VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Uyen Nguyen

STREAMLINING PROJECT MANAGEMENT

Enhancing Efficiency and Progress Tracking Through a CRM Website

Technology and Communication

2024

ACKNOWLEDGEMENTS

I extend my sincere gratitude to all those who helped me complete this thesis and the development of the CRM web application it explores.

First and foremost, I would like to express my appreciation to my thesis supervisor for his invaluable guidance, insightful feedback, and unwavering support throughout this journey. His expertise and encouragement have been instrumental in shaping the direction and quality of this research.

I am also profoundly thankful to the participants who generously shared their time and insights, providing valuable perspectives that enriched the study. Additionally, I extend my gratitude to all the friends who helped me fix the bugs and whose dedication and expertise have been essential to its success. Furthermore, I am grateful to my family and friends for their unwavering encouragement and understanding during the demanding phases of this project. Their support has been a constant source of strength and motivation.

Lastly, I acknowledge the academic and institutional resources that have facilitated this research, including libraries and research thesis library. This thesis would not have been possible without the collective contributions of these individuals and organizations, and for that, I am genuinely thankful.

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Uyen Nguyen
Title	Streamlining Project Management: Enhancing Efficiency and Progress Tracking Through a CRM Website
Year	2024
Language	English
Page	55
Name of Supervisor	Jari Töyli

As business expands, managing customer relationships alongside internal projects becomes increasingly complex. Traditionally, organizations have relied on separate systems to handle these areas, often resulting in data duplication, communication breakdowns, and inefficient processes. This thesis outlines the journey from conceptualization to deployment of a CRM website designed to address these challenges by combining customer relationship management and project management into a single, cohesive platform.

This integrated approach enables businesses to streamline operations, enhance communication, and eliminate redundancies, thereby saving time and improving overall efficiency. The CRM web application developed as part of this thesis not only supports the comprehensive monitoring of project progress and efficient distribution of tasks but also enhances customer management by providing detailed insights into customers. The research methodology includes a quantitative analysis, supplemented by case studies that demonstrate the application's impact on organizational productivity and workflow optimization.

Key aspects such as task allocation, progress tracking, customer management, sale tracking and managerial oversight are explored in depth. The system' architecture, data security measures, and the integration of dynamic management features support real-time project tracking and robust customer relationship management. The implementation of this integrated CRM system has demonstrated significant improvement in project oversight and customer management capabilities, underscoring the effectiveness of merging these two critical business functions into a powerful tool.

Keywords Java, Spring Boot, React, Redux, MySQL, CRM

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	Application programming interface
AWS	Amazon Web Service
CRM	Customer Relationship Management
CRUD	Create-Read-Update-Delete
CSS	Cascading Style Sheets
DOM	Document Object Model
GUI/UI	Graphical/User Interface
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JDK	Java Development Kit
JSON	JavaScript Object Notation
JWT	JSON Web Token
MVC	Model-View-Controller
RPC	Remote Procedure Call
XML	Extensible Markup Language

1 INTRODUCTION

In response to the increasingly complex landscape of project management and the growing demand for streamlined task allocation and progress tracking, this project aims to develop and implement a web-based Customer Relationship Management (CRM) system. The system is specifically designed for modern organizations seeking to enhance their project management capabilities. Recognizing the inherent inefficiencies in conventional project management methodologies, such as disjointed communication and limited real-time visibility, the development of this CRM web application seeks to streamline these processes. By providing a centralized hub for task management and progress monitoring, it aims to empower managers with comprehensive oversight capabilities while fostering seamless collaboration among team members.

The primary objective of this initiative is to develop a robust web-based CRM system tailored to modern project management needs, focusing on enhancing task allocation and progress tracking capabilities. By integrating best practices from project management methodologies, software development principles, and user experience design, the system is designed to ensure effectiveness and usability. The overarching goal is to give managers real-time visibility into project progress, facilitating informed decision-making and resource allocation while promoting seamless communication and collaboration among team members. Ultimately, the project seeks to optimize task management workflows, reduce redundancy, minimize delays in project delivery, improve organizational productivity, and promote a culture of transparency and accountability within the organization.

2 BACKGROUND

In response to the increasingly complex landscape of managing both customer relationships and internal projects, this thesis has developed and deployed a web-based Customer Relationship Management (CRM) system. This integrated system is tailored to meet the needs of modern organizations that aim to streamline operations and improve efficiency across project management functions. Driven by the challenges posed by traditional separate systems, such as data duplication and inefficient processes, this CRM web application integrated project management and customer relationship management into a single, cohesive platform. By offering a centralized hub for task management, progress tracking, and customer insights, the system provides managers with strong oversight capabilities and enhances collaboration among team members. Implementing this integrated CRM solution addresses the inherent inefficiencies in conventional methodologies. It demonstrates significant improvement in project oversight and customer management, showcasing the effectiveness of merging these critical business functions.

2.1 Customer Relationship Management (CRM)

Customer Relationship Management (CRM) encompasses a blend of methodologies, strategies, and technological solutions companies employ to oversee and analyze customer interactions and data across the entire customer journey. The primary objective is to elevate customer service relationships, facilitate customer retention, and drive sales expansion. In contemporary business discourse, the term CRM predominantly refers to CRM software—a unified platform consolidating sales, marketing, and customer support activities alongside organizational processes, policies, and workforce management.

In modern business management, Customer Relationship Management (CRM) is crucial for enhancing customer satisfaction, fostering loyalty and driving growth. CRM encompasses processes, technologies, and strategies to manage customer interactions effectively. This includes lead management, onboarding, communi-

cation, support, and feedback collection. CRM strategies focus on customer segmentation, relationship building, delivering exceptional experiences, and data-driven approaches. Collaborative efforts across sales, marketing, and support teams ensure seamless interactions-effective CRM practices in building long-term relationships, driving growth, and maintaining competitiveness.

Figure 1 illustrates the positive impact of CRM on business performance, showing substantial improvements in lead conversion rates, revenue per salesperson, customer retention, sales cycle duration, and sales and marketing costs. The data is collected through an internal survey of Zoho.

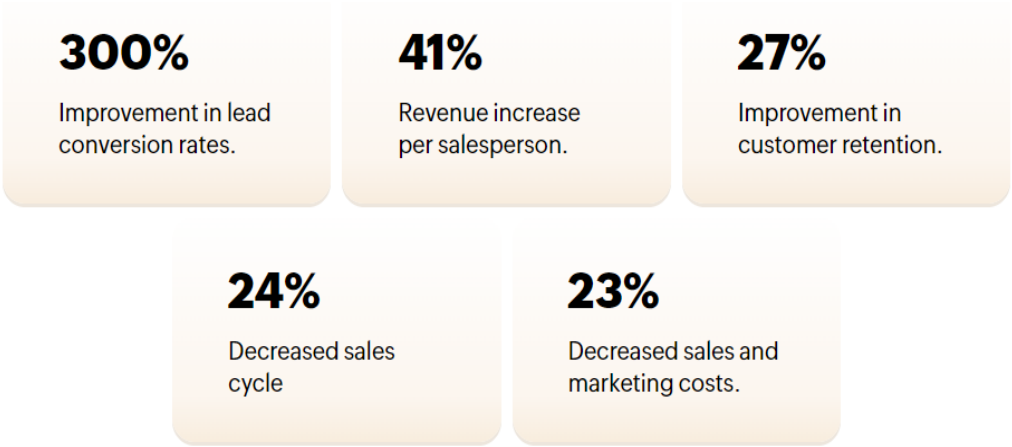


Figure 1. Growth metrics using CRM (What does CRM software do?, n.d.)

CRM project management offers a structured approach to coordinating essential functions and tasks within CRM initiatives. It aligns strategies with business objectives, optimizes resource allocation, mitigates risks, and encourages continuous improvement. Built on CRM fundamentals based on customer needs and data analysis, CRM project management integrates these principles into a project management framework. This integration allows project managers to utilize CRM insights, technologies, and cross-functional collaboration to successfully implement and oversee CRM initiatives, ensuring organizational success in a competitive environment. Figure 2 below describes a flowchart illustrating the stages of a Project-Based CRM system, from relationship building and bid invitations through bid tracking to project execution.

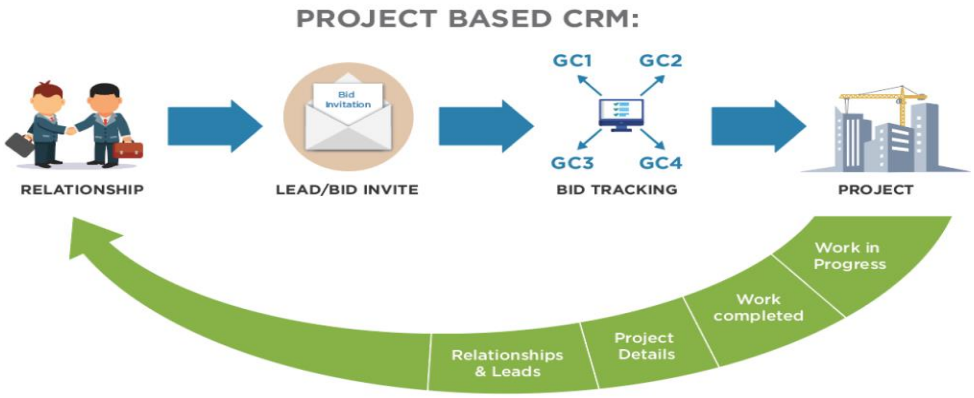


Figure 2. CRM integrated with Project Management (Lorek, 2018)

3 SYSTEM ARCHITECTURE

3.1 System Requirements

The Customer Relationship Management (CRM) system must meet several functional requirements to effectively support its intended operational objectives. First, the system must provide secure login capabilities for all users, including administrators, project managers, and employees. Once authenticated, the user can access various functionalities based on their roles.

The system must facilitate comprehensive user management by allowing the addition, editing, deletion, and viewing of user accounts. In addition, it should enable the management of client information and roles, ensuring that administrators can add and update this data as needed. The system will also support project management by allowing users to create, modify, delete, and view project-related information.

Task Management is another critical requirement, the system must enable the delegation and tracking of tasks within projects, ensuring that project managers can assign tasks to team members and monitor their progress. Additionally, the system should provide tools for tracking the work progress of users on various projects, allowing project managers to monitor and manage tasks effectively.

To support decision-making and oversight, generating aggregated statistics and reports on the progress of projects shall be a function of the system. This functionality will enable administrators to gauge overall productivity, identify potential bottlenecks, and make informed decisions to steer project directions effectively.

The CRM systems must also meet several non-functional requirements to ensure reliability, security, and performance. The system shall employ encryption methodologies for storing passwords, ensuring that user credentials remain secure even in the event of unauthorized database access. Furthermore, all system data

shall be accessible only after a successful login, safeguarding sensitive information from unauthorized users. Role-Based Access Control (RBAC) is implemented to enhance data security and ensure appropriate access levels within the system, assigning system permissions based on the user's role within the organization.

3.2 System Analysis

The system analysis aims to thoroughly understand the requirements and constraints of the new CRM system. This detailed analysis facilitates the design of a system architecture that meets the specific needs of users and aligns with the organization's overall objectives.

Table 1 outlines the responsibilities and system interactions for different user roles within an organization, including Administrators, Project Managers, and Employees.

Table 1. User Role Analysis

User	**Administrator (ADMIN)**	**Project Manager (LEADER)**	**Employee (MEMBER)**
Responsibility	-Managing user accounts and permissions. -Adding, editing, and deleting employee, client, role, and project information.	- Creating and managing projects. - Assign tasks to team members. -Monitoring project progress and performance, client for whom responsible.	- Updating work progress by updating task status - Viewing assigned tasks
System Interaction	- Full access to all system functionalities. - Ability to configure system settings and preferences.	- Access to project management tools. - Ability to view project-specific	- Access personal task dashboard - Access to their profile page

Table 2 provides an overview of the critical modules within a management system, detailing the features and purposes of each module. The modules covered include User Management, Project Management, Task Management, Role Management, and Client Management. Each section highlights the specific functionalities such as user authentication, project tracking, task assignment, role modification, and client data management, explaining how these contribute to the overall operational efficiency and strategic execution within the organization.

Table 2. Module Analysis

Modules	**User Management**	**Project Management**	**Task Management**	**Role Management**	**Client Management**	**Event Management**
Features	Authentication	View, Creation, Editing and deleting	View, Creation, Editing and deleting	Role assignment.	View, Creation, Editing and deleting	View, Creation, Editing and deleting
	Role-base access	Tracking	User assignment		Lead Management	
	Profile Management	Task implement	Tracking		Assign Project	
Purpose	Manage accounts and permissions.	Effective planning, execution, monitoring	Organize work, assign tasks	Enforce role permissions	Manage client data	Manage events

4 RELEVANT TECHNOLOGY

This section provides an overview of the technologies used in the development of the CRM system, focusing on the frontend technologies that contribute to the system's architecture and enhance user interaction and experience.

4.1 Frontend Technology Used

The frontend of the CRM system is developed using several key technologies that work together to create an interactive and user-friendly interface. These technologies include JavaScript, React, Redux, and Bootstrap.

JavaScript is a programming language that developers use to make interactive web pages. From refreshing social media feeds to displaying animations and interactive maps, JavaScript functions can improve a website's user experience. As a client-side scripting language, it is one of the core technologies of the World Wide Web. (What is JavaScript?, n.d.)

ReactJS is an open-source library that is utilized for building up the UIs explicitly for single-page applications. ReactJS empowers software engineers to make immense web applications that can use data and can change after some time without reloading the page. Along these lines, React has a savvy diffing calculation that it uses to just recover in its DOM hub what should be recovered while it keeps all that else with no guarantees. The utilization of reusable parts gives a simple method of building the application. The brilliant thought of React moreover makes arranging UI reliable and takes a huge weight off from programmers so they could focus on more huge limits and business reasoning. Respond likewise does not force a particular method to play out a specific undertaking. It gives a rich arrangement of libraries from which clients can choose to play out a specific undertaking. Lifecycle strategies and React Hooks are other significant highlights that handle the arrangement of occasions that get called during the lifecycle of a segment (Prateek et al., 2020).

Utilizing React's capabilities, this CRM project leverages the framework to build complex user interfaces by breaking the code into manageable components, simplifying code management.

ReactJS has a rich ecosystem that includes must-have tools like Redux or Flux. Redux is a predictable state container for JavaScript applications. It is a standalone library, but it is used most often as a state management layer with React. Like Flux, its major goal is to bring consistency and predictability to the data in applications. Redux divides the responsibilities of state management into a few separate units (Garreau and Fauro, 2018)

Figure 3 illustrates how Redux operates, actions are triggered by user interactions, the store reducers to compute a new state, and the UI displays these updated values.

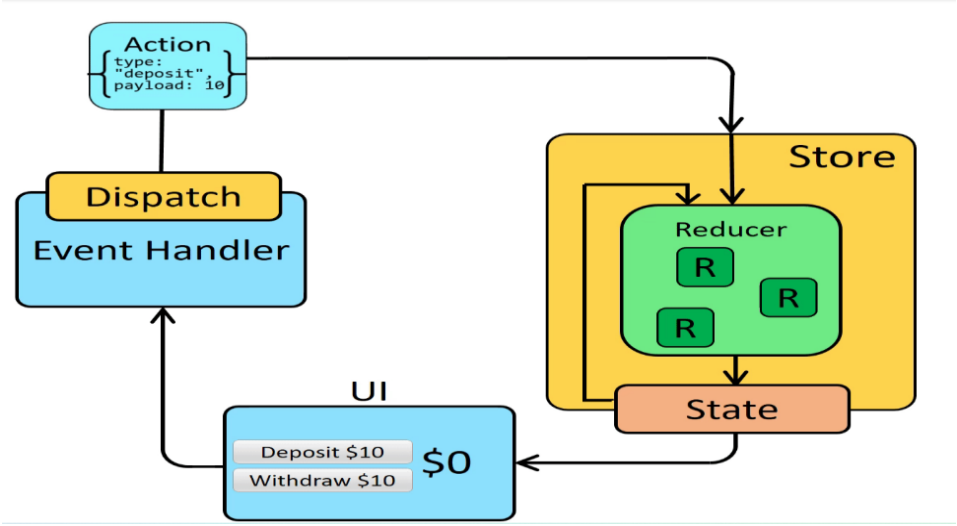


Figure 3. Redux data flow (Abramov D. et al., 2023)

Bootstrap is a free, open-source frontend development framework for the creation of websites and web applications. Designed to enable responsive development of mobile-first websites, Bootstrap provides a collection of syntax for template designs (Zola, 2022)

As a framework, Bootstrap encompasses the fundamentals of responsive web development, simplifying the process for developers who can simply integrate code into a pre-established grid system. Bootstrap is constructed using HTML, CSS, and JavaScript. By leveraging Bootstrap, web developers can accelerate website development significantly, bypassing the need to concern themselves with fundamental commands and functions.

4.2 Backend Technology Used

The backend of the CRM system is built using robust technologies that ensure reliable performance, security, and scalability. These technologies include the Spring Framework/Spring Boot, Java, RESTful APIs, MySQL, and AWS S3 storage

Java is a modern programming language for high-level programming, open-source, and easy to learn. Using Java is widespread, with a well-developed ecosystem, numerous libraries, and technology frameworks, and accordingly, it provides an infinite number of perspectives for development. Java combines paradigms of procedural, object-oriented, and functional programming in a modern way with easy syntax, although it is more descriptive than Python and JavaScript. (Svetlin Nakov & Team, 2021)

The Spring Framework fundamentally serves as a sophisticated container for dependency injection, enhanced with several utility features, such as database access, proxies, aspect-oriented programming, RPC, and a web MVC framework. These features are part of a comprehensive programming and configuration model that facilitates the development of modern Java-based enterprise applications across any deployment platform. According to the Spring documentation on spring.io, at its core, Spring focuses on providing robust infrastructural support at the application level, managing the complex "plumbing" of enterprise applications so developers can concentrate on application-level business logic. This infrastructure focus allows teams to work more efficiently without being bogged down by the specifics of deployment environments, making the development process faster and more streamlined. Spring’s approach ensures that enterprise

applications have strong support for foundational operations, allowing developers to focus on core business objectives without unnecessary ties to specific deployment environments. (Rod et al., 2008).

Figure 4 provides an overview of the Spring Framework architecture, highlighting its core components: DAO, ORM, JEE, Web, AOP, and Core, which serve various functions from data access and transaction management to web development and aspect-oriented programming.

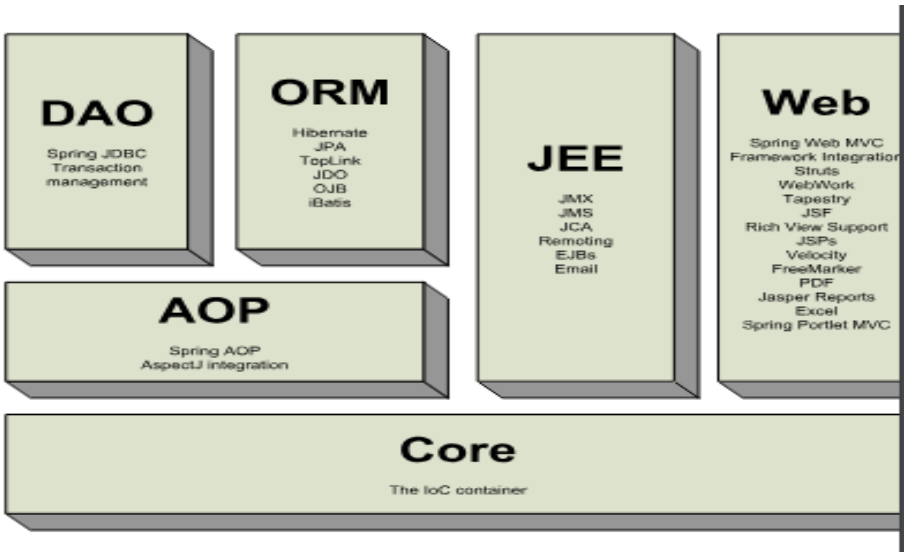


Figure 4. Spring Framework Architecture (Rod et al., 2008)

REST API is an architectural style that allows for communication between different systems over the internet. A RESTful API is designed to be simple, scalable, and maintainable, with a set of constraints that define how the system should behave. (Masse, 2021)

API developers can design APIs using several different architectures. APIs that follow the REST architectural style are called REST APIs. Web services that implement REST architecture are called RESTful web services. The term RESTful API generally refers to RESTful web APIs. However, the terms REST API and RESTful API can be used interchangeably (What is a RESTful API?, n.d.)

The primary function of a RESTful API is the same as browsing the internet. The client contacts the server by using the API when a resource is required. Figure 5 below illustrates how Rest API works, 5 showcasing the interaction between consumers (clients) and service providers (servers). Clients send HTTP requests to REST APIs on the web server, which interacts with the database and returns HTTP responses.

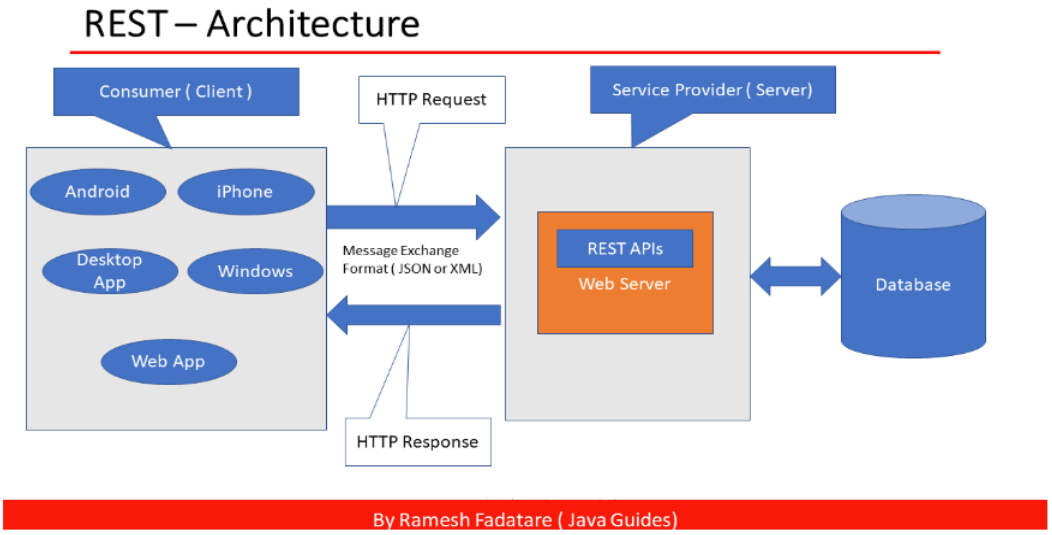


Figure 5. REST Architecture (Fadatare, n.d.)

MySQL, developed by Oracle, operates as an open-source relational database management system, employing Structured Query Language (SQL) for data manipulation within its structure. Data in MySQL is structured across multiple tables, streamlining storage and organization. These tables are interconnected through predefined relationships. Compatible with various operating systems like macOS, Linux, FreeBSD, and Windows, MySQL comprises two key components: a data management server and client interfaces responsible for executing tasks such as data modification and report generation. Using the Client-Server Architecture model, users access resources via a central server through network services facilitated by client computers. Clients interact with the server through a graphical user interface (GUI), with the server responding to requests promptly upon matching instructions. Figure 6 below shows the interaction between the client and when using MySQL. The database management system (DBMS) com-

municates with the database using SQL, facilitating data exchange with various applications such as enterprise resource planning (ERP), web applications, dynamic websites, and other applications, which are accessed by the DBMS end user (client).

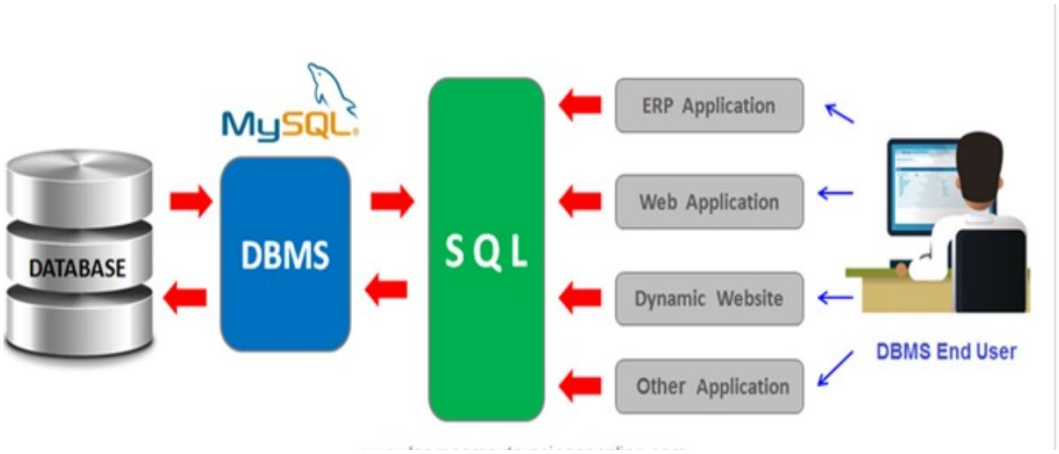


Figure 6. How client and server interact in MySQL (Hausman, 2023)

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 to optimize, organize, and configure access to data, meeting specific business, organizational, and compliance requirements. (Cloud Object Storage - Amazon S3, n.d.)

5 SYSTEM ARCHITECTURE DIAGRAM

5.1 System Functional Diagram

Figure 7 illustrates a hierarchical access and control diagram for a CRM website, detailing the different levels of permissions and functionalities assigned to various user roles such as Administrator, Project Manager, and Member/Staff. The diagram shows how each role interacts with different modules, including Role Management, User Management, Client Management, Project Management, Task Management, and Statistics.

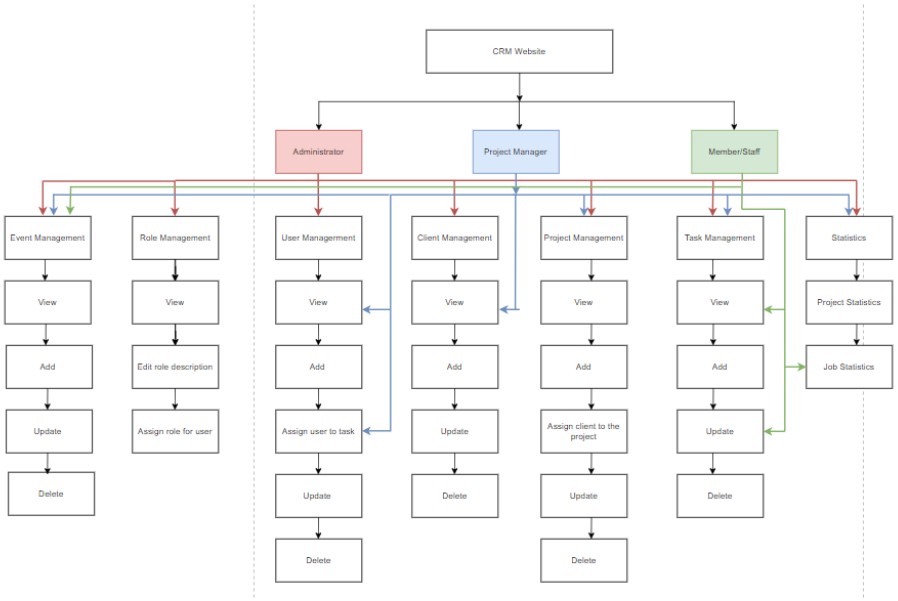


Figure 7. System functional diagram.

5.2 General Use Case Diagram

A use case describes how a system interacts with its users, typically represented as a sequence of steps or actions that the system performs to achieve a specific goal. It outlines the functionality of the system from the perspective of an external user or actor.

Figure 8 below represents a use case diagram for a CRM system, showing the interactions between various user roles and system functionalities. The central us-

ers are Admin, Project Manager, and Member, all interacting through a standard "User" interface that includes a login mechanism.

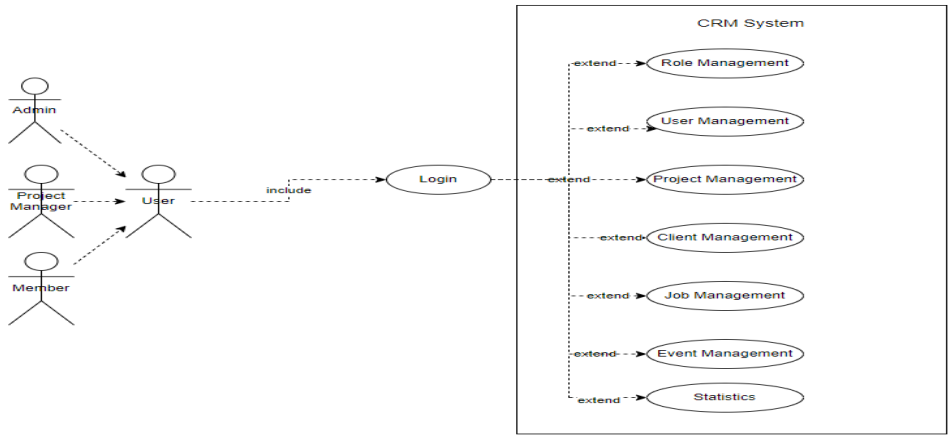


Figure 8. General use case diagram.

Once users log in, they access various functionalities based on their roles. Administrators can perform comprehensive roles and perform user management, configure the system to reflect organizational hierarchies, maintain client records, and oversee project statistics to inform decision-making and ensure smooth operation. Project managers focus on managing projects and tasks, creating, and modifying project details, assigning tasks to team members, and monitoring task progress. They also stay informed about client interactions to ensure projects meet expectations and deadlines. Members update their work progress, access their task dashboards, track assigned tasks, view work statistics, and report task completion, promoting accountability and alignment with project goals.

The system design ensures secure and role-appropriate interactions, leveraging a centralized login process to streamline access while maintaining robust data security and role-based access control. This structured interaction flow helps each user efficiently perform their duties, enhancing the overall effectiveness of the CRM system in managing employee work and project progress.

6 IMPLEMENTATION AND APPROACH

6.1 Design Database

Based on the insights gained from the system analysis and system architecture diagrams outlined earlier, the design of the database tables is significantly informed. In particular, this facilitates the structured organization of all tables within the system.

The Clients Table stores vital information about clients. It includes fields for the client's information, such as full name, company name, and phone number. The client details are added as needed to support business operations and client management. The primary key for this table is id, which ensures that each client's record is uniquely identifiable.

The Users Table manages user account data necessary for system login. It includes email and password fields, which are essential for authenticating users. Both email and password are required when creating a new user account. For security, the password is encrypted before storage in the database. The field labeled username stores the user's full name. Additional details such as phone numbers and images are also maintained to provide more comprehensive user profiles.

The Roles Tables catalogs all the roles available within the system, with each role possessing distinct access privileges. Roles are crucial for assigning permission to users and ensuring appropriate access control. The relationship between users and roles is many-to-many, necessitating an intermediary table named users_roles. It stores the users' foreign keys and roles tables to link them effectively. The setup facilitates flexible and secure permission management across the system.

The Projects table is designed to record comprehensive information about each project, including the project's name, start date and end date, status and deal. The id is the primary key, ensuring each project record is unique. Additionally,

the client_id field identifies the client associated with the project, clarifying client ownership. The originator_id indicates the project's originator, typically the project manager, providing clarity on project leadership and responsibility. This structure supports effective project management and client relationship tracking.

The Tasks table is essential for managing and organizing project tasks. It stores comprehensive details about each task, including task's name, start date, end date, and status. The implementer_id is a critical field identifying the individual responsible for the task and facilitating accountability and oversight. The relationship between tasks and users(implementers) is many-to-many, necessitating an intermediary table, implementers_tasks, which contains the foreign keys from both the Tasks and Users tables to link them. Additionally, project_id in the Tasks table indicates which project the task belongs to, enabling project-specific task management. The fields for date and status are beneficial for implementers and managers to monitor and track task progress effectively.

The Events table manages event details, including title, date time, and category. It establishes a one-to-many relationship with the users table, indicated by including the owner_id as a foreign key.

Figure 9 provides an oversight visualization of the database schema, detailing each entity, its fields, and the relationships between entities. The schema includes several tables, as mentioned above. The figure serves as an essential tool for understanding the structure of the database and the flow of information between different parts of the system, ensuring data consistency and integrity across the database.

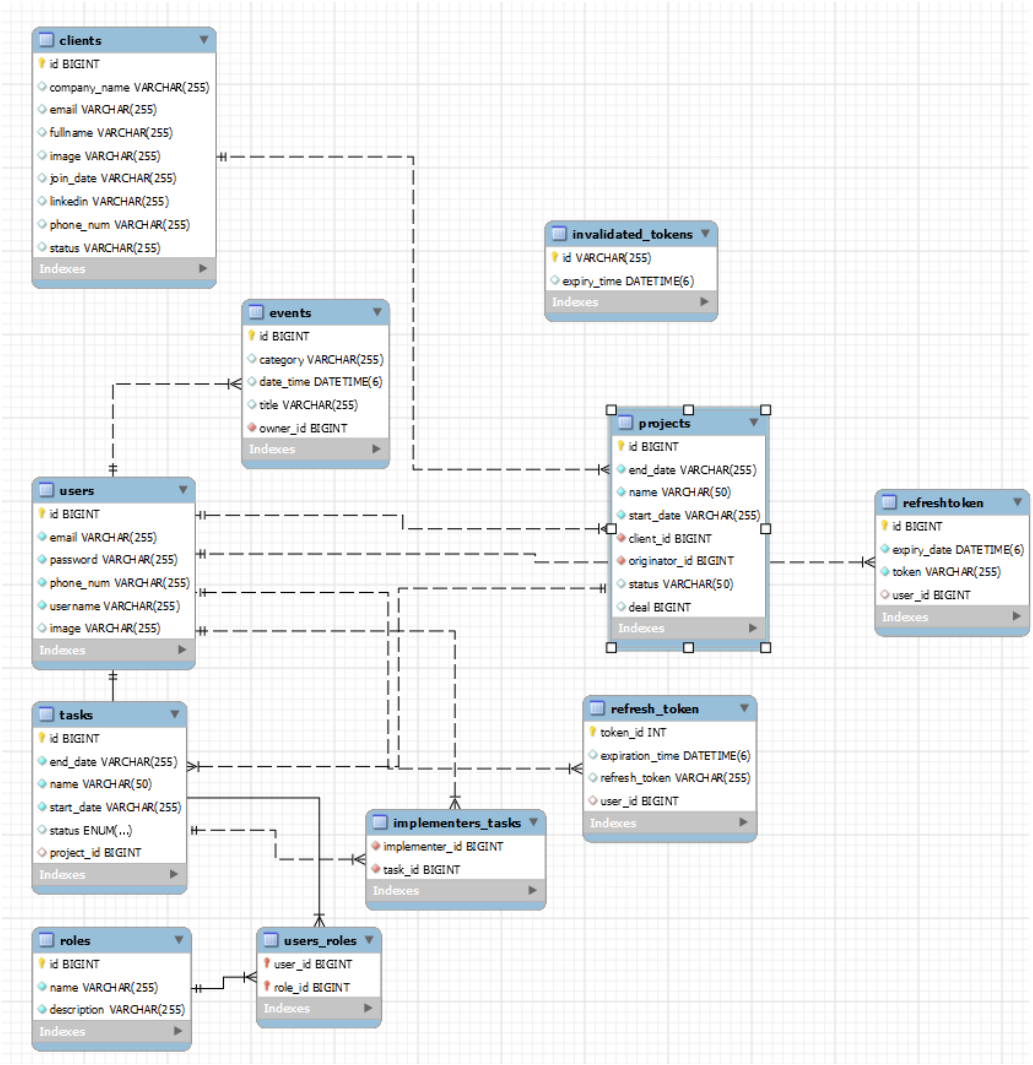


Figure 9. The Enhanced Entity-Relationship Diagram (EER)

6.2 Backend and Frontend Setup

For the backend, A CRM database was established on a MySQL localhost and it was configured it using MySQL Workbench to commence the technical setup. Subsequently, the project package was initialized at https://start.spring.io/, where the necessary dependencies were selected. IntelliJ IDEA was selected as the Integrated Development Environment (IDE) for the development of the backend. Figure 10 shows the structure of the project when it is opened in IntelliJ. The pom file contains dependencies added to the project.

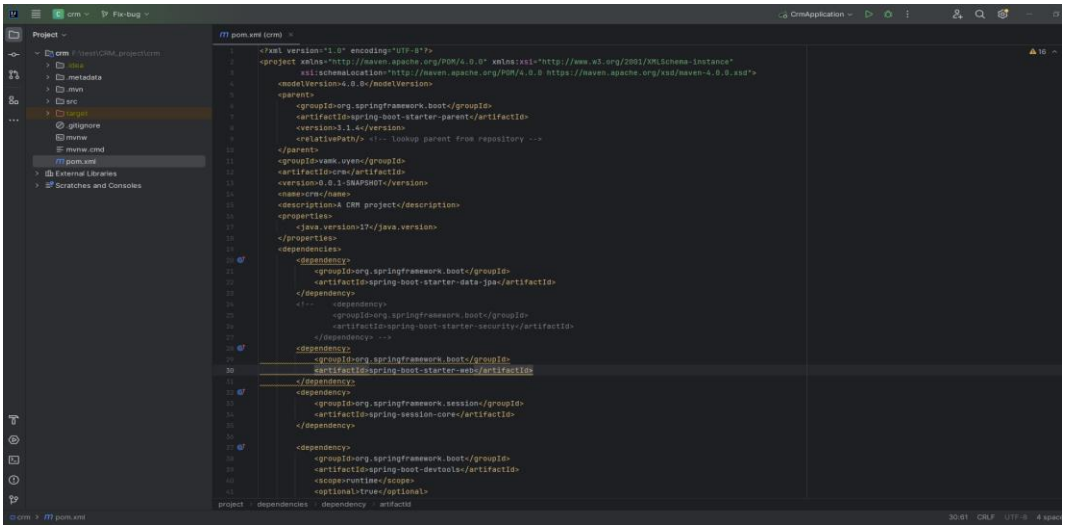


Figure 10. Open project package in IntelliJ.

For the frontend, Visual Studio Code for React development is an ideal code editor for implementing React, it is a free and open-source code editor. It is widely used by developers for web development because VS code provides a rich set of features designed to enhance productivity and streamline the development process with some key features: IntelliSense, integrated terminal, extensions, and version control. Figure 11 shows the workspace visualization in Visual Studio Code, as well as the structure of the project. All dependencies required for the project, such as React libraries and other packages, are in package.json file.

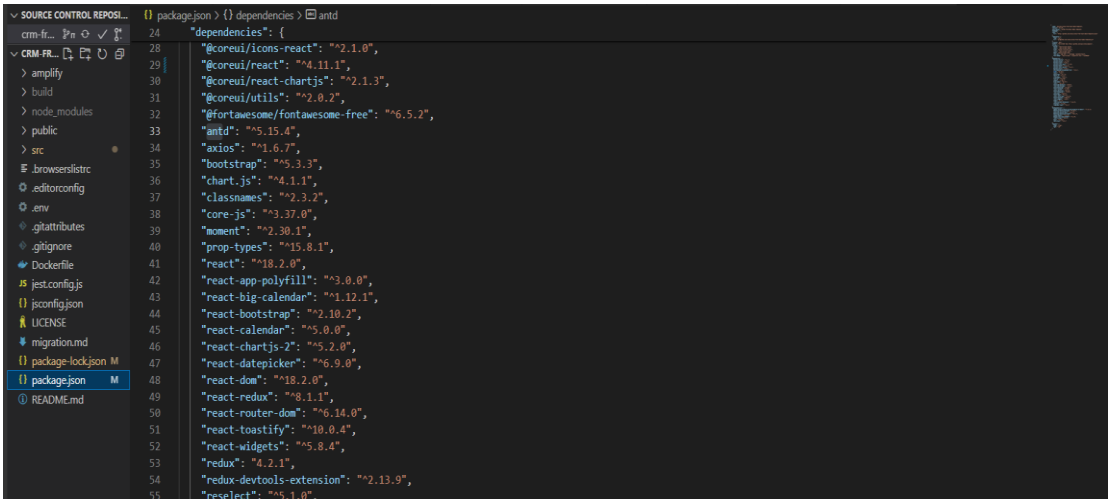


Figure 11. Frontend workspace

6.3 Version Control

The primary purpose of using a version control system in this project is to act as a backup for the project, safeguarding against data loss or corruption on local machines. Additionally, they offer a comprehensive history of changes made to the project, which can be valuable for auditing purposes or understanding past decisions. Figures 12 and 13, respectively, depict the repositories for the front end and back end. The git ignore is added to exclude unnecessary files or folders from being tracked in the repository.

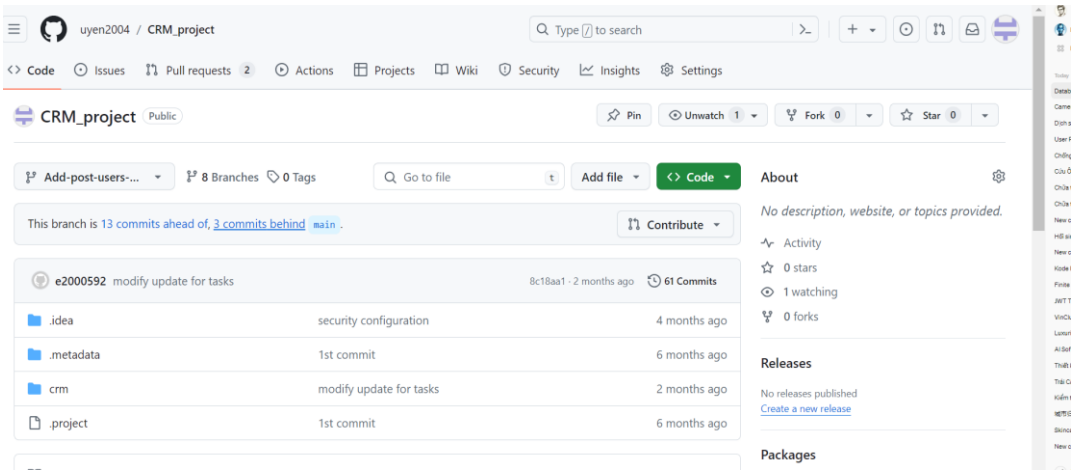


Figure 12. GitHub repository for backend.

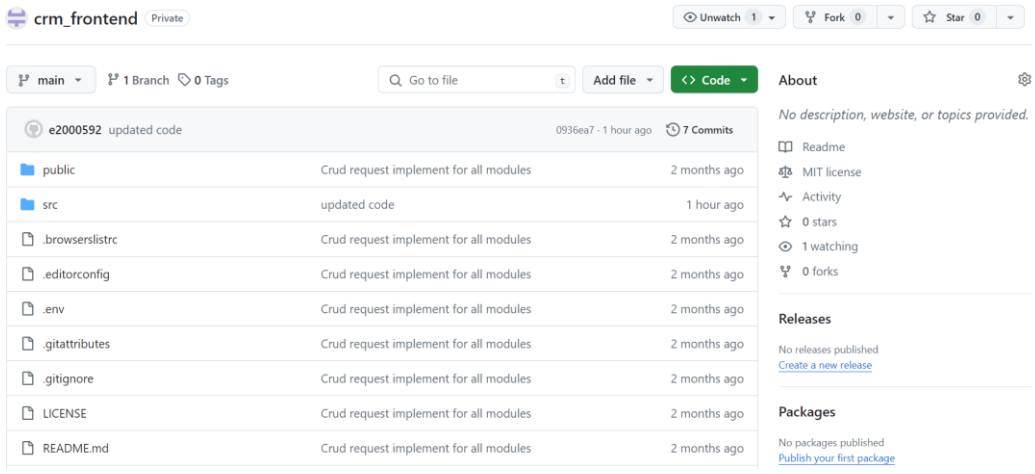


Figure 13. GitHub repository for frontend.

6.4 Postman

Postman is used to test backend API. Postman allows the sending of HTTP requests with different methods such as GET, POST, PUT, DELETE, and PATCH, and allows the posting of data as a form(key-value), text, and JSON. Postman also supports various types, including text, image, XML, and JSON. Additionally, it offers features like authorization support and the ability to modify request headers as needed.

CRM workspace is created in Postman and organized based on module management, structuring folders shown in Figure 14 below, configuring a module folder in this manner not only facilitates the logical inheritance of authorization settings from the parent folder to the service within the module (since all endpoints require authorization to access them) but also simplifies the management of APIs for each module.

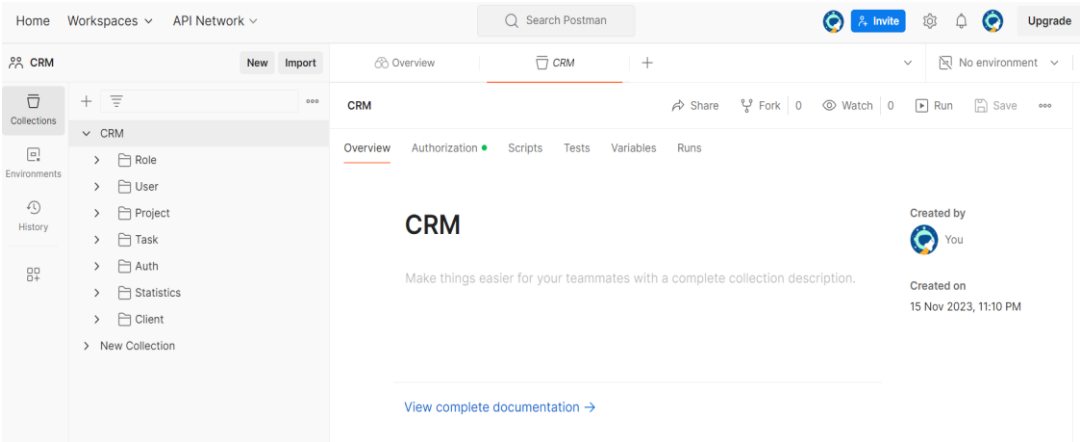


Figure 14. Create CRM workspace in Postman

6.5 Backend Logic

The backend, also known as the server side, refers to the part of the application that manages data storage, business logic, and server configuration. It is responsible for processing requests from the frontend, performing database operations, and implementing core functionalities that the user does not directly interact with.

6.5.1 RESTful API Endpoint

Given the system's stringent adherence to role-based access and its myriad features, compiling a list of endpoints streamlines the development process, preventing leaving out any functionalities and guaranteeing compliance with role-based access rules. The tables below categorize endpoints based on feature purposes within each entity module and corresponding role-based access.

Table 3 lists endpoints used for authentication, facilitating access control, and user verification within the system.

Table 3. Authentication endpoint

Method	**Endpoint**	**Role Access**	**Purpose**
POST	/api/v1/auth/login	All role	Authorize user
	/api/v1/auth/register/{roleId}	Admin	Register new user and assign role to the user
	/api/v1/auth/refreshToken	All role	Obtain new access token when token is expired
	/api/v1/logout	All role	Log out user

Table 4 presents a comprehensive list of endpoints dedicated to client management within the system. It provides an overview of accessing and managing client-related functionalities and data.

Table 4. Client Endpoint

Method	**Endpoint**	**Role Access**	**Purpose**
GET	/api/v1/clients	Admin, manager	View all clients

POST	/api/v1/clients	Admin	Add new client
PUT	/api/v1/clients/{id}	Admin	Update client
DELETE	/api/v1/clients/{id}	Admin	Delete client
POST	/api/v1/client/image	Admin	Upload client image

Similar to Table 4, Table 5 provides a detailed compilation of endpoints, tailored explicitly for project modules. It offers an organized overview of endpoints crucial for managing various aspects of project modules within the system.

Table 5. Project Endpoint

Method	**Endpoint**	**Role access**	**Purpose**
GET	/api/v1/projects	All role	View all projects
GET	/api/v1/projects/{id}	Admin, manager	View specific project
POST	/api/v1/projects	Admin, manager	Add new project
PUT	/api/v1/projects/{id}/originator/{originatorId}	Admin, manager	Update project
DELETE	/api/v1/projects/{id}	Admin, manager	Delete project

Table 6 outlines endpoints relevant to the role module, offering a comprehensive breakdown for managing user roles within the system.

Table 6. Role Endpoint

Method	**Endpoint**	**Role Access**	**Purpose**
GET	/api/v1/roles	Admin	View all roles
PUT	/api/v1/roles		Update role description
DELETE	/api/v1/roles		Delete role

Table 7 provides a list of endpoints pertinent to user management. It offers insights into accessing and manipulating user-related functionalities within the system, facilitating efficient user administration and interaction.

Table 7. User Endpoint

Method	**Endpoint**	**Role Access**	**Purpose**
GET	/api/v1/users	Admin, manager	View all users
GET	/api/v1/users/{id}	Admin	View specific user
GET	/api/v1/users/profile	All role	View user profile when login successful
POST	/api/v1/users/role/{roleId}	Admin	Add new user and assigned role to

			the user
POST	/api/v1/{userId}/uploadImage	All role	Allow user to upload image profile
PATCH	/api/v1/users/{id}	Admin	Update user
DELETE	/api/v1/users/{id}	Admin	Delete user

Table 8 provides a comprehensive of endpoints specifically designated for task module operations. It serves as a comprehensive guide for navigating and managing task-related functionalities within the system, facilitating streamlined task administration and tracking.

Table 8. Task Endpoint

Method	**Endpoint**	**Role Access**	**Purpose**
GET	/api/v1/tasks	All role	View all tasks
GET	/api/v1/tasks	All role	View specific task
POST	/api/v1/tasks/projects/{projectId}/users/{userId}	Admin, manager	Add new task and assign an implementer to the task
PUT	/tasks/id	Mem-	Update

		ber/staff	task status
PUT	/tasks/{id}/implementer/{implementerId}	Admin, manager	Update task
DE-LETE	/tasks/{id}	Admin, manager	Delete task

Table 9 compiles endpoints dedicated to statistics retrieval and analysis within the system.

Table 9:.Statistics Endpoint

Method	**Endpoint**	**Role Access**	**Purpose**
GET	/api/v1/taskStatistics	All role	View task statistics
GET	/api/v1/jobStatitstic/{projectId}	Admin, manager	View project statistics

Table 10 provides an overview of the event module management endpoints, detailing various functionalities and operations associated with event management within the system.

Table 10. Event endpoint

Method	**Endpoint**	**Role access**	**Purpose**
GET	/api/v1/event	All role	View all event
GET	/api/v1/event/{id}	All role	View specific event with its detail

POST	/api/v1/event	All role	Add new event
PUT	/api/v1/event/{id}	All role	Update the existing event
DELETE	/api/v1/event/{id}	All role	Delete specific event

6.5.2 Database Connection

Before, when initializing the project package, the package already added the MySQL driver. Now need to configure application properties to connect it with the CRM database in the localhost. Figure 15 shows how the database is configured in application properties. With this configuration, the server will access a database with the username and password provided beside the **spring.jpa.hibernate.ddl-auto=update** automatically generates and updates database schemas based on entity mapping in the application.

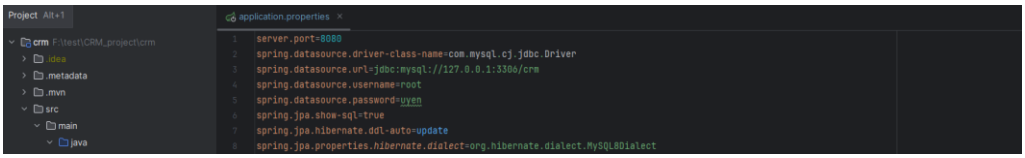


Figure 15. Database config in the backend.

6.5.3 Feature Implementation

Figure 16 below shows how the backend is structured. This source code is organized to ensure a three-tier model comprising a controller, business logic, and data access layers. Apart from packages structured within these layers, additional packages like util and exception exist.

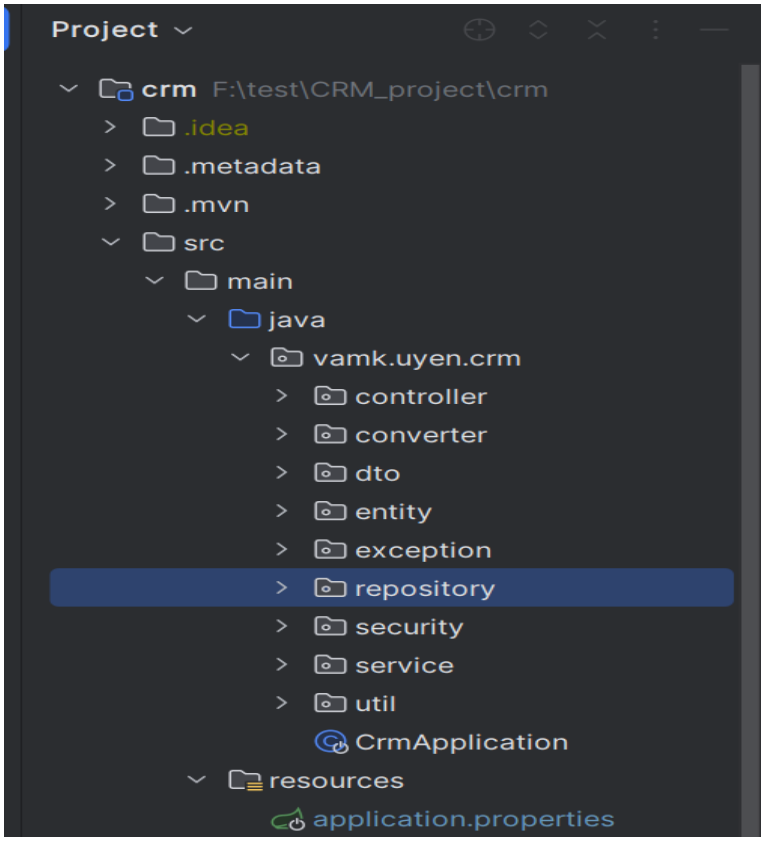


Figure 16: Backend Module structure.

The login and logout features are crucial for managing user authentication within the system. When users attempt to log in, they provide their email and password. If the credentials are valid, Spring Security generates an authentication token (JWT) to represent the user's identity, allowing them to access the system. The system invalidates the user's token for logging out, effectively ending their session, and ensuring a secure logout. The application enforces strict security measures to ensure that users cannot register externally. Only administrators possess the privilege to register new users. Upon registering a new user, administrators also assign specific roles to them. Consequently, newly created users gain access to the system using their assigned role-based accounts, enabling them to interact with the application according to their role.

A scheduled task status update feature is crucial in the implementation of task service management. This function retrieves all tasks and updates their status based on the start and end dates.

In Figure 17, within the component, the @Scheduled is employed with the cron expression to automate the daily update of the task and project status. This scheduling ensures that task and project status are refreshed precisely at midnight each day, thereby maintaining the system's accuracy and timeliness of task and project management data.

```
@Scheduled(cron = "0 0 0 * * ?")
public void updateTaskStatus() {
    taskService.updateTaskStatus();
    projectService.updateProjectStatus();
}
```

Figure 17. Schedule for update task and project status.

For the comprehensive management of projects, tasks, clients, users and events within the system, CRUD (Create, Read, Update, Delete) operations are essential. These operations are facilitated through the repository for each respective entity-projects, clients, tasks, users, and events. Each entity is supported by service layer methods designed to manage creation, retrieval, modification, and deletion. These methods interface with the JPA repository layer to execute database operations efficiently.

Additionally, the system incorporates validation checks to ensure data integrity and consistency. When adding a project or task, the system validates that the end date is after the start date, preventing the creation of logically incorrect records. Furthermore, the system checks for any associated tasks or projects when attempting to delete a user, project, or client. Suppose any tasks or projects are linked to the entity being deleted. In that case, the delete operation will fail, ensuring that dependent records are not left orphaned, which maintains the integrity of the data relationships within the system.

Moreover, the system includes functionality for users to recover their passwords if they forget them. The server is implemented to handle password recovery by

verifying the existence of the user's email in the system. If the email is verified, the server sends the user a one-time password (OTP). If the OTP is valid, the server allows users to change their passwords, ensuring a secure and efficient password recovery process.

6.6 Frontend Logic

The frontend, or client-side, is the part of the application that interacts directly with the user. It visually encompasses everything the user experiences on their screen, including layout, design, and user interface elements. The frontend communicates with the backend to display data, submit forms, and ensure a dynamic and responsive user experience.

On the login page, users can log in using their email and password. This function sends login credentials to the server and handles the response, which includes an access token, refresh token and user role. It saves these credentials in the local storage by using the saveToken function for later use. After logging in successfully, it will navigate to the dashboard page. It will notify the user of login success or failure. Figure 18 illustrates the implementation of the login function, managing server request and response.

```
const login = async (email, password) => {
    try {
        const response = await http.post('/v1/auth/login', { email, password });
        const { accessToken, refreshToken, role } = response.data;
        saveToken(accessToken, refreshToken, role);
        navigate('/dashboard');
        sendToast('Login successful!');
    } catch (error) {
        console.error('Error logging in:', error.message);
        sendToastError('Failed to login.');
    }
}
```

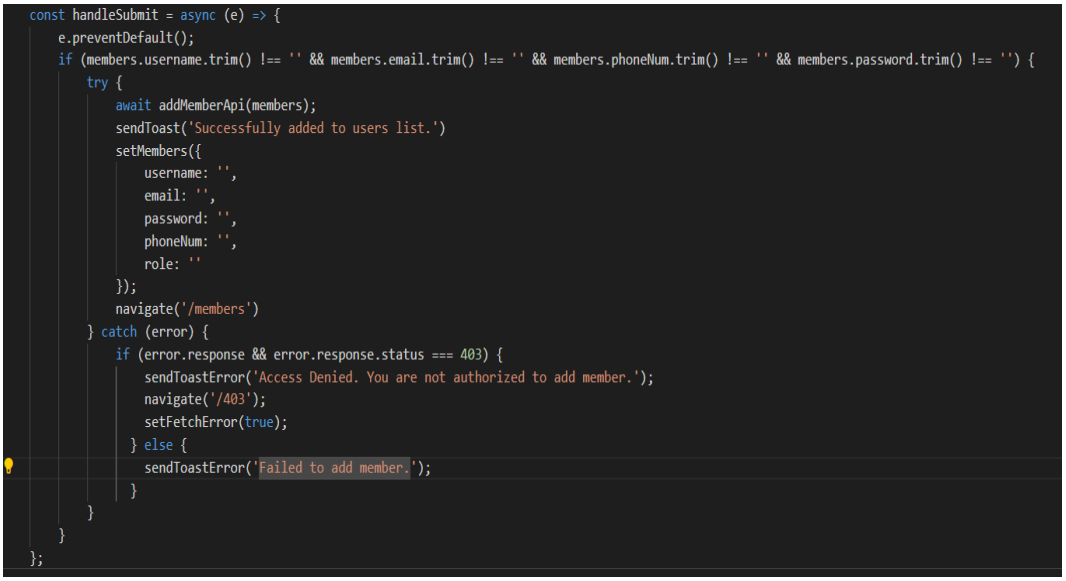
Figure 18. Login Function

Users can view their work progress on the dashboard page through task statistics displayed in a chart. A function handles server requests and responses for pages displaying lists or information. The page content is displayed correctly if the server responds with a 200 HTTP status code. If a 403-status code is returned, indicating insufficient permissions, the user is redirected to a 403-error page, and a notification informs them of the access restriction. For other server errors, a notification will alert the user that the request has failed. Figure 19 illustrates the implementation for displaying members in response to server requests. If the response is successful, the list of members will be shown. If not, the system will display an error message indicating the specific issue and will direct to a 403-error page if the user cannot access the member page.

```
const fetchApiMembers = async () => {
  try {
    const response = await http.get('/v1/users');
    return response.data.content;
  } catch (error) {
    console.log(error);
    if (error.response && error.response.status === 403) {
      sendToastError('Access Denied. You are not authorized to view this page.');
      navigate('/403');
      setFetchError(true);
    } else {
      sendToastError('Failed to fetch member.');
    }
    return [];
  }
```

Figure 19. Fetch members function.

Similarly, the Add or Update pages provide a form for users to submit new or updated data. These pages handle server requests and responses like list pages. Upon a successful request, the user is redirected to the list page to view the updated changes in the database. Figure 20 provided code snippet is a JavaScript function handle submit that handles form submission for adding a new member. The function trim removes any leading or trailing whitespace in the user inputs and checks for non-empty fields.



```
const handleSubmit = async (e) => {
    e.preventDefault();
    if (members.username.trim() !== '' && members.email.trim() !== '' && members.phoneNum.trim() !== '' && members.password.trim() !== '') {
        try {
            await addMemberApi(members);
            sendToast('Successfully added to users list.')
            setMembers({
                username: '',
                email: '',
                password: '',
                phoneNum: '',
                role: ''
            });
            navigate('/members')
        } catch (error) {
            if (error.response && error.response.status === 403) {
                sendToastError('Access Denied. You are not authorized to add member.');
                navigate('/403');
                setFetchError(true);
              } else {
                sendToastError('Failed to add member.');
              }
        }
    }
};
```

Figure 20. Handle submission for adding member

7 TESTING AND OUTCOMES

7.1 Testing

To ensure the appropriate execution of functions and ascertain the adequacy of the front end in managing server requests and responses, a comprehensive manual testing strategy was implemented. This approach involves utilizing a browser to execute a series of test cases that rigorously assess the frontend interface's management of server interactions and its response to errors through HTTP status codes. These test cases are meticulously designed to evaluate all aspects of the system functionality to ensure its operational integrity and user interface effectiveness.

The testing process was methodically structured to encompass all critical stages of development and deployment. Initially, the project was executed in a local environment on a laptop, serving as the primary test bed. This local testing phase involved using a web browser to interact with the application, providing immediate feedback and facilitating necessary adjustments. The project was launched in a local development environment using a local host server setup. Various browsers, including Google Chrome and Microsoft Edge, were employed to verify cross-browser compatibility. The test cases validated the frontend's capability to handle server requests and responses correctly, with particular attention given to HTTP status codes and error handling. After local testing, the application was deployed to a live domain. The application was then accessed via a browser at its live domain to confirm its functionality, ensuring that any issues related to server configuration or network conditions are identified and resolved.

7.2 Outcomes

Figure 21 displays the appearance of the login interface. The login page displays a minimalist and user-friendly login form that requires users to input their email and password. It also includes an option for users who have forgotten passwords, enhancing accessibility and user support. If users attempt to access other pages within the system without being logged in, they will be immediately redirected to the login page to provide their credentials.

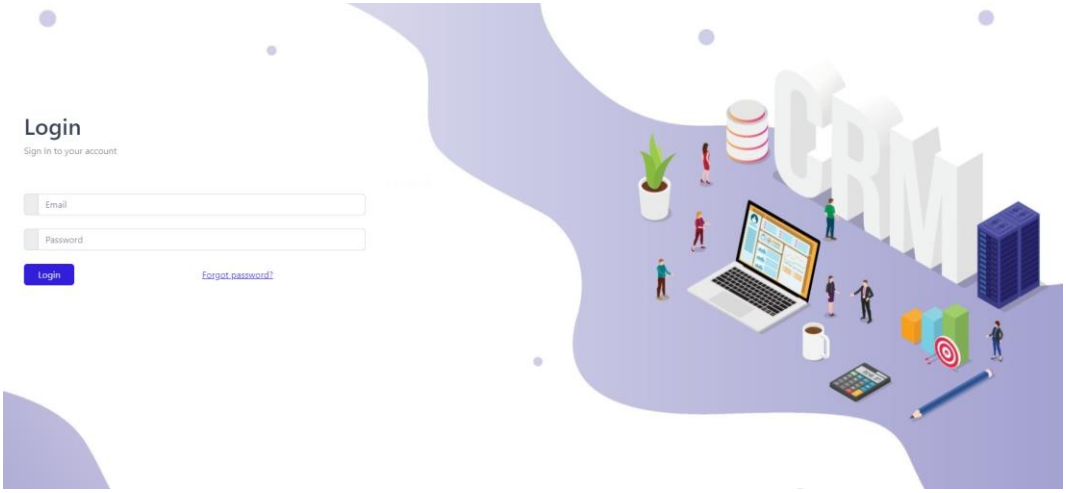


Figure 21. Login Page

Once successfully logged into the system, users are redirected to their dashboard, which serves as a central hub displaying their work progress overview. Throughout the system, a consistent layout features a left-side sidebar facilitating easy navigation to other pages. Additionally, a header bar showcases the user's profile image, which, upon clicking, triggers a dropdown menu allowing access to the user's profile page, as depicted in Figure 22.

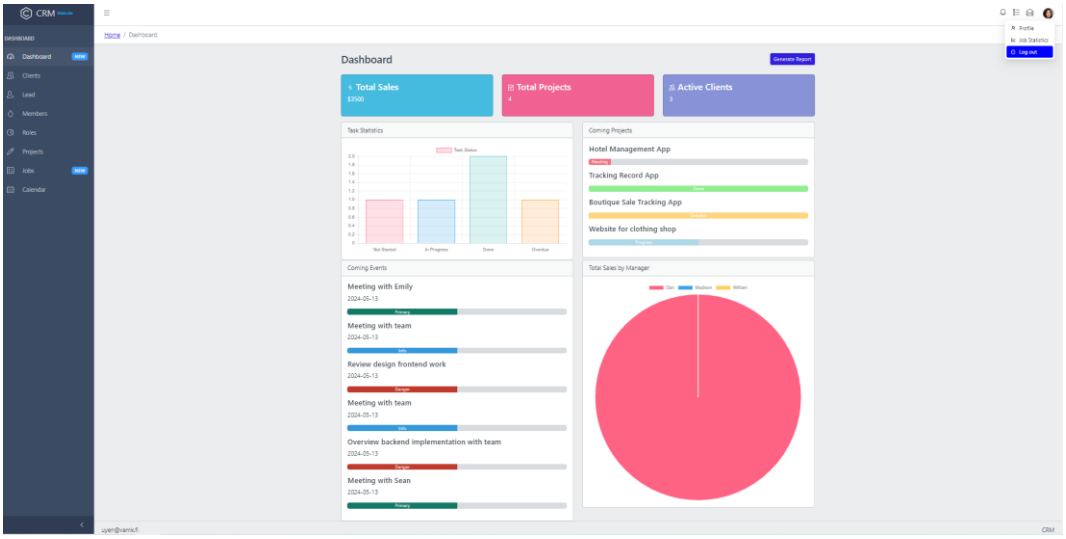


Figure 22. Navigate to the dashboard after successful login.

Figure 23 shows the client page, which contains a comprehensive list of clients along with their respective details. Within this page, users are provided with an Add button to add new clients. Each client card includes a dropdown menu, allowing users to either update or delete the desired client entry.

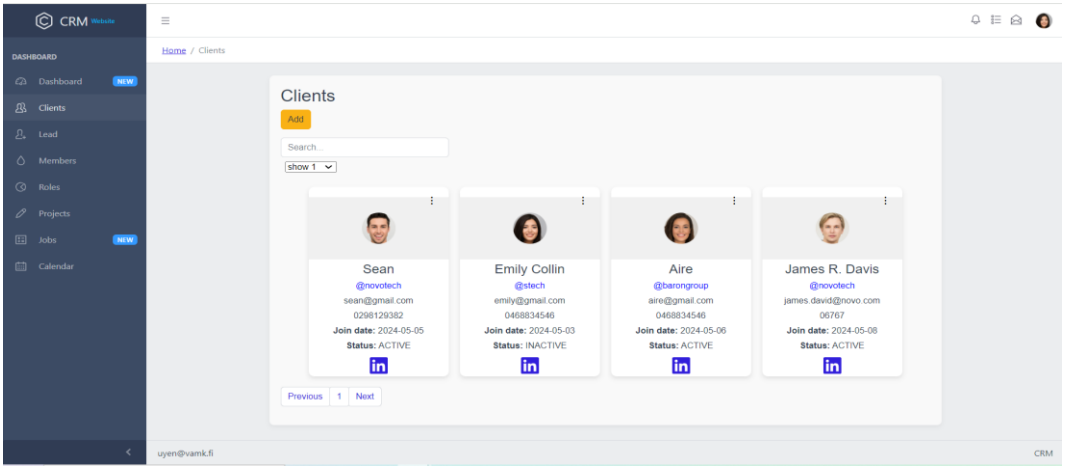


Figure 23. Client page.

The add or update client, as in Figure 24, the form displays an enabling the user to input or modify client details as needed. Upon submission by the user, the system promptly notifies them regarding the success or failure of the action. In case of errors, the system displays an error message to alert the user.

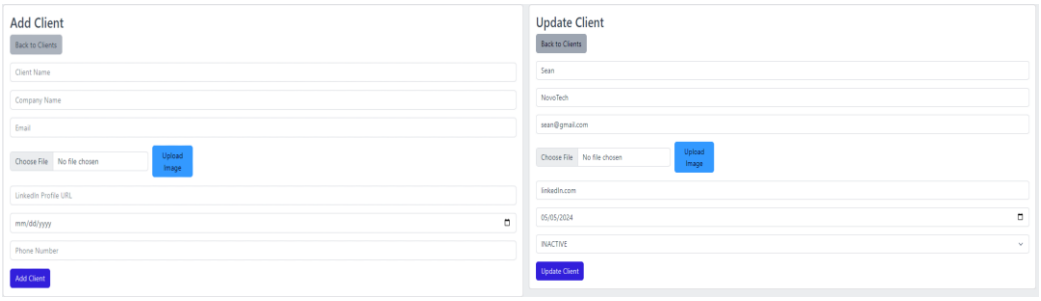


Figure 24. Add and Update Client page.

Figure 25 illustrates the UI of the member page, which displays a list of members and their details. This page includes a search bar allowing users to search for specific member information and it also includes an add, update, and delete button.

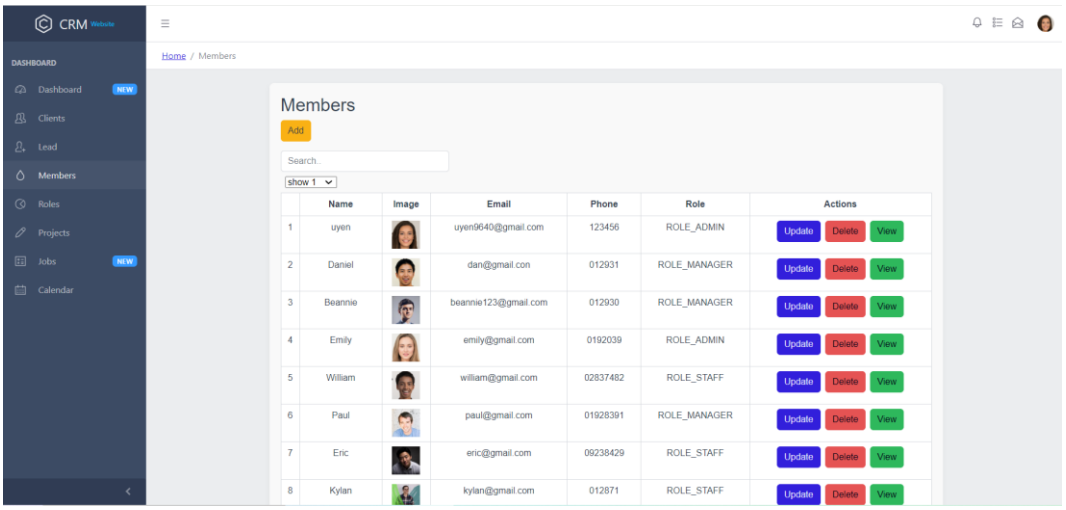


Figure 25. Member page.

Figure 26 shows the form of adding and updating members. These forms present a user-friendly form for inputting or updating member details. Additionally, it includes an error handler to manage input errors effectively, ensuring a seamless user experience.

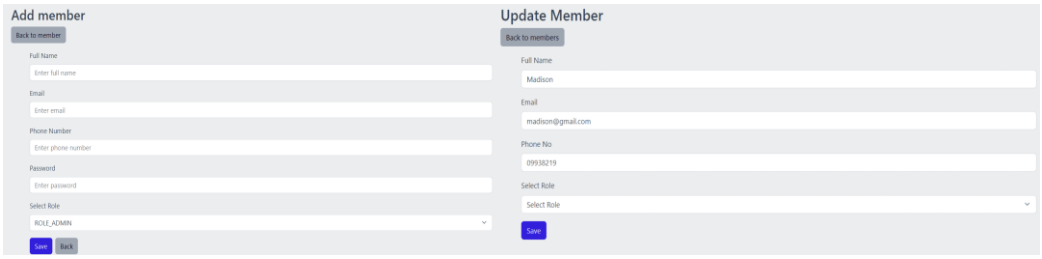


Figure 26. Add and Update member page.

The role page displays all available roles in the system, as shown in Figure 27. Access to this page is restricted solely to administrators. Administrators possess the capability to update the description associated with each role, given that the system has already established access permissions for each role.

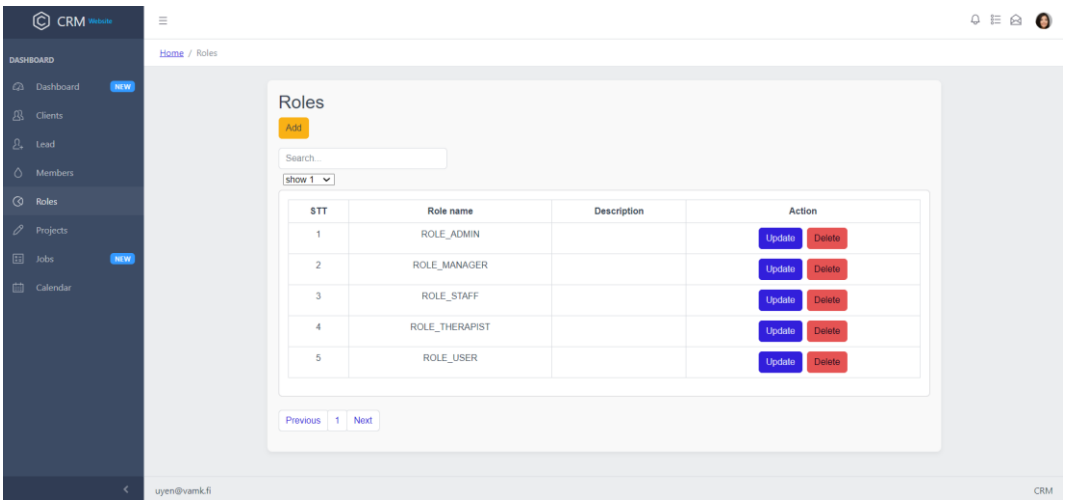


Figure 27. Role page.

The project page renders the response from the server based on the user's role. Administrators see all projects listed, while project managers view only the projects they manage. Employees are presented with a list of projects associated with tasks they are currently implementing, as depicted in Figure 28. This approach ensures that users access project information according to their roles without dynamic adjustments on the page itself.

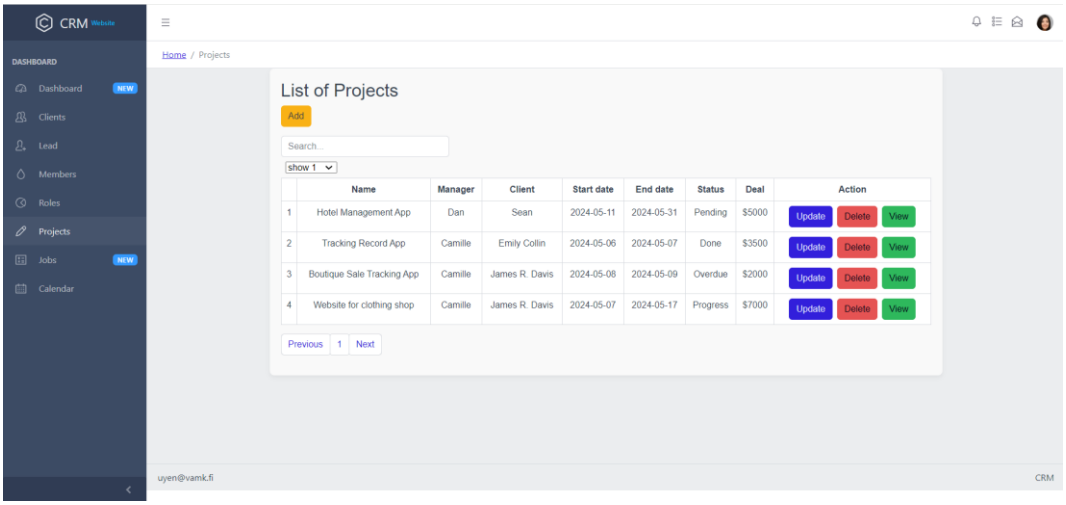


Figure 28. Project page for Admin role.

Figure 29 is respectively the add and update project form. The add/update project feature provides users a form to input or update project details. Typically, project managers are responsible for creating projects. Therefore, when a project is created, it is automatically assigned to the manager who initiated its creation. This streamlined process ensures efficient assignment and management of projects within the system.

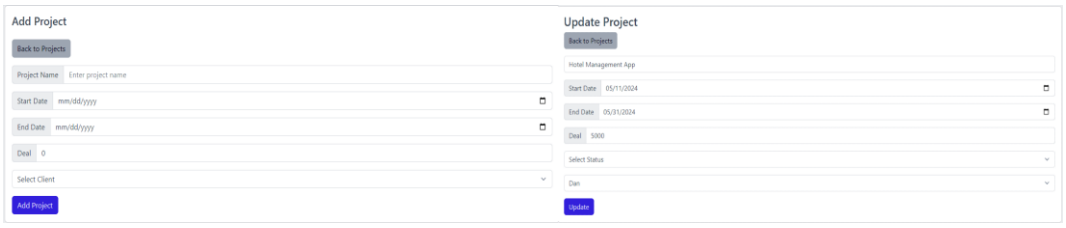


Figure 29. Add and Update project page.

The job page in Figure 30 displays all tasks assigned to the user, with the task list tailored to the user's role. Administrators have access to view all tasks across the system. Project managers are limited to viewing tasks under the projects they manage. Regular users can only view tasks they are responsible for implementing. This role-based rendering ensures that users see task information relevant to their responsibilities within the system.

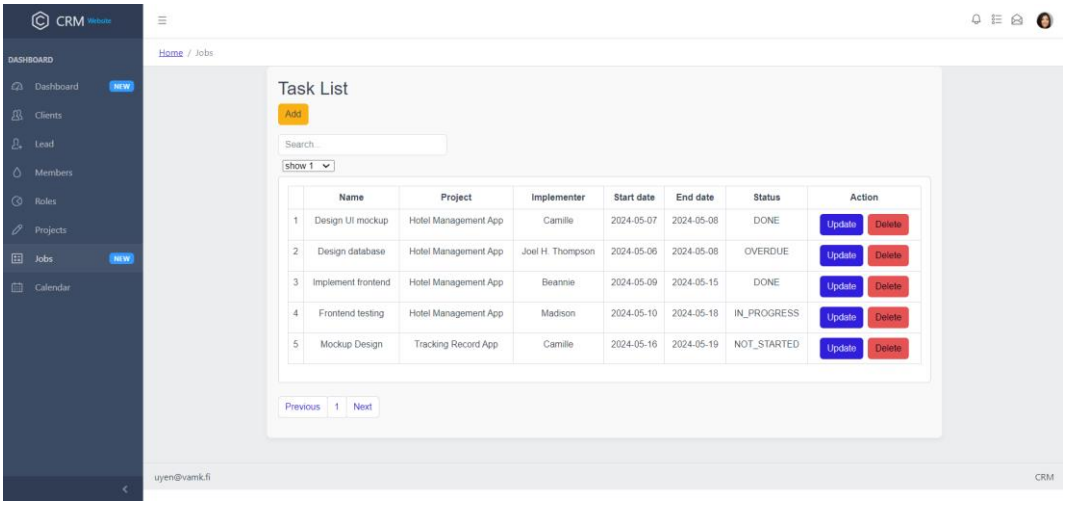


Figure 30. Task page.

In the add/update task feature, administrators and project managers have the ability to input new task details or update existing tasks through a form. Additionally, they can assign an implementer for the task, enabling efficient task management. Add and update job forms are shown in Figure 31. These forms allow the user to fill in the job details such as job name, start date, end date, select existing project and implementer, and able to select task status.

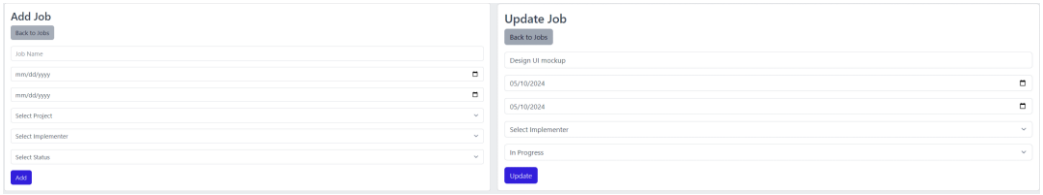


Figure 31. Add and Update task page.

Users can access their profile, as shown in Figure 32, by clicking on their profile image in the header bar, as previously described. Users can view their profile information, including statistics related to their tasks on their profile page, as well as a comprehensive task list. This centralized location offers users convenient access to their personal information and task-related data.

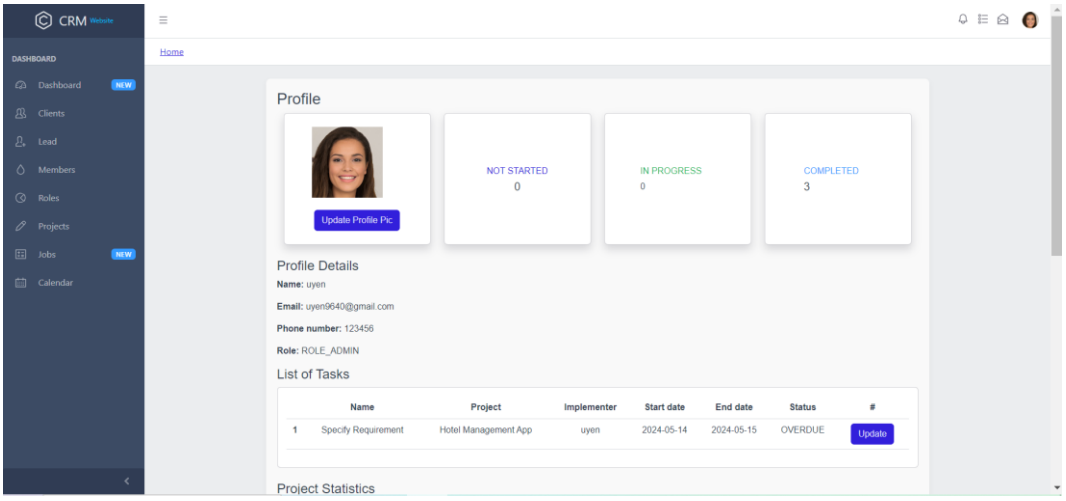


Figure 32. Profile page.

Users can view project progress within the system according to their roles by accessing job statistics in the dropdown header, illustrated in Figure 33. Administrators can view the progress of all projects, providing a comprehensive overview of the entire systems status. Project managers can view the progress of their projects, allowing them to monitor their specific responsibilities and ensure timely completion. Employees can see a task progress chart for their assigned tasks, helping them stay on track and maintain accountability for their work.

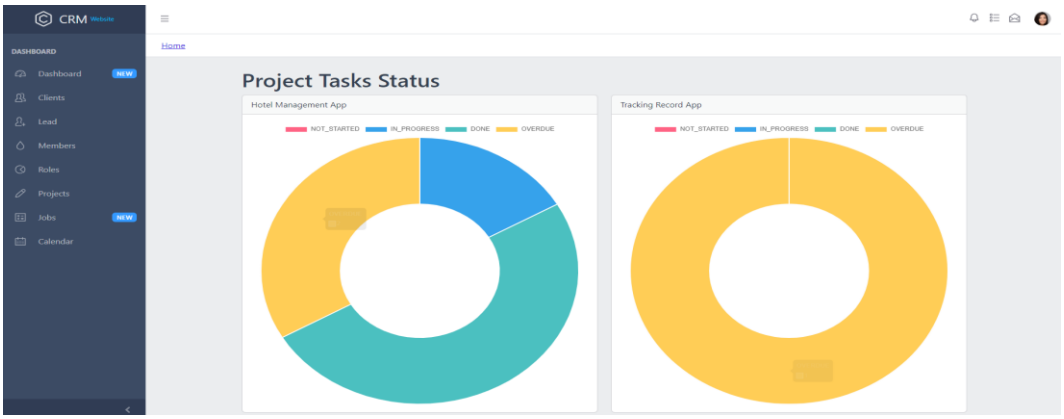


Figure 33. Job statistics.

The lead page is shown in Figure 34. On the lead page, all lead information, which is active clients with projects, is displayed. To view more details about the projects associated with each lead, toggle the button at the top of each lead table.

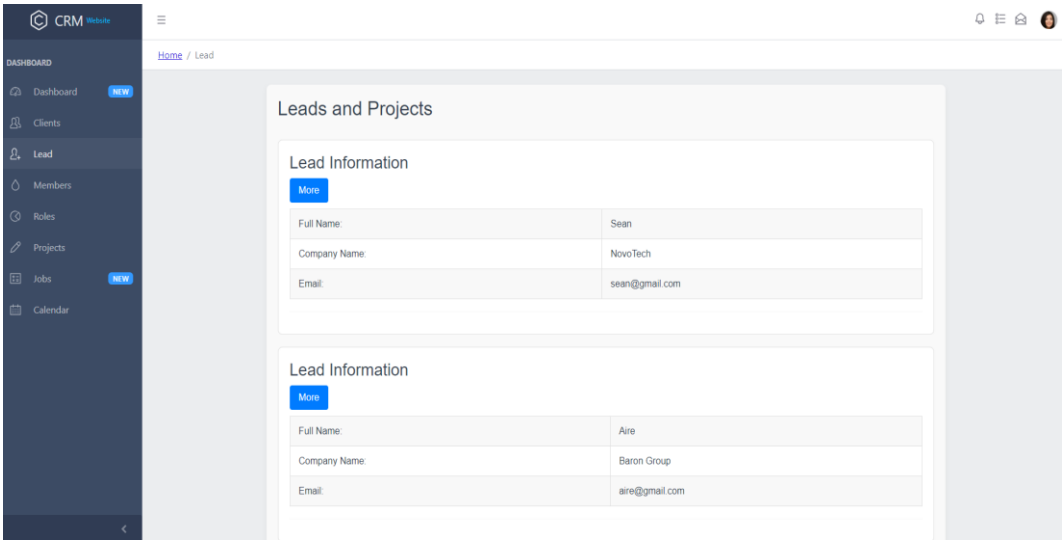


Figure 34. Lead page.

Moreover, clicking to view a specific project will present a list of tasks associated with that project, accompanied by statistics job status visualized through a chart, as depicted in Figure 35 below.

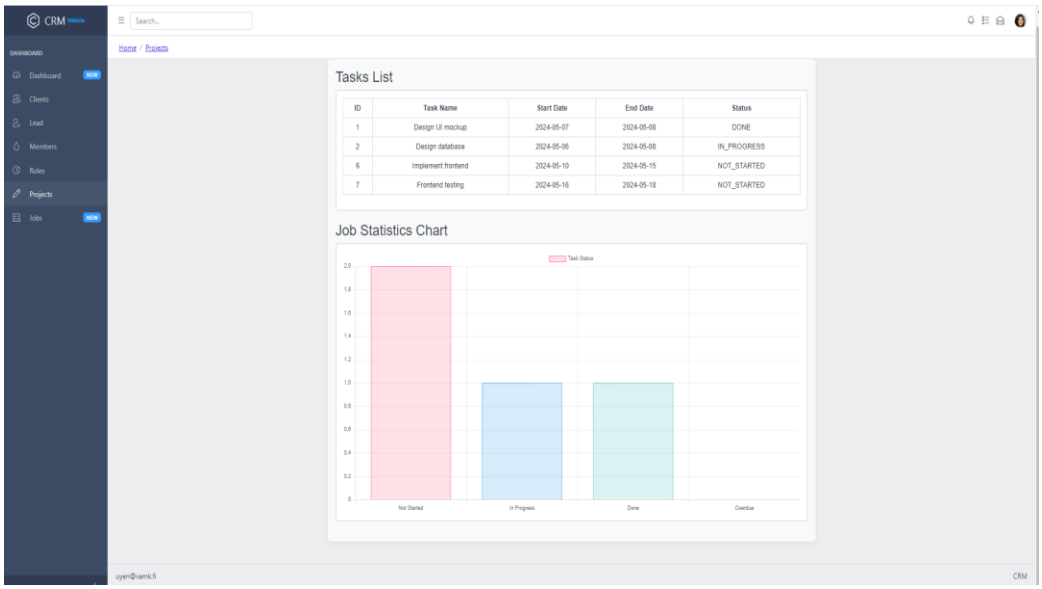


Figure 35. View Specific Project

Figure 36 is the event calendar page. On this page, users can view all scheduled events, which are color-coded by category. Users can schedule new events by

selecting a date on the calendar. This interface facilitates easy event management and interaction.

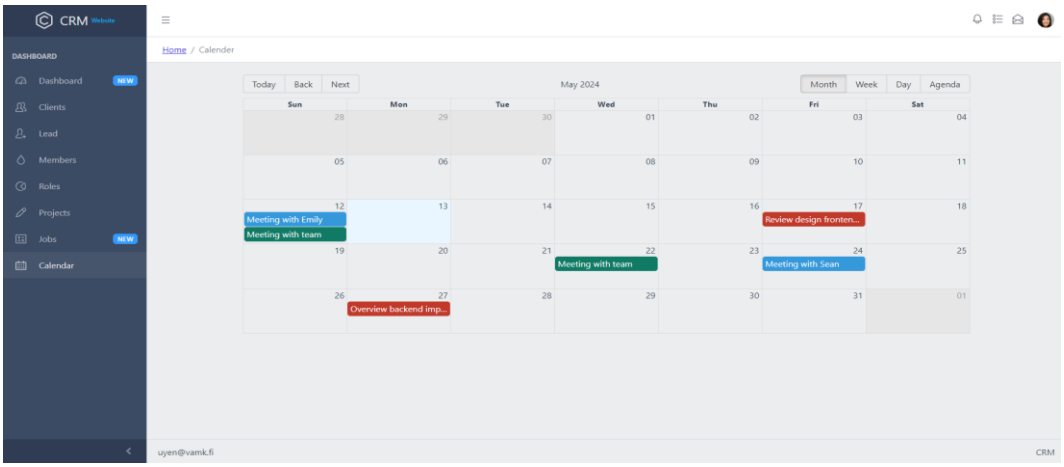


Figure 36: Event Calendar

Figure 37 displays the edit form. To update or delete a specific event, the event shown in the calendar is clicked. This action opens the edit form where changes can be made to the event details, or the event can be deleted. This streamlined process ensures efficient management of calendar events.

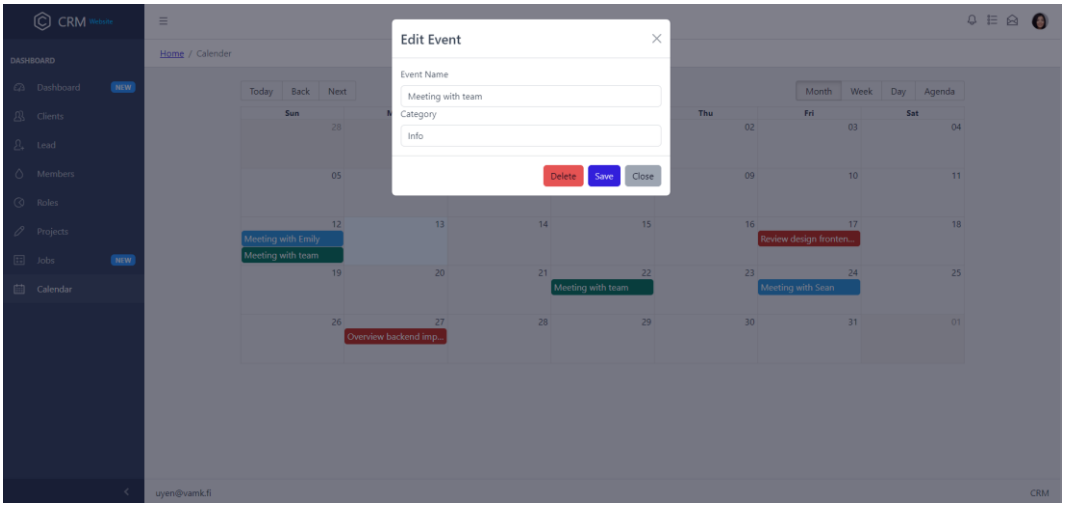


Figure 37: Editing event popup.

When users want to recover their password, they can click on "Forgot Password" on the login page. This action navigates them to a series of forms, as shown in Figure 38 where they enter their email address, verify the OTP sent to their

email, and then set a new password. This process ensures secure password recovery and allows users to regain access to their accounts.

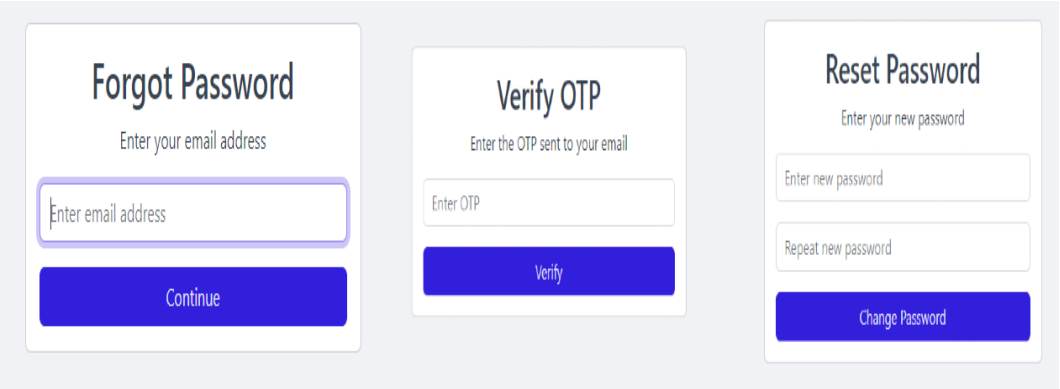


Figure 38: Forms for forgot password.

8 CONCLUSIONS

In the culmination of this research endeavor for this thesis, a comprehensive exploration into the development and integration of a Customer Relationship Management (CRM) website with project management functionality has been undertaken. The synthesis of CRM concepts, project management principles, and web development techniques has yielded valuable insights into the potential benefits and implications of such an integrated platform.

Centralizing customer data and project details within a unified interface presents a compelling advantage for businesses seeking to streamline operations and bolster productivity. This holistic solution facilitates efficient management of customer relationships and project workflows, thereby enhancing overall organizational effectiveness.

The meticulous development process, characterized by thoroughly considering user requirements, system architecture, and technological selections, has resulted in a robust CRM website. Leveraging advanced front-end technologies, such as ReactJS and Redux alongside the robust backend framework Spring Boot has endowed the platform with a user-friendly interface, comprehensive functionality, and seamless integration capabilities.

Reflecting on this project, it is evident that the insights gained, and skills honed throughout the development process are invaluable. From identifying critical aspects of the subject matter to making informed decisions about system architecture and programming languages, this project has provided fertile ground for personal and professional growth. Notably, in the pursuit of technological advancement, the importance of safeguarding data privacy and implementing stringent security measures in the CRM website, the strategic choice of Spring Boot for its robust security features underscores the importance of aligning technology choices with project requirements.

In conclusion, the culmination of this thesis marks not only the attainment of research objectives but also the acquisition of practical skills and insights that will continue to inform and enrich future endeavors in web development and project management. Developing a CRM integrated with project management has yielded several notable results. Real-time dashboards provide immediate access to key metrics and performance indicators, enabling data-driven decision-making. Increased efficiency is achieved by streamlining operations by centralizing client information and project details, significantly reducing time spent on administrative tasks. Data-driven insights are leveraged to uncover patterns and trends, aiding in strategic planning, and enhancing customer relationship management.

Additionally, centralized client contact management organizes all client interactions and communications within a single platform, improving accessibility and management. The integrated lead tracking system further enhances the ability to monitor and manage potential sales opportunities. For future advancements, several avenues for research and development are proposed to improve the system further. These include the incorporation of a chat box to facilitate real-time user communication, the implementation of a notification system to keep users informed of new records and essential updates, mobile optimization to ensure the platform is fully accessible on mobile devices and the development of advanced algorithms to forecast customer behavior and potential interactions. These future enhancements and continued research efforts aim to elevate the platform, ensuring it remains a robust and indispensable tool for businesses seeking to optimize their customer relationship management and project management processes.

REFERENCES

Cloud Object Storage - Amazon S3. (n.d.). Retrieved from Amazon Web Service: https://aws.amazon.com/pm/serv-s3/?gclid=EAIaIQobChMI-c7J3sfvhQMVAheiAx3jewu2EAAYASAAEgLyqPD_BwE&trk=b45f363b-5d02-4b3f-87df-b7b1908ff05c&sc_channel=ps&ef_id=EAIaIQobChMI-c7J3sfvhQMVAheiAx3jewu2EAAYASAAEgLyqPD_BwE:G:s&s_kwcid=AL!4422!3!536452769228!e!!g!!

Dan Abramov and the Redux documentation authors. (2023, November 25). *Redux Essentials, Part 1: Redux Overview and Concepts*. Retrieved from Redux: https://redux.js.org/tutorials/essentials/part-1-overview-concepts

Fadatare, R. (n.d.). *REST API Tutorial*. Retrieved from Java Guide: https://www.javaguides.net/p/rest-api-tutorial.html

Hausman, A. (2023, August 21). *The Evolution of Database Monitoring Services*. Retrieved from Market Maven: https://www.hausmanmarketingletter.com/the-evolution-of-database-monitoring-services/

Johnson, Juergen Hoeller, Arendsen, Alef, Sampaleanu,Colin Sampaleanu. (2008). *The Spring Framework - Reference Documentation.* Retrieved from Spring: https://docs.spring.io/spring-framework/docs/2.5.x/spring-reference.pdf

Kantor, I. (2007). *The Modern JavaScript Tutorial-Part 1.*

Lorek, S. (2018, November 2). *What exactly is CRM … And How Does It Help Construction Firms?* Retrieved from Unanet: https://unanet.com/blog/what-exactly-is-crm-and-how-does-it-help-construction-firms

Marc Garreau and Will Fauro. (2018). *Redux in action.* Manning.

Masse, M. (2021). *REST API Design Rulebook.* O'Reilly Media, Inc.

Prateek Rawat, Archana N. Mahajan. (2020). ReactJS: A Modern Web Development Framework. *International Journal of Innovative Science and Research Technology* , 1.

Svetlin Nakov & Team. (2021). *Programming Basics with Java.* Faber Publishing, Sofia.

Zoho CRM. (n.d) Retrieved from: What does CRM software do? https://www.zoho.com/crm/what-is-crm.html#:~:text=The%20World's%20Favorite%20CRM%2C%20Zoho,support%20in%20a%20single%20system.

What is a RESTful API? (n.d.). Retrieved from Amazon Web Service: https://aws.amazon.com/what-is/restful-api/

What is JavaScript? (n.d.). Retrieved from Amazon Web Service: https://aws.amazon.com/what-is/javascript/#:~:text=AWS%20SDK%20for%20JavaScript%20is,js%20for%20server

Zola, A. (2022, 8). *Bootstrap*. Retrieved from TechTarget: https://www.techtarget.com/whatis/definition/bootstrap