**REPUBLIQUE DU CAMEROUN**

Paix-Travail-Patrie

**REPUBLIC OF CAMEROON**

Peace-Work-Fatherland

A logo with a map and a globe

Description automatically generated 

**N-TEC LABS**

PO BOX : Grand Chefferie Simbock, Immeuble DC

Tel: (+237)651834545

Website: www.nteclabs.com

Email: contact@nteclabs.com

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

PO Box: 13719 Yaoundé

Tel: (+237)242729957 / (+237)242729958

Website: [www.iaicameroun.com](http://www.iaicameroun.com)

Email: contact@iaicameroon.com

Email: contact@iaicameroun.com

INTERNSHIP REPORT

RESTAURANT MANAGEMENT SYSTEM

Internship Carried out from the 01st July to 30th September 2023

In view of obtaining a **Higher Technician Diploma (HTD)** in computer sciences option

**Software Engineering**

Submitted by

**NDIP LUCY-DIANE BANYI**

**Level 2 Student at AICS Cameroon**

Supervisors

**Academic Supervisor**

**Mr. NGUH PRINCE**

**Lecturer at AICS-Cameroon**

**Professional Supervisor**

**Miss NFORMI VICTORY**

**Developer at N-tech labs**

**ACADEMIC YEAR 2023-2024**

# ACKNOWLEDGEMENTS

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

* We give thanks to God who made me able to this project.
* The Resident Representative of African Institute of Computer Sciences Yaounde Cameroon, **Mr. ARMAND CLAUDE ABANDA.**
* **Mr. NGUH PRINCE** our academic supervisor and lecturer at AICS-Cameroon for his availability, advice and moral support throughout theinternship period which helped in therealization of this project.
* The CEO of N-Tech Labs Mr. **NGWANYI JOEL** for having welcomed us and validate our application.
* **Miss FONGWA BLESSING** for having welcomed us, encouragements, availability, support, supervision, counseling and motivation throughout the internship period.
* **Miss** **NFORMI VICTORY** our professional supervisor, for her warm welcome, encouragements, availability, support, supervision, counseling and motivation throughout the internship period.
* To all the teachers of the African Institute of Computer Sciences Yaounde Cameroon for their help and support throughout our training.
* **MR MBAH ROYCE CHUM ALEM** for his constant verification, support, counselling
* To my parents and family for their guidance and support and for have done the required sacrifice to have made me integrate in this prestigious and talented institution
* To all friends and classmates at AICS for their knowledge, ideas, collaboration and most importantly the unforgettable moments spent together.
* To a special friend **BONG SHALOM** for her help, support, advice for the constant encouragement in the realization of our report

# DEDICATION

**THANKS TO MY PARENTS AND FAMILY**

FOR THEIR LOVE, ADVICE AND UNCONDITIONAL SUPPORT

# CONTENT

[ACKNOWLEDGEMENTS 2](#_Toc176268886)

[DEDICATION 3](#_Toc176268887)

[CONTENT 4](#_Toc176268888)

[LIST OF TABLES i](#_Toc176268889)

[LIST OF FIGURES ii](#_Toc176268890)

[GLOSSARY iii](#_Toc176268891)

[ABSTRACT iv](#_Toc176268892)

[RESUME 1](#_Toc176268893)

[GENERAL INTRODUCTION 2](#_Toc176268894)

[PART I 3](#_Toc176268895)

[INSERTION PHASE 3](#_Toc176268896)

[INTRODUCTION 5](#_Toc176268897)

[Conclusion 16](#_Toc176268898)

[PART II 17](#_Toc176268899)

[EXISTING SYSTEM 17](#_Toc176268900)

[INTRODUCTION 20](#_Toc176268901)

[CONCLUSION 26](#_Toc176268902)

[SPECIFICATION BOOK 27](#_Toc176268903)

[INTRODUCTION 29](#_Toc176268904)

[CONCLUSION 42](#_Toc176268905)

[ANALYSIS PHASE 43](#_Toc176268906)

[INTRODUCTION 45](#_Toc176268907)

[CONCLUSION 89](#_Toc176268908)

[CONCEPTION PHASE 90](#_Toc176268909)

[INTRODUCTION 92](#_Toc176268910)

[CONCLUSION 105](#_Toc176268911)

[REALIZATION PHASE 106](#_Toc176268912)

[INTRODUCTION 108](#_Toc176268913)

[CONCLUSION 117](#_Toc176268914)

[Functionality Test 118](#_Toc176268915)

[INTRODUCTION 120](#_Toc176268916)

[USER GUIDE 121](#_Toc176268917)

[USER GUIDE 122](#_Toc176268918)

[INTRODUCTION 123](#_Toc176268919)

[GENERAL CONCLUSION 135](#_Toc176268920)

[Annexe 136](#_Toc176268921)

[BIBLIOGRAPHY 137](#_Toc176268922)

[WEBOGRAPHY 138](#_Toc176268923)

[TABLE OF CONTENT 139](#_Toc176268924)

# LIST OF TABLES

[Table 1 Hardware resources of HADRON SA 13](#_Toc147113584)

[Table 2 Software resources of HADRON SA 13](#_Toc147113585)

[Table 3: Criticism of the system 21](#_Toc147113586)

[Table 4 Project's Plan 32](#_Toc147113587)

[Table 5 Software resources of the project 35](#_Toc147113588)

[Table 6 hardware resources of the project 36](#_Toc147113589)

[Table 7 Human Resources of the project 36](#_Toc147113590)

[Table 8: Comparison between UML and MERISE 47](#_Toc147113591)

[Table 9: formalism of textual description of a use case 61](#_Toc147113592)

[Table 10: textual description of use case Report Lost Card 62](#_Toc147113593)

[Table 11 textual description of use case Found Card with Image 63](#_Toc147113594)

[Table 12: Formalism of an activity diagram 67](#_Toc147113595)

[Table 13: Components of an activity Diagram 68](#_Toc147113596)

[Table 14: formalism of Communication Diagram 74](#_Toc147113597)

[Table 15: Component of a class Diagram 97](#_Toc147113598)

[Table 17: description of element in deployment diagram 105](#_Toc147113599)

[Table 16: component of a Component Diagrams 107](#_Toc147113600)

[Table 18: Software tools used in the project 110](#_Toc147113601)

[Table 19: languages used in the project. 111](#_Toc147113602)

# LIST OF FIGURES

[Figure 1: geographical location of HADRON SA 10](#_Toc147113555)

[Figure 2 organization chart of HADRON SA 12](#_Toc147113556)

[Figure 3: Gannt project planning 34](https://d.docs.live.net/4be1740105e2a1f3/Documents/presentation%20of%20hadron%20(Repaired).docx#_Toc147113557)

[Figure 4: Gantt project 34](#_Toc147113558)

[Figure 5 System Design Phases 43](#_Toc147113559)

[Figure 6 Two-Track Unified Process 49](https://d.docs.live.net/4be1740105e2a1f3/Documents/presentation%20of%20hadron%20(Repaired).docx#_Toc147113560)

[Figure 7: Formalism of Use Case Diagram 52](#_Toc147113561)

[Figure 8: General Use Case Diagram of our system 57](#_Toc147113562)

[Figure 9: Use case diagram of Manage card 58](#_Toc147113563)

[Figure 10: Use case diagram of Upload card. 59](#_Toc147113564)

[Figure 11 use case diagram of Authenticate. 60](#_Toc147113565)

[Figure 12: Activity Diagram of Signup 70](#_Toc147113566)

[Figure 13:Activity diagram of report lost card 71](#_Toc147113567)

[Figure 14: Activity diagram of match Card 72](#_Toc147113568)

[Figure 15: Communication diagram of Lost card 75](#_Toc147113569)

[Figure 16: Communication diagram of found card. 76](#_Toc147113570)

[Figure 17: Communication Diagram of Register 77](#_Toc147113571)

[Figure 18: formalism of Sequence diagram 78](#_Toc147113572)

[Figure 19: reporting lost card Sequence Diagram 82](#_Toc147113573)

[Figure 21 SignUp Sequence Diagram 83](#_Toc147113574)

[Figure 31: MVT pattern in Django 89](#_Toc147113575)

[Figure 23: package diagram of the system 92](#_Toc147113576)

[Figure 24 formalism of state machine diagram 93](#_Toc147113577)

[Figure 25: state machine diagram <<User's Card>> 95](#_Toc147113578)

[Figure 26: State machine diagram <<Finder's Entry>> 96](#_Toc147113579)

[Figure 27: Class Diagram for our system 99](#_Toc147113580)

[Figure 28: Visual components of Deployment Diagram 104](#_Toc147113581)

[Figure 29:deployment diagram of the solution 106](#_Toc147113582)

[Figure 30 Component Diagram of the Software solution 109](#_Toc147113583)

# ABSTRACT

Restaurants play an important role in society by contributing to culture, economy, and social life, in order to facilitate management of services restaurants should have computer systems.

Most restaurants do not have computer systems which aid them to manage services properly, hence they struggle with manual order taking and processing leading to errors and delays, services are slow leading to longer wait for customers, cash handling errors where money can be stolen by employees, difficulty in tracking sales trends, customer preferences.

In this report we are proposing the design and development of a **RESTAURANT MANAGEMENT APPLICATION**. To do this, studies were conducted as well as a series of interviews with users (customers, servers, kitchen staff, managers). We used 2TUP methodology practices for the development of our application, the requirements specifications allowed us to formalize the data interaction with UML (Unified Modelling Language). In terms of structuring and storing data, we used the MySQL Relational Database Management System (RDBMS). The implementation in PHP allowed us to set up a flexible, secure, and easy to use and perfectly manage services in restaurants. By developing this application restaurants can significantly improve their operational efficiency, enhance customer satisfaction, and gain valuable insights to drive growth and profitability.

# RESUME

Les restaurants jouent un rôle important dans la société en contribuant à la culture, à l'économie et à la vie sociale. Pour faciliter la gestion des services, les restaurants devraient disposer de systèmes informatiques.

La plupart des restaurants ne disposent pas de systèmes informatiques qui les aident à gérer correctement les services, ce qui les amène à lutter avec la prise de commandes et le traitement manuels, entraînant des erreurs et des retards. Les services sont lents, ce qui entraîne une attente plus longue pour les clients, des erreurs de manipulation d'argent où l'argent peut être volé par les employés, et des difficultés à suivre les tendances des ventes et les préférences des clients.

Dans ce rapport, nous proposons la conception et le développement d'une APPLICATION DE GESTION DE RESTAURANT. Pour ce faire, des études ont été menées ainsi qu'une série d'entretiens avec les utilisateurs (clients, serveurs, personnel de cuisine, gestionnaires). Nous avons utilisé les pratiques de la méthodologie 2TUP pour le développement de notre application, les spécifications des exigences nous ont permis de formaliser l'interaction des données avec UML (Langage de Modélisation Unifié). En termes de structuration et de stockage des données, nous avons utilisé le Système de Gestion de Base de Données Relationnelle (SGBDR) MySQL. L'implémentation en PHP nous a permis de mettre en place un système flexible, sécurisé et facile à utiliser pour gérer parfaitement les services dans les restaurants. En développant cette application, les restaurants peuvent améliorer considérablement leur efficacité opérationnelle, augmenter la satisfaction des clients et obtenir des informations précieuses pour stimuler leur croissance et leur rentabilité.

# GENERAL INTRODUCTION

Restaurants are multipurposed establishments that offer much more than just food, they are places of social connection, cultural exchange, entertainment, and economic activity, serving a wide range of purposes for individuals and communities. Cameroonians go to restaurants to have an experience and they expect to be served well. Thus, it is important to manage restaurant service to enhance this experience.

The purpose of this report with the theme **“COMPUTERIZED MANAGEMENT OF RESTAURANT SERVICES”** will help enhance customers experience and to do this we went through all the following phases. We divided this report into 6 main parts which are shown below:-

1. **The insertion phase:** Here, we present the company in which we did our internship, and the integration of the interns into the company.
2. **Existing system**: Here, we present the present system and evaluate its advantages and most importantly its downfalls.
3. **The specification book**: Here, we identify the need of the future system users and point out different constraints of the project.
4. **Analysis phase:** Here, we choose our analysis method and provides all the diagram needed to visually represent our system both the infrastructure and the data structure.
5. **The Conception phase**: this presents the generic and detailed conception of the project and the architecture used to bring into reality our analysis.
6. **Realization phase:** In this phase, we will visualize the implementation process of the solution

# PART ONE :

# INSERTION PHASE

Preamble

This phase presents the details of how we were integrated in the host company, the company presentation and organisation.

Overview

INTRODUCTION

1. WELCOME AND INTEGRATION
2. GENERAL PRESENTATION OF THE COMPANY
3. ORGANISATION OF THE XOMPANY
4. HARDWARE AND SOFTWARE RESOURCES OF THE COMPANY
5. BRIEF PRESENTATION OF THE PROJECT THEME

CONCLUSION

INTRODUCTION

The insertion phase is a period (generally of 02 weeks) reserved for the different interns to discover and to familiarize themselves with the working environment. Here, we got to know about the staff, the different hardware and software resources used, the different departments which constitute the enterprise, how the company functions both internally and externally and we were introduced to our work space. During this period, we were also attributed an internship master often called professional supervisor and a theme. We also had a time to discuss amongst us interns on topics like what we love doing most, what we dislike, our beliefs and experiences.

## WELCOME AND INTEGRATRION

### Welcome

We arrived at N-TEC LABS on Monday July 2024 at 8:00am. we were welcomed, by **Mme Fongwa Blessing** the enterprise’s main Secretary, who introduced us to our work space and her collaborators. She then introduced us to the boss **Mr Ngwanyi Joel** and our supervisor **Mme Nformi Victory.** Later on, we were logged onto the Intern portal and we were given a task to test our competence

### Integration

A working day at N-TEC LABS start from 08:00am to 06:00pm. Our tasks in the company generally follow a predefined routine. Every day we were assigned tasks to carry out. There was also an opportunity to present our difficulties and challenges to everyone and receive directives.

## GENERAL PRESENTATION OF THE COMPANY

### Geographical location

### History

N-Tec Labs is a Cameroon based non-governmental tech start-up institute founded by **Mr. Ngwanyi Joel** in 2022 which proposes IT solutions and empowers the use of new technologies and also trains various people in Various IT domains in Cameroon .

### Mission

The mission of N-Tec Labs mainly relates to the empowerment and the perpetuation of technology use. These missions include:

* Design and realization of mobile-oriented software for companies and individuals;
* Offering training in computer technologies and information technology
* Taking an active part in the sustainable development of the world through innovative solutions and virtual reality.

### Vision

At N-Tec Labs, we believe that the true potential of every organization and individual can be unlocked through the harmonious fusion of analysis, development, realization, and data analytics. Our vision propels us forward, guiding us to make a lasting impact on the digital landscape and shape a future where innovation knows no bounds.

### Activities

The activities of N-Tec Labs range from computer sciences, engineering, and training. We can outline the following:

* Computer Training: Hardware , WebMastery, Secretaryship
* Software development and maintenance ;
* Training in Software related fields;
* CCTV Installations
* Wired / Wireless Networking
* IT support

## ORGANISATION OF THE ENTERPRISE

### Administrative Organisation of N-Tec Labs

N-Tec Labs is administratively organized as follows;

#### The General Management

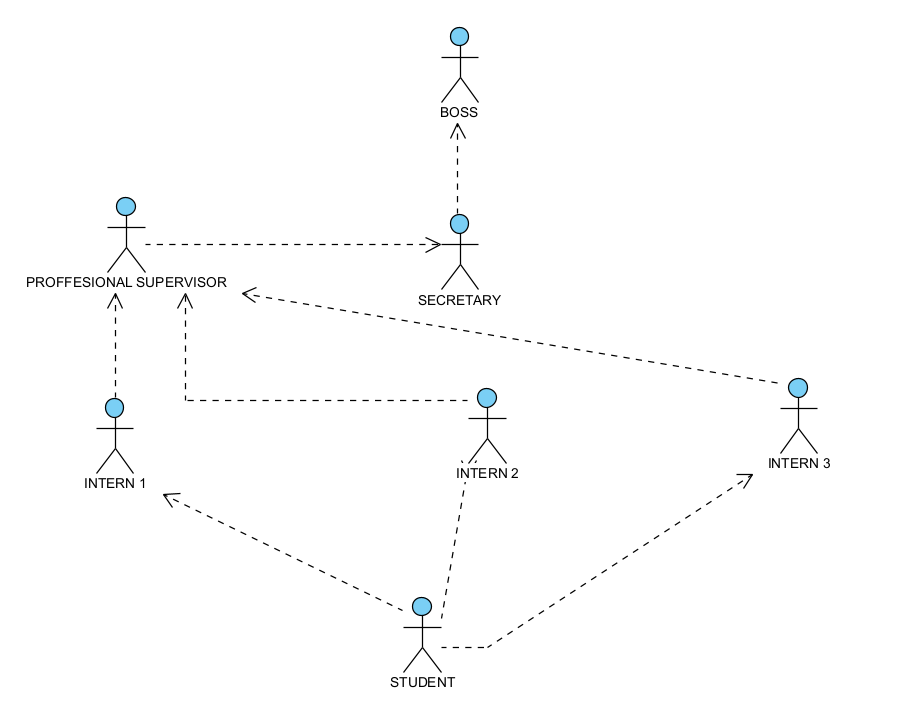
**Boss/ Founder:** Ngwanyi Joel

**Secretary:** Fongwa Blessing

**Supervisor:** Nformi Victory

### Functional organization of N-Tec Labs

The functional branch of N-Tec Labs is organised as follows;



## HARDWARE AND SOFTWARE RESOURCES OF THE COMPANY

### Hardware Resources

**Table 1**: Hardware resources of N-tec labs

|  |  |  |  |
| --- | --- | --- | --- |
|  | Designation | Quantity | Characteristics |
| 1 | Mac Desktop | 4 | APPLE |
| 2 | Hp Desktop | 1 | HP |
| 3 | Starlink | 1 | / |
| 4 | Printer | 2 | HP |
| 5 | Training equipment and office furniture | 22 | / |

### Software Resources

As an enterprise specialised in IT, N-Tec also comprises of software resources such as;

**Table 2**: Software resources of N-Tec Labs

|  |  |
| --- | --- |
| Designation | Software |
| Operating system | Windows 11, Mac OS |
| Design tools | Photoshop |
| Integrated development environment (IDE) | Vs Codium |
| Text editor | Sublime text, Notepad++ |
| Database management system (DBMS) | MongoDB, PostgreSQL, MySQL |
| Web browser | Google Chrome, Microsoft Edge, Safari |
| Document editor | Microsoft Office Word |
| Presentation | Microsoft Office PowerPoint |

1. **BRIEF PRESENTATION OF THE PROJECT**

On our arrival in the enterprise, we were given a task to work on **“RESTAURANT MANAGEMENT SYSTEM”.** In this section, was mainly for the presentation, history, Mission, key realization of Hadron SA and their geographic location. The objective of these phase was to scan the enterprise at its deep level and bring out the problem faced by the company and provide a related IT solution for the problem. In order to make it more efficient. Although the company largely encountered no issues, there was a specific problem I observed, especially in Cameroon. This problem pertained to the **COMPUTERIZED MANAGEMENT OF LOST ID CARDS Case Study Cameroon**

PART III

# SPECIFICATION BOOK

**PREAMBLE**

This document outlines the comprehensive specifications for our project, detailing functional and non-functional requirements, UI design, system limitation, and database structures serving as the development guide.

**CONTENT**

INTRODUCTION

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

CONCLUSION

## INTRODUCTION

The specification book sets the foundation for understanding the software project at hand. It provides an overarching summary of the software’s purpose, the scope of its functionality and the specific objectives it aims to achieve. It also lays out the intended audience of the software, any assumptions or prerequisites for its use, and the strategic approach to its development. Following this, the specification book delves deeper into the intricate details of the software, covering aspects like functional and non-functional requirements, system limitations and constraints, preliminary user interface design, and data structures of database design. In this document, it will serve as a pivotal roadmap guiding the software development process. We will also include project details such as the team involved, constraints, the budget, deadlines, and deliverables.

### CONTEXT AND JUSTIFICATION OF STUDIES

In Cameroon, an ID card isn't just an identification tool—it's pivotal to daily life. It serves as a passport to essential services, from mundane activities like parcel collection at the post office to critical ones like employment or medical consultations. Hence, losing an ID card isn't a mere inconvenience; it's a significant setback. The process of obtaining a replacement is not only long but often fraught with bureaucratic delays and ambiguity.

These challenges spurred a series of reflections:

* Could there be an efficient way to broadcast the loss of an ID card?
* If someone stumbles upon a lost ID card, how might they seamlessly connect with the rightful owner?
* Can technology be harnessed to alleviate this widespread problem?
* What if there existed a platform, a communal hub, where individuals assist each other in tracing lost ID cards?
* Is it possible to negate the prolonged wait and anxiety associated with a lost ID card?

Pondering over these questions gave birth to our solution: "Computerized Management of Lost ID Cards." Envisioned as a bridge linking those who've misplaced their ID cards with those who discover them, this system taps into the spirit of community. It ensures that in Cameroon, no one faces undue hardship just because of a momentarily misplaced ID card.

### OBJECTIVES OF THE PROJECT

#### **General objectives**

To design and implement a digital platform that aids in the efficient recovery and management of lost ID cards in Cameroon.

#### **Specific Objectives**

In order to achieve the main objective, there are specific objectives to be achieved.

* **Identification & Reporting**: Create a user-friendly interface where individuals can sign up and login and easily report their lost ID cards.
* **Search & match**: implement an advanced search algorithm that matches reported lost ID cards with found ones in the system.
* **Notifications**: Establish an effective notification system that alerts the user who lost when the card is found
* **Safety and Security**: Ensure that the platform protects users ‘sensitive information, maintaining confidentiality while also being accessible for recovery purposes.
* Name of the project: **LoFiCard**
* Project target: Cameroon population
* Technical specification : mobile application mainly.

### EXPRESSION OF NEEDS

In this section we are going to outline the specific needs or requirements that the software project is intended to meet. It is a bridge between the problem and the solution detailing the ‘what’ and not the ‘how’. And this need is going to be divided into functional and non-functional needs.

#### **Functional needs**

Here, we are talking about **what** the system is supposed to do for the user of the application, and these features are.

* User should signup and sign in (we should be able to identify the user)
* User can enter information of lost id cards (name, card matricula, type, day lost)
* User can enter information of found id cards (matricula, day found, type, his contact, image)
* User can get notification in case his card has been found and the contact of the person who founds.

**Users who lost their cards:**

* Must login
* Provides information about their lost card (how was it lost, the card details, the day it was lost and if possible, the time they have lost the cards)
* Receives a notification when his cards have been found.
* Knows where and how to take it (option to call the police in case it was assaults)

**User who found the cards:**

* Must login
* Register the found cards in the database with image.
* Provide information concerning him and how to contact him.
* Gives where is the cards.

#### **NON-FUNCTIONAL NEEDS**

The non-functional need we are referring to here, is the ‘quality attributes’ of the software, that is, how the software should behave in order to increase user expectations and satisfaction and just below we are going to see the non-functional need that our application needs to have.

* **Security:** User personal data should be securely stored and transmitted in order to respect user privacy.
* **Performance:** The application should be able to load quickly and operate smoothly to ensure maximum user satisfaction
* **Usability:** it must be easy to use and navigate in order to avoid making the app seems complicate to use by the user
* **Reliability:** it must be reliable in order to meet up market expectation and to promote the needs of the application

### PLANNING OF THE PROJECT

Table 4 Project's Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Week | period | task | method |
| INSERTION PHASE | | | |
| Weeks 1-2 | **Morning**  (08h30 – 12h) | Documenting on the structure of  the organization | theoretical |
| **Afternoon**  (13h-17h) | 1. General presentation of Hadron    * History and Evolution of Hadron    * Missions and activities    * Key Realization    * Geographical location 2. Administrative and Functional Organization of Hadron 3. Hardware and Software Resources of Hadron 4. Brief Presentation of the project theme   Conclusion (13h-17h) | practical |
| TECHNICAL PHASE | | | |
| Weeks 3 Specification | Morning  (08h30 – 12h) | Understanding the themes   * Ask frequent questions to understand the basic needs. * Understand the basic objectives of the application with the stakeholder | practical |
| Afternoon (13h-17h) | Requirements   * Gathering and defining the requirements * Documenting the specifications accurately | Theoretical/practical |
| Weeks 4-7  Analysis phase | Morning  (08h30 – 12h) | Documenting on the method of  analysis (UML), diagrams and | Theoretical |
| Afternoon (13h-17h) | Implementing on the detailed  structure of the application using UML diagrams | Practical |
| Weeks 8-9 Design | Morning  (08h30 – 12h) | Documenting the application architecture and  component/modules and UI | Theoretical |
| Afternoon  (13h-17h) | Designing the components and UI | Practical |
| Week 10-12 deploying and testing the application | Morning  (08h30 – 12h) | * Writing the source code * Setting up the   development environment | Practical |
| Afternoon (13h-17h) | * Preparing the application for deployment * Testing the application for deployment | Practical |

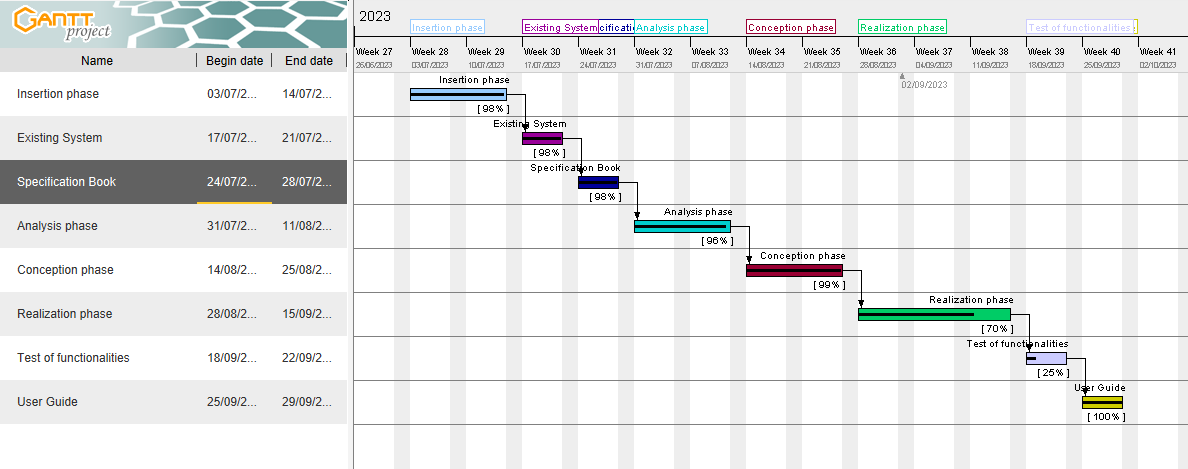


Figure 3: Gannt project planning

Figure 4: Gantt project

### ESTIMATION OF THE PROJECT

#### **SOFTWARE RESOURCES**

Table 5 Software resources of the project

|  |  |  |  |
| --- | --- | --- | --- |
| SOFTWARE | USAGE | QUANTITY | PRICE(FCFA) |
| Microsoft office 2016 | Used for the creation of the report and power point | 1 | 287,500 |
| Visual studio | Open and free software for the coding part of the project | 1 | Free |
| PowerAMC | For doing the design of the diagrams to represent the structure and the interaction between the different actors | 1 | 85000 |
| Microsoft edge | For doing the research on the internet and testing of the application | 1 | Free |
| Windows 11 (64 bits) HOME | Operating system | 1 |  |
| Gannt Project | For building a gannt chart for the planning of the work | 1 | free |
| Postgresql | For creating database | 1 | Free |
| TOTAL 1 |  |  | 372,500 FCFA |

#### **HARDWARE RESOURCES**

Table 6 hardware resources of the project

|  |  |  |  |
| --- | --- | --- | --- |
| MATERIAL | USAGE | QUANTITY | PRICE (FCFA) |
| LAPTOP (HP CORE i5 11gen, SSD) | Used to build the system, type the report, create the power point, make research, etc. | 1 | 700,000 |
| USB flash disk 32gb | Used for storage and to print the report | 1 | 16,000 |
| Modem camtel | Used for internet connection | 1 | 65,000 |
| Mobile phone (iPhone XR) | Used for carrying research and storage | 1 |  |
| Printer HP Officejet 4630 | Used to print the scientific document | 1 | 86,520 |
| Total 2 |  |  | 867,520 FCFA |

#### **HUMAN RESOURCES**

Table 7 Human Resources of the project

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Role | Pay (FCFA)/day | Number | Duration(days) | Price (FCFA) |
| Project manager | 250000 | 1 | 60 | 15,000,000 |
| Analyst | 300000 | 1 | 30 | 9,000,000 |
| Designer | 500000 | 1 | 14 | 7,000,000 |
| Developer | 485000 | 1 | 35 | 16,975,000 |
| TOTAL 3 |  |  |  | 47,975,000 FCFA |

|  |  |  |  |
| --- | --- | --- | --- |
| TOTAL 1 | TOTAL 2 | TOTAL 3 | TOTAL (FCFA) |
| 372500 | 867520 | 47975000 | 49,215,020 FCFA |

Total in letter:

Forty-nine million two hundred and fifteen thousand and twenty Francs CFA

### PROJECT CONSTRAINTS

#### **CRITERIA OF ACCEPTABILITY**

The delivered project is judge acceptable if it respects all the criteria, that is the functionality that was presented.

#### **TIME CONSTRAINTS**

The project will be realized within 3 months that is the time allocated by the school for the internship and realizing of the project that is from July to September

### DELIVERABLES

At the end of this project, we are expected to submit the following:

A complete report containing:

* A User’s guide of the application
* A CD-ROM containing:
* The applications’ source code.
* Database (exported sql file)

## CONCLUSION

This section was a great step in the accomplishment of the project, and we were expected to clarify the different stakeholders the effectiveness and the use of the application in the market and its necessities by giving a detail structure of how the application will behave and help Cameroonians and give the estimation of the project cost for its realization and date of accomplishment of the project.

PART IV

# ANALYSIS PHASE

**PREAMBLE**

The analysis phase is pivotal in software development, setting the stage for the project’s trajectory. This document delves into the core components of our scientific software project, outlining objectives, requirements, risks, existing system, and the analysis method used. Our goal is to establish a blueprint, ensuring the software aligns with stakeholders’ expectations and scientific rigor.

**CONTENT**

INTRODUCTION

1. SOME MODELLING TECHNIQUES
2. PRESENTATION OF ANALYSIS METHODS
3. MODELING WITH UML

CONCLUSION

## INTRODUCTION

Software development is a journey and like any voyage, it requires a clear understanding of the starting point to determine the best route forward. This phase is dedicated to precisely that-providing a comprehensive analysis of our current system and charting a course for its evolution.

From there, the focus shifts to the crux of our analysis-the problematic. Armed with this understanding, we then present advanced analytical and modeling methodologies tailored for our project’s unique needs. Every stage of this analysis is driven by a singular objective: to forge a pathway for the software that is both innovative and meticulously informed. By the conclusion of this phase, we hope to present a vision shaped by rigorous study and forward thinking. The scenario for design can be represented like this:

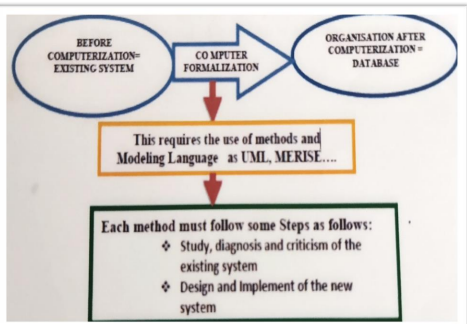


Figure 5 System Design Phases (Source: past report folder)

### SOME MODELLING TECHNIQUES

In the analysis of the software project, it is important to identify the model, or the method used to design and visualize globally the structure of the project depending on the project needs. And we will discuss some of the modelling techniques and the choice made for the realization of the project.

#### **MERISE**

Stands for « Méthode d’Etude et de Réalisation Informatique pour des Systèmes d’Entreprise ». It is a comprehensive system development methodology that originated in France during the 1970s and 1980s. Rooted in a holistic approach, Merise places a strong emphasis on the clear separation of data (what the system must know) and processes (what the system must do). This separation facilitates an in-depth analysis of the system from two complementary perspectives. Merise employs several models to guide the development process, from the conceptual phase to the physical implementation, including the **Conceptual Data Model (CDM), Conceptual Process Model (CPM), and Physical Data Model (PDM).** It is particularly favored for its structure approach to database design, making it a staple in many Francophones educational institutions and businesses.

#### **SCRUM**

Scrum is a widely adopted Agile framework for managing complex product development. Introduced in the early 1990s, Scrum is built around the principles of collaboration, adaptability, and short iterative feedback loops that are not implemented once. Projects are divided into cycles known as Sprints, which typically last two to four weeks. Within these sprints, specific roles like the Scrum Master (who ensures the process runs smoothly), the Product Owner (who defines the product requirements) and the Development team (who execute the work) collaborate to complete a predefined set of tasks. Scrum also involves key aspects like **Daily Stand-ups, Product Backlog, Sprint planning, and Sprint retrospectives to facilitate communication, planning, and continuous improvement.** Whereas other software modelling method Scrum provides iteration for each feature or functionality to be implemented and tested to see if it satisfies the user expectations before diving into the next step as compared to the other methods that implement the whole application in one iteration before presenting to the product owner.

#### **UP**

Stands for “Unified Process” it is sometimes associated with the **Rational Unified Process (RUP)** due to a particular commercial variant, is an object-oriented and use-case-driven approach to software development. These iterative and incremental framework structure the development phase into four sequential phases: **Inception (**initial planning and requirements gathering), **Elaboration (**refining the architecture and resolving high-risk elements), **Construction (**developing the bulk of the software), and **Transition (**deploying the solution to users). Throughout, UP employs the **Unified Modeling Language (UML)** as a standard notation to visualize and design system architecture. With its disciplined approach to assignments and roles, UP integrates various best practices from software engineering.

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

DSDM is one of the earliest Agile methodologies and is part of the Agile Alliance. Introduced in the 1990s, DSDM emphasizes collaboration and flexibility to meet business objectives. Rooted around principles like focusing on the business need, delivering on time, and collaborating effectively. The methodology categorizes requirements as either “must have”, “should have”, “could have", or "won’t have”, ensuring that critical functionalities are always prioritized. DSDM’s iterative approach is accompanied by regular reviews and feedback sessions, ensuring alignment with user requirements and promoting continuous improvement.

### PRESENTATION OF THE ANALYSIS METHOD

#### **Presentation of UML Language**

UML, or the Unified Modeling Language is a standardized, general-purpose modeling language in the field of software engineering and system design. Introduced in the 1990s by **Grady Booch, Ivar Jacobson, and James Rumbaugh**, UML was developed to provide a consistent way to visualize, design, and document models of software systems. Over the years, it has evolved to become an essential tool not only for software architects and developers but also for business analysts, system engineers, and stakeholders. And the key features and aspects of UML is that it provides a comprehensive set of diagrams divided into two broad categories: **Structural Diagrams and Behavioral Diagrams.**

1. **Structural Diagrams**

These represent the static structure of a system. Examples include:

* Class Diagrams;
* Object Diagrams;
* Component Diagrams;
* Deployment Diagrams;
* Package Diagrams;
* Composite Structure Diagrams;

1. **Behavioral Diagrams**

These capture the dynamic behavior of a system. Examples include:

* Use Case Diagrams;
* Sequence Diagrams;
* Activity Diagrams;
* State Machine Diagrams;
* Communication Diagrams;
* Interaction Overview Diagrams;
* Timing Diagrams;

Important to note that UML is not a method but a modeling language. However, while UML provides the language and notation, it does not prescribe a specific process or method for software development. Instead, UML can be used in conjunction with various software development methodologies and processes, such as Rational Unified Process (RUP) , Agile methodologies, or the waterfall model. But our modeling approach will be 2TUP (Two-track Unified Process).

#### **Comparison study between UML and MERISE**

Table 8: Comparison between UML and MERISE

|  |  |  |
| --- | --- | --- |
| In terms of | UML | MERISE |
| Origin | UML is an international standard adopted by the Object Management Group (OMG). It was developed in the mid-1990s by integrating various earlier modeling methods and notations | MERISE is a French methodology developed in the early 1980s. It has been predominantly used in French speaking countries |
| Focus | It is primarily object-oriented, meaning it emphasizes objects, their behaviors, and their interaction. UML is versatile and can be used across different software development methodologies and is not tied to a specific process | MERISE is more data centric. It emphasizes the clear separation between data (what the system knows) and processes (what the system does). This methodology is especially known for its structured approach to database design |
| Methodology | UML offers a set of diagrams to model various aspects of systems, both structural (e.g., class diagrams, object diagrams) and behavioral (e.g., sequence diagrams, activity diagrams). | MERISE uses a phased approach to system development with specific models guiding the process from conceptual design to physical implementation, like the conceptual Data Model (CDM), Physical Data Model (PDM) |
| Scope | UML is a generalized modeling language suitable for various systems, not just software. Its scope spans from system architecture to detailed design | MERISE can be applied to different systems, its primary strength lies in information system modeling and database design. |
| Adoption and Usage | Widely adopted internationally, UML has become a global standard in software engineering for modeling and design. Most modern software development tools provide support for UML. | Merise is largely within French-speaking regions. However, its principles and techniques are foundational and can be recognized in other methodologies. |

#### **The Two Track Unified Process (2TUP)**

In the world of software development, both the process model and the tools used to capture and communicate requirements play pivotal roles in ensuring project success. The 2TUP (Two-Track Unified Process) model represents one such strategic approach to software development, delineating the journey from conceptualization to realization. Parallel to this, the Unified Modeling Language (UML) serves as a standardized visual language, adept at articulating the intricate nuances of software systems. When the structured, Y-shaped flow of the 2TUP meets the expressive, visual paradigms of UML, it results in a powerful synergy. This union ensures that both the ‘what’ (**Functional Branch**) and the ‘how’ (**Technical Branch**) of software development are seamlessly transitioned into the ‘actualization’ (**Realization branch**) and are captured with precision, clarity and cohesion. By intertwining the 2TUP model with UML diagrams, teams can better visualize, communicate, and implement software requirements and design.

**Capture of business requirements**

**Analysis**

**Capture of technical needs**

**Generic Design**

**Preliminary Design**

**Detailed Designed**

**Coding and Testing**

**Recipe**

**Prototype**

**FUNCTIONAL BRANCH (TRACK)**

**TECHNICAL BRANCH (TRACK)**

**REALISATION BRANCH**

Figure 6 Two-Track Unified Process

1. **Functional Branch**

This branch of 2TUP model prioritizes understanding business objectives. Activities begin with **Capturing business requirements**, involving stakeholder interviews and user studies to document essential needs. This is followed by **requirement analysis**, where these needs are validated, refined and modeled into structured specifications. Techniques like **use case diagrams** might visualize complex business interactions. The outcomes include a business requirement document capturing strategic needs, and a functional Specification Document detailing specific functionalities. This branch ensures the software is aligned with business goals and user needs.

1. **Technical Branch**

This branch delves into the ‘how’ of the software functionality. It starts with the **capture of technical needs**, identifying system constraints, performance requirements, scalability needs, and non-functional necessities. This phase ensures the software’s robustness, efficiency, and security. Following this, **generic design (or prototype)** development takes center stage. Designers create high-level architecture diagrams and preliminary interface mock-ups to offer a visual representation of the software’s structural and user-interface layout. This prototype serves as a tangible blueprint enabling stakeholders to visualize the end product, and developers to understand the foundational structure they will build upon.

1. **Realization Branch**

Finally, this branch involves bringing abstract concepts to reality. Starting with **Preliminary Design**, the system’s initial architectural framework is outlined, detailing broad modules and interactions. This foundation is further refined in **Detailed Design**, where every system component gets a granular blueprint, encompassing algorithms, data structures, and interactions subsequently, the **Coding** phase sees developers meticulously crafting the software, adhering to these blueprints. Parallelly, **Testing** ensures the software’s robustness, functionality, and quality by identifying and rectifying discrepancies. Culminating the branch, **Recipe (or Deployment)** involves prepping the software for its final user environment, which includes setting up necessary infrastructure, final validation in real-world conditions, and post- launch monitoring.

#### **Justification and Motivation for the chosen Approach**

The reasons for our choice of UML modeling language and 2TUP method is being justify due to the following reasons:

Justification:

* 2TUP’s structured approach when mapped with UML diagrams, provides a visual overview of the software’s functional needs, technical designs, and realization paths.
* UML, being a standard modeling language, ensures that everybody that takes part in the project has a common understanding and helps minimize misinterpretations.
* 2TUP’s iterative nature combined with UML’s ability to easily update and modify diagrams makes it easier to adapt to changes and refine requirements or design over multiple cycles.
* From capturing business requirements (use case diagrams) to defining system interactions (sequence diagrams) and deployment considerations (deployment diagrams), UML offers tools for every phase of the 2TUP model.

Motivation:

* Visualization often leads to better comprehension. By representing ideas using UML diagrams, teams can achieve a clearer requirement and design of the system.
* The approach foster collaboration by using UML diagram we can bridge the gap between stakeholder and development team to have the same visual language of the system.
* Early visualization and iterative development allow potential issue to be solved as soon as possible reducing project risks.

In essence, integrating UML with the 2TUP model brings together the best of strategy and visualization enhancing clarity, collaboration, and efficiency throughout the software development lifecycle.

### MODELING WITH UML AND 2TUP STRUCTURE

FUNCTIONAL BRANCH

**CAPTURE OF BUSINESS REQUIREMENTS and Analysis**

In this phase we capture the objectives of the system, vision, expected outcomes and understanding user requirements or functionalities. This can be represented in UML using the following diagrams:

* Use Case Diagrams
* Activity Diagrams

#### **USE CASE DIAGRAMS**

DEFINITION:

A use case diagram is a visual representation of the interactions between actors (actors or external systems) and a system under consideration. It depicts the functionality or behavior of a system from the user’s perspective.

Formalism

A diagram of a system

Description automatically generated

Figure 7: Formalism of Use Case Diagram

Explanation:

|  |  |  |
| --- | --- | --- |
| Elements | Diagrammatic representation | Description |
| Actor |  | This refers to someone who interacts with the functionalities of the system. It is similar to the concept of a user of a system, but a user can play different roles for example: a prof can be an instructor and a researcher playing 2 roles. Actors trigger event in the system. |
| Use case |  | A use case represents a functionality or features of the system that interact with actor of the system. |
| Communication Link |  | The participation of an actor and a use case is shown by connecting an actor to a use case and this is done with using a solid link to represent it |
| Boundary of System |  | This is the overall system having in its all the functionalities (use case) inside the system and their interaction |
| Extends |  | This is a use case relationship that is used when child use case may not necessarily been called when the event (base use case) is triggered but it can be called based on specific conditions done by the actors. Example as we can see on the image the base use case is login account and the child use case is invalid and the child use case in an extension relationship always point to the base use case with a label “Extend” |
| include |  | The include relationship is a relationship where the parent use case depend on another use case in order to continue its process |
| generalization |  | This refers to a parent-child use cases relationship and the child case is an enhancement of the parent use case is shown as directed arrow with a triangle arrowhead with the child use case connected at the base of the arrow |
| Association |  | An association between an actor and a use case somehow interact with each other |
| Inheritance |  | Relates a generalized and a specialized element. The generalized element is at the arrowhead end. A specialized use case inherits the goals and actors of its generalization |

1. **List of actors of our system**

* **User**: he is the one that uses the system or interacts with the features and functionalities of the system. They can have two main role
* Signal a lost card
* Adding detail of a found card
* **Database**: This is more of a system actor that stores, retrieves, and matches card details also identification of the user
* **Notification System**: to notify the user when his card is found.

##### **GENERAL USE CASE DIAGRAM OF OUR SYSTEM**

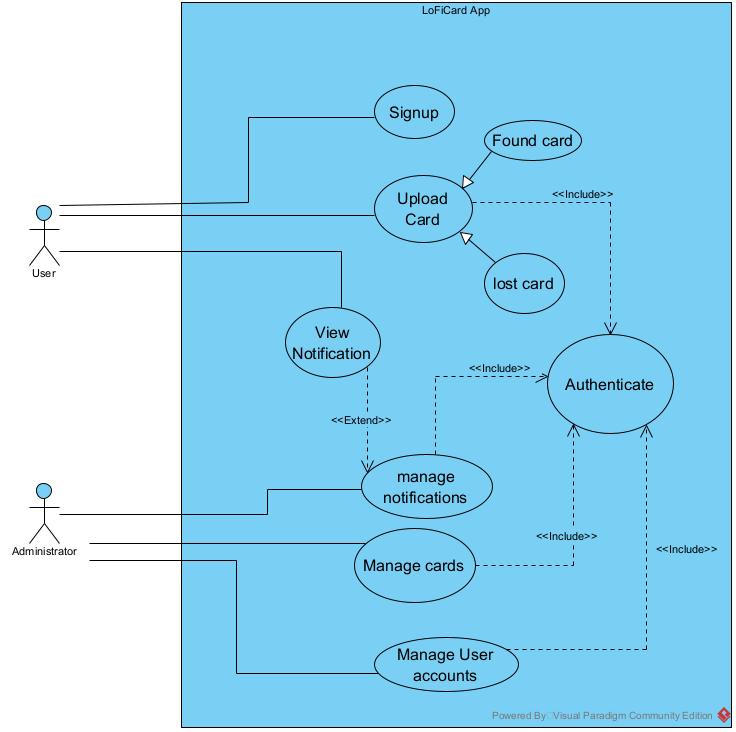


Figure 8: General Use Case Diagram of our system

##### **USE CASE DIAGRAM OF MANAGE CARDS**

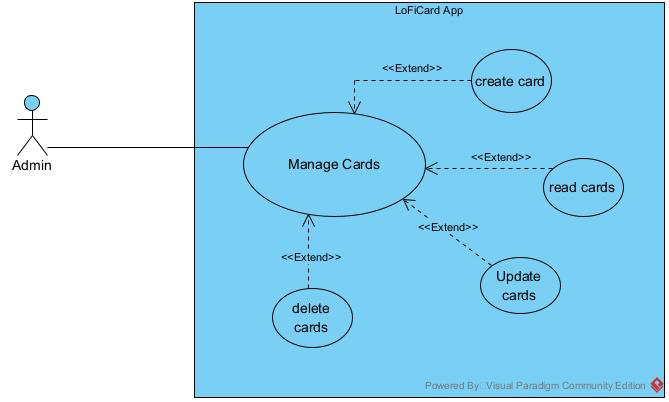


Figure 9: Use case diagram of Manage card

##### **USE CASE DIAGRAM OF UPLOAD CARD**

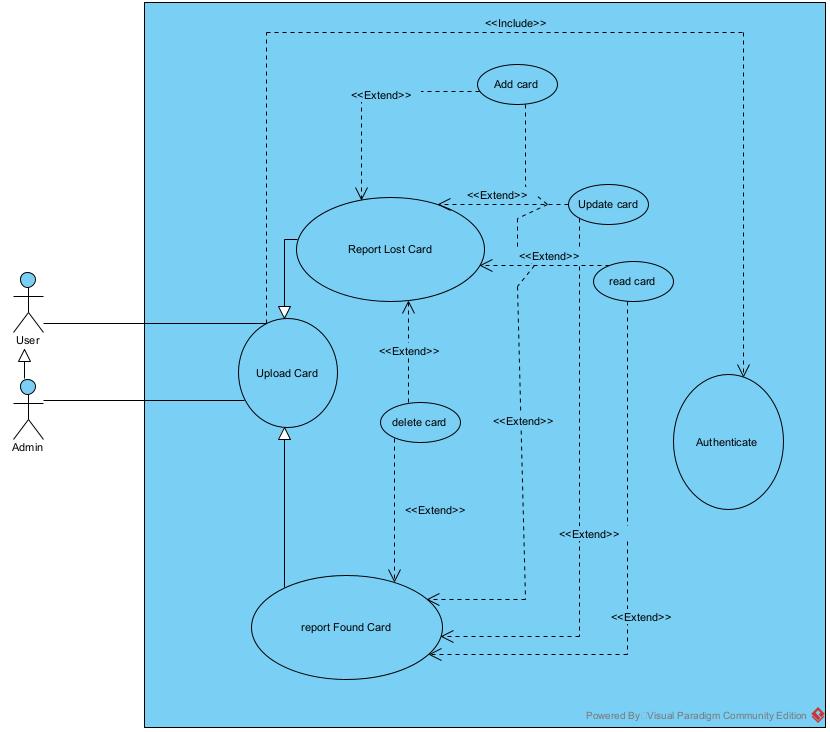


Figure 10: Use case diagram of Upload card.

##### **USE CASE DIAGRAM OF AUTHENTICATION IN THE SYSTEM**

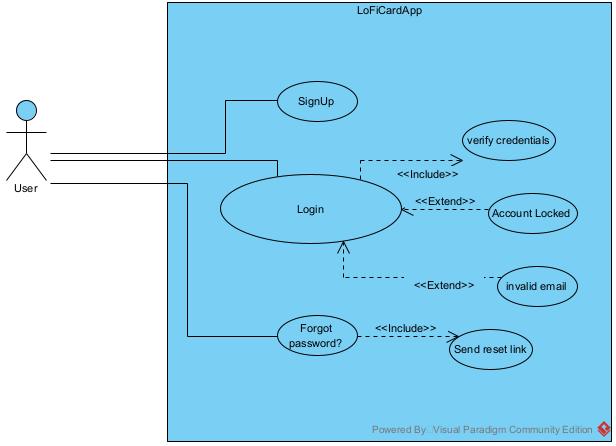


Figure 11 use case diagram of Authenticate.

TEXTUAL DESCRIPTION

The textual description of a use case provides detailed written information about a specific use case. This description offers a deeper understanding of the system’s functionality by outlining how different actors interact with the system and under what condition.

FORMALISM

Table 9: formalism of textual description of a use case

|  |  |
| --- | --- |
| USE CASE NAME | |
| ACTORS | List the primary actor (who initiates the use case) and secondary actors (other participants or systems that might be involved). |
| Description | A brief summary of the use case’s main objective |
| Preconditions | Any conditions that must be true before the use case can begin |
| Postconditions | The state of the system once the use case has been successfully completed |
| Nominal scenario | A step-by-step description of the typical sequence of events when everything goes as expected. |
| Alternative scenario | Descriptions of variations from the nominal scenario due to exceptions or specific conditions. These might include errors, alternative decisions made by the actor, or any other variations from the main path. |
| Exceptions | Specific conditions or errors that might prevent the use case from achieving its goal, along with how the system responds to these exceptions. |
| Special requirements | This refers to any non-functional requirements or special conditions that are relevant to this use case (e.g., performance constraint, security considerations). |

Table 10: textual description of use case Report Lost Card

|  |  |
| --- | --- |
| USE CASE OF REPORT LOST CARD | |
| ACTORS | * **Primary Actor**: User (Person who lost their card) |
| description | This use case allows a user to report his/her lost id card. Which involve signaling, the system logs the details and check if there is a matching with found cards in the database. |
| Preconditions | * User is signup with the app. * User has access to the internet and the app’s platform * User is authenticate in the app |
| Postconditions | * Lost card details are stored in the database. * System checks for a match with found id cards and if a match is found, triggers a notification to the user. |
| Nominal scenario | 1. User authenticated 2. User selects the option to report a lost card. 3. User inputs necessary details about the lost card. 4. System verifies the details and logs the cards as “lost” in the database. 5. System initiates a check for matching found cards. |
| Alternative scenario | 4.a) User inputs incomplete or ambiguous details:   * System prompts the user to provide complete and clear details. |
| Exceptions | 1. Database connection failure:  * System displays an error message and prompts the user to try again later. |
| Special requirements | * Secure encryption of user data to ensure privacy. * Real-time database check for quick matching |

Table 11 textual description of use case Found Card with Image

|  |  |
| --- | --- |
| USE CASE OF REPORT FOUND CARD WITH IMAGE | |
| description | This use case enables a user to input detail of a card they’ve found and upload an image of the card. The system logs the information and checks if anyone has previously reported the card as lost |
| Preconditions | * User has the app installed and an active internet connection. * User is authenticated * User has access to details of the found card and can take a clear image of the card |
| Postconditions | * Found card details, along with the image, are saved in the database. * System checks for a matching lost card and, and if a match is identified, triggers a notification to the original owner |
| Nominal scenario | 1. User logs into the app. 2. User selects the option to report a found card. 3. User provides necessary details about the found card. 4. User upload/takes a clear image of the found card. 5. Systems verifies the details, stores the image, and logs the cards as “found” in the database. |
| Alternative scenario | 4.a) user uploads an unclear or unrelated image:   * System prompts the user to provide a clear image of the found card.   5.a) User inputs incomplete or ambiguous details:   * System prompts the user to provide complete and clear details. |
| Exceptions | 1. Database connection failure:  * System displays an error message and prompts the user to try again later.  1. Image upload error or large file size:  * System alerts the user about the upload issue and suggests trying a different image or reducing the file size. |
| Special requirements | * Prompts user verification to minimize false entries. * Secure handling and encryption of card data and images to maintain privacy. * Image compression and optimization for efficient storage and fast retrieval. |

|  |  |
| --- | --- |
| USE CASE OF SIGNUP | |
| Actors | * **Primary Actor**: user (person looking to create an account in the app * **Secondary Actor**: Database (System component that stores and retrieves user details) |
| Description | This use case facilitates a new user in creating an account, providing their essential details to the system |
| Preconditions | * User has app install and an active internet connection. * User does not have an existing account with the app |
| Postconditions | * User details are saved in the database. * User has a new active account and can access app features. |
| Nominal scenario | 1. User launches the app. 2. User selects the option to “signup” or “create account.” 3. User provides necessary details (name, password, email, phone number). 4. Upon successful verification, the system logs the user details into the database and confirms account creation. 5. User is directed to the login interface |
| Alternative scenario | 3.a) if the user provides an already registered username or email:   * System prompts the user to choose a different username or log in if they already have an account.   6.a) if the user provides a weak or non-compliant password:   * System suggests creating a stronger or compliant password based on set criteria. |
| Exceptions | 1. Database connection failure:  * System displays an error message and prompts the user to try again later.  1. Verification code/link expiry or inaccuracy:  * System offers an option to resend the verification code/link |
| Special requirements | * Secure encryption of user passwords to ensure data security. * Possible integration of CAPTCHA or similar systems to prevent bots from registering. * Option for email verification to confirm the user’s email address |

#### **ACTIVITY DIAGRAMS**

Activity diagrams represent workflows or processes, showcasing the sequence of activities and the conditions that dictate the flow from one activity to the next.

Formalism:

Table 12: Formalism of an activity diagram

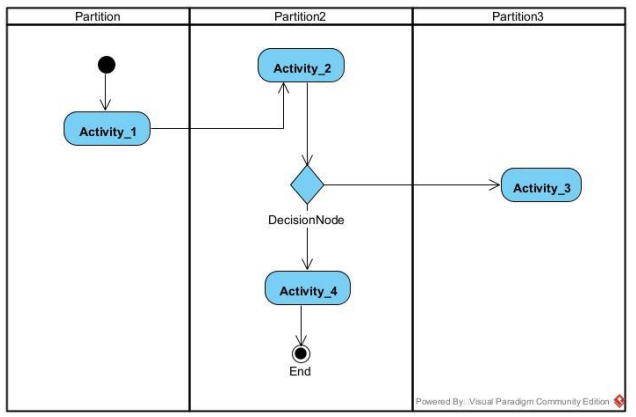


Table 13: Components of an activity Diagram

|  |  |  |
| --- | --- | --- |
| ELEMENTS | NOTATION | DESCRIPTION |
| Initial node |  | Portrays the beginning of a set of actions or activities |
| Activity |  | Is used to represent a set of actions |
| Action |  | A set of actions to be performed. |
| Control flow |  | Show the sequence of execution. |
| Object flow |  | Show the flow of an object from one activity to another activity |
| Object node |  | Represent an object that is connected to a set of Object flows |
| Decision node |  | Represent a test condition to ensure that the control flow or object flow only goes down one path |
| Merge node |  | Bring back together different decision path that were create using decision nodes |
| Fork node |  | Split behavior into a set of parallel or concurrent flow of activities (or actions) |
| Join node |  | Bring back together a set of concurrent or parallel flow of activities |
| partition |  | A way to group activities perform by the same actor on an activity diagram or to group activities in a single thread |
| Activity final node |  | Stop all control flow or object flow in an activity or action |

##### ACTIVITY DIAGRAM OF SIGNUP

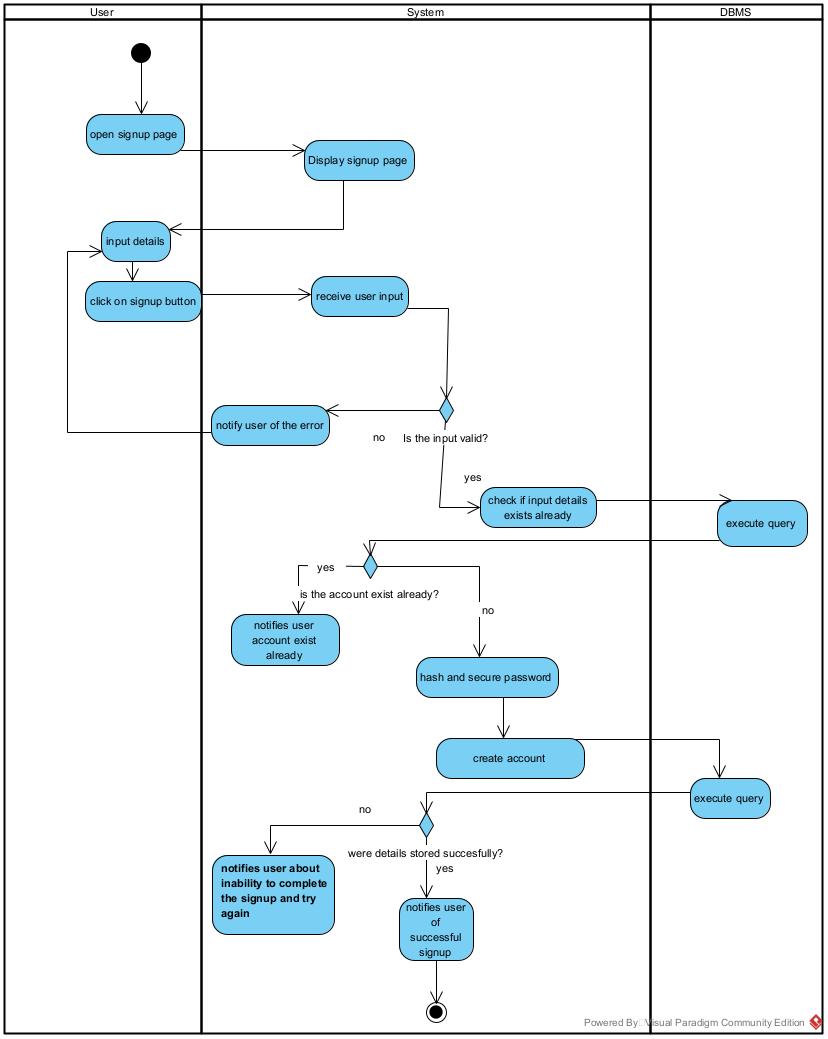


Figure 12: Activity Diagram of Signup

##### ACTIVITY DIAGRAM OF REPORT LOST CARD

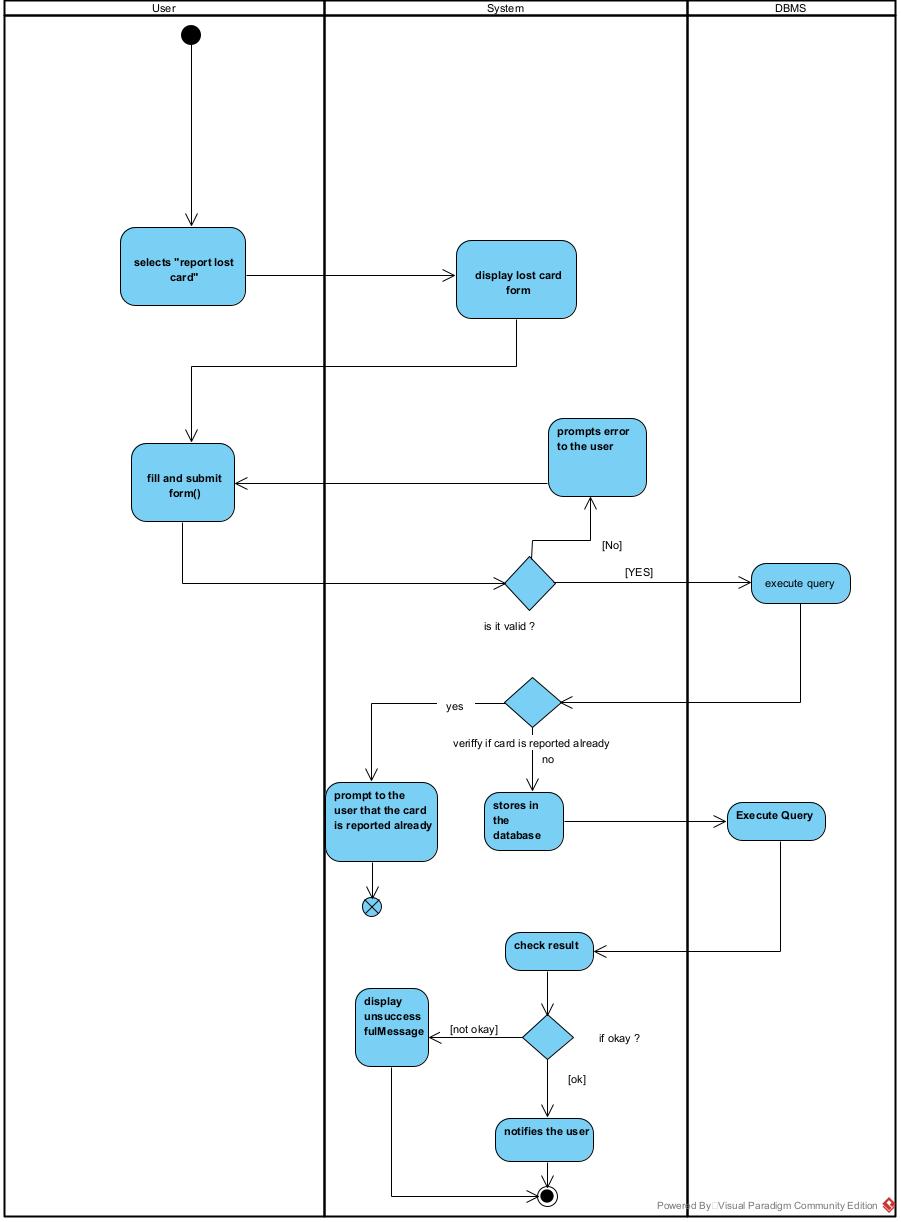


Figure 13:Activity diagram of report lost card

##### ACTIVITY DIAGRAM OF MATCH CARD

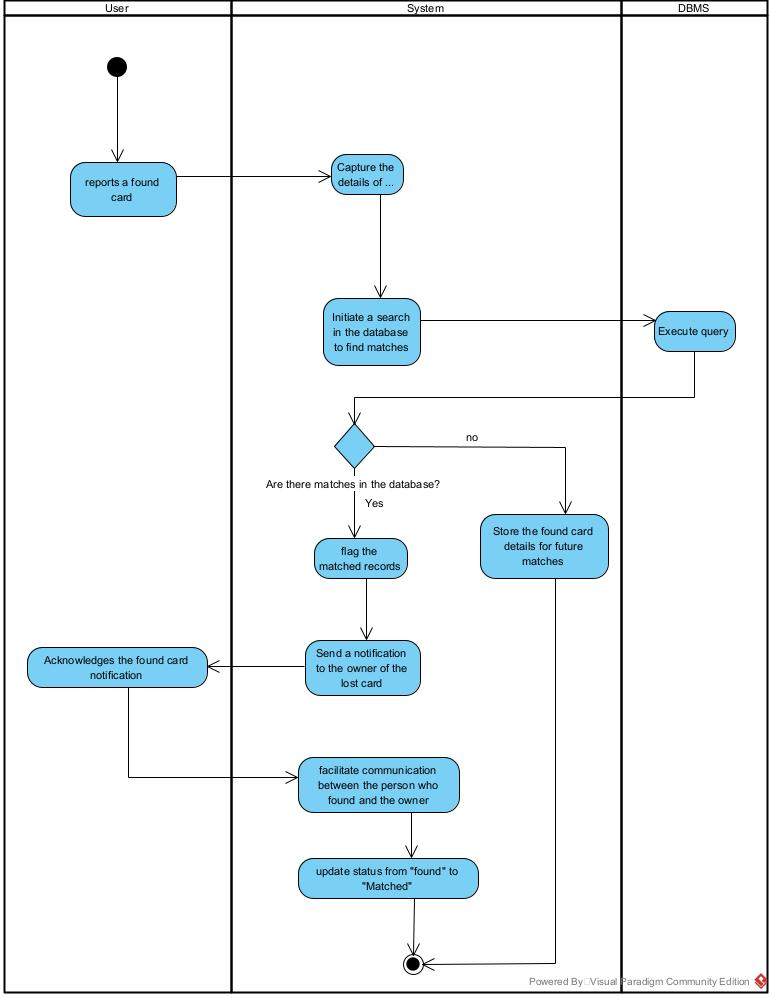


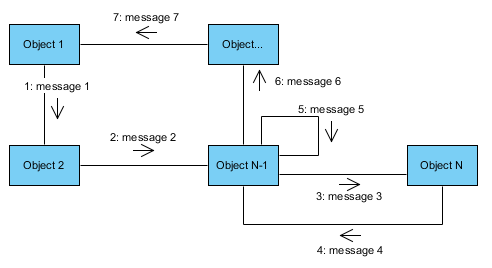
Figure 14: Activity diagram of match Card

#### **COMMUNICATION DIAGRAMS**

Definition:

A communication diagram, in the context of UML, showcases the interactions between objects or parts. It emphasizes the relationships (or links) between these objects and represents the flow of messages exchanged in a sequence. While similar to a sequence diagram, a communication diagram emphasizes the organization and links between objects rather than the time sequence of interactions.

Formalism:



Components of a communication diagram

Table 14: formalism of Communication Diagram

|  |  |  |
| --- | --- | --- |
| Element | Notation | Description |
| Frame | A white rectangular object with black text  Description automatically generated | Could be shown within a rectangular frame with the name in a compartment in the upper left conner. |
| Object |  | Is a participant in the communication process that can send or receive messages. It can be labeled with its name and role. For example, this is an object named :user |
| Message |  | Is an arrow from one object to another that shows the direction and order of the message flow. It can be labeled with a number and a name |
| Link |  | Is a straight line that connects two objects and indicates a relationship between them. It means that the object can communicate with each other |

##### COMMUNCATION DIAGRAM OF LOST CARD



Figure 15: Communication diagram of Lost card

##### COMMUNCATION DIAGRAM OF FOUND CARD



Figure 16: Communication diagram of found card.

##### COMMUNCATION DIAGRAM OF SIGNUP



Figure 17: Communication Diagram of Register

#### **SEQUENCE DIAGRAM**

**Definition**

A sequence diagram is a visual representation of how different parts of a system interact over time. It showcases the sequence of events, detailing which components communicate Dwith another and in what order. Using vertical lines to represent lifetimes and horizontal arrows for interactions, it offers a chronological view of processes, making it essential for understanding system functionality and flow.

**Formalism**

**A diagram of a diagram of a lifeline

Description automatically generated**

Figure 18: formalism of Sequence diagram

|  |  |  |
| --- | --- | --- |
| ELEMENTS | NOTATION | DESCRIPTIONS |
| Actor |  | **Represents an external entity, often a user, that interacts with the system. Typically illustrated as a stick figure.** |
| Lifeline |  | **The vertical dashed line extending from an object or actor, representing the existence of that entity over time** |
| Message |  | **Depicts the communication or interaction between objects or actors. It’s shown as horizontal arrow with the arrow’s direction indicating the message path.** |
| Return Message |  | **An arrow depicting the response sent back to the originating object or actor after a message. Usually represented as a dashed arrow** |
| Activation Bar |  | **A thin rectangle drawn on the lifeline, representing the period an object is active or executing something in response to a message.** |
| Destroy |  | **A large X on a lifeline, indicating the point at which an object is destroyed or ceases to exist.** |
| Combined Fragment |  | **Used to represent conditions, loops or other control structures. Often a rectangle that encompasses a portion of the lifelines and contains labels like ‘alt’ for alternatives or ‘loop’ for repetitive sequences.** |
| Note |  | **A comment or annotation that can be attached to any part of the diagram to provide additional information or clarification.** |

##### **SEQUENCE DIAGRAM FOR REPORTING A LOST CARD**

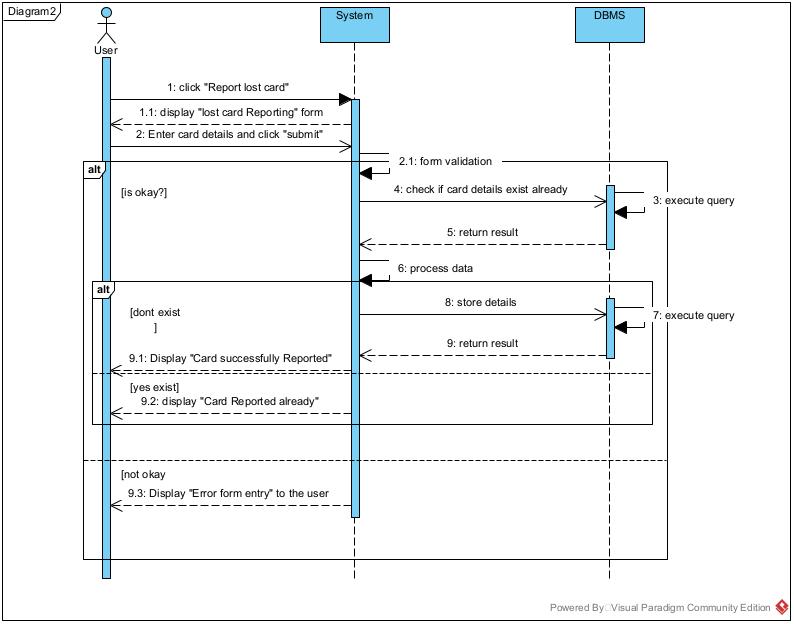


Figure 19: reporting lost card Sequence Diagram

##### **SEQUENCE DIAGRAM OF SIGNUP**

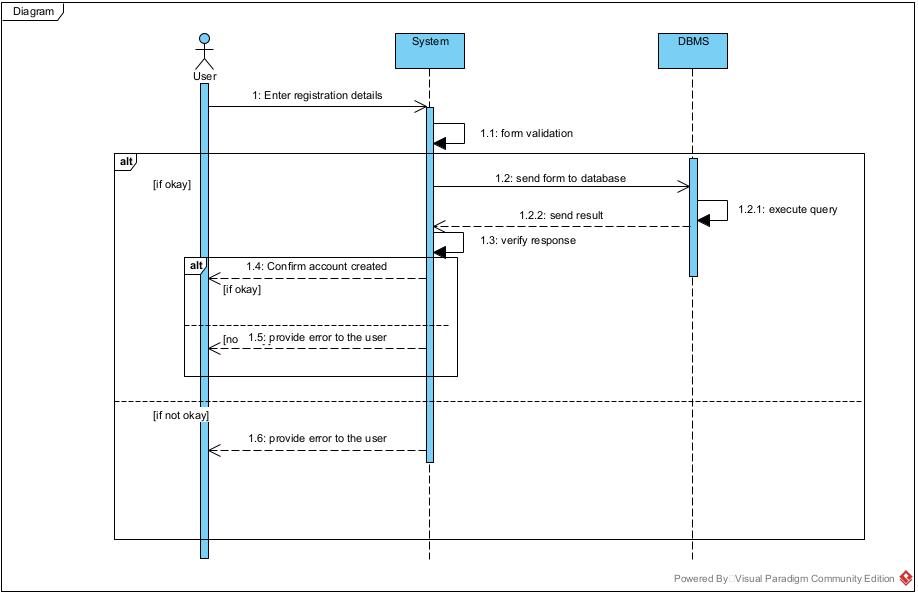


Figure 20 SignUp Sequence Diagram

## CONCLUSION

As we conclude the analysis phase, we’ve mapped out the core functions and user interactions of our LoFiCard app using some UML diagrams. This foundational blueprint, rooted in comprehensive examination, will now guide our transition into the conception phase, ensuring a user-centric design and seamless development process. Our groundwork promises a successful evolution of the project.

PART V

# CONCEPTION PHASE

**PREAMBLE**

The conception phase, often termed as the design phase is pivotal in transitioning from abstract requirement to tangible blueprints. This stage meticulously outlines the system’s architecture, database design, user interfaces, and other critical components, ensuring a cohesive roadmap for developers. Emphasizing user experience, security, and scalability, the conception phase is the keystone that underpins successful software development, ensuring the envisioned product aligns seamlessly with user needs and technical feasibility.

**CONTENT**

INTRODUCTION

1. TECHNICAL BRANCH
   1. Generic Design
2. IMPLEMENTATION BRANCH

A. Preliminary Design

B. Detailed Design

CONCLUSION

## INTRODUCTION

The conception phase is a critical bridge in software development, transitioning from abstract requirements to concrete design blueprints. Utilizing the 2TUP methodology, this phase is segmented into two main branches: technical and implementation. The former sketches a high-level system overview, while the latter dives deep into precise details. Together, they form a comprehensive roadmap, setting the foundation for seamless development.

### TECHNICAL BRANCH

#### **Generic Design**

Generic Design is a component of the technical branch in the 2TUP methodology, aiming to lay out a high-level and abstract representation of the software system. It avoids delving into minutes implementation details, and instead focuses on defining the overall structure and architecture.

##### **Physical architecture**

The physical architecture of an application refers to the physical components and the interactions between them. This typically involves servers, databases, and client devices, describing how these components are arranged and interconnected.

Our application follows the 3-tier architecture and in this architecture, the physical components can be represented as follows:

1. Presentation Tier: This is where the user interacts with the application. It can be a mobile app, a web browser, or any other interface that presents data to the user. And in my app context the front-end of my app is developed with React Native (cross-platform) and provides a user-friendly interface.
2. Application Tier: This tier contains the logic of the application. It processes the data between the presentation and the data tier, handling business rules, computations and transformations and in my app is the Django Backend Server who is responsible for it.
3. Data Tier: This consists of databases and file storage systems where the data is stored and retrieved. In my app what is used here is PostgreSQL database.

##### **Logical Architecture**

The logical architecture refers to the internal software design of the system, the patterns, and practices used in the code and application structure. So in the context of my application the pattern use is MVT (MVC pattern but in Django framework) which will be explain next in the upcoming lines.

1. Model (M in MVC/MVT): it is part of the data tier in the physical architecture, it represents the data structures and interactions with the database. In Django, this is usually the ‘models.py’ file where you define the data schema and the relationships.
2. View (V in MVC): it is part of the application tier in the physical architecture, it is concerned with displaying the data to the user. In Django’s MVT pattern, this is a combination of what Django calls ‘views’ and ‘templates. The view handles the logic of which data is presented, and the template determines how this data looks.
3. Controller (C in MVC): it is part of the application tier in the physical architecture. It manages user input and updates the Model and View accordingly. In Django, this is mostly done behind the scenes, hence why Django’s architecture is sometimes to refer to as MVT (Model View Templates) with Django itself managing the controller aspect.

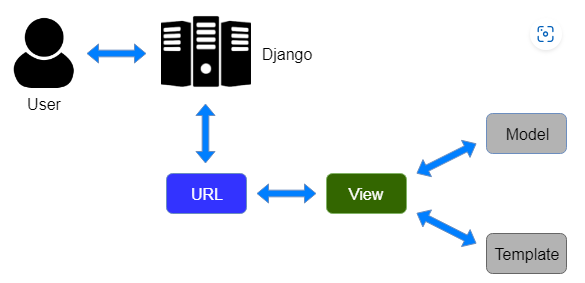


Figure 21: MVT pattern in Django

### IMPLEMENTAION BRANCH

#### **PRELIMINARY DESIGN**

Here, the design is still high-level but more concrete than in the technical branch.

### PACKAGE DIAGRAM

Definition:

A Package Diagram in the Unified Modeling Language (UML) illustrates how different components or classes of a system are grouped (packaged) together and how they depend on each other. It is a mechanism to organize related elements into grouped namespaces, providing a high-level modular view of a system’s architecture. This view aids in identifying shared components, reducing redundancies, and clarifying system structure.

Component of a package diagram

|  |  |  |
| --- | --- | --- |
| Elements | Notation | Description |
| Package |  | A namespace used to group together elements that are semantically related as a file folder or a tabbed folder. Packages can contain other packages, creating a nested structure. |
| Dependency |  | This is a relationship in which one package depends on another. It’s represented by dashed arrow pointing towards the depended-upon package. This means that if something changes in the package being pointed to, it package being pointed to, it might affect the package that’s doing the pointing |
| Package import |  | Indicates that all the publicly visible members of a package are accessible in another package. |
| Generalization |  | Represents inheritance between packages, allowing one package to inherit the behavior of another. |
| Stereotypes |  | These are labels or tags you can add to give more semantic meaning to a package. Common stereotypes might include <<utility>>, <<system>>, or <<subsystem>> |

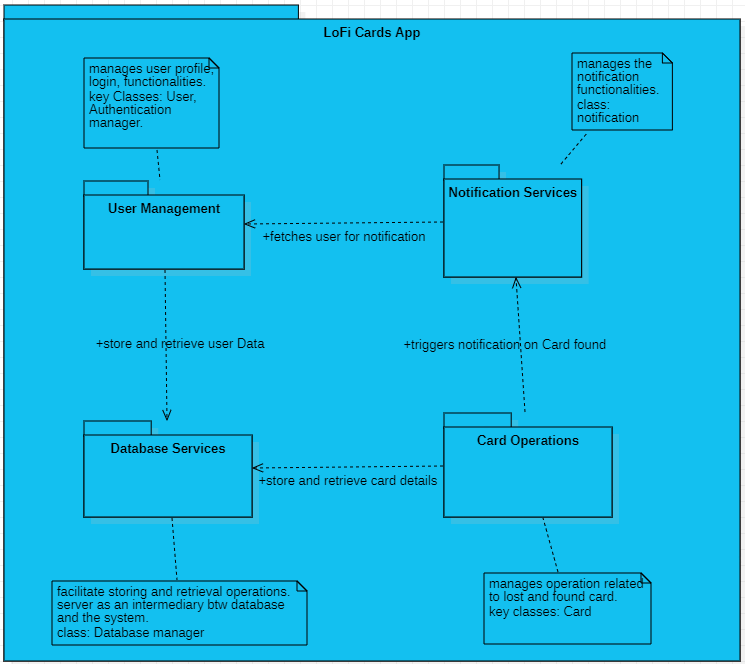


Figure 22: package diagram of the system

### STATE MACHINE DIAGRAM

Definition :

A state machine diagram is a visual guide that displays how a system behaves. It pinpoints the various “situation” or “states” a system can find itself in and explains the shifts from one state to another based on triggers. This diagram was crafted to provide clarity in understanding complex systems. By mapping out the system’s behaviors, designers, developers, and users can predict its reactions, making it easier to design, troubleshoot, and interact with the system effectively.

Formalism:

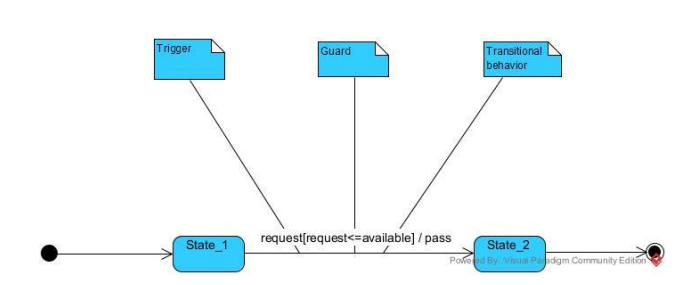


Figure 23 formalism of state machine diagram

|  |  |  |
| --- | --- | --- |
| ELEMENT | NOTATION | DESCRIPTION |
| Initial State |  | It defines the initial state (beginning) of a system, and it is represented by a black filled circle |
| State-box | A blue rectangle with black text  Description automatically generated | It depicts the conditions or circumstances of a particular object of a class at a specific point of time. A rectangle with round corners is used to represent the state box |
| Decision-box | A blue diamond with black outline  Description automatically generated | It is of diamond shape that represents the decisions to be made based on an evaluated guard. |
| Transition |  | A change of control from one state to another due to the occurrence of some event is termed as a transition. It is represented by an arrow labeled with an event due to which the change has ensued. |
| Final state |  | It represents the final state (end) of a system. It is denoted by a filled circle present within a circle |

##### STATE MACHINE DIAGRAM FOR A USER’S CARD



Figure 24: state machine diagram <<User's Card>>

##### STATE MACHINE DIAGRAM FOR A FINDER’S ENTRY



Figure 25: State machine diagram <<Finder's Entry>>

### DETAILED DESIGN

### CLASS DIAGRAM

Definition:

A class diagram is a type of static structure diagram in the Unified Modeling Language (UML), which represents the structured patterns of a system. It illustrates how classes are interconnected and how they collaborate with each other. Essentially, it offers a blueprint of the system.

Components of a Class Diagram

Table 15: Component of a class Diagram

|  |  |  |
| --- | --- | --- |
| Elements | Notation | Description |
| Class | A blue box with black text  Description automatically generated | Symbolizes a rectangle divided into three sections: Name at the top, Attributes in the middle, Operations/Methods at the bottom. Represent entities or things and their features. |
| Association |  | Represented by a solid line between two classes, indicating that there’s a relationship between the classes. |
| Multiplicity |  | Represent how many instances of one class can be associated with a single instance of another class. Expressed as a cardinal numbers (1,2) or ranges(0..1, 1..\*) at the end of associations |
| Generalization (Inheritance) | A diagram of a diagram  Description automatically generated | Represented by a hollow triangle pointion to the parent class. Indicates an “is-a” relationship between the child class (subclass) and the parent class (superclass). |
| Aggregation | A diagram of a diagram  Description automatically generated | Represented by a hollow diamond on the side of the whole. It is a “whole-part” relationship, where the part can exist from the whole |
| Composition | A diagram of a diagram  Description automatically generated | Represented by a filled diamond on the side of the whole. It’s a stronger form of aggregation, indicating that the part cannot exist separately from the whole. |
| Dependency | A diagram of a connection between two squares  Description automatically generated | Represented by a dashed line with an arrow. Indicates that one class depends on another. As we can see on the diagram at the left class1 depends on class2. |
| visibility |  | Indicators (+,-,#,~) for public, private, protected, and package-level visibility respectively |

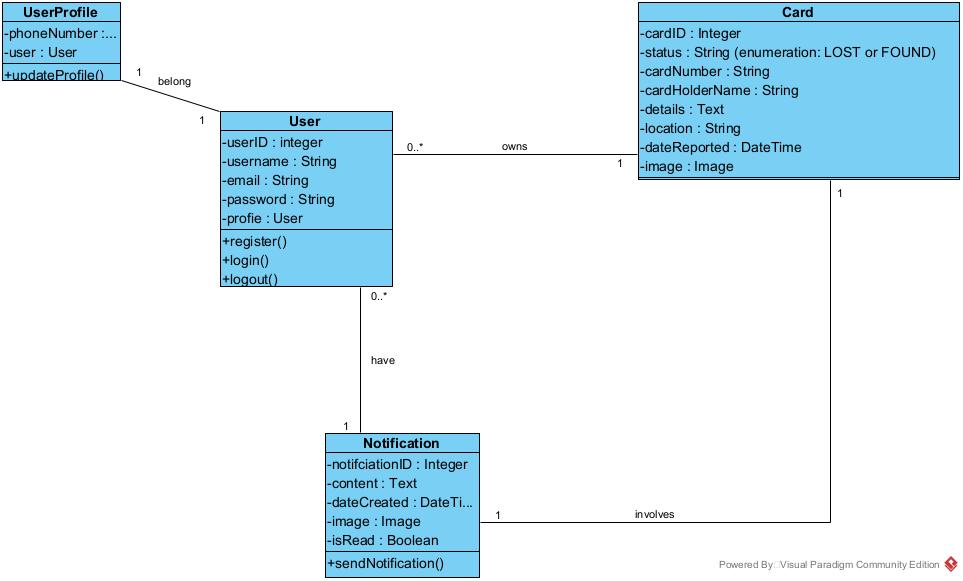


Figure 26: Class Diagram for our system

## CONCLUSION

In the conception phase, we have designed the architecture and structural blueprint of our application, ensuring a seamless fusion of user experience with backend functionalities. Leveraging the 2TUP model, our design incorporated both the technical and implementation branches, utilizing UML diagrams to provide clarity. Through class, state machine and package diagrams, we have achieved a comprehensive understanding of the system’s behavior and their state transitions. With this robust design framework in place, we are now poised to transition into the realization phase where these conceptual designs will be transformed into tangible, working modules, setting the stage for the deployment of our innovative card management application.

PART VI

# REALIZATION PHASE

**PREAMBLE**

In the dynamic journey of software development, every step we take builds upon the last, each one vital in shaping the final product. Having navigated to the intricate maze of conceptualization and design, we now find ourselves on the threshold of the realization phase. It’s in this phase that our concepts crystallize, transforming into tangible components and systems. We begin to see the intricate dance of software modules and their symphony with hardware laying the foundation for a seamless user experience. The path ahead will illuminate the essence of our application, breaking it down, piece by piece, and showcasing how each element fits into the grand mosaic of our digital solution.

**CONTENT**

INTRODUCTION

1. COMPONENT DIAGRAM
2. DEPLOYMENT DIAGRAM
3. PRESENTATION OF THE TOOL USED
4. PRESENTATION OF THE LANGUAGES USED

CONCLUSION

## INTRODUCTION

In this phase, we are focusing on two major tools: our Component Diagram and Deployment Diagram. Thinking of the component Diagram as our application family tree, showing every part and how they are linked. Meanwhile the Deployment Diagram is like a game plan for launch day, show casing where everything will sit and how it will all connect. By the end of this phase, we will have a clear game plan setting the stage for the development ahead. Let’s dive in and bring our vision to life.

### DEPLOYMENT DIAGRAM

Definition:

A Deployment Diagram is just a special type of Class Diagram that focuses on system’s node, it provides a visual representation of the physical deployment of artifacts and nodes. Essentially it describes the hardware components used in the system and how software components installed on that hardware communicate with each other. It helps showcasing the components that run on these nodes.

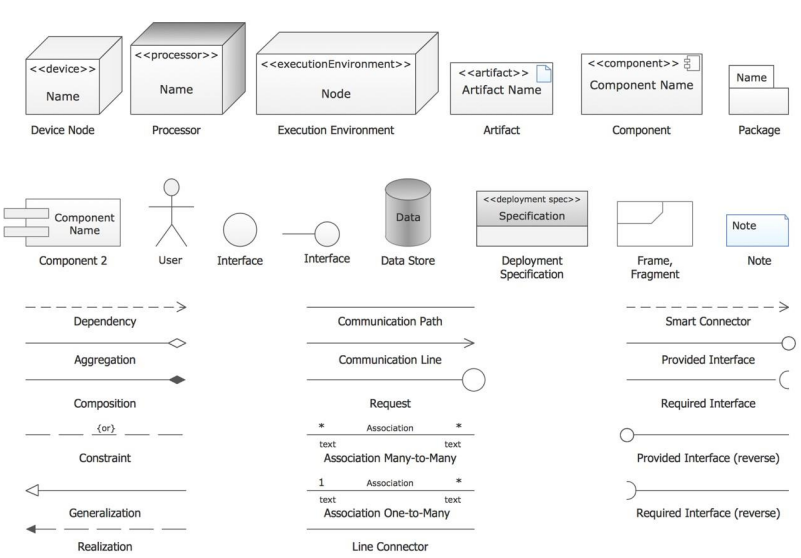


Figure 27: Visual components of Deployment Diagram

Table 16: description of element in deployment diagram

|  |  |
| --- | --- |
| Elements | Description |
| Node | Represents physical pieces of hardware where a software runs. Nodes are depicted as three-dimensional boxes |
| Artifact | Represents a tangible physical piece, usually software that resides on a node. Artifacts could be executables libraries, archives, database table, configuration files, etc. |
| Communication path | Illustrates how different nodes in the system interact with each other. It indicates the flow of information between nodes. |
| Dependency | Shows the reliance of one element (node or artifact) on another for its functionality of deployment. |
| Association | Represents relationship between two nodes, indicating communication or connectivity between them. |
| Component | While not exclusive to deployment diagrams, components represent specific functional parts of a software system. In deployment diagrams, components are used to showcase the high-level software parts that are deployed on specific nodes. |
| Package | A logical grouping of elements. In the context of a deployment diagram, it might be used to group related nodes or artifacts for better clarity |

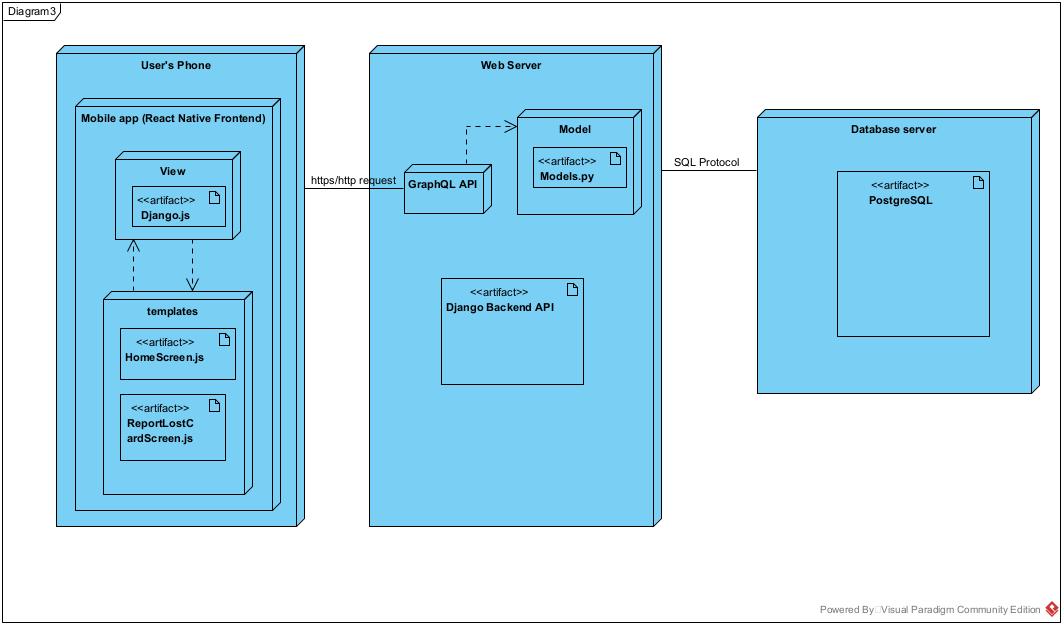


Figure 28:deployment diagram of the solution

### COMPONENT DIAGRAM

Definition:

Component Diagram is a type of structural diagram in the Unified Model Language (UML), which visualizes the organization and dependencies among a set of software components. These components can represent classes, interfaces, or entire software packages or modules. The diagram provides a high-level view of a system’s structure, illustrating how software components interact and fit together within the system.

Components of a component Diagram

Table 17: component of a Component Diagrams

|  |  |  |
| --- | --- | --- |
| Elements | Notation | Description |
| Component |  | Represented as a rectangle with the component’s name inside. Often it also has an icon of two smaller rectangles on its left side. Symbolizes a modular part of a system which can encapsulate certain functionality or a group of functionalities. |
| Interface |  | Depicted as a circle (or lollipop representation) and sometimes as a semi-circle attached to a component.  Defines a set of operations which specify the responsibility of a component. |
| Dependency |  | Illustrated dashed arrow. Signifies that one component depends on other components to function properly. |
| Port |  | Represented as a small square on the edge of a component. Defines an entry or exit point from the component for data or control flow. |
| Connector |  | Shown as a solid line between two components or ports. Signifies the communication path between components. |

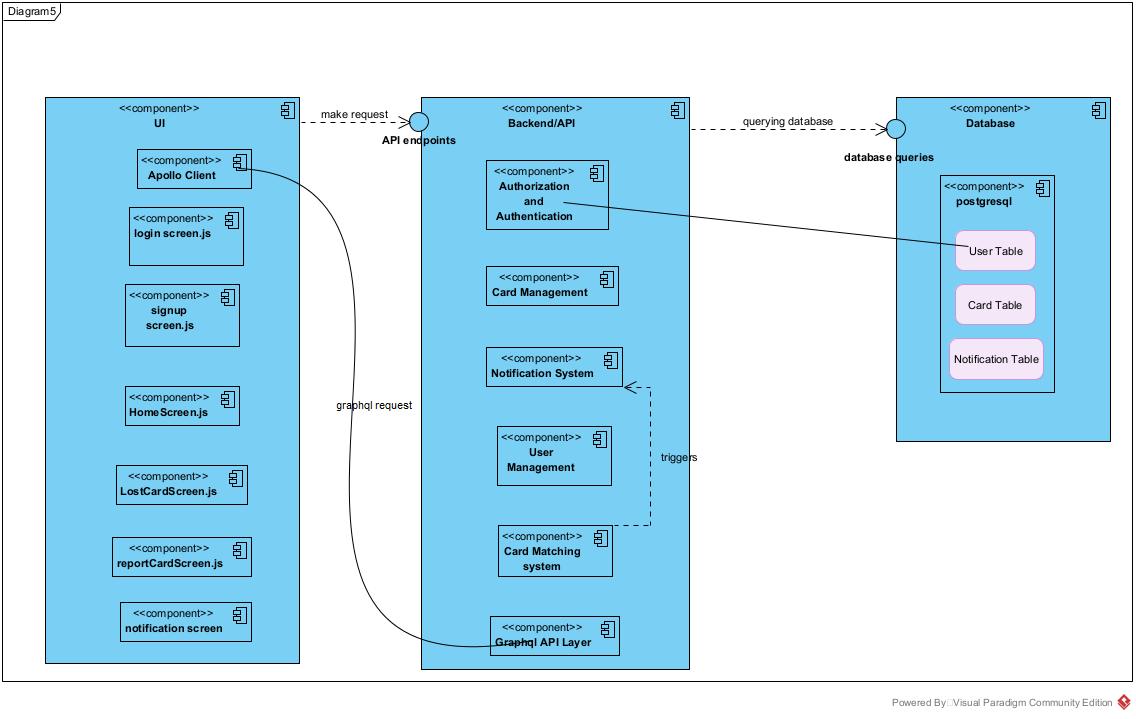


Figure 29 Component Diagram of the Software solution

### PRESENTATION OF THE TOOL USED

### SOFTWARE RESOURCES

Table 18: Software tools used in the project

|  |  |  |
| --- | --- | --- |
| Software | Logo | Roles |
| Windows 11 OS Home |  | The software where all was done that is management of the computer resources |
| Browser |  | Used in doing researches |
| StarUML | A colorful star with black text  Description automatically generated | Used in drawing UML diagrams |
| Gantt Project |  | Used to do planning of the realization of the work and to present the schedule of the work |
| Microsoft word |  | Software for the treatment of text, used in drafting the report |
| Visual studio | A blue ribbon with a cross  Description automatically generated | IDE or a studio to write the code of the application both front end and backend |

### HARDWARE RESOURCES

* Operating System windows 11 64-bit
* Memory 12GB RAM
* HARD DISK 1To
* Processor: intel iris core™ i5 11gen

### PRESENTATION OF THE LANGUAGE USED

Table 19: languages used in the project.

|  |  |
| --- | --- |
| Software | Roles |
| JavaScript (React Native) | Is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, etc. so used the framework of js (react native) in the coding section of the front end of the application. |
| Python (Django) | A python framework that makes it easier to create web sites using Python. Used to do the backend of LoFiCardApp and the server-side |
| GraphQL | A query language for APIs and a runtime for fulfilling those with your existing data. It was used to query data or fetch data via API to the frontend |
| UML | A modeling language used for the modeling of project diagrams. |
| PostgreSQL | The DBMS of my application to store and retriev e |

## CONCLUSION

In this phase we finalized with UML diagrams, and we went forth to introduce the architecture in which our application relies on and also we presented the tools used and the language used in the development of the application. Now we are going to do the functionality test of the application.

# Functionality Test

**PREAMBLE**

In today’s fast-paced world, the value of an ID card goes beyond a mere identification tool. It server as a gateway to numerous essential services and function integral to our daily lives. This guide has been meticulously crafted to assist you in navigating and harnessing the full part of the potential of our application. Whether you are first time user wanting to familiarize yourself with our platform or a returning user looking for a specific functionality, this manual provides clear step-by-step instructions tailored to ensure a seamless user experience.

INTRODUCTION

1. Test of functionality

CONCLUSION

## INTRODUCTION

**USER GUID**

# USER GUIDE

# USER GUIDE

**PREAMBLE**

In today’s fast-paced world, the value of an ID card goes beyond a mere identification tool. It server as a gateway to numerous essential services and function integral to our daily lives. This guide has been meticulously crafted to assist you in navigating and harnessing the full part of the potential of our application. Whether you are first time user wanting to familiarize yourself with our platform or a returning user looking for a specific functionality, this manual provides clear step-by-step instructions tailored to ensure a seamless user experience

INTRODUCTION

1. INSTALLATION OF THE APPLICATION
2. SHOW CASE

CONCLUSION

## INTRODUCTION

This is the final phase of our report, in this phase we will walk through in the requirements of our system, the necessary installation process, accessing our system and it futures, all this in a step by step manner to facilitate the setting up of the platform form the first time users.

### INSTALLATION OF THE APPLICATION

#### DATABASE SERVER INSTALLATION

To deploy a software, we need to install it database server, in our case PostgreSQL will be installed in our system which is windows.



Figure 30: PostgreSQL logo

1. Installing the database server

**Step 1:** Open your browser.

Go to <https://www.postgresql.org/download> and select windows

**Step 2)** Check options.  
You are given two options 1) Interactive Installer by EnterpriseDB and 2) Graphical Installer by BigSQL.

BigSQL currently installs pgAdmin version 3 which is deprecated. It’s best to choose EnterpriseDB which installs the latest version 4.

Step 3: Select PostgreSQL version.

1. You will be prompted to desired PostgreSQL version and operating system. Select the latest PostgreSQL version and OS as per your environment
2. Click the Download Button

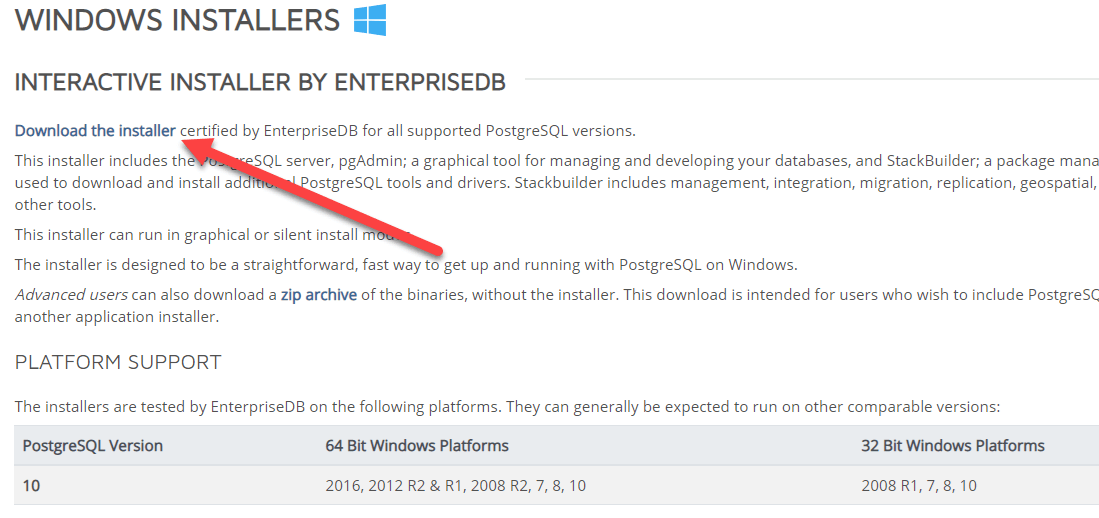


Figure 31: postgresql installer package

After download finishes, open the exe file.

Step 4: as illustrated below. A screenshot of a computer

Description automatically generated

Figure 32: postgreSQL download button

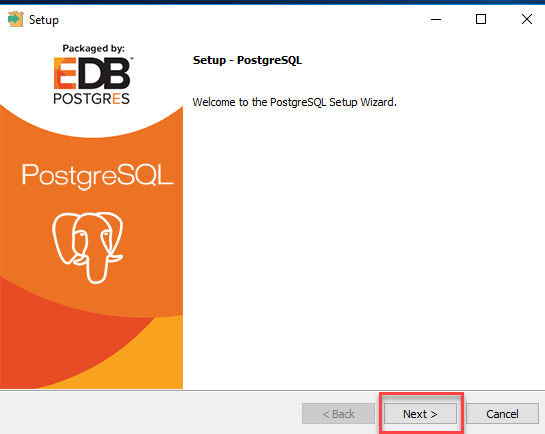


Figure 33: setup wizard postgreSQL

Step 5: update the location i.e change the installation directory if required else leave it to default and then click Next

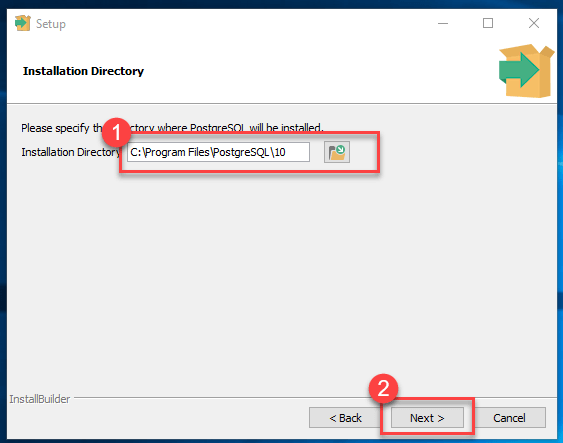


Figure 34: update installation directory

Step 6: selects components you want to install in your system You may uncheck Stack Builder and then click Next

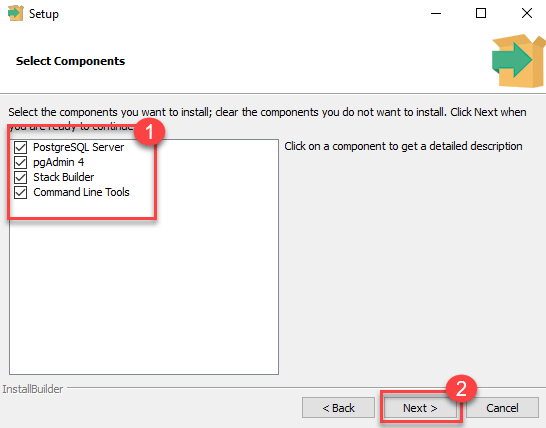


Figure 35: selects components you want to install

Step 7: check data location

You may change the data location and click Next

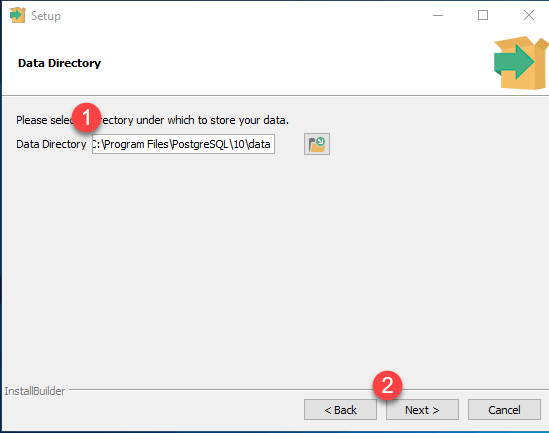


Figure 36: selects data directory

Step 8: Enter password. Enter super user password. Make a note of it and then click next

A screenshot of a computer

Description automatically generated

Figure 37: provide password

Step 9: check port option and leave the port number default then click Next.

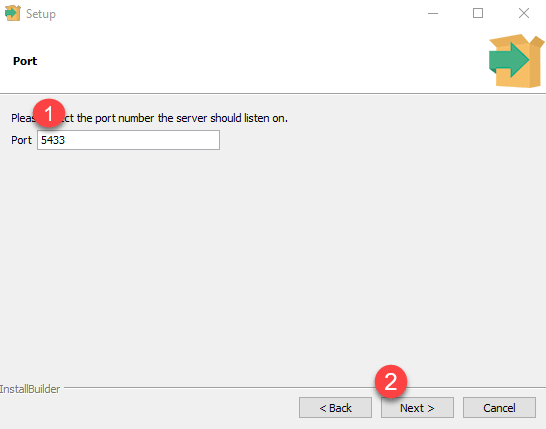


Figure 38: provide port number

Step 10: check the pre installation summary and then click next and now ready to install

Once installed you will see the stack build prompt. Uncheck that option and click Finish

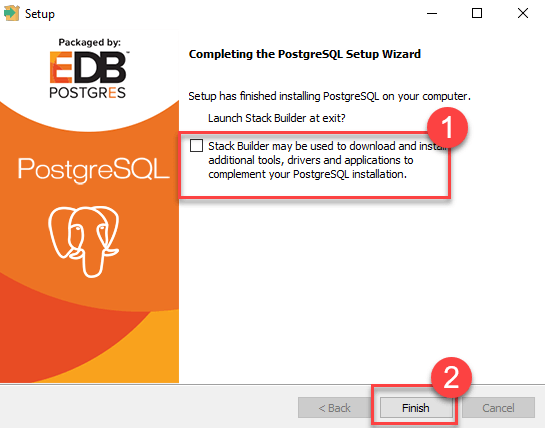


Figure 39: completing the postgreSQL Setup Wizard

#### MOBILE APPLICATION INSTALLATION

To be able to use LoFiCard mobile application, we need to download the APK file or IPA file and install it. After installing it, run the application.

### SHOWCASES

1. Here we have the Welcome Page of LoFiCard.

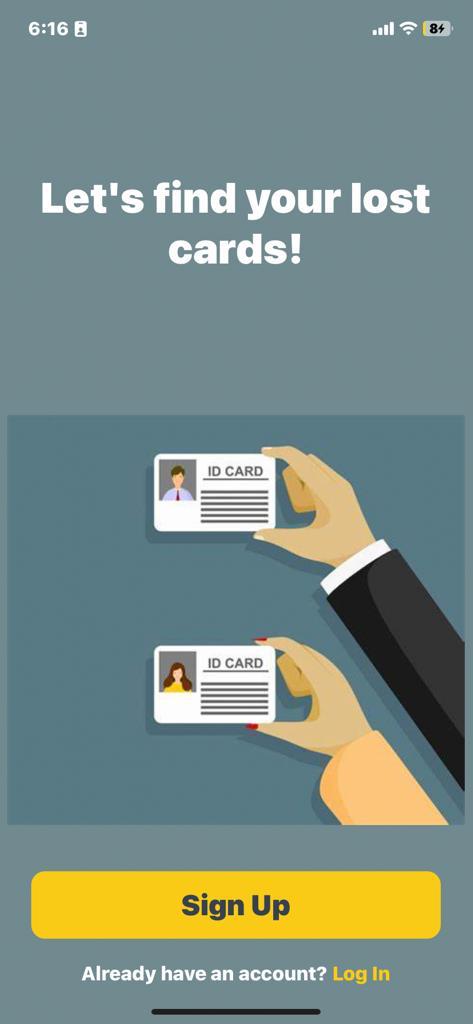


Figure 40: Welcome screen

1. Here we have the login page of LoFiCard

A screenshot of a phone

Description automatically generated

Figure 41: login screen

1. This is the home screen where the user choose a feature he want to interact with.

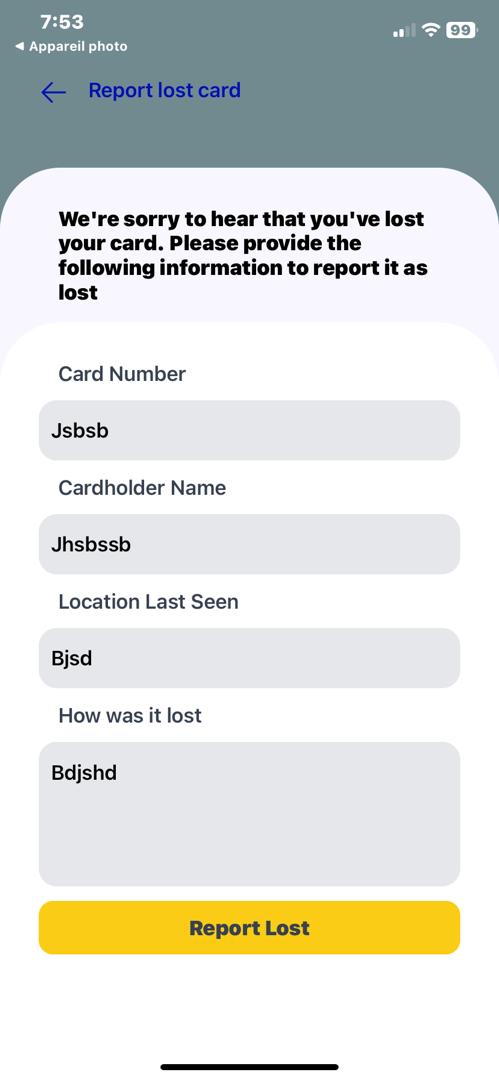


Figure 42: report lost card feature

# GENERAL CONCLUSION

As we reach the culmination of our project and our invaluable internship at Hadron SA, we find ourselves reflecting on the myriad of experiences and learnings we've imbibed. The journey, though filled with challenges, moments of stress, and instances of setbacks, was equally punctuated with triumphs and accomplishments. Our time at Hadron SA was not just professionally enriching, but it also taught us adaptability, resilience, and the value of collective teamwork. Our venture was driven by the theme “**COMPUTERIZED MANAGEMENT OF LOST ID CARDS – A Case Study of Cameroon**”. Our motivation was the Cameroonian citizen, who, due to bureaucratic processes, faces challenges in replacing a lost ID card. With a straightforward yet impactful objective, we aimed to simplify and digitize the process of recovering lost ID cards. Our journey was systematic. It commenced with identifying the needs of our users, followed by delineating project constraints and requirements. Using the UML-2TUP methodology, we dived deep into system analysis, with the versatile Visual Paradigm tool aiding our modeling efforts. Django anchored our backend, while GraphQL facilitated API functionalities. The frontend was crafted with React, ensuring users enjoyed a seamless interface. Our application, in its essence, aims to be a lighthouse for those lost in the tumultuous sea of lost ID cards. It offers a beacon of hope, guiding them towards recovery. The spirit of community is central to our platform, where users aid one another in the retrieval of lost cards. Yet, our aspirations go beyond the present. We remain steadfast in our commitment to continually refine and amplify our application's potential. As we bid adieu to Hadron SA, we are filled with gratitude and carry forward a zeal for innovation, determined to further the reach and impact of our solution.

# Annexe

# BIBLIOGRAPHY

* Mme Alice UML course AICS-Cameroon 2022-2023
* OMG, OMG Unified Modelling Language 2.5, Object Management Group, March 2015, 794 pages.
* React-native course (from basics to advanced) from UDEMY
* Django course from UDEMY
* Mr MESSIO PostgreSQL course AICS-cameroon 2022-2023

# WEBOGRAPHY

<https://htmlcolorcodes.com/> html color code for the colors use in my report

<https://www.udemy.com/course/the-complete-python-django-rest-api-development-bootcamp/learn/lecture/24481164?start=165#overview> for learning the basics in Django

<https://www.hippotechgroup.com/2021/01/how-to-get-your-cameroon-national-id-card-number.html> know the steps to have a card in Cameroon, September 28 2023

<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/> know how to draw UML diagrams, September 27 2023

<https://v2.crowdfind.com/site_login> study of related software doing the same application, September 25 2023

<https://www.postgresql.org/download/> downloading PostgreSQL, 22 september 2023

<https://www.visual-paradigm.com/download/?platform=windows&arch=64bit> downloading visual paradigm for the UML diagrams, 21 september 2023

<https://github.com/chrispirit/LoFiCardApp/tree/donfack/assets/image> putting work project on git hub, September 13 2023

<https://dribbble.com/get-started#welcomeDesigner> inspiration for the home page, 12 september 2023

<https://reactnative.dev/docs/environment-setup> setting up react native, sep 6 2023

<https://docs.expo.dev/get-started/installation/> documenting on expo sep 6 2023

<https://staruml.io/download/> downloading star UML for the August 29 2023

<https://www.javatpoint.com/uml-state-machine-diagram> learn state machine diagram, August 2023

# TABLE OF CONTENT

[DEDICATION i](#_Toc147113492)

[ACKNOWLEDGEMENTS ii](#_Toc147113493)

[CONTENT iii](#_Toc147113494)

[LIST OF TABLES v](#_Toc147113495)

[LIST OF FIGURES vi](#_Toc147113496)

[GLOSSARY vii](#_Toc147113497)

[ABSTRACT viii](#_Toc147113498)

[RESUME 1](#_Toc147113499)

[GENERAL INTRODUCTION 2](#_Toc147113500)

[PART I 3](https://d.docs.live.net/4be1740105e2a1f3/Documents/presentation%20of%20hadron%20(Repaired).docx#_Toc147113501)

[INSERTION PHASE 3](https://d.docs.live.net/4be1740105e2a1f3/Documents/presentation%20of%20hadron%20(Repaired).docx#_Toc147113502)

[INTRODUCTION 5](#_Toc147113503)

[I. WELCOME AND INTEGRATION at Hadron SA 6](#_Toc147113504)

[II. General presentation of Hadron 7](#_Toc147113505)

[III. Administrative and Functional Organization of Hadron 10](#_Toc147113506)

[IV. Hardware and Software resources of Hadron 13](#_Toc147113507)

[Conclusion 15](#_Toc147113508)

[PART II 16](https://d.docs.live.net/4be1740105e2a1f3/Documents/presentation%20of%20hadron%20(Repaired).docx#_Toc147113509)

[EXISTING SYSTEM 16](https://d.docs.live.net/4be1740105e2a1f3/Documents/presentation%20of%20hadron%20(Repaired).docx#_Toc147113510)

[INTRODUCTION 18](#_Toc147113511)

[I. Brief Presentation of the project theme 19](#_Toc147113512)

[II. STUDY OF THE EXISTING SYSTEM 20](#_Toc147113513)

[III. CRITICISM OF THE EXISTING SYSTEM 21](#_Toc147113514)

[IV. PROBLEMATICS 22](#_Toc147113515)

[V. PROPOSED SOLUTION 23](#_Toc147113516)

[CONCLUSION 24](#_Toc147113517)

[SPECIFICATION BOOK 25](https://d.docs.live.net/4be1740105e2a1f3/Documents/presentation%20of%20hadron%20(Repaired).docx#_Toc147113518)

[INTRODUCTION 27](#_Toc147113519)

[I. CONTEXT AND JUSTIFICATION OF STUDIES 28](#_Toc147113520)

[II. OBJECTIVES OF THE PROJECT 29](#_Toc147113521)

[III. EXPRESSION OF NEEDS 30](#_Toc147113522)

[IV. PLANNING OF THE PROJECT 32](#_Toc147113523)

[V. ESTIMATION OF THE PROJECT 35](#_Toc147113524)

[VI. PROJECT CONSTRAINTS 38](#_Toc147113525)

[VII. DELIVERABLES 39](#_Toc147113526)

[CONCLUSION 40](#_Toc147113527)

[ANALYSIS PHASE 41](https://d.docs.live.net/4be1740105e2a1f3/Documents/presentation%20of%20hadron%20(Repaired).docx#_Toc147113528)

[INTRODUCTION 43](#_Toc147113529)

[I. SOME MODELLING TECHNIQUES 44](#_Toc147113530)

[II. PRESENTATION OF THE ANALYSIS METHOD 46](#_Toc147113531)

[III. MODELING WITH UML AND 2TUP STRUCTURE 52](#_Toc147113532)

[CONCLUSION 84](#_Toc147113533)

[CONCEPTION PHASE 85](https://d.docs.live.net/4be1740105e2a1f3/Documents/presentation%20of%20hadron%20(Repaired).docx#_Toc147113534)

[INTRODUCTION 87](#_Toc147113535)

[I. TECHNICAL BRANCH 88](#_Toc147113536)

[II. IMPLEMENTAION BRANCH 90](#_Toc147113537)

[a. PACKAGE DIAGRAM 90](#_Toc147113538)

[b. STATE MACHINE DIAGRAM 93](#_Toc147113539)

[A. DETAILED DESIGN 97](#_Toc147113540)

[a. CLASS DIAGRAM 97](#_Toc147113541)

[CONCLUSION 100](#_Toc147113542)

[REALIZATION PHASE 101](https://d.docs.live.net/4be1740105e2a1f3/Documents/presentation%20of%20hadron%20(Repaired).docx#_Toc147113543)

[INTRODUCTION 103](#_Toc147113544)

[I. DEPLOYMENT DIAGRAM 104](#_Toc147113545)

[II. COMPONENT DIAGRAM 107](#_Toc147113546)

[III. PRESENTATION OF THE TOOL USED 110](#_Toc147113547)

[A. SOFTWARE RESOURCES 110](#_Toc147113548)

[B. HARDWARE RESOURCES 110](#_Toc147113549)

[IV. PRESENTATION OF THE LANGUAGE USED 111](#_Toc147113550)

[CONCLUSION 112](#_Toc147113551)

[USER GUIDE 116](#_Toc147113552)

[USER GUIDE 116](https://d.docs.live.net/4be1740105e2a1f3/Documents/presentation%20of%20hadron%20(Repaired).docx#_Toc147113553)

[INTRODUCTION 118](#_Toc147113554)