Western Norway University of Applied Sciences

BACHELOR'S THESIS

Notification Centre Development

for Wide Assessment (WA)'s Recruiting Management System

Bachelor, Computer Engineering

Department of Computing, Mathematics and Physics

Faculty of Engineering and Science

Submission date: 03.06.2019

Number of words: 7 639

Number of pages: 33

Group nr: 08

Jianyou Dai

Marcus Joar Hauge

I confirm that the work is self-prepared and that references/source references to all sources used in the work are provided, cf. Regulation relating to academic studies and examinations at the Western Norway University of Applied Sciences (HVL), § 10.

TITLE FOR MAIN PROJECTS

Report title: Notification Centre Development for Wide Assessment (WA)'s Recruiting Management System	*Date:* *03.06.2019*
Author(s): Jianyou Dai, Marcus Joar Hauge	*Number of pages: 33*
	Number of appendix pages: 4
Field of study: Computer engineer	*Number of DVDs/CD-er:* 0
Contact person at field of study: Richard Kjepso	*Grading: (Select: None or Limited to ddmmyy)*
Remarks:	

Assigner: Wide Assessment AS	*Assigner reference:*
Oppdragsgivers kontaktperson: Andreas Hammerbeck	*Phone:* 97661466
Summary: Development of notification centre for WA' recruiting system using React + TypeScript as frontend and C#(.Net Core) as backend.	

Keywords:

Notification	React	.Net Core
Poke	Typescript	Web API

Høgskulen på Vestlandet, Fakultet for ingeniør- og natuvitskap
Post address: Postbox 7030, 5020 BERGENBesøksadresse: Inndalsveien 28, Bergen
Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

Table of Contents

PREFACE

This document of bachelor assignment “Notification Centre Development” is completed in spring of 2019 by Marcus Hauge and Jianyou Dai at Western Norway University of Applied Sciences (HVL).

We would like to thank the people over at Wide Assessment(WA) for giving us this challenging and exciting bachelor’s assignment. Special thanks to Arve Andreassen and Stine Andreassen for bringing this project to our attention and allowing us to work at WA’s office. A great thanks to Andreas Hammerbeck and Vilja Rolfsen for all their continued support throughout the project.

Last but not least, we would like to thank Richard Kjepso, our project advisor at HVL, for all the guidance and encouraging feedbacks.

1. INTRODUCTION

Wide Assessment is a company developing an IT recruiting system that is used for candidates and companies to find their match. On the platform employers can post available positions and information about their company, and search for certain skill sets they want in a potential candidate. The candidates can edit their profile with prior experience, skills, education , they can upload their resume and link to other social media sites. A company or candidate can make contact by "poking" the other. This will for instance let the candidate know the company is interested in them.

1.1 Goal and motivation

The goal for this bachelor thesis is to create a notification centre and add this feature to WA's existing platform. By using this feature on the platform both candidates and companies are able to view their own notifications all at one place and make it easy for them take an action on it. In addition, the notification will be marked as "read" if it has been clicked by user or it will remain at an "unread" status.

The motivation is that Wide Assessment is constantly improving their platform for all users, and the new feature is expected to improve user experiences and boost the interaction between candidates and companies. The current solution is an email-based notification, that is, when a user pokes another, a notification will be sent and notify the recipient by email with a link in it. Then the user has to click the link for further info. Although the notification is not always seen by the recipient due to many reasons, for example when the recipient's mailbox is full, notification email could be landing in the spam folder or even the user forgets there's a notification in mailbox if he or she does not check often. By using the new feature notification centre that will bring benefits not only candidates and companies but also Wide Assessment itself marketing.

1.2 Context

As mentioned above, Wide Assessment wants to have a notification centre to be integrated to their existing recruitment platform. Currently the platform is an email-based notification when a candidate or company pokes to another, a message will be sent and notify recipient by email. However, the inconvenient are obvious that it has been described from above which is not user friendly enough for all WA users.

In this bachelor assignment, Marcus Hauge and Jianyou Dai will be working as a group and help WA to develop and integrate the notification centre to the platform. The new feature therefore will meet their requirements and bring a positive change for their already existing platform.

1.3 Limitations

There are some limitations for the notification centre development that it needs to be taken into consideration.

C# and .Net Core - The programming language C# and frameworks .Net Core are used in the existing platform in the backend. For our group, java, JavaScript and MVC are the most familiar programming language and framework. None of us have experience in C# or .Net Core. However, C# was built specifically as a Java competitor, there are many similarities between the two languages. Even the .NET framework has some similarities with Java. Object-oriented languages: Both Java and .NET (C# and VB.NET) are object-oriented languages.
Therefore, the limitations are existed, but it can be an opportunity for us to make a breakthrough point to the limitations.

React and typescript - when it comes to frontend, the platform uses typescript and React as their language and framework that are also new for us. Although TypeScript is a superset of JavaScript, in typescript there are many JavaScript code in it. With help of typescript documentation, it is not difficult for us to overcome the limitation. When it comes to React framework, it is one of the most popular frameworks which are being used by numbers of web developers and many companies prefer to use it as their frontend framework according to the research on GitHub. That means there are many already existing resources that can be used, in addition with help of online documentation of React and typescript, the limitation will not be a big hurdle for us to complete the project. It is what the purpose of doing bachelor project. To learn new things and update our programming language and frameworks which will be a practical benefit for our future work.

Integration - the notification centre system is considered as a part of the existing platform, it means the first step for us is to clearly understand the already implemented frameworks, models and the relevant codes that chosen and written by others, there will be a little room for us to decide freely. However, one of the most important way to break through this limitation is starting early and studying other's code by practising with our own code to test.

1.4 Resources

First and foremost, our contact person Andreas Hammerbeck who is a full stack developer and the CTO of Wide Assessment. He is available to provide us with tips and help along the development process, especially about the architecture of the system. Besides, Vilja Rolfsen, a developer at WA who is always available to help us whenever he is at office or on Slack. Therefore, both of them will be our most valuable resource to give a hand for our project development.

Another important resource is the internet. Our group will first go through some general beginner tutorials on React, and then look up the documentation for more specific concepts. The feature we will work on implementing is not something new, so similar solutions may be looked at for inspiration.

In addition, the school library has many valuable books related to programming languages and frameworks. It is uncertain to what extent that will be used, but it is at least a good alternative option.

1.5 Organization of the report

Chapter 1: An introduction to the project.
Chapter 2: A more detailed description of the project.
Chapter 3: The design of the project, discussing different approaches and planning.
Chapter 4: A detailed description of the actual solution developed in this project.
Chapter 5: How the work was evaluated, and the results of the evaluation.
Chapter 6: Discussion regarding the result based on the goals of the project.
Chapter 7: The conclusion of the report.
Chapter 8: Literature and appendices.
Chapter 9: appendices.

2. PROJECT DESCRIPTION

2.1 Project description

Our aim in this chapter is to describe the background for the project. In the sections below are project owner, previous work, initial requirements specifications, initial solution idea and literature background.

2.1.1 Project owner

The owner of the project is Wide Assessment AS which is a subsidiary company of Gyril Norway who provides employment recruiting services focusing on IT industry and helps especially IT firms to find talent and to locate individuals who meet specific job requirements. With the widespread impact of digitalization, the company started to digitalize the process. Eventually the digitized version is born in Bergen which is today's WA.works.

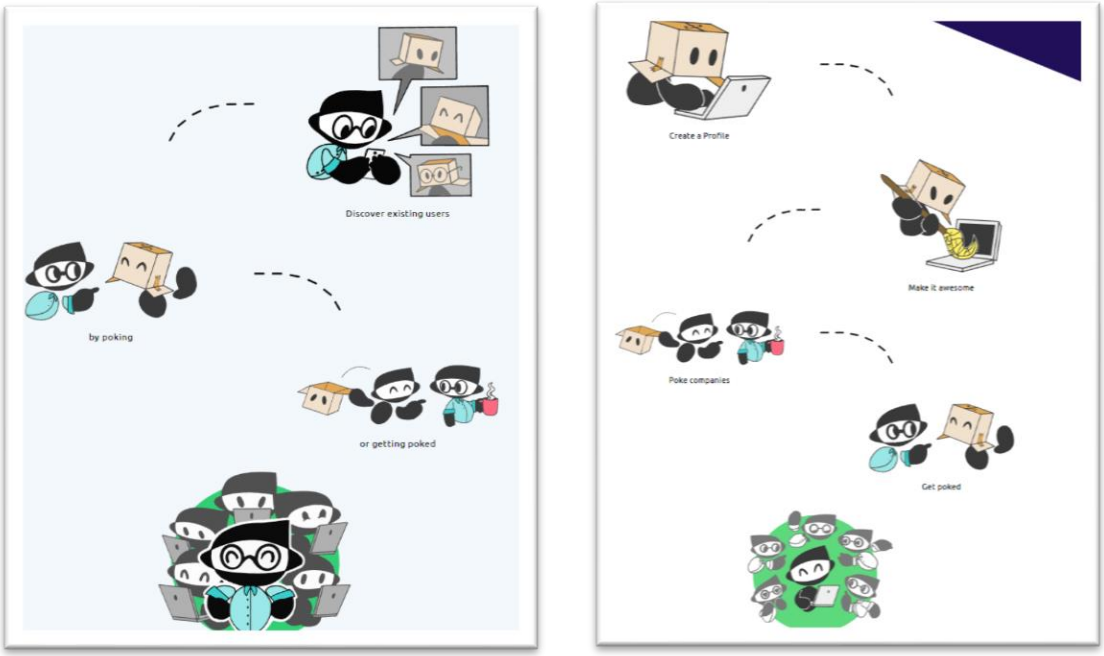


Figure 1: Wide Assessment

The platform provides a profiling for tech candidates and employers in a way that they can easily be approached and shared on the IT market. The candidates are able to show off their skills without being limited by formal experience or lack of communication skill. The platform allows them to create their profiles that can be viewed by multiple companies, and this makes it simple for companies to find their wanted candidates quickly and accurately.

The project team consists of one UX-designer and two full stack developers. Their office is located in the centre of Bergen. It is allowed for us to work at the office any time we want, and they are available to help us with both backend and frontend issues of the application.

Why the project is important to the project owner.
The current notification system is using email-based notification. Due to the email could be delayed, not received or it may be staying in user's spam folder. In this case either the candidate or company could consider that no response means the receiver has no interest in them. If delayed, it could cause loss for both sides because they could already have achieved something else before the late response.

The trend today is shifting to instant notifications and the new technology clearly has more benefits than these older mediums. The quick, real and effective request or response will be the final result. This will definitely bring a great benefit for candidates, companies as well as project owner itself.

2.1.2 Previous work

The development of our project is based on the existing product on wa.works platform. The current product integrated with an email notification system. Each time the poke button is pressed either from a candidate or a company, the system will send out a notification with an email form including an accessing link address for the receiver. The receiver will then be notified by the email. Clicking the provided link address enables the receiver to access and view more details of the sender's profile if he or she is interested.

2.1.3 Initial requirements specification

The initial requirements specification for the project is to understand how the email-form notification works between client and server. And then to further develop the new notification system based on the current frameworks. The extra challenge of the assignment is to make a real time notification work, that is, once the poke button is pressed, the notification should be visible on receiver's screen without reloading its webpage.

2.1.4 Initial solution idea

The initial solution idea is to create a notification centre system based on the existing platform. There are mainly two situations to handle the default settings of notifications once the poke-button is triggered.

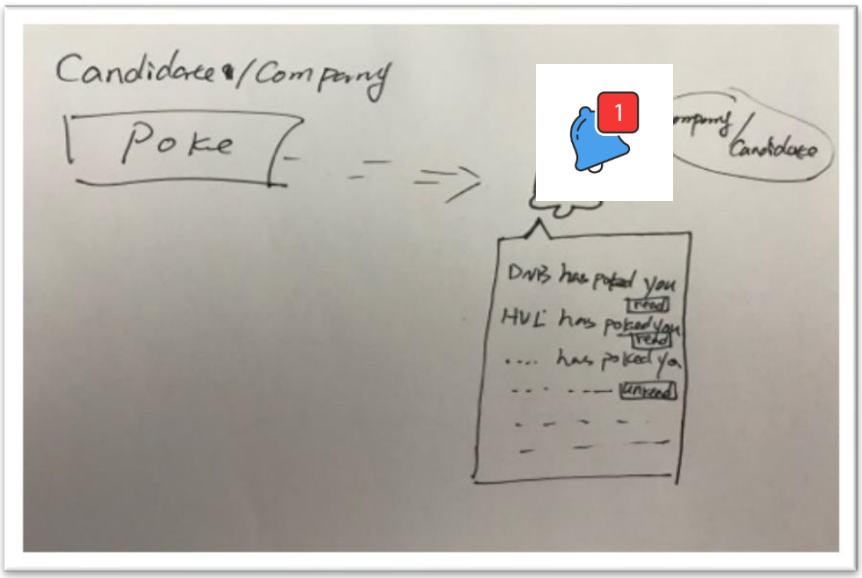


Figure 2: Notification sketching

The first situation is when the user is logged in, the notification should be set to display in a corner of the screen so that the user can have a quick view of it. Multiple notifications are

available to be shown as a stack or a list and then disappear after a set amount of time. User can click the notification-block for further view if he or she has logged on the platform.

The second situation is when the user is logged in. The system should enable end users to get notifications with an alert appearing in the status bar on the top of the webpage once the poke-button is triggered. On the alert status it should show the user the total numbers of unread notifications. Under the dropdown list of the status-icon it should be displaying all the latest read and unread notifications.

Although the real time function for the notification is not included in the assignment, we would like to take this as our challenge. With real time notification, the system should have a WebSocket connection or SignalR that enables Two-way full duplex communication with low latency between candidates and companies.

When it comes to the programming, it is desired that the code should be written in the language of React JavaScript and typescript for the frontend design. C# and .Net Core will be used in the backend. There is also a requirement to follow the company's programming practice in order for the code to be consistent and easy to read.

2.2 Literature background

References used in this document are mainly from the official website of .NET Core by Microsoft, Reactjs organization by Facebook, typescript organization by Microsoft and some published books and previous research papers.

Major resources referred to throughout this paper are dotnet core and Reactjs which are respectively the leading frameworks of backend and frontend that adapted in this project development. In the dotnet core webpage, it presents an overview of the structure to get an understanding of how to build web API, how to integrate with client-side framework React and so on. In the Reactjs documentation that provides the main concepts and advanced guides for building the frontend web in great details.

3. PROJECT DESIGN

3.1 Possible approaches

The possible approach is to create a new feature that either company or candidate can be able to poke and receive notification from each other. However, it is important to follow WA's architecture and programming practices in order for the implementation to function optimally. The aspect which has some wiggle room for design choices is the frontend. The backend logic will depend on how the frontend and the notification object is designed.

3.1.1 Alternative approach 1

On the client side there would be a notification icon button at the top menu bar. By clicking the icon button, all this user's read, or unread notifications will be displayed. User can take further action to view detailed information of the one who poked him or her. The notifications can be rendered in real time or fetched from the server on a set interval such that the user can see the notification without reloading the website.

3.1.2 Alternative approach 2

The frontend could have a page dedicated for notifications with details of each one. The page is basically a nested list from the icon-dropdown list which shows not only the latest notifications but also older notifications that user has received.

3.1.3 Discussion of alternative approaches

There is not that much of a difference between the approaches. A page only for notifications might be a bit excessive, but it depends on what kind of data the notification will hold. The frontend is WA's decision. We might come up with better looking designs along the way which they might approve of, but now in the beginning we could go with either approach and then afterwards evaluate how it turned out and discuss possible changes which would make an improvement.

3.2 Specification

We have chosen to start with approach 1 because the notifications are not very big or detailed, so an entire page only for notification might not be needed. Instead, an icon with a dropdown similar to sites like Facebook for example which will highlight unread notifications will probably give the best user experience.

3.3 Selection of tools and programming languages (if necessary)

As specified by WA, the frontend will be programmed in JavaScript plus typescript with the React framework. The backend will be programmed in C# with the ASP.NET core framework. Visual studio code is the preferred code editing tool, but any kind of text editor is fine. GitHub is used of easy version control and keeping track of changes.

3.4 Project development method

In order to achieve the goals and planned result, it is important to have a good and suitable methodology that puts a primary focus on the most crucial tasks that need to be done.

3.4.1 Development method

There are a number of software development methodologies such as Agile, Waterfall, Spiral development and so on. However, our team is not large, it consists of only 2 people plus about 3 developer from WA, a lightweight methodology is much suitable for us, like Agile Scrum, which is also recommended by our project owner. It focuses mainly on short iterative cycles and rely on the knowledge within a team.

Scrum is the most popular Agile development framework because it is relatively simple to implement but also because it solves a lot of problems that software developers have struggled with in the past such as convoluted development cycles, inflexible project plans, delayed production. The Scrum framework enables our small team to work in a short cycle of one or two weeks which called “sprints”. In addition, we can have a daily meeting if we want to discuss what has been done and what the current problems are. The methodology allows for quick development and testing.

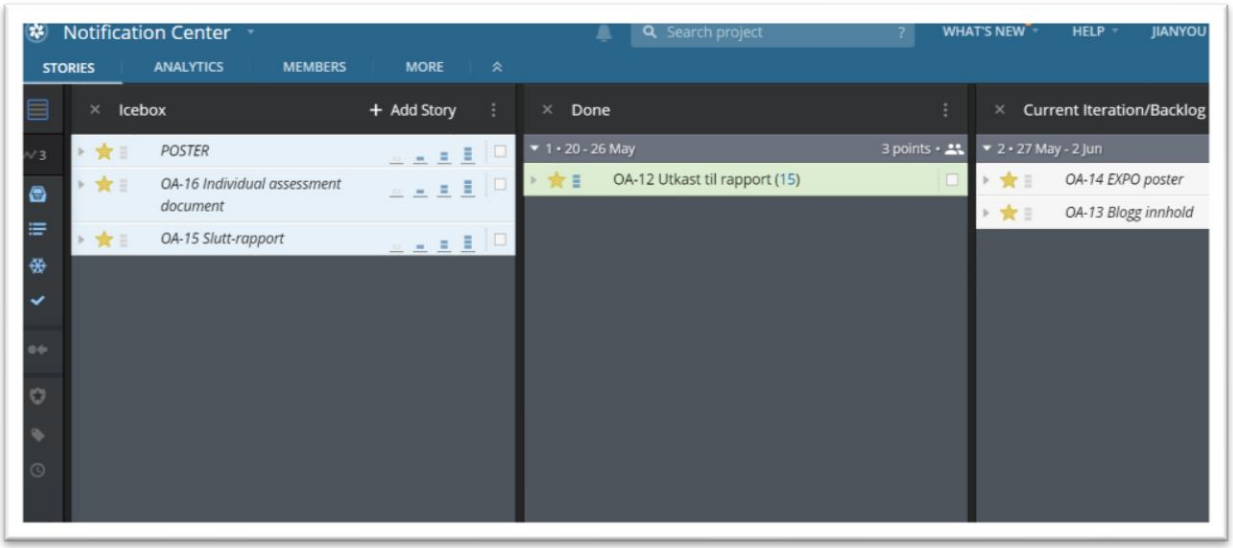


Figure 3: Pivotal Tracker Kanban-board

In additional, a Pivotal Tracker tool is recommended by our project owner. The Pivotal Tracker for Kanban-board embraces the simplicity and make the scrum run easier rather than creating more work. The main goal is to track the process of your tasks. It tracks what you are actually doing, how far along you are into a project and what's left to do.

By utilizing these methodologies ensure that every member of the project group is up to date on what is started, what is in the process , what is getting stuck in which phase, and what is finished. Combining the Scrum methodology together with The Pivotal Tracker board provide an actual feeling of everything running as planned.

3.4.2 Project Plan

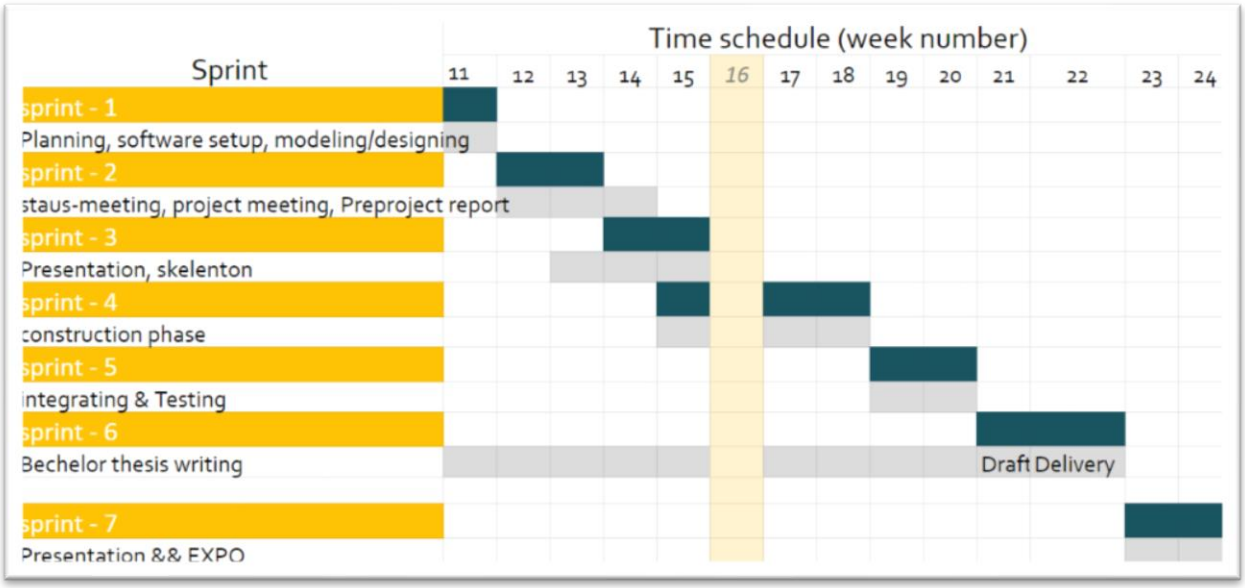


Figure 4: Gantt diagram

Gantt diagram explained:

Sprint 1
The goal at this part is basically to make a general preparation for the project development. I consist of highlighting the main tasks such as making a schedule time, selecting the working method, setting up software requirement, understanding the basic requirements of the project. Besides M2 assignment needs to be delivered then.

Sprint 2
At this part we are going to break down each task into small pieces then fill it in our agenda. As well the group can start modelling for the application to have a better understanding of the app structure through a data flow diagram. In addition, the group should decide our project name in the report, meet our supervisor and deliver the status report afterwards. In week 13

our focus is on our pre- project report. It should be completed and have a review by our supervisor before delivery.

Sprint 3
Our main job at this part is to start building the application structure and making a simple notification demo work at the front end at first with fake data.
Secondly the group have to prepare for our presentation for week 15th.
At last the group can try to connect frontend with backend. In the meantime, we need some feedbacks from our UX design about the frontend.

Sprint 4
This part will be the complete core function development for the skeleton which includes adding database connection to all elements. The group may want to work sometime during the Easter holiday.

Sprint 5
Here we will focus on testing and debugging. It should not be a big problem, because we will keep testing along the development process from the beginning. If all is good, it is time to integrate our project to the platform and make it work. After that if we have good time, we can try to take the challenge that is implementing WebSocket to make the notification in real time.

Sprint 6
We will keep writing and editing our thesis through the whole process, such that we can have a quick feedback responding from our supervisor on each sprint and deliver the draft and the final report on time. After delivering we can start the poster for the expo. We can ask our project owner for tips.

Sprint 7
Writing reflection notes for individual. Then preparing presentation and expo supposed to be at last sprint of the schedule.

3.4.3 Risk management

There are various types of risks through every step of the development process. It may cause part of our project failure, delay or prevent us from achieving our goals. In order to avoid and minimize those threats, we need the risk management to handle it properly. Therefore, we should monitor this from early stages until the project is delivered.

In the risk management, all possibilities of the threats should be listed and identified in the beginning of the schedule.

Time management
The time schedule is set up in the beginning of the development, but we cannot be certain that the scheduled time will accurately meet to our actual tasks' requirement. To avoid failures which may cause delay or other results, we should have good communication between group members, and keep updating the time schedule accordingly.

Not achieving the goals as expected.
It is possible that we may get stuck in the code and using too much time on debugging. To avoid this, we should contact our project owner as early as possible after we have tried our best efforts. In addition, we have to prioritize our tasks for not affecting other processes.

Lack of competence
Neither of us have any experience in using c#, .net core and react, although it is not completely new programming languages or frameworks so it shouldn't take too long to get familiar with them. In the developing process it may cause some extra time on debugging. To minimize too much time consuming at this part, we should share the learning together and dare to ask questions and solve problem as quick as possible.

Illness or absence
Illness and absence are possible to happen and definitely will affect the project quality and working process. Therefore, it is important to maintaining the work-life balance. the absence is better in control as long as we can reschedule and prioritize the task then it won't affect the process too much.

3.5 Evaluation method

The evaluation method of the project is to maintain a good communication with our project owner and keep our them up to date in order to make sure that every task is done based on the initial goal as requested initially.

The second important is self-verification after each small goal achieved and fix the problem as quick as possible with project owner if necessary. In the end to have complete review the whole project together with project owner ensure the final goal is achieved.

4. DETAILED DESIGN

This chapter contains details about the use cases, architecture, the designs of the backend, frontend and database, and implementation.

4.1 Use cases

The use cases are quite few and simple.

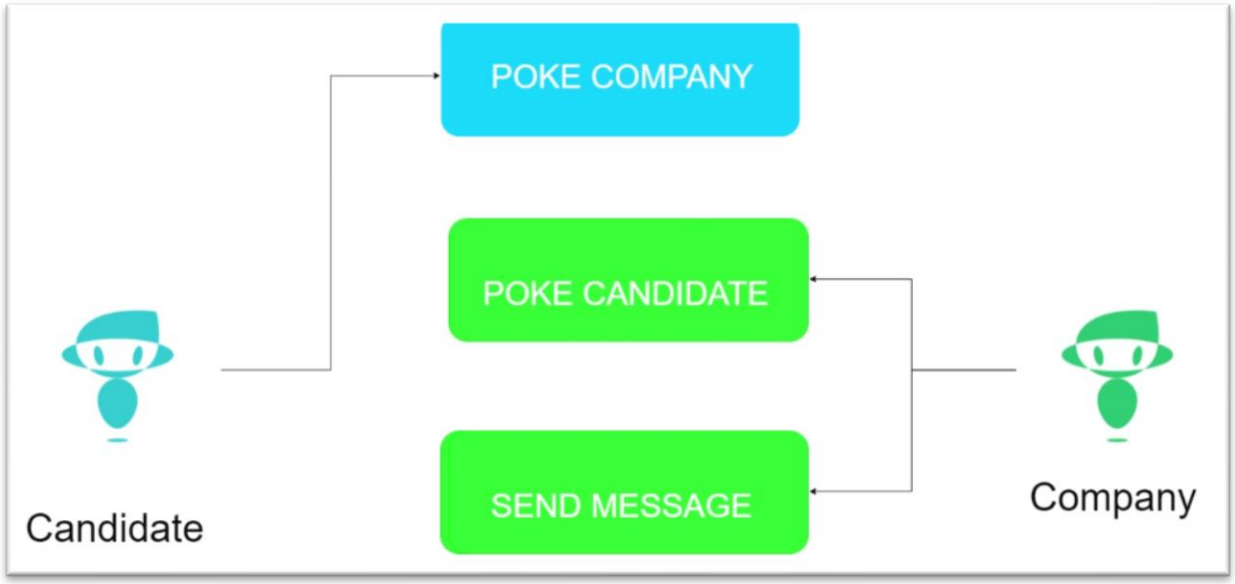


Figure 5: use case

Flow:
A candidate will navigate to the “companies” page where he/she will filter them based on preference, and then click poke on some company. The poke will send a an initial notification object to the backend where it will be further processed. A notification will be generated for each admin in the poked company.

A company poking a candidate works similar. The company first creates a search with wanted education, experience etc, and then click poke on the relevant candidate(s). A “base” notification will be sent to the backend, processed, and stored in the database.

When a company wants to send a message, they first click create message which opens up a modal box, then fills in the form data and clicks send. The form data is sent to the backend which will generate a notification for each specified recipient.

4.2 Architecture:

The architecture follows the MVC pattern where the view is controlled only by the client, which sends requests to the controller to fetch the relevant data. In contrast to the more traditional design where the view in prepared in the backend based on the request and then served to the user, fetching only some data is faster because it won't send an entire page each time.

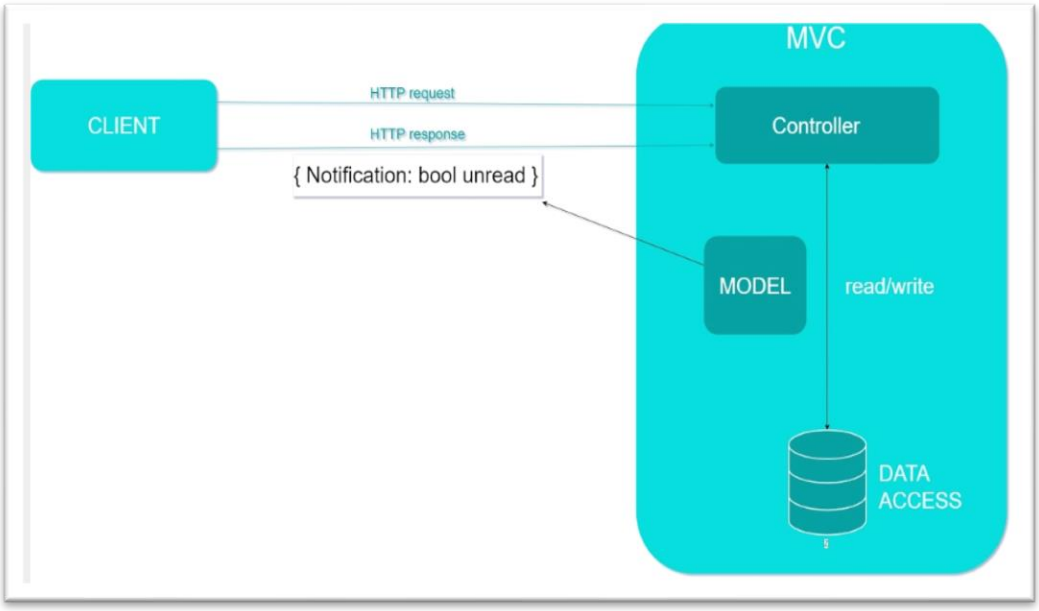


Figure 6: Overview of the total architecture

4.2.1 Database model

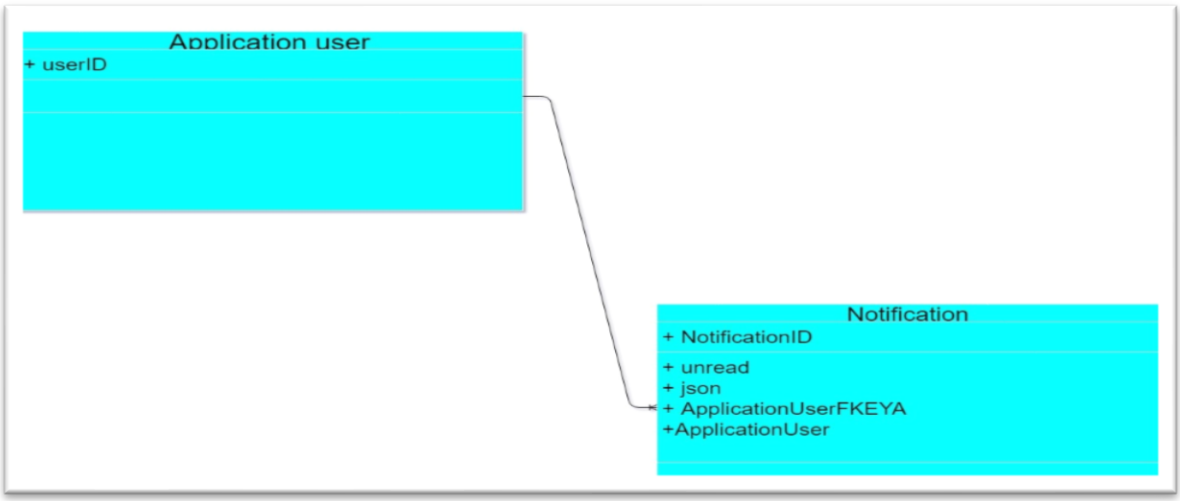


Figure 7: Model shows what data the Notification object contains.

4.2.2 Server architecture

As mentioned, the server architecture utilizes the MVC pattern, which stands for model-view-controller. This way of development separates parts of the application making it easy to maintain, scale and reuse any of them. The backend consists of the model and controller parts. The model is the data structure of the application, and the controller takes input and processes it and performs some actions on the model.

4.2.3 Client architecture

The client side is created with the React framework and is a single-page website. This means that the client will not request a new page when navigating the site, but rather load the website once on first visit, and then content will be created and removed dynamically based on the user inputs. A big advantage with this approach is a much quick and responsive experience for the user.

React is component based, so a notification component is created with its own state which holds the current notifications the user has received. The component, as with all other react components, has a render() function which is the part where the HTML elements on the site is modified. This function will render the notifications in the dropdown menu/icon by creating a HTML element for each notification object currently stored in the component's state. The render() function is called whenever the state changes, and for the state to change, a GET request to the notification controller on the server is made. The returning response is a list of notification objects which are put into the component's state.

4.3 Implementation

4.3.1 Database/model

Entity Framework Core is the framework used for data access. A subclass of DbContext handles data access. The class contains properties of the type DbSet<SomeDataObject>, which represents collections of each data object in the system. Since this is already done and configured by WA, all that is needed to be added is a Dbset for the notification object to access notification data in the database.

4.3.1 Server application

A controller is implemented to handle all http requests related to notifications. The controller is designed in a way that an action to be executed is routed to with http[verb] attributes. This is a typical way of routing for web APIs.

There are three actions which can be performed, which means three different methods are defined in the controller. These actions are retrieving notification data, creating notifications, and changing the data in a notification.

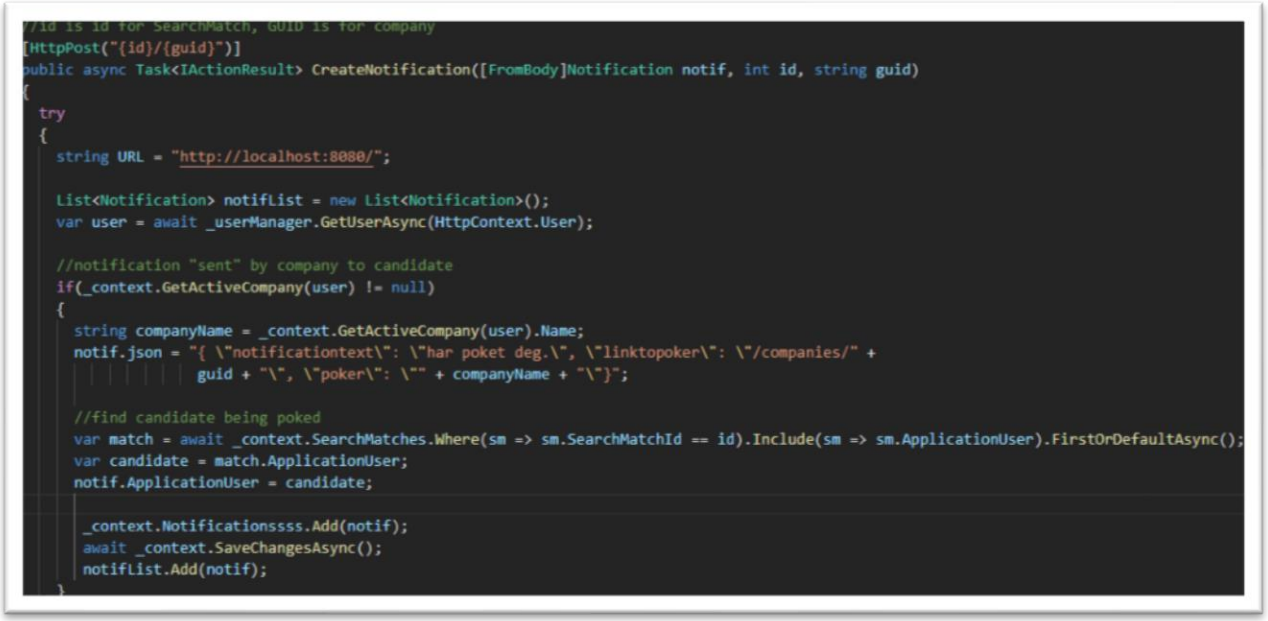
```
[Authorize]
[HttpGet]
0 references
public async Task<IActionResult> GetNotifications()
{
  try
  {
    var user = await _userManager.GetUserAsync(HttpContext.User);
    var notificationList = _context.Notificationssss.Where(n => n.ApplicationUser == user);

    var result = new[] { notificationList };
    return SimpleHTTP.OK(result);
  }
  catch (Exception e)
  {
    Console.WriteLine(e);

    return SimpleHTTP.BadRequest(new { });
  }
}
```

Figure 8: GET-request method in the Notification Controller.

The [Authorize] attribute on the top means that if the user making the request does not pass an authorization check, the action will not be performed. The second attribute [HttpGet] routes incoming get-requests to this method. The logic inside the method is surrounded by a try-catch to handle errors. The user making the request is fetched from _userManager, which is then put in a query to the database for all the notification objects relevant to this user. It is a query expression from LINQ, which is a component from .Net framework that makes for easy data extraction from several different data sources, including relational databases. The query returns a list of the objects which is returned to the client to handle.



```
//id is id for SearchMatch, GUID is for company
[HttpPost("{id}/{guid}")]
public async Task<IActionResult> CreateNotification([FromBody]Notification notif, int id, string guid)
{
  try
  {
    string URL = "http://localhost:8080/";

    List<Notification> notifList = new List<Notification>();
    var user = await _userManager.GetUserAsync(HttpContext.User);

    //notification "sent" by company to candidate
    if(_context.GetActiveCompany(user) != null)
    {
      string companyName = _context.GetActiveCompany(user).Name;
      notif.json = "{ \"notificationtext\": \"har poket deg.\", \"linktopoker\": \"/companies/" +
                   guid + "\", \"poker\": \"" + companyName + "\"}";

      //find candidate being poked
      var match = await _context.SearchMatches.Where(sm => sm.SearchMatchId == id).Include(sm => sm.ApplicationUser).FirstOrDefaultAsync();
      var candidate = match.ApplicationUser;
      notif.ApplicationUser = candidate;

      _context.Notificationssss.Add(notif);
      await _context.SaveChangesAsync();
      notifList.Add(notif);
    }
```

Figure 9: Snippet of part of the post method

This is the first section of the post request method. The method takes three parameters, a notification object, a number, and a string. The notification object only contains some data of the notification, the Boolean unread status. It is done this way for simplicity reasons. The snippet shown on the figure is for the case where a company pokes candidate user. A JSON string is made with the guide value from the parameter. GUID is an acronym for Globally Unique Identifier, a value each company in the database has. These values in the json will utilized in the client. The id parameter is for a search match retrieved from which candidate is being poked in the client. A query is made to the database to find the correct candidate, which will be put into the ApplicationUser attribute of the notification. Finally, the notification is added to the notification DbSet of the DbContext, and then SaveChangeAsync() persists the change to the database.

In the case where it is the candidate who pokes a company, a notification is made for each admin in the company. The implementation is quite similar to the one described above but with some differences. The poked company is fetched with the GUID parameter, and a query is made which selects all users which is admin the company. For each admin user in the company, a notification object is made for them and persisted to the database.

The last case is when sending a “custom” notification message. This will only be used by WA admins to send out notifications to either all company admins or all candidates. If GUID equals “massNotification” in the post request, then based on the id, a query is made to fetch all candidates or all company admins and then create a notification for each one.

The final action, changing a notification, is done with a PUT request. The actual change to the notification object is done in the client. The clients send the updated object in the put request which is then passed to the dbContext which updates the database.

4.3.2 Client

As described in the architecture section above, the client is built with the React framework and is a so called Single-page application. Since React is component-based, a separate component for notifications was implemented. Components are kind of similar to regular functions, in that they take some input and returns some elements to be rendered on the page. There are two different types, functional and class components. The functional one really is just defined like a regular function, while a class component is defined as a class extending the React.Component class with its own state and functions. Components takes something called “props”(properties) as input, which works basically how regular functions take some parameter as input. This is an easy way to reuse a component several places on the website but where it acts different each place based on the props it was passed.

The notification component is of the class type as it needs its own logic state. State has similarities to props, but instead of it being passed to the component and is only read-only, state is variable and handled from inside the component.

Something related to state is the render() function which is the only function components must implement. Every time the state is changed, render() will automatically be executed. This function returns the elements to be displayed on the page, with the help of JSX. JSX stands for JavaScript Extension and makes it simpler to make changes to the page by writing the code like HTML instead of having to use DOM which could require significantly more code. It makes for better readability and easier to debug.

```
public render() {
  var items = this.prepareNotifications();

  return (
    <div>
      <div className="dropdown">
        <a className="dropbtn" onClick={this.btnClick}>
          <i className="fa fa-bell"> </i>
          <mark>23</mark> Notification </a>

        <div id="myDropdown" className="dropdown-content">
          {items}
        </div>

      </div>
    </div>
  );
}
```

Figure 10: Render function

Figure 10 is a snippet of the render function in the notification component. The prepareNotifications() function on top goes through each notification object in the state, and fetches the relevant data to be displayed, such as text and URL to whoever poked the user, and then put the data into a HTML <p> element which is then put into an array. In render(), the items is inside curly brackets, which is a way to write any JavaScript expression inside JSX. Since items is an array of <p> elements, all these elements will be rendered inside the .

Lifecycle
http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

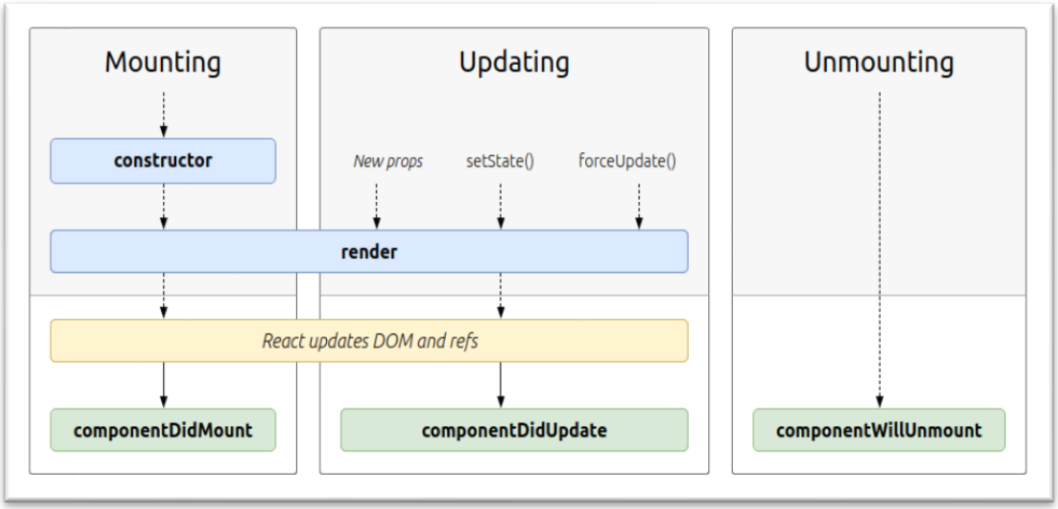


Figure 11: React component lifecycle functions.

The constructor is for initializing state and binding methods, which is not necessary in the notification component. After the component is mounted, which means added to the DOM, componentDidMount is called. Here, an interval is created which will automatically calls the function that fetches data from the server every 30 second.

```
39  public getNotifs = () => {
40    axios.get("http://localhost:8080/api/notifications")
41      .then((response) => {
42        let notifications = response.data[0];
43        this.setState({ objects: notifications });
44      });
45  }
```

Figure 12: Function of notifications request from the web API.

Using the Axios library, a get request is made to the API and the “objects” field in state is set to the notification objects returned for the server.

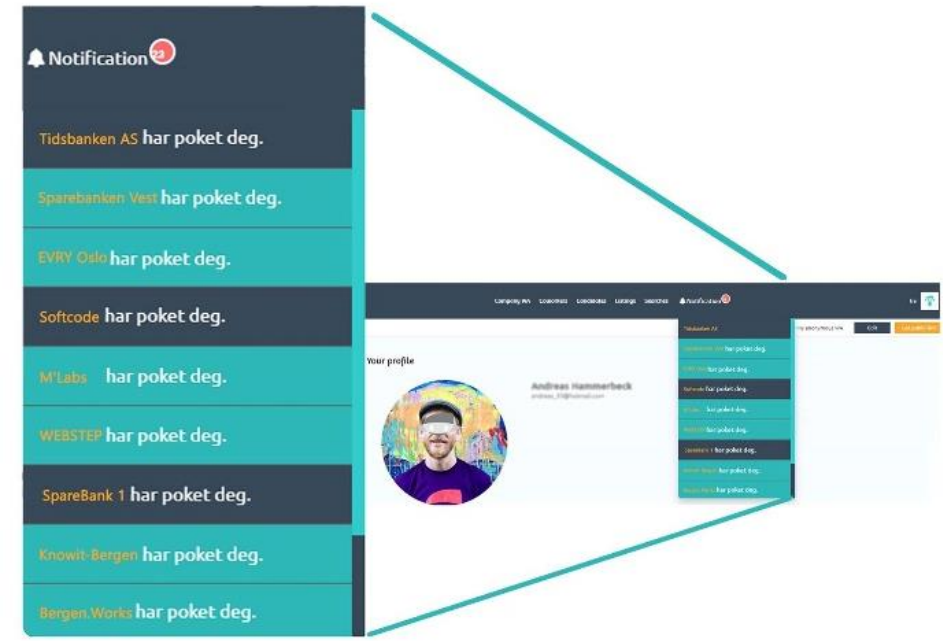


Figure 13: how the notifications is rendered after the get request.

Last function in the notification component is markAsRead(), which changes the read status of the notification.

```
public markAsRead = (id) => {
  var notification = this.state.objects.find(element => element.notificationId === id);
  notification.unread = false;

  axios.put("http://localhost:8080/api/notifications/" + id, notification)
    .then((response) => {

    });

  this.forceUpdate();

}
```

Figure 14: Method markAsRead() function in notification component

The function retrieves the notification to be changes based on the id parameter, sets unread to false, and makes a put request to the web API with the entire object. ForceUpdate() is called in order to re-render the component so that the notification renders with a different look.

Creation of notifications is not a part of the notification component, but rather another component for the buttons for poking someone. There are two buttons, one for poking a company, and one for poking a candidate. Both of them makes a post request to the WEB api, with either ID for candidate or GUID for company. An option to create a “custom” notification is available to WA’s admins. This is a separate component which opens a modal box when a button is clicked, with a text field for the message and a dropdown for selecting the receivers.

5. EVALUATIONS

The goal in this section is to describe the methods are used for evaluating and determining whether the project meets the goal as it is initially expected. The evaluation methods mainly based on the development method scrum and functionality testing which will be presented in the sections below.

5.1 Evaluation method

Evaluation by Scrum

As mentioned earlier in section 3.4, the development methodology Agile Scrum is implemented for the project development. The Scrum framework enables project team to work in a short cycle of sprint. Each sprint will get feedbacks by comparing the the solution with the project requirements. Then the feedbacks will be used to evaluate whether the it is heading in the right direction. By following scrum framework, the team can give a quick response to the problem addressed by project owner through a meeting or daily communication. The essential adjustment will be made accordingly so that the project always meets the requirements for usability.

Evaluation by project owner

Another method of evaluation is maintaining a good communication with our project owner and keep them up to date in order to make sure that every task is done based on the initial goal as requested. When it comes to front-end development, it is about user events and rendering the right views at the right time. Due to React is used in the frontend, therefore it is easy to use the build-in TestUtils which is Jest that can be used to test the UI structure, interactions. By maintaining a good communication with UI designer from WA to evaluate the user interface design should always follow the existing theme and not changed unexpectedly from user end.

In addition, the functionality testing is important to ensure the quality of the code and the results of all functions should be working as planned. After a function was implemented or changed, it will be pushed to the GitHub, and then the project owner will evaluate the code and the result regularly through GitHub repository. Project owner can make a comment to notify that a specific part of the code should be modified, as well keep tracking of the progress and reducing the expectation gap between initial concept and delivered reality.

5.2 Evaluation results

By keeping track of the progress and evaluating the task during each sprint and after it is completed by the project owner. Meanwhile the group follows tight after the scrum framework and remain good communication with project owner to make the modification when it is needed. An actual testing of the solutions enhanced the robust code and reduce the chances of occurrence of risk.

6. DISCUSSION

6.1 Final product

One of the biggest influences on the results is that none of the group members had any prior experience in any of the technologies used in WA's system, except for for SQL, which was only used a little bit for testing purposes. This in combination with the fact that the codebase was completely new to us, lead to a lot of time being spent on figuring out how things worked and how to all work together. At the start of the project, we did read up on the basics of React and created a separate project to test and experiment how our solution would be like, but eventually moved over to WA's codebase. This was very overwhelming at first and we didn't know where to begin, but some guidance from WA's developers helped out alot. Not a lot of reading was done before we dived right into implementation and began our trial and error process. This method involved a lot of guessing and assumptions, but progress was made, nevertheless. In hindsight, this method was not optimal and led to us not following all the right coding and design practises. More studying up the principles behind .net core, react, redux etc would have prevented us in having to spend time correcting and changing things up in the later parts of the project.

6.3 Choices

Throughout this project there hasn't been that much room for different approaches and design choices as it would have been if the project was to develop a new system, in contrast to new features in an existing system. Nevertheless, there was several ways to go about this project.

One of the major choices that has been made in the project is how the Notification object would look like. Since the developers at WA talked about extending the notification feature or change in the future, we decided to use a JSON string as an attribute of Notification in addition to the other must-have attributes. This way it wouldn't be necessary to create a new model later, but instead add data to the JSON string. A disadvantage to this is you cannot directly make queries to the database based on data in the JSON string. Also, there is a slight performance decrease because of the need to parse the string, but not enough to be noticeable. There was also the choice of having the notifications on a separate page for itself, or just have them in the dropdown icon on the top menu. The decision landed on not adding the extra page because of the nature of the notifications. It would be a little excessive for such "simple" content and might impact the user experience negatively.

As stated before, since the codebase and technologies in this project was completely unfamiliar, there was a fair share of trial and error. This led to us sticking with a solution when it worked but should maybe have taken some more time to think if there is a better way

to do it. Of course, we knew the principles behind the MVC architecture and we follow best practises to the best of our abilities.

6.4 Notes

Even though there might have been more trial and error than learning from reading, we did learn very much about the technologies used. React in particular was very intuitive and enjoyable to work with, with separation of components making it very modular and flexible. We also value the experience of working on an existing codebase a lot, because this is something we most likely are going to do after graduating.

7. CONCLUSIONS

7.1 Summary of goals

The main goal of the project is to develop a notification system and integrate it to existed recruiting platform for wa.works. In such way that a company or candidate can make contact by “poking” the other and then a notification will be sent or received by a candidate immediately once the page is reloaded, instead of checking the notification by email. In additional a company can create a notification and send it to a candidate. The programming languages frameworks are used for the project are C#-.NET core for the backend and React-Typescript for the frontend.

7.2 Confirmation of reached goals

As the poke button is pressed, the notification will be sent to a candidate, then it will also appear in receiver's dropdown menu with unread status. By clicking the notification item, it will lead user to the sender’s profile page. As well user can view all read or unread notifications from a single page. The features above are integrated to the existed platform for wa.works such that one can conclude that the main goal for the project has been reached.

7.3 Further work

Further work for the project, the adaptation of WebSocket has been left for the future due to lack of time. Although it is not required in the project, we would like to try during the development. With the implementation of WebSocket protocol, a notification will be sent or received in real-time. User will be notified by the pushed notification whenever user is busy with other things, the real-time notification will be an effective way of getting user back to

the site and easily view the message. In such way that the interaction between candidate and company would be boost increasingly. In addition, the real-time notification enable user to have personalized custom notification through settings.

8. LITERATURE/REFERENCES

Microsoft.com (n.d.). .NET Core Guide. [online] Available at: https://docs.microsoft.com /en-us/dotnet/core/#net-core-22 [Accessed on 15 May 2019].

Upwork.com. (n.d.). Java vs. .NET: Determining the Right Software Platform for Your Project. [online] Available at: https://www.upwork.com/hiring/development/java-vs-net-determining-right-software-platform-project/ [Accessed on 02 April 2019].
Medium.com. (n.d.). React vs Angular vs Vue.js: A Complete Comparison Guide. [online] Available at: https://medium.com/front-end-weekly/react-vs-angular-vs-vue-js-a-complete -comparison-guide-d16faa185d61 [Accessed 02 April 2019].

Reactjs.org (n.d.). A JavaScript library for building user interfaces. [online] Available at: https://reactjs.org/docs/getting-started.html [Accessed on 10 April 2019].

Typescriptlang.org (n.d.). Typescript doc. [online] Available at:https://www.typescriptlang .org/docs/handbook/release-notes/typescript-3-3.html [Accessed on 07 April 2019].

Microsoft.com (n.d.).Tutorial: Create a web API with ASP.NET Core MVC. [online] Available at: https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view= aspnetcore-2.2&tabs=visual-studio [Accessed on 19 May 2019].

WA.works (n.d.). We help companies grow better [online] Available at: https://wa.works/candidate?s=candidate) [Accessed on 11 April 2019].

Code.visualstudio.com. (n.d.). Visual Studio Code - Code Editing. Redefined . [online] Available at: https://code.visualstudio.com/ [Accessed on 25 May 2019].

W3schools.com. (n.d.). JSON Introduction. [online] Available at: https://www.w3schools.com/js/js_json_intro.asp [Accessed on 02 April 2019].

Typescriptlang.org. (n.d.). Migrating from JavaScript [online] Available at: https:// www.typescriptlang.org/docs/handbook/migrating-from-javascript.html [Accessed on 03 April 2019].

Tomdalling.com. (2009). Model View Controller Explained. [online] Available at: https://www.tomdalling.com/blog/software-design/model-view-controller-explained/ [Accessed 13 May 2019].

Google.com (n.d.). Model View Controller Explained. [online] Available at: https://developers.google.com/web/ilt/pwa/introduction-to-push-notifications [Accessed on 15 Apil 2019].

Npmjs.com (n.d.). React-awesome-modal [online] Available at: https://www.npmjs.com/package/react-awesome-modal [Accessed on 05 May2019].

9. APPENDIX

9.1 APPENDIX A: Risk list

Risk identification and mitigration				
Risk situation	Likelihood (1-5)	impact (1-5)	Risk level (0-10)	mitigration
poor time management	2	5	10	scheduling and keep updating
poor communicaiton	2	3	6	Keeping update with project owner
computer problem (installation	3	4	12	backup solution
illness or absence	2	3	6	balance life-work, prioritizing task
implementation faliure	3	5	15	debugging, ask help from p-owner
Lack of competence	1	3	3	self-learing, ask question often
github collision	2	3	6	test a demo first or ask help from p-owner

Figure 15: Risk identification and mitigation

9.2 APPENDIX B: Gantt diagram

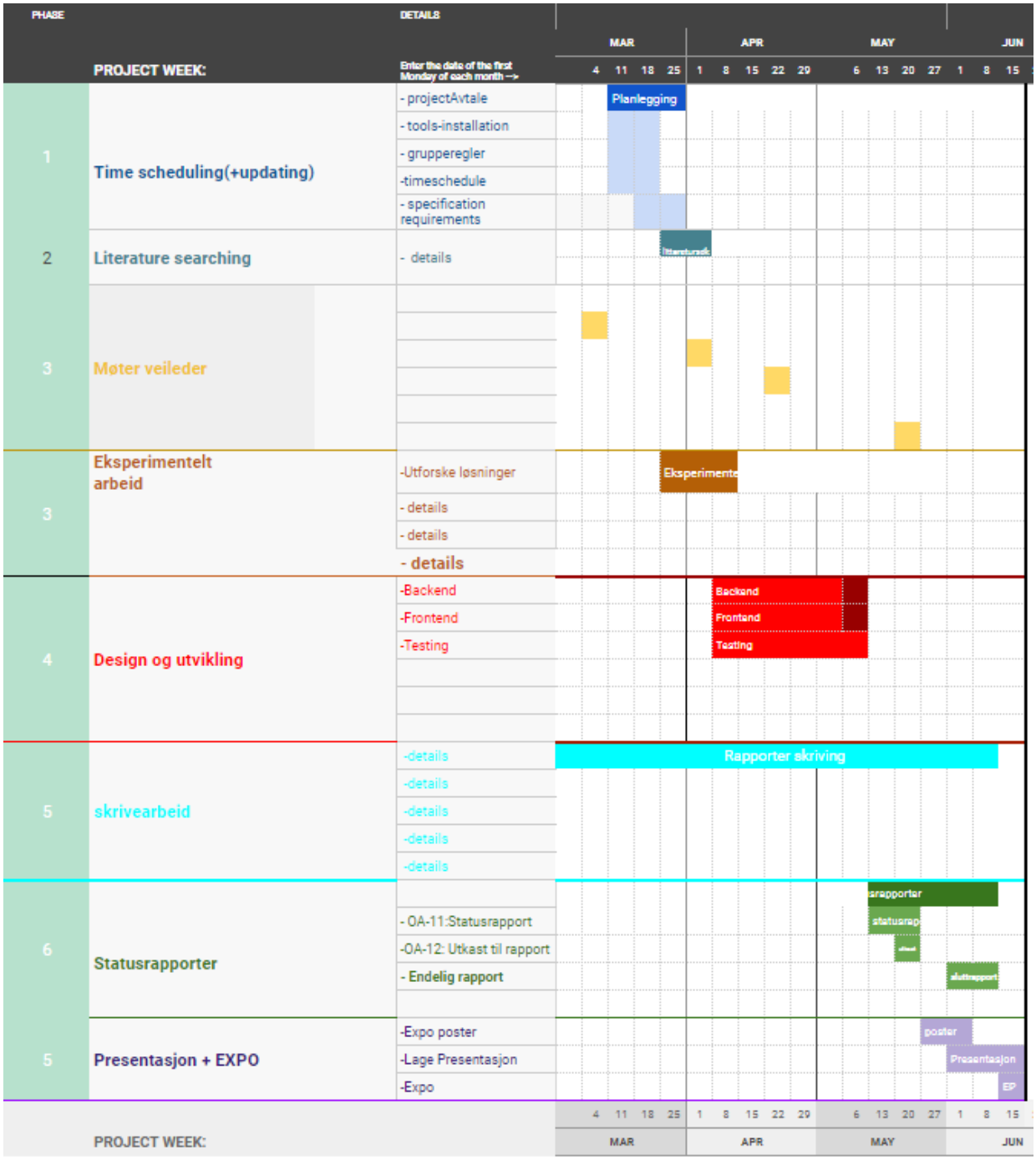


Figure 17: Gantt diagram

9.3 APPENDIX C: Acronyms

API	Application Programming Interface
HVL	Høgskulen på Vestlandet (Western Norway University of Applied Sciences)
JS	JavaScript
MVC	Model-View-Controller
WA	Wide Assessment
UI	User Interface
UX	User Experience
CTO	Chief technical officer
JSX	JavaScript XML
DOM	Document Object Model
http	Hyper Text Transfer Protocol
MVC	Model-View-Controller

9.4 APPENDIX D: Vocabulary

API	A set of clearly defined methods to communicate between a service and any other software or component
Backend	Commonly the database and web server layers of a web application. Where most of the business logic and data are stored.
Frontend	The presentation layer of a web application. What the user actually can see and interact with. Also called client.
User	Company and Candidate
Database	An organized collection of data, structured with tables holding different attributes.
React	A JavaScript framework for building user interfaces.
State	An object that is managed within a React component.
Component	Components are the building blocks of any React app and a typical React app will have many of these