Se former autrement

2022 - 2023

GRADUATION PROJECT

NATIONAL ENGINEERING DEGREE

SPECIALTY :INFORMATION TECHNOLOGY

Design and development of a business opportunity management application

By: Mohamed Dhia Zoghlami

Academic supervisor: Mr. Lassaad Saidani

Corporate Internship Supervisor: Mr. Mabrouk Zhili

Je valide le dépôt du rapport PFE relatif à l’étudiant nommé ci-dessous / I validate the submission of the student’s report:

- Nom & Prénom /Name & Surname : …………………………………………..

Encadrant Entreprise/ Business site Supervisor

- Nom & Prénom /Name & Surname : …………………………………………..

Cachet & Signature / Stamp & Signature

Encadrant Académique/Academic Supervisor

- Nom & Prénom /Name & Surname : …………………………………………..

Signature / Signature

Ce formulaire doit être rempli, signé et **scanné**/This form must be completed, signed and **scanned**.

Ce formulaire doit être introduit après la page de garde/ This form must be inserted after the cover page.

Acknowledgment

No matter the words chosen, I will never be able to express to them my sincere love. To the Eternal, **Allah**, the Almighty for having helped me reach the end of my studies, for giving me the strength to survive, as well as the courage to overcome all difficulties.

To My Dear Parents, no dedication could express my respect, eternal love, and consideration for the sacrifices you have made for my education and well-being. I thank you for all the support and the love you have shown me since my childhood, and I hope that your blessings always accompany me. May this modest work be the fulfillment of your many expressed wishes, the result of your countless sacrifices, although I can never repay you enough. May **Allah**, the Most High, grant you health, happiness, and a long life and ensure that I never disappoint you.

To all my Friends, thank you immensely for your help, your time, your encouragement, your assistance, and support.

To all my teachers, I dedicate this work. To all those who have contributed directly or indirectly to the completion of this work, may **Allah** grant you health and prosperity.

It is with great pleasure that I reserve this page as a symbol of gratitude and deep appreciation for all those who have contributed to the completion of this work. This experience has taught me a lot, especially how to manage stress during challenging times and how to persevere regardless of the situation.

First and foremost, I would like to express my heartfelt thanks to the management of the company **'Progress Engineering'** for giving me this wonderful opportunity to undertake my final internship with them.

I would also like to thank **Mr. Mabrouk Zhili** and **Mr. Lassaad Saidani**, my supervisors, for their valuable advice and availability. I have been well guided and have always been able to rely on my professor, who never ceased to encourage me and have confidence in me.

I cannot let this opportunity pass without expressing my deep gratitude to the entire educational and administrative team at **ESPRIT** for all the efforts they have made to provide us with quality education.

Lastly, I extend my most sincere thanks to the members of the jury for agreeing to evaluate this work. I hope that they find in it the qualities of clarity and synthesis they are expecting.

List of Contents

List of Figures

General introduction

In recent times, the landscape for IT companies has undergone a notable shift, marked by increasing challenges in securing new projects and clients, as well as the struggle to retain their existing customer base. The rapid evolution of technology, coupled with a highly competitive market, has led to a proliferation of IT service providers, intensifying the battle for limited opportunities. Clients now demand not only exceptional technical expertise but also a keen understanding of their specific industry needs and challenges. This necessitates a constant investment in upskilling employees and staying at the forefront of emerging technologies, which can strain resources. Moreover, as project scopes expand and become more complex, maintaining effective communication, meeting deadlines, and delivering superior quality have become paramount, further testing the capacities of IT companies. Amidst this, retaining existing customers has become equally daunting, with loyalty hinging on consistently surpassing expectations and offering ongoing value. As a result, IT companies find themselves in an intricate dance of innovation, adaptability, and customer-centricity to navigate this arduous landscape.

In this End-Of-Studies project, I will try to explain how *Progress Engineering* is responding to these changes and how we have developed a software that addresses all of these challenges.

The report is structured as follows: At first, I defined the project scope from presenting the hosting company to the problem statement, our proposed solution and its goals as well as laying out the working methodology. Throughout the second chapter, I specified the functional and non-functional requirements alongside identifying the main actors and refining the product global use case. After that, in the 3rd chapter, I wrote about the technical environment (Runtime environment, Frameworks, libraries, tools etc...) that we used during the internship to develop the application. Following that, in the 4th chapter, I defined the software architecture, the Database design and gave some use case scenarios. Finally, the 5th chapter is dedicated to showcase the project from development to production and a realization of the final application.

Chapter 1: Project Scope

This chapter will be dedicated to provide information about the hosting company, we will also be stating the project context by defining the problem, a study of the existing, criticism of the existing, our proposed solution and finally we will be ending the chapter by talking about the work methodology.

1.1-Hosting Company:

Progress Engineering is an innovative High-tech company in the IT sector, it is offering a variety of development and consultancy solutions focusing on the e-services and secured intranet. It was founded in 2000 by a highly qualified team of engineers acting in the field of software development and telecommunications, this company has acquired expertise, skills, and experience that enable it to successfully carry out all IT projects, from specification to maintenance. Its main mission is to be the World leader in thought and technology in e-services and consulting. They an attentive partner, flexible in tuning the established engagement methods and helping each customer to make the most out of the partnership with low cost and error-free services.
They also provide an integrated and highly flexible mix of on site, off site, near shore and offshore services Worldwide.

Figure 1:Progress Engineering Logo

In this section we will be talking about the project context, the description of the problem and the end goal of our project.

1.2-Project Context:

The project is entitled <u>“Design and development of a business opportunity management application”</u>.
It concludes the courses of the third year of Software Engineering, specifically **TWIN** (**T**echnologies of **W**eb and **I**nformation), at **ESPRIT** School of Engineering as a graduation project.
I was led through it to make a professional experience in **Progress Engineering** for the position of "Full Stack Software Engineer”, from March to September 2023.

1.3-Problem Statement:

In today's rapidly evolving business landscape, customers stand as the cornerstone of revenue generation, holding a pivotal role in shaping a company's success. However, this vital revenue stream is encountering new challenges due to the escalating competition fueled by the integration of cutting-edge technologies. As businesses harness the power of technological advancements, they find themselves operating in an era of heightened customer expectations and unprecedented options.

The traditional paradigm of customer-business interactions has undergone a seismic shift. Empowered by digital transformation, customers are now equipped with an array of choices that span geographical boundaries and industry sectors. This proliferation of options has elevated the competition to a level where companies must vie not only with local competitors but also with global counterparts, intensifying the battle for customer loyalty and market share.

In parallel, the dynamics of customer expectations have evolved drastically. Customers, armed with the ability to compare products, services, and experiences instantly, are increasingly discerning in their demands. They expect businesses to be not just responsive, but hyper-responsive, providing seamless interactions, tailored solutions, and rapid issue resolution.

This demand for personalized and instantaneous service has put immense pressure on companies to reengineer their customer engagement strategies and elevate their customer service capabilities to new heights. As a result, businesses find themselves in a dual challenge: they must not only stay ahead of the technological curve to remain competitive but also recalibrate their approach to customer engagement to meet the heightened expectations of the modern consumer.

A misstep in either direction can lead to loss of market share, erosion of customer trust, and ultimately, diminished revenue. Under the pressure of situations like this, how do we ensure the continuity of business and the winning of new projects?

1.4-Existing products:

1.4.1-Salesforce:

Salesforce is one of the most well-known and widely used CRM platforms. It offers a comprehensive suite of tools for managing sales, customer service, marketing, and more. Salesforce allows businesses to centralize customer information, track interactions, manage leads and opportunities, automate tasks, and provide a holistic view of the customer journey.

1.4.2-HubSpot CRM:

HubSpot CRM is a robust CRM solution designed to help businesses streamline their sales, marketing, and customer service efforts. It offers tools for managing contacts, tracking deals, automating workflows, creating marketing campaigns, and providing customer support.

1.5-Criticism of the existing:

1.5.1-Salesforce:

Complexity and Learning Curve: Salesforce's extensive feature set and customization options can lead to a steep learning curve for new users. Implementation and configuration might require significant time and resources, especially for smaller businesses without dedicated IT teams.

Cost: Salesforce's pricing can be considered high, particularly for smaller businesses. While they offer various plans, the more advanced features often come at a premium, making it a substantial investment for some companies.

1.5.2-HubSpot CRM:

Limited Advanced Functionality: While HubSpot CRM is excellent for small to medium-sized businesses, larger enterprises might find that it lacks some of the more advanced features and scalability they require.

Pricing Tiers: some advanced features are locked behind higher-priced tiers, which might not be cost-effective for all businesses especially smaller ones.

1.6-Proposed solutions:

A comprehensive web application to streamline the end-to-end process of managing business opportunities and contracts. The solution should empower the company to efficiently store and manage customer data, make informed decisions on participation in consultations or bids, and navigate the entire lifecycle from opportunity identification to contract realization.
The application should serve as a collaborative platform for IT agents, commercial agents, and administration members to seamlessly share crucial project information. It should enable the tracking of opportunity statuses, evaluation of agent performance through scores and statistics, and integration of an email notification system to ensure prompt updates on offers. The ultimate goal is to enhance operational efficiency, improve decision-making, and foster effective communication among different stakeholders within the company.
This solution is adequate for small, medium, and large businesses.

We could summarize our solution by referring to the 5W1H problem-solving method (see figure below):

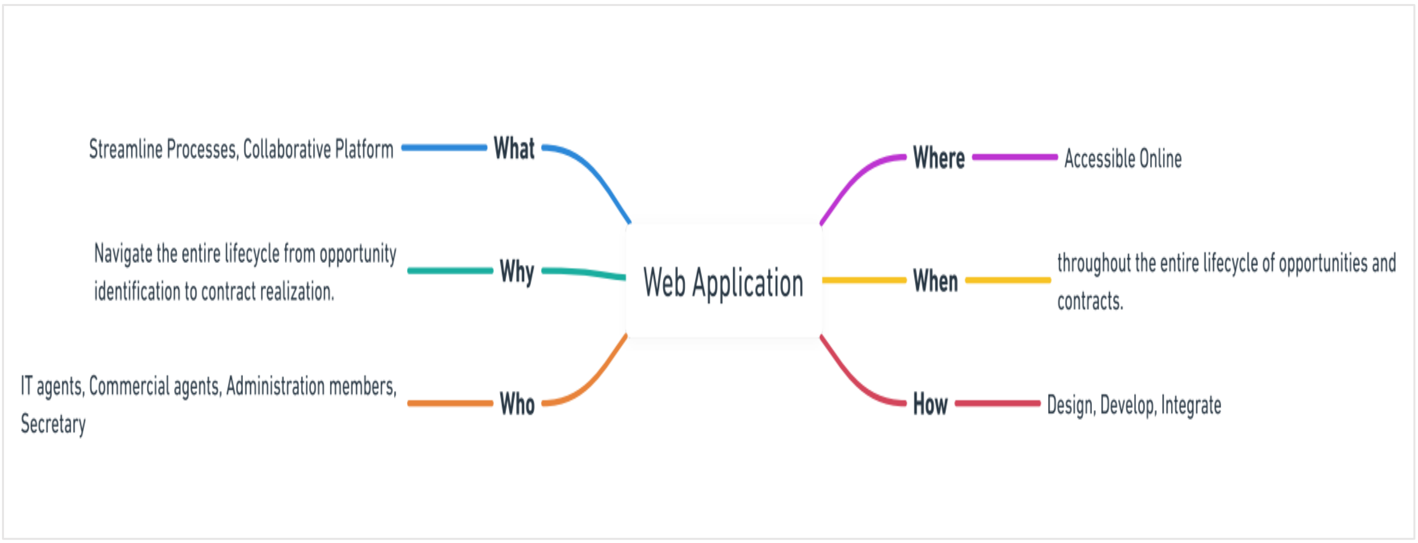

Streamline Processes, Collaborative Platform
What
Navigate the entire lifecycle from opportunity identification to contract realization.
Why
IT agents, Commercial agents, Administration members, Secretary
Who
Web Application
Where
Accessible Online
When
throughout the entire lifecycle of opportunities and contracts.
How
Design, Develop, Integrate

Figure 25W1H Method of our solution

1.7-Work Methodology:

1.7.1-Agile Scrum:

What is agile?

Agile is a process that allows a team to manage a project more efficiently by breaking it down into several stages, each of which allows for consistent collaboration with stakeholders to promote steady improvements at every stage. [R1]

What is scrum?

In short, scrum is a framework for effective collaborations among teams working on complex products. Scrum is a type of agile technology that consists of meetings, roles, and tools to help teams working on complex projects collaborate and better structure and manage their workload. Although it is most often used by software development teams, scrum can be beneficial to any team working toward a common goal. [R1]

What is agile scrum?

Agile scrum methodology is a project management system that relies on incremental development. Each iteration consists of two- to four-week sprints, where the goal of each sprint is to build the most important features first and come out with a potentially deliverable product. More features are built into the product in subsequent sprints and are adjusted based on stakeholder and customer feedback between sprints. [R1]

In our case, we had 12 sprints over 6 months, each sprint lasts for 2 weeks, and we combined each 4 sprints into a release so overall we have 3 main releases.

The methodology we adapted is as follows:

- Daily meeting:
  - What I have been working on?
  - What will I work on today?
  - What is blocking me?

- Weekly meeting:
  - The made Progress.
  - Re-order priorities

- Sprint review:
  - Every 2 weeks we represent the work that we accomplished during the sprint.

- Release:
  - Every 2 months (4 sprints) we deliver the complete work that we have done during the release.

1.7.2-Software Development Life Cycle:

The Software Development Life Cycle (**SDLC**) is the series of steps an organization follows to develop and deploy its software. There isn't a single, unified software development lifecycle. Rather, there are several frameworks and models that development teams follow to create, test, deploy, and maintain software.

Planning:

This is the first phase, and it was dedicated to define the project scope, goals, and objectives in collaboration with stakeholders, gather and document detailed requirements, including features, functionalities, and user roles, create a project plan outlining tasks, responsibilities, and timelines.

Code & Build:

This is the second phase of the process which we as developers do our main job and that is design the global architecture and the database architecture, develop the necessary frontend and backend functionalities, implement business logic.

Test:

Testing is an essential part of any software development lifecycle. In addition to security testing, performance tests, unit tests, and non-functional testing such as interface testing all take place in this phase.

Deployment:
This phase is dedicated to the deployment of the application, this is where we introduced DevOps into our project by implementing a continuous integration/continuous delivery (CI/CD) with the use of Infrastructure as Code (IaC) tools to define and manage the application's infrastructure, making deployment consistent and repeatable.

Conclusion:

At this point, we've gained a comprehensive grasp of the project's overall scope and its alignment with *Progress Engineering*'s mission to face the above challenges. In the upcoming sections, our focus will be on establishing the fundamental principles that we will build upon our application.

Chapter 2: Requirements and Specifications

The requirements analysis and specification phase of a project's development cycle is crucial. As a result, we will be introducing our main actors in this chapter. The definition of our application's functional and non-functional needs follows. Finally, in order to provide a simplified graphical representation of the behavior of our system, we will build the global use case diagram.

2.1-Actors Identification:

An "actor" in the context of a web application usually refers to any entity that engages with or makes use of the application. These entities could be people, machines, or other external elements that operate the application.
Understanding the various roles that actors play in a web application's operation helps in creating and putting the application's functionality and security measures into place.
For our application we can identify 4 total actors as follows:

- Admin: This entity has a full access to everything of the app, it mainly refers to administrative members who will have the final decision making of every operation.
- Secretary: Employee of the company and his/her main job is to add necessary documents as needed.
- Commercial Agent: Employee of the company that will be working on every opportunity assigned to him/her.
- IT Agent: Employee of the company that will be working on every project or task assigned to him/her.

2.2-Functional Requirements:

Since we have 4 actors and each has its own role, we have 4 dashboards, so we decided to divide the functional requirements into 5 parts: 4 of them depends on the view of each actor and the final one is system functional requirements.

- **Admin space management:** Being an admin, you have the full control of the application:

☞ -Full access to user management.

- ☞ -Full access to customer management.
- ☞ -Full access to contact management.
- ☞ -Full access to opportunities management.
- ☞ -Full access to project management.
- ☞ -Full access to task management.
- ☞ -Full access to bill management.
- ☞ -Full access to contract management.
- ☞ -Full access to offer management.
- ☞ -Assign a commercial agent to an opportunity.
- ☞ -Assign a team to a project.
- ☞ -Assign an IT agent to a specific task.
- ☞ -Ability to view commercial agent statistics.
- ☞ -Ability to view commercial agent scores.
- ☞ -Ability to view the opportunity lifecycle from A to Z.
- ☞ -Ability to view full reports (Money gained, Customers gained/lost, percentage of opportunities won, etc...)

- ▪ **<u>Secretary space management:</u>**
- ☞ -Full access to customer management.
- ☞ -Full access to contact management.
- ☞ -Ability to add/read opportunities.
- ☞ -Ability to change user profile.

- ▪ **<u>Commercial agent space management:</u>**
  This actor is mainly created to handle opportunity assigned to him/her and make an adequate offer, therefore:
- ☞ -Update an opportunity status from "Assigned" to "Working On" then to "Negotiating".
- ☞ -Update an opportunity status from "Negotiating" to "Closed won" or "Closed lost".
- ☞ -Create an offer related to an opportunity assigned to him/her.
- ☞ -Submit an offer.
- ☞ -Update an offer from "On going" to "Accepted" or "Rejected".
- ☞ -Delete offers made by him/her that are not submitted.
- ☞ -Ability to change user profile.

- ▪ **<u>IT Agent space management:</u>**
  After winning an opportunity, it becomes a project so this actor is handling the technical part of the project as follows:
- ☞ -Update status of tasks from "Assigned" to "Working on" to "Test" then finally to "done".
- ☞ -Create feedback (if missing information about his/her task) if necessary.

- <u>**System functional requirements:**</u>
- ☞ -Authentication and authorization.
- ☞ -An upload/download file system.
- ☞ -Emailing service.
- ☞ -Reminder service for each related actor.

2.3-Non-functional requirements:

- **Ergonomic:** The application has an intuitive and user-friendly interface that is easy to navigate and use. Security: We implemented authentication and authorization mechanisms to ensure that only authorized users have access to specific features and data.

- **Reliability:** Using some error handling techniques and through consistent testing and monitoring we ensured that our application is reliable.

- **Scalability:** The application is designed to accommodate both vertical scaling (adding more resources to a single server) and horizontal scaling (distributing load across multiple servers) so it will always be scalable. Availability: We ensured that the application is highly available by adding load balancers and using cloud services to deploy the application.

2.4-Use Case:

Use case diagrams are a valuable tool in system modeling and design, particularly when there are multiple actors involved. In this scenario, with four distinct actors - admin, secretary, technical agent, and commercial agent - use case diagrams help illustrate the interactions and functionalities within a system as the next figure shows:

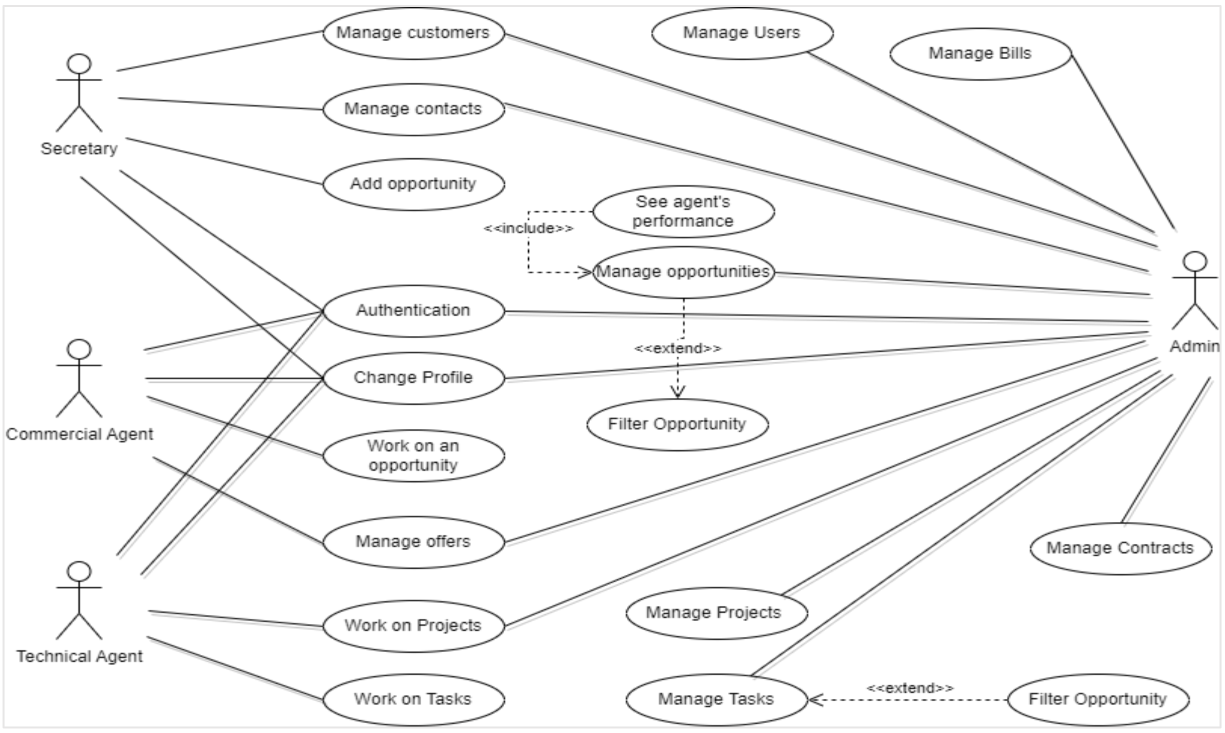


Figure 3: Global use Case Diagram

Conclusion:

By analyzing our project objectives and business specifications, we took the first step in clarifying the large modules that our application will tackle in this chapter. The technical environment in which we will develop this application is described in the following chapter.

Chapter 3: Technical environment

After defining the requirements and specifications of our project, we will dedicate this chapter to talk about the different technologies that we have used to build our solution, we used a lot of technologies to achieve our end goal and that’s why we will be dividing them into parts, Frontend, Backend, Database, DevOps so we have a clearer way for every technology used and where and why we used it.

3.1-Technologies:

3.1.1-FrontEnd technologies:

ReactJS:

Figure 4: React Logo

This is an open-source JavaScript library used for building user interfaces for web applications. It was developed by Facebook and later open-sourced, gaining widespread adoption in the web development community. React allows developers to create reusable UI components and manage the dynamic rendering of those components based on changes in application state.

- **Advantages**:
  - ☞ **Component-Based Architecture:** React's component-based architecture allows developers to break down complex user interfaces into smaller, reusable components. This modularity makes it easier to manage and maintain code, promotes reusability, and facilitates collaboration among developers working on different parts of an application.

  - ☞ **Virtual DOM and Performance Optimization:** React's use of a virtual DOM (Document Object Model) provides a significant performance boost.

  - ☞ **Strong Ecosystem and Community Support:** React has a large and active community of developers, which means there are numerous resources, tutorials, libraries, and tools available to help developers learn and build with React.

Material UI:

Figure 5: Material UI Logo

It is a popular open-source UI framework for building user interfaces in React applications. It provides a set of reusable components and styles based on Google's Material Design principles. Material Design is a design language developed by Google that emphasizes a clean and modern aesthetic, along with a focus on usability and consistency across platforms.

3.1.2-Backend Technologies:

Springboot:

Figure 6: SpringBoot Logo

Java Spring Boot (Spring Boot) is a tool that makes developing web application and microservices with Spring Framework faster and easier through **three** core capabilities:

- Autoconfiguration.
- An opinionated approach to configuration.
- The ability to create standalone applications.

These features work together to provide you with a tool that allows you to set up a Spring-based application with minimal configuration and setup. [R2]

Spring security:

Figure 7: SpringBoot Logo

It is a framework that provides authentication, authorization, and protection against common attacks. With first class support for securing both imperative and reactive applications, it is the de-facto standard for securing Spring-based applications. [R3]

3.1.3-Database:

MySQL:

Figure 8: MySQL Logo

It is an open-source relational database management system (RDBMS) that is widely used for storing and managing structured data. It is one of the most popular and commonly used database systems in the world.

3.1.4-Versioning:

Git:

Figure 9: Git Logo

It's an open-source distributed version control system (VCS) that is widely used for tracking changes in source code during software development. It was initially developed by Linus Torvalds in 2005 and has since become one of the most popular and essential tools in the field of software development.

GitHub:

Figure 10: GitHub Logo

This platform is a web-based platform and service that provides tools for version control and collaborative software development using the Git version control system. It offers a wide range of features to help developers and teams manage, collaborate on, and track changes to their codebase.

3.1.5-Cloud Services:

AWS CodeBuild:

Figure 11: AWS CodeBuild Logo

It is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy. AWS CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left in the queue. [R4]

AWS CodePipeline:

Figure 12: AWS CodePipeline Logo

It helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously. You can integrate AWS CodePipeline with third-party services like GitHub or use an AWS services such as AWS CodeCommit or Amazon ECR. [R4]

Amazon Elastic Container Registry (Amazon ECR):

Figure 13: AWS ECR Logo

It is a fully managed registry that makes it easy for developers to store, manage, and deploy Docker container images. Amazon ECR is integrated with Amazon ECS to simplify your development-to-production workflow. Amazon ECR hosts your images in a highly available and scalable architecture so you can deploy containers for your applications reliably. Integration with AWS Identity and Access Management (IAM) provides resource-level control of each repository. [R4]

Amazon Elastic Container Service (Amazon ECS):

Figure 14: AWS ECS Logo

It is a highly scalable, high-performance container orchestration service that supports Docker containers and allows you to easily run and scale containerized applications on AWS. Amazon ECS eliminates the need for you to install and operate your own container orchestration software, manage and scale a cluster of virtual machines, or schedule containers on those virtual machines. [R4]

AWS Fargate:

Figure 15: AWS Fargate

It's a compute engine for Amazon ECS that allows you to run containers without having to manage servers or clusters. With AWS Fargate, you no longer have to provision, configure, and scale clusters of virtual machines to run containers. This removes the need to choose server types, decide when to scale your clusters, or optimize cluster packing. [R4]

Amazon Relational Database Service (Amazon RDS):

Figure 16: AWS RDS Logo

It is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks. Amazon Aurora is a fully managed relational database engine that's built for the cloud and compatible with MySQL and PostgreSQL. Amazon Aurora is part of Amazon RDS. [R4]

Amazon S3:

Figure 17: Amazon S3 Logo

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements. [R4]

Amazon CloudFront:

Figure 18: AWS CloudFront Logo

Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users. CloudFront delivers your content through a worldwide network of data centers called edge locations. When a user requests content that you're serving with CloudFront, the request is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance. [R4]

Amazon Elastic Load Balancer (ELB):

Figure 19: AWS ELB Logo

Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones. It monitors the health of its registered targets, and routes traffic only to the healthy targets. Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic. [R4]

3.1.6-Other Technologies:

Docker:

Figure 20: Docker Logo

Docker is an open-source platform that enables programmers to automatically deploy, scale, and manage applications inside of isolated, lightweight containers. Containers, a type of virtualization, offer a consistent and portable environment for running software across many settings, including development, testing, and production. They package an application and its dependencies together.

CDK:

Figure 21: CDK Logo

This stands for "Cloud Development Kit." It is an open-source software development framework that allows developers to define cloud infrastructure in familiar programming languages, such as TypeScript, Python, Java, and others. The CDK simplifies the process of provisioning and managing cloud resources by providing a higher-level abstraction over cloud providers' native infrastructure-as-code tools.

3.2-Software environment:

3.2.1-Visual Studio Code:

Figure 22: VS Code Logo

Visual Studio Code (VS Code) is a lightweight yet powerful source code editor created by Microsoft, widely used by developers for its flexibility and extensive extension library. It provides a streamlined coding experience with a user-friendly interface, making it a top choice for coding tasks across various programming languages and platforms.

3.2.2-IntelliJ IDEA:

Figure 23:IntelliJ IDEA Logo

IntelliJ IDEA is a renowned integrated development environment (IDE) crafted by JetBrains, favored for its exceptional Java development capabilities. It boasts smart code assistance, strong debugging tools, and excellent support for version control. IntelliJ IDEA's popularity stems from its productivity-enhancing features and widespread use in the Java community.

3.3.3-NodeJs:

Figure 24: NodeJs Logo

Node.js is a cross-platform, open-source server environment that can run on Windows, Linux, Unix, macOS, and more. Node.js is a back-end JavaScript runtime environment, runs on the V8 JavaScript engine, and executes JavaScript code outside a web browser. [R6]

3.3.4-Maven:

Figure 25: Maven Logo

Maven is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. The Maven project is hosted by The Apache Software Foundation, where it was formerly part of the Jakarta Project. [R7]

Conclusion:

In this chapter we talked and we defined the different technologies we have used to achieve our goals, we also ended up talking and defining about some software environments that we installed to develop the application. In the next one we will dive deep into the design, architecture, and the implementation.

Chapter 4: Design and Architecture

After talking about the functional requirements, non-functional requirements, the actors, technical environment that we have used, we will be dedicating this chapter to dive deeper into the design and architecture of the whole application. We will start by talking about the software architecture in general then the database modeling, after that we will talk about and explain some concepts we have used to achieve our end goal solution and finally we will be finishing by walking through some different scenarios.

4.1-Software architecture:

The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.

4.1.1-Logical architecture:

The logical architecture is defined as the organization of the subsystems, software classes, and layers that make the complete logical system. In our case, we decided to go with the multi-tier (n-tier) architecture specifically three-tier architecture, and we will be explaining why and we will talk about its benefits.

The three-tier architecture is the most popular implementation of a multi-tier architecture and consists of a single presentation tier, logic tier, and data tier. The following illustration shows a high level on how to implement physical architecture of our generic three-tier application.

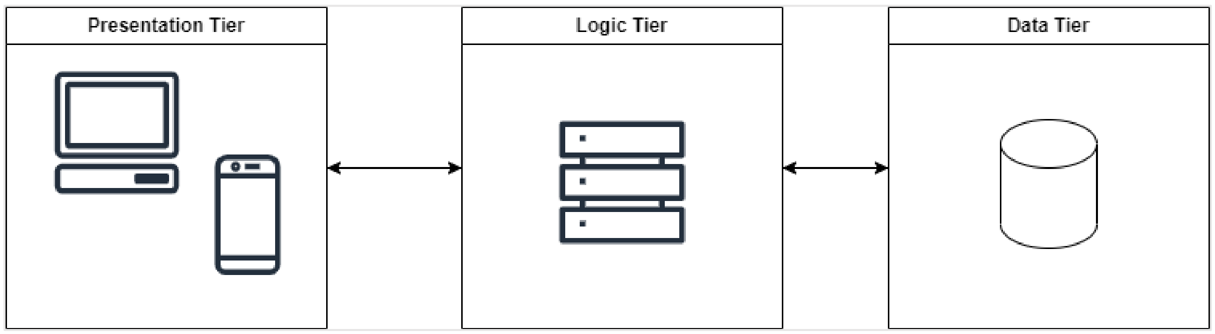


Figure 26: High Level Physical Architecture

The chief benefit of three-tier architecture is that because each tier runs on its own infrastructure, each tier can be developed simultaneously by a separate development team and can be updated or scaled as needed without impacting the other tiers.

Adding to the faster development and improved scalability, the three-tier architecture provides an improved reliability, an outage in one tier is less likely to impact the availability or performance of the other tiers.

Finally, the three-tier architecture provides a huge improvement on the security of the whole application, because the presentation tier and data tier can't communicate directly, a well-designed application tier can function as a sort of internal firewall, preventing SQL injections and other malicious exploits.

Presentation Layer:

The presentation layer is the user interface and communication layer of the application, where the end user interacts with the application. Its main purpose is to display information to and collect information from the user. This top-level tier can run on a web browser, as desktop application, or a graphical user interface (GUI), for our application, the web presentation tiers are developed using ReactJs and Material UI.

Logic Tier:

The logic tier, also known as the application tier or middle tier, is the heart of the application. In this tier, information collected in the presentation tier is processed - sometimes against other information in the data tier - using business logic, a specific set of business rules. The application tier can also add, delete or modify data in the data tier. In our case we used Java and Springboot to develop this tier and build our application logic.

Data Tier:

The data tier, sometimes called database tier, data access tier or back-end, is where the information processed by the application is stored and managed. We have used MySQL as a database to store and manage all the datas that we need.

In a three-tier application, all communication goes through the logic tier. The presentation tier and the data tier cannot communicate directly with one another. So, through what do these tiers communicate?

The presentation tier communicates with the logic tier by sending requests in the form of HTTP(S), The application tier processes these requests, executes the necessary business logic, and generates a response. The response is then sent back to the presentation tier, which renders it for the user. We also need to add that data between these tiers are in the format of JSON following the REST API architectural style.

The logic tier communicates with the data tier to retrieve or update data required for the application's functionality. This communication is typically done using database queries or data access libraries. The logic tier sends queries to the data tier to fetch specific data or to perform data updates.

- REST API: REST is an architectural style for designing networked applications and APIs. RESTful APIs use HTTP methods (such as GET, POST, PUT, DELETE) to perform operations on resources (e.g., data objects) identified by URLs (Uniform Resource Locators). They are widely used for building web services and are known for their simplicity and scalability.
- HTTPS: Modern users expect a secure and private online experience when using a website. HTTPS is a secure version of the HTTP protocol, which is used for transferring data over the internet.
- JSON: JSON is a lightweight data interchange format that is easy for both humans to read and write and machines to parse and generate. It is a common format for data

exchange in RESTful APIs, as it is easy to work with in various programming languages.

4.1.2-Physical architecture:

Cloud architecture plays a critical role in modern IT strategies, providing the flexibility, scalability, and cost-efficiency needed to meet the demands of businesses and organizations of all sizes. It continues to evolve with new services and technologies, making it an exciting field within the broader realm of cloud computing. Refers to the design and structure of cloud computing environments. Cloud computing has become a fundamental technology in the modern digital landscape, enabling organizations to access and use computing resources (such as servers, storage, databases, networking, software, and analytics) over the internet on a pay-as-you-go basis. The figure below shows our end solution of the cloud architecture of our application.

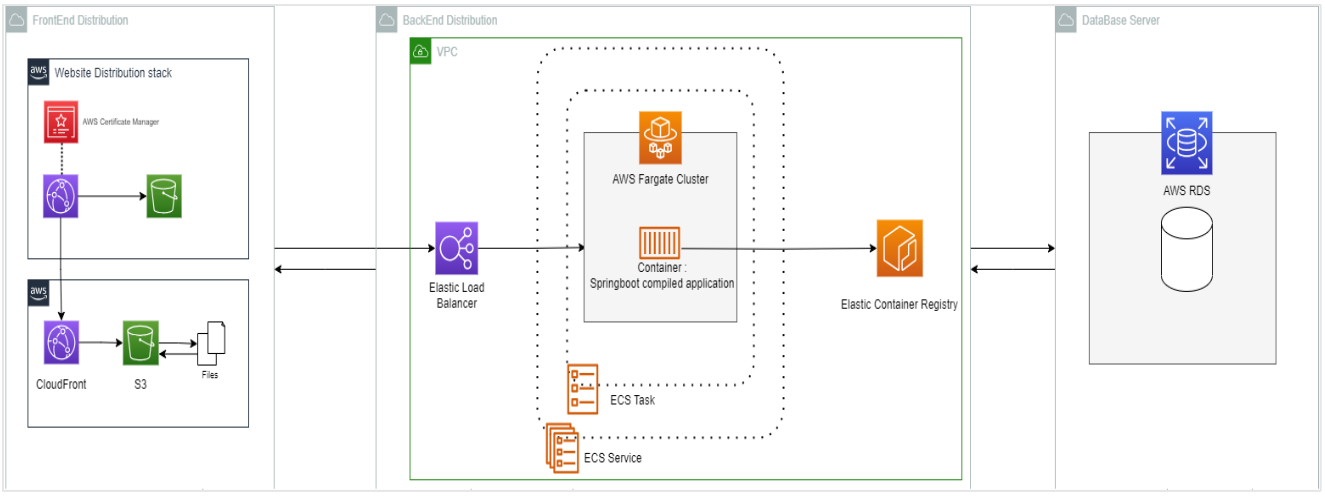


Figure 27: phyical architecture

Frontend Distribution:

To efficiently host and distribute a React application while enabling direct file uploads from the client side to Amazon S3, an AWS architecture was implemented. The frontend distribution stack comprises two key components. Firstly, the React app is deployed on an AWS S3 bucket, allowing for the hosting and scaling of the application. Secondly, Amazon S3 is leveraged as a reliable and scalable storage solution for directly uploading files from the client side. To ensure low-latency content delivery and enhanced security, Amazon CloudFront acts as a Content Delivery Network (CDN) in front of both the React app and the S3 bucket. This setup not only ensures a responsive and highly available frontend but also offers efficient file storage and retrieval, all while benefiting from the AWS ecosystem's robustness and scalability.

Backend Distribution:

To establish a resilient and scalable backend distribution for a Spring Boot application on AWS, a comprehensive infrastructure was orchestrated. The backend distribution stack consists of several vital components. First and foremost, the Spring Boot application is containerized and stored in Amazon Elastic Container Registry (ECR), ensuring easy deployment and version control. AWS Fargate is employed to manage and orchestrate the application containers within Amazon Elastic Container Service (ECS), ensuring automatic scaling and efficient resource utilization. A Virtual Private Cloud (VPC) is configured to isolate and secure the backend services. To distribute incoming traffic and enhance availability, an AWS Load Balancer is utilized to evenly distribute requests across multiple Fargate tasks. This setup not only provides a highly available and fault-tolerant backend infrastructure but also ensures scalability and efficient resource management, all while leveraging the power of AWS services.

Database Server:

To ensure robust and scalable data storage for the application, Amazon RDS (Relational Database Service) was chosen to host a MySQL database. Amazon RDS simplifies database management tasks such as provisioning, patching, backup, recovery, and scaling. By hosting the MySQL database on Amazon RDS, the development team can focus on application logic without the need to worry about the underlying database infrastructure. Amazon RDS also provides features like automated backups, high availability through multi-AZ deployments, and the ability to scale the database instance up or down as needed, ensuring data reliability and performance. Additionally, it integrates seamlessly with other AWS services, making it a powerful choice for managing and maintaining the application's database while benefiting from the security, scalability, and reliability of Amazon Web Services.

4.1.3-Other software implementation:

Security: Security is a paramount concern in this architecture, and it is addressed comprehensively. JSON Web Tokens (JWT) are employed for both authentication and authorization, enhancing the overall security posture. JWTs provide a secure and efficient means of verifying the identity of users and ensuring they have the necessary permissions to access specific resources. In addition to JWTs, the AWS infrastructure also contributes to security. Amazon VPC isolates the backend services, creating a secure network environment. AWS Identity and Access Management (IAM) is used to finely control access to AWS resources. Moreover, Amazon RDS implements encryption at rest and in transit, safeguarding sensitive data. Regular security updates and patches are automatically applied to the MySQL database, thanks to Amazon RDS. Altogether, the combination of JWTs and AWS services ensures robust authentication, authorization, and data security throughout the application, mitigating potential risks and protecting sensitive user information.

Reliability: The reliability of this stack is paramount, thanks to a well-orchestrated blend of AWS services and security measures. With Amazon RDS ensuring high availability and data integrity through multi-AZ deployments and automated backups, the backend and MySQL database remain consistently accessible. AWS Fargate's automatic scaling capabilities handle variable workloads seamlessly, guaranteeing responsiveness during traffic surges. With Amazon S3 storing files and Amazon CloudFront as a content delivery network, the frontend benefits from data durability and low-latency content distribution, bolstering user experience. Altogether, this architecture safeguards against disruptions, ensures data consistency, and delivers reliable performance to our users.

Scalability: This architecture's scalability ensures that our application can handle increased user demand and traffic, providing a responsive and reliable user experience even during periods of high activity. This scalability is essential for accommodating growth and ensuring our application's performance remains consistent as it attracts more users in the future.

4.2-Data Modelling:

Database modeling is a crucial step in the development of a robust and efficient database system. It involves the conceptualization and design of the database structure, which serves as the foundation for storing, organizing, and managing data. In our case of MySQL, a popular relational database management system, this modeling typically includes defining tables, their attributes, relationships between tables, and constraints to ensure data integrity. To visualize the database structure and relationships, a class diagram can be a valuable tool. A class diagram represents the entities (tables) in the database as classes, their attributes as class attributes, and the relationships between tables as associations between classes. This diagram provides a clear and concise overview of the database schema, making it easier for developers and stakeholders to understand the system's data structure and relationships, ultimately contributing to effective database design and development.

Why MySQL?

The choice of a relational database is because a lot of entities we are dealing with are connected to each other in multiple ways so we made the choice to go with MySQL as our relational database.

Why Amazon RDS?

Amazon RDS (Relational Database Service) offers automated backup and restore capabilities to help us protect and manage our relational database instances in the Amazon Web Services (AWS) cloud. Amazon RDS provides automated backups of our database instances. These backups are created automatically and retained for a specified period, allowing us to restore our database to a point in time.

By default, RDS takes a daily backup of our entire database instance during a user-defined backup window. We can choose the start time for this window to minimize the impact on our database's performance.

Class Diagram:

The class diagram presented in our database design represents a comprehensive visualization of our MySQL database structure, reflecting the critical entities and their relationships within our system. The diagram encapsulates key tables, including 'user,' 'role,' 'opportunity,' 'contract,' 'contact,' 'customer,' 'bill,' 'project,' 'task,' and 'notification,' each depicted as a class with its respective attributes. Relationships between these classes are skillfully illustrated, signifying the intricate connections between data entities. For example, we've modeled a many-to-many relationship between the 'user' and 'role' classes, denoting that users can assume multiple roles, and roles can be assigned to various users. This class diagram serves as an indispensable reference tool for us, fostering a shared understanding of our data architecture, and will be instrumental in the ongoing maintenance and optimization of our MySQL database.

The next figure represents the whole class diagram of our application and the different relationships the entities had with each other.

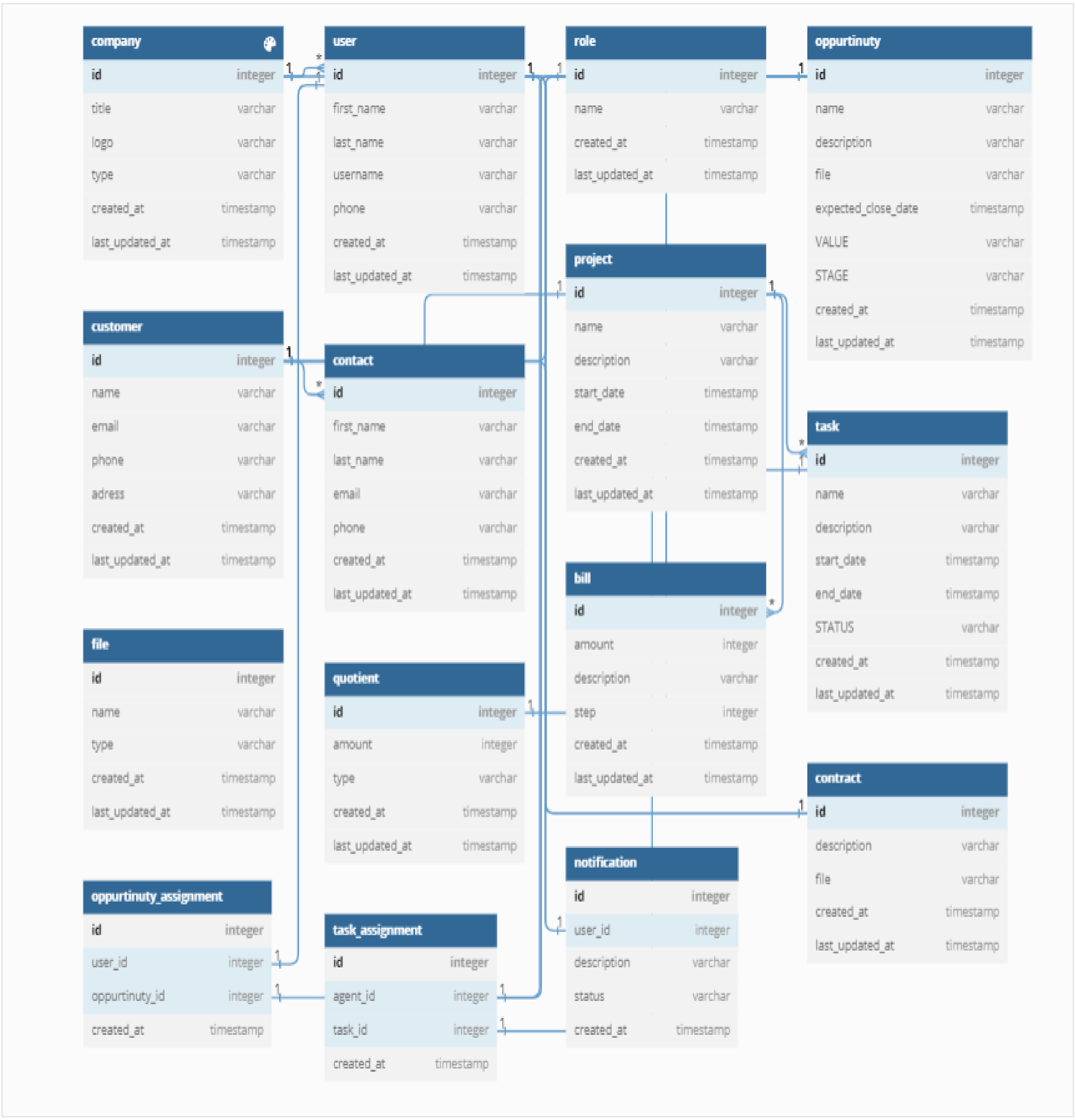


Figure 28: Class Diagram

4.3-Scenarios:

In software architecture, a scenario refers to a specific situation or use case that describes how a software system or component behaves or interacts with its environment. Scenarios are used to illustrate and analyze different aspects of a system's behavior, functionality, and performance. In our case, we will be illustrating different scenarios of opportunities life cycle, from creation till it becomes a valid project.

We will assume from now on that every actor is logged in and has access to his own dashboard.

4.3.1-Opportunity creation:

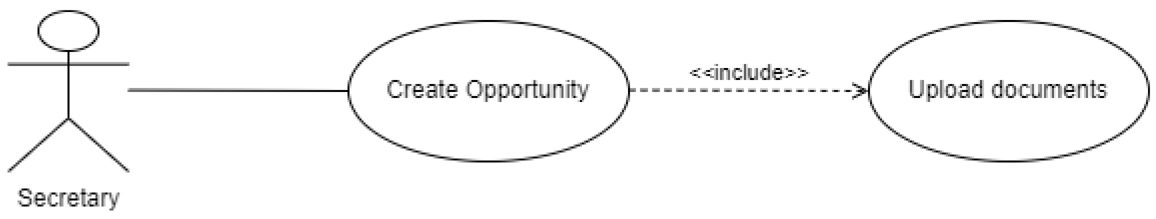


Figure 29: Use Case of opportunity creation

A secretary in our application refers to an actor that its only job is to type information from adding customers contacts and opportunities. It works like an opportunity hunter, this entity can get information from contacts directly or through social media or even through newspaper, the secretary then has to add every opportunity found to our database.

Now we will be discussing the steps to add an opportunity:

1. Upload necessary documents related to this opportunity, such as PDF files, images, bidding documents, full report about the customer or the opportunity.
2. Add different information about the customer, the expected closed date, from where did the secretary found out about this opportunity, etc. …
3. Once you finish step 1 & 2, the secretary will press the "Create Opportunity" button which will save those information into our database.
4. The emailing service will automatically send a notification to the administrators so they can decide to engage in this opportunity or just ignore it.

In the next illustration, we will delve into an in-depth technical exposition of the sequential steps involved in this scenario. By adopting a highly detailed and granular approach, we aim to provide everyone reading this with a comprehensive and nuanced understanding of the intricate workings and processes underlying this situation.

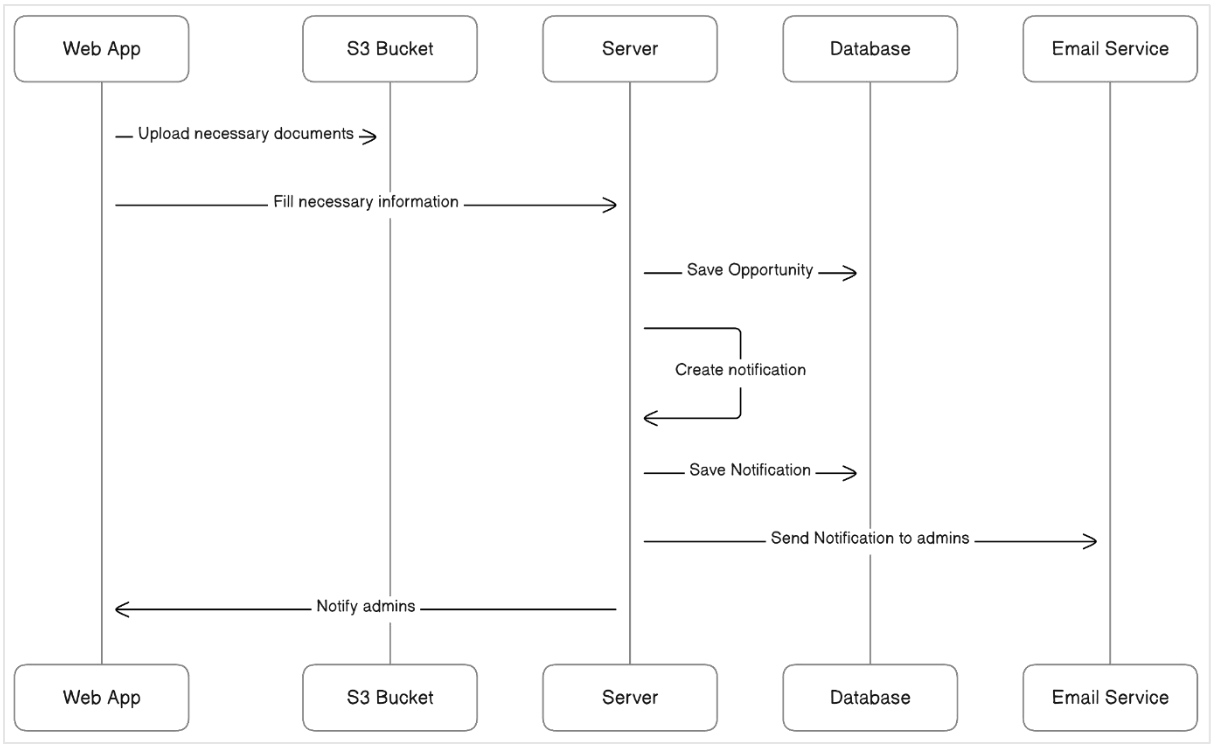


Figure 30: Sequence diagram of opportunity creation

4.3.2-Decision Making:

In this section we will be talking about the admin's interference with the opportunity life cycle We will assume from this point that the admin got already notified about the creation of a new opportunity as it shows the previous figure.

So the decision making refers to the process of admins reading all information about the newly added opportunity, they download all of documents, they check them out then decide to engage or ignore, so that leaves us with 2 main scenarios, first is to ignore the opportunity as it doesn't fit to the company's work strategies or for any other reason, the second will be to engage in that opportunity.

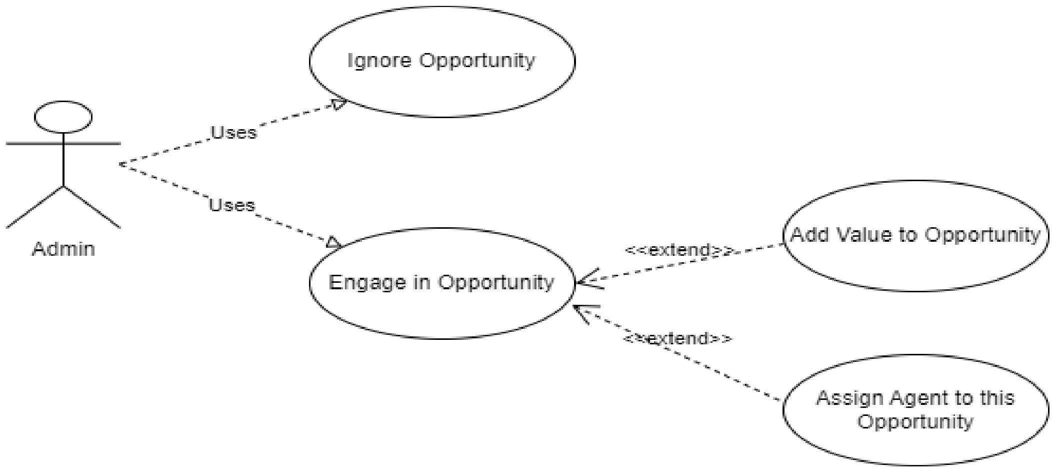


Figure 31: Use case diagram of the decision making

We won't be talking about the first scenario as it is a straight forward one , for any reasons the company has such as : lack of time , lack of commercial agents , busy schedule or even finance problems , the admins can decide to just ignore the opportunity but it will be saved in the database with an attribute of ignored.

We will be focusing in the section on deciding to engage, so that in mind we have these steps that the admin will do to interact with our application:

1. The admin will press the button of "Engage" and that will update the opportunity decision from "deciding" to "accepted".
2. The admin will give a value to this opportunity: low, medium, high depending on a lot of factors that the company works upon, and that will update the opportunity value.
3. The admin then will assign a commercial agent to that opportunity and by that the opportunity status will be updated from "accepted" to "assigned" and a notification will be created and sent to the assigned commercial agent.

The figure below shows the different technical steps for our decision-making process.

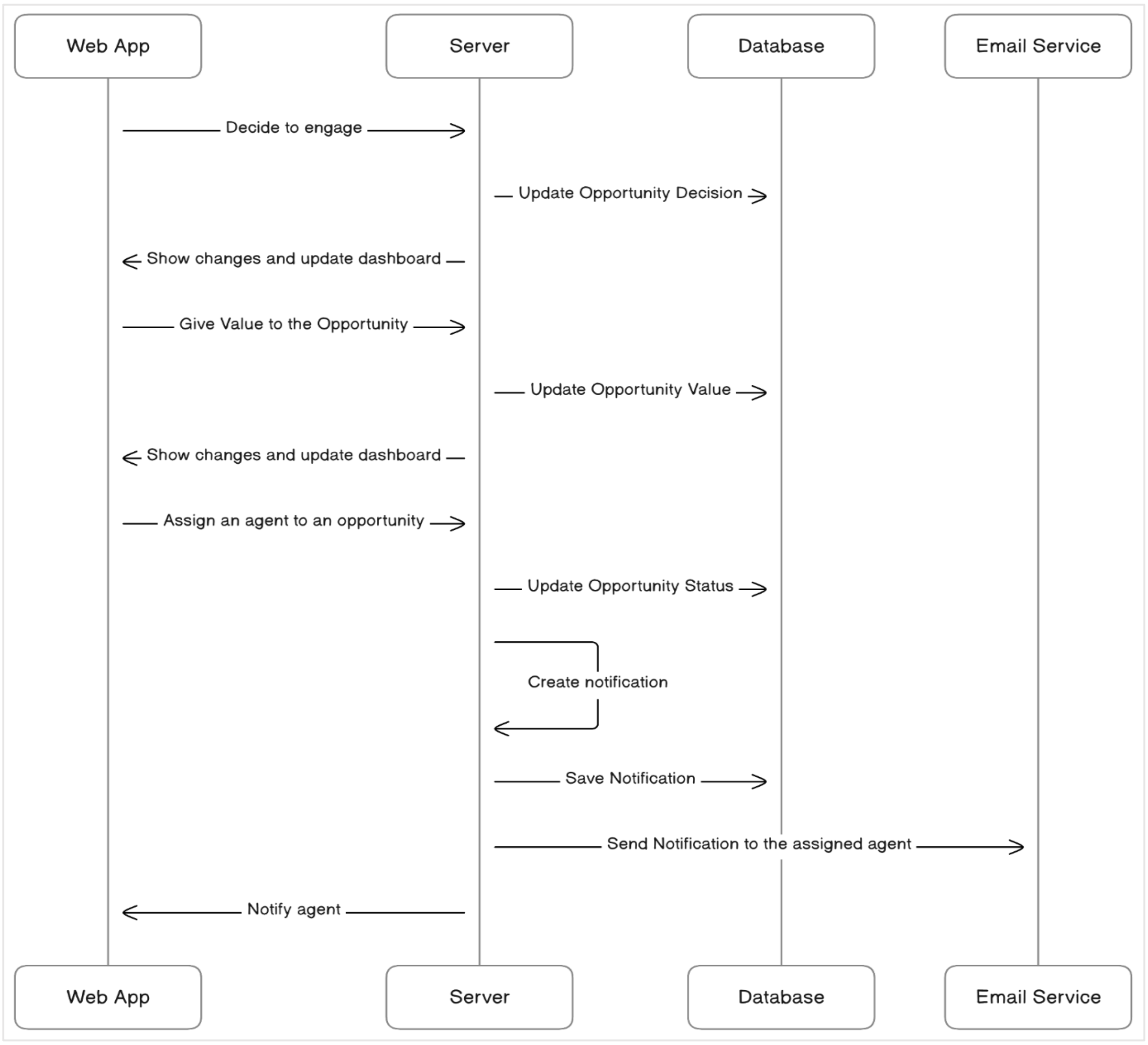

Web App
Server
Database
Email Service
Decide to engage
Update Opportunity Decision
Show changes and update dashboard
Give Value to the Opportunity
Update Opportunity Value
Show changes and update dashboard
Assign an agent to an opportunity
Update Opportunity Status
Create notification
Save Notification
Send Notification to the assigned agent
Notify agent
Web App
Server
Database
Email Service

Figure 32: Sequence diagram of decision making.

4.3.3-Commercial Agent handling the Opportunity:

As shown in the previous figure, the commercial agent got notified that he was assigned to a new opportunity, now its his/her turn to manage that opportunity and make offers directly to the specific customer, so that leaves us with multiple steps, one dedicated to updating the opportunity, one to managing offers, from creating to updating , deleting and reading, but we need to add that every commercial agent can create multiple offers for the same opportunity but he/she can only submit one , if an offer is submitted, he/she can no longer update it or delete it as it will affect his final score.

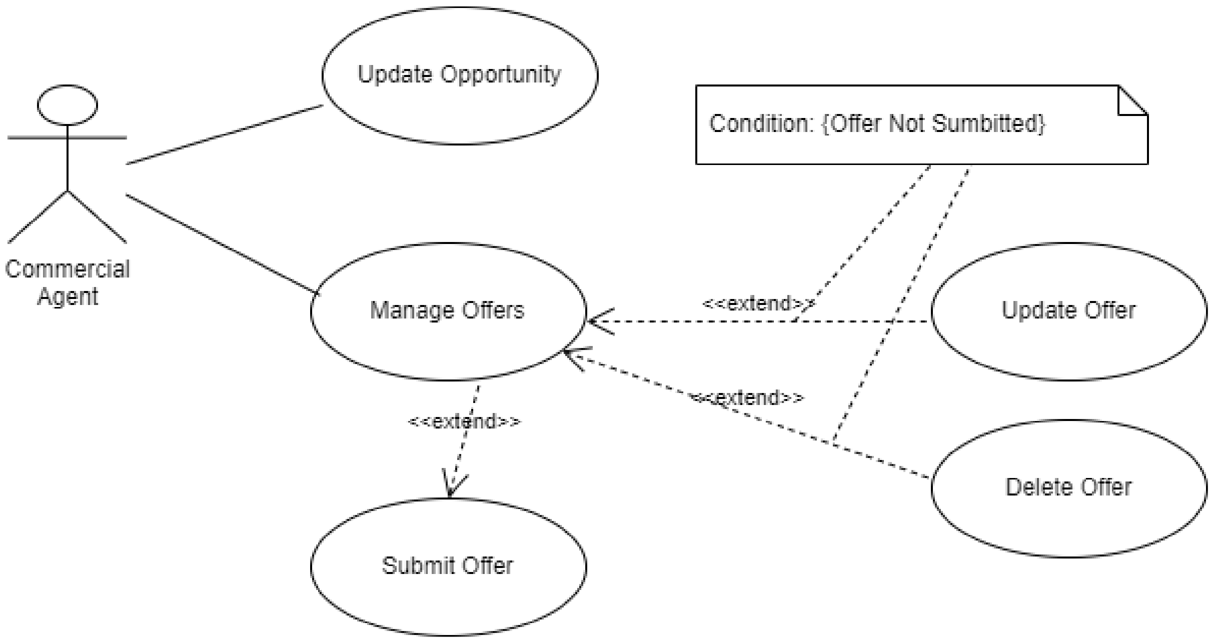


Figure 33: Use Case of Commercial Agent

For this scenario, There is a lot of steps going on under the hood and we will try to explain them one by one, at first here is the different steps a commercial agent takes to achieve his process in the opportunity life cycle, we may need to say that these steps happen after the commercial agent received his notification that he is assigned to the opportunity and after he reads all the details including necessary documents about the opportunity and the customer :

1. The agent presses "Start Working On" button and that would update the opportunity status from "Assigned" to "Working On" which will provide the admins of better following the whole process.
2. The agent now will create an offer or multiple offers related to that opportunity, he can either choose to submit one directly or talk with his supervisors about the best offer to submit, as we said he/she can only submit one offer and he/she can no longer update or delete the submitted offer. Once submitted the status of the Opportunity will automatically be updated from "Working On" to "Negotiating".
3. After the "Negotiating" phase that the agent will do with the customer he/she then can update the result, its either "Accepted" or "Rejected", and that by clicking one of two buttons "Offer Accepted" and "Offer Rejected" and that will update the offer status from "Negotiating" to either "Accepted" or "Rejected" and it will automatically update. the opportunity status from "Negotiating" to "Closed Lost" or "Closed Won".

Now this leaves us with two different scenarios, one is that the opportunity is closed lost and that will update negatively the score of that agent and also will notify the administration, the second is the opportunity is closed won and that certainly will update positively the agent's score and also will automatically create a contract between the company and the specific customer and it will automatically generate a PDF file with all the information necessary for the administration to read and sign that contract.

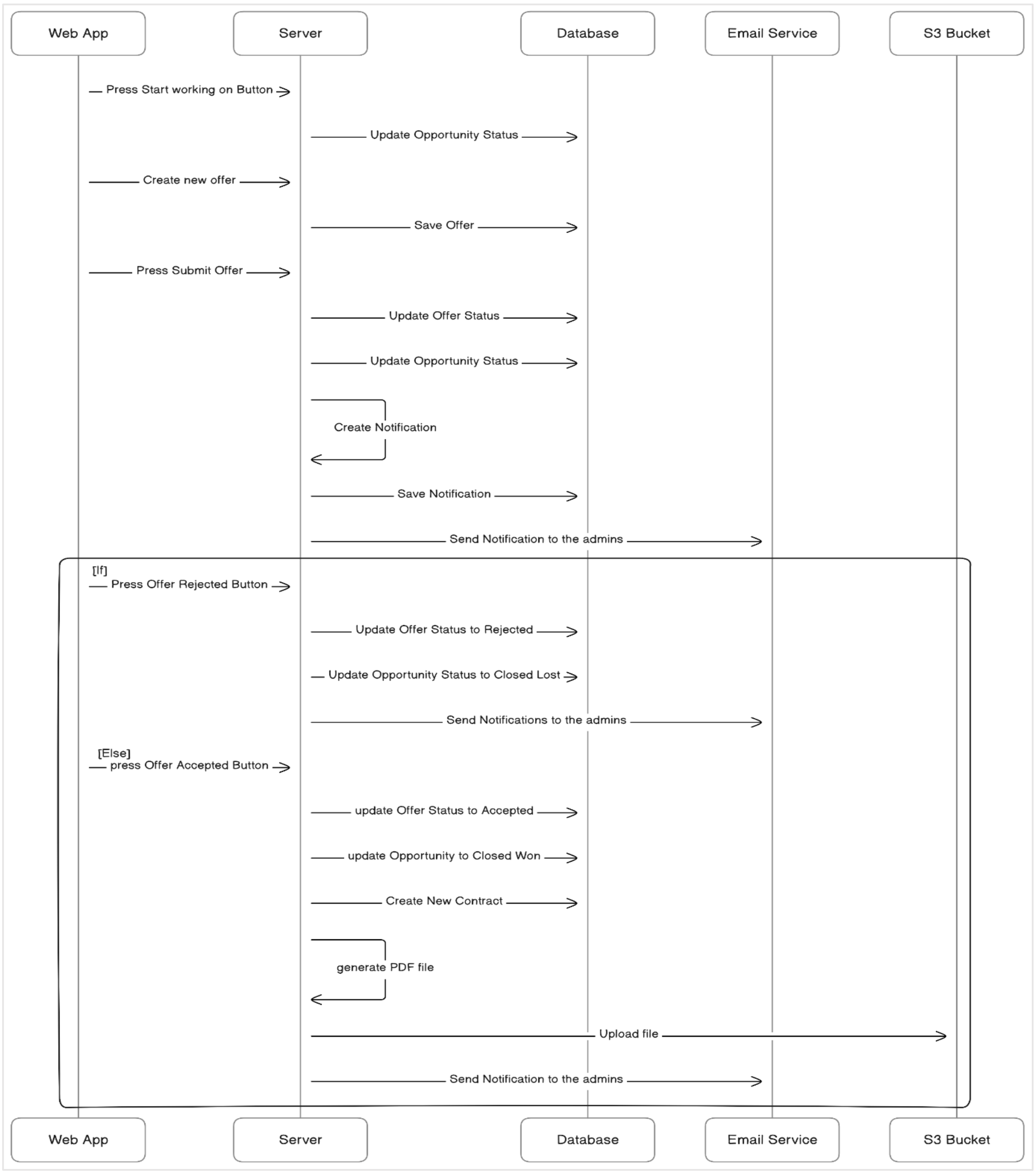

Web App
Server
Database
Email Service
S3 Bucket
Press Start working on Button
Update Opportunity Status
Create new offer
Save Offer
Press Submit Offer
Update Offer Status
Update Opportunity Status
Create Notification
Save Notification
Send Notification to the admins
[If]
Press Offer Rejected Button
Update Offer Status to Rejected
Update Opportunity Status to Closed Lost
Send Notifications to the admins
[Else]
press Offer Accepted Button
update Offer Status to Accepted
update Opportunity to Closed Won
Create New Contract
generate PDF file
Upload file
Send Notification to the admins
Web App
Server
Database
Email Service
S3 Bucket

Figure 34: Sequence Diagram of Commercial Agent handling the Opportunity.

4.3.4-Handling project and tasks:

In the previous sections we covered the process from typing an opportunity until that opportunity becomes a valid project, now in this section will be covering how both admin and IT agent can handle the project and the tasks.

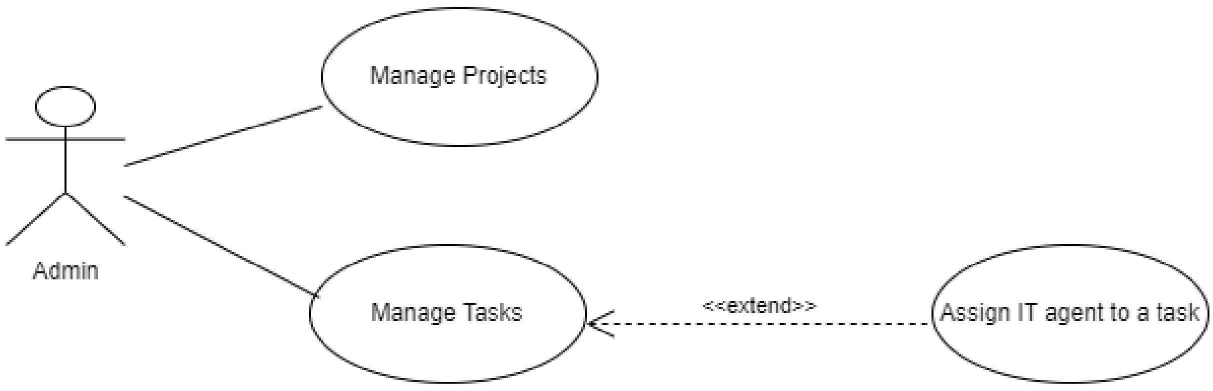


Figure 35: Use case of admin handling projects and tasks

After signing the contract between the company and the customer, now the admin will create a new project and fill it with all the necessary details such as the start date, the end date, description etc. …

After that he will create specific tasks related to that project and then assign IT agents to those tasks.

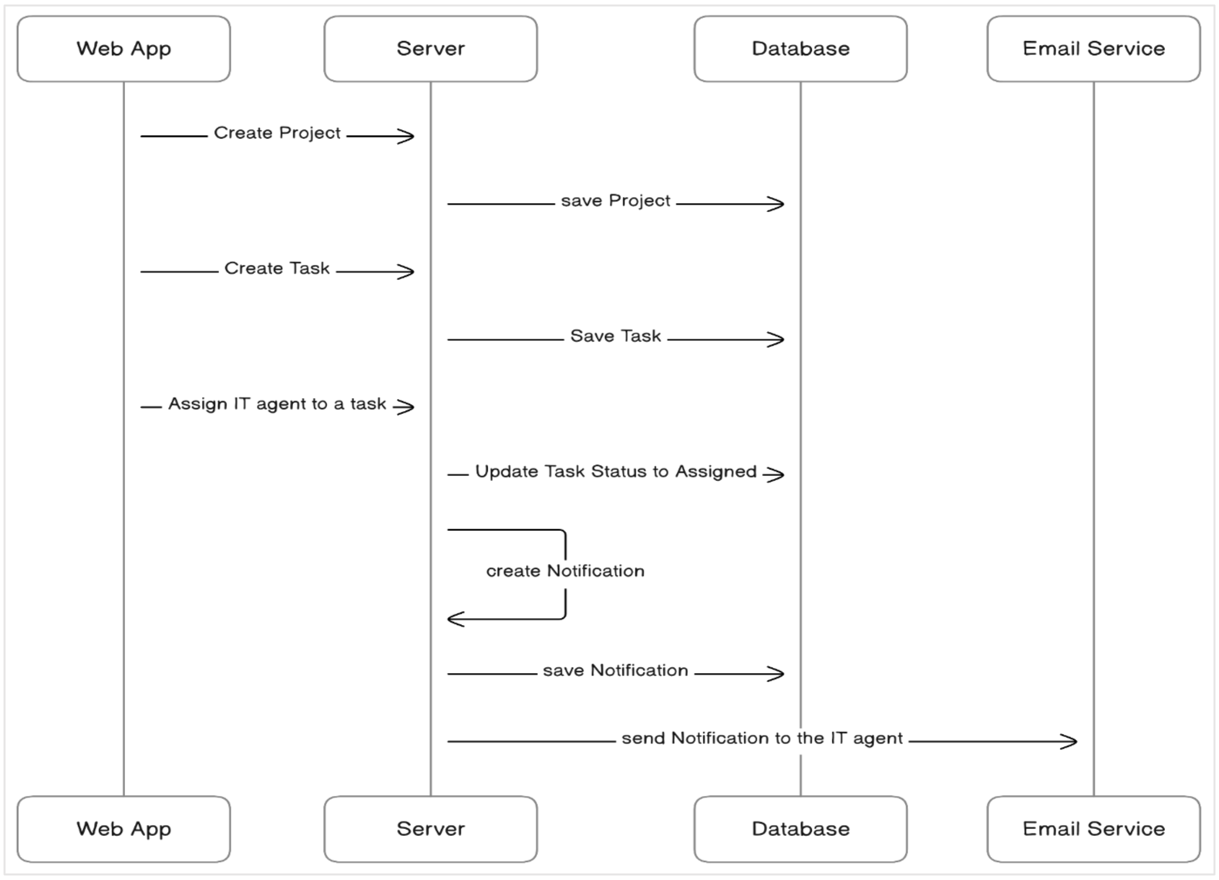


Figure 36: Sequence diagram of creating and assigning tasks

Now the IT agent gets a notification that he is assigned to a new task and we will be discussing his role in the application as follows:

1. The agent can update the task when he starts working on it by pressing the "Start Working On" button, and that will update the status of the task from "Assigned" to "Working On".
2. When he/she finishes that task, he/she will just drag and drop that task to the "Test" field which will automatically update the status from "Working On" to "Test".
3. After the testing phase the admin will decide whether it's done or not, if it's done the status will be updated to "Done" else will go back to "Working On".

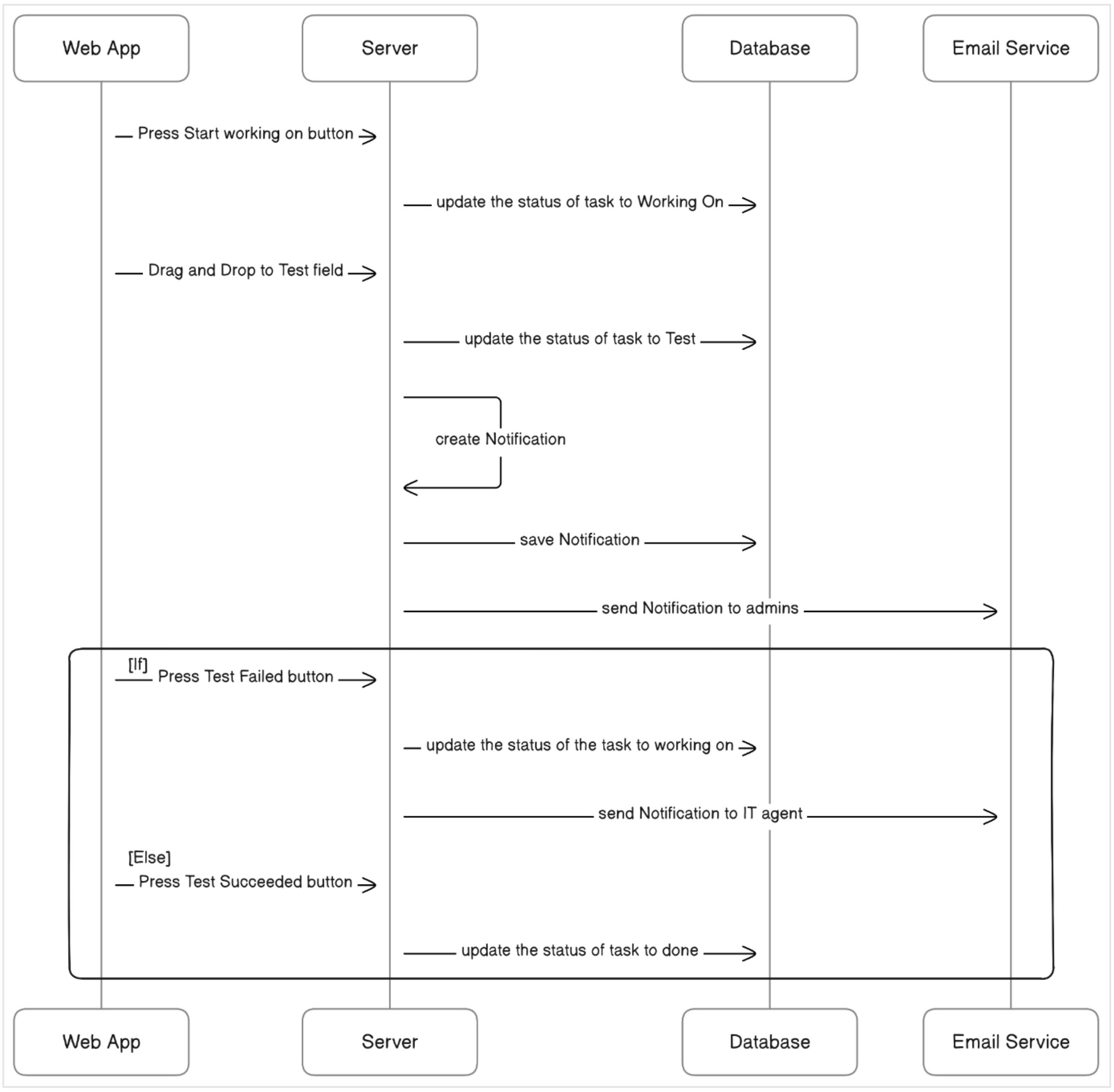

Web App
Server
Database
Email Service
Press Start working on button
update the status of task to Working On
Drag and Drop to Test field
update the status of task to Test
create Notification
save Notification
send Notification to admins
[If]
Press Test Failed button
update the status of the task to working on
send Notification to IT agent
[Else]
Press Test Succeeded button
update the status of task to done
Web App
Server
Database
Email Service

Figure 37: Sequence diagram of handling tasks

Conclusion:

In this chapter, we talked about everything that deals with the architecture, the design of the database, the different ideas we had to achieve our goals for the application, then we shed light on different scenarios, and we focused on the scenarios that take an opportunity from creation till it gets modified and updated through multiple process until it becomes a project.

However we didn’t talk about the other functionalities such as the reminder service that will automatically send emails to remind certain agents that they have 3 days to complete a task or submit an offer for example, we didn’t also talk about the upload and download system but we briefly explained how we deal with it in some scenarios, also we didn’t talk about the agent’s scores and statistics but we will shed some light on it in the next chapter that will be dedicated to the realization of the application.

Chapter 5: Realization

In the previous chapter we discussed all about the design, the infrastructure, the architecture, and other software implementation that we have used to achieve our end goal. Now in this chapter we will dive deep into the realization, and we will provide some screenshots and we will focus on the things that we provided as scenarios in the last chapter, but before we will describe the workflow we used throughout the software development lifecycle, going through our deployment Architecture.
We decided to go with a DevOps methodology which, is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market. [R5]
We need to add there is multiple process that a good DevOps strategy needs and those are CI/CD, and that is essentially the process of pushing code, automatic build, and deployment.
We have used CDK (cloud development kit) to create, maintain all the cloud infrastructure.

5.1-Continuous Integration "CI":

5.1.1- Pushing changes to the Code Repository:

Continuous integration is a software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates. [R5]
It is good to explain that I have worked on this project alone, and the strategy I adapted is anytime I code a new feature I just create a new branch and push the code there, then I merge the main branch if there is no conflicts, since the main branch is dedicated and related to the pipeline we have used (more about this later), I only merge the code when I'm sure that everything is working well.
We follow a git rebase workflow to spot conflicts as soon as possible before pushing to the git repository (GitHub). The workflow is as follow:

1. When I start development, I always make sure the code on my local machine is synced to the latest commit from the remote repository.

```
# check if there is anything wrong by pulling the latest code
git pull
```

2. Next, I will create a new branch so after I finish coding I can add and commit code to this new branch that is separated from the main one.

```
# Create new branch and checkout
git checkout -b new-feature
```

3. After I finish the code, I add everything and create a commit as follows.

```
#Add changes and commit to local repository
git add .
git commit -m “new-feature : Add API endpoint”
```

4. Last but not least, I push to this new branch.

```
#push code to the new branch
git push origin new-feature
```

5. After that I need to check if there is nothing wrong with my new work to the latest code pushed in the main branch

```
git checkout main
git merge new-feature
```

6. If there is conflicts, I try to debug and fix them but if there is nothing wrong I just push the code to the main branch

```
git push origin main
```

5.1.2-Pre-deployment testing:

As a rule of thumb, all types of tests that do not require code to be deployed to a server should be run in this phase. Unit tests always fall into this category. But we decided to skip them since I am the only developer working on this project. So instead, we do test our changes manually on the development environment before pushing them and we rely on a simple test to check whether the application is running or not as a trigger for our "Red" or "Green" CI Pipeline policy. In the case of a "Red", I get notified with an email and I immediately start working on fixing the issue that made the pipeline fails, it becomes a priority for me since the changes that I have made are a few minutes away since my last commit. Finally, when it is "Green" the changes are deployed to our staging environment where they go through extensive testing from our product owner.

5.1.3-Packaging:

I need to insist in this chapter that I have worked on 2 different git repositories, one for the backend and the other is dedicated for the frontend, all of the above information are the same as I work with the same strategy for the continuous integration and pre-deployment testing but in this phase, things get more complicated as the packaging is different for each work.

For the frontend, the method to do it in the JavaScript realm by minimizing the code through a bundling tool such as webpack: e.g., React does endure the hood automatically when building our app for production or staging.
For the backend, we used docker to package the entire application, the process is using a Docker file that after pushing code and the pipeline is triggered, on the building phase that Docker file will generate a Docker image that contains not only the bundled code but also all other dependencies our application might need (Linux libraries...) and push it to Elastic Service Registry (AWS cloud service for Docker Images).
We will go deeper on this on the "CI/CD workflow summary" where we discuss both pipelines.

5.2-Continuous Delivery "CD":

Continuous delivery is a software development practice where code changes are automatically built, tested, and prepared for a release to production. It expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage. When continuous delivery is implemented properly, developers will always have a deployment-ready build artifact that has passed through a standardized test process. [R5]

5.2.1-Deployment to staging environment:

Once the deployment package is created, we can proceed to deploy it to a staging environment, There are 2 different staging environments, one for the frontend and the other for the backend, the first is using AWS S3 with AWS CloudFront, the other is using AWS ECS, AWS Fargate And AWS ECR, again we will dive deep into details of each one in the next section.

5.2.2-Pre-deployment testing:

Once our code is deployed to the staging environment, our product owner will check whether the integration was successful. Through making functional and integration tests to assure that our changes are working as intended and didn't break anything within the application.
There are two possible outcomes that we can expect from this phase:

- Not working as expected Request changes from me.
- Everything is working as expected; Therefore, we start working on the next feature.

5.3-Global CI/CD workflow:

To summarize the workflow we have adopted throughout the entirety of our application development lifecycle, we will be dividing them into 2 parts, frontend pipeline and backend pipeline and that way we get a deeper look and better understanding of what's going on under the hood, and we will get all the details needed for both distributions as each distribution works differently than the other due to different programming languages, different building tools, different deployment services etc. …

Both distributions share the use of git, GitHub, AWS codePipeline and AWS codeBuild but the rest of the services are completely different.

5.3.1-Frontend Pipeline:

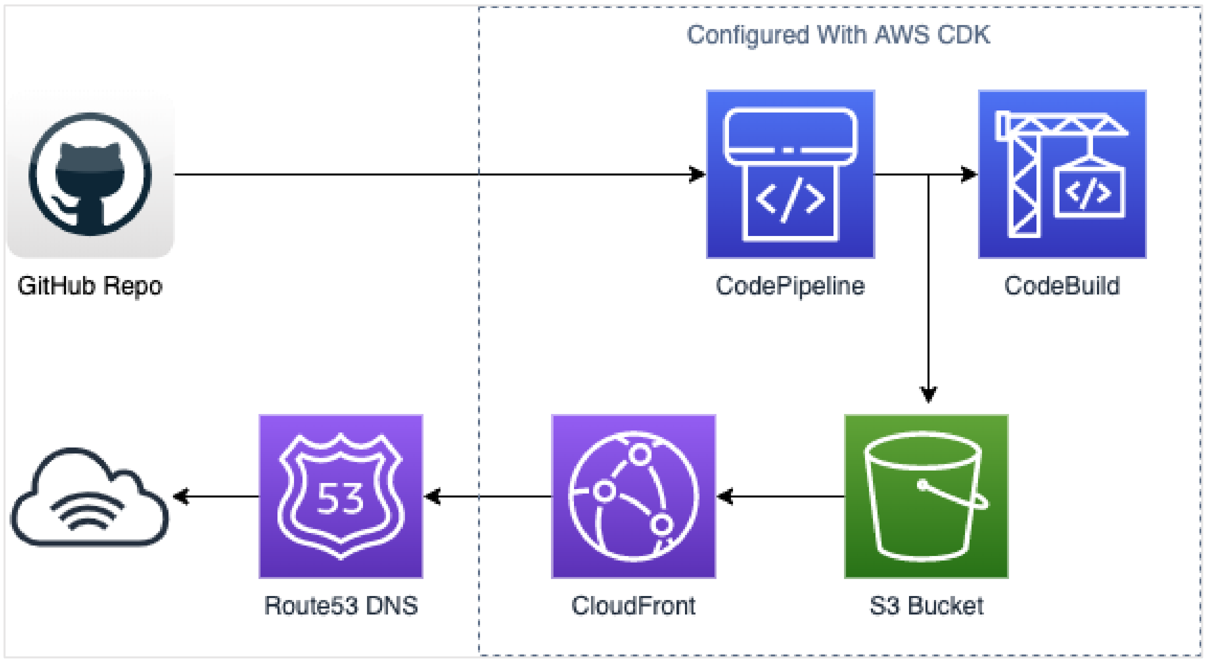


Figure 38: Frontend Pipeline

As shown in the previous figure, the process begins with me pushing the code changes to GitHub to our main branch (as other feature branch won't trigger the pipeline), where our source code resides.

AWS CodePipeline monitors this repository and, upon detecting changes, triggers the pipeline.

The pipeline itself manages a sequence of stages, including source code retrieval from GitHub, automated testing with CodeBuild, and potentially more stages for deployment to various environments.

AWS CodeBuild plays a crucial role in this process by compiling your React application, running tests, and generating deployment-ready artifacts. Once all stages pass successfully, the pipeline proceeds to automatically deploy our React application.

Here's where AWS S3 and CloudFront come into play:

- After successful testing, CodePipeline can upload the build artifacts (e.g., minified JavaScript files, HTML, CSS) to an Amazon S3 bucket. S3 serves as a highly scalable and durable object storage service. It securely stores our application assets and allows for easy retrieval.
- To deliver our React application with low latency and high availability, AWS CloudFront, a content delivery network (CDN), is employed. CloudFront acts as a global edge server that caches and distributes our application's content to endpoints closer to your users. This minimizes latency and enhances the overall user experience.

With S3 and CloudFront in the mix, our React application is not only efficiently built and tested but also hosted in a scalable, reliable, and performant manner. This comprehensive pipeline ensures that our software is continuously integrated, thoroughly tested, and seamlessly delivered to end-users, enhancing development agility and maintaining high standards of code quality and user experience.

We then configured Amazon Route 53 to manage the DNS records for our custom domain or subdomains, directing traffic to our CloudFront distribution.
This ensures that users access our React application via a user-friendly domain name (e.g., www.example.com).
N.B: I cannot provide the actual domain name of the application as it is confidential to the company.

5.3.2-Backend Pipeline:

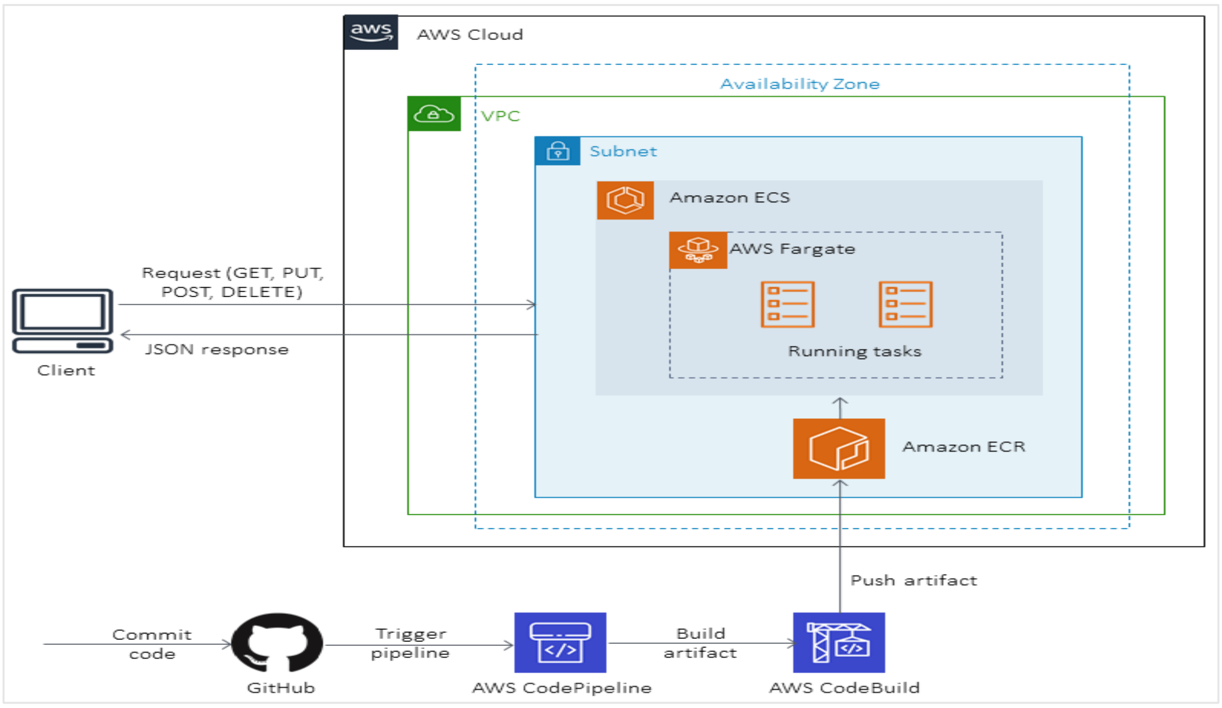


Figure 39: Backend Pipeline

The backend pipeline is also configured using AWS CDK, the process as shown in the previous figure is more complicated than the frontend pipeline, but we will explain everything going on. And here is a full step guide to the whole process:

1. The process starts with committing my code changes to the main branch after I worked on the Spring Boot REST API code.

2. AWS CodePipeline monitors the GitHub repository for changes. When code is pushed into the main branch, CodePipeline is triggered to initiate the deployment process. (See Annex B)

3. CodePipeline invokes AWS CodeBuild to build the Spring Boot application. CodeBuild uses the specified build specifications and dependencies defined in the pom.xml file (managed by Maven). This step compiles the code, runs unit tests, and packages the application into a JAR file. (See Annex A)

4. The Docker image for the Spring Boot application is created by CodeBuild. It uses a Dockerfile that defines how the application should be packaged within a container. This Docker image includes the Java runtime, the Spring Boot application JAR file, and any other dependencies. (See Annex C)

5. The newly built Docker image is pushed to Amazon ECR, a fully managed Docker container registry. ECR stores and manages our Docker image securely.

6. AWS CodePipeline, in conjunction with Amazon ECS, orchestrates the deployment of the Docker container to the staging environment. ECS is responsible for managing containerized applications and tasks. In this case, it deploys the Spring Boot API as a container. (See Annex D)

7. The Spring Boot API runs in a VPC for network isolation and enhanced security. We configured VPC settings to control inbound and outbound traffic, ensuring that only authorized requests can access the API.

8. Within Amazon ECS, AWS Fargate is used as a compute engine to manage the containers. Fargate abstracts away the underlying infrastructure, making it easier to deploy and manage containers without the need to provision or manage servers.

This deployment process leverages AWS services and tools to automate the building, packaging, and deployment of our Spring Boot REST API, providing consistency and reliability while allowing for thorough testing in a controlled environment before updates reach production. It ensures that changes are deployed in a secure and scalable manner.

5.4-Screenshots:

In this section we will be giving a full demonstration of the application by providing screenshots about the application and how we use it to its full potential. We will be starting by the authentication process, then we will talk about adding and managing opportunities then we will finish by generating a contract and saving it.

5.4.1-Authentication:

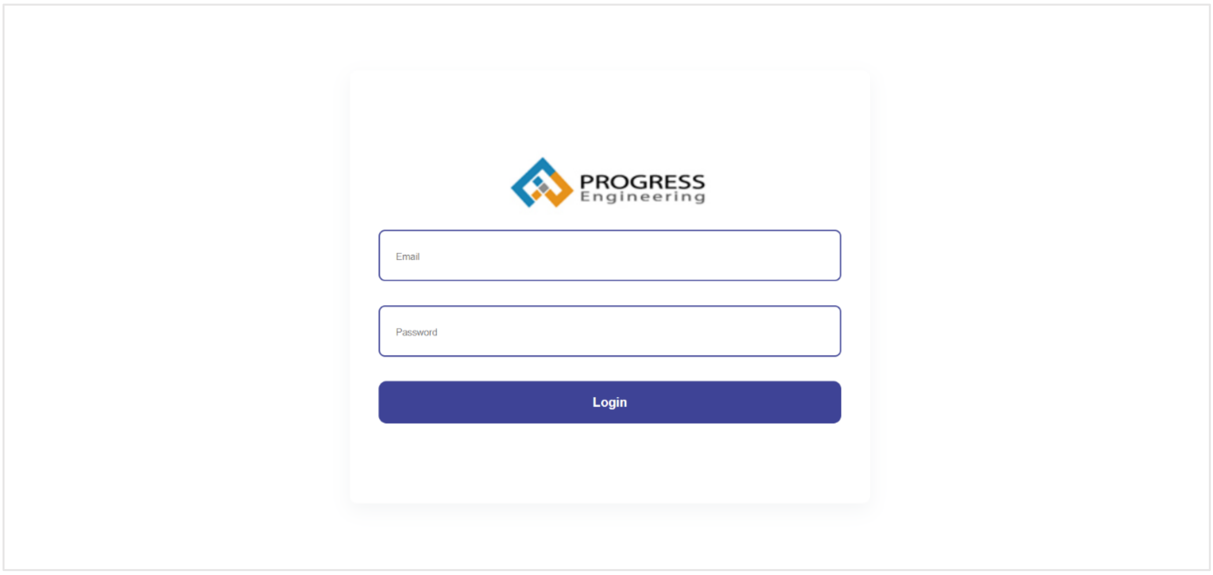


Figure 40: Login Screenshot

The login page shown in the figure above is shared between all actors (Admin, Secretary, Commercial Agent, IT Agent), they put their required information and then each actor will be redirected to its own interface(dashboard), this is a role-based authentication & authorization and this is one of the best benefits of working with a single page application framework such as ReactJs.

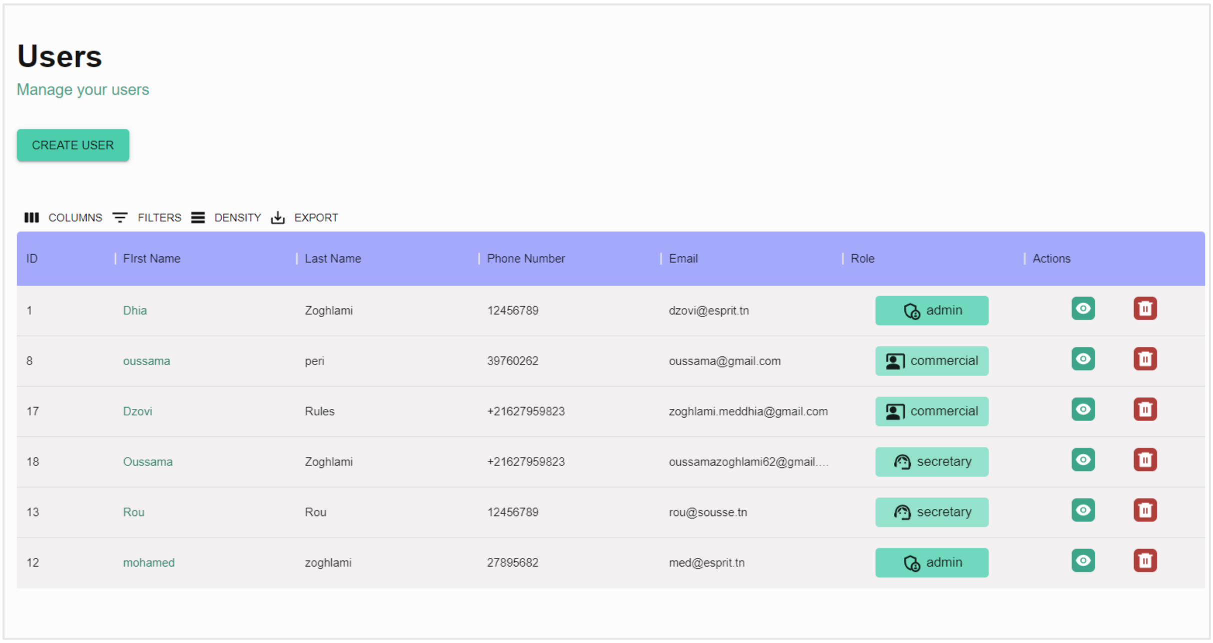
The process of authentication works as follow:

1. The user types its email and password.
2. That information will be sent to the server where it will check if the user exist and if the password is correct, if there is any error it will return an exception, if everything is correct, it will generate an access token and a refresh token and those will be sent to the client.
3. The client side (Browser) will store those tokens in a secure way, after that it will check the role of the user and then redirect him/her to its own proper dashboard.
4. After a successful login, each actor now is logged in to its own dashboard while he/she can start performing actions they are allowed to.

I need to mention that from now on I will be using random data and random users because of confidentiality reasons.

5.4.2-Creating an Employee:

Let’s assume that an admin is logged in successfully, now we will share the table of all users and how he can manage his users.



Users

Manage your users

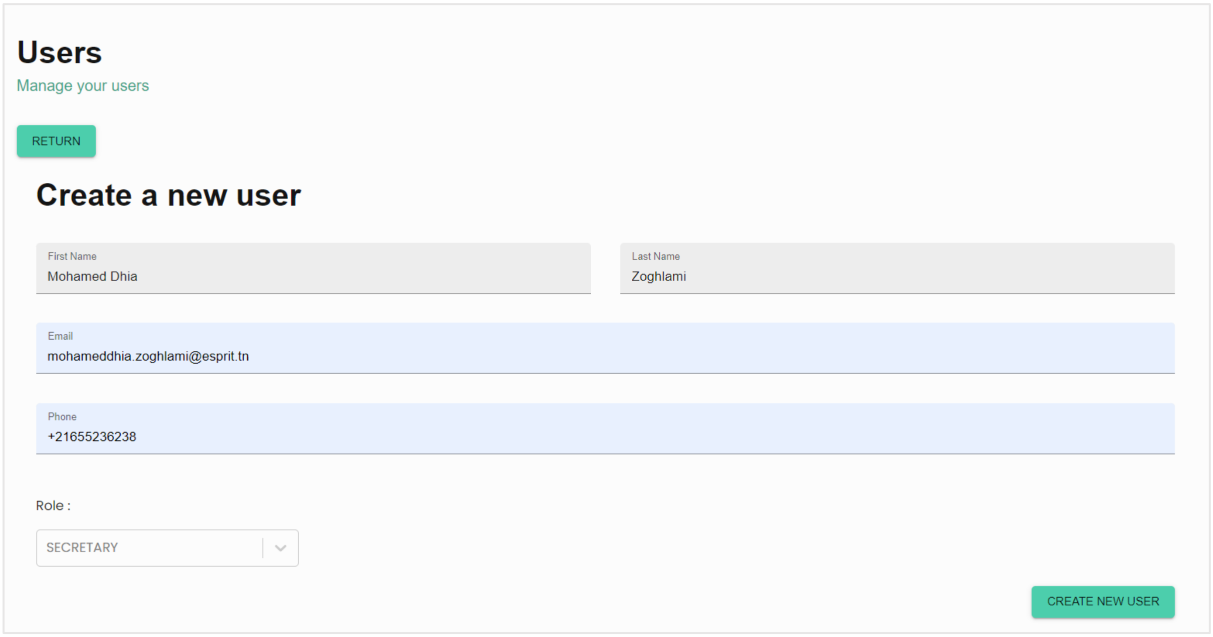
CREATE USER

COLUMNS FILTERS DENSITY EXPORT

ID	First Name	Last Name	Phone Number	Email	Role	Actions
1	Dhia	Zoghlami	12456789	dzovi@esprit.tn	admin	
8	oussama	peri	39760262	oussama@gmail.com	commercial	
17	Dzovi	Rules	+21627959823	zoghlami.meddhia@gmail.com	commercial	
18	Oussama	Zoghlami	+21627959823	oussamazoghlami62@gmail....	secretary	
13	Rou	Rou	12456789	rou@sousse.tn	secretary	
12	mohamed	zoghlami	27895682	med@esprit.tn	admin	

Figure 41: List of Users

The previous figure shows a table of all users, the only actor that has the access to this table is the admin, he/she can manage their employees as necessary by creating a new one, see a specific user information, or even deleting a user.



Users

Manage your users

RETURN

Create a new user

First Name: Mohamed Dhia

Last Name: Zoghlami

Email: mohameddhia.zoghlami@esprit.tn

Phone: +21655236238

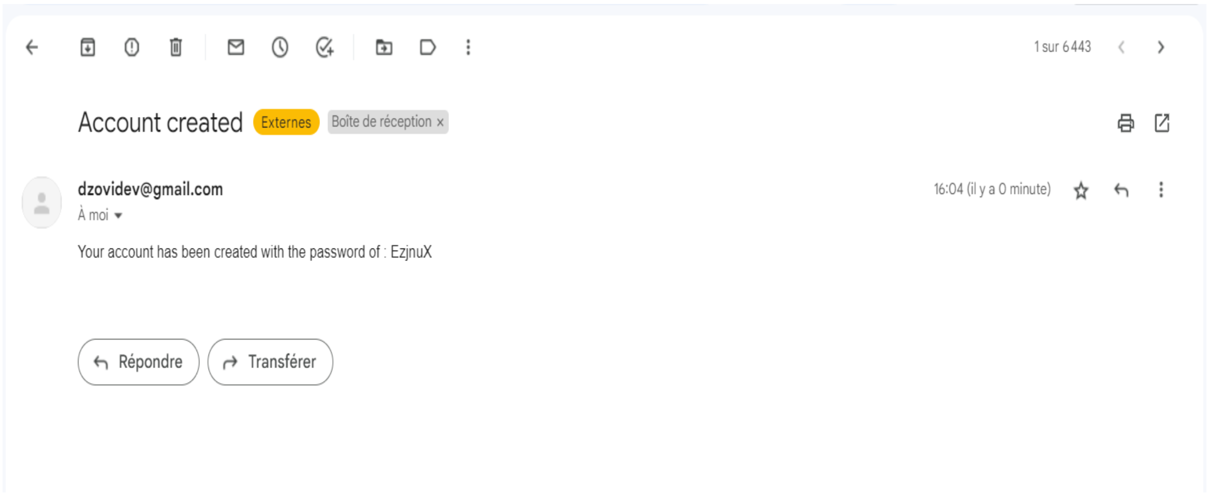
Role : SECRETARY

CREATE NEW USER

Figure 42: List of Users

As shown above in the figure, the steps to creating a user are as follows:

1. Fill up necessary information about the user (first name, last name, email, phone and more importantly its role).
2. Press the "Create New User" button which will generate a random password, send an email to that new user giving his/her password, and saving that user into the database.



Account created

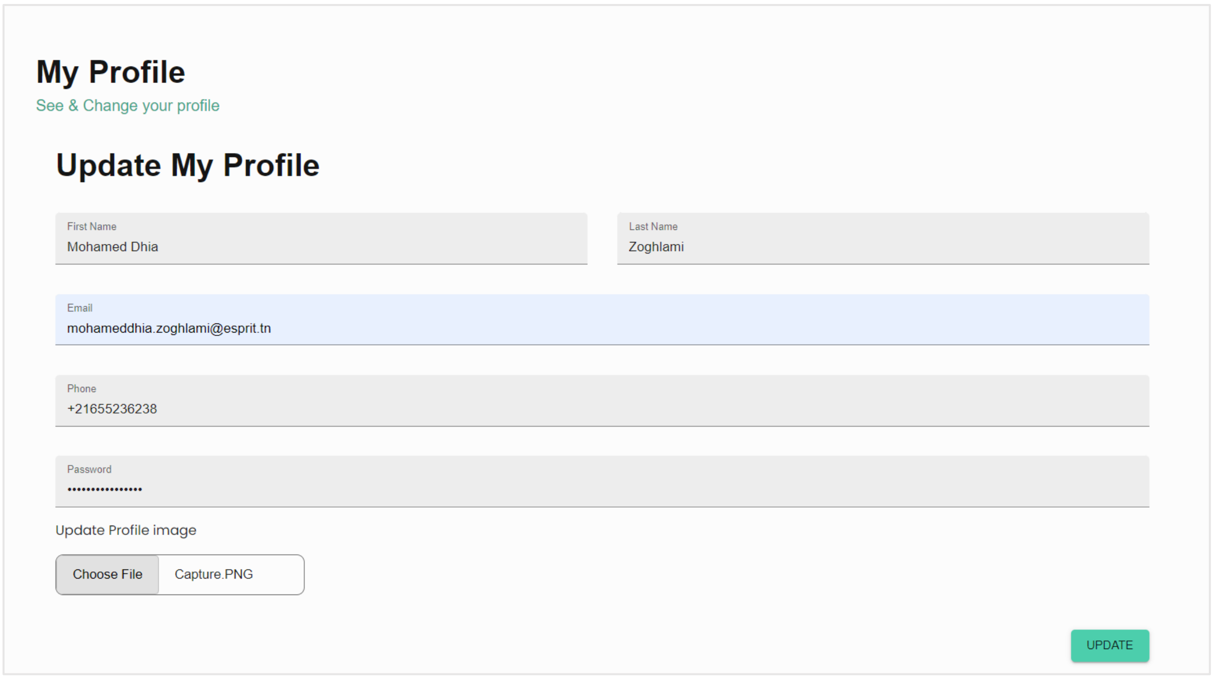
dzovidev@gmail.com
À moi
16:04 (il y a 0 minute)

Your account has been created with the password of : EzjnuX

Figure 43: Account created Email.

As I mentioned before, the new user will receive an email letting him/her that his/her account has been created and providing him with a password that he/she can use to get access to the application and start working on its role.

I also need to mention that the new user and every other user can change its information including the password automatically generated in his/her own interface and also can add a profile picture, as the figure below shows:



My Profile

See & Change your profile

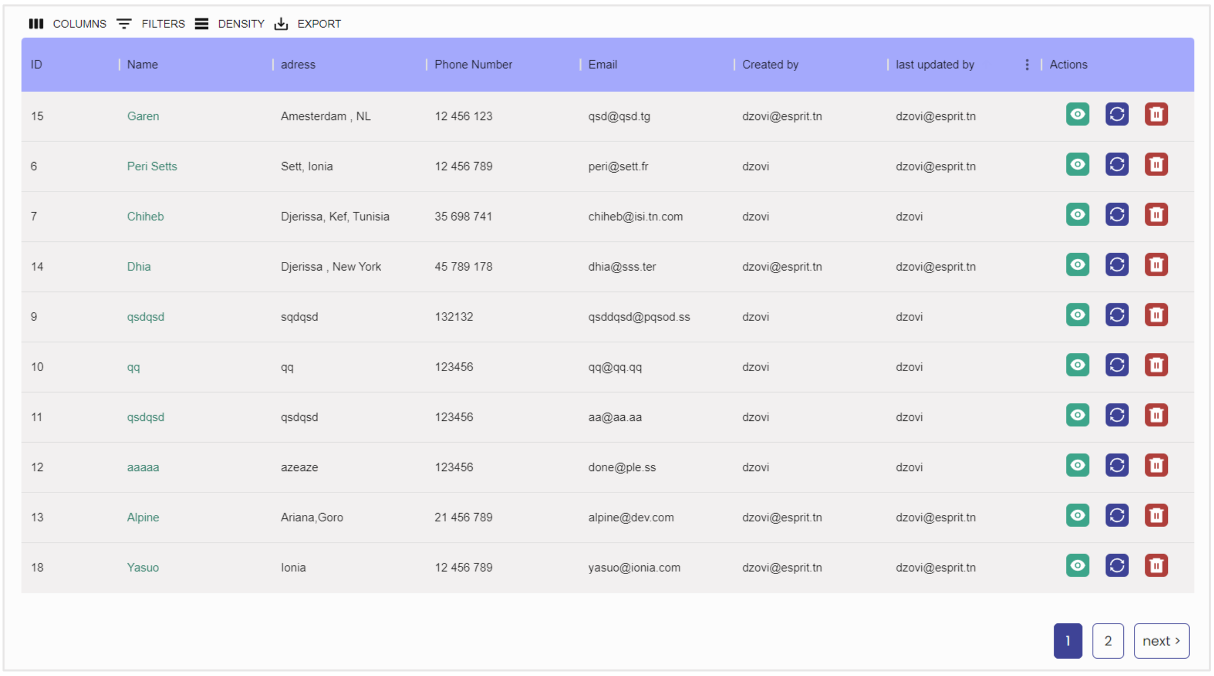
Update My Profile

First Name: Mohamed Dhia
Last Name: Zoghlami
Email: mohameddhia.zoghlami@esprit.tn
Phone: +21655236238
Password: ••••••••••••••••
Update Profile image: Capture.PNG

Figure 44: Update Profile

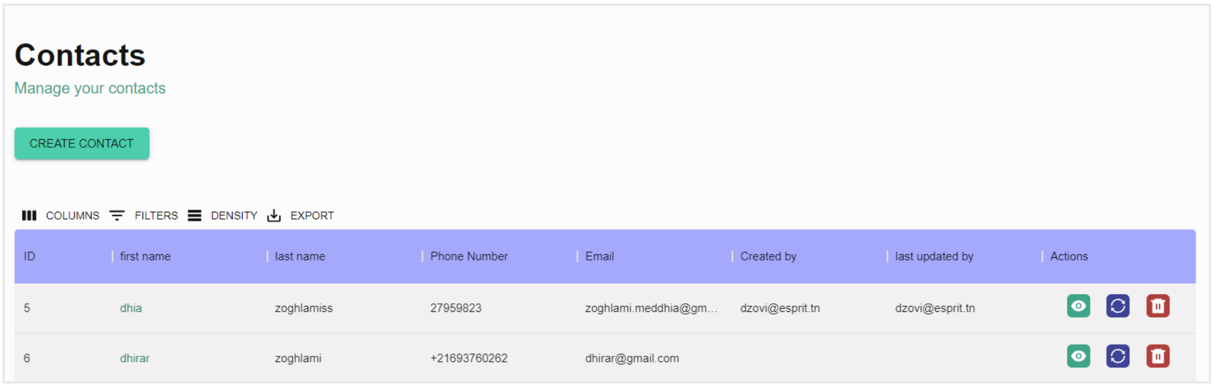
5.4.3-Secretary Dashboard:

As we talked about before, the secretary has access to customers, contacts and opportunities where he/she can add new customers or contacts, and more importantly can add opportunities.



ID	Name	adress	Phone Number	Email	Created by	last updated by	Actions
15	Garen	Amesterdam , NL	12 456 123	qsd@qsd.tg	dzovi@esprit.tn	dzovi@esprit.tn	👁 🔄 🗑
6	Peri Setts	Sett, Ionia	12 456 789	peri@sett.fr	dzovi	dzovi@esprit.tn	👁 🔄 🗑
7	Chiheb	Djerissa, Kef, Tunisia	35 698 741	chiheb@isi.tn.com	dzovi	dzovi	👁 🔄 🗑
14	Dhia	Djerissa , New York	45 789 178	dhia@sss.ter	dzovi@esprit.tn	dzovi@esprit.tn	👁 🔄 🗑
9	qsdqsd	sqdqsd	132132	qsddqsd@pqsod.ss	dzovi	dzovi	👁 🔄 🗑
10	qq	qq	123456	qq@qq.qq	dzovi	dzovi	👁 🔄 🗑
11	qsdqsd	qsdqsd	123456	aa@aa.aa	dzovi	dzovi	👁 🔄 🗑
12	aaaaa	azeaze	123456	done@ple.ss	dzovi	dzovi	👁 🔄 🗑
13	Alpine	Ariana,Goro	21 456 789	alpine@dev.com	dzovi@esprit.tn	dzovi@esprit.tn	👁 🔄 🗑
18	Yasuo	Ionia	12 456 789	yasuo@ionia.com	dzovi@esprit.tn	dzovi@esprit.tn	👁 🔄 🗑

Figure 45: Customers table



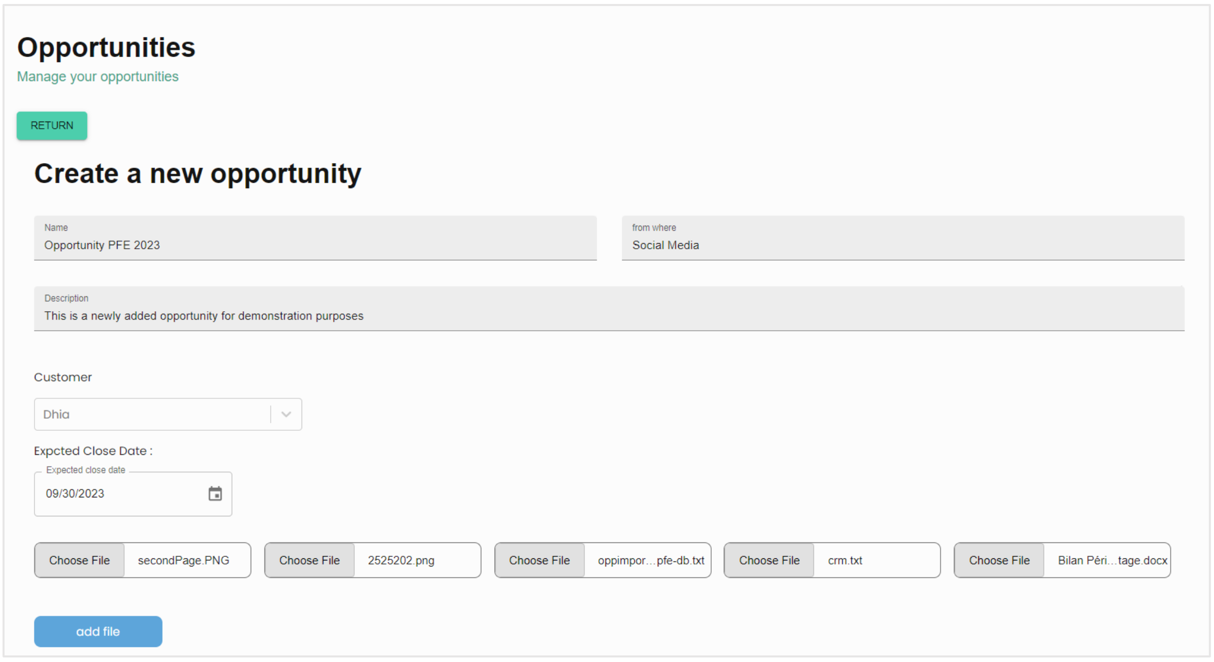
Contacts

Manage your contacts

ID	first name	last name	Phone Number	Email	Created by	last updated by	Actions
5	dhia	zoghlamiss	27959823	zoghlami.meddhia@gm...	dzovi@esprit.tn	dzovi@esprit.tn	👁 🔄 🗑
6	dhirar	zoghlami	+21693760262	dhirar@gmail.com			👁 🔄 🗑

Figure 46: Contacts table

As shown in the 2 previous figures, we provided both the customers and the contacts table, but I need to add that we will not provide the opportunity table here but we will provided in the section dedicated to the admin dashboard. However, in this section we will also provide the create opportunity interface to showcase what the secretary can do.


Opportunities

Manage your opportunities

RETURN

Create a new opportunity

Name: Opportunity PFE 2023

from where: Social Media

Description: This is a newly added opportunity for demonstration purposes

Customer: Dhia

Expcted Close Date :

Expected close date: 09/30/2023

Choose File secondPage.PNG | Choose File 2525202.png | Choose File oppimpor…pfe-db.txt | Choose File crm.txt | Choose File Bilan Péri…tage.docx

add file

Figure 47: Create Opportunity

The secretary needs to fill every information, and he/she can add as many documents as possible, the more documents or files provided the easier it gets for admins to decide if they have to ignore this opportunity or engage in it.
I also need to add that these are not the only attributes necessary, there is a lot going on under the hood and multiple attributes are generated adequately such as “created by”, “created at”, “opportunity_decision” is set to “Deciding” etc. …

This opportunity is also in relation with the customer field in case this opportunity is related to a new customer, the secretary needs to create a new customer then create the new opportunity.

Also, the “From where” attributes is provided so the admins can have an understanding of where they heard from this opportunity and also the commercial agent will benefit from that information.

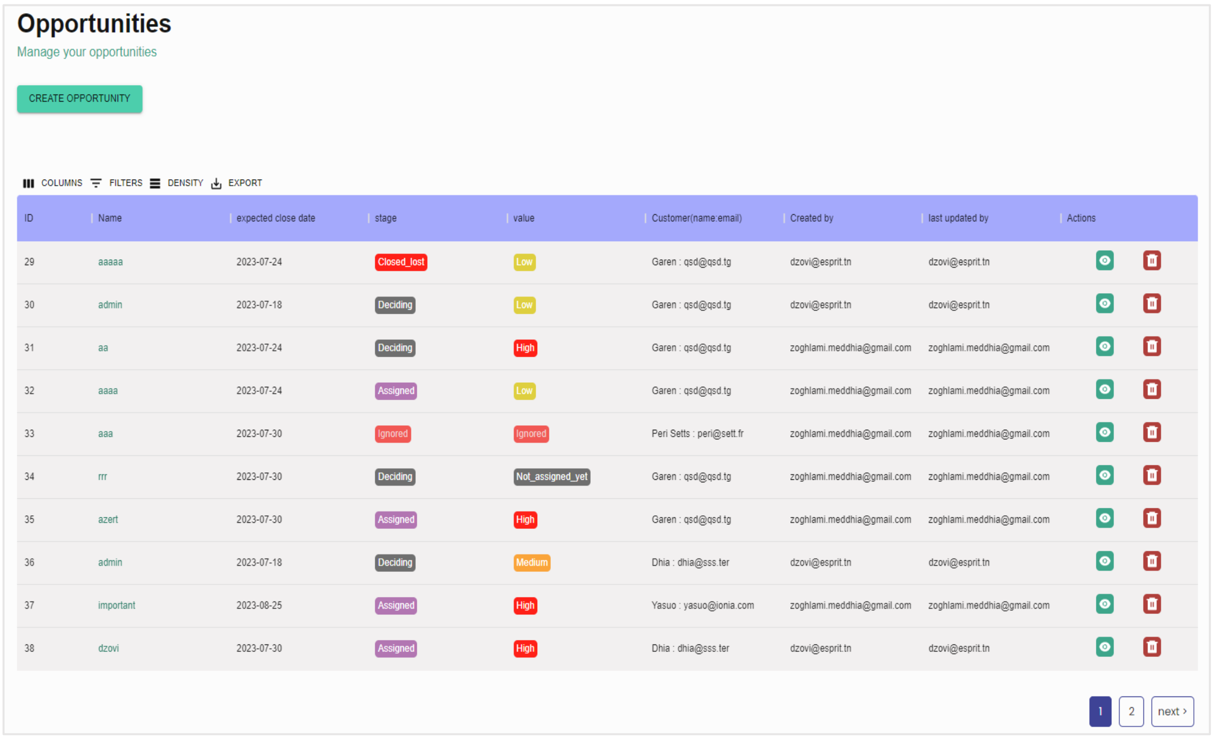
In case that this opportunity is provided by a contact, the “From where” field needs to be typed as the email of that contact.

With this information being saved in the database, now both the admins and the commercial agent who will be assigned to this opportunity will have knowledge about the customer, the contact if there is any which will surely help both parties to make a better offer and even increase the chances of successfully closing that opportunity.

5.4.4-Admin Dashboard:

As we all know by now, the admin has full control of the entire application, but in this section, we will be covering the opportunity management then after we cover the commercial agent dashboard, we will come back to talk more about the admin dashboard.

We won't be covering the customers table and the contacts table because they are the same as the ones shown in the secretary dashboard section.



Opportunities

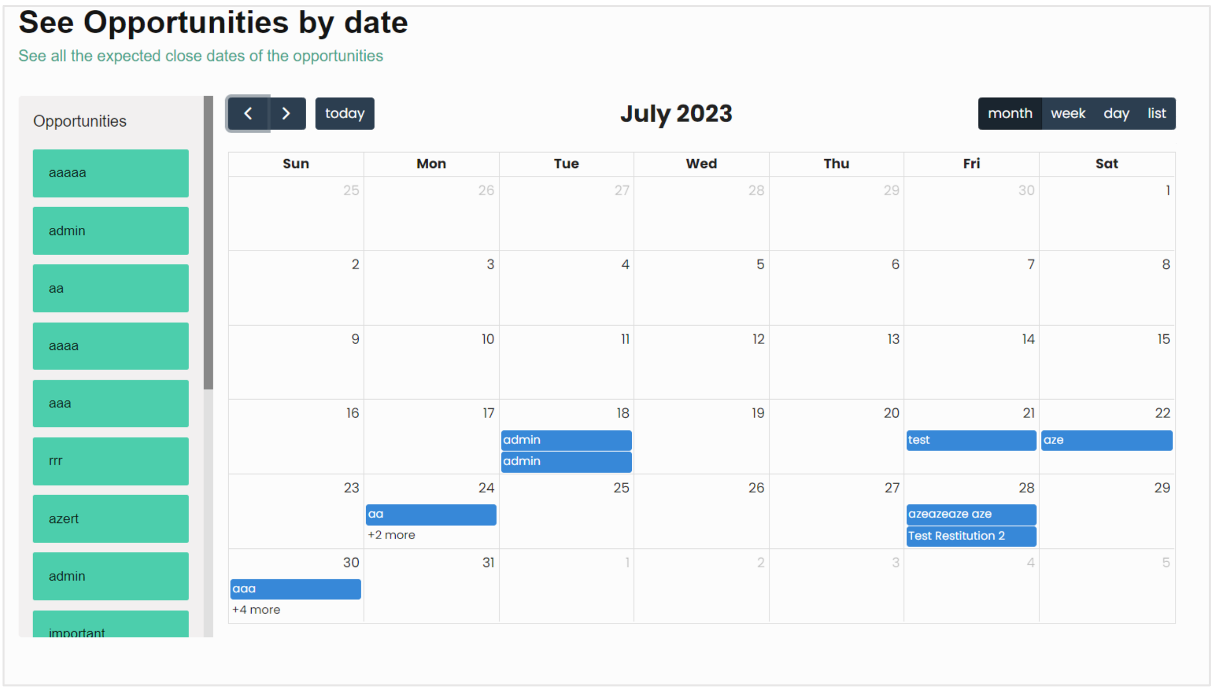
Manage your opportunities

ID	Name	expected close date	stage	value	Customer(name:email)	Created by	last updated by	Actions
29	aaaaa	2023-07-24	Closed_lost	Low	Garen : qsd@qsd.tg	dzovi@esprit.tn	dzovi@esprit.tn	
30	admin	2023-07-18	Deciding	Low	Garen : qsd@qsd.tg	dzovi@esprit.tn	dzovi@esprit.tn	
31	aa	2023-07-24	Deciding	High	Garen : qsd@qsd.tg	zoghlami.meddhia@gmail.com	zoghlami.meddhia@gmail.com	
32	aaaa	2023-07-24	Assigned	Low	Garen : qsd@qsd.tg	zoghlami.meddhia@gmail.com	zoghlami.meddhia@gmail.com	
33	aaa	2023-07-30	Ignored	Ignored	Peri Setts : peri@sett.fr	zoghlami.meddhia@gmail.com	zoghlami.meddhia@gmail.com	
34	rrr	2023-07-30	Deciding	Not_assigned_yet	Garen : qsd@qsd.tg	zoghlami.meddhia@gmail.com	zoghlami.meddhia@gmail.com	
35	azert	2023-07-30	Assigned	High	Garen : qsd@qsd.tg	zoghlami.meddhia@gmail.com	zoghlami.meddhia@gmail.com	
36	admin	2023-07-18	Deciding	Medium	Dhia : dhia@sss.ter	dzovi@esprit.tn	dzovi@esprit.tn	
37	important	2023-08-25	Assigned	High	Yasuo : yasuo@ionia.com	zoghlami.meddhia@gmail.com	zoghlami.meddhia@gmail.com	
38	dzovi	2023-07-30	Assigned	High	Dhia : dhia@sss.ter	dzovi@esprit.tn	dzovi@esprit.tn	

Figure 48: Opportunities table

This table provide a global overview of all opportunities and their specific information, most importantly their stage, value and customers which benefit the admins the track the progress of every opportunity without requesting many details from their employees.

Also, there is a calendar providing an overview of all opportunities to better keep knowing about the recent added ones, or the ones that their date are close as show in the figure below:

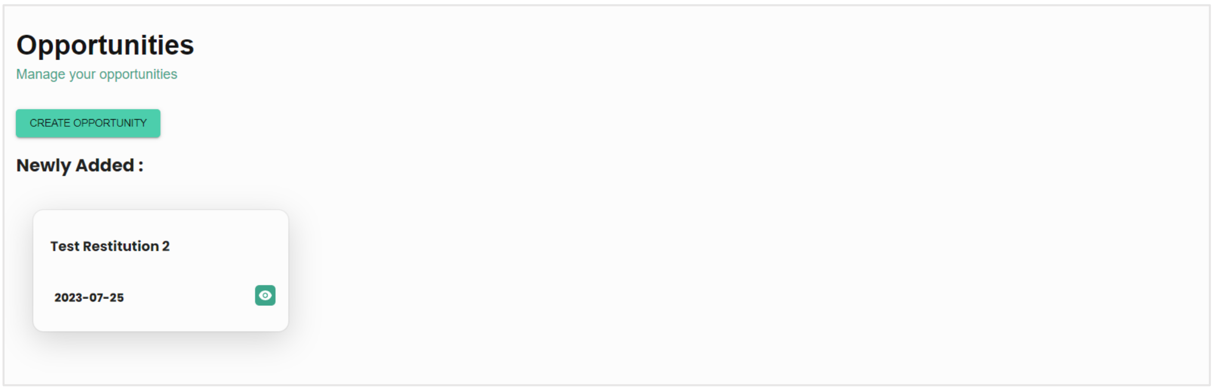

See Opportunities by date

See all the expected close dates of the opportunities

Figure 49: Opportunities Calendar

Now, let’s talk about the case when the secretary creates a new opportunity, but now we will talk about the admin’s point of view.

Once a new opportunity has been added to the database, the admins will get notified in 2 ways, The first is through their email, the second is inside of the dashboard as the next figure shows:


Opportunities

Manage your opportunities

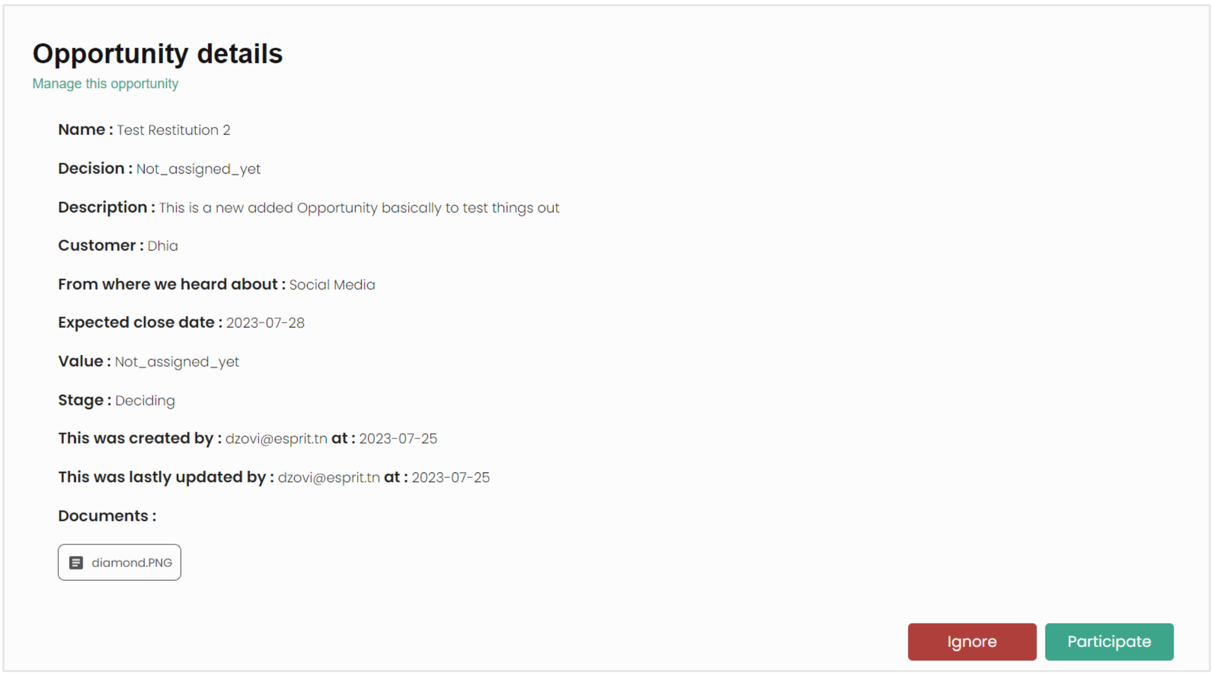
Newly Added :

Test Restitution 2

2023-07-25

Figure 50: Newly added Opportunity

The admin then can just click on that opportunity and checks its details and download its files and documents:


Opportunity details

Manage this opportunity

Name : Test Restitution 2

Decision : Not_assigned_yet

Description : This is a new added Opportunity basically to test things out

Customer : Dhia

From where we heard about : Social Media

Expected close date : 2023-07-28

Value : Not_assigned_yet

Stage : Deciding

This was created by : dzovi@esprit.tn **at :** 2023-07-25

This was lastly updated by : dzovi@esprit.tn **at :** 2023-07-25

Documents :

diamond.PNG

Figure 51: Opportunity Details

After reading all about the opportunity, the admin now can decide whether to ignore or engage, by pressing the “Ignore” button, that opportunity will be ignored but still stored in the database, but in case the admin decides to engage, he/she presses the “Participate” button and then needs to provide a value of that opportunity:

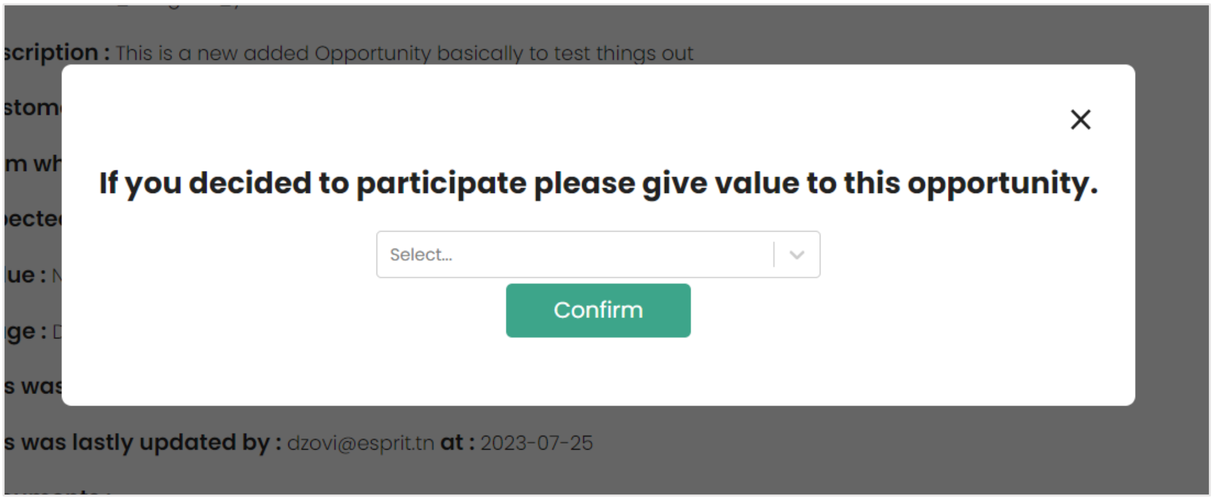

If you decided to participate please give value to this opportunity.

Figure 52: Opportunity value

The value of the opportunity is one of three: “Low”, “Mid” and “High”.

After providing a proper value to the opportunity, the admin now has the ability to assign a commercial agent to the opportunity:

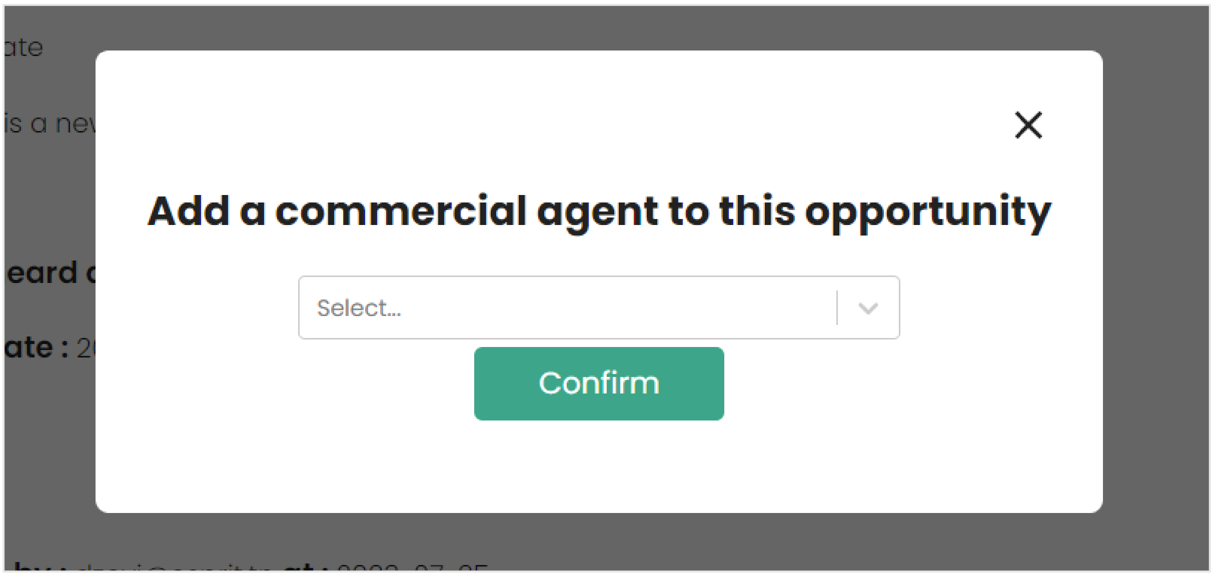


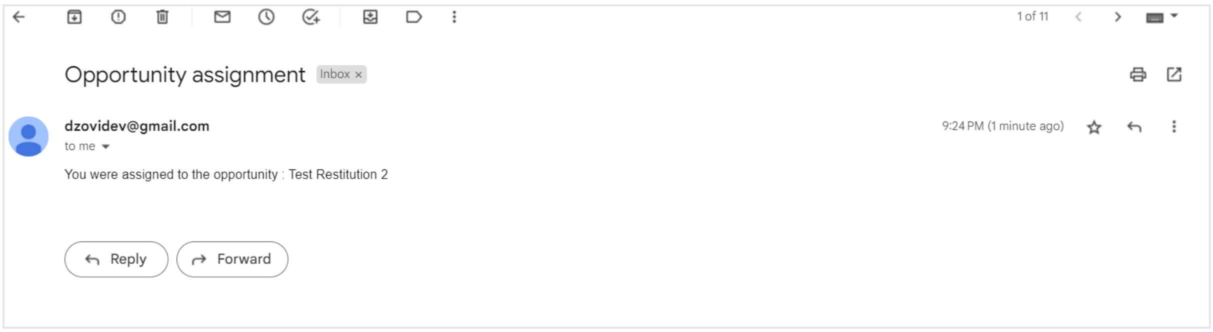
Figure 53: Opportunity assignment

By pressing the "Confirm" button, a commercial agent is assigned to this opportunity, and the opportunity stage is automatically updated from "Deciding" to "Assigned".

5.4.5-Commercial Agent Dashboard:

The commercial agent can read the opportunities he/she is assigned to, manage offers and submit them.

After the admin assign a specific commercial agent to an opportunity, that agent get notified by email:



Opportunity assignment

dzovidev@gmail.com

to me

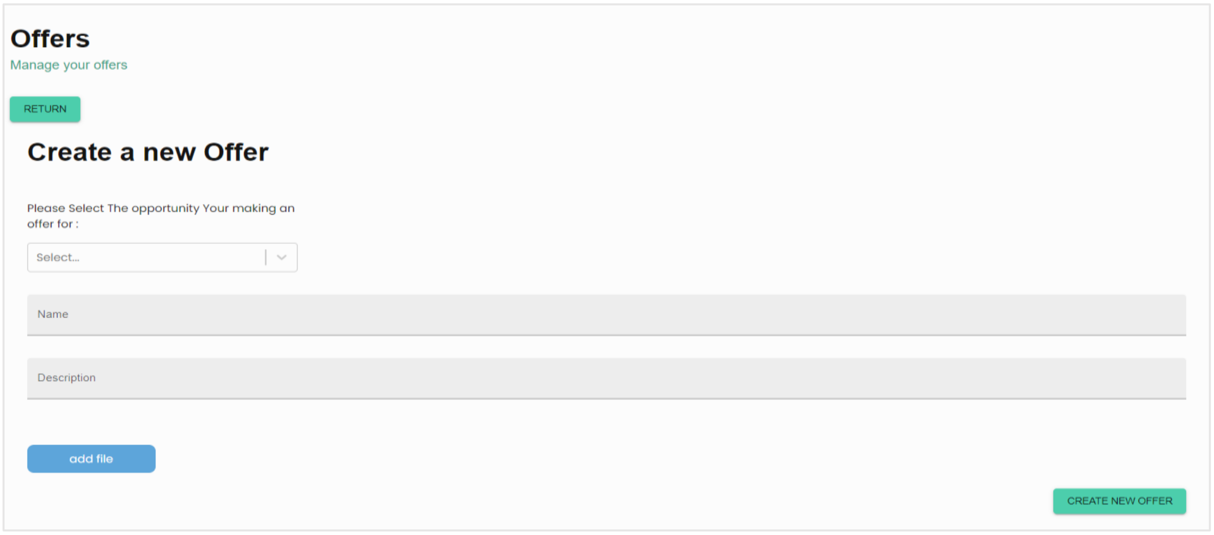
9:24 PM (1 minute ago)

You were assigned to the opportunity : Test Restitution 2

Figure 54: Opportunity assignment Email

Afterwards, the commercial agent can log in to his dashboard and start his role in the process, he/she can (after reading the details of the opportunity), start working on an offer by creating one or multiple offers associated with that particular opportunity but as we mentioned before he/she only can submit one offer.

I need to mention that the offer has to be done in a PDF or DOC format, then the agent can just upload it to the application with adding just the offer name and a small description.



Offers

Manage your offers

RETURN

Create a new Offer

Please Select The opportunity Your making an offer for :

Select...

Name

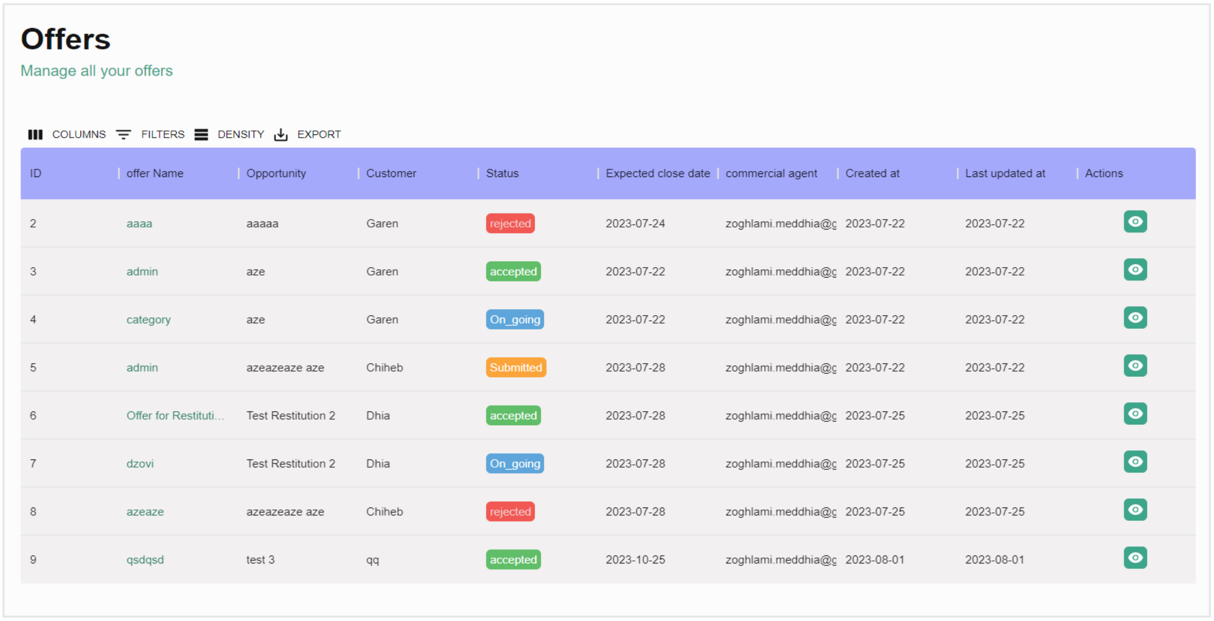
Description

add file

CREATE NEW OFFER

Figure 55: Create Offer

All the offers are stored in the database and the commercial agent can access the offers he/she created but the admin of course can access all offers:



Offers

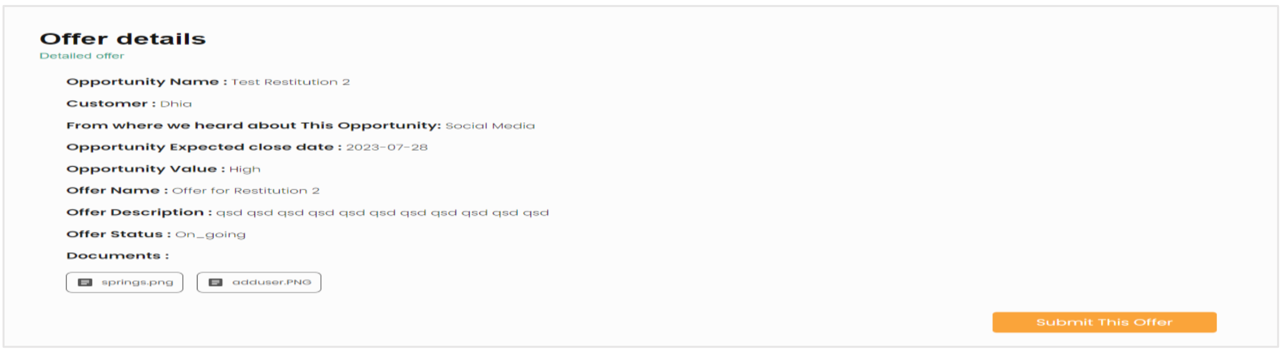
Manage all your offers

COLUMNS FILTERS DENSITY EXPORT

ID	offer Name	Opportunity	Customer	Status	Expected close date	commercial agent	Created at	Last updated at	Actions
2	aaaa	aaaaa	Garen	rejected	2023-07-24	zoghlami.meddhia@g	2023-07-22	2023-07-22	👁
3	admin	aze	Garen	accepted	2023-07-22	zoghlami.meddhia@g	2023-07-22	2023-07-22	👁
4	category	aze	Garen	On_going	2023-07-22	zoghlami.meddhia@g	2023-07-22	2023-07-22	👁
5	admin	azeazeaze aze	Chiheb	Submitted	2023-07-28	zoghlami.meddhia@g	2023-07-22	2023-07-22	👁
6	Offer for Restituti...	Test Restitution 2	Dhia	accepted	2023-07-28	zoghlami.meddhia@g	2023-07-25	2023-07-25	👁
7	dzovi	Test Restitution 2	Dhia	On_going	2023-07-28	zoghlami.meddhia@g	2023-07-25	2023-07-25	👁
8	azeaze	azeazeaze aze	Chiheb	rejected	2023-07-28	zoghlami.meddhia@g	2023-07-25	2023-07-25	👁
9	qsdqsd	test 3	qq	accepted	2023-10-25	zoghlami.meddhia@g	2023-08-01	2023-08-01	👁

Figure 56: Offers table.

The commercial agent has the ability to submit an offer by clicking the "Submit this offer" button:


Offer details
Detailed offer

Opportunity Name : Test Restitution 2
Customer : Dhia
From where we heard about This Opportunity: Social Media
Opportunity Expected close date : 2023-07-28
Opportunity Value : High
Offer Name : Offer for Restitution 2
Offer Description : qsd qsd qsd qsd qsd qsd qsd qsd qsd qsd qsd
Offer Status : On_going
Documents :

Figure 57: Offer Details & Submit

Now the offer status will be updated from “On going” to “Submitted”, and the agent can no longer delete or update this offer.

Once the commercial agent hears back from the customer about the submitted offer, he/she has to update the offer status from “Submitted” to “Rejected” or “Accepted”:

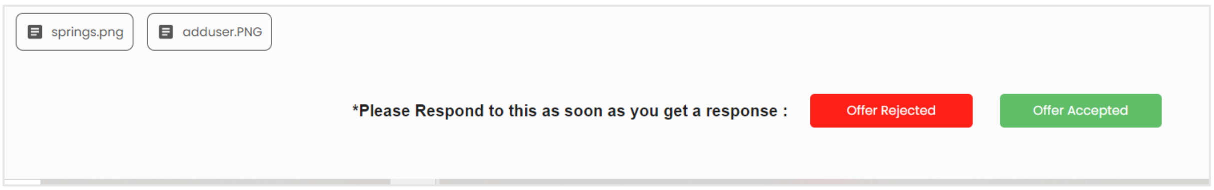
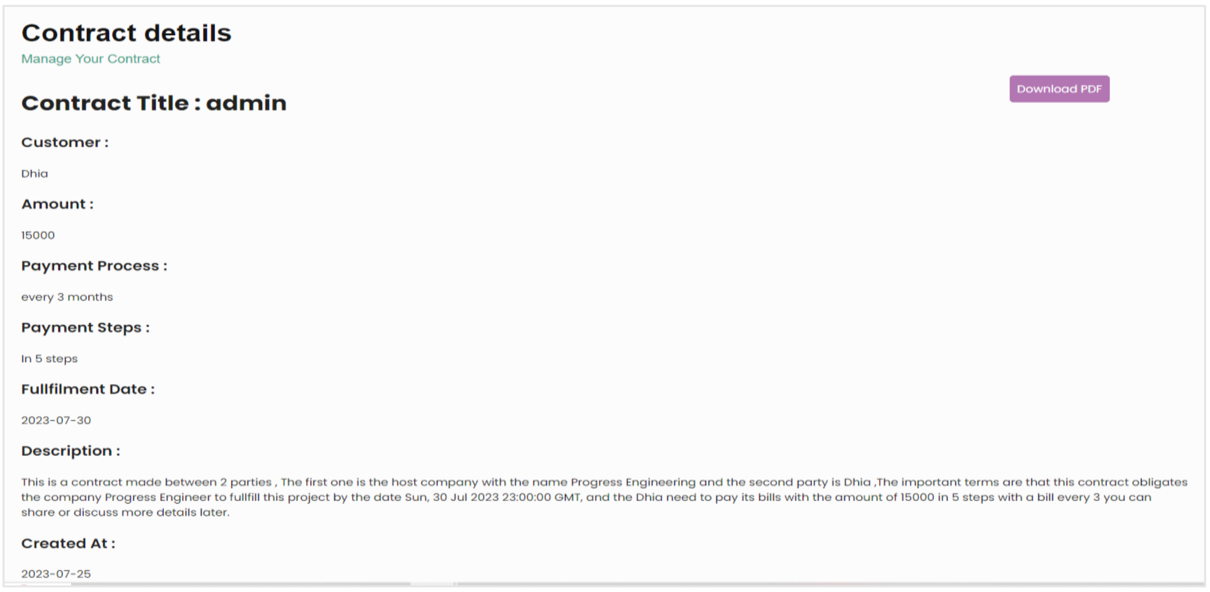

*Please Respond to this as soon as you get a response :

Figure 58: Offer result.

By changing the offer status that will automatically change the opportunity status from “Negotiating” to either “Closed Lost” or “Closed Won” depending on the offer result.
The admin will get an email notification allowing him/her to know about the offer result.
In case of the offer is accepted, the admin will start creating a new contract, a PDF file will be automatically generated giving all the details of contract:

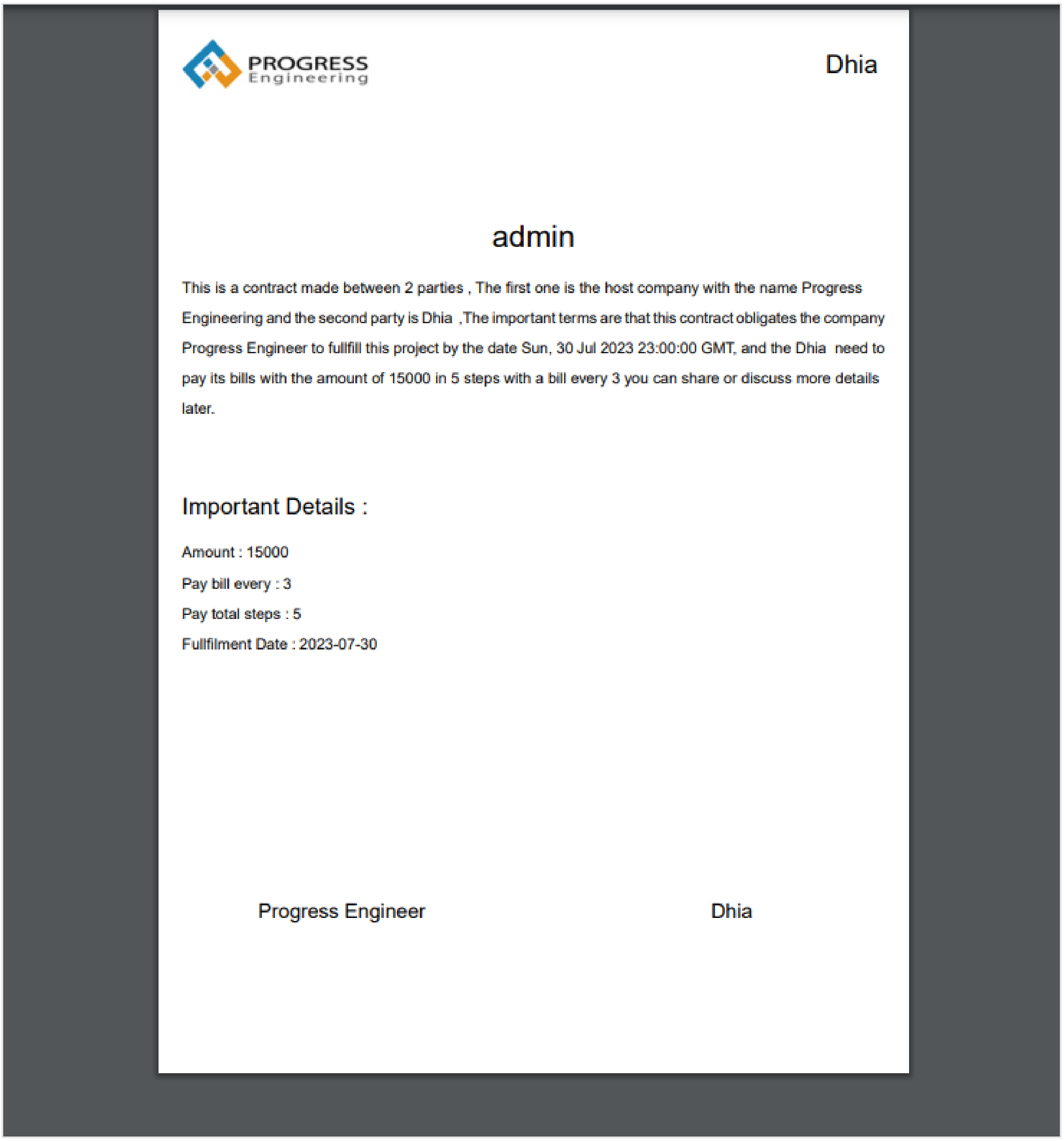

Contract details
Manage Your Contract

Contract Title : admin

Customer :
Dhia
Amount :
15000
Payment Process :
every 3 months
Payment Steps :
In 5 steps
Fullfilment Date :
2023-07-30
Description :
This is a contract made between 2 parties , The first one is the host company with the name Progress Engineering and the second party is Dhia ,The important terms are that this contract obligates the company Progress Engineer to fullfill this project by the date Sun, 30 Jul 2023 23:00:00 GMT, and the Dhia need to pay its bills with the amount of 15000 in 5 steps with a bill every 3 you can share or discuss more details later.
Created At :
2023-07-25

Figure 59: Contract Details

The contract has sole information about the amount of the company is going to gain and the payment process, and the payment steps which will help us know and calculate the amount of money generated by the company each month or each period or each year and also the total amount.
By clicking the "Download PDF" button, the PDF generated automatically will be downloaded:



PROGRESS Engineering

Dhia

admin

This is a contract made between 2 parties , The first one is the host company with the name Progress Engineering and the second party is Dhia ,The important terms are that this contract obligates the company Progress Engineer to fullfill this project by the date Sun, 30 Jul 2023 23:00:00 GMT, and the Dhia need to pay its bills with the amount of 15000 in 5 steps with a bill every 3 you can share or discuss more details later.

Important Details :

Amount : 15000

Pay bill every : 3

Pay total steps : 5

Fullfilment Date : 2023-07-30

Progress Engineer

Dhia

Figure 60: Contract PDF

Now, we covered the entire opportunity life cycle, from typing a new opportunity to take a decision on it, after it assign an agent, that commercial agent will start working on it and create an offer then submit the adequate offer, then in case of the offer being accepted, the creation of a contract with all details.

Now the next step is to create a project associated with that particular project then create necessary tasks and assign IT agents to specific tasks, but we will not be covering that and we will move to cover the “Commercial Agent scores and stats” in the next section.

5.4.6-Statistics and Scores:

In this section, we will be covering the statistics and the scores of the commercial agent, we will be starting by giving a chart that gives insight on how many opportunities agents closed successfully each month throughout the year as the figure below shows:

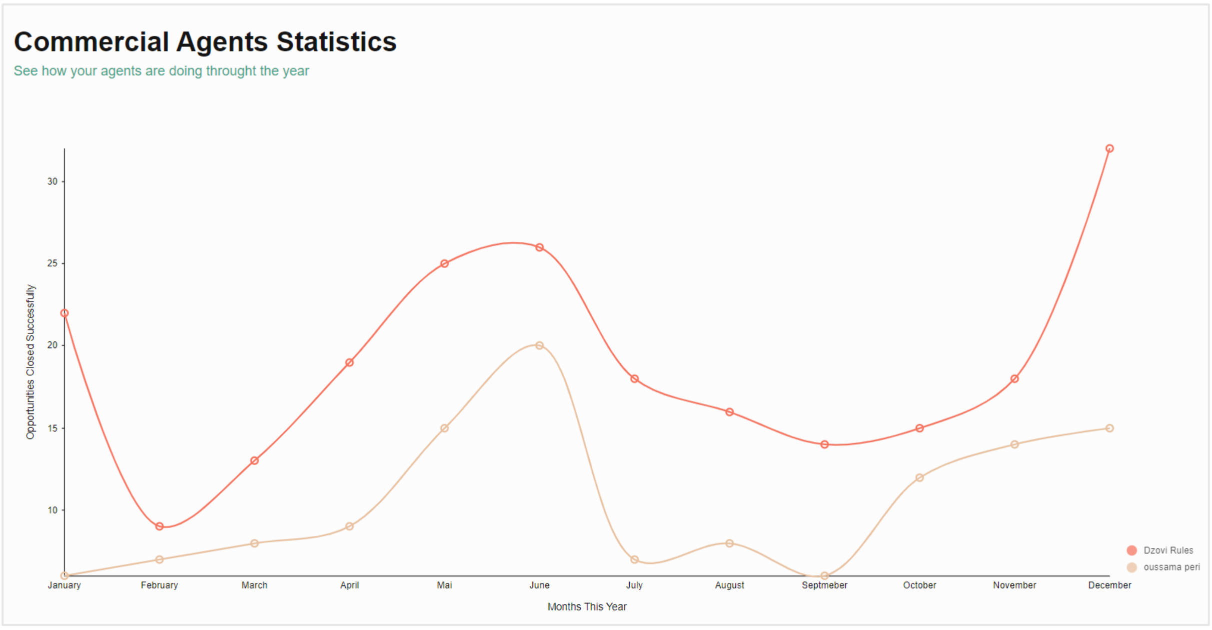


Figure 61: Statistics chart

This chart gives a very detailed comparison between commercial agents and who performed the most during the year, which gives the admins an insight on who to assign for the high value opportunities, and who to count on if necessary, in case they want to change the assignment from an agent to another.

Although this chart works quite well as a comparison chart, it really doesn’t give much insight on an individual commercial agent performance that’s why we have another chart that can give detailed information on how a certain commercial agent is performing, at first you will see 2 headers giving the agent overall score and the agent success rate, for first you would think that these are the same but in fact they are not, let’s take the case of the agent success rate, the formula to achieve that is simple as follows:

(number of opportunities closed successfully)*100 / (total opportunity closed)

This formula is quiet important, but it will not provide good insights because of one simple reason : someone closed successfully 60 opportunities out of 100 closed has 60% success rate with let’s say 30 high value 20 mid value and 10 low value, and there is another agent that has

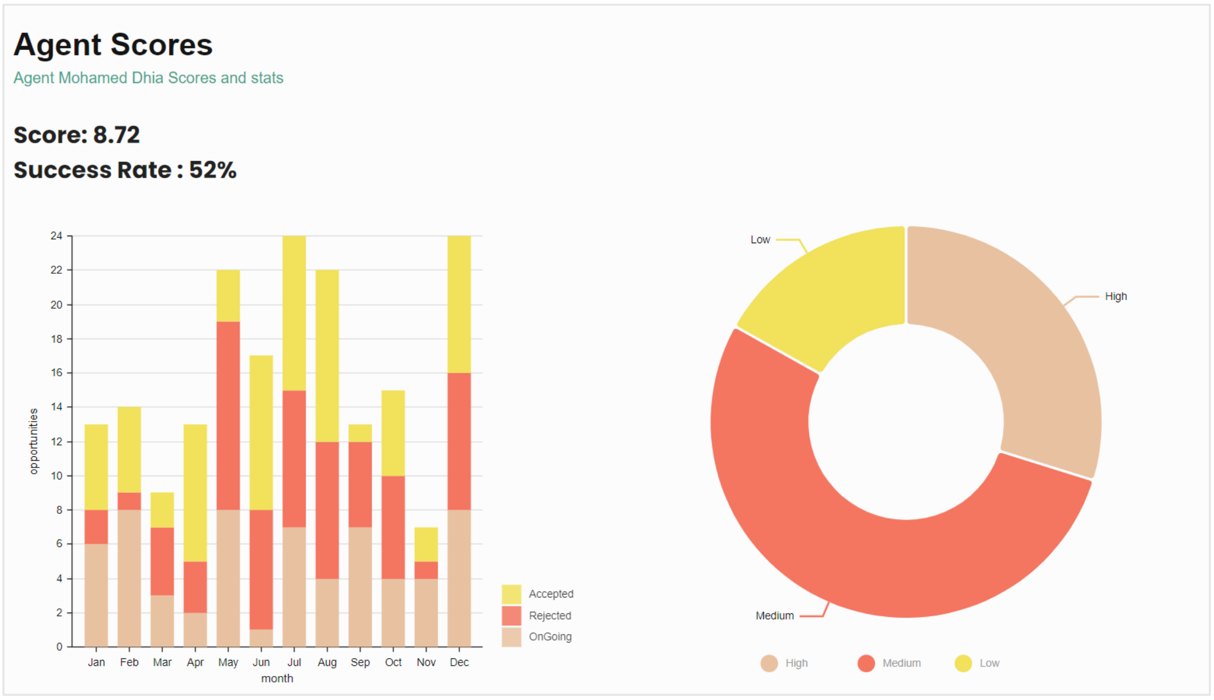
also 60% success rate but 50 of them low value and 8 mid and only 2 high, then those 2 agents can't possibly have the same performance.
That's why we have the agent overall score that has its own different formula as follows :

Sum_of [((opportunity closed successfully)*value) / Sum_of (opportunity closed*value)]*10

The value is as follows, if an opportunity value is high than its 3, if its mid its 2 and if its low it is 1.
Let's assume that the closed lost opportunities are the same for both agent with 10 of them value high, 20 value mid and 10 value low.
Let's now apply this to the previous example:

Agent1: [(30*3+20*2+10*1)/(40*3+40*2+20*1)]*10 = 6.63
Agent2: [(2*3+8*2+50*1)/(12*3+28*2+60*1)]*10 = 4.73

So now the admin can distinguish between those 2 agents on who is better.


Agent Scores
Agent Mohamed Dhia Scores and stats

Score: 8.72
Success Rate : 52%

Figure 62: Commercial agent score

The first chart is how many opportunities a certain agent is assigned to per month with the indication of how many he closed successfully and how many he lost and how many that are still on going with the negotiating phase.

The second chart is just an overview of all the opportunities that got closed successfully by that agent by value (Low, Mid, High)

With all of these the admins have a clear vision about the performance of each commercial agent and they can easily pick who is right for a certain job.

Conclusion:

This chapter has provided a comprehensive overview of the realization phase of our application development journey, with a specific focus on the implementation of the DevOps pipeline. Throughout this chapter, we have explored the key steps involved in setting up the pipeline, from version control and continuous integration to automated testing and deployment.

Additionally, we have supplemented our discussion with relevant screenshots of the application at various stages of development. These screenshots offer a visual representation of the progress made and serve as a valuable reference for understanding the evolution of our application.

General conclusion

At this end of study project, the main goal was to complete building an entire application that helps “Progress Engineering” deal with some complex problems related to business opportunities and how to deal with them.

In this comprehensive report, we have embarked on a journey through the various phases of our Business Opportunity Management Application project. Beginning with the establishment of the project scope, where we defined the boundaries and objectives, we stated the problem, then talked about some existing products and their main disadvantages then we talked about our proposed solution. We moved on to outline the project requirements, from providing the readers with all the actors, functional requirements and even non-functional requirements ensuring a clear roadmap for development. The exploration of the technical environment shed light on the tools and technologies that would underpin our application's functionality. Subsequently, the design and architecture phase provided the structural foundation, we talked in this chapter about the logical architecture, the cloud architecture, the class diagram and shed some lights on the most important scenarios. Finally, in the realization phase, we implemented a robust DevOps pipeline to streamline development and presented visual snapshots of our application's evolution.

Throughout this report, we have underscored the significance of each phase and its role in shaping our CRM-like Business Opportunity Management Application. As we conclude this journey, it is evident that meticulous planning, attention to detail, and a commitment to best practices in software development have been instrumental in our progress. Looking ahead, this holistic approach will continue to guide us towards delivering a valuable and efficient tool for managing opportunities, meeting the needs of the company and contributing to organizational success.

I will dedicate my final words to talk about the perspectives and the next goals of the application, for now it is just an internal application that “Progress Engineering” start using, however, we designed the database and the backend for it to be a SaaS (Software as a Service) application providing a clear use case and solving problems that every IT company is facing nowadays, I really can’t wait for it to go live and get a lot of success.

References:

R1:https://www.businessnewsdaily.com/4987-what-is-agile-scrum-methodology.html

R2:https://www.ibm.com/topics/java-spring-boot

R3:https://docs.spring.io/spring-security/reference/index.html

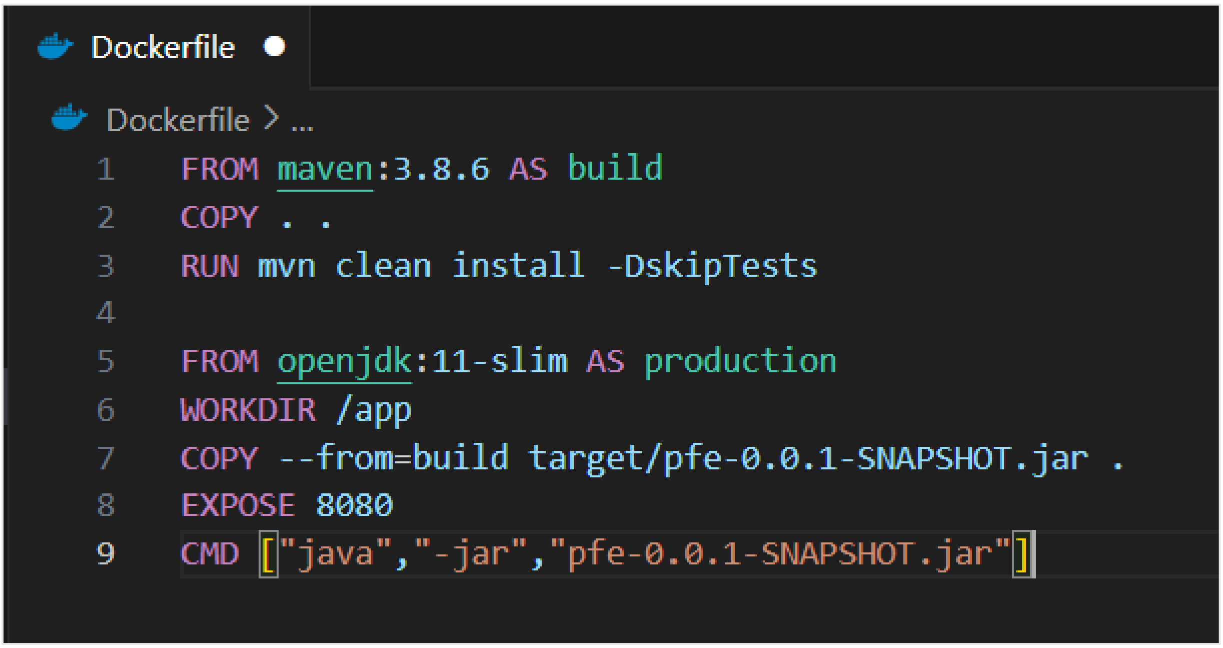
R4:https://docs.aws.amazon.com/

R5:https://aws.amazon.com/devops/what-is-devops

R6:https://en.wikipedia.org/wiki/Node.js

R7: https://en.wikipedia.org/wiki/Apache_Maven

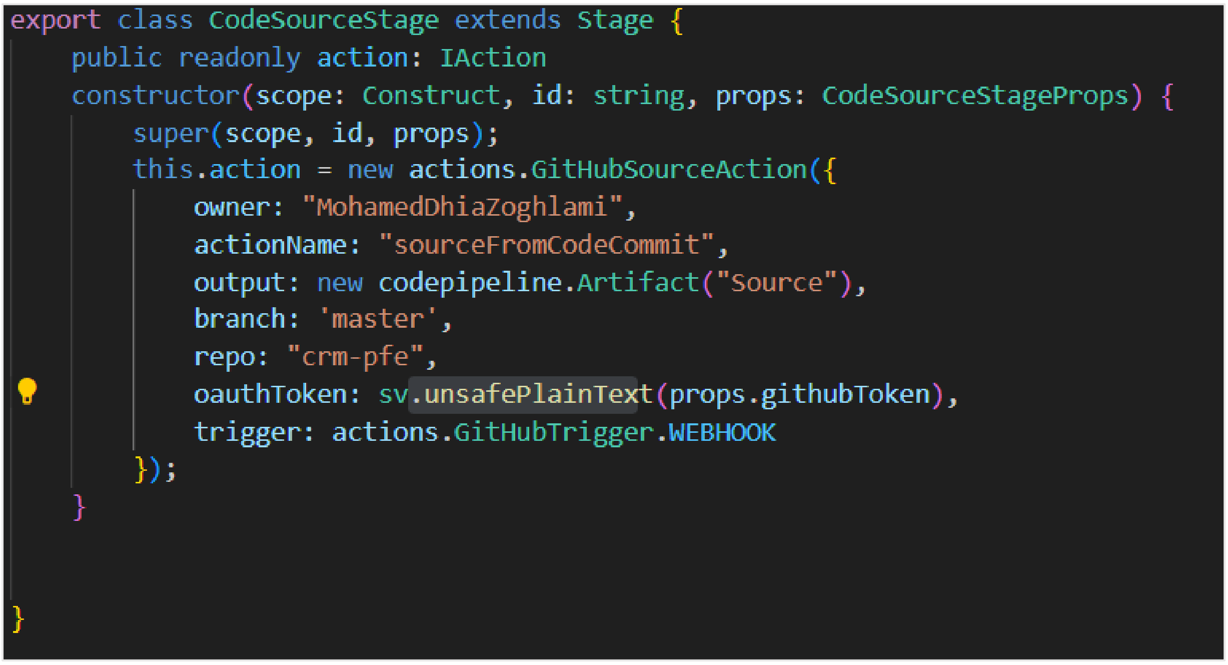
Annex A:



```
FROM maven:3.8.6 AS build
COPY . .
RUN mvn clean install -DskipTests

FROM openjdk:11-slim AS production
WORKDIR /app
COPY --from=build target/pfe-0.0.1-SNAPSHOT.jar .
EXPOSE 8080
CMD ["java","-jar","pfe-0.0.1-SNAPSHOT.jar"]
```

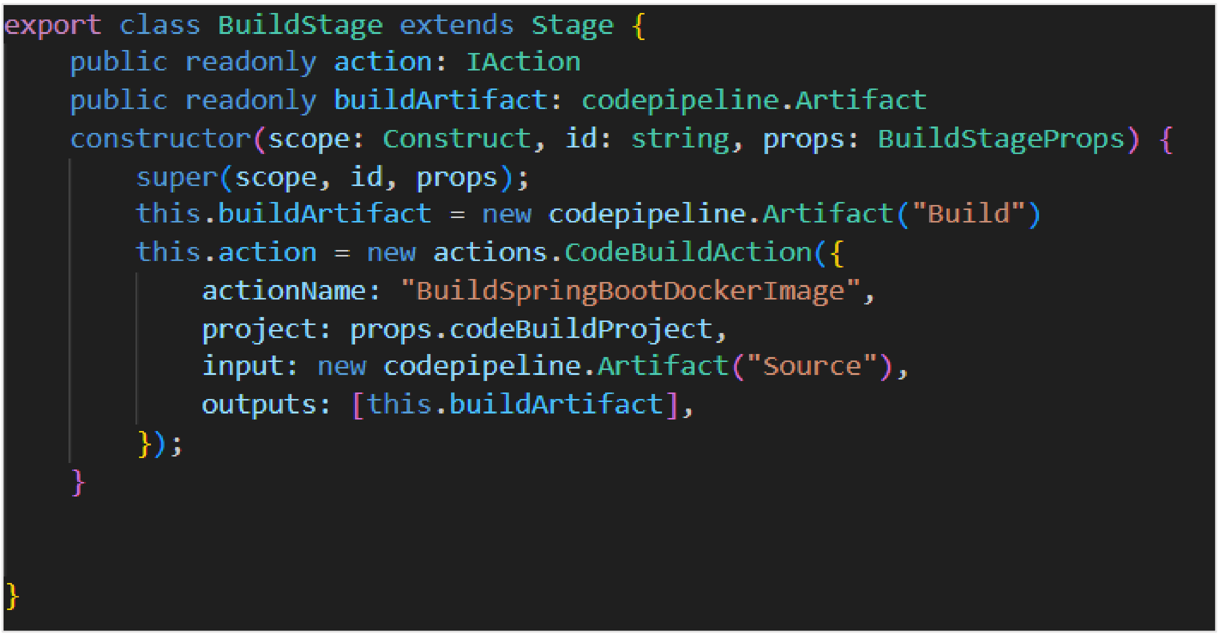
Annex B:



```
export class CodeSourceStage extends Stage {
    public readonly action: IAction
    constructor(scope: Construct, id: string, props: CodeSourceStageProps) {
        super(scope, id, props);
        this.action = new actions.GitHubSourceAction({
            owner: "MohamedDhiaZoghlami",
            actionName: "sourceFromCodeCommit",
            output: new codepipeline.Artifact("Source"),
            branch: 'master',
            repo: "crm-pfe",
            oauthToken: sv.unsafePlainText(props.githubToken),
            trigger: actions.GitHubTrigger.WEBHOOK
        });
    }


}
```

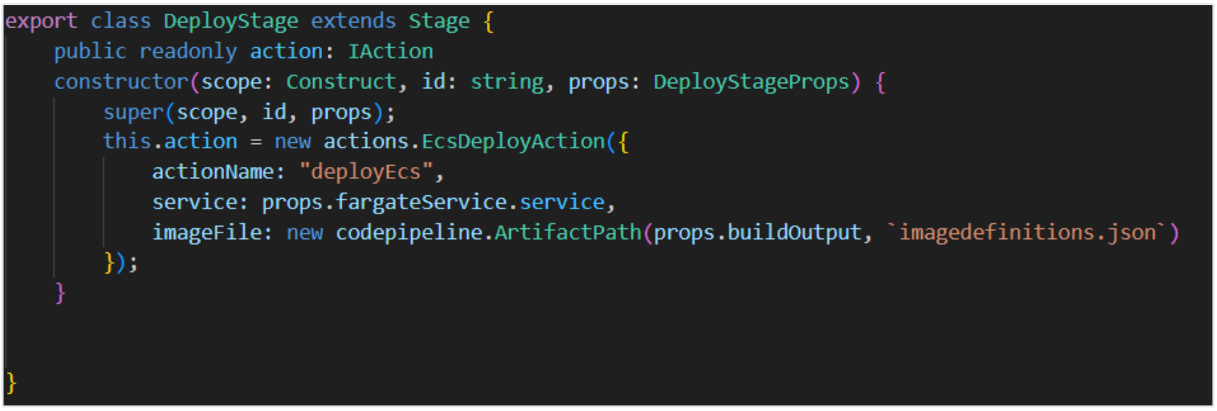
Annex C:



```
export class BuildStage extends Stage {
    public readonly action: IAction
    public readonly buildArtifact: codepipeline.Artifact
    constructor(scope: Construct, id: string, props: BuildStageProps) {
        super(scope, id, props);
        this.buildArtifact = new codepipeline.Artifact("Build")
        this.action = new actions.CodeBuildAction({
            actionName: "BuildSpringBootDockerImage",
            project: props.codeBuildProject,
            input: new codepipeline.Artifact("Source"),
            outputs: [this.buildArtifact],
        });
    }


}
```

Annex D:



```
export class DeployStage extends Stage {
    public readonly action: IAction
    constructor(scope: Construct, id: string, props: DeployStageProps) {
        super(scope, id, props);
        this.action = new actions.EcsDeployAction({
            actionName: "deployEcs",
            service: props.fargateService.service,
            imageFile: new codepipeline.ArtifactPath(props.buildOutput, `imagedefinitions.json`)
        });
    }


}
```

ESPRIT SCHOOL OF ENGINEERING