الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research

University Abderrahmane Mira of Bejaia
Faculty of Exact Sciences
Department of Computer Science

Master's Thesis

Towards the completion of the Professional Master's degree in

Software Engineering

THEME

Development of a Web Application for managing Interns and Apprentices

Case Study: Bejaia Port Enterprise

Written by:
Mr. TSHUMA Marvellous Courage

Supervised by:
Mrs. AIT HACENE Souhila

Before the jury:

President: Mrs. GADOUCHE Hania

Examiner: Mrs. TASSOULT Nadia

2023/2024

ACKNOWLEDGEMENTS

Grateful acknowledgments are extended to Almighty God, whose strength, courage, and guidance enabled me to complete this work with determination and dedication.

I would like to express my heartfelt gratitude to my supervisor, Mrs. Ait Hacene Souhila, for all the time she dedicated to me, her valuable advice, and her support throughout this work.

I extend my gratitude to the staff of Bejaia Port Enterprise, particularly the head of the Computer Science department, and Mr. Bilal Bidaouche, my internship supervisor, for their kindness, availability, and guidance in addressing all of my inquiries during my internship.

I express my deepest gratitude to the esteemed members of the jury, Mrs. Gadouche Hania and Mrs. Tassoult Nadia, for their valuable time, insightful feedback, and guidance throughout the evaluation of my thesis. Their expertise and thoughtful comments have significantly contributed to the improvement and finalization of this work.

Finally, I would like to express my appreciation to my family and to all those who contributed directly or indirectly to the completion of this work.

Dedications

.

This thesis is dedicated to my beloved family, whose unwavering support and encouragement have been the cornerstone of my academic journey. To my late father, Themba Tshuma, whose wisdom, resilience, and unwavering belief in education continue to inspire me profoundly. His guidance and values have shaped my aspirations and determination, and I carry his memory with me as I pursue this academic milestone.

To my mother, Tryonce Matsweru, your boundless love, sacrifices, and nurturing spirit have been the bedrock of my life and academic achievements. Your strength and perseverance have guided me through every challenge, and this thesis stands as a testament to the values you instilled in me.

To my dear sisters, Thandekile Tshuma and Fungai Chapungu, your steadfast support, encouragement, and sisterly love has been a constant source of inspiration. I am grateful for your belief in my abilities and unwavering support throughout this journey.

To my nephew, Brighton Mlambo, your joyful presence and innocence have reminded me of the importance of resilience and determination in the face of challenges.

I extend my deepest gratitude to my uncle, Peter Hoko, and aunt, Irene Sibhuku, for your guidance, encouragement, and unwavering support throughout my academic pursuits. Your belief in my potential has been a source of strength.

To my brother, Behold Sandile Chikomo, your companionship, encouragement, and shared experiences have enriched my life and strengthened my resolve.

To my best friend, Mzwakhe Mlalazi, your unwavering friendship, encouragement, and belief in my potential have been invaluable in navigating the complexities of academic life.

To my special friend, Shingirirai Dube, your insightful conversations, encouragement, and support have enriched my academic pursuits and broadened my perspective.

To Bhekimpilo Mlambo, Thank you for always believing in me, supporting, and checking up on me throughout my journey of study.

This thesis is a reflection of the love, support, and sacrifices of my family and friends. Your belief in me has been a guiding light, and I dedicate this work to each of you with profound gratitude and appreciation.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

EPB:	Entreprise Portuaire de Bejaia
ISD:	Information System Department
HRD:	Human Resources Department
UP:	Unified Process
CSS:	Cascading Style Sheet
HTML:	Hyper Text Markup Language
UML:	Unified Modeling Language
API:	Application Programming Interface
AI :	Artificial Intelligence
IP :	Internet Protocol
IDE:	Integrated Development Environment
VCS:	Version Control System
HTTP:	Hypertext Transfer Protocol
AWS:	Amazon Web Services
REST:	Representational State Transfer
DNS:	Domain Name System
IOS:	iPhone Operating System
URL:	Uniform Resource Locator
CI:	Continuous Integration
CD:	Continuous Deployment
GCP:	Google Cloud Platform
CMS:	Content Management System
RIA:	Rich Internet Application
UI:	User Interface
UX:	User Experience
NOSQL:	Not Only SQL
VS CODE:	Visual Studio Code

GENERAL INTRODUCTION

General Introduction

In the past, information was manually recorded on paper, leading to various problems such as significant time loss in searching for information and the degradation of documents. Today, however, we are experiencing a significant technological evolution across all sectors, driven by advancements in computing. Manual data storage and processing have been automated across numerous domains, including administrations, businesses, associations, and educational institutions. In this context, data is centralized into databases managed by sophisticated database management systems.

The Bejaia Port Enterprise (EPB) is one of the largest companies in Algeria. To maintain its competitive edge and ensure continuous and sustainable development, it must equip itself with modern tools and technologies. However, it has been observed that the current system for managing interns and apprentices operates on an outdated desktop application. This system, managed solely by an administrator, lacks essential features critical for effective management, such as task submission, task evaluation, and a communication platform for interaction between interns and supervisors.

My study aims to develop a web application to enhance the management of interns and apprentices within the Human Resources Department (HRD) at EPB. The objective is to streamline and improve the overall internship and trainee management process within the company. Some of the functionalities of the system to be developed include a chat feature to allow trainees and supervisors to communicate effectively, a task submission interface, and a task validation interface.

This thesis is organized into five chapters:

Chapter 1: This chapter introduces the host organization and examines the problem statement. It discusses the project's motivations and outlines the development methodology.

Chapter 2: This chapter covers the background of web applications, including design principles, user interface development, and backend architecture. It also addresses challenges and best practices in deploying and maintaining web-based systems.

Chapter 3: This chapter specifies the system requirements by identifying the actors involved in the application. It includes a use case diagram with textual descriptions to define the system's functionalities.

Chapter 4: This chapter focuses on the analysis and design phase. It starts with system sequence diagrams and concludes with a class diagram representing the company’s data in a database.

Chapter 5: The final chapter focuses on the implementation phase. It presents the development environment and languages used for the project, showcases key interfaces of the application, and concludes with a summary of the main achievements and a general conclusion.

CHAPTER 1

STUDY OF THE EXISTING SYSTEM

Chapter 1: Study of the Existing System

1 Introduction

In this chapter, I will present my host company, **The Bejaia Port Enterprise (EPB)**, along with some basic background information, the problem statement, and the objective of our work.

2 Presentation of the Company

2.1 Historical Background

Located in the heart of the Mediterranean region, the city of Bejaia has numerous natural sites and historical remains dating back over 10,000 years, as well as numerous archaeological sites containing artifacts dating back to the Neolithic period. Bejaia played a significant role in knowledge transmission in the Mediterranean basin, thanks to the dynamism of its port, regional security, sound policies, and customs advantages. Bougie managed to attract many powerful merchants. [1]

2.2 Missions and Activities of EPB

The port of Bejaia plays a very important role in international transactions due to its location and geographical position. Today, it is ranked as the second largest port in Algeria for general goods and the third largest oil port. It is also the first port in the Mediterranean basin certified to ISO 9001:2000 for all its services, having implemented a quality management system. This marks a step in the process of continuous improvement of its services for the benefit of its clients. The Bejaia Port Enterprise has achieved other successes since then, notably being certified to ISO 14001:2004 for environmental management and to the OHSAS 18001:2007 standard for occupational health and safety. [1]

The management, operation, and development of the port area are the essential tasks of The Bejaia Port Enterprise, with the aim of promoting the country's foreign trade. It is responsible for law enforcement and security within the country. It is tasked with maintenance, development, renewal, and creation of infrastructure. The Bejaia Port Enterprise also provides commercial services, including towing, handling, and stevedoring. [1]

The main activities of the enterprise are:

- ✓ Operation of port equipment and facilities
- ✓ Execution of maintenance work on port structures
- ✓ Monopoly on stevedoring operations

✓ Monopoly on towing operations

✓ Port police and security within the geographical limits of the public port area

2.3 Organization Chart of the EPB

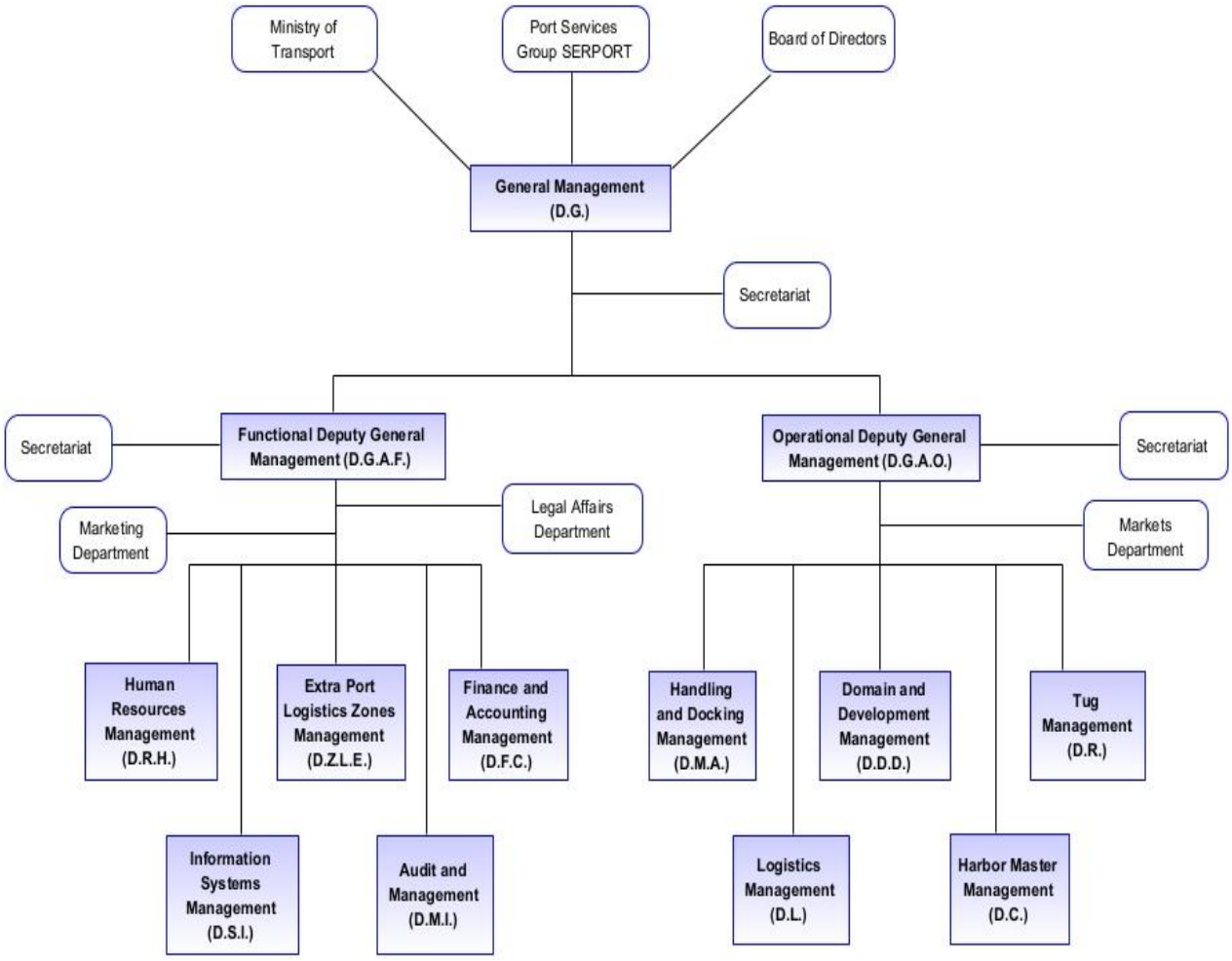

Figure 1. 1: The general organizational chart of the EPB [1]

2.4 Information System Department (ISD):

The purpose of this department is to provide essential technological support for operations, manage and safeguard data, facilitate decision-making, enhance communication and collaboration, manage IT resources, and provide cybersecurity measures.

Figure 1. 2 Missions of information system department [1]

2.5 Human Resources Department (HRD):

It is responsible for planning, organizing, and executing all actions related to human resources management while ensuring strict compliance with laws and social regulations. It performs the following tasks:

✓ Implementation of the remuneration policy, recruitment, and staff training.

✓ Management of staff careers (recordkeeping).

✓ Management of general resources (routine purchases, vehicle fleet, insurance, etc.).

3 Problem Statement

After presenting the scope of the study, we observed that within the framework of intern and apprentice management, the Human Resources Department (HRD) currently operates on an outdated desktop application. This system, managed solely by an administrator, lacks vital features crucial for effective management. Notably, there is a glaring absence of communication channels between supervisors and interns, impeding collaboration and feedback exchange. Additionally, interns encounter hurdles in task submission, given the system's static nature, primarily serving as a data repository rather than fostering interactive engagement. Task validation takes some time since tasks are submitted and validated manually. This hampers the efficiency and effectiveness of intern and apprentice management processes.

4 Proposed Solution

To address the deficiencies outlined in the current system, a modernized web-based application is proposed. This application will feature comprehensive functionalities tailored to streamline intern and apprentice management:

Communication Platform: Implementing real-time communication channels between supervisors and interns to facilitate seamless collaboration and feedback exchange.

Task Submission Interface: Providing a user-friendly platform for interns to submit tasks and supervisors to review, validate, and provide feedback, enhancing workflow efficiency.

Task Evaluation: Introducing a reliable evaluation system to assess internal performance, ensuring clear criteria and consistent feedback to support internal development.

Interactive Dashboard: Developing a dynamic dashboard for supervisors to monitor intern progress, deadlines, track task statuses, and deliver timely guidance and support**.**

User Authentication and Authorization: Ensuring data security through robust user authentication mechanisms and access controls to safeguard sensitive information.

Document Management: Implementing a centralized document repository for storing interns related documents securely and facilitating easy access and retrieval.

Automated Workflows: Streamlining administrative tasks such as internship assignment, evaluation, and documentation through automated workflows, reducing manual intervention and enhancing efficiency.

In the next section, we move on to selecting the development methodology. We begin by defining the development methodology, followed by the presentation of the chosen Unified Process (UP) method for developing our application.

5 Work Methodology

The choice of a software development methodology is a critical process that depends on the nature of the project itself, its size, and its complexity.

5.1 Development Methodology

Software development methodology is defined as a framework for developing information systems, focusing on planning and organization. It benefits both teams and customers by improving efficiency and adaptability to changes.

To develop our web application, we choose the Unified Process (UP) based on the Unified Modeling Language (UML), a solution that adapts to all types of projects because it is guided by use cases, directed towards architecture, and iterative and incremental.[6] [7]

5.2 The Unified Process [8]

The Unified Process (UP, for Unified Process) is a software development process associated with UML. It is "iterative and incremental", centered on architecture driven by use cases and driven by risks.

UP is an iterative and incremental process: the project is divided into short-duration iterations that help track overall progress better. At the end of each iteration, an executable part of the final system is produced, incrementally.

UP is centered on architecture: any complex system must be broken down into modular parts to ensure easy maintenance and evolution. This architecture must be modeled in UML.

UP is driven by use cases: the project is conducted taking into account the needs and requirements of users. The future system's use cases are identified and accurately described.

UP is driven by risks: the project's major risks must be identified as early as possible and, most importantly, mitigated as quickly as possible.

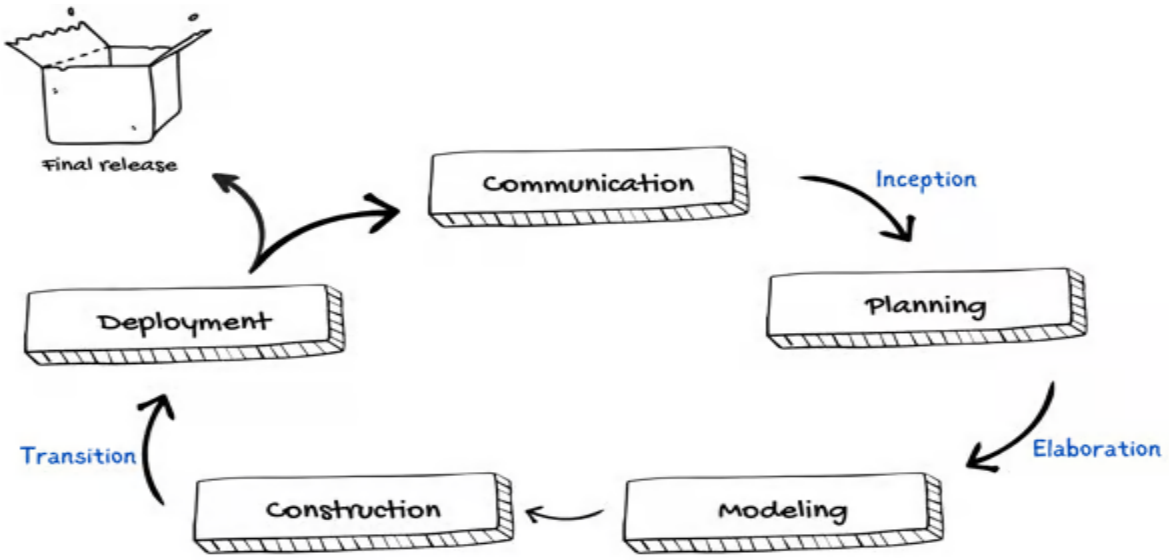

Figure 1. 3: A visual representation of the Unified Process

5.3 Phases

We can represent a unified process model as a series of cycles. Each cycle ends with the release of a new system version for the customers. We have four phases in every cycle:

i. Inception
ii. Elaboration
iii. Construction
iv. Transition

5.4 Inception

The main goal of this phase involves delimiting the project scope. This is where we define why we are making this product in the first place. It should have the following:

- What are the key features?
- How does this benefit the customers?
- Which methodology will we follow?
- What are the risks involved in executing the project?
- Schedule and cost estimates.

5.5 Elaboration

We build the system given the requirements, cost, and time constraints and all the risks involved. It should include the following:

- Develop with the majority of the functional requirements implemented.
- Finalize the methodology to be used.
- Deal with the significant risks involved.

5.6 Construction

This phase is where the development, integration, and testing take place. We build the complete architecture in this phase and hand the final documentation to the client.

5.7 Transition

This phase involves the deployment, multiple iterations, beta releases, and improvements of the software. The users will test the software, which may raise potential issues. The development team will then fix those errors.

6 The formalism UML

UML, short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. [9]

7 Conclusion

Throughout this chapter, I have introduced EPB, which is the hosting organization, as well as the issues encountered within the company and the objective of providing them with an intern management system that meets their expectations. The study of the existing system allowed us to have a clearer vision of the project and its requirements. Therefore, we were able to choose the approach to follow in order to successfully carry out our project.

CHAPTER 2

BACKGROUND OF WEB APPLICATIONS

Chapter 2: Background of Web Applications

1 Introduction

With the advent of the Internet, native applications began using the network to exchange data between clients. However, the arrival of the web opened new perspectives by offering the possibility of distributing information via static websites. Thanks to the evolution of web technologies, new possibilities have emerged, allowing the development of interactive and dynamic web applications.[2]

2 Definition

A Web Application (or web app) is an application hosted on a server and accessible from a web browser. Unlike a mobile application, no installation is required, opening the door to many benefits. Businesses must exchange information and provide services remotely. They use web applications to connect with customers conveniently and securely.[2] [3]

3 Characteristics and Benefits of a Web Application [4]

Web Applications have four characteristics that distinguish them from other types of applications, namely:

✓ They require a single development for any device: A single development in HTML5 is sufficient for any operating system.

✓ They do not need to be downloaded: The application is hosted on a server and is accessible from a browser.

✓ They are portable: Web Applications are accessible from any installed browser (Firefox, Safari, Chrome...).

✓ They appear as results in traditional search engines: Since they do not need to be downloaded, you will not find them in app stores, but they will accordingly appear in search engines such as Google.

4 How do Web Applications work?

Web Applications have a client-server architecture. Their code is divided into two components: client-side scripts and server-side scripts.

4.1 Client-side architecture

Client-side scripting concerns user interface functionality, such as the presence of buttons and drop-down lists. When the end user clicks on the Web Application link, the web browser loads the client-side script and displays the graphics as well as text for the user to interact with them. The user can, for example, read content, watch videos or fill out a contact form. Actions such as clicking the submit button are transmitted to the server as a request from the client.[3]

4.2 Server-side architecture

The server-side script takes care of data processing. The Web Application server processes and responds to client requests. Requests generally concern sending additional data, modifying or saving new data. For example, if the user clicks the Learn More button, the web application server will return content to the user. If he clicks on the Submit button, the application server will save the user's data in the database. In some cases, the server responds to the data request and returns the full HTML page to the client. This action is called server-side rendering.[3]

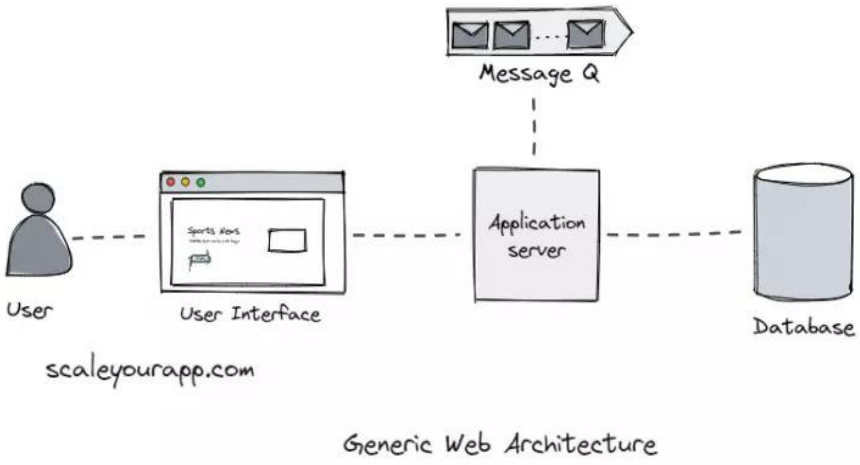

Figure 1. 4: Web applications architecture [3]

5 What is the difference between a Web Application and a Website?

In the early days of the Internet, Websites were much less functional than Web Applications and were only capable of providing information to users through static content. It was necessary to install and run software with complex functionalities. Web Applications were designed to bridge the gap between software and static sites. They had interactive features and user elements like software but were provided using a Web browser url. Since then, Web technology has evolved dramatically, and most of today's Websites are complex Web Applications. [3]

6 What is the difference between a Web Application and a Native Application?

A native application is a computer program specifically designed for a particular user environment. Mobile Applications are the most common Native Applications and are developed with specific programming code. Unlike Web Applications, Native or Mobile Applications are downloaded by the user onto their mobile device, typically from app stores. Native applications can only be accessed from the device on which they were downloaded, and users must download updates themselves.

Developers must design different versions of their Mobile Applications if they want to make them available on multiple operating systems like iOS or Android and reach the maximum number of users. They must also ensure that their Mobile Applications comply with app store standards to avoid rejection from the platform.[3]

7 Types of Web Applications

There are ten types of web applications: **Static Web Applications**, **Dynamic Web Applications**, **E-Commerce Web Applications**, **Portal Web Applications**, **Progressive Web Applications, Single-page Web Application (SPA), Multi-page Web Application, Animated Web Application, Rich Internet Web Application (RIA)** and **Content Management system (CMS)** applications. In our case, we will develop a dynamic Web Application. This type of Web Application is technically complex as they use databases whose content is updated every time the user accesses the application. Thus, dynamic web applications offer an interactive, personalized experience to users by providing real-time content, making them ideal for many projects and businesses.[5]

8 Design Principles for Web Applications [10]

Effective website UI design is both pleasant to look at and easy to use. To create such interface you have to understand the fundamentals: how users interact with UI, what they expect from it and what can potentially cause them problems. Here we will look at graphical UI web design principles used for creating beautiful and functional web UI.

User-Centric Design: Prioritize the needs and preferences of the end-users throughout the design process to create a positive user experience.

Responsive Design: Ensure that the web application adapts seamlessly to various devices and screen sizes, providing a consistent experience across desktops, tablets, and smartphones.

Accessibility: Design the application to be accessible to users with disabilities, adhering to accessibility standards such as WCAG (Web Content Accessibility Guidelines).

Intuitive Navigation: Implement clear and intuitive navigation menus and pathways to help users easily find the information or features they need.

Consistent UI/UX: Maintain consistency in design elements, such as colors, typography, and layout, to create a cohesive and familiar user interface and experience.

Performance Optimization: Optimize the performance of the web application by minimizing load times, reducing unnecessary data transfers, and optimizing code and asset delivery.

Scalability: Design the application architecture to scale seamlessly with increasing user traffic and data volume, ensuring that performance and responsiveness are maintained.

Security: Implement robust security measures to protect user data and prevent unauthorized access, including encryption, secure authentication mechanisms, and regular security updates.

Error Handling and Feedback: Provide clear error messages and feedback to users to help them understand and resolve issues encountered while using the application.

Mobile-First Approach: Design the application with a mobile-first mindset, prioritizing mobile usability and responsiveness before considering desktop layouts.

Cross-Browser Compatibility: Ensure that the web application functions correctly across different web browsers, including popular options such as Chrome, Firefox, Safari, and Edge.

Usability Testing: Conduct usability testing with real users to gather feedback and identify areas for improvement in the design and functionality of the web application.

9 Web Application development methodologies and tools [11]

The workflow is built differently in every approach, the tools used vary too, as well as the principles and rules that are endorsed. Here below is a list of the tools and methodologies often adopted in web app development

Agile Methodology: Agile is an iterative approach to software development that emphasizes collaboration, flexibility, and customer feedback. It involves breaking down the development process into small increments called sprints, with continuous testing and adaptation to changing requirements.

Scrum: Scrum is a specific framework within the Agile methodology that organizes development work into short, time-boxed iterations called sprints. It includes roles such as Scrum Master, Product Owner, and Development Team, along with various ceremonies like sprint planning, daily stand-ups, sprint reviews, and retrospectives.

Waterfall Methodology: Waterfall is a traditional sequential approach to software development where each phase (e.g., requirements, design, implementation, testing, deployment) is completed before moving on to the next. It is characterized by its structured and linear nature.

Kanban: Kanban is a visual project management method that emphasizes continuous delivery and flow. It uses a visual board with columns representing different stages of the development process (e.g., to do, in progress, done) to visualize work items and limit work in progress.

DevOps: DevOps is a set of practices that combines software development (Dev) and IT operations (Ops) to shorten the development lifecycle and improve collaboration and efficiency. It involves automation, continuous integration, continuous delivery, and infrastructure as code.

Version Control Systems (VCS): Version control systems like Git are essential tools for managing and tracking changes to source code. They allow developers to collaborate, track changes, revert to previous versions, and manage different branches of code.

Integrated Development Environments (IDEs): IDEs such as Visual Studio Code, IntelliJ IDEA, and Eclipse provide developers with a comprehensive environment for writing, testing, and debugging code. They often include features like syntax highlighting, code completion, and integrated debugging tools.

Testing Frameworks: Testing frameworks like JUnit, Selenium, and Jest are used to automate the testing process and ensure the quality and reliability of software. They include tools for writing and executing automated tests, generating test reports, and managing test cases.

Continuous Integration/Continuous Deployment (CI/CD) Pipelines: CI/CD pipelines automate the process of building, testing, and deploying software changes. They allow developers to continuously integrate code changes into a shared repository, run automated tests, and deploy changes to production environments quickly and safely.

Project Management Tools: Project management tools like Jira, Trello, and Asana help teams organize and track project tasks, timelines, and resources. They provide features for creating and assigning tasks, setting deadlines, and visualizing project progress.

10 Deployment and Hosting [12]

Deployment and hosting are crucial aspects of web application development, ensuring that the application is accessible to users and functions reliably. This section explores the key considerations and strategies involved in deploying and hosting web applications.

10.1 Types of deployment strategies

Recreate Deployment: This is a standard deployment strategy example that requires developers to scale the previous version of the software before the app's deployment. First, the developers shut down the initial version of the app to upload a new update. After the first version is scaled to zero and can be removed, the team uploads and deploys the next version.

Blue-Green Deployment: This strategy involves maintaining two identical production environments, with one serving active user traffic while the other remains inactive. New application versions are deployed to the inactive environment, allowing for testing and validation before switching the active environment, minimizing downtime and risk.

Canary Deployment: Canary deployment gradually rolls out new application versions to a subset of users or servers, allowing for real-time monitoring and validation of performance and stability. If issues arise, deployment can be halted or rolled back before affecting the entire user base.

Rolling Deployment: In a rolling deployment, new application versions are gradually deployed across the production environment, with each server or instance updated sequentially. This approach minimizes downtime and ensures a smooth transition between

10.2 Web Hosting Services

Amazon Web Services (AWS): AWS offers a comprehensive suite of cloud computing services, including compute, storage, database, and networking services. With features such as Elastic Beanstalk and EC2, AWS provides scalable and reliable hosting solutions for web applications of any size.

Microsoft Azure: Azure provides a range of cloud services, including virtual machines, app services, and serverless computing options. With features like Azure App Service and Azure Kubernetes Service, Azure offers flexible and cost-effective hosting solutions for web applications.

Google Cloud Platform (GCP): GCP offers a wide array of cloud services, including compute, storage, and networking services. With offerings such as Google App Engine and Google Kubernetes Engine, GCP provides scalable and secure hosting solutions for web applications.

11 Domain Management

Domain Registration: Registering a domain name is the first step in establishing an online presence for a web application. Domain registrars such as GoDaddy, Namecheap, and Google Domains offer domain registration services, allowing users to choose and register a unique domain name for their application.

DNS Configuration: Domain Name System (DNS) configuration involves mapping domain names to IP addresses, allowing users to access web applications using human-readable domain names. DNS providers such as Amazon Route 53 and Cloudflare offer DNS management services, allowing users to configure DNS settings and manage domain records.

12 Deployment Automation

Automation Tools: Deployment automation tools such as Jenkins, CircleCI, and GitHub Actions streamline the deployment process by automating build, test, and deployment tasks. By integrating with version control systems and continuous integration/continuous deployment (CI/CD) pipelines, these tools enable developers to automate repetitive tasks and ensure consistent deployment processes.

13 Security Considerations

Secure Deployment: Deploying web applications securely involves implementing best practices such as using HTTPS encryption, securing server configurations, and regularly updating software dependencies to mitigate security vulnerabilities. Additionally, monitoring and logging tools such as AWS CloudTrail and Azure Monitor help detect and respond to security incidents in real time.

14 Maintenance and Support [13]

Web Applications are dynamic and complex software systems that require regular maintenance and support to ensure their optimal performance, security, and usability. However, maintaining and supporting web applications can be difficult and expensive, especially as the web application grows and evolves over time.

14.1 Routine Maintenance Tasks

✓ Software Updates: Regularly update software dependencies, libraries, and frameworks to patch security vulnerabilities, improve performance, and add new features.

✓ Database Backups: Implement regular backups of the application's database to prevent data loss in case of hardware failure, human error, or security breaches.

✓ Performance Monitoring: Monitor application performance metrics such as response time, server load, and error rates to identify and address performance issues proactively.

✓ Security Patching: Apply security patches and updates to the underlying operating system, web server, and application components to protect against security threats and vulnerabilities.

15 Bug Tracking and Issue Resolution

✓ Bug Tracking: Use bug tracking systems such as Jira, Bugzilla, or GitHub Issues to log and prioritize bugs reported by users or discovered during testing.

✓ Issue Resolution: Assign and track bug fixes, enhancements, and feature requests through the bug tracking system, ensuring timely resolution and communication with stakeholders.

16 Iterative Improvements

✓ User Feedback Analysis: Gather and analyze user feedback through surveys, user interviews, and analytics tools to identify areas for improvement and prioritize feature enhancements.

✓ A/B Testing: Conduct A/B tests to compare different versions of the application and measure the impact of changes on user engagement, conversion rates, and other key metrics.

✓ Feature Prioritization: Prioritize feature development based on user feedback, business goals, and resource constraints to deliver maximum value to users and stakeholders.

17 Conclusion

To sum up, web applications are crucial to modern digital interactions since they offer a wide range of functions and services across several sectors. They need to be flexible enough to adapt to shifting user needs and technology developments in order to stay in business.

Applying best practices to web application design, development, deployment, and maintenance can help organizations ensure the dependability, security, and efficiency of their applications. This consistent commitment to excellence is essential in the digital age to meet people's diverse requirements and promote innovation.

CHAPTER 3

REQUIREMENTS SPECIFICATION

Chapter 3: Requirements Specification

1 Introduction

In this chapter, we will present our work following the UP (Unified Process) methodology. We will specify the various functional and non-functional requirements necessary to embark on the implementation phase. Subsequently, we will identify the actors who use the system, followed by a use case diagram describing the functionalities of our future application.

2 Requirements specification [14]

Use case modeling is one of the specification documentation strategies that facilitates the developer to specify the intended user functionalities. The purpose of use cases is to delineate the requirements such that people without high-level training (e.g., the stakeholders, users) can understand and review them. It usually consists of two parts – use case diagram and the use case textual descriptions. These requirements are divided into two types: functional and non-functional.

2.1 Functional Requirements

The functional requirements, as their name suggests, describe the functionalities of the system to be implemented clearly and precisely. Our application must cover the functional requirements described below:

✓ Authentication, where the system must be able to retrieve the information of each user based on their username and password to grant them access to the application.

✓ The management and monitoring of trainees by an administrator, who is the head of department.

✓ Consultation of lists of users including the information of all interns and their supervisors.

✓ Message functionality enhances communication, collaboration, and productivity between trainees and supervisors, ultimately contributing to the success of internship programs and the development of trainees' skills and knowledge.

✓ Submit task provides interns/trainees with a structured and efficient way to submit their completed tasks or assignments to their supervisors.

✓ Validate task allows supervisors to officially approve or validate the completion of tasks submitted by interns/trainees.

✓ Notification feature serves to alert users about important events, updates, or actions within the system.

2.2 Non Functional Requirements

The non-functional requirements represent the implicit requirements that the system must meet. Among these requirements, we mention:

Security: The system must provide data security and guard against intrusions or unauthorized access.

Reliability: To reduce the chance of malfunctions or breakdowns, the system should function consistently and dependably both at normal and high loads.

Availability: There should be little downtime for updates or maintenance, and the system should be available 24/7.

Usability: The system should be user-friendly, intuitive, and easy to navigate, ensuring a positive user experience.

Ergonomics: The application must meet the following ergonomic criteria: readability, guidance, error correction capabilities, and adaptation to the user's working conditions.

3 Identification of Actors

An actor represents an external entity that interacts with a system. For example, an actor can be a (human user, hardware device or other systems).[15]

In our application, we have three actors:

The application administrator: Manages users, meaning the administrator can create, modify, search, or delete a user account. They can activate or deactivate a user account and access the information of all trainees who completed their internship in the company. The administrator can view and consults the evaluations of all trainees in their assigned tasks and see the names of their supervisor.

The supervisor: Monitors the progress of interns and apprentices. They provide guidance through messaging with trainees, create tasks, delete, search and modify assigned tasks, and ensure compliance with policies. The supervisor also evaluates tasks and documents progress and performance. They can also change their account password.

The interns and apprentices: They perform similar roles within the system and are classified under a single actor, referred to as a trainee. They engage and submit assigned tasks or projects, develop skills under supervision, search assigned tasks and comply with company policies. Trainees communicate regularly with their supervisor through messaging for guidance and feedback, and they can consult their evaluations and view performance. Just like supervisors, they can also change their account password.

4 Identification of use cases

Use cases describe the functional requirements of a system from the end user's perspective, creating a goal-focused sequence of events that is easy for users and developers to follow. The main objective is that the set of use cases must comprehensively describe the functional requirements of the system.[16]

In the system to be developed, we have identified the following use cases:

<table>
<tr><td colspan="3">Use Case</td><td>Actor</td></tr>
<tr><td colspan="3">Authentication

Consult Evaluations</td><td>The Administrator of the application
The Supervisor
The Trainees</td></tr>
<tr><td colspan="2" rowspan="3">Manage User Account</td><td>Create</td><td rowspan="6">The administrator of the application</td></tr>
<tr><td>Delete</td></tr>
<tr><td>Modify</td></tr>
<tr><td colspan="2" rowspan="3">List of users

Consult trainee history</td><td>Activate</td></tr>
<tr><td>Deactivate</td></tr>
<tr><td>Search user</td></tr>
<tr><td colspan="2">Send Message
Receive Message
Change Password
Search Task</td><td>Sending a Message triggers a Notification</td><td>The Supervisor

The Trainees</td></tr>
<tr><td rowspan="4">Manage Task</td><td rowspan="4">Creating a Task triggers a Notification</td><td>Create Task</td><td rowspan="4">The Supervisor</td></tr>
<tr><td>Delete Task</td></tr>
<tr><td>Modify Task</td></tr>
<tr><td>Evaluate Task</td></tr>
<tr><td colspan="3">Submit Task</td><td>The Trainees</td></tr>
</table>

Table 1. 1-List of System Use Cases

4.1 System Use Case Diagram

It's a formalism that allows modeling the operation of a system by breaking it down into functionalities. [17]

This diagram illustrates the different functional behaviors of our system.

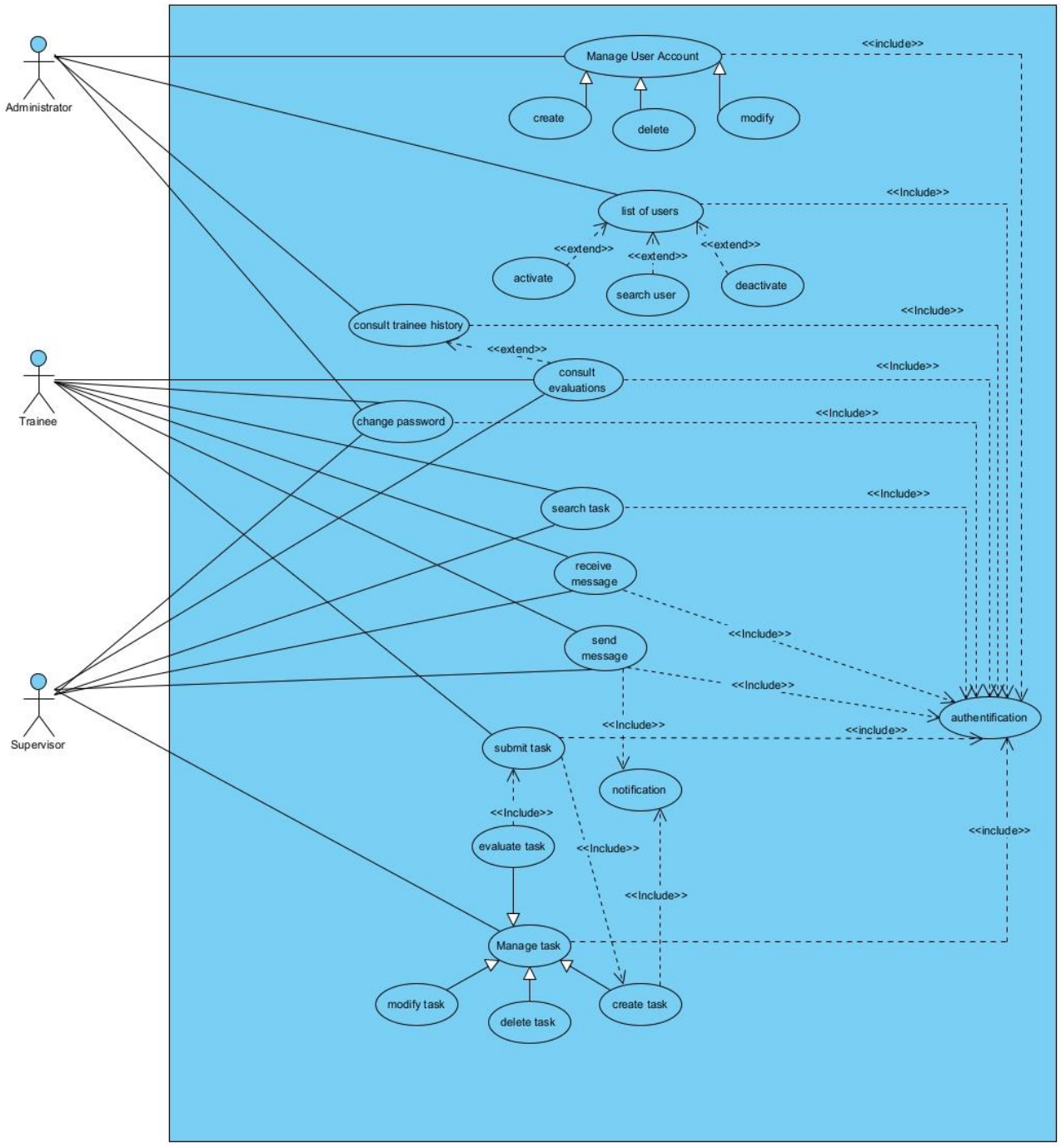

Figure 1. 5: System Use Case Diagram

5 Textual Description of Use Cases

The textual description is flexible. However, this description often takes a written form that is more suitable for communication with users. Below, we present a detailed description of each use case. This involves defining the possible scenarios that occur when each use case is triggered.

5.1 Use Case Authentication

Name	Authentication
Actor	All actors
Description	Allows the user to authenticate in order to access the application's functionalities
Nominal Scenario	-The user accesses the application. -The system displays the home interface. -When a user clicks the login button the system displays the authentication interface. -The user enters their login and password. -The system verifies the correctness of the information. -The system displays the corresponding interface.
Alternative Scenario	If the provided credentials are incorrect: -The system displays an error message. -The user has the option to retry by entering their credentials again

Table 1. 2-Textual Description Use Case Authentication

5.2 Use Case Manage User Account

Name	Manage User Account
Actor	Application Administrator
Description	This use case allows the administrator to manage user accounts (create, modify and delete)
Precondition	Authentication
Nominal Scenario	-The application administrator requests the form to (create or modify) a user account. -The system displays the requested form. -The administrator enters the data and submits it. -The system verifies the data and then displays a confirmation message.
Alternative Scenario	The system displays an error message if the data is incomplete

Table 1. 3- Textual Description Use Case Manage User Account

5.3 Use Case Change Password

Name	Change Password
Actor	Supervisor, Trainee
Description	-This use case allows the supervisors and the trainees to change their account passwords.
Precondition	-Authentication
Nominal Scenario	-The supervisor or trainee requests the form to change their account password. -They enter their new password twice in the "New Password" and "Confirm Password" fields respectively and submit. -The system verifies the data and then displays a confirmation message.
Alternative Scenario	-The system displays an error message if the data doesn’t match

Table 1. 4- Textual Description Use Case Change Password

5.4 Use Case List of users

Name	list of users
Actor	Application Administrator
Description	This use case allows the application administrator to view a table listing all users grouped into the categories of supervisor, intern, and apprentice.
Precondition	Authentication
Nominal Scenario	-The admin accesses the list of users interface, categorized as supervisor, intern, or apprentice. -The system displays the table of users in each category. -The admin enters the entry in the search field. -The system displays the search results. -They can also activate or deactivate a user account by clicking the active or disabled button.

Table 1. 5- Textual Description Use Case List of users

5.5 Use Case Consult Evaluations

Name	Consult Evaluations
Actor	All actors
Description	The administrator consults the evaluations of all trainees, the supervisor consults the evaluations of the trainees assigned to them, and trainees consult their individual evaluations.
Precondition	Authentication
Nominal Scenario	The user clicks the "Evaluations" tab, and the system will display a table with evaluations

Table 1. 6- Textual Description Use Case Consult Evaluations

5.6 Use Case Manage Task

Name	Manage Task
Actor	Supervisor
Description	This use case allows the supervisor to manage tasks assigned to trainees, including creating, modifying, deleting, and evaluating tasks.
Precondition	Authentication
Nominal Scenario	-The supervisor requests the form to (create, modify) a task. -The system displays the requested form. -The supervisor enters the data and submits it. -The system verifies the data and then displays a confirmation message. - After the trainees have submitted the task, the supervisor evaluates their performance and assigns marks accordingly
Alternative Scenario	The system displays an error message if the data is incomplete

Table 1. 7- Textual Description Use Case Manage Task

5.7 Use Case Submit Task

Name	Submit Task
Actor	Trainee
Description	-This use case allows the trainees to submit their tasks or assignments so that they can be evaluated by the supervisor.
Precondition	Authentication
Nominal Scenario	-The trainee submits the task using the form. -The system verifies the data and then displays a confirmation message
Alternative Scenario	-The system displays an error message if the data is incomplete

Table 1. 8- Textual Description Use Case Submit Task

5.8 Use Case Send Message

Name	Send Message
Actor	Supervisor, Trainee
Description	- This use case allows the trainees and supervisors to send messages to each other.
Precondition	Authentication
Nominal Scenario	- The trainee or supervisor types a text in the input field. -The trainee or supervisor clicks the send button to send the message.

Table 1. 9- Textual Description Use Case Send Message

5.9 Use Case Receive Message

Name	Receive Message
Actor	Supervisor, Trainee
Description	- This use case allows the trainees and supervisors to receive messages.
Precondition	Authentication
Nominal Scenario	- The trainee or supervisor receives a message sent by the other party

Table 1. 10- Textual Description Use Case Receive Message

5.10 Use Case Search Task

Name	Search Task
Actor	Supervisor, Trainee
Description	- This use case allows the trainees and supervisors to search for tasks. -The supervisor searches for the tasks they created or submitted by trainees. -The trainees search for the list of tasks assigned to them.
Precondition	Authentication
Nominal Scenario	-They input the search query into the search field. -The system verifies the data and then displays the corresponding search results.
Alternative Scenario	If the search query doesn't match any information regarding the tasks, the system will respond with no results

Table 2. 1- Textual Description Use Case Search Task

5.11 Use Case Notification

Name	Notification
Actor	Supervisor, Trainee
Description	- This use case allows the trainees and supervisors to receive notifications.
Precondition	Authentication
Nominal Scenario	-This action is triggered when the supervisor creates a task or when a trainee or supervisor sends a message to the other party.

Table 2. 2- Textual Description Use Case Notification

5.12 Use Case Consult Trainee History

Name	Consult Trainee History
Actor	Application Administrator
Description	This use case allows the application administrator to view a table listing all trainee who did their internship in the company.
Precondition	Authentication
Nominal Scenario	-The admin accesses the history interface. -The system displays the table of all trainees who did their internship in the company starting with the recent trainees -The admin enters a query in the search field to search for an individual trainee or they can search a list of trainees by end or start year of the internship. -The system displays the search results.

Table 2. 3- Textual Description Use Case Consult trainee history

6 Conclusion

In this third chapter, we focused on expressing the requirements of the system. This allowed us to gain a better understanding of it in order to define its main functionalities accurately. In the next chapter, we will start the design phase of our application.

CHAPTER 4

ANALYSIS AND DESIGN

Chapter 4: Analysis and Design

1 Introduction

In this chapter, we will address the phases of analysis and design. The models presented in the requirements specification will be translated into a language close to that of computer scientists. The analysis models will describe what the system must do, interpreted in sequence diagrams that clearly depict the structuring and architecture of our application. The design models, on the other hand, will describe the system's structure, meaning the system's components, in a class diagram.

2 Presentation of sequence diagrams

A sequence diagram describes interactions between objects. Usually, the diagram shows a single use case or scenario. Sequence diagrams are a type of interaction diagram and are not as good for showing object implementation details. A sequence diagram focuses on time sequencing or time ordering of messages or the order in which messages are sent. The emphasis in these diagrams is what happens first, second, and so on. They represent the passage of time graphically. Sequence diagrams have two axes: the horizontal axis displays the objects, and the vertical axis shows the progression of time. These diagrams illustrate the interactions between objects in the order that they occur. We identify operations from the use case description and represent them in sequence diagram. Since a sequence diagram can be developed for each primary use case, included use case, and extended use case, multiple sequence diagrams will be created. [18] [19]

3 Development of sequence diagrams for the system's use cases

3.1 Sequence diagram for the use case: « Authentication»

Authentication involves verifying the user's identity to access the application. To do this, the user requests to login, and the system displays the form page where they enter their username and password. If the information is correct, the user can access the corresponding interface, otherwise, the system will display an error message.

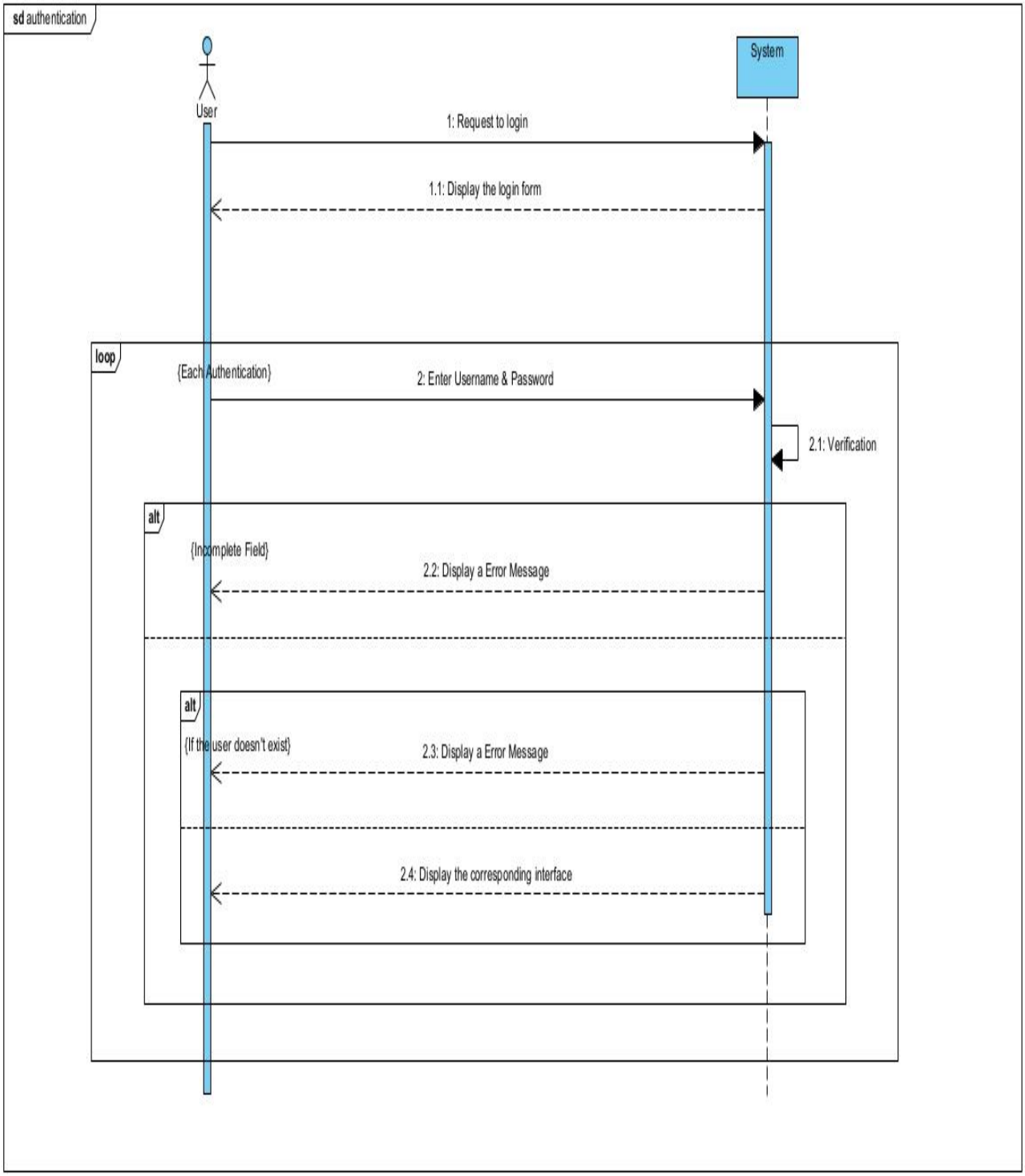

Figure 1. 6: Sequence Diagram « Authentication »

3.2 Sequence Diagram for the Use Case: «Create User »

The diagram below illustrates the use case « Create User »

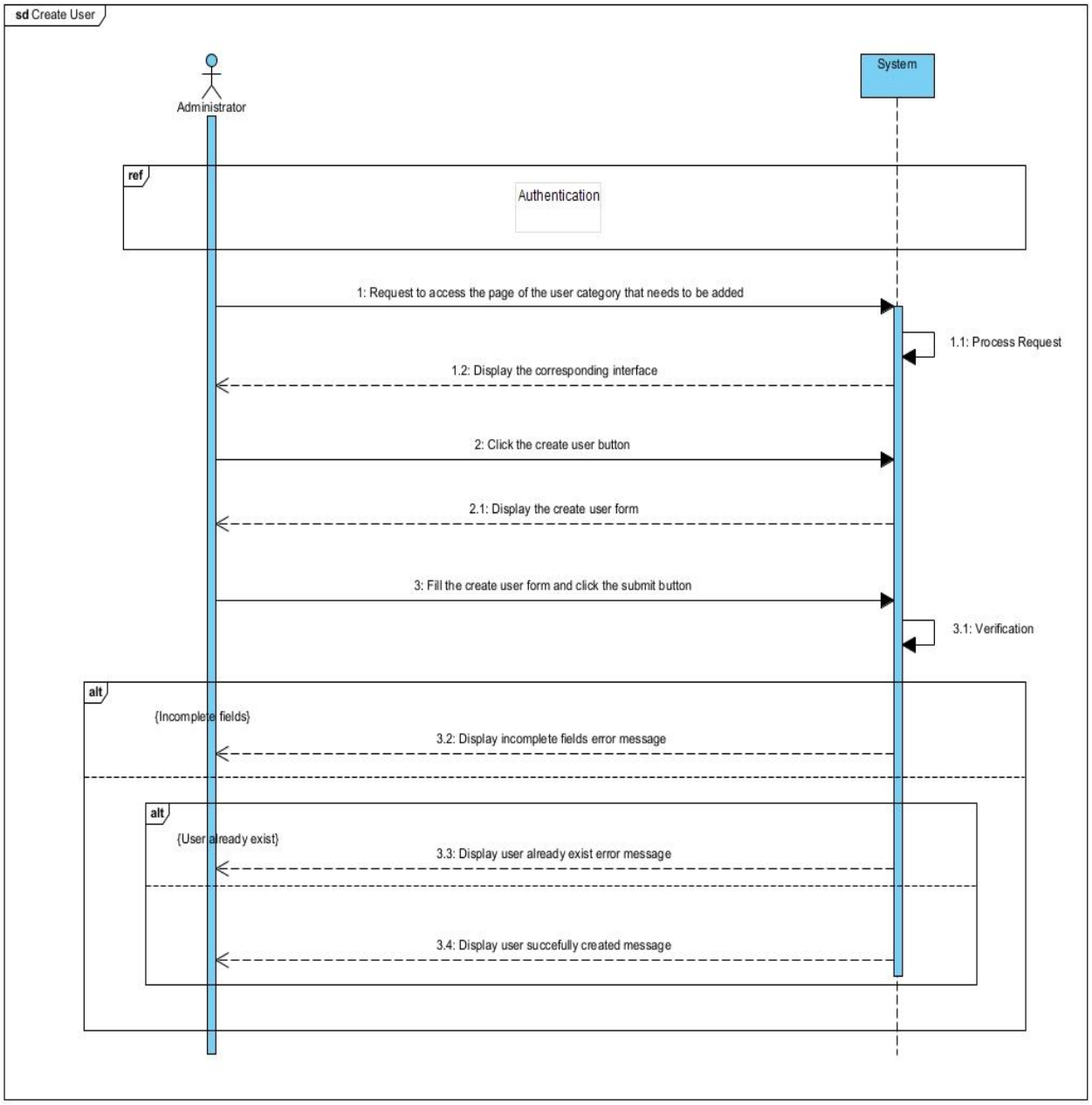

Figure 1. 7: Sequence Diagram « Create User »

3.3 Sequence Diagram for the Use Case: « Search User»

To search for a user (supervisor, intern, apprentice), the administrator must first authenticate. After authentication, they can access the list of users by category. Then, they can enter their search query (first name, surname) in the search bar. The system will immediately search for the corresponding user in the database and display the results.

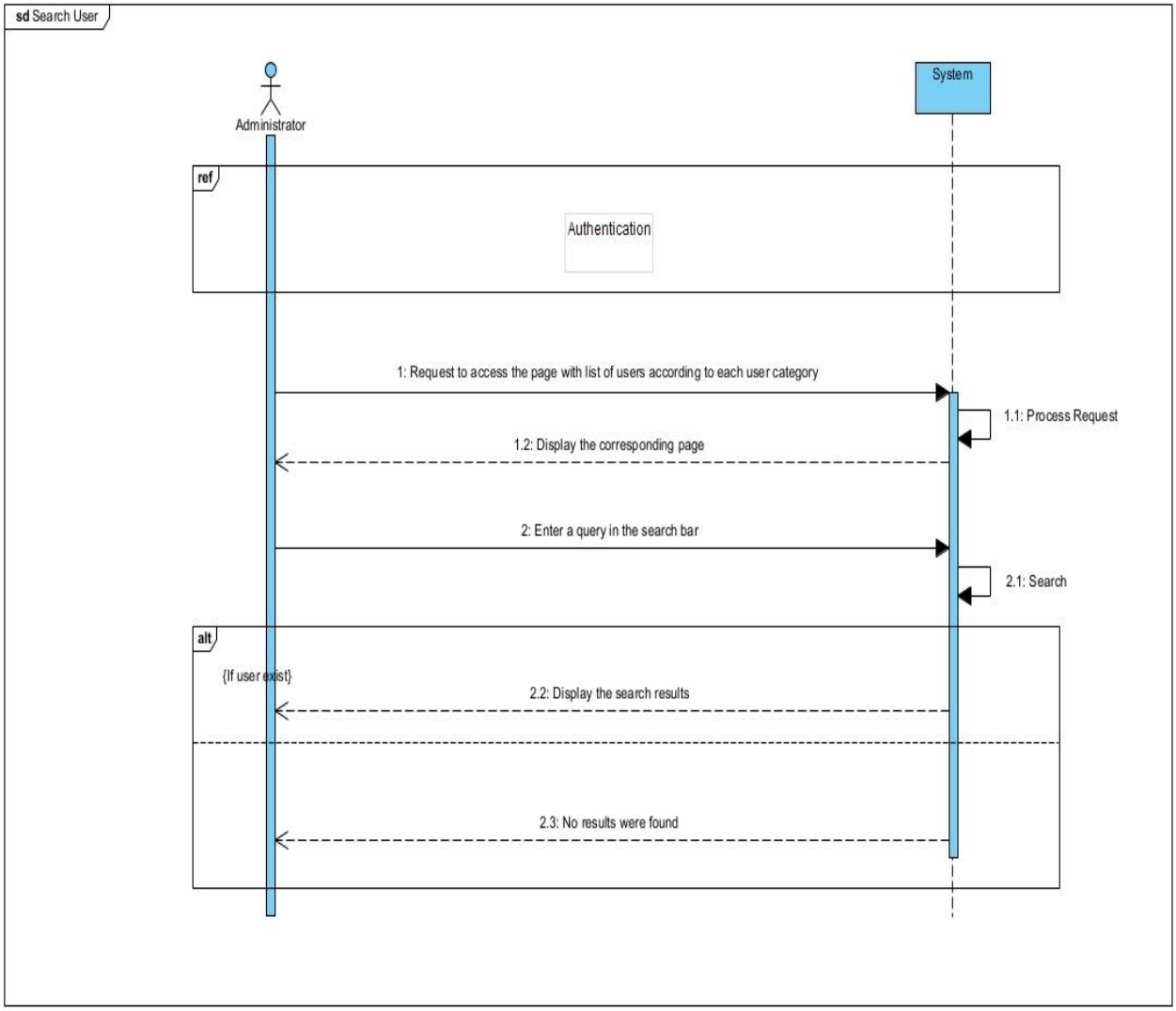

Figure 1. 8: Sequence Diagram « Search User »

3.4 Sequence Diagram for the Use Case: «Modify User»

The diagram below illustrates the use case « Modify User »

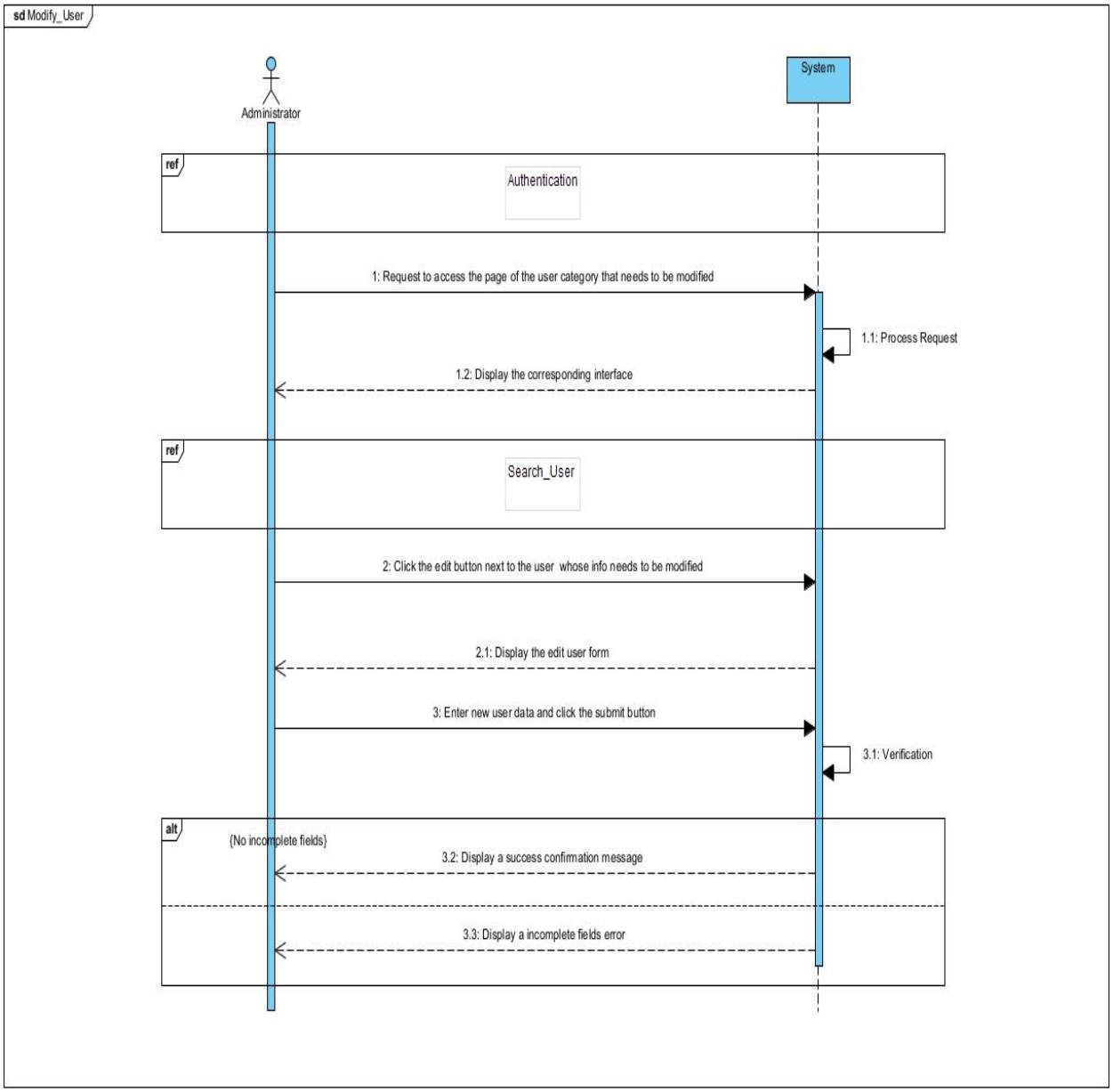

Figure 1. 9: Sequence Diagram « Modify User »

3.5 Sequence Diagram for the Use Case: «Delete User»

The diagram below illustrates the use case « Delete User »

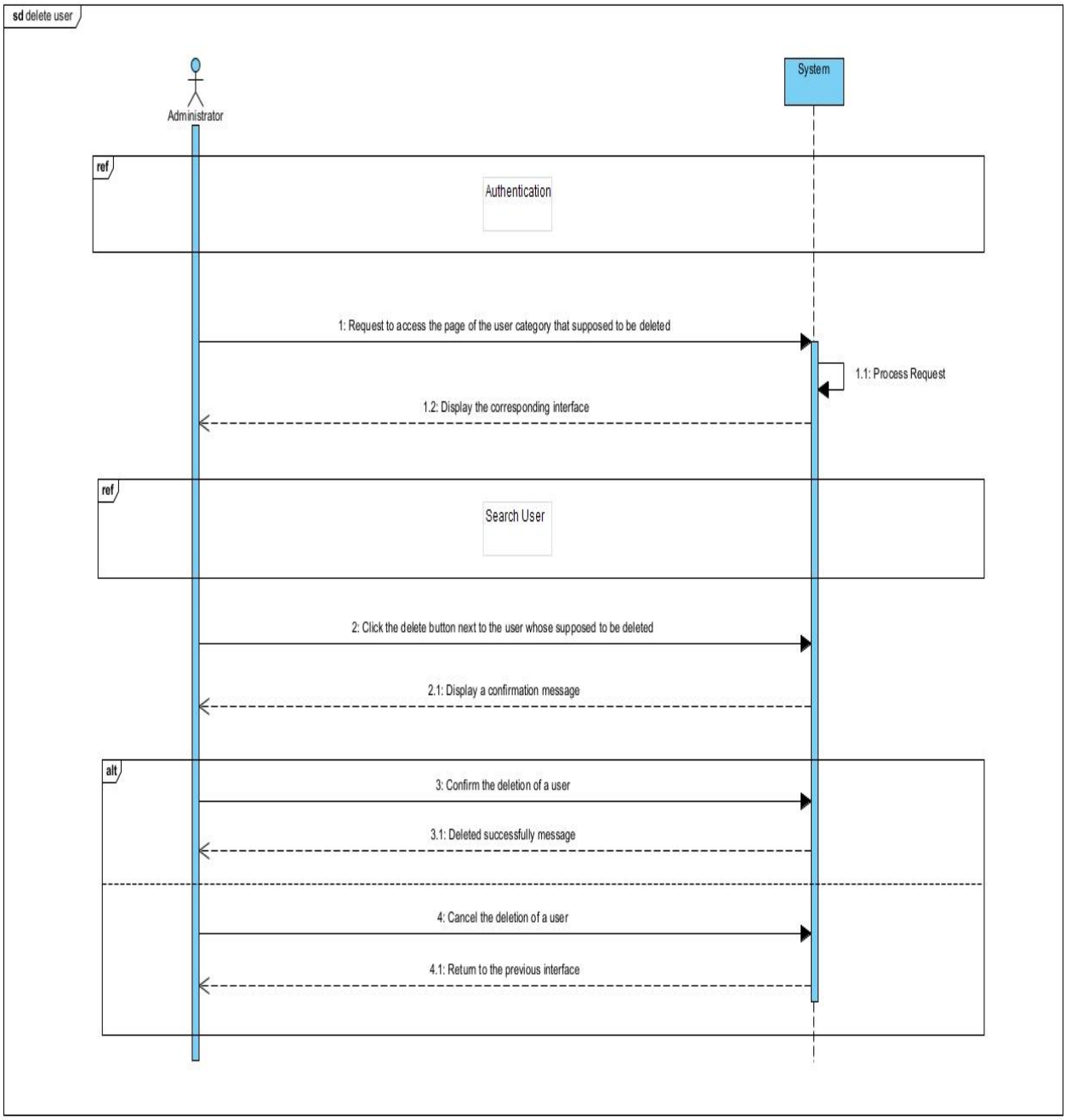

Figure 2. 1: Sequence Diagram « Delete User »

3.6 Sequence Diagram for the Use Cases: « Activate / Deactivate»

The diagram below illustrates the use cases of « Activating or Deactivating a User » by the administrator.

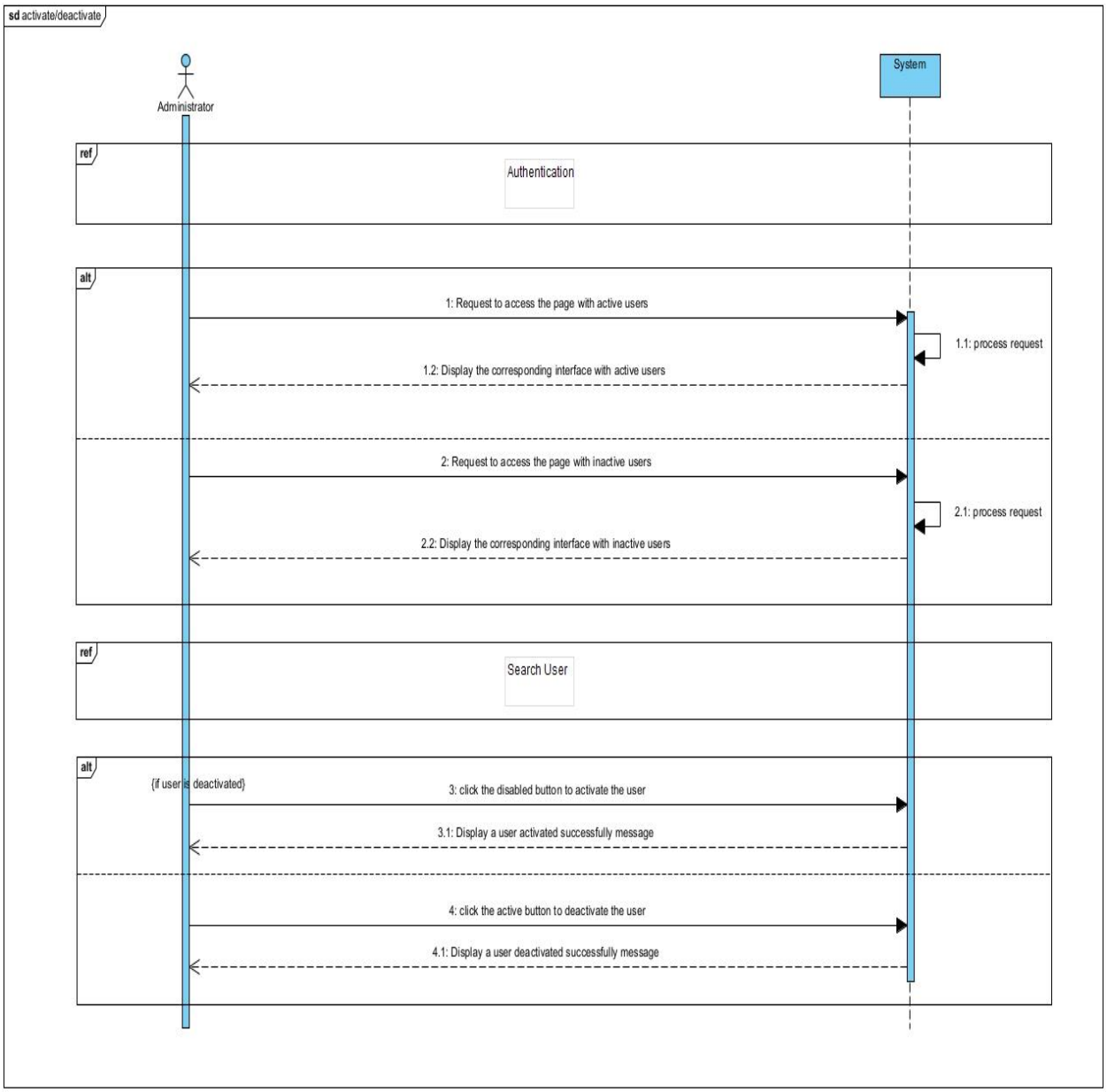

Figure 2. 2: Sequence Diagram « Activate / Deactivate »

3.7 Sequence Diagram for the Use Case: « Create Task»

The diagram below illustrates the use case « Create Task »

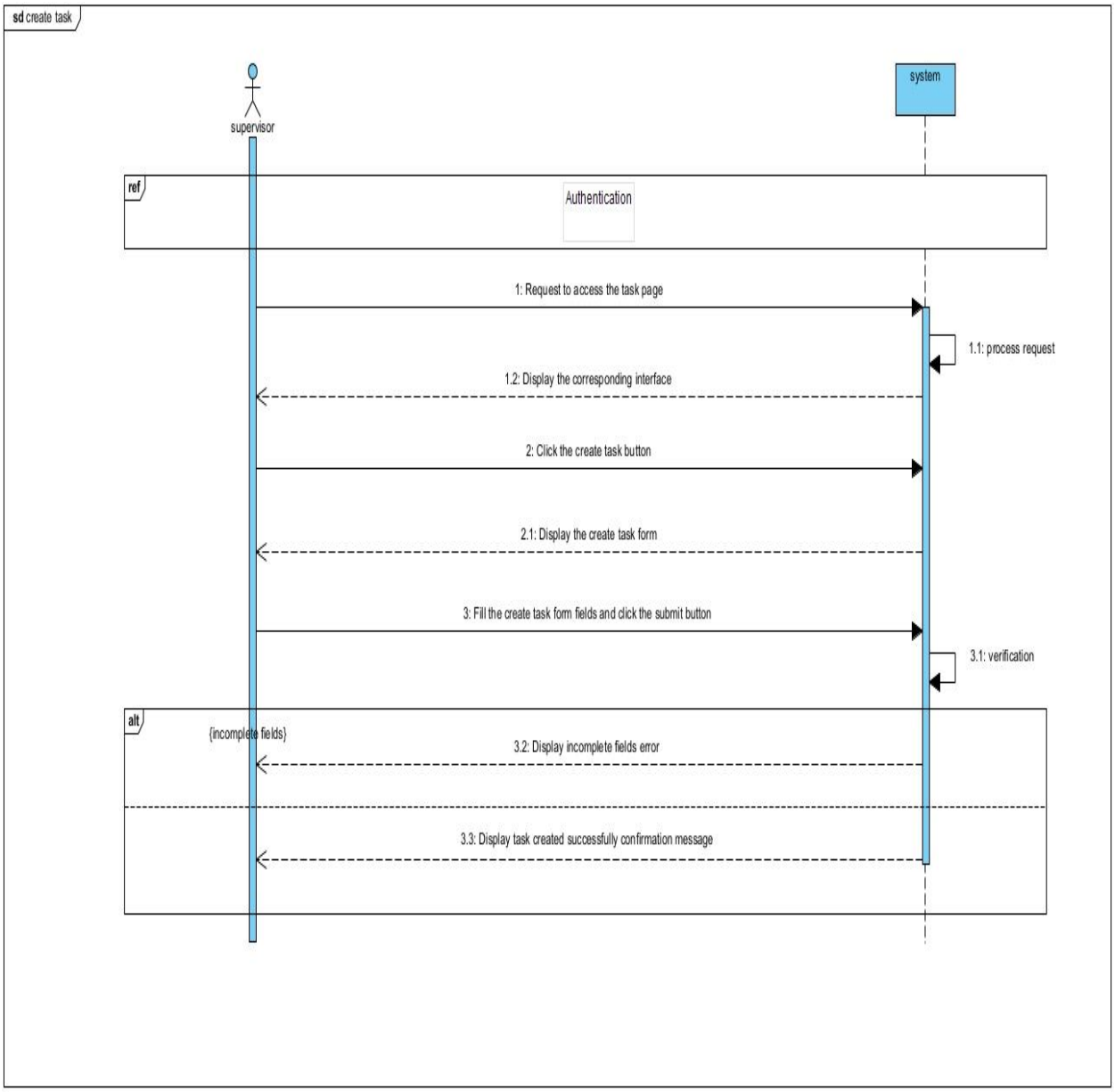

Figure 2. 3: Sequence Diagram « Create Task »

3.8 Sequence Diagram for the Use Case: « Search Task»

The diagram below illustrates the use case « Search Task »

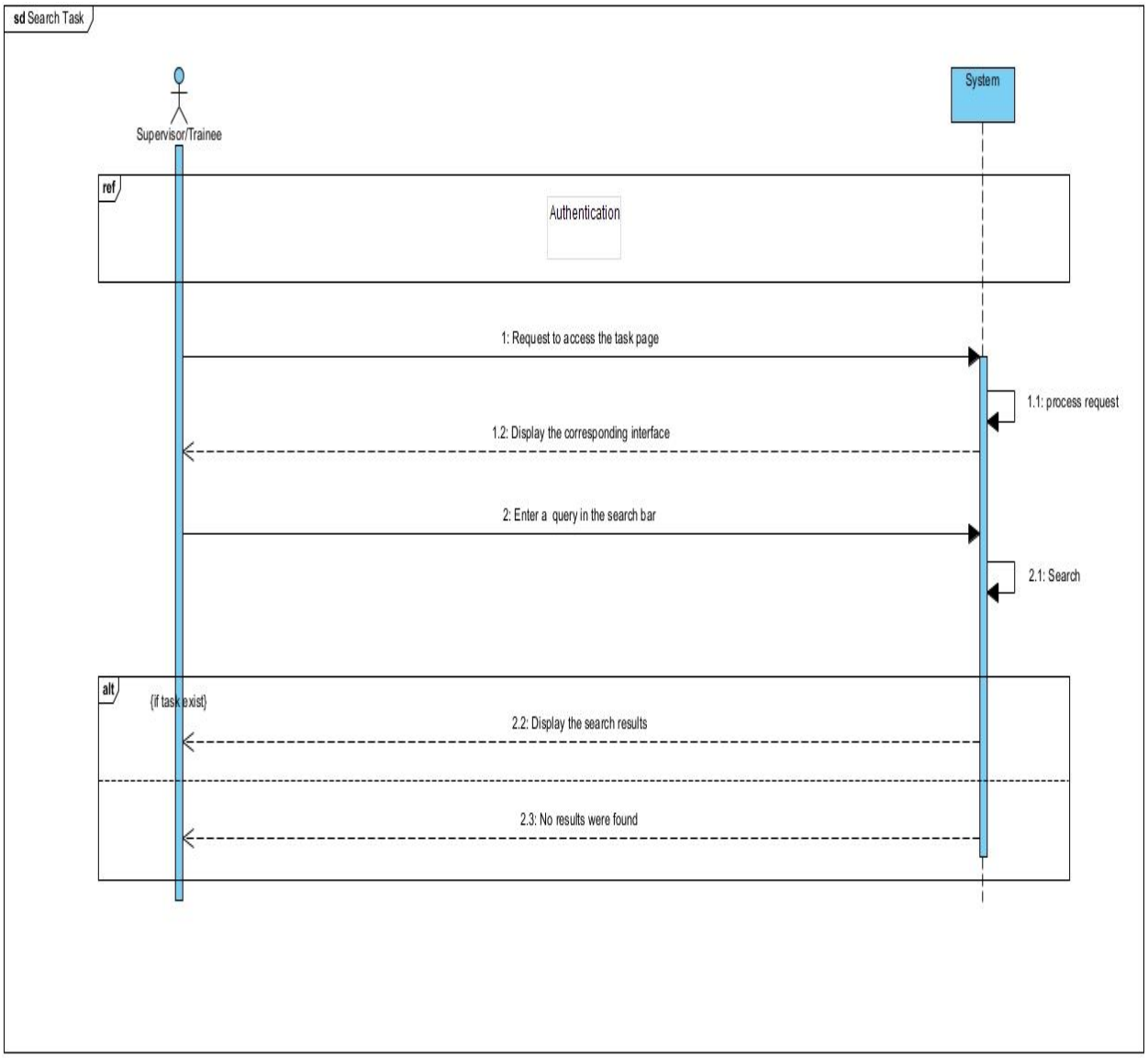

Figure 2. 4: Sequence Diagram « Search Task »

3.9 Sequence Diagram for the Use Case: « Delete Task »

The diagram below illustrates the use case « Delete Task »

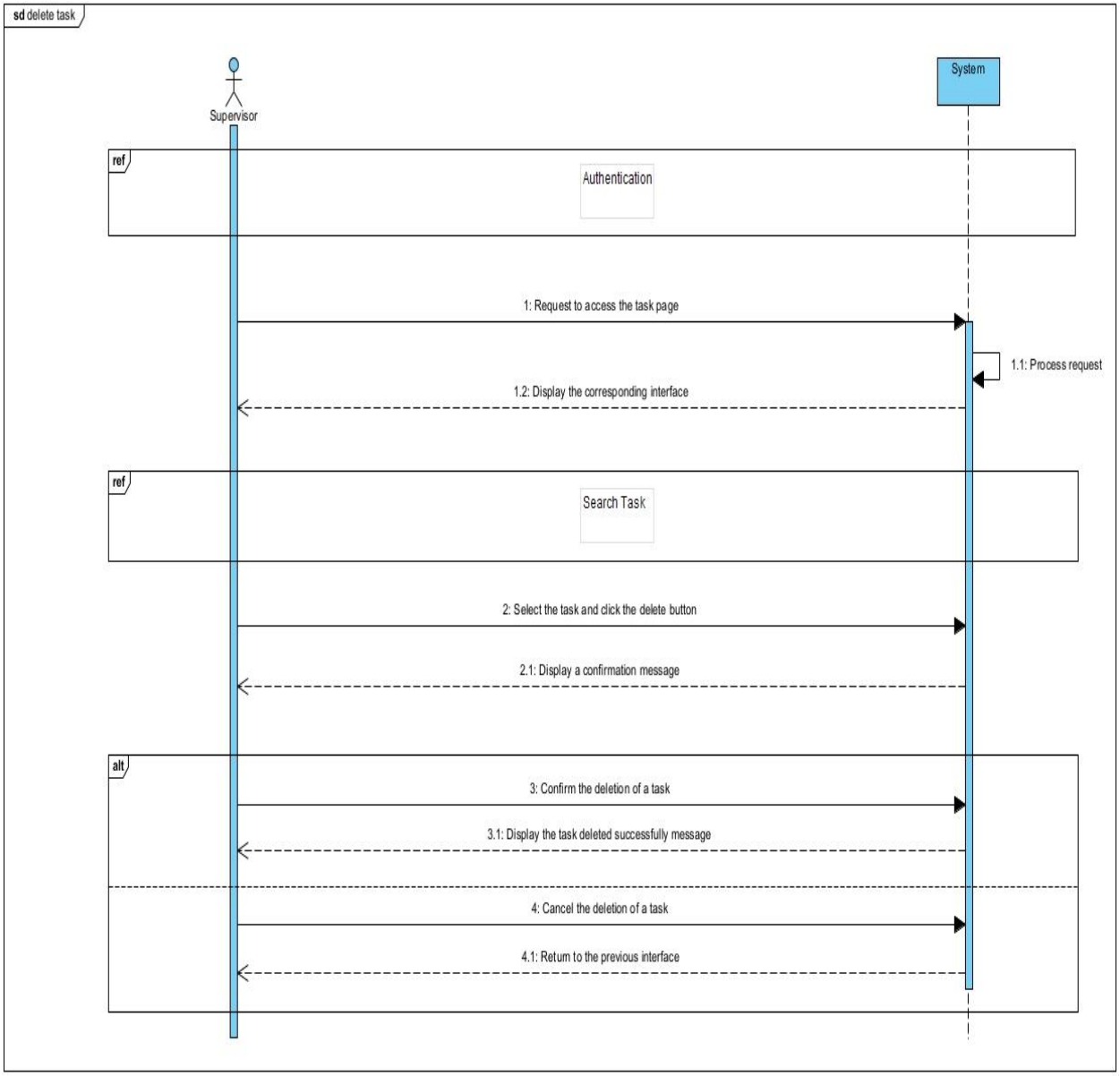

Figure 2. 5: Sequence Diagram « Delete Task »

3.10 Sequence Diagram for the Use Case: « Modify Task »

The diagram below illustrates the use case « Modify Task »

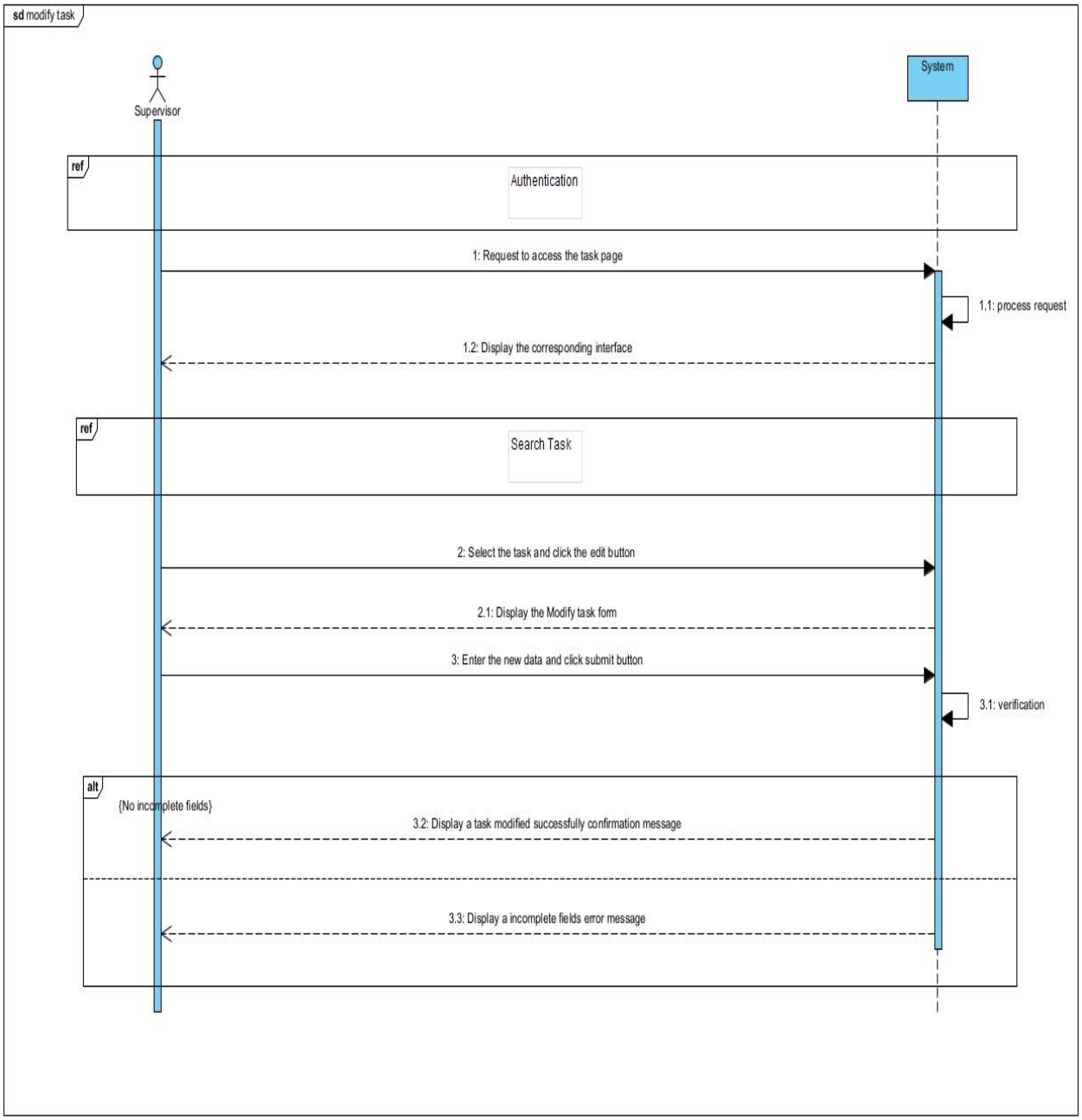

Figure 2. 6: Sequence Diagram « Modify Task »

3.11 Sequence Diagram for the Use Case: « Submit Task»

The diagram below illustrates the use case « Submit Task »

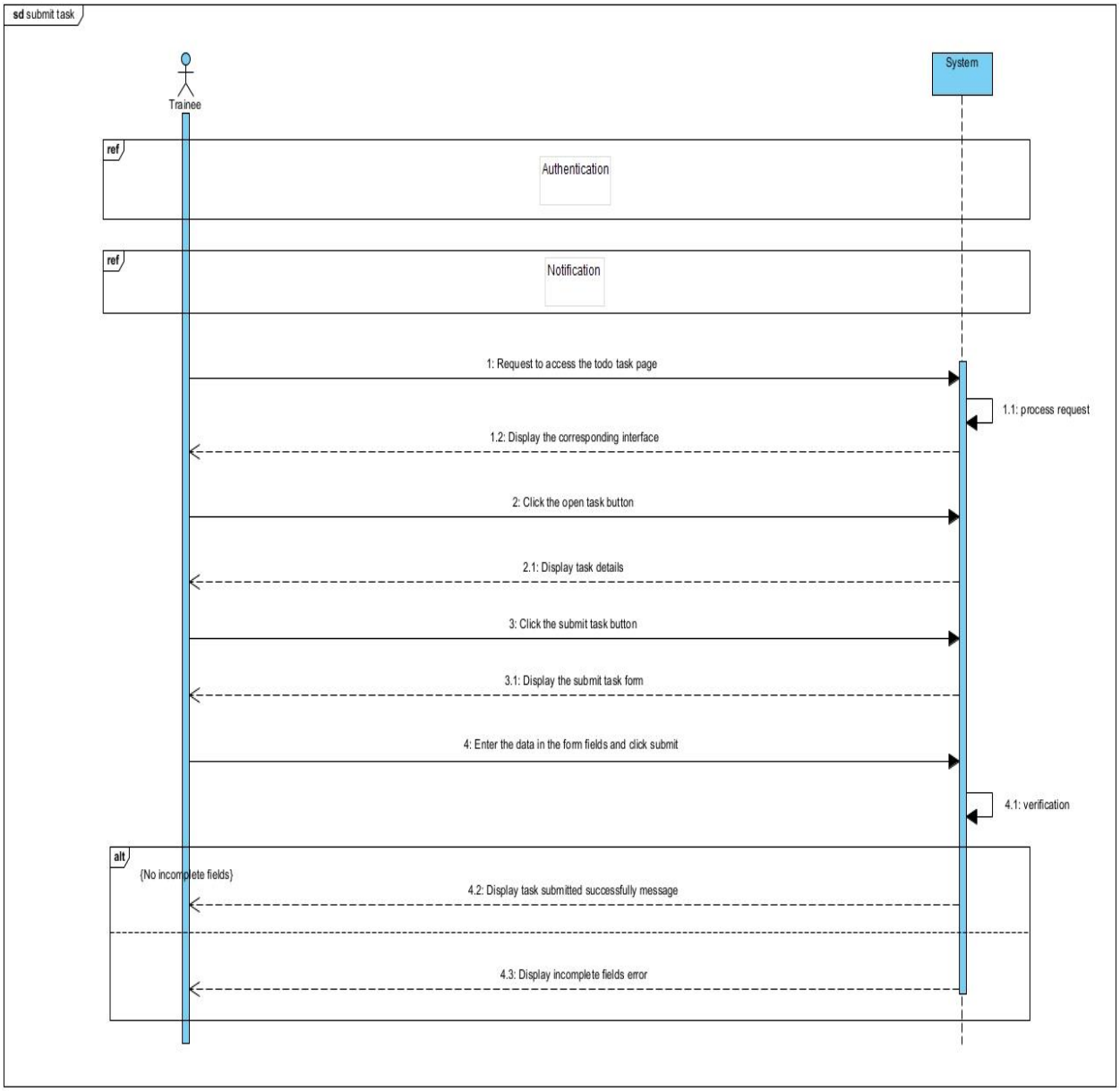

Figure 2. 7: Sequence Diagram « Submit Task »

3.12 Sequence Diagram for the Use Case: «Evaluate Task»

The diagram below illustrates the use case « Evaluate Task »

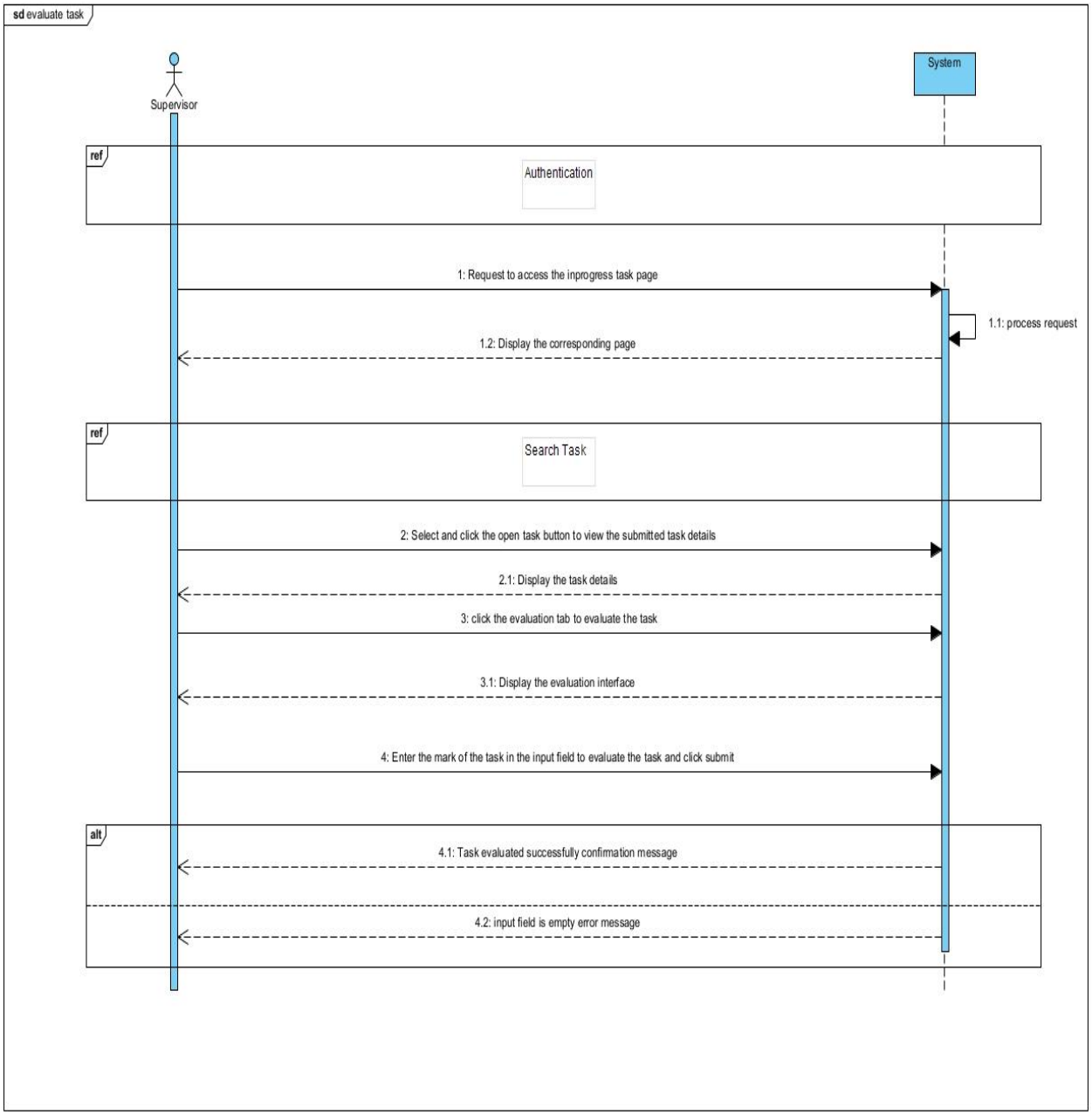

Figure 2. 8: Sequence Diagram « Evaluate Task »

3.13 Sequence Diagram for the Use Case: «Consult Trainee History»

The diagram below illustrates the use case « Consult Trainee History »

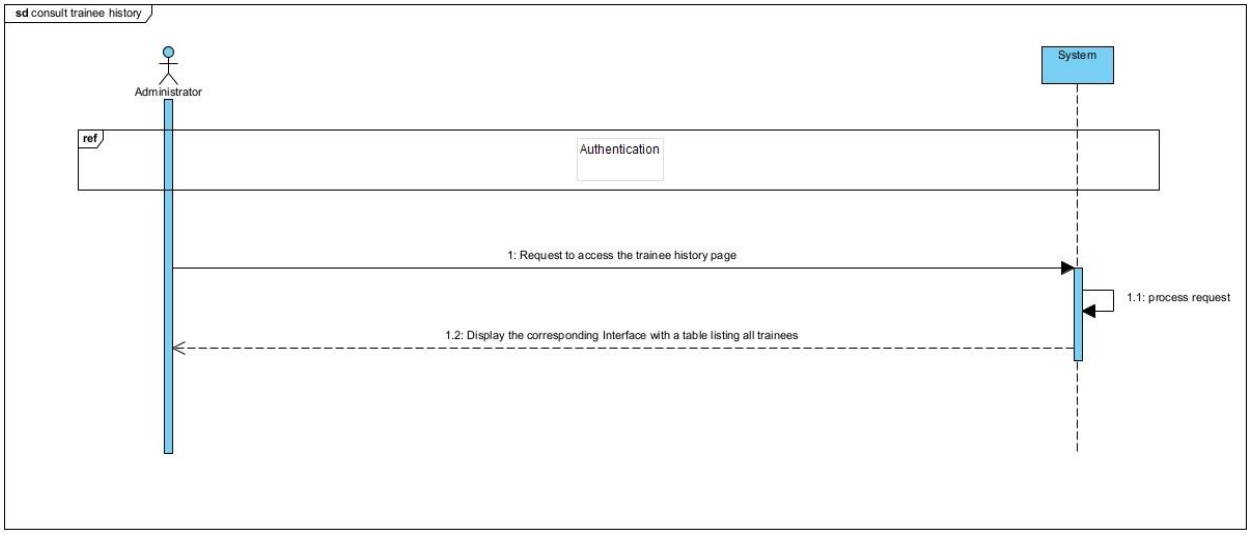

Figure 2. 9: Sequence Diagram « Consult Trainee History »

3.14 Sequence Diagram for the Use Case: «Change Password»

The diagram below illustrates the use case « Change Password »

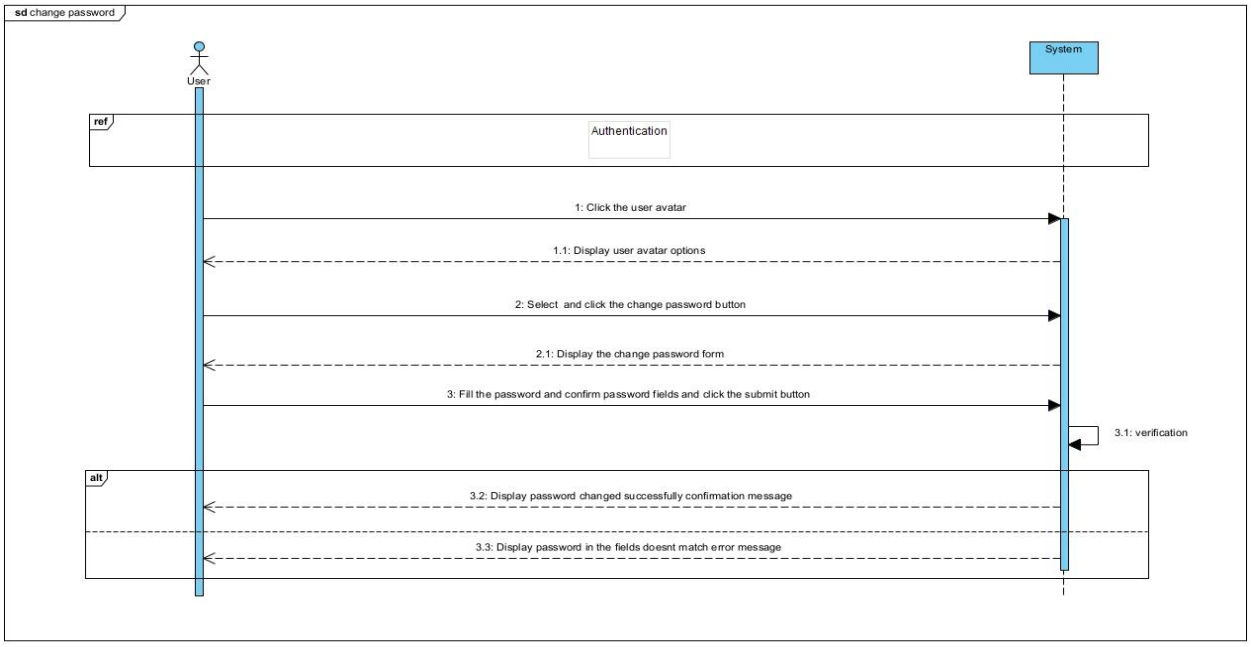

Figure 2. 10:Sequence Diagram « Change Password»

3.15 Sequence Diagram for the Use Case: «Consult Evaluations»

The diagram below illustrates the use case « Consult Evaluations »

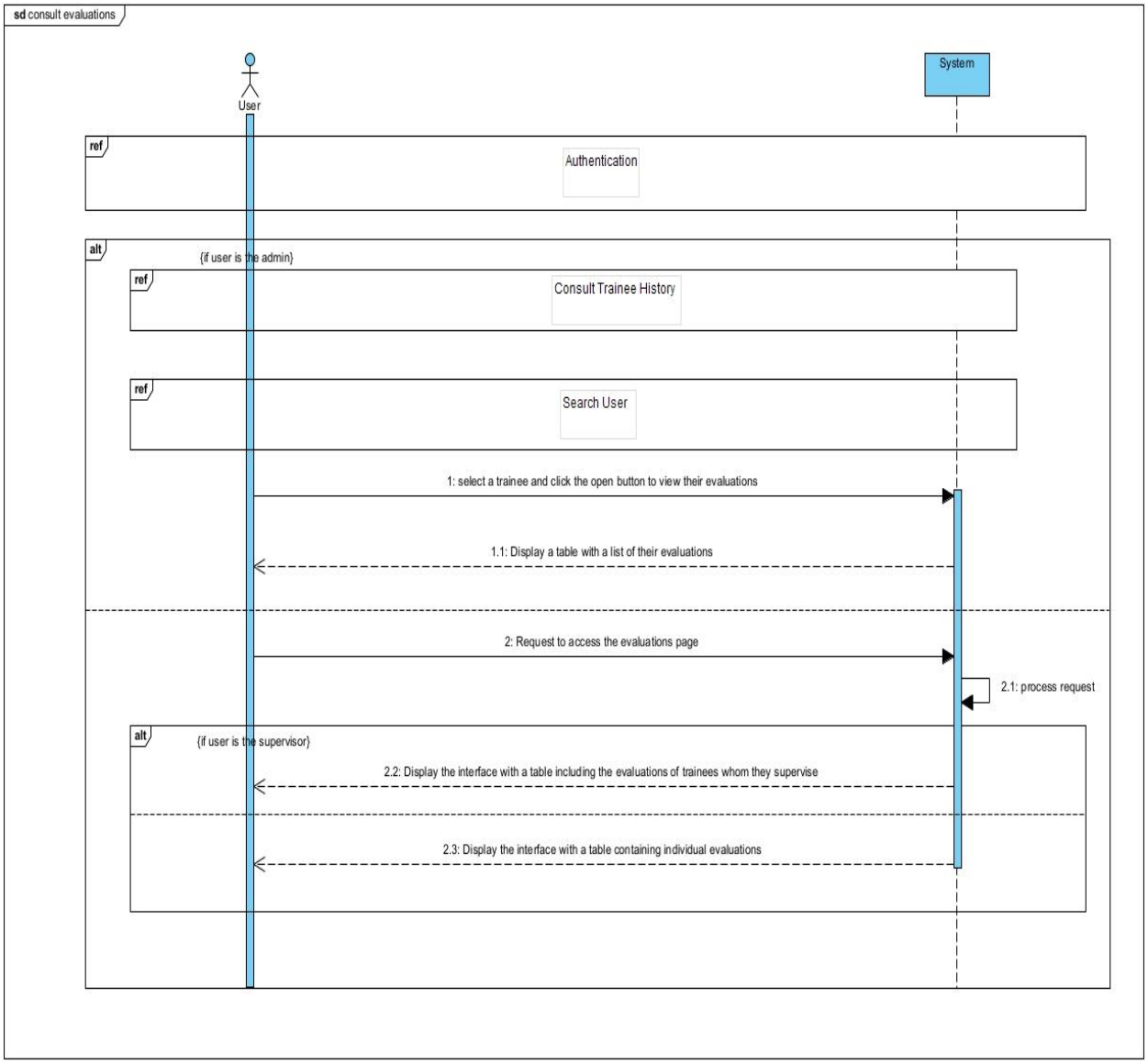

Figure 3. 1: Sequence Diagram « Consult Evaluations »

3.16 Sequence Diagram for the Use Case: « Send Message»

The diagram below illustrates the use case « Send Message»

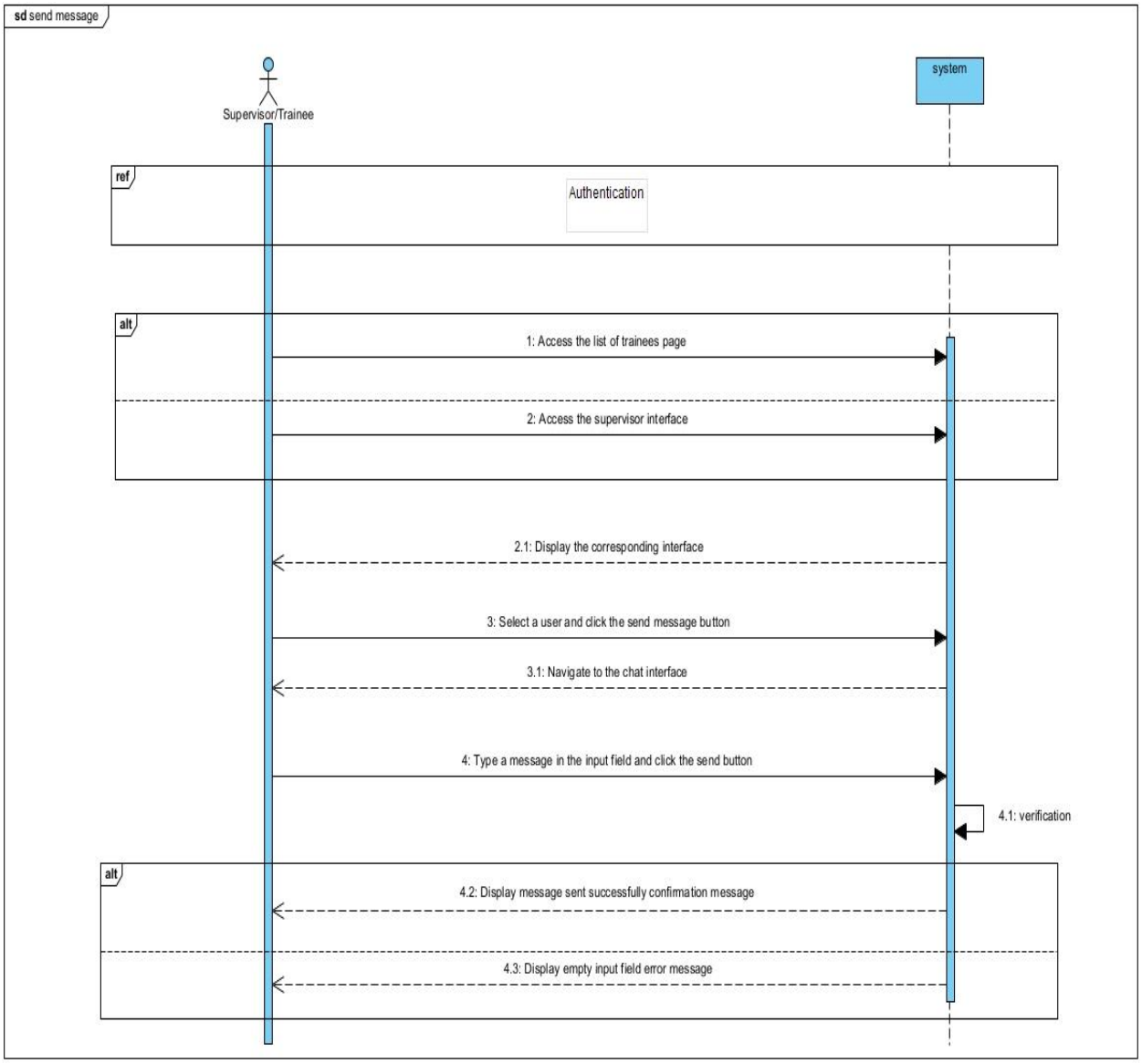

Figure 3. 2: Sequence Diagram « Send Message »

3.17 Sequence Diagram for the Use Case: «Notification»

The diagram below illustrates the use case « Notification»

Figure 3. 3: Sequence Diagram « Notification »

3.18 Sequence Diagram for the Use Case: «Receive Message»

The diagram below illustrates the use case « Receive Message»

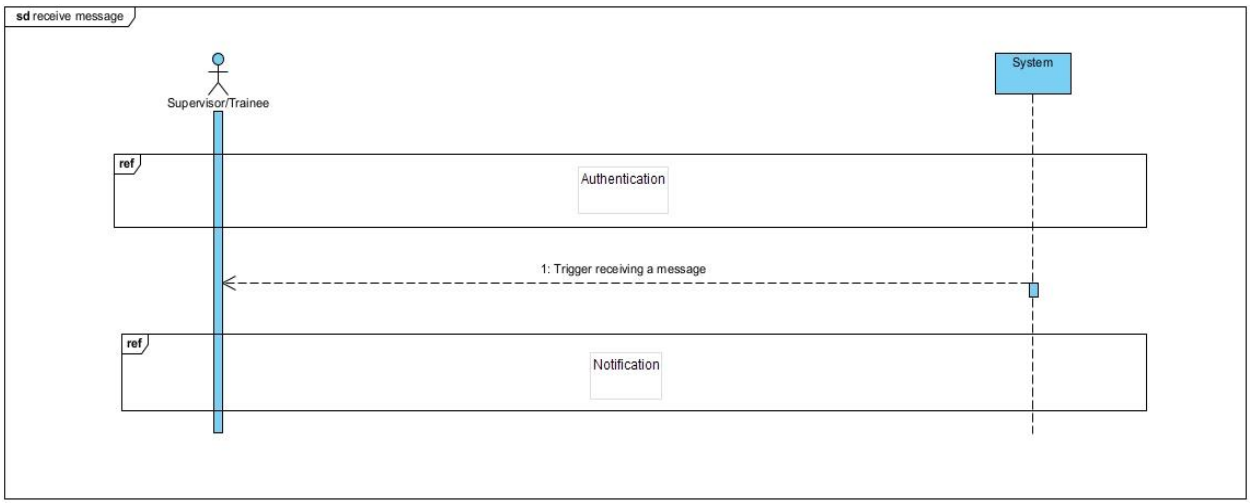

Figure 3. 4: Sequence Diagram « Receive Message»

4 Data dictionary

The data dictionary brings together all the data identified, specified and collected, and which we will have to keep in our database (and which will therefore appear in the class diagram). The data dictionary of our application is represented in the table below:

Class Name	Codification	Designation	Type
User	Id_User	User Identification	**ObjectId**
	Surname	Surname of the User	**String**
	FirstName	User FirstName	**String**
	D.O.B	User Date of birth	**Date**
	Start_Date	Trainee internship startdate	**Date**
	Department	User department	**String**
	Role	Role of the user	**String**
	End_Date	Trainee internship enddate	**Date**
	Phone_Number	User phone number	**String**
	Theme	Trainee internship theme	**String**
	Email	User email	**String**
	Password	User account password	**String**
	IsAdmin	Admin login variable	**Boolean**
	IsSupervisor	Supervisor login variable	**Boolean**
	Supervisor	Id of the supervisor	**ObjectId**
	IsActive	Active status of the user	**Boolean**
	IsTrashed	Soft deletion state	**Boolean**

Table 2. 4-Data dictionary

Task	Id_Task	Task identification	ObjectId
	Title	Title of the task	String
	Date	Task Submission date	Date
	Priority	Task Priority	String
	Stage	Task Stage	String
	Evaluation	Task Evaluation	Object
	Sub_Task	Submitted Task	Array
	Assets	Task Assets	String
	Team	Task team	Array
Notification	Id_notification	Notification Identification	ObjectId
	Text	Notification message	String
	Task	Task identification	ObjectId
	NotiType	Notification Type	String
	IsRead	Identification of users who read the notification	Array
	Team	Identification of users	Array
Chat	Id_Chat	Chat Identification	ObjectId
	Participants	Members participating in the chat	Array
Message	Id_Message	Message Identification	ObjectId
	Chat_Id	Chat Identification	ObjectId
	Sender	User Identification	ObjectId
	Text	Message sent or received	String

5 UML Class Diagram [20]

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.

5.1 Purpose of Class Diagrams

The main purpose of class diagrams is to build a static view of an application. It is the only diagram that is widely used for construction, and it can be mapped with object-oriented languages. It is one of the most popular UML diagrams. Following are the purpose of class diagrams given below:

✓ It analyzes and designs a static view of an application.

✓ It describes the major responsibilities of a system.

✓ It is a base for component and deployment diagrams.

✓ It incorporates forward and reverse engineering.

5.2 Benefits of Class Diagrams

✓ It can represent the object model for complex systems.

✓ It reduces the maintenance time by providing an overview of how an application is structured before coding.

✓ It provides a general schematic of an application for better understanding.

✓ It represents a detailed chart by highlighting the desired code, which is to be programmed.

✓ It is helpful for the stakeholders and the developers.

The diagram below illustrates the class diagram for our system:

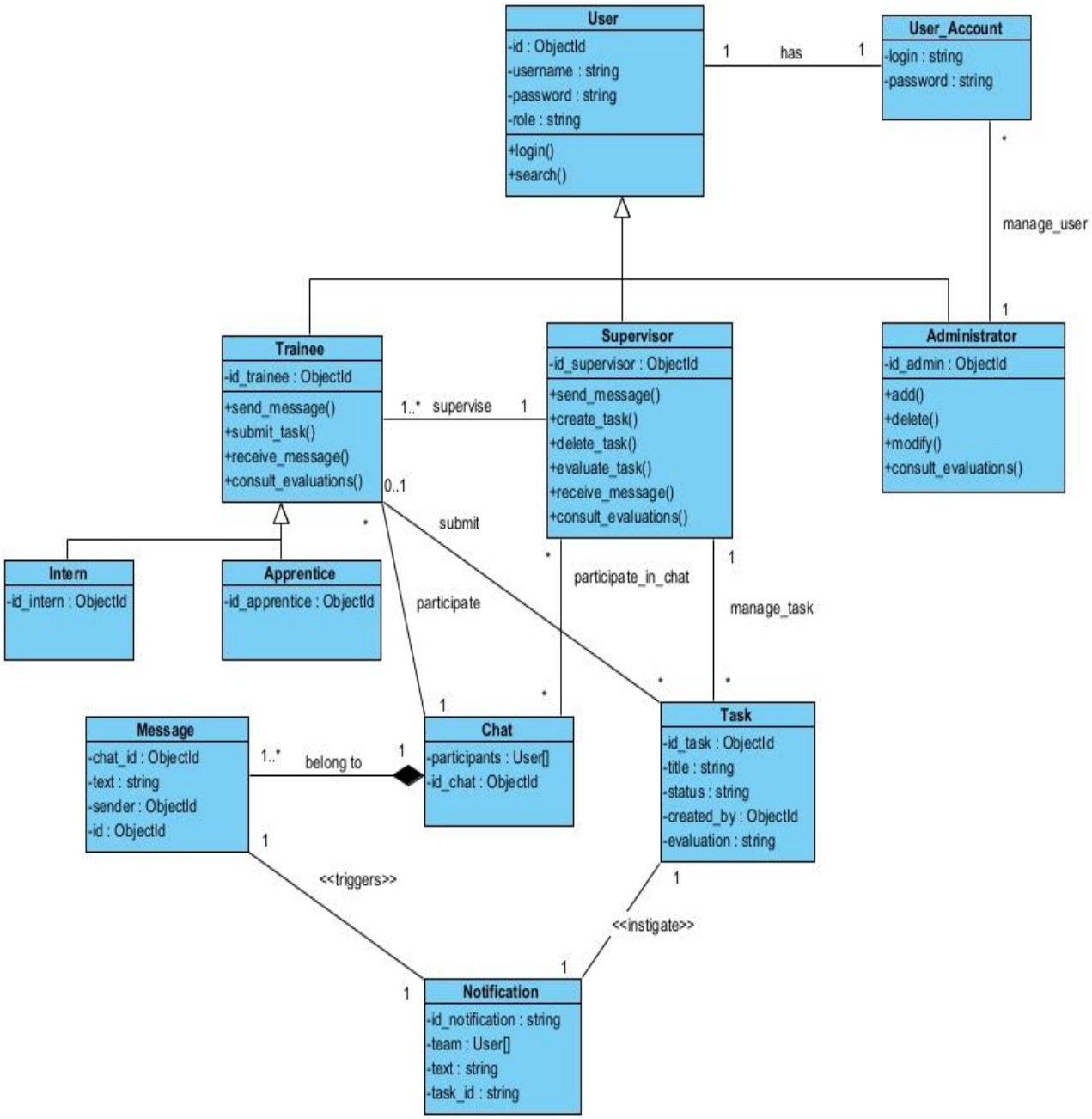

Figure 3. 5: Class Diagram

6 Rules Applied for MongoDB Schema Design [34]

When transitioning from a class diagram to a MongoDB schema, certain guidelines are followed to create an effective schema design. Below are the rules applied to our class diagram to model data in MongoDB:

6.1 Rule 1: Class Transformation

Each object class becomes a MongoDB collection.

Object properties are mapped to fields within MongoDB documents.

Object identifiers are often represented by the _id field, MongoDB's primary key, within documents.

6.2 Rule 2: Association Transformation

We distinguish three types of associations:

-Association 1..*: Add a foreign key attribute in the child relationship of the association. The attribute name should match the primary key of the parent relation of the association.

-Association *..* , n-ary and class-association: The class-association becomes a collection. The primary key of this collection is the concatenation of the identifiers of the classes connected to the association.

-Association 1..1: Add a foreign key attribute in the derived relation of the class having a minimum multiplicity equal to one. The attribute name should match the primary key of the derived relation of the class connected to the association.

6.3 Rule 3: Presence of Generalization (Method 1: Push-up)

Create a collection with all attributes of the classes. Add an attribute to distinguish the types of objects. These rules guide the process of transforming a class diagram into a MongoDB schema, ensuring a structured and efficient representation of data in MongoDB collections.

6.4 MongoDB schema design

User{

```
_id: ObjectId,
surname:{type:String,required:true},
   firstname:{type:String,required:true},
   dob:{type:Date,default:new Date()},
   startdate:{type:Date,default:new Date()},
   department:{type:String,required:true},
   role:{type:String,required:true},
   enddate:{type:Date,default:new Date()},
   phonenumber:{type:String,required:false},
   theme:{type:String,required:false},
   email:{type:String,required:true,unique:true},
   password:{type:String,required:true},
   isAdmin:{type:Boolean,required:true,default:false},
   isSup:{type:Boolean,required:true,default:false},
   supervisor:{type:Schema.Types.ObjectId,ref:"User"},
   isActive:{type:Boolean,required:true,default:true},
   isTrashed:{type:Boolean,default:false}
```

}

Chat{

```
_id: ObjectId,
   chatId: { type: Schema.Types.ObjectId, ref: "Chat" },
   senderId: {type:Schema.Types.ObjectId,ref:"User"},
   text:  String
```

}

Task{

```
_id: ObjectId,
  title:{type:String,required:true},
  date:{type:Date, default:new Date()},
   priority:{type:String,default:"normal",enum:["high","medium","normal","low"},
   stage:{type:String,default:"todo",enum:["todo","in progress","completed"]},
   Evaluation:{
    performance:{ type:String, default:"",enum:["very bad", "bad","fair","good","very good",
                    "excellent","pending" ]},
      Mark:String,
      date:{type:Date, default:new Date()},
      by:{type:Schema.Types.ObjectId,ref:"User"}
   },
  subTask:[{title:String,tag:String,assets:[String],by:{type:Schema.Types.ObjectId,ref:"User"}
     ,submitedAt:{type:Date, default:new Date()} }],
   assets:[String],
   team:[{type:Schema.Types.ObjectId,ref:"User"}]
}
```

Notification{

```
_id: ObjectId,
team: [{ type: Schema.Types.ObjectId, ref: "User" }],
   text: { type: String },
   task: { type: Schema.Types.ObjectId, ref: "Task" },
   notiType: { type: String, default: "alert", enum: ["alert", "message"] },
   isRead: [{ type: Schema.Types.ObjectId, ref: "User" }],
```

}

Message{

```
 _id: ObjectId,
   chatId: { type: Schema.Types.ObjectId, ref: "Chat" },
   senderId: {type:Schema.Types.ObjectId,ref:"User"},
   text: {
     type: String,
   },
```

}

7 Conclusion

In this fourth chapter, we presented and detailed the design stage of our system. The development of the class diagram as well as the mongoDB schema helped us build the various collections of our application that we will use in the development of our application. We also presented sequence diagrams, emphasizing the chronology of messages exchanged between system objects. In the next chapter, we will describe the implementation and testing stage.

CHAPTER 5

REALIZATION

Chapter 5: Realization

1 Introduction

This chapter marks the concluding stage of the software development process. We will leverage the data, information gathered, and insights gained from the earlier chapters. We start by outlining the development environment and languages employed in building our application, followed by a discussion of the system's physical architecture, and a preview of several interfaces.

2 Development Environment

2.1 Visual Studio Code

Visual Studio Code (VS Code) is a code editor that is developed by Microsoft. It is a lightweight and cross-platform code editor that can be used to develop a wide range of applications, from simple scripts to complex web applications. Visual Studio Code provides a number of features that make it easier to write and debug code, including code completion, debugging tools, and support for source control systems.

One of the main benefits of using Visual Studio Code is that it provides a rich set of editing and debugging tools. This includes features such as code completion, which suggests possible completions for partially written code, and error highlighting, which alerts the developer to any syntax errors or other problems in the code. Visual Studio Code also includes a powerful debugger that allows developers to step through their code, set breakpoints, and inspect the state of their application while it is running.

Another benefit of Visual Studio Code is that it is highly customizable, and can be extended with a wide range of plugins and extensions. This allows developers to tailor the code editor to their specific needs and preferences, and to add additional features and capabilities to the editor. Visual Studio Code also integrates with many popular tools and services, such as Git and Azure, which allows developers to easily use these tools and services within the code editor. [21]

Figure 3. 6: Logo «Vs Code»

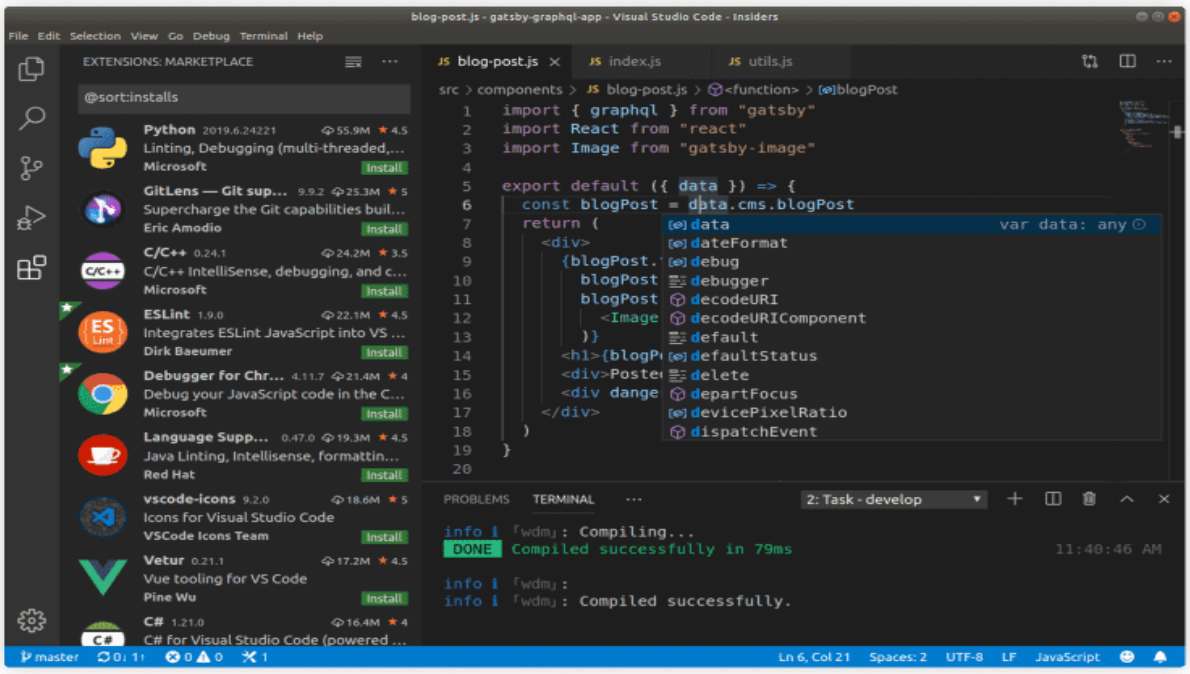

Figure 3. 7: Code Editor « Vs Code »

3 DataBase

3.1 MongoDB

MongoDB is an open source NoSQL database management program. NoSQL (Not only SQL) is used as an alternative to traditional relational databases. NoSQL databases are quite useful for working with large sets of distributed data. MongoDB is a tool that can manage document-oriented information, store or retrieve information.

MongoDB is used for high-volume data storage, helping organizations store large amounts of data while still performing rapidly. Organizations also use MongoDB for its ad-hoc queries, indexing, load balancing, aggregation, server-side JavaScript execution and other features.[22]

Figure 3. 8: Logo « MongoDB»

3.2 Firebase

Firebase is a Backend-as-a-Service (BaaS) app development platform that provides hosted backend services such as a realtime database, cloud storage, authentication, crash reporting, machine learning, remote configuration, and hosting for your static files.[23]

Figure 3. 9: Logo « Firebase»

4 Development Tools

4.1 Java Script

JavaScript is a lightweight, cross-platform, single-threaded, and interpreted compiled programming language. It is also known as the scripting language for webpages. It is well-known for the development of web pages, and many non-browser environments also use it.

JavaScript is a weakly typed language (dynamically typed). JavaScript can be used for Client-side developments as well as Server-side developments.[24]

Figure 4. 1: Logo «Java Script»

4.2 Node Js

Node.js is an open-source and cross-platform JavaScript runtime environment. It runs on Chrome's V8 JavaScript engine. It allows developers to run JavaScript code on the server. Node.js enables developers to get into the server-side world.[25]

Figure 4. 2: Logo «Node Js»

4.3 HTML (HyperText Markup Language)

HTML, or HyperText Markup Language, is the standard markup language used to create web pages. It's a combination of Hypertext, which defines the link between web pages, and Markup language, which is used to define the text document within tags to structure web pages. This language is used to annotate text so that machines can understand and manipulate it accordingly. HTML is human-readable and uses tags to define what manipulation has to be done on the text.[26]

Figure 4. 3: Logo «HTML»

4.4 CSS (Cascading style sheets)

CSS stands for Cascading style sheets. It describes to the user how to display HTML elements on the screen in a proper format. CSS is the language that is used to style HTML documents.[27]

Figure 4. 4: Logo «CSS»

4.5 Tailwind CSS

Tailwind CSS is a utility-first CSS framework that streamlines web development by providing a set of pre-designed utility classes. These classes enable rapid styling without writing custom CSS, promoting consistency and scalability. Tailwind's approach shifts focus from traditional CSS components to functional classes, empowering developers to efficiently build responsive and visually appealing interfaces with minimal effort.[28]

Figure 4. 5: Logo «Tailwind CSS»

4.6 Headless UI

Headless UI is a library focused on creating accessible, unstyled UI components. Unlike traditional UI libraries, it offers developers bare-bones components, void of predefined styles.[29]

Figure 4. 6: Logo «Headless UI»

4.7 React Js

React is a JavaScript-based UI development library. Although React is a library rather than a language, it is widely used in web development. The library first appeared in May 2013 and is now one of the most commonly used frontend libraries for web development.[30]

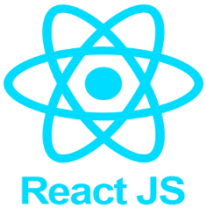

Figure 4. 7: Logo «React»

4.8 Redux Toolkit

Redux Toolkit or RTK is a node package that simplifies the development by providing utility functions. It is made to simplify the creation of redux store and provide easy state management. It can be easily installed using simple npm commands.[31]

Figure 4. 8: Logo «Redux Toolkit»

4.9 Socket.IO

Socket.IO is a library that enables real-time, bidirectional and event-based communication between the browser and the server. The client will try to establish a WebSocket connection if possible, and will fall back on HTTP long polling if not. WebSocket is a communication protocol which provides a full-duplex and low-latency channel between the server and the browser. [32]

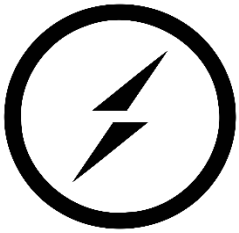

Figure 4. 9: Logo «Socket.IO»

5 Software Design Tool

5.1 Visual Paradigm

It is a software design tool tailored for agile software projects. It supports UML, BPMN, ERD, DFD, SysML. It also supports use cases, wireframing, code engineering etc. Visual Paradigm is a tool in the Diagramming category of a tech stack.[33]

Figure 4. 10: Logo «Visual Paradigm»

6 Presentation of the interfaces of our application

We represent the interfaces of our application with the following figures:

6.1 Home Interface

The homepage is the first page that users of the web application access. The name of the application is 'EPB Career Craft'. This page will display an image of the organization as well as the functionalities to authenticate and to learn more about internships in the FAQ section.

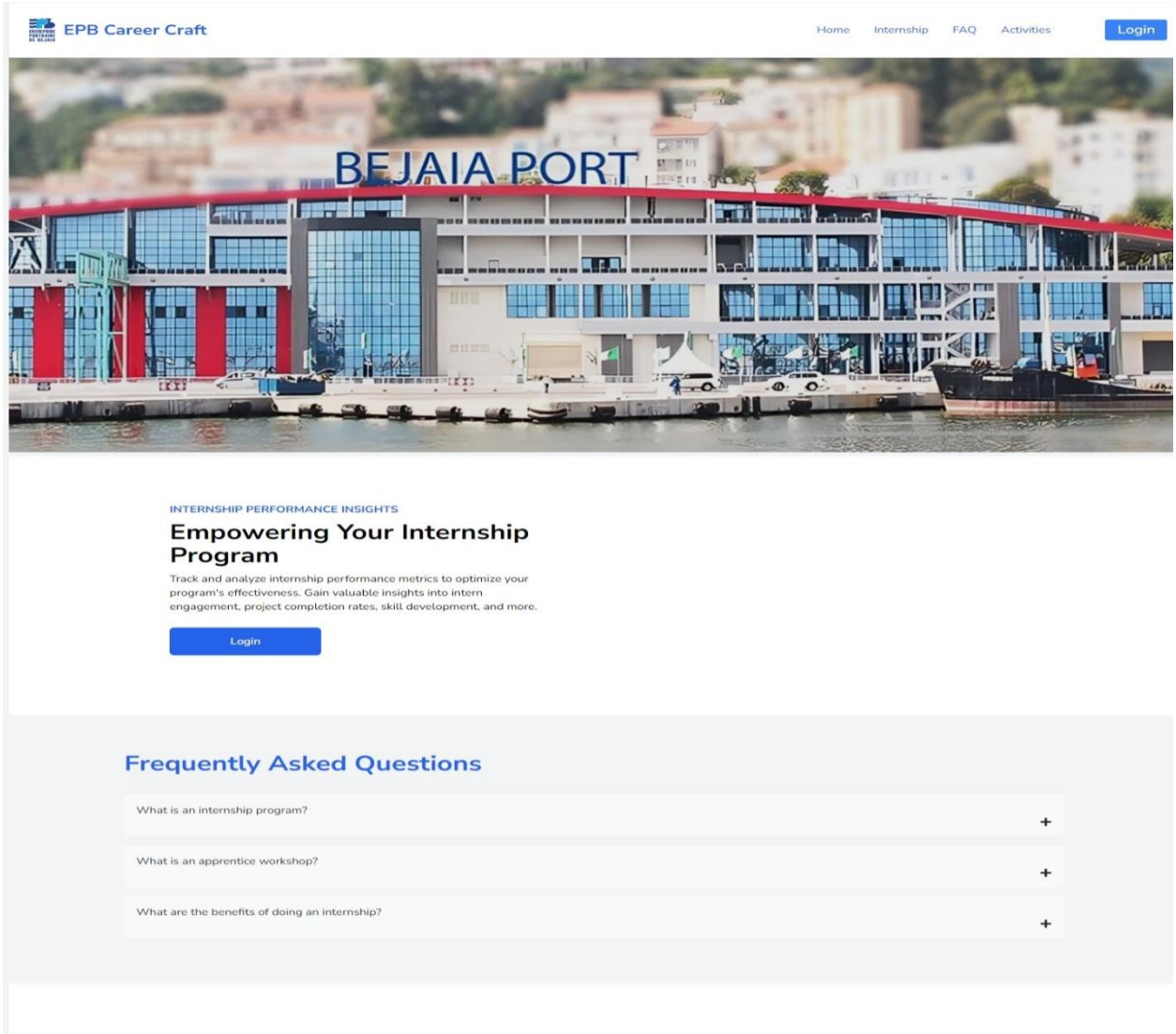

Figure 5. 0: «Hero section of Home Page»

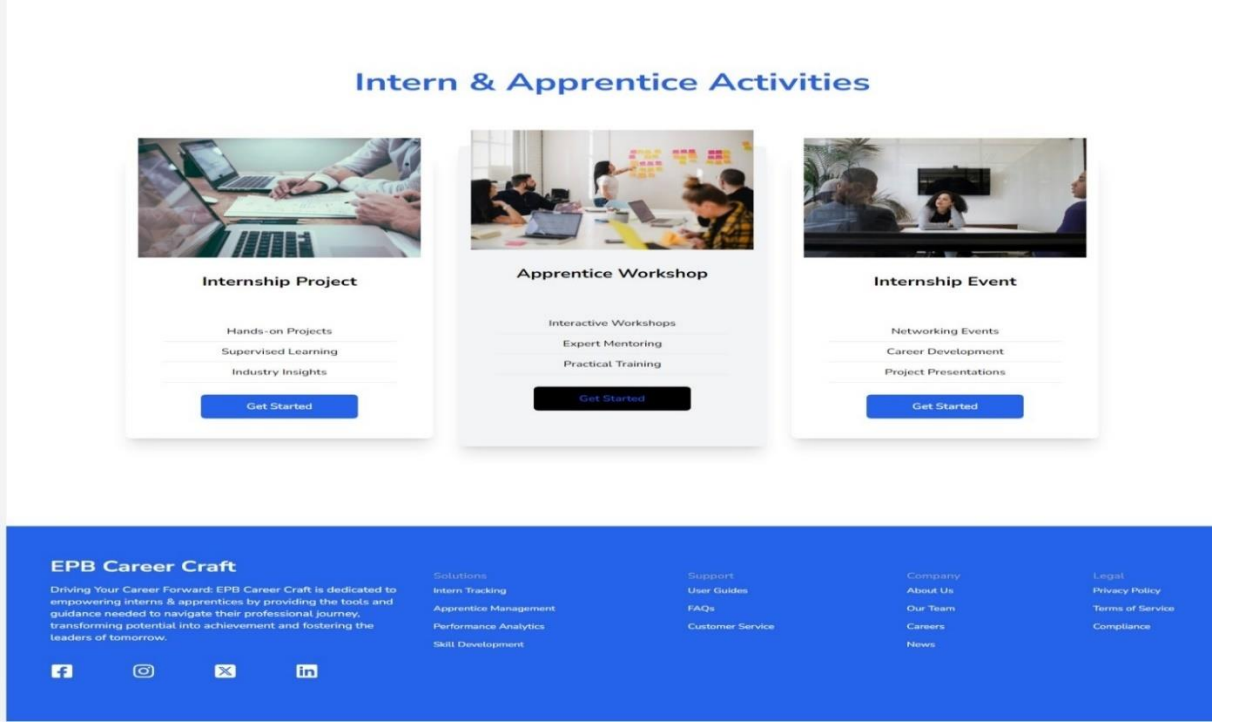

Figure 5. 1: «Footer section of Home Page»

6.2 Authentication Interface

After accessing the application and clicking on the login button, the user is prompted to enter their username and password. If the entered information is either invalid or missing, the system will display an error message to notify the user of the issue.

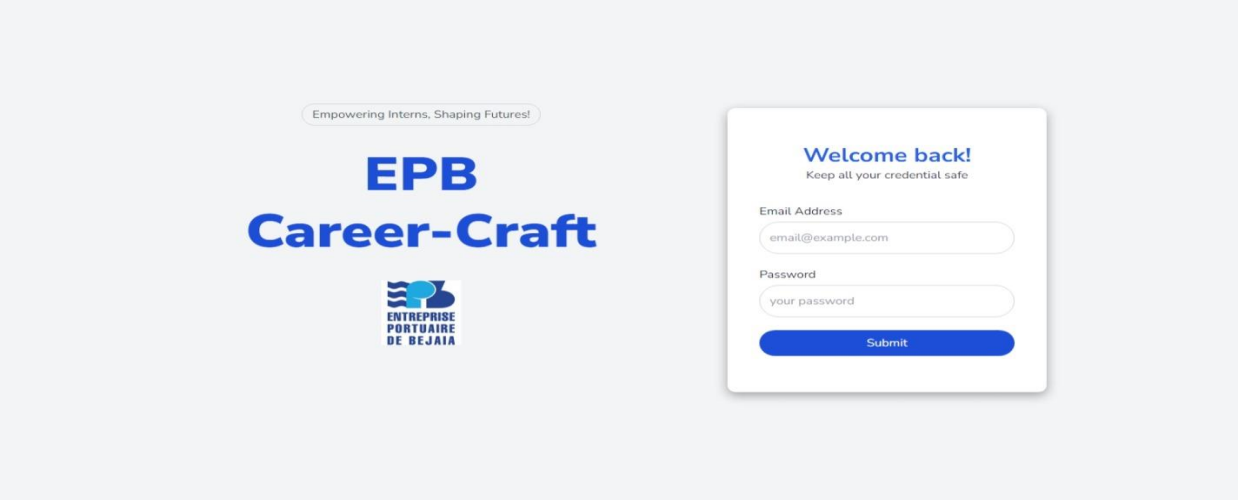

Figure 5. 2: «Authentication Page»

6.3 Admin Dashboard Interface

After authenticating with the correct credentials and an active account, the administrator page will be displayed for a user with admin privileges.

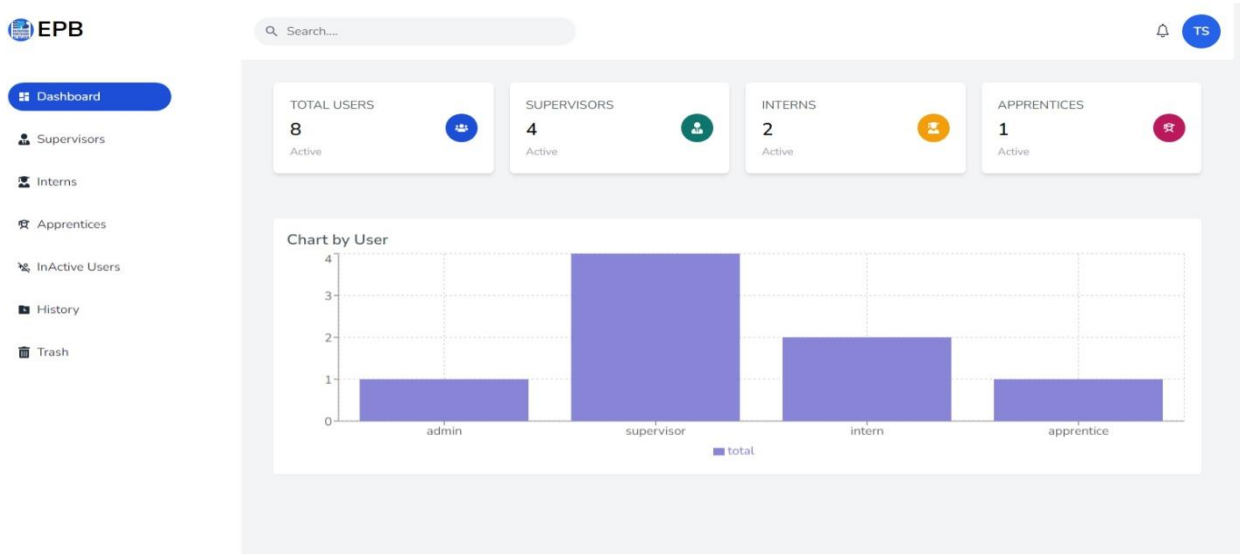

Figure 5. 3: «Admin Dashboard»

6.4 List of Active Supervisors Interface

The administrator can consult the list of active supervisors in the supervisor interface as illustrated by the diagram below:

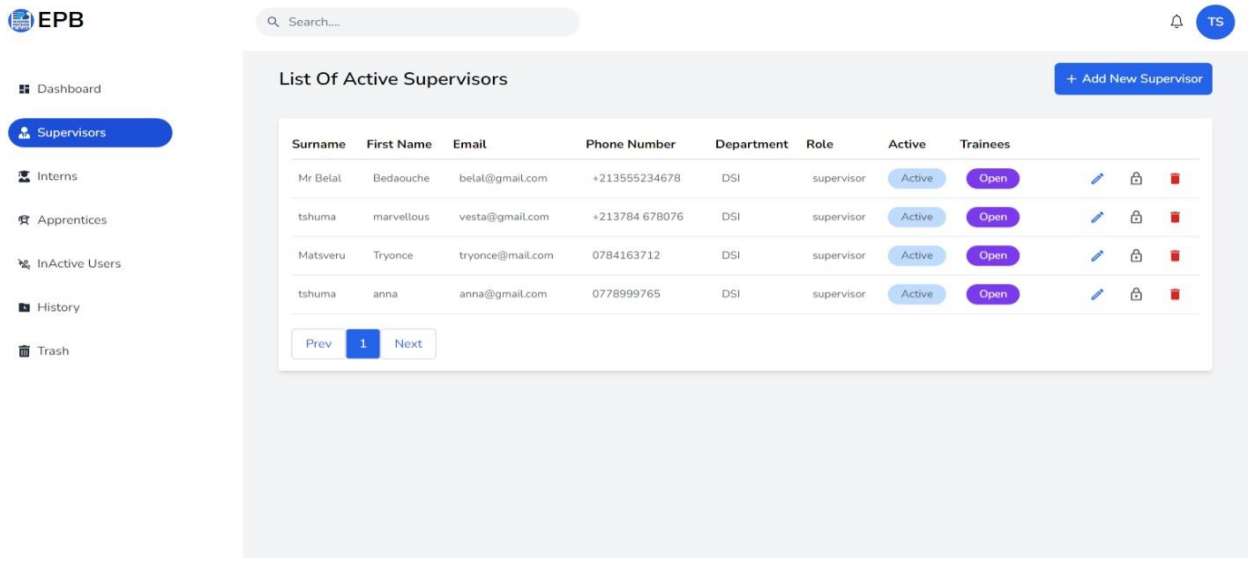

Figure 5. 4: «List of Active Supervisors»

6.5 List of Active Interns Interface

The diagram below illustrates the list of active interns interface.

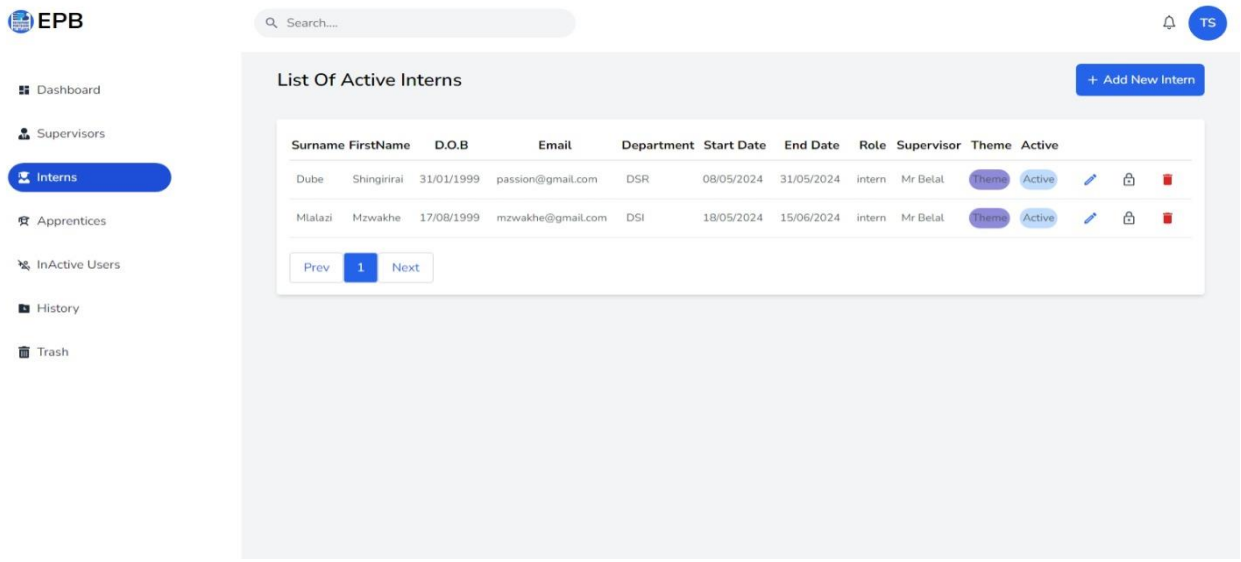

Figure 5. 5: «List of Active Interns»

6.6 List of Active Apprentices Interface

The diagram below illustrates the list of active apprentices interface.

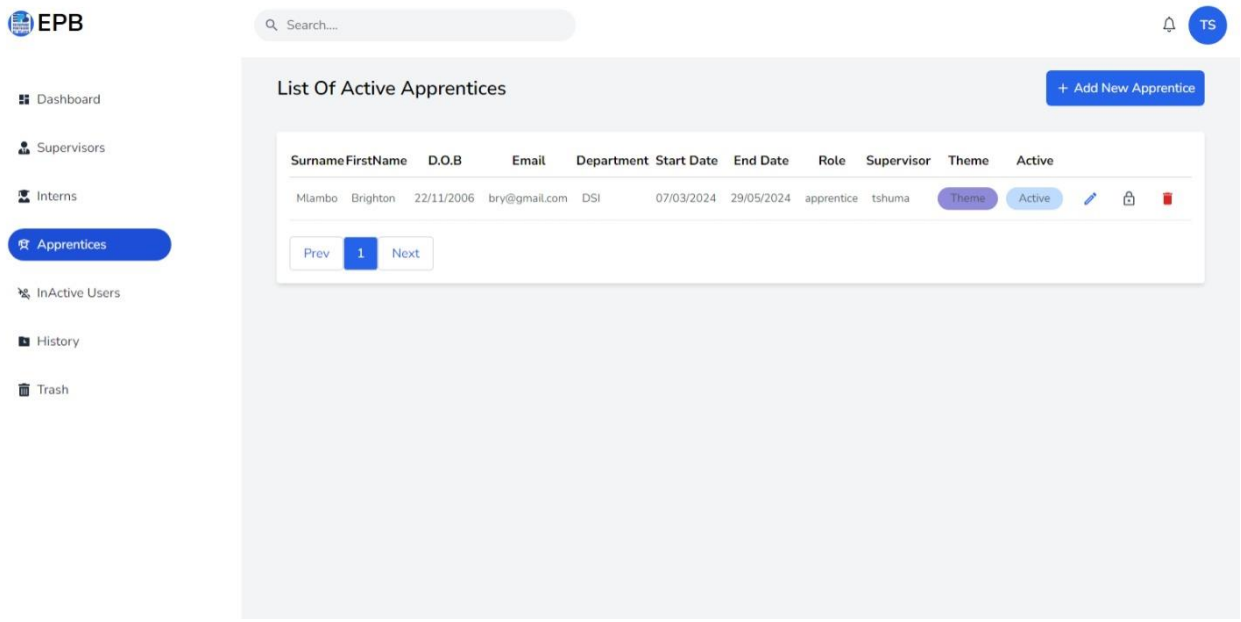

Figure 5. 6: «List of Active Apprentices»

6.7 List of InActive Users Interface

The diagram below depicts the list of all inactive users in the system that can be consulted by the administrator by clicking the inactive users tab.

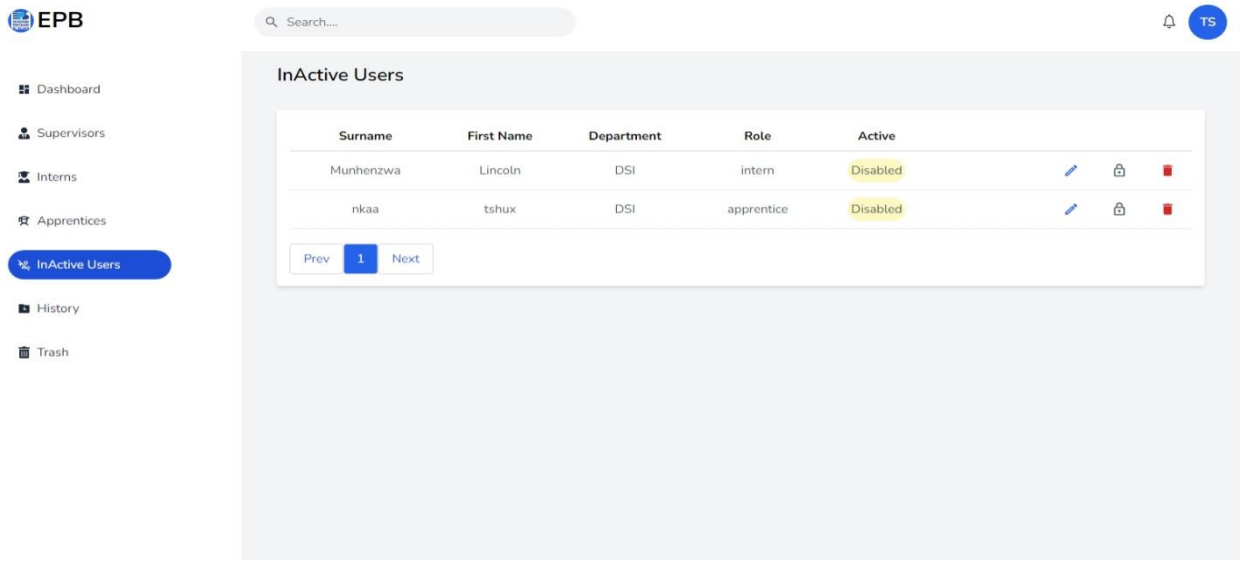

Figure 5. 7: «List of InActive Users»

6.8 Trainee History Interface

The history of all trainees can be consulted by the administrator as show by the diagram below.

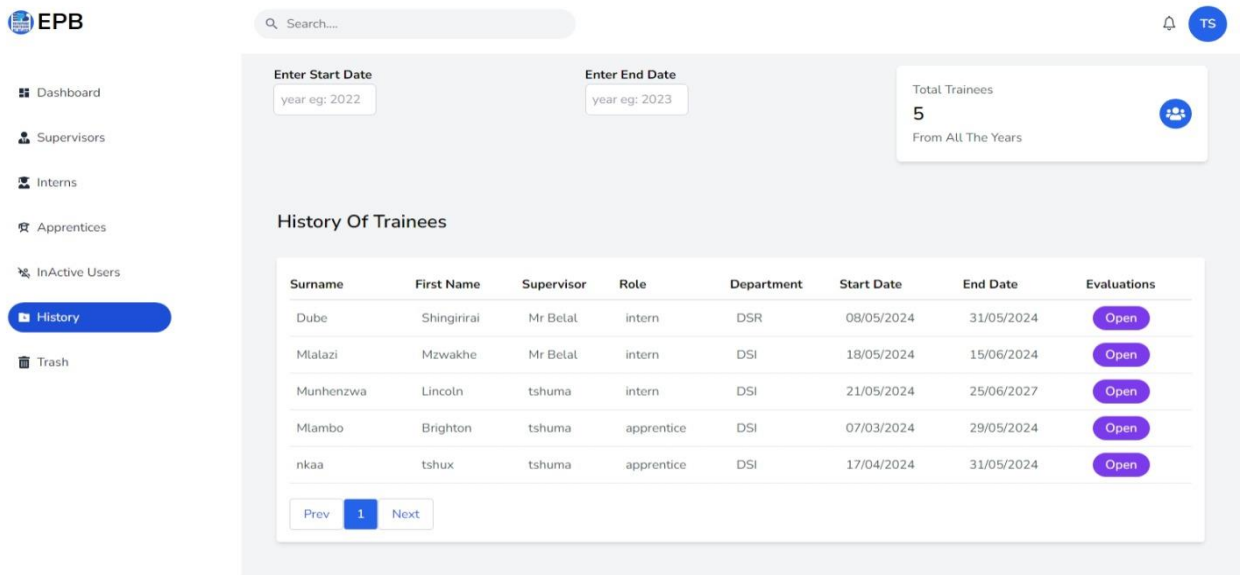

Figure 5. 8: «Trainee History»

6.9 Trashed Users Interface

The deleted users will first appear in the trashed interface as shown by the diagram below, this is where the administrator has the options to restore a user account or permanently delete it from the database.

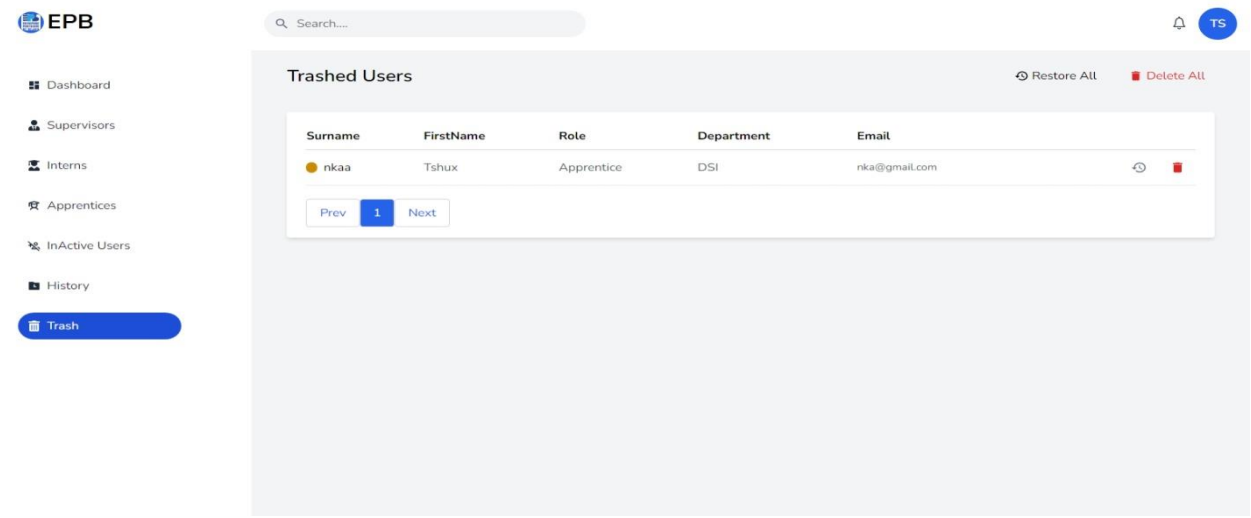

Figure 5. 9: «Trashed Users»

6.10 Create Supervisor Account Interface

The diagram below illustrates the interface to create a supervisor account.

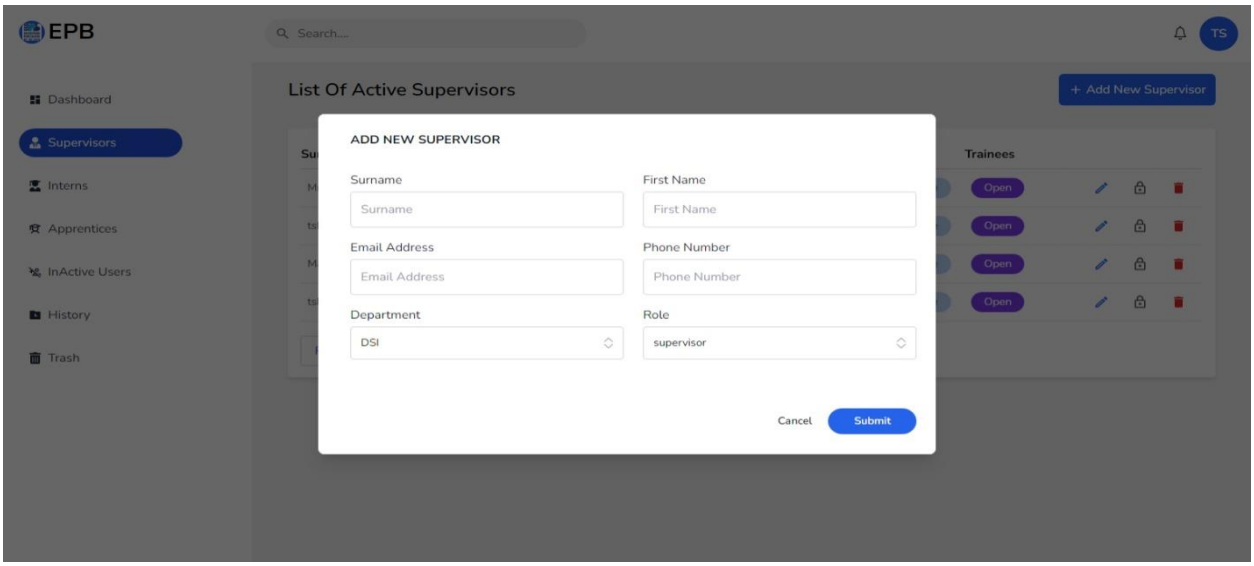

Figure 5. 10: «Create Supervisor Account»

6.11 Supervisor and Trainee Dashboard Interface

The diagram below shows the supervisor and trainee dashboard interface.

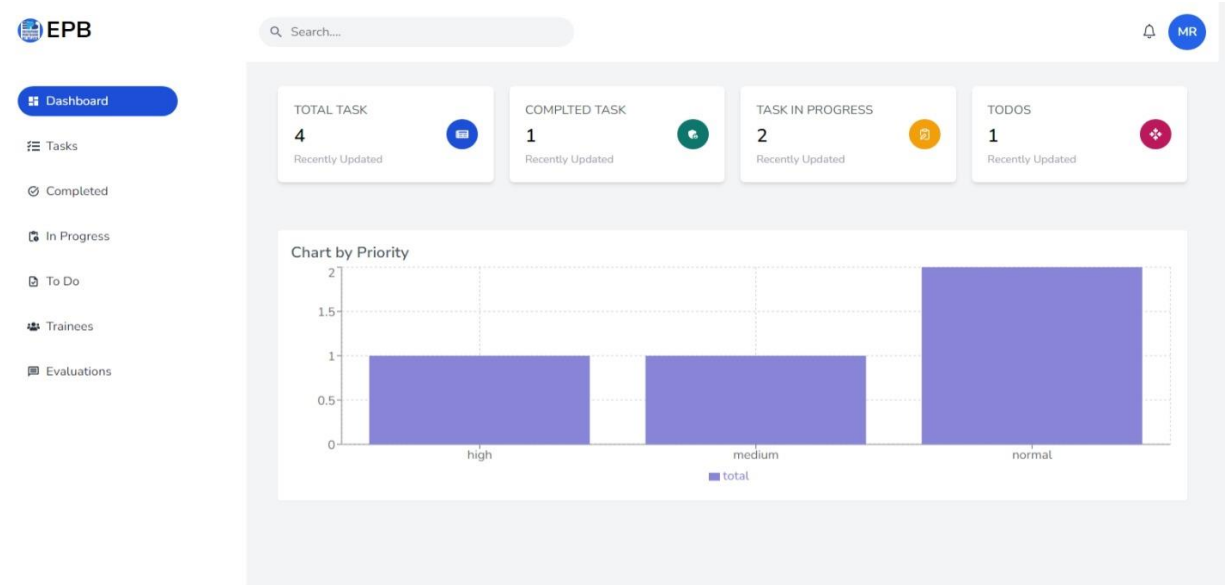

Figure 6. 1: « Supervisor and Trainee Dashboard »

6.12 Delete User Confirmation Interface

The diagram below illustrates the confirmation to delete a user account interface.

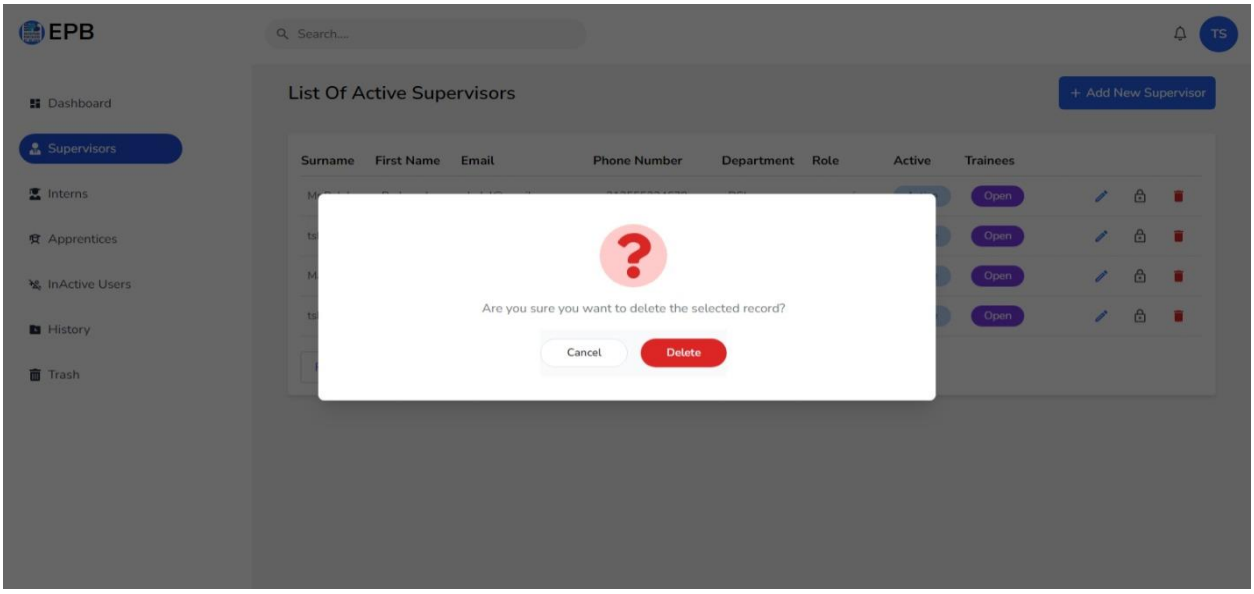

Figure 6. 2: «Delete User Confirmation»

6.13 Change Password Interface

The diagram below shows the interface to change a user account password.

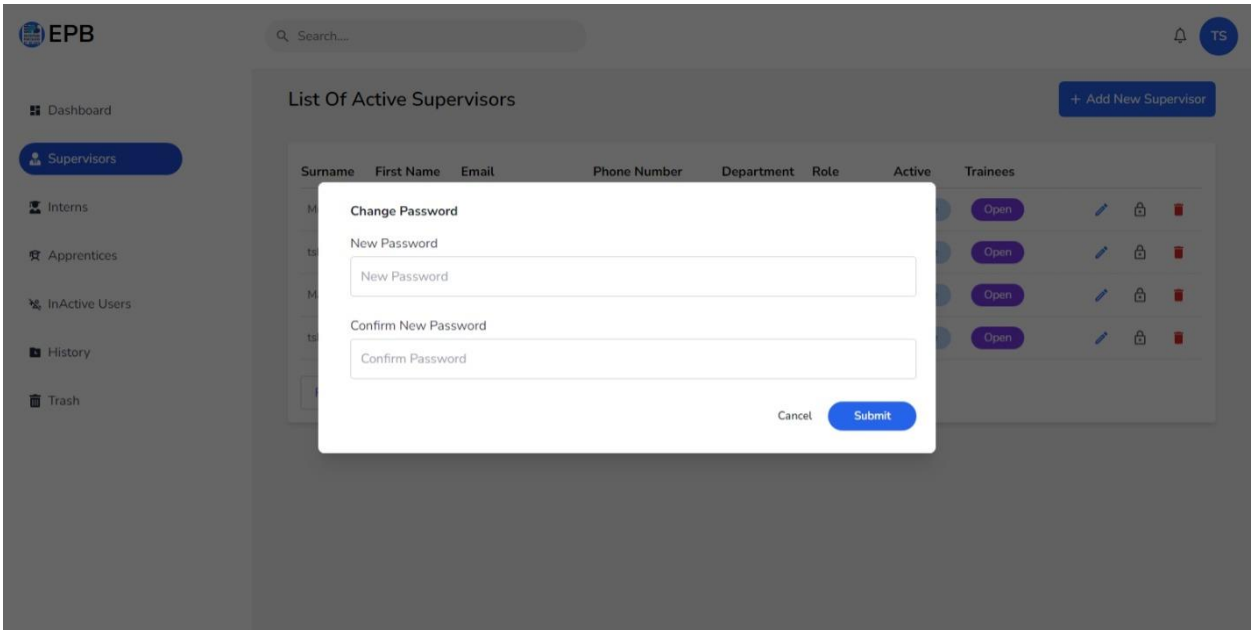

Figure 6. 3: «Change Password »

6.14 Create Trainee Account Interface

The diagram below depicts the interface to create a trainee account.

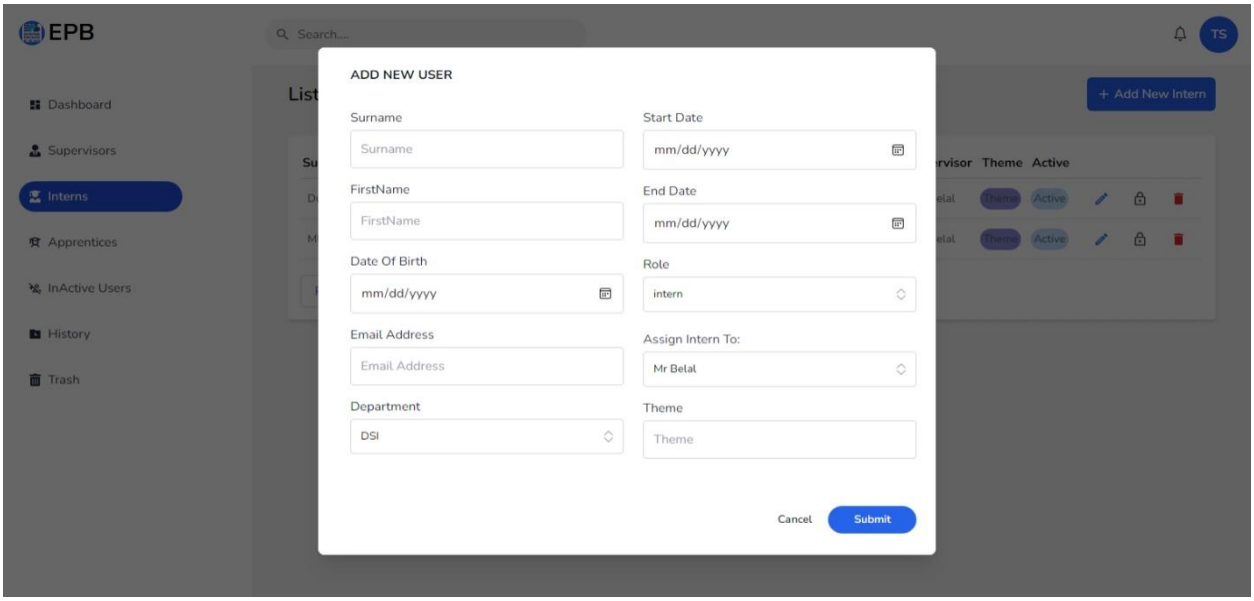

Figure 6. 4: «Create Trainee Account»

6.15 Task Interface

The diagram below illustrates the list of tasks created by the supervisor assigned to the trainees.

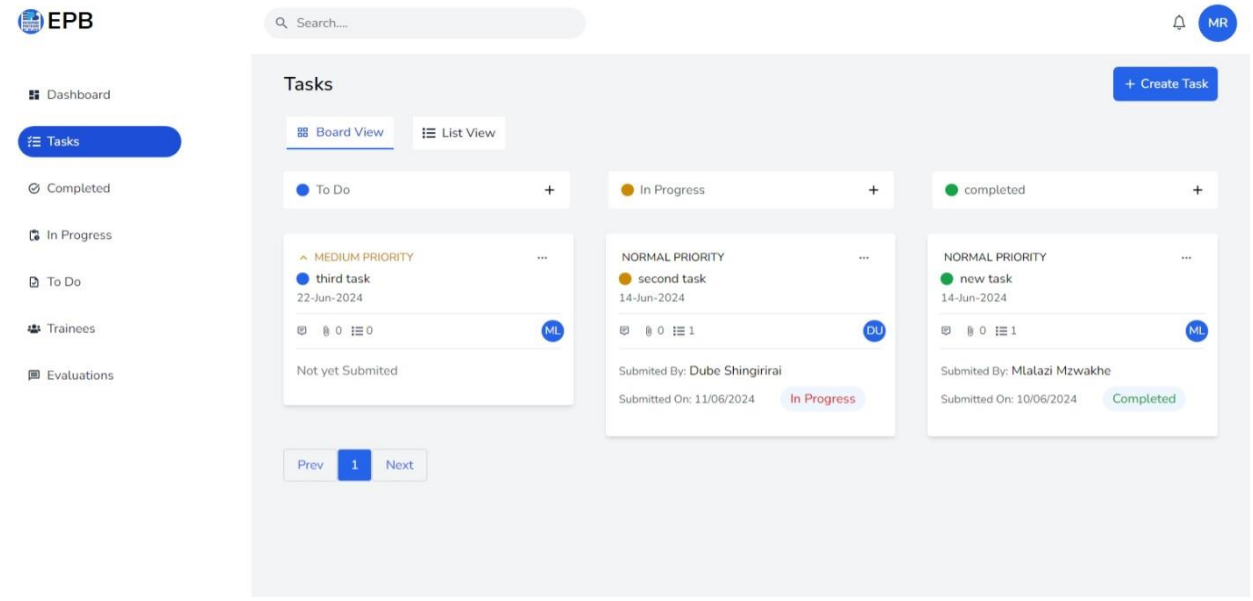

Figure 6. 5: «Task Page»

6.16 Create Task Interface

The diagram below shows the interface to create a task.

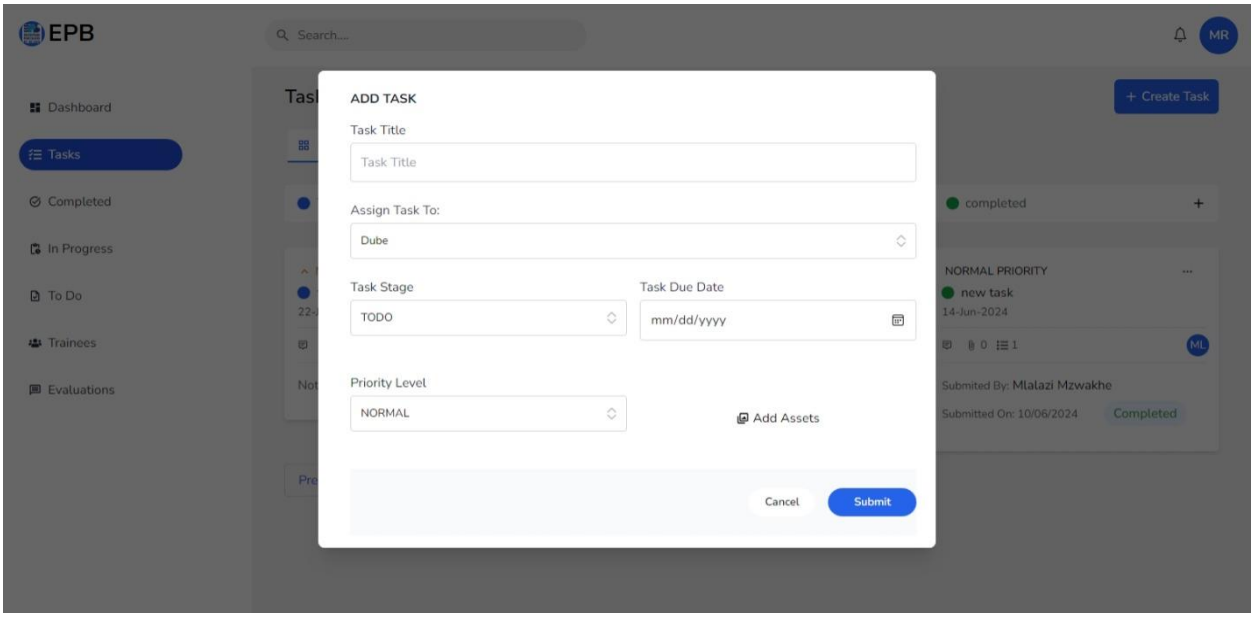

Figure 6. 6: «Create Task»

6.17 Submit Task Interface

The diagram below shows the interface to submit a task for evaluation.

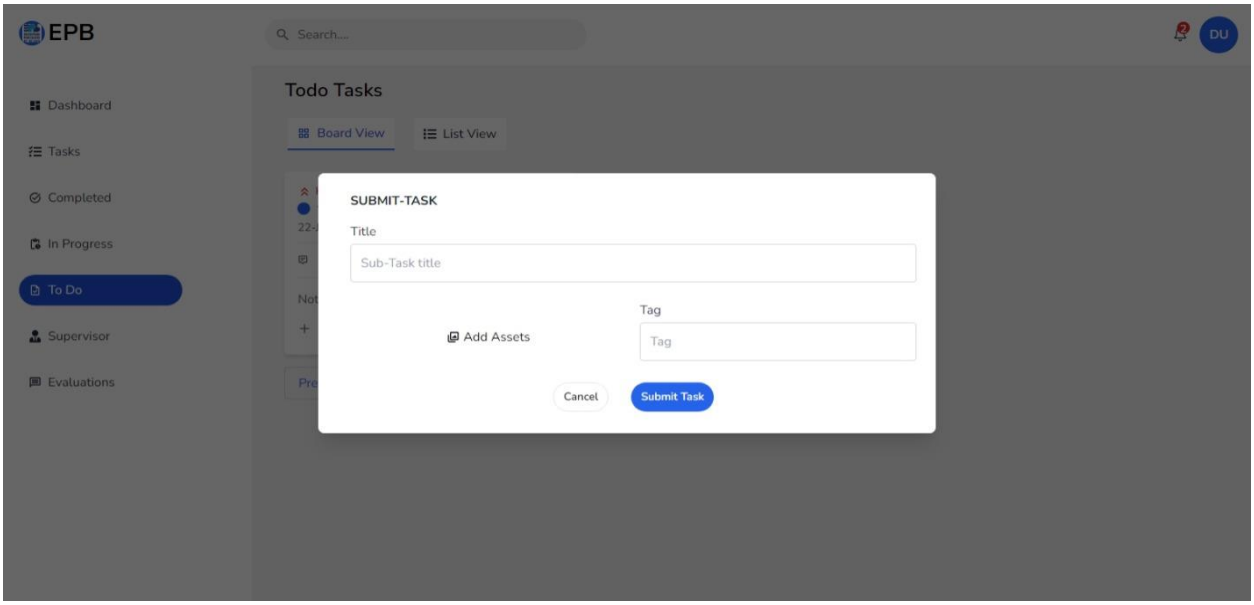

Figure 6. 7: «Submit Task»

6.18 Task Details Interface

The diagram below illustrates the details of a task.

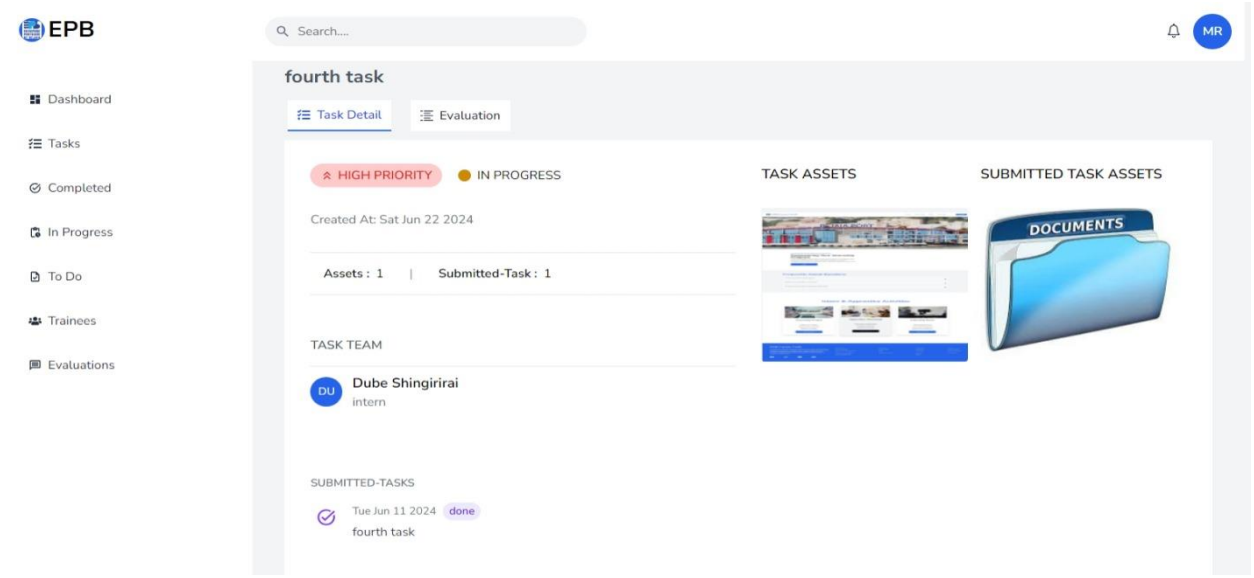

Figure 6. 8: «Task Details»

6.19 Task Evaluation Interface

The diagram below shows the interface to evaluate the submitted task.

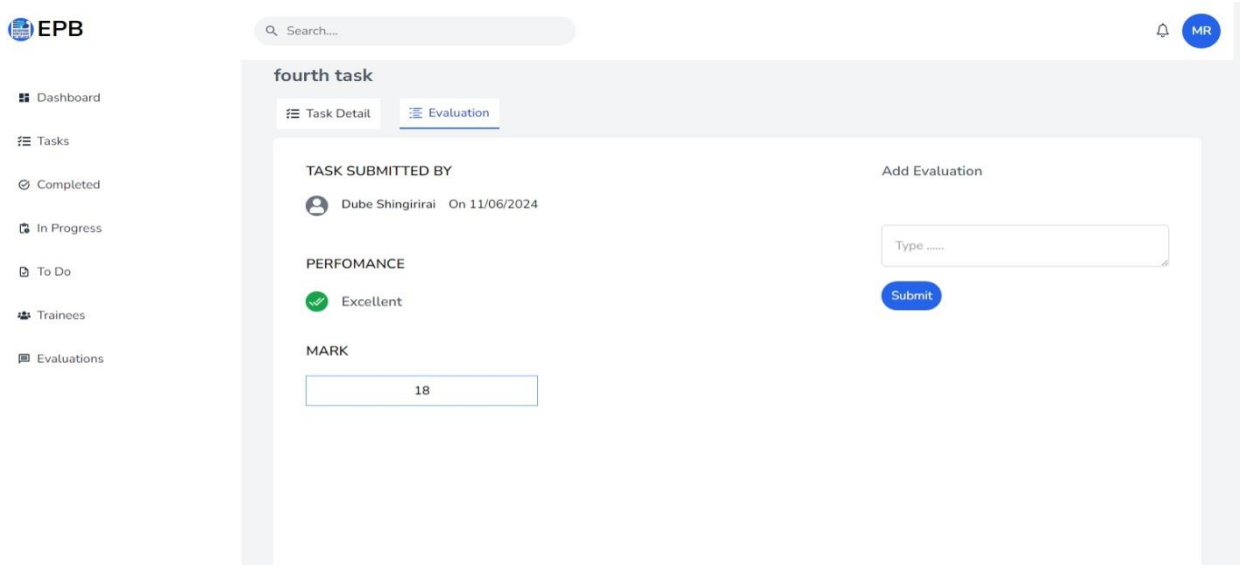

Figure 6. 9: «Task Evaluation»

6.10 Chat Interface

The diagram below shows the chat interface of trainees and their supervisors.

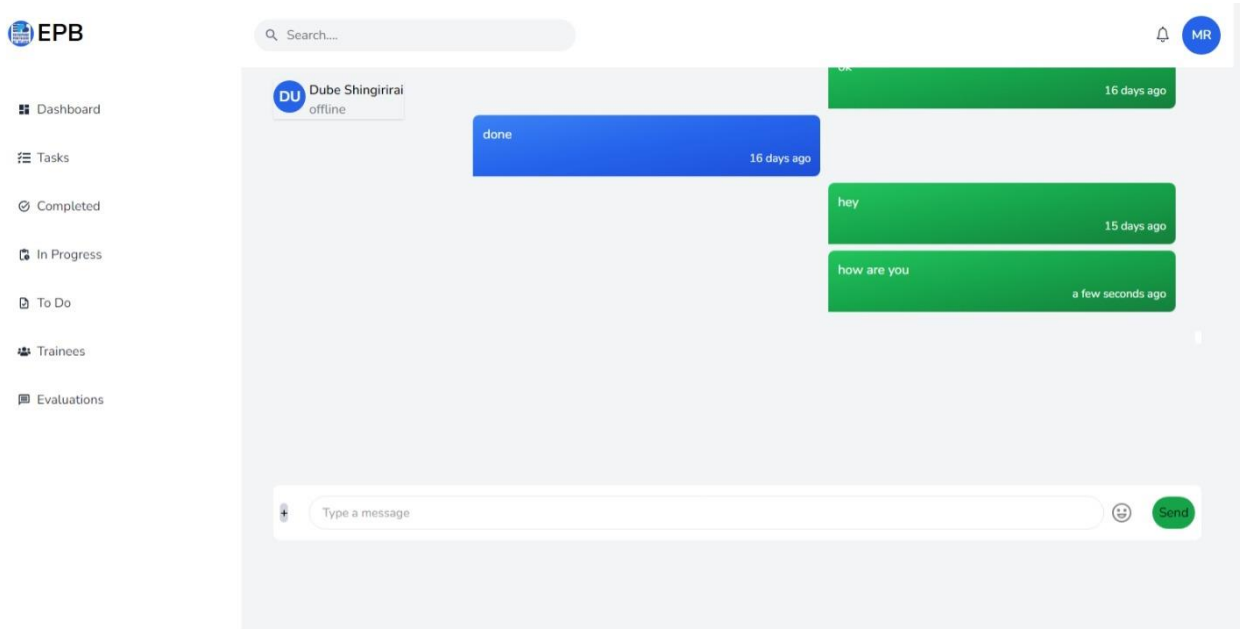

Figure 6. 10: «Chat»

6.11 Notifications Interface

The diagram below shows all unread notifications of the user.

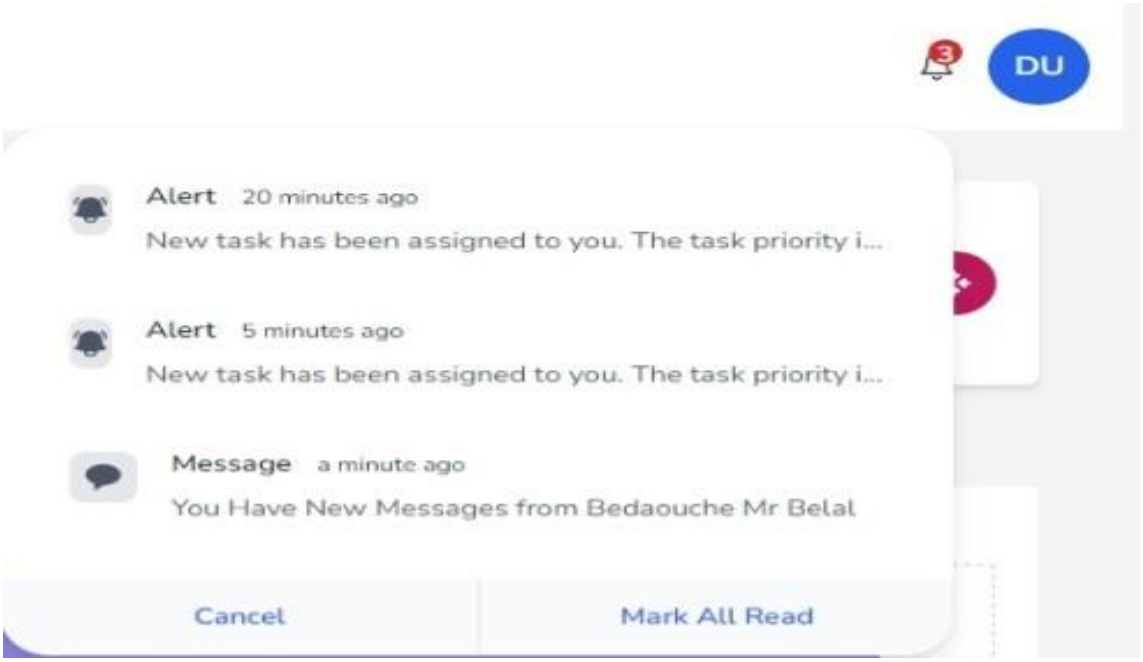

Figure 6. 11: «Notifications»

6.12 Evaluations Interface

The diagram below illustrates the interface of evaluations.

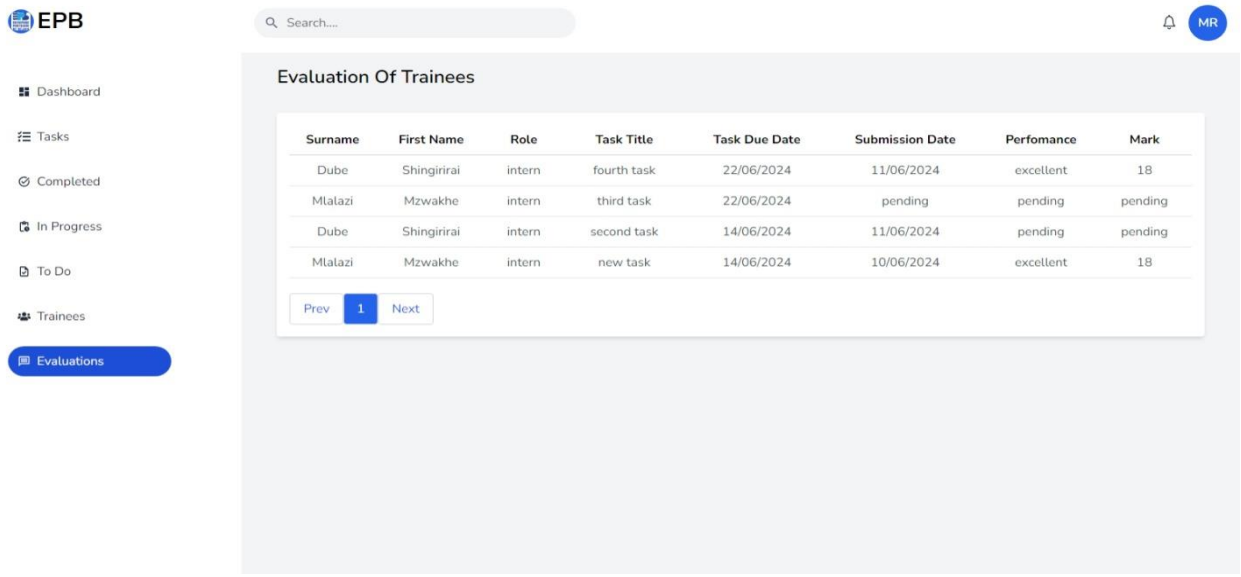

Figure 6. 12: «Evaluations»

7 Conclusion

In this final chapter, we initially introduced the various tools and languages that facilitated the completion of this project. Subsequently, we demonstrated some graphical user interfaces of our application.

General Conclusion

General Conclusion

As part of my final year project, we proposed to set up a web application for managing interns and apprentices at the EPB. This solution addresses the needs of the Human Resources Department by enabling effective tracking of interns and apprentices. It facilitates the consultation of their progress, submission of tasks, validation by supervisors, and the preparation of administrative documents using an adapted model. This system helps overcome daily challenges such as difficulty in tracking the progress and evaluations of interns, time loss, and delays in preparing printable documents.

The web application we developed includes several advanced features to streamline the management process. Among these features is a chat functionality that allows real-time communication between interns, apprentices, and their supervisors. This ensures quick resolution of queries and fosters better communication. Additionally, the system enables the submission of tasks by interns and apprentices, which can then be validated by supervisors. This feature helps in maintaining a structured workflow and ensures that all tasks are properly monitored and evaluated.

We opted for the UML formalism to carry out the system analysis and design. Using a series of diagrams, we were able to clearly and precisely model the system's behavior.

Finally, we began the implementation using JavaScript and its frameworks and libraries such as React.js and Node.js, along with other programming languages such as HTML, CSS, and Socket.IO for real-time communication. For the database management system, we opted for MongoDB.

The completion of this project has not only provided me with a valuable opportunity to enhance my design and programming skills while acquiring new knowledge but also allowed me to familiarize myself with the requirements and challenges of the professional world.

Perspectives

Although the main objectives of our project have been achieved, the application we developed could be enriched with additional advanced features. Potential improvements include incorporating AI-driven analytics to provide detailed insights into the performance and progress of interns and apprentices, aiding in better decision-making for the training department. Furthermore, AI-powered chatbots could be integrated to assist interns and apprentices by answering common questions, providing instant feedback, and guiding them through various tasks, thereby enhancing the overall user experience

References

Webliography

[2] https://ouidou.fr/2019/04/30/quest-ce-quune-application-web--definition-2019

[3] https://aws.amazon.com/fr/what-is/web-application/

[4] https://digitiz.fr/blog/definition-application-web/

[5] https://www.bairesdev.com/blog/types-web-application-development/

[6] https://www.simplilearn.com/software-development-methodologies-article

[7] https://www.educative.io/answers/what-is-a-unified-process-model

[9]https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/#:~:text=UML%2C%20short%20for%20Unified%20Modeling,business%20modeling%20and%20other%20non%2D

[10] https://stfalcon.com/en/blog/post/user-interface-web-design-principles

[11]https://www.adcisolutions.com/knowledge/web-development-methodologies-and-approaches

[12] https://www.techmagic.co/blog/best-application-deployment-strategies

[13] https://www.linkedin.com/advice/1/what-best-way-handle-web-application-maintenance

[15] https://www.geeksforgeeks.org/use-case-diagram/

[16] https://www.techtarget.com/searchsoftwarequality/definition/use-case

[19] https://www.academia.edu/49616533/Developing_Sequence_Diagrams_in_UML

[20] https://www.javatpoint.com/uml-class-diagram

[21] https://www.sandoba.com/en/knowledge-base-glossary/ide-visual-studio-code-vs-code/

[22] https://www.techtarget.com/searchdatamanagement/definition/MongoDB

[23] https://docs.flutter.dev/data-and-backend/firebase

[24] https://www.geeksforgeeks.org/introduction-to-javascript/

[25] https://www.geeksforgeeks.org/nodejs/

[26] https://www.geeksforgeeks.org/html-introduction/

[27] https://www.javatpoint.com/what-is-css

[28] https://www.geeksforgeeks.org/introduction-to-tailwind-css/

[29]https://medium.com/@jill6666/what-is-headless-ui-unlocking-flexibility-and-accessibility-3c7f9bec5a23

[30] https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs

[31] https://www.geeksforgeeks.org/redux-toolkit/

[32] https://socket.io/docs/v3/

[33] https://stackshare.io/visual-paradigm

[34]https://www.mongodb.com/developer/products/mongodb/mongodb-schema-design-best-practices/

Bibliography

[1] Internal documents of the EPB

[8] Pascal ROQUES, Les Cahiers du Programmeur UML 2 :Modéliser une Application Web, ISBN :978-2-212-12389-0, 4 ème édition, Groupe Eyerolles, 2008.

[14] Tiwari, S., & Gupta, A. (2015). A systematic literature review of use case specifications research. Information and Software Technology, 67, 128-158.

[17] Gilles Roy « Conception de base de données ave UML », Université de Quebec, 1 ère édition, 2009.

[18] Letaw, L. (2024). Unified Modeling Language Class and Sequence Diagrams. *Handbook of Software Engineering Methods*.

Abstract

Abstract

A Web Application is an application hosted on a server and accessible through a web browser. This technology allows the company, its subsidiaries, and mobile users to access the application conveniently. In our work, we developed a Web Application for managing interns and apprentices at EPB. The project includes sections for requirements specification, analysis and design, and concludes with the implementation of the final application.

We choose the Unified Process (UP) methodology based on the Unified Modeling Language (UML) for describing the main functionalities of our application. For the implementation phase, we utilized various tools including MongoDB as our database management system, Node.js and Express.js for building our backend RESTful APIs, React.js for building our user interfaces, Socket.IO for real-time communication between the client and the server, and several programming languages such as HTML, CSS, and JavaScript.

Keywords: Web Application, MongoDB, UML, UP, HTML, CSS, Java Script, Node.js Express.js, Socket.IO, RESTful APIs, Manage Trainees, Task Submission, Task Validation, Chat functionality, Manage Task

Résumé

Une Application Web est une application hébergée sur un serveur et accessible via un navigateur web. Cette technologie permet à l'entreprise, à ses filiales et aux utilisateurs mobiles d'accéder facilement à l'application. Dans notre travail, nous avons développé une Application Web pour gérer les stagiaires et apprentis à l'EPB. Le projet comprend des sections pour la spécification des exigences, l'analyse et la conception, et se termine par la mise en œuvre de l'application finale.

Nous avons choisi la méthodologie du Processus Unifié (UP) basée sur le Langage de Modélisation Unifié (UML) pour décrire les principales fonctionnalités de notre application. Pour la phase de mise en œuvre, nous avons utilisé divers outils, notamment MongoDB comme système de gestion de base de données, Node.js et Express.js pour construire nos API RESTful backend, React.js pour construire nos interfaces utilisateur, Socket. IO pour la communication en temps réel entre le client et le serveur, et plusieurs langages de programmation tels que HTML, CSS et JavaScript.

Mots-clés : Application Web, MongoDB, UML, UP, HTML, CSS, JavaScript, Node.js, Express.js, Socket.IO, API RESTful , Gérer les stagiaires, Soumission de tâches, Validation des tâches, Fonctionnalité de discussion, Gérer les tâches