OULU UNIVERSITY OF APPLIED SCIENCES

Thang Nguyen

COMPANY-WIDE ABSENCES VIEW

COMPANY-WIDE ABSENCES VIEW

Thang Nguyen
Bachelor's Thesis
DIN19SP
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Bachelor's Degree in Information Technology

Author(s): Thang Nguyen
Title of the bachelor's thesis: Company-Wide Absences View
Supervisor(s): Janne Kumpuoja
Term and year of completion: Summer 2022 Number of pages: 46

With the steady growth in company size and team(s) spanning over the world, the need to know whether or not a colleague is present is necessary. In this regard, the prime motivation of this thesis is to propose a company-wide absences view (CWAV) to dynamically have easy access to information and simplify processes.

The first contribution of this thesis focuses on data migration & processing that is permitted under German Federal Data Protection Act (BDSG). Moreover, optimizing data solutions will be beneficial to the application scalability.

In the second contribution, a company-wide absences view is implemented whereby a list of employees will be displayed clearly and nicely in an out-of-office calendar at the first glance. Finally, the main search bar with a date-time filter will help users the best way to find out the information.

Keywords: Company absences view, Out of office tracker

PREFACE

This thesis was carried out at The Qt Company (Oulu-Tutkijantie) office, Oulu, Finland from January 2022. However, this work would not have been done without the patience, encouragement, help, and guidance that I received from many individuals.

First, I would like to express my sincerest acknowledgment to my company supervisor, Mr. Olli Puurunen, for providing me with the opportunity, the best guidance, the vast knowledge, and inspiring ideas. Without his instructions, the thesis could have been unable to be completed.

I would also like to thank my follow-up group, Janne Anttila, Petri Kasper, Nam Nguyen, Merlin Albert, Manigandan Dhamodagan, Kari Miettinen, Matti Tuomikoski, and Pasi Kaapolafrom from the Qt Company for the countless productive discussions and meetings with invaluable advice and supports. It was an honor to work with the team.

I would like to express my sincerest gratitude to my parents, for their endless love, unconditional support, motivation, and guidance in every walk of life. Thanks to them, I had many opportunities to interact with the advanced modern technology in the world.

I further wish to express my gratitude to the pre-manager, Mr. Juha Hautala from Nome Oy, in Finland for his insightful advice and discussions during my bachelor studies.

Success is best when it’s shared. I wish to thank to Quan Do, Kien Vu, and Tai Nguyen for their friendships and memorable moments and the rest who are and were here during the past couple of years. Life in Oulu would not have been as wonderful without them. My special thanks also go to Juha-Pekka and Mai, Ida & Timi Valtonen, and Thanh Nguyen for their friendships and for making our gatherings so memorable.

Last and definitely not least, I would not be standing here without the endless love, support, and inspiration from Trang, my girlfriend for their love and caring, and for being the soulmate and strength of my life. Trang, you rock.

CONTENTS

LIST OF ABBREVIATIONS

AD	Active Directory
API	Application Program Interface
BDSG	Bundesdatenschutzgesetz
CEO	Chief Executive Officer
CPU	Central Processing Unit
CWAV	Company Wide Absence View
DOM	Document Object Model
DPO	Data Protection Officer
HR	Human Resource
HTTP	Hypertext Transfer Protocol
ID	Idem
ISO	International Organization for Standardization
JWT	JSON Web Token
MSAL	Microsoft Authentication Library
NPM	Node Package Manager
SPA	Single Page Application
SSO	Single Sign-On
SWR	State While Revalidate
UI	User Interface
URI	Uniform Resource Identifier

1 INTRODUCTION

Nowadays, a company are tracking vacations, sick leaves, and other kinds of absences in their internal HR system. This data is only accessible to the employee itself, line manager, and HR admins - colleagues in general cannot see it. In addition to filing their status to the HR system, the company encourages people also to notify colleagues of absences themselves, for instance: by setting an Out-of-office status in Microsoft Outlook or blocking out their respective calendars. Anyhow, this is strictly speaking not required, and quite some colleagues forget about it.

People are missing easily accessible out-of-office information for colleagues. Getting an answer to the simple question 'Is a colleague working today?' is difficult. The company has been trying with mailing lists or shared outlook calendars, but it is all not really structured, and far away from a single source that is updated automatically.

Sometimes a business needs to know whether a colleague is available or not. In such a case the only reliable way currently is to check with HR or the respective line manager - this is time-consuming though and assumes that HR and the line manager are available and will promptly reply.

For these reasons, an automated out-of-office view tracker provides a source of truth to help user easy access information and simplifies processes:

- An employee would be able to correct absence information by themselves via a single system.
- Employees would have less informing to do. A phone call to HR would be sufficient to let co-workers know about their absence.
- An automated absence view would be more reliable than additional data entry or individual message.
- No traces left behind by email or chat messages that are occasionally distributed right now.

2 DPO STATEMENT

Out-of-office is sensitive information, and the company wants to protect employee privacy. This chapter describes the privacy of employees and the data protection officer statements.

2.1 Privacy

The characteristics of the processed data in CWAV:

- No additional data would be collected by the company. Absences need to be reported by every employee already.
- View to out-of-office data shall be simplified as much as possible: sick leaves, vacations, or other reasons will be reported as general "out-of-fice" status.
- The system does not need to display extensive information about the past, but past information can be limited to the past 6 weeks.
- No statistics about, for example: number of sick days would be exposed.

2.2 DPO Statement

If it is necessary for business processes to find out as quickly as possible whether a colleague is absent and it is also ensured that no information is given about the specific reasons for the absence of the respective employee, there is no problem in the data processing. If the request only applies to the GmbH and the above criteria are met, it is believed that such data processing is generally permitted under Section 26 of the German Federal Data Protection Act (BDSG). Provided that the cooperation between the individual entities is closely structured, the creation of an overview of employee absences can, in our opinion, be considered permissible in principle. There is no concern about the migration of data from the HR system to another tool, as long as this serves the project purposes and the overview of the plans.

3 THEORETICAL BACKGROUND

This chapter refers to all knowledges which should be learn before developing a CWAV product.

3.1 React.js

React.js is a free and open-source front-end library for building components or user interfaces, maintained by Facebook and a community of developers and companies [1].

React can be used as a base to develop single-page application or mobile applications. However, React.js is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires using additional libraries for routing, as well as the other certain client-side functionality.

React is described as “the V in MVC.” In other words, React.js components act as the view layer or the user interface for your JavaScript applications [2]. Figure 1 shows the lifecycle a React component.

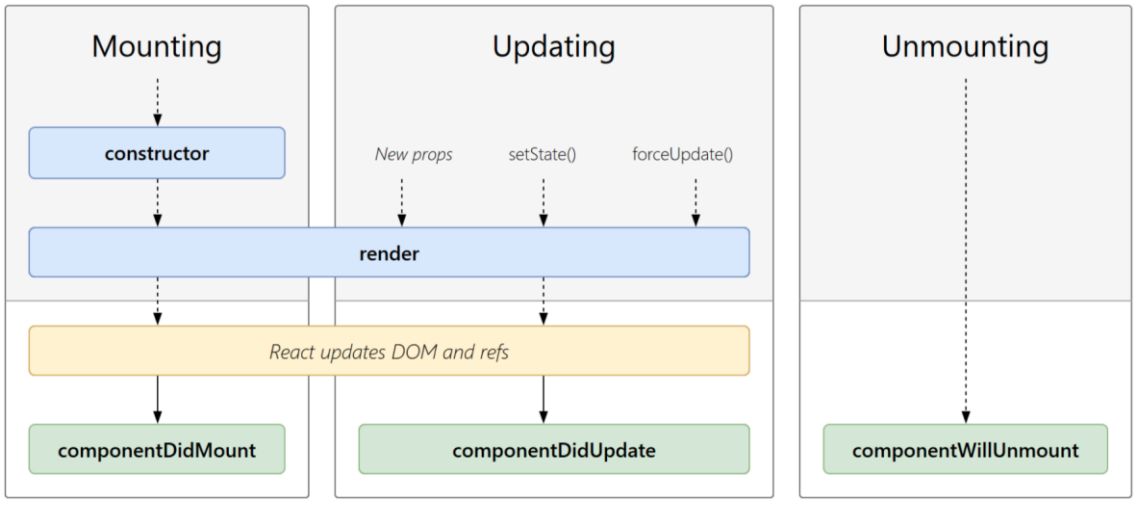


FIGURE 1. React component lifecycle [3.]

React components are created by being mounted onto the DOM, they update, change or grow through updates, and finally, components can be unmounted from the DOM. These three referred milestones are React component lifecycle.

3.2 Redux

Redux is an open-source JavaScript library for centralizing and managing application state. It is most commonly compatible with libraries such as React or Angular for building user interfaces, create logic with powerful capabilities.

Redux is a simple and small library with limited API designed to be a predictable container for application state. It helps developers to write applications that behave consistently, run in different environments, and easy to test. Figure 2 illustrates a Redux lifecycle.

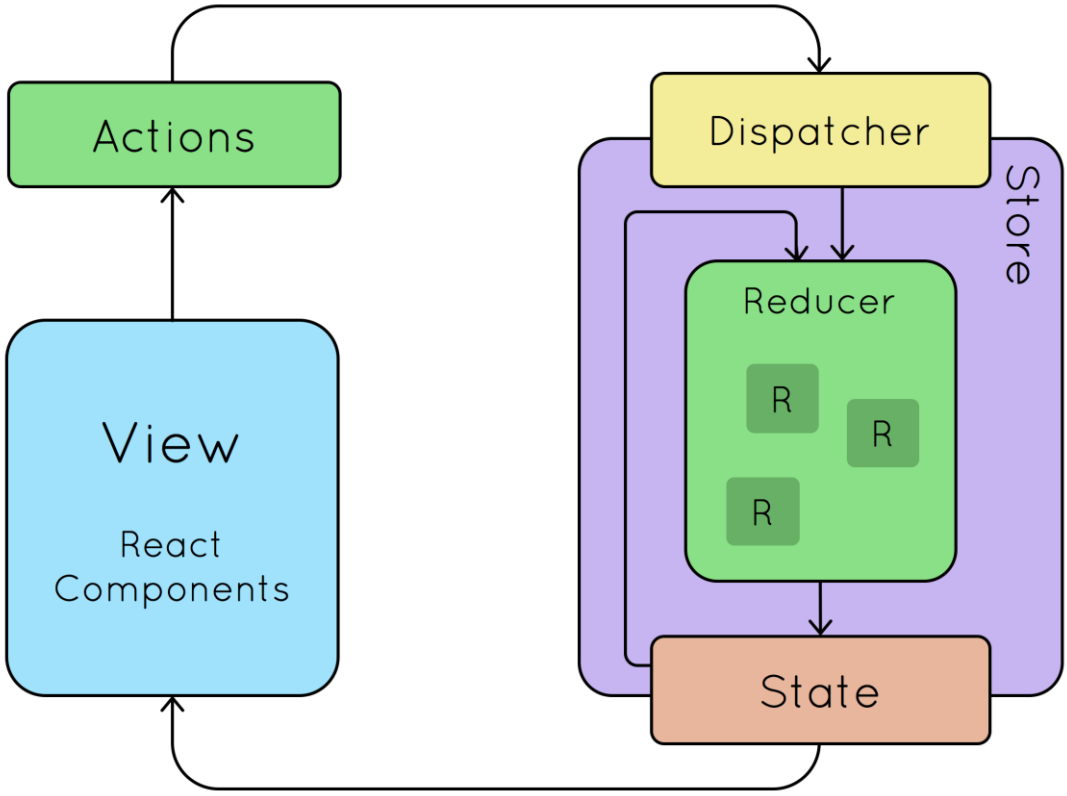


FIGURE 2. Redux component lifecycle [4.]

When an action is dispatched, the root reducer receives it and passes the same action object to all reducers for them to react to it independently. However, instead of passing the entire state tree to each reducer, Redux will only pass the reducer’s exclusive state slice. The return value from each reducer is then merged back into the complete state tree.

This is the basic redux architecture, and it is the most common one as well. With careful planning and smart division of the application state, most small-scale apps can work with this architecture without any added complexities. [5.]

3.2.1 Redux Toolkit

Many aspects of Redux involve writing verbose code such as immutable updates, action types, action creators, and normalizing state. There are some behind-the-scenes reasons why those patterns exist but writing that code "by hand" is difficult. In addition, the process for setting up a Redux store takes several steps, and developers have to come up with their own logic for things like dispatching "loading" actions in thunks or processing normalized data. As the result, many times developers are not sure what the right way is to write Redux logic. [6.]

Therefore, Redux team has created Redux Toolkit that contains packages and functions that are essential for building a Redux app. Redux Toolkit build suggested best practices, simplifies most Redux tasks, prevents common mistakes, and makes life much easier to write Redux applications. Figure 3 shows a structure of the basic Redux toolkit

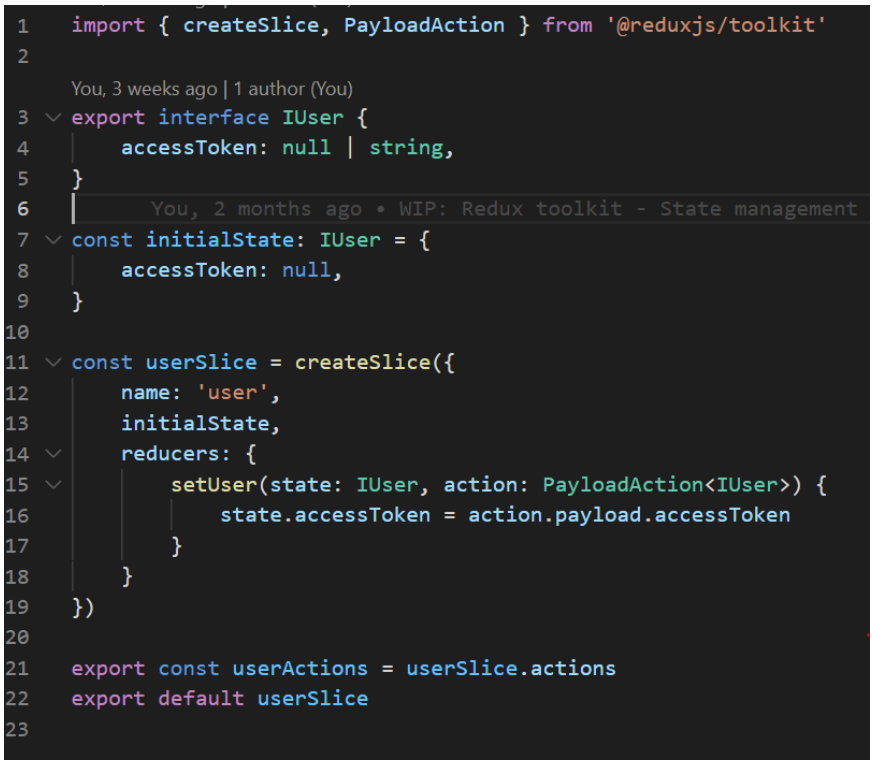


FIGURE 3. A basic template of Redux Toolkit

From the figure, create a Redux reducer is simple. A Redux Toolkit reducer includes utilities to simplify common use cases like store setup, creating reducers, and immutable update logic.

3.2.2 Redux Toolkit Query

Web applications request data from a server in order to use or display it. Also, they need to update new data, send those updates to the server and keep the cached data on the client in sync with the data on the server [7.]. This is made more complicated by the need to implement UI behaviours:

- Tracking loading state in order to show loading spinners.
- Avoiding duplicate requests with the same data.
- Optimistic updates to make the UI faster.
- Manage cache lifetimes as the interaction of users.

RTK Query takes inspiration from other tools that have pioneered solutions for data fetching, like Apollo Client, React Query, and SWR, but adds a unique approach to its API design:

- Data fetching and caching logic is built on top of RTK Query, including how to generate query parameters from arguments and data transformation for caching.
- Generate React hooks that encapsulate the entire data fetching process, provide data and is-loading fields to components, and manage the lifetime of cached data.
- Provides cache entry lifecycle options that enable use cases that help cache updates via WebSocket messages.
- Completely written in TypeScript and is designed to provide an excellent TypeScript usage experience.

3.3 Typescript

TypeScript is an open-source and free programming language developed and maintained by Microsoft. It is a strict superset of JavaScript [8.].

TypeScript is an extension of JavaScript intended to address the deficiency and enable easier development for large-scale JavaScript applications. TypeScript adds class-based object-oriented programming and optional static typing to the JavaScript language. The TypeScript programming language adds optional types to JavaScript, with support for interaction with existing JavaScript libraries via interface declarations.

TypeScript offers a module system, interfaces, classes, and a rich gradual type system [8.]. TypeScript provides a smooth transition for JavaScript developers — well-established JavaScript programming idioms are supported without any need for rewriting or annotations. Below is an example of functions written in Typescript.

```
type filterFn = (item: string) => bool;

// The higher-order-function takes an array and a function as arguments
function filterItems(arr: string[], fn: filterFn): string[] {
    const newArray: string[] = [];
    arr.forEach(item => {
        if(fn(item)){
          newArray.push(item);
        }
    });
    return newArray;
}

function checkNameLength(name: string) {
  return name.length >= 10;
}

const doctorList = ["DoctorOne", "DoctorTwo", "DoctorThree", "DoctorFour"];

// We are passing the array and a function as arguments to filterItems method.
const output = filterItems(doctorList, checkNameLength);

console.log(output); // ["DoctorThree", "DoctorFour"]
```

FIGURE 4. Example of First-class functions [9.]

Here function can also be referred as First-class citizen. A first-class function is treated like any other variable. In this type of programming language, a function can be assigned to multiple variables, and can be passed as an argument to another one and also return one from another function.

3.4 Node.js

Node.js (Node) is currently one of the most popular open-source development platforms for executing JavaScript code on the server-side. Node is said to be useful for building applications that require persistent connections from the browser to the server and versa. It is said to be often used for real-time applications such as chat, news feeds, and web push notifications. [10.]

Node.js is intended to run on a dedicated HTTP server and has a single thread with one process at a time. Node.js applications are event-based and asynchronous. Code built on the Node platform does not follow the traditional model of receive, process, send, wait, and receive. Instead, Node processes incoming

requests in a constant event stack and sends small requests one after the other without waiting for responses. A shift away from mainstream models that run larger, more complex processes and run several threads concurrently, with each thread waiting for its appropriate response before moving on to the next one.

One of the major advantages of Node.js is that it does not block input/output (I/O). Some developers are highly critical of Node.js and point out that if a single process requires a significant number of CPU cycles, applications will block and that the blocking can crash the application. Proponents of the Node.js model said that CPU processing time is less of a concern because of the high number of small processes that Node code is relied on. Figure 5 shows an example of a Node.js server written with TypeScript and Express.

FIGURE 5. Simple Node.js server [11.]

From the figure, a simple Node.js with Express framework server setup is written in TypeScript. The server initializes routes and subsequently listens to a defined secret port.

3.5 Express.js

Express.js (or Express) is known to be a minimal as well as flexible web application framework of Node.js which gives robust features for use of the web as well as hybrid applications [12]. Express.js is also considered an open-source

framework and it was developed and maintained by the foundation of Node.js as well as varied contributors of open source.

Express.js gives a minimal interface and needed tools in order to make our applications. It is also flexible since there exist various modules that are made available on npm and that can be directly plugged into it. Figure 6 shows an example of express route and middleware setup.

FIGURE 6. Setup routes and middleware for an Express.js application

Express middleware is executed in order from the request-response cycle. Every single middleware has access to the request and the response object as

well as a next function which are passed from one another to another. Middleware receives the request, executes the code logic, and finally changes the request and response objects and calls the next function which activates the next middleware in the queue. An express application can do multiple-access middleware, such as: application-level middleware, router-level middleware, error-handling middleware, built-in middleware, and third-party middleware.

In addition, express routes divide the application into several mini express applications and combine them as a parent express application. The express server is composed of an HTTP verb (GET, POST, PUT, DELETE).

3.6 Azure AD SSO

Azure Active Directory Single Sign-On (Azure AD SSO) automatically signs users in when they have identified themself once on their devices. When it is enabled, it is unnecessary to type in any passwords to sign in, and usually, user only needs to type in their email address. It provides users with very easy access to your cloud-based applications without needing any additional on-premises components.

There are many benefits when using Azure AD SSO:

- The user has a better experience. There is no need for re-typing any password.
- Users are not only automatically signed into both on-premises, but also cloud-based applications.
- Application is easy to deploy & administer.
- No additional components are needed on-premises to make the authentication service work.

3.7 MongoDB

MongoDB is an open-source, cross-platform database document-oriented NoSQL database that is used for high-volume data storage. Instead of using

traditional tables and rows from relational databases, MongoDB makes a new usage of collections and documents. Documents consist of the pairs of key-value which are the basic unit of data in MongoDB. Collections contain the number of sets of documents and functions which is the same as relational database tables. MongoDB is a database that came to popular recently around the mid-2000s.

There are some major key features of MongoDB:

- Each MongoDB database contains the number of collections which in turn contain documents. Every single document is different with a varying number of fields. The sizes and contents of the document can be different from each other.
- The document structure fits with how developers construct their classes and objects in their respective programming languages. Developers said their classes are not rows and columns but have a clear structure with key-value pairs.
- The rows (documents in MongoDB) do not need to have a schema defined first. Instead, the fields can be added on the fly.
- The data model available within MongoDB allows developers to represent hierarchical relationships, store arrays, and apply other more complex structures more easily.
- MongoDB is scalable. Companies across the world have defined clusters with run more than 100 nodes with around millions of documents within the database.

Figure 7 illustrates how Relational Database Management System (RDBMS) maps to MongoDB.

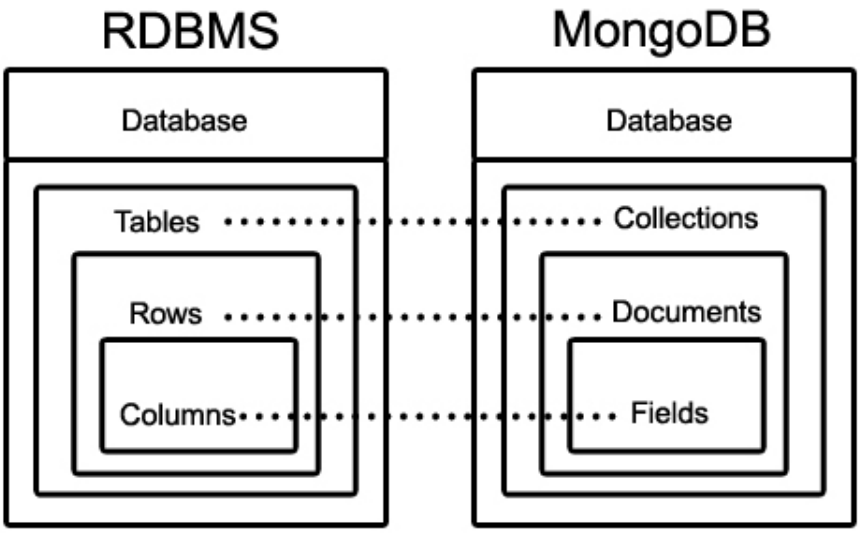


FIGURE 7. Mapping RDBMS to MongoDB [13.]

As can be seen from the figure, Collections in MongoDB is equivalent to the tables in RDBMS, Documents in MongoDB is equivalent to the rows in RDBMS, and Fields in MongoDB is equivalent to the columns in RDBMS. [13]

4 IMPLEMENTATION

Currently the application has come through code, build, and test stages. Three completed stages would be analysed elaborately. They are self-experiences, self-perspective, self-obstacles as well as self-research gathered for solution. Functionalities worth mentioning are user identification, search filtering, data migration & processing, and visual view. Coming to the development, the workflow would be step by step: UI design, database creation and server connection, and application functions.

4.1 UI design

In the scratch and wireframe period, the application design was easy to be utilized covering all the required functions. The user interface was set to be as minimal and convenient as possible. In addition, when designing the interfaces, it also needs to be consistent with the brand design system. With the desire to build professionalism and be aware that colour effects can affect the user's emotions as well as the impact on the meaning of interactive objects in the application, the Qt colour scheme was evaluated. See Figure 8 below.

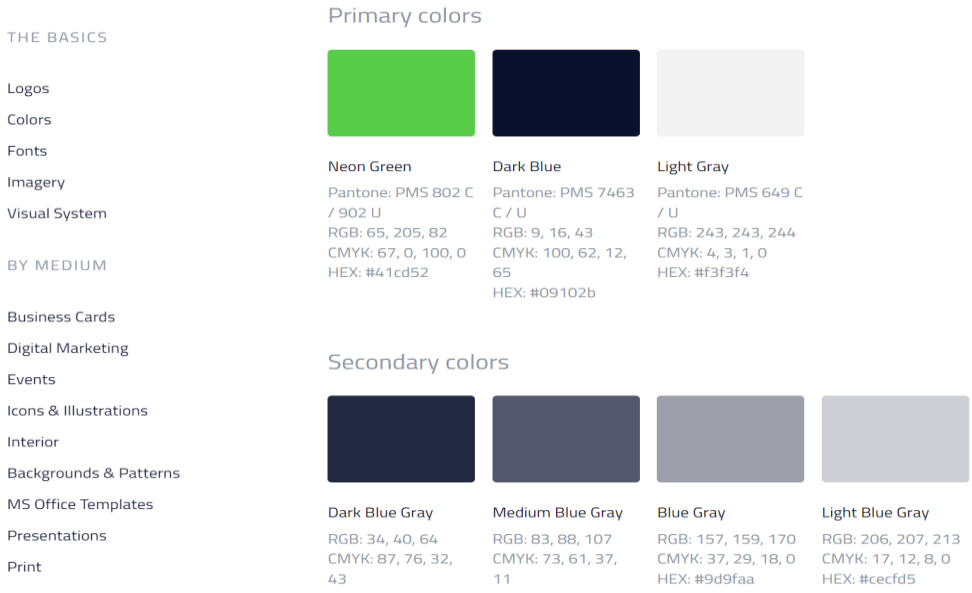


FIGURE 8. Color Palette, Qt brand book

The main screens took the grid out-of-office calendar and the filtering as a pivotal point. The UI design circled around the concept called WYSIWYG. This abbreviation stands for What You See Is What You Get. The design was also constantly evolving by the time.

Figure 9 illustrates how the initial application design has been made.

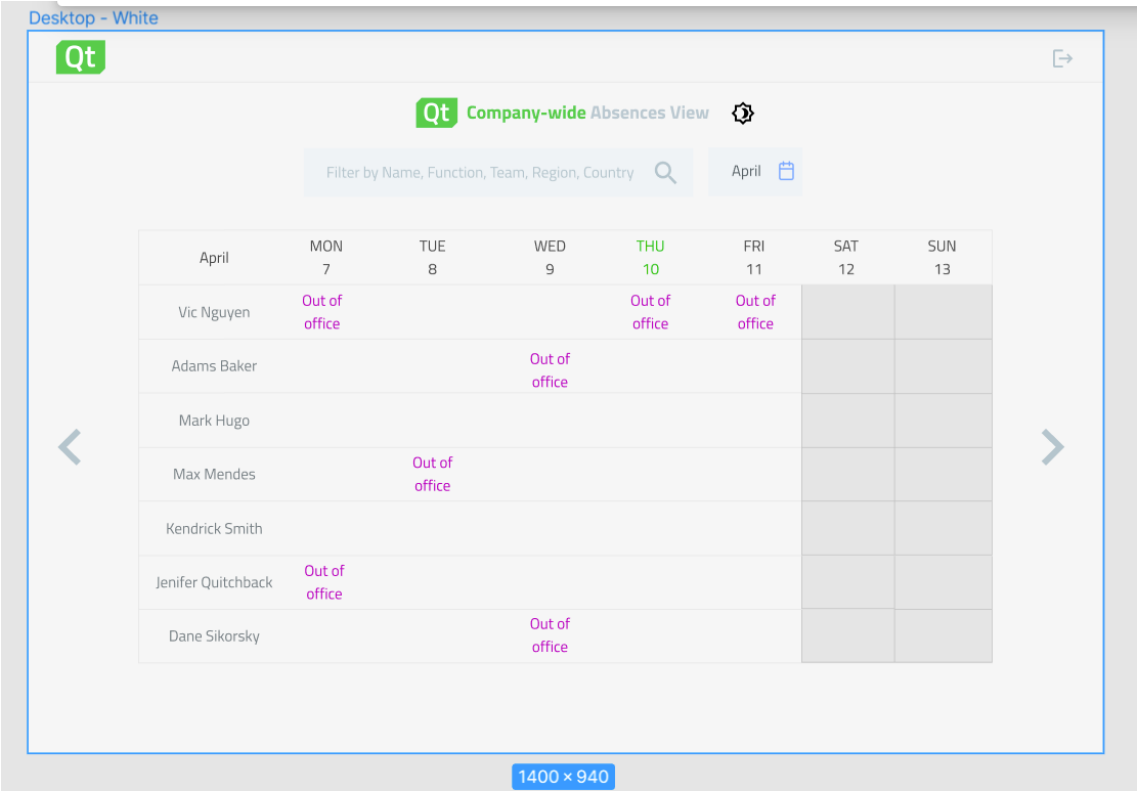


FIGURE 9. Application desktop design

The UI was designed in a way that provides a quick and easy way to see the overall absence status of multiple employees at the first glance.

Everything created was built from small components. Components were interactive building blocks for creating a user interface. There were three main components built from the application: Filtering bar, calendar date picker, and employee grid calendar. The figures below show how these components have been designed in Figma.

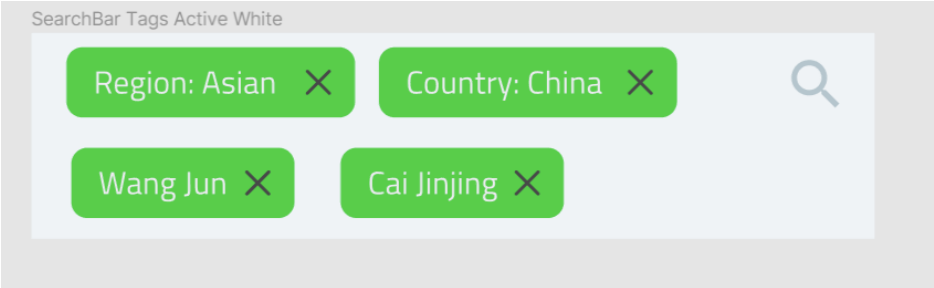


FIGURE 10. Filtering bar

From Figure 10, the filtering bar provided a quick filter that users will have the options suggestion when clicking on the bar. It brought quick data accessibility and avoided typo mistakes.

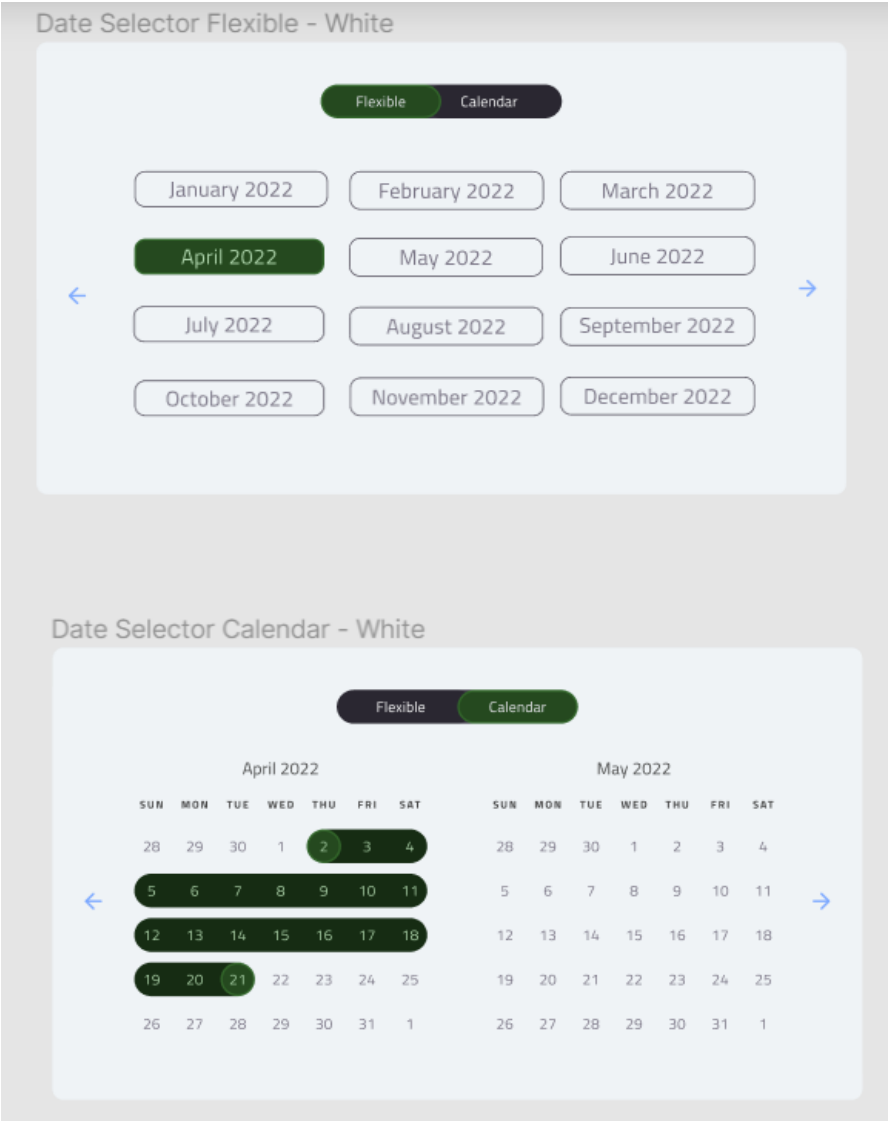


FIGURE 11. Calendar date picker

As can be seen from Figure 11, the calendar date picker allowed users to select a specific view date range or flexible month.

The mobile interface needed to minimize the information displayed, otherwise it would create a negative feeling when being used. The overwhelmed feeling would reduce the emotions forwards the application. Figure 12 shows the minimal design on mobile view.

FIGURE 12. Application mobile view

4.2 Data migration & processing

A human resources software that helps CEOs and HR professionals streamline employee time tracking, time-off management, and performance. Sympa is a complete, fully customizable HR software that helps to focus on people and smart decision-making.

Sympa HR Integration API provides methods for exposing data from the Sympa HR solution, and for importing data to Sympa HR for integration purposes. An administrator can dynamically create interfaces that provide access to basically any data stored in the HR system. Interfaces can, at this point, be configured to return data from and/or modify data in a specific card base.

Sympa HR API has a throttling limitation. The default throttling level is up to a hundred requests per hour time per interface. As a result, the need to migrate employee data is necessary to avoid reaching maximum requests to the Sympa HR API.

Last but not least, since having huge data from the resource, the need to arrange them in a way that can be practically beneficial for the application purpose. It helps to increase productivity and profits, more accurate, and reliable. Furthermore, cost reduction, and ease in storage. This makes it easy to understand and retrieve the specific information anytime.

4.2.1 Data migration from HR software

A company decided to sync employee data from the Sympa HR system to application database by using CronJob. A CronJob creates migrating jobs on a repeating schedule. The figure below describes how the CronJob is configured from the application server.

```
// SympaHR is up to 100 requests per hour.
// We will sync the data every 6 hours - “At minute 0 past every 6th hour.”. Only allow in staging,
if (configs.env !== 'development') {
    new CronJob(
        '0 */6 * * *',
        () => {
            syncDataFromSympaHR()
        },
        null,
        true
    )
```

FIGURE 13. Cronjob for data migration

From the described configuration, the cronjob will be triggered at every sixth hour, that means four times a day. From this context, the function syncData-FromSympaHR() will run, get employee data from Sympa API, process, and store it in the MongoDB database.

Figure 14 explains how the request was made from the server to Sympa HR

```
const getOoOEmployeeFromSympaHr = () => {
    const method: Method = 'GET'
    const { dateToQueryHistoryData, dateToQueryFutureData } =
        sympaHRDateToQuery()

    const sympaBuildQuery = `
        Absences_FIN($filter=(Start_date gt ${dateToQueryHistoryData}) and (End_date lt ${dateToQueryFutureData})),
        Holidays_FIN($filter=(Start_date gt ${dateToQueryHistoryData}) and (End_date lt ${dateToQueryFutureData})),
        Absences_INT($filter=(Start_date gt ${dateToQueryHistoryData}) and (End_date lt ${dateToQueryFutureData})),
        Holidays_INT($filter=(Start_date gt ${dateToQueryHistoryData}) and (End_date lt ${dateToQueryFutureData})),
        Absences_NOR($filter=(Start_date gt ${dateToQueryHistoryData}) and (End_date lt ${dateToQueryFutureData})),
        Holidays_NOR($filter=(Start_date gt ${dateToQueryHistoryData}) and (End_date lt ${dateToQueryFutureData}))
    `

    const config = {
        method,
        headers: {
            'X-ApiKey': configs.sympaHrAPISecret,
        },
        url: configs.sympaHrURL + '[redacted]',
        params: {
            $expand: sympaBuildQuery,
        },
    }

    return axios(config)      You, 2 months ago • Change directory naming: src to server …
        .then((res: AxiosResponse) => {
            logger.debug(`GET Data from SympaHR success, syncing....`)
            return res.data
        })
        .catch((error) => {
            logger.error(
                `GET Data from SympaHR failed. Reason: ` +
                    JSON.stringify(error.response.data)
            )
            logger.debug(`Stop syncing...`)
            return false
        })
}
```

FIGURE 14. Get employee API

Before sending the request to Sympa HR API to get extensive employee information, the system started to build the query that limit the history & future data according to the setting, that are set and can be easily change from the server application.

4.2.2 Data schema

From Sympa HR API configuration, the possibility to restrict the transferable fields to only ones required by the calling system and no unnecessary data is expose. The schema was designed on top of the Mongoose model. See the Figure below.

```
const QtOoOSchema = new Schema({
    Employee_number: {
        type: String,
    },
    'E-mail_address': {
        type: String,
    },
    sAMAccountName: {
        type: String,
    },
    Country: {
        type: String,
    },
    Region: {
        type: String,
    },
    Function: {
        type: String,
    },
    Area: {
        type: String,
    },
    Location: {
        type: String,
    },
    Employee_group: {
        type: String,
    },
    Team: {
        type: String,
    },
    Fullname : {
        type: String
    },
    OoO: {
        type: Array,
        required: true,
    },
    Manager : {
        type: String
    },
    isManager : {
        type: Boolean
    },
    PublicHolidays: {
        type: Array,
    }
})
```

FIGURE 15. Employee out-of-office schema

The schema was designed in a way close to the original data schema from API. There are fields that have been transformed and added to use in the application, such as Fullname (first name and last name), OoO (out-of-office), IsManager (Does the person has managerial role?), PublicHolidays (public national holidays). The referred data transformation made it a better-organized format that would be useful from the application level.

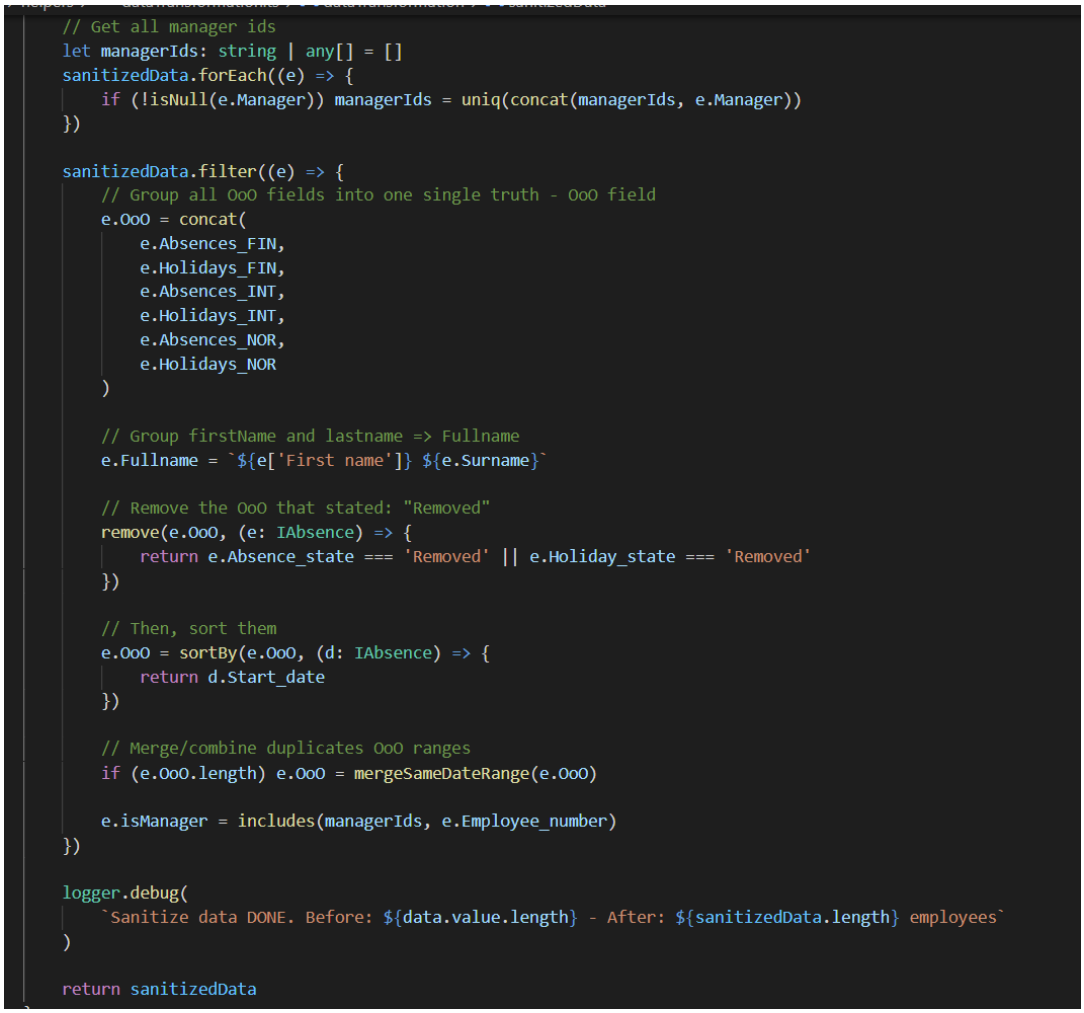
4.2.3 Data processing

Firstly, anonymize data is a type of information that intends privacy protection, and that needs sanitization. Considering anonymous data was simplified: A user without an e-mail address, invalid email domain organization, and all the filterable fields are missing. Figure 16 below illustrates how the data processing was made.

```
const sanitizedData = reject(
    data.value,
    ({
        'E-mail_address': emailAddress,
        Country,
        Region,
        Function,
        Area,
        Location,
        Team,
        Surname,
        'First name' : firstName
    }) =>
        emailIsMissing(emailAddress) ||
        allFilterableFieldsAreMissing(Country, Region, Function, Area, Location, Team, Surname, firstName) ||
        invalidEmailDomain(emailAddress)
)
```

FIGURE 16. Data sanitization function

Secondly, all absences of the employee, such as: sick leaves, vacation or other reasons will be reported as general out-of-office status. Furthermore, the need of converting data to an organized form is warranted. It makes easy to understand and retrieve the specific information anytime. Data processing is described in the Figure 17.



```
    // Get all manager ids
    let managerIds: string | any[] = []
    sanitizedData.forEach((e) => {
        if (!isNull(e.Manager)) managerIds = uniq(concat(managerIds, e.Manager))
    })

    sanitizedData.filter((e) => {
        // Group all OoO fields into one single truth - OoO field
        e.OoO = concat(
            e.Absences_FIN,
            e.Holidays_FIN,
            e.Absences_INT,
            e.Holidays_INT,
            e.Absences_NOR,
            e.Holidays_NOR
        )

        // Group firstName and lastname => Fullname
        e.Fullname = `${e['First name']} ${e.Surname}`

        // Remove the OoO that stated: "Removed"
        remove(e.OoO, (e: IAbsence) => {
            return e.Absence_state === 'Removed' || e.Holiday_state === 'Removed'
        })

        // Then, sort them
        e.OoO = sortBy(e.OoO, (d: IAbsence) => {
            return d.Start_date
        })

        // Merge/combine duplicates OoO ranges
        if (e.OoO.length) e.OoO = mergeSameDateRange(e.OoO)

        e.isManager = includes(managerIds, e.Employee_number)
    })

    logger.debug(
        `Sanitize data DONE. Before: ${data.value.length} - After: ${sanitizedData.length} employees`
    )

    return sanitizedData
```

FIGURE 17. Employee data processing function

As can be seen from the figure, sanitized data was processed step by step: group absences to a sorted and non-duplicated single source of truth - out of office, transform employee full name, and manager state. The result is shown in the Figure 18.

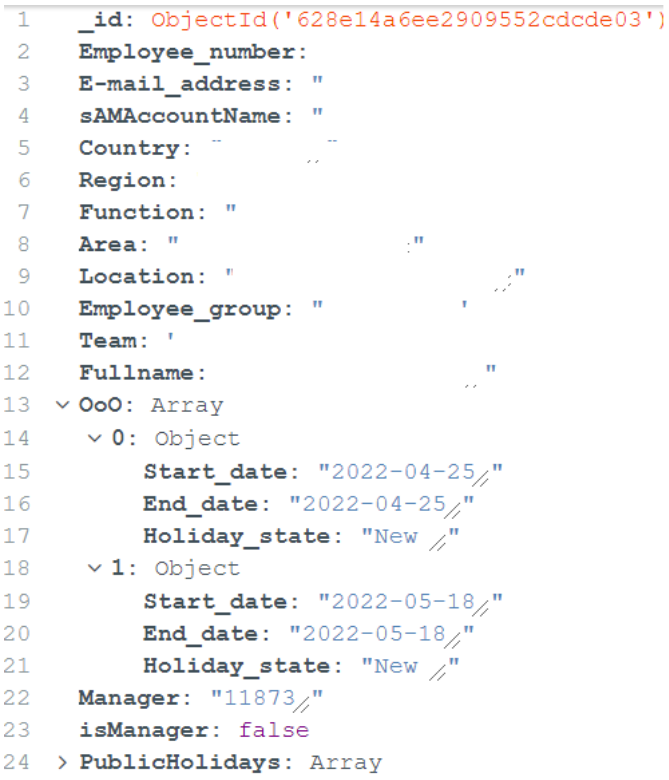


FIGURE 18. Result of data processing

4.3 Server

In this section, the main contributions are focused on Azure authentication & authorization and data responding from the server.

4.3.1 Authentication & authorization

Access tokens allow clients a secure call to protected web API and are used by web APIs to perform authentication and authorization. Microsoft identity platform uses different token formats depending on how the API's configuration accepts the token. APIs registered by developers from the Microsoft identity platform can select from two different formats of JSON Web Tokens (JWTs), v1 or v2.

For validation, developers usually decode JWTs using a site like jwt.io. But unfortunately, the signature from the Microsoft JWT cannot easily be verified because the website does not know what kind of key to use to validate the signature. The header of the JWT does provide information about the algorithm used and the id of the key used but this by itself is not enough to locate the key to be used.

Luckily, thank for the library 'validate-azure-ad-token' that makes the validation steps much easier. Here is the explanation of how the library has made the validation steps:

- Receive token and all required props are passed in
- Decode token with jsonwebtoken library
- Send a request to Microsoft decode key website with the configuration settings to receive all public keys unique only
- Verify all required access token claims
- Guarantee whether the token belongs to the application and can be successfully sent to Microsoft's Graph API
- Compare the ID received from Graph API with the one from the access token payload.

An example of token validation setup can be seen as a middleware from Figure 19 below.

```
export const ensureAuth = async (
    req: any,
    res: Response,
    next: NextFunction
) => {
    const azureJWTValidateOptions: IValidationOptions = {
        tenantId: configs.azureTenantId,
        applicationId: configs.azureClientId,
        audience: configs.azureAudience,
        scopes: configs.azureScopes,
    }       You, 2 months ago • Change directory naming: src to server …

    try {
        const authorization = req.headers['authorization']
        if (!authorization || !authorization.startsWith('Bearer '))
            throw { code: 'credentials_required' }

        const jwtToken = authorization!.split('Bearer ')[1].trim()
        const verifyUser = await validate(jwtToken, azureJWTValidateOptions)
        // to do: Should we store user's session anyway?
        req.user = {
            email: verifyUser.payload.upn,
        }

        return next()
    } catch (error: any) {
        if (error.message) logger.warn(error.message)

        error = {
            name: 'UnauthorizedError',
            code: error.code ? error.code : 'invalid_token',
        }
        next(error)
    }
}
```

FIGURE 19. Example of token validation middleware

From the example, the ensureAuth middleware started to get the access token that linked to the request header. Subsequently, it will be added along with the validation options to the encrypt library to process. If found user, the middleware saves user email to the server and calls next(), otherwise returns found error to the user

4.3.2 Data response with world national holiday

World national holiday

Currently, there were not many tools to detect whether today is a holiday in the specified country. Usually, users by themself need to look up the calendar or

google, and also need to ensure the source is accurate. This was time-consuming work.

The need to know whether the colleagues from the other countries are having a holiday today is necessary. It brings a business many benefits:

- Employees have fewer things to do. A quick check from the tool will be sufficient to know about national country holidays of colleagues.
- No holiday notice e-mails or chat messages that are occasionally distributed from admin countries.

Millions of users use Google Calendar API to track their events. Luckily, it provides accurate worldwide holidays that can be used for the application. Figure 20 is a high-level conceptual architecture of how to implement worldwide national holiday.

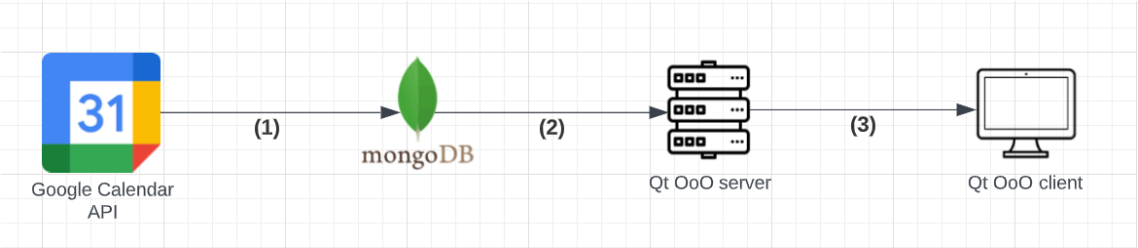


FIGURE 20. Get national holiday architecture

Table 1 explains the national holiday architecture specification.

TABLE 1. The specification for the national holiday architecture

No.	Description/purpose
1	Migrate worldwide national holidays from Google Calendar API to MongoDB database. This also included data processing.
2	Server retrieves employee data that included his/her national holidays from MongoDB database

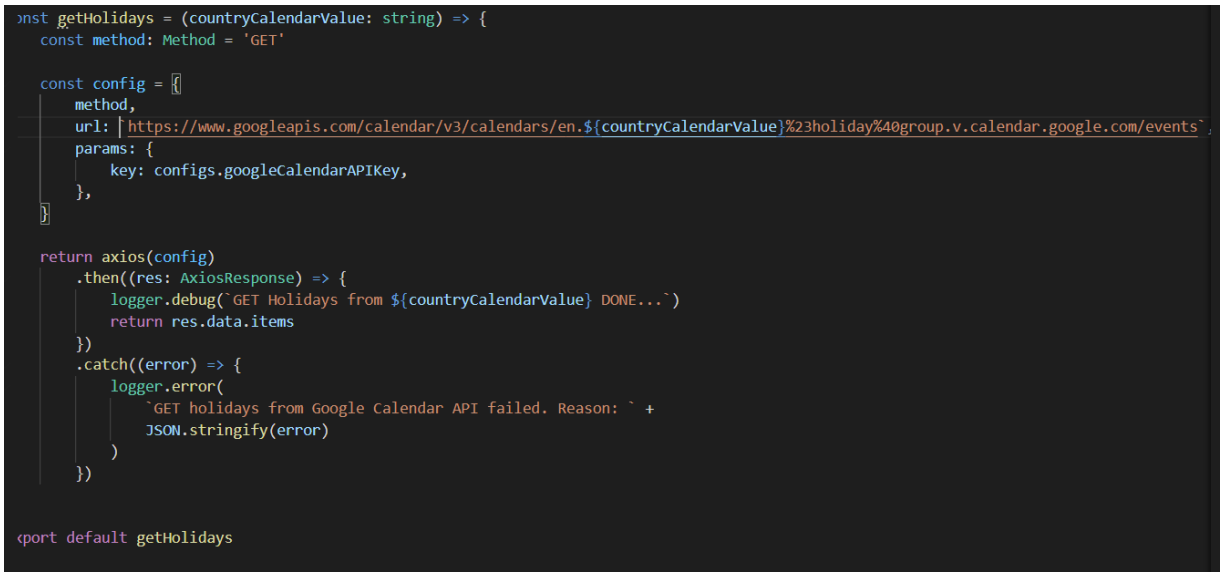
3	Users see colleague's national holidays along with out of office status

Firstly, Google Calendar API requires a list of countries and an API key from the Google cloud platform. The worldwide national holiday migration was carried out by mapping all the world's countries to the requests and send them to the Google Calendar API. Figure 21 is an example of ISO 3166 two-letter country codes with description values for using in Google Calendar API.

```
const languages: any = {
    "ad": "ad",
    "ae": "ae",
    "af": "af",
    "ag": "ag",
    "ai": "ai",
    "al": "al",
    "am": "am",
    "ao": "ao",
    "ar": "ar",
    "as": "as",
    "at": "austrian",
    "au": "australian",
    "aw": "aw",
    "az": "az",
    "ba": "ba",
    "bb": "bb",
    "bd": "bd",
    "be": "be",
    "bf": "bf",
```

FIGURE 21. Example of country code and description used in Google calendar URL

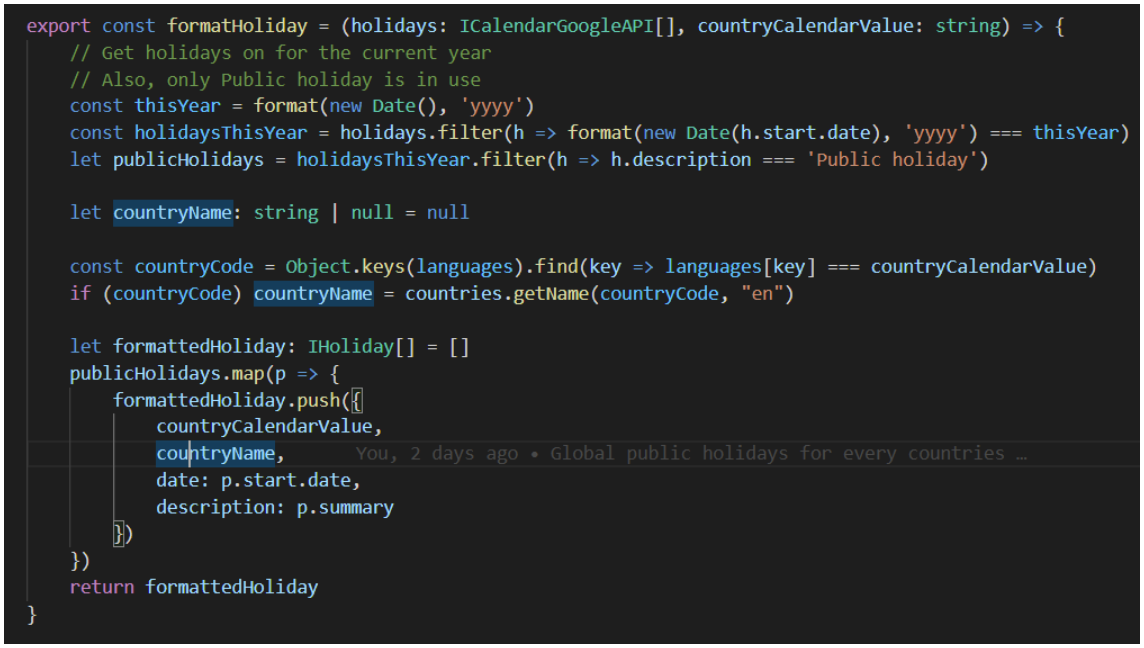
Next, this list is being mapped, and the description value of a corresponding country will be sent to the Google Calendar API to get the holidays. An example can be seen from Figure 22.



```
onst getHolidays = (countryCalendarValue: string) => {
  const method: Method = 'GET'

  const config = {
      method,
      url: `https://www.googleapis.com/calendar/v3/calendars/en.${countryCalendarValue}%23holiday%40group.v.calendar.google.com/events`,
      params: {
          key: configs.googleCalendarAPIKey,
      },
  }

  return axios(config)
      .then((res: AxiosResponse) => {
          logger.debug(`GET Holidays from ${countryCalendarValue} DONE...`)
          return res.data.items
      })
      .catch((error) => {
          logger.error(
              `GET holidays from Google Calendar API failed. Reason: ` +
              JSON.stringify(error)
          )
      })


xport default getHolidays
```

FIGURE 22. Example of Google Calendar API query

After querying worldwide holidays, to minimize the amount of data stored in the database, it will be collected and translated into usable information. See in Figure 23.



```
export const formatHoliday = (holidays: ICalendarGoogleAPI[], countryCalendarValue: string) => {
    // Get holidays on for the current year
    // Also, only Public holiday is in use
    const thisYear = format(new Date(), 'yyyy')
    const holidaysThisYear = holidays.filter(h => format(new Date(h.start.date), 'yyyy') === thisYear)
    let publicHolidays = holidaysThisYear.filter(h => h.description === 'Public holiday')

    let countryName: string | null = null

    const countryCode = Object.keys(languages).find(key => languages[key] === countryCalendarValue)
    if (countryCode) countryName = countries.getName(countryCode, "en")

    let formattedHoliday: IHoliday[] = []
    publicHolidays.map(p => {
        formattedHoliday.push({
            countryCalendarValue,
            countryName,       You, 2 days ago • Global public holidays for every countries …
            date: p.start.date,
            description: p.summary
        })
    })
    return formattedHoliday
}
```

FIGURE 23. Transformation of holiday

From the figure, only the public holidays of the current year are collected and then converted into a more readable format, giving it the form and context necessary to be interpreted by computers and utilized by employees throughout an organization. Once the processing of the worldwide holiday is completed, the application started to save the data into the database, which is going to be used from the application.

Data response

To show the absence of the employee, the combination of out-of-office data with the national holiday is carried out. Figure 24 shows how the data combination is made.

```
QtOooController.get(
    '/',
    middleware.ensureAuth,
    middleware.asyncHandler(async (req: any, res: Response) => {
        const colleagues = await getColleagues(req.user.email)      You, 2 days ago • Global public holiday

        // Merge the National holidays to the data
        // Get all available countries
        let allCountries: string[] = []
        colleagues.forEach((e) => {
            if (!isNull(e.Country)) allCountries = uniq(concat(allCountries, e.Country))
        })

        const allPublicHolidays = await getAllPublicHolidays(allCountries)

        colleagues.filter((colleague) => {
            // Country public holidays
            if (colleague.Country) {
                const countryCode = countries.getAlpha2Code(colleague.Country, 'en').toLocaleLowerCase()
                const countryCalendarValue = languages[countryCode]
                colleague.PublicHolidays = allPublicHolidays.filter(
                    (holiday: IHoliday) => holiday.countryCalendarValue === countryCalendarValue
                )
            }
        })

        res.status(200).json(colleagues)
    })
)
```

FIGURE 24. Data combination between out-of-office data and national holiday

As the result of the figure, all the colleagues and available national holidays of the user will be merged and returned to the request of the client. The holidays will be matched with the country of the user.

4.4 Web client features

In this chapter, the author proposes how the main features have been implemented that return the best user experience and performance. The solution can be extended when the technology has to be changed and updated to a new version.

4.4.1 Azure AD SSO

The SSO method depends on how the application is configured for authentication. Depending on the platform or device the application is targeting, additional configuration may be required such as redirect URIs, specific authentication settings, or fields specific to the platform. The SSO application settings were created from Azure Portal in which the application platform was a single-page application. SPA platform is browser client applications and progressive web applications with modern JavaScript.

Figures 25, 26, and 27 illustrate the basic setup for Azure AD SSO on React.js

```
// Msal Configurations
export const msalConfig = {
    auth: {
        authority: 'https://login.microsoftonline.com/' + configs.azureTenantId,
        clientId: configs.azureClientId,
        redirectUri: configs.azureRedirectURI,
    },
    cache: {
        cacheLocation: 'localStorage',
        storeAuthStateInCookie: false,
    },
}       You, 2 months ago • Azure SSO configuration …

// Add scopes here for ID token to be used at Microsoft identity platform endpoints.
export const loginRequest = {
    scopes: ['user.read'],
}
```

FIGURE 25. Azure AD SSO Settings

From the figure 25, there were mandatory variables need to be provided: application client ID, redirect URI, and tenant ID to enable the SSO from the application level.

Subsequently, a new Microsoft identity platform application was created by using a msal-browser library with the injected Azure setting. See Figure 26.

```
// Azure AD SSO
import { PublicClientApplication } from '@azure/msal-browser'
import { MsalProvider } from '@azure/msal-react'
import { msalConfig } from './configs/msalConfig'

import App from './App'

const msalInstance = new PublicClientApplication(msalConfig)

const rootElement = document.getElementById('root')

render(
    <MsalProvider instance={msalInstance}>
        <App />
    </MsalProvider>,
    rootElement
)
```

FIGURE 26. Basic MSAL provider setup

Then, A Microsoft Authentication Library for React was created with a custom authenticate components. See Figure 27.

```
const App: FC = () => {
    const ErrorComponent: React.ElementType<MsalAuthenticationResult> = (error) => {
        return <BodyContainer>
                <p>An Error occurred while trying to login: {error}</p>
        </BodyContainer>
    }

    const LoadingComponent: React.ElementType<IMsalContext> = () => {
        return <BodyContainer>
            <h5>Authentication in progress...</h5>
        </BodyContainer>      You, last month • CWAV-routes …
    }

    return (
        <MsalAuthenticationTemplate
            interactionType={InteractionType.Redirect}
            errorComponent={ErrorComponent}
            loadingComponent={LoadingComponent}
        >
            <Navbar/>
            <BodyContainer>
                <Routes>
                    <Route path="/" element={<OoOMain />} />
                    <Route path="*" element={<NotFound />} />
                </Routes>
            </BodyContainer>

        </MsalAuthenticationTemplate>
    )
}
```

FIGURE 27. Azure AD SSO Settings

As can be seen from the Figure 27, custom error and authenticate pages were injected into the MsalAuthenticationTemplate, which created a better user experience while authenticating.

4.4.2 Filtering bar

Search bar

After receiving data from API, the filtering component starts to classify search fields and save them into the reducer of the Redux store. Once it is completed, the search fields are provided to React-select options – a select input control for ReactJS with multiselect. The example is shown in Figure 28.

```
export const classifyAndSaveSearchableFields = (OoOEmployees: IOoOEmployee[], dispatch: Dispatch) => {
    const countries = sanitizeField(OoOEmployees, 'Country')
    dispatch(searchFieldAction.setSearchableCountries(countries))       You, 4 weeks ago • Search logic …

    const regions = sanitizeField(OoOEmployees, 'Region')
    dispatch(searchFieldAction.setSearchableRegions(regions))

    const functions = sanitizeField(OoOEmployees, 'Function')
    dispatch(searchFieldAction.setSearchableFunctions(functions))

    const areas = sanitizeField(OoOEmployees, 'Area')
    dispatch(searchFieldAction.setSearchableAreas(areas))

    const locations = sanitizeField(OoOEmployees, 'Location')
    dispatch(searchFieldAction.setSearchableLocations(locations))

    const teams = sanitizeField(OoOEmployees, 'Team')
    dispatch(searchFieldAction.setSearchableTeams(teams))

    const fullname = sanitizeField(OoOEmployees, 'Fullname')
    dispatch(searchFieldAction.setSearchableFullname(fullname))
}
```

FIGURE 28. Azure AD SSO Settings

From the code example, it started to sanitize the raw data and save all the filter options for the filtering function. The result flow can be seen from Figure 29.

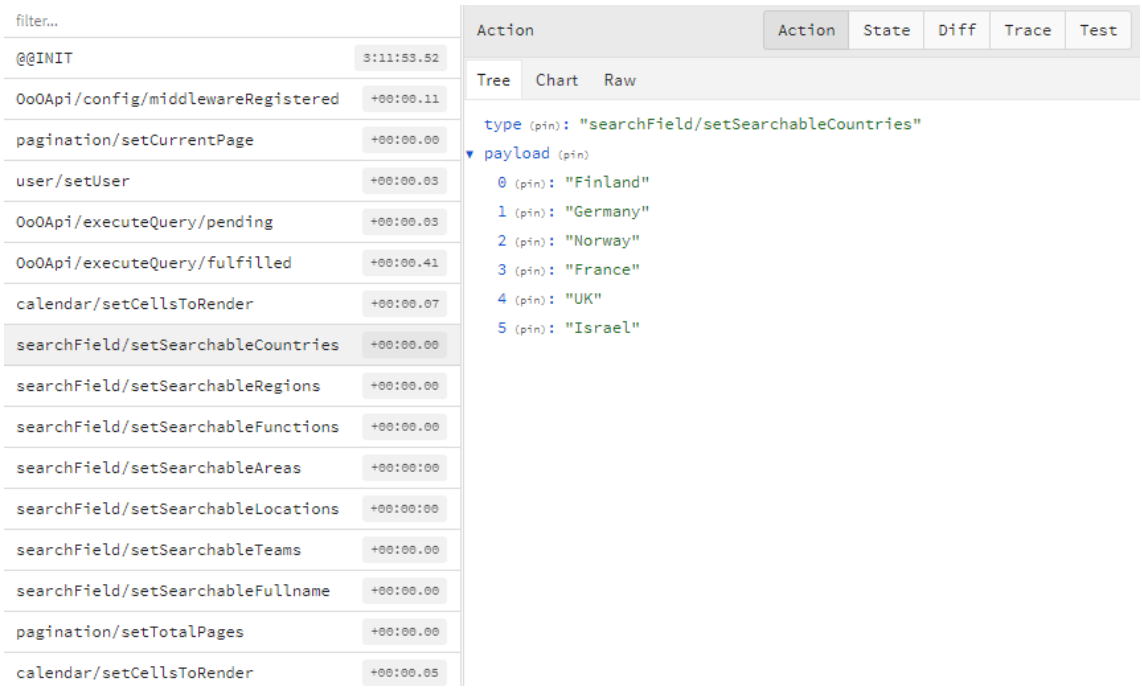


FIGURE 29. Application actions from Redux Dev tool

React-select supports customize themes and formatted display options that bring an important part to UI design and user experience. From Figure 30 below, a custom theme was used for the search filter component.

FIGURE 30. A custom search filter component

Calendar date picker

The calendar date picker was carried out by using react-datepicker - a simple Datepicker component for React.js. See from Figure 31.

```
<DatePicker
    selected={currentDate}
    // Remove selected day repeat to all months
    disabledKeyboardNavigation
    onChange={(date) => {
        if (date) {
            dispatch(calendarActions.setCurrentDate(date))
            dispatch(calendarActions.setCurrentMonth(date))
            dispatch(calendarActions.setCurrentWeek(getWeek(date)))
        }
    }}
    dateFormat="MMMM d, yyyy"
    customInput={<DateSelectionButton />}
    showMonthDropdown
    showPopperArrow={false}
    minDate={subDays(new Date(), configs.daysToTravelHistoryData)}
    maxDate={addMonths(new Date(), configs.monthsToTravelFutureData)}
```

FIGURE 31. Example of calendar picker

From the figure, onSelect event handler fires each time some calendar date has been selected, that triggers to set the current date view to the selected date.

4.4.3 Employee grid calendar

An employee grid calendar is a system to organize absence days. It includes the current range of days and a list of planned employee out-of-office events and holidays.

Day's component

The basic concept to implement the days component was a simple loop, by increasing the start day until it reaches the end day. Figure 32 illustrates the pseudo code of printing dates.

```
# Print the N days
startDate   = new Date('2022-05-20')
endDate     = new Date('2022-05-30')
while (startDate < endDate) {
    # Print this date
    print(startDate)

    # Increase one day
    startDate = addDays(startDate, 1)
}
```

FIGURE 32. Pseudo code of printing dates loop

The first element will be displayed in a month and year format from the current date range, also the current date was highlighted by the green colour. See in Figure 33.

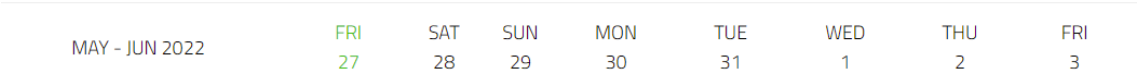


FIGURE 33. Day's component

Employee calendar grid

The solution was made step by step. Firstly, the name of the employee and avatar was displayed to give the user a quick glance at the status of the employee. Subsequently, the logic started to print temporary working days, then overrides with absences, holidays, and weekends. Figure 34 explains the rendering logic by a pseudo code.

```
# Employee calendar grid logic
let employeeRow = []

let startDate = new Date('2022-05-05')

// First, print all the working days, do not care.
for (let cell = 0; cell < cellsToRender; cell++) {
    employeeRow.push('working')
    # Now employeeRow has all 'working' state
}


for (let cell = 0; cell < cellsToRender; cell++) {

    // Override with the absences
    for (let j = 0; j < eachOoODays.length; j++) {
        if (isSameDay(startDate, eachOoODays[j])) {
            employee[cell] = 'absence'
        }
    }

    // Override with the holiday
    for (let j = 0; j < PublicHolidays.length; j++) {
        if (isSameDay(startDate, PublicHolidays[j])) {
            employee[cell] = 'holiday'
        }
    }

    // Override if this day is weekend
    if (isWeekend(startDate)) {
        employee[cell] = 'weekend'
    }

    # done logic, plus one day
    startDate = addDays(startDate, 1)

}
```

FIGURE 34. Rendering absence logic

From the employee grid, by hovering or toggling the grid UI, the annotation will pop up that display useful information to the user. Figure 35 shows an example of when the user interacts with the national holiday cell.

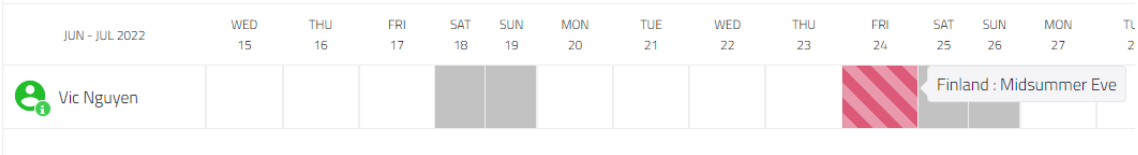


FIGURE 35. Employee calendar grid

5 CONCLUSION

The aim of this thesis was to create a web application using a modern web development framework. This application serves as the company's private tool for tracking employee absences for the Qt Company. The objective of this project was to provide a single source of truth provide easy access to absence information and simplify processes.

At the end of the project, all the major requirements were met, and the development processes were explicated in a shape order. In the process of designing and implementing, the application is constantly being optimized and evolving. Subsequently, the final result was much better than was originally expected. The simple user interface is considered a favourable aspect, not only allowing the user to find their information the way faster but also a better experience.

The project developed serves as an example to give rise to future works that involve task assignments in a practical way to solve everyday problems. The production of this graduation thesis is a process of learning and improving. All of this not only enriches the knowledge structure but also expands the knowledge and enhances the potential for analysis and problem-solving. Last by not least, it creates a potential for self-studying and lays the foundation for future work.

REFERENCES

[1] Tutorials Point Pvt. Ltd. "ReactJS – Overview," *Tutorials Point Pvt. Ltd*, 2022. [Online]. Available: https://www.tutorialspoint.com/reactjs/reactjs_overview.htm. [Accessed May 27, 2022].

[2] R. Savaram, “What is ReactJs,” *mindmajix.com*, 2022. [Online]. Available: https://mindmajix.com/introduction-to-react-js. [Accessed Oct. 27, 2021].

[3] K. Guerra, “The Lifecycle of a React Component,” Jan. 5, 2021. [Online]. Available: https://medium.com/codex/the-lifecycle-of-a-react-component-8e01332a068d. [Accessed Oct. 27, 2021].

[4] M. Bergh, “React Redux: Building Modern Web Apps with the ArcGIS JS API,” Sept. 8, 2017. [Online]. Available: https://www.esri.com/arcgis-blog/products/3d-gis/3d-gis/react-redux-building-modern-web-apps-with-the-arcgis-js-api/. [Accessed May 27, 2022].

[5] S. Zibbu, “Redux Lifecycle,” Aug. 3, 2018. [Online]. Available: https://hackernoon.com/redux-lifecycle-8e93908af03a. [Accessed May 27, 2022].

[6] “Redux Fundamentals, Part 8: Modern Redux with Redux Toolkit”. [Online]. [Accessed May 27, 2022]. Retrieved from https://redux.js.org/tutorials/fundamentals/part-8-modern-redux.

[7] “RTK Query Overview”. [Online]. [Accessed May 27, 2022]. Retrieved from https://redux-toolkit.js.org/rtk-query/overview#motivation.

[8] Bhattacharyya. S, Nath. A, “Application of TypeScript Language: A Brief Overview,” *IJIRCCE journal in Computer and Communication Engineering,* no. 1, p. 10585, 2021. [Accessed May 27, 2022].

[9] P. Behera, “Functional Typescript,” Jun. 4, 2021. [Online]. Available: https://blogs.halodoc.io/functional-typescript/. [Accessed May 27, 2022].

[10] J. Denman, “What is Node.js?”. [Online]. Available: https://www.techtarget.com/whatis/definition/Nodejs. [Accessed May 27, 2022].

[11] J. Vadgama, “A minimal web application structure with technologies like node.js, typescript, express,” Apr. 25, 2018. [Online]. Available: https://www.linkedin.com/pulse/minimal-web-application-structure-technologies-like-nodejs-vadgama/. [Accessed May 27, 2022].

[12] P. Pedamkar, “What is ExpressJS?”. [Online]. Available: https://www.educba.com/what-is-expressjs/. [Accessed May 27, 2022].

[13] P. Pedamkar, “What is MongoDB?”. [Online]. Available: https://www.educba.com/what-is-mongodb/. [Accessed May 27, 2022].