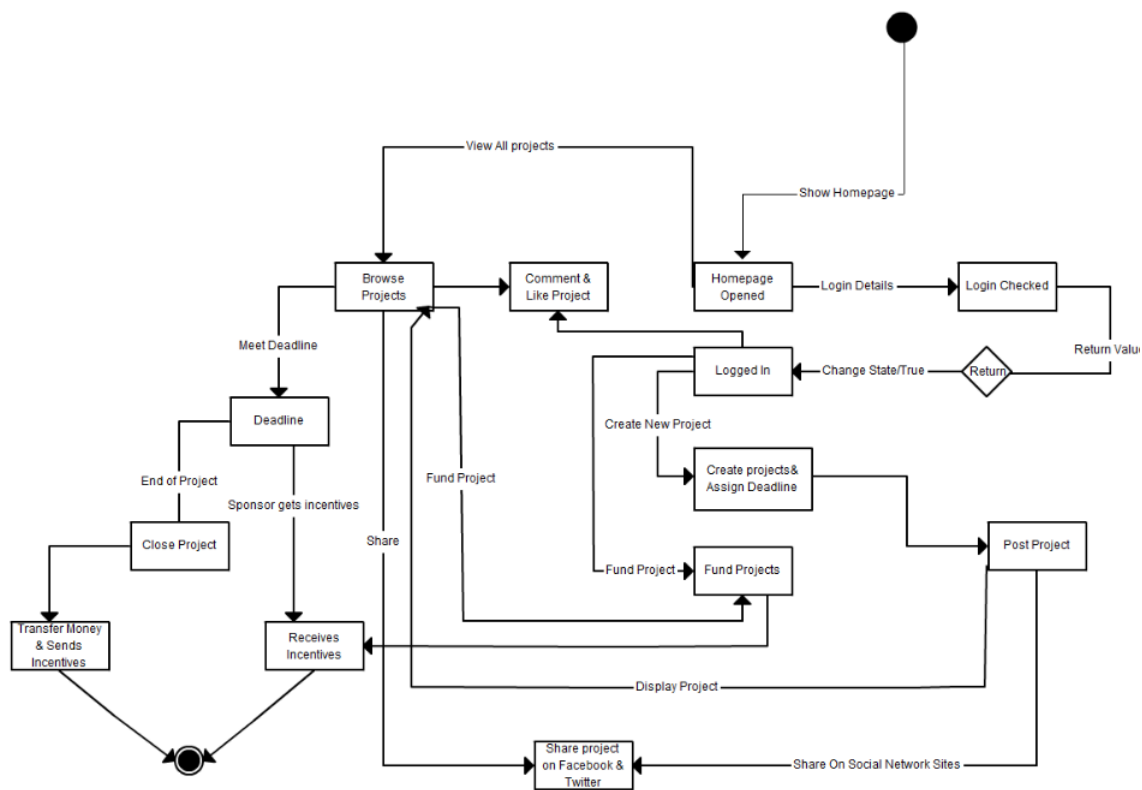


**Figure 8.** User's interactions with projects Use Cases

## 2.4 Dynamic Analysis

Dynamic analysis is the testing and evaluation of a program by executing data in real-time. The objective is to find errors in a program while it is running, rather than by repeatedly examining the code offline. By debugging a program in all the scenarios for which it is designed, dynamic analysis eliminates the need to artificially create situations likely to produce errors. Other advantages include reducing the cost of testing and maintenance, identifying and eliminating unnecessary program components.

First of all, we start by creating an activity diagram, that summarizes the life cycle of all the user's interactions with the « Sahem » system (Figure 9), then we created different Sequence diagrams modeling the different functionalities of « Sahem » system and the interactions between its components (Figures 10, 11, 13, and 12).



**Figure 9.** « Sahem » Activity Diagram

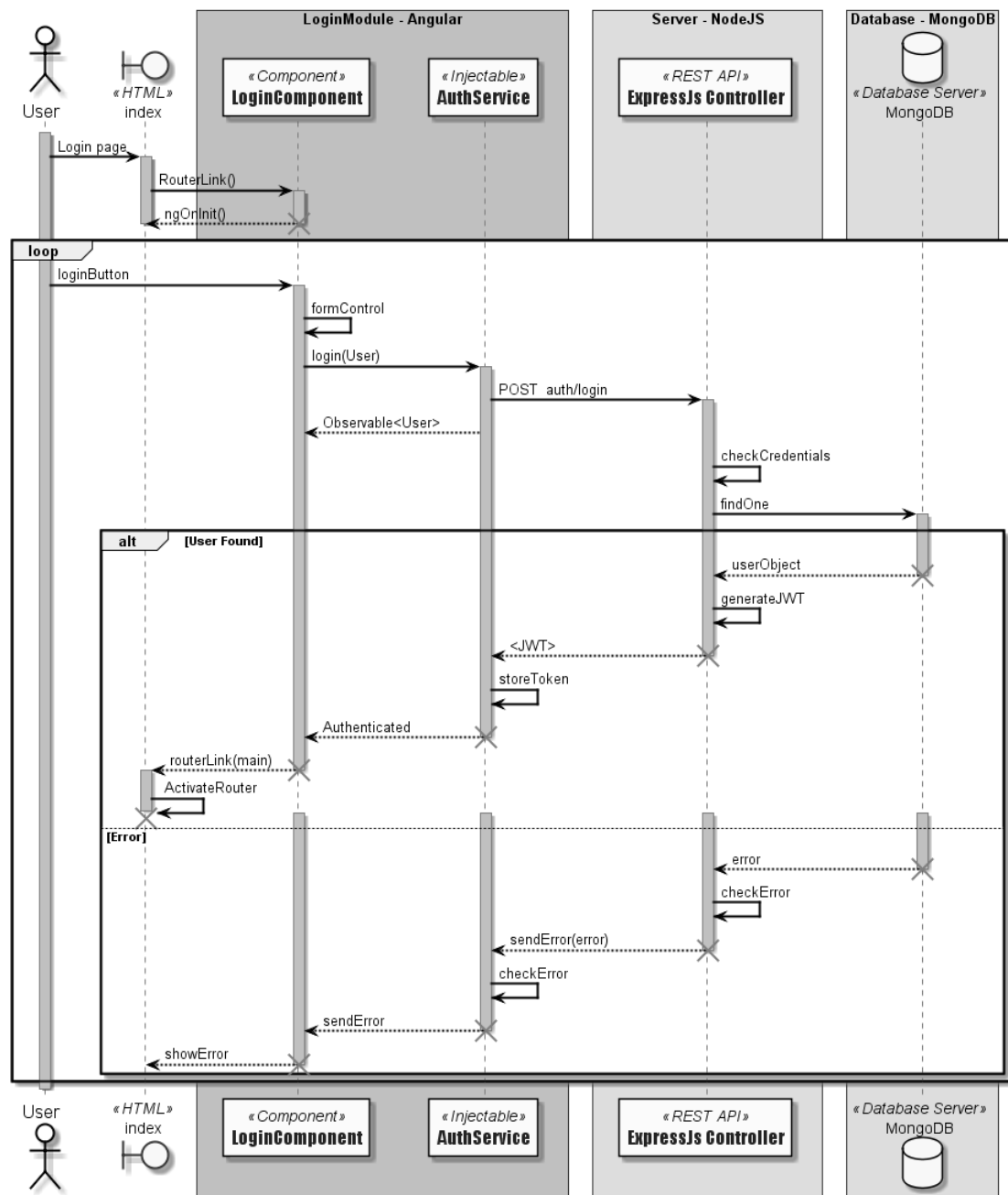


Figure 10. User Authenticating Sequence Diagram

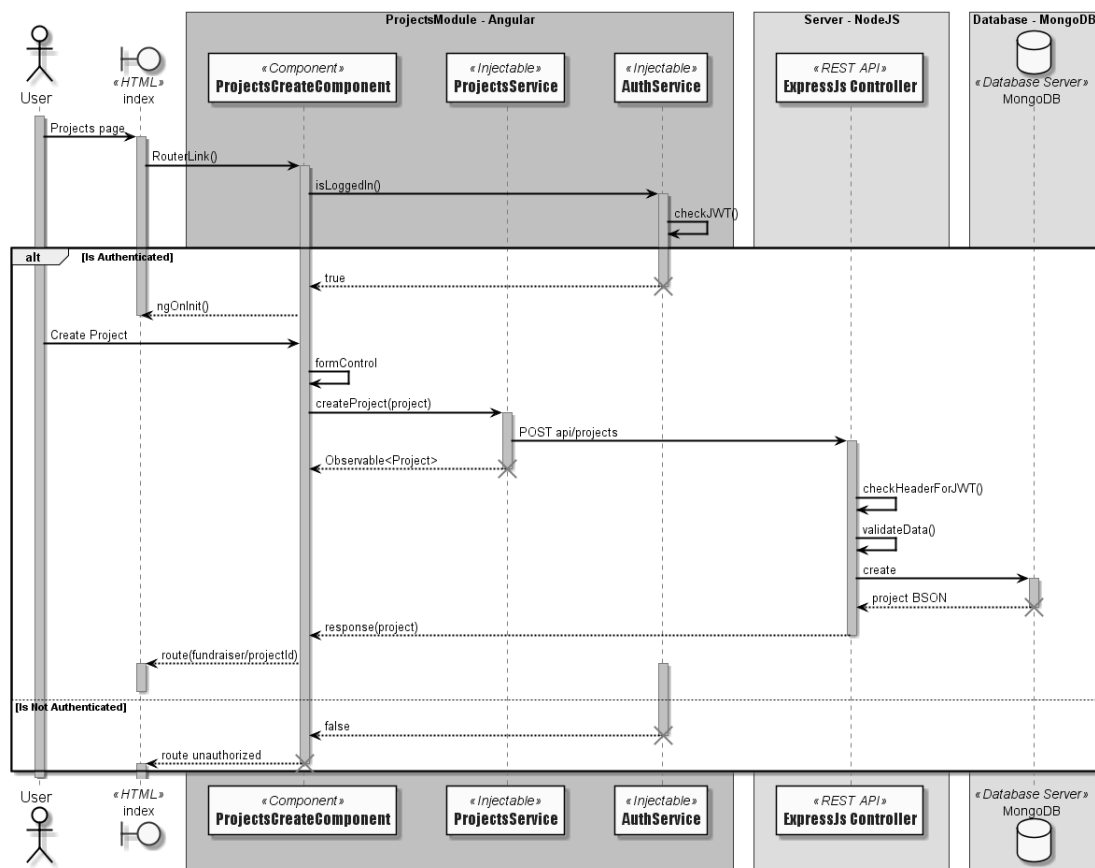


Figure 11. Create a Project Sequence Diagram

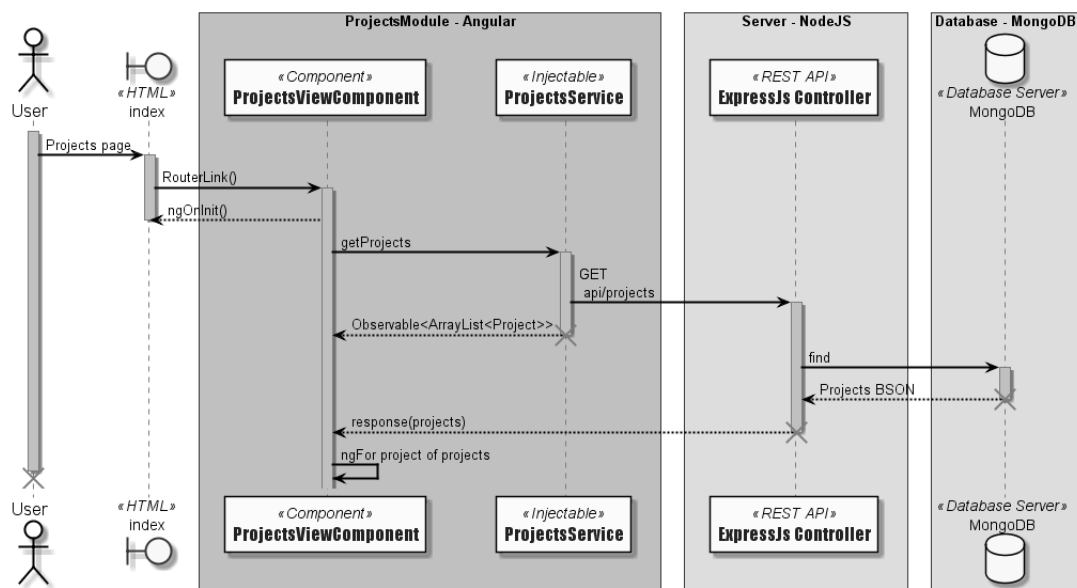
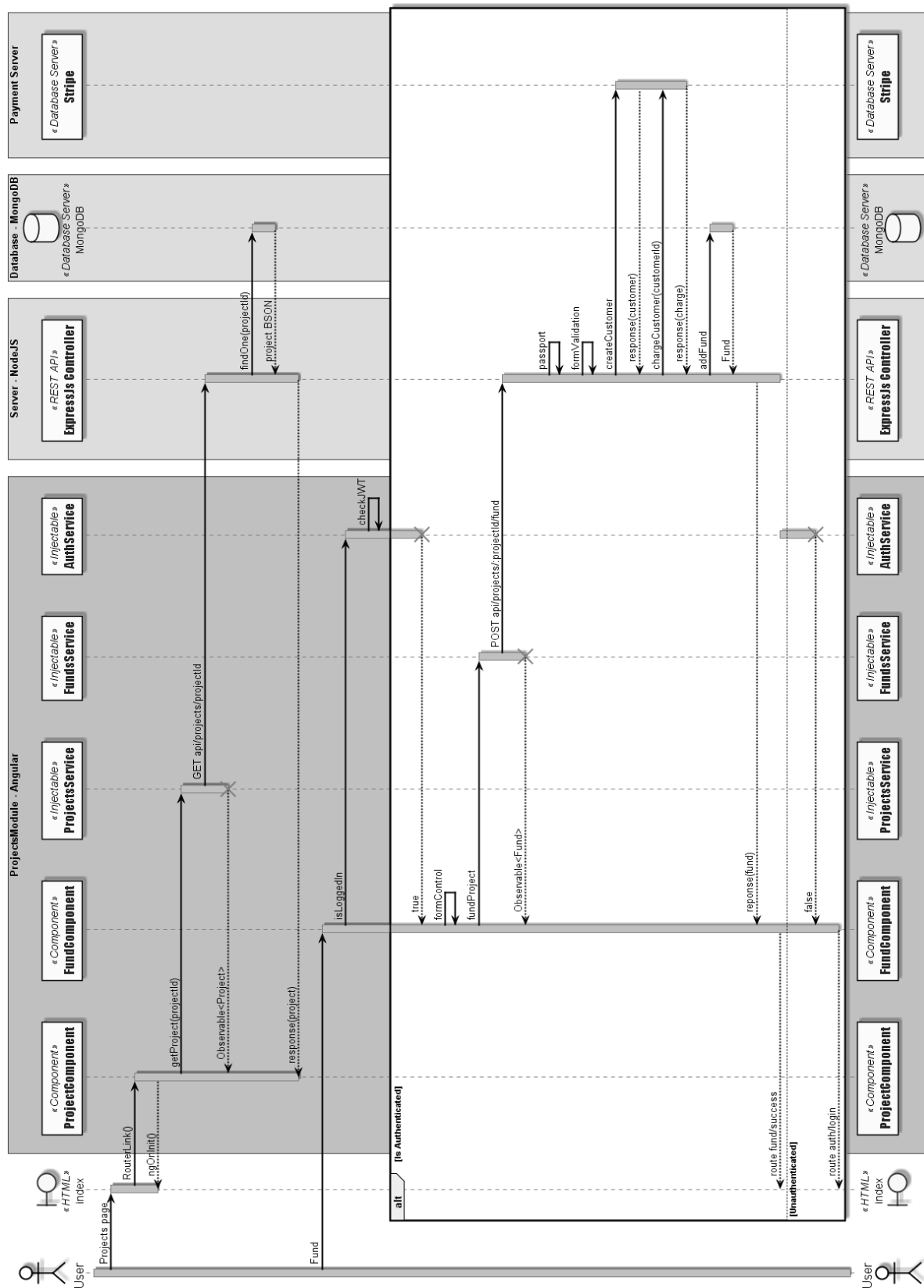


Figure 12. Get Projects Sequence Diagram

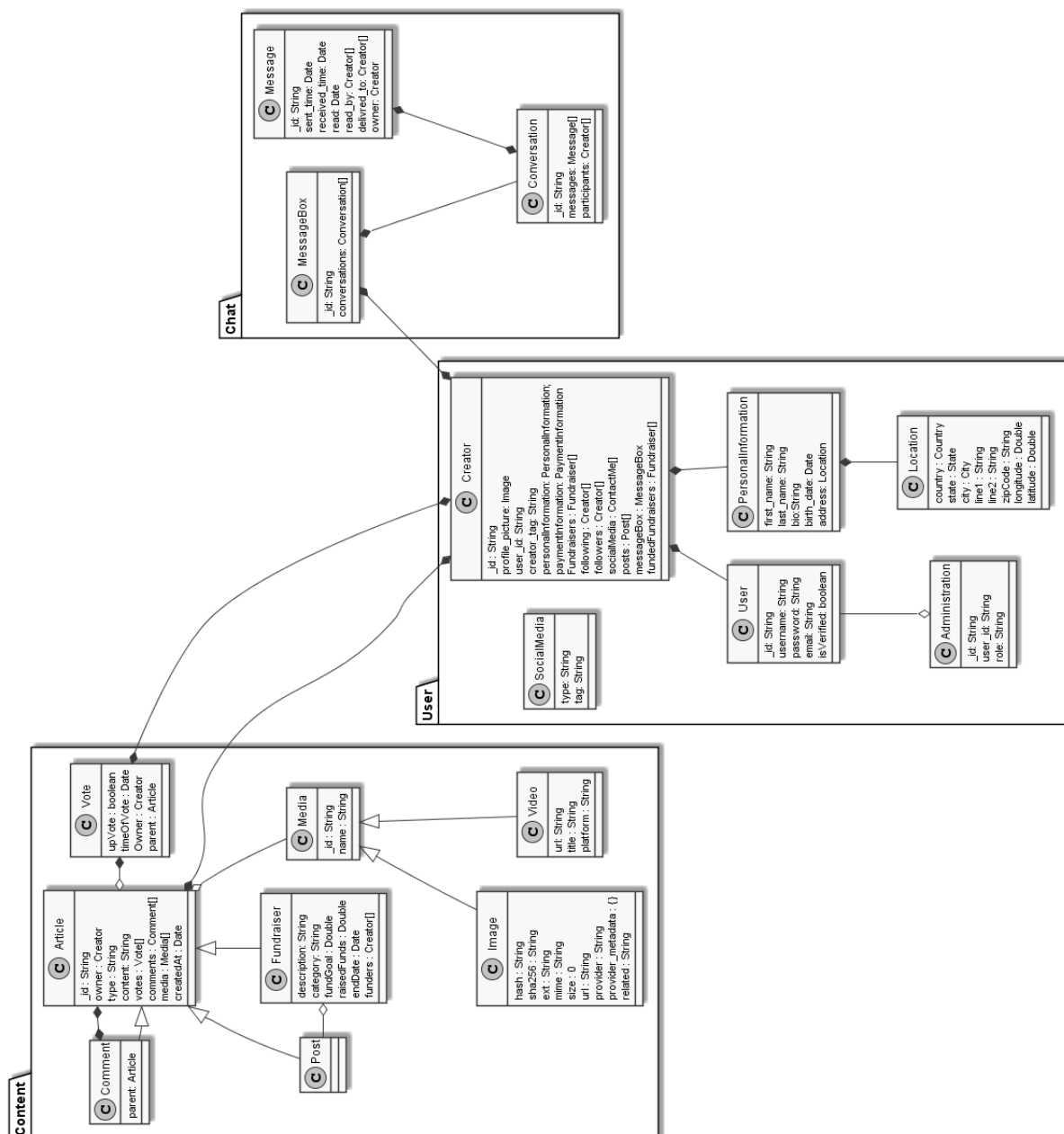


**Figure 13.** *Get And Fund Project Sequence Diagram*

## 2.5 Structured Analysis

Structured Analysis is a development method that allows the analyst to understand the system and its activities logically.

It is a systematic approach, which uses graphical tools that analyze and refine the objectives of an existing system and develop a new system specification that can be easily understandable by the user.



**Figure 14.** « Sahem » Class Diagram

## Chapter

**3****Technical And Environmental Study****Contents**

|            |  |           |
|------------|--|-----------|
| <b>3.1</b> | <b>Capture of technical needs . . . . .</b>    | <b>28</b> |
| 3.1.1      | Capturing Software Specifications . . . . .    | 29        |
| <b>3.2</b> | <b>Adopted Software Architecture . . . . .</b> | <b>30</b> |
| 3.2.1      | Software Architecture Pattern . . . . .        | 31        |
| 3.2.2      | Codebase Architecture . . . . .                | 31        |
| 3.2.3      | Controller Architecture . . . . .              | 31        |
| 3.2.4      | Cloud Architecture . . . . .                   | 32        |
| <b>3.3</b> | <b>Frameworks . . . . .</b>                    | <b>32</b> |
| 3.3.1      | Front-End . . . . .                            | 33        |
| 3.3.2      | Back-End . . . . .                             | 34        |
| <b>3.4</b> | <b>Programming Languages . . . . .</b>         | <b>35</b> |
| 3.4.1      | Conception . . . . .                           | 35        |
| 3.4.2      | General . . . . .                              | 35        |
| 3.4.3      | Front-End . . . . .                            | 35        |
| 3.4.4      | Back-End . . . . .                             | 36        |

**3.1 Capture of technical needs**

The capture of technical needs identifies all the constraints and the choices dimensioning the design of the system.

The tools and materials selected as well as the management of integration constraints generally conditioning the prerequisites of general architecture.

The capture of technical needs is as follows:

- Capturing software specifications.
- Capturing specifications related to hardware configuration.



### 3.1.1 Capturing Software Specifications

These are technical features that the system will provide to the user regardless of the business or functional term.

We will, therefore, list a list of technical functionalities that our system will be able to provide and offer to users, which is why we have introduced the concepts of operator and technical use cases:

#### a) Technical actors

Also called "technical actors" of the system, these are the actors who benefit from the technical functionalities of the system:

- User: These are the people who use the system functionality in one way or another.
- Administrator: this is the person responsible for managing all the management of the platform as well as that of the users.

#### b) Identification of technical use cases

| Aspects             | Use Cases           | Sub-Use Cases                                |
|---------------------|---------------------|--|
| Security management | Authentication      | Register                                     |
|                     |                     | Login  |
|                     |                     | Reset Password                               |
|                     |                     | Change password                              |
|                     |                     | Logout                                       |
|                     | Confidentiality     | Role management                              |
|                     | Secure payment      | Send Payment                                 |
|                     |                     | Receive Payment                              |
| Data management     | Backup and restore  | Save the database                            |
|                     |                     | Restore the database                         |
|                     | Dynamic data        | Dynamic data display                         |
|                     | Integrity           | Simultaneous updates                         |
|                     | Interoperability    | Network deployment                           |
|                     | Compatibility       | Run on different browsers and platforms      |
|                     |                     | Get all the user created projects fundraised |
|                     | History             | Get all the user created projects fundraised |
| Help management     | Statistics          | Display platform statistics                  |
|                     | Fund                | Fund migration from anonymous user           |
|                     | Gestion des erreurs | Logging Viewing types of errors              |

|                    |                         |  |
|--------------------|-------------------------|--|
|                    |                         | Send the error by email to the administrator               |
|                    | Aide en ligne           | Consult the user help Guide                                |
|                    | Notifications           | Confirmation or error message                              |
|                    | Mails                   | Sending emails   |
| Payment management | Multi-currency platform | Choose a currency for purchase and payment                 |
|                    | Payment method          | Possibility to choose a payment method                     |
| Optimization       | Navigation              | Nested operations  |
|                    | Sharing                 | Possibility to share your donation                         |
|                    | Theming                 | Possibility to switch themes between dark and light themes |

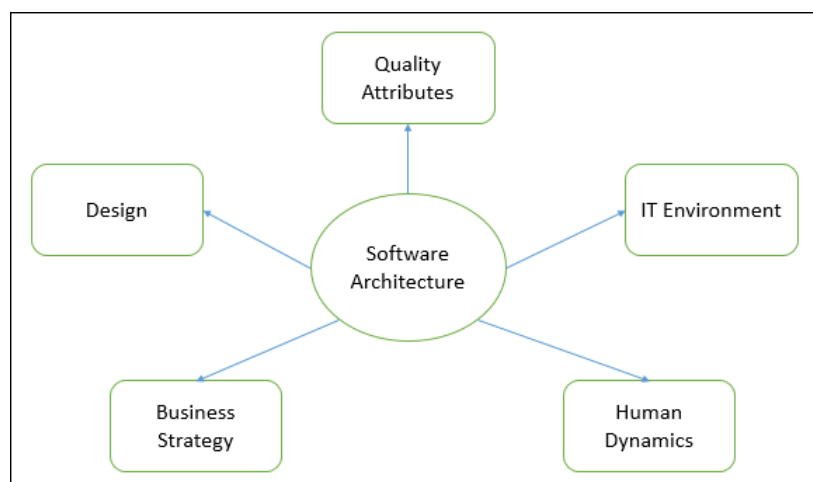
**Table 2.** *Identification of technical objectives and use cases*

## 3.2 Adopted Software Architecture

The software architecture of a system describes its major components, their relationships, and how they interact with each other.

It essentially serves as a blueprint. It provides an abstraction to manage the system complexity and establish communication and coordination among components.

It helps us to determine how we should handle the codebase of our application, the technology to go with, and what cloud architecture is best for our application.

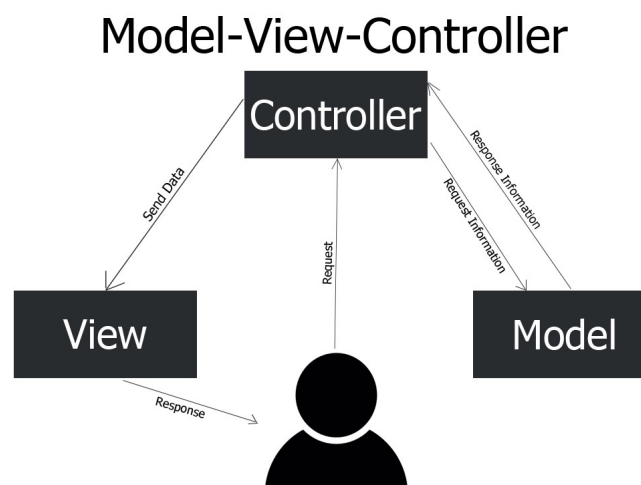


**Figure 15.** *Software Architecture Factors*

### 3.2.1 Software Architecture Pattern

The Model View Controller Model View Controller (MVC) design pattern specifies that an application consists of a data model, presentation information, and control information. The pattern requires that each of these be separated into different objects.

The MVC architecture is a software architectural pattern in which the application logic is divided into three components on the basis of functionality. These components are called: Models - represent how data is stored in the database Views - the components that are visible to the user, such as an output or a GUI Controllers, the components that act as an interface between models and views



**Figure 16.** *Model View Controller*

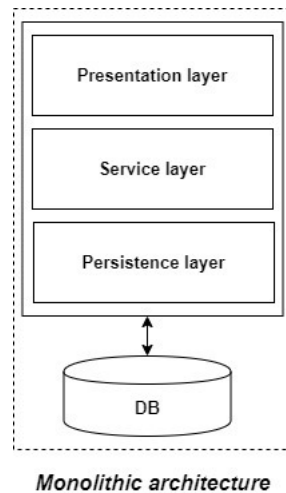
### 3.2.2 Codebase Architecture

Monolithic applications fit best for use cases where the requirements are pretty simple, the app is expected to handle a limited amount of traffic.

If all the functionalities of a project exist in a single codebase, then that application is known as a monolithic application. We all must have designed a monolithic application in our lives in which we were given a problem statement and were asked to design a system with various functionalities. We design our application in various layers like presentation, service, and persistence and then deploy that codebase as a single jar/war file. This is nothing but a monolithic application where “mono” represents the single codebase containing all the required functionalities.

### 3.2.3 Controller Architecture

Representational State Transfer Representational state transfer (REST), is an architectural style for providing standards between computer systems on the web, making it easier for systems to communicate with each other. REST-compliant systems, often called RESTful systems, are characterized by how they are stateless and separate the concerns of client and server.



**Figure 17.** *Monolithic Applications*

### 3.2.4 Cloud Architecture

Software as a service (SaaS) , which allows us to set up services, such as application servers and databases, as needed to create the apps we need. This type of system typically works in a cloud or hybrid cloud environment, which often makes provisioning of servers and software as simple as entering some configurations and clicking a button.

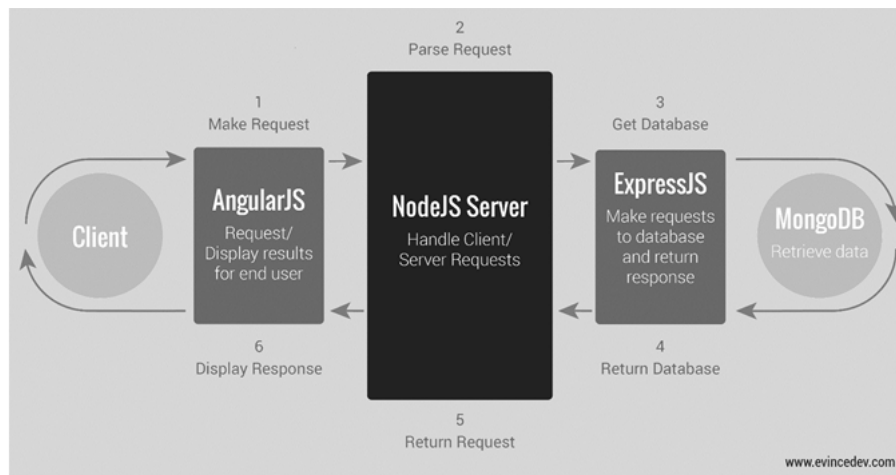
This type of setup is perfect for the MEAN stack in the deployment stage, as each piece of the MEAN stack is easily placed into the cloud and thus can be easily provisioned, so they are widely available on PaaS services and can be used quickly when needed.

## 3.3 Frameworks

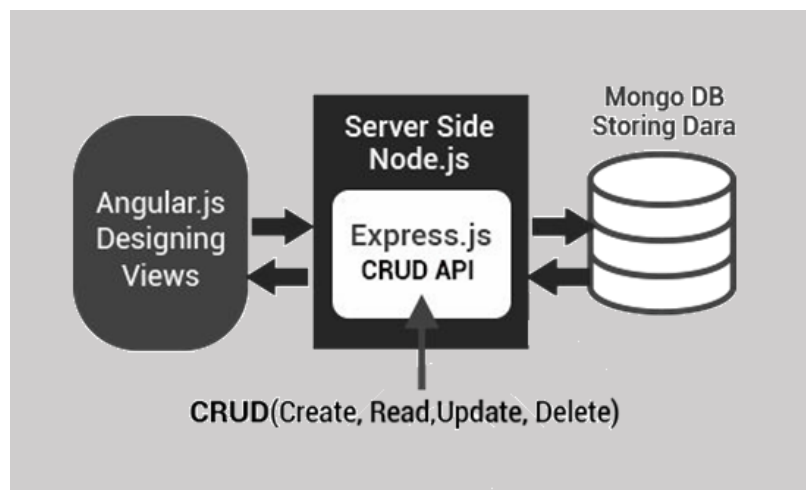
### MEAN stack

While the name sounds like “mean”, it stands for the software pieces that are used to create a particular development stack: MongoDB, ExpressJS, Angular, and NodeJS. One of the biggest advantages of using this particular development stack is the ability to allow developers to use one consistent data model across the stack, using JavaScript Object Notation (JSON) and BSON (for MongoDB). This allows for quick transitions between the various pieces of the stack, especially when a single programmer has to handle more than one portion of the stack.

1. When a client makes any request, it is firstly processed by the Angular.
2. After that, the Request enters phase 2 which is NodeJS.
3. After that Request enters phase 3, and ExpressJs it send a request to the database.
4. After that, MongoDB retrieves the data and return the response to the ExpressJs.
5. Then ExpressJs return response to the NodeJS and then NodeJS return it to the AngularJS to display the result.



**Figure 18.** *The Flow In Mean Stack*



**Figure 19.** *Communication In Mean Stack*

- Create (POST) Make something
- Read (GET) Get something
- Update (PUT) Change something
- Delete (DELETE) Remove something

### 3.3.1 Front-End

#### 3.3.1.1 Single Page Application

Single-page application Single Page Applications (SPA) is an app that works inside a browser and does not require page reloading during use. SPAs are all about serving an outstanding UX by trying to imitate a “natural” environment in the browser, no page reloads, no extra wait time. It is just one web page that you visit which then loads all other content using JavaScript, which they heavily depend on. SPA requests the markup and data independently and renders pages

straight in the browser. Single-page sites help keep the user in one, comfortable web space where content is presented to the user in a simple, easy, and workable fashion.

### **3.3.1.2 Angular 9**

Angular is an app-design framework and development platform for creating efficient and sophisticated single-page apps in Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and Typescript which is a superset of JavaScript. Angular provides built-in features for animation, HyperText Transfer Protocol (HTTP) service, and materials which in turn have features such as auto-complete, navigation, toolbar, menus, etc.

## **3.3.2 Back-End**

### **3.3.2.1 Node Js**

Node.js is open-source, and a cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux. Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

The following are some of the important features that make Node.js the first choice of software architects.

1. Asynchronous and Event-Driven
2. Very Fast
3. Single-Threaded but Highly Scalable
4. No Buffering
5. Node.js is released under the MIT license

### **3.3.2.2 Express Js**

ExpressJS is a web application framework that provides us with a simple API to build websites, web apps, and back ends. With ExpressJS, we need not worry about low-level protocols, processes, etc. Express provides a minimal interface to build our applications. It provides us the tools that are required to build our app. It is flexible as there are numerous modules available on npm, which can be directly plugged into Express. Express was developed by TJ Holowaychuk and is maintained by the Node.js Foundation and numerous open source contributors.

### **3.3.2.3 MongoDB**

MongoDB is a cross-platform, document-oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on the concept of collection and document.

1. Database: The database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

2. Collection: A collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purposes.
3. Document: A document is a set of key-value pairs. Documents have a dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structures, and common fields in a collection's documents may hold different types of data.

#### **3.3.2.4 Mongoose**

Mongoose is an Object Data Modeling Object Data Modelin (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.

## **3.4 Programming Languages**

### **3.4.1 Conception**

#### **PlantUML**

PlantUML is an open-source tool allowing users to create Unified Modeling Language (UML) diagrams from a plain text language. The language of PlantUML is an example of a Domain-specific language. It uses Graphviz software to lay out its diagrams.

### **3.4.2 General**

#### **JavaScript Object Notation**

JSON is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

### **3.4.3 Front-End**

#### **3.4.3.1 Typescript**

TypeScript is an open-source programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript and adds optional static typing to the language. TypeScript is designed for the development of large applications and transcompiles to

JavaScript. As TypeScript is a superset of JavaScript, existing JavaScript programs are also valid TypeScript programs.

### **3.4.3.2 HTML**

Hypertext Markup Language HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

### **3.4.3.3 CSS**

Cascading Style Sheets CSS is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

## **3.4.4 Back-End**

### **3.4.4.1 Javascript**

JavaScript often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.



## Chapter

**4****Realization, GUI And Tests****Contents**

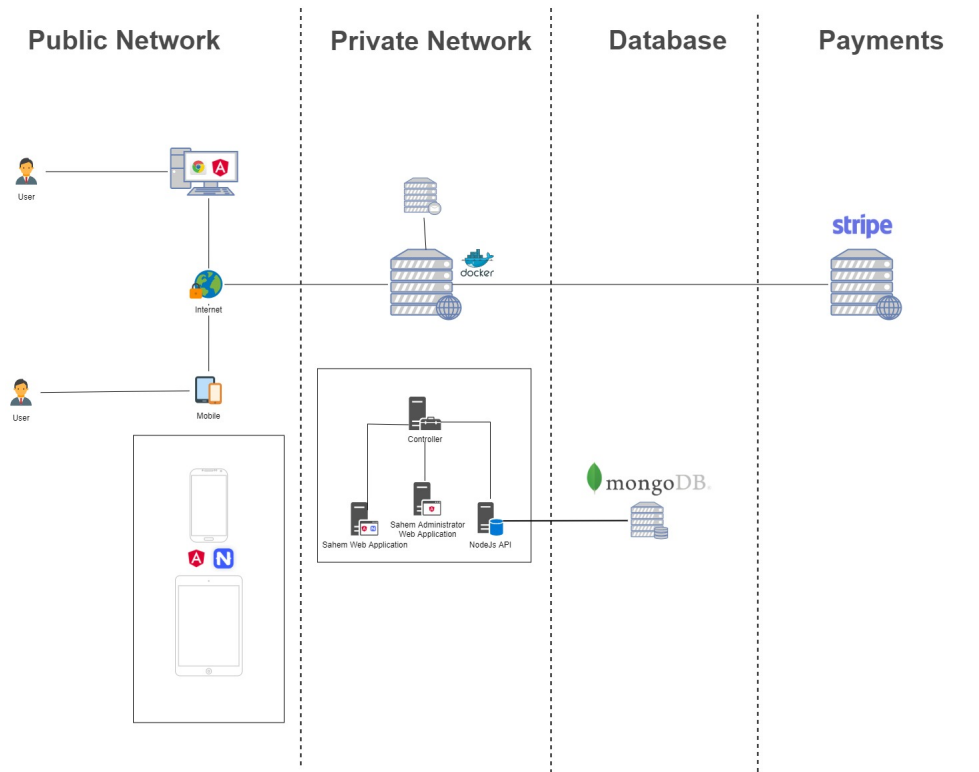
|            |   |           |
|------------|---|-----------|
| <b>4.1</b> | <b>Hardware Environments . . . . .</b>      | <b>37</b> |
| 4.1.1      | Development hardware . . . . .              | 37        |
| 4.1.2      | Production hardware . . . . .               | 38        |
| <b>4.2</b> | <b>Development Environments . . . . .</b>   | <b>38</b> |
| 4.2.1      | Design And Planing . . . . .                | 38        |
| 4.2.2      | User Interface And Product Design . . . . . | 39        |
| 4.2.3      | Development . . . . .                       | 39        |
| <b>4.3</b> | <b>Platform security . . . . .</b>          | <b>41</b> |
| 4.3.1      | Security Principles . . . . .               | 41        |
| 4.3.2      | Physical level . . . . .                    | 42        |
| 4.3.3      | Logical level . . . . .                     | 42        |
| <b>4.4</b> | <b>Project Tests . . . . .</b>              | <b>45</b> |
| <b>4.5</b> | <b>Deployment . . . . .</b>                 | <b>45</b> |
| <b>4.6</b> | <b>Interfaces . . . . .</b>                 | <b>46</b> |

**4.1 Hardware Environments****4.1.1 Development hardware**

The good news is that you don't need anything particularly special to run the development of mean stack. A single laptop or even a virtual machine Virtual Machine (VM) is enough to develop a MEAN application. All components of the stack can be installed on Windows, macOS, and most Linux distributions.

### 4.1.2 Production hardware

The approach to production hardware architecture isn't all that different from development hardware. The main difference is that production hardware is normally higher-spec and open to the internet to receive public requests.



**Figure 20.** *< Sahem > Platform's Hardware Architecture*

This model is commonly used when it comes to using Platform as a service (PaaS) provider for hosting.

## 4.2 Development Environments

### 4.2.1 Design And Planing

In the modeling phase, we found many solutions and software that helped to create diagrams, but they all are graphic based solutions and felt a bit short. But we decided to work with the following tools.

#### PlantUml

PlantUML is an open-source tool allowing users to create UML diagrams from plain text language. The language of PlantUML is an example of a Domain-specific language. It uses Graphviz software to layout its diagrams. It has been used to allow blind students to work with UML. PlantUML also helps blind software engineers to design and read UML diagrams.

## **Gantt chart**

A Gantt chart is a horizontal bar chart that visually represents a project plan over time. Modern Gantt charts typically show us the status of—as well as who’s responsible for—each task in the project.

### **4.2.2 User Interface And Product Design**

#### **Adobe Photoshop**

Adobe Photoshop is a software application for image editing and photo retouching and logo creation. Photoshop offers users the ability to create, enhance, or otherwise edit images, artwork, and illustrations. It is the most widely used software tool for photo editing, image manipulation, and retouching for numerous image and video file formats.

#### **Adobe XD**

Adobe XD is a vector-based user experience design tool for web apps and mobile apps, developed and published by Adobe Inc. It is available for macOS and Windows, although there are versions for iOS and Android to help preview the result of work directly on mobile devices.

### **4.2.3 Development**

#### **4.2.3.1 Version Control**

When building software it’s always important to track your changes. This is especially critical when collaborating on projects where multiple people will be updating the same code. Software that can keep track of all these changes are called Version Control.

#### **Git**

Git is a distributed revision control and source code management system that allows several people to work on the same codebase at the same time on different computers and networks. These can be pushed together, with all changes stored and recorded. It’s also possible to roll back to an earlier state if necessary.

#### **Github**

At a high level, GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code.

## **Github Desktop**

GitHub Desktop is a fast and easy way to contribute to projects, and is designed to simplify all processes and workflow in our GitHub.

### **4.2.3.2 Dependency Managers**

A large software project often makes use of many third-party packages and libraries. In turn, these packages often rely on several other packages and so on. To keep track of all these dependencies, software developers use-package managers. Below we introduce the ones we have used.

## **Node Package Manager**

Node Package Manager Node Package Manager (NPM), is two things: first and foremost, it is an online repository for the publishing of open-source Node.js projects; second, it is a command-line utility for interacting with the said repository that aids in package installation, version management, and dependency management.

## **Angular CLI**

The Angular CLI is a command-line interface tool that we use to initialize, develop, scaffold, and maintain Angular applications.

### **4.2.3.3 IDEs**

## **Visual Studio Code**

Visual Studio Code is a source code editor developed by Microsoft for Windows, Linux, and macOS. It includes support for debugging, embedded Git control and GitHub, syntax highlighting, intelligent code completion, snippets, and code refactoring.

## **MongoDB Compass Community**

MongoDB Compass is the defacto GUI tool for MongoDB much like MySQL Workbench is MySQL's associated tool. It allows us to visually explore our data, run ad hoc queries, interact with our data with full CRUD functionality, as well as view and optimize our queries' performance.

## **Postman**

Postman is an interactive and automatic tool for verifying the APIs of our project. Postman is a Google Chrome app for interacting with HTTP APIs. It presents us with a friendly GUI for constructing requests and reading responses. It works on the backend, and makes sure that each API is working as intended.

#### 4.2.3.4 Report And Presentation

##### Boost Note

Boostnote is an Open source note-taking app for programmers.

##### Latex

L<sup>A</sup>T<sub>E</sub>X is a tool used to create professional-looking documents. It is based on the WYSIWYM (what we see is what we mean) idea, meaning we only have to focus on the contents of our document and the computer will take care of the formatting. Instead of spacing out text on a page to control formatting, as with Microsoft Word or LibreOffice Writer, users can enter plain text and let L<sup>A</sup>T<sub>E</sub>X take care of the rest.

##### Markdown

Markdown is a lightweight markup language with plain-text-formatting syntax. Its design allows it to be converted to many output formats, but the original tool by the same name only supports HTML. Markdown is often used to format readme files, for writing messages in online discussion forums, and to create rich text using a plain text editor.

##### Marp

Marp is the ecosystem to write our presentation with plain Markdown.

## 4.3 Platform security

Security is a constant worry when it comes to information technology. Data theft, hacking, malware, and a host of other threats are enough to keep any IT professional up at night. In this article, we'll look at the basic principles and best practices that IT professionals use to keep their systems safe.

When you create a website or application, it means you are ready to show thousands of people on the internet. Your customer/audience may be legit, but some of them will try to tamper your application. So we need to follow the following golden rules:

1. "Never trust your audience when it comes to your application's security"
2. Always validate the user input before getting it into the server.
3. Always encode the user inputs before printing them on to the screen.

### 4.3.1 Security Principles

The main three overarching principles in information security are:

1. Confidentiality: This means that information is only being seen or used by people who are authorized to access it.
2. Integrity: This means that any changes to the information by an unauthorized user are impossible (or at least detected), and changes by authorized users are tracked.
3. Availability: This means that the information is accessible when authorized users need it.

### 4.3.2 Physical level

For the Physical level, we will be hosting and deploying our application to an IaaS cloud service, the main reason to use the cloud services is the ability they provide to our application to be decentralized, and available all time.

Physical security on IaaS is managed by one or several processes, which include:

1. Area security definition
2. Controlled access to those areas
3. Uninterrupted power supplies
4. Monitoring critical parameters
5. Alarms
6. Air and particle filtering
7. Fire protection
8. Others, such as proper risk and issue management

### 4.3.3 Logical level

Logical Security consists of software safeguards for an organization's systems, including user identification and password access, authenticating, access rights, and authority levels. These measures are to ensure that only authorized users are able to perform actions or access information in a network or a workstation. We have implemented multiple levels and layers of security to control the integrity of the data exchanged between our users and « Sahem » platform, from creating form controls to ensure that the data sent from the user is well structured, and when received on the controller, we created multiple layers and implemented some great middlewares and libraries to help us protect the system from different vulnerabilities.

And for the Confidentiality, on the Frontend we put guards on our routes, to only give some specific users the accessibility to the functionalities they have the permission to do, and when a request is received on the controller, we check the user's Identity, and check if they are allowed to take the action requested.

#### 4.3.3.1 Stripe Payment API

Stripe was founded in 2010 with the mission of making it easier to accept payments over the internet. At the time, taking credit cards meant working with a legacy processor or a middleman broker who would provide you with access to a processor. Stripe has been audited by a PCI-certified auditor and is certified to PCI Service Provider Level 1. This is the most stringent level of certification available in the payments industry.

They use the best-in-class security tools and practices to maintain a high level of security at Stripe:

1. HTTPS and HSTS for secure connections
2. Encryption of sensitive data and communication
3. PGP keys to encrypt your communications with Stripe

#### **4.3.3.2 Frontend**

Angular has some great protections against common web-application vulnerabilities and attacks to provide it's applications with a good, and secure experience, preventing cross-site scripting Cross-Site Scripting (XSS) that prevent attackers from injecting malicious code into web pages. Such code can then, for example, steal user data (in particular, login data) or perform actions to impersonate the user. This is one of the most common attacks on the web.

And has built-in support to help prevent two common HTTP vulnerabilities, cross-site request forgery (Cross-Site Request Forgery (CSRF) or XSRF) and cross-site script inclusion Cross-Site Script Inclusion (XSSI). Both of these must be mitigated primarily on the server-side, but Angular provides helpers to make integration on the client-side easier.

#### **Route Guards**

Angular's route guards are interfaces that can tell the router whether or not it should allow navigation to a requested route. They make this decision by looking for a true or false return value from a class that implements the given guard interface.

#### **Form Control**

This is one of the three fundamental building blocks of Angular. It implements most of the base functionality for accessing the value, validation status, user interactions, and events.

#### **4.3.3.3 BackEnd**

Security is really hard to get right on the backend. There are so many different factors to consider, countless different ways to break an application. This is just as true with Express applications as it is with any other web framework. There's no instant way to make sure an application won't be taken down by a Denial of Service (DoS) attack because of how it handles one type of user input, or how it routes a specific request.

#### **Error Handling**

Error handling is considered one of the main factors in creating a good production software for any type of applications, so we have made it our main

## Check for Vulnerabilities

There are a few tools in the Node ecosystem that allow easy checking for vulnerabilities in Node and Express application dependencies. These tools are highly valuable in ensuring that no vulnerabilities are currently in the packages an application relies on, and none are added into that application when its packages are updated.

1. Snyk
2. Node Security Project
3. Retire.js

## Helmet

### HTTP headers

The helmet middleware is really like a helmet for our applications. It protects our application by setting up various HTTP headers.

## Session Data

The express-session middleware stores session data on the server; it only saves the session ID in the cookie itself, not session data.

### cookie-session

Cookie-session is a simple cookie-based session middleware, that allows storing cookies. It doesn't require any database/resources on the server-side, though the total session data cannot exceed the browser's max cookie size.

## Authentication

express-jwt-permissions is a middleware that checks JSON Web Token (JWT) tokens for permissions. It is very useful to build a user access control system.

## Data sanitize

Express-mongo-sanitize is a middleware that sanitizes user-supplied data to prevent MongoDB Operator Injection.

## Errors Logging

Morgan is a library adds some logging capabilities to our Express API.



## Cross-Origin Resource Sharing

We will use Cors dependency to configure Express to add headers stating that your API accepts requests coming from other origins. This is known as Cross-Origin Resource Sharing (CORS).

## Server's Information

Dotenv is a zero-dependency module that loads environment variables from a .env file into process.env.. Storing configuration in the environment separate from code is based on The Twelve-Factor App methodology [11].

## 4.4 Project Tests

Project testing and validation is the last step of the software engineering process. The Testing is conducted during or after the implementation in order to ascertain that the product is compliant with the requirements and modeling specifications initially stipulated in the analysis and design phase, i.e. whether or not the product, « Sahem » Crowdfunding Platform in our case, satisfies and meets the needs, requirements, and design initially stipulated. The testing process for our application comprised of Unite testing through Jasmine and Karma for the angular application, and Mockgoose for our express js controller, then use Protractor.js to perform end to end testing. This is achieved by test targeting specific and small portions of code such as individual classes or functions. The Integration Testing is done by verifying and testing the interaction between the different components of our app as they were integrated in addition to the data flow between them. Many errors and system failures occurred during this step and are being dealt with. Concerning System Testing, this is conducted by testing the system as a whole to ensure that the requirements, both functional and non-functional are met. The final step of the testing phase, which represents the Acceptance Testing will be conducted by Dr. BELAQZIZ SALWA to see whether the project meets the requirements stated earlier or not.

## 4.5 Deployment

Today, there exist multiple PaaS on the internet, that gives a lot of rich environments with NodeJs included. Sadly, most of them are paid, and those that provide the free tier services, puts a lot of constraints on the application, such as Heroku, that we will be using to deploy our application, it does delete all the static files on every server restart, and shuts down the server when it does not receive any request for some time. But one of the positive points was that we could deploy the application by linking it to the development repository on Github.

<https://sahem.herokuapp.com/>

## 4.6 Interfaces

One of the main requirements for our Frontend application was to create a responsive design to fit our application in different devices and screen sizes.

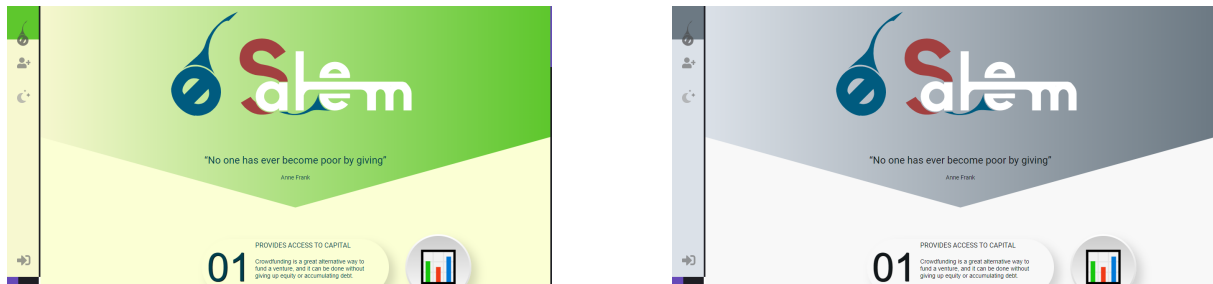


**Figure 21.** *Responsive Design*

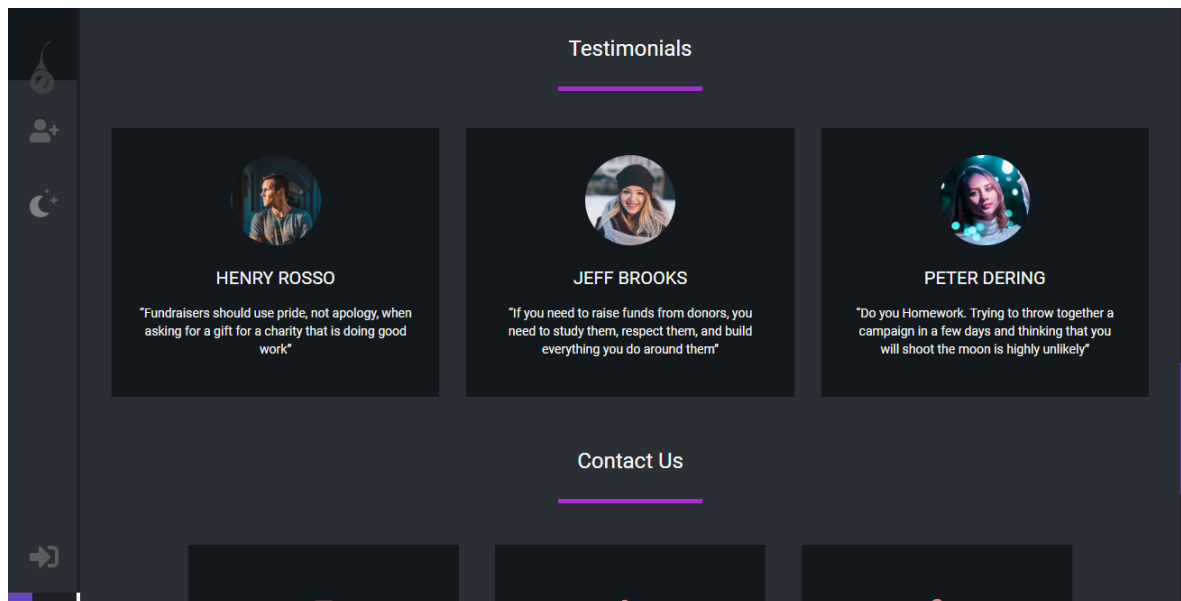


**Figure 22.** *« Sahem » Main Page - Navbar Activated*

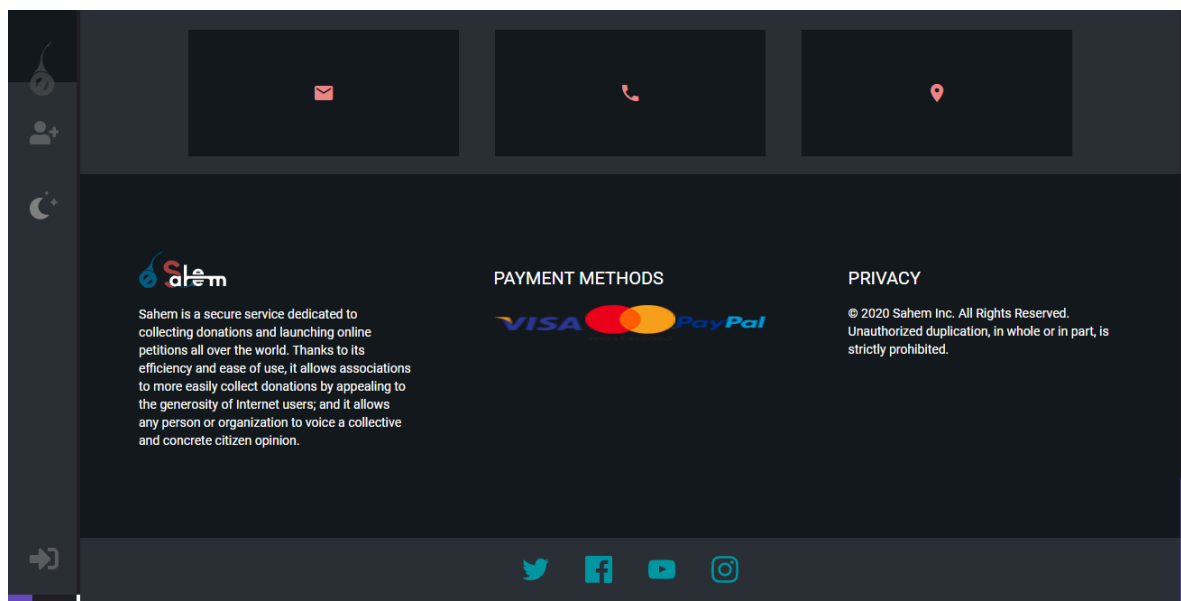
The application comes with different themes and colors to satisfy multiple varieties of users.



**Figure 23.** « Sahem » Main Page - Themes



**Figure 24.** « Sahem » Main Page - Testimonials

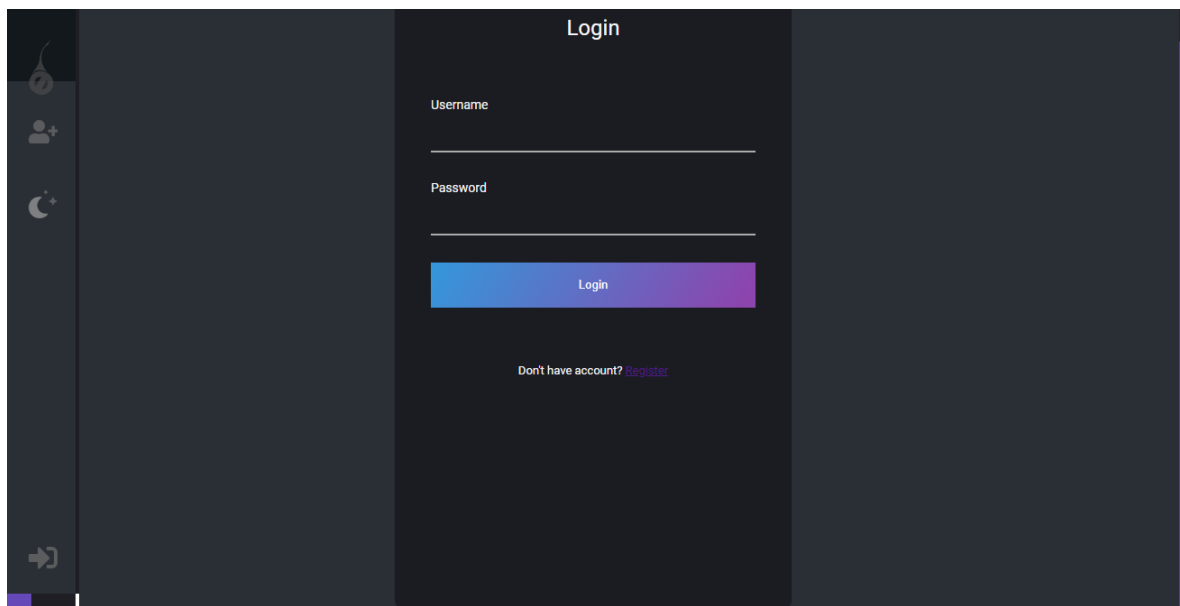


**Figure 25.** « Sahem » Main Page - Footer

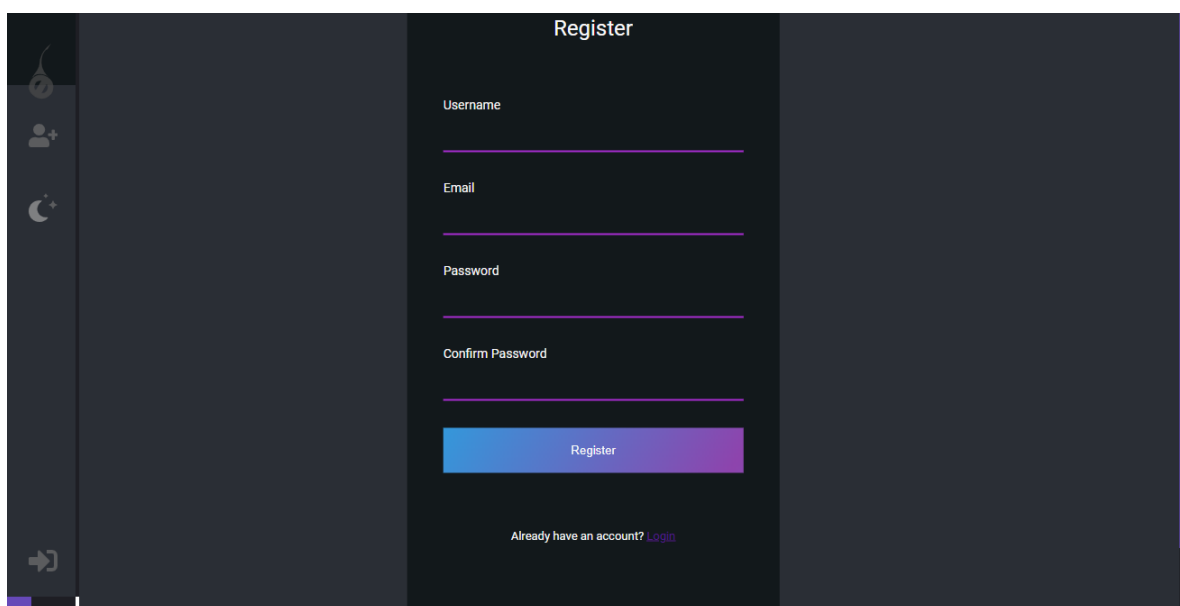


**Figure 27.** *Login Page - Themes*

The Angular framework provides us with a rich form control experience. With it, we designed and created multiple delicate forms Figure 26 and Figure 28.

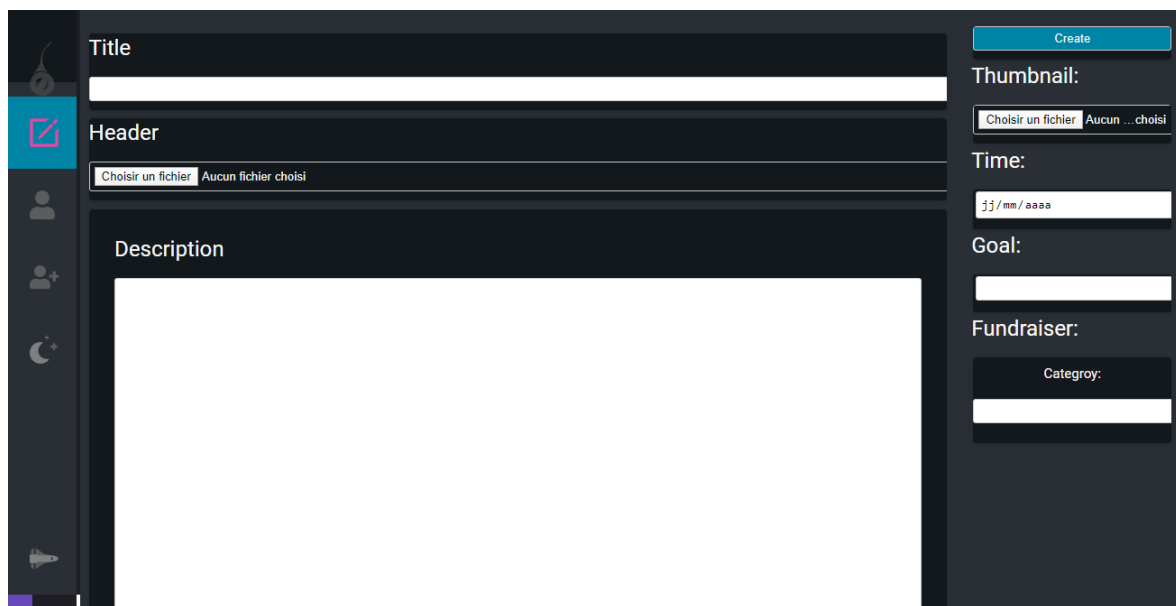


**Figure 26.** *Login Page*



**Figure 28.** *Register Page*

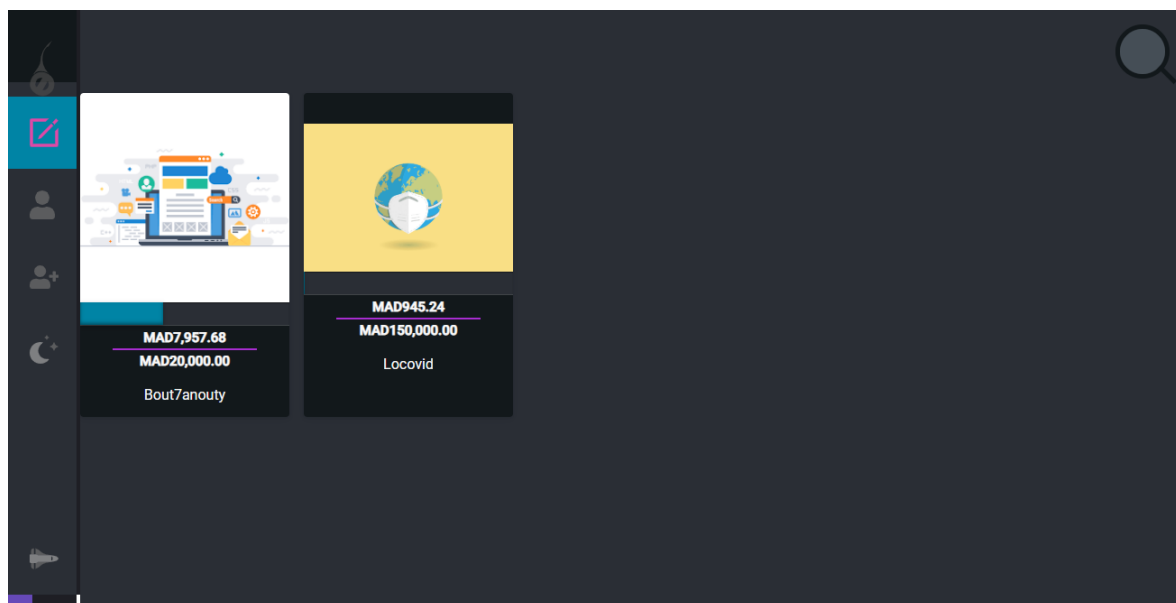
Figure 29 is a form that allows the user to create a fundraiser.



The screenshot shows a web form for creating a fundraiser. On the left is a dark sidebar with icons for home, edit, users, settings, and a search icon. The main form area has a dark header with a 'Title' label and a text input field. Below the title is a 'Header' section with a file upload button labeled 'Choisir un fichier' and the text 'Aucun fichier choisi'. The main body of the form is a large white 'Description' text area. On the right side, there is a vertical stack of form elements: a blue 'Create' button, a 'Thumbnail:' section with a file upload button 'Choisir un fichier' and the text 'Aucun ...choisi', a 'Time:' section with a date input field showing 'jj/mm/aaaa', a 'Goal:' section with a text input field, a 'Fundraiser:' section with a text input field, and a 'Category:' section with a text input field.

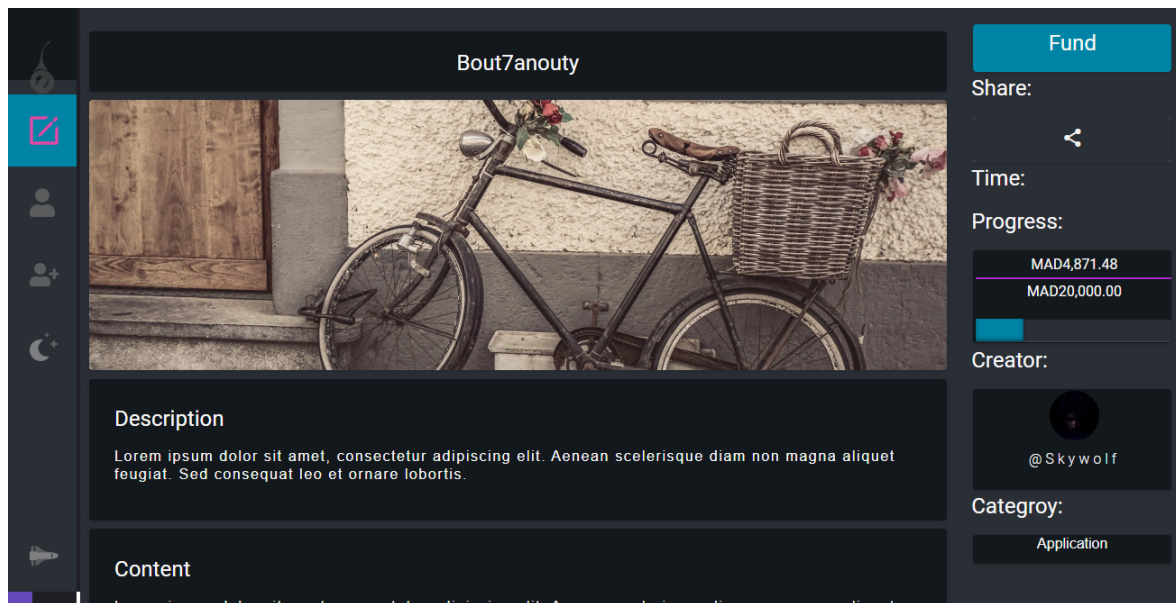
**Figure 29.** *Fundraiser Form*

Figure 30 is a showcase with the in-progress projects existing on the platform.



**Figure 30.** *Fundraisers List*

Figure 31 is an example of the content of a created fundraiser.

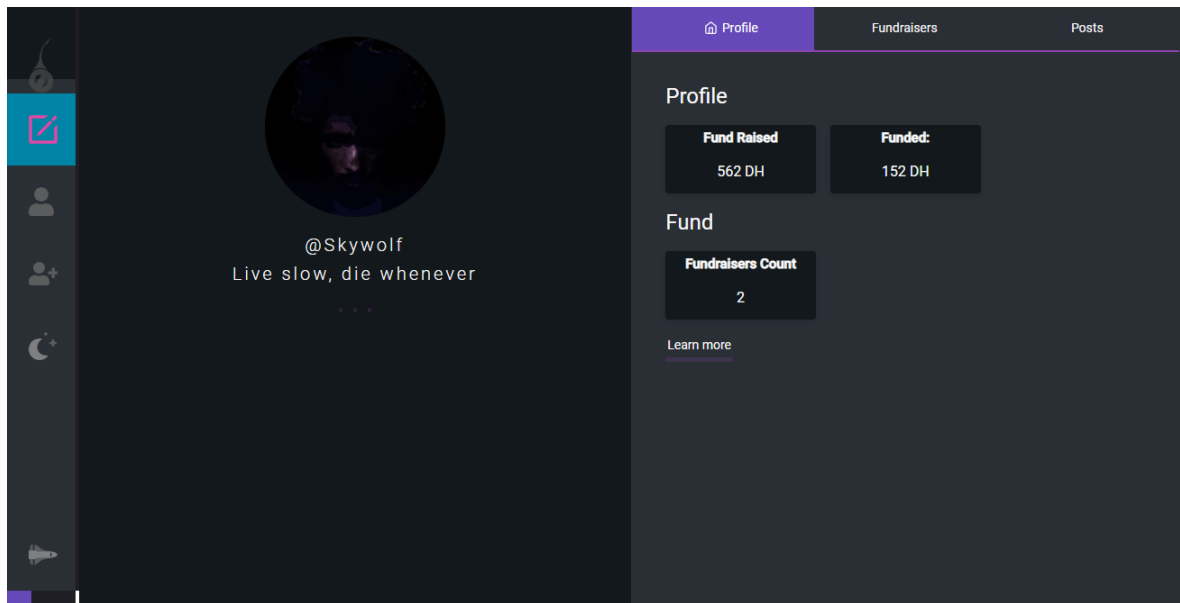


**Figure 31.** *Fundraiser View*

When a user accesses his profile for the first time, he will be asked to fill the form in Figure 32 to create his Profile, he won't be able to create a fundraiser otherwise.

**Figure 32.** *Profile Form*

For every user there exist a profile that holds his information, Figure 33 is page showcase the different fundraisers he created and successfully funded.

**Figure 34.** *Profile Page - Themes***Figure 33.** *Profile Page*



---

# Conclusion And Perspective

## Conclusion

Ideas are cheap, execution is expensive, even if you create a beautiful application with a beautiful User Interface, if you neglect even one aspect of the main functionalities, and built it with a half effort, you might result in a product that won't live for long, or if you were to use the wrong technologies for the project, you might as well find yourself limited and won't be able to create and bring the most out of your Idea.

With this in mind, and from the very start of the project we gave enough time for the design phase, we aimed to create not a prototype of a project that will be neglected afterward, but a well structured project, that respects the nowadays requirements, implementing the MEAN stack technologies, and multiple layers of security, especially that we have an application that handles payments transactions and sensible pieces of information, from debit cards to the user's personal information.

And it goes without saying that I learned a lot of different skills during the development of this project, it helped me to widen my knowledge and use different practices and concepts I acquired throughout my academic years, and my journey in self-learning. The research on the literature side of this project also helped me to link between my coding skills, and the ability to create solutions, as well as the communication skills I acquired during the rich talks I had with Dr. BELAQZIZ SALWA .

## Perspective

Unfortunately, due to time, material, and some uncontrollable constraints, I was not able to arrive at the result, we had envisioned for this project. However, this does not mean that the progress of this project is over, with the previous work we did as the base, there are still, several aspects and functionalities to include in order for this app to take the shape we intended for it. This continuous progress will go into the future work category and will include the following actions:

- Creating a native application with Nativescript, reusing the logic we built in Angular.



- Move from a monolithic architecture to microservices.
- Deploy the platform to a better PaaS such as Google Cloud.
- Linking the application with a domain name.
- Create an ai model to handle the suggestion to users during the creation of a fundraiser.
- Finish the messaging Business Logic.
- Finish the posts Business Logic.
- Finish the Administration application.
- Finish the comment section of a fundraiser.
- Improving UX and error handling.
- Studying the possibility of integrating Inwi Money or an alternative for Moroccan Users.

The future work is not restricted to these actions, as we know a software lifespan is a cycle of continuous refinement and prototyping into improved products and so will be the case for this platform.



---

## Webography

- [11] *The Twelve-Factor App*. URL: <https://12factor.net/config> (visited on 0004–2020).
- [12] *Latex @ Wikipedia*. URL: <https://en.wikipedia.org/wiki/LaTeX> (visited on 0004–2016).
- [13] *FSA*. URL: <http://www.fsa.ac.ma/> (visited on 0003–2020).
- [14] *Peer-to-Peer (P2P) Lending*. URL: <https://www.investopedia.com/terms/p/peer-to-peer-lending.asp> (visited on 0004–2020).



---

## Bibliography

- [1] Mohamed Amine Korchi, Driss Daoui, and Hafid Barka. “Crowdfunding in Morocco”. In: 27 (Nov. 2019), pp. 1037–1057.
- [2] Andrea Ordanini et al. “Crowd-funding: Transforming customers into investors through innovative service platforms”. In: *Journal of Service Management* 22 (Aug. 2011), pp. 443–470. DOI: 10.1108/09564231111155079.
- [3] Alexandra Moritz and Joern Block. “Crowdfunding: A Literature Review and Research Directions”. In: Jan. 2016, pp. 25–53. ISBN: 978-3-319-18016-8. DOI: 10.1007/978-3-319-18017-5\_3.
- [4] Elizabeth Gerber, Julie Hui, and Pei-Yi (Patricia) Kuo. “Crowdfunding: Why People are Motivated to Post and Fund Projects on Crowdfunding Platforms.” In: vol. 10. Feb. 2012.
- [5] Paul Belleflamme, Thomas Lambert, and Armin Schwienbacher. “Individual crowdfunding practices”. In: *Venture Capital* 15.4 (2013), pp. 313–333. DOI: 10.1080/13691066.2013.785151. eprint: <https://doi.org/10.1080/13691066.2013.785151>. URL: <https://doi.org/10.1080/13691066.2013.785151>.
- [6] Ajay K Agrawal, Christian Catalini, and Avi Goldfarb. *The Geography of Crowdfunding*. Working Paper 16820. National Bureau of Economic Research, 2011. DOI: 10.3386/w16820. URL: <http://www.nber.org/papers/w16820>.
- [7] Yan Lin, Wai Fong Boh, and Kim Huat Goh. “How Different are Crowdfunders? Examining Archetypes of Crowdfunders and Their Choice of Projects”. In: *Academy of Management Proceedings* 2014.1 (2014), p. 13309. DOI: 10.5465/ambpp.2014.209. eprint: <https://doi.org/10.5465/ambpp.2014.209>. URL: <https://doi.org/10.5465/ambpp.2014.209>.
- [8] Venkat Kuppuswamy and Barry Bayus. “Crowdfunding Creative Ideas: The Dynamics of Project Backers in Kickstarter”. In: *SSRN Electronic Journal* 2013 (Mar. 2013). DOI: 10.2139/ssrn.2234765.
- [9] Elizabeth Gerber and Julie Hui. “Crowdfunding: Motivations and Deterrents for Participation”. In: *ACM Transactions on Computer-Human Interaction* 20 (Jan. 2014), pp. 34–32.
- [10] Julie Hui, Michael Greenberg, and Elizabeth Gerber. “Understanding the role of community in crowdfunding work”. In: Feb. 2014, pp. 62–74. DOI: 10.1145/2531602.2531715.

- [15] Ashbindu Singh. “Review article digital change detection techniques using remotely-sensed data”. In: *International journal of remote sensing* 10.6 (1989), pp. 989–1003.
- [16] Mi (Jamie) Zhou et al. “Project description and crowdfunding success: an exploratory study”. In: *Information Systems Frontiers* 20.2 (2018), pp. 259–274. ISSN: 1572-9419. DOI: 10.1007/s10796-016-9723-1. URL: <https://doi.org/10.1007/s10796-016-9723-1>.
- [17] Andrea Ordanini et al. “Crowd-funding: Transforming customers into investors through innovative service platforms”. In: *Journal of Service Management* 22 (Aug. 2011), pp. 443–470. DOI: 10.1108/09564231111155079.
- [18] Bhuvan Unhelkar. *Software Engineering with UML*. 1st. USA: Auerbach Publications, 2017. ISBN: 1138297437.

