Palestine Techinical University (Khadoorie)

Faculty of Engineering and Technology

Department of Computer Systems Engineering

“Introduction to Graduation Project” Thesis

HR and Correspondence Workflow Messaging System

Prepared By:

Fares Hatem Abu Ali

Ahmad Mohammad Hamadneh

Sondos Ghassan Jarrar

Ahmad Othman Marie

Supervised by:

Mohammad Khalil, Ph.D.

Tulkarm – Palestine

January 2022

This document was submitted in fulfilment of the requirements for the "Introduction to Graduation Project" course, during the academic year 2021-2022 (First Semester).

ACKNOWLEDGEMENT

In the name of Allah, the Most Gracious and the Most Merciful.

We would like to take this opportunity to first and foremost thank God, the merciful for being our guide and for giving us the strength to write this thesis and carry on this project and for blessing us with many great people who have been our greatest support in both our professional and personal life.

First and foremost, we would like to sincerely thank our project's supervisor and head of the department - Dr. Mohammad Khalil - for his guidance, effort, and patience. All thanks to our lecturers and instructors. They have always provided positive encouragement and continuous support to finish this thesis.

Special thanks to Eng. Ahmad Qerm, a Software Engineer at Dimensions Healthcare Company – Ramallah since the project's idea was first inspired by a meeting held with him. We should also not forget Eng. Iyad Atallah, a Software Engineer & Web Developer at the PTUK Computer Center, for his time and effort spent to provide us with a better understanding of the requirements of such a project.

Finally, all thanks to all the group members for sharing the positivity and invaluable assistance.

ABSTRACT

The idea of developing an HR and Correspondence Workflow Messaging System was inspired by discovering our university's HRM "PTUK Human Resources Management System".

By questioning the head of the department, Dr. Mohammad Khalil, and some of our lecturers on how the communication process runs among these lecturers themselves, and among the lecturers and their heads of departments and deans for example, we concluded that the process of workflow communications and messaging inside our university organization is almost automated and pretty much comfortable.

This facilitation was made possible by dint of the PTUK HRM software system - which they have been using for a while - for managing the communication and messaging process.

Hence, we began thinking of why not trying to adopt a new similar software system that offers the same essential functions and modules which the PTUK HRM system offers, but with an attempt to generalize the system by making it flexible to fit the needs of any organization, not only a university, since most of the establishments share a similar workflow communication and messaging process, with some additional custom differences.

After achieving our ambitions mentioned above, we can think of ways to financially benefit from our system in the post-graduation level. One of the ways may be by marketing the system through the internet for interested clients, such as organizations and startups, and if an organization is interested in our system, then they can request a demo or pay for the system to be deployed on the organization's domain. Then our responsibility will be to offer the consistent support for the system

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table Page

CHAPTER 1

INTRODUCTION

1.1 Overview

First, let's clarify a terminology that might look strange to some people:

What is a Correspondence?

A **Correspondence** is a written form of communication between two endpoints - a sender and a recipient, or a group of recipients (consignees) - In other words, correspondence is the letters, emails, or any other type of technical writing used within an organization as a way of communication between its employees.

Nowadays, the demand for web-based correspondence management services is continuously expanding. Written communication has become extremely important for modern businesses. Organizations expecting its employees to respond to different types of correspondences in a timely manner and with high accuracy. We need to design a Correspondence & Messaging Management system that assists organizations to do that by streamlining the whole communication process with the help of a comprehensive web-based platform. [1.1]

1.2 Problem Statement

Consider your day-to-day business interactions – all the emails you send, documents you share and any type of business information you exchange between colleagues, customers or third parties – all these activities fall under the umbrella of business correspondence. As any other complex process within a business or governmental establishment, business correspondence requires proper management.

According to International Data Center, the average worker spends about 2.5 hours per day or roughly 30% of the workday writing business correspondence.

Considering average yearly salary as $80,000, the inability to find and retrieve documents costs organization (that employs 1000 workers) $2.5 million per year. The enterprise also wastes $5 million per year because employees spend too much time duplicating information that already exists within the enterprise.

So that is why we have decided through our graduation project to analyze, design, and implement a Correspondence Tracking System. It will be a new experience through which we aspire to offer the needed automation which allows companies to reduce costs and improve their workflow.

* In the references chapter, you will find links of all websites and research papers which we accessed and helped us in the literature review process and in writing this Introduction Chapter.

CHAPTER 2

RELATED WORKS

2.1 PTUK HRMS

(Palestine Technical University's Human Resources Management System)

PTUK HRMS is also a Correspondence Management System which is mainly used to create and track all the incoming and outgoing internal correspondences and workflows within the university's environment and make them available to the system's users (Almost all PTUK staff) according to their roles and positions in the hierarchy which determine their permissions.

We can say that the process of workflow communications and messaging among our university's staff is almost automated and pretty much comfortable. This facilitation was made possible by dint of the PTUK HRM software system which have been used recently, for facilitating the communication process and making it look much easier.

However, the system lacks some functionalities and features, such as the ability to create teams' channels, or what so called chatrooms (groups of employees of different positions who work on a current project). This feature will be elaborated in detail in the coming paragraphs.

2.2 Microsoft Outlook

Outlook is a personal information manager software system from Microsoft. Though primarily an email client, Outlook also includes such functions as calendaring, task managing, contact managing, note-taking, journal logging, and web browsing.

With Outlook on your PC, you can organize email to let you focus on the messages that matter most. Microsoft Outlook is designed to operate as an independent personal information manager. [2.1]

2.3 Slack

slack is not a correspondence management system. It differs a lot. However, we include it here as a case study because of its great functionalities that we aspire to include some in our project.

Slack is a workplace communication tool, “a single place for messaging, tools, and files.” This means Slack is an instant messaging system with lots of add-ins for other workplace tools. The add-ins aren’t necessary to use Slack, though, because the main functionality is all about talking to other people. There are two methods of chat in Slack: *channels* (group chat), and *direct message* or *DM* (person-to-person chat).

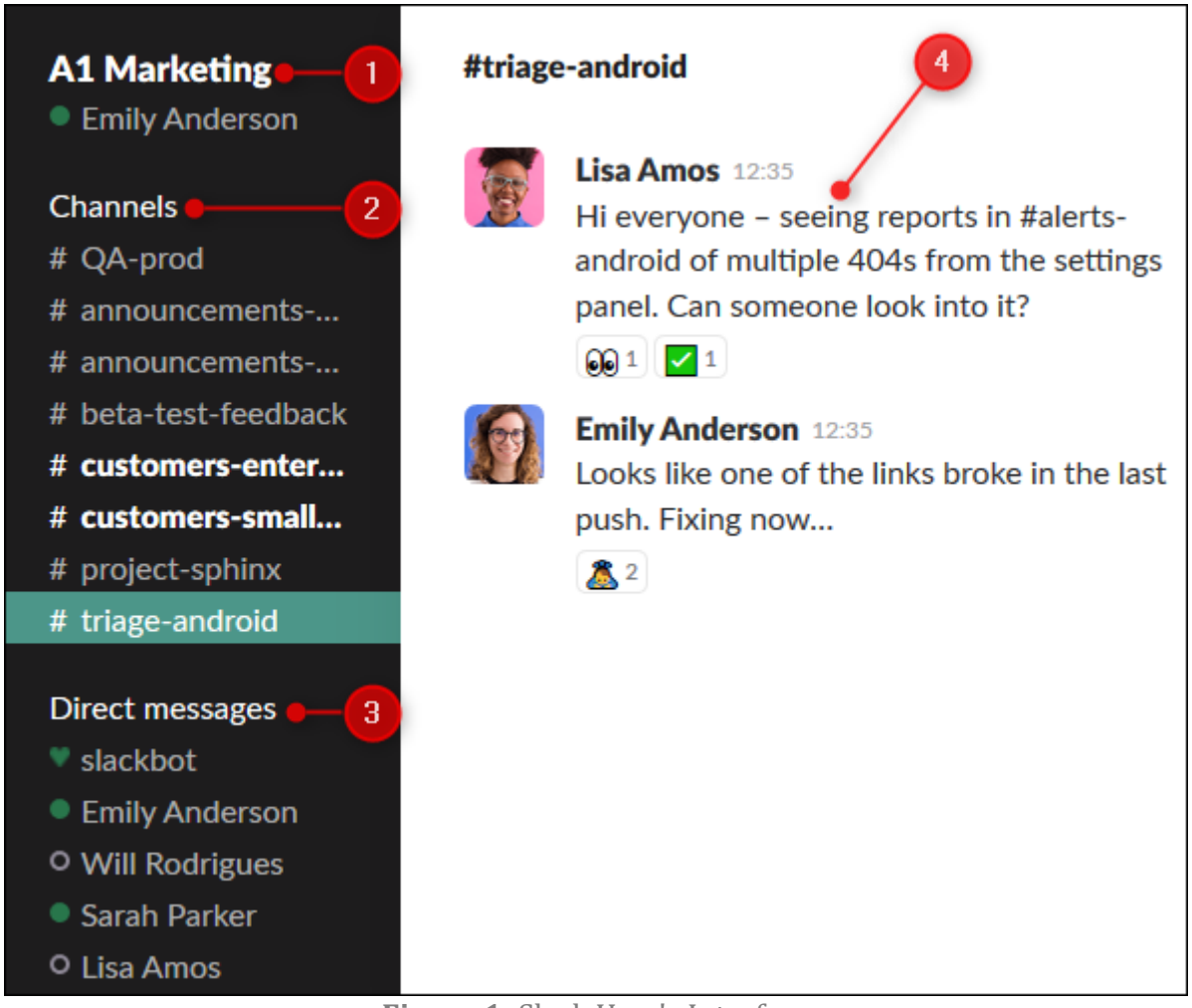

Figure 1: Slack User's Interface

There are four main things to pay attention to in Slack:

1. The name of the Slack instance.
2. The list of channels you’re a member of.
3. The list of people you’ve direct messaged.
4. The chat window.

When Slack came along, there were no real competitors in the market. That's not to say there weren't other chat apps, but Slack combined an intuitive UI with both group and person-to-person messaging. It also allows companies to have a measure of control over who can use it through the invitation system. Other tools could do the same, but without the same usability. [2.2]

2.4 Zoho Mail

Zoho Mail is an excellent email service that is clean, fast, and offers better protection against fake emails.

Zoho mail allows you to create a domain for your business and set up custom email addresses for users. Unique and professional email addresses give your company the visibility and authenticity it deserves. [2.3]

2.4.1 Streams - the newer way to email

ZohoMail offers a feature called 'Streams', which adds a social media flavor to your mailboxes with Streams. Replace unending email threads with comments, tag your teammates, share files, manage tasks, and much more, right from within your inbox. [2.4]

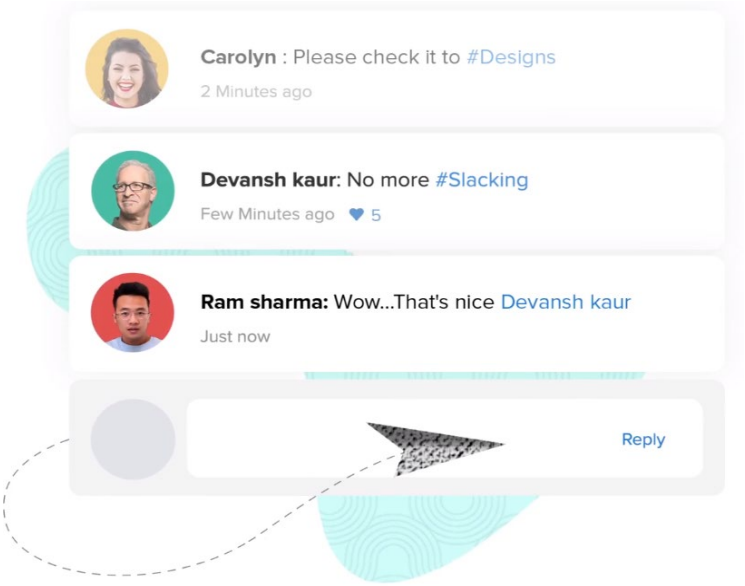

Figure 2: ZohoMail Streams UI

2.4.2 Email Retention and e-Discovery

Retain emails across your organization for a specified period to comply with company standards and to counter legal attacks. e-Discovery helps discover such retained emails quickly. [2.5]

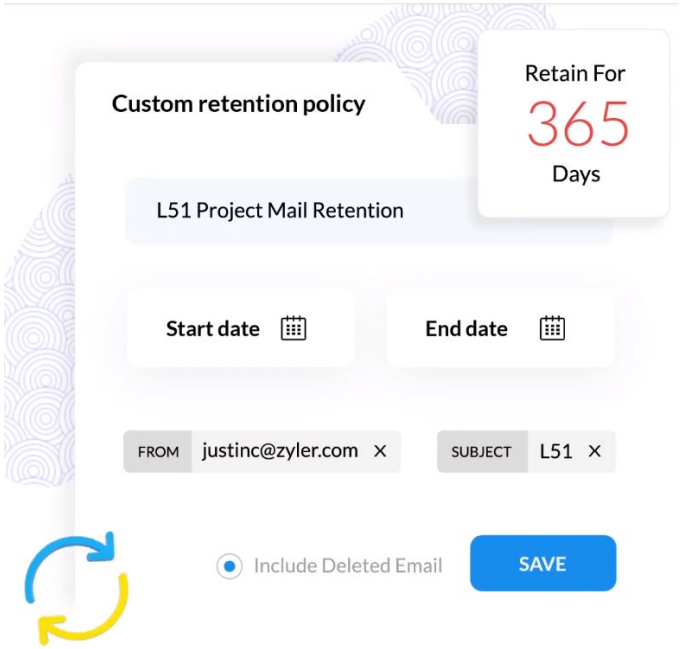

Figure 3: E-mail Retention Feature

2.5 Comparison

2.5.1 Successive Replies

Table 1: Comparison by "Successive Replies" Feature

	Feature: Successive Replies
PTUK HRM	One of the PTUK HRM System's downsides is that once the person replies to a received workflow, he loses the ability to add another reply for the same workflow. So imagine if you are writing a reply and hit send, then remember you forgot to add the attachment to it, or you remembered to add some details to your reply. You cannot send another reply. Instead, you will have to wait for one of the consignees (recipients) to read your reply, then reply to it so that you can again reply to the workflow after it is forwarded to you.
MS Outlook	Outlook gives you the freedom to add multiple successive replies
Slack	Slack is different. It is an instant messaging system which provides group and person-to-person messaging. Not used for sending workflows, so we will exclude it from this comparison.
Zoho Mail	Zoho Mail also gives you the freedom to add multiple successive replies

2.5.2 Messaging Policy

Table 2: Comparison by "Messaging Policy" Feature

	Feature: Messaging Policy
PTUK HRM	One of the great features provided in the PTUK HRM is its messaging policy which adds levels of security to the system and ensures its proper use. What I am trying to say is that the employee within an organization – such as a university - cannot send workflows to whoever he desires inside the organization, nor be able to see all the workflows. Instead, a set of fields (such as the employee's position in the hierarchy, his/her classification, and job title) who determine his/her roles and privileges, thus state to whom he/she is allowed to send, and from whom he/she is allowed to receive workflows.
MS Outlook	Microsoft Outlook is designed to operate as an independent personal information manager. Its mailing system and interface is so appealing, but it does not provide the organization the ability to fully control the roles and privileges of its employees.

Slack	Slack is different. It is an instant messaging system which provides group and person-to-person messaging. Not used for sending workflows, so we will exclude it from this comparison.
Zoho Mail	ZohoMail offers a feature called 'Streams', which adds a social media flavor to your mailboxes with Streams. Replace unending email threads with comments, tag your teammates, share files, manage tasks, and much more, right from within your inbox.

2.5.3 Creating Projects (Team Channels, or Chatrooms)

Usually in IT companies, the employees work on projects, each project consists of a group of employees (team) which must work cooperatively to finish the project.

With a lot of projects being worked on at the same time, there will be a lot of teams, and usually one employee would be working on multiple projects in parallel.

Table 3: Comparison by "Creating Chatrooms" Feature

	Feature: Creating Projects (Chatrooms)
PTUK HRM	Although the workflow messaging policy is good to guarantee the proper flow of workflows in a way complying to the organization's rules, these restrictions may make it difficult to work on projects within the organization. Working on projects sometimes require a combination of employees that do not have any common link in-between in the hierarchy tree. For example, in the university, it happens that there is a project that needs collaboration between the head of computer engineering department, and employees from the faculty of agriculture. The process of sending workflows between "the head of computer engineering department" form a side, and let's say, "a specific lecturer in the department of environment" from the other side, will be a tedious job. Why? Because the two sides are far away from each other in the hierarchy tree. First, "the head of computer engineering department" will have to send the workflow to "the dean of engineering faculty", then the dean will read it and forward it to "the dean of agriculture faculty", then the dean of agriculture will have to pass the workflow to "the head of environment

	department", who in turn will pass it to "a specific lecturer in the department of environment". (See Figure 4 for visual illustration) **Sender's Position in the hierarchy tree:** * PositionID = Department of Computer Engineering (subordinate to - Deanship of Engineering) * PositionNature = responsible (رئيس قسم) * JobTitle = Head of Computer Engineering **Recipient's Position in the hierarchy tree:** * PositionID = Department of Environment (subordinate to Deanship of Agriculture) * positionNature = staff (موظّف عادي/مُحاضر، ليس رئيس قسم) * JobTitle = Lecturer
MS Outlook	Microsoft Outlook does not serve this feature.
Slack & MS Teams	Here comes the power of Slack as a communication tool, it is also similar to Microsoft Teams [2.6] in that both provide the feature of creating channels. A channel is a chat room where instant messages and conversations are held among teammates of a specific project. There are two methods of chat in Slack: channels (group chat), and direct message or DM (person-to-person chat) [2.7]
Zoho Mail	Just like Microsoft Outlook.

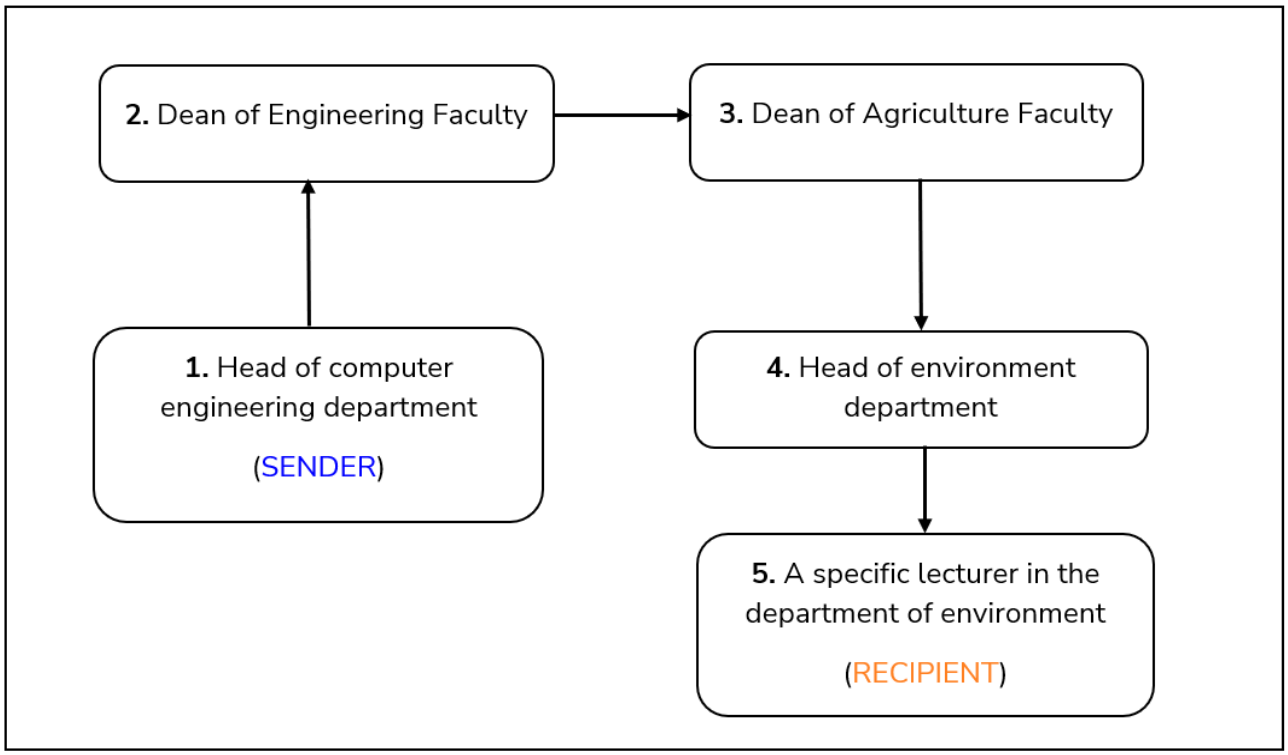

Figure 4: A complicated forwarding process of a workflow (e.g. letter or email) that was meant to be sent from "the head of computer engineering department" to a lecturer in a different department in different faculty (e.g. "lecturer in the department of environment which is within the faculty of agriculture").

The instant Team Channels (Chatrooms) module should be used in such a case to simplify the messaging process.

2.6 Conclusion of Comparison

From the above comparison, our correspondence Workflow Messaging System is aimed to offer the three main features that we mentioned:

- **Successive replies** to a workflow.
- Ability to control and customize the **messaging policy** to comply with the organization's hierarchy tree (Specify who can send to whom).
- Ability to create **project** channels (**chatrooms**) to allow direct communication among employees who work on a project. In other words, make a real-time chat without the headache of sending a lot of emails (workflows), nor being restricted nor prevented from messaging someone because of your position in the hierarchy.

Table 4: Conclusion of the Comparison

Case Study / Feature	PTUK HRMS	Slack & MS Teams	Outlook	ZohoMail	Our Project
Successive Replies	✗	☑	☑	☑	☑
Customize Messaging Policy (Organization Structure)	☑	✗	✗	✗	☑
Creating Projects (Chatrooms)	✗	☑	✗	☑	☑

CHAPTER 3

SOLUTION AND OBJECTIVES

3.1 Solution

Unlike the general-purpose traditional mailbox systems such as Microsoft Outlook, we aim to make the decision of determining authorities and roles for each user (Who can send to whom) fully controlled by the organization itself. So the employee cannot send workflows to whoever he desires inside the organization, nor be able to see all the workflows. Instead, a set of fields (such as the employee's position in the hierarchy, his/her classification, and job title) who determine his/her roles and privileges, thus state to whom he/she is allowed to send, and from whom he/she is allowed to receive workflows. So, this customization adds levels of security based on each organization's needs and on its own policy and method of treating its employees.

Each organization will have the freedom to customize the roles, privileges, and permissions according to its hierarchy tree. We aim to make all these security features dynamic as much as possible and fully customizable by the organization itself, not by the system's developers.

3.2 General Objective (Requirement)

Our correspondence workflow messaging system is aimed to become a fully integrated web-based solution, enabling organizations to transfer and track incoming and outgoing internal correspondences within an organization.

3.3 Specific Objectives

Correspondence management is of paramount importance for all organizations, government agencies and other parties that have to deal with business correspondence, and seek the highest possible level of productivity, with the help of an efficient control system to effectively govern the ever-increasing correspondence channels. This is where correspondence management systems come into play.

Correspondences such as emails, letters, fax, invoices, and others are core to any establishment’s everyday activities. However, tracking and responding to these correspondences timely could become an annoying task if it is not managed correctly.

The moment a correspondence is received, it requires an appropriate action, which could be either redirecting it to the concerned individual/department, sending a reply, or archiving it.

The way these communications are managed directly impacts an organization's effectiveness.

Once the employee joins the organization, he/she will by default acquire an account that allows him/her to use the system. Then he/she can easily be merged and become a part of the organization's family. Then start participating in receiving and sending the required workflows that help support the organization's direction towards paperless traditional communication that reduces expansions, enhances performance, and raise productivity level.

Our system aims to accomplish the following objectives: [3.1]

- Manage inbound, outbound, internal, and related actions of a workflow.
- Link workflows and obtain related history at glance.
- Let the enterprise owners build their own Organization Structure (Hierarchy Tree) without any limit to number of levels or users.
- Each organization will have the freedom to customize its employees' roles, privileges, and permissions according to its hierarchy tree. (We plan to make the process of specifying organization structure as general as possible, to be customizable as per business owner's needs).
- Make all the security features dynamic as much as possible and fully customizable by the organization itself.
- Ability to send copies of same workflow action to different recipients with one signature (Specify a group of recipients/consignees)
- Easy-to-use and comprehensive search tools in addition to full text search that can help accessing any workflow easily when needed.
- Notification system.
- Ability to expand and include more business process UI Features.

- HTML5 & CSS3 responsive design UI, in addition to using React.js in rendering UI components effectively. ReactJS is one of the best and most successful JavaScript libraries.
- Intuitive Easy-to-navigate UI with minimal steps.
- Manage Inbox, follow-up, CC, notifications, and drafts.
- Full log of workflows history. (Monitoring)
- Easy administration, with comprehensive admin module to manage users' roles and track all workflows history.

CHAPTER 4

TECHNOLOGIES AND IMPLEMENTATION

In this chapter we will discuss the IDE on which we plan to work, all used technologies, programming languages, frameworks, and database we plan to use in building our website.

4.1 Overview of Implementation Stack

Integrated Development Environment

- Visual Studio Code

User Interface Design

- Adobe XD

Front-End Technologies:

- HTML5
- SASS SCSS Preprocessor
- React.js JavaScript Library
- React Material UI
- Gulp.js Task Runner

Back-End Technologies

- Node.js
- Express.js Framework

Database

- MySQL database
- Sequelize ORM (Object Relational Mapping) package

4.2 What is Implementation Stack

For developing large-scale apps like this, we need to follow a specific development stack, and what we mean by this term is, what are the technologies we plan to use to build this application or this prototype? what languages were used in this process, and why?

4.2.1 MERN

MERN is a web development stack which stands for (MongoDB, Express.js, React.js, Node.js). It's a development stack used to build fully functioning applications.

However, MongoDB is a NoSQL database program, and since our software system has a full structured relational database schema, we figured out that the traditional SQL Database Management Systems - such as MySQL - should be more than enough to develop our database. So we plan to remove MongoDB from our development stack and replace it with MySQL.

So our custom MERN implementation stack will become as follows:

- **M**: MySQL database
- **E**: Express.js (Back-end framework)
- **R**: React.js in Front-end (JavaScript Library for Client-Side Work)
- **N**: Node.js for Back-end (Server-Side JavaScript Environment)

4.3 Integrated Development Environment

4.3.1 Visual Studio Code (VSCode)

For the implementation of the Correspondence & Workflow Management System prototype, we used VSCode to write the needed markup and script code for the website. Particularly, Microsoft Visual Studio Code 2022 will be used in the development process.

Figure 5: VSCode Logo

Microsoft Visual Studio is a lightweight integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with websites, web applications, and web services. [4.1]

Visual Studio Code can do pretty much everything regarding web development reasonably well. Good thing VSCode literally has a catalog of over 24,000 different extensions to extend its own capabilities.

VSCode already supports a plethora of different programming syntaxes out of the box, but you still can install a language extension to expand support for useful features, such as IntelliSense auto-complete and advanced linting. The sky is the limit, from Python and C++ to COBOL and Lisp. All you have to do is click on an “Install” button right inside VSCode.

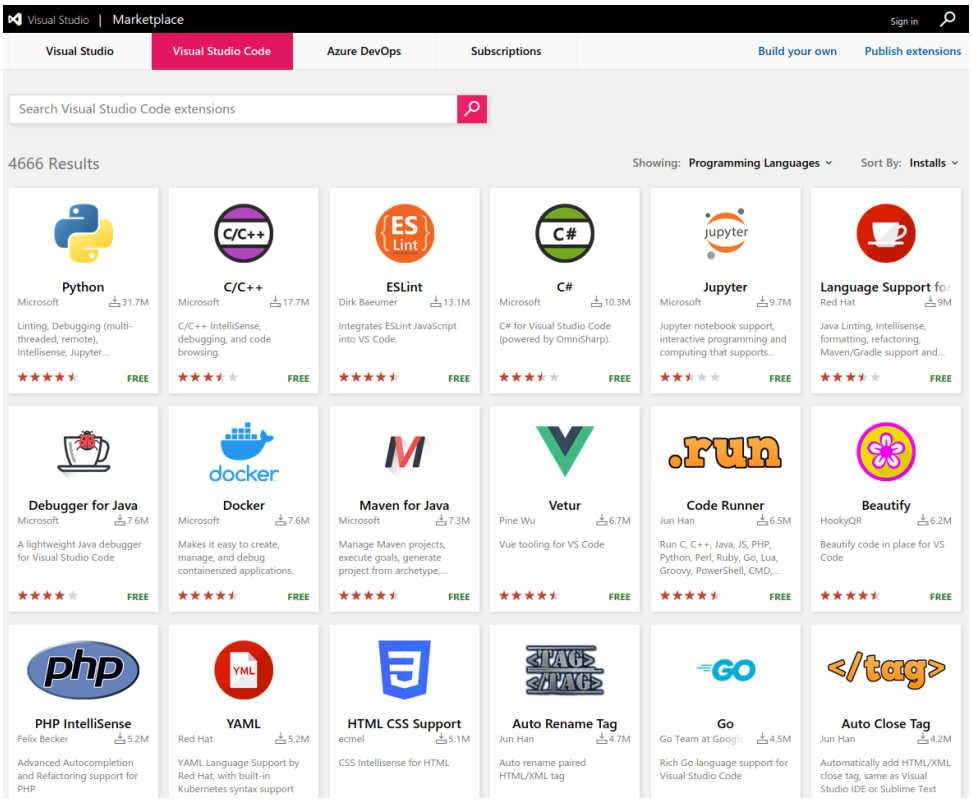

Figure 6: A catalog of Visual Studio Code programming language extensions. Note that there are **4666** results at the time of writing. That’s a humongous amount!

4.4 Front-End Technologies

4.4.1 HTML 5

Figure 7: HTML5 Logo

HTML5 is the latest version of Hypertext Mark-up Language, the standard programming language for describing the contents and appearance of Web pages. HTML5 was developed to solve compatibility problems that affect the current standard HTML4.

One of the biggest differences between HTML5 and previous versions of the standard is that older versions of HTML require proprietary plugins and APIs. (This why a Web page that was built and tested in one browser may not load correctly in another browser) HTML5 provides one common interface to make loading elements easier.
HTML5 has been designed to deliver almost everything you'd want to do online without requiring additional software such as browser plugins. It does everything from animation to apps, music to movies, and can also be used to build incredibly complicated applications that run in your browser. [4.3]

4.4.2 SASS (a CSS Preprocessor)

What is a CSS Preprocessor?

- A layer between stylesheet you make and CSS file you give to the browser.
- A program that takes one type of data and converts it to another.

Figure 8: SASS Logo

What is SASS?

- SASS stands for Syntactically Awesome Stylesheets.
- SASS is CSS extension language.

Why are we planning to use a CSS Preprocessor like SASS or SCSS?

- It saves time.
- Makes our code organized and maintained.
- Allows us to write DRY Code (Don't Repeat Yourself), faster, reliable, and more efficient.
- Gives our CSS style the missing features. [4.4]

SCSS is **S**assy **C**ascading **S**tyle **S**heets. SCSS can be separated by a semicolon and run on the same line. SCSS is a pre-processor which lets you use features that aren't a part of the wider CSS standard yet and provides better workflows for maintaining your stylesheets. With SCSS pre-processor, you can reduce the amount of time you repeat yourself and ensure you're writing clean, maintainable code for the future. SCSS can take CSS code and work. SCSS is fully compatible with the syntax of CSS, while still supporting the full power of Sass.

SCSS is an extension of the syntax of CSS. This means that every valid CSS stylesheet is a valid SCSS file with the same meaning. In addition, SCSS understands most CSS hacks and vendor-specific syntax.

Figure 9: Simple Comparison between CSS and SCSS

4.4.3 React.js

React is an open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. [4.5]

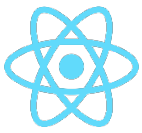

Figure 10: React.js Logo

Why we plan to use React.js over other JavaScript frameworks and libraries:

We went with React over (Angular.js, Vue.js, Vanilla `Pure` JavaScript), for various reasons:

❖ **Learning curve:** the learning curve of this React.js library is really good, and you can pick it up in no time.

❖ **Reusability:** It's well known that React is oriented around building small components and then reuse them in multiple areas on the website, and that makes the development process much more enjoyable and faster.

❖ **Popularity:** As a team, we needed to find out what is the common ground between us so we can make the communication easier, and we found out that we all know a little bit of React.js and so we decided to go with it.

❖ **Scalability:** Using something that could be reused again and again, is always something good for scalability, and as discussed before, we are planning to scale this project up later on so we choose React.js because it's the best fit for such a goal "in the meantime at least".

❖ **Market's Need:** React.js library is robust and the demand for it in the market is really high, so by learning it we are keeping up with what the market in the real world needs the most.

According to stack overflow trends, React.js took the lead among other JavaScript front-end libraries in web developments in the last couple of years. See (Figure 11).

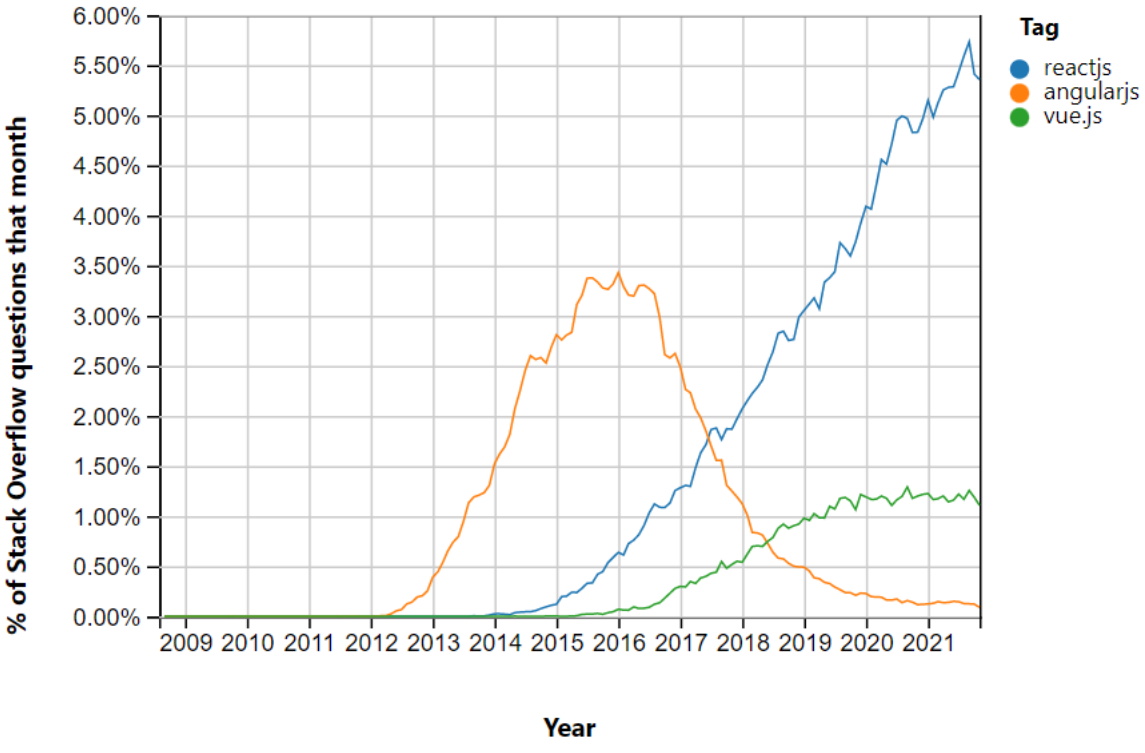

Figure 11: This graph is a comparison that shows how the most popular JavaScript front-end libraries have trended over time based on use of their tags since 2008, when Stack Overflow was founded. [4.6]

The graph was taken on 11-Dec-2021. Visit the link below to see up-to-date comparison that could change each month:

https://insights.stackoverflow.com/trends?tags=reactjs%2Cangularjs%2Cvue.js

4.4.4 React Material UI

Material-UI is simply a library that allows us to import and use different components to create a user interface in our React applications. This saves a significant amount of time since the developers do not need to write everything from scratch.

Figure 12: Material UI Logo

Material-UI widgets are heavily inspired by Google's principles on building user interfaces. It is, therefore, easy for developers to build visually appealing applications. [4.7]

4.4.5 Gulp.js

Gulp.js is an open-source JavaScript toolkit. It is a task runner built on Node.js and npm (Node Package Manager), used for automation of time-consuming and repetitive tasks involved in web development like CSS and JS files minification, files concatenation, cache busting, unit testing, linting, optimization, and much more. [4.8]

Mainly we will use it to automate repeated tasks in our development environment, such as tasks mentioned above.

Figure 13: Gulp.js Logo

4.5 Back-End Technologies

4.5.1 Node.js

Node.js is an open-source, cross-platform, back-end, JavaScript runtime environment that executes JavaScript code outside the web browser, it executes it on the back-end servers.

Figure 14: Node.js Logo

We used Node.js to build our restful APIs, separating the logic of the app from its front-end code, that way we can handle any back-end issues or features in a stand-alone manner, without ever affecting any other code.

We went with node because it is a lightweight and super-fast back-end framework, easy to learn, works asynchronously, has a large active community to help if needed. Furthermore, the global demand for it and here in Palestinian market is growing amazingly. [4.9]

4.5.2 Express.js

Express.js, or simply Express, is a back-end web application framework for node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

Figure 15: Express.js Logo

It helps create routes within the Node.js app, so we can easily know to which function the coming requests should go on the server.
By splitting the backend application into multiple routes, it provided us with multiple features: [4.10]

- We can separate the logic of each functionality in a stand-alone file so it would be easier to read and understand.
- Maintenance in case of failure, by splitting them we know where to look (in which file) depending on the error message and could debug that problem alone.

- Scaling the application, later, would be feasible, since it's already split into multiple small chunks.
- Express doesn't just create routes, it also can help us use other libraries with Node.js, like CORS (Cross-Origin Resource Sharing), and more. It even helps with reading JSON format from the requests rather than doing it manually.

Figure 16: Express.js Advantages & Disadvantages

4.5.2.1 How popular are Node and Express?

The popularity of a web framework is important because it is an indicator of whether it will continue to be maintained, and what resources are likely to be available in terms of documentation, add-on libraries, and technical support.

There isn't any readily available and definitive measure of the popularity of server-side frameworks (although you can estimate popularity using mechanisms like counting the number of GitHub projects and StackOverflow questions for each platform). A better question is whether Node and Express are "popular enough" to avoid the problems of unpopular platforms. Are they continuing to evolve? Can you get help if you need it? Is there an opportunity for you to get paid work if you learn Express?

Based on the number of high profile companies that use Express, the number of people contributing to the codebase, and the number of people providing both free and paid for support, then **yes, *Express* is a popular framework!** [4.10]

4.5.3 MVC Design Pattern

4.5.3.1 What is the MVC?

MVC stands for Model, View, Controller is an architectural design pattern that separates an application into three main logical components: the model, the view, and the controller. Each one of these components is built to handle specific development aspects of an application.

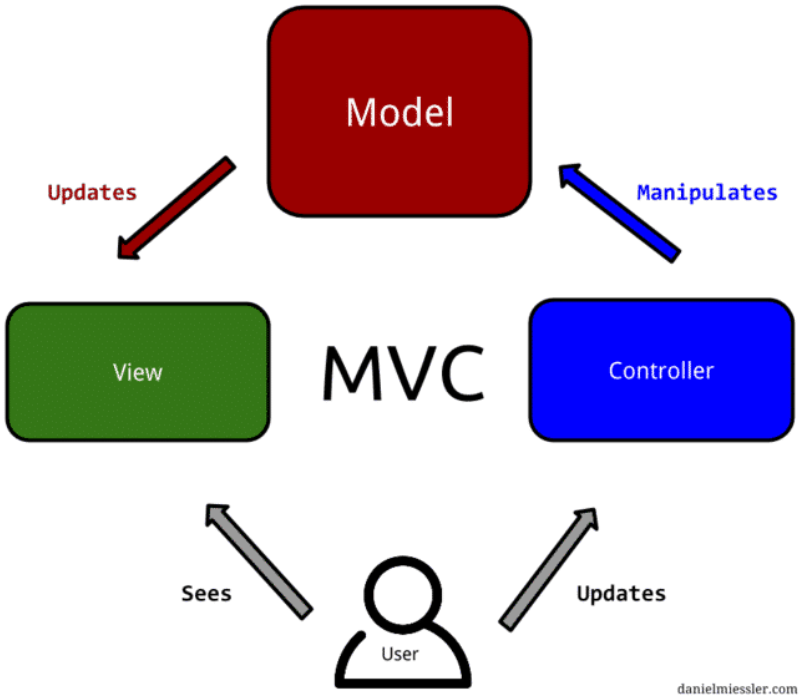

Figure 17: Model-View-Controller Design Pattern Overview

MVC is architecture used in building Web Servers that focus on reliability and making the development process much simpler and easier since it composes the Web Server into three separate parts.

- **Controller** is the part that takes care of client request processing which handles the HTTP Request and returns a response the response could be either a JSON if you're calling an API endpoint or regular HTML webpage.

- **Model** is the database interface which lets you interact with the database API and create different entity schemas of your app on the database (MySQL, MongoDB), it gets called from controller depending on the client's request if it needs a stored data then the controller will ask the model interface for providing it with the needed data.

- **View** is what compiles and renders into plain HTML and what the client in most cases going to get back as a response of what he requested (for ex: to view his profile details), the view needs to use a Template Engine for doing the rendering process where the controller feed it with needed data (data from database and client) and the view renders and convert everything into plain HTML that the browser could understand and display.

So the three different components interact with each other perfectly to ensure reliable and flexible client request processing, either for simple tasks like accessing the home page or updating his profile data, that's why building your Web Server around and MVC Architecture is really cool and will let you with tons of options and flexible thoughts. [4.11]

4.5.3.2 What does Express code look like?

Express.js is an ***unopinionated*** web framework, which means that you as a developer using this framework, have the freedom on what design pattern or what method you use to structure your code. You don't have to abide by a specific design architecture such as MVC for example.

Instead, you can insert almost any compatible middleware you like into the request handling chain, in almost any order you like. You can structure the app in one file or multiple files and using any directory structure. You may sometimes feel that you have too many choices!

In a traditional data-driven website, a web application waits for HTTP requests from the web browser (or another client). When a request is received the application works out what action is needed based on the URL pattern and possibly associated information contained in POST data or GET data. Depending on what is required it may then read or write information from a database or perform other tasks required to satisfy the request. The application will then return a response to the web browser, often dynamically creating an HTML page for the browser to display by inserting the retrieved data into placeholders in an HTML template.

Express provides methods to specify what function is called for a particular HTTP verb (GET, POST, SET, etc.) and URL pattern ("Route"), and methods to specify what template ("view") engine is used, where template files are located, and what template to use to render a response. You can use Express middleware to add support for cookies, sessions, and users, getting POST/GET parameters, etc. You can use any database mechanism supported by Node (Express does not define any database-related behavior). [4.10]

4.5.3.3 Node.JS Express MVC

After talking in detail about the NodeJS Express, and how it is an unopinionated framework, the main parts of an Express app, like (routes, middleware, error handling, and template code), the way in which all these parts should be pulled together is up to us!

For developing our project, mainly we will follow the MVC design architecture. In the following figure, you can see an overview of a simple Express app I was working on while learning Node.js and Express.js. As you can see this is how the code structure usually will be when following the MVC design pattern while programming in NodeJS Express framework:

You can obviously notice the 'models, 'views', 'routes', and 'controllers' folders.

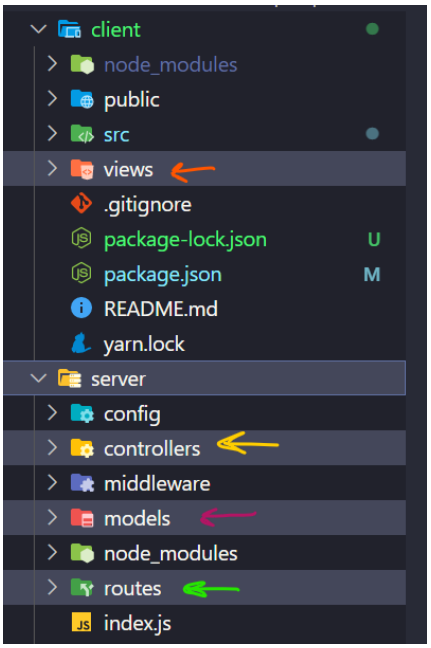

Figure 18: MVC in Express.js

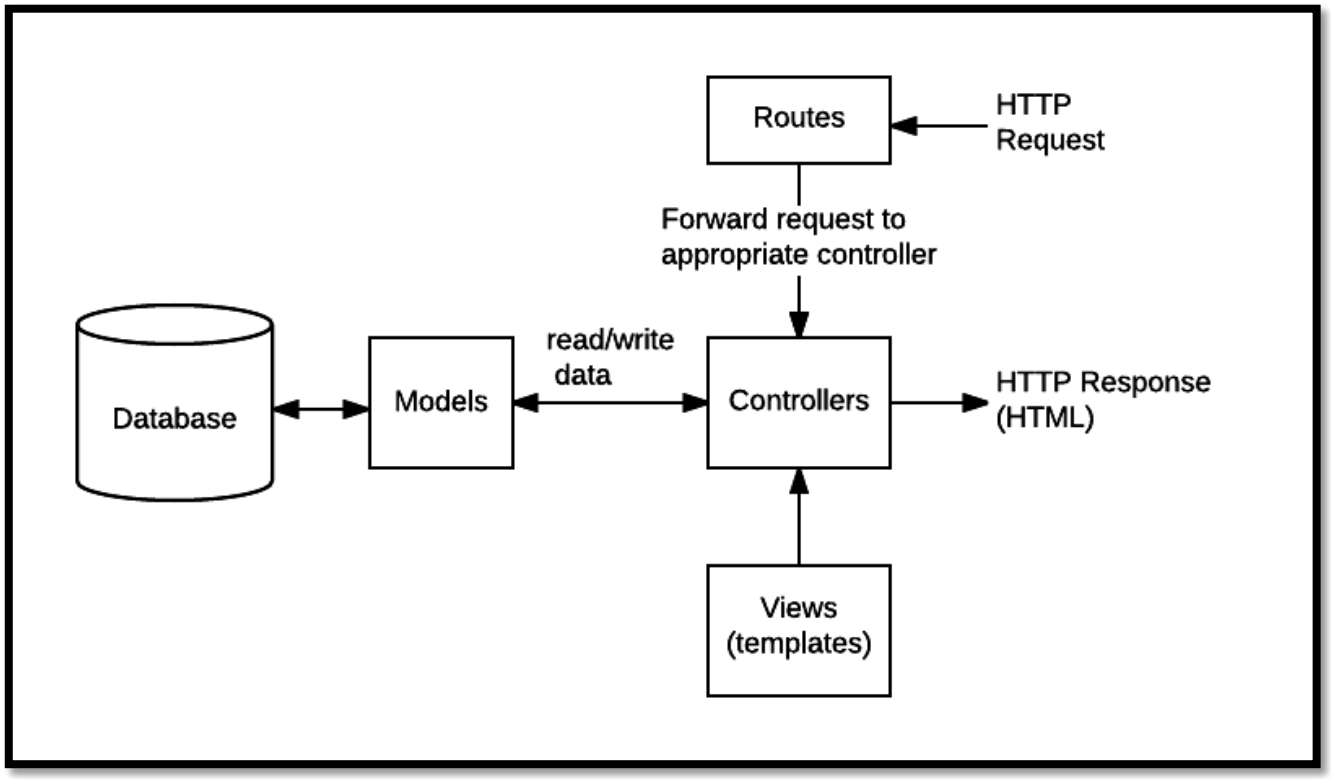

Figure 19: A closer look to interactions between the Models, Views, and Controllers

4.6 Database

4.6.1 MySQL

MySQL is a widely used open-source relational database management system (RDBMS). It is ideal for both small and large applications. A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data.

SQL (Structured Query Language) is a language programmers use to create, modify, and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access, and facilitates testing database integrity and creation of backups. [4.12]

4.6.2 Sequelize ORM

Sequelize is a JavaScript promise-based Node.js ORM (Object–relational Mapping) tool for Postgres, **MySQL**, MariaDB, SQLite, DB2 and Microsoft SQL Server. It features solid transaction support, relations, eager and lazy loading, read replication and more. Sequelize follows Semantic Versioning and supports Node v10 and above.

Figure 20: Sequelize Logo

4.6.2.1 How Sequelize Works

Sequelize is a powerful library in JavaScript that makes it easy to manage a SQL database. Sequelize can layer over different protocols, (PostgreSQL, MySQL, etc.).

At its core, Sequelize is an **Object-Relational Mapper** – meaning that it maps an object syntax onto our database schemas. Sequelize uses Node.JS and JavaScript's object syntax to accomplish its mapping.

Under the hood, Sequelize used with PostgreSQL (or could be MySQL) is several layers removed from our actual database:

1. First, we write our Sequelize, using JavaScript objects to mimic the structure of our database tables.
2. Sequelize generates a SQL string and passes it to a lower-level library called pg (PostgreSQL). (In our project it will be MySQL library).
3. pg connects to your PostgreSQL database and queries it or transforms its data.
4. pg passes the data back to Sequelize, which parses and returns that data as a Javascript object. [4.13]

4.6.2.2 Quick Example, Creating a DB

Sequelize uses a constructor function to connect to your database. It return a Sequelize instance for you to interact with.

Here's all you need to do:

```
const db = new Sequelize('postgres://localhost:3000/yourDatabaseNameHere')
                         //^ Our protocol   ^Path to DB     ^Database name
```

This line creates a new instance of Sequelize, which we have called db. We'll need to pass in a string with the protocol (PostgreSQL), the database's location (an IP or other path), and the database name.

4.6.2.3 From "Sequelize Model" to "Database Schema" and vice versa.

Defining Models:

Sequelize is all about *models*. In database-speak, these are our *schemas* – the shape that our data takes. Your models are both the objects that you'll interact with in your application and the primary tables that you'll create and manage in your database.

Here's how to define a new model called *Album*.

```
var Album = db.define('Album', {
  title: {
    type: Sequelize.STRING,
    allowNull: false
  }
  tracks: {
    type: Sequelize.ARRAY(Sequelize.STRING),
    allowNull: false
  }
  price: {
    type: Sequelize.INTEGER,
    allowNull: false,
    defaultValue: 10
  }
});
```

db.define() defines a new table in our database with the name 'Album'. As our second parameter to define(), we pass in an object that will hold the attributes of our database schemas.
Inside this object, our properties create the names of our columns. Now, our Album table will have a title, price, and tracks column. For each column, we specify *another* object that instructs Sequelize how to create and manage that column.
In our example, we specify the type of each column using Sequelize's data types. This constrains our schemas, allowing attributes to be assigned only certain types of data.
We can (and should) add some validation to our columns as well. [4.14]

4.6.2.4 Promises and Sequelize

Sequelize operates with JavaScript Promises. Promises allow us to escape from callback hell when running asynchronous code in Node.js. It's beyond the scope of this article to go into detail about Promises. Read more about them here.

When we call Sequelize methods like Model.create() or Model.select(), we are interacting with the file system and therefore making an asynchronous call. These methods return a

Promise object, which will resolve or reject based on the successful or failed interaction with your database. [4.14]

4.6.2.5 Summary

The above explanation and code are written only to give an overview of what Sequelize ORM is, how it works behind the scenes, and to show how it facilitates dealing with database (the Models in the MVC) while working on Node.js Express framework. [4.14]

CHAPTER 5

SOFTWARE DESIGN

ANALYSIS OF THE PROPOSED SYSTEM

5.1 Analysis of the Proposed System

5.1.1 Types of Workflows

The system supports following workflows

- Internal Correspondence
- External Correspondence – Incoming
- External Correspondence – Outgoing
- Letter
- Decision
- Report
- Instructions
- Invitation

5.1.2 Actors

This system consists of three actors:

- Administrator
- Employee

  Employees belong to positions in the hierarchy tree, or what so called the organizational structure of the company. This organizational structure will be planned by the company itself. So, depending on the industry and the nature of the company, business roles can range. Here is an example of the organizational structure at our university:

Table 5: Example of The Organizational Structure (PTUK):

ID	Position Name		ParentPositionID
1	President	رئيس مؤسسة	0
2	Vice president	نائب رئيس	1
3	Dean	عميد	1
4	Head of Unit	رئيس وحدة	1
5	Director of Department	مدير دائرة	1
6	Head of Department	رئيس قسم (إداري أو أكاديمي)	1
7	Head of Division	رئيس شعبة	1
8	Deanship of Engineering	عمادة الهندسة	3
9	Deanship of Science	عمادة العلوم	3
10	Deanship of Economy	عمادة الاقتصاد	3
11	Deanship of Arts	عمادة الآداب والفنون	3
12	Department of Computer Engineering	قسم هندسة الحاسوب	8
13	Department of Electrical Engineering	قسم هندسة الكهرباء	8

And so on …

5.2 System's Architecture

This section will present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules.

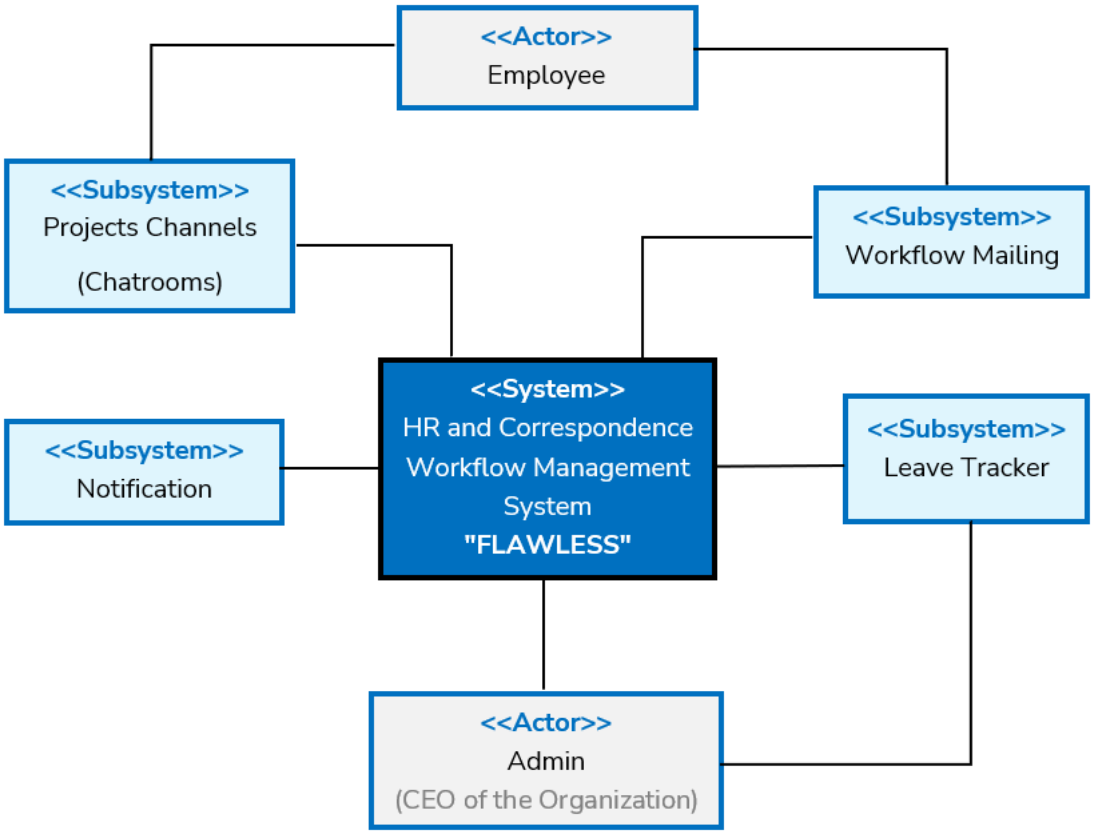

Figure 21: General Overview of our System's Architecture

5.3 Use Case Diagrams

Here we will try to show the interactions between the system and its environment in a simplified way:

5.3.1 Employee's Use Case Diagram

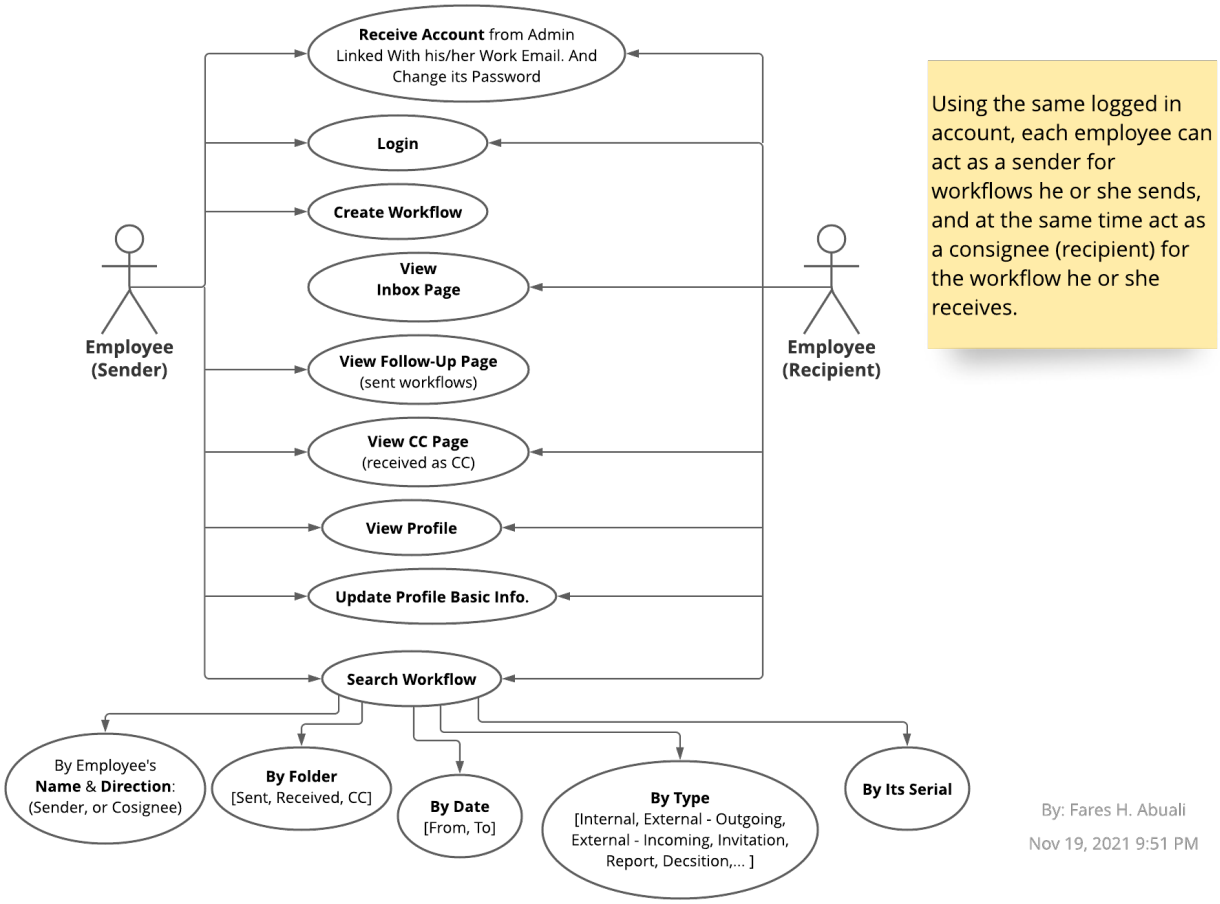

Figure 22: Employee's Use Case

5.3.2 Administrator's Use Case Diagram

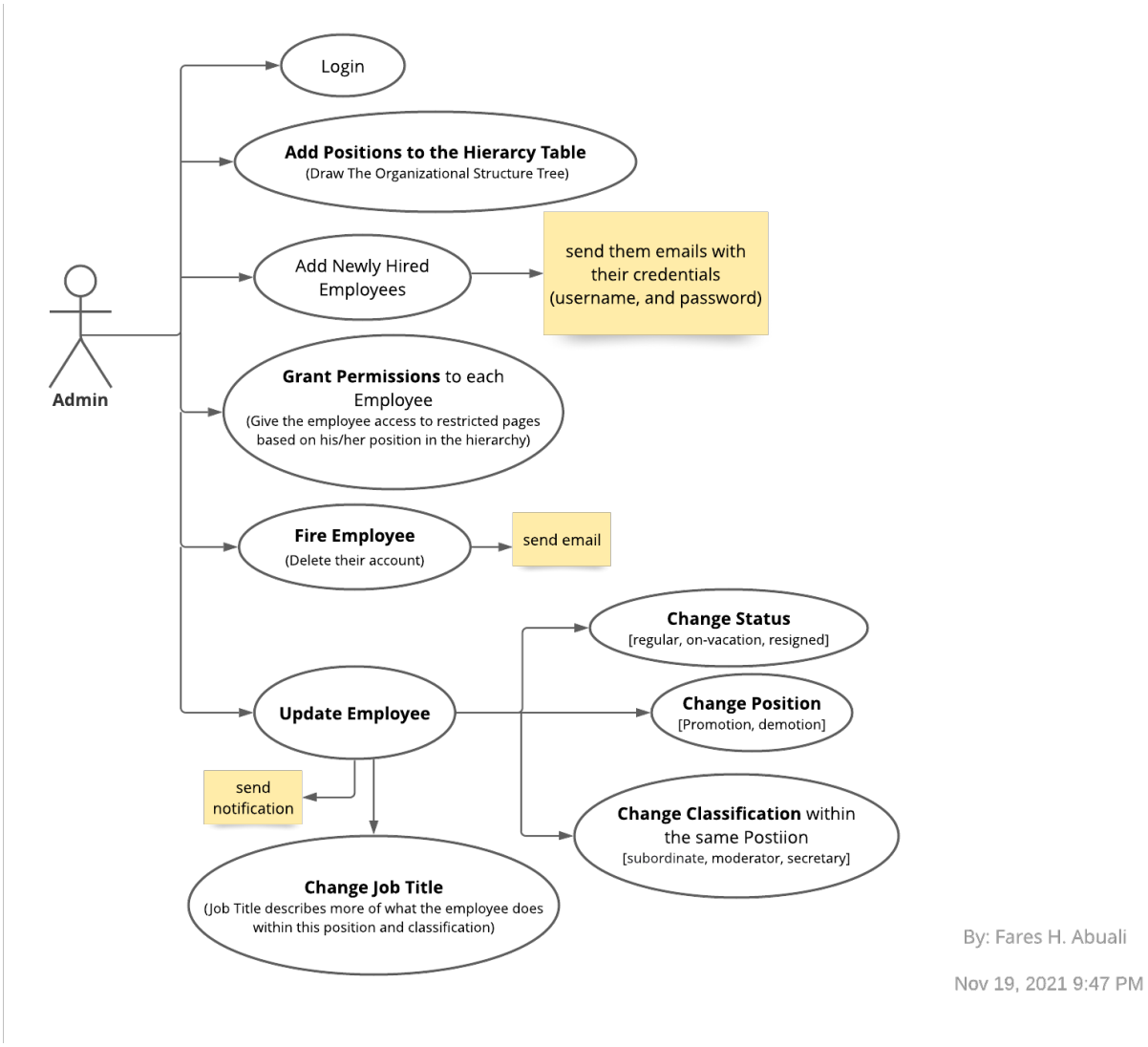

Figure 23: Administrator's Use Case

5.3.3 Brief Description of Use Cases

5.3.3.1 Employee's Use Case Description

Table 6: Employee's Use Case Description

Use Case	Description
Create account (Register)	The employee will be added to the organization automatically once he or she is hired, so it is the mission of the admin to create the account for the employee and give him or her a fixed username that will be used then to login to the system. The scenario will be that after the admin fills in the basic information for the employee (username, email, and randomly generated password), the employee will receive a verification email on his / her email address which contains his or her given username, and a random password. The employee has to click on the link provided in the email, which will redirect him/her to the login page where he can login and change the password.
Can create a workflow	**1.** Must specify the workflow's type (internal, external – incoming, external – outgoing, etc.) **2.** Must fill-in the subject field. **3.** Must specify consignee (recipients) from a dropdown list. Here comes the mission of the sender employee's position, classification, and job title. These factors restrict the sender and allow him/her only to send the workflow to the permitted people based on their position, classification, and job title. This adds levels of security and organizes the process of workflow communication and ensures that the process is working as the organizational hierarchy requires.

	For example, here in the university's HRM correspondence system, if a lecturer wants to send a workflow, he can send it to his co-workers (his siblings in the hierarchy), plus his/her head of the department, which is his/her direct president in the hierarchy. So, if the lecturer tries to send to the faculty's dean, he is not allowed to do so. **1.** He can send to the head of his department, then the head of the department will forward the workflow to the dean. **2.** After specifying the consignee (recipients), the employee starts writing the message body, he can make use of the rich text editor to style the message body. **3.** The employee then can upload files to send them as attachments enclosed with the workflow.
Search workflows	**1.** Search for workflows in a specific folder (Inbox: received, Follow-Up: sent, CC: received as CC). **2.** Search for workflows that were sent in specific time (Date From -> Date To) **3.** Search for workflows of a specific type (internal, external – incoming, external – outgoing, etc.) **4.** Search for workflows which are associated with a specific employee, and whether this employee participates as a sender or recipient. **5.** Search for a specific workflow by its serial number.
View and update his/her profile	Employee's profile show - Employee's info. [Table 7]

5.3.3.2 Employee's Profile Info Description

Table 7: Employee's Profile Info

Use Case	Description
	- firstName - middleName

Personal information	- lastName - email (Unique, used for login) - hireDate - status: [regular, in-vacation, retired] - privilegeGroup: [employee, admin, externalEmployee]
Optional Info Social media links	LinkedIn, Github, Facebook, Twitter
Work History	- Previously occupied positions: [many, with time period (startDate, endDate) for each] - Current assigned positions [many, with start date for each]

5.3.3.3 Workflow's Use Case Description

Table 8: Workflow Details

Use Case	Description
Insertion operation into [Workflow, Actions, and Users] **Tables must be treated as one single transaction:**	- **Each workflow must initially have at least one action**. (When an employee creates a new workflow, the workflow's id must be mentioned in the `actions` table at least once. Because there is no workflow without actions. Each workflow once created, must by initially have one single action caused by the sender. Then when one of the recipients reply, a new action will be added to the same workflow, and so on...) - **Each action must initially have at least two actions.** (When a new action is added to a workflow, the action's id must be mentioned at least in 2 records in the `users` table, one record is the sender, and the second is at the recipient).
Inbox Page	- Show all workflows that are associated with the employee who is currently logged in, ordered descending by their lastUpdate date. (Each time a new action added to the workflow, the workflow's lastUpdate Date is updated)

Inbox Page	- Notice that when an employee has more than one position, For example, academic lecturer could be a lecturer, and at the same time be the head of his department, so here the employee will have only one account in all cases, but at the login page, he can specify whether he wants to login as a lecturer, or as a head of department. - So, based on the position the employee is currently logged in, the workflows shown in his inbox will differ. - Imagining the inbox for an employee who is currently logged in will be something like: SELECT FROM `Workflows` as W JOIN `Users` as U on (W.id = U.workflowID) WHERE U.posIndex = ${position that the employee is currently logged in as} AND U.actionType = 'Recipient' Order By W.updateDate DESC
Follow-Up Page	The same logic as the above 'Inbox Page', but for the sake of classification, here in this page we let the employee see his / her *sent* workflows, while in the Inbox Page we show the received workflows. The same query but change this part: U.actionType = 'Recipient' To: U.actionType = 'Sender'
CC Page	The same logic as the above 'Inbox Page', and 'Follow-Up Page', but for the sake of classification, here in this page we let the employee see workflows that are associated with him as a CC (so he/she is not a recipient in these workflows, instead, he is mentioned in these workflows as CC). The same query as above but change this part: U.actionType = CC

Show the actions of a specific Workflow when clicked:	Imagine the inbox page, it only shows the workflows you received, each workflow with its subject, sender, date, and number of attachments. Now when the employee clicks on a specific workflow to open it, he or she can view all actions associated with this clicked workflow ordered descending from latest to oldest.

5.4 Activity Diagrams

5.4.1 Super-Admin's Activity Diagram

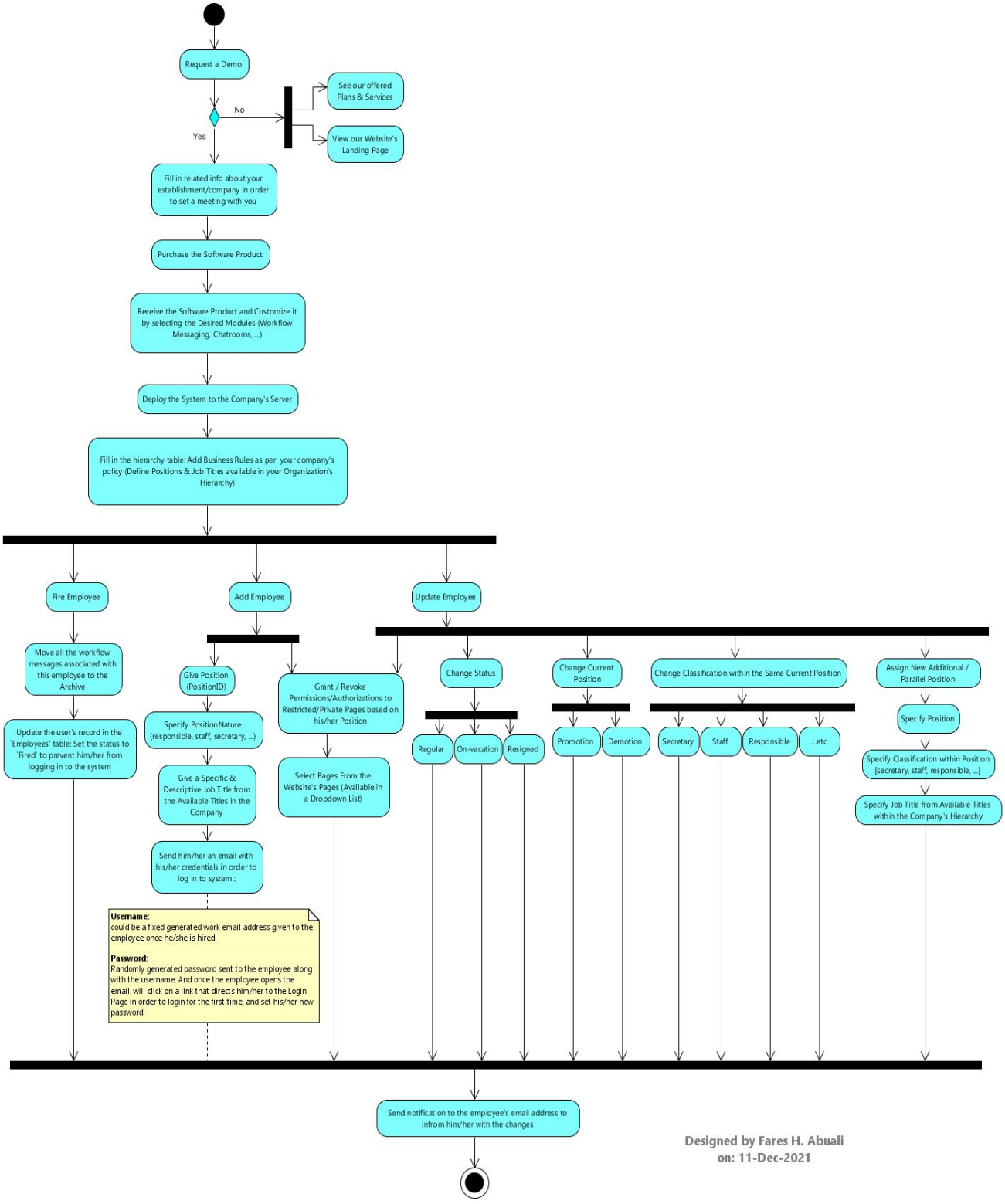

Figure 24: Super-Admin's Activity Diagram

5.4.2 Employee's Activity Diagram

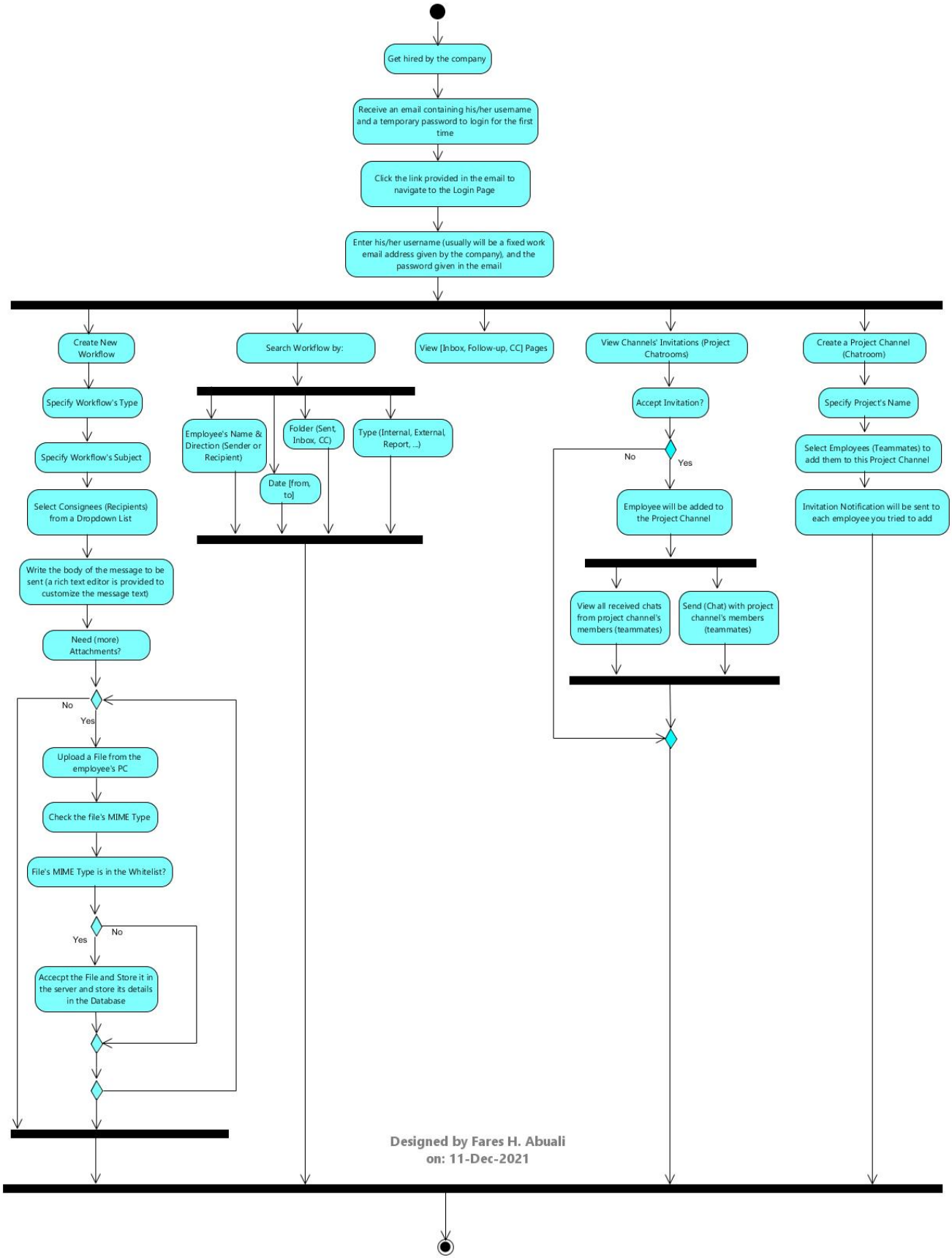

Figure 25: Employee's Activity Diagram

5.5 Sequence Diagrams

5.5.1 Login Sequence Diagram

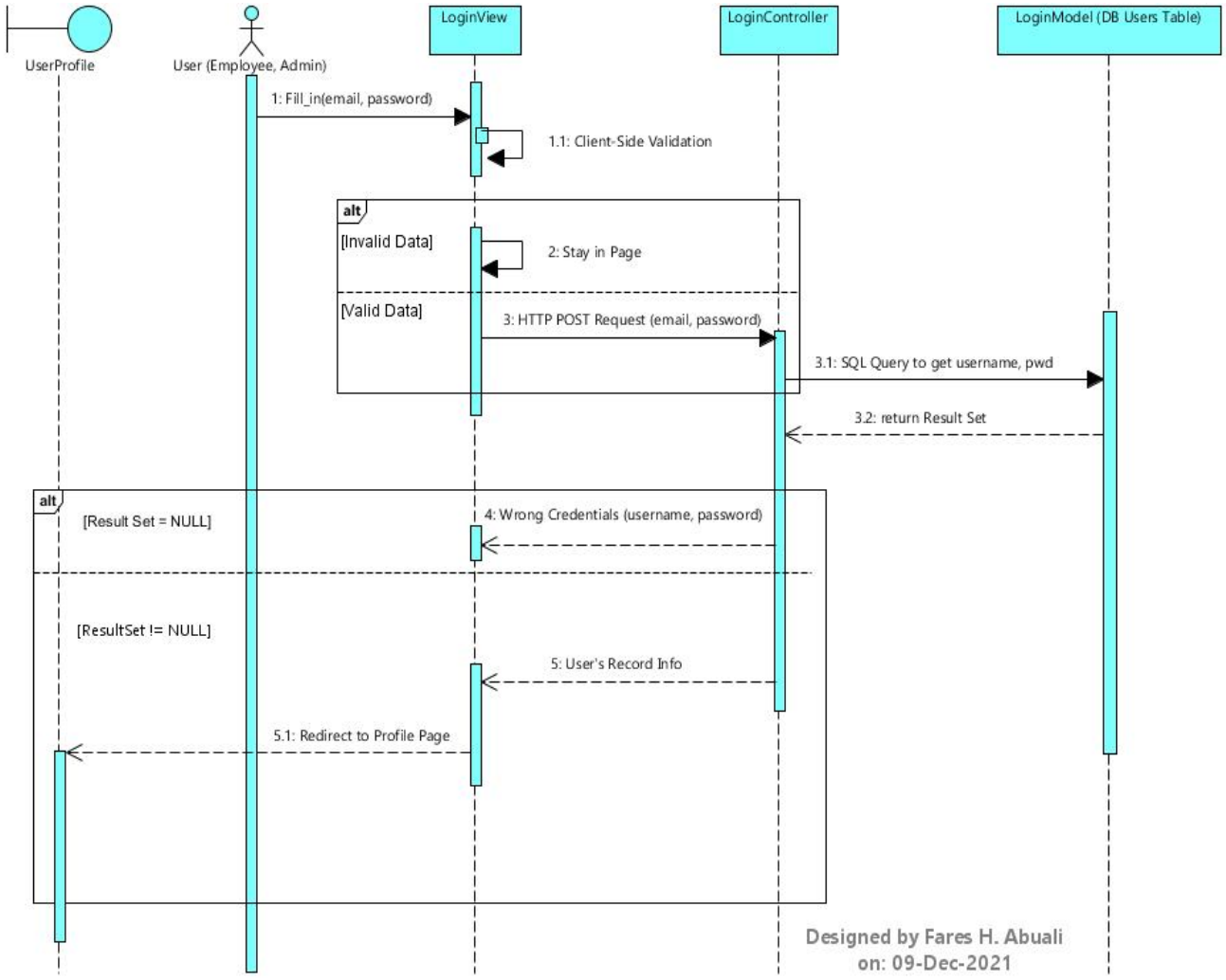

Figure 26: User Login Sequence Diagram

(This applies to all types of users, even admins)

5.5.2 Logout Sequence Diagram

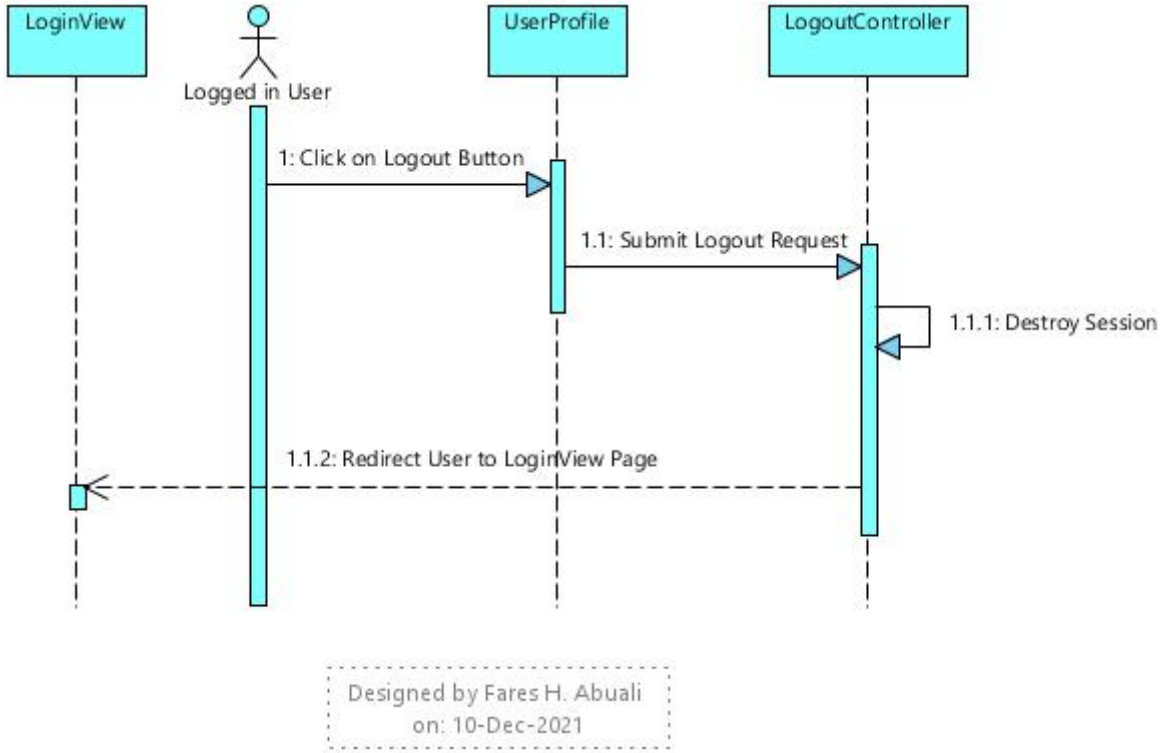

Figure 27: User Logout Sequence Diagram

5.5.3 "Creating a Workflow" Sequence Diagram

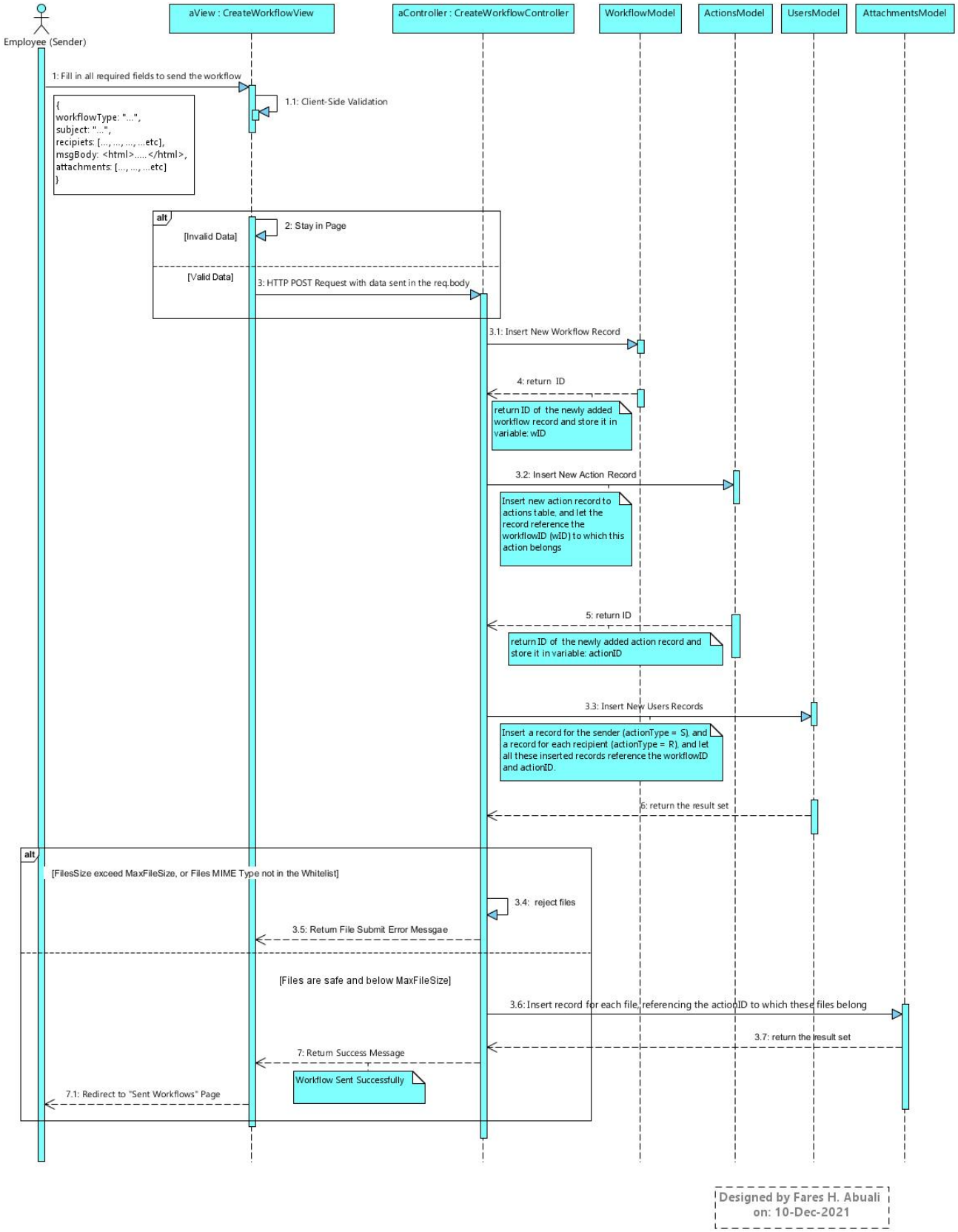

Figure 28: "Creating a Workflow" Sequence Diagram

5.6 Relational Schema Diagrams

*UML Diagrams were designed using Visual Paradigm for UML 8.0 Desktop Software [5.1]

*5.6.1 "Relational Schema Diagram" was designed using MS Word.

*5.6.2 "Closer Look at the Schema Structure" was designed using MySQL Workbench desktop software tool.

5.6.1 Relational Schema Diagram

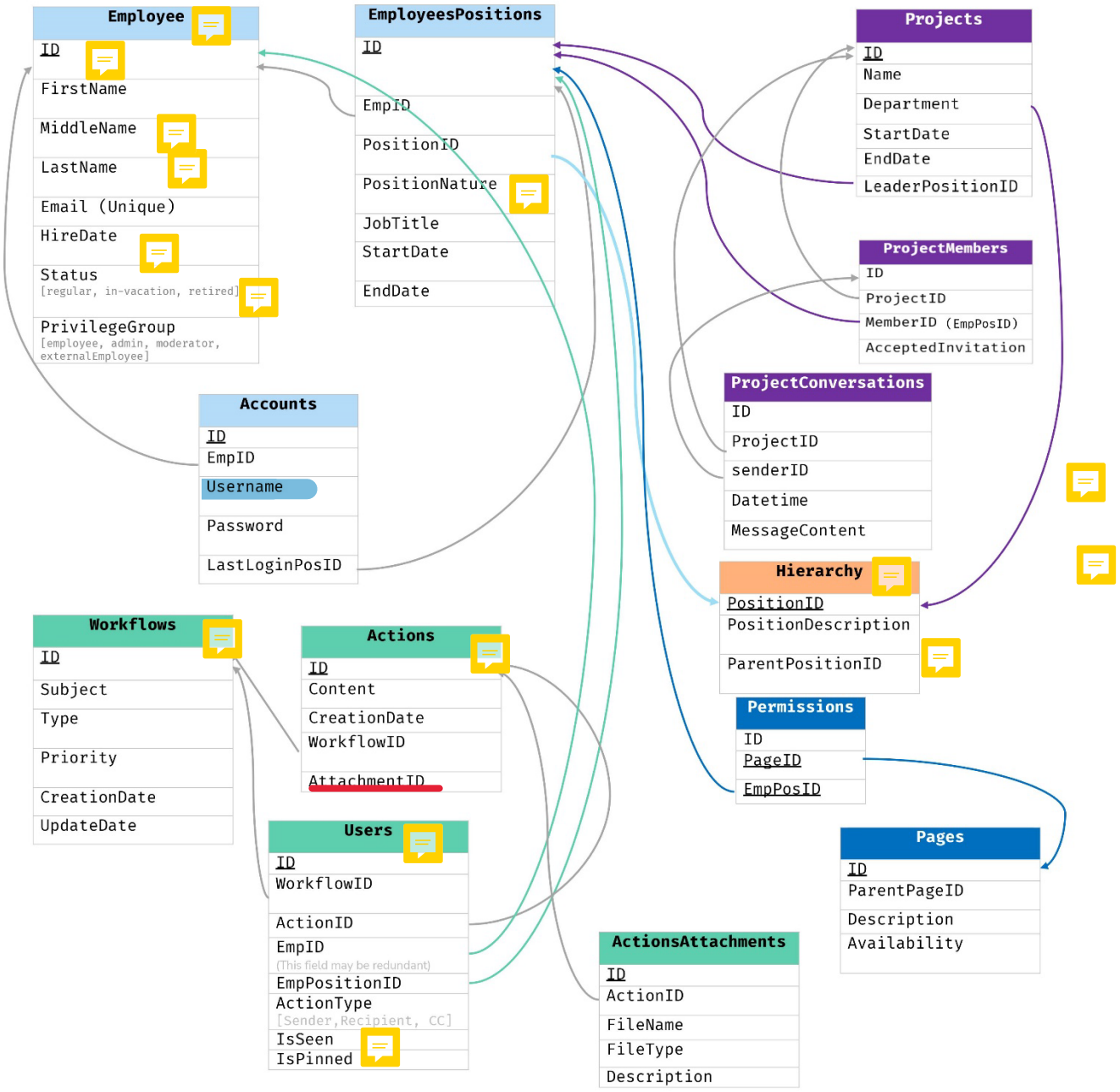

Figure 29: Relational Schema Diagram for the Database of our Graduation Project: "Flawless", A Correspondence Workflow Management & Human Resources Software System (Version 1.0 | 26-Dec-2021)

Note: Different colors of tables and arrows do not reflect any special meaning. These colors are only for the sake of avoiding any confusion may be caused by the overlapping of arrows.

5.6.2 Closer Look at the Schema Structure

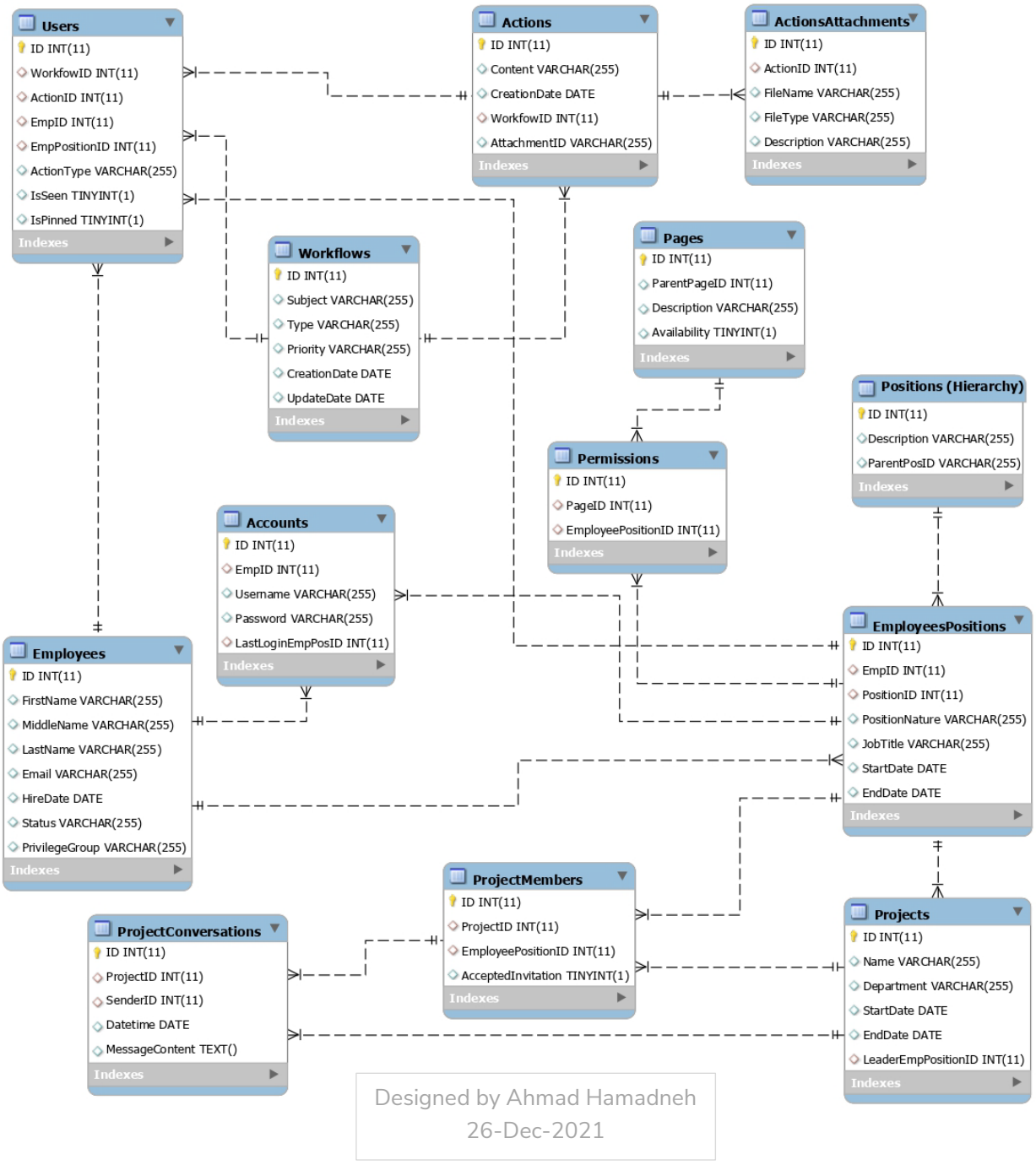

Figure 30: Closer Look at the Relational Schema Structure

5.6.3 Aside Notes regarding the Database

- When a new employee is added to the 'Employees' table, a record for him/her will also be added in the 'Accounts' table. (His email will be his username).

- The field 'privilegeGroup' in the 'Employees' table is added because we want all types of users (Whether a regular employee, or admin, or any other type may be added later), all users stored in one single table 'Employees'.

- The empID field actually references the EmployeesPositions(ID), references the 'ID' field in the EmpoloyeesPositions, not the 'empID' field, because as we said for example, Dr. Mohammad Khalil can have 2 positions (one as a lecturer محاضر, and other as a 'Head of Department' رئيس قسم), and each position has different permissions. So it depends whether he is logged in as a lecturer, or as a head of department (Each position will have different combination of permitted pages).

CHAPTER 6

PLANNING

6.1 Process Model

For our project to be done, we plan to follow the Agile process model. **Agile process model** refers to a software development approach based on iterative development. We are planning to break project tasks into small iterations (scrums), where each scrum will take two weeks. The division of the entire project into scrums helps to minimize the project risk and to reduce the overall project delivery time requirements.

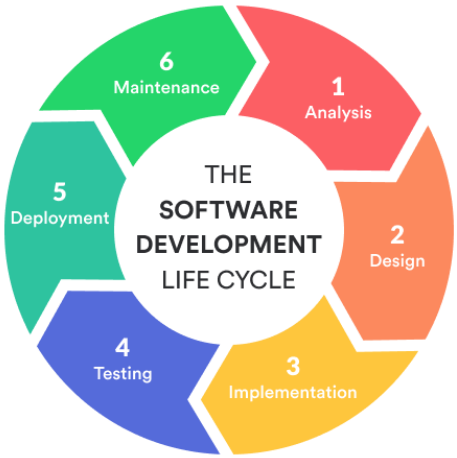

Figure 31: Agile Model Lifecycle

6.2 Time Scheduling

Table 9: Our Project's Time Scheduling

Sprint #	Feature/s
1	Project setup
	Install framework dependencies
	Implement project skeleton
	Implement users' management APIs
2	Implement main layout (header, body, and footer)
	Implement workflow APIs
	Implement common components
	Implement login page

3	Implement Inbound, Outbound, Internal, and related Actions of a Workflow
	Implement organization dashboard APIs
4	Implement organization dashboard #1
	Implement email notifications APIs
5	Implement tasks history APIs
	Implement organization dashboard #2
	Implement mailbox management APIs
6	Implement mailbox management
	Implement task delegation APIs
	Implement search APIs
7	Implement task delegation
	Implement search feature
8	Project deployment and testing

CHAPTER 7

REFERENCES

[1.1] Definitions for correspondence
https://www.definitions.net/definition/correspondence

https://dictionary.cambridge.org/dictionary/english/correspondence

[1.2] Software Engineering HRM System Project Proposal. By Betsegaw Demeke, Admas University, Feb. 2020

https://www.yumpu.com/en/document/read/63076879/hrms-project-proposal-assignment [Accessed: 04-Oct-2021].

[1.3] TRASUL - Correspondence Management System. By Xerox Emirates LLC:

https://www.opentext.com/products-and-solutions/partners-and-alliances/partner-solutions-catalog/partner-solutions-catalog-detail?id=a103z00000FrlxlAAB [Accessed: 04-Oct-2021].

[1.4] Tamkeen Organize, Saudi Correspondence and Document Management System:

https://correspondence.tamkeentech.sa/page/homepage [Accessed: 19-Nov-2021].

[1.5] E-Diwan Correspondence Management System "نظام الديوان الإلكتروني", By Echo Technology:

https://echo-tech.com/pages/viewpage/139/CorrespondenceManagementSystem [Accessed: 19-Nov-2021].

[1.6] Intalio Correspondence Tracking System:

https://www.intalio.com/products/process-management/correspondence-tracking/ [Accessed: 03-Dec-2021]

[1.7] Neologix Correspondence Management System. By Neologix Software Solutions:

https://neologix.ae/solutions/correspondence-management-system-sharepoint-uae/ [Accessed: 19-Nov-2021].

[2.1] Microsoft Outlook

https://en.wikipedia.org/wiki/Microsoft_Outlook [Accessed: 19-Nov-2021].

[2.2] What Is Slack, and Why Do People Love It? By ROB WOODGATE, JUL 17, 2019

HTTPS://WWW.HOWTOGEEK.COM/428046/WHAT-IS-SLACK-AND-WHY-DO-PEOPLE-LOVE-IT/ [Accessed: 22-Dec-2021]

[2.3] Zoho Mail: https://www.zoho.com/mail/ [Accessed: 19-Nov-2021]

[2.4] Zoho Streams - Enhanced collaboration inside your mailbox:

https://www.zoho.com/mail/help/streams.html [Accessed: 24-Nov-2021]

[2.5] Zoho Email retention and eDiscovery

https://www.zoho.com/mail/ediscovery.html [Accessed: 22-Dec-2021]

[2.6] What is Microsoft Teams and who should be using it?

https://www.compete366.com/blog-posts/microsoft-teams-what-is-it-and-should-we-be-using-it/ [Accessed: 24-Nov-2021]

[2.7] Microsoft Teams vs. Slack: Which is Better?

https://www.avepoint.com/blog/microsoft-teams/microsoft-teams-vs-slack-which-is-better/ [Accessed: 24-Nov-2021]

[3.1] Creating an Organizational Structure
https://opentextbc.ca/strategicmanagement/chapter/creating-an-organizational-structure/

[Accessed: 04-Oct-2021]

[4.1] Welcome to the Visual Studio IDE, Article by Microsoft contributors, on 14-Sep-2021:

https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022 [Accessed: 11-Dec-2021]

[4.2] What Makes VSCode so Popular?, by Richard So, on 24-Feb-2021:

https://codeburst.io/what-makes-vscode-so-popular-11e1b3c59ffd [Accessed: 11-Dec-2021]

[4.3] HTML5, by TechTarget Contributor

https://whatis.techtarget.com/definition/HTML5 [Accessed: 12-Dec-2021]

[4.4] Sass, CSS with Superpowers

https://sass-lang.com/ [Accessed: 12-Dec-2021]

[4.5] React (JavaScript library)

https://en.wikipedia.org/wiki/React_(JavaScript_library) [Accessed: 16-Dec-2021]

[4.6] Stack Overflow Trends, Comparison between Angular.js, React.js, and Vue.js

https://insights.stackoverflow.com/trends?tags=reactjs%2Cangularjs%2Cvue.js [Accessed: 16-Dec-2021]

[4.7] What is React Material UI and How to Implement Material-UI in React

https://www.section.io/engineering-education/how-to-implement-material-ui-in-react/ [Accessed: 04-Feb-2022]

[4.8] gulp.js

https://en.wikipedia.org/wiki/Gulp.js [Accessed: 16-Dec-2021]

[4.9] Why the Hell Would I Use Node.js? A Case-by-Case Tutorial

https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js [Accessed: 21-Dec-2021]

[4.10] Express/Node introduction, MDN Web Docs (moz://a). [Accessed: 11- Dec -2021].

https://developer.mozilla.org/en-US/docs/Learn/Serverside/Express_Nodejs/Introduction

[4.11] What is the MVC, Creating a [Node.js-Express] MVC Application, by Islem Maboud, on Oct 09, 2020.

https://medium.com/@ipenywis/what-is-the-mvc-creating-a-node-js-express-mvc-application-da10625a4eda [Accessed: 11- Dec -2021].

[4.12] MySQL

https://en.wikipedia.org/wiki/MySQL [Accessed: 22- Dec -2021].

[4.13] Sequelize npm package [Accessed: 11- Dec -2021].

https://www.npmjs.com/package/sequelize [Accessed: 22- Dec -2021].

[4.14] Sequelize: Step-By-Step, written by zcaceres on Github Gist. [Accessed: 22- Dec -2021].

https://gist.github.com/zcaceres/742744b708393c022703b615d1bffbb1#sequelize-step-by-step

[5.1] Visual Paradigm for UML 8.0 VP-UML is a UML CASE tool supporting full software development life cycle. VP-UML features the latest UML notations

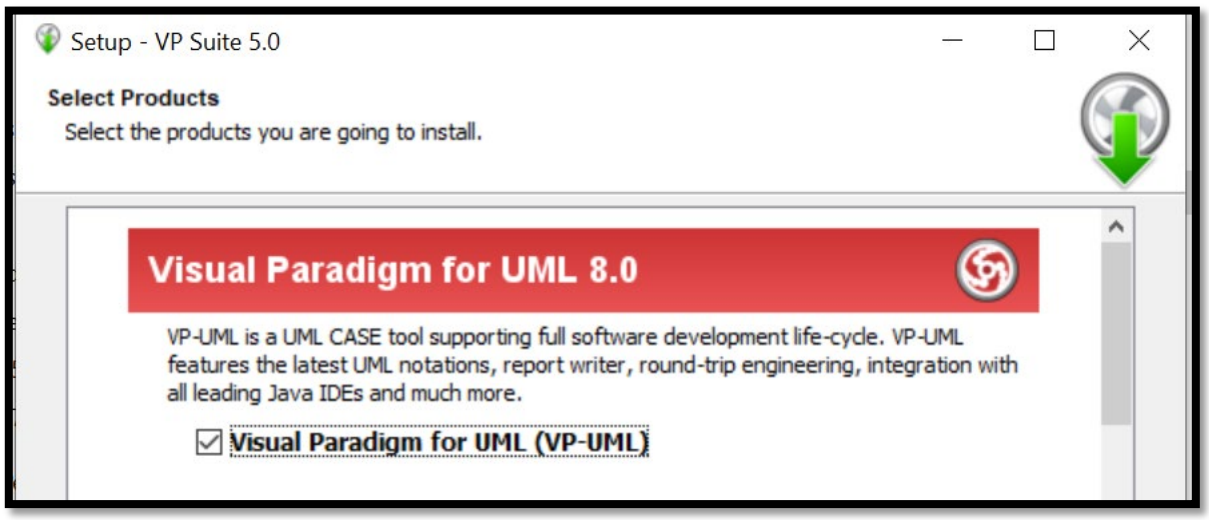

Figure 32: Visual Paradigm Software for UML Diagrams