REPUBLIQUE DU CAMEROUN  
Paix-Travail-Patrie

REPUBLIC OF CAMEROON  
Peace-Work-Fatherland



**INSTITUT AFRICAIN D’ INFORMATIQUE – CAMEROUN**

**CENTRE D’EXCELLENCE TECHNOLOGIQUE PAUL BIYA**

B.P: 13719 Yaoundé

Tel: +237.242.729.957 Fax: 22729958

Courriel: contact@iaicameroun.com

**AFRICAN INSTITUTE OF COMPUTER SCIENCES - CAMEROON PAUL BIYA TECHNOLOGICAL CENTER OF EXCELLENCE**

P.O. Box: 13719 Yaoundé

Tel: +237.242.729.957 Fax: 22729958

E-mail: contact@iaicameroun.com

**PERSONALIZED PROJECT REPORT**

**DESIGN AND IMPLEMENTATION OF AN ONLINE SALARY CALCULATION AND AUTOMATED BATCH PAYROLL SYSTEM**

Written By:

**MENGUE NZAMBO JULIENNE DAPHNEY**

Level II student at AICS Cameroon

**Supervised By:**

**Mme Nzali**

**Academic year  
2024-2025**

# DEDICATION

****

**TO THE ENTIRE NZAMBO FAMILY.**

# ACKNOWLEDGEMENT

Drafting this document would have not been possible without the contribution of some people who took upon themselves to see this work being accomplished. Our gratitude goes to the following people:

* The Resident Representative of AICS-Cameroon, Mr. **ARMAND Claude Abanda**, for his support, words of encouragement and the different advice on how to approach situations.
* Our academic supervisor Mr **ASANE** for his moral support, effort and advice throughout the academic year.
* To my academic teachers Mr. **MESSIO**, Mme. **NZALI** and Mrs. **TCHOUTOUO ISABELLE** for their advice and assistance in realizing this document.
* To all my teachers of AICS CAMEROON for the knowledge they impacted me with for me to be where I am today.
* A special thanks to my academic seniors, brothers and sisters who helped me all through this project from beginning to the end.
* To all my classmates for their collaborative work throughout the academic year.

# SUMMARY

[DEDICATION 1](#_Toc201590845)

[ACKNOWLEDGEMENT 2](#_Toc201590846)

[SUMMARY 3](#_Toc201590847)

[LIST OF FIGURES 4](#_Toc201590848)

[LIST OF TABLES 6](#_Toc201590849)

[GLOSARY 7](#_Toc201590850)

[ABSTRACT 8](#_Toc201590851)

[RESUME 9](#_Toc201590852)

[LIST OF ABBREVIATIONS 10](#_Toc201590853)

[GENERAL INTRODUCTION 1](#_Toc201590854)

[CHAPTER 1: EXISTING 4](#_Toc201590855)

[CHAPTER 2: SPECIFICATION BOOK 11](#_Toc201590856)

[CHAPTER 3: ANALYSIS PHASE 26](#_Toc201590857)

[CHAPTER 4: CONCEPTION PHASE 64](#_Toc201590858)

[CHAPTER 5: REALIZATION PHASE 81](#_Toc201590859)

[CHAPTER 6: TEST OF FUNCTIONALITIES 93](#_Toc201590860)

[CHAPTER 7: INSTALLATION GUIDE AND USER GUIDE 98](#_Toc201590861)

# LIST OF FIGURES

[Figure 1: Gantt project planning 19](#_Toc201682319)

[Figure 2: UML 2.5 diagrams overview (source: https://creately.com/blog/diagrams/uml-diagram-types-examples/) 30](#_Toc201682320)

[Figure 3: 2TUP diagram (source: https://www.mysciencework.com/omniscience/pervasive-mobile-healthcare-system-based on- cloud-computing 33](#_Toc201682321)

[Figure 4: Use case diagram formalism 36](#_Toc201682322)

[Figure 5: General Use Case Diagram 40](#_Toc201682323)

[Figure 6: Access Store Specific Use Case Diagram 41](#_Toc201682324)

[Figure 7: Manage store use case diagram 42](#_Toc201682325)

[Figure 8: Communication diagram formalism 47](#_Toc201682326)

[Figure 9: Authentication Communication Diagram 49](#_Toc201682327)

[Figure 10: Buy product Communication Diagram 50](#_Toc201682328)

[Figure 11: Add product Communication Diagram 51](#_Toc201682329)

[Figure 12: Formalism of a Sequence Diagram 52](#_Toc201682330)

[Figure 13: Authentication Sequence Diagram 54](#_Toc201682331)

[Figure 14: Buy product Sequence Diagram 55](#_Toc201682332)

[Figure 15: Add product Sequence Diagram 56](#_Toc201682333)

[Figure 16: Formalism of an Activity Diagram 57](#_Toc201682334)

[Figure 17: Authentication Activity Diagram 60](#_Toc201682335)

[Figure 18: Buy Product Activity Diagram 61](#_Toc201682336)

[Figure 19: Add Product Activity Diagram 62](#_Toc201682337)

[Figure 20: Physical Architecture 67](#_Toc201682338)

[Figure 21: The MVC architecture Source: https://www.freecodecamp.org/news/the-model-view-controller-pattern- mvcarchitecture-and-frameworks-explained 68](#_Toc201682339)

[Figure 22: Formalism of a Class Diagram 69](#_Toc201682340)

[Figure 23: Materio Class Diagram 72](#_Toc201682341)

[Figure 24: Formalism of a State Machine Diagram 73](#_Toc201682342)

[Figure 25: Order state machine diagram 76](#_Toc201682343)

[Figure 26: Formalism of a Package Diagram 77](#_Toc201682344)

[Figure 27: Materio Package Diagram 79](#_Toc201682345)

[Figure 28: Formalism of a Deployment Diagram 83](#_Toc201682346)

[Figure 29: Materio Deployment Diagram 85](#_Toc201682347)

[Figure 30: Formalism of Component Diagram (Source: https://www.pinterest.com/pin/551128073157994549/) 86](#_Toc201682348)

[Figure 31: Materio Component Diagram 89](#_Toc201682349)

[Figure 32: /POST/http://127.0.0.1:8000/auth/register/ 96](#_Toc201682350)

[Figure 33: /GET/http://127.0.0.1:8000/auth/get\_user\_profile/ 96](#_Toc201682351)

[Figure 34: Materio Home Page 102](#_Toc201682352)

[Figure 35: : Materio Store 103](#_Toc201682353)

[Figure 36: Materio Login Page 104](#_Toc201682354)

[Figure 37: Materio Signup Page 105](#_Toc201682355)

# LIST OF TABLES

[Table 1: Existing system name, description and country 7](#_Toc201663410)

[Table 2: Criticism of the existing system 8](#_Toc201663411)

[Table 3: Chronogram of activities 17](#_Toc201663412)

[Table 4: Actors of the Project 18](#_Toc201663413)

[Table 5: Software Resources (source: Mercurial 2025) 20](#_Toc201663414)

[Table 6: Hardware Resources (source: Mercurial 2025) 21](#_Toc201663415)

[Table 7: Human Resources (source: Mercurial 2025) 22](#_Toc201663416)

[Table 8: Global Estimation 23](#_Toc201663417)

[Table 9: Differences between UML and MERISE 31](#_Toc201663418)

[Table 10: Use Case Diagram Components 37](#_Toc201663419)

[Table 11: Actors of Our System 39](#_Toc201663420)

[Table 12: Buy Product Textual Description 43](#_Toc201663421)

[Table 13: Manage Store Specific Textual Description 45](#_Toc201663422)

[Table 14: Components of a Communication Diagram 48](#_Toc201663423)

[Table 15: Components of an Activity Diagram 58](#_Toc201663424)

[Table 16: Components of a Class Diagram 70](#_Toc201663425)

[Table 17: Components of a State Machine Diagram 74](#_Toc201663426)

[Table 18: Components of a Package Diagram 78](#_Toc201663427)

[Table 19: Components of a Deployment Diagram 84](#_Toc201663428)

[Table 20: Components of a Component Diagram 87](#_Toc201663429)

[Table 21: Softwares used for the development of Materio 90](#_Toc201663430)

# GLOSARY

**2TUP:** Two Track Unified Process.

**AICS:** African Institute of Computer Sciences.

**ERD:** Entity Relational Diagram.

**MVT:** Model View Templates.

**UML:** Unified Modelling Language

# ABSTRACT

In the domain of construction, getting the right materials for a construction project can be a nightmare. The current system is slow, frustrating, and riddled with problems. From drowning in paperwork to unreliable deliveries and unexpected stock issues, the process can stall projects and cost contractors both time and money. Our solution here details the conception and implementation of a web application designed to revolutionize the construction industry by streamlining the ordering and delivery of materials. The application tackles the inefficiencies of traditional methods, where contractors face time-consuming processes, limited supplier options, and potential for errors. For this reason, we decided to develop an application to facilitate the acquisition of construction materials. This application known as MATERIO is a web application that aims to empower contractors to save time and resources while ensuring timely delivery of the correct materials chosen from a wide variety of products directly to their project sites. We explore the conceptualization phase, outlining the identified need within the construction sector and the application’s functionalities designed to address it. We also talk about the application’s potential impact on the construction industry, promoting a more streamlined and cost-effective approach to material management. To accomplish this project, we will use UML (Unified Modelling Language) as our modelling language coupled with 2TUP (2 Track Unified Process) to form a method.

Keywords:

 Construction.  Contractors.  Materials.

 MATERIO.

 Streamlined

# RESUME

Dans le domaine de la construction, obtenir les bons matériaux pour un projet de construction peut être un cauchemar. Le système actuel est lent, frustrant et semé d'embûches. De la noyade dans la paperasse aux livraisons peu fiables et aux ruptures de stock inattendues, le processus peut retarder les projets et coûter aux entrepreneurs du temps et de l'argent. Notre solution détaille ici la conception et la mise en œuvre d'une application web destinée à révolutionner le secteur de la construction en rationalisant la commande et la livraison des matériaux. L'application s'attaque aux inefficacités des méthodes traditionnelles, où les entrepreneurs sont confrontés à des processus longs, à des options limitées de fournisseurs et à des risques d'erreurs. C'est pourquoi nous avons décidé de développer une application pour faciliter l'acquisition de matériaux de construction. Cette application, appelée MATERIO, est une application web qui vise à permettre aux entrepreneurs de gagner du temps et des ressources tout en garantissant la livraison en temps voulu des matériaux corrects choisis parmi une large variété de produits directement sur leurs sites de projet. Nous explorons la phase de conceptualisation, en soulignant le besoin identifié dans le secteur de la construction et les fonctionnalités de l'application conçues pour y répondre. Nous évoquons également l'impact potentiel de l'application sur le secteur de la construction, en promouvant une approche plus rationnelle et plus rentable de la gestion des matériaux. Pour mener à bien ce projet, nous utiliserons UML (Unified Modelling Language) comme langage de modélisation, associé à la méthode 2TUP (2 Track Unified Process).

Mots-clés :

 Construction.  Entrepreneurs.  Matériaux.

 MATERIO.

 Rationalisé.

# LIST OF ABBREVIATIONS

* **2TUP:** Two Track Unified Process.
* **AICS:** African Institute of Computer Sciences.
* **API:** Application Programming Interface.
* **DBMS:** Database Management System.
* **HTTP:** Hypertext Transfer Protocol.
* **MVC:** Model View Controller.
* **OS:** Operating System.
* **UML:** Unified Modelling Language.
* **USB:** Universal Serial Bus.

# 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 **“DESIGN AND IMPLEMENTATION OF AN ONLINE CONSTRUCTION MATERIALS ORDERING AND DELIVERY APPLICATION WITH 3D VISUALIZATION”**. 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:

1. **Existing System:** Here, we shall present the already present system in place, that is the one used for consultation and follow-up purposes.
2. **Specification Book:** In this book, we specify the needs of the user taking into considerations the time and cost of the project.
3. **Analysis Document:** Here, we shall present the analysis method chosen together with the presentation of all the diagrams used for the analysis of this project.
4. **Conception phase:** This presents the generic and detailed conception of the project to bring out real world constituents.
5. **The Realization phase**: This phase will permit us to visualize the implementation process of the solution.
6. **Test of functionalities:** In this phase, we shall present to you the different functionalities or modules of our application and how they work.
7. **The User Guide:** This elaborates on all conditions necessary to use the application and how to use it.

**TECHNICAL PHASE**

# CHAPTER 1: EXISTING

### Preamble

This section of our report will cover details of the different research did for the realization of this project.

The existing study, criticism of the existing system, problem statement, proposed solutions.

### Content overview



**INTRODUCTION**

1. **THEME/PROJECT PRESENTATION**
2. **EXISTING SYSTEM STUDY**
3. **CRITISISM OF THE EXISTING SYSTEM**
4. **PROBLEMATIC**
5. **PROPOSED SOLUTIONS**

**CONCLUSION**

**INTRODUCTION**

In this section of our report, we will dive into the various research conducted for the realization of this project. We will explore the existing system, critique the current system, identify the problem statement, and propose a solution. Through thorough analysis and investigation, we aim to provide a comprehensive understanding of the research undertaken to address the challenges and develop an effective solution. By examining the existing landscape and presenting our proposed approach, we strive to contribute towards the advancement and improvement of the project's objectives.

#### THEME/PROJECT PRESENTATION

Our theme is **“DESIGN AND IMPLEMENTATION OF AN ONLINE CONSTRUCTION MATERIALS ORDERING AND DELIVERY APPLICATION WITH 3D VISUALIZATION”.**

The traditional way of going to hardware stores to buy construction materials has many limitations such as waste of time, physical efforts, stock availability, risk of aggressions and many others. Today, we present an innovative solution that solves the problem of acquiring these materials. Our web application called MATERIO which is there to address all of these problems by presenting many features such as a store which permit users know the availability of the materials and choose the best material suitable for their various projects among a wide variety presented to them, the 3D visualization and presentation of the materials, an integrated AI which will help the users know which materials are best for a proposed project, a chat box for users to express themselves whenever they encounter problems with the ordering and delivery process. Our web application revolutionizes the domain of construction in Cameroon because with some few clicks users can access a vast inventory, purchase easily their desired products, and manage their projects efficiently. Experience the future of construction supply chain management today.

#### EXISTING SYSTEM STUDY

##### Hardware stores

Hardware stores today generally operate through physical locations where customers browse shelves and aisles for products like tools, building materials, and home improvement supplies. These stores often rely on manual inventory systems, leading to inaccurate stock records. When there is limited stock and selection of materials, inefficient customer support, customers either leave dissatisfied or spend extra time waiting for restocks, which causes time wastage.

##### Existing system

Table 1: Existing system name, description, and country

|  |  |  |
| --- | --- | --- |
| NAME | DESCRIPTION | COUNTRY |
| Materials Xpress | It proposes a system of delivery the same day in all the metropolitan zones. | USA |
| Baumit | Proposes a service which permits users test the products before ordering them. | Austria |
| iBuildNew | iBuildNew is an online application which permits users to hire services of pose and maintenance and the ordering of materials. | Brazil |

#### CRITISISM OF THE EXISTING SYSTEM

Criticism of the existing system

Table 2: Criticism of the existing system

|  |  |  |
| --- | --- | --- |
| LIMITATION | CONSEQUENCE | PROPOSED SOLUTION |
| Limited stock and Selection | Projects may face delays if essential materials are not available. | * Implement real-time updates to ensure real- time stock updates to ensure customers know exactly what’s available. * Implement a wide selection online, including items not typically stocked in physical stores. |
| Time and Convenience | Time spent travelling and shopping can delay project completion and limited store hours can disrupt your schedule and planning. | - Ensure the website is easy to navigate with powerful search allowing customers to order online. |
| Inadequate Customer Support | Customers may become frustrated if they can’t get timely help with their issues. | - Offer 24/7 customer support through multiple channels such as a chat box. |

#### PROBLEMATIC

We asked ourselves the question **"HOW CAN WE OPTIMIZE THE ORDERING AND DELIVERY PROCESS FOR CONSTRUCTION MATERIALS TO ENSURE PROMPT AND ACCURATE DELIVERIES, WHILE MINIMIZING THE NEED FOR CUSTOMERS TO VISIT HARDWARE STORES IN PERSON?"** We will try to offer a

solution through our application.

#### PROPOSED SOLUTION

After our study and criticism of the existing situation, we propose to design a Web application designed to optimize the construction materials ordering and delivery process. This platform will allow customers to browse available products, and place orders eliminating the need to visit hardware stores. The application will feature real-time inventory updates, and secure payment options, this solution will ensure timely deliveries, reduce errors, and offer a more convenient and efficient way to procure construction materials.

**CONCLUSION**

Having reached the end of this part, we studied the existing system by questioning different health professionals in different institutes through the survey that enabled us to come out with the limitations of the system that leads to the consequences, and we proposed solutions to the various limitations. Finally, we proposed our solution which is a web application. This step is necessary because we need to understand how the system put in place currently works before we can facilitate or ameliorate the processes been carried out in this system.

# CHAPTER 2: SPECIFICATION BOOK

Preamble

After specification book, we have the Analysis phase which permits us to represent a detailed analysis of the limitations identified in our context, and our solution, through a software development process and modelling language.

Content overview



**INTRODUCTION**

1. **CONTEXT AND JUSTIFICATION**
2. **OBJECTIVES OF THE PROJECT III.BENEFICIARIES AND TARGETS**
3. **EXPRESSION OF NEEDS**
4. **PLANNING OF THE PROJECT**
5. **ACTORS OF THE PROJECT**
6. **ESTIMATION OF THE PROJECT**
7. **CONSTRAINTS**
8. **DELIVERABLES**

**CONCLUSION**

**INTRODUCTION**

The specification book gives us the different directives on the product to be delivered with the agreement of the client(s) and the solution provider. In this section, we are going to present the context in which we are to but in place a platform, what the system should do and how the system should do it. These specifications are to avoid the production of inadequate results. We will also include project detail such as the team involve, constraints, the budget, deadlines, constrains and the deliverable.

#### CONTEXT AND JUSTIFICATION OF STUDIES

* 1. Context:

The online construction materials market in Cameroon is poised for significant growth, projected to increase from 1.8 billion CFA francs in 2021 to 2.8 billion CFA francs by 2026, a CAGR (Compound Annual Growth Rate). of 8.7%. This trend has been accelerated by a 45% rise in online ordering among Cameroonian construction professionals during the COVID-19 pandemic. However, the industry still faces challenges, such as limited supplier access and higher material procurement costs. Implementing a comprehensive online ordering and delivery platform could help construction firms in Cameroon address these issues. On average, online ordering has been shown to reduce material costs by 8-12% by enabling access to a wider range of suppliers and more competitive pricing. The top materials ordered online are cement (61%), steel reinforcement bars (52%), and roofing materials (47%), indicating strong demand for digitalized solutions. Proposing an online platform tailored to the Cameroonian construction sector could therefore yield substantial improvements in efficiency, cost-effectiveness, and overall competitiveness.

* 1. Justification:

The construction industry in Cameroon stands to benefit significantly from the implementation of a robust online ordering and delivery platform for construction materials. The market data clearly demonstrates the growth potential in this space, with the online construction materials market projected to expand at a CAGR of 8.7% over the next five years. This trend reflects the increasing demand among construction professionals in Cameroon for more convenient, efficient, and cost-effective procurement methods. For this reason, MATERIO, will allow users to order from a wide variety of materials thereby limiting waste of time primarily. MATERIO will also propose suitable materials for proposed projects by the contractors.

#### OBJECTIVES

1. General Objective

The general objective in this project is facilitate the ordering and delivery of construction materials by eliminating time waste.

1. Specific Objectives

* The customers will be able to log into the platform, and search products from a wide variety presented to them.
* Make complaints about problems encountered through the ordering or delivery of their various products through a chat-box.
* To know which materials are best suitable for their upcoming projects.
* Visualize the materials through 3D presentation to have a better perspective of the material.
* Add their products to a cart thereby choosing their quantity for instant or future purchase.

1. Beneficiaries and Target
   1. *Beneficiaries*

The beneficiaries of our project will be the customers.

* 1. *Target*

Large hardware retailers and suppliers (FOKOU, COGENI, SICAM).

#### EXPRESSION OF NEEDS

* 1. Functional Needs

In this context, functional requirements describe what the system or application should do. The modules are as follows.

* 1. Non-Functional Needs

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

* Performance:

1. Responsiveness: The application should respond quickly to user interactions and provide a smooth user experience.
2. Efficiency: The application should be optimized to perform tasks efficiently, minimizing resource usage and response times.
3. Scalability: The application should be capable of handling increasing user loads and data volumes without significant performance degradation.
4. Reliability: The application should be reliable and available, minimizing downtime and disruptions.
   * Usability:
5. Intuitive Interface: The application should have a user-friendly and intuitive interface that is easy to navigate and understand.
6. Accessibility: The application should be accessible to users with disabilities, complying with accessibility standards and guidelines.
7. Multilingual Support: The application should support multiple languages to cater to a diverse user base.
8. Consistency: The application should maintain consistent design elements and interactions throughout different screens and modules.
   * Security:
9. User Data Protection: The application should securely store and handle user data, following best practices for encryption and data privacy.
10. Authentication and Authorization: The application should provide secure user authentication mechanisms and access controls to protect user accounts and data.
11. Secure Transactions: The application should ensure secure transmission and handling of sensitive information, such as payment details.
12. Data Backup and Recovery: The application should have robust backup and recovery mechanisms to protect against data loss and enable quick recovery.
    * Compatibility:
      1. Cross-Platform Compatibility: The application should be compatible with different operating systems and devices, including mobile and web platforms.
      2. Browser Compatibility: The web application should be compatible with major web browsers, ensuring consistent functionality and appearance.
      3. API Integration: The application should be able to integrate with external systems or APIs seamlessly.
    * Performance Monitoring and Analysis:
      1. Logging and Monitoring: The application should log events and errors for monitoring and troubleshooting purposes.
      2. Analytics and Insights: The application should provide analytics and insights on user behavior, system performance, and usage patterns.
      3. Performance Optimization: The application should continuously optimize its performance based on analytics and monitoring data.
    * Maintainability:
      1. Modularity: The application should be designed with a modular structure, making it easier to maintain and enhance individual components.
      2. Code Quality: The application's code should follow best practices and coding standards, ensuring readability and maintainability.
      3. Documentation: The application should have comprehensive documentation, including installation instructions, user guides, and API documentation.

#### PLANNING OF THE PROJECT

1. Chronogram of activities

Table 3: Chronogram of activities

|  |  |  |  |
| --- | --- | --- | --- |
| PHASE | OBJECTIVE | OUTPUT | DURATION |
| EXISTING SYSTEM | Study of the existing system | Existing system | 5 days |
| SPECIFICATION BOOK | Specification of the user needs | Specification book | 5 days |
| ANALYSIS | Capture of needs Use case and textual description modelling | Analysis book | 15 days |
| CONCEPTION | Preliminary conception and detailed conception | Conception book | 15 days |
| REALIZATION | Implementation Unitary test Integration test Development, Deployment, Conception diagrams | Realization book | 30 days |
| TEST OF FUNCTIONALITIES | Testing of the software and debugging | Test of  functionalities | 10 days |
| REPORT | Writing of our report | Word Document | 1 week |

1. LIST OF PARTICIPANTS

*List of Participants*

Table 4: Actors of the Project

|  |  |  |
| --- | --- | --- |
| NAME | TITLE ROLE | |
| FONO KENGNE Lionel Emery Nasser | Software Engineering student at  AICS-Cameroon | Analyst and developer |

1. Gantt Project Planning

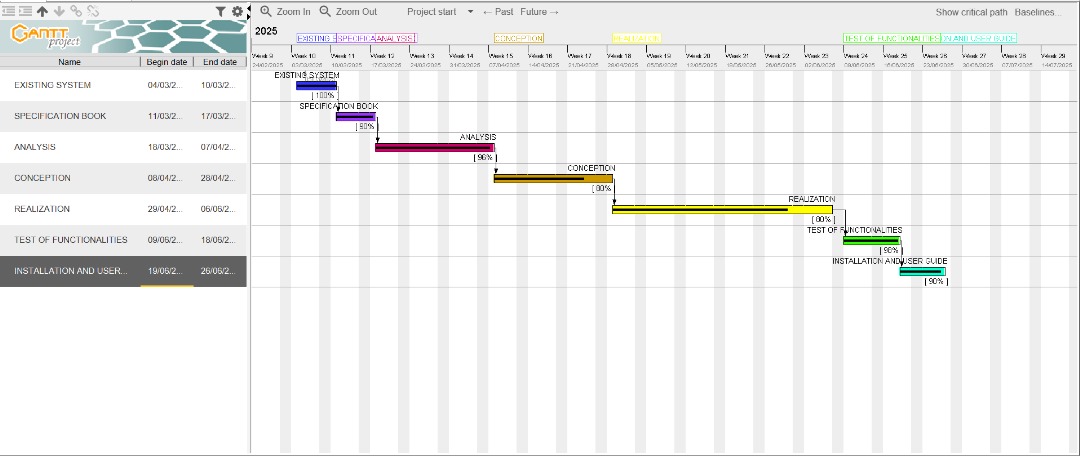


Figure 1: Gantt project planning

#### ESTIMATION OF THE PROJECT

1. Software Resources

Table 5: Software Resources (source: Mercurial 2025)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RESOURCES | DESIGNATION | USAGE | QUANTITY | UNIT COST (FCFA) |
| Formatting | Microsoft  Offie 2019 | Writing of the report | 1 | 177,000 |
| Cloud Storage | MEGA | Saving our report | 1 | Freemium |
| Web Browser | Google Chrome | View web pages | 1 | Freeware |
| Code Editor | Visual Studio Code | For writing the code of the  application | 1 | Freeware |
| Project Planning | Gantt Project | For building a Gantt chart | 1 | Freemium |
| UML Analysis | Sybase PowerAMC | For drawing UML diagrams | 1 | Freemium |
| TOTAL 1 |  |  | 7 | 177,000 |

1. Hardware Resources

Table 6: Hardware Resources (source: Mercurial 2025)

|  |  |  |  |
| --- | --- | --- | --- |
| RESOURCES | HARDWARE | QUANTITY | UNIT COST (FCFA) |
| Computer | Acer 11th Gen Intel(R) Core (TM) i3-1115G4 @ 3.00GHz 3.00 GHz | 1 | 250,000 |
| Printer | Printers | 1 | 425,500 |
| Smart phone | iPhone XR | 2 | 150,000 |
| Removable disk | 8GB USB Key | 1 | 6,612 |
| Modem | 4g LTE-Advanced mobile WI-FI hotspot mq531 150mps | 1 | 23,000 |
| TOTAL 2 |  | 6 | 855,112 |

1. Human Resources

Table 7: Human Resources (source: Mercurial 2025)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ROLE | NUMBER OF DAYS | QUANTITY | COST PER DAY | TOTAL PRICE (FCFA) |
| Project Manager | 90 | 01 | 30 000 | 1,500,000 |
| Analyst | 21 | 01 | 25 000 | 450,000 |
| UI/UX  Designer | 07 | 01 | 20 000 | 140,000 |
| Programmer | 30 | 01 | 15 000 | 320,000 |
| Margin error | / | / | / | 1,475,000 |
| TOTAL 3 |  |  |  | 2,410,000 |

1. Global Estimation

Table 8: Global Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| TOTAL 1(FCFA) | TOTAL2 (FCFA) | TOTAL 3(FCFA) | OVERALL TOTAL (FCFA) |
| 177,000 | 855,112 | 2,410,000 | 3,786,323.2 |
| THREE MILLION, SEVEN HUNDRED AND EIGHTY-SIX THOUSAND, THREE HUNDRED AND TWENTY-THREE POINT TWO (FCFA) | | | |

#### CONSTRAINTS

1. **Technical constraint**

For the development of our system, we have sufficiently robust tools to guarantee a minimum of security, extensibility and excellent scalability. Moreover, the programming phase will have to follow all the technical standards for a better performance in a reduced execution time, this is why the choice of the development technologies is crucial.

1. **Time Constraint**

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

1. **Cost Constraint**

The realization of our project will require expenditures in human resources, material and software a total cost of 3,786,323.2

#### DELIVERABLES

1. In project management, any component materializing the result of a realization service is called a deliverable. In the case of our project, the deliverables are: A report composed of the following document

* The application.
* The user guide.
* The PowerPoint.

**CONCLUSION**

The specifications book contains detailed information about the project's objectives, requirements (both functional and non-functional), team members, project plan, cost, and constraints. This document serves as a guide to develop a solution that meets the project's needs while avoiding errors and compatibility issues. The specifications book acts as a foundation for the analysis phase of the project, where we carefully examine the provided information to determine the best approach and ensure a successful outcome.

# CHAPTER 3: ANALYSIS PHASE

### Preamble

This section of our report will cover details on the product to be delivered with the agreement of the client(s) and the solution provider. In this section, we are going to present the context in which we are to put in place a platform, what the system should do and how the system should do it.

### Content overview



**INTRODUCTION**

1. **METHODOLOGY**
   1. **COMPARATIVE STUDY OF UML AND MERISE**
   2. **COMPARATIVE STUDY OF UNIFIED PROCESS**
2. **MODELING**
   1. **USE CASE DIAGRAM**
   2. **COMMUNICATION DIAGRAM III.SEQUENCE DIAGRAM**

**IV. ACTIVITY DIAGRAM**

**CONCLUSION**

**INTRODUCTION**

The analysis book permits us to examine in an explicit way the existing system, it’s limitations and how we can remedy them. We will also describe in detail the modeling language known as UML (Unified Modeling Language) which is coupled with Two Tract Unified Process (2TUP) to form a method and its justification why we decided to use it in preference of another. then we will dive directly into the modeling of the proposed solution consisting of diagrams that meets the requirements of the functional needs.

#### METHODOLOGY

* + - 1. COMPARATIVE STUDY OF UML AND MERISE
         1. MERISE

MERISE stands for “Méthode d’Etude et de Réalisation Informatique pour des Systèmes d’Entreprise”. Although it is prescriptive to some extent, MERISE permits the participant of end user’s and senior management as well as data processing professionals in its decision cycle. MERISE is a method for designing, developing and carrying out IT projects. The goal of this method is to achieve the design of an information system. The MERISE method is based on the separation of data and processing to be carried out in several conceptual and physical models. Indeed, the arrangement of data does not have to be often overhauled, while treatments are more frequently.

* + - * 1. UML

UML (Unified Modelling Language) is a standard notation for the modelling of real world objects as a first step in developing an object-oriented design methodology. Its notation is derived from and unifies the notations of three object-oriented design and analysis methodologies: Grady Booch's methodology for describing a set of objects and their relationships, James Rumbaugh's Object-Modelling Technique (OMT), Ivar Jacobson's approach which includes a use case methodology. Other ideas also contributed to UML, which was the result of a work effort by Booch, Rumbaugh, Jacobson, and others to combine their ideas, working under the sponsorship of Rational Software. UML captures information about the static and dynamic view of a system. UML 2.5 comprises of 14 diagrams which represent the different views of a system. The 14 diagrams can be subdivided into two, Static or structural and Dynamic diagrams. These diagrams include.

*STATIC OR STRUCTURAL DIAGRAMS*

 Class Diagram.  Object Diagram.

 Component Diagram.  Deployment Diagram.

 Composite Structure Diagram.  Package Diagram.

 Profile Diagram.

*BEHAVIOURAL DIAGRAMS*

 Use Case Diagram.  Activity Diagram.

 State Machine Diagram.  Sequence Diagram.

 Communication Diagram.  Global Interaction Diagram.  Timing Diagram.

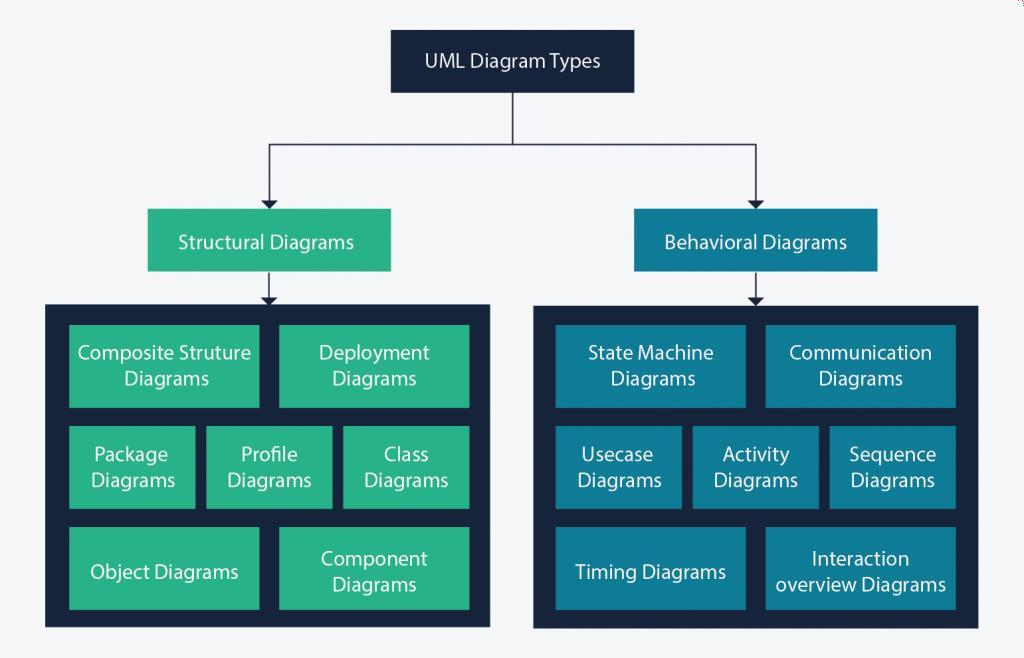


Figure 2: UML 2.5 diagrams overview (source: https://creately.com/blog/diagrams/uml-diagram-types-examples/)

A very important notice is that UML is not a method but a modelling language. As such to give it an approach we need to associate UML to a Unified Process (UP) in other to give our conception a methodology to follow. There exist several Unified Processes, but our modelling approach will be the 2TUP (Two-track unified process) which we will use during our project.

Table 9: Differences 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 |
| MERISE 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; RUP 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. |
| Rational | Object |

* + - 1. COMPARATIVE STUDY OF UNIFIED PROCESS

1. *Unified Process*

The first step of the left (functional) branch of Two Track Unified Process (2TUP) is the capture of the functional needs. At this step, we capture the intended behavior of the system that maybe express as services, tasks or functions the system is required to perform.

1. *The Two Track Unified Process (2 TUP)*

2TUP is a unified process which is built on UML and has as objective to bring solution to constraints of functional and technical changes imposed on information systems by strengthening controls on development capacities. It proposes a Y-sharped development life cycle that separates the functional aspect from the technical aspects, and the merging of these two forms the implementation aspect. 2TUP distinguishes therefore two branches: the functional and technical branches, the combination of the result of these two branches forms the third: the realization branch – where we realize our system. The diagram below illustrates the branches of 2TUP.



Figure 3: 2TUP diagram (source: https://www.mysciencework.com/omniscience/pervasive-mobile-healthcare-system-based on- cloud-computing

1. The left branch (functional branch)

It captures the functional needs of a system. This ensures the production of software that meets the needs/requirements of the user. The analysis here consists of studying precisely the functional specification to obtain an idea of what the system is going to realize, and its result does not depend on any technology.

1. The right branch (Technical branch)

The technical branch enumerates the technical needs and proposes a generic design validated by a prototype. The technical needs include 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.

1. The middle branch (Realization or implementation branch)

In this branch, we study the preliminary conception, detailed conception, and documentation of the system. The realization branch supports the following: Preliminary conception: This is the most sensitive step of 2TUP as it is the confluence of the functional and technical branch. It is completed when the deployment model, theoperating model, the logical model, interphases and the software configuration model are defined. We have the following diagrams:

 Component Diagram.  Deployment Diagram.  Package Diagram.

 Composite Structure Diagram.

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

 Class Diagram.  Object Diagram.

 Sequence Diagram.

 Timing Diagram.

Coding and testing: This is the phase where we program the designed features and testthe coded features.

The recipe: Also known as the deliverables is the validation phase of the functions ofthe developed system.

#### MODELLING

a. Capture of Functional Needs

The first step of the left (functional) branch of Two Track Unified Process (2TUP) is the capture of the functional needs. At this step, we capture the intended behavior of the system that maybe express as services, tasks or functions the system is required to perform.

1. USECASE DIAGRAM
   1. *Definition*

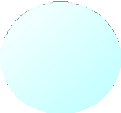
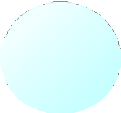
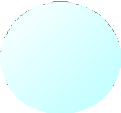
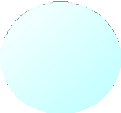
Use case diagram shows the functionalities of a system, their interdependencies and how they relate with actors of the system. A use case is a specification of behavior. The main objectives of the use case diagram are:

* Provide a high-level view of the system.
* Identify the functions of the system.
* Use case diagrams are completed with a textual description of each use case that is intended to define the use case in greater details.
* Use case diagrams are completed with a textual description of each use case that is intended to define the use case in greater details.
  1. *Formalism*



Acteur\_3

Figure 4: Use case diagram formalism



System

Acteur\_1

Usecase 1

Usecase 2

<<include>>

Cas\_2

<<extend>>

Usecase 3

Acteur\_2

Table 10: Use Case Diagram Components

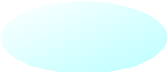
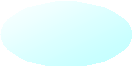
|  |  |  |
| --- | --- | --- |
| Elements | Notation | Description |
| Actors |  | Represents an entity that directly interacts with the system. The actor is what performs the different possible actions of the system. |
| Use case |  | A use case represents a functionality of the system. It is an action that can be performed by an actor. |
| Association |  | It indicates that an actor takes part in a use Case. |
| Include |  | An inclusion denotes that an included action must be performed before the including action can be performed. |
| Extend |  | An extension denotes that an extending action may be performed while an extended action is being performed. |
| Generalization |  | This shows that an actor or a use case is a kind of another abstract or concrete actors can be defined and later specialized using generalization relationship. |
| System |  | It is a container of use cases which interact with external actors |

* 1. *The Actors of Our System*

Table 11: Actors of Our System

|  |  |
| --- | --- |
| Actor | Role |
| Visitor | View the website and creating account |
| Customer | He performs actions like buying, send complaints, make a 3D view and others. |
| Delivery Agent | He receives orders made by the customers |
| System Administrator | The administrator is responsible of managing the accounts of customers and delivery agents (validating, deleting accounts) and general functioning of the system. |
| Payment API | Provides payment services to the application in terms of NEXUS PAY made in Welldone Planet. |

* 1. *General Use Case Diagram*



MATERIO

Create Account

Visitor

<<include>>

Make 3D View

Customer

Make Complaint

<<include>>

Authenticate

Describe Project

<<include>>

Access Store

<<include>>

Add to Cart

<<extend>>

<<extend>>

<<extend>>

Search Product

Buy Product

Delete Product

View Order

<<include>>

Delivery Agent

<<extend>>

In Transit

<<include>>

Update Order Status

Delivered

<<extend>>

Manage Store

<<include>>

Administrator

<<extend>>

<<extend>>

Delete product

Add Product

<<include>>

Manage Delivery AgentAccount

<<extend>>

<<extend>>

Delete

<<include>>

Create

<<extend>>

Reply

Consult Complaint List

Payment API

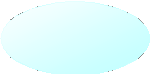
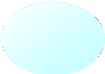
Figure 5: General Use Case Diagram

* 1. *Access Store Specific Use Case Diagram*



Customer

Figure 6: Access Store Specific Use Case Diagram



<<include>>

Authenticate

Access Store

<<extend>>

Add to Cart

<<extend>>

<<extend>>

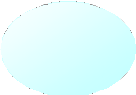
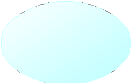
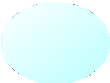
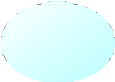
Buy Product

Delete Product

Payment API

Access Store

* 1. *Manage Store Specific Use Case Diagram*



<<include>>

Authenticate

Manage Store

<<extend>>

Administrator

<<extend>>

Mark out of stock

Add Product

Manage Store

Figure 7: Manage store use case diagram

* 1. *Textual Description of Use Cases*

Table 12: Buy Product Textual Description

|  |  |
| --- | --- |
| TITLE | BUY PRODUCT |
| Summary | The Customer needs to acquire a product. |
| Actor (s) | Customer. |
| Precondition (s) | The Customer must be logged into the system. He must have the necessary permission to buy. The Customer clicks on the add to cart button. The Customer clicks on the cart icon. |
| Trigger (s) | The Customer clicks on the buy button. |
| Nominal Scenario | 1. The system displays the payment form. 2. The Customer fills and clicks on buy button. 3. The system confirms that payment has been done by displaying successful message. |
| Alternative Scenario | At step 2 of the Nominal Scenario, the Customer enters invalid values.  2.1 The system displays the error message. |
| Exceptional Scenario | At step 3 of the Nominal Scenario, the API encounters a payment failure.  3.1 The system displays an error message. |
| Post Condition of Success | The system displays a payment receipt. |
| Post Condition of Failure | The system brings the Customer back to the payment form. |
| Non-Functional Requirements | The interface should be user-friendly. |

Table 13: Manage Store Specific Textual Description

|  |  |
| --- | --- |
| TITLE | MANAGE STORE (ADD PRODUCT) |
| Summary | The Administrator needs to oversee the store. |
| Actor (s) | Administrator. |
| Precondition (s) | The Administrator must be logged in.  He must have the necessary permission to manage the store. |
| Trigger (s) | He must click on “Add” button. |
| Nominal Scenario | 1. The system displays the Administrator dashboard. 2. The Administrator clicks on “Add” button. 3. The system displays the Add product form. 4. The Administrator defines the product parameters. 5. The Administrator clicks on the “SAVE” button. 6. The system confirms that product has been added and updates the store inventory. 7. The system displays a list of added products. |
| Alternative Scenario | At step 7 of the Nominal Scenario, the Administrator selects a specific product.   * 1. The Administrator clicks on the “DELETE” button.   2. The system updates the store inventory. |
| Exceptional Scenario | At step 6 of the Nominal Scenario, the system encounters a database error.  6.1 The system displays an error message and sends back the Administrator to step 4 of the Nominal Scenario. |
| Post Condition of Success | The Administrator receives an error message indicating that the product has not been added. |
| Non-Functional Requirements | The interface should be user-friendly. |

1. COMMUNICATION DIAGRAM
   1. *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.

* 1. *Formalism*

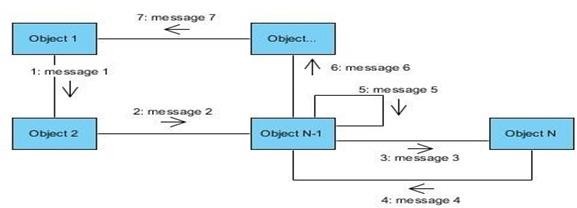


Figure 8: Communication diagram formalism

* 1. *Components of a Communication Diagram*

Table 14: Components of a Communication Diagram

|  |  |  |
| --- | --- | --- |
| ELEMENT | NOTATION | DESCRIPTION |
| Message |  | Designs a particular communication between lifelines. |
| Connectors |  | It represents the relationships that exist between lifelines |
| 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. |

* 1. *Authentication Communication Diagram*



4: Check conformity

8: Send signin response

6: Send login request

System

DBMS

10: Display successful message 9: Display error message 5: Display error message 3: Fill and submit form

2: Display signin interface

1: Request signin form

Customer

7: Execute signin request

Figure 9: Authentication Communication Diagram

* 1. *Buy product Communication Diagram*



15: Treat and send response

DBMS

23: Display payment form 22: Display error message 21: Display payment form 20: Display error message 19: Display payment form 18: Display error message

17: Display successful message 10: Fill and submit form 9: Display payment form

8: Click on buy button

Customer

16: Receive response

14: Send result request

System

7: Display cart interface5: Adds product to cart 6: Click on cart icon

4: Select product and click on add to cart

3: Display shop interface 2: Click on shop now button

1: Display home interface

13: Receive and treat payment result 12: Send payment request

API

11: Treat and send payment request

Figure 10: Buy product Communication Diagram

* 1. *Add product Communication Diagram*



8: Recieve Add result

6: Send add request

System

13: Display Add product form 10: Display error message

9: Display successful message 5: Fill and submit form

4: Display Add product form 3: Click on Add button

2: Display Admin dashboard interface

1: Click on Add button

7: Treat and send add request

Administrator

12: Display error message

DBMS

Figure 11: Add product Communication Diagram

1. SEQUENCE DIAGRAM
   1. *Definition*

A Sequence diagram describes interactions among classes in terms of an exchange of messages over time. They are also called event diagrams. A Sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modelling a new system.

* 1. *Formalism*



sequence diagram formalism

Objet\_1

Objet\_2

Objet\_3

Message\_1

Message\_2

alt ok

Message\_3

not ok

Message\_4

Mess ge\_5

Figure 12: Formalism of a Sequence Diagram

* 1. *Components of a Sequence Diagram*

*Table 18: Components of sequence diagram*

|  |  |  |
| --- | --- | --- |
| ELEMENT | NOTATION | DESCRIPTION |
| **Lifelines** |  | They represent rows or objects instances that participate in the sequence being modelled. |
| **Messages** | A close-up of a message  Description automatically generated | These are arrows which shows the direction of message flow. We have the synchronous, the asynchronous and the self- messages. |
| **Combined Interaction Fragment** |  | An articulation of interaction diagram, defined by an Operator and Operands. |
| **Activation** |  | It describes the period in which an operation is performed by an element |

* 1. *Authentication Sequence Diagram*

Ex



Display successful message

{credentials==valid}

Display error message

{credentials==invalid}

alt

Send signin response

Execute signin request

Send login request

{conformity==true}

{conformity==false}

Display error message

alt

Check conformity

Fill and submit form

Request signin form

Display signin interface

Customer

DBMS

System

Authentication

Figure 13: Authentication Sequence Diagram

* 1. *Buy product Sequence Diagram*



Diispllay payment form

{if information is not okay}

Diispllay error message

{If payment results are not okay}

Diispllay error message

Diispllay payment form

{If response is not okay}

Diispllay error message Diispllay payment form

alt {If response is okay}

Diispllay successfull message

Receiive response

Treat and send response

Send resullt request

alt {if payment results are okay}

Receiive and treat payment resullt

Treat and send payment request

Send payment request

alt {If information is okay}

Diispllay home iinterface

Clliick on shop now button Diispllay shop iinterface

Sellect product and clliick on add to cart

Adds product to cart

Clliick on cart iicon Diispllay cart iinterface

Clliick on buy button Diispllay payment form

Fiillll and submiit form

Authentication()

ref

Customer

DBMS

API

System

Buy product

Figure 14: Buy product Sequence Diagram

Tre

* 1. *Add product Sequence Diagram*

ea



DisplMayesesrarogrem\_1e2ssage

Display Add product form

{If information is not okay}

Display error message

{If results not okay}

Display successful message

alt {If results okay}

Treat and send add request

Send add request

Recieve Add result

alt {If information okay}

Click on Add button Display Add product form

Fill and submit form

Click on Add button

Display Admin dashboard interface

Authentication()

ref

Administrator

DBMS

System

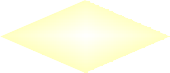
Add product

Figure 15: Add product Sequence Diagram

1. ACTIVITY DIAGRAM
   1. *Definition*

An activity diagram is a graphical representation of workflows that show the steps needed in the realization of a process; showing the details from a start point to an end point through all decisions and actions that can possible be performed. Activity diagrams are intended to model both the computational and organizational process. They flow can be sequential, branched or concurrent. Below is an activity diagram formalism.

* 1. *Formalism*



Acti vi te\_1

Acti vi te\_2

Acti vi te\_3

Acti vi te\_4

[Not ok]

Acti vi te\_6

Acti vi te\_5

Deci sion\_1

Acti vi te\_7

Acti vi te\_8

[Not ok]

Deci sion\_2

[Ok]

UniteOrganisation\_3

UniteOrganisation\_2

UniteOrganisation\_1

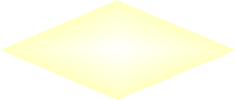
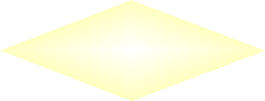
Figure 16: Formalism of an Activity Diagram

* 1. *Components of an Activity Diagram*

Table 15: Components of an Activity Diagram

|  |  |  |
| --- | --- | --- |
| ELEMENTS | DIAGRAMMATIC REPRESENTATION | DESCRIPTION |
| Activity |  | Used to represent a set of actions. |
| Action |  | Represent a task to be Performed. |
| Activity edge |  | A directed connection between two activity nodes through which tokens may flow. |
| Initial node |  | Shows the beginning of an activity or set of actions. |
| Final node |  | Stops all controls and object flows in an activity. |
| Object node |  | Represents an object connected to a series of object flows. |
| Decision node |  | Represents a test condition that slits an incoming activity edge into opposite outgoing activity edges. |
| Merge node |  | Reunite different decision paths created using a decision node. |
| Fork node |  | Slits behavior into parallel or concurrent flows of activities (or actions). |
| Join node |  | Unites a set of parallel or concurrent flows of activities or actions. |
| Swimlane and Partition |  | A way of grouping activities performed by the same actor in an activity diagram or to group actions in the same thread. |

* 1. *Authentication Activity Diagram*



Display signin form

Execute authentication query

Display error message

[Invalid]

Check form validity

Fill and submit form

[Valid]

Send query results

Displays error message

Process query result

[Not ok]

Check validation

[Ok]

Display home page

Send authentication query

Request signin form

DBMS

System

Customer

Figure 17: Authentication Activity Diagram

* 1. *Buy Product Activity Diagram*

Customer

System

API

DBMS

Displays home interface

Authenti cate

Cli ck on Shop Now button

Displays store interface

Select product and cl i on add to cart

ck

Ad

ds product to cart

Cli ck on cart i con

Displays cart interface

Cli ck on buy button

Display paym

ent form

Fill and submit form

[Not ok]

Form check Dispalays error message

[Ok]

Send payment request

Treat payment request

Send payment result

Recei ve and treat result

Treat result request

[Ok]

Payment result check

[Not ok]

Send request

Send response

Dispaly error message

Recei ve response

[Ok]

Response check

[Not ok]

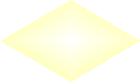
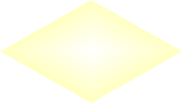
Display error message



Display success message

Figure 18: Buy Product Activity Diagram

* 1. *Add Product Activity Diagram*



Display Add product form

Parameters check

[Not ok]

[Ok]

Send add request

Treat add request

Receive add result

[Not ok]

Result check

[Ok}

Display successful message

Send add result

Fill and submit form

Click on Add button

DBMS

System

Administrator

Displays error message

Display error message

Display the administrator dashboard

Authenticate

Figure 19: Add Product Activity Diagram

**CONCLUSION**

In the analysis phase, we chose a software development process and modelling language, after which we explained the functional need of our system, we saw the use case diagram which shows the relationship between the actors and use case (the action the actor can perform on the system),we saw the communication diagram which represents the architecture of the system based on object oriented programming, we saw the sequence diagram which represents the flow of messages between elements in the system, and lastly the activity diagram which shows the workflow of our system. We will now move to the conception phase in which we will present the technical branch of our system together with related diagrams.

# CHAPTER 4: CONCEPTION PHASE

### Preamble

The conception phase will permit us to present in an orderly manner the components necessary for the good functioning of our software and the architecture used for the proposed solution. It bridges the gap between the analysis phase and the realization phase.

### Content overview



**INTRODUCTION**

1. **TECHNICAL PHASE**
   1. **GENERIC DESIGN**
2. **CAPTURES OF TECHNICAL NEEDS**
3. **RELATED UML DIAGRAMS**
   1. **CLASS DIAGRAM**
   2. **STATE TRANSITION DIAGRAM**
   3. **PACKAGE DIAGRAM. CONCLUSION**

**INTRODUCTION**

The conceptual phase will describe in detail the necessary specifications, features and operations that will satisfy the functioning requirements of the proposed system as modelled in the analysis phase. This phase is meant to identify and consider essential components (hardware

/Or software), structure (network capabilities), processes and procedures for the system to accomplish it objectives. We will look at some diagrams such as the class diagram, state machine diagram and package diagram.

* + 1. TECHNICAL BRANCH

1. GENERIC DESIGN

*I. Hardware diagram of the system.*

The hardware diagram simply shows how the system components of our system are deployed; it shows the positioning of each device into it right proportion.

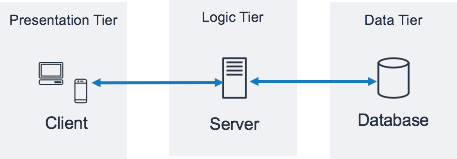
*ii. High Level Architectural Diagram of the Software*

The high-level architecture diagram provides an overview of the entire system, identifying the main components that would be developed for the product and their interfaces. The high-level architecture diagram below illustrates this.

* + 1. CAPTURE TECHNICAL NEEDS.
       1. *Physical architecture*

When designing an application, it is important before going any further to know the architecture of this application. The choice of your architecture can be based on the number of users, the data volume, transaction volume, performance requirements and security needs.

The architecture of our project is an n tier architecture. We made this choice because we are making use of external APIs. In our case a payment API.



Payment API

Figure 20: Physical Architecture

* + - 1. *Logical architecture:*

Model View controller or MVC as it is popularly called, is a software design patten for developing applications. A model view controller pattern is made up of the following three parts.

Model: The lowest level of the patten which is responsible for maintaining data. View: This is responsible for displaying all or a portion of data to the user.

Controller: It handles software codes that control the interactions between the model and the view.

MVC is popular as it isolates the application logic from the user interface and supports separation of concerns. Here the controller receives all requests for the application then works

with the model to prepare data needed by the view. The view then uses the data prepared by the controller to produce a final response. The MVC can be represented as follows.

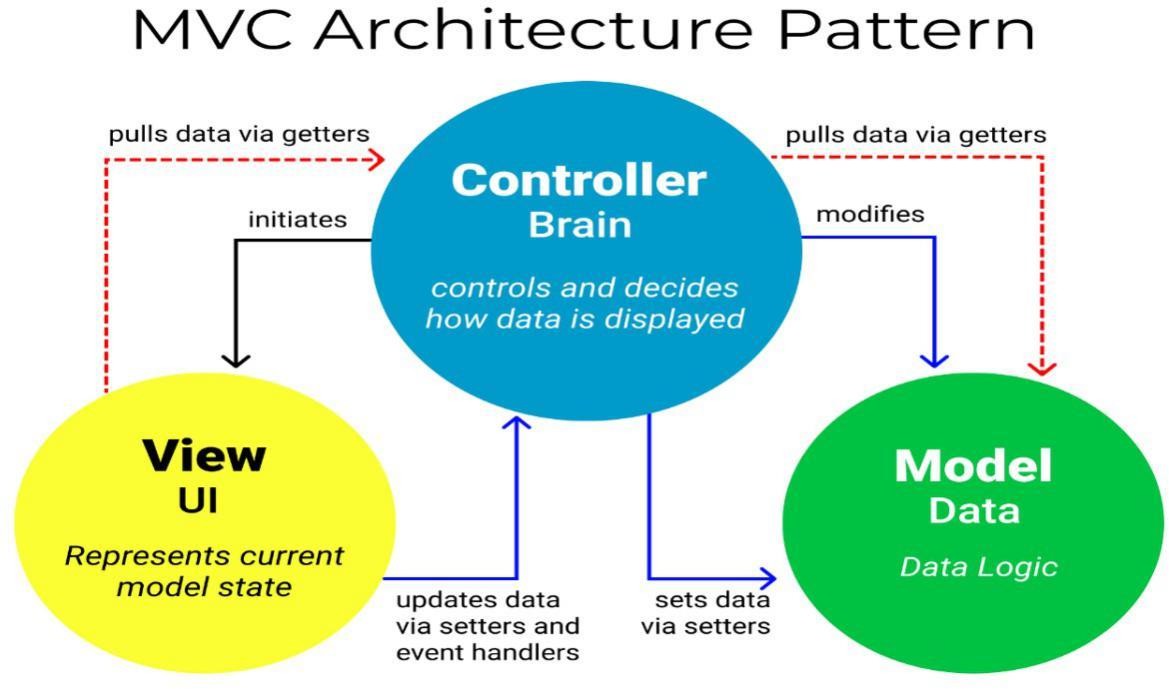


Figure 21: The MVC architecture Source: <https://www.freecodecamp.org/news/the-model-view-controller-pattern-> mvcarchitecture-and-frameworks-explained

#### B. RELATED UML DIAGRAMS

1. CLASS DIAGRAM
   1. *Definition*

A class diagram is a static diagram. It represents the static view of an application. class diagram is not only used for visualizing, describing and documenting different aspect of the system but also for constructing executable code of the software application. Class diagram describes the attribute and operation of a class, and constraints imposed on the system. Its purpose is to model the static view of an application.

* 1. *Formalism*

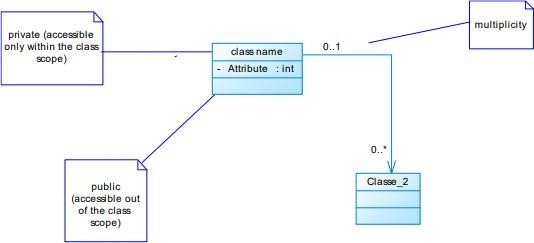


Figure 22: Formalism of a Class Diagram

* 1. *Components of a Class Diagram*

Table 16: Components of a Class Diagram

|  |  |  |
| --- | --- | --- |
| ELEMENT | REPRESENTATION | DESCRIPTION |
| Class |  | A class is an element that defines the attributes and behaviors that an object can generate |
| Composition |  | If a parent of a composite is deleted, usually, all its parts are deleted with it. |
| Aggregation |  | If the parent of the aggregate is deleted, usually the children are not deleted. |
| Dependency |  | It existed between two classes, if one changes it may cause the change in the order, but the other way around. |
| Generalization |  | it a relationship between a whole thing (called superclass) and a more specific thing (called subclass) |
| Association |  | It is a general type of relationship between elements, it may include cardinality, roles etc. |

* 1. *Materio Class Diagram*



consists of

- id

contains

1..1

assigned to

paid with

1..1

+ removeIteme () : int

+ updateQuantity () : int

+ calculateTotal () : int

processes payment through

+ assignDeliveryAgent () : int

: int

: int

1..\*

* totalAmount : float
* status : String
* orderDate : Date

+ calculateTotal ()

Cart

- total : float

+ addItem ()

1. .1
2. .1

Order

: int

Product

* id : int
* name : int

1. .\* - description : int
   * price : int
   * List<string> images : int

CartItem

* quantity : int
* subtotal : float

+ calculateSubtotal () : int

1..1

represents

references

1..1

0..1

+ processPayment () : int

+ verifyPayment () : int

PaymentMethod

* id : int
* type : String
* details : String
* isDefault : boolean

orderItem

* quantity : int
* subTotal : float

+ calculateSubtotal () : int

1..1

DeliveryAgent

- List<Order> assignedOrders : int

+ updateOrderStatus () : int

+ completeDelivery () : int

1..\*

0..\*

PaymentAPI

0..\*

1..1

1..1

: int

: int

+ addToCart ()

+ updateProfile () : int

0..1

- list<Order>OrderHistory : int

+ placeOrder () : int

1..1

: String

: String

+ register ()

+ login ()

+ removeFromCart () : int

+ cancelOrder () : int

+ payOrder () : int

- id

0..1

Customer

# username : String # email : String # Password : String

1..1

has

User

: int

# id

places

0..\*

0..\*

0..\*

+ manageUsers () : int

+ manageProducts () : int

+ manageOrders () : int

+ viewProductDetails () : int

CustomerOrder

* customer\_id : int
* order\_id : int

: int

: int

+ browseProducts ()

+ searchProducts ()

1..1

Administrator

Visitor

Address

* Street : String
* City : String
* Quarter : String

0..\*

uses

0..\*

owns

1..1

Figure 23: Materio Class Diagram

1. STATE MACHINE DIAGRAM
   1. *Definition*

*A* state machine diagram describes the behavior of a single object in response to a series of events in a system. Also known as the state machine diagram, it models the dynamic flow of control from the state of a particular object within a system.

* 1. *Formalism*

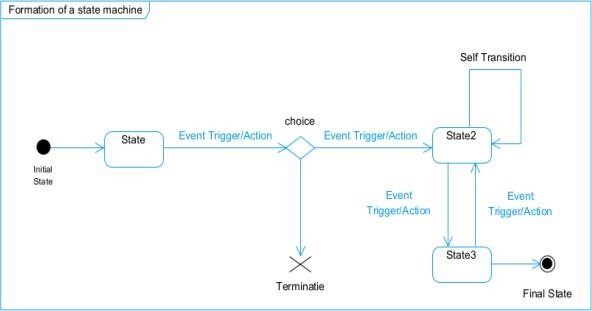
**

Figure 24: Formalism of a State Machine Diagram

* 1. *Components of a State Machine Diagram*

Table 17: Components of a State Machine Diagram

|  |  |  |
| --- | --- | --- |
| ELEMENTS | REPRESENTATION | DESCRIPTION |
| State |  | Models a situation during which a certain invariant condition holds. |
| First (Initial State) |  | It represents a default vertex, that is, a source for a single transaction to the default or composite state. |
| Final State |  | A state specifying that the enclosing region is complete. |
| Transition |  | A direction relation between a source and a target vertex. |
| Choice Pseudo State |  | A diamond symbol that indicates a dynamic condition with branched potential results |
| Terminate |  | Implies that the  execution of a state by means of its context is terminated |
| Diagram Overview |  | A placeholder for the linked states in a state machine diagram. |

* 1. *Order State Machine Diagram*



Customer places order

Created

Order verified

Processing

Order completed

Delivered

Order satisfaction period ends

Completed

Figure 25: Order state machine diagram

1. PACKAGE DIAGRAM
   1. *Definition*

This is a structural diagram used to show the organization and arrangement of various model elements in the form of packages. A package diagram is the grouping of related UML elements such as classes, diagrams or even other packages.

* 1. *Formalism*

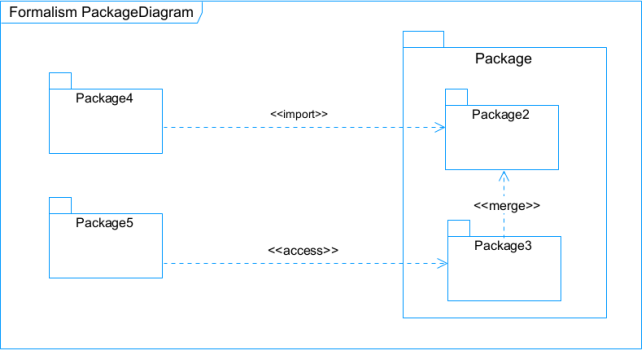
**

Figure 26: Formalism of a Package Diagram

* 1. *Components of a Package Diagram*

Table 18: Components of a Package Diagram

|  |  |  |
| --- | --- | --- |
| ELEMENTS | REPRESENTATION | DESCRIPTION |
| Package |  | A package is a namespace use to group related elements; it is a mechanism used to group elements into a better structure in a system. |
| Package Import |  | A relationship Indicate that, functionality has been imported from one package to another. |
| Package Access |  | A relationship Indicates that one package requires assistance from the function of another package. |
| Package Merge |  | It is a relationship which shows that, the functionality of two packages are combines to a single function. |

* 1. *Materio Package Diagram*

Package 15



User

Functionalities

User

<<merge>>

<<access>>

Buy product

<<merge>>

DeliveryAgent

<<merge>>

<<access>>

Visitor

Access store

4

<<access>>

<<merge>>

Add to cart

<<access>>

Make 3D view

Administrator

Customer

<<access>>

Search product

<<access>>

Manage store

Figure 27: Materio Package Diagram

Package\_1

**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. 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

In this phase we will to straight forward in the implementation of our solution, we will base ourselves on the analysis and conception phases and present the deployment and component diagrams.

### Content overview



**INTRODUCTION**

1. **DEPLOYMENT DIAGRAM**
2. **COMPONENT DIAGRAM**
3. **TECHNOLOGICAL CHOICES**

**A. SOFTWARE CHOICES CONCLUSION**

**INTRODUCTION**

Here in the realization phase, we will see some diagrams related to the physical aspect of the system like libraries, documents, as well as the physical topology of the components of the system when the software is been deployed.

* + 1. DEPLOYMENT DIAGRAM

1. *Definition*

Deployment diagram is a structural diagram used to visualize the topology of the physical components of a system, where the software is deployed. They consist of nodes and their relationship. It is related to the component diagram because the components are deployed using the deployment diagram. A deployment diagram consists of nodes. Nodes are nothing but physical hardware used to deploy the application.

1. *Formalism*

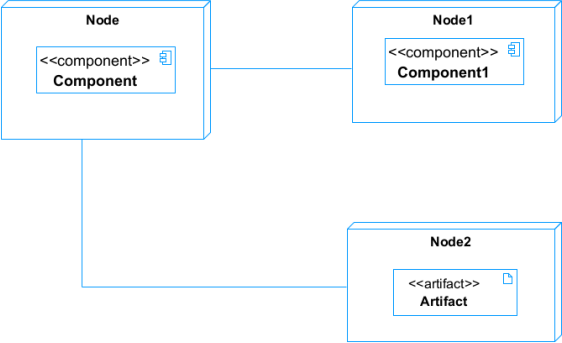


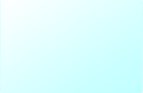
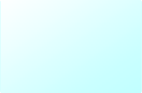
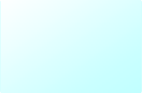
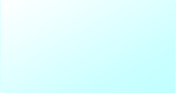
Figure 28: Formalism of a Deployment Diagram

1. *Components of a Deployment Diagram*

Table 19: Components of a Deployment Diagram

|  |  |  |
| --- | --- | --- |
| ELEMENTS | REPRESENTATION | DESCRIPTION |
| Node |  | It is a hardware used to deploy the application |
| Artifact |  | An artifact is a major product, which is produced or used during the development of a software. E.g. diagrams, data models, setup scripts |
| Component |  | It represents a modular part of a system that encapsulates its content and whose manifestation is replaceable within its environment. |
| Association | association | An association helps to connect two nodes together which permits them to communicate together |

1. *Materio Deployment diagram*



1.1

serves

Database server

TCP/IP

is connected

1.1

POSTGRESQL

Frontend

Client server

is connected HTTP

1.1

1.1

serves

<<access>>

Noeud\_2 Backend

API's

is connected

1.1

TCP/IP

1.1

serves

Payment API

Django

Navigator (Google Chrome)

React js

Figure 29: Materio Deployment Diagram

* + 1. COMPONENT DIAGRAM
       1. *Definition*

Component diagrams are used to model the physical aspect of a system. Now the question is what are this physical aspect? They are elements such as Executables, libraries, files, document etc. which resides in a node. The component diagram does not describe the functionality of the system, but it describes the components used to make those functionalities.

1. *Formalism*

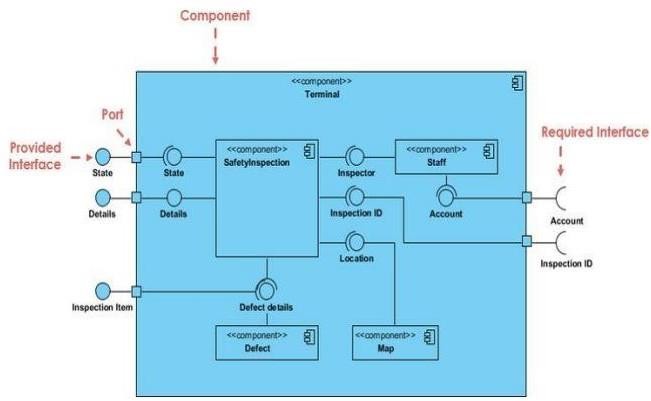


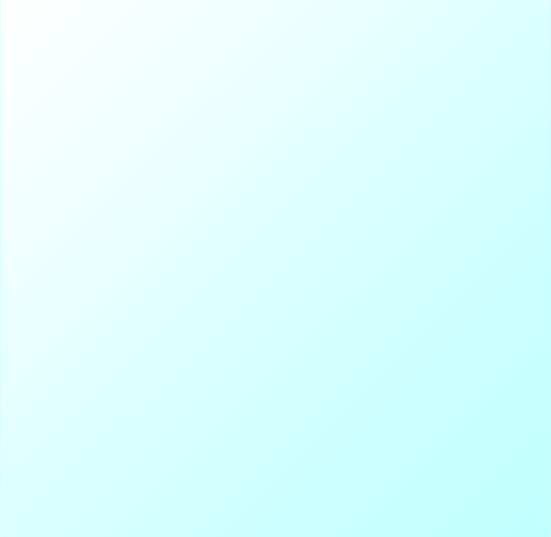
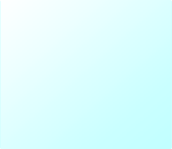
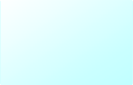
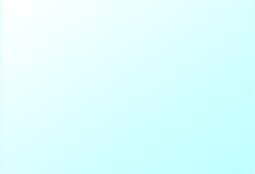
Figure 30: Formalism of Component Diagram (Source: https://www.pinterest.com/pin/551128073157994549/)

1. *Components of a Component Diagram*

Table 20: Components of a Component Diagram

|  |  |  |
| --- | --- | --- |
| ELEMENTS | REPRESENTATION | DESCRIPTION |
| A Component |  | A component is an abstract logical unit block of a system. It is represented as a rectangle with smaller rectangle in the upper right corner which saves as it icon for recognition. |
| Dependency |  | Dependency is a directed relationship which is used to show that some components are dependent on others for their correct functioning. |
| Required Interface |  | It is a straight line from the component box with an attached half circle representing interfaces where a component requires information to perform its own functions. |
| Provided Interface |  | It is a straight line from the component box with an attached circle representing interfaces where a component produces information used by required interfaces |
| Port |  | A port (represented by a small square at the end of a required or provided interface) is used when the components delegate the interfaces to an internal class. |

1. *Materio Component Diagram*



HTTP request (GET)

HTTP request (POST,PUT,PATCH)

HTTP GET

request

HTTP/POST

request

Backend

HTTP GET request

HTTP request (POST,PUT,PATCH)

Web Server

Nexus Payment API

Database

View

Model

Django Rest Framework

Python

Web app

Frontend JavaScript

React JS

Component

PostgreSQL

Figure 31: Materio Component Diagram

* + 1. TECHNOLOGICAL CHOICES
       1. *Software used for the development of Materio*

Table 21: Softwares used for the development of Materio

|  |  |  |  |
| --- | --- | --- | --- |
| SOFTWARE | VERSION | ROLE | IMAGE |
| OS WINDOWS 11 | 23H2 | The operating system required to run the software is the one on which we have worked. | Description : A close-up of a blue object  Description automatically generated |
| Visual studio code | 1.70.0.8 | Text editor used to write down lines of codes that would be interpreted by the navigator. | Description : A blue triangle with a cross  Description automatically generated |
| Postman | 9.4 | It is a platform that simplifies every step of the API lifecycle and streamlines collaboration, making it easier and faster to create better APIs. | Description : A white figure in a circle  Description automatically generated |
| GitHub | 3.0.6 | It is a website and cloud service that helps developers store and manage their code. | Description : A logo with a cat  Description automatically generated |
| Framework reactjs | 0.69.0 | React Native is an English framework built on JavaScript for creating applications. | A blue and white logo  Description automatically generated |
| EDRAW MAX | 11.5.0 | the software you're referring to is a 2D technical and business diagram creation tool that helps create organizational charts, flowcharts, mind maps, network diagrams, floor plans, workflow diagrams, business charts, and engineering diagrams. |  |
| SQLite | v0.14.1 | Database Management System |  |
| Power AMC | 17.1.0.0 | The software engineering workshop used for solution modeling. We utilized this tool for modeling the various diagrams of our system. |  |

**CONCLUSION**

In our realization phase, we implemented our application. In other to accomplish this phase, we made used of our analysis and conception phase. We also drew the deployment and component diagrams which depict the structure of our system in terms of modules, files, assets, how the different elements interact with each other. We will move to the test of functionalities phase, where we will examine the different modules present in our app and how beneficial they are to its different users.

# CHAPTER 6: TEST OF FUNCTIONALITIES

### Preamble

The purpose of the user guide is to provide users of our platform with step-by-step instructions on how to install and use the system.

### Content overview



**INTRODUCTION**

1. **APPLICATION FUNCTIONALITIES**
2. **TESTS SHOWCASES CONCLUSION**

**INTRODUCTION**

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 Materio.

#### 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.

 Deliveries

This functionality manages the delivery process after an order has been placed. It enables the delivery agent to view orders and update the delivery status.

 Cart management

It helps the users keep a desired product in a cart either for instant purchase or future purchase. Here, the user can decide to add or reduce the quantity of the product in the cart or even delete it.

 User Complaints

The user here can type a message addressed to the admin if encountering a problem on the application. This is mostly done in the process of product acquisition.

 3D View

The user here can have a better perspective of some products in the store. This is to help him/her feel more confident about the quality and appearance of the product.

 Logout

It enables the users to close a working session.

#### TEST SHOWCASES

These tests were carried out to test the API of the user.

**Testing the POST**

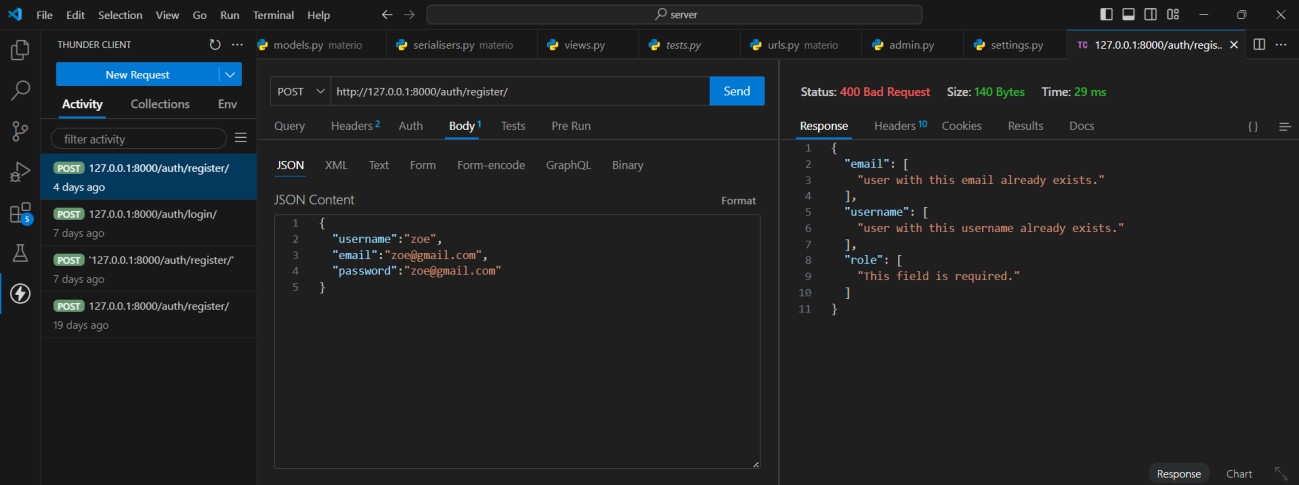
****

Figure : /POST/http://127.0.0.1:8000/auth/register/

**Testing the GET**

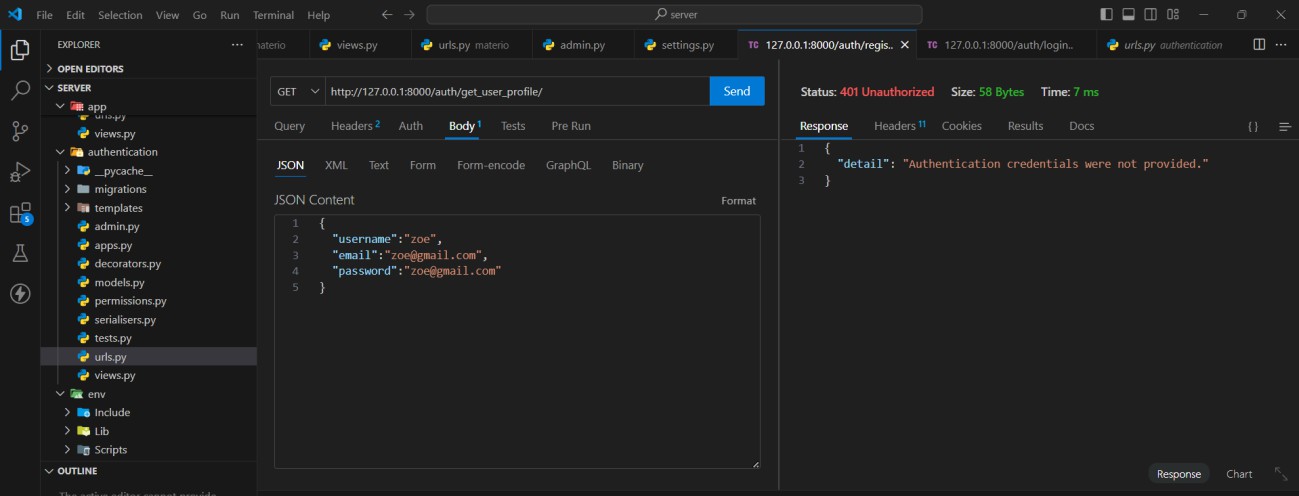
****

Figure : /GET/http://127.0.0.1:8000/auth/get\_user\_profile/

**CONCLUSION**

In the test of functionalities phase, we explained the different functionalities of our application and the benefits to the various users (customer, delivery agent, admin). 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 install and use Materi

# CHAPTER 7: INSTALLATION GUIDE AND USER GUIDE

### Preamble

In this phase, we will present the various functionalities of our application.

### Content overview



**INTRODUCTION**

**I.**

**II.**

**INSTALLATION OF THE APPLICATION**

**USER GUIDE**

**CONCLUSION**

**INTRODUCTION**

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 features, 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 showcase our application by viewing the different screens.

#### INSTALLATION OF THE APPLICATION

* 1. Prerequisites

To use this application the user would need.

* To have a Desktop.
* Have a browser installed.

### Steps for Installation

## 1. Install Web Application

Install the necessary web server components (Node.js, Python).

**Node.js**

## Download the Installer

* Go to the Node.js-Download-page and download the Windows Installer for the latest LTS (Long-Term-Support) version

## Run the Installer

* Double clicks the downloaded ‘.msi’ file to launch the Node.js Setup Wizard.
* Click ‘Next’ to proceed through the installation steps
* Accept the license agreement and choose the installation location (default is usually fine)
* Select components to install (default options are recommended)
* Click ‘Install’ and provide administrator permissions if prompted

## Verify Installation

Open Command Prompt and run

node-v

npm-v

* Deploy the web app to a local server.
* Connect the web app to the database or API that stores sensor data.

## Python

**Download the Installer:**

* Go to the official Python website (python.org).
* Click on "Downloads" and select the latest version for Windows.
* Choose either the 64-bit or 32-bit installer based on your system.

## Run the Installer:

* Launch the downloaded .exe file.
* Check the box that says, "Add Python to PATH".
* Click "Install Now" for standard installation.

## Verify Installation:

Open Command Prompt and type:

II. USER GUIDE

python--version

Here we have the first interface displayed when the Materio application is opened. It automatically displays the home page.

Figure : Materio Home Page

After the home page, we move to the store page. This page is displayed after clicking on the shop now button. It presents some materials for customers to add them to the cart and procced to payment.

Figure : Materio Store

This interface is our Sign In page. This page is displayed to the visitors who want to perform a payment. If we already have an account, we can simply enter our credentials (username and password) and click on “Sign” to submit. If we don’t have an account, we are supposed to click on the “Sign Up” link to create an account.

Figure : Materio Login Page

As we were previously on the Sign In page, this is now our Sign Up page where we are required to enter our username, email, and password. Then click on “Sign Up” if one of the pieces of information does not respect the formalism. The account will not be created.

Figure : Materio Signup Page

**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 to run this application without any problem and how the users will use this platform once the environment is set up.

**GENERAL CONCLUSION**

Having reached the end of our project, we are proud of the progress made despite the challenges encountered. Working on the theme **“Design and Implementation of an Online Construction Materials Ordering and Delivery Application with 3D Visualization”**, we aimed to simplify the process of acquiring construction materials by reducing the need for customers to visit physical stores. We began by identifying the existing system, defining the requirements, and analyzing the solution using the UML-2TUP methodology. Sybase Power AMC was used to model our diagrams. The system, built with React.js for the frontend, Django for the backend, and SQLite as the database, allows users to order materials and have them delivered directly to project sites. Further updates will be made to enhance functionality, security, and reliability.

**PERSPECTIVES**

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

* Implementing an AI to help the customers know the construction materials they need for a project if they don’t.
* Using Virtual and Augmented Reality for construction materials proposed to the customers.

**BIBLIOGRAPHY**

**-IC3 (Internet Computing Core Certification) Course Material,** 1st year, African Institute of Computer Science – Paul BIYA Technological Excellence Center, 2022, by Mrs. TCHOUTOUO.

**-UML Course Material,** 2nd year, African Institute of Computer Science – Paul BIYA Technological Excellence Center, 2023, by Mr. MESSIO Ulrich

**-The Small Guide for Writing a thesis or Internship Report (1999),** Charles de Gaulle University, Lille III, Faculty of Mathematics, Economic and Social Sciences.

**-Scientific Writing Course Material,** African Institute of Computer Science – Paul BIYA Technological Excellence Center, 2024, by Mrs. EBENYE.

**-Database Design Course Material,** African Institute of Computer Science – Paul BIYA Technological Excellence Center, 2023, by Mr. FOMEKONG

**-Database Implementation Course Material,** African Institute of Computer Science – Paul BIYA Technological Excellence Center, 2023, by Mr. MESSIO Ulrich

**-OMG, OMG Unified Modelling Language 2.5,** Object Management Group, March 2015, 794 pages.

**WEBOGRAPHY**

* <https://dribbble.com/shots/24780061-OpenProperties-Visual-Identity-Designs> for the design of our Web Application.
* [https://www.figma.com/design/9qqanpJ9iuOy4Y86PAQ47r/Untitled?node-id=1-](https://www.figma.com/design/9qqanpJ9iuOy4Y86PAQ47r/Untitled?node-id=1-148&t=dKbjaPXOIy7EY8Ja-0) [148&t=dKbjaPXOIy7EY8Ja-0](https://www.figma.com/design/9qqanpJ9iuOy4Y86PAQ47r/Untitled?node-id=1-148&t=dKbjaPXOIy7EY8Ja-0) also for the design of our application.
* [https://www.google.com/search?q=w3schools&oq=&gs\_lcrp=EgZjaHJvbWUqBggBEE](https://www.google.com/search?q=w3schools&oq&gs_lcrp=EgZjaHJvbWUqBggBEEUYOzIGCAAQRRg5MgYIARBFGDsyBggCEEUYQDIMCAMQABhDGIAEGIoFMgcIBBAAGIAEMgYIBRBFGDwyBggGEEUYPDIGCAcQRRg80gEJMjI3MTBqMGo0qAIAsAIB&sourceid=chrome&ie=UTF-8) [UYOzIGCAAQRRg5MgYIARBFGDsyBggCEEUYQDIMCAMQABhDGIAEGIoFMgc](https://www.google.com/search?q=w3schools&oq&gs_lcrp=EgZjaHJvbWUqBggBEEUYOzIGCAAQRRg5MgYIARBFGDsyBggCEEUYQDIMCAMQABhDGIAEGIoFMgcIBBAAGIAEMgYIBRBFGDwyBggGEEUYPDIGCAcQRRg80gEJMjI3MTBqMGo0qAIAsAIB&sourceid=chrome&ie=UTF-8) [IBBAAGIAEMgYIBRBFGDwyBggGEEUYPDIGCAcQRRg80gEJMjI3MTBqMGo0qA](https://www.google.com/search?q=w3schools&oq&gs_lcrp=EgZjaHJvbWUqBggBEEUYOzIGCAAQRRg5MgYIARBFGDsyBggCEEUYQDIMCAMQABhDGIAEGIoFMgcIBBAAGIAEMgYIBRBFGDwyBggGEEUYPDIGCAcQRRg80gEJMjI3MTBqMGo0qAIAsAIB&sourceid=chrome&ie=UTF-8)

[IAsAIB&sourceid=chrome&ie=UTF-8](https://www.google.com/search?q=w3schools&oq&gs_lcrp=EgZjaHJvbWUqBggBEEUYOzIGCAAQRRg5MgYIARBFGDsyBggCEEUYQDIMCAMQABhDGIAEGIoFMgcIBBAAGIAEMgYIBRBFGDwyBggGEEUYPDIGCAcQRRg80gEJMjI3MTBqMGo0qAIAsAIB&sourceid=chrome&ie=UTF-8) to document ourselves on the programming languages used in the development of our Web Application.

* <https://fr.freepik.com/> to get all the necessary images for our Web Application.

**VIDEOGRAPHY**

* [https://www.youtube.com/watch?v=QFaFIcGhPoM&list=PLC3y8-](https://www.youtube.com/watch?v=QFaFIcGhPoM&list=PLC3y8-rFHvwgg3vaYJgHGnModB54rxOk3) [rFHvwgg3vaYJgHGnModB54rxOk3](https://www.youtube.com/watch?v=QFaFIcGhPoM&list=PLC3y8-rFHvwgg3vaYJgHGnModB54rxOk3) to help us get started with React js.
* [https://www.youtube.com/watch?v=UmljXZIypDc&list=PL-](https://www.youtube.com/watch?v=UmljXZIypDc&list=PL-osiE80TeTtoQCKZ03TU5fNfx2UY6U4p) [osiE80TeTtoQCKZ03TU5fNfx2UY6U4p](https://www.youtube.com/watch?v=UmljXZIypDc&list=PL-osiE80TeTtoQCKZ03TU5fNfx2UY6U4p) to help us get started with Django.
* [https://www.youtube.com/watch?v=ur6I5m2nTvk&list=PL4cUxeGkcC9ixPU-](https://www.youtube.com/watch?v=ur6I5m2nTvk&list=PL4cUxeGkcC9ixPU-QkScoRBVxtPPzVjrQ) [QkScoRBVxtPPzVjrQ](https://www.youtube.com/watch?v=ur6I5m2nTvk&list=PL4cUxeGkcC9ixPU-QkScoRBVxtPPzVjrQ) to help us get started with React Native.
* [https://www.youtube.com/watch?v=2nZiB1JItbY&list=PLDyQo7g0\_nsX8\_gZAB8KD1l](https://www.youtube.com/watch?v=2nZiB1JItbY&list=PLDyQo7g0_nsX8_gZAB8KD1lL4j4halQBJ) [L4j4halQBJ](https://www.youtube.com/watch?v=2nZiB1JItbY&list=PLDyQo7g0_nsX8_gZAB8KD1lL4j4halQBJ) to master React js.

**TABLE OF CONTENT**

[DEDICATION 1](#_Toc201682376)

[ACKNOWLEDGEMENT 2](#_Toc201682377)

[SUMMARY 3](#_Toc201682378)

[LIST OF FIGURES 4](#_Toc201682379)

[LIST OF TABLES 6](#_Toc201682380)

[GLOSARY 7](#_Toc201682381)

[ABSTRACT 8](#_Toc201682382)

[RESUME 9](#_Toc201682383)

[LIST OF ABBREVIATIONS 10](#_Toc201682384)

[GENERAL INTRODUCTION 1](#_Toc201682385)

[CHAPTER 1: EXISTING 4](#_Toc201682386)

[Preamble 4](#_Toc201682387)

[Content overview 4](#_Toc201682388)

[CHAPTER 2: SPECIFICATION BOOK 11](#_Toc201682389)

[CHAPTER 3: ANALYSIS PHASE 26](#_Toc201682390)

[Preamble 26](#_Toc201682391)

[Content overview 26](#_Toc201682392)

[CHAPTER 4: CONCEPTION PHASE 64](#_Toc201682393)

[Preamble 64](#_Toc201682394)

[Content overview 64](#_Toc201682395)

[CHAPTER 5: REALIZATION PHASE 81](#_Toc201682396)

[Preamble 81](#_Toc201682397)

[Content overview 81](#_Toc201682398)

[CHAPTER 6: TEST OF FUNCTIONALITIES 93](#_Toc201682399)

[Preamble 93](#_Toc201682400)

[Content overview 93](#_Toc201682401)

[CHAPTER 7: INSTALLATION GUIDE AND USER GUIDE 98](#_Toc201682402)

[Preamble 98](#_Toc201682403)

[Content overview 98](#_Toc201682404)

[Steps for Installation 100](#_Toc201682405)

[1. Install Web Application 100](#_Toc201682406)

[Download the Installer 100](#_Toc201682407)

[Run the Installer 100](#_Toc201682408)

[Verify Installation 101](#_Toc201682409)

[Python 101](#_Toc201682410)

[Run the Installer: 101](#_Toc201682411)

[Verify Installation: 101](#_Toc201682412)