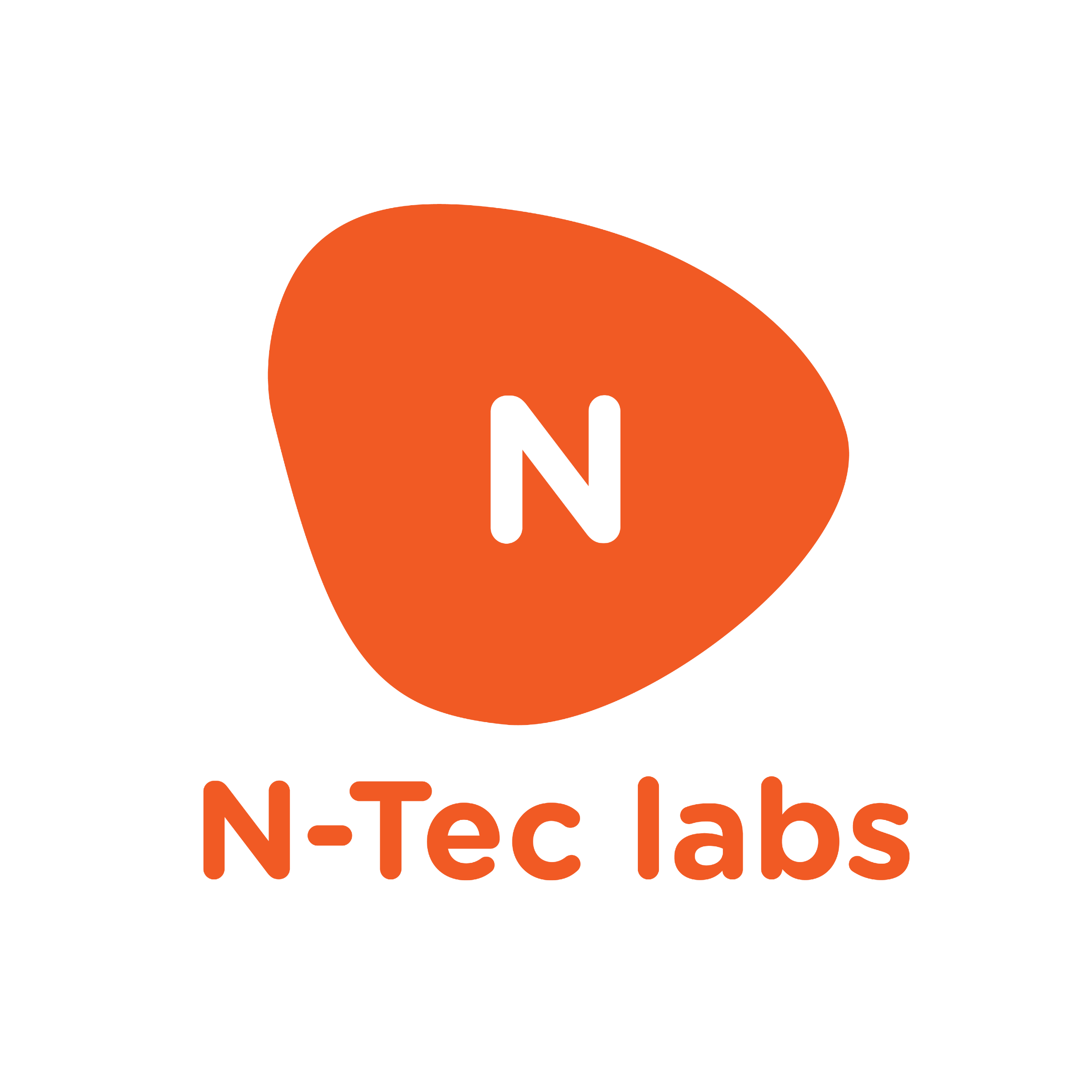
**REPUBLIC OF CAMEROON**

Peace-Work-Fatherland



A logo with a map and a globe

Description automatically generated

**REPUBLIQUE DU CAMEROUN**

Paix-Travail-Patrie

**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

**COMPUTERISED MANAGEMENT OF RESTAURANT SERVICES**

**CASE STUDY: SNAP BURGER CRADAT YAOUNDE**

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**

ACKNOWLEDGEMENTS

**ACADEMIC YEAR 2023-2024**

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 the internship period which helped in the realization 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

**LIST OF ABBREVIATIONS**

* **2TUP:** Two Track Unified Process
* **AICS:** African Institute of Computer Sciences
* **API:** Application Programming Interface
* **CSS:** Cascading Style Sheet
* **DBMS:** Database Management System
* **ER:** Entity Relationship
* **ERD:** Entity Relationship Diagram
* **HTML:** Hyper Text Markup Language
* **HTTP:** Hypertext Transfer Protocol
* **PDM:** Physical Data Model
* **RMS;** Restaurant Management System
* **SQL:** Structured Query Language
* **Mr.:** Mister
* **Mrs.:** Miss

**LIST OF FIGURES**

[Figure 1: geographical location of N-TEC LABS 10](#_Toc147113555)

[Figure 2 organization chart of N-TEC LABS 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 Ordering 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)

**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)

**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

The insertion phase is a part of t he internship report where we will present the detailed structure and characteristics of our enterprise where our internship was carried out.

Content

INTRODUCTION

1. WELCOME AND INTEGRATION
2. GENERAL PRESENTATION OF THE COMPANY
3. ORGANISATION OF THE COMPANY
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.

1. **WELCOME AND INTEGRATRION**
2. **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

1. **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.

1. **GENERAL PRESENTATION OF THE COMPANY**
2. **Geographical location**



*Figure1: Location of N-tec Labs*

1. Company’s Identification Form

|  |  |
| --- | --- |
| IDENTIFICATION FORM | |
| Company name | N-TEC LABS |
| Type of Enterprise | Institution, Digital Agency |
| President of the Company | Mr. NWANYI Joel |
| Office Telephone | (+237)651834545 |
| Website | nteclabs.com |
| Language | English and French |
| Logo | A logo with a map and a globe  Description automatically generated |

*Table 1: Company Identification form*

1. **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.

1. **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.

1. **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.

1. **Activities**

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

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

1. **ORGANISATION OF THE ENTERPRISE**
2. Administrative Organisation of N-Tec Labs

N-Tec Labs is administratively organized as follows;

1. The General Management

This is the decision-making department of the company. They perform some of the following functions

* Ensure the office runs properly
* Evaluate the realization of projects by the company

1. The Teaching Department

This department is in charge of the school activities of the enterprise.

1. The Development Department

This is the heart of the company because they carry out the company’s projects. They perform some of the following functions:

* Reenforce the application standards of the company
* Write training content
* Maintenance
* Innovations

1. Functional organization of N-Tec Labs

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

PRESIDENT OF N-TEC LABS

SECRETARY

DEVELOPMENT TEAM

SUPERVISOR

INTERNS

STUDENTS

*Figure 2: Organisation Chart of N-Tec Labs*

1. **HARDWARE AND SOFTWARE RESOURCES OF THE COMPANY**
2. **Hardware Resources**

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

***Table 2****: Hardware resources of N-tec labs*

### Software Resources

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

|  |  |
| --- | --- |
| 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 |

***Table 3****: Software resources of N-Tec Labs*

1. **BRIEF PRESENTATION OF THE PROJECT**

During our insertion phase at N-Tec Labs, we were asked to look for themes which solve a specific problem and something innovative. Our professional supervisor took some time to look into what we proposed as themes and he also proposed some theme ideas. Finally, we were assigned the theme **“RESTAURANT MANAGEMENT SYSTEM”**.

**CONCLUSION**

To end, our insertion phase into N-tec labs was a serene, warm and convenient experience with the company’s personnels. One of the things we learned during this phase was collaboration and discipline which is essential for a successful career.

**PART II:**

**SPECIFICATION BOOK**

Preamble

The primary objective of the specification book is to outline 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 OF STUDY
2. OBJECTIVES OF THE PROJECT
3. EXPRESSION OF USER NEEDS
4. PLANNING OF THE PROJECT
5. ESTIMATION OF COST OF THE PROJECT
6. CONSTRAINTS OF THE PROJECT
7. DELIVERABLES

CONCLUSION

INTRODUCTION

**INTRODUCTION**

The Specification Book lays the vital groundwork for this software project by mapping out the vision, requirements, designs and plan that will guide us from idea to completion. This thorough documentation serves as a roadmap and compass to smoothly navigate the development journey and ensure the end goal is achieved.

1. CONTEXT AND JUSTIFICATION OF STUDY

In the modern dining experience, convenience is key, and managing a restaurant's operations goes beyond just food preparation. It involves providing seamless, fast, and personalized services to customers. However, many restaurants like Snap Burger face challenges in keeping up with customer demands while managing orders, tracking inventory, and ensuring a smooth dining experience.

This led to a series of important reflections:

* Could there be a more efficient way to manage customer orders without overwhelming the staff?
* How can technology be used to create a personalized and fast ordering experience?
* What if there was a way to allow customers to customize their meals and track orders in real-time?
* Can restaurant owners reduce wait times and enhance customer satisfaction with a streamlined system?

These reflections inspired the creation of our "Computerised Management Of Restaurant Services." This platform is designed to bridge the gap between the restaurant and its customers by providing an intuitive, tech-driven experience that enhances both operational efficiency and customer satisfaction. From real-time order tracking to easy meal customization, the system empowers restaurants to deliver exceptional service, ensuring no customer is left waiting for the perfect dining experience.

1. OBJECTIVES OF THE PROJECT
2. GENERAL OBJECTIVES

To develop and implement an intuitive digital platform that streamlines restaurant operations and enhances the dining experience through efficient management of orders, meal customization, and customer satisfaction.

1. SPECIFIC OBJECTIVES

In order to achieve the overarching goal, several specific objectives must be realized:

* **User-Friendly Ordering**: Design an easy-to-navigate interface where users can browse menus, customize their orders, and make real-time requests.
* **Order & Customization Management**: Implement a robust order management system that allows users to seamlessly customize meals and track orders from preparation to delivery.
* **Real-Time Notifications**: Establish a notification system that alerts customers of order status updates, ensuring clear communication between the kitchen, staff, and customers.
* **Safety & Security**: Ensure the platform maintains high standards of security to protect sensitive customer data and payment information, while making the platform accessible for all users.
* **Name of the Project**: SERVESOFT
* **Target Audience**: Restaurants and diners looking for an enhanced, tech-driven dining experience.
* **Technical Specification**: A web-based and mobile application for ordering, meal customization, and order management.

1. EXPRESSION OF USER NEEDS

This section unveils the essential requirements of the software project, serving as a vital link between the problem at hand and the proposed solution. It emphasizes the "what" instead of the "how," painting a clear picture of the project's goals. These needs are artfully divided into two categories: functional requirements, which define the system's capabilities, and non-functional requirements, which outline quality attributes that enhance user experience.

1. FUNCTIONAL NEEDS

In this context, this expresses **what** the system is supposed to do and these features are as follows:

* **Sign Up & Login for staff**: Staff must sign up and log in to access their profiles.
* **Meal Selection**: Customers can browse a comprehensive menu, categorized into different food types such as deserts, salads, beverages, and more.
* **Meal Customization**: The app offers a customizable meal ordering system, allowing users to modify dishes to their preferences (e.g., extra mayonnaise, pepper options).
* **Order Tracking & Notifications**: Once the order is placed, users receive real-time updates about the order status (in preparation, en route, delivered).
* **Reorder from Recent Orders**: Users can easily reorder meals they’ve enjoyed before from their recent order history, saving time.

1. NON-FUNCTIONAL NEEDS

In this context, non-functional needs refer to the **quality attributes** that determine how the app should operate to enhance user experience and satisfaction. These attributes are essential for the long-term success of the platform. These features include:

* **Security**: Personal information, such as staff details, addresses, and order history, should be securely stored and encrypted to ensure user privacy.
* **Performance**: The application should load quickly, even under high traffic, and ensure a smooth user experience. This includes responsive pages, fast processing of orders, and minimal downtime to avoid frustrating the customers and staff.
* **Usability**: The app must be intuitive and easy to navigate, ensuring that users can browse menus, customize orders, and place requests without confusion. A simple and clean interface will reduce complexity, encouraging repeated use.
* **Reliability**: The platform must consistently function as expected, without crashes or failures, ensuring that orders are processed and delivered correctly. A reliable system will build trust with users and meet market expectations.

1. PLANNING OF THE PROJECT
   1. ESTMATION OF RESOURCES NEEDED
2. Software Resources

|  |  |  |  |
| --- | --- | --- | --- |
| SOFTWARE | USAGE | Logo | PRICE(FCFA) |
| Microsoft 365 | Used for the creation our report and PowerPoint |  | 500,500 |
| Visual studio code | Open and free software for the coding part of the project |  | Free |
| PowerAMC | For doing the design of the diagrams to represent the structure and the interaction between the different actors |  | 150,000 |
| Google | For doing the research on the internet and testing of the application |  | Free |
| Windows 11, version 23H2 | Operating system |  | Free |
| Gantt Project | For building a gantt chart for the planning of the work |  | Free |
| XAMPP | For creating database |  | Free |
| Visual Paradigm Enterprise version | For doing the design of the diagrams to represent the structure and the interaction between the different actors |  | 500,000 |
| TOTAL | **1,150,500** | | |

Table 5: Software resources of the project

1. Hardware Resources

|  |  |  |  |
| --- | --- | --- | --- |
| Material | USAGE | QUANTITY | PRICE (FCFA) |
| LAPTOP (LENOVO i5 8th Gen, SSD) | Used to build the system, type the report, create the power point, make research, etc. | 01 | 425,000 |
| USB flash disk 64gb | Used for storage and to print the report | 01 | 16,000 |
| Internet Modem | Used for internet connection | 01 | 65,000 |
| Samsung Tablet | Used for carrying research and storage | 01 | 150,000 |
| Printer HP Officejet 4630 | Used to print the scientific document | 01 | 86,550 |
| TOTAL: | **742,550** | | |

Table 6: Hardware resources of the project

1. Human Resources

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Role | Duration (Week) | Quantity | Unit Price (FCFA) | Total Price (FCFA) |
| Project manager | 08 | 01 | 250,000 | 2,000,000 |
| DevOps | 06 | 01 | 205,000 | 1,230,000 |
| Analyst | 02 | 01 | 200,000 | 4,00,000 |
| Designer | 02 | 01 | 150,000 | 300,000 |
| Developer | 04 | 01 | 100,0000 | 400,000 |
| Tester | 02 | 02 | 50,000 | 50,000 |
| TOTAL | **4,680,000** | | | |

Table 7: Hardware resources of the project

1. Overall Resources

|  |  |
| --- | --- |
| Designation | Cost (FCFA) |
| Software Resources | 1,150,500 |
| Hardware Resources | 742,550 |
| Human Resources | 4,680,000 |
| Unforeseen | 500,000 |
| Total in figures | **7,073,050** |
| Total in Words | **Seven million seventy-three thousand fifty.** |

Table 8: Overall resources of the project

* 1. ESTMATION OF RESOURCES NEEDED

|  |  |  |  |
| --- | --- | --- | --- |
| Phase | Task / Objective | Output | Duration (WEEKS) |
| Insertion Document | Collection of Information concerning the Enterprise | Insertion Report | 02 |
| Specification Book | Specification of the user’s need | Specification Book | 01 |
| Analysis | Study of the existing system, modelling user interactions | Analysis Report | 02 |
| Conception | Preliminary Conception  Detailed Conception | Conception Report | 01 |
| Realization | Implementation, Unitary tests, Integration test, development | Realization document | 01 |
| User Guide | User guide for solution | User guide | 02 |
| Presentation | Design slides showing how work was realised | PowerPoint | 01 |

Table 9: Estimation of time required for project

* 1. GANTT DIAGRAM

This diagram in planning, scheduling and monitoring the project. It was realised using the software **GANTT PROJECT.**

Figure3: Gantt Diagram

1. PROJECT CONSTRAINTS
2. Criterions Of Acceptability

The delivered product is judged acceptable if it respects the different functionalities that have been presented.

1. Time Constraint

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

1. DELIVERABLES

Regarding this project, below lies its deliverables:

* A report composed of the following:
  + The analysis documents
  + The conception documents
  + The realisation document
  + The user guides

**CONCLUSION**

In conclusion, this section represents a crucial milestone, demonstrating the application's market value, its benefits for Cameroonians, and providing a clear timeline with cost estimates for successful project completion.

# **PART THREE:**

# **ANALYSIS PHASE**

Preamble

The main goal of the analysis phase is to thoroughly capture the users' needs, define the project’s boundaries, and obtain a clear understanding of the restaurant system being developed. To achieve this, we will utilize UML (Unified Modeling Language) combined with the 2TUP (2-Track Unified Process) methodology, which is specifically applied to analyze the system’s functionality. Content

INTRODUCTION

1. DESCRIPTION OF THE EXISTING SYSTEM
2. LIMITS OF THE EXISTING SYSTEM
3. PROBLEMATIC
4. DELIMITATION OF THE FIELD OF STUDY
5. PRESENTATION OF THE MODELING APPROACH
6. CHOICE OF THE ANALYSIS METHOD
7. MODELING OF THE PROPOSED SOLUTION

CONCLUSION

**INTRODUCTION**

Building a restaurant app is like embarking on a creative journey, where the first step is to deeply understand the existing system and set clear goals for the future. This phase goes beyond just identifying where we are but it’s about pinpointing key challenges and providing targeted solutions that enhance the dining experience. We apply innovative methodologies and advanced modeling techniques, not only to address the app's requirements but to resolve existing pain points within the system. Whether it's streamlining orders, improving customer engagement, or optimizing restaurant operations, every step of the analysis is aimed at delivering a solution that is both forward-thinking and meticulously crafted. The ultimate goal is to transform identified challenges into opportunities for creating a refined, efficient, and user-friendly app experience.

* 1. DESCRIPTION OF THE EXISTING SYSTEM

The existing system refers to the framework already in place, which has led to the decision to develop a more efficient solution. The current system presents several challenges that disrupt smooth operations, making it crucial to fully understand these issues before moving forward. Through thorough research and investigation, we have identified key problems that are affecting the user experience and operational efficiency. This phase of analysis serves as foundational research to outline the framework for our proposed solution. After all possible studies carried out on the existing system we identified the following consequences of the various problems. in addition to these consequences, we provided some proposed solutions:

* 1. LIMITS OF THE EXISTING SYSTEMF THE EXISTIN

|  |  |  |
| --- | --- | --- |
| Criticism | Consequences | Proposed Solutions |
| Slow ordering process via waiter | Customers experience longer wait times, leading to frustration and dissatisfaction. | Implement an app-based ordering system allowing customers to place orders directly through their phones. |
| Limited menu visibility | Customers may not be aware of special offers or detailed dish descriptions, limiting their choices. | Provide a digital menu within the app that includes item details, pictures, and real-time specials. |
| Human error in order taking | Miscommunication between waiters and the kitchen can lead to incorrect orders or missed items. | The app automates order transmission directly to the kitchen, reducing the chances of human error. |
| Difficulty tracking customer preferences | Waiters have difficulty remembering individual customer preferences for future visits. | Use the app to store customer preferences and order history, enabling personalized recommendations. |
| Waiter availability issues | Waiters may be busy or unavailable when customers need assistance, causing delays in service. | Allow customers to request assistance or modifications through the app, ensuring quicker responses. |
| Inaccurate billing due to manual calculations | Errors in manually calculating bills can cause discrepancies and customer dissatisfaction. | Automate bill generation through the app to ensure accuracy and transparency in billing. |
| Difficulty handling large groups efficiently | Large parties may face delays in placing and receiving orders, impacting their overall dining experience. | The app can offer group ordering features, allowing multiple guests to place their orders simultaneously. |

Table 10: Limits of the existing system (criticisms, consequences and proposed solutions)

* 1. PROBLEMATIC

With the current restaurant system described and its limitations outlined, we can now confidently identify the problem, which is **“HOW CAN WE COMPUTERIZE AND FACILITATE THE MANAGEMENT OF SERVICES IN RESTAURANTS?**” This question guided the development of the new system, which aims to address, if not entirely resolve, many of the issues highlighted. By leveraging technology, the proposed solutions will enhance service speed, reduce errors, and elevate the overall dining experience, creating a more seamless and satisfying environment for both customers and staff.

* 1. DELIMITATIONS OF THE FIELD OF STUDY

In this restaurant app project, the focus will be on the following modules:

* CUSTOMER
  + View their order history
  + Place new orders
* STAFF
  + Users can sign up and log in to their accounts
* ADMIN
  + Update menu items
  + Track restaurant reports
  + Manage user accounts
  1. PRESENTATION OF THE MODELING APPROACH
     1. SOME MODELLING TECHNIQUES

In analyzing the software project, it's essential to select a model that effectively designs and visualizes the project's structure based on its specific needs. Here, we explore various modeling techniques and discuss the chosen approach for the project's development.

1. MERISE

**MERISE** (Méthode d'Étude et de Réalisation Informatique pour les Systèmes d'Entreprise) is a structured methodology used in the development and management of information systems. Developed in France in the late 1970s, it primarily focuses on the design and modeling of data and processes within an organization. MERISE is based on the separation of data and processes, allowing for a more modular and flexible approach to system development. It separates system design into three levels: conceptual (high-level understanding of data), logical (structured data and processes), and physical (implementation). MERISE emphasizes data modeling using the Entity-Relationship model and is particularly useful for complex, data-driven projects. It’s widely used in French-speaking regions for systematic and modular software development.

1. SCRUM

Scrum is an Agile framework introduced in the early 1990s, designed to manage complex product development through collaboration, adaptability, and continuous feedback. It breaks projects into short cycles called **Sprints**, typically lasting 2-4 weeks, during which a cross-functional team focuses on delivering specific, prioritized tasks. Key roles include the **Scrum Master** (ensuring smooth process), **Product Owner** (defining and prioritizing requirements), and the **Development Team** (executing the work). Scrum also emphasizes practices like **Daily Stand-ups**, **Sprint Planning**, and **Retrospectives** to promote communication and continuous improvement. Unlike traditional methods, Scrum delivers small, testable product increments throughout development, allowing for early validation and quick adaptation to feedback.

1. UP

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.

1. DYNAMIC SYSTEMS DEVELOPMENT METHOD (DSDM)

**DSDM (Dynamic Systems Development Method)** is one of the earliest Agile methodologies, introduced in the 1990s as part of the Agile Alliance. It is designed to help organizations deliver high-quality software solutions quickly and efficiently, while ensuring flexibility and collaboration throughout the development process. The core focus of DSDM is on meeting business objectives through a balanced approach that emphasizes **collaboration**, **timely delivery**, and **adaptability.**

* + 1. PRESENTATION OF UML, 2TUP AND ERD
       1. UML

**UML (Unified Modeling Language)** is a standardized, versatile modeling language used extensively in software engineering and system design. Introduced in the 1990s by pioneers Grady Booch, Ivar Jacobson, and James Rumbaugh, UML was created to offer a consistent framework for visualizing, designing, and documenting software systems.

Since its inception, UML has evolved into an indispensable tool for software architects, developers, business analysts, system engineers, and stakeholders. It facilitates clear communication and precise documentation across various aspects of software development.

UML 2.O comprises of 13 diagrams categorized into two main groups

1. **Structural Diagrams**: These diagrams represent the static aspects of a system, illustrating its components and their relationships. Diagrams in this category include:
   * + - Class Diagram
       - Component Diagram
       - Composite Structure Diagram
       - Deployment Diagram
       - Package Diagram
       - Object Diagram
2. **Behavioral Diagrams**: These diagrams depict the dynamic aspects, showcasing how the system behaves and interacts over time. Diagrams in this category include:
   * **Use Case Diagram**
   * **Sequence Diagram**
   * **State Machine Diagram**
   * **Activity Diagram**
   * **Communication Diagram**.
   * **Interaction Overview Diagram**.
   * **Timing Diagram**

With its comprehensive set of diagram types, UML provides a robust foundation for understanding and managing complex software systems, ensuring clarity and coherence throughout the development process. However, it is important to note that UML is not a modeling methodology on its own. To effectively apply UML, it needs to be associated with a structured approach, such as the Unified Process (UP). While there are several variations of the Unified Process, our modeling approach for this project will be the 2TUP (Two-Track Unified Process). This methodology will guide our conceptualization and ensure a systematic approach throughout the project.

* + - 1. COMPARISON BETWEEN UML AND MERISE

|  |  |  |
| --- | --- | --- |
| Aspect | UML (Unified Modeling Language) | MERISE |
| Primary Focus | General-purpose modeling language for software engineering. | Methodology for designing and managing information systems, with a focus on data and processes. |
| Diagram Types | 14 diagram types, including Class, Use Case, Sequence, and Activity diagrams. | 6 main diagrams, including Context Diagram, Data Flow Diagram, and Entity-Relationship Diagram. |
| Modeling Paradigms | Object-oriented, emphasizing objects and their interactions. | Entity-relationship and process-oriented, focusing on data modeling and process flow. |
| Level of Abstraction | Provides various levels of abstraction from high-level overviews to detailed designs. | Focuses more on data and process modeling with a less diverse set of diagrams. |
| Use Case Focus | Strong emphasis on use cases and user interactions. | Less emphasis on use cases, more focus on data and process modeling. |
| Methodological Approach | Iterative and incremental approach to software development. | More structured and sequential approach, particularly for data analysis and system design. |

Table 11: Comparison of MERISE and UML

This comparison highlights the differences in focus, methodology, and application between UML and MERISE. UML is a broader, internationally recognized modeling language with a focus on object-oriented design, while MERISE is more specialized for data and process modeling, with strong roots in French methodologies.

* + - 1. TWO TRACK UNIFIED PROCESS (2TUP)

In software development, both the process model and the tools for capturing and communicating requirements are crucial for project success. The **2TUP (Two-Track Unified Process)** model provides a strategic approach by integrating conceptual and technical aspects into a structured, Y-shaped flow. This model divides the development process into two parallel tracks: the **Functional Branch** focusing on defining what the system should do, and the **Technical Branch** concentrating on how it will be implemented. These branches converge in the **Realization Branch**, where both tracks are integrated to achieve the final system. The figure below gives us more details on how software development follows the three branches of 2TUP.



Figure 3 : Two-Track Unified Process

This diagram illustrates a software development process involving two main branches: the **Functional Branch** and the **Technical Branch.** Both of these branches work in parallel, and the process converges towards the **Realization Branch** to complete the project.

* + **Functional Branch**
* **Capture of business requirements**: This step involves gathering the business needs and goals. It focuses on understanding the features, functionality, and objectives of the application from a business perspective. Techniques like **use case diagrams** might visualize complex business interactions.
* **Analysis**: After gathering the business requirements, an analysis is performed to evaluate the needs, align them with technical feasibility, and understand how the system will fulfill these requirements.
  + **Technical Branch**
* **Capture of technical needs**: In this step, the technical requirements are identified. It includes the technical specifications necessary to build the system, such as hardware, software, and technical constraints.
* **Generic Design**: The technical requirements are translated into a generic design, which outlines the system's overall architecture, defining how different components will work together.
  + **Realization Branch**

The two branches merge at the realization stage, where the functional and technical designs are implemented in a step-by-step manner:

* **Preliminary Design**: A rough system design that incorporates both business and technical requirements.
* **Detailed Design**: A more refined and detailed version of the system design that specifies every element of the system and how it will be built.
* **Coding and Testing**: The actual coding of the application takes place, followed by testing to ensure it meets both functional and technical requirements.
* **Recipe**: This step likely represents the finalization of the system and could involve preparing the application for deployment or outlining the procedures for implementation.
  1. CHOICE OF THE ANALYSIS METHOD

1. JUSTIFICATION AND MOTIVATION FOR THE CHOSEN APPROACH

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

* **Justification:**

The Two-Track Unified Process (2TUP), when integrated with Unified Modeling Language (UML) diagrams, provides a clear and structured way to visualize a software system’s functional requirements, technical design, and development plan.

UML provides a common standard, ensuring clear communication among stakeholders and reducing miscommunication risks. Its flexibility, combined with 2TUP's iterative approach, enables ongoing refinement of diagrams as requirements and designs evolve. UML offers a variety of diagram types, such as: use case diagrams (capturing business needs), sequence diagrams (for system interactions), and deployment diagrams (for technical implementation). These diagrams effectively represent each stage of the 2TUP methodology.

* **Motivation:**

1. **Improved understanding:** UML diagrams help teams better comprehend system requirements and designs, leading to a more efficient development process.
2. **Enhanced collaboration:** UML serves as a common language between stakeholders and developers, promoting clear communication and shared understanding.
3. **Early issue identification:** Visualizing the system early on, combined with the iterative approach, helps teams spot and resolve potential issues before they become major problems, reducing risks.

By combining UML with the 2TUP methodology, teams can integrate strategic planning with visualization, improving clarity, collaboration, and efficiency throughout the software development lifecycle.

* 1. MODELING OF THE PROPOSED SOLUTION
     1. FUNCTIONAL BRANCH

Here we are focusing on modeling business requirements and system functionality, ensuring clear communication among stakeholders.

* + - * 1. USE CASE DIAGRAM

**Definition**

A use case diagram visually depicts the interactions between users (actors) and a system. It focuses on how actors engage with the system to accomplish specific tasks. This diagram helps outline the system's functional requirements from the user's perspective.

**Formalism**

A diagram of a system

Description automatically generated

Figure 4 : Use case Diagram Formalism

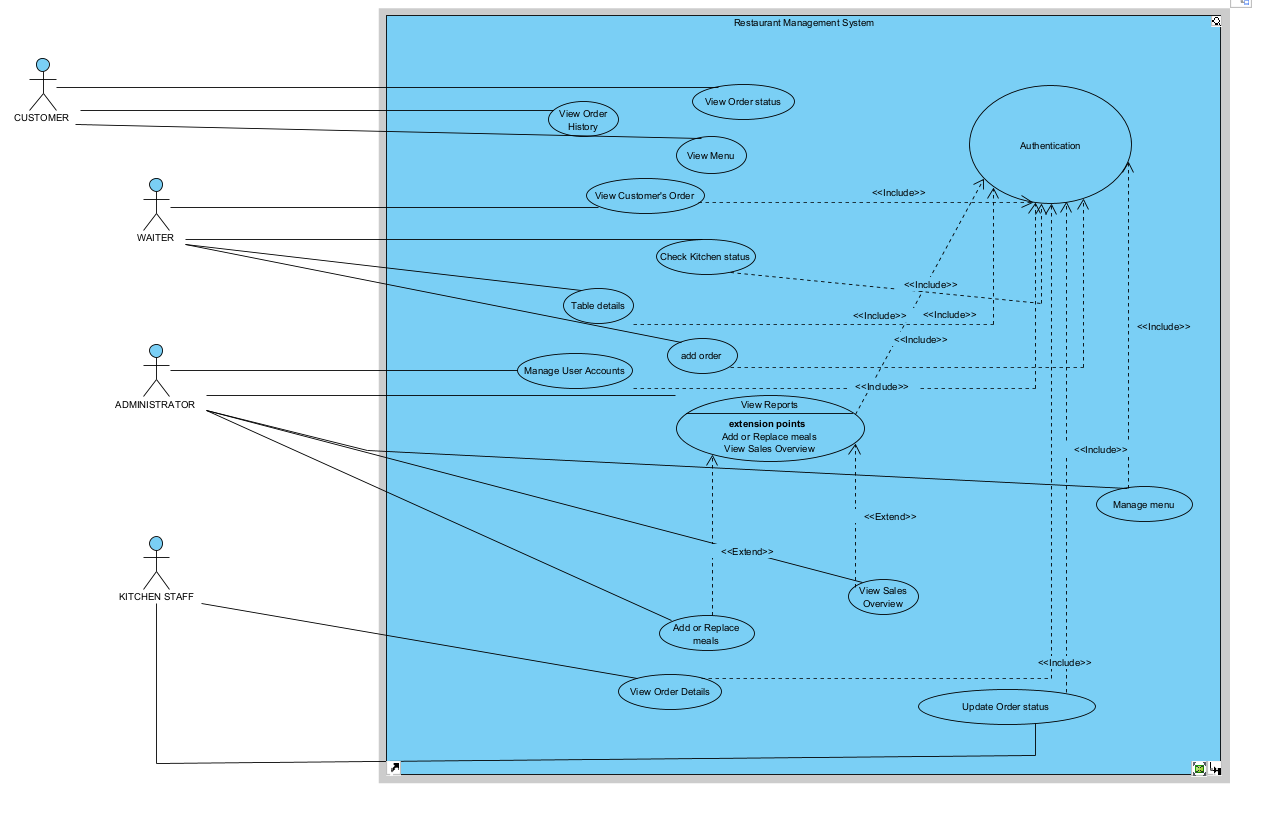
**Components Of the Use case Diagram**

|  |  |  |
| --- | --- | --- |
| ELEMENTS | DESCRIPTION | DIAGRAMMATIC REPRESENTATION |
| Actor | An actor in a use case diagram represents an external entity, such as a user, system, or device, that interacts with 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 | Extends represents a conditional relationship where one use case (the extending use case) adds additional behavior to another use case (the base use case) under specific conditions**.** |  |
| Include | Include represents a relationship where one use case (the base use case) always incorporates the behavior of another use case (the included use case) as part of its process. |  |
| Generalization | Generalization represents an inheritance relationship where a child use case or actor inherits the behavior and characteristics of a parent use case or actor. |  |

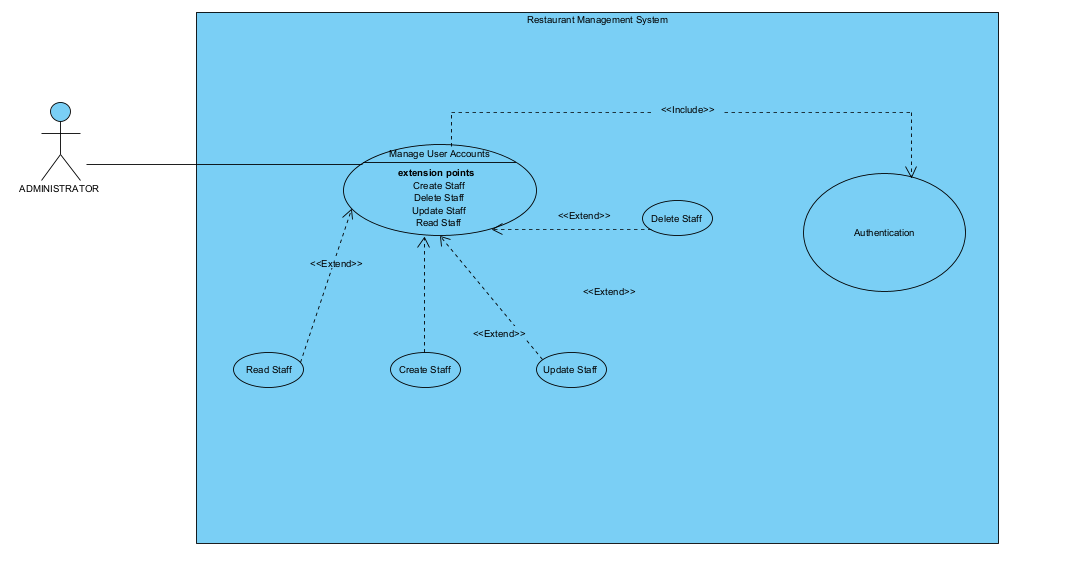
|  |  |  |
| --- | --- | --- |
| ELEMENTS | DESCRIPTION | DIAGRAMMATIC REPRESENTATION |
| Association | Association represents a relationship between an actor and a use case, showing how the actor interacts with the system. |  |
| Inheritance | Inheritance refers to a relationship where a child actor or use case inherits the properties and behaviors of a parent actor or use case**.** |  |

*Table 12: Components Of the Use case Diagram*

* LIST OF ACTORS
  + **Administrator:** Responsible for the system including updating the menu, handling orders, overseeing customer service, and monitoring restaurant operations, administration of users.
  + **Customer:** Interacts with the app to browse the menu, place orders.
  + **Waiter:** interacts with the app, place orders for people who want to have traditional restaurant services.
  + **Kitchen Staff:** Updates order status.



*Figure 5: General Use case Diagram*



*Figure 5: Specific Use case Diagram for staff Management*

**TEXTUAL DESCRIPTION**

**Definition**

A textual description of a use case is a detailed written explanation of how a system functions. It provides a comprehensive understanding of the system by outlining the interactions between various actors and the system, under specific conditions.

**Formalism**

|  |  |
| --- | --- |
| COMPONENT | EXPLANATION |
| Actors | Identify the primary actor (the entity initiating the use case) and any secondary actors (participants or systems that support or interact with the primary actor). |
| Description | Provide a brief, concise summary of the use case’s goal or main objective, outlining what the system is supposed to accomplish from the actor's perspective. |
| Preconditions | Outlines the necessary conditions that must be true or fulfilled before the use case can begin. This might include system states or actor prerequisites. |
| Postconditions | Define the expected state of the system once the use case has been successfully completed. This ensures that the intended goal has been achieved. |
| Nominal Scenario | Describe the typical sequence of events, step-by-step, when the use case is executed under normal conditions and everything proceeds as expected. |
| Alternative Scenario | Provide descriptions of possible deviations from the nominal scenario. This includes variations based on exceptions, errors, or alternate decisions by the actor. |
| Exceptions | Highlight specific conditions or errors that may prevent the use case from reaching its goal. Include how the system should respond to these exceptions. |
| Special Requirements | Detail any non-functional requirements (such as performance, security, or regulatory needs) or other special conditions necessary for this use case to function properly. |

**TEXTUAL DESCRIPTION FOR STAFF MANAGEMENT**

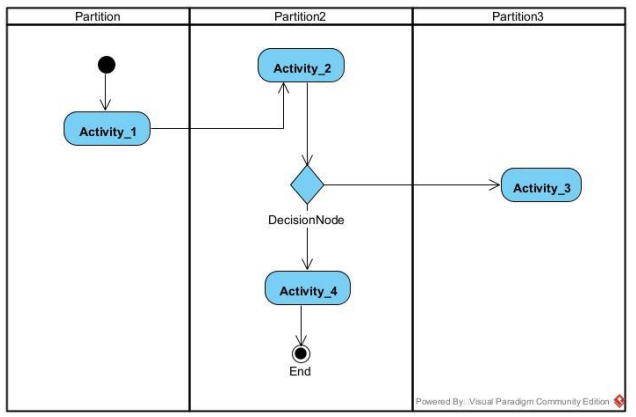
|  |  |
| --- | --- |
| COMPONENT | EXPLANATION |
| Actors | The **primary actor** is the **Administrator**, who manages user accounts. There is no secondary actor explicitly shown in the diagram, but the system itself plays a role in processing requests. |
| Description | The diagram illustrates how the administrator manages the staff by creating, reading, updating, and deleting staff accounts. It includes **authentication** as part of the system to control access. |
| Preconditions | The administrator must be authenticated to access the system and perform operations related to managing user accounts. The system must be functional and accessible. |
| Postconditions | Once the use case is completed, the staff records will be updated according to the administrator’s actions (e.g., adding, deleting, or modifying staff). |
| Nominal Scenario | * + 1. The administrator logs into the system using **authentication**.     2. The admin chooses to manage user accounts (staff).     3. They create, read, update, or delete staff details. |
| Alternative Scenario | In case of authentication failure, the administrator won’t be able to access the system or perform any operations on user accounts. |
| Exceptions | If there is an error during the operation (e.g., database unavailability or system crash), the administrator won’t be able to manage user accounts, and error messages will be displayed. |
| Special Requirements | The system should meet security requirements for **authentication** to ensure that only authorized personnel can manage user accounts. Performance efficiency is essential to handle large amounts of data. |

* + - * 1. ACTIVITY DIAGRAMS

**Definition**

These are visual representations of workflows or processes that outline the sequence of activities involved in completing a task or achieving an objective. They illustrate how different actions are performed and how decisions or conditions influence the flow from one activity to the next.

**Formalism**



*Figure 6: Formalism of an activity diagram*

COMPONENTS OF THE ACTIVITY DIAGRAM

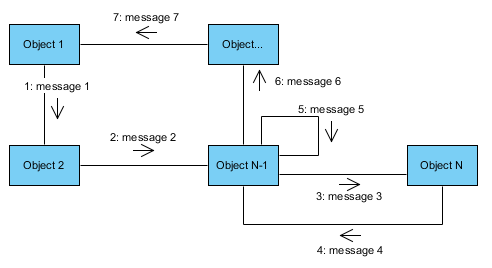
|  |  |  |
| --- | --- | --- |
| ELEMENTS | DESCRIPTION | NOTATION |
| Initial node | Represents the starting point of a set of actions or activities within an activity diagram. |  |
| Activity | Denotes a set or sequence of actions to be performed, typically used to encapsulate larger processes or behaviors within the system. |  |
| Action | Refers to a specific operation or task that is to be executed, forming part of an activity. |  |
| Control flow | Illustrates the order in which actions or activities are carried out within the process. |  |
| Object flow | Shows the path of an object as it moves from one activity to another within the process. |  |
| Object node | Represents an object in the activity diagram, often connected to object flows, indicating the role of objects in the workflow. |  |
| Decision node | Depicts a branching point where a condition is tested to determine which path the control or object flow should follow. |  |
| Merge node | Combines different paths that were previously separated by decision nodes, consolidating them into a single flow. |  |
| Fork node | Splits a single flow into multiple concurrent or parallel flows of activities, enabling actions to occur simultaneously. |  |
| Join node | Reunites multiple concurrent or parallel flows back into a single flow, ensuring synchronization of activities or actions. |  |
| partition | Organizes activities by actor or role, or separates activities into threads, highlighting who performs which actions or in which thread they occur. |  |
| Activity final node | Terminates all ongoing control or object flows, marking the completion of an activity or process within the diagram. |  |

* + - * 1. COMMUNICATION DIAGRAMS

**Definition**

A communication diagram in UML visualizes the interactions between objects, emphasizing their relationships and the flow of messages exchanged to perform tasks. Unlike sequence diagrams, which focus on the timing of interactions, communication diagrams highlight the structure and connections between objects, providing a clear view of how components collaborate within a system.

**Formalism**



COMPONENTS OF THE COMMUNICATION DIAGRAM

|  |  |  |
| --- | --- | --- |
| ELEMENT | DESCRIPTION | NOTATION |
| Frame | The overall use case or sequence is represented within a rectangular frame. The name of the diagram is placed in a separate compartment at the upper-left corner for identification. | A white rectangular object with black text  Description automatically generated |
| Object | Represents an entity involved in the communication process. Objects participate by sending or receiving messages and are typically labeled with their name and role (e.g., User). |  |
| Message | An arrow between two objects that illustrates the flow of communication, specifying the direction and sequence of the interaction. Messages are labeled with a number and a description to indicate the order and type of communication. |  |
| Link | A straight line connecting two objects, symbolizing an established relationship or connection between them. This indicates that the objects can communicate with each other. |  |

* + - * 1. SEQUENCE DIAGRAM

**Definition**

A sequence diagram visually represents how system components interact over time, showing the order of events and communications between them. With vertical lines depicting object lifetimes and horizontal arrows for interactions, it provides a clear, chronological view of processes, helping to understand system functionality and flow.

**Formalism**

**A diagram of a diagram of a lifeline

Description automatically generated**

COMPONENTS OF THE SEQUENCE DIAGRAM

|  |  |  |
| --- | --- | --- |
| ELEMENTS | DESCRIPTIONS | NOTATION |
| Actor | Represents an external entity, typically a user, that interacts with the system. Illustrated as a stick figure, the actor initiates actions or receives outcomes from the system. |  |
| Lifeline | A vertical dashed line extending from an object or actor, symbolizing the entity’s presence over time within the sequence diagram. It represents the duration for which the object or actor is active. |  |
| Message | A horizontal arrow illustrating communication between objects or actors. The direction of the arrow indicates the sender and receiver of the message, marking the path of interaction. |  |
| Return Message | A dashed arrow indicating the response sent back to the original object or actor after a message. It typically follows an action or request and is used to show feedback or result. |  |
| Activation Bar | A thin vertical rectangle placed on the lifeline, representing the duration in which an object or actor is actively processing a message or task. It indicates the time frame of execution. |  |
| Destroy | A large "X" at the end of a lifeline, showing the point at which an object is terminated or ceases to exist. It marks the end of the object’s involvement in the process. |  |
| Combined Fragment | A rectangular area used to represent conditions, loops, or control structures in the diagram. It often includes labels such as ‘alt’ for alternative paths or ‘loop’ for repetitive actions, and groups related lifelines. |  |
| Note | A comment or annotation box attached to elements in the diagram to provide additional context or clarification. It serves as a means of offering supplementary information about specific parts of the sequence. |  |

**CONCLUSION**

As we conclude the analysis phase for the Restaurant Management App, we've outlined the core functionalities and interactions using UML diagrams. This solid blueprint will guide us into the design phase, ensuring a user-friendly system and a smooth development process.

**PART IV:**

**CONCEPTUAL PHASE**

Preamble

The conception phase, or design phase, translates requirements into clear blueprints, defining architecture, database design, and user interfaces. It ensures the product is user-friendly, secure, and scalable, providing developers with a structured roadmap for a smooth development process.

Content

INTRODUCTION

1. TECHNICAL BRANCH

* Generic Design

1. IMPLEMENTATION BRANCH

* Preliminary Design
* Detailed Design

CONCLUSION

**INTRODUCTION**

The conception phase is a key transition in software development, moving from abstract requirements to detailed design. Using the 2TUP methodology, this phase splits into two branches: technical, which outlines the system architecture, and implementation, which focuses on detailed execution. Together, they form a clear roadmap for smooth development.

* 1. TECHNICAL BRANCH
* **Generic Design**  
  This stage of the technical branch focuses on creating a high-level design blueprint of the software without getting into detailed implementation. It outlines the core structure and overall architecture of the application.
  + 1. **Physical Architecture**

The physical architecture defines the system's components and how they interact. Our app follows a **3-tier architecture**, with the layers described below:

* **Presentation Tier**: This is the front-end of the application, where users interact via web browsers. In our case, it's built with **HTML, Bootstrap, and CSS**, offering a responsive and user-friendly interface.
* **Application Tier**: This tier contains the core logic, processes requests, and applies business rules. Our app’s back-end is developed in **PHP**, managing requests between the user interface and the database.
* **Data Tier**: This layer handles data storage and retrieval. We use a **MySQL database** for storing restaurant details, user information, orders, and more.
  + 1. **Logical Architecture**

The logical architecture focuses on internal software design and the patterns used within the code structure. Our app uses the **MVC (Model-View-Controller)** design pattern, breaking the system into three primary components:

* **Model**: Represents the data structure, interacting with the database to manage data operations. In our app, this is where the business logic is applied, linking to the **MySQL database**.
* **View**: This is where data is presented to the user. Built using **HTML, Bootstrap, and CSS**, the view layer displays the restaurant menu, orders, and other details to the user.
* **Controller**: The controller manages user input, handling requests and coordinating between the model and the view. In our case, **PHP** serves as the controller, processing user inputs like orders and managing the flow of information.
  1. IMPLEMENTATION BRANCH

The Implementation Branch begins with the Preliminary Design, where UML diagrams like Class Diagrams and Component Diagrams are used to visualize the system’s structure and high-level components. These models help guide the early stages of development by providing a blueprint for implementation.

* PRELIMINARY DESIGN

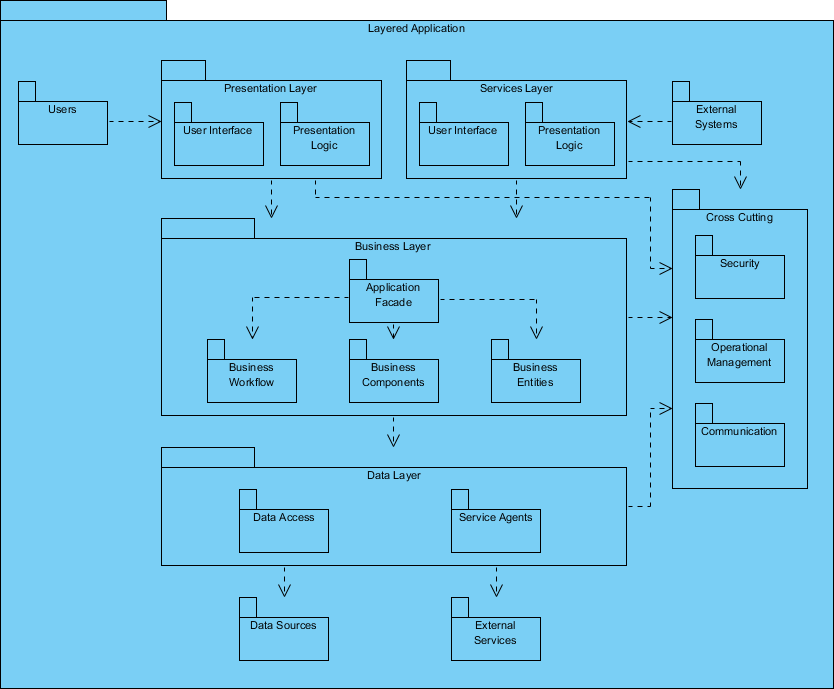
This phase serves as the foundation for refining system requirements and guiding subsequent development efforts.

* + 1. PACKAGE DIAGRAM

**Definition**

A Package Diagram in UML showcases how system components or classes are organized into groups (packages) and their dependencies. It offers a high-level modular view of the system's architecture, helping to identify shared components, minimize redundancies, and clarify the overall structure.

**Formalism**



**Component of a package diagram**

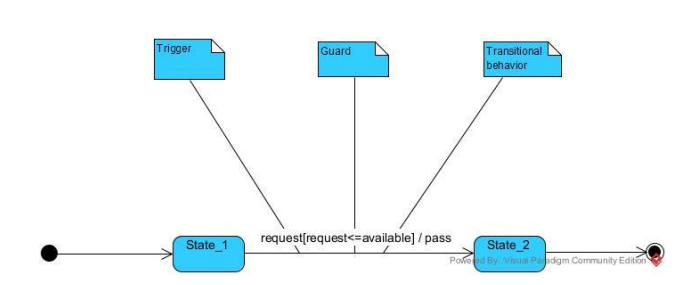
|  |  |  |
| --- | --- | --- |
| ELEMENTS | DESCRIPTION | NOTATION |
| Package | A namespace used to group semantically related elements, similar to a folder. Packages can contain other packages, creating a hierarchical structure. |  |
| Dependency | A relationship where one package depends on another. Represented by a dashed arrow, it indicates that changes in the target package may affect the dependent package. |  |
| Package import | Indicates that all publicly visible members of a package are accessible in another package. |  |
| Generalization | Represents inheritance between packages, allowing one package to inherit characteristics or behavior from another. |  |
| Stereotypes | |  | | --- | |  |  |  | | --- | | Labels or tags added to packages to provide additional semantic meaning. Common examples include <<utility>>, <<system>>, or <<subsystem>>. | | // |

* + 1. STATE MACHINE DIAGRAM

**Definition**

A state machine diagram serves as a visual representation of a system's behavior. It identifies the different states or conditions that the system can enter and illustrates how transitions occur from one state to another based on specific triggers. This diagram is designed to enhance understanding of complex systems. By mapping out the system's behaviors, designers, developers, and users can anticipate its responses, facilitating more effective design, troubleshooting, and interaction with the system.

**Formalism**



|  |  |  |
| --- | --- | --- |
| ELEMENT | DESCRIPTION | NOTATION |
| Initial State | It defines the initial state (beginning) of a system, and it is represented by a black filled circle |  |
| State-box | 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 | A blue rectangle with black text  Description automatically generated |
| Decision-box | It is of diamond shape that represents the decisions to be made based on an evaluated guard. | A blue diamond with black outline  Description automatically generated |
| 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 |  |

**Components of the State Machine Diagram**

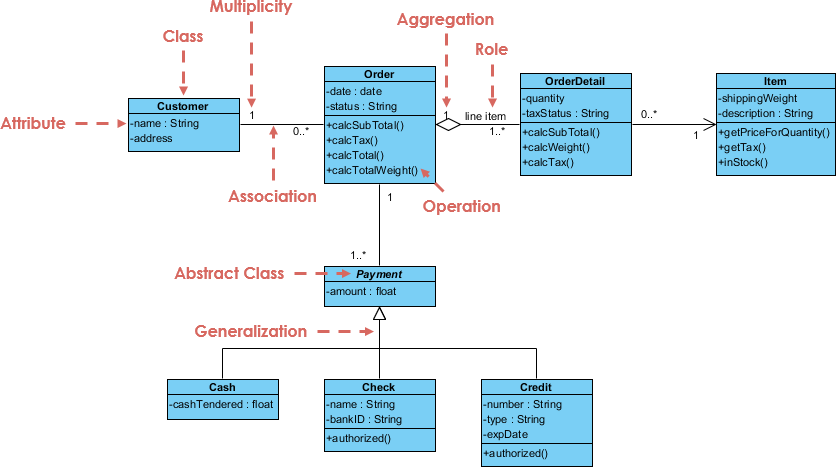
* DETAILED DESIGN

1. CLASS DIAGRAM

**Definition**

A class diagram is a static structure diagram within the Unified Modeling Language (UML) that depicts the structural patterns of a system. It illustrates the relationships between classes and how they interact with one another. Essentially, it serves as a blueprint for the system's architecture.

**Formalism**



**Components of the Class Diagram**

|  |  |  |
| --- | --- | --- |
| Elements | Description | Notation |
| Class | Symbolized by a rectangle divided into three sections: the top section for the Name, the middle for Attributes, and the bottom for Operations/Methods. Represents entities or objects along with their characteristics. | A blue box with black text  Description automatically generated |
| Association | Represented by a solid line connecting two classes, indicating a relationship between them. |  |
| Generalization (Inheritance) | Depicted by a hollow triangle pointing to the parent class, indicating an "is-a" relationship between the subclass (child class) and the superclass (parent class). | A diagram of a diagram  Description automatically generated |
| Aggregation | Represented by a hollow diamond on the side of the whole, indicating a "whole-part" relationship where the part can exist independently of the whole. | A diagram of a diagram  Description automatically generated |
| Composition | Shown with a filled diamond on the side of the whole, indicating a stronger "whole-part" relationship where the part cannot exist separately from the whole. | A diagram of a diagram  Description automatically generated |
| Dependency | Depicted by a dashed line with an arrow, indicating that one class relies on another. For example, in the diagram, Class1 depends on Class2. | A diagram of a connection between two squares  Description automatically generated |
| Visibility | Indicated by symbols (+, -, #, ~) representing public, private, protected, and package-level visibility, respectively. |  |
| Multiplicity | Represents the number of instances of one class that can be associated with a single instance of another class, expressed as cardinal numbers (e.g., 1, 2) or ranges (e.g., 0..1, 1..\*), typically found at the ends of associations. |  |

**CONCLUSION**

In the conception phase, we created the architectural framework and structural blueprint for our application, ensuring a seamless integration of user experience and backend functionalities. Utilizing the 2TUP model and UML diagrams, we achieved a clear understanding of the system's behaviors and state transitions through class, state machine, and package diagrams. With this solid design framework in place, we are now prepared to enter the realization phase, where we will transform these designs into functional modules for our innovative card management application.

**PART VI**

**REALIZATION PHASE**

Preamble

In the software development journey, each step is crucial in shaping the final product. After navigating the complexities of conceptualization and design, we are now on the brink of the realization phase, where our ideas transform into tangible components. This phase reveals the interplay of software modules with hardware, laying the groundwork for a seamless user experience. As we move forward, we will dissect our application, demonstrating how each element contributes to the cohesive whole of our digital solution.

Content

INTRODUCTION

1. COMPONENT DIAGRAM
2. DEPLOYMENT DIAGRAM
3. PRESENTATION OF THE TOOL USED
4. TECHNOLOGY STACK
5. ARCHITECTURE OF THE APPLICATION

CONCLUSION

**INTRODUCTION**

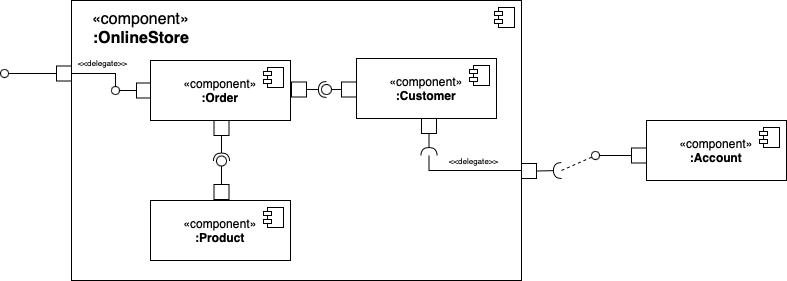
In this phase, we are focusing on two key tools: the Component Diagram and the Deployment Diagram. Think of the Component Diagram as our application's family tree, illustrating each part and their connections. Meanwhile, the Deployment Diagram serves as a launch day game plan, detailing where everything will be positioned and how it will all connect. By the end of this phase, we will have a clear strategy that sets the stage for the upcoming development. Let’s dive in and bring our vision to life!

* 1. COMPONENT DIAGRAM

**Definition**

A Component Diagram is a type of structural diagram in the Unified Modeling Language (UML) that visualizes the organization and dependencies among software components. These components may include classes, interfaces, or entire software packages and modules. The diagram offers a high-level overview of a system's architecture, illustrating how software components interact and integrate within the overall system.

**Formalism**



**Components of the Component Diagram**

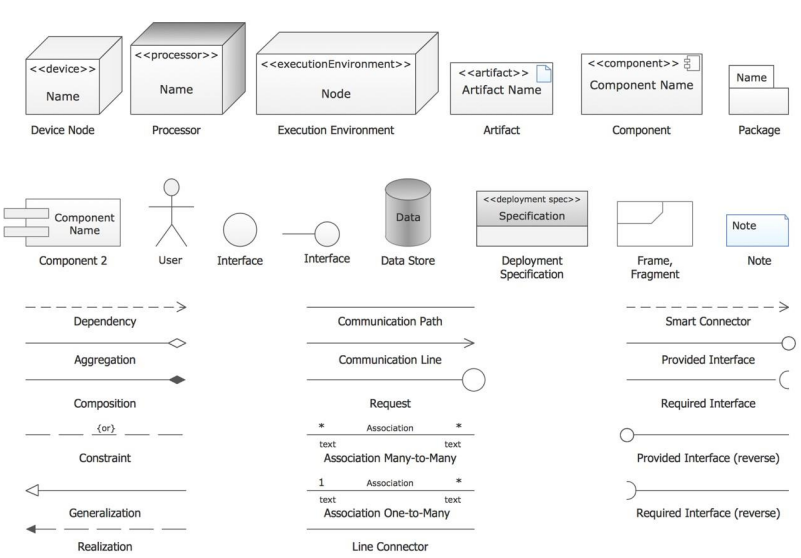
|  |  |  |
| --- | --- | --- |
| ELEMENTS | DESCRIPTION | NOTATION |
| Component | Represented as a rectangle with the component's name inside, often with an icon of two smaller rectangles on the left. Symbolizes a modular part of the system that encapsulates specific functionality. |  |
| Interface | Depicted as a circle (or "lollipop" symbol), sometimes as a semi-circle attached to a component. Defines a set of operations that outline a component's responsibilities. |  |
| Dependency | Illustrated with a dashed arrow, indicating that one component relies on another for proper functioning. |  |
| Port | Shown as a small square on the edge of a component. Represents an entry or exit point for data or control flow. |  |
| Connector | Depicted as a solid line between two components or ports, representing a communication path between them. |  |

* 1. DEPLOYMENT DIAGRAM

**Definition**

A Deployment Diagram is a specialized type of UML diagram that focuses on a system’s physical nodes. It visually represents the deployment of artifacts and nodes, detailing the hardware components and how software components installed on them interact. Essentially, it illustrates how the system's software runs on the hardware and how these components communicate with one another.

**Formalism**



**Description of some components of the Deployment Diagram**

|  |  |
| --- | --- |
| ELEMENTS | DESCRIPTION |
| Node | Represents physical hardware where software runs, depicted as a three-dimensional box. |
| Artifact | Represents a tangible piece of software residing on a node, such as executables, libraries, archives, database tables, or configuration files. |
| Communication path | Illustrates the flow of information between nodes, showing how they interact within the system. |
| Dependency | Depicted as a reliance of one element (node or artifact) on another for functionality or deployment. |
| Association | Represents a relationship between two nodes, indicating communication or connectivity between them. |
| Component | Although not exclusive to deployment diagrams, components represent functional parts of a software system. In deployment diagrams, they show high-level software deployed on nodes. |
| Package | A logical grouping of related elements, such as nodes or artifacts, to provide better clarity within the deployment diagram. |

* 1. PRESENTATION OF TOOL USED

|  |  |  |
| --- | --- | --- |
| SOFTWARE | USAGE | LOGO |
| Microsoft 365 | Used for the creation our report and PowerPoint |  |
| Visual studio code | Open and free software for the coding part of the project |  |
| PowerAMC | For doing the design of the diagrams to represent the structure and the interaction between the different actors |  |
| Google | For doing the research on the internet and testing of the application |  |
| Windows 11, version 23H2 | Operating system |  |
| Gantt Project | For building a gantt chart for the planning of the work |  |
| XAMPP | For creating database |  |
| Visual Paradigm Enterprise version | For doing the design of the diagrams to represent the structure and the interaction between the different actors |  |

1. SOFTWARE RESOURCES
2. HARDWARE RESOURCES

|  |  |
| --- | --- |
| MATERIAL | USAGE |
| LAPTOP (LENOVO i5 8th Gen, SSD) | Used to build the system, type the report, create the power point, make research, etc. |
| Internet Modem | Used for internet connection |
| Samsung Tablet | Used for carrying research and storage |
| Printer HP Officejet 4630 | Used to print the scientific document |

* 1. TECHNOLOGY STACK

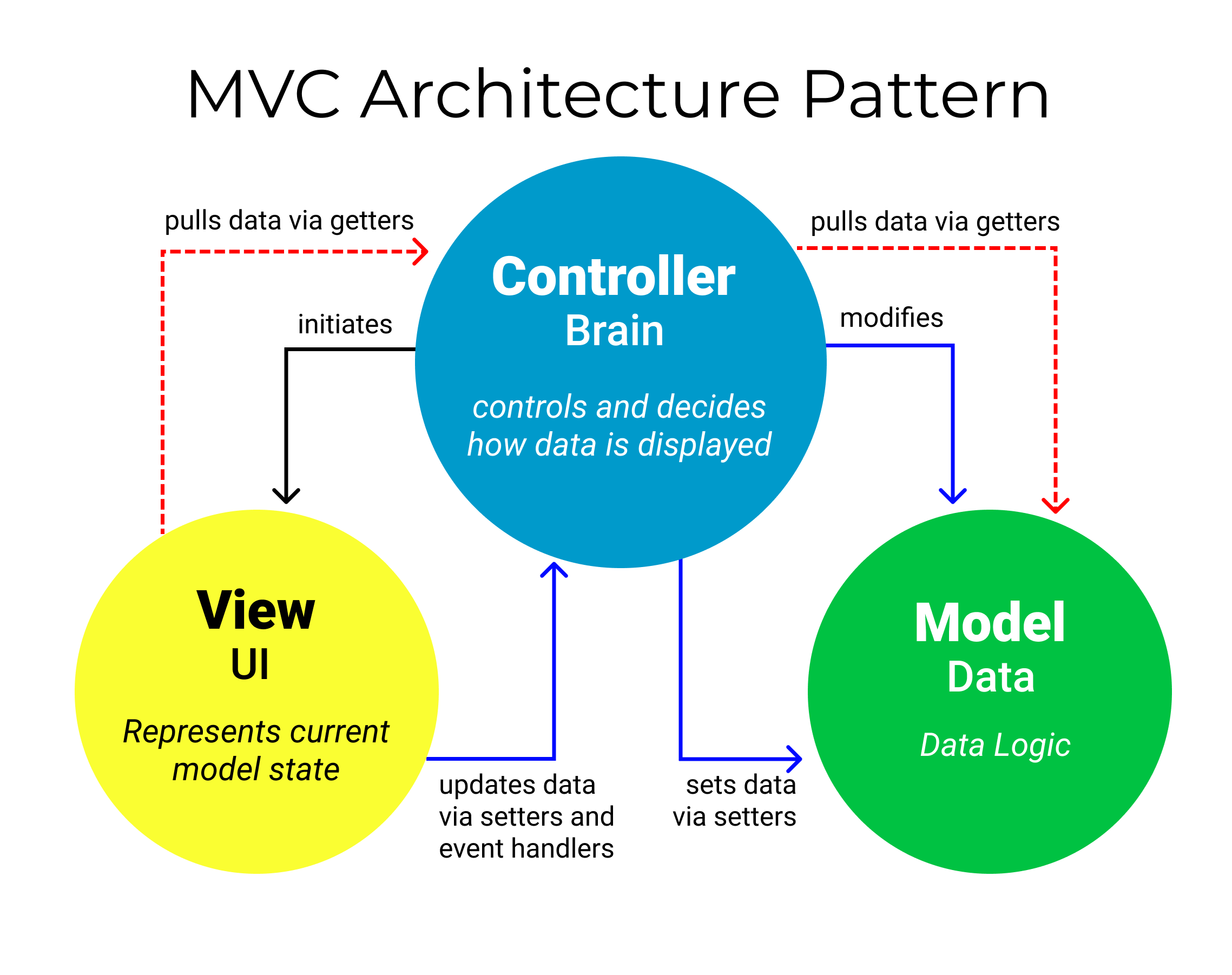
|  |  |
| --- | --- |
| Software | Roles |
| HTML/CSS (Bootstrap) | These are used for building and styling the front-end of the restaurant app, ensuring a responsive and user-friendly interface. **Bootstrap** provides pre-designed UI components for faster front-end development. |
| PHP | PHP is used as the server-side scripting language for processing user inputs, handling orders, and managing the communication between the front-end and the database. |
| MySQL | A relational database management system used to store and retrieve restaurant data, such as menus, user profiles, orders, and transaction records. |
| UML | Unified Modeling Language (UML) is employed to model and visualize system diagrams during the analysis and design phases of the restaurant app development. |
| PostgreSQL | The DBMS of my application to store and retriev e |
| |  | | --- | | **JavaScript** |  |  | | --- | |  | | Enhances interactivity on the front end, like form validation, dynamic content updates, and handling asynchronous requests. |
| Ajax | Used to make asynchronous calls between the client (front-end) and server (back-end) to update parts of the webpage without refreshing the entire page. |
| Apache | A web server that hosts the restaurant app, handling HTTP requests and serving web pages to users. |

* 1. ARCHITECCTURE OF THE APPLICATION
     1. LOGICAL ARCHITECTURE

The logical architecture defines the overall structure of the software system, including the design pattern used and how different components of the system interact.

**Design Pattern: MVC (Model-View-Controller)**

* **Model:**  
  This layer handles the database and data-related logic. It communicates with the database to fetch, save, or modify restaurant data (such as menu items, user profiles, and order history).
* **View:**  
  The view layer is responsible for the presentation and user interface. The front-end will be developed using **HTML, CSS (Bootstrap)**, and **JavaScript** to create responsive layouts that display the restaurant's menu, handle ordering, and display user profiles.
* **Controller:**  
  The controller acts as the middleman between the Model and View. It processes incoming requests (like making an order or updating a user profile) and sends the appropriate data to the Model. It also selects the correct View to display the data to the user.



* + 1. PHYSICAL DATA MODEL

The **physical data model** represents how the data is stored and managed within the application, typically in the form of a database schema. Since this restaurant app uses **MySQL**, the physical data model includes the following key tables and their relationships:

* **Users Table**: Stores customer and admin information, such as username, password, contact details, and role (customer or admin).
  + Columns: user\_id (PK), name, email, password, role, created\_at
* **Menu Table**: Stores details about the items available in the restaurant.
  + Columns: menu\_id (PK), item\_name, description, price, category
* **Orders Table**: Records the orders placed by customers.
  + Columns: order\_id (PK), user\_id (FK), menu\_id (FK), quantity, total\_price, order\_status, created\_at

**CONCLUSION**

**In this phase, we completed the UML diagrams and introduced the architecture on which our application is built. We also outlined the tools and programming language used in its development.** Now, we are moving forward to conduct functionality testing for the application.

**CONTENT**

[ACKNOWLEDGEMENTS 2](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268886)

[DEDICATION 3](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268887)

[CONTENT 4](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268888)

LIST OF ABBREVIATIONS,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,5

[LIST OF TABLES i](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268889)

[LIST OF FIGURES ii](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268890)

[GLOSSARY iii](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268891)

[ABSTRACT iv](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268892)

[RESUME 1](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268893)

[GENERAL INTRODUCTION 2](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268894)

[PART I 3](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268895)

[INSERTION PHASE 3](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268896)

[INTRODUCTION 5](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268897)

[Conclusion 16](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268898)

[PART II 17](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268899)

[EXISTING SYSTEM 17](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268900)

[INTRODUCTION 20](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268901)

[CONCLUSION 26](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268902)

[SPECIFICATION BOOK 27](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268903)

[INTRODUCTION 29](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268904)

[CONCLUSION 42](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268905)

[ANALYSIS PHASE 43](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268906)

[INTRODUCTION 45](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268907)

[CONCLUSION 89](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268908)

[CONCEPTION PHASE 90](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268909)

[INTRODUCTION 92](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268910)

[CONCLUSION 105](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268911)

[REALIZATION PHASE 106](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268912)

[INTRODUCTION 108](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268913)

[CONCLUSION 117](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268914)

[Functionality Test 118](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268915)

[INTRODUCTION 120](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268916)

[USER GUIDE 121](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268917)

[USER GUIDE 122](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268918)

[INTRODUCTION 123](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268919)

[GENERAL CONCLUSION 135](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268920)

[Annexe 136](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268921)

[BIBLIOGRAPHY 137](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268922)

[WEBOGRAPHY 138](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268923)

[TABLE OF CONTENT 139](file:///C:\Users\user\Desktop\FINAL%20FILES\INTERNSHIP%20REPORT.docx#_Toc176268924)