|  |  |
| --- | --- |
| REPUBLIC OF CAMEROON  Peace - Work - Fatherland | REPUBLIQUE DU CAMEROUN  Paix – Travail - Patrie |
| C:\Users\lenovo\Downloads\school docs\iai logo.jpg |  |
| **AFRICAN INSTITUTE OF COMPUTER SCIENCES - CAMEROON OFFICE**  **PAUL BIYA TECHNOLOGICAL CENTER OF EXCELLENCE**  P.O. Box: 13719 Yaoundé  Tel: +237.242.729.957 Fax: 22729958  E-mail: contact@iaicameroun.com  Web site: www.iaicameroun.com | **REALIZE**  Tel: (237) 620256960/654486985  Email : realize.contactus@gmail.com  Web site: [www.realize.cm](http://www.realize.cm)  **THEME:** **CONCEPTION AND IMPLEMENTATION OF A WASTE MOBILE APP COLLECTION SYSTEM** |

**INTERNSHIP REPORT**

Internship carried out from the 03rd of July till the 30th of September 2024. In view of obtaining a Software Engineering Diploma in Computer Sciences Option: Software Engineering

Written by:

**NSAN SAMUEL PENIEL**

Level II Software Engineering student

Supervision:

**Academic Supervisor:**

Mr. ASSANE DERICK

*Lecturer at AICS-Cameroon*

AICS-Cameroon

**Professional Supervisor:**

Mr. MENGOT NSO

*Software Engineer*

AICS-Cameroon

**Academic Year 2024 - 2025**

0

DEDICATION

This work is dedicated to the Dibango’s Family for their love, support, encouragement and their unconditional sacrifices towards my academic success.

ACKNOWLEDGEMENTS

As a prelude to this internship report, we would like to express our sincere gratitude to all individuals and entities whose guidance, support, and encouragement have contributed to the successful completion of this work and to a productive academic year. We are particularly grateful to:

* **Mr. ARMAND CLAUDE ABANDA**, the Resident Representative of the African Institute of Computer Sciences, for his wise and invaluable instructive advice.
* **Mr. NDELOGAKEH DANIEL,** the Chief Executive Officer of REALIZE, and his staff for their trust, warm welcome, encouragement, availability, and moral support throughout our internship period.
* **Mr. ASANE DERICK**, our academic supervisor and lecturer at AICS Cameroon, for his constant guidance, advice, and moral support during our academic and internship journey, which contributed to the realization of this project.
* **Mr. MENGOT NSO**, our professional supervisor at REALIZE, for his help, advice, and encouragement that greatly supported the successful development of this project.
* The **Administrative and Teaching Staff of AICS Cameroon**, for their dedication, teaching, and guidance throughout the 2023-2025 academic year.
* Our **families and parents**, for their spiritual, moral, and financial support, which was instrumental in enabling us to complete this work.
* Lastly, all other individuals and organizations who contributed in one way or another to the realization of this project

CONTENT

TABLE OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABSTRACT

RESUME

GENERAL INTRODUCTION

In today’s fast-paced world, technological advancements are transforming various sectors, including waste management. Digital innovations are helping cities and enterprises improve efficiency, reduce costs, and address environmental challenges. As a developing country, Cameroon is increasingly embracing the digital economy to boost growth and sustainability.

To contribute to this progress, students at AICS Cameroon are required to undertake a three-month internship to apply their academic knowledge in real-world scenarios. During our internship at **REALIZE**, an IT solutions company, we were assigned the theme: **“Conception and Implementation of a Waste Collection System.”**

Our project aims to design and develop a digital platform that optimizes waste collection processes, enhances efficiency, and promotes better environmental management. To achieve this, we conducted comprehensive research and structured our work into eight parts:

1. **Insertion Document:** Introducing the company and our internship experience, along with the project theme.
2. **Existing System:** Analyzing current waste collection methods.
3. **Specification Book:** Defining user needs, considering costs and deadlines.
4. **Analysis Document:** Presenting analysis methods and diagrams for understanding the system.
5. **Conception Phase:** Designing the overall and detailed architecture of the waste collection system.
6. **Realization Phase:** Demonstrating how the system will be built and implemented.
7. **Functionality Testing:** Highlighting the modules and how they operate.
8. **User Guide:** Providing instructions for users to operate the system effectively.

PART 1

INSERTION PHASE

Preamble

This report summarizes the company, the project we worked on, the tasks we completed, the tools used, and the results achieved during the internship.

Content Overview

INTRODUCTION

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

CONCLUSION

INTRODUCTION

Insertion is the way in which man can integrate a community, a group. Within the

framework of the academic internship, the insertion phase is a period in which the different

Interns have to know more about their host company. Additionally, we have to know about the staffs, the different hardware and software used, the different department, which constitute the company, and we were introduced to our workspace, how the company functions both internally (that is how the different task is schedule, the rules and regulations, working periods and so on). We also had a time to discuss amongst us interns on topics like what we love doing most, what we dislike, our believes, our best meals, sports, songs, our temperaments, our inborn talents and those we learn as we grow up. We shared about different realizations and failures in life.

1. WELCOME AND INTEGRATION

We arrived at REALIZE on Thursday, July 3, 2025, at 08:00. Greeting us were **Mr. TANUE MONET**, the Software Designer, and **Mr. MENGOT NSO**, the Senior Developer, who delivered an inspiring welcome and introduced the company’s operations and scheduling for our internship period. The initial meeting covered the challenges from the coding test required for the internship, REALIZE’s organizational structure and internal policies, and the allocation of workstations to each intern.

First-week activities: We set up at our designated workstations and received our initial task. A foundational verification of concepts using HTML and CSS ensured all interns started on an even footing. We brainstormed project ideas, with an emphasis on individual initiative; prior experience helped me contribute and refine these ideas. We conducted research on the proposed ideas, as REALIZE encouraged initiative and did not prescribe themes.

Second-week activities: JavaScript concept verification exercises were used to assess understanding and reasoning. A detailed discussion with the supervisor reviewed the specification book and its components, clarifying expectations for each section. Project ideas were further developed into concrete plans, leveraging familiarity with REALIZE’s expectations and processes.

1. GENERAL PRESENTATION OF REALIZE

### A. History

Realize is a Cameroon-based NGO and tech start-up founded by **Mr. NDELOGAKEH Daniel** in 2022. It offers IT solutions and promotes the use of new technologies in Cameroon.

### B. What is REALIZE all about?

#### i. Missions

Realize mission is to empower and sustain the use of technology through:

* Designing mobile software for businesses and individuals
* Supporting startups for rapid growth using IT
* Providing training and certifications to build qualified human resources
* Contributing to global sustainable development with innovative solutions and virtual reality

#### ii. Vision

Realize believes that unlocking the potential of organizations and individuals is possible through analysis, development, realization, and data analytics. Their vision is to shape a future of limitless innovation in the digital world.

#### iii. Activities

Realize activities include:

* Designing and hosting websites
* Software development and maintenance
* Providing training in software fields
* Creating multimedia content
* Offering IT consulting, support, and innovation.

##### C. GEOGRAPHICAL LOCATION

REALIZE is located at Yaoundé-Valle Amadou

*Figure 1: Realize location*

1. ORGANISATION OF REALIZE

### A. Administrative Organization

**Realize’s administrative structure is divided into several departments:**

#### 1. General Management

* Oversees the entire company
* Ensures departments function properly
* Sets project strategies
* Provides leadership and guidance
* Makes critical decisions impacting operations and reputation
* Acts as a point of contact for stakeholders like investors

#### 2. Human Resources

* Handles recruitment and hiring
* Acts as a link between employees and management
* Manages salaries, benefits, and employee policies
* Maintains employee records and HR systems

#### 3. Communication Department

* Manages public relations and company image
* Handles messaging to external stakeholders (customers, partners)
* Ensures effective internal communication
* Creates and manages content for various platforms

#### 4. Financial Affairs

* Develops and manages budgets and financial plans
* Prepares financial reports for management and authorities
* Controls costs and expenses
* Ensures tax compliance

#### 5. Technical Department

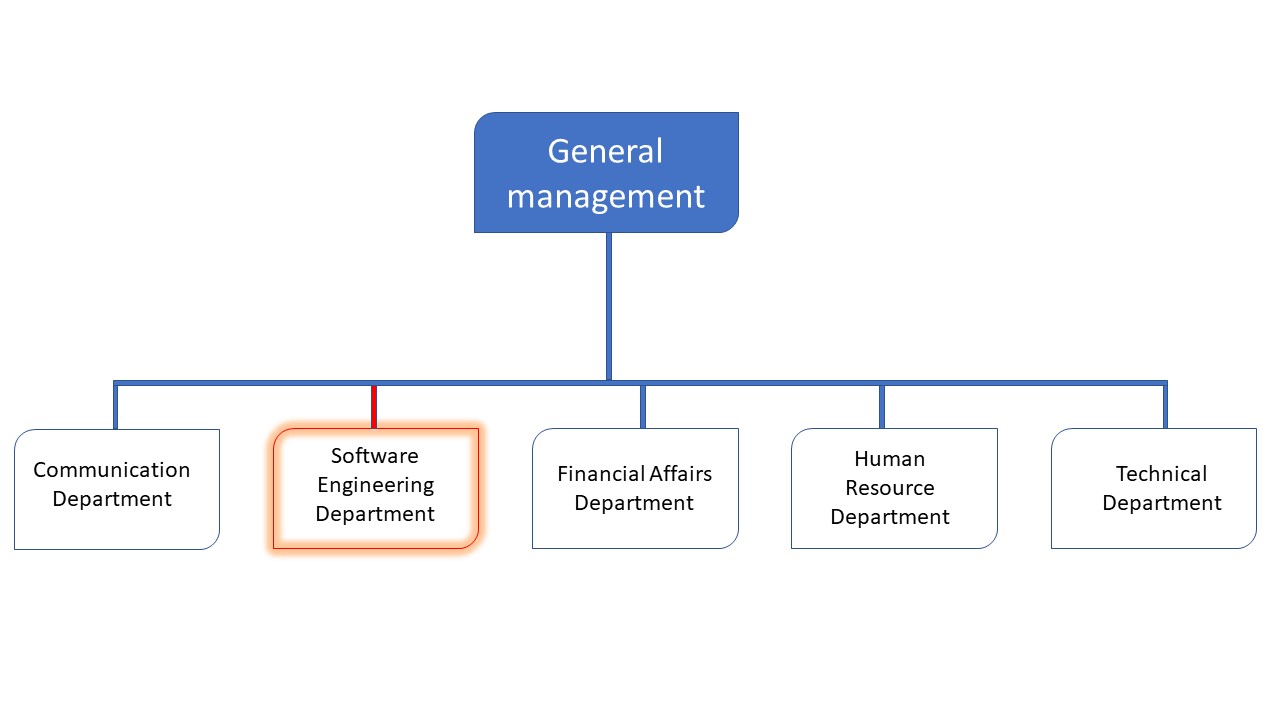
* Manages infrastructure like networks and hardware
* Handles data management and cybersecurity
* Maintains relationships with vendors and service providers

#### 6. Software Engineering Department

* Develops and supports software applications
* Maintains software throughout its lifecycle
* Executes and evaluates project realization

##### B. FUNCTIONAL ORGANISATION OF REALIZE

The functional branch of realize is organized as follows,



*Figure*

*2*

*:*

*Company functional organization*

*(*

*Source: Realize*

*)*

1. HARDWARE AND SOFTWARE RESOURCES OF REALIZE
2. **HARDWARE RESOURCES**

*Table 1: Hardware resources (Source from REALIZE)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **DESIGNATION** | **QUANTITY** | **CHARACTERISTICS** | **OBSERVATION** |
| 1 | MacBook Pro | 1 | APPLE | Good |
| 2 | Modem | 1 | CAMTEL | Good |
| 3 | LCD Screen | 1 | HP | Excellent |
| 4 | Training  equipment and  office furniture | / | / | / |

1. **SOFTWARE RESOURCES**

*Table 2: Software resources (Source REALIZE)*

|  |  |  |
| --- | --- | --- |
| **DESIGNATION** | **SOFTWARE** | |
| **Operation system** | | Windows 10, Mac OS, parrot Linux |
| **Design tools** | | Photoshop |
| **Integrated development environment**  **(IDE)** | | Visual studio code, |
| **Text editor** | | Sublime text, notepad++ |
| **Database management system (DBMS)** | | MongoDB, PostgreSQL, MySQL |
| **Web browser** | | Google chrome, Microsoft edge |
| **Document editor** | | Microsoft office word |
| **Presentation** | | Microsoft office PowerPoint |

1. BRIEF PRESENTATION OF THE PROJECT

During our insertion phase at REALIZE, we identified themes addressing specific problems and introducing innovation. After reviewing our initial ideas, our professional supervisor took the time to evaluate our themes and proposed some improvements. Finally, we had to go for the selected theme: **"CONCEPTION AND REALIZATION OF A WASTE COLLECTION SYSTEM."**

This platform aims to improve waste collection processes, making them more efficient and organized. It will help waste collectors and community members by providing a digital solution to manage their waste collection activities. Each contributor (community member or household) will have the ability to view their assigned collector, see the history of waste collected, and make online requests for waste pick-up. Collectors will be able to register waste contributions through the application and issue receipts to contributors for each collection. Additionally, the admin will have the capability to assign collectors to specific contributors, thereby streamlining the entire waste management process and reducing environmental nuisances caused by uncollected waste.

CONCLUSION

The insertion phase at REALIZE established the foundation for our internship, enabling rapid integration into the Software Engineering host unit and a clear understanding of the organization’s structure, workflows, and resources. We completed initial HTML/CSS exercises to validate core concepts and began shaping project ideas, preparing us to transition from onboarding to hands-on work. The phase culminated in the shift toward conceiving and implementing a waste collection system, setting the stage for practical design, stakeholder engagement, and scalable, impact-driven development.

PART 2

TECHNICAL PHASE

# FILE 1: EXISTING SYSTEM

Preamble

This section explains the existing system we are studying and outlines the app we will build. It describes how the existing system works, where it falls short, and the problems this causes. It then presents practical fixes and the initial steps to implement them.

Content Overview

INTRODUCTION

1. THEME PRESENTATION
2. DESCRIPTION OF THE EXISTING SYSTEM
3. LIMITATIONS OF THE EXISTING SYSTEM
4. PROBLEMATIC
5. PROPOSED SOLUTION

CONCLUSION

INTRODUCTION

In this part of our report, we will explain our theme. We will also study the current system and identify its strengths and weaknesses. Based on this analysis, we will present the problem statement, which is the main challenge or gap that the current system faces. Finally, we will propose a solution that addresses this problem and improves the system.

* 1. THEME PRESENTATION

Our theme, **“Conception and Implementation of a Waste collection System”**, our project focuses on designing and developing a digital waste collection system to improve how waste is collected, tracked, and managed in urban and hard-to-reach areas.

The system connects households, businesses, and waste collectors through an intuitive interface, addressing critical inefficiencies in traditional waste management (like **HYSACAM’s** limitations in Cameroon).

The conception phase involves analyzing existing waste management practices (by **AMAH Association and HYSACAM**), identifying limitations such as inefficiency, poor payment tracking, and inaccessible neighborhoods, and proposing a digital solution.

The implementation phase consists of creating a mobile application that allows residents to register, subscribe, and make payments online, integrates **GPS tracking** for precise pickups, coordinates agents’ routes, and provides management with real-time monitoring and reporting tools.

Overall, this project aims to optimize waste collection, increase operational efficiency, and ensure better service coverage, particularly in areas that are difficult to reach, while making the system more transparent and convenient for both residents and waste management agents.

* 1. DESCRIPTION OF THE EXISTING SYSTEM

In Cameroon, companies such as HYSACAM (Hygiène ET Salubrité au Cameroun) and smaller organizations like the AMAH Association mainly handle waste management. Both are responsible for maintaining hygiene and sanitation in various communities.

**HYSACAM** focuses largely on servicing accessible urban and suburban areas. Their teams consist of drivers, cleaners, and inspectors who clean trash bins, collect waste, and ensure equipment is in working order. While their operations cover large zones, they face challenges in reaching certain hard-to-access neighborhoods, especially those located on steep terrain, near cliffs, or in narrow, unpaved streets. Trucks may be unable to navigate these areas, leaving residents with irregular waste collection services.

In contrast, **AMAH Association** operates on a smaller scale but takes a more personalized approach, especially in less accessible areas. AMAH agents often go door-to-door, directly informing residents of their presence and collecting garbage even in neighborhoods that trucks cannot reach. They cover households, restaurants, and community spaces; ensuring waste was collected from places that larger trucks overlook.

However, both systems face operational challenges:

* No real-time tracking of bins or households: Teams do not know exactly when waste bins are full, leading to unnecessary trips or delayed collections.
* Limited communication with residents: Many residents wait for trucks without knowing the exact collection time, and some miss the opportunity to dispose of their waste.
* Accessibility issues: HYSACAM struggles with hard-to-reach locations, while AMAH manual, approach is labor-intensive and time-consuming.
* Inefficient resource usage: Wasted fuel, increased travel time, and higher operational costs.

These challenges result in uncollected waste in some areas, increased health risks, and higher operational expenses. Your proposed system aims to support AMA Association by integrating a GPS-based tracking solution that can monitor households, restaurants, and communities, enabling optimized route planning and timely pickups. This would ensure residents no longer have to wait unnecessarily for trucks and that waste collection teams operate more efficiently.

* 1. LIMITATIONS OF THE EXISTING SYSTEM

Table 3: Limitations, Consequences and proposed solutions

|  |  |  |
| --- | --- | --- |
| Limitations | Consequences | Proposed Solutions |
| No central digital system for households to request waste collection | Residents must wait for fixed schedules, often missing pickups | Mobile app where residents can log in, subscribe, and request collection directly |
| Lack of real-time location sharing between clients and collectors | Collectors waste time searching or driving to wrong addresses | GPS integration that automatically shares client location after subscription or pickup request |
| Inconvenient and manual payment methods | Delays in payment, need for cash handling | Integrated payment API supporting Orange Money and Mobile Money for instant, cashless transactions |
| No option for urgent pickup requests | Waste can pile up in special situations (meetings, events, etc.) | “Request Immediate Pickup” feature allowing clients to notify collectors outside normal schedules |
| No tracking of subscription and collection history | No transparency for clients or operators, disputes may arise | User dashboard to view payment history, collection records, and subscription status |
| No efficient communication channel between clients and collectors | Misunderstandings and delays in service | In-app messaging or push notifications for updates and confirmations |

* 1. PROBLEMATIC

In Cameroon, urban waste management faces significant challenges. While companies like HYSACAM provide general waste collection, they are often unable to reach hard-to-access neighborhoods, leaving some households without consistent service. Associations like AMAH attempt to fill this gap with door-to-door collection, but their operations remain manual, time-consuming, and inefficient, relying on paper-based records and physical visits.

The lack of digital tools for tracking households, managing subscriptions, processing payments, and scheduling pickups creates several issues:

1. Inefficient waste collection, especially in areas with difficult terrain.

2. Delayed or missed services, leading to unhygienic conditions.

3. Difficulty in monitoring payments and subscriptions, increasing the risk of revenue loss.

4. Limited communication between residents and collection agents, causing frustration and reduced satisfaction.

Therefore, there is a need for a digital system that integrates real-time household tracking, payment processing, and pickup scheduling, to improve efficiency, transparency, and accessibility in waste management, particularly for underserved neighborhoods.

* 1. PROPOSED SOLUTION

The proposed solution is a mobile-based waste collection system that allows residents to subscribe and pay online, tracks households via GPS for efficient pickups, and provides digital management tools to coordinate services and improve coverage, efficiency and transparency in to our homes and country.

1. Mobile Application for Residents

Develop a user-friendly app allowing residents to register, subscribe, and make payments directly through mobile money, Enable residents to schedule pickups or request services.

2. Real-Time Location Tracking

Use GPS APIs to track households’ locations, especially in hard-to-reach or inaccessible areas, allowing collection agents to locate homes efficiently.

3. Digital Subscription and Payment Management

Maintain a database of subscribers, tracking payment status and subscription history in real time.

4. Integration of AMAH and HYSACAM Operations

Coordinate the activities of AMAH (door-to-door collection) and HYSACAM (main city collection) to ensure assign pickups intelligently based on accessibility, subscription, and location.

5. Pickup Request and Agent Communication System

Allow residents to notify agents if they need urgent pickup or rescheduling.

The solution is a digital waste collection system that combines a mobile app, GPS tracking, payment integration, and intelligent pickup scheduling. It aims to increase efficiency, transparency, and accessibility while reducing the limitations of manual collection in hard-to-reach areas.

CONCLUSION

In conclusion, the current waste collection system faces challenges in efficiency, accessibility, and payment management, particularly in hard-to-reach areas. The proposed digital solution, combining a mobile app, GPS tracking, online payments, and integrated management tools, offers a practical and effective way to optimize waste collection, ensure timely service, and improve overall transparency and customer satisfaction.

# FILE 2: SPECIFICATION BOOK

Preamble

The primary goal of the specification book is to outline with great precision the requirements or need of the users and the description of the resources necessary to realize the project. The project's success will depend on careful allocation and management of the various resources available, including physical, financial, and human assets. Therefore, this document outlines the criteria that the developer must adhere to in order to ensure the success of the project.

Content Overview

INTRODUCTION

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

CONCLUSION

INTRODUCTION

The report's specification book enables us to provide comprehensive details about our topic, enhancing our understanding and improving its chances of success. To establish the project's scope, we will describe the context of our topic and identify the issues within it that we aim to solve. Once the solution is proposed, we will outline the objectives we have set for ourselves and for the project. It establishes a certain agreement between the client of the project and the person who is supposed to realize the project. It presents the objectives and needs of the system to be developed. The present specification book relates to the context and justification, the needs of the project, the plan and the deliverables.

1. CONTEXT AND JUSTIFICATION
2. CONTEXT

Cameroon, with a population of nearly 28.7 million (2025 estimate) inhabitants, especially in major cities like Douala, Yaoundé, and Bafoussam. Which faces an increase in challenges in urban waste management. The rapid growth of cities, coupled with irregular and inefficient waste collection methods, has contributed to unsanitary living conditions, public health risks, and environmental degradation.

Within an annual year waste generation generates approximately 6million tons of waste per year. Key statistics include:

* **Urban Waste Production**: Douala and Yaoundé generates approximately 3000 and 1800 tons of waste per day.
* **Recycling Rate**: Less than 10% of waste is formally recycled.
* **Open Dumping**: Over 60% of waste is disposed of in open dumps or roadside areas.
* **Landfill Capacity**: Existing landfills are often overburdened and environmentally unsafe.
* **Recycling and Reuse**: Informal waste pickers play a role, but formal systems remain underdeveloped.
* **Waste Composition**: It includes household, commercial, and industrial waste with a notable amount of plastic waste.

Currently, the system heavily depends on manual processes and irregular waste truck schedules. Households often miss collectors due to the lack of fixed collection times. This leads to waste accumulation, unpleasant odors, and the proliferation of diseases. Collectors themselves lack traceability and coordination, while communities have no access to data on waste production trends.

1. JUSTIFICATION

The justification for developing a computerized platform (Zerodech) for smart waste management lies in addressing the inefficiencies of the current system. Automation and digitalization will:

* Streamline operations and provide **real-time notifications** of collection schedules.
* Reduce missed pickups and enhance **coordination between households and collectors**.
* Ensure **traceability of collection activities**, minimizing errors and improving accountability.
* Provide **data insights** on household and community waste, supporting decision-making and environmental planning.
* Improve **quality of life**, promote cleaner environments, and encourage sustainable waste practices.

Ultimately, Zerodech contributes not only to health and sanitation but also to smart city development, community empowerment, and environmental sustainability.

1. OBJECTIVES OF THE PROJECT

1. General Objective

To develop a computerized platform that automates and optimizes household and community waste management, thereby improving efficiency, traceability, and sustainability.

2. Specific Objectives

The specific objectives of this project are to:

➢ Facilitate household and community waste tracking.

➢ Provide real-time notifications of collection schedules to users.

➢ Improve transparency by showing households which collector is assigned to them.

➢ Strengthen the relationship between collectors and contributors through communication features.

➢ Enable access to historical records of collections for households and communities.

➢ Provide data analytics for better decision-making by municipalities and waste agencies.

The project is characterized by the following:

* **Project Name:** Zerodech
* **Project Target:** Individuals, businesses, and communities seeking reliable and efficient waste management services.
* **Technical Specification:** Mobile application (React Native) integrated with a backend (Node.js, MongoDB) for real-time data management, authentication, and payment processing.

1. EXPRESSION OF NEEDS

### 1. FUNCTIONAL NEEDS

In this context, functional requirements describe what the system (Zerodech) should do. The modules are as follows:

**Customer (Households & Businesses):**

* **Fill Waste Registration Form:** Customers can register their bins and provide details such as household type, waste category, and pickup preferences.
* **Track Collection:** Customers can monitor the status of their waste pickup and receive real-time notifications when a collector is near.
* **View History:** Customers can access their past collection records for transparency.
* **Payment Processing (optional):** Customers can make payments for premium services (special pickups, recycling, etc.) via Mobile Money or other methods.
* **Chat:** Customers can chat with the assigned collector or support for clarifications.

**Collector (Waste Collectors & Agencies):**

* **Manage Profile:** Collectors can update their profile, service areas, and working hours.
* **View Collection Requests:** Collectors can see assigned households/communities and confirm when waste is collected.
* **Update Availability:** Collectors can mark their availability to optimize scheduling.
* **Communicate with Customers:** Collectors can contact households for specific instructions.
* **Upload Reports:** Collectors can upload photos or reports to confirm successful collection.

**Administrator (Community / Waste Agency Admin):**

* **Manage Collectors:** Verify collector registrations, assign them to neighborhoods, and oversee performance.
* **Manage Collection Records:** Monitor household/collector interactions and resolve disputes.
* **Track Platform Performance:** Access reports on community cleanliness, collection efficiency, and waste data statistics.
* **Manage User Roles:** Assign roles (customer, collector, admin) and permissions.

# 2. NON-FUNCTIONAL NEEDS

In this section, non-functional requirements define the quality attributes, standards, and performance measures that the Zerodech system must adhere to. These aspects ensure the overall performance, security, and usability of the application.

### ****1. Performance Requirements****

* The system should be able to handle multiple simultaneous user requests (households, collectors, administrators) without lag, especially during peak collection times.
* Pages, dashboards, and notifications must load and respond in under **3 seconds** for a seamless experience.

### ****2. Security Requirements****

* All sensitive data (user credentials, household information, collection history, and payment details where applicable) must be **encrypted** using industry-standard protocols (e.g., AES-256).
* Implement robust **authentication and authorization** mechanisms, including JWT for authentication and role-based access control (RBAC) for customers, collectors, and admins.

### ****3. Scalability Requirements****

* The platform should be **scalable** to support growing numbers of users (households, businesses, collectors) without impacting performance.
* Both frontend and backend must support **horizontal scaling** to accommodate user demand in multiple communities and cities.

### ****4. Reliability and Availability****

* The application must ensure an uptime of at least **99.9%** to guarantee availability for households and collectors.
* Robust **error-handling and fallback mechanisms** must be implemented to maintain service continuity in case of server or network failures.

### ****5. Data Integrity****

* The system must maintain the integrity of all data across transactions (no missing, inconsistent, or duplicated collection records).
* **Automatic backups and transaction logs** must be in place to ensure no data is lost in case of failures.

### ****6. Geolocation Accuracy****

* The geolocation feature for tracking collectors must provide real-time location data with a margin of error not exceeding **50 meters**.
* The system should refresh the collector’s location on the map every **15–30 seconds** for accurate positioning.

### ****7. Maintainability****

* The platform should be **modular** to allow updates, bug fixes, and new feature integration without disrupting other modules.
* The codebase must follow best practices with proper **documentation** to ensure easy maintenance and scalability by future developers.

1. PROJECT PLANNING
2. **Chronogram of activities**

Table 4: Chronogram of activities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PHASE** | **OBJECTIVE** | **OUTPUT** | **DURATION** | **PERIOD** |
| **INSERTION** | Collection of information on the enterprise | Insertion book | 2 weeks | 03th July to 22th July |
| **EXISTING SYSTEM** | Study of the existing system | Existing System | 1 week | 23nd July to 31th July |
| **SPECIFICATION BOOK** | Specification of the user needs | Specification Book | 1 week | 1st August to 08th August |
| **ANALYSIS** | Capture of needs Use case and textual description Modelling | Analysis Book | 2 weeks | 11th August to 21st August |
| **CONCEPTION** | Preliminary conception and Detailed conception | Conception book | 2 weeks | 22th  August to 01st September |
| **REALIZATION** | Implementation Unitary test Integration Test Development | Realization book | 3 weeks | 02nd September to 22nd September |
| **TEST OF FUNCTIONALITIES** | Testing of the software and debugging | Test of functionalities | 1 week | 15th September to 25th September |
| **INSTALLATION AND USER GUIDE** | Documenting software | User Guide | 1 week | 26th September to 03rd October |

1. **Gantt Project Planning**

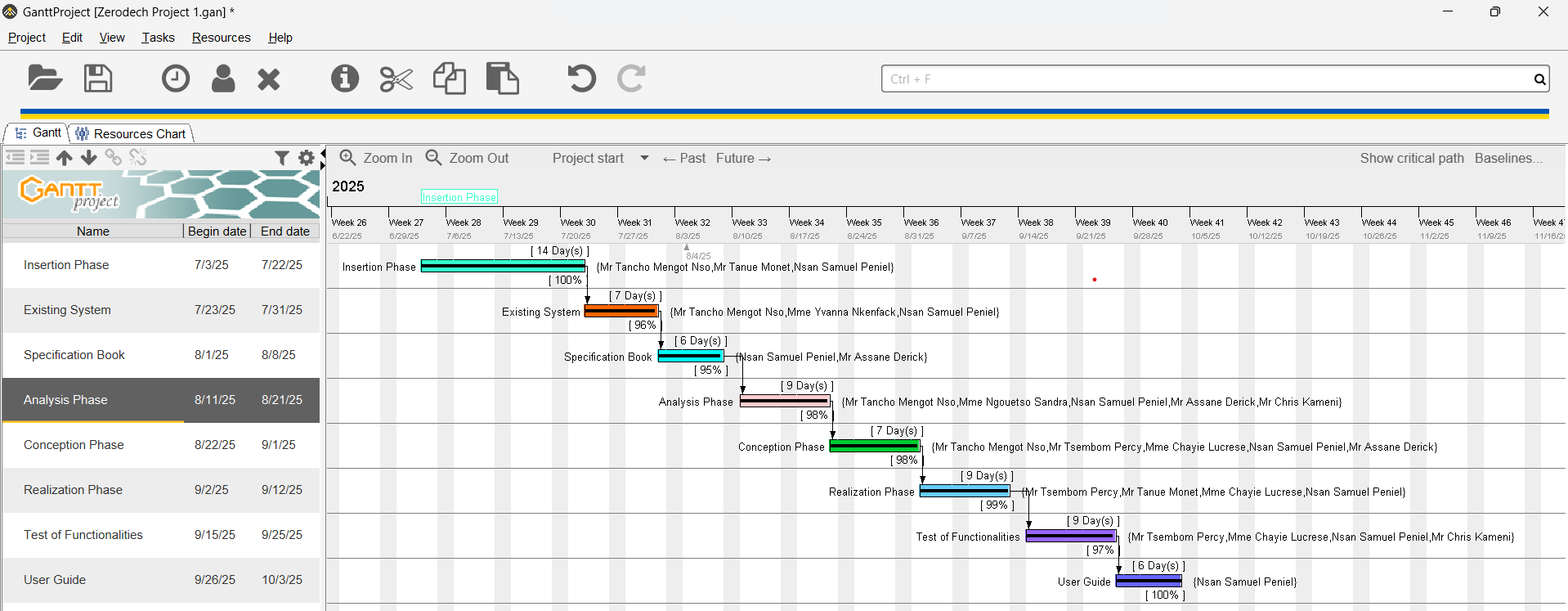
****

Figure 3: Project Gantt chart

1. PROJECT ESTIMATION COST

### **Software Resources**

Table 5: Software resources (source: MERCURIALE-2025)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RESOURCES | DESIGNATION | USAGE | QUANTITY | UNIT COST (FCFA) |
| **Formatting** | MICROSOFT OFFICE PROFESSIONAL 2016 | Writing of the report | 1 | 355,465 |
| **Web browser** | Google Chrome | Access and view web pages on the internet | 1 | Freeware |
| **Code Editor** | Visual Studio Code | For writing the code of the application | 1 | Freemium |
| **Project Planning** | Gantt Project | For building a Gantt chart | 1 | Freemium |
| **Graphic Editor** | Icogram | For building a geographical location of the enterprise | 1 | Freemium |
| **UML Analysis** | Visual Paradigm | For drawing UML diagrams | 1 | Community Version |
| **Version Control** | Git & GitHub | For tracking and managing changes to software code | 1 | Free Version |
| **TOTAL 1** |  |  | **7** | **355,465** |

1. Hardware Resources

Table 6: Hardware resources (source: MERCURIALE-2025)

|  |  |  |  |
| --- | --- | --- | --- |
| RESOURCES | HARDWARE | QUANTITY | UNIT COST (FCFA) |
| Computer | HP ZBook 15 G4 Laptop, Core i7, 500GB SSD, 16GB RAM | 1 | 1,295,000 |
| Printer | Printers | 1 | 514,528 |
| Network | Local network installation | 1 | 300,000 |
| Removable Disk | Toshiba USB Flash Drive V225W – 2GB | 1 | 14,160 |
| Smartphone | Techno Spark 20 | 2 | 150,000 |
| Network | Local network installation | 1 | 300,000 |
| Total 2 |  | 7 | **2,273,688** |

### **Human Resources**

Table 7: Human resources (source: Mercurial 2025)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ROLE | NUMBER OF DAYS | QUANTITY | COST PER DAY | TOTAL PRICE  (FCFA) |
| Project Manager | 90 | 01 | 30 000 | 2,700,000 |
| Analyst | 21 | 01 | 25 000 | 525,000 |
| UI/UX Designer | 30 | 01 | 10 000 | 300,000 |
| Programmer | 30 | 01 | 15 000 | 450,000 |
| Tester | 14 | 02 | 10 000 | 280,000 |
| Margin error | / | / | / | 1,475, 000 |
| TOTAL 3 |  |  |  | 5,730,000 |

### **Global Estimation**

Table 8: Global estimation

|  |  |  |  |
| --- | --- | --- | --- |
| TOTAL 1 (FCFA) | TOTAL 2 (FCFA) | TOTAL 3 (FCFA) | OVERALL TOTAL (FCFA) |
| **355,465** | **2,273,688** | 5,730,000 | 8,359,153 |
| **EIGHT MILLION THREE HUNDRED FIFTY-NINE THOUSAND ONE HUNDRED FIFTY-THREE** | | | |

1. PROJECT CONSTRAINTS

1. Technical Constraints

The project's technical feasibility is constrained by our dependency on the municipal service's infrastructure; the core-tracking feature requires their vehicles to be equipped with GPS and their willingness to grant us API access to their real-time data.

2. Time Constraints

The project must be delivered by October 2nd, 2025, following a 13-week (3-month) development cycle starting July 3rd, 2025.

3. Cost constraint

The implementation of our project will necessitate expenses on work force, materials, and software, amounting to 8,359,153.

1. LIST OF PARTICIPANTS AND DELIVERABLES

### **LIST OF PARTICIPANTS**

Table 9: List of participants

|  |  |  |
| --- | --- | --- |
| NAME | FUNCTION | ROLE |
| Mr. MENGOT NSO | Follows up interns at the company level | Professional Supervisor |
| Mr. ASANE DERICK | Follows up student at the academic level | Academic Supervisor |
| NSAN SAMUEL PENIEL | AICS Student Intern | Student at AICS |

### **DELIVRABLES**

In project management, any component materializing the result of a realization service is a deliverable. In the case of our project, the deliverables are:

A report is composed of the following document:

* A User Guide
* Software Application
* PowerPoint

CONCLUSION

In conclusion, this part has come to a close, wherein we have effectively listed and expounded on our goals. The specifications book provided us with the means to present the project's different stakeholders, as well as the prerequisites and tentative schedule essential for its realization. Moving forward, we will directly proceed with the analysis phase, where we will utilize a modelling language and a unified approach to exemplify our system. Furthermore, we will carry out a comparative evaluation of UML and Merise methodologies.

# FILE 3: ANALYSIS PHASE

Preamble

The main objective of the analysis phase is to capture the user’s need, the delimitation of the field of study and to have a clear understanding of the system in study. To achieve this, we will use UML (Unified Modelling Language) with the 2TUP (two Track Unified Process) as method applied to UML to analyze the system. We will start by performing a preliminary study of the software, followed by the identification of the different stakeholders that interact with the system, finally perform the analysis of the functional needs through a global and detail view of the application.

Content Overview

INTRODUCTION

1. I. COMPARATIVE METHODOLOGY
2. STUDY OF UML AND MERISE
3. STUDY OF UNIFIED PROCESSES
4. II. CHOICE OF THE ANALYSIS METHOD

III. JUSTIFICATION OF THE ANALYSIS METHOD

1. IV. MODELLING OF THE PROPOSED SOLUTION

CONCLUSION

INTRODUCTION

System development can be thought of as having two major components: System analysis and system design which both help in understanding the details of the existing system or the system to be designed. The analysis and design of information systems has most of the time vocation to allow the creation of databases, which must represent as closely as possible the reality of the field studied thus requiring the use of a design method. This is why our choice will be directed on the UML method as it offers much to developers seeking a user-centered approach and / or a wide scope in design. This part of the report consists of the comparative study of UML and MERISE unified processes and finally the various diagrams that meet the functional need requirements.

1. COMPARATIVE METHODOLOGY
2. STUDY OF UML AND MERISE
3. **MERISE**

MERISE stands for “Méthode d’Etude et de Réalisation Informatique pour des Systèmes d’Entreprise”. Although it is prescriptive to some extent, MERISE permits the participation of end users and senior management as well as data processing professionals in its decision cycle. MERISE is a method for designing, developing and carrying out IT projects. The goal of this method is to achieve the design of an information system. The MERISE method is based on the separation of data and processing to be carried out in several conceptual and physical models. The essentials of the approach lie in its three cycles: the decision cycle, the life cycle and the abstraction cycle, which cover data and process elements equally. The separation of data and processing ensures longevity in model. Indeed, the arrangement of data does not have to be often overhauled, while treatments are more frequently.

1. UML

UML (Unified Modelling Language) is a standard notation for the modelling of real world objects as a first step in developing an object-oriented design methodology. Its notation is

derived from and unifies the notations of three object-oriented design and analysis

methodologies: Grady Booch's methodology for describing a set of objects and their

relationships, James Rum Baugh’s Object-Modelling Technique (OMT), Ivar Jacobson's

approach, which includes a use case methodology. Other ideas also contributed to UML,

which was the result of a work effort by Booch, Rum Baugh, Jacobson, and others to combine

their ideas, working under the sponsorship of Rational Software. UML captures information

about the static and dynamic view of a system. UML 2.0 comprises of 13 diagrams which

represent the different views of a system. The 13 diagrams can be subdivided into two, Static

alternatively, structural and dynamic diagrams. These diagrams include;

1. STATIC OR STRUCTURAL DIAGRAMS

❖Class diagram

❖Object diagram

❖Component diagram

❖Deployment diagram

❖Package diagram

❖Profile Diagram

2. BEHAVIOURAL OR DYNAMIC DIAGRAMS

❖Use case diagrams.

❖Activity diagram.

❖State machine diagram.

❖Sequence diagram.

❖Communication diagram

❖Global Interaction diagram.

❖Timing Diagram

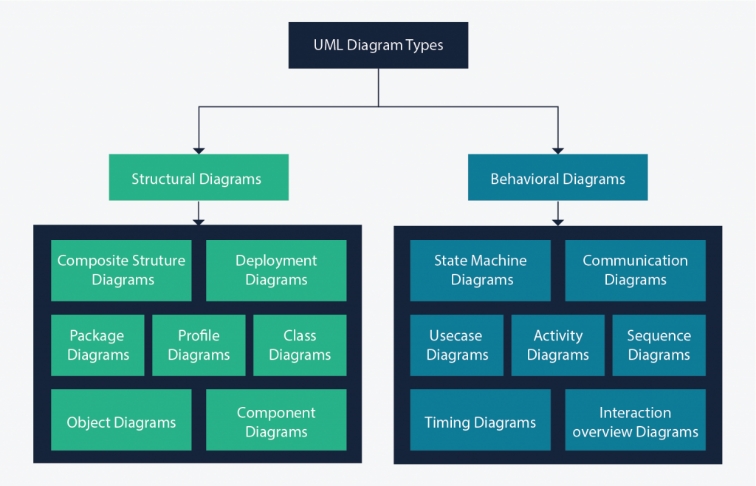


Figure 4: UML 2.5 diagrams overview (source: <https://creately.com/blog/diagrams/uml-diagram-types-examples/> )

1. STUDY OF UNIFIED PROCESSES
2. **Unified Process (UP)**

A Unified Process is a generic software development method. Generic means that it is necessary to

adapt UP to the context of the project, team, domain and/or organization and process of development of software constructed on UML; it is iterative, incremental, centered on architecture, driven by use cases and requirements.

**Iteration** is distinct sequence of activities with a basic plan and evaluation criterion that produces an internal or external output. Either the content of an iteration is improved, or users evaluate the evolution of the system.

**An increment** is the difference between two released products at the end of two iterations. Each iteration that the group is capable of integrating the technical environment in order to develop a final product and give users the possibility of having tangible results.

**Centered on architecture** the different models derived during the establishment of system must be reliable and coherent.

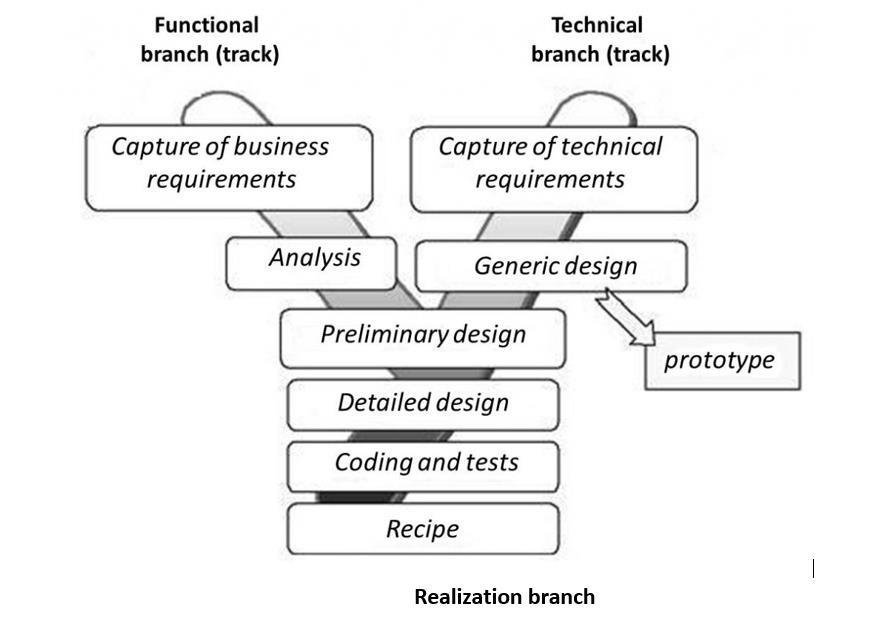
**Driven by use case and requirements** enables the clear definition of a user’ needs and priorities respectively thereby minimizing the risk of project failure.

The essential characteristics of the unified process are:

* It is component-based.
* It uses the UML language (set of tools and diagrams).
* It is driven by use cases.
* It is centered on architecture.
* It is iterative and incremental.

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

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



*Figure5:2TUP diagram (Source:https://www.mysciencework.com/omniscience/pervasive-mobile-healthcare-system-basedon*

*cloud-computing)*

a. The left branch (functional branch)

It captures the functional needs of a system. This ensures the production of

software that meets the needs/requirements of the user. The analysis here consists of

studying precisely the functional specification in order to obtain an idea of what the

system is going to realize, and its result does not depend on any technology.

b. The right branch (Technical branch)

The technical branch enumerates the technical needs and proposes a generic

design validated by a prototype. The technical needs include constraints and choices

related to the conception of the system, the tools and equipment as well as the

integration constraint with the existing system condition.

c. The middle branch (Realization or Implementation branch)

In this branch, we study the preliminary conception, detailed conception, and

documentation of the system. The realization branch supports the following:

**Preliminary conception:** This is the most sensitive step of 2TUP as it is the

confluence of the functional and technical branch. It is completed when the

deployment model, the operating model, the logical model, interphases and the

software configuration model are defined. We have the following diagrams:

* -Component Diagram
* Deployment Diagram
* Package Diagram
* Composite Structure Diagram

**Detailed conception:** This is the detailed design of each feature of the system. We

have the following diagrams:

* Class
* Object
* Sequence
* Timing Diagram

**Coding and testing**: This is the phase where we program the designed features and

test the coded features.

**The recipe:** Also known as, the deliverables is the validation phase of the functions of

the developed system.

CHARACTERISTICS OF 2TUP

The two-track unified process was built to be used with the Unified Modeling

Language to become a method and has the following characteristics:

* **It is user oriented**: 2TUP is built from user’s expectation.
* **It is component oriented**: it offers flexibility to the model by supporting the re-use

of components.

* **It is an iterative process**: processes are being done in iterations and each iteration

shows a precise level of abstraction.

* **It is an incremental process:** allowing a better functional and technical risk management and thus constituting the deadlines and the cost control.

1. CHOICE OF THE ANALYSIS METHOD

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

Table 10: Comparison between MERISE and UML

|  |  |
| --- | --- |
| MERISE | UML |
| It is based on procedural approach | Based on object approach |
| Suitable for relational databases design and implementation | Suitable for object-oriented design and implementation |
| It stands for Méthode d'Étude et de Réalisation Informatique pour les Systèmes d'Entreprises | Unified Modeling Language |
| MERISE is a systemic method of analysis and design of information systems. That is, it uses a systems approach. | UML is however not a method but rather an  object modeling language to which it is necessary to associate an approach to make  it a method. This is the case with the 2TUP  method: RUP and XP. |
| Merise is a method of analysis | UML is modeling language |
| Emphasizes data analysis and modelling using data flow diagrams (DFD), entity-relationship diagrams (ERD), and data dictionaries. | Emphasizes system behaviour and structure using 14 diagram types, such as use case diagrams, class diagrams, and activity diagrams |

1. JUSTIFICATION OF THE ANALYSIS METHOD

The reason why we chose UML modelling language and the software development process 2TUP instead of many others that exist, include:

* UML is the current standard for programming in an object-oriented language. For this reason, it is widely understood and well known making it easy for a new programmer to join the project and be productive from the very first day.
* UML diagrams allow teams to virtualize how a project is or will be working, and they can be used in any field, not just software engineering. The diagrams will allow teams to virtualize together how a system or a process will work or did work. It can provide new ideas for how teams have to collaborate to achieve the goal of the workflow process.
* 2TUP is centered around the creation and maintenance of a model, rather than the production of mountain documents.
* 2TUP is user oriented as it permits the development of software that responds to the needs of the users through the study of the user needs.
* 2TUP is iterative and incremental, hence it enables the project team to produce refined amelioration if necessary and easily integrate it in the already existing system.
* 2TUP by permitting the project team identify and test the key functionalities of the system limits the risk related to building the system.

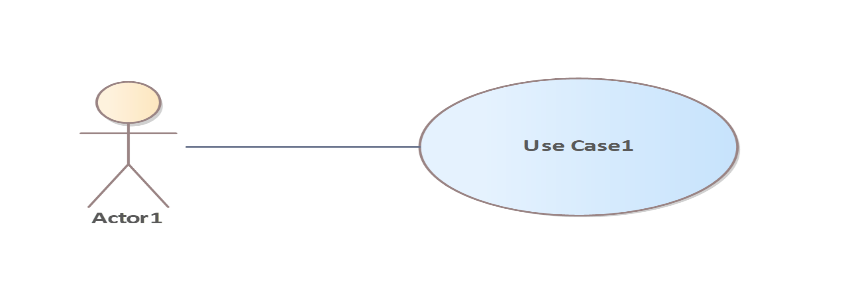
1. MODELING OF THE PROPOSED SOLUTION
2. USE CASE DIAGRAM
3. Definition

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

* Provide a high-level view of the system.
* Identify the functions of the system.

Use case diagrams are completed with a textual description of each use case that is intended to define the use case in details.

1. **Formalism**



*Figure 6: Use case formalism*

1. **Components of use case diagram**

*Table 11: Component of use case diagram*

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

1. **Actors of our system**

Table 12: Actors of our system

|  |  |
| --- | --- |
| Actor | Role |
| Household/Client | A resident or business that generates waste and requires collection services. They can subscribe to waste collection plans, track collection status, view collector locations, and make payments. |
| Waste Collector | Individuals or companies that provide waste collection services. They can view their assigned routes, see subscribed households, update collection status, and communicate with clients. |
| Admin | Responsible for overseeing platform operations. They manage users (households and collectors), handle collector verifications, resolve issues, and monitor system activities. |
| Geolocation API | Provides real-time location data for mapping household addresses, optimizing collection routes, and tracking collector vehicles during operations. |
| Payment API | Handles secure financial transactions for subscription payments from households and disbursements to waste collectors. |

* 1. General Use Case Diagram of ZERODECH

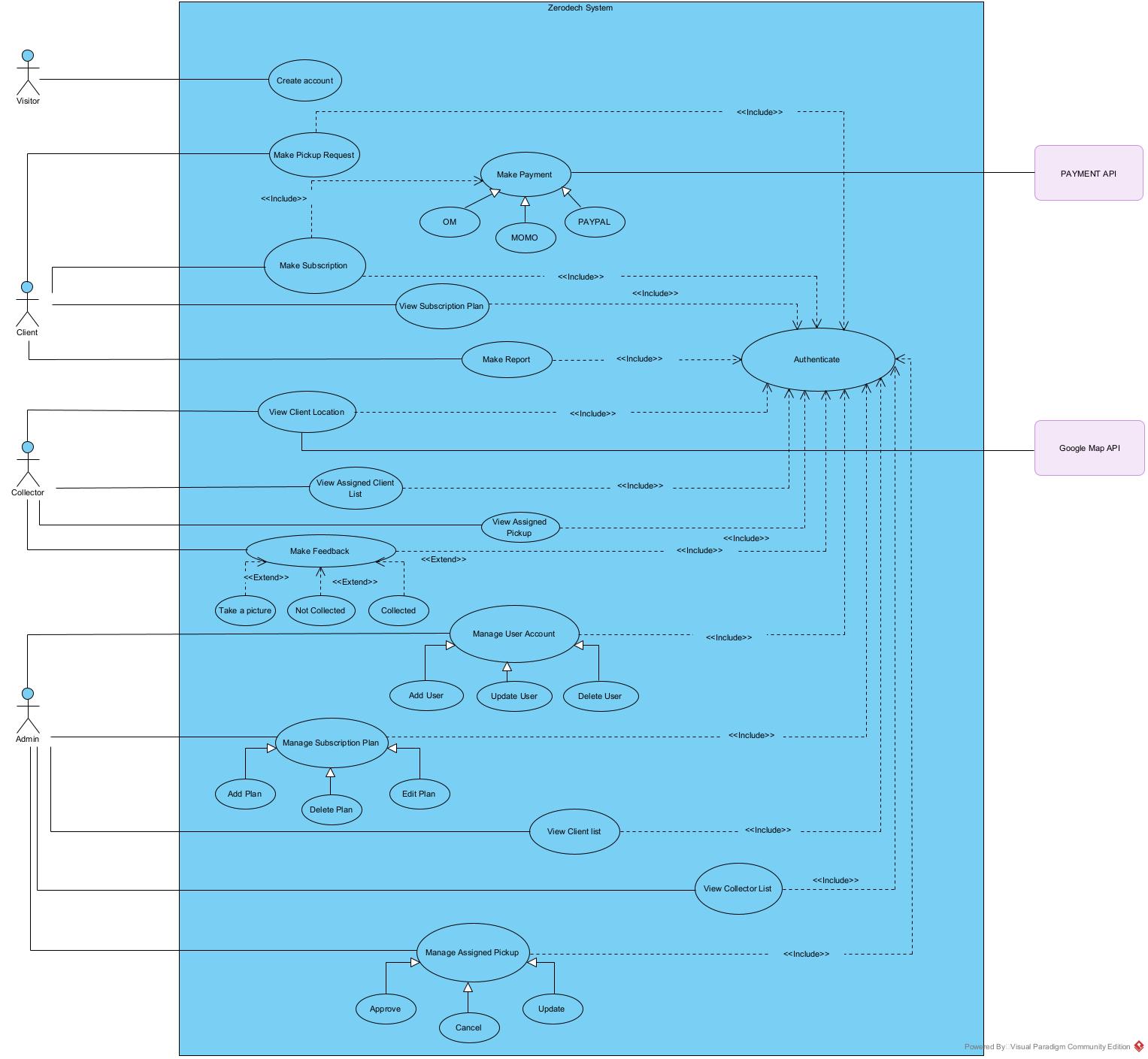


Figure 7: General use case diagram

**Textual Description**

UML allows the execution of a use case to be described in a textual way, in a form called nominal scenario. A nominal scenario describes in more detail the execution of a use case by an actor until it is successfully completed. Scenarios that describe the logical sequence of action that constitute this use case do the description of a use case.

**Formalism**

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

|  |  |
| --- | --- |
| NAME OF THE USE CASE | |
| Summary | Aim of the use case |
| Actors | Users |
| Pre-condition | Set of actions that must be completed before the launching of the use case |
| Trigger | The event that causes the use case to be initiated |
| Nominal scenario | It corresponds to a normal development of a use case |
| Alternative scenario | These are variants of the nominal scenario. |
| Post condition | Set of mechanism that can lead to the end of the use case |

Table 13: Textual description formalism

1. **Create Account**

Table 14: Textual description of Create account

|  |  |
| --- | --- |
| **Field** | **Description** |
| **Name of the Use Case** | Create Account |
| **Summary** | Allows a new client to register in the system to gain access to waste management services. |
| **Actors** | Client (new user) |
| **Pre-condition** | User has internet access and a valid device. |
| **Trigger** | User selects Create Account on the app/website. |
| **Nominal Scenario** | 1. User clicks Create Account. 2. System displays registration form. 3. User enters name, email/phone, and password. 4. System validates input. 5. Account created and saved. 6. Confirmation displayed. |
| **Alternative Scenario** | Input invalid (weak password, duplicate email/phone) → error displayed. |
| **System Response** | Validates and stores new user record; sends confirmation. |
|  |  |
| Post-condition | Client has a valid account and can log in. |

2. **Authenticate**

Table 15: Textual description of Authenticate

|  |  |
| --- | --- |
| **Field** | **Description** |
| **Name of the Use Case** | Create Account |
| **Summary** | Allows a new client to register in the system to gain access to waste management services. |
| **Actors** | Client (new user) |
| **Pre-condition** | User has internet access and a valid device. |
| **Trigger** | User selects Create Account on the app/website. |
| **Nominal Scenario** | 1. User clicks Create Account. 2. System displays registration form. 3. User enters name, email/phone, and password. 4. System validates input. 5. Account created and saved. 6. Confirmation displayed. |
| **Alternative Scenario** | Input invalid (weak password, duplicate email/phone) → error displayed. |
| **System Response** | Validates and stores new user record; sends confirmation. |
|  |  |
| Post-condition | Client has a valid account and can log in. |

3. Make Subscription

Table 16: Textual description of Create account

|  |  |
| --- | --- |
| **Field** | **Description** |
| **Name of the Use Case** | Make Subscription |
| **Summary** | Enables a client to subscribe to a waste collection plan and receive a bucket. |
| **Actors** | Client (new user) |
| **Pre-condition** | Client is logged in. |
| **Trigger** | Client selects Subscribe under a plan. |
| **Nominal Scenario** | 1. Client goes to Plans. 2. Views available options. 3. Selects a plan. 4. Provides address + map location. 5. Confirms subscription. 6. System saves subscription as PENDING. 7. Admin validates and assigns a bucket. |
| **Alternative Scenario** | Client cancels before confirming. Invalid data → correction required. |
| **System Response** | Saves subscription record; notifies Admin. |
|  |  |
| Post-condition | Client has an **active subscription** linked to a bucket |

4. Make Pickup Request

Table 17: Textual description of Make Pickup Request

|  |  |
| --- | --- |
| **Field** | **Description** |
| **Name of the Use Case** | Make Pickup Request |
| **Summary** | Allows a subscribed client to request waste collection. |
| **Actors** | Client, Collector |
| **Pre-condition** | Client has an active subscription and a delivered bucket. |
| **Trigger** | Client clicks Request Pickup on dashboard. |
| **Nominal Scenario** | 1. Client logs in → dashboard. 2. Clicks Request Pickup. 3. Confirms request. 4. System records pickup with status = PLANNED. 5. Collector views request on map. 6. Collector collects waste → marks as PICKED. 7. Client notified of completion. |
| **Alternative Scenario** | Client cancels before collection. Collector unavailable. System error → retry needed. |
| **System Response** | Records pickup request, updates status, notifies collector and client. |
|  |  |
| Post-condition | Waste is collected and recorded in the system. |

5. Make Payment

Table 18: Textual description of Make Payment

|  |  |
| --- | --- |
| **Field** | **Description** |
| **Name of the Use Case** | Make Payment |
| **Summary** | Allows client to pay for subscription using supported methods (e.g., Mobile Money, Card). |
| **Actors** | Client, Payment Gateway |
| **Pre-condition** | Client has an active subscription pending payment. |
| **Trigger** | Client selects Pay Now on subscription |
| **Nominal Scenario** | 1. Client selects payment method.  2. Enters details (Mobile Money/Bank Card).  3. System forwards request to payment gateway.  4. Payment gateway confirms transaction.  5. System updates subscription as ACTIVE. |
| **Alternative Scenario** | Insufficient funds, payment declined, or timeout. Client retries or selects another method. |
| **System Response** | Processes transaction securely via payment gateway; updates subscription status. |
|  |  |
| Post-condition | Subscription is paid and activated. |

1. Activity Diagram

An activity diagram provides a behavior view of the system by describing the sequence of

action in a process or a particular activity. The activity diagram makes it possible to emphasis the processing and to graphically formalize the sequence of action carried out in the use case.

1. **Formalism**

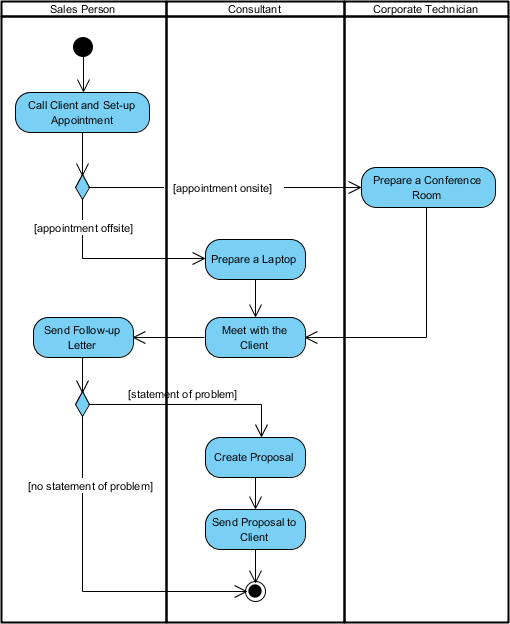


Figure 8: Activity diagram formalism

1. **Component of an Activity diagram**

Table 19: Component of an Activity diagram

|  |  |  |
| --- | --- | --- |
| Element | Notation | Description |
| Activity | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps3.jpg | Is used to represent a set of actions |
| Action | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps4.jpg | A task to be performed |
| Activity Edge | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps5.jpg | An **activity edge** in a UML activity diagram is a directed connection between two activity nodes. It represents the flow of control or data between these nodes. |
| Initial node | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps6.jpg | Portrays the beginning of a set of actions or activities |
| Activity Final Node | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps7.jpg | Stop all control flows and object flows in an activity (or action) |
| Object node | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps8.jpg | Represent an object that is connected to a set of Object Flows |
| Decision node | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps10.jpgC:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps9.png | Represent a test condition to ensure that the control flow or object flow only goes down one path |
| Merge node | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps12.jpgC:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps11.png | Bring back together different decision paths that were created using a decision-node. |
| Fork node | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps13.jpg | Split behavior into a set of parallel or concurrent flows of activities (or actions) |
| Join node | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps14.jpg | Bring back together a set of parallel or concurrent flows of activities (or actions). |
| Swimlane and Partition | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps15.jpg | A way to group activities performed by the same actor on an activity diagram or to group activities in a single thread |

1. **Activity diagram of Authenticate**

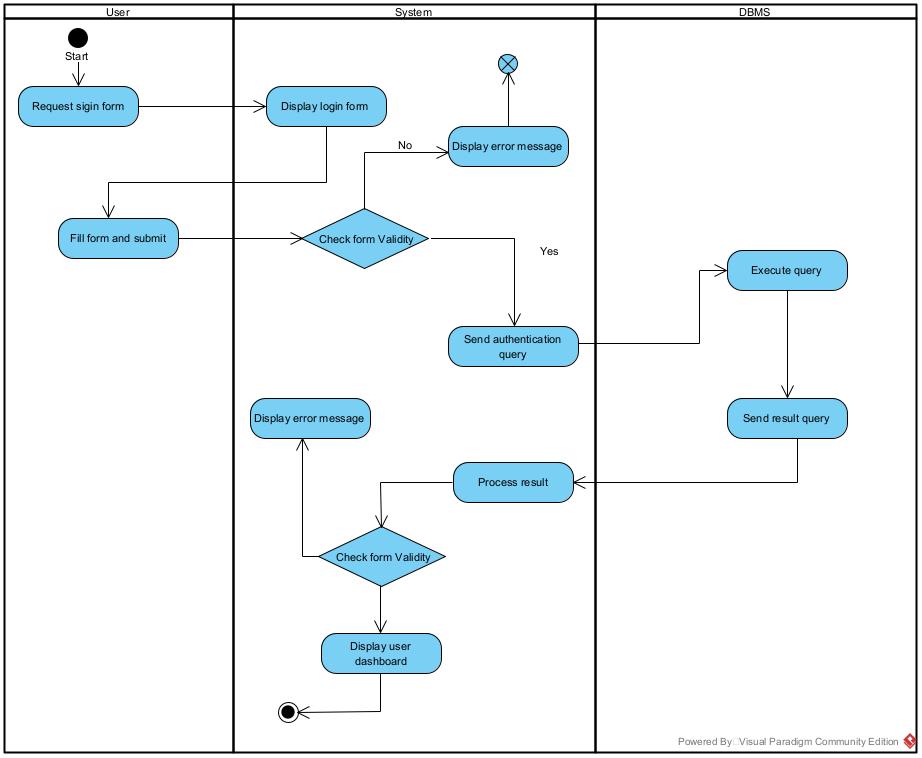


Figure 9: Authenticate activity diagram

1. **Activity diagram of Account Creation**

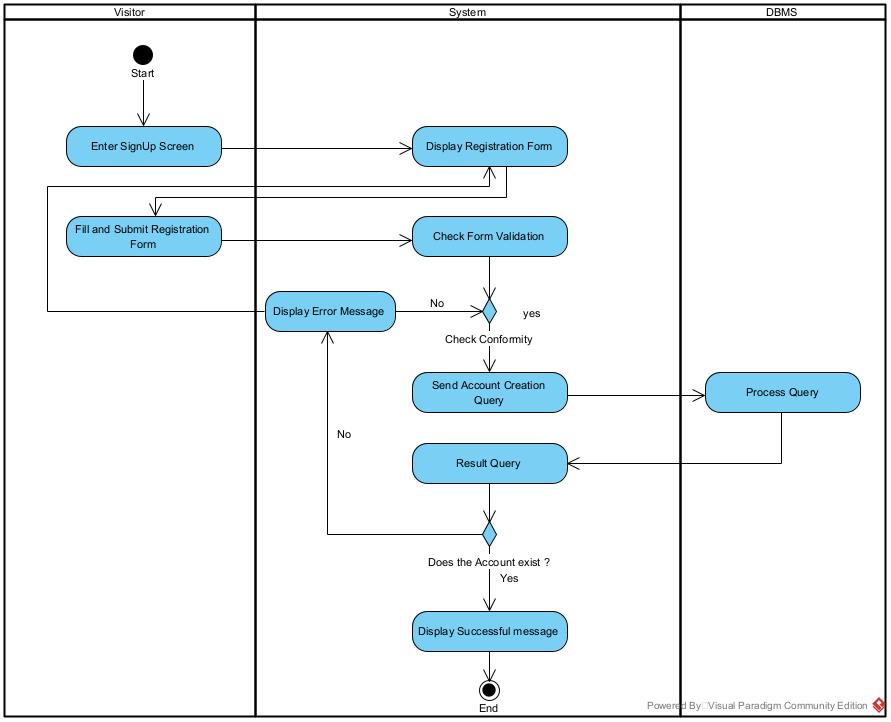


Figure 10: Create account activity diagram

1. **Activity diagram of Make Subscription**

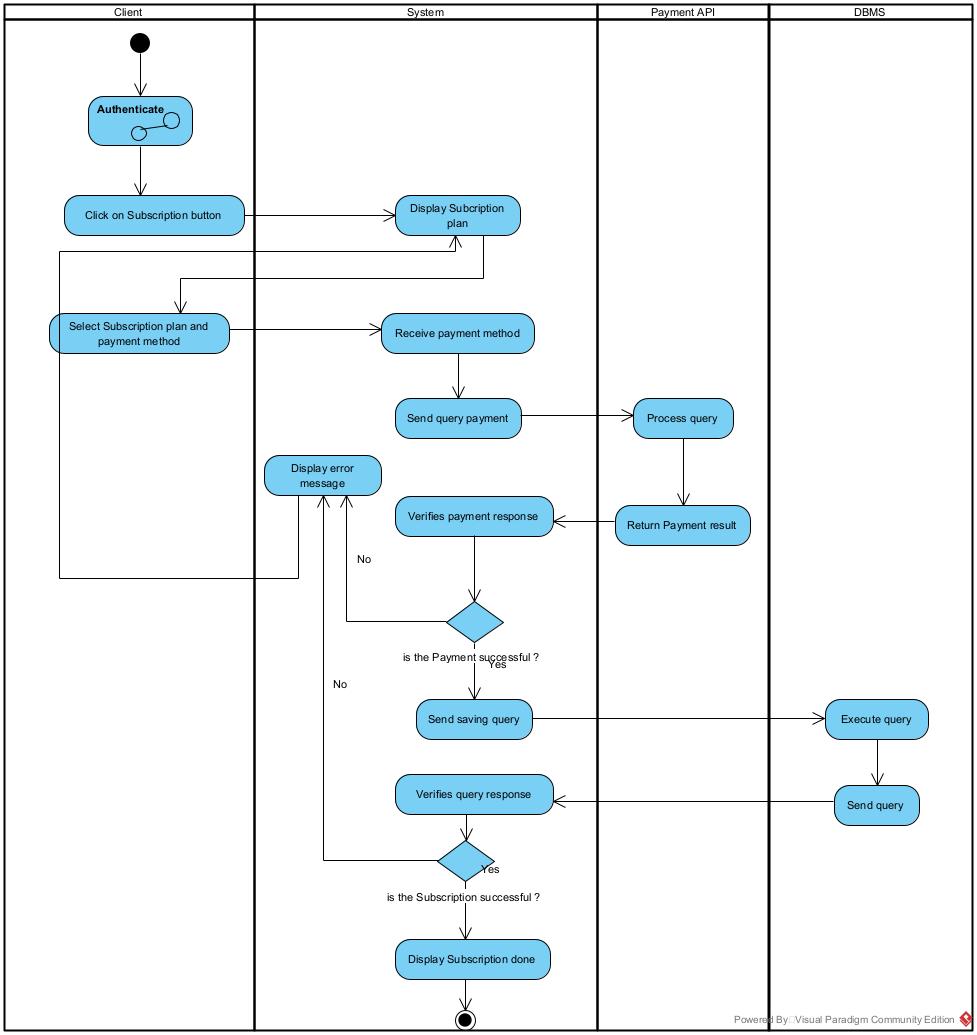
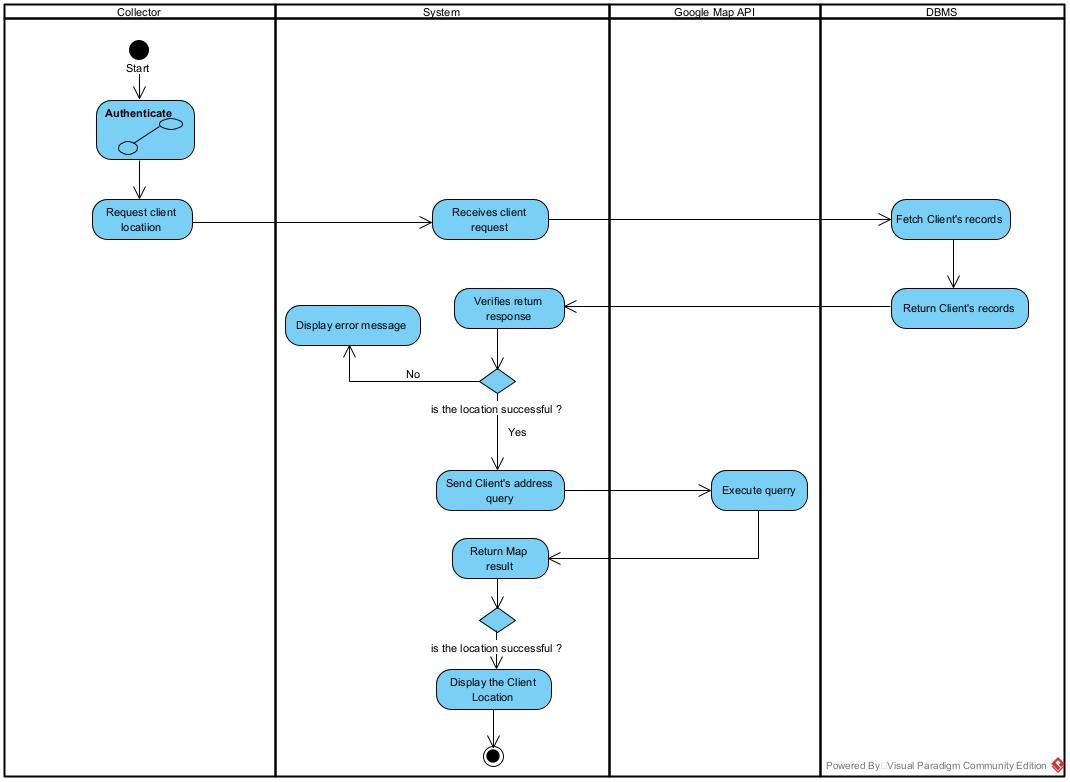


Figure 11: Make Subscription activity diagram

1. **Activity diagram of Make Subscription**



1. **Activity diagram of View Client Location**

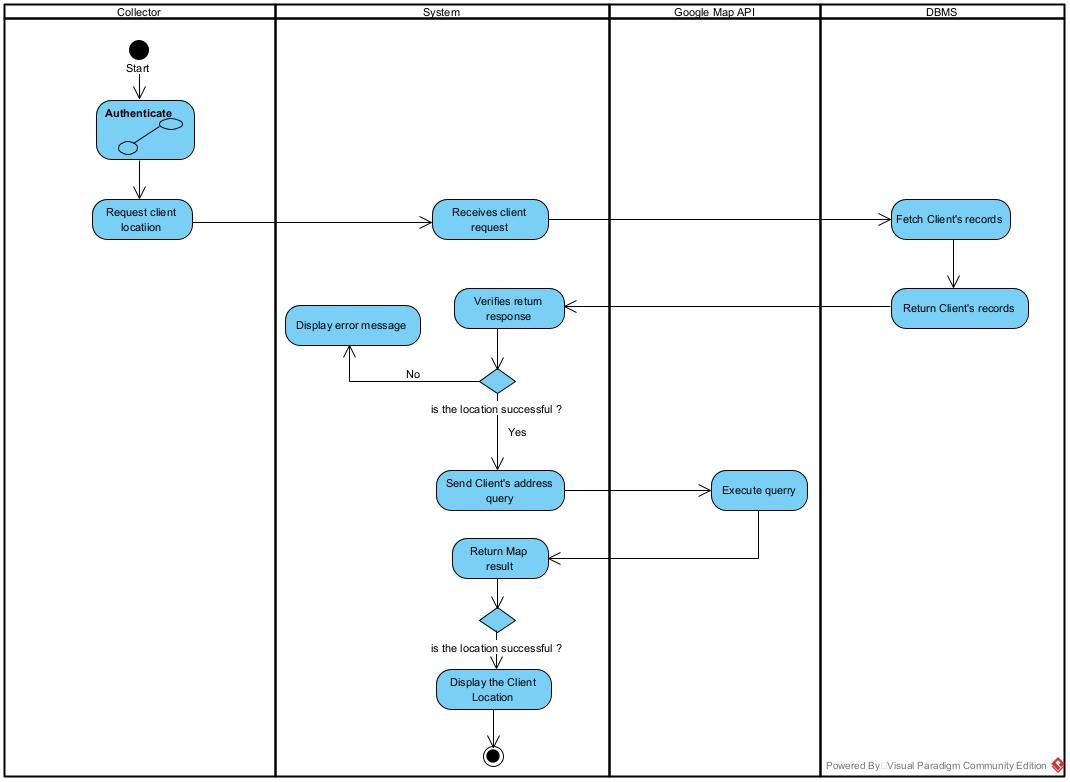


Figure 12: View Client Location activity diagram

1. **SEQUENCE DIAGRAM**
2. **Definition**

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

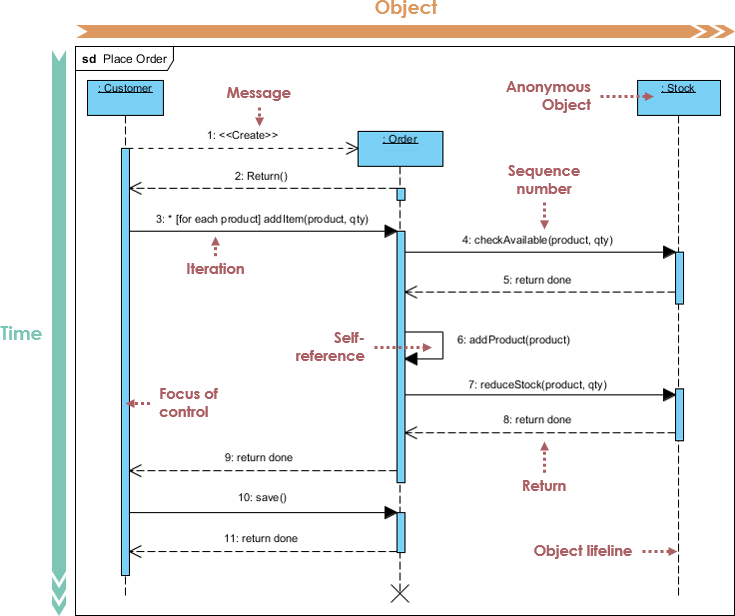
1. **Formalism**

Figure 13: Sequence diagram formalism

1. **Component of sequence diagram**

Table 20: Component of sequence diagram

|  |  |  |
| --- | --- | --- |
| Element | Notation | Description |
| Lifelines | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps16.jpg | They represent rows or objects instances that participate in the sequence being modelled. |
| Asynchronous Message | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps17.jpg | It is a message that receives an indirect response. |
| Synchronous Message | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps18.jpg | It is a message that sends and want response before it continues a process |
| Self-Message | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps19.jpg | Self-message is a kind of message that represents the invocation of message of the same lifeline. |
| Return Message | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps20.jpg | It represents the response of a message. |
| Actor | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps21.jpg | They send and receive message. |
| Combined Interaction Fragment | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps22.jpg | An articulation of interaction diagram, defined by an operator and operands. |
| Object | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps23.jpg | They send and receive messages |

##### **Sequence diagram of Authenticate**

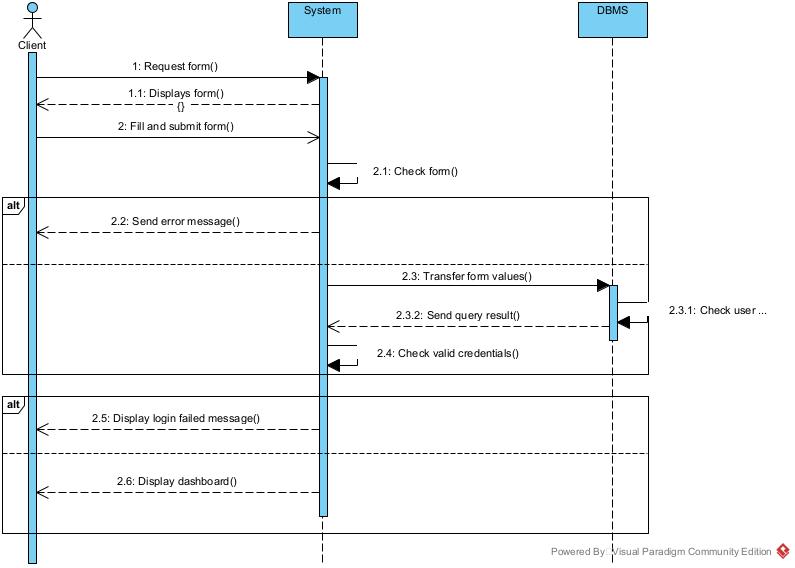


Figure 14: Sequence diagram of Authenticate

##### **Sequence diagram of Account Creation**

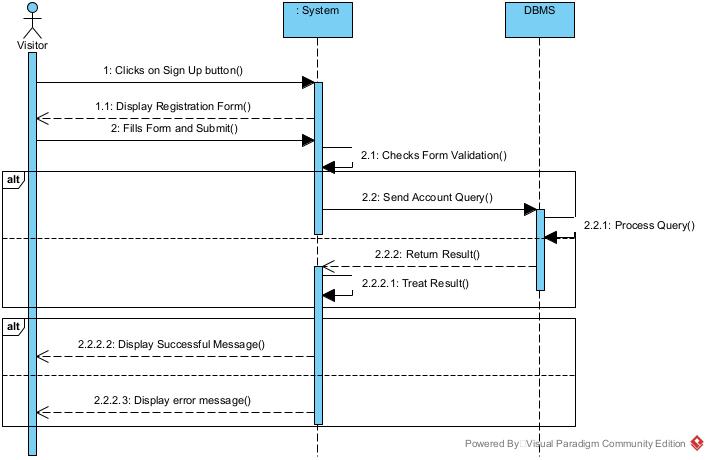


Figure 15: Sequence diagram of Account Creation

##### **Sequence diagram of Make Subscription**

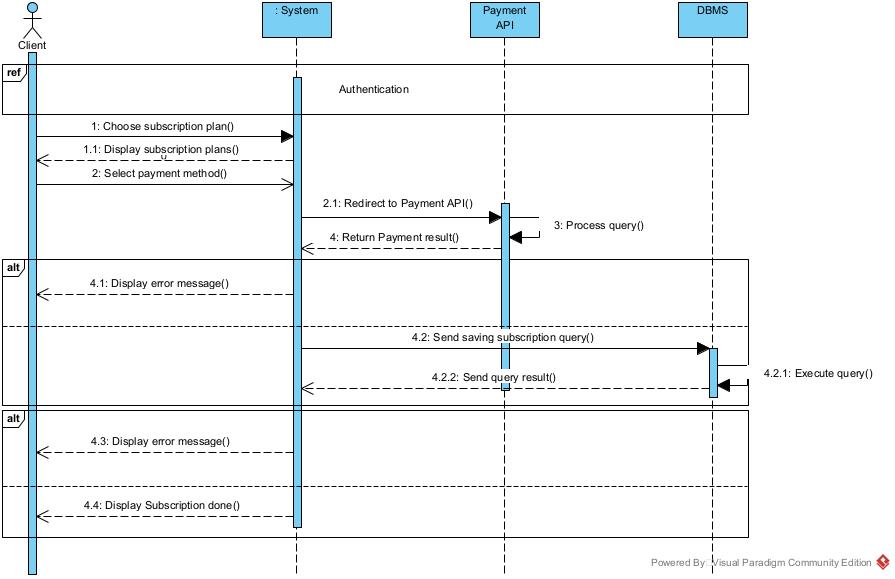


Figure 16: Sequence diagram of Make Subscription

##### **Sequence diagram of View Client Location**

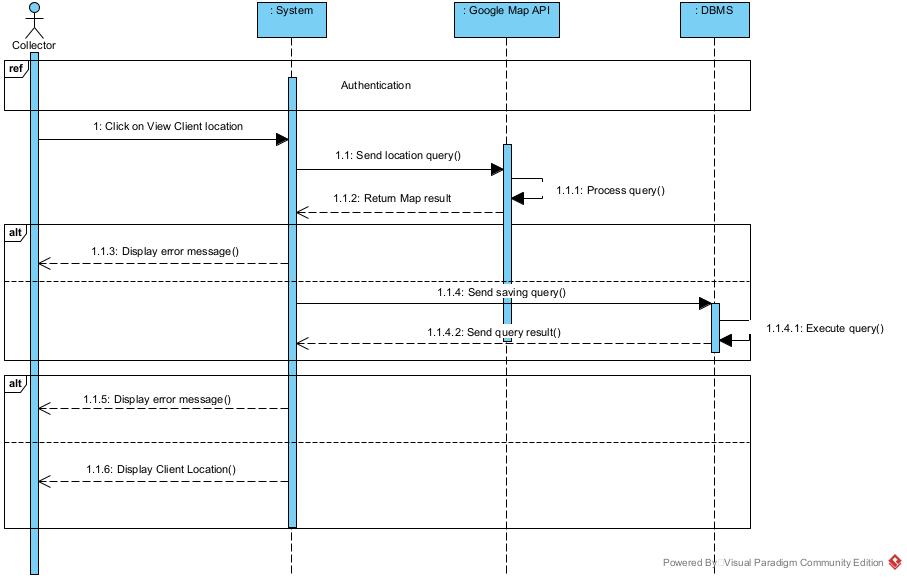


Figure 17: Sequence diagram of View Client Location

1. **COMMUNICATION DIAGRAM**
2. **Definition**

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

1. **Formalism**

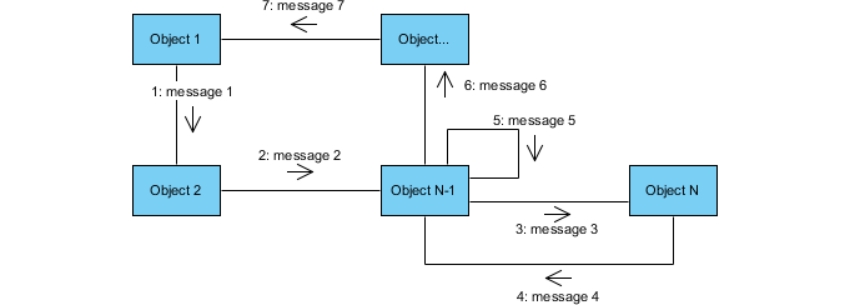


Figure 14: Communication diagram formalism

1. **Components of a communication diagram**

Table 21: Component of a communication diagram

|  |  |  |
| --- | --- | --- |
| Element | Notation | Description |
| Message | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps24.jpg | Designs a particular communication between lifelines. |
| Actors | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps25.jpg | A role play by an entity that interacts with the subjects |
| Link | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps26.jpg | It initiates an association it connects two objects together for them to communicate. |
| Object | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps27.jpg | An actor represents an individual participant in the interaction conversation |

1. **Communication diagram of Authenticate**

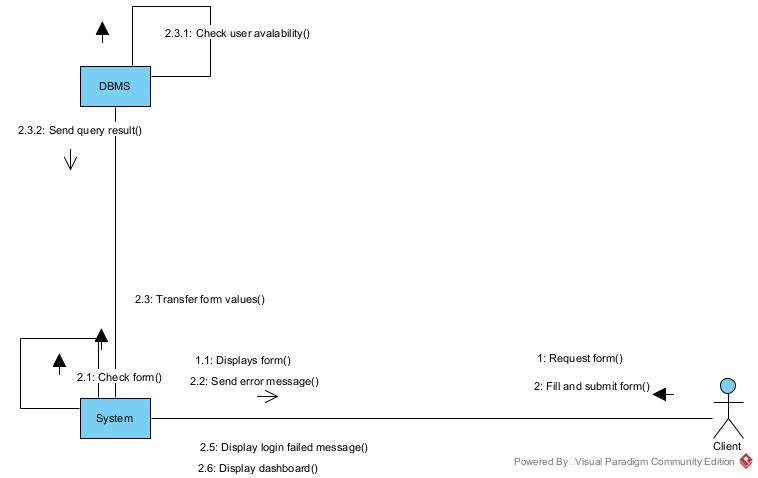


Figure 18: Authenticate Communication diagram

1. **Communication diagram of Make Subscription**

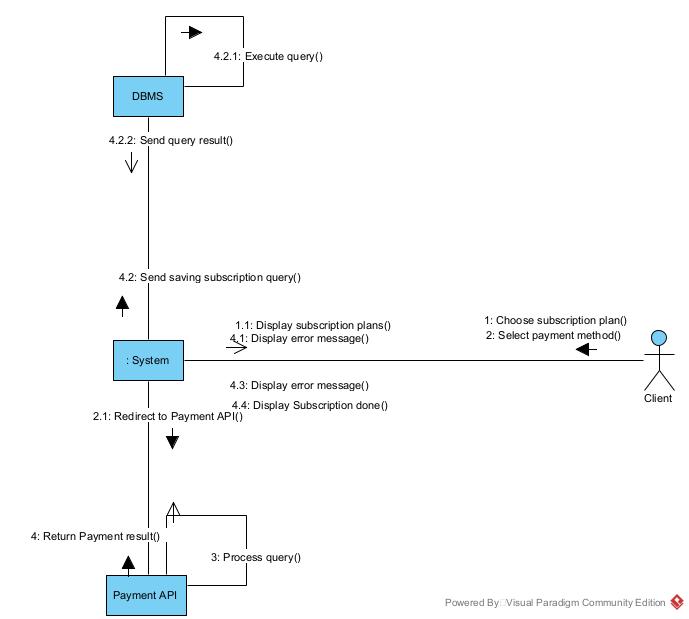


Figure 19: A Make Subscription Communication diagram

1. **Communication diagram of View Client Location**

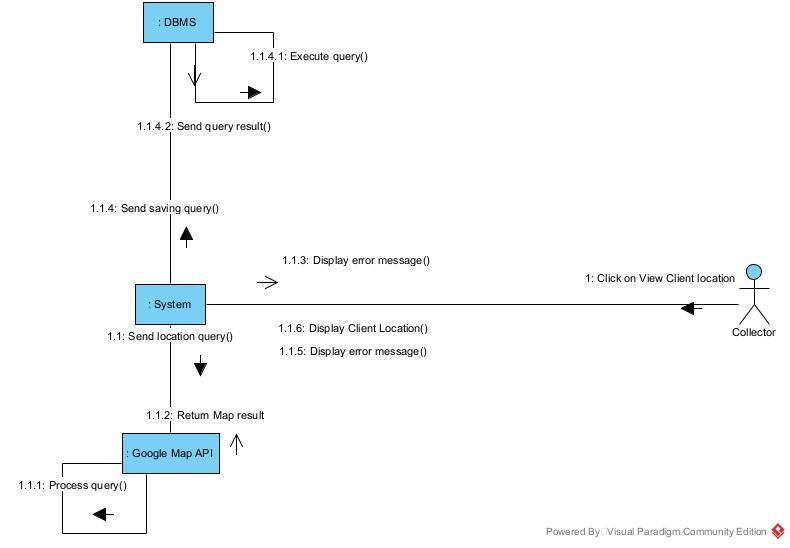
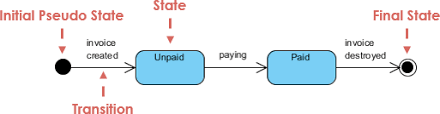


Figure 20: View Client Location Communication diagram

1. **STATE MACHINE DIAGRAM**
2. **Definition**

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

1. **Formalism**



*Figure 18: Formalism of a state machine diagram*

2. **Component of a state machine diagram**

Table 22: Component of state machine diagram

|  |  |  |
| --- | --- | --- |
| Element | Notation | Description |
| State | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps32.jpg | Models a situation during which a certain invariant condition holds. |
| First (Initial State) | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps33.jpg | It represents a default vertex, that is, a source for a single transaction to the default or composite state. |
| Final state | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps34.jpg | A state specifying that the enclosing region is complete |
| Transition | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps35.jpg | A direction relation between a source and a target vertex. |
| Choice pseudo state | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps36.png | A diamond symbol that indicates a dynamic condition with branched potential results |
| Terminate | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps37.pngC:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps38.png | Implies that the execution of a state by means of it context is terminated. |
| Diagram overview | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps39.pngC:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps40.png | A placeholder for the linked states in a state machine diagram. |

1. **State machine diagram of Account Creation**

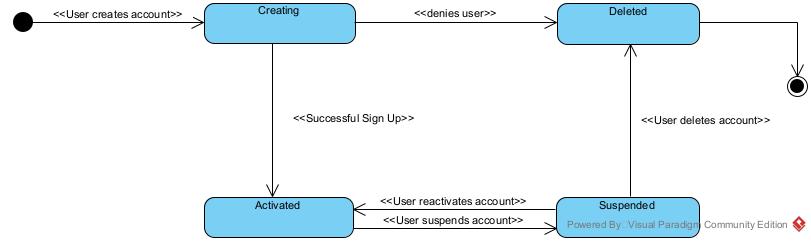


Figure 21: Account Creation State machine diagram

1. **State machine diagram of Make Subscription**

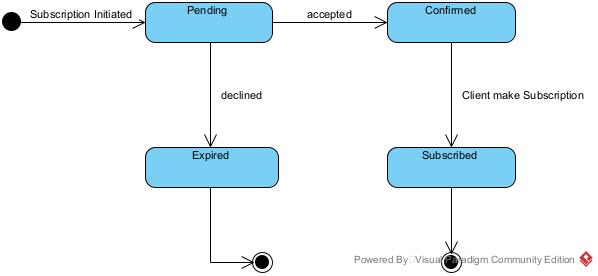


Figure 22: Make Subscription State machine diagram

CONCLUSION

During the analysis phase, we selected a software development process and modelling language. Next, we elaborated on the functional requirements of our system. We reviewed the use case diagram, which outlines the relationship between actors and the actions they can perform in the system, as well as the communication diagram that illustrates the architecture of the system based on object-oriented programming. We also analysed the sequence diagram, which delineates the flow of messages between elements in the system. Lastly, we reviewed the activity diagram, which displays the workflow of our system. We will proceed to the conception phase, during which we shall highlight the Technical branch of our system, accompanied by relevant diagrams.

# FILE 4: CONCEPTION PHASE

Preamble

The conception phase is the part of the document that shows the link between the analysis and the realization phases. It is a continuation of the analysis phase, which represents the technical aspects used in modelling our system.

Content Overview

INTRODUCTION

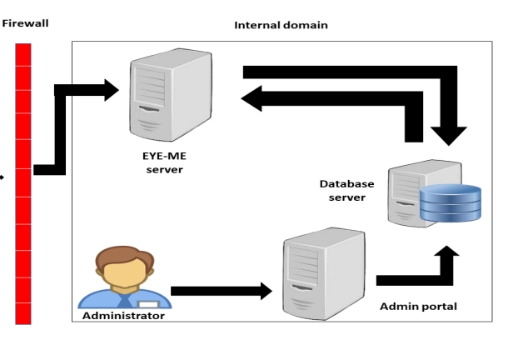
1. TECHNICAL BRANCH
2. GENERIC DESIGN
3. CAPTURE OF TECHNINCAL NEEDS
4. RELATED UML DIAGRAMS
5. CLASS DIAGRAM
6. PACKAGE DIAGRAM

CONCLUSION

INTRODUCTION

The concept stage will outline the precise requirements, characteristics and activities needed to fulfil the operational needs of the suggested system, as defined during the analysis stage. The objective during this phase is to recognize and examine key components (either hardware or software), configuration (network capacity), processes and procedures, which are indispensable for the system to achieve its desired outcomes. We will scrutinize several diagrams such as the class diagram, state machine diagram and package diagram.

1. TECHNICAL BRANCH
2. **GENERIC DESIGN**
3. **Hardware diagram of the system**

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

C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps42.jpgC:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps43.png

Zerodech Server

Figure 23: Hardware diagram

1. **High Level Architecture of the System**

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

1. **CAPTURE OF TECHNICAL NEEDS**
2. **Physical Architecture**

The design of the DBMS depends on its architecture. An n-tier architecture partitions overall system into related but separated n modules, which can be independently modified, altered, changed or replace. A large amount of data on web servers, personal computers (pc) and others are link with networks with the help of basic client or server architecture.

Within the scope of our project, we made use of the 3-tier architecture this architecture separate it tiers from each other based upon the user and the manipulated data in the database. Each layer has a well-defined communication interface, and the evolution of the layer is independent of the other. The 3-tier of our system is made up of:

* + The client tier, which is also known as our presentation interphase.
  + Application Tier, which represents our webserver.
  + The data tier, which represents our DBMS server

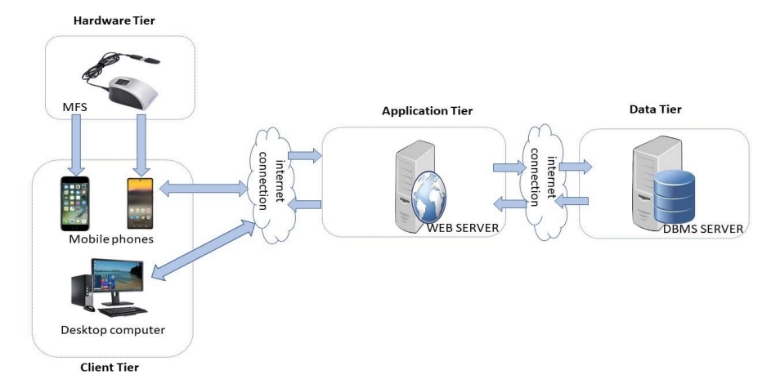
C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps47.png

Figure 24: n-tier Architecture (Source: <https://www.pinterest.es/pin/752241943987565648/> )

1. **Logical Architecture**

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

* Model: The lowest level of the pattern, which is responsible for maintaining data.
* View: This is responsible for displaying all or a portion of data to the user.
* Controller: It handles software codes that controls the interactions between the model and the view.

MVC is popular as it isolates the application logic from the user interface and supports separation of concerns. Here the controller receives all requests for the application then works with the model to prepare data needed by the view. The view then uses the data prepared by the controller to produce a final response. The MVC can be represented as follows:

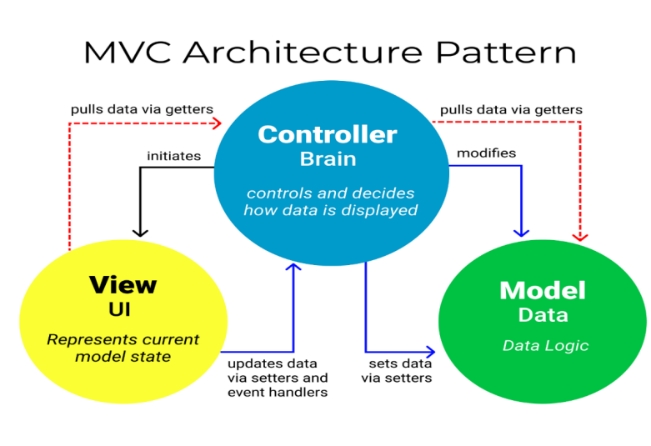


Figure 25: The MVC architecture (source: https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/)

1. **RELATED UML DIAGRAMS**
2. **CLASS DIAGRAM**
3. **Definition**

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

1. **Formalism**

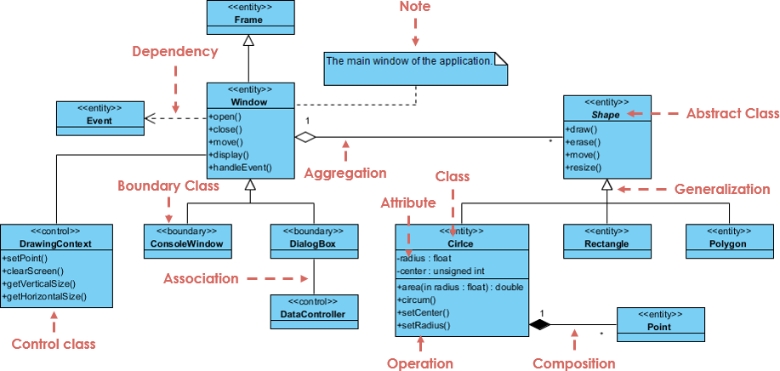


Figure 27: Class diagram formalism

1. **Component of a Class diagram**

Table 21: Components of class diagram

|  |  |  |
| --- | --- | --- |
| Element | Notation | Description |
| Class | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps53.jpg | A class is an element that defines the attributes and behaviors that an object can generate |
| Generalization | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps54.jpg | it a relationship between a whole thing (called superclass) and a more specific thing (called subclass) |
| Association | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps55.jpg | It is a general type of relationship between elements, it may include cardinality, roles etc. |
| Aggregation | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps56.jpg | If the parent of the aggregate is deleted, the children are not deleted. |
| Composition | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps57.jpg | If a parent of a composite is deleted all its parts are deleted with it. |
| Dependency | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps58.jpg | It existed between two classes, if one changes it may cause the change in the order, but the other way around. |

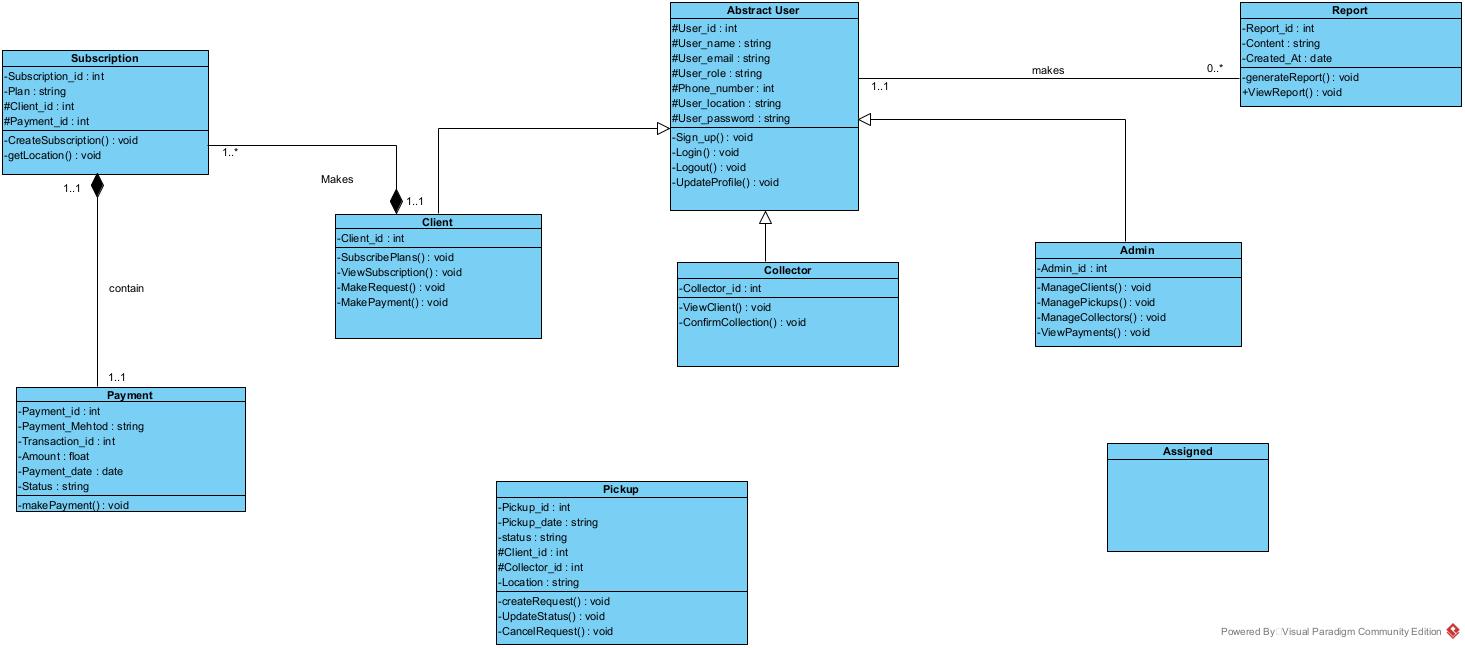


Figure 28: Zerodech Class diagram

1. **PACKAGE DIAGRAM**
2. **Definition**

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

1. **Formalism**

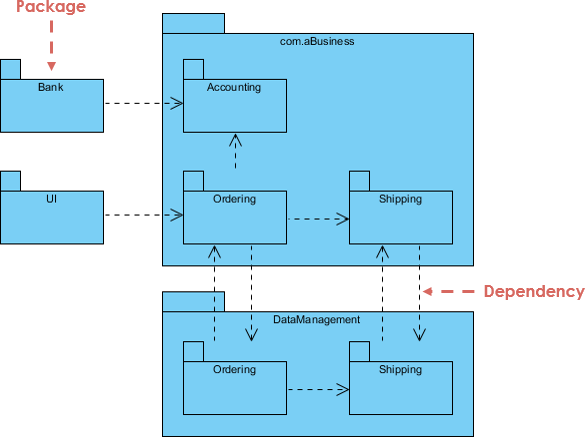


Figure 29: Package diagram formalism

1. **Component of package diagram**

Table 23: Component of package diagram

|  |  |  |
| --- | --- | --- |
| Element | Notation | Description |
| Package | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps60.jpg | A package is a namespace use to group related elements; it is a mechanism used to group elements into a better structure in a system. |
| Package import | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps61.jpg | A relationship Indicate that, functionality has been imported from one package to another. |
| Package access | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps62.jpg | A relationship Indicates that one package requires assistance from the function of another package. |
| Package merge | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps63.png | A relationship shows that, the functionality of two packages are combines to a single function. |

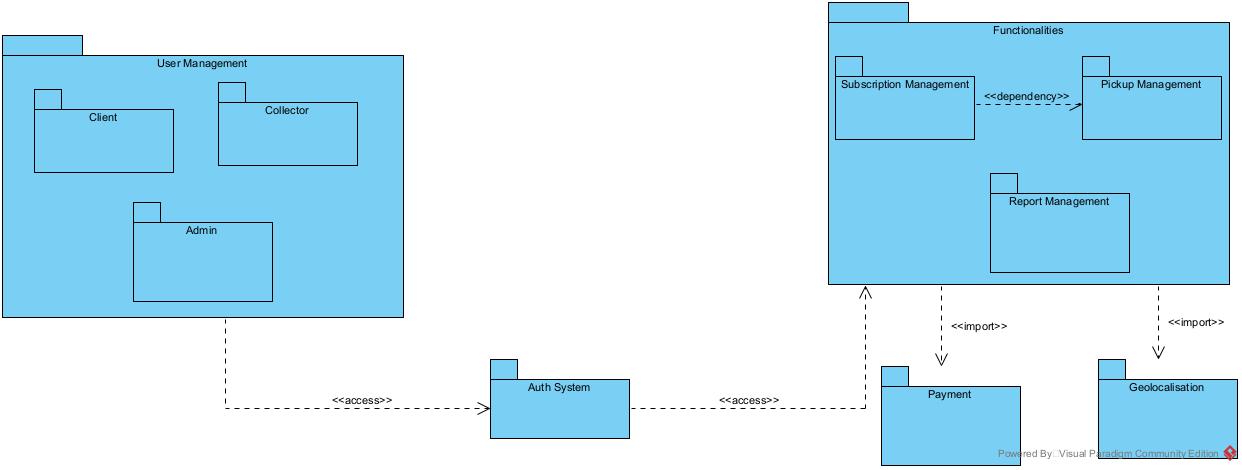


Figure 32: Zerodech package diagram

CONCLUSION

We presented the structure of our system and its integration into the existing technical architectures. In the initial stage, we aimed to investigate various aspects of our system. Primary actions included assessing the technical limitations, identifying the system's components with their classification and deployment onto relevant deployment targets. We then concluded this phase by evaluating the interactions between different components and participants of our system. Moving forward, our report will now progress to the implementation phase, which will entail concentrating on executing our system-related elements.

# FILE 5: CONCEPTION PHASE

Preamble

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

Content Overview

INTRODUCTION

I. COMPONENT DIAGRAM

II. DEPLOYMENT DIAGRAM

CONCLUSION

INTRODUCTION

Here in the realization phase, we will concentrate on building or implementing our solution, we will review diagrams concerning the physical aspects of the system, such as libraries, documents and the physical topology of the system components when the software is deployed. Based on the various analyses and concepts developed earlier, this phase is as critical as the previous ones. It focuses on the physical topology of deployed components, the tools used to realize the application, the chosen programming language, and the overall architecture as it is implemented in code and infrastructure.

1. COMPONENT DIAGRAM
2. **Definition**

It is mainly use to describe the dependencies between various software component such as the dependency between executable file and source files. It represent the internal structure of your software (Software component). You must represent the different classes, libraries, modules, or files. It represents the technical architecture of your system.

1. **Formalism**

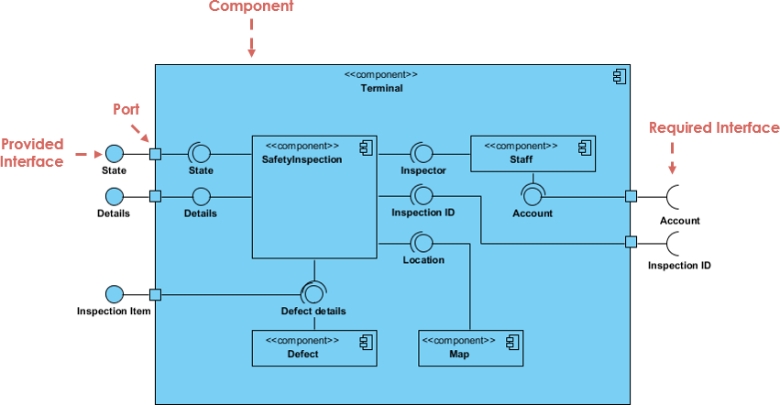


Figure 35: Component diagram formalism

1. **Component of Component diagram**

Table 25: Component of deployment diagram

|  |  |  |
| --- | --- | --- |
| Element | Notation | Description |
| Dependency | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps65.jpg | A dependency is a relationship that signifies that a single or a set of model elements requires other model elements for their specification or implementation. |
| Port | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps66.jpg | A port is often used to help expose required and provided interfaces of a component. |
| Interface | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps67.jpg | It is a point of interaction between a component and its environment or other components. It specifies the services that a component provides or requires from other components |
| Component | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps68.jpg | It represents a self-contained and replaceable unit that interacts with other components via provided and required interfaces |

1. **Zerodech Component diagram**

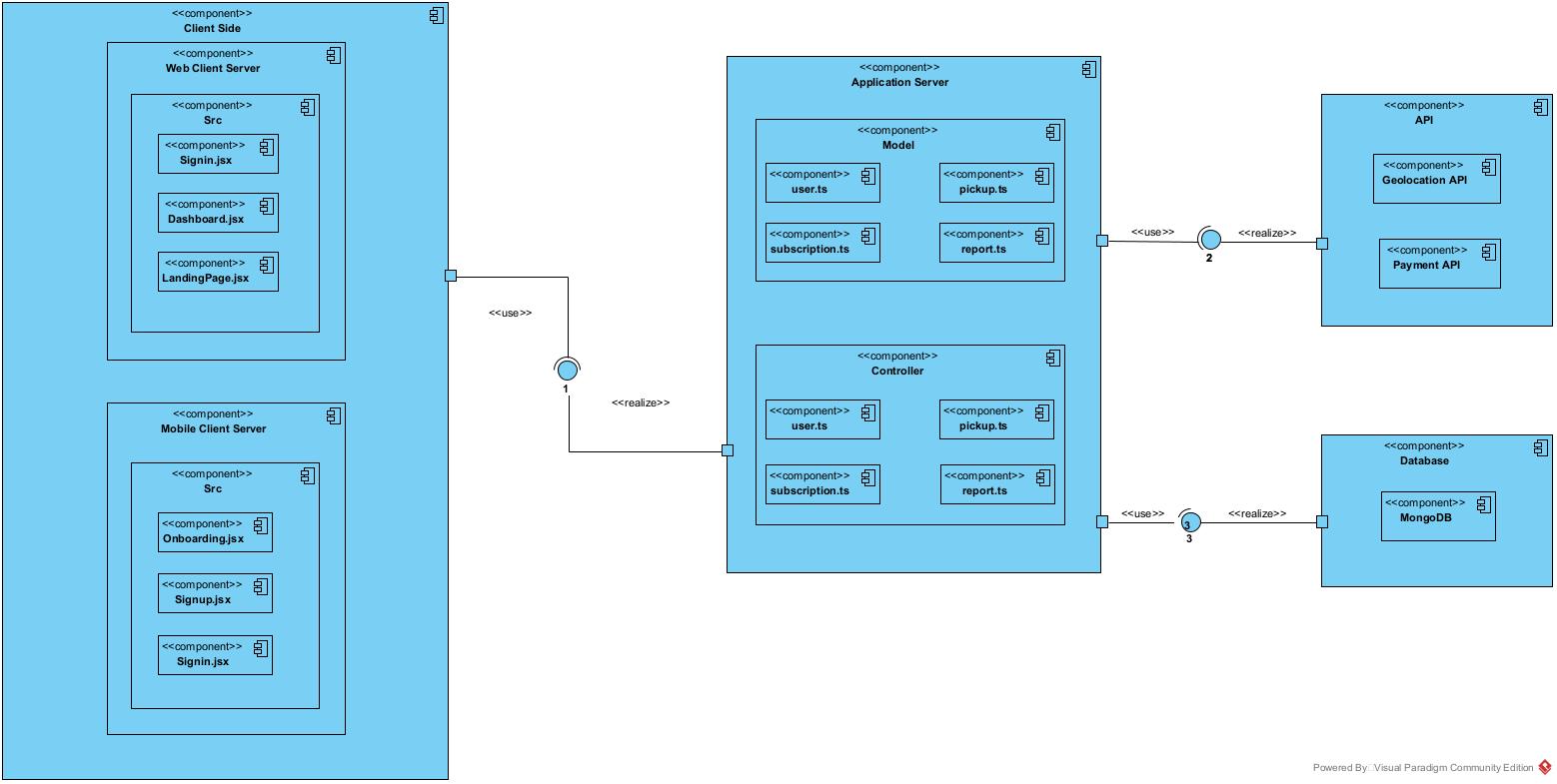
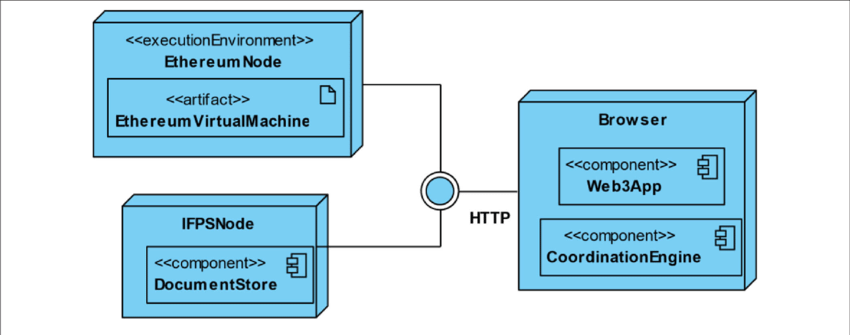


Figure 36: Zerodech component diagram

1. DEPLOYMENT DIAGRAM
2. **Definition**

A deployment diagram is a visual representation of the physical architecture of a computer system. The diagram represents the physical disposition of materials resources that constitute the system and shows the component repartition (software elements) that are executed inside these materials. It shows how the different hardware and software components of a system are connected and deployed on different nodes.

1. **Formalism**



*Figure 33: Formalism of Deployment diagram*

1. **Component of Deployment diagram**

Table 24: Component of deployment diagram

|  |  |  |
| --- | --- | --- |
| Element | Notation | Description |
| Node | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps71.jpgC:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps72.jpg | A hardware or software object, shown by a three-dimensional box |
| Artifact | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps73.jpgC:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps74.jpg | An artifact is a major product, which is produced or used during the development of a software. E.g. diagrams, data models, setup scripts |
| Component | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps75.jpg | It represents a modular part of a system that encapsulates its content and whose manifestation is replaceable within it environment. |
| Association | C:\Users\jasam\AppData\Local\Temp\ksohtml8540\wps76.png | An association helps to connect two nodes together which permits them to communicate together |

1. **Zerodech Deployment diagram**

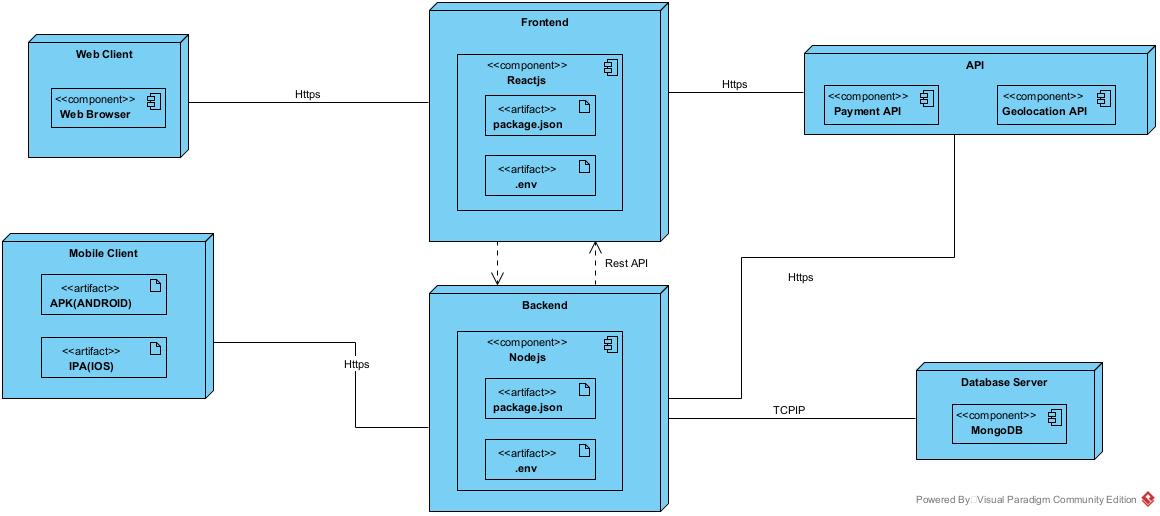


Figure 34: Zerodech deployment diagram

CONCLUSION

During our realization phase, we successfully developed and implemented our application. To achieve this, we utilized our analysis and conception phase. We also created deployment and component diagrams that illustrate the modules, files, and assets, as well as how different elements interact with each other within our system. Our next step is to move onto the functionalities testing phase, where we will assess the efficacy of the various modules present in our app for different users.

# FILE 6: TEST OF FUNCTIONALITIES

Preamble

In this report, we will explore the different aspects of the test of functionalities and its significance in the software development life cycle.

Content Overview

INTRODUCTION

1. TEST OF FUNCTIONALITIES
2. TEST SHOWCASES

CONCLUSION

INTRODUCTION

The test of functionalities plays a crucial role in ensuring the quality and reliability of software systems. By thoroughly examining each functionality and validating its behavior, developers can identify potential issues early in the development process. This helps in reducing the overall cost and effort required for bug fixing and maintenance.

1. TEST OF FUNCTIONALITIES
2. **What is Unit Testing**

Unit testing involves testing the smallest parts of an application, such as functions or methods, independently from the rest of the application. The primary goal is to validate that each unit of the software performs as expected.

1. **Unit Testing Importance**

Unit testing ensures that each component of the application works independently and correctly. It helps catch bugs early in the development process and guarantees that future changes do not break existing functionality.

1. **Testing Strategy**

We implemented **unit testing** to isolate and test individual parts of the app, specifically the API endpoints and backend services. Unit testing provides fast feedback on code quality and behavior.

1. **Tools Used**

* **Express:** testing framework used for running unit tests.
* **Postman**: For testing API requests.
* **Mongoose**: Used in some instances to mock MongoDB models for tests.

1. **Key Functionalities Tested**
2. User Registration

* Tests ensure that users can successfully register with valid details.
* Handles test cases where invalid inputs or missing fields cause registration failure.

1. User Authentication

* Tests confirm that users can log in with correct credentials.
* Includes scenarios where login fails due to invalid credentials (e.g., wrong password).

1. Subscription Creation
   * Tests verify that subscription are created successfully with all required information.
   * Includes test cases where an error is returned if a user tries to make a duplicate subscription for the same provider.
2. TEST SHOWCASES

PART SEVEN: ANNEX

BIBLIOGRAPHY

WEBOGRAPHY

TABLE OF CONTENTS