**REPUBLIC OF CAMEROON**

Peace - Work - Fatherland

**REPUBLIQUE DU CAMEROUN**

Paix – Travail - Patrie



**PERSONALISED PROJECT REPORT**

**INSTITUT AFRICAIN DE**

**L’INFORMATIQUE – CAMEROUN**

**CENTRE TECHNOLOGIQUE DE L’EXCELLENCE PAUL BIYA**

**B.P: 13719 Yaoundé**

**Tel: +237.242.729.957 Fax: 22729958**

**Courriel:contact@iaicameroun.com**

**AFRICAN INSTITUTE OF COMPUTER SCIENCE**

**CAMEROON OFFICE, PAUL-BIYA TECHNOLOGICAL CENTER OF EXCELLENCE**

P.O Box: 13719 Yaoundé

Tel: +237.242.729.957 Fax: 22729958

E-mail: [contact@iaicaameroun.com](mailto:contact@iaicaameroun.com)

**THEME :**

**COMPUTERISED PLATFORM FOR THE DAILY COLLECTION IN A MICROFINANCE**

**Case study: VISION FINANCE**

**:**

Written by:

**WIRBA JORDAN WIRBA**

Level III Software Engineering

**Supervised by.**

**MR. ASANE DERICK**

**Lecturer at AICS Cameroon**

**Academic Year 2024 – 2025**

**DEDICATION**

**THIS BOOK IS DEDICATED**

**TO THE ALMIGHTY GOD**

# **ACKNOWLEDGEMENTS**

This work would not have been accomplished without the help, encouragements, advices, teachings and precious time of the following people:

❖ We give thanks to the almighty God who made it possible without whom nothing would have been done.

❖ The Resident Representative of the African Institute of Computer Sciences, Mr. Armand Claude ABANDA.

❖ Mr. AGBOR Anderson lecturer at AICS-Cameroon for his Inspirational pieces of advices, encouragements and moral support.

❖ To all our teachers of the African Institute of Computer Sciences for their support throughout our training.

❖ To my family most especially to my parents for their guidance and support without which we could not attain the end of this work.

❖ To all our classmates and friends at AICS, for their knowledge, ideas, collaboration and most importantly the unforgettable moments spent together.

# **CONTENT**

[ACKNOWLEDGEMENTS ii](#_Toc201277274)

[CONTENT iii](#_Toc201277275)

[LIST OF FIGURES iv](#_Toc201277276)

[LIST OF TABLES v](#_Toc201277277)

[LIST OF ABREVIATIONS vi](#_Toc201277278)

[ABSTRACT vii](#_Toc201277279)

[RESUME viii](#_Toc201277280)

[GENERAL INTRODUCTION 1](#_Toc201277281)

[**CHAPTER 1: THE EXISTING SYSTEM** 3](#_Toc201277282)

[**CHAPTER 2: SPECIFICATION BOOK** 9](#_Toc201277283)

[**CHAPTER 3: ANLYSIS PHASE** 22](#_Toc201277284)

[**CHAPTER 4: CONCEPTION PHASE** 54](#_Toc201277285)

[**CHAPTER 5: REALIZATION PHASE** 65](#_Toc201277286)

[**CHAPTER 6: FUNCTIONALITY TEST** 80](#_Toc201277287)

[**CHAPTER 7: USER GUIDE** 84](#_Toc201277288)

[GENERAL CONCLUSION 98](#_Toc201277289)

[ANEXES ix](#_Toc201277290)

[BIBLIOGRAPHY x](#_Toc201277291)

[WEBOGRAPHY xi](#_Toc201277292)

[TABLE OF CONTENT xii](#_Toc201277293)

# **LIST OF FIGURES**

[*Figure 2:UML diagrams* 27](#_Toc199848096)

[*Figure 3:Branches of 2TUP* 29](#_Toc199848097)

[*Figure 4:Use case diagram formalism* 33](#_Toc199848098)

[*Figure 5:General use case diagram of our system* 35](#_Toc199848099)

[*Figure 6:Detailed use case of Manage account* 36](#_Toc199848100)

[*Figure 7:Detailed use case of Manage contributor* 36](#_Toc199848101)

[*Figure 8: Formalism of communication diagram* 40](#_Toc199848102)

[*Figure 9:Communication diagram of Authenticate* 42](#_Toc199848103)

[*Figure 10:Communication diagram for register contribution* 42](#_Toc199848104)

[*Figure 11:Formalism of sequence diagram* 43](#_Toc199848105)

[*Figure 12:Sequence diagram for Authentication* 45](#_Toc199848106)

[*Figure 13:Sequence diagram for register contribution* 46](#_Toc199848107)

[*Figure 14:Formalism of activity diagram* 47](#_Toc199848108)

[*Figure 15:Activity diagram for Authentication* 49](#_Toc199848109)

[*Figure 16:Activity diagram verify assigned collector* 50](#_Toc199848110)

[*Figure 17:Activity for Add collector* 51](#_Toc199848111)

[Figure 18:Formalism of class diagram 56](#_Toc199848112)

[*Figure 19:Components of a class diagram* 57](#_Toc199848113)

[*Figure 20:Class diagram of our system* 58](#_Toc199848114)

[*Figure 21:State machine diagram of Notification state* 61](#_Toc199848115)

[*Figure 22:Formalism of Package diagram* 62](#_Toc199848116)

[*Figure 23:Components of a package diagram* 62](#_Toc199848117)

[*Figure 24:Package diagram of our system* 63](#_Toc199848118)

[*Figure 25:Formalism of deployment diagram* 67](#_Toc199848119)

[*Figure 26:Deployment diagram of our system* 69](#_Toc199848120)

[*Figure 27:Formalism of Component diagram* 70](#_Toc199848121)

[*Figure 28:Component diagram of our system* 72](#_Toc199848122)

[*Figure 29:The N-tier architecture* 73](#_Toc199848123)

[*Figure 30:MVC design pattern* 74](#_Toc199848124)

# **LIST OF TABLES**

[Table 3:Limits, consequences and solutions of existing system 6](#_Toc199848077)

[*Table 4:Time schedule* 14](#_Toc199848078)

[*Table 5:Actors of the project* 16](#_Toc199848079)

[*Table 6:Hardware resources* 17](#_Toc199848080)

[*Table 7:Software resources* 18](#_Toc199848081)

[*Table 8:Human resources* 19](#_Toc199848082)

[*Table 9:Total estimation of the project* 19](#_Toc199848083)

[*Table 10:UML diagrams* 26](#_Toc199848084)

[*Table 11:Comparative between UML and MERISE* 28](#_Toc199848085)

[*Table 12:Components of a use case diagram* 34](#_Toc199848086)

[Table 13:Formalism of textual description 37](#_Toc199848087)

[Table 14:Textual description of Authenticate 38](#_Toc199848088)

[Table 15:Textual description of register contribution 39](#_Toc199848089)

[*Table 16:Components of Communication diagram* 41](#_Toc199848090)

[*Table 17:Components of a sequence diagram* 44](#_Toc199848091)

[*Table 18:Components of activity diagram* 48](#_Toc199848092)

[*Table 19:Components of a state machine diagram* 60](#_Toc199848093)

[*Table 20:Components of a deployment diagram* 68](#_Toc199848094)

[*Table 21:Components of a component diagram* 71](#_Toc199848095)

# **LIST OF ABREVIATIONS**

➢ 2TUP: Two Track Unified Process.

➢ AICS: African Institute of Computer Sciences.

➢ DBMS: Database Management System

➢ ERD: Entity Relational Diagram.

➢ GUI: Graphical User Interface.

➢ IDE: Integrated Development environment

➢ JSON: JavaScript Object Notation

➢ MVC: Model View Controller.

➢ PDM: Physical Data Model.

➢ UML: Unified Modelling Language.

# **ABSTRACT**

Humanity has not stopped evolving since its creation from agricultural revolution to industrial revolution and now towards an information and communication technology revolution. This is due to its exponential evolution in various domains such as transport, agriculture, communication, education, health, and many leading to innovations in these domains. Thus, we cannot see the future without technology. For this reason, we decided to develop an application **“DAILY COLLECT”** to facilitate the transaction process between the contributor and collector in a more secure and efficient way. Following our theme, we did the feasibility studies, followed by the analysis of the system using the UML methodology associated to 2TUP process, from the analysis passing through conception and realization phase. The application was developed using php, html, CSS and java script some open-source API’s in order to obtain a flexible, light and secure solution for the best user experience.

**Keywords:**

• API (Application Programming Interface)

• UML (Unified Modeling language)

2TUP

# **RESUME**

L'humanité n'a cessé d'évoluer depuis sa création, de la révolution agricole à la révolution industrielle et maintenant vers une révolution des technologies de l'information et de la communication. Cela est dû à son évolution exponentielle dans divers domaines tels que le transport, l'agriculture, la communication, l'éducation, la santé, et bien d'autres, menant à des innovations dans ces domaines. Ainsi, nous ne pouvons pas envisager l'avenir sans la technologie. Pour cette raison, nous avons décidé de développer une application « DAILY COLLECT » pour faciliter le processus de transaction entre le contributeur et le collecteur de manière plus sécurisée et efficace. Suivant notre thème, nous avons réalisé des études de faisabilité, suivies de l'analyse du système en utilisant la méthodologie UML associée au processus 2TUP, de l'analyse à la conception, puis à la phase de réalisation. L'application a été développée en utilisant PHP, HTML, CSS et JavaScript, ainsi que des API open-source pour obtenir une solution flexible, légère et sécurisée, offrant la meilleure expérience utilisateur.

**Mots clés:**

• API

• UML

• 2TUP

# **GENERAL INTRODUCTION**

Technology is advancing at an incredible rate, spanning almost every sector from business to agriculture and other sectors. Due to the huge technological advancement, we now live in a fast-paced society where everything is gradually being digitalized making life easier. As a developing country, most businesses in Cameroon strive to make advantage of the digital economy to grow their businesses thereby boosting the economy as well. To achieve such a goal enterprise, need qualified personnel having skills in computer sciences and related fields. It is in this regard that institutions like AICS Cameroon are the place to be nowadays in order to acquire such skills. Level III students at AICS are required to produce at the end of the year, a personalized project. As the name implicates, it is a project that is or might be of moral or psychological importance to us. We then decided to work on the theme “**COMPUTERISED PLATFORM FOR DAILY COLLECTION IN A MICROFINANCE**”. Based on our theme, we needed to make thorough research in order to provide an acceptable solution. We divided this report into seven (7) parts which are as follows:

.

* **Existing system:** This phase consists of presenting the actual system.
* **Specification book:** The specification book is the part of the document that describes exactly what is expected from us taking into consideration the time and the cost of the project. In other to proposed a suitable solution for the user.
* **Analysis phase:** It presents the analysis method chosen with presentation of all diagrams use for the analysis of the project.
* **Conception phase:** This phase enables us to model the solution based on the criticism done at the analysis phase.
* **Realization phase:** It permits to visualize the implementation of the process in an organized step by step process.
* **The installation and user manual:** It demonstrates how the system is being used**.**

**PART 1: TECHNICAL PHASE**

**Preamble**

The existing system is a part of the internship report where we shall give a detailed explanation of our theme. That is, what the theme is all about and also to give more precision concerning the application we are to produce. Also, it provides a deep understanding of the system currently in placed associated to the various limitations, the problems that result from these and the solution we propose.

**Content**

**INTRODUTION**

**CHAPTER 1: THE EXISTING SYSTEM**

**CHAPTER 2: SPECIFICATION BOOK**

**CHAPTER 3: ANALYSIS PHASE**

**CHAPTER 4: CONCEPTION PHASE**

**CHAPTER 5: REALIZATION PHASE**

**CHAPTER 6: FUNCTIONALITY TESTING**

**CHAPTER 7: INSTALLATION AND USER GUIDE**

**CONCLUSION**

## **CHAPTER 1: THE EXISTING SYSTEM**

**Preamble**

The study of the existing system is a crucial step in the field of IT. It provides an in-depth understanding of the systems, processes and technologies already in place, in order to make informed decisions for the improvement or creation of new IT systems. A well-conducted study of the existing situation contributes to the efficiency, performance and success of IT projects.

**Content**

**INTRODUTION**

1. **DESCRIPTION OF THE EXISTING SYSTEM**
2. **CRITICISMS OF THE EXIXTING SYSTEM**
3. **PROBLEMATIC**
4. **PROPOSED SOLUTION**

**CONCLUSION**

### **INTRODUTION**

Successful achievements of a project need clear instructions to avoid failures. Hence, realizing the document phase called the specification book will help avoid the failure of the project. The specification book permits to specify the expectations of the customer as well as the standards that will govern the project for a good, smooth, precise and concrete project. It establishes a certain agreement between the client of the project and the person who is supposed to realize the project. It presents the objectives and needs of the system to be developed. The present specification book refers to the context and justification, the needs of the project, the plan and deliverables.

### **DESCRIPTION OF THE EXISTING SYSTEM**

#### **Objectives**

This study enables us to assemble information related to the management of the present system before the proposition of a solution or amelioration. This study can be done through interviews or from documents (manuals, archives…). Therefore, communication is very important between the user and the developer of the future system. The application we are putting in place will be adapted to the user’s needs.

#### **Description of the existing system**

Based on our research and information collections we noticed that, the process of daily collection in a microfinance is very long and inefficient, there is little or no management of contributors, no direct contact between the microfinance institution and their contributors.

Daily collection at **VISION FINANCE** goes on as follows;

* + Every morning collectors arrive at the institution to have their working tools that is usually made up of: registration book, stamp pad, **VISION FINANCE** jacket and the barge, then reach out to their various contributors.
  + At the contributor’s market place, the contributor gives the collector the amount of money he’s willing to save.
  + The collector goes further registering the contributor’s saving in his big register then the contributor signs in the big register to confirm the transaction.
  + Furthermore, the collector does the same in a small book own by the contributors, signs and put the stamp to confirm the transaction.
  + After the collector is done collecting from his different contributors, he goes back directly to the institution, deposits the money and the register.
  + To add, the money and the register are been controlled to make sure everything matches with what is been written and vice-versa.
  + After the personnel in charge of registering the different daily collection of money in an excel spreadsheets takes the different register book of every collector and enters the different of their daily collection in a spreadsheet.

### **CRITICISMS OF THE EXISTING SYSTEM**

|  |  |  |
| --- | --- | --- |
| LIMIT | CONSEQUENCES | SOLUTION |
| Transactions are processed and stored largely if not exclusively on physical  registers | The waste of time registering a new contributor | Collectors have a dedicated space through which they carry out operations with complete traceability |
| Access to transaction history is painful and costly for  contributors | Leads to stress and confusion. | Contributors have real-time access to their transaction  histories |
| The contributor has no secure and infallible way to authenticate and differentiate assign collectors from fake collectors | Contributors being reluctant and vulnerable to theft. | Contributors can view the profile of collectors assigned to them and scan the Qrcode for more security |
| The administrator has no information on the geographical position of the collectors and their interactions once in activity. | The collector can embezzel the money and run away. | Developing a platform that permits to the administrator to track his personel when on field of work. |
| Archiving, balance sheet and reports are processed and exported manually | Lost of important information with no backup. | The archiving of data and files is digitized in order to offer adequate and restrictive use as needed. |

#### Table 3:Limits, consequences and solutions of existing system

### **PROBLEMATIC**

Since Security is the basis of financial transaction, it is more than necessary that it be done in a simpler and more explicit way as possible because human living conditions are at stake. Though these various observations, we ask ourselves the question of Knowing **How to facilitate the transaction process between contributors and collectors in a secured and efficient way?** We will try to offer a solution through our application **(Daily Collect**).

### **PROPOSED SOLUTION**

After our study and criticism of the existing situation, we propose to design a Web application to solve the problems stated aboveby allowing:

**The collectors**

* + To register the contributions of the contributors via the application
  + He will view the contributors assigned to him
  + Generate receipts
  + Show the Qrcode for verification
  + View transactions history

**The contributor**

* + Scan the Qrcode to verify the collector
  + View his transactions
  + Deposit contribution
  + Call customer support
  + Create account
  + Sign

### **CONCLUSION**

The Existing system file allows you to understand the current landscape, identify good practices, examine lacunes, make informed decisions, define objectives and requirements. This leads to solutions being taken to overcome the problems found.

## **CHAPTER 2: SPECIFICATION BOOK**

**Preamble**

The primary goal of the specification book is to outline with great precision the requirements or need of the users and the description of the resources necessary to realize the project. It is considered as a communication and description tool which permits us to avoid inadequate results.

**Content**

INTRODUCTION

1. CONTEXT AND PROBLEM DEFINITION
2. OBJECTIVES OF THE PROJECT
3. EXPRESSION OF NEEDS
4. PLANNING OF THE PROJECT
5. ESTIMATION OF RESOURCES NEEDED
6. CONSTRAINTS
7. DELIVERABLES

CONCLUSION

### **INTRODUTION**

In order to successfully achieve a project, we need to have clear instructions so as to avoid failure. Hence, realizing the document phase called the specification book will help avoid the failure of our project. The specification book permits to specify the expectations of the customer as well as the standards that will govern the project for a good, smooth, precise and concrete project. It establishes a certain agreement between the client of the project and the person who is supposed to realize the project. It presents the objectives and needs of the system to be developed. The present specification book relates to the context and justification, the needs of the project, the plan and the deliverables.

### **CONTEXT AND JUSTIFICATION**

#### **Context**

Cameroon is a home to closely 28,697,813 million inhabitants and an estimated **37.5 percent** of the country's population lives below the poverty line; microfinance would seemingly be an ideal solution to the economic problems of the country poorest citizens. Even though microfinances try to solve a peculiar problem we have noticed that they are still some lacunae in their way of doing that is mainly in their way of recording daily collections where everything is done on papers, lack of security which leads to mothers in the market being duped, and no traceability of their collectors.

#### **Justification**

This platform (Daily Collect) will help microfinance to ease the daily collection process without wastage of time, have traceability of their transaction history in real time and also have the localization of the collector during the working time.

### **OBJECTIVES OF THE PROJECT**

#### **General Objective**

The general objective is to build an application that facilitate the daily collection process of money in a microfinance in a secured and efficient way and allow the contributor to view his past and present financial records.

#### **Specific Objectives**

The specific objectives of this project are to:

* Facilitate daily collection
* Keeping track of financial records that will help in decision making
* Improve security by showing to the contributor the collector that is assigned to him
* Have real time localization of collectors
* Improve the relationship between collector and contributor.
* Facilitate the listing of previous transaction done by the contributor.

### **EXPRESSION OF NEEDS**

#### **Functional needs**

##### **ADMINISTRATOR**

* Authenticate
* Manage contributor
* Manage collector
* View the localization of collector on field
* Assigns collectors to contributors

##### **CONTRIBUTOR**

* + - Authenticate
    - Manage account
    - Update profile
    - Scan QR code
    - View transaction history
    - View assign collector
    - Receive notification
    - Rate collector

##### **COLLECTOR**

* + - Authenticate
    - Register contributions
    - View assign contributor
    - Update profile

#### **Non-Functional needs**

Specifies the quality attribute of a software system. The judge the software system or application based on Performance, Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to its success. Failing to meet nonfunctional requirements can result in systems that fail to satisfy user needs.

* + **Performance:** defines how fast a software system, or its piece responds to certain user’s action under certain workload. In most cases, this metric explains how much a user must wait before the target operation happens (the page renders, a transaction is processed, etc.) given the overall number of users at the moment. But it’s not always like that. Performance requirement may describe background processes invisible to users. Our goal will be to provide our users with the best performance as it affects the overall user experience.
  + **Scalability:** accesses the highest workloads under which the system will still meet the performance requirements. In this project we will mainly leverage the power of cloud storage and third-party API’s.
    - * The application should have a friendly user interface (UI) and should be easy to use.
      * The code should be clear to facilitate future development and improvement.
      * The web application should be resizable when opened on any device (android phone, computer, tablet web browsers

• **Security:** it is also one of the most important aspects of any system, especially the ones dealing with sensitive user’s information.

* + - * The application should provide a strong security mechanism to reassure user’s that they can trust it for their information.
      * Some of the security principles include reassuring confidentiality, accountability, Integrity, Authentication.

### **PROJECT PLANNING**

#### **Time scheduling**

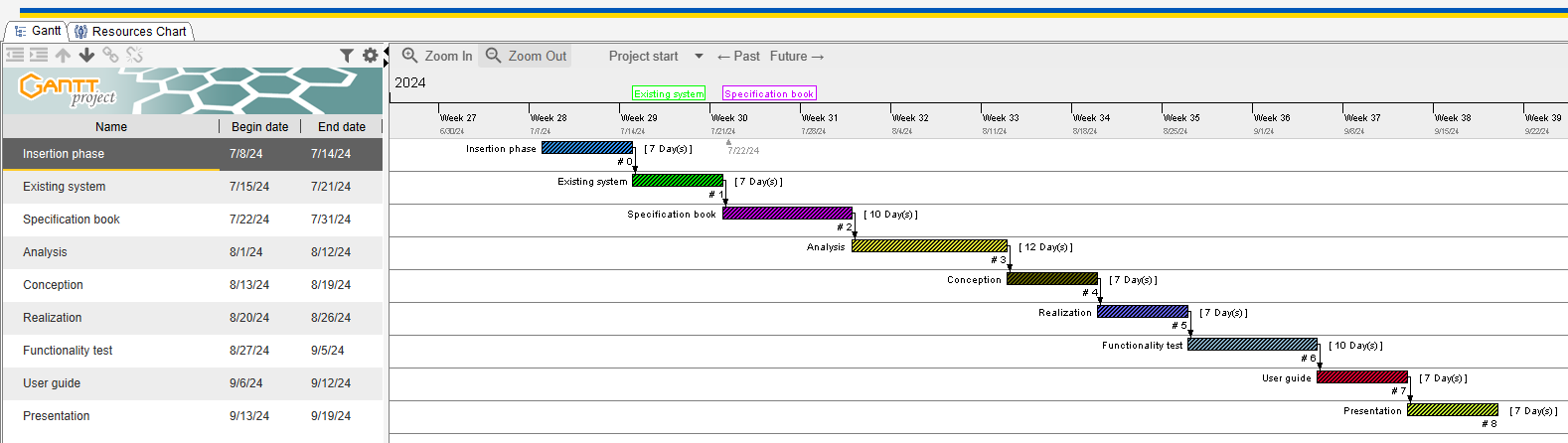
This section presents how the work phases was scheduled throughout the internship period. We will present it on a table and on a Gantt diagram

|  |  |  |  |
| --- | --- | --- | --- |
| PHASE | OBJECTIVE | OUTGOING | DURATION |
| INSERTION | Collection of information on the enterprise | Insertion book | 1 week |
| EXISTING SYSTEM | Study of the existing system | Existing System | 1 week |
| SPECIFICATION BOOK | Specification of the user needs | Specification Book | 1 week |
| ANALYSIS | Capture of needs Use case and textual description  Modelling | Analysis Book | 2 weeks |
| CONCEPTION | Preliminary conception and  Detailed conception | Conception book | 1 week |
| REALIZATION | Implementation Unitary test, Integration Test  Development,  Deployment,Component diagrams | Realization book | 2 weeks |
| TEST OF FUNCTIONALITIES | Testing of the software and debugging | Test of functionalities | 1 week |
| INSTALLATION AND USER GUIDE | Documenting software | User Guide | 1 week |

##### *Table 4:Time schedule*

#### **Gantt project**

The GANTT project software is one of the efficient tools to present the various projects tasks, their dates and the order in which they should be



##### Figure 2*: Gantt* Project

#### **Actors of the project**

##### *Table 5:Actors of the project*

|  |  |  |
| --- | --- | --- |
| NAME | FUNCTION | ROLE |
| WIRBA JORDAN WIRBA | Software engineering student at AICS-Cameroon | Analyst and developer |

### **ESTIMATION OF RESOURCES NEEDED**

#### **Hardware resources**

##### *Table 6:Hardware resources*

|  |  |  |
| --- | --- | --- |
| Materials | Quantity | Price (FCFA) |
| LAPTOP HP  Intel Core i3  500 GO, RAM 4 GO, | **1** | **250,000** |
| Printers | **1** | **500,820** |
| Local network installation | **1** | **300, 000** |
| 8GB USB Key | **1** | **6325** |
| Smart phone | **2** | **150,000** |
| Total2 | **6** | **1,207,145** |

#### **Software resources**

##### *Table 7:Software resources*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RESOURCES | DESIGNATION | USAGE | QUANTITY | UNIT COST  (FCFA) |
| Text Editor | Microsoft Office  2019 | Used for the creation of our report and  PowerPoint | 1 | **500,500** |
| Web browser | Google Chrome | View web pages | 1 | **Free** |
| Code Editor | Visual Studio  Code | For writing the code of the application | 1 | **Free** |
| Project planning | Gantt Project | For building a  Gantt chart | 1 | **Free** |
| Geo-location plan  designing tool | Ichogram | To draw the location plan | 1 | **Free** |
| UML  Analysis | Visual paradigm  (Community edition) | Modeling tool | 1 | **Free Trial** |
| TOTAL 1 |  | | **8** | **500,500** |

#### **Human resources**

##### *Table 8:Human resources*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ROLE | NUMBER OF DAYS | QUANTITY | COST PER DAY | TOTAL PRICE (FCFA) |
| Project  Manager | **90** | **01** | **30 000** | **2 700 000** |
| Analyst | **21** | **01** | **25 000** | **525 000** |
| UI/UX Designer | **07** | **01** | **20 000** | **140 000** |
| Programmer | **30** | **01** | **15 000** | **450 000** |
| Tester | **14** | **02** | **10 000** | **140 000** |
| Margin error | **/** | **/** | **/** | **1,475,000** |
| TOTAL3 |  |  |  | **4 840 000** |

#### **Total estimation of the project**

##### *Table 9:Total estimation of the project*

|  |  |
| --- | --- |
| Item | Amount (FCFA) |
| Hardware Resources | **1,207,145** |
| Software Resources | **500,500** |
| Human Resources | **4 840 000** |
| Total | **6,574,645** |

### **CONSTRAINS**

With any project, some limitations and risks need to be taken into account and addressed to ensure the project’s ultimate success. So, we have as constraints:

#### **Criterions of Acceptability**

The delivered product is judged acceptable if it respects all the need of the user and all the different functionalities that have been presented.

#### **Time constrains**

The project will be realized in 14 weeks starting from the beginning date coupled with many other school projects

### **DERIVABLES**

A deliverable is an expected result of a task or project, measurable or verifiable which results from the completion of a delivery service. During delivery, it will be a question for us to present an application with all the functional modules. In addition to this, we must present the elements below:

* A complete report including:
* The existing file
* The specification book,
* The analysis file,
* The design file,
* The production file,
* The user guide,
* A CD-ROM in which will be recorded:
* The source code of the application
* The database script,

### **CONCLUSION**

Reaching the end of this part. The list of objectives to be achieved has been enumerated and made clear. The specification book permitted us to present the different actors associated with the project as well as the requirement and the provisional planning needed for the achievement of our project. We will move directly to the next part which is the analysis phase. In the analysis phase, we will model our system with a modelling language and a unified process, do a comparative study between UML and Merise.

## **CHAPTER 3: ANLYSIS PHASE**

**Preamble**

The development of any system can be put into two major phases: Analysis and Design. The analysis document appears following the specification book, allows us to present a detailed analysis of the problem and solution, The chosen analysis method and the reasons that motivated this choice. To achieve this, we will use UML2.5 (Unified Modeling Language) with 2TUP (2 Track Unified Process) as method applied to UML to analyze the system.

**Content**

**INTRODUTION**

1. **PRESENTATION OF THE MODELING APPROACH**
2. **PRESENTATION OF UML AND 2TUP**
3. **CHOICE OF THE ANALYSIS METHOD**
4. **MODELING OF THE PROPOSED SOLUTION**

**CONCLUSION**

### **INTRODUTION**

The analysis phase is a systematic study that enables to differentiate between the parts of a problem and find solutions following a well-defined methodology. Thus, after the specifications of our needs, we are now going to do our analysis using the Unified Modeling Language (UML) Following the 2TUP process. Here, we are going to determine diagrams that intervene in the analysis phase, priority functions and take into considerations all the elements that can help in the solutions due to the fact that any change in the existing policies in an organization may require the existing information system to be restructured to have a complete development of a new information system. In case of an organization functioning manually and planning to computerize its functionality, the development of a new information system would be required.

### **PRESENTATION OF THE MODELING APPROACH**

#### **SOME ANALYSIS METHODS/APPROACH**

##### **MERISE**

MERISE stands for “Méthode d’Etude et de Réalisation Informatique pour des Systèmes d’Entreprise”. This is a method to build an automated (computerized) information system which is efficient, flexible and adapted to the organization. Like any other method of analysis and design of the Information system, MERISE includes: a **language** (vocabulary, rules of syntax...), a **process** (by level and in stages), t**ools** (software such as POWER AMC, Designer or Win Design …) and **models.** Its models listed as follows:

* CMC (Conceptual model of communication)
* CMT (Conceptual model of treatment)
* OMT (Organizational model of treatment)
* CDM (Conceptual data model)
* LDM (Logical data model)
* PDM (Physical data model)

##### **SCRUM**

In the agile Scrum world, instead of providing complete, detailed descriptions of how everything is to be done on a project, much of it is left up to the Scrum software development team. This is because the team will know best how to solve the problem they are presented. Agile scrum methodology is a project management system that relies on incremental development. Each iteration consists of two- to four-week sprints, where each sprint's goal is to build the most important features first and come out with a potentially deliverable product. More features are built into the product in subsequent sprints and are adjusted based on stakeholder and customer feedback between sprints. Whereas other project management methods emphasize building an entire product in one iteration from start to finish, agile scrum methodology focuses on delivering several iterations of a product to provide stakeholders with the highest business value in the least amount of time.

##### **UNIFIED PROCESS (UP)**

The UP is an iterative and incremental software development methodology. The Unified Process is an iterative, architecture-centric software development process driven by use cases and geared towards reducing risk. It is a process pattern that can be adapted to a wide class of software systems, to different areas of application, to different types of businesses, to different skill levels and to different sizes of the business and different data. It qualifies a process or a procedure that performs a group of operations repeatedly until a well-defined condition is met.

##### **DYNAMIC SYSTEM DEVELOPMENT METHOD (DSDM)**

It is an organized, commonsense process focused on delivering business solutions quickly and efficiently rather than just team creativity. It is similar in ways to SCRUM and XP, but it has its best uses where the time requirement is fixed.

### **PRESENTATION OF UML AND 2TUP**

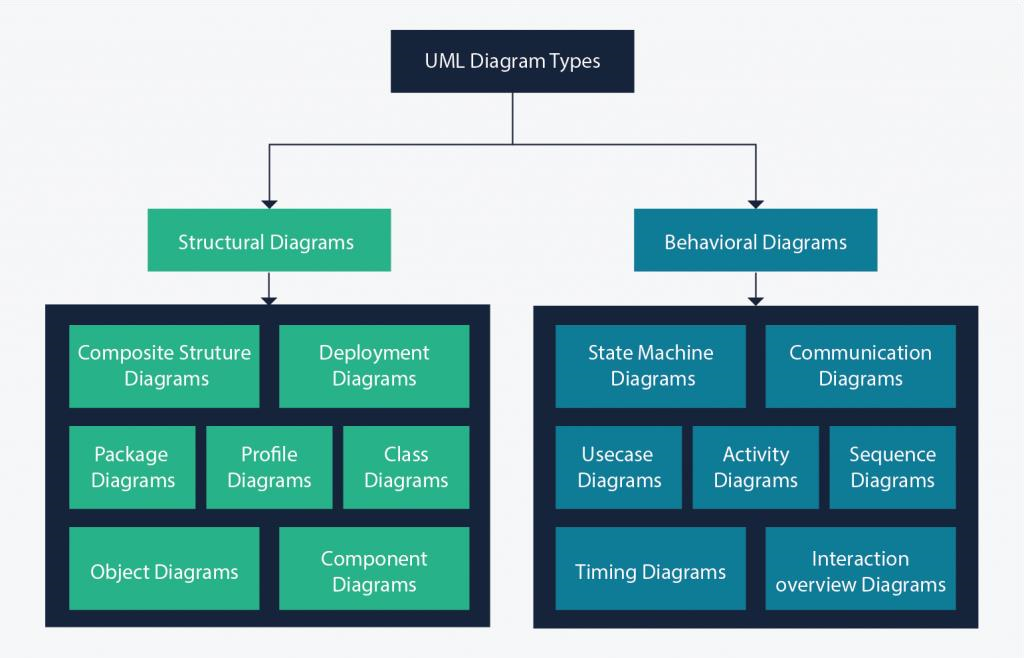
#### **UNIFIED MODELING LANGUAGE (UML)**

The Unified Modeling Language (UML) is a general-purpose visual modeling language that is used to specify, visualize, construct and document the artifacts of a software system. It captures decisions and understandings about systems that must be constructed. It is used to understand, design, browse, configure, maintain and control information about a system. It is intended for use with all development methods, life cycle stages, application domains and media. UML captures information about the static structure and the dynamic behavior of a system. A system is modeled as a collection of discrete objects that interact to perform work that ultimately benefits an outside user. UML 2.5 comprise 14 diagrams which represents the different views of a system. The 14 diagrams can be subdivided into Static or structural and Dynamic or Behavioral diagrams. These diagrams are represented in the table below:

|  |  |
| --- | --- |
| STATIC DIAGRAMS | DYNAMIC DIAGRAMS |
| 1 – Class Diagram | 8 – Use case Diagram |
| 2 – Component Diagram | 9 – Activity Diagram |
| 3 – Deployment Diagram | 10 – State Machine Diagram |
| 4 – Object Diagram | 11 – Sequence Diagram |
| 5 – Package Diagram | 12 – Communication Diagram |
| 6 – Profile Diagram | 13 – Interaction Overview Diagram |
| 7 – Composite Diagram | 14 – Timing Diagram |

##### *Table 10:UML diagrams*

The most important point to note here is that UML is not a method but a modeling language. As such, to give it an approach, we need to associate UML to a Unified Process (UP) in order to give our conception a methodology to follow. There exist several Ups but our modeling approach will be the 2TUP (Two-track unified process) which we will use in the course of our project.



##### *Figure 2:UML diagrams*

#### **COMPERATIVE BETWEEN UML AND MERISE**

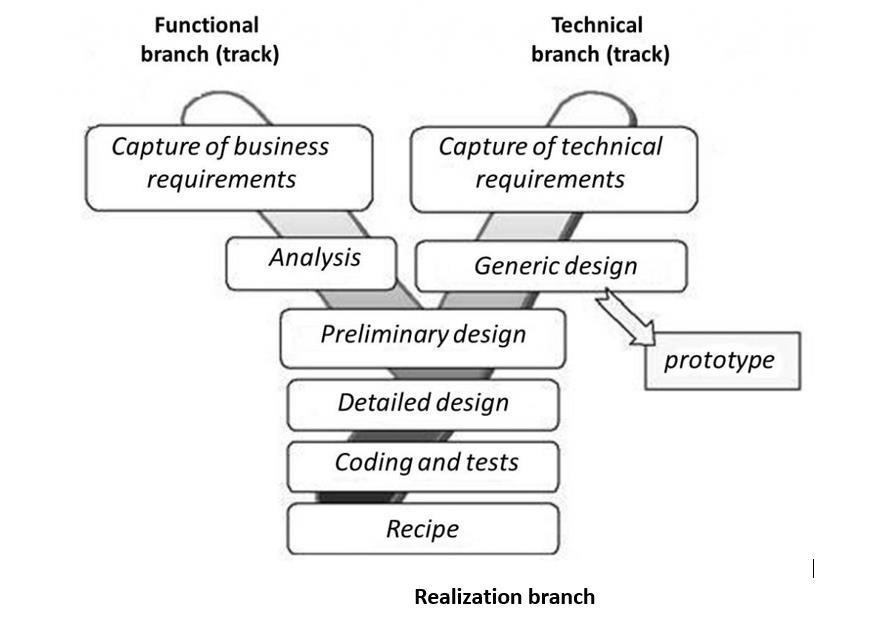
##### *Table 11:Comparative between UML and MERISE*

|  |  |
| --- | --- |
| MERISE | UML |
| - It stands for Méthode d'Étude et de Réalisation Informatique pour les Systèmes d'Entreprises. | - It stands for Unified modeling Language |
| - It is a systemic method of analysis and design of information systems. That is, it uses a systems approach. | - UML is however not a method but rather an object modeling language to which it is necessary to associate an approach to make it a method. This is the case with the 2TUP method, RUT and XP. |
| - MERISE proposes to consider the real system from two points of view:  - A static view (data)  - A dynamic view (treatments).  That is, with the MERISE method, we have a separate study of the data and the treatments. | - UML offers a different approach from that of MERISE in that it combines data and processing. Because with UML, centralizing the data of a type and the associated processing makes it possible to limit the maintenance points in the code and facilitates access to information in the event of software development. In addition, UML describes the dynamics of the information system as a set of operations attached to the objects of the system. |
| - Relational | - Object |

#### **TWO TRACK UNIFIED PROCESS (2TUP)**

In 2TUP, system analysis and design activities are generalized following 5 workflows namely; the **capture of needs**, **analysis**, **conception**, **implementation**, and **test**. 2TUP proposes life cycle in **Y**, which separates the **technical** and the **functional** aspects. It begins with preliminary studies which essentially consists of identifying the different actors of a system. This is followed by the terms of reference and the context of the project. This is articulated around 3 essential phases:

* The functional branch
* The technical branch
* The realization phase.

The following figure gives more details on how software development follows the three branches of the 2TUP.

##### *Figure 3:Branches of 2TUP*

##### **The functional branch**

This branch captures the functional needs which help in preventing the production of software that does not fit the needs of the user. The analysis here consists of studying precisely the functional specification in order to obtain an idea of what the system is going to realize and its result does not depend on any technology. This branch consists of the following stages:

* **Capture of Business (functional) needs**: This part minimizes the risk of producing an inadequate system with the needs of users and also verifies its consistency.
* **Analysis**: This is the study of specifications to find out what the system will be made of in terms of business rules. We have the following diagrams: **Use case**, **Activity**, **State Machine** and **Communication**.

##### **The Technical branch**

This branch captures all the constraints and choices related to the conception of the system, the tools and equipment as well as the integration constraint with the existing system condition. This branch consists of the following stages:

* **Capture of Technical Needs**: This stage consists of the identification of tools, materials and technologies to use in developing the system. This technical architecture will be presented in this stage.
* **Generic Design**: The technical architecture will be presented in this stage. We have the following diagrams: **Mockup design**.

##### **The Implementation branch**

The preliminary conception, the detailed conception and the documentation of the system are studied here. This branch consists of the following stages:

* **Preliminary Design**: This is the stage where the analysis model is integrated into the technical architecture. The goal here is to know what technical component will be used depending on the features from the analysis. We have the following

diagrams: **Interaction** **Overview**, **Component**, **Deployment**, **Package** and **Composite structure diagrams**.

* **Detailed Design**: This is the detailed design of each feature of the system. We have the following diagrams: **Class,** **Object**, **Sequence** and **Timing Diagrams**.
* **Coding and Tests**: This is the programming phase of the designed features, alongside testing of the coded features.
* **Recipe (Results)**: This is the validation phase of the functions of the system developed.

##### **The capture of technical requirements**

Identification of Tools, materials and technologies to use constraints (maximum response time, integration with the existing constraints) all these will lead to a first design of the technical architecture.

##### **The middle branch (Implementation phase)**

* **Preliminary Design**

This is a delicate stage in which the analysis model is integrated into technical architecture. The goal here is to know what technical component we put our feature from the analysis. We have the following diagrams:

* Component Diagram
* Deployment Diagram
* Package Diagram
* Composite Structure Diagram
* **Detailed design**

This is the detailed design of each feature of the system. We have the following diagrams:

* Class Diagram
* Object Diagram
* Sequence Diagram
* **Coding and tests**

This is the programming phase of the designed features alongside with the coded features.

### **CHOICE OF THE ANALYSIS APPROACH**

#### **JUSTIFICATION AND MOTIVATION FOR THE CHOSEN APPROACH**

Our choice for the UML modelling language and the 2TUP development processes are based on the following criteria:

* UML is a language which is centered on the user’s needs
* UML is based on the object-oriented approach
* UML produces good standards for software development
* UML has large visual elements to conduct and easy to follow
* 2TUP is a process based on object approach and is constructed in UML
* 2TUP facilitates the modeling of complex systems which will undergo many evolutions with time
* 2TUP offers a deployment cycle (the Y shape development cycle) which dissociates the technical aspects from the functional aspects.
* 2TUP is a unified process, so it processes all the characteristics of this process (Incremental, Iterative, User-oriented and component-oriented).

### **MODELING OF THE PROPOSED SOLUTION**

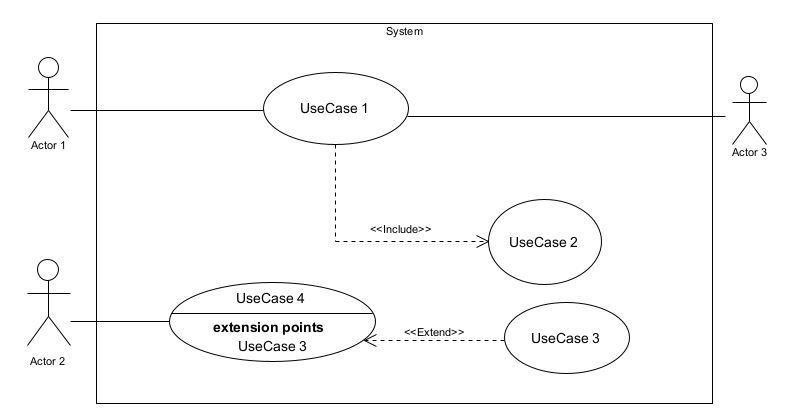
#### **USE CASE DIAGRAM**

##### **Definition**

The use case diagram identifies the functionality provided by the system, the users who interact with the system, and the interactions between them. Use cases are used in the analysis phase to define the high-level actors of the system. The main purposes of a use case diagram are:

* Provide a high-level view of what the system does,
* Identifies the users (actors) of the system
* Determine sectors which requires human-machine interfaces.

##### **Formalism**



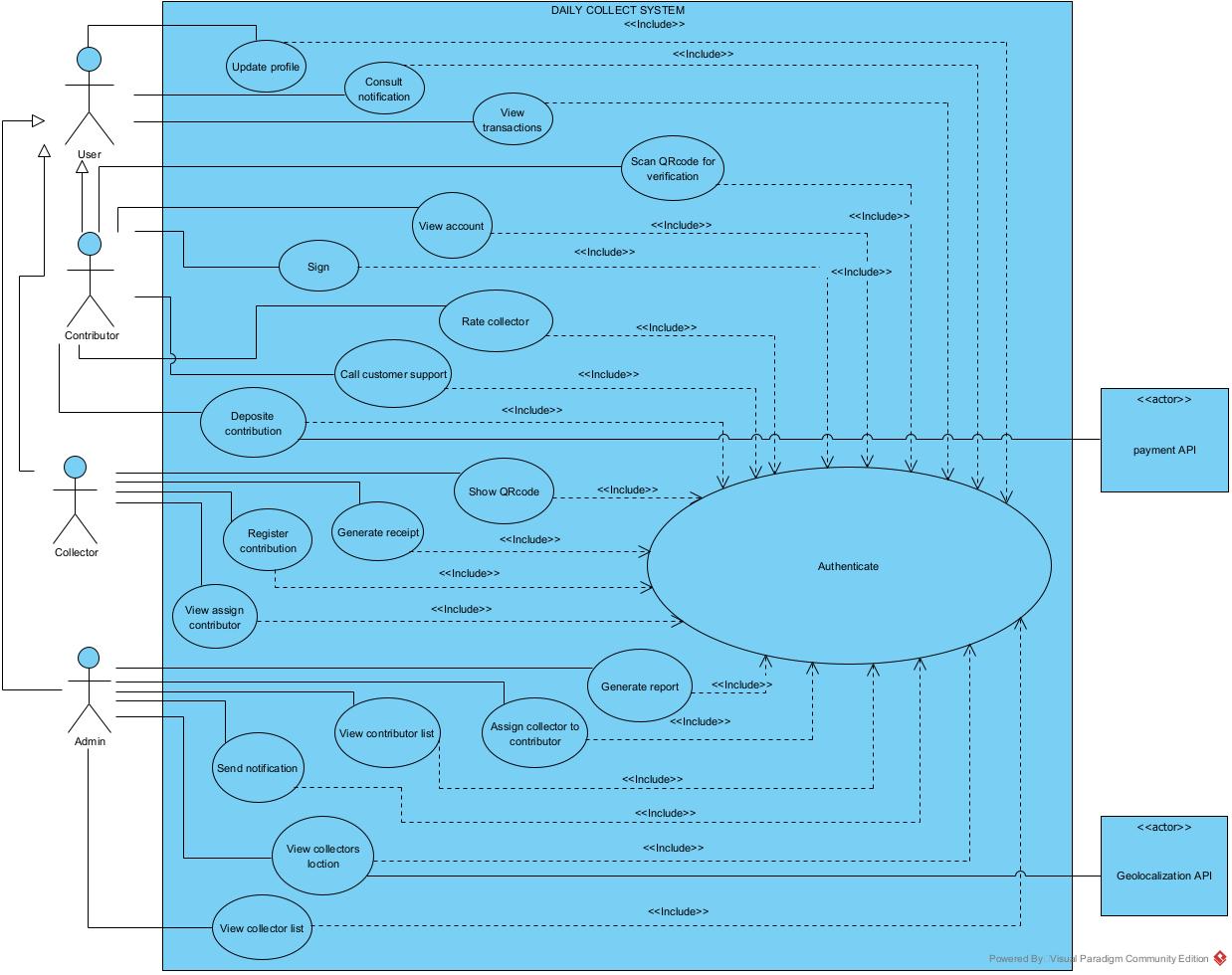
###### *Figure 4:Use case diagram formalism*

##### **Components of Use case diagram**

|  |  |  |
| --- | --- | --- |
| Element | Description of main properties | Notation |
| Actor | Represents any entity that directly interacts with the system. The actor is what performs the different possible actions of the system. |  |
| Use case | A use case represents a functionality of the system. It is an action that can be performed by an actor. | Name of use case 1 |
| Association | It indicates that an actor takes part in a use case. | B |
| Inclusion | This is a relation that denotes that an included action **must** be performed before another action is can be performed. A includes B signifies that B is a compulsory part of A. | <<Include>>  A  B |
| Exclusion | This is a relation that denotes that an action **may** be performed while another one is being performed. A use case B extends A means that B is an optional part of A. | <<Extend>>  A  B |
| Inheritance | It is the only possible relationship between actors | Acteur\_3  Acteur\_4 |
| Generalization | This shows that the use case is **kind of another**. This relation also permits to decompose a complex case into smaller and simple cases. | A  B  Actor\_1  Actor\_2 |
| System | It is a container of use case which interacts with the external actors |  |

###### *Table 12:Components of a use case diagram*

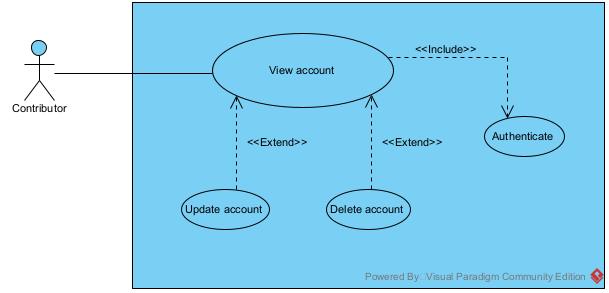
##### **Diagrams**

1. 

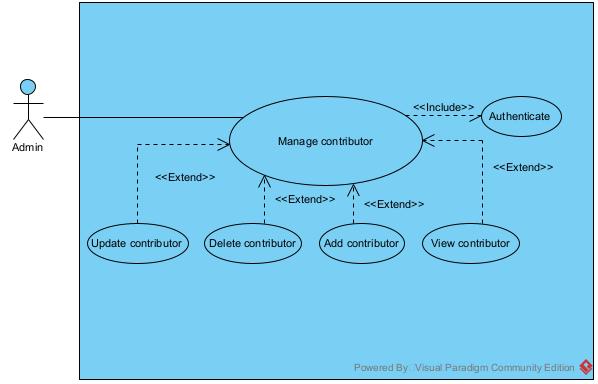
###### *Figure 5:General use case diagram of our system*

* **Detail** **Use case diagram of Manage Account**

###### *Figure 6:Detailed use case of Manage account*



* Detail use case of Manage contributor



###### *Figure 7:Detailed use case of Manage contributor*

##### **TEXTUAL DESCRIPTION OF USE CASE DIAGRAM**

###### **Definition**

UML allows the execution of a use case to be described in a textual way, in a form called nominal scenario. A nominal scenario describes in more detail the execution of a use case by an actor until it is successfully completed.

###### **Formalism**

A textual description of a use case is represented in the following form:

Table 13:Formalism of textual description

|  |  |
| --- | --- |
| Name | EXAMPLE |
| Objective | A user should login into the application |
| Actors | All users |
| Trigger | Users clicks on trigger button |
| Pre-condition | The user should be in a particular page |
| Nominal scenario | The interaction steps between the user and the system |
| Alternative scenario | What happens when an action in the nominal scenario goes wrong |
| Post condition of success | What happens when the request is successful |
| Post condition of failure | What happens when request fails |

###### **Diagrams**

Table 14:Textual description of Authenticate

|  |  |
| --- | --- |
| Name | Authenticate |
| Objective | A user should login into the application |
| Actors | Visitor/Student/Mentor/Preparation center / Administrator |
| Trigger | The user clicks on login button |
| Pre-condition | The user should be in the home page |
| Nominal scenario | 1. User clicks on login button 2. System Display login form 3. User Fills and submit form 4. System Check conformity 5. System sends query to database 6. System receives results from database 7. System checks results from the database 8. System displays home page |
| Alternative scenario | 4.a. In case the user enters wrong credentials  4.b. The system displays an error message and returns to step 2  7.a. In case an error occurs during the process  7.b. The system displays an error message and returns to step 2 |
| Post condition of success | The user’s session is displayed |
| Post condition of failure | An error message is displayed for incorrect credentials, system error or account inexistence |

Table 15:Textual description of register contribution

|  |  |
| --- | --- |
| Title | Register contribution |
| Actor: | Collector |
| Goal: | The collector should register the contribution of a particular contributor |
| Pre-condition: | The collector should be authenticated |
| Nominal scenario: | 1. The collector clicks on transaction 2. The system displays the transaction screen 3. Collector choose the contributor and fill in the amount of money 4. The system shows the password modal 5. The collector fills in the password 6. The system shows the signature pad 7. Collector signs on signature pad 8. The system verifies the password in the database. 9. The data are saved in the database.   The system route to home screen |
| Alternative Scenario (invalid password): | 8.1 The system displays invalid password  8.2 The system returns to step3 of the nominal scenario |
| Post-condition: | Successful transaction |

#### **COMMUNICATION DIAGRAM**

##### **Definition**

Communication Diagrams model the interactions between objects in a sequence. They describe both the static structure and the dynamic behavior of a system. It is a simplified version of a Collaboration Diagram introduced in UML 2.0. A communication diagram is more focused on showing the collaboration of objects rather than the time sequence.

##### **Formalism**

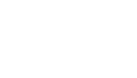


Objet\_1



1:

Message\_1



2:

Message\_2



3:

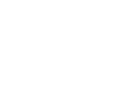
Message\_3



Objet\_4



Objet\_3



Objet\_2

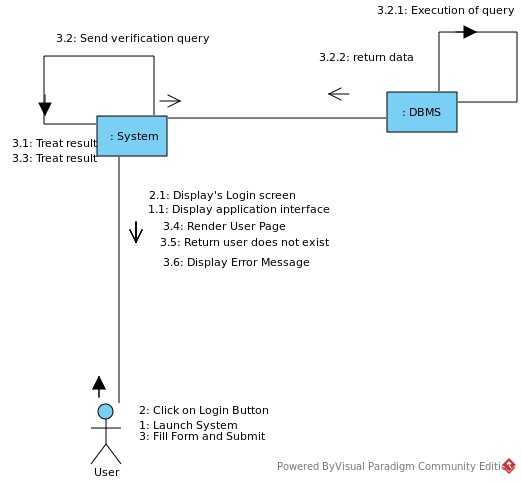
###### *Figure 8: Formalism of communication diagram*

##### **Components of Communication diagram**

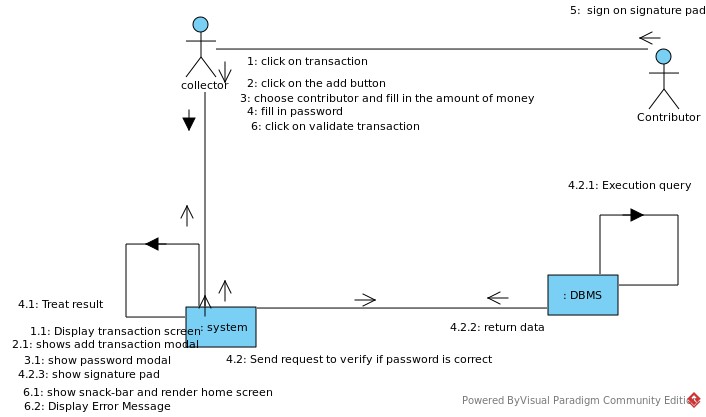
###### *Table 16:Components of Communication diagram*

|  |  |  |
| --- | --- | --- |
| ELEMENT | DESCRIPTION | NOTATION |
| Call message | A Call message define a particular communication between lifeline of the interaction that represent an innovation of the target  lifeline |  |
| Dependency | A dependency is a relationship that signifies a single or a set of model elements for their specification |  |
| Lifeline | An object represents an individual participant in the interaction conversation |  |
| Generalization | A generalization is a taxonomic relationship between a more general classifier and a more specific classifier. |  |

##### **Diagrams**



###### *Figure 9:Communication diagram of Authenticate*



###### *Figure 10:Communication diagram for register contribution*

#### **SEQUENCE DIAGRAM**

##### **Definition**

A sequence diagram is a form of interaction diagram which shows objects as life lines running down the page with their interactions over time represented as messages drawn as arrows from the source life line to the target life line. Sequence diagrams are good at showing which objects communicate with which other object and what messages trigger this communication. Sequence diagrams are not intended to show complex procedural logic.

##### **Formalism**



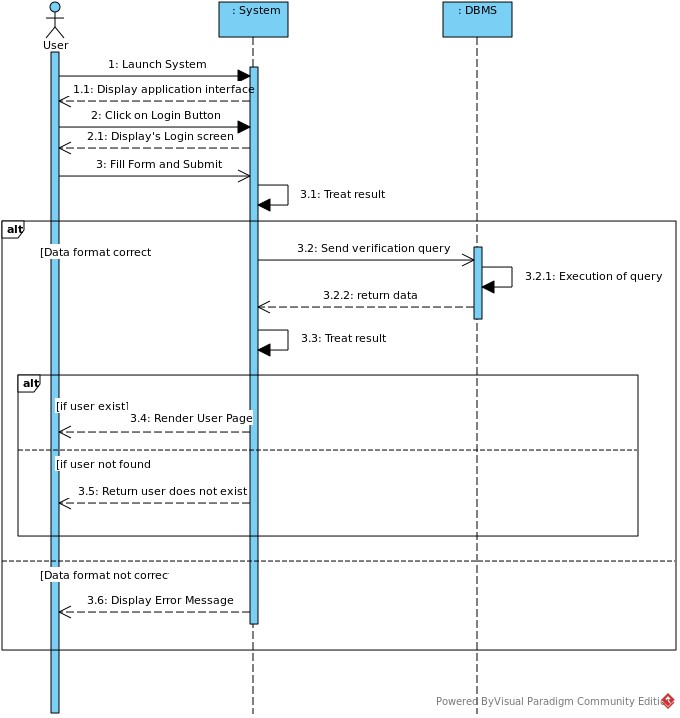
###### *Figure 11:Formalism of sequence diagram*

##### **Components of Sequence diagram**

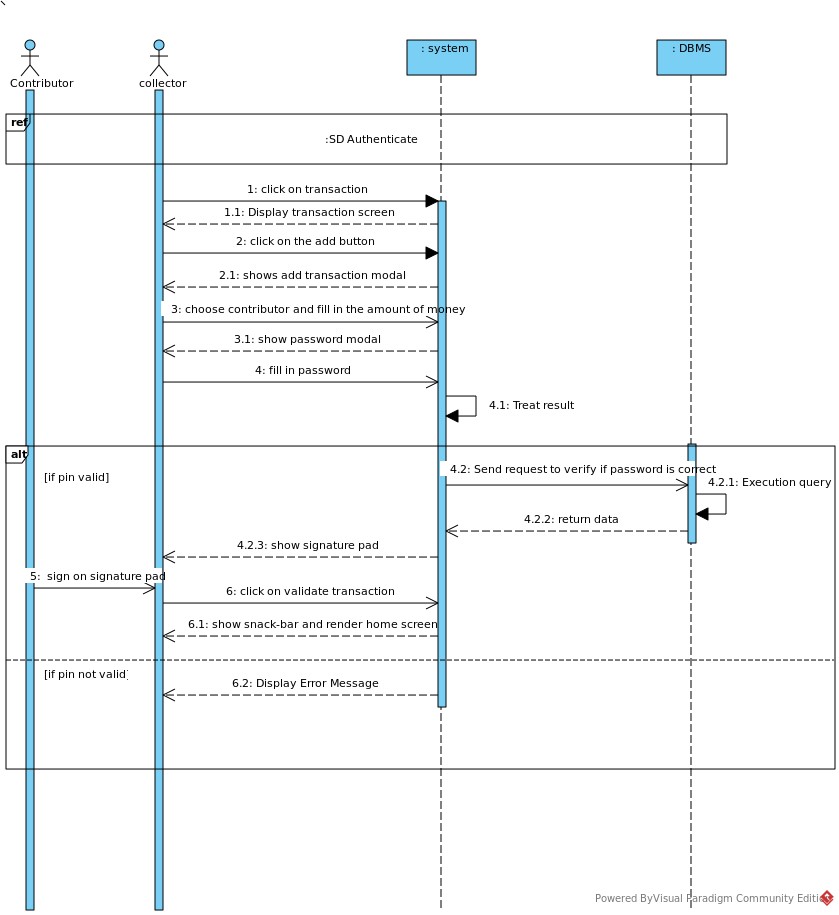
|  |  |  |
| --- | --- | --- |
| ELEMENT | DESCRIPTION | NOTATION |
| Lifeline | A lifeline represents and individual participant in a sequence diagram. A lifeline will usually have a rectangle containing its object name. |  |
| Asynchronous message | It is a message which does not require a response for the  interaction to continue. |  |
| Synchronous message | It is a message which requires a respond for the interaction to continue. |  |
| Self-message | A self-message can represent a recursive call of an operation, or one method calling another method belonging to the same object. |  |
| Return message | Return message as results of asynchronous messages. |  |
| Actor | The entities which perform actions in the system. |  |
| Fragment | It’s one or more processing sequence enclosed in a frame and executed under named circumstances. |  |
| Objects | They send and receive messages. |  |

###### *Table 17:Components of a sequence diagram*

##### **Diagrams**



###### *Figure 12:Sequence diagram for Authentication*



###### *Figure 13:Sequence diagram for register contribution*

#### **ACTIVITY DIAGRAM**

##### **Definition**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams are intended to model both computational and organizational processes. Activity diagrams show the overall flow of control. The basic purpose of activity diagrams is that it captures the dynamic behavior of the system.

##### **Formalism**

###### *Figure 14:Formalism of activity diagram*



Partition 1



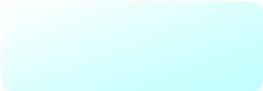
Partition 2



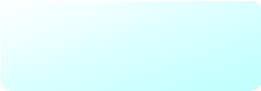
Partition 3



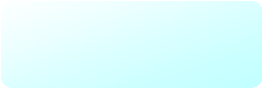
Activity\_1



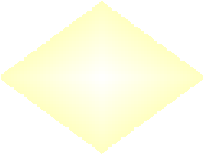
Activity\_3



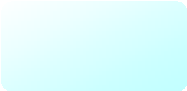
Activity\_2



Activity\_6



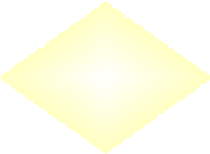
Decision\_



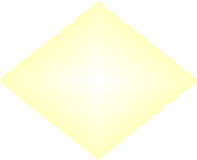
Activity\_4



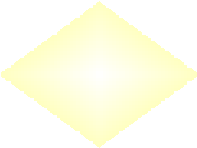
Activity\_5



Decision\_



Decision



Decision

Condition

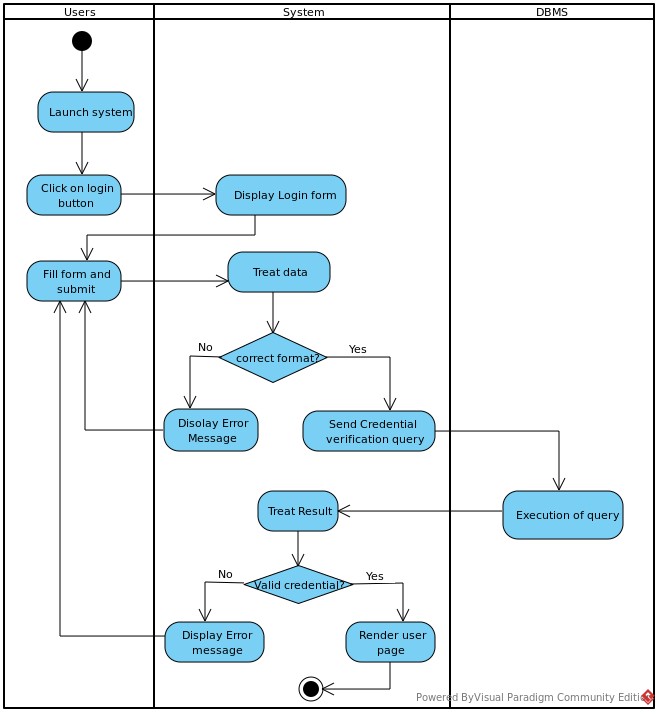
Condition

|  |  |  |
| --- | --- | --- |
| ELEMENT | DESCRIPTION AND MAIN PROPERTIES | NOTATION |
| Activity | An activity is a rounded corned rectangle enclosing all actions control flows and other elements that make up an activity. |  |
| Control Flow | It shows the flow of control form one end to the next |  |
| Initial Node | An initial node is a control node that splits a flow into multiple concurrent flows |  |
| Final Node | It denotes the end of all control flows |  |
| Decision and merge nodes | Indicates the check of a condition from a control flow |  |
| End nodes | Indicates the end of a single control flow |  |
| Swim Lanes | Used for partitioning the children in an activity diagram |  |

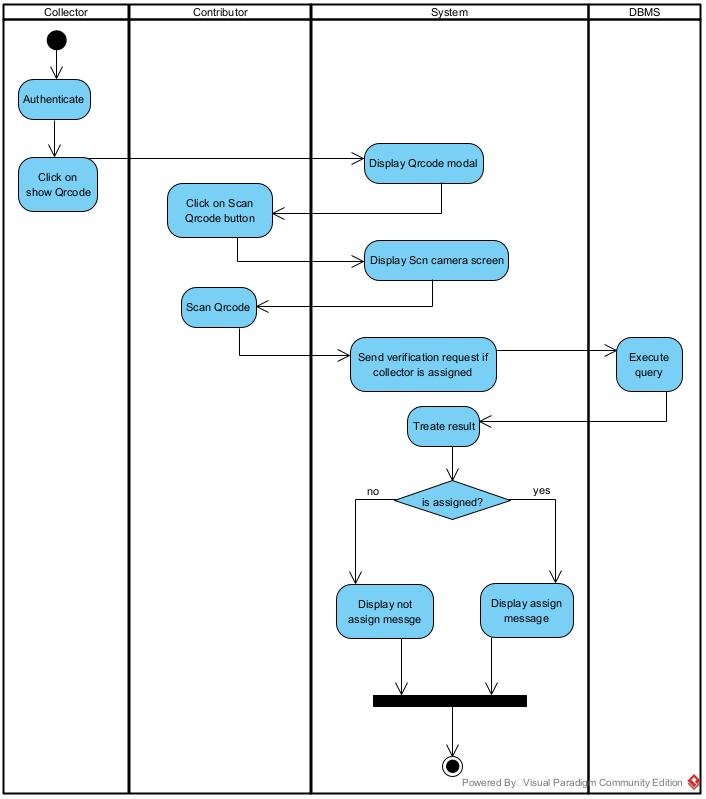
##### **Components of activity diagram**

###### *Table 18:Components of activity diagram*

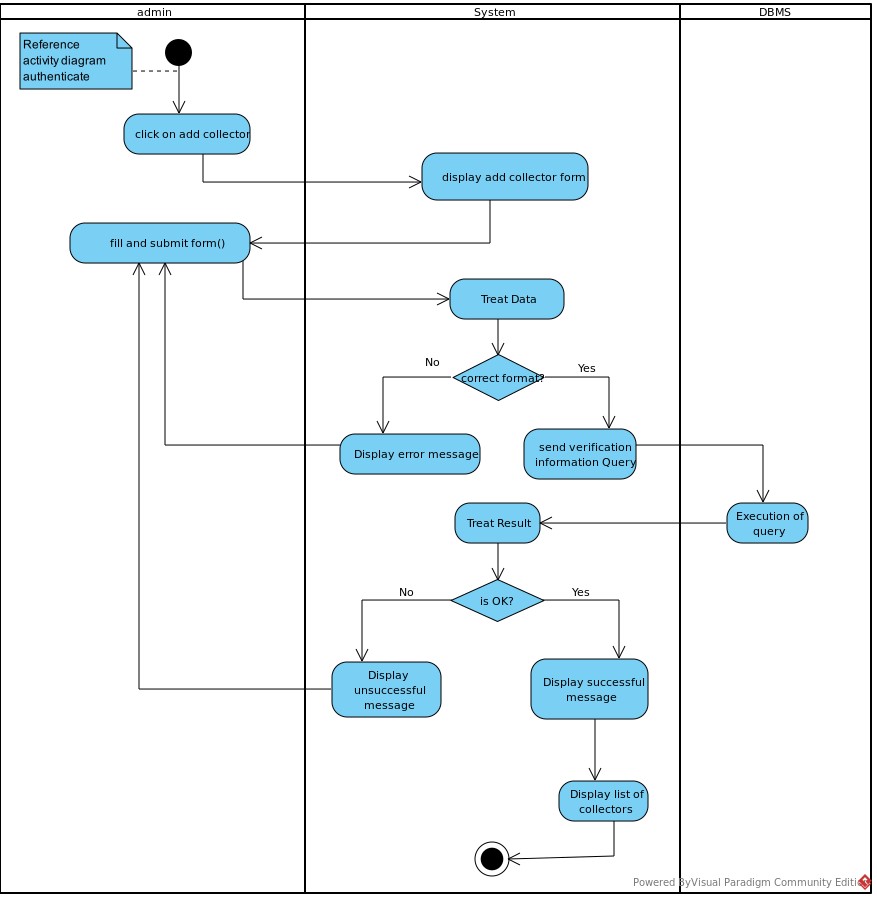
##### **Diagrams**



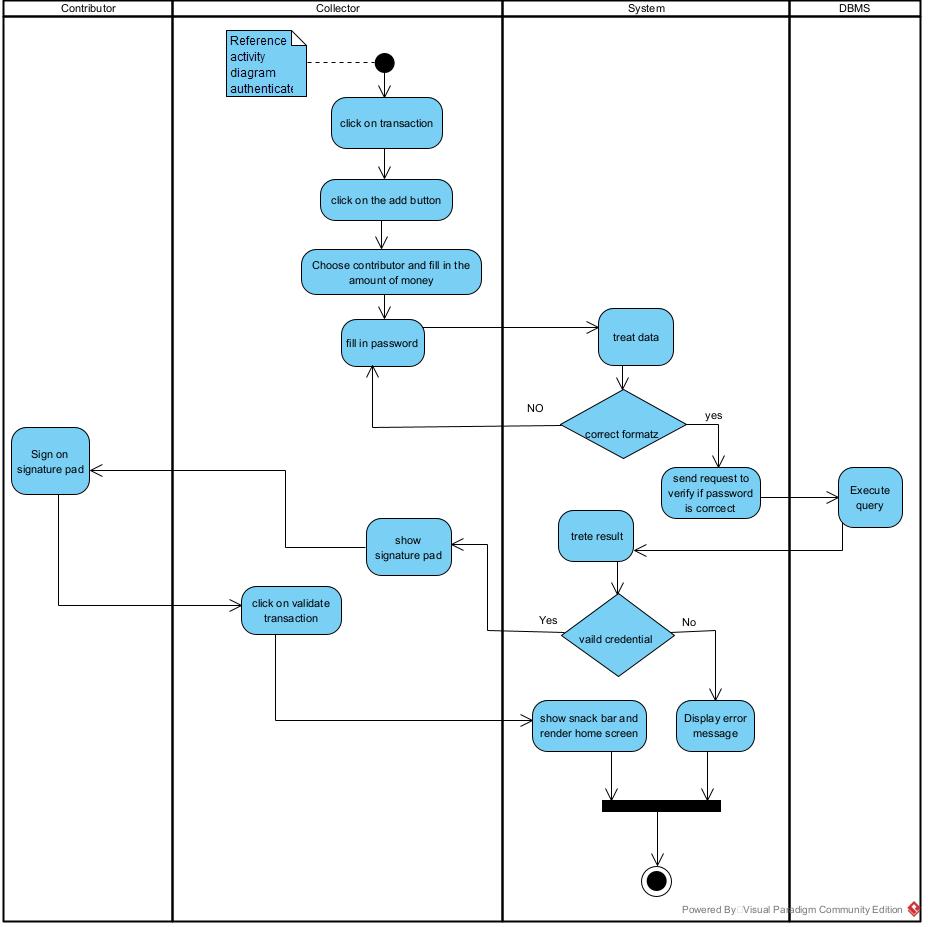
###### *Figure 15:Activity diagram for Authentication*



###### *Figure 16:Activity diagram verify assigned collector*



###### *Figure 17:Activity for Add collector*



*Figure 18:Activity diagram for Register contribution*

### **CONCLUSION**

Having reached the end of our analysis, we were asked to present, then illustrate the modeling language and the method used and finally present the diagrams of the functional branch to carry out our analysis phase. We can say that this analysis file has allowed us to well delimit the functional needs of the web application that will be designed and to have a detailed overview of the new system to be set up. The chosen modeling language and associated process will allow us to subsequently start the conception phase.

## **CHAPTER 4: CONCEPTION PHASE**

**Preamble**

The conception phase which appears directly after the analysis phase, permits us to present in a precise manner the group of components necessary for the good functioning of the software and also the architecture used for the proposed solution. It bridges the analysis and the realization phase and is a continuation of the analysis phase, which represents the technical aspects used in modeling our system.

**Content**

INTRODUCTION

I.

CLASS DIAGRAM

II.

STATE MACHINE DIAGRAM

**III**.

PACKAGE DIAGRAM

CONCLUSION

### **INTRODUTION**

The conception phase describes, in detail, the necessary specifications, features and operations that will satisfy the functional requirements of the proposed system as modelled in the analysis phase. This phase is meant to identify and consider the essential components (hardware and/or software), structure (networking capabilities), processes and procedures for the system to accomplish its objectives.

### **CLASS DIAGRAM**

#### **Definition**

A class diagram shows the building block of any object-oriented system. Class diagrams portray a static view of the model, or part of the model, describing what attributes and behavior it has rather than detailing the method for achieving operations. Class diagrams are most useful in illustrating relationships between classes and interface, composition or usage and connections respectively.

A class is a set of functions and date (attributes) that are linked together by semantic field. Classes are used in object-oriented programming to model programs and to curt a complex risk into several small simple jobs.

#### **Formalism**

##### Figure :Formalism of class diagram

#### **Components of class diagram**

##### *Figure 19:Components of a class diagram*

#### **Diagram**



##### *Figure 20:Class diagram of our system*

### **STATE MACHINE DIAGRAM**

#### **Definition**

State machine diagrams (or sometimes referred to as state diagram or state chart diagram) show the different states of an entity. State machine diagrams can also show how an entity responds to various events by changing from one state to another.

1. Formalism

request [request <= available] / pass

Transitional

behavior

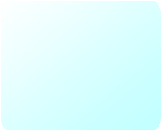
Guard

Trigger

State machine Formalism



State\_1



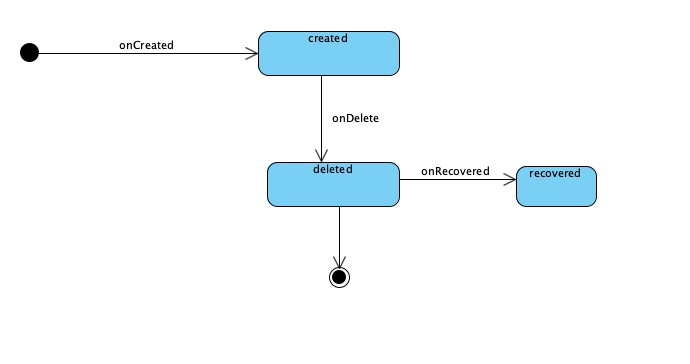
State\_2

#### **Components of state machine diagram**

##### *Table 19:Components of a state machine diagram*

|  |  |  |
| --- | --- | --- |
| ELEMENT | DESCRIPTION AND MAIN PROPERTIES | NOTATION |
| State | A state represents situations during the life of an object |  |
| Transition | A solid arrow that represents the path between different states of an object. A state can have a transition that points back to itself. |  |
| Initial state | A filled circle followed by an arrow represents an object’s initial state. |  |
| Final state | An arrow pointed to a filled circle nested inside another circle represents an object’s final state. |  |
| Join | It is used to split transition or reduce two or more transitions into one |  |

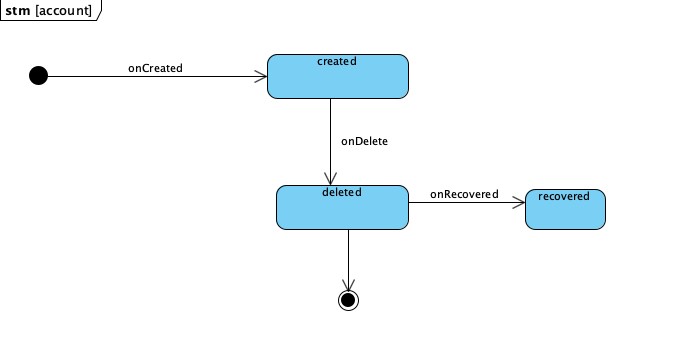
#### **Diagram**



*24*

*:*

##### *Figure 21:State machine diagram of Notification state*



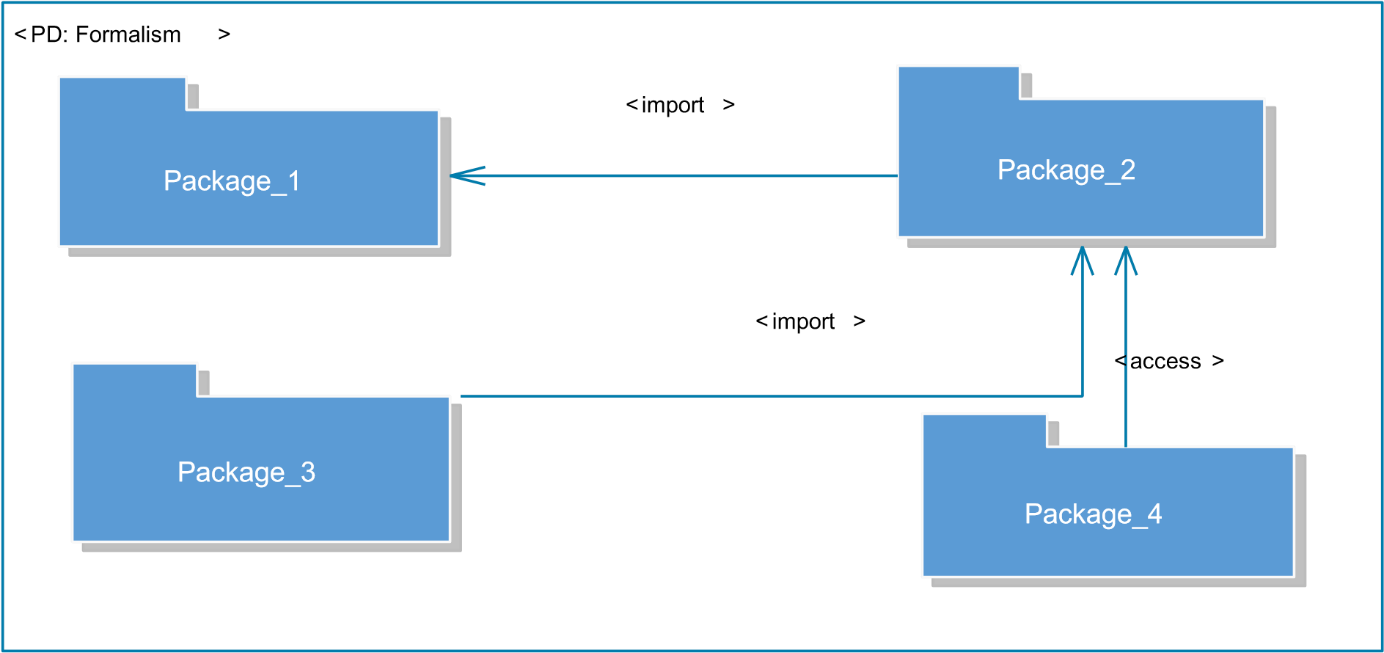
*Figure 22: Account state machine diagram*

### **PACKAGE DIAGRAM**

#### **Definition**

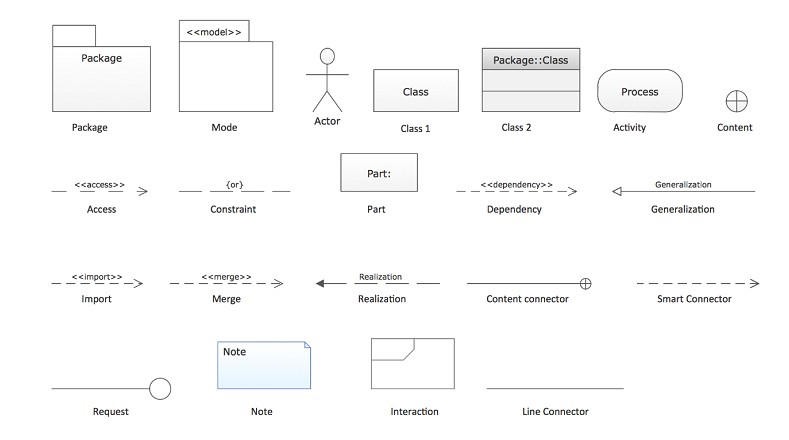
1. Package is a namespace used to group together elements that are semantically related and might change together. It is a general-purpose mechanism to organize elements into groups to provide better structure for system model. Owned members of a package should all be packageable elements. If a package is removed from a model, so are all the elements owned by the package. Package by itself is packageable element, so any package could also be a member of other packages.

#### **Formalism**



##### *Figure 22:Formalism of Package diagram*

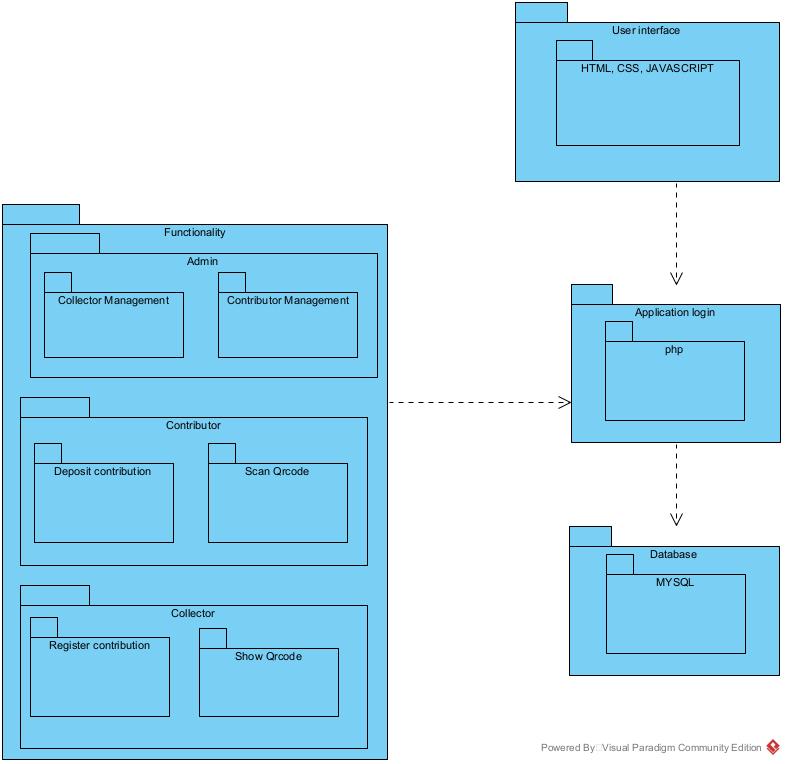
#### **Components of a package diagram**



##### *Figure 23:Components of a package diagram*

#### **Diagram**

##### *Figure 24:Package diagram of our system*



### **CONCLUSION**

In the conception phase, we set as objective to plane the different aspect of our system by showing how it will be structure and deployed within existing technical architectures. We began by considering the technical constraints for our system, after which we proceeded to identify the components of our system, how they are grouped together and how they should be deployed on appropriate deployment targets. We finished this phase by looking at interactions between the various aspects and actors of our system. The next phase of our report is the realization phase where we will look at aspects concerning the implementation of our system

## **CHAPTER 5: REALIZATION PHASE**

**Preamble**

The realization document has as aim to present in details the significant points on how our application is developed. The aim here is to present the database, the tools used for the realization of our mobile and wen application, the architecture of our application, the programming language used and the result obtained.

**Content**

**INTRODUTION**

1. **DEPLOYMENT DIAGRAM**
2. **COMPONENT DIAGRAM**
3. **ARCHITECTURE OF THE APPLICATION**
4. **TECHNOLOGICAL CHOICE**

**CONCLUSION**

### **INTRODUTION**

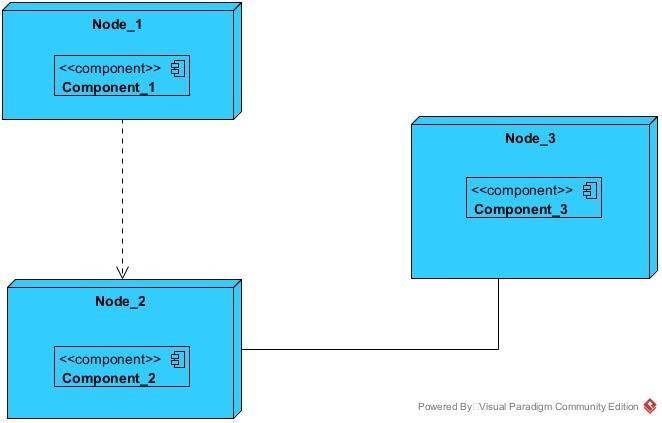
The realization document is the one that contains all the information relating to the actual implementation of a project. The latter aim to present the elements involved in the development of our standalone application, this by presenting the working environment and the various software that will come into play. In this part, we will take a look at the development tools, the programming language we have used and the presentation of the physical architecture of our theme

### **DEPLOYMENT DIAGRAM**

#### **Definition**

Deployment diagram is a structural diagram which shows the architecture of the system as deployment (distribution) of software artifacts to deployment targets. Artifacts represent concrete elements in the physical world that are the result of a development process.

#### **Formalism**



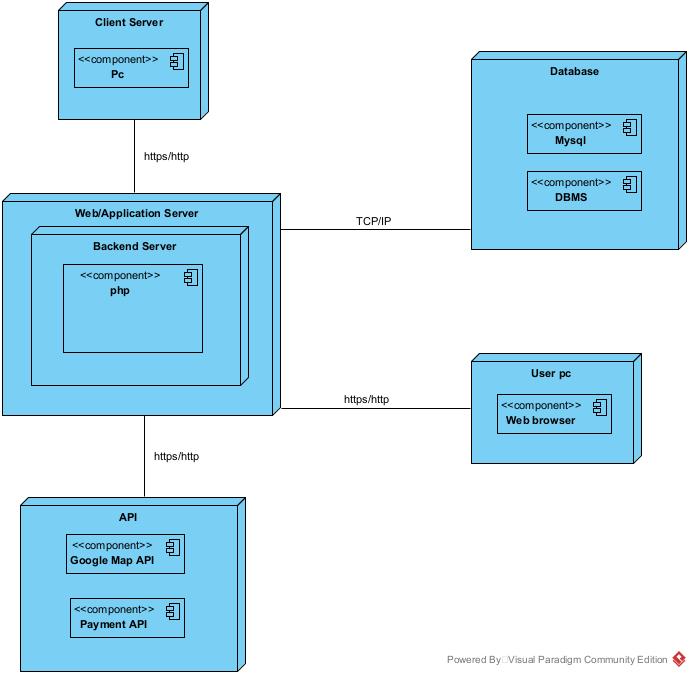
##### *Figure 25:Formalism of deployment diagram*

#### **Components of deployment diagram**

|  |  |  |
| --- | --- | --- |
| ELEMENTS | DESCRIPTION AND MAIN PROPERTIES | NOTATION |
| Node | A node is either a hardware or a software |  |
| Artefact | An artefact is a product of a software development process |  |
| Component | It represents a modular part of a system that encapsulates it content and whose manifestation is replaceable within its environment |  |
| Association | An association represents a communication path between nodes. |  |

##### *Table 20:Components of a deployment diagram*

#### **Diagram**



##### *Figure 26:Deployment diagram of our system*

### **COMPONENT DIAGRAM**

#### **Definition**

Component diagrams show the organization and dependencies involved in the implementation of a system. A Component diagram describes the organization and wiring of physical components in a system. Physical components are elements such as executables, libraries, files, documents, etc. which reside in a node. Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

#### **Formalism**



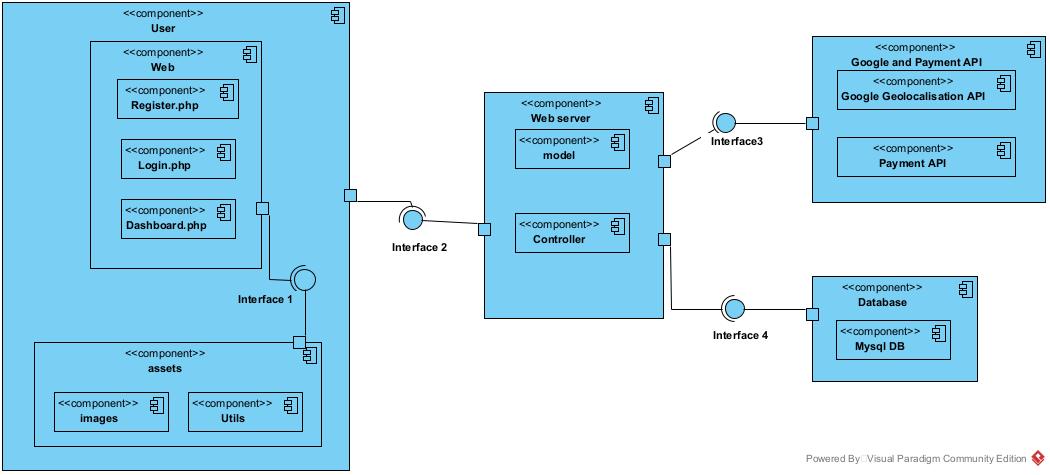
##### *Figure 27:Formalism of Component diagram*

#### **Components of component diagram**

##### *Table 21:Components of a component diagram*

|  |  |  |
| --- | --- | --- |
| ELEMENT | DESCRIPTION AND MAIN PROPERTIES | NOTATION |
| Component | It represents a module part of a system that encapsulate it content and whose manifestation is replaceable within it environment |  |
| Interface | It is a small circle or semi-circle on a stick describing a group of operations used (required) or created (provided) by components. A full circle represents an interface created or provided by the component. A semi-circle represents an interface required, like a person’s input. |  |
| Port | These are square along the edge of a system or component. A port is often used to help expose the required and provided interfaces of a component. |  |

#### **Diagram**

****

##### *Figure 28:Component diagram of our system*

### **ARCHITECTURE OF THE APPLICATION**

#### **Physical architecture**

The physical structure layer describes the system’s hardware, software, and network environment. The physical architecture is based on the non-functional requirements such as availability, performance, security. The physical architecture of our system follows the N-tier client-server architecture. Three-tier architecture is a client –server software architecture pattern in which the user interface (presentation), functional process logic (“business rules”), computer data storage and data access are developed and maintained as independent modules, most often on separate platforms. 3-tier architecture has the following layers:

##### **The user (presentation) tier**

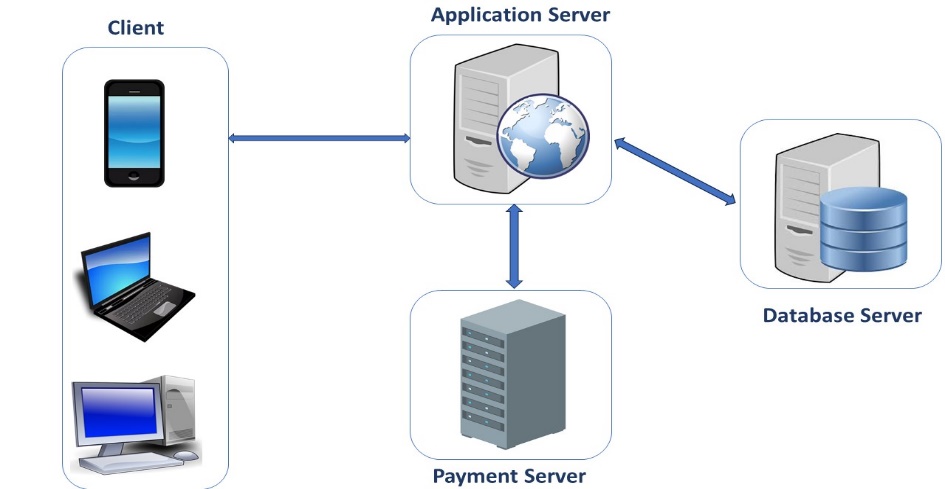
This is the tier which users can access directly. That is, this tier provides interfaces for the user and it is also responsible for transmitting users request to other tiers of the application and displaying the results of their actions back to them. It may be a web page or an operating system’s GUI (That is our case).

##### **The application (logic or middle) tier**

Here reside the application server and the programs in charge of accessing the data storage tier and providing results back to the presentation tier. It controls application functionality by performing detailed processing and reinforcing business rules.

##### **The database (data) tier**

At this layer, we found the database which holds all or almost all the applications data along with its query processing programs.



##### *Figure 29:The N-tier architecture*

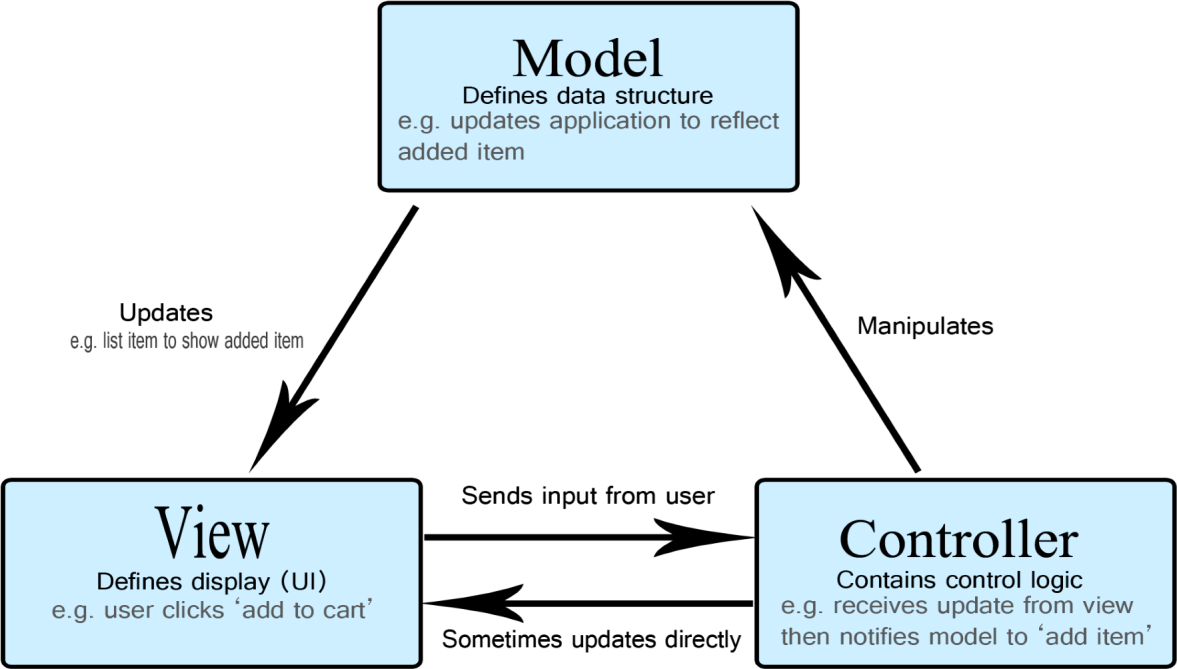
#### **Logical architecture**

The application logic of our project is based on the MVC design pattern which is a very practical way to organize its different classes in an application development project facilitating error detection, collaboration and code reusability.

The MVC (Model View Controller) design pattern is an architecture and a design method that organizes the human-machine interface (HMI) of a software application. This paradigm divides the HMI into a model (data model), a view (user interface) and a controller (control logic, event management, synchronization), each having a specific role in the interface.

The MVC design pattern allows you to organize your source code well. It will help you to know which files to create, but especially to define their roles. The goal of the MVC is to separate the logic of the code into three parts that are found in separate files.

* **Model:** This part manages the data of your application. Its role is to retrieve "raw" information from a local or remote database, organize it and assemble it so that it can be processed by the controller. This data can be stored in databases, in the cloud or in binary files.
* **View:** this part focuses on the display. It does almost no computation and just retrieves variables to know what to display. In our application we find mainly Flutter code but also some very simple loops and dart conditions.
* **Controller:** this part manages the logic of the code that makes decisions. It is the intermediary between the model and the view: the controller will ask the model for data, analyze it, make decisions and send the text to be displayed to the view. The controller contains exclusively dart code. In particular, it determines whether the user has the right to view the page or not (access rights management). Representation of an MVC architecture



##### *Figure 30:MVC design pattern*

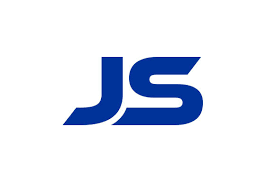
### **TECHNOLOGICAL CHOICE**

#### **Choice of languages used**

##### **Front end**

* **HTML**

HTML, or HyperText Markup Language, is the standard markup language used to create and design documents on the World Wide Web. It structures web content and consists of a series of elements or tags that define the layout, formatting, and hyperlinks of a webpage. Key components of HTML include:

* **Elements:** Fundamental building blocks, usually represented by opening and closing tags (e.g., <p>...</p> for paragraphs).
* **Attributes:** Provide additional information about elements (e.g., <a href="https://example.com">Link</a>).
* **Headings:** Defined with tags like <h1> to <h6>, indicating the hierarchy of content.
* **Lists:** Ordered (<ol>) and unordered (<ul>) lists for organizing items.
* **Images and multimedia:** Embedding images (<img>), audio, and video using respective tags.
* **JavaScript**

JavaScript is a high-level, dynamic, and interpreted programming language primarily used for enhancing the interactivity and functionality of websites. It allows developers to create interactive elements, handle events, manipulate the Document Object Model (DOM), and communicate with servers asynchronously.

Key features of JavaScript include:

* **Client-Side Scripting:** JavaScript runs in the user's browser, enabling real-time updates without requiring a page reload.
* **Event Handling:** It can respond to user actions like clicks, form submissions, and keyboard events.
* **Asynchronous Programming:** JavaScript supports asynchronous operations, allowing tasks to run concurrently (e.g., using Promises and async/await).
* **Object-Oriented and Functional:** It supports both object-oriented and functional programming paradigms, enabling flexible coding styles.
* **Cross-Platform Compatibility:** JavaScript code can run on various devices and platforms, making it a versatile choice for web development.

##### **Back end**

**PHP (Hypertext Preprocessor)**

**PHP** (Hypertext Preprocessor) is a popular server-side scripting language designed primarily for web development. It can be embedded into HTML and is widely used to create dynamic web pages and applications. PHP is open-source, which means it is free to use and has a large community of developers contributing to its improvement.

**Advantages of PHP**

1. **Open Source**: PHP is free to use, which lowers the cost of development.
2. **Cross-Platform**: It runs on various platforms (Windows, Linux, macOS), making it versatile for different server environments.
3. **Easy to Learn**: PHP has a straightforward syntax that is easy for beginners to grasp, especially for those familiar with C or Java.
4. **Rich Library Support**: PHP has a wide range of built-in libraries and frameworks (like Laravel, Symfony) that expedite development.
5. **Database Integration**: It supports various databases (MySQL, PostgreSQL, SQLite), making it easy to work with data-driven applications.
6. **Community Support**: Being widely used, PHP has a large community that offers extensive documentation, tutorials, and forums for support.
7. **Performance**: PHP is optimized for web development and can handle a large number of requests efficiently.
8. **Security**: PHP provides several security features to protect against common web vulnerabilities, such as SQL injection and cross-site scripting (XSS).
9. **Flexible and Scalable**: PHP can be used for small-scale projects as well as large enterprise-level applications, providing flexibility in development.
10. **Integration Capabilities**: It easily integrates with various web technologies, including HTML, CSS, JavaScript, and APIs.

### **CONCLUSION**

In this phase, the implementation of the proposed solution was the main subject. It began with the presentation of the various architectures, which has permitted us to present in a general manner the components that are attached to the DBMS on which our software is based, the tools used in order to put the software in place and the choice of the programming language. In the next phase, we are going to provide a functionality test.

## **CHAPTER 6: FUNCTIONALITY TEST**

**Preamble**

Once a solution is analyzed, conceived, and deployed, it is now necessary to produce a functionality test. This part of our document concerns the functionalities of the implemented application.

**Content**

INTRODUCTION

1.

APPLICATION FUNCTIONALITIES

2.

TESTS SHOWCASES

CONCLUSION

### **INTRODUTION**

The test of functionalities phase helps us to know more about the solution we are building be it web or mobile. It provides the different functionalities or modules found in our application and how they are beneficial to the users. Hence, we are going to explore the different functionalities present in Daily Collect

1. **APPLICATION FUNCTIONALITIES**

* **Authentication**

This functionality enables the users to have access to their workspace or dashboard. In case he/she does not have an account he/she will register.

* **Deposit contribution**

This functionality enables the contributor to deposite the amount of money the he/she is willing to save.

* **View contributors list**

This functionality will enable administrator to view the list of all contributors in the system

* **View collectors list**

This functionality will enable administrator to view the list of all collectors in the system

* **Notification Management**

Notifications are send by the administrator and is received by collectors and contributors.

* **Assigned a collector to a contributor**

This functionality will enable the administrator to assign a collector to the contributor.

* **Manage reports**

This functionality will enable the administrator to generate reports about users, transactions and notifications.

* **Register contribution**

This functionality will enable the collector to register the contribution of the collector

* **View transactions**

This functionality will enable the users of the system to view their transaction history

* **Log out**

This functionality will enable the users to close a working session

### **CONCLUSION**

In the test of functionalities phase we explained the different functionalities of our application and the benefits to the various users (Contributor, Collector, Administrator). This phase is essential in the understanding of the application. We will move to the last phase which is the installation and user guide where we will elaborate on how to use Daily Collect

## **CHAPTER 7: USER GUIDE**

**Preamble**

Once a solution is analyzed, conceived, and deployed, it is now necessary to produce a guide for its users. This part of our document concerns the installation of the necessary tools to set up an environment for the platform in local and the user guide.

**Content**

INTRODUCTION

I.

INSTALLATION GUIDE

II.

PERSPECTIVE

CONCLUSION

### **INTRODUTION**

This is the final phase of our report. In this phase we will walk through the requirements for our system, the necessary installation process, accessing our system and its futures, all this in a step-by-step manner to facilitate the setting up of the platform for the first-time users. The steps of different processes will be accompanied by images. After we will showscase our application by viewing the different screens and some brief explanation.

### **INSTALLATION GUIDE**

The installation of the tools for the functioning of our project will be in few steps, that is the installation of XAMPP server, bootstrap and then the importation of our project.

1. Installation of XAMPP server

XAMPP is a completely free, easy to install Apache distribution containing MariaDB, PHP, and Perl. The XAMPP open source package has been set up to be incredibly easy to install and to use.

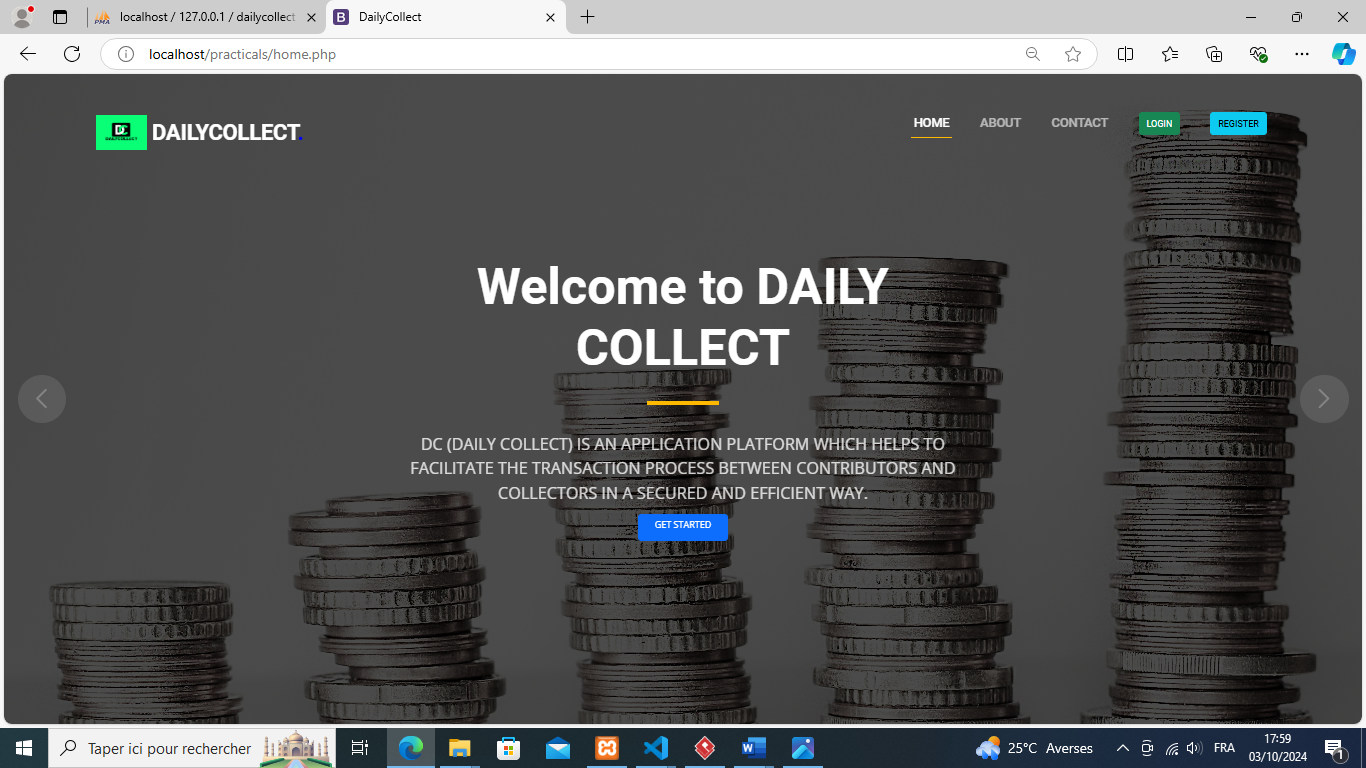
* S**tep 1:** Download XAMPP from [www.apachefriends.org](http://www.apachefriends.org).
* **Step 2:** Run the .exe file.
* **Step 3:** Start the setup wizard.
* **Step 4:** Choose software components.
* **Step 5:** Choose the installation directory and start the installation process.
* **Step 6:** Complete installation.

Click next for the next two steps.



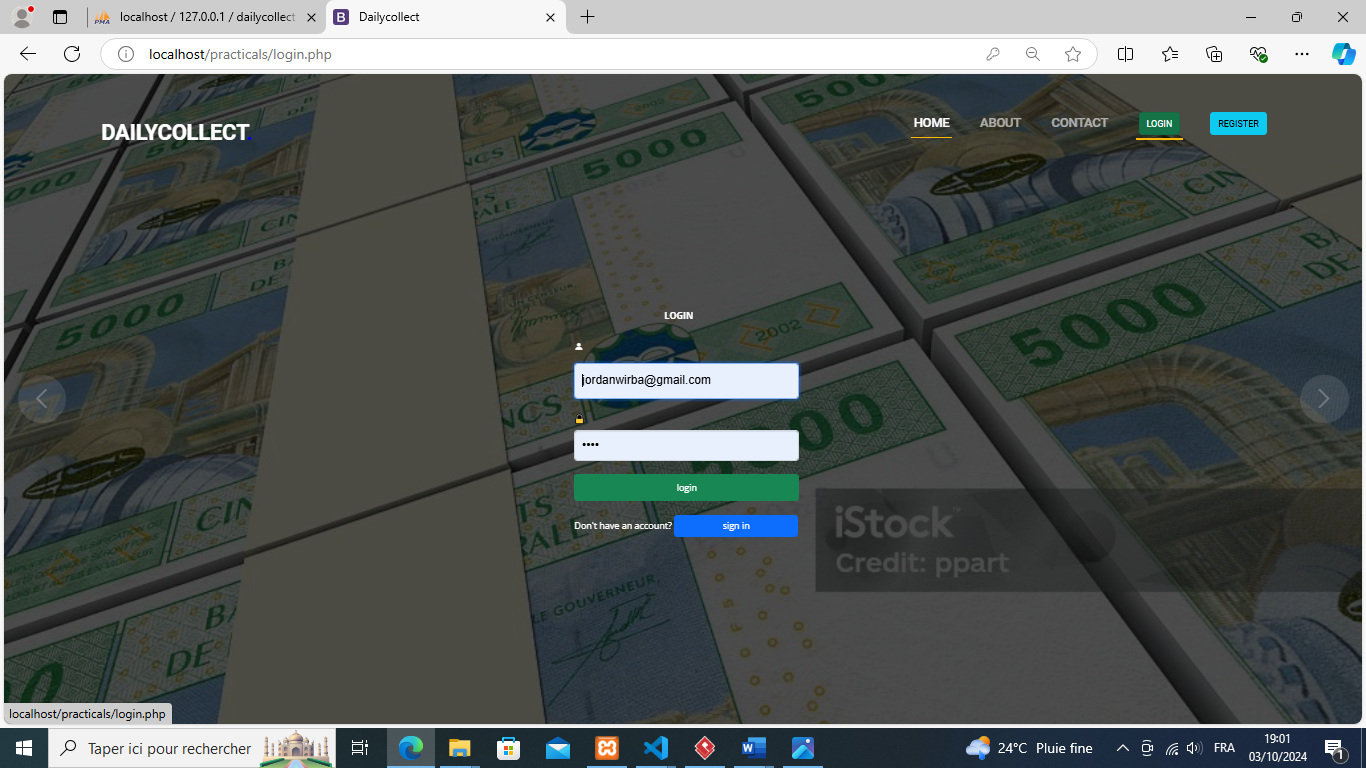


1. Here is the home page of Daily Collect system



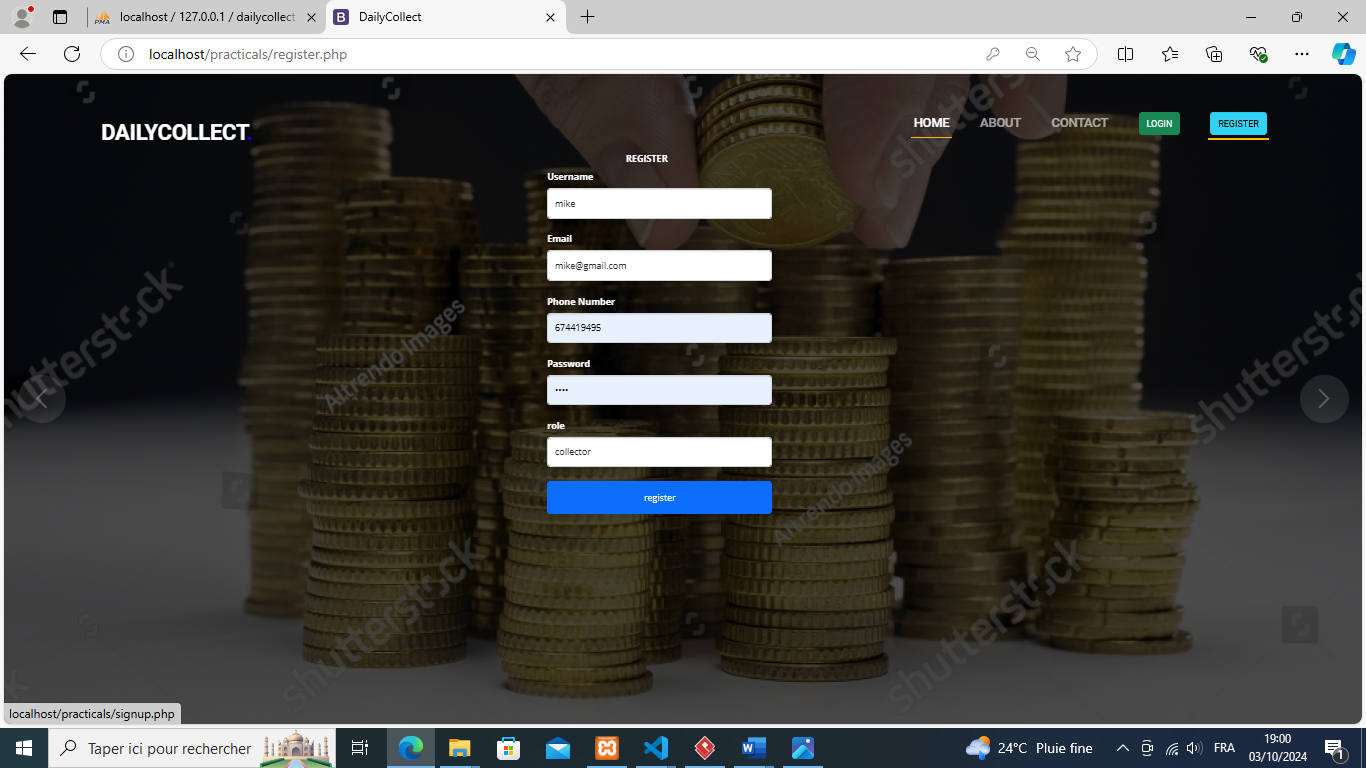
##### *Figure31: home screen of the system*

1. Here is the login screen



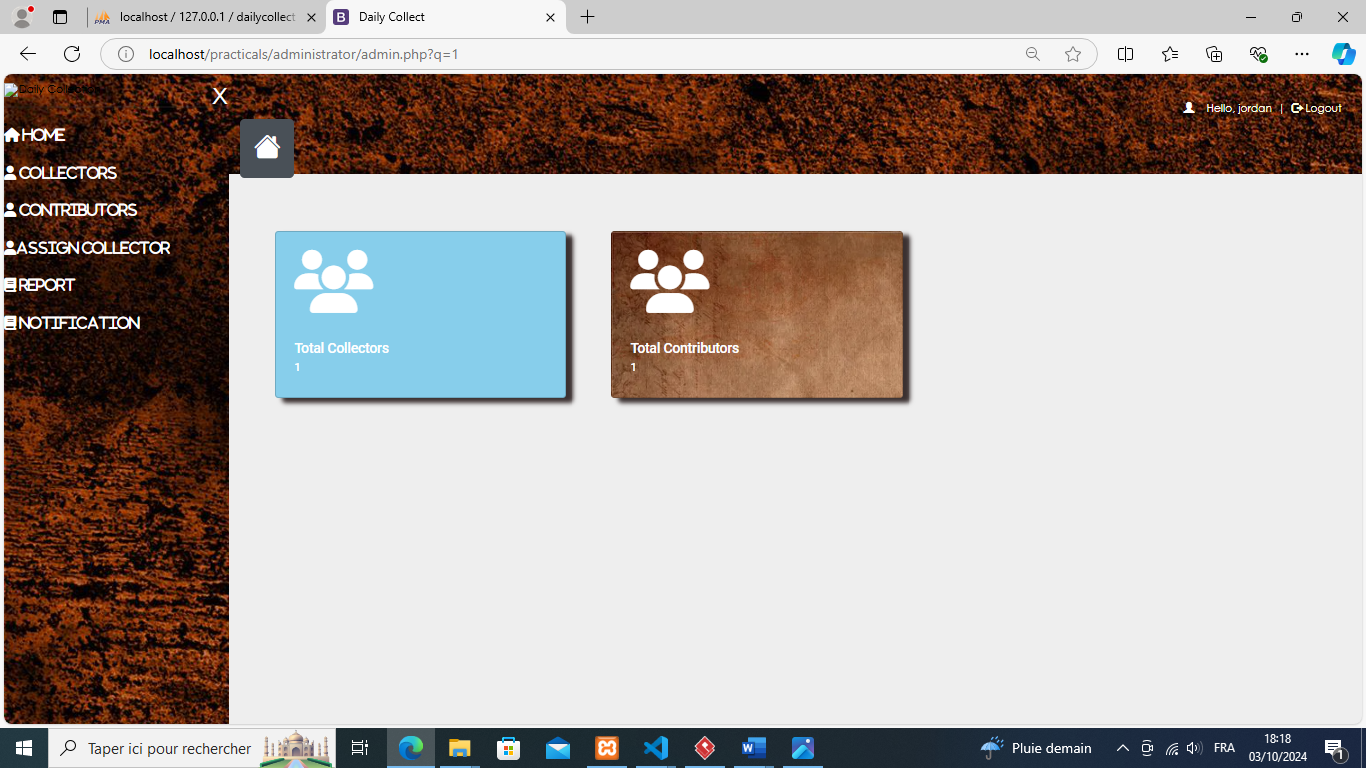
##### *Figure32: login screen*

1. Here is the registration screen



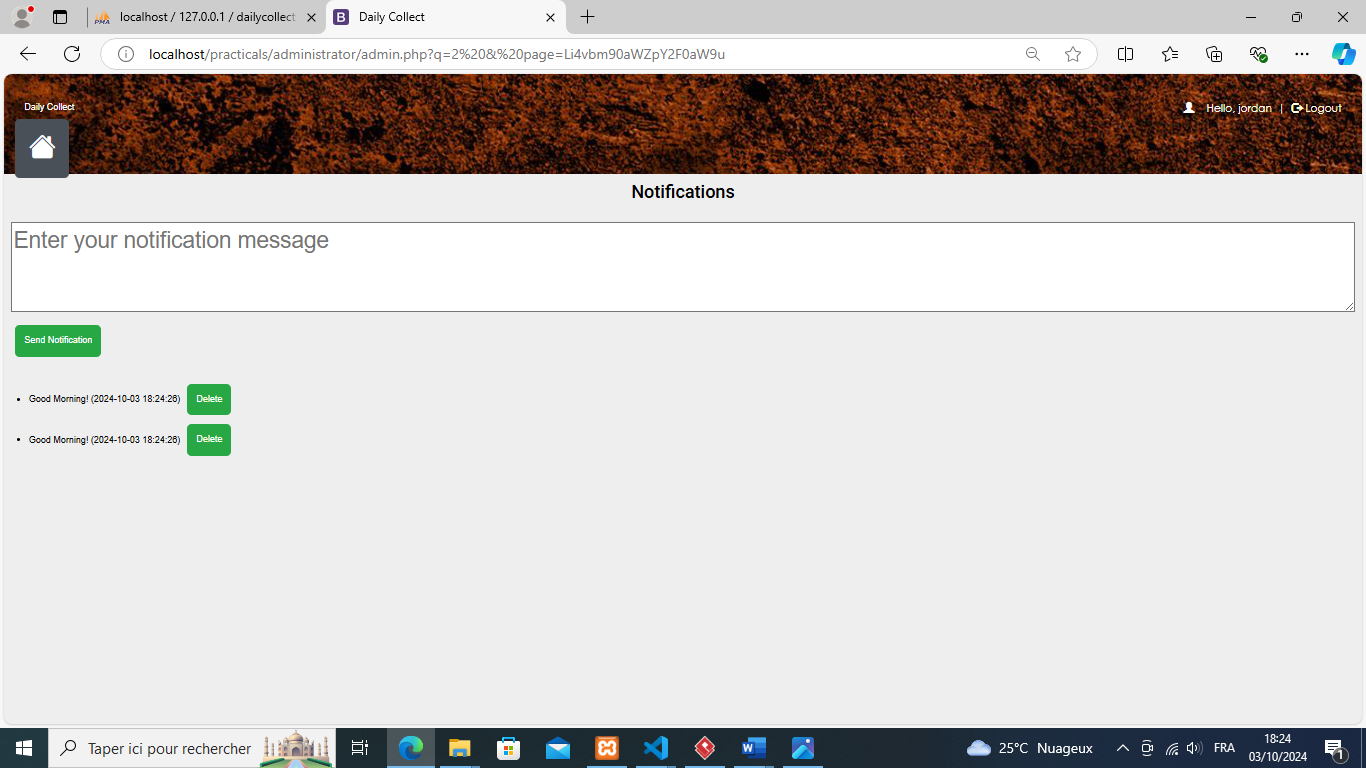
##### *Figure33: register screen*

1. Hello is the administrator dashboard



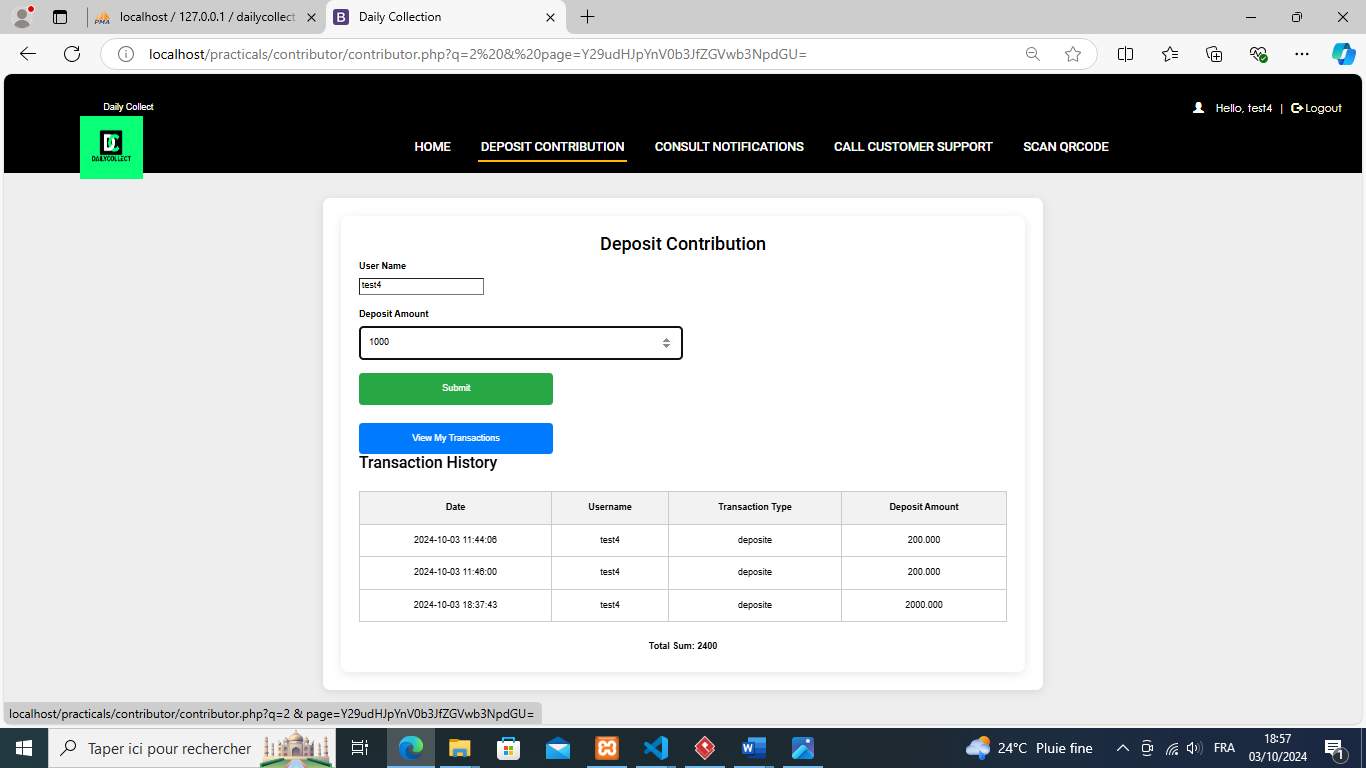
##### *Figure34: admin dashboard screen*

1. Here is the notification screen in the administrator dashboard



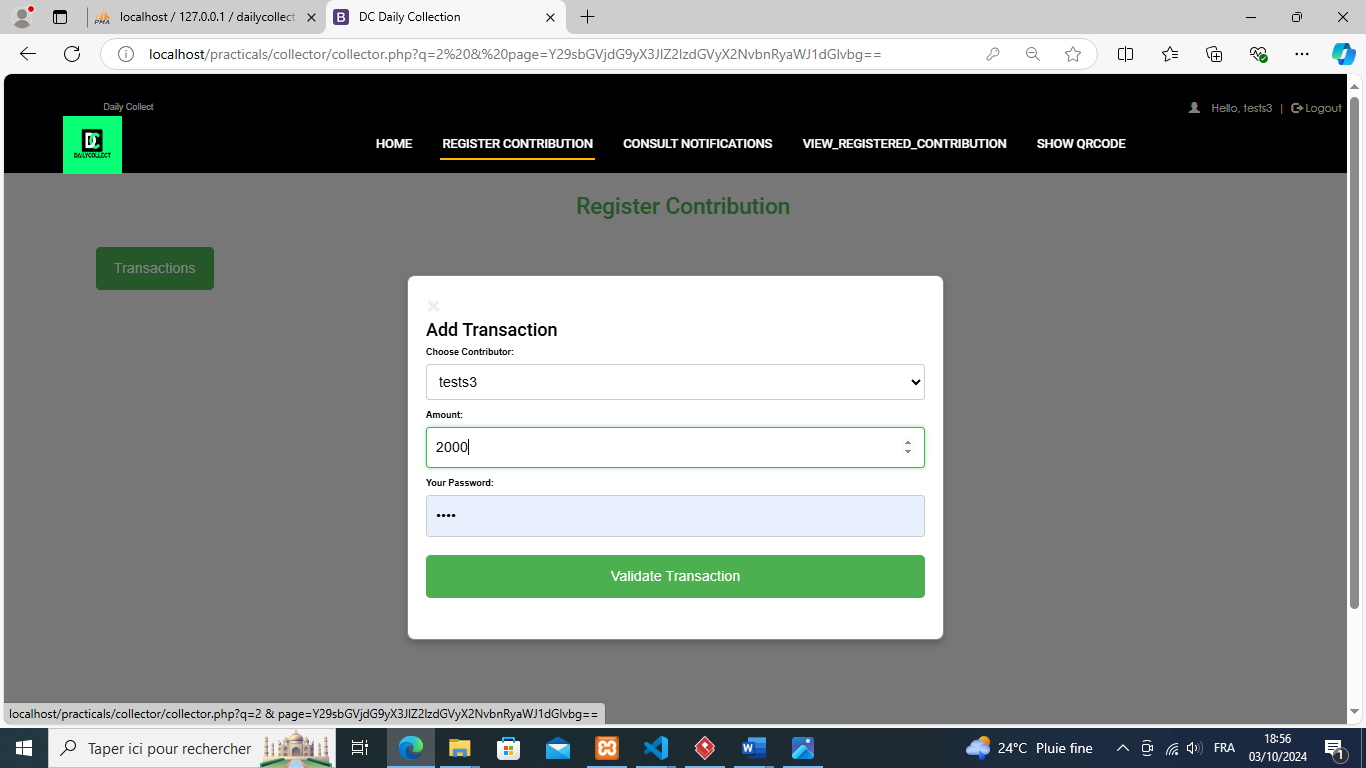
##### *Figure35: notification screen*

1. Here is the contributor dashboard screen for deposite contribution



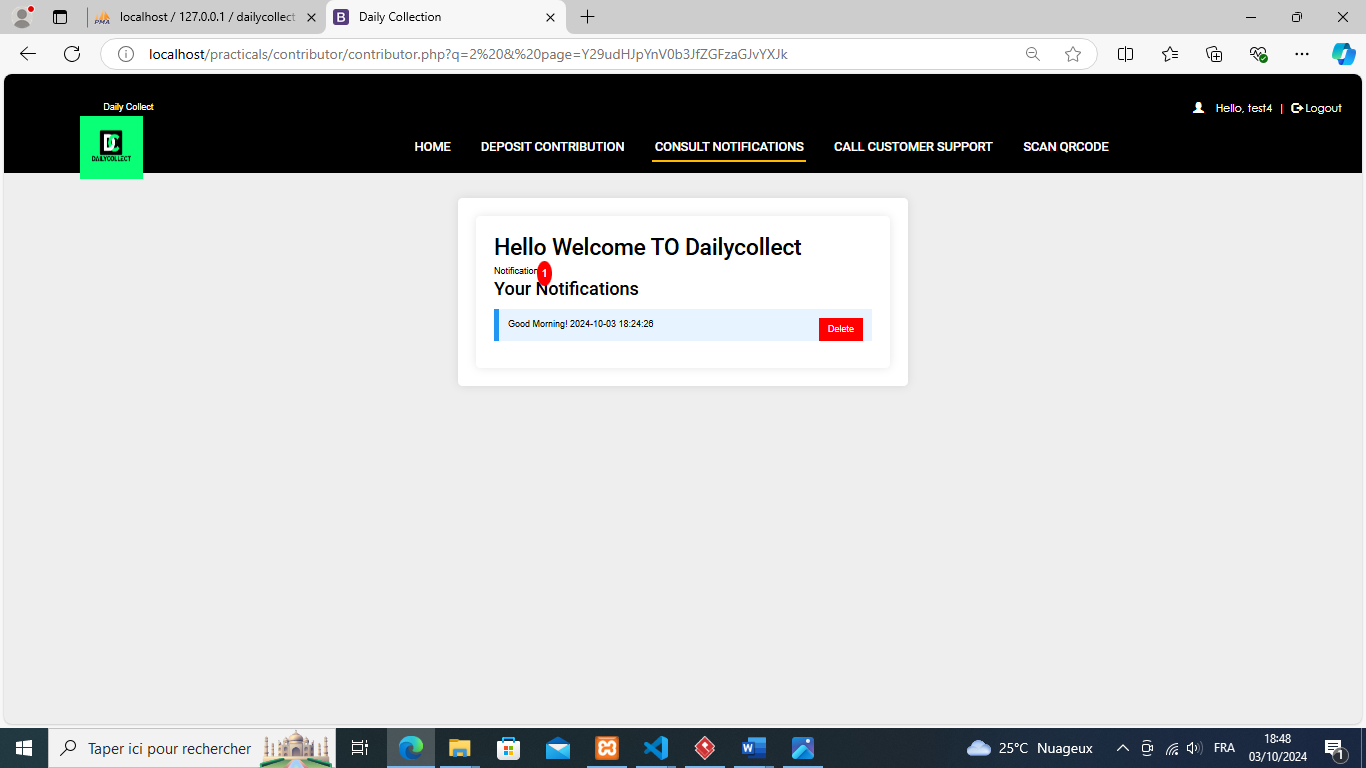
##### *Figure36: deposit contribution screen*

1. Here is the collector dashboard screen for register contribution



##### *Figure37: register contribution screen*

1. Here is the contributor dashboard consult notification screen



##### *Figure38:consult notification screen*

### **PERSPECTIVES**

Considering the fact that this project can be improved to a higher extend, as perspectives we can bring some ameliorations such as:

* Implementing a geolocalisation API for the administrator to view the location of a collector when he/she is at work(field).
* Implementing a payment API for the contributor to to carry out transaction either from mobile money, orange money or a bank account.
* Developing a mobile application for the project.
* Providing a means for the contributor to do withdrawals through the system

### **CONCLUSION**

Having put in place the platform, it was not sufficient for we had to produce a manual that will help its various users. That is why we presented the different tools to be installed and how they are to be installed in order to run this application without any problem and how the users will use this platform once the environment is set up

# **GENERAL CONCLUSION**

Having come to an end of our project, we learned how to manage stress and time with multiple school projects. For our project we worked on the theme “**COMPUTERISED PLATFORM FOR DAILY COLLECTION IN A MICROFINANCE**”, our greatest desire is to facilitate the transaction process between the collector and contributor, project constraints and requirements in the specification book. Following our theme, we did the feasibility studies, followed by the analysis of the system using the UML methodology associated to 2TUP process, from the analysis passing through conception and realization phase. The application was developed using php, html, CSS and java script some open-source API’s in order to obtain a flexible, light and secure solution for the best user experience.

# **ANEXES**

# **BIBLIOGRAPHY**

* The unified modelling Language User Guide by Graddy Booch et All.
* UML Distilled by Martin Fowler.
* Learn PHP in 24 Hours or Less by Dwight Robert.
* Object Oriented Modular Programming using HTML5, CSS by Steve Prettyman

# **WEBOGRAPHY**

* <https://www.google.com/search?q=uml+tutorial> ; for learning more about uml diagrams. 10th January 2025 11:AM
* <https://www.google.com/search?q=tuto+on+php&oq=tuto+on+php> ; to learn how to use php. 25th January 2025 2:PM
* <https://www.google.com/search?q=tuto+on+bostrap&oq=tuto+on+bostrap> ; to learn how to code the front end of the application with javascripts 10th February 2025 8:AM.
* <https://www.google.com/search?q=tuto+on+2tup&oq=tuto+on+2tup> ; to know more about 2TUP process. 20th  February 2024 10:AM
* <https://www.w3schools.com/php/> ; To learn how to code in php.25th  March 2025 12:PM
* <https://www.youtube.com/results?search_query=php+html+crud+operations> ; to know more about CRUD (create, read, update, delete) operations. 1st  April 2025 5:PM

# **TABLE OF CONTENT**

Table of Contents

[ACKNOWLEDGEMENTS ii](#_Toc199847485)

[CONTENT iii](#_Toc199847486)

[LIST OF FIGURES iv](#_Toc199847487)

[LIST OF TABLES v](#_Toc199847488)

[LIST OF ABREVIATIONS vi](#_Toc199847489)

[ABSTRACT vii](#_Toc199847490)

[RESUME viii](#_Toc199847491)

[GENERAL INTRODUCTION 1](#_Toc199847492)

[**CHAPTER 1: THE EXISTING SYSTEM** 3](#_Toc199847493)

[**INTRODUTION** 4](#_Toc199847494)

[**I.** **DESCRIPTION OF THE EXISTING SYSTEM** 5](#_Toc199847495)

[**II.** **CRITICISMS OF THE EXISTING SYSTEM** 6](#_Toc199847496)

[**III.** **PROBLEMATIC** 7](#_Toc199847497)

[**IV.** **PROPOSED SOLUTION** 7](#_Toc199847498)

[**CONCLUSION** 8](#_Toc199847499)

[**CHAPTER 2: SPECIFICATION BOOK** 9](#_Toc199847500)

[**INTRODUTION** 10](#_Toc199847501)

[**I.** **CONTEXT AND JUSTIFICATION** 11](#_Toc199847502)

[**II.** **OBJECTIVES OF THE PROJECT** 11](#_Toc199847503)

[**III.** **EXPRESSION OF NEEDS** 12](#_Toc199847504)

[**IV.** **PROJECT PLANNING** 14](#_Toc199847505)

[**V.** **ESTIMATION OF RESOURCES NEEDED** 17](#_Toc199847506)

[**VI.** **CONSTRAINS** 20](#_Toc199847507)

[**VII.** **DERIVABLES** 20](#_Toc199847508)

[**CONCLUSION** 21](#_Toc199847509)

[**CHAPTER 3: ANLYSIS PHASE** 22](#_Toc199847510)

[**INTRODUTION** 23](#_Toc199847511)

[**I.** **PRESENTATION OF THE MODELING APPROACH** 24](#_Toc199847512)

[**II.** **PRESENTATION OF UML AND 2TUP** 25](#_Toc199847513)

[**III.** **CHOICE OF THE ANALYSIS APPROACH** 32](#_Toc199847514)

[**IV.** **MODELING OF THE PROPOSED SOLUTION** 33](#_Toc199847515)

[**CONCLUSION** 53](#_Toc199847516)

[**CHAPTER 4: CONCEPTION PHASE** 54](#_Toc199847517)

[**INTRODUTION** 55](#_Toc199847518)

[**1.** **CLASS DIAGRAM** 56](#_Toc199847519)

[**2.** **STATE MACHINE DIAGRAM** 59](#_Toc199847520)

[**3.** **PACKAGE DIAGRAM** 62](#_Toc199847521)

[**CONCLUSION** 64](#_Toc199847522)

[**CHAPTER 5: REALIZATION PHASE** 65](#_Toc199847523)

[**INTRODUTION** 66](#_Toc199847524)

[**1.** **DEPLOYMENT DIAGRAM** 67](#_Toc199847525)

[**2.** **COMPONENT DIAGRAM** 70](#_Toc199847526)

[**3.** **ARCHITECTURE OF THE APPLICATION** 73](#_Toc199847527)

[**4.** **TECHNOLOGICAL CHOICE** 75](#_Toc199847528)

[**CONCLUSION** 79](#_Toc199847529)

[**CHAPTER 6: FUNCTIONALITY TEST** 80](#_Toc199847530)

[**INTRODUTION** 81](#_Toc199847531)

[**CONCLUSION** 83](#_Toc199847532)

[**CHAPTER 7: USER GUIDE** 84](#_Toc199847533)

[**INTRODUTION** 85](#_Toc199847534)

[**1.** **INSTALLATION GUIDE** 86](#_Toc199847535)

[**2.** **PERSPECTIVES** 96](#_Toc199847536)

[**CONCLUSION** 97](#_Toc199847537)

[GENERAL CONCLUSION 98](#_Toc199847538)

[ANEXES ix](#_Toc199847539)

[BIBLIOGRAPHY x](#_Toc199847540)

[WEBOGRAPHY xi](#_Toc199847541)

[TABLE OF CONTENT xii](#_Toc199847542)