



**Title: Internship in web development at ØB Innovation**

**Author(s): Viktor Nerland Engvall & Magnus Munthe-Dahl**

**Total page count: 74**

**Submission date: 30.05.2022**

## Obligatorisk egenerklæring/gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

Du/dere fyller ut erklæringen ved å klikke i ruten til høyre for den enkelte del 1-6:		
1	Jeg/vi erklærer herved at min/vår besvarelse er mitt/vårt eget arbeid, og at jeg/vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.	<input checked="" type="checkbox"/>
2	Jeg/vi erklærer videre at denne besvarelsen: <ul style="list-style-type: none"><li>• ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands.</li><li>• ikke refererer til andres arbeid uten at det er oppgitt.</li><li>• ikke refererer til eget tidligere arbeid uten at det er oppgitt.</li><li>• har alle referansene oppgitt i litteraturlisten.</li><li>• ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.</li></ul>	<input checked="" type="checkbox"/>
3	Jeg/vi er kjent med at brudd på ovennevnte er å <u>betrakte som fusk</u> og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. <a href="#">Universitets- og høgskoleloven</a> §§4-7 og 4-8 og <a href="#">Forskrift om eksamen</a> §§16 og 36.	<input checked="" type="checkbox"/>
4	Jeg/vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert, jf. <a href="#">høgskolens regler og konsekvenser for fusk og plagiat</a>	<input checked="" type="checkbox"/>
5	Jeg/vi er kjent med at høgskolen vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens <a href="#">retningslinjer for behandling av saker om fusk</a>	<input checked="" type="checkbox"/>
6	Jeg/vi har satt oss inn i regler og retningslinjer i bruk av <a href="#">kilder og referanser på biblioteket sine nettsider</a>	<input checked="" type="checkbox"/>

# Personvern

## Personopplysningsloven

Forskningsprosjekt som innebærer behandling av personopplysninger iht. Personopplysningsloven skal meldes til Norsk senter for forskningsdata, NSD, for vurdering.

Har oppgaven vært vurdert av NSD?

ja  nei

- Hvis ja:

Referansenummer:

- Hvis nei:

Jeg/vi erklærer at oppgaven ikke omfattes av Personopplysningsloven:

## Helseforskningsloven

Dersom prosjektet faller inn under Helseforskningsloven, skal det også søkes om forhåndsgodkjenning fra Regionale komiteer for medisinsk og helsefaglig forskningsetikk, REK, i din region.

Har oppgaven vært til behandling hos REK?

ja  nei

- Hvis ja:

Referansenummer:

# Publiseringsavtale

Studiepoeng: 15

Veileder: Ketil Danielsen

## Fullmakt til elektronisk publisering av oppgaven

Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven. §2).

Alle oppgaver som fyller kriteriene vil bli registrert og publisert i Brage HiM med forfatter(ne)s godkjennelse.

Oppgaver som er unntatt offentlighet eller båndlagt vil ikke bli publisert.

Jeg/vi gir herved Høgskolen i Molde en vederlagsfri rett til å

gjøre oppgaven tilgjengelig for elektronisk publisering:

ja  nei

Er oppgaven båndlagt (konfidensiell)?

ja  nei

(Båndleggingsavtale må fylles ut)

- Hvis ja:

Kan oppgaven publiseres når båndleggingsperioden er over?

ja  nei

Dato: 30.05.2022

# IBE600 Bacheloroppgave

Viktor Nerland Engvall, Magnus Munthe-Dahl

## Contents

<b>1</b>	<b>Abstract</b>	<b>1</b>
<b>2</b>	<b>Introductory information</b>	<b>1</b>
2.1	Getting the most out of this report . . . . .	1
2.2	Entry test . . . . .	1
2.2.1	The VBOOnline REST API (and REST APIs generally) . . . . .	1
2.2.2	API documentation using Swagger . . . . .	2
2.2.3	Authenticating using the bearer token . . . . .	2
2.2.4	Why we need to use asynchronous code to talk to remote systems . . . . .	2
2.2.5	Example async/await functions . . . . .	3
2.2.6	The test . . . . .	5
2.3	About the workplace . . . . .	7
2.3.1	ØB Innovation . . . . .	7
2.3.2	VBOOnline Web . . . . .	7
2.4	How we worked . . . . .	8
2.4.1	Location . . . . .	8
2.4.2	Developer tools . . . . .	8
2.4.3	Source control . . . . .	8
2.4.4	Project management . . . . .	9
2.4.5	Documentation . . . . .	10
<b>3</b>	<b>Viktor's Tasks</b>	<b>11</b>
3.1	Structure of this section . . . . .	11
3.2	Save PDF for results report (Issue #286) . . . . .	11
3.2.1	Introduction . . . . .	11
3.2.2	Research . . . . .	11
3.2.3	Implementation . . . . .	12
3.2.4	Testing . . . . .	13
3.2.5	Conclusion . . . . .	13
3.3	Save PDF reports for customer/supplier ledger (Issue #324/ #325 respectively) . . . . .	14
3.3.1	Introduction . . . . .	14
3.3.2	Research . . . . .	14
3.3.3	Implementation . . . . .	14
3.3.4	Testing . . . . .	19
3.3.5	Conclusion . . . . .	19
3.4	Other tasks . . . . .	20
3.4.1	Issue #297 – Purchase History - Export to CSV sometimes fails . . . . .	20
3.4.2	Issue #361 – Basic Data/Customer Data - Export failure . . . . .	21
3.4.3	Issue #361 - Continued . . . . .	22
3.4.4	Issue #332 – Better display of credit note . . . . .	23
3.4.5	Issue #388 – Project field resetting when line data updated in new order/invoice menu . . . . .	25
3.4.6	Issue #414 - VBOOnline Exchange Rate Updater Prototype . . . . .	25

<b>4</b>	<b>Magnus' Tasks</b>	<b>28</b>
4.1	Structure of this section . . . . .	28
4.2	Two factor login (Issue #161) . . . . .	28
4.2.1	Introduction . . . . .	28
4.2.2	Research . . . . .	29
4.2.3	Implementation . . . . .	30
4.2.4	Testing . . . . .	35
4.2.5	Conclusion . . . . .	36
4.3	Other tasks . . . . .	36
4.3.1	Issue #311 - Remove generation of externalOrderId . . . . .	36
4.3.2	Issue #295 - Removal of check for disabling dropdown menu . . . . .	36
4.3.3	Issue #282 - Update license agreement URL on login-page . . . . .	36
<b>5</b>	<b>Report Conclusion</b>	<b>37</b>
<b>6</b>	<b>References</b>	<b>38</b>

## Preface

This report has been written as a part of the course IBE600 Bacheloropp-gave.

We (Magnus and Viktor) both decided to pursue the school's offer of doing an internship at a business and writing a report on that instead of a normal bachelor thesis. The internship was done over the span of our 6th semester of the Bachelor IT and Digitalization study at Molde University College.

We got the opportunity to work for a business that allowed us to study production code, work close to user requirements as well as receive quick feedback on our work. Therefore, we would like to thank Egil Stavik Tautra, Stian Sundsbø and Frode Finnøy for their guidance and mentorship throughout this semester. It has been a very valuable and enjoyable experience for the both of us.

Additionally, we would like to thank Ketil Danielsen for guidance and valuable feedback.

# 1 Abstract

We both felt a gap in our education of hands-on experience and practical application of what we were learning. We arrived at this conclusion on our own, as we had not even met before this semester. The desire to close the gaps in our hands-on experience encouraged us to do an internship for our last semester instead of the typical bachelor thesis.

We were tested before our internship began, and were quickly given access to source code with expectation to learn how to develop the features asked of us. Meeting these expectations and being able to contribute with bug fixes and product enhancements taught us a lot about working in an organization. During the course of our internship, we were given valuable opportunities to network. This networking included being introduced to local business leaders, which helped us both secure employment before the semester was over.

After a concluded semester, we would both highly encourage any student considering the opportunity to intern for a local business to do so. We believe the experience gained and the networking opportunities can result in a significant advantage when the studies are over and it's time to look for a job.

## 2 Introductory information

### 2.1 Getting the most out of this report

We will try to include appropriate information within every chapter in order to make sense of it. We understand this might lead to some redundancies, but attempts will be made to minimize this, it is not our intent to bloat the report with needless explanations. The editing process is primarily one from top to bottom, so information on terminology might be contained within an earlier chapter, for us to then use shortened versions later.

We will try to explain abbreviations or expressions that are not a part of common English, but a fundamental grasp of programming is expected, as it is not within the scope of this report to explain those concepts.

### 2.2 Entry test

Before we were accepted as interns, we were administered an entrance test at the end of our 5th semester. We are writing about it first, because it was our first interaction with the business, and while it prompted us to learn some things, those things didn't prove essential to our work throughout the semester.

The test involved performing some basic tasks in the VBOnline web app, as well as creating a simple application demonstrating integration by sending calls to the VBOnline API to retrieve and modify specific information.

To accomplish this, we had to familiarize ourselves with how to write code to interact with external resources. This meant we had to learn how to interface with REST APIs by writing asynchronous code.

In the following segments, we will cover some basics about what we had to learn to complete our test. This will include some basics about REST APIs and the related tool Swagger. It will also describe how we obtain a bearer token for authenticating our requests to the API. Lastly, it will briefly go over the significance of asynchronous code versus traditional code in writing applications with these types of technologies.

After that, we will write some specifics about the test itself.

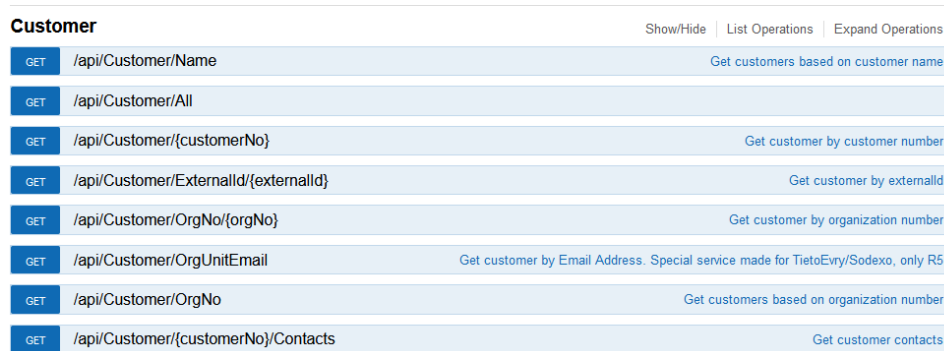
#### 2.2.1 The VBOnline REST API (and REST APIs generally)

The VBOnline API is a REST API that lets you retrieve data from a Visma Business (which is where the VB in VBOnline comes from) database by sending HTTP requests to the server, which sends a response back. The request format is defined by which endpoint you're requesting data from, where an endpoint is a parameter which you append to an URL.

For example, the `"/customer/all"` endpoint will serve you different data than the `"/customer/<customer number>"` endpoint. But both can be appended to the same base URL. In these two examples, the first endpoint gets you every customer in the database and in the second one gets you the details for one customer you specified using their customer number. The specifications of each endpoint request and response are documented by an automatically generated website using a tool called Swagger.



## 2.2.2 API documentation using Swagger



The screenshot shows a Swagger API documentation page for a 'Customer' API. At the top, there are links for 'Show/Hide', 'List Operations', and 'Expand Operations'. Below this, a list of API endpoints is displayed, each with a 'GET' method and a description of the operation. The endpoints are:

Method	Endpoint	Description
GET	/api/Customer/Name	Get customers based on customer name
GET	/api/Customer/All	
GET	/api/Customer/{customerNo}	Get customer by customer number
GET	/api/Customer/ExternalId/{externalId}	Get customer by externalId
GET	/api/Customer/OrgNo/{orgNo}	Get customer by organization number
GET	/api/Customer/OrgUnitEmail	Get customer by Email Address. Special service made for TietoEvy/Sodexo, only R5
GET	/api/Customer/OrgNo	Get customers based on organization number
GET	/api/Customer/{customerNo}/Contacts	Get customer contacts

Figure 1: Screenshot of swagger for a subset of customer endpoints.

Swagger is commonly used to automatically generate documentation for API end points, and it lets you see sample requests and responses for most available endpoints (see fig. 1). It is something you can enable as you develop an API, and as long as your endpoints and responses are formatted a certain way, it will automatically generate documentation for your API with sample requests and responses. This is ideal when your business model involves selling API access to other businesses, as it gives them access to format specifications without requiring you to write a book about it. In VBOOnline's implementation it also lets you send requests and receive responses as long as you have an appropriate bearer token.

### 2.2.3 Authenticating using the bearer token

The bearer token serves as authentication and is something you can get from VBOOnline's OAuth2<sup>1</sup> service if you can provide a valid user name and password to the "/token" endpoint, it will send a response back with a valid bearer token<sup>2</sup> you can use to authenticate yourself against the other endpoints.

### 2.2.4 Why we need to use asynchronous code to talk to remote systems

To make use of all these systems we had to execute logic using asynchronous code. A thorough explanation of asynchronous code is outside the scope of this report, but an explanation of the differences from synchronous code deserve mention. With synchronous code like you would find in most local applications, the computer will attempt to execute the code as quickly as possible. With asynchronous code you might send a request to an external server and want to do something with the response that is returned to you. The way this is commonly solved is referred to as async/await[15]. The await keyword when used in an async function will pause the execution of your code until your request has received a response. This lets a programmer write logic flow very similar to synchronous code by altering some parts of the syntax.

<sup>1</sup>OAuth2 is a form of authentication provider, you can read more at <https://oauth.net/2/>

<sup>2</sup>Typically a string of hexadecimal characters the endpoint can recognize as an authorized use.

## 2.2.5 Example async/await functions

These are the functions<sup>3</sup> used to complete the entrance test. We include them here to reduce the need for long explanations while describing the test itself. Note the keyword `async` at the beginning of the function definitions. This is important, as it allows us to use the `await` keyword in our code to signal to the interpreter that we need to wait for this to resolve before moving on with code execution.

```
async function getToken(api, user, password){
  const response = await fetch(api + "/token", {
    method: "POST",
    headers: {
      "Content-Type":"application/x-www-form-urlencoded"
    },
    body: "grant_type=passwordandusername=" + user + "andpassword=" + password
  })
  const token = await response.json();
  return token;
}
```

**Codeblock 1:** Code to get token for intern test

This function takes the parameters of the API base url, as well as a username and password. The final resource endpoint is dedicated within the function itself using a base URL from the `api` parameter.

As this is a POST-request[16], the username and password is formatted appropriately to be a part of the request body using the API's requested content type, which in this case is "application/x-www-form-urlencoded".

```
async function getCustomer(api, customer, token){
  const response = await fetch(api + "/api/Customer/" + customer, {
    method: "GET",
    headers: {
      "Authorization":"bearer " + token.access_token,
      "Content-Type":"application/json"
    }
  })
  const customerData = JSON.stringify(await response.json());
  return customerData;
}
```

**Codeblock 2:** Code to get customer data for internship test

This function takes the parameters: `api`, `customer` and `token`. These correspond to the API base URL, the customer's customer number and the bearer token gained from `getToken()` respectively. This means in order to use `getCustomer()` we first need to request a token and resolve the request with a response. If the customer exists and the token is correct, a customer's data will be returned in the JavaScript Object Notation (JSON) format, as specified in the `Content-Type` field.

---

<sup>3</sup>Written by Viktor in this case

```
async function displayCustomerData(){
  document.getElementById("customerh2").innerHTML = "Customer(#" + customerNumber + "):";

  const token = await getToken(apiURL, userName, password);
  document.getElementById("token").innerHTML = token.access_token;

  const customer = await getCustomer(apiURL, customerNumber, token)
  document.getElementById("customer").innerHTML = customer;
}
```

**Codeblock 3:** Code to get and display token and customer in application

This function packages the previous functions and handles `getCustomer()`'s dependency on a token by running `getCustomer()` once the request for a token has been resolved.

`displayCustomerData()` is a very rudimentary function which doesn't take parameters and simply uses the variables at the top of the application's code. But from this point, it wouldn't be very complicated to parameterize it and add a web form front end to customize which API provider you are using, which user name and password to authenticate with, as well as which customer ID to retrieve information for. But this was outside the scope of the requested application for the test.

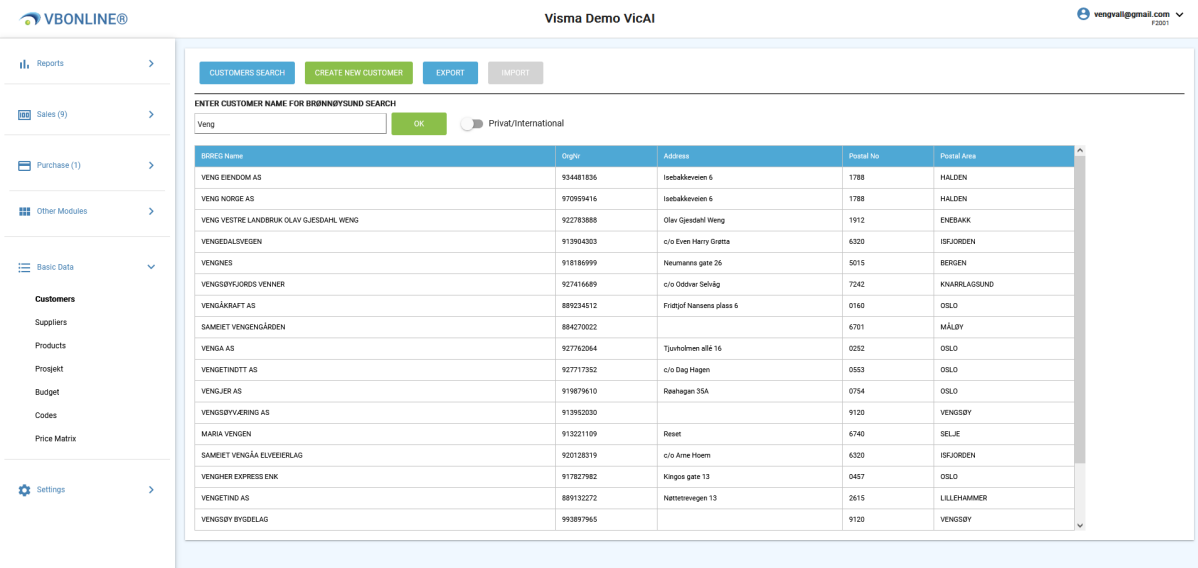
## 2.2.6 The test

The test consisted of two parts. The first part was to navigate and use VBOOnline Web, and the second part was to develop a small application to request information from the VBOOnline API.

### Part 1: Using VBOOnline Web to create a customer and invoice

The task was to create a customer in the web app, then create an invoice. This was simple enough and was much like using any web-based interface. We were free to choose which customer to make (see figure 2 for an example of the user interface), as long as we registered them with our own email address. The reason for registering with our own email address was in order to receive confirmation emails when posting invoices. This entire step amounted to simply clicking through some menus and it was fairly intuitive. No new learning was required for either of us to accomplish this step.

Everything we had access to in the web app was in a development environment, so nothing we did was going to disrupt anything important.



The screenshot shows the VBOOnline web interface. The top navigation bar includes the VBOOnline logo, the text 'Visma Demo VicAI', and a user profile dropdown for 'vengvalt@gmail.com'. The left sidebar contains navigation menus for Reports, Sales (9), Purchase (1), Other Modules, Basic Data, Customers, Suppliers, Products, Prosjekt, Budget, Codes, Price Matrix, and Settings. The main content area features a search bar with 'Veng' entered and a 'Private/International' toggle. Below the search bar is a table of customer records.

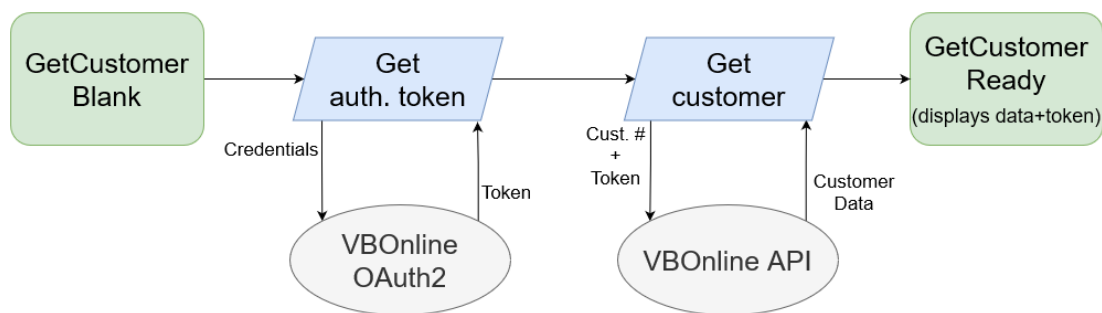
BRREG Name	Orgnr	Address	Postal No	Postal Area
VENG EIENDOM AS	934481836	Isebakkeveien 6	1788	HALDEN
VENG NORGE AS	970959416	Isebakkeveien 6	1788	HALDEN
VENG VESTRE LANDBRUK OLAV GJESDAHL WENG	922783888	Olav Gjesdahl Weng	1912	ENEBAKK
VENGEDALVEGEN	913904303	c/o Eren Harry Gresta	6320	ISJORDEN
VENGES	918186999	Neumanns gate 26	5015	BERGEN
VENGSSBYJORDS VENNER	927416689	c/o Oddvar Selvig	7242	KNARRLAGSUND
VENGAKRAFT AS	889234512	Fridtjof Nansens plass 6	0160	OSLO
SAMEIET VENGENGÅRDEN	884270022		6701	MÅLBY
VENGA AS	927762064	Tjuholmen alle 16	0252	OSLO
VENGETINDTT AS	927717352	c/o Dag Hagen	0553	OSLO
VENGER AS	919879610	Røshagan 35A	0754	OSLO
VENGSSBYVÆRING AS	913992030		9120	VENGSSBY
MARIA VENGEN	913221109	Riset	6740	SELJE
SAMEIET VENGÅA ELVEVEIERLAG	920128319	c/o Arne Hoem	6320	ISJORDEN
VENGER EXPRESS ENK	917827982	Kings gate 13	0457	OSLO
VENGETIND AS	889132272	Nettesvegen 13	2615	LILLEHAMMER
VENGSSBY BYGDELAG	993897965		9120	VENGSSBY

Figure 2: VBOOnline Interface Example: Customer Menu

### Part 2: Developing a small application to retrieve data from the VBOOnline API

This step was aimed at testing our technical abilities and had to do with creating our own application that fetched information from the VBOOnline API.

The calls were supposed to retrieve data for the customer we had created in the first part of the test. In order to do this, we both studied the specifications of the REST API endpoints we had to use, which was made conveniently available through the VBOOnline Swagger site[13].



**Figure 3:** A simple figure showing the flow of retrieving a customer from the API.

The application described in this text uses the code from the code examples in chapter 2.2.5 of this report. It first retrieves a token from the “/token” end point for an authentication token. Then the application sends a request to the “/api/Customer/{customerNo}” end point with the appropriate token and customer number, which retrieves the data for the specified customer. The application was written as a simple HTML page with the previously described functions baked in. The customer field output after a customer has been retrieved is shown in figure 4.

```

Customer(#59928):
{"id":11019661,"externalId":"aec6ad3b-9ea8-79ca-ebf2-14d63b9b122d","customerNo":59928,"companyNo":"934481836","name":"VENG EIENDOM AS","address1":"Isebakkeveien 6","address2":null,"address3":null,"address4":null,"postalNo":"1788","postalArea":"HALDEN","emailAddress":"vengvall2@gmail.com","phone":null,"paymentTerms":null,"paymentMethod":null,"taxCode":null,"documentDeliveryMethod1":53687091,"documentDeliveryMethod2":null,"mobilePhone":"123456789","countryNo":47,"customerPriceGroup1":null,"customerPriceGroup2":null,"customerPriceGroup3":null,"deliveryMethodForCustomer":null,"languageNo":47,"currencyNo":47,"customerPreferences1":null,"group1":null,"group2":null,"group3":null,"group4":null,"group5":null,"group6":null,"group7":null,"group8":null,"group9":null,"group10":null,"group11":null,"group12":null,"creditLimit":null,"creditDenied":null,"euVatRegistrationNo":null,"information1":null,"information2":null,"information3":null,"information4":null,"information5":null,"information6":null,"information7":null,"information8":null,"associateProcessing":null,"invoiceCustomerNo":null,"autoInvoiceOperatorNo":null,"autoInvoiceEmailAddress":null,"personId":null,"yourReference":null,"district":null,"eanLocationCode":null,"tradeArea":null,"createdDateTime":"2021-12-21T19:15:00","createdByUser":"vengvall@gmail.com","changedDateTime":"2022-04-10T03:31:00","changedByUser":"API@himolde.no","warehouseNoForCustomer":null,"factoringCompanyNo":0,"factoringNo":null,"formId":0,"directDebitingStatus":0,"bankAccount":null,"bankNo":0,"autoInvoiceEndpointId":null,"autoInvoiceEndpointScheme":null,"OrgUnitTypeNo":0,"OrgUnitTypeValue":0,"prescribers":[]}
  
```

**Figure 4:** The customer output for the second part of the test

## Results

We were both able to finish the test in a relatively short time without outside help, so we were both accepted. We began working on our other tasks at ØB Innovation at the beginning of our 6th semester.

## 2.3 About the workplace

### 2.3.1 ØB Innovation

ØB Innovation was started on September 1. in 2016 as a collaboration between ØB Solutions AS, also known as ØkonomiBistand, and Molde-based entrepreneur Egil Stavik Tautra.

Their idea was to develop a complete API for Visma Business, a prolific enterprise resource planning (ERP) system.

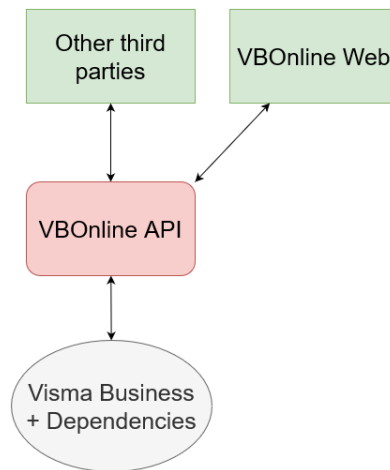


Figure 5: Simplified look at how the API helps build services such as VBOOnline Web.

In addition to their API, ØB Innovations develops and delivers a web-based interface called VBOOnline Web. VBOOnline Web leverages their own VBOOnline API to offer services based on Visma Business.

We were given tasks primarily focused around working on VBOOnline Web to deliver product enhancements and bug fixes. The majority of the report will be about our work on this.

### 2.3.2 VBOOnline Web

VBOOnline Web is an SPA<sup>4</sup> based on the AngularJS<sup>5</sup> framework by Google, which is designed around the MVC architecture<sup>6</sup>. The application relies on ØB Innovation's API to fetch relevant data for its business processes using asynchronous code similar to what was exhibited in our entry test. Unlike our entry tests, these asynchronous functions are grouped in structures referred to as service-modules<sup>7</sup> in the source code. Service modules should be categorized based on what data they fetch.

Providing a browser-based web application allows smaller clients to leverage some of the value in Visma's ERP system and does so by implementing a middle-layer of business logic<sup>8</sup> to handle some of the complications for them.

<sup>4</sup>Single-page Applications (SPA) dynamically create site content by using logic to fetch and present data from a server. It stays on one page rather than loading new pages using URLs

<sup>5</sup>AngularJS is a web development framework for developing SPAs

<sup>6</sup>MVC stands for Model-View-Controller, wherein the model defines data, view defines the display and the controller defines logic.[18]

<sup>7</sup>The functions contained within customerService.js fetches customer-related data, etc.

<sup>8</sup>What we call the algorithms and rules implemented to handle the calculation, routing and transformation of data to conduct business.

## 2.4 How we worked

### 2.4.1 Location

For office space we were allowed to use the meeting room at Sandvegen 9. Being at the office afforded us the ability to quickly communicate with staff as required. Access to quick feedback or hands-on assistance with a problem was pleasant after the majority of our studies had been done throughout the COVID-19 pandemic. Being able to work in the same room allowed us (Viktor and Magnus) to co-operate on tasks at work as well as group projects in other courses when we realized we worked well together.

We noticed good value in having a colleague in the same room to discuss problems with, even if we weren't working on the same thing. A second set of eyes can catch an obvious solution because you're fixated on a bigger challenge. We didn't implement pair programming<sup>9</sup>, but when we got stuck it was very valuable to have someone to discuss the problem with. It is a phenomenon<sup>10</sup> observed by some coders that simply explaining a problem to someone else leads to a solution.

Being at the office also gave us the opportunity to sit in on a client meeting. This was a product demo for VBOOnline Web, as well as listening to the prospective client's needs as an organization. The two parties were trying to get a sense of the others' needs, rather than being a pure sales pitch. This was interesting to witness, and something we would not have had the opportunity to do at school.

### 2.4.2 Developer tools

Visual Studio Code (VSCode) was the IDE<sup>11</sup> we used for development on the web app. It is an IDE very suitable for JavaScript and supports many plugins that make working on a large composed website easier and is in the case of a codebase as large as VBOOnline Web essential.

We both used extensions for VSCode to run local web servers<sup>12</sup>. These extensions are used to run instances of the VBOOnline website on a localhost address. They can still make calls to the VBOOnline API and allows for development against a version of the website that quickly represents changes made to the code.

The ability to host our version of the code on a local web server was vital. It enabled us emulate a live environment to run and test our code, as the data our code fetched was from a live server. This allowed us to make sure our features worked, except in edge cases<sup>13</sup> where user data caused critical differences. Unless specifically stated otherwise, all the code from the issues we worked on has gone out to customers to be tested on their systems.

We will be covering code libraries and specific third party services in the task where they were relevant, as these weren't uniform or lacked equivalents across our separate tasks.

### 2.4.3 Source control

Source control is the practice of tracking and managing changes to code. We used GitHub to accomplish this.

GitHub was used extensively to track branches<sup>14</sup> and code commits<sup>15</sup>. GitHub provides a VSCode plugin that makes this easier to manage while actively developing without needing to dedicate much time and energy to source control.

The theory behind source control is something we studied in a course called IBE205 Agile Methods, but we did not get a chance to apply it extensively at the time. Working on a large project such as this necessitates the ability to separate what each of us is working on. Especially in allowing feature development and bug fixing

---

<sup>9</sup>A concept we heard about in IBE205 Agile Methods

<sup>10</sup>It's called rubber duck debugging<sup>[4]</sup>, and some coders even have a rubber duck on their desk for this purpose.

<sup>11</sup>An integrated development environment is software dedicated to software development and should include tools for, for example: debugging, syntax highlighting and source control. VSCode supports a lot more than this, too much to list.

<sup>12</sup>IIS Express was used by Viktor and Live Server was used by Magnus. The difference was mainly based on IIS Express lacking Mac support

<sup>13</sup>For example Issue #297 where user data caused site functionality to break.

<sup>14</sup>Branches are splits from the main source code that allow us to isolate feature development

<sup>15</sup>Commits are the ability to submit changes to code along with a comment

in relative isolation, knowing we're not affecting each other with experimental changes during development. We finish the work on our own feature branch before it gets merged in the master branch.

### Some other GitHub terminology:

When we refer to issue numbers in the format of "Issue #<number>", it is GitHub's issue tracker format we are referring to.

Pushes take the current backlog of commits and sends them to the GitHub repository, merging changes and creating newer versions of the source code containing your changes.

Pulls are requests from the GitHub repository for source code from a branch or the master branch.

Pull requests are ways to notify that we have made changes that we believe are ready to be merged with the main branch. They can be approved or declined, if the person reviewing the request thinks it looks good or could use some more work.

## 2.4.4 Project management

As ØB Innovation is a fairly small team, they use GitHub's project boards<sup>16</sup> to assign and track issues. It covers the team's needs by providing the necessary issue tracking, commit message and discussion board functionality in case any of this would be desirable. We were assigned our semester projects through this tool, and it kept track of our code commits to the related issues throughout the process.

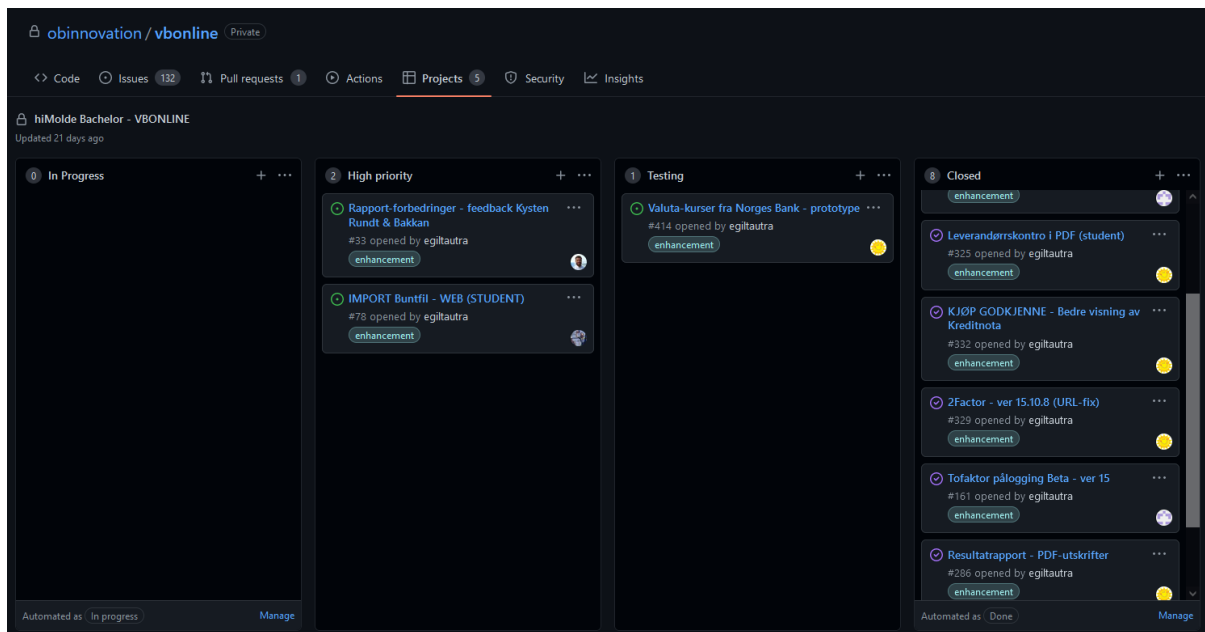


Figure 6: Project kanban board

What wasn't covered on GitHub gets communicated via company email, as it's simpler to integrate customer feedback there when the team is too small to have an employee dedicated solely to project management.

### Writing tools

We began documenting our work by using local copies of Microsoft Word.

When work began to take shape, we started writing the majority of our report into the cloud-version of Word, as it let us co-operate and we were running into issues with Google Docs.

To write the final version of the report, we used the online LaTeX editor called Overleaf.

<sup>16</sup>This is a rather typical kanban board that tracks goals and their respective status



#### **2.4.5 Documentation**

Documentation for systems was not a priority. Effort was instead made to name variables and functions well so that those maintaining the system can read the code. We also saw that customers were more prone to contacting the office about any potential problems rather than spending much time troubleshooting on their own. This aligns well with modern ideas of businesses treating their products as a service, as we've been taught about in IBE505 Industriell Digitalisering.

## 3 Viktor's Tasks

### 3.1 Structure of this section

This section is my report on the tasks I (Viktor) worked on for this semester. It begins with the main tasks. These will be split into segments for introduction, research, implementation, testing and conclusion. I also worked on other issues unrelated to my main tasks, but I will cover these after the main tasks. The section for smaller tasks will suit their scope, and might not be as extensive as the main ones.

The main task assigned to me for my internship was to add features that would allow users to download PDF reports for the result report as well as the customer and supplier ledger in VBOOnline Web.

### 3.2 Save PDF for results report (Issue #286)

#### 3.2.1 Introduction

This issue was a feature request to let users save a report of a VBOOnline Web results report to PDF. This was the first main task I worked on.

	Results 2017 Jan - Sep	Budget Name 2017 Jan - Sep	Results 2016 Jan - Sep	Results 2017 Sep	Budget Name 2017 Sep	Results 2016 Sep	Prognosis
Revenue	1 544 485	0	0	398 213	0	0	1 544 485
Cost of goods:	54 025	0	0	0	0	0	54 025
<b>Contributions:</b>	<b>1 490 460</b>	<b>0</b>	<b>0</b>	<b>398 213</b>	<b>0</b>	<b>0</b>	<b>1 490 460</b>
Operational expenses:	369 800	0	54 550	50 625	0	54 550	369 800
Salaries:	1 108 261	0	102 883	165 590	0	102 883	1 108 261
Depreciation:	0	0	0	0	0	0	0
<b>Sum operation:</b>	<b>1 478 061</b>	<b>0</b>	<b>157 433</b>	<b>216 215</b>	<b>0</b>	<b>157 433</b>	<b>1 478 061</b>
<b>Operational res.:</b>	<b>12 399</b>	<b>0</b>	<b>-157 433</b>	<b>181 998</b>	<b>0</b>	<b>-157 433</b>	<b>12 399</b>
Financial income:	0	0	0	0	0	0	0
Financial costs:	111	0	0	111	0	0	111
<b>Financial results:</b>	<b>-111</b>	<b>0</b>	<b>0</b>	<b>-111</b>	<b>0</b>	<b>0</b>	<b>-111</b>
<b>Result before tax:</b>	<b>12 288</b>	<b>0</b>	<b>-157 433</b>	<b>181 887</b>	<b>0</b>	<b>-157 433</b>	<b>12 288</b>
<b>EBITDA:</b>	<b>12 399</b>	<b>0</b>	<b>-157 433</b>	<b>181 998</b>	<b>0</b>	<b>-157 433</b>	<b>12 399</b>

Figure 7: VBOOnline Web Results Report Example

The format of the table was a little peculiar and difficult for me to get into alongside other on-boarding<sup>17</sup> activities. The result is that I learned a lot, but I could have done this better in hindsight.

#### 3.2.2 Research

The advice I got when I began working on this issue was to look at the result report's CSV<sup>18</sup>-export function. As I had already been tasked with looking at fixing a bug in this function as my first minor task<sup>19</sup>, I was already a little familiar with how it worked.

After investigating the CSV-export function, I saw it gathered the data for the export by using a form of web scraping. I decided to solve my problem similarly, and started looking into it.

My next step was to use the browser inspector tool to figure out how the report was built. This is an important step if I was to use a form of web scraping to access the data in the results table. This proved difficult when the structure used for the table was built using <div> elements rather than a normal HTML <table> structure. This made me consider a simpler solution of exporting the entire element to the PDF file, thus freeing up time to work on other things rather than unravelling the structure.

This was when I started looking for existing solutions to export HTML elements to PDF files and found a JavaScript library called jsPDF<sup>[12]</sup> which had solved all of this rather nicely, and had good documentation<sup>[11]</sup>. I decided this library would be able to accomplish what I needed and more, so I decided to move ahead with it.

I considered using html2canvas<sup>[20]</sup>, which is a library which can render a website's elements to a static image. That image could then be pushed into a PDF. I found it more difficult to use than jsPDF alone, and decided

<sup>17</sup> Learning the tools and getting used to the work environment.

<sup>18</sup> CSV is a standard data format for tables using very simple syntax, it is well supported in Excel for example.

<sup>19</sup> Issue #297 in chapter 3.4.1 of the report

it offered no benefit for my needs. I mention it because it comes up often when researching exporting HTML content to PDF.

### 3.2.3 Implementation

In order to implement the functionality to save the report, I first added a new button next to the existing button to export to CSV, but now for PDF. Then I looked at how the CSV button linked itself to its corresponding application function and did something similar.

This turned out to have some rather specific syntax tied to it, called a directive in Angular. When referring to a directive in an AngularJS application, an HTML-element containing a dash-delimited attribute is tied to a corresponding directive using camel case naming. For example:

```
<div tablename="accountingmatrixtable" filename="ResultsTable.pdf" export-div-to-pdf></div>
```

will attach to an AngularJS directive named with the following convention:

```
exportDivToPdf
```

Any other attributes on the HTML-element are passed into the directive and can be accessed using the `attrs` object. For example, I can get the value of the `tablename` attribute by referring to `attrs.tablename` within the directive, and this is in fact how my code identifies the correct element to export to the PDF.

As this particular directive didn't end up being very large, I will include it in its entirety to provide an example of a functional AngularJS directive:

```
app.directive('exportDivToPdf',function(){
return {
  restrict: 'A',
  link: function (scope, element, attrs) {
    element.bind('click', function(e){
      const fileName = attrs.filename;
      const tableName = attrs.tablename;
      savePdfFromElement(tableName, fileName);

      function savePdfFromElement(elementID, filename){
        window.jspdf = window.jspdf.jspdf;
        const width = parseInt(window.getComputedStyle(document.getElementById(elementID)).width.slice(0,-2))+20;
        const height = parseInt(window.getComputedStyle(document.getElementById(elementID)).height.slice(0,-2))+30;
        let pdf = new jsPDF('l','px', [height, width]);

        pdf.html(document.getElementById(elementID), {
          callback:function(pdf) {
            pdf.save(filename);
          },
          x: 10,
          y: 10,
        });
      }
    });
  });
});
```

**Codeblock 4:** The complete directive for exporting the results table to a PDF file

This directive begins with a fair amount of boilerplate<sup>20</sup> code. For more information about this code and AngularJS in general, there is documentation<sup>[7]</sup> online.

The directive was attached to the button so that it takes the table and file name from the button's properties and passes it into a function within the directive called `savePdfFromElement()`.

`savePdfFromElement()` begins by setting a jsPDF object on the current window. Based on my understanding of JavaScript it creates a jsPDF object within the window object, cloning data in `window.jspdf.jspdf`<sup>21</sup>. The

<sup>20</sup>Boilerplate is a common term for syntax that needs to exist in order to make something work, but from a surface level might not make sense.

<sup>21</sup>The documentation instructed this use, but I couldn't find an explanation as to why exactly.

function then creates and sets the width and height constants to the values of the computed style of the table element that was passed into the function, and adds some extra pixels to prevent the exported element from being cut off. This is done so that when it creates a new pdf document, it can use the width and height values as dimensions for the page size. Then it calls the `html()` and `save()` method on the pdf object using the element ID and filename respectively. The x and y values represent the margin set for the placement of the object on the finished page.

### 3.2.4 Testing

The extent of the testing on this issue was myself saving various versions of the table in different sizes and shapes and seeing that it didn't break down. This helped me zero in on good values for padding values on height and width as well as good values for margins on the finished document.

I received feedback that users shouldn't be allowed to save the report if a business rule relating to suspension dates was being violated. This was implemented by simply disabling the PDF export button if `isSuspension-DatePassed()` was true. As this condition was checked for elsewhere on the page it was as trivial as also adding the check to the button for PDF exports.

### 3.2.5 Conclusion

I feel the work on issue #286 is not in line with my current standards. But I recognize it as part of the process of learning how to work with Angular and gaining some familiarity with the VBOnline Web code base. It was a necessary step in gaining better understanding for my future work.

Keeping it simple was a compromise made on my part when I saw the underlying data structure of the element I was working with. The report has a complex structure, and I wasn't comfortable iterating over it, though I suspect it would be doable with a tool like jQuery<sup>22</sup>. I also had not yet worked enough with Angular and the way it can store data in a global variable in order to leverage that.

Using what I have learned since, I am confident I could go back and make a better version of this feature. I am electing to apply what I have learned about project management in this semester to the issue, however. I can't always take all the time I want to go back and refine my work until it's perfect, sometimes I have to move on and get other work done.

---

<sup>22</sup>A common JavaScript library for HTML document traversal

### 3.3 Save PDF reports for customer/supplier ledger (Issue #324/ #325 respectively)

The customer and supplier ledger tables share a lot of fields in their table columns, with a few minor deviations. In this light, it made sense to create a single function for them both. The rationale for this was that it's good practice to create reusable code, and that it is easier to maintain one function than two.

Since the two issues encapsulated in this chapter were solved using the same function, I will write about both of them at once. The only major differences in the report are shown as which type of report it is, and some minor tweaks to which table columns are included.

#### 3.3.1 Introduction

Users had requested the ability to export the customer and supplier ledgers to PDF format.

In my opinion, the data structure of the two tables is easier to understand than that of issue #286. And since I didn't foresee needing to wrestle with the underlying data, I saw more potential to work with it. These conclusions lead me to writing a more robust solution that produces a cleaner export to PDF with a proper layout suitable more typical to examples I've seen of similar reports.

Code examples will be included under the corresponding section of program flow.

#### 3.3.2 Research

In the process of resolving issue #286 (chapter 2.2), I also observed a lot of material on the uses for jsPDF. For example how it handles adding text and visual elements to the page by calling methods on the document object, such as `document.text()` and `document.line()`. You define the element to be added through the method parameters. An example using the text method on a jsPDF document object:

```
document.text(<text string>, <width integer>, <height integer>)
```

So I figured out the basics of using jsPDF to place various elements on the page, but not how to represent a table cleanly in a report. I considered writing my own little module to generate tables in a jsPDF document. But I quickly found a JavaScript library called jsPDF-AutoTable<sup>[3]</sup>(AutoTable) which would do exactly what I needed, and decided to use it. It also conveniently came with a demo page<sup>[2]</sup> very similar to jsPDF.

AutoTable is rather convenient since it can render a table in a PDF document. It can do this directly from a referenced HTML element, but also be fed a properly structured two-dimensional array and render a table based on that. On top of handling the table-generation, it also comes with styling features such as row alignment, border styles, cell padding, colors and more. The AutoTable library extends the jsPDF library, so the new methods it adds get appended to the jsPDF document class. This makes it so that using it to add tables is as simple as calling the `document.autoTable()` method with the appropriate data and style configuration you desire.

Since jsPDF and AutoTable both offer good examples to learn from, it was now just a matter of handling the logic of which columns to include and how to structure the header, but this falls more under implementation.

#### 3.3.3 Implementation

The first step was to create a minimum viable product (MVP) that could take the table data from the website and output a PDF containing a neatly formatted table. The first logical step was to implement the jsPDF-AutoTable library and input the table data. I opted to convert the HTML table from the customer report to a two-dimensional JavaScript array. I did this because it would make subsequent operations before table-generation much simpler to code, as interacting with an array in code is simpler than editing DOM-structure as it has a lot of superfluous objects attached to it for this purpose.

After the MVP was produced, I received feedback from within ØB Innovation that it would be nice to get a header that would inform the user what type of report it was, along with the sort of typical data you might find in a report header. Following an example I was given, I set out to make it so that the top of the first page would display relevant metadata; the type of report, the report subject, what time period the report is for, when the report was saved and so on. Implementing the header became the main workload for this issue.

The table containing the report data was simple to generate with jsPDF-AutoTable and had practically been accomplished already with the MVP, but I would also add to this by removing some superfluous columns that would not be required in a local version of the report.

While adding a header and removing some columns seems like a modest request, I didn't want to underestimate the amount of work required to address these requests (I turned out to be correct in this assessment). I spent some time thinking about the structure of this project, and I ended up splitting the program into functions with what I thought were reasonable names as a result. I split the code up into the following main segments:

1. Add header - It is the first thing on the first page of the report, so I add it first.
2. Add tables - This is the main content, and it belongs under the header.
3. Add page count - This can only be added once everything else, otherwise you can have no expectation of the page count being accurate.
4. Save the PDF - There are no more elements to add, so the document finished and we can save it.

These segments may contain their own functions, for the sake of splitting code into functions that accomplish a specific task. This is easier to maintain than one long block of code.

With order of operations explained, I will go into some more detail for each operation and explain how I handled them.

## Header

In order to add a header with useful information I first needed to establish some ground data. I concluded I would need the date the report is being saved, the type of report, the subject of the report and the period the report is for. Once I had all these pieces of information I could add the header I had in mind. Fortunately using a structured approach I could write these functions one at a time. The majority of this information is gathered from the table data itself, except for the timestamp for report generation, as well as which period the report is for.

The timestamp is generated using the current system time. This is handled by the function `getCurrentTime()`, which is called using Norwegian time formatting by default. The function simply acts as a wrapper for a native JavaScript function which handles date and time.

```
function getCurrentTime(localization){
    return new Date().toLocaleString(localization);
}
```

**Codeblock 5:** The `getCurrentTime()` function

The type of report is determined by the function `getReportType()`, which searches the header<sup>23</sup> of the data and looks for specific column names. The presence of a column name corresponding to either a customer or a supplier determines if this is a customer or supplier report.

The report period is determined by a mixture of table data and the AngularJS scope<sup>24</sup> variables to ensure that if the table only contains values for one month but is meant to an annual report it is still registered as an annual report. In this instance, the scope variable called `scope.selectedMonth` is checked, and if it is undefined then the report must be annual, since a month has not been selected by the user.

Customer Payments Account	Page: 1 of 1
Customer: E2 SOFTWARE AS	Report generated: 19.5.2022, 09:17:03
Period: 1 - 12/2020	

**Figure 8:** An example header from the customer report

Drawing the various objects on the page is done by utilizing a tracker variable that acts as a cursor for where to draw. The program advances the cursor every time an element is drawn on the page. This helps ensure

<sup>23</sup>The first row which contains the column titles

<sup>24</sup>Angular tracks its global state with a variable called `scope`. It is effectively global.

there are no inconsistencies or unwanted overlapping of elements on the page. When each of the elements are drawn, they are drawn by calling functions such as `drawLine()` and `writeHeadline()`, which each add their own content to the header of the document. Each element added advances the cursor by an amount appropriate to their corresponding element size. I cover this here because I only needed to use this approach for the header, as the code for the header generates a very predictable format (as seen in figure 8), and the table element is the element which should change significantly in size depending on document content.

## Tables

### The table data

Early in the process of deciding what the report should contain, it was mentioned by management at ØB Innovation that a few of the columns in the web version of the report should be removed from a saved report. The rationale behind removing columns was that some of them would be unnecessary in a printed version of the report, as they refer to online resources that would be unavailable in a printed report. The requirement to remove columns led me to converting the HTML table to an array. I find it simpler to manipulate an array rather than resorting to DOM-traversal<sup>25</sup>.

In order to handle the removal of specific columns, I wrote a few functions to find and remove columns based on their names. The first step was a function that finds the array index of a column based on its name. It does so by searching row 0 of the array, which is typically the header and contains column names in most standard tables.

```
function findColumnByName(columnName, table = arrayTable){
  const tableHeader = table[0];
  for(let cell = 0; cell < tableHeader.length; cell++){
    if (tableHeader[cell] == columnName){
      return cell;
    }
  }
}
```

**Codeblock 6:** Function to return a column's index based on its name.

The second step incorporates that function to walk through a list of names and return an array of indices for all the relevant columns. It was necessary to write a function within it to remove undefined elements as the function to find columns by name is undefined if there is no matching column name.

```
function findIndexofColumns(titleArray, table = arrayTable){
  let indexArray = [];
  for(let title = 0; title < titleArray.length; title++){
    if (!indexArray.includes(findColumnByName(titleArray[title], table))){
      indexArray.push(findColumnByName(titleArray[title], table));
    }
  }
  return removeUndefinedElementsFromArray(indexArray);

  function removeUndefinedElementsFromArray(array){
    return array.filter(function (element){
      return element !== undefined;
    });
  }
}
```

**Codeblock 7:** Function to create a list of column indices based on names input.

The third step of the process is to finally remove all the columns by iterating through the full table array and returning a new array that excludes the specified columns. This is done by first calling `findIndexofColumns()`

---

<sup>25</sup>Such as using a tool like jQuery to select and filter child elements in order to get rid of columns that way.

on a list of column names to find the desired indices. Then the code walks through `arrayTable` and builds a new array that excludes the index of the undesired columns. In the end the new array is returned, free of the unwanted columns.

```
function removeTableColumns(nameArray){
  const filterArray = findIndexOfColumns(nameArray);
  let tempArray = [];
  for (let row = 0; row < arrayTable.length; row++){
    try{
      let tempRow = [];
      for(let column = 0; column < arrayTable[row].length; column++){
        if(!filterArray.includes(column)){
          tempRow.push(arrayTable[row][column]);
        }
      }
      tempArray[row] = tempRow;
    } catch(err) {
      console.error(err);
    }
  }
  return tempArray;
}
```

**Codeblock 8:** Function to remove columns.

Being able to find and remove columns based on their name rather than a hard coded index is good because the column order can change, and it won't cause the code to break. Another benefit is an increased level of abstraction as it makes it easier to spot the column names in the code in case it needs maintenance later. I find that burying low-level code deeper in the program improves readability of the code.

#### Adding the table to the document

After removing the columns from the table, the table has to be added to the PDF document in the appropriate location and with the appropriate styling.

AutoTable handles this by letting you apply style parameters to an added table. The only significant coding hurdle here was solved by using an earlier function to identify which columns are numerical columns and designating them to be right-aligned, as this is an industry standard and looks more orderly. As the style parameters for AutoTable are handled like a JavaScript object, I had the idea to write a function that returns an object I could insert into the AutoTable styling options. The function takes an array of column names, the desired alignment and which array (table) to search as parameters.

```
function addColumnAlignment(columnNameArray, alignment, columnStyleTable){
  const styleString = {
    halign: alignment
  }
  const numberColumnIndex = findIndexOfColumns(columnNameArray, columnStyleTable);

  let workerStyleObject = {};
  for (let columnIndex of numberColumnIndex){
    workerStyleObject[columnIndex] = styleString;
  }
  return workerStyleObject;
}
```

**Codeblock 9:** Column alignment function.

With the columns aligned, the rest of the AutoTable style settings are set using JavaScript Object Notation, and this all gets handled in the `addTables()` function which takes the parent document and the table array as parameters.



```

function addTables(document, table){
  const numberColumnNames = [
    "Dokument", "Document",
    "Debet Beløp", "Debit Amount",
    "Kredit Beløp", "Credit Amount",
    "Cross Reference"
  ];
  const columnStylesObject = addColumnAlignment(numberColumnNames, "right", table);

  document.autoTable({
    startY: 70,
    margin: {
      top:documentMargin+10,
      right:documentMargin,
      bottom:documentMargin,
      left:documentMargin,
    },
    styles: {fontSize: 8, lineWidth: 0.5, cellPadding: 3, },
    columnStyles: columnStylesObject,
    head: table.slice(0, 1),
    body: table.slice(1),
  })
}

```

**Codeblock 10:** Table adding function.

The function begins by defining a list of column names referring to the numerical columns of the table. The list is used to create the style object which tells AutoTable to justify those columns to the right. Then a start position is set, followed by margin and style properties. The choice to hard code the start position comes from the header function producing a very reliably sized element. The margin is derived from a document margin variable set earlier in the larger scope of the wider report function. The styles settings were arrived at by playing with various properties and settings. A full list of the style setting options can be found in AutoTable's documentation on style<sup>26</sup>.

## Page count

The page count is handled after the rest of the document is built. This makes sense, since counting the pages before adding the rest is out of order and pointless.

```

function addPageCount(document, margin = 10){
  const PageCountText = localizePageCount();
  const pageCount = document.internal.getNumberOfPages();
  const documentWidth = document.internal.pageSize.getWidth();

  for(let i = 0; i < pageCount; i++) {
    document.setPage(i);
    let pageCurrent = document.internal.getCurrentPageInfo().pageNumber;
    document.setFontSize(7);
    document.text(PageCountText[0]+pageCurrent+PageCountText[1]+pageCount,documentWidth-margin,margin,"right");
  }

  function localizePageCount(){
    const lang = navigator.language;
    if (lang == "nb-NO" || lang == "no-NO" || lang == "nb"){
      return ["Side: ", " av "];
    } else {
      return ["Page: ", " of "];
    }
  }
}

```

**Codeblock 11:** Page number function.

<sup>26</sup><https://github.com/JonatanPe/jsPDF-AutoTable>

To go through the code, the page count gets localized to either Norwegian or English based on browser language settings. It then gets the total amount of pages using the document's object properties. Then it obtains the document's width. This is required to properly place the page number on the right-hand side of the page by subtracting the desired margin from the document width. The page count is placed on each page by going through the document using a loop that runs iterates once for every page of the document.

## Save

The final step is to save the report as a PDF file. Initially the files were saved as "CustomerReport.pdf" and "SupplierReport.pdf", but I carried out refinements in this area and changed it. The current one to save the file is a single line and looks like this:

```
pdf.save(pdfname+" "+getReportSubject()+" "+getReportPeriod().replace(/ /g, "")+".pdf");
```

The file name will now come out as:

```
"<report type> <the subject of the report> <the time period of the report>.pdf"
```

These changes improve usability by automatically differentiating between reports for different subjects and time periods. It took me very little work, and the information comes from functions that were already written.

## Customer Payments Account

Page: 1 of 1

Customer: E2 SOFTWARE AS  
Period: 1 - 12/2020

Report generated: 24.5.2022, 08:50:20

Voucher Date	Value Date	Text	Document	Debit Amount	Credit Amount	Currency	Cross Reference
		Previous balance		0,00			
05.08.2020	05.08.2020	Utgående faktura	1000800	1 187,50			
05.08.2020	05.08.2020	Utgående kreditnota	1000801	- 1 187,50			1000800
05.08.2020	05.08.2020	Utgående kreditnota	1000802	1 187,50			1000801
18.08.2020	18.08.2020	Utgående kreditnota	1000805	- 1 187,50			1000802
23.09.2020	23.09.2020	Utgående faktura	1000825	25 000,00			
19.10.2020	19.10.2020	Utgående faktura	1000867	0,00			
		New balance		25 000,00			

Figure 9: Complete report for a test customer("E2 Software AS") with few entries.

### 3.3.4 Testing

After showing the MVP to management, they were able to request more features to guide development. These features were the removal of specified columns and the addition of header data. It was up to me to decide how to shape this header, but I was given example reports for inspiration. The columns for removal were decided by them after some debate.

Testing of the code was mostly carried out by me trying out various permutations of the functionality that I could think of. I tested the localization functions to make sure I got the language I would expect based on settings. I also tested things like choosing different customers or suppliers and making sure different time periods would work as expected.

### 3.3.5 Conclusion

Resolving this issue went well. I got an MVP built extremely quickly which showed a proof of concept, and proceeded from there to flesh it out with the requested functionality.

My goal in delivering this feature was to create a function that would produce both ledger reports at the same time, and I succeeded in doing so. This has the benefit of putting less unnecessary code into the source code, and it also helps ensure both reports maintain similar styles. So I am quite happy with the outcome.

On the other hand, two major problems with my work stand out to me.

The occasional lack of parameters in lower level<sup>27</sup> functions hinders readability. I would change this if I started over from scratch. Currently, a lot of the lower level functions rely on variables and information found in the scope of their parent blocks. This is a bad practice and makes it harder to read and understand a single isolated low level function, as it isn't apparent where the variables come from.

I used the table data rather than the source data when the function assembles the report. Using the source data is a bit more correct than simply reading the HTML elements on the page, as it would allow them to replace the current table with another one without breaking the report functionality. There is a potential benefit to doing it the way I have done. Taking the data from the table as seen by the user lets ØB Innovation change how the table gets its data. This might be necessary at a later date due to changes in business logic brought about by for example new legislation.

### 3.4 Other tasks

The smaller issues I was assigned will be covered here. If an issue lacks a research or testing section, it is because there is nothing to write about in it.

#### 3.4.1 Issue #297 – Purchase History - Export to CSV sometimes fails

##### Introduction

This was the first thing I worked on.

There was a bug in the CSV-export function for the results report. When the button was pressed, nothing would happen and the JavaScript function BtoA() threw an error when attempts were made.

This issue turned out to be a character set issue introduced by a user using an unusual hyphen character in a name. An error occurs with the JavaScript function BtoA() when a character in a string is outside the Latin1 range of character encoding.

##### Research

I didn't have to do very much research to solve this issue, but I looked into what BtoA() was good for and why it might break.

BtoA() is a JavaScript function that takes a binary string and converts it to ASCII base-64 encoded data. The method is used to encode data into a format that is less likely to cause communication errors. It can then be decoded using the corresponding AtoB() function on the other end to get the original string back.

According to the sources<sup>28</sup> I found, it can not convert characters that occupy more than one byte as this is not considered binary data. The suggested solution is to convert the character such that it only occupies one byte, but as BtoA() was being used to export to a CSV-document in this use case, I wouldn't be able to access the data on the other end and reverse the conversion.

##### Implementation

The fix for the bug ended up being to expand the existing function that generates the exported string. I added additional filtering to remove the offending character and replace it with a normal hyphen.

```
cellData = cellData.replace(/-/g, "-");
```

The syntax used for this replace is RegExp shorthand in JavaScript.<sup>29</sup>

After adding this line of code to replace the "bad" hyphen with a regular one, the function worked as expected.

##### Conclusion

Feedback from the user with the problem was that this fixed their problem.

<sup>27</sup>A higher level function is made up solely of functions/methods and shows little to no explicit logic. Calculations, logic and data structures is often referred to as lower level.

<sup>28</sup><https://developer.mozilla.org/en-US/docs/Web/API/btoa>

<sup>29</sup>More information on this can be found at [https://www.w3schools.com/jsref/jsref\\_replace.asp](https://www.w3schools.com/jsref/jsref_replace.asp)

### 3.4.2 Issue #361 – Basic Data/Customer Data - Export failure

#### Introduction

This is a similar issue to Issue #297. A lot of the basis for understanding this will have been covered there.

The export function for customer data was failing due to the JavaScript function Btoa() failing to handle some characters in the database.

#### Implementation

On closer inspection the string intended for CSV export in the user data wasn't being cleaned for unsupported characters at all, so I got permission to create a new AngularJS service<sup>[8]</sup> for string utilities, so I created a service module called utilityStringService. My first addition to this utility library was a string sanitization<sup>30</sup> function. It also has support for adding additional terms to replace by giving the function an optional dictionary parameter.

```
let _sanitizeString = function (workerString, additionalTerms = {}){
  const replacementDictionary = {
    "&nbsp;": " ",
    "&amp;": "&",
    "&apos;": "'",
    "&quot;": '"',
    "&gt;": ">",
    "&lt;": "<",
    "&reg;": "®",
    "&copy;": "©",
    "%": "% ",
    "-": "-"
  }
  addToDictionary();
  replaceUsingDictionary();
  return workerString;

  function addToDictionary(){
    if (Object.keys(additionalTerms).length > 0){
      for (let key in additionalTerms){
        replacementDictionary[key] = additionalTerms[key];
      }
    }
  }

  function replaceUsingDictionary(){
    for (let key in replacementDictionary){
      workerString = workerString.replace(new RegExp(key, 'g'), replacementDictionary[key]);
    }
  }
}
```

**Codeblock 12:** String sanitization utility function.

The main work is done by the replacementDictionary data structure which contains a list of common problematic entities. If you look at the end of the list, this function also covers the hyphen from issue #297.

I applied this function to the export functions for all basic data categories (customer, supplier, product, project), not just the one that was a problem in this case (customer). Hopefully this can prevent some future problems before they happen.

#### Conclusion

I'm happy with this solution. The function itself is fairly simple, as choosing the correct data structure did most of the work for me. It let me quickly begin applying string utilities in controllers<sup>[6]</sup> with data export functionality by importing the utility service and calling the correct function.

<sup>30</sup>A common term for cleaning a string of unwanted characters or sequences.

Having a utility service also allows for storing commonly used functionality in a shared resource. This increases code reuse and reduces code repetition<sup>31</sup>.

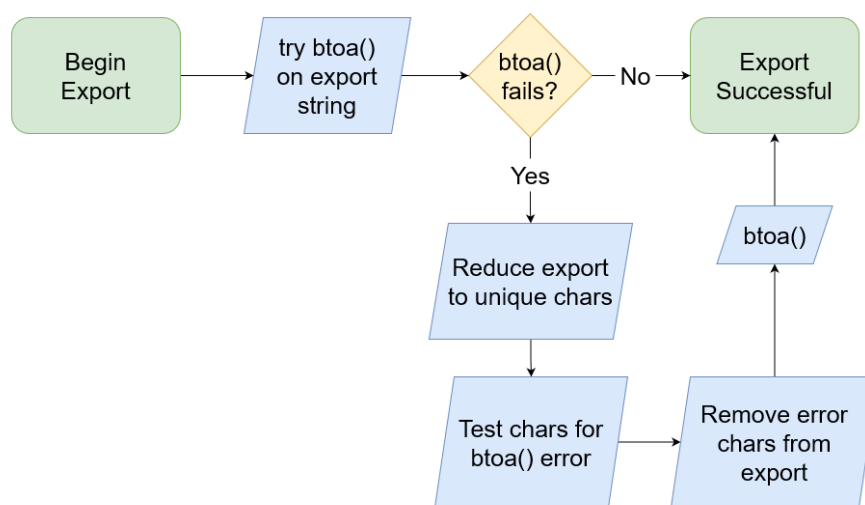
### 3.4.3 Issue #361 - Continued

#### Introduction

Users were having problems with data exports again. Since we weren't given access to user databases, I assumed it was a new unsupported character. I started thinking of a way to create a function that could detect and get rid of incompatible characters. This happened often enough that I wanted a fix that would deal with this once and for all.

#### Implementation

My solution is based on an idea I had to use exception-handling to identify and remove error-causing characters.



**Figure 10:** The flow of the btoa() wrapper.

I wrote a try-catch wrapper function for the export that calls BtoA(). This is so if the function fails it can go into a different function that iterates through the exported string character by character until it finds the offending character and removes it before attempting the export again.

<sup>31</sup>This is in accordance with something called the DRY principle. DRY stands for 'Don't Repeat Yourself'.

```

let _btoaCleaner = function (btoaString){
  try{
    return btoa(btoaString)
  }catch(error){
    return btoa(removeBadSymbols(findBadBtoASymbols(_returnUniqueSymbols(btoaString)), btoaString));
  }

  function removeBadSymbols(badSymbolArray, stringToClean){
    let workerString = stringToClean;
    for (let badSymbol of badSymbolArray){
      workerString = workerString.replace(new RegExp(badSymbol, 'g'), "");
    }
    return workerString;
  }

  function findBadBtoASymbols(stringContainingBadSymbols){
    let badSymbols = [];
    for (let symbol of stringContainingBadSymbols){
      if(tryBtoAOnSymbol(symbol) == false && !badSymbols.includes(symbol)){
        badSymbols.push(symbol);
      }
    }
    return badSymbols;
  }

  function tryBtoAOnSymbol(symbol){
    try {
      btoa(symbol);
      return true;
    } catch (error) {
      return false;
    }
  }
}

```

**Codeblock 13:** Function to remove BtoA()-incompatible characters.

The function works by:

1. Taking the export string and creating a new string consisting of all unique characters in the original string<sup>32</sup>. This is so the function won't keep testing characters that have already been tested.
2. Putting each character through a try-catch block that determines if the character being checked makes BtoA() crash. If it does, it gets added to an array which is returned at the end of the function.
3. Iterating over the array of incompatible characters to remove them from the export string using regular expressions. In this specific method, the regular expression replaces all instances of the incompatible characters with nothing. I chose to replace with nothing because I can't think of a way to determine a suitable replacement character.

## Conclusion

This is a brute force solution, but the export should only fail in cases with incompatible characters in the data. This means the code should only run if an error occurs in the first place. A user would probably like an export even if it takes a moment longer to execute.

Wrapping all strings put through BtoA() with this should prevent these types of errors in the future, so I'm pleased with the solution.

A more permanent and ideal solution would be to sanitize the database to ensure there are no incompatible characters that make their way to the export functions.

### 3.4.4 Issue #332 – Better display of credit note

#### Introduction

<sup>32</sup>This is handled by a function called `_returnUniqueSymbols` which returns a list of unique characters in a string.

This issue was cosmetic. The product manager wanted the user to be able to see whether a posting was a credit note or invoice without having to click on it.

## Research

While this issue seems simple on the surface, a lot of time was spent tracking down what underlying data determines if something is a credit note or invoice.

It was found that it was determined by the field called `documentType` in the global scope variable. A value of 3 meant invoice and a value of 4 meant credit note.

## Implementation

After finding the correct values, it was trivial to write a function called `isCreditNote()` that returned true if `documentType` is equal to 4, and false if not.

```
$scope.isCreditNote = function(documentsItem){  
  if (documentsItem.documentType == 4){  
    return true;  
  }  
  return false;  
}
```

**Codeblock 14:** `isCreditNote()` function.

This function can now be attached to HTML elements and Angular will check its logic. In this case it only shows the appropriate element if the function returns true. For example this span element contains a minus sign that only shows<sup>33</sup> if the document is a credit note:

```
<span ng-show="isCreditNote(item)">-</span>
```

This way the visibility of elements simply get toggled based on whether we're looking at a credit note or not. As shown below.

DocID	Approve	Type	Department	Project	Supplier	Credit Amount	Due Date	Comment	Invoice No
<a href="#">4098</a>	<input checked="" type="radio"/> Partly	Inngående fakturaer			Duett AS	2 048,00	02.05.2022		<a href="#">4326542432</a>
<a href="#">4096</a>	<input type="radio"/> Not Set	Inngående fakturaer			\$ ETERO AS	2 889,00	04.04.2022		<a href="#">S0628181</a>
<a href="#">4095</a>	<input checked="" type="radio"/> Partly	Inngående fakturaer			\$ E2 SOFTWARE AS	5 000,00	28.03.2022		<a href="#">323232</a>
<a href="#">4091</a>	<input checked="" type="radio"/> Approved	Faktura			\$ E2 SOFTWARE AS	- 450,00	10.03.2022		<a href="#">150</a>

**Figure 11:** Credit amount displays a minus on credit notes.

## Conclusion

This was a small request that was implemented with a minimal amount of code. Figuring out which values corresponded to a credit note took more time than I would like. Overall I am happy with the result.

I believe access to a resource where a developer could quickly look up which data field corresponds to which property would be useful. It would have saved me a lot of time on this feature.

<sup>33</sup>As indicated by `ng-show`. `ng-hide` would flip the behavior and hide the element if true.

### 3.4.5 Issue #388 – Project field resetting when line data updated in new order/invoice menu

#### Introduction

After users had selected a project in the web form to create a new order/invoice it would clear itself whenever they updated an order line. This was unintended behavior that negatively impacted the user experience.

#### Research

The problem took some time to track down but was eventually narrowed down to a call to a service function that retrieved a new response from the VBOOnline API that overwrote the existing project with a new order object lacking the project property.

#### Conclusion

My contribution was to figure out why the project field would reset. I did this, but it took a few hours of walking through the application using breakpoints. It taught me I probably have a lot to learn about troubleshooting in web development.

This issue lacks an implementation segment because I didn't implement it. The code to fix this bug was written by a senior developer.

### 3.4.6 Issue #414 - VBOOnline Exchange Rate Updater Prototype

#### Introduction

Management requested an application written in C# that could update the current exchange rate in the VBOOnline API based on the current exchange rate found in the The Bank of Norway(bank/the bank) API.

I spent a weekend (8/04-10/04) working on this. Despite being a small console application, it took a lot of time because I had never used C# to develop an application before.

#### Research

I had done a few integrations<sup>34</sup> in JavaScript before, so I was confident I could do the same in C#. It turned out to be more work than I thought it would be. What I estimate might have taken less than half an hour in NodeJS<sup>35</sup> ended up taking an entire weekend in C#. It took a while because JavaScript is dynamically typed and can create usable objects out JSON formatted strings, but C# is statically typed and needs predefined models<sup>36</sup> to deserialize<sup>37</sup> the response before it can be used.

C# has its own library to serialize/deserialize the JSON format, but not CSV format. I found a NuGet package<sup>38</sup> called CsvHelper[5], which I used to deserialize the bank's API response.

```
List<VBOGetCurrency> CurrencyList;  
CurrencyList = JsonSerializer.Deserialize<List<VBOGetCurrency>>(response.Content);
```

**Codeblock 15:** Example of deserializing a JSON response.

In the example above (Codeblock 15), an empty variable called CurrencyList of the type List<VBOGetCurrency> is declared. VBOGetCurrency is a class intended to contain information about a currency from the VBOOnline API. response.Content is a string in JSON format containing multiple currencies, so it makes sense we read it into a list of currencies. The JsonSerializer class handles this for us with the Deserialize method.

Another NuGet package I used is called RestSharp[14], which is often used to build REST APIs, but I used it to simplify the use of HTTP calls in C#.

<sup>34</sup>Making two systems talk to each other.

<sup>35</sup>A tool to use JavaScript for server code.

<sup>36</sup>A class made to hold all the properties of the API response.

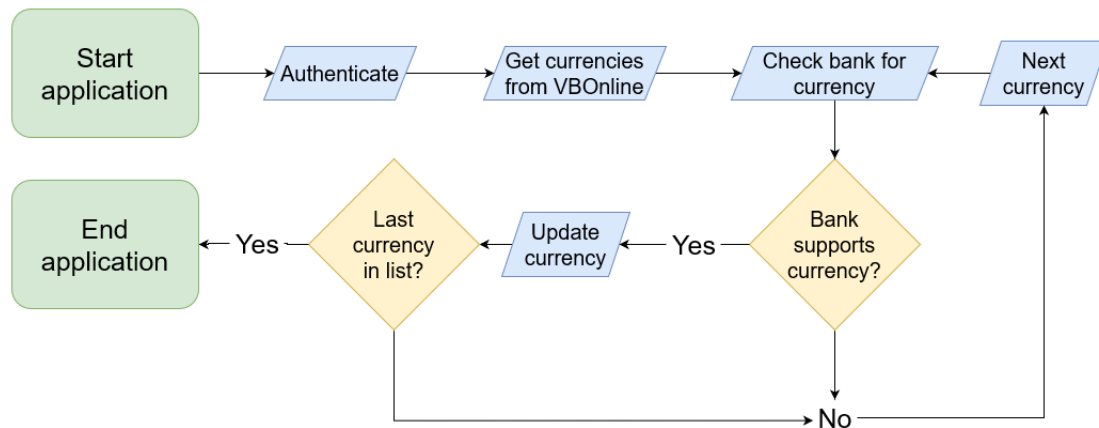
<sup>37</sup>Serialize meaning to take an object and turn it into a data string, deserialize goes the other way.

<sup>38</sup>These can be considered similar to a JavaScript library.



## Implementation

As this application needs two different APIs to talk to each other, it is an integration project. That means a lot of asynchronous code had to be used. The concepts of this was briefly covered in the introductory section on the test we were given ahead of our internships.



**Figure 12:** Simplified program flow for exchange rate updater.

The order of operations is as follows:

1. The application must first get authorization to use the VBOOnline API. I used the credentials given to us for the entry test to develop the application. I made it possible to use different endpoints and credentials when starting the application.
2. The application then requests a list of currencies from the VBOOnline API.
3. It goes through these currencies one at a time and requests the exchange rate from the bank.
4. If the currency is found in both API endpoints a PUT request is created using data from both sources and sent to the VBOOnline end point with updated exchange rate information and timestamp.
5. It prints out status updates and what it does to the console prompt, so it is possible to see where things go wrong if they do. For instance, if a currency is found in VBOOnline that the bank doesn't support it will say so. If an update is sent to VBOOnline it will notify of this, and likewise if not.

The user must first supply a valid endpoint, username and password (fig. 13 shows an example using nonsense credentials). There is a maximum of 5 tries per execution of the program.

```
Use test data? (y/n)
n
Authentication attempts remaining: 5

Endpoint: custom
Username: credentials
Password:
```

**Figure 13:** Inputs for endpoint and credentials.

If valid credentials are entered, the program will run. and provide status updates on what was updated and what wasn't, as well as the reason if it wasn't updated.

For the main sequence of the program I made some efforts to give the application a user friendly printout as it updates the currencies (as seen in fig. 14) so that it is easier to follow what the application does. For

example the red text and cross next to Norges Bank for NOK means that the exchange rate for NOK was not found in the bank's database<sup>39</sup>.

```
SEK found in:
- [v] https://vbi.okonomibistand.no/VBInterfaceOBIDev
- [v] Norges Bank

Last observation on 2022-05-23:
- 0.9262 NOK per SEK
Update for SEK to https://vbi.okonomibistand.no/VBInterfaceOBIDev successful

NOK found in:
- [v] https://vbi.okonomibistand.no/VBInterfaceOBIDev
- [x] Norges Bank

JPY found in:
- [v] https://vbi.okonomibistand.no/VBInterfaceOBIDev
- [v] Norges Bank

Last observation on 2022-05-23:
- 0.07023 NOK per JPY
Update for JPY to https://vbi.okonomibistand.no/VBInterfaceOBIDev successful

EUR found in:
- [v] https://vbi.okonomibistand.no/VBInterfaceOBIDev
- [v] Norges Bank

Last observation on 2022-05-23:
- 9.5693 NOK per EUR
Update for EUR to https://vbi.okonomibistand.no/VBInterfaceOBIDev successful
```

Figure 14: Console application output of exchange rate updater.

Some use of colors to highlight important lines is an easy but worthwhile change.

## Testing

This code has not been tested in a live environment, it ended up being a proof of concept.

Testing consisted of me running the application and trying various cases such as entering the wrong credentials or endpoints. At first this led to some crashes which I fixed by more rigorously checking for null values and handling exceptions.

## Conclusion

For my first complete C# application I am happy with it. Looked at as a learning experience in statically typed languages and object oriented programming, I consider it a great success. Looked at as a good example of object oriented programming principles<sup>40</sup> it is not a great success. But it does what it was supposed to and is stable.

An improvement I would make is to change the exchange rate updates to be run in parallel, rather than queuing them. Currently the application gets a list of currencies in the VBOnline API endpoint that is being updated, then checks the bank's exchange rate API for whether they support that currency. If the currency is found, it updates the one in the VBOnline API. Instead of going through and checking each currency individually I could request all supported currencies from the bank up front. This could be used to create a matched list of supported currencies from VBOnline. Afterwards, the application could create a list of tasks that all execute simultaneously against the VBOnline API endpoint. This way, each task could complete in parallel rather than a series, which would result in a shorter run time.

<sup>39</sup>This makes sense in this case because the exchange rate is based on what something is worth compared to NOK in the first place, so this would always return 1 on any day.

<sup>40</sup>For example the SOLID principles[17].

## 4 Magnus' Tasks

### 4.1 Structure of this section

This section contains a report on my (Magnus) specific tasks for this internship. It is divided into introduction, research, implementation, testing and conclusion.

My main assignment for the internship was to implement a two-factor authentication solution for VBOOnline. The assignment was considered larger than the other tasks available to us, so it was decided that I was to focus on it alone. This proved to be a good decision as the assignment turned out to take up my time for most of the internship.

### 4.2 Two factor login (Issue #161)

#### 4.2.1 Introduction

Some customers using VBOOnline had requested two-factor authentication to be added to the login system. The existing system only required a username and password and was built with C#<sup>41</sup> and OAuth2<sup>42</sup> running on the VBOOnline back-end. My solution is meant to exist alongside it and can be activated for specific customers requiring the extra security.

#### What is two-factor authentication

Two-factor authentication, also called Multi-factor authentication, broadly describes a authentication system where the user has to confirm their identity with more than one source. From here on I will refer to it as 2FA, which is a commonly used abbreviation. In a traditional login scenario where the user enters a username and password, there is only one confidential source, namely the password. This opens up a plethora of possibilities for adversaries gaining unauthorized access. By adding another layer of defense, like verifying that the user also has access to the email they are trying to log in with, the chance of getting compromised is reduced. An adversary would now have to both hack/crack the password as well as have access to the user's email in order to break in.

In addition to verification of email-ownership, there are several other methods available. For example, a number can be sent to the user's phone number by SMS, which the user then enters in the app to confirm they have access to their phone. Another one is using an app like Google Authenticator, which works like the BankID-tags many people use to log into their online bank. In this case, verification by email was chosen. I will get back to why in the Research part of the report.

---

<sup>41</sup>C# is a general purpose, multi-paradigm programming language created by Microsoft.

<sup>42</sup>OAuth2 is a industry standard protocol for authorization.

## 4.2.2 Research

When I got the assignment, I was not sure whether I would be able to successfully implement a working 2FA system. Due to this research and implementation happened somewhat alongside each other. However, the choice of which technology to use had to be made first. Other than management suggesting a library called ASPSMS<sup>43</sup>, I was given free reins. Due to my limited experience with C# and back-end coding, I decided to pursue a solution built with Javascript in the front-end application.

In a previous project I have worked on in the IBE205 Agile Methods course, I used Google Firebase to create a login system for a web page. VBOnline was already using some Firebase functionality for other parts of the application, which coincided nicely with my experience using the tools.

Firebase is an app development platform provided by Google. It offers several services like authentication, cloud database, hosting and more. One of the authentication methods offered is Email Authentication, which is password-less. This means it only requires the user to have access to their email to authenticate and made it an excellent fit for my solution as password authentication already existed in VBOnline.

The majority of research time spent went to reading and understanding the VBOnline codebase. My method for this was to use the inspection tool<sup>44</sup> to find which file or function I had to look into. I then used VSCode's search tool to find the exact location in the codebase. Over time, I gained a decent overview of how VBOnline is put together, letting me place my code at the correct places for the functionality I wanted.

To learn Firebase, I used the Firebase Docs[9] extensively. It provides boilerplate code and suggestions on how to use it. The docs are written for modern web pages and web frameworks, so I had to do some tweaking here and there to make the boilerplate code work with the older Angular 1.6 of which VBOnline is built with.

---

<sup>43</sup>ASPSMS is a C# library and messaging service for ASP.NET systems.

<sup>44</sup>The inspection tool is a browser developer tool. When activated, the user can click a element on a webpage and the tool will show the corresponding code.

### 4.2.3 Implementation

#### Prototype

Figure 15 shows a web form titled "Firebase Login Demo". At the top is a dropdown menu with "Email link" selected. Below the dropdown are two text input fields: "Email address" and "Password". At the bottom of the form are two buttons: a blue "SIGN IN" button and an orange "SIGN UP" button.

Figure 15: Prototype

To get started I was asked to create a prototype demonstrating the use of the Firebase tools. As the only purpose was to show I was able to make it work, I created a simple web page without the use of Angular. In the dropdown menu you can choose between email link authentication or regular username/password authentication. I chose to also implement the latter to refresh my knowledge of the tools. I did a short presentation of it to management and was given a go to continue implementation.

#### Flowchart

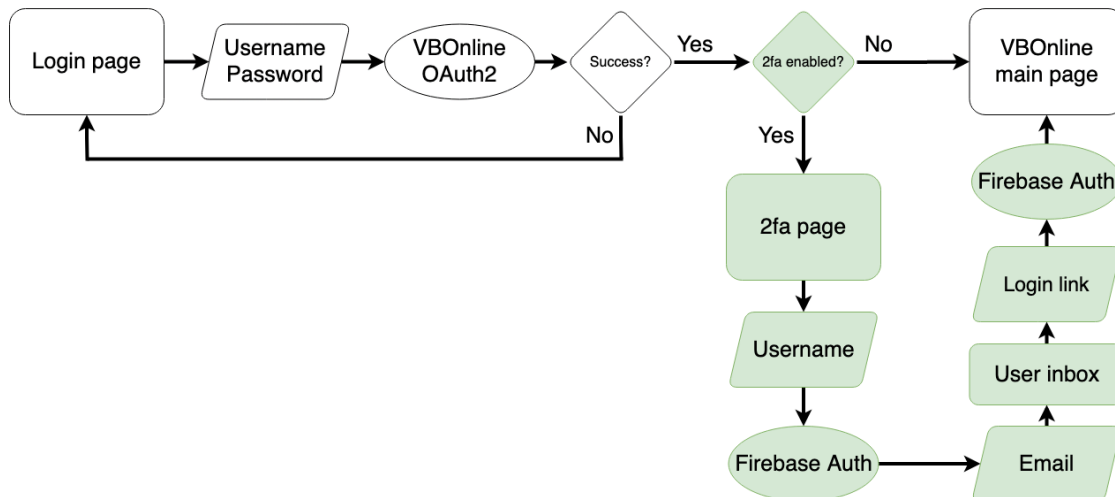


Figure 16: Flowchart of the login system

The flowchart in figure 16 shows the flow of the login procedure, with my implementation highlighted in green. During development I tried different approaches to some of the steps, but the overall flow stayed the same throughout.

## Setting up Firebase

```
var config = {
  apiKey: "AIzaSyD08v40AwEZ7Gc2J3GVosGMKXwR600c8B0",
  authDomain: "vbonline-87855.firebaseio.com",
  databaseURL: "https://vbonline-87855.firebaseio.com",
  projectId: "vbonline-87855",
  storageBucket: "vbonline-87855.appspot.com",
  messagingSenderId: "543591023862"
};
firebase.initializeApp(config);
```

**Codeblock 16:** Firebase configuration.

The Firebase module is loaded to the site with a script-tag on the parent HTML-page of the application. It is then initialized with an object containing configuration information as seen in codeblock 16. The API-key is safe to show as it only serves as an address to the correct Firebase account. After initialization, the Firebase tools can be used throughout the application.

## Adding users to Firebase Authentication database

```
def NewUser(email,password):
    details = {
        "email": email,
        "password": password,
        "returnSecureToken" : True
    }

    r = requests.post(url+"?key={}".format(apikey),data=details)
```

**Codeblock 17:** Register user to Firebase Authentication database.

For any authentication in Firebase the user needs to exist in the Firebase Authentication database. VBOOnline has about 1600 accounts which needed to be added to this database. Doing it manually was unfeasible due to the sheer amount of time it would take, so I wrote a Python<sup>45</sup> script that automated the registering process.

To register a user in Firebase you need an email and a password. To get the emails I exported all users from VBOOnline to a .csv<sup>46</sup> file. I then extracted the emails from it and added them to a array. The passwords were created with a simple password-generator from an article on Geekflare[19] which generates a string of a set size composed of random characters. The script loops through the email array and for each item it makes a post request to the Firebase API with a payload containing an email and a generated 8-character password.

To prevent misuse, Firebase has a limit of 1000 new users per hour. To deal with this limit I split the email array into 3 batches and ran the script three times one hour apart. I later discovered there are ways to batch-register new users with the Firebase SDK<sup>47</sup>. Good to know if I were to face a similar task in the future.

## Enabling 2FA for a user

Every account in VBOOnline has a list of claims linked to it. Claims are flags for giving access to different parts of the application. One of the claims, called SalesContractAccess, was no longer in use and I was asked to use it to flag a user for 2FA. Management also requested the claims VBOALLE and VBOSYSADM to be added to the check. These claims can be activated by a user with admin privileges in the VBOOnline User Administration page.

---

<sup>45</sup>Python is a high-level, interpreted, general-purpose programming language.

<sup>46</sup>CSV stands for Comma Separated Values and is a common way to store data in a table structured format.

<sup>47</sup>SDK: Software Development Kit

## Redirecting user to the 2FA page

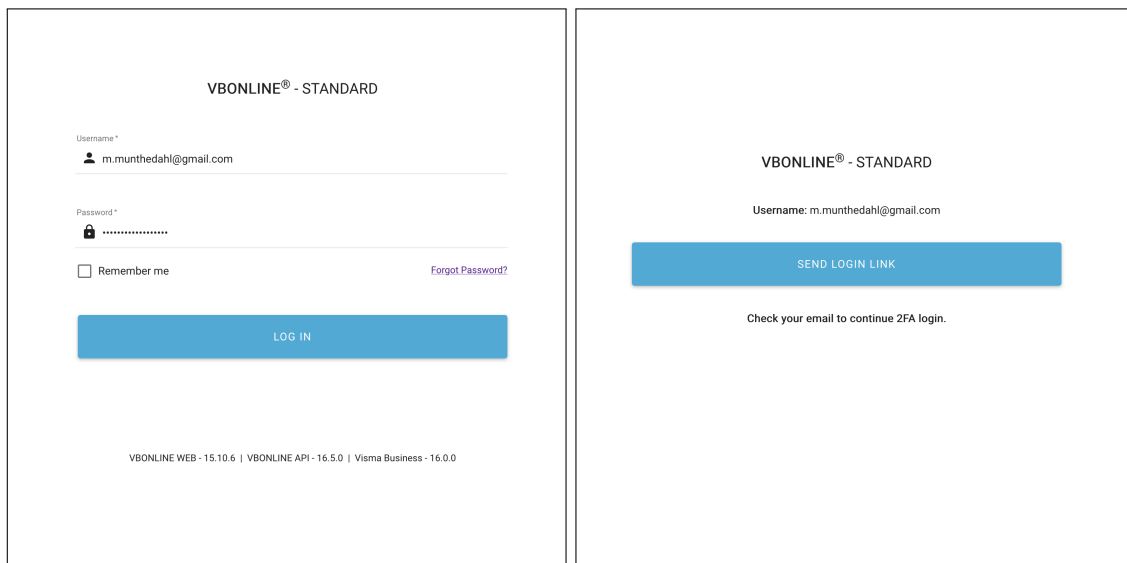


Figure 17: VBOonline login and 2FA page

Shown in figure 17 is the login and 2FA pages of VBOonline. To keep the styling consistent, I made the 2FA page by copying the login page and making some small modifications to fit the 2FA functionality.

```
angular.forEach(response, function(item, key) {
  if(item.ClaimValue == 'VBOALLE' || item.ClaimValue == 'VBOSYSADM') {
    TwoFaEnableCheck1 = true;
  }
  if(item.ClaimType == 'SalesContractAccess' && item.ClaimValue == 'true'){
    TwoFaEnableCheck2 = true;
  }
});

if (TwoFaEnableCheck1 && TwoFaEnableCheck2){
  $scope.use2FA = true;
};
```

Codeblock 18: Check for enabling 2FA.

When a user logs in, the API returns the login token as well as the list of claims which is then stored in localStorage<sup>48</sup>. During the login procedure the list is looped through using an `angular.forEach()`-function and the necessary checks are being made. If both checks pass, the `use2FA`-flag is set to true.

```
if ($scope.use2FA){
  let url = app.urls.TWOFACOR_LOGIN
  console.info("Change location to " + url);
  window.location.assign(url);
}
```

Codeblock 19: Check for redirecting to 2FA page.

If the `use2FA`-flag is set to true, the user is redirected to the 2FA page by the `window.location.assign()`-function.

<sup>48</sup>localStorage is a way of storing data in the browser. The whole application can access this data.

## Mobile page

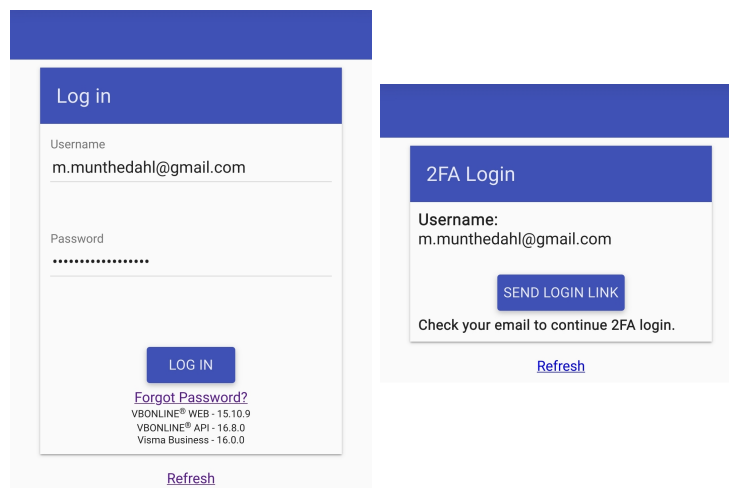


Figure 18: VBOOnline mobile login and 2FA page

VBOOnline has a mobile UI (figure 18) that is activated when the window is smaller than 1280px wide. I created the mobile 2FA page in the same way as the desktop version.

## Sending the email

```
$scope.sendTwoFaEmail = function(isMobileView) {
  let email = window.localStorage.getItem('username');
  document.getElementById("loader").style.display = "";
  let url = window.location.origin + window.location.pathname + "#/";

  if (isMobileView) {
    url = url + 'shortcuts';
  }

  let actionCodeSettings = {
    url: url,
    handleCodeInApp: true,
  }

  auth.sendSignInLinkToEmail(email, actionCodeSettings)
    .then(() => {
      document.getElementById("loader").style.display = "none";
      document.getElementById("sentText").style.display = ""
      document.getElementById("sentTextMobile").style.display = ""
    })
    .catch((error) => {
      console.info("Code: "+error.code+" Message: "+error.message);
    });
};
```

Codeblock 20: Email sender function.

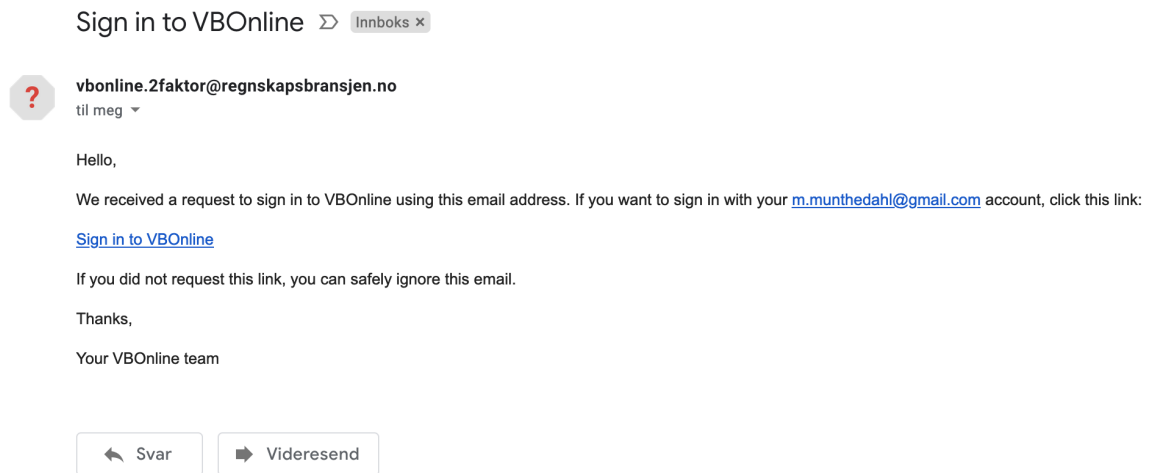
Codeblock 20 shows the email sender function, which is called when the user clicks the Send Login Link button. The function picks up the email from the `username` item in `localStorage`<sup>49</sup> and displays a spinning loader icon to the user. It then creates the destination URL from the window origin, which is the website URL, and pathname which will be the name of the instance of the application being logged into. For example, `OBIDev` for the testing instance we used during development.

<sup>49</sup>In VBOOnline, every user account has the user's email as it's username.



If the mobile UI is active, "shortcuts" is appended to the URL to direct the user to an alternative main page with mobile friendly shortcuts. The URL is then placed in a object which is passed to the `sendSignInLinkToEmail()`-function together with the email. This function is the Firebase Auth function for sending emails and I learned how to use it at the Authenticate with Firebase using email link[10] section of Firebase Docs. If the function call is successful, the loader is disabled and a small text notifying the user to check their email is displayed.

By default, the email is sent from Firebase's own SMTP<sup>50</sup> server and I kept it like that during development.



**Figure 19:** 2FA email

Figure 19 shows the email sent to the user. To prevent misuse, the body of the email is predefined by Firebase and can not be changed. The link is shown in figure 20, split up for readability. It is generated by Firebase and composed of the following URL variables. API-key, mode, a token and the target domain address.

- `https://vbonline-87855.firebaseio.com/__/auth/action`
- `?apiKey=AIzaSyD08v40AwEZ7Gc2J3GVosGMKXwR600c8B0`
- `&mode=signIn`
- `&oobCode=ZR2t2vYdntRYcRE2JhGe0MCuiUaZNsxtlMaD_6AH2noAAAGA-skfVQ`
- `&continueUrl=https://vbonline.no/OBIDev/%23/&lang=en`

**Figure 20:** 2FA link URL

---

<sup>50</sup>SMTP is an internet standard communication protocol for electronic mail transmission.

## Confirming the redirect form email link

```
if ($scope.use2FA() && $scope.fbAuth == "false"){
  let email = window.localStorage.getItem('username');
  firebase.auth().signInWithEmailLink(email, window.location.href)
    .then((result) => {
      console.info("Email link is valid.")
      localStorage.setItem("fbAuth", "true")
      $scope.userLoggedIn = true;
      $scope.init();
    })
    .catch((error) => {
      console.info("Code: "+error.code+" Message: "+error.message);
      console.info("Email link is invalid. Logging out.")
      $scope.userLoggedIn = false;
      window.location.assign(app.urls.LOGIN_PAGE);
    });
} else {
  $scope.userLoggedIn = true;
  $scope.init();
}
```

**Codeblock 21:** Email link validation.

During loading of the VBOnline main page the same claims check that was done in the login procedure is made to see if the user has 2FA enabled. If true, the code shown in codeblock 21 attempts to confirm that the user accessed the page by following the login link from the email. This is done by the Firebase Auth function `signInWithEmailLink()`. It takes in the email from `localStorage` and the URL of the webpage by referring to the `href` property of the DOM window object. If the page is accessed from the email the URL should be in the format shown in figure 20. If the function returns a valid response, loading of the page continues and the user is now logged in. If not, the user is returned to the login page and has to start over.

### Known issues

As the email of the user is stored in `localStorage`, it is possible to hijack the login email by utilizing the developer console and debugger of the browser to change the email. This does require both knowledge of where in the code to look as well as actually getting past the regular username/password authentication. Management was briefed on this and I was advised to leave it be and focus on the rest of the assignment.

After deploying the solution to customers for testing it was discovered that the login email sometimes got caught by their spamfilters. It was hard to tell why, so I attempted to remedy it by changing the SMTP server. After testing with several servers used by ØB Innovation, one was found that seemed to not trigger the spamfilters. At the time of writing this solved the issue. However, there were some inconsistency in the feedback I got on whether it worked for everyone, so I choose to leave it in as a known issue.

### 4.2.4 Testing

Testing was done continuously throughout the whole development process. Whenever a new step of the process was completed, it was deployed to the OBIDev testing environment and tested by both myself and management. In the later phases of the development, some customers where also allowed to test it. The functionality to be tested was always related to sending the email or validating it, and most of my testing consisted of stepping through the login process and monitoring each step with the browser debugger tool. I also created additional VBOnline accounts with different emails so I could quickly switch between an account with 2FA activated and one without. Management and customers were testing by attempting to log in with different accounts and privileges.

#### **4.2.5 Conclusion**

Completing this assignment was a slightly bumpy, but successful ride. My solution covered the requirements given by management and has since been deployed to several customers of VBOnline. Making the Firebase tools work by themselves went fairly straightforward and figuring out how to implement them into an existing codebase was challenging, but rewarding. My design philosophy was mainly dictated by how VBOnline already worked and not wanting to make it more complicated than necessary. I tried to keep the coding style consistent with what was already there as well as not making any big changes to the existing logic.

If I were to start over again today, I would put in the extra effort to learn C# and build the system on the back-end. While I'm happy with how my solution turned out given the boundaries of the assignment, I've learned that a authentication system is better served as a self contained module. Building one with different technologies placed in different parts of the stack makes it more difficult to maintain and have felt somewhat hacky.

#### **4.3 Other tasks**

These were very minor issues that didn't require much research. I solved them in between main assignment sessions during the start of the internship.

##### **4.3.1 Issue #311 - Remove generation of externalOrderId**

The variable `ExternalOrderId` sometimes interfered with other functions in VBOnline. Fixed by disabling generation of the variable in functions where it was interfering.

##### **4.3.2 Issue #295 - Removal of check for disabling dropdown menu**

A customer asked for the `AnsattNr` dropdown menu to be available to use for all users and not just users with all privileges. Completed by removing the privilege-check in their installation of VBOnline.

##### **4.3.3 Issue #282 - Update license agreement URL on login-page**

The license agreement for VBOnline is presented as a pdf-file and was updated. The link on the login page was changed to reflect this. An infotext urging users to use Google Chrome as their browser when using VBOnline was also added to the login page.

## 5 Report Conclusion

We are both very pleased at the outcome of our decision to do an internship this semester. It built our confidence that we can be useful in a team and that there is a strong path forward for both of us doing what we want to do.

We gained experience working in an organization while putting to use some of the theory taught to us about agile methodology<sup>51</sup>. Being familiar with the concepts of modern project management let us both adapt to the organizational side of things with ease. Coming in with an understanding that perfect is the enemy of good, and that one of the most important factors to retaining velocity<sup>52</sup> in a project is modest goals with regular course corrections and room for feedback.

We also learned about working with legacy code<sup>53</sup>, which was a challenge at first, but gratifying to overcome. Having some familiarity with the concepts of web development<sup>54</sup> from before helped us progress to more advanced concepts. Previously, during our studies, we hadn't had to understand, fix and add to legacy code. It built a lot of confidence in us that we were able to figure these things out. As a result, we feel better prepared for joining teams working on existing projects.

While a lot of the lessons we've learned have built on existing learning from the school, the internship exposed us to some of the effects of regularly going to an office and being around others who are working towards a common goal. We've felt the motivation it gives to want to be a team player and contribute. While there have been plenty of group projects throughout our study, we both felt there is always the problem of mismatched motivations and contributions. Getting to be a part of a small team that all did their best to pull in the same direction inspired us to do the same. While this was a good and enjoyable experience for us, it brings us to a potential trap. As we were both introduced to a new environment where we got to enjoy success and progress that was entirely detached from our existing obligations to other school subjects, a disproportionate amount of time often ended up going into what we enjoyed most. We managed by agreeing to work together whenever possible for the other courses in the semester, which helped us stay on top of deadlines and mandatory assignments.

To us, having the chance to leave the classroom and work on systems that are in use in the real world proved invaluable. We got to use our existing knowledge, learn what we excel at and be told what we needed to do better. Additionally it afforded us opportunities to network that we would not have gotten otherwise. The internship will likely have an impact on the rest of our lives, as we have made friends and gotten jobs through it. We can only encourage anyone who doesn't intend to become researchers within academia to consider the opportunity that this presents.

---

<sup>51</sup>From IBE205 Agile Methods

<sup>52</sup>Velocity[1] is a term used around agile methods and it measures rate of progress.

<sup>53</sup>A large amount of preexisting source code, often written in a way that no longer matches current practices due to advances in the field.

<sup>54</sup>From IBE102 Webutvikling

## 6 References

- [1] Agile Alliance. What is Velocity in Agile?, 2022. URL <https://www.agilealliance.org/glossary/velocity>.
- [2] Simon Bengtsson and open source contributors. jsPDF-AutoTable documentation page, 2022. URL <https://simonbengtsson.github.io/jsPDF-AutoTable/>.
- [3] Simon Bengtsson and open source contributors. jsPDF-AutoTable GitHub page, 2022. URL <https://github.com/simonbengtsson/jsPDF-AutoTable>.
- [4] Kolade Chris. freeCodeCamp Rubber Duck Debugging page, 2022. URL <https://www.freecodecamp.org/news/rubber-duck-debugging/>.
- [5] Josh Close and other contributors. CsvHelper page, 2022. URL <https://joshclose.github.io/CsvHelper/>.
- [6] Google. AngularJS controller page, 2022. URL <https://docs.angularjs.org/guide/controller>.
- [7] Google. AngularJS directive page, 2022. URL <https://docs.angularjs.org/guide/directive>.
- [8] Google. AngularJS services page, 2022. URL <https://docs.angularjs.org/guide/services>.
- [9] Google. Firebase Docs page, 2022. URL <https://firebase.google.com/docs>.
- [10] Google. Firebase authenticate with firebase using email link in javascript, 2022. URL <https://firebase.google.com/docs/auth/web/email-link-auth>.
- [11] James "MrRio" Hall. jsPDF documentation page, 2022. URL <https://rawgit.com/MrRio/jsPDF/master/docs/index.html>.
- [12] James "MrRio" Hall. jsPDF GitHub page, 2022. URL <https://github.com/parallax/jsPDF>.
- [13] ØB Innovation. VBOnline Swagger page, 2022. URL <https://vbi.okonomibistand.no/VBInterface0BIDev/swagger/ui/index#/>.
- [14] Alexey Zimarev John Sheehan, Andrew Young and RestSharp community. RestSharp page, 2022. URL <https://restsharp.dev/>.
- [15] Mozilla. MDN article page for asynchronous JavaScript, 2022. URL <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Introducing>.
- [16] Mozilla. MDN article page for HTTP POST requests, 2022. URL <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST>.
- [17] Samuel Oloruntoba. Digital Ocean SOLID page, 2022. URL [https://www.digitalocean.com/community/conceptual\\_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design](https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design).
- [18] W3 Schools. MVC W3Schools page, 2022. URL <https://www.w3schools.blog/mvc-in-angularjs>.
- [19] Hafeezul Kareem Shaik. How to create a random password generator in python, 2022. URL <https://geekflare.com/password-generator-python-code/>.
- [20] Niklas von Herten. html2canvas GitHub page, 2022. URL <https://github.com/niklasvh/html2canvas>.

## List of codeblocks

1	Code to get token for intern test . . . . .	3
2	Code to get customer data for internship test . . . . .	3
3	Code to get and display token and customer in application . . . . .	4
4	The complete directive for exporting the results table to a PDF file . . . . .	12
5	The getCurrentTime() function . . . . .	15
6	Function to return a column's index based on its name. . . . .	16
7	Function to create a list of column indices based on names input. . . . .	16
8	Function to remove columns. . . . .	17
9	Column alignment function. . . . .	17
10	Table adding function. . . . .	18
11	Page number function. . . . .	18
12	String sanitization utility function. . . . .	21
13	Function to remove BtoA()-incompatible characters. . . . .	23
14	isCreditNote() function. . . . .	24
15	Example of deserializing a JSON response. . . . .	25
16	Firebase configuration. . . . .	31
17	Register user to Firebase Authentication database. . . . .	31
18	Check for enabling 2FA. . . . .	32
19	Check for redirecting to 2FA page. . . . .	32
20	Email sender function. . . . .	33
21	Email link validation. . . . .	35

## **End of report**

This page marks the end of our report for IBE600. Any following pages will contain logs, source code or other artifacts we were able to include at the end.

Thank you for reading.

# Attachment 1

## Kontrakt om utplassering for IT-studenter ved Høgskolen i Molde

IBE600 Bacheloroppgave for IT-studenter utgjør 15 studiepoeng, som er halv semesterbelastning.

Den kan utføres som en utplassering i bedrift, da med halv arbeidstidsbelastning, for eksempel med 2 eller 3 fulle arbeidsdager i uka, eller 5 dager med halv arbeidsdag.

Emnet tas vanligvis på våren i siste (sjette) semester av bachelorutdanningen. Utplasseringen starter ved semesterstart i januar og går ut mai måned.

Studenten skal i utplasseringen a) få erfaring fra arbeidslivet, b) prosjekterfaring, c) øvelse i å skrive og fremlegge en sluttrapport.

Bedriften (eller organisasjonen) skal være en aktør i IT-bransjen, ha flere ansatte, kontorlokaler og tilby fast kontorplass til studenten i utplasseringsperioden.

Studenten skal ha to veiledere, en fra bedriften og en fra høgskolen. Disse signerer kontrakten som inngås i forkant av utplasseringen. Denne skal vedlegges sluttrapport.

Etter utplasseringen er ferdig skal bedriften signere en bekreftelse på at utplasseringen er utført som beskrevet. Denne skal legges ved sluttrapport.

Sluttrapporten leveres digitalt til Høgskolens eksamenssystem, teller 100 % og gis en bokstavkarakter.

Utplasseringen skal inneholde omtrent like mye av

- Daglige gjøremål innen digitalisering, eksempelvis drift, vedlikehold, bruker- og kundestøtte, eksperimentering, igangsetting, testing, feilsøking, programmering, dokumentasjon og innhenting av informasjon og utredning
- Et spesifikt og relevant prosjekt i bedriften, som man blir enige om i de første ukene av utplasseringen med tilhørende prosjektplan

Arbeidstiden må koordineres slik at studenten kan delta i undervisning, eksamen og forberedelser.

Dersom bedriften ønsker å omplassere studentene i løpet av utplasseringen, må dette skje i samtykke med student og Høgskole. Bedrift dekker de merkostnader dette forårsaker.

Når det gjelder krav til arbeidsinnhold og skaderisiko, skal ta hensyn til at studenten er under opplæring og utdanning.

Studenten kan ikke motta lønn fra bedriften i utplasseringsperioden.

Bedriften har ikke arbeidsgiveransvar for den utplasserte. Studenten mottar ikke lønn.

Studentene har normal taushetsplikt vedrørende bedrift, jmfør universitets- og høyskoleloven §4-6. Egen taushetserklæring er med i standard rapportmal ved Høgskolen, og må signeres hvis ønsket.

Ved å signere aksepterer man de forhold som er beskrevet over.

Kontrakt er inngått mellom Høgskolen i Molde, studentene **Magnus Munthe-Dahl** og **Viktor Engvall**, og bedrift **ØB Solutions AS** i Molde

Utplasseringsperiode: fra **1.1.2022** til **31.05.2022**


Arbeidssted: Sandv 9, 6413 Molde.



Bedriftens veileder og kontaktinfo: **Egil Stavik Tautra**. Denne har ansvar for å sette seg inn i ordningen med utplassering, og veilede studentene i sitt arbeide samt å rapportere særlige avvik fra ordningen til Høgskolen i Molde.

Høgskolens veileder: **Ketil Danielsen**. Denne har ansvar for å veilede studentene i arbeidet med rapporten og følge opp bedriftens rapportering.

Sted/dato: Molde 2.jan 2022

  
Viktor N Engvall

Studenter

  
For Bedrift

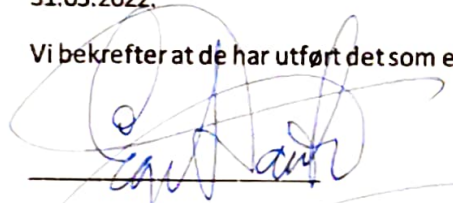
  
For Høgskolen i Molde

## Bekreftelse på gjennomført utplassering

(vedlegg inn i sluttrapporten)

Dette er en bekreftelse fra bedriften ØB Solutions as ved veileder Egil Stavik Tautra på at studentene Magnus Munthe-Dahl og Viktor Engvall har vært utplassert i vår bedrift i perioden 1.1.2022 til 31.05.2022.

Vi bekrefter at de har utført det som er beskrevet i sluttrapporten.

  
Molde 31.mai 2022

Sted og dato. (signert av veileder/bedriftsrepresentant)

## Attachment 2

# Utplassering ØB Innovation - Logg

Magnus Munthe-Dahl

## Tirsdag 18.Jan

Peppol problemstilling

- XML dokument mangler cac/cbc namespace

Git og setup av vscode

2factor

- Design
  - Hovedlogin side skal fungere som vanlig. Brukere med 2faktor pålogging blir tatt til en annen side før de får tilgang til appen.

## Torsdag 20.Jan

2factor

- Videre arbeid med å få oversikt over hvordan VBOnline er satt sammen.
- Har fått tilgang til Firebase Console
- Database for brukere
  - Root -> Users -> ØB Innovation AS

Zoom-møte med idrettslag

- Presentasjon av VBOnline v/ Egil

## Tirsdag 25.Jan

### 2factor

- Gjennomgang med Egil, Stian og Frode
- Plan lagt for videre arbeid
  - Lage stand-alone prototyp for innlogging mot Firebase

### Nettleser tekst

- "Use Google Chrome browser at PC and MAC." skal vises ved siden av lisensavtale på login side.
  - Også lagt til i translate liste
- Første commit og merge med master

### PDF

- Gjennomgang med Egil og Stian
- Inspisering av kildekode for å finne variablene som blir brukt i Purchase -> History
  - Transactions

## Torsdag 28.Jan

### 2factor

- Arbeid med å lage prototype med vanilla js
  - Prototype for innlogging med passord ferdig
  - Viste frem demo for Egil og Frode

## Tirsdag 01.Feb

### 2factor

- Videre arbeid med prototyp
- Email link sendes riktig
  - Kan ikke redirekte url til localhost
    - Solved -> måtte bruke http uten s
- Phone login lagt til
  - Evig warnings - funker ikke
  - Beskjed fra Egil: trenger ikke phone login
- Opprettet github repo for 2factor kode

### Planlegging

- Opprettet Kanban side for 2factor prosjekt

## Torsdag 03.Feb

### 2factor

- Demo av prototype vist fram
- Plan for videre implementasjon lagt

## Tirsdag 08.Feb

### 2factor

- Arbeid med å finne ut hvordan identifisere sysadm bruker
  - Må sendes fra API
  - Foreløpig plan -> lage dummy funksjon

## Torsdag 10.Feb

### 2factor

- 2faktor løsning fungerer halveis
- Opprettet egen controller og view for 2faktor
- Lagt til i routing
- Firebase oppdatert til versjon 8.2.4
- Hacky løsning på redirect fra email link
  - Link blir bekreftet, men man er bare soft-locked ut av vbonline

## Lørdag 12. Feb

### 2factor

- 1900 brukere lagt til i firebase auth database via python script
  - Autogenererte passord for sikkerhet

## Tirsdag 15. Feb

### 2factor

- Lagt til forskjellig infotekst på 2faktor-bilde og User Admin side

## Torsdag 17. Feb

### 2factor

- Bugfix url redirect
- Begynt på powerpoint presentasjon av 2faktor

### admin

- Opprettet og registrert VBO brukere for seminar deltakere

## Tirsdag 22. Feb - Torsdag 24. Feb

Syk

## Tirsdag 01. Mar

### 2factor

- Arbeid med standalone vbonline.no/twofa

## Torsdag 03. Mar

### 2factor

- Arbeid med standalone vbonline.no/twofa

## Tirsdag 08. Mar

### 2factor

- Arbeid med standalone vbonline.no/twofa
- Refresh bug fixed!
- 2FA funker nå for mobile view

## Torsdag 10. Mar

### 2factor

- Arbeid med email sjekk

## Tirsdag 15. Mar

### 2factor

- Arbeid med email sjekk
- Email sjekk funker!

## Torsdag 17. Mar

### 2factor

- Forsøk på edge-case bugfixing av email-sjekk

## Tirsdag 22. Mar

### 2factor

- Revert tilbake til tidligere løsning. Alt ser ut til å fungere lokalt.

## Torsdag 24. Mar

### 2factor

- Testing av mobillogin. Alt fungerer
- Assignment complete!

## Tirsdag 29. Mar

### 2factor

- Byttet email server til ØB's egen smtp server. Alle får nå login link.

## Torsdag 31. Mar

### Presentasjon

- Presentasjon av 2faktor for Konsulentsjef ØB Nic

### Rapport

- Lagt inn issues jeg har gjennomført under Other tasks

## Attachment 3

# Code - Magnus

## twofactorController.js - Whole file

```
1  'use strict';
2
3  app.controller('twofactorController', ['$scope', '$location', '$window', '$translate', '$rootScope', function($scope) {
4    console.log('Start of twofactorController');
5
6    const auth = firebase.auth();
7
8    $scope.getEmail = function(){
9      return window.localStorage.getItem('username');
10   }
11
12   // Two Factor prototype email sender
13   $scope.sendTwoFaEmail = function(isMobileView) {
14     let email = window.localStorage.getItem('username');
15     document.getElementById("loader").style.display = "";
16     let url = window.location.origin + window.location.pathname + "#/";
17
18     if (isMobileView) {
19       url = url + 'shortcuts';
20     }
21
22     let actionCodeSettings = {
23       url: url,
24       handleCodeInApp: true,
25     }
26
27     auth.sendSignInLinkToEmail(email, actionCodeSettings)
28       .then(() => {
29         console.info("Email sent to "+email)
30         console.info("Link: "+ url)
31         document.getElementById("loader").style.display = "none";
32         document.getElementById("sentText").style.display = ""
33         document.getElementById("sentTextMobile").style.display = ""
34       })
35       .catch((error) => {
36         console.info("Code: "+error.code+" Message: "+error.message);
37       });
38   };
39
40   console.log('End of twofactorController');
41
42 }]);
```

## loginController.js - use2FA check

```
142     var TwoFaEnableCheck1 = false;
143     var TwoFaEnableCheck2 = false;
144
145     if(isMobileView && response.length > 0) {
146         sendUserGUID(response[0].UserId);
147     }
148
149     angular.forEach(response, function(item, key) {
150         console.log(item);
151
152         if(item.ClaimType.toUpperCase() == 'VBCLIENT') {
153             isVBClientUser = true;
154         }
155         // Check for 2factor
156         if(item.ClaimValue == 'VBOALLE' || item.ClaimValue == 'VBOSYSADM') {
157             TwoFaEnableCheck1 = true;
158         }
159         if(item.ClaimType == 'SalesContractAccess' && item.ClaimValue == 'true'){
160             TwoFaEnableCheck2 = true;
161         }
162     });
163
164     //Enable 2factor if checks pass
165     if (TwoFaEnableCheck1 && TwoFaEnableCheck2 && window.location.hostname != "localhost"){
166         $scope.use2FA = true;
167     }
};
```

## loginController.js - 2FA redirect

```
189     if ($scope.use2FA){
190         let url = app.urls.TWOFACOR_LOGIN
191         console.info("Change location to " + url);
192         window.location.assign(url);
193     }
194     else {
195         if(isMobileView) {
196             $scope.$emit('userLogIn');
197             window.location.assign("#/shortcuts");
198         }
};
```

## index.html - Main Controller - use2FA check

```
2595     $scope.use2FA = function(){
2596         let ret = false;
2597         let claims = JSON.parse(localStorage.getItem('claims'));
2598         let roleCheck = false;
2599         let SalesContractAccessCheck = false;
2600
2601         angular.forEach(claims, function(item,key){
2602             if(item.ClaimValue == 'VBOALLE' || item.ClaimValue == 'VBOSYSADM') {
2603                 roleCheck = true;
2604             }
2605             if(item.ClaimType == 'SalesContractAccess' && item.ClaimValue == 'true'){
2606                 SalesContractAccessCheck = true;
2607             }
2608         })
2609         if (roleCheck && SalesContractAccessCheck && window.location.hostname != "localhost"){
2610             ret = true;
2611         }
2612         return ret;
2613     }
};
```



## index.html - Main controller - 2FA link verification

```
2639 // Check if email link is valid if 2FA is enabled and firebase user has not yet been authenticated.
2640 if ($scope.use2FA() && $scope.fbAuth == "false"){
2641     let email = window.localStorage.getItem('username');
2642     firebase.auth().signInWithEmailLink(email, window.location.href)
2643         .then((result) => {
2644             console.info("Email link is valid.")
2645             localStorage.setItem("fbAuth", "true")
2646             $scope.userLoggedIn = true;
2647             $scope.init();
2648         })
2649         .catch((error) => {
2650             console.info("Code: "+error.code+" Message: "+error.message);
2651             console.info("Email link is invalid. Logging out.")
2652             $scope.userLoggedIn = false;
2653             window.location.assign(app.urls.LOGIN_PAGE);
2654         });
2655     } else {
2656         $scope.userLoggedIn = true;
2657         $scope.init();
2658     }
2659 }
2660 else {
2661     $scope.userLoggedIn = false;
2662     console.info("Token not found.");
2663 }
2664 }
```

## twofa.html - whole file

```
1 <!-- NEW GUI - for WEB view only -->
2 <div class="container" ng-controller="twofactorController" ng-cloak layout="column" hide-xs>
3
4 <div layout="row" layout-align="center center">
5
6 <div layout="row" style="width: 50%;padding: 20px;" layout-align="center center">
7
8 <div style="width: 70%;" >
9
10 <div layout="row" layout-align="center center">
11
12 </div>
13
14 <div layout="row" layout-align="center center">
15 <span layout="row" layout-align="center center" style="font-size: 22px;margin-bottom: 3rem;display: block;">
16 <strong>VBONLINE</strong><sup></sup> - {{'LOGIN_SUB_TITLE' | translate}}
17 </span>
18 </div>
19
20 <form class="md-form" ng-submit="submit()" layout="column">
21
22 <div layout="row" layout-align="center center">
23 <span style="font-weight:bold;font-size: 16px;">{{'LOGIN_USERNAME' | translate}}:</span>
24 <span>&nbsp;&nbsp;&nbsp;{{getEmail()}}</span>
25 </div>
26
27 <md-button class="md-raised md-primary" ng-click="sendTwoFaEmail(false)" style="width: 100%;
28 background-color: #4E885D;
29 color: #fff;
30 font-size: 16px;
31 font-weight: 400;
32 letter-spacing: 1.1px;
33 text-transform: uppercase;
34 padding: 14px 20px;
35 margin: 2.4em 0;
36 border: none;
37 border-radius: 4px;
38 cursor: pointer;" >
39 {{'TWOFA_SEND_EMAIL' | translate}}
40 </md-button>
```



## Attachment 4

### ØB Internship Log - Viktor

#### 18/01/2022 - Onboarding

Original plan was to get a walkthrough in work tools. Turns out there was a bit of an ongoing crisis with an integration and they were busy. Spent a bit of time talking about that and getting a walkthrough in the system as it stands. I did not understand all of it, but it was an educational experience.

I was assigned a bug fixing case to investigate and resolve.

Then we had a walkthrough on github and IIS Express.

#### 20/01/2022

Worked on assigned bug and think I have arrived at a conclusion. The btoa function fails because it is being fed an unsupported character, an extra-long hyphen “-“. The current solution is to replace it with a regular hyphen.

This was a simple fix but getting to grips with IIS express and making change visible was more difficult because the browser I was using was caching the code from previous attempts. Looked into ways to solve it and it was as simple as clearing the cache of the browser using Ctrl+F5.

The day was largely spent familiarizing myself with Chrome web dev tools using break points in source documents and inputting dummy test data in JS variables.

#### 25/01/2022

I wrote a couple functions over the weekend to clean up the .csv export output of the sales history page. It might not be used, but it was good training for string-handling in my PDF project. A lot of the same concepts will probably be reused and it was an opportunity to think about data structure and how to segment text up to be easier to program against.

I have decided that to accomplish printing the PDF in a satisfactory way I should assemble an MVP (minimum viable product). Preliminarily I think I should proceed about that in two ways:

1. Get to grips with the concepts of Angular JS and what makes it different from normal JS. Perhaps gain insight into the site controllers for VBOOnline in case I must hook my code into that to retrieve correct data more easily for table construction.
  - a. Most importantly try to master the ng-repeat function for tables in Angular because that is what the purchase controller uses to list the retrieved data. It seems to take the data from a fetch in an array/list and iterates through it using a foreach like structure, adding a row for each item on the list.
2. Code a simple table into jsPDF pdf constructor with download link formatted in the intended format of the table on VBOOnline's reports page. This will be a work of reverse engineering, but I might be able to use the data format of the existing CSV export to extrapolate the fields I need.
  - a. Did find references to a Mozilla-owned and open-source PDF library in existing source code. Might investigate it and try to create examples using both it and jsPDF. Determine which is better for my purposes.

In addition to these, we had a workplace meeting discussing how to best implement Magnus' two-factor authentication scheme. What data might be needed and how he might be able to solve it.

We also did a deep dive into the JS controller and found out how VBOnline constructs its tables. It uses an object called transactions and pipes the object properties into an Angular table builder.

## 27/01/2022

Spent the day working on a way to generate tables with corresponding content for testing of automatic generation of PDFs.

Some hurdles have been met, but progress has also been made. The function that lets me save tables to PDF has now been implemented and I am free to experiment with various style and sizing elements to get the desired result.

If the desired effect can't be had with jsPDF I will have to branch out and try to use a different library. The nuclear solution will be an html2canvas-generated image of the table put into the pdf file, but I don't want to resort to the laziest solution as the first solution.

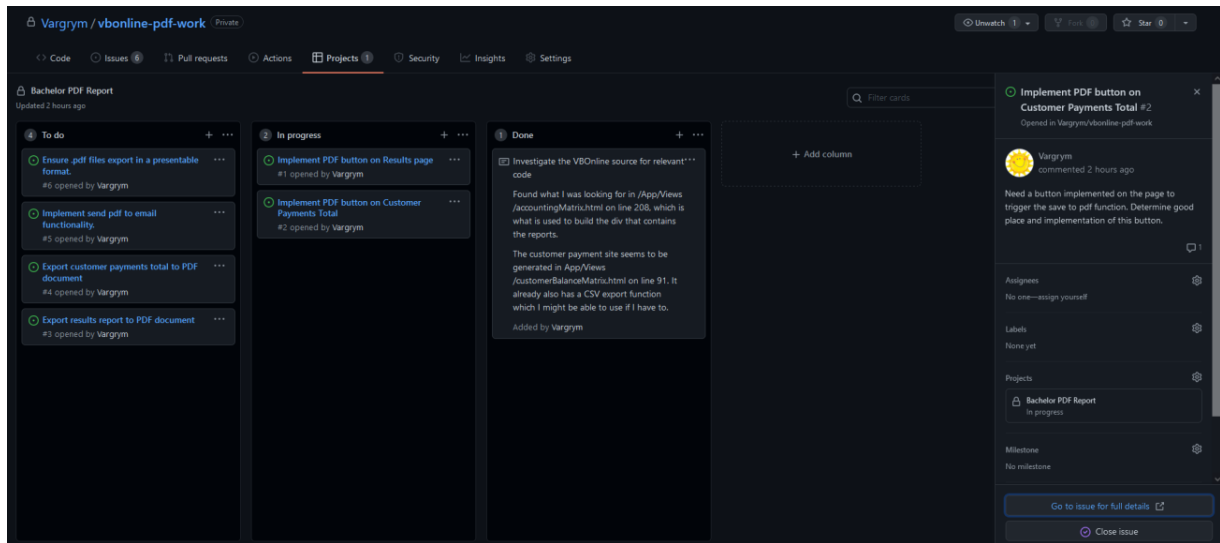
The problem of the "dirty" data within the VBOnline tables persists and it is a cause for concern, but I have decided that I will bring it up with the senior developers to see if we can't do something to clean it up for easier export/scraping.

Noteworthy sources used today: W3Schools HTML table articles and W3Schools CSS articles.

## 01/02/2022

Spent the day looking into source code again, trying to figure out the best approach for gathering data for a pdf report.

Decided it would be wise to organize my workflow in a Kanban system using GitHub. I will also use the repo to store the code for any functions or features I develop to help me in creating the desired functionality for VBOnline. So I spent a lot of the day organizing code snippets and establishing issues and to do list items in the form of user stories on a private GitHub repository.



03/02/22

Did research into Angular directives and figured out how to start integrating the function I need into the website. There isn't too much to write about it except that it took far too long because Angular's naming conventions don't follow from any other language that I know of. It parses elements written with hyphens (ex: such-as-this) and attaches directives named using camel case to it (ex: suchAsThis). So a lot of confusion arose from a function called export-to-PDF which had to be written as exportToPdf.

08/02/2022

Made a first working version of the PDF save for result reports. Mostly aesthetic fiddling trying to get the PDF document to scale appropriately to height and width. I hit a snag but found out that the document height and width values were being returned as strings with px appended, so I wrote a basic function to trim and cast the values to int before feeding it to the system as parameters.

I relied heavily on jsPDF html documentation ([source](#)) and MrRio's demos ([source](#)).

10/02/2022

No progress due to sickness.

15/02/2022

Was tasked with resolving some refinements on previous work. Worked on issue #286 and disabled the PDF export button if the account suspension period flag was enabled. Looked into various resources on how to resolve this but decided to opt into simply using the existing logic to set the flag. Hopefully this way if they make the change in one location they'll persist in others.

Went on to work a bit on my other PDF export features in implementing a button and the required information foundation to export the correct tables on the customer payment accounts reports page. I have a good idea of how to handle this using JavaScript to iterate over the table on the site. I believe if I succeed on the customer side, I can also succeed on the supplier side as they use a rather similar format.

Made good progress by implementing a JavaScript library to extend jsPDF functionality called AutoTable. It can take html data and feeding it into a table to automatically format it. I was planning on manually coding a function to parse the HTML table row by row and constructing a table to feed into the PDF using jsPDF.

[24/02/2022](#)

Came down with COVID-19 right after getting my vaccination booster shot, so this impeded work progress quite a bit.

It was requested of me to refine the AutoTable solution and I have been working on it by trying to code a robust report-generator with localization that will work dynamically across both pages. This should hopefully save me some work in the long run by letting me maintain a single function for both instead of having to check my work and apply fixes in both places.

Update: Spent most of the day coding. I believe I have now implemented a saveable PDF report for both client and supplier ledgers.

Some weaknesses of my design: If they change the order of the columns a lot of the functionality will no doubt break, but there aren't many good ways around this for now. I opted to write the code to be readable instead so if someone does change the table structure in the future they can repair it.

[1/03/2022](#)

Logged my work and tracked down code snippets for the boss. Did some minor bug-fixing and added a few missing translations.

[3/03/2022](#)

Pushed a more final version of my pdf report generation code to repo and waited for that to be tested and approved. New version handles column orders better, so small changes shouldn't break the export function. Spent a lot of the day preparing for a meeting with project group in other course. Had a phone call about maybe getting a job.

[8/03/2022](#)

Tasked with digging up code and a reference to the API call method which determined if a user was active. It was obfuscated and hidden behind a few layers. Eventually had to resort to scrubbing the scope to find it.

Started looking at adding document type to a sales approval menu. Made some headway but by no means done.

**10/03/2022**

Asked to investigate credit note display in purchase menu. Solved by adding function to the purchase controller module to return true if the document type is a specific value.

**15/03/2022**

Got permission to start implementing utility services in VBOnline. So I established a utility service for string manipulation in the VBOnline app.

Utility service for strings implemented and attached to all exports where btoa might fail.

Also added some information to a form in VBOnline for credit note documents.

**17/03/2022**

Minor problem with the new utility service function; the Regular Expression shorthand doesn't work with variables, so I had to go back to long form regular expressions in JavaScript. Switching them out fixed everything and the change has been rolled out.

**22/03/2022**

Opting to spend more time on writing the report for the internship from now on. I have already worked out a lot of the issues assigned to me and it's a better use of our collective time if I work on the report. The work will take on a more constant effort both at home and the office, so the logs will probably stop here.

**12/04/2022**

Worked up an application in C# to update exchange rates over the previous weekend because Egil asked for it. I'm not proud of it, but it works. Wasted a fair amount of time at first making an API that redirected an exchange rate value. After this has concluded I will get back to focusing on writing.

## Attachment 5

### Code – Viktor

Syntax highlighting and codeblock backgrounds removed to save toner in case of printing.

#### Results report

```
app.directive('exportDivToPdf',function(){
  return {
    restrict: 'A',
    link: function (scope, element, attrs) {
      element.bind('click', function(e){
        const fileName = attrs.filename;
        const tableName = attrs.tableName;
        savePDFfromElement(tableName, fileName);

        function savePDFfromElement(elementID, filename){
          window.jspdf = window.jspdf.jspdf;
          const width = parseInt(window.getComputedStyle(document.getElementById(elementID)).width.slice(0,-2)) + 20;
          const height = parseInt(window.getComputedStyle(document.getElementById(elementID)).height.slice(0,-2)) + 30;
          let pdf = new jsPDF('l','px', [height, width]);

          pdf.html(document.getElementById(elementID), {
            callback:function(pdf) {
              pdf.save(filename);
            },
            x: 10,
            y: 10,
          });
        }
      });
    }
  });
});
```

#### Customer/Supplier ledger reports

```
app.directive('exportToPdf',function(){
  return {
    restrict: 'A',
    link: function (scope, element, attrs) {
      element.bind('click', function(e){
        const tableElement = document.getElementById(attrs.tableName);
        const fileName = attrs.filename;
        const jsTable = convertHtmlTableToArray(tableElement);
```



```
savePdfFromArray(jsTable, fileName);
```

```
function savePdfFromArray(arrayTable, pdfname){  
  window.jsPDF = window.jspdf.jsPDF;  
  let pdf = new jsPDF('p', 'pt', 'a4');  
  const documentMargin = 20;  
  
  addHeader(pdf, arrayTable);  
  const columnsToRemove = ["Kundesøk", "Customer", "Leverandør", "Supplier"];  
  addTables(pdf, removeTableColumns(columnsToRemove));  
  addPageCount(pdf, documentMargin);  
  pdf.save(pdfname + " " + getReportSubject() + " " + getReportPeriod().replace(/ /g, "") + ".pdf");  
}
```

```
function addHeader(document, dataArray) {  
  const documentWidth = document.internal.pageSize.getWidth();  
  const dateGenerated = getCurrentTime("no-NO");  
  const reportType = getReportType(dataArray);  
  const reportSubject = getReportSubject(dataArray);  
  const reportPeriod = getReportPeriod(dataArray);  
  
  let widthWrite = documentMargin;  
  let heightWrite = documentMargin;  
  document.setFont("helvetica");  
  
  writeHeadline();  
  drawLine();  
  writeGenerationInformation();  
  writeReportTypeAndSubject();  
  writeReportTimePeriod();  
  drawLine();  
  
  function drawLine(){  
    document.setLineWidth(0.2);  
    document.line(documentMargin, heightWrite, documentWidth-documentMargin, heightWrite);  
    heightWrite += 12;  
  }  
  
  function writeHeadline(){  
    document.setFontSize(15);  
    document.text(reportType + localizeTitle(), widthWrite, heightWrite);  
    heightWrite += 7;  
    document.setFontSize(8);  
  
    function localizeTitle(){
```

```

const lang = navigator.language;
if (lang == "nb-NO" || lang == "no-NO" || lang == "nb"){
    return "reskontro";
} else {
    return " Payments Account";
}
}
}

function writeReportTypeAndSubject(){
    document.text(reportType + ": " + reportSubject, widthWrite, heightWrite);
    heightWrite += 10;
}

function writeReportTimePeriod(){
    document.text(localizePeriod() + reportPeriod, widthWrite, heightWrite);
    heightWrite += 10;

function localizePeriod(){
    const lang = navigator.language;
    if (lang == "nb-NO" || lang == "no-NO" || lang == "nb"){
        return "Periode: ";
    } else {
        return "Period: ";
    }
}
}

function writeGenerationInformation(){
    document.text(localizeGenerationInfo() + dateGenerated, documentWidth - widthWrite, heightWrite,
"right");

function localizeGenerationInfo(){
    const lang = navigator.language;
    if (lang == "nb-NO" || lang == "no-NO" || lang == "nb"){
        return "Rapport generert: ";
    } else {
        return "Report generated: ";
    }
}
}
}

function addTables(document, table){
    const numberColumnNames = [

```

```

    "Dokument", "Document",
    "Debet Beløb", "Debit Amount",
    "Kredit Beløb", "Credit Amount",
    "Cross Reference"
  ];
  const columnStylesObject = addColumnAlignment(numberColumnNames, "right", table);

  document.autoTable({
    startY: 70,
    margin: {
      top:documentMargin+10,
      right:documentMargin,
      bottom:documentMargin,
      left:documentMargin,
    },
    styles: {fontSize: 8, lineWidth: 0.5, cellPadding: 3, },
    columnStyles: columnStylesObject,
    head: table.slice(0, 1),
    body: table.slice(1),
  });

  function addColumnAlignment(columnNameArray, alignment, columnStyleTable){
    const styleString = {
      halign: alignment
    }
    const numberColumnIndex = findIndexOfColumns(columnNameArray, columnStyleTable);

    let workerStyleObject = {};
    for (let columnIndex of numberColumnIndex){
      workerStyleObject[columnIndex] = styleString;
    }
    return workerStyleObject;
  };
}

function addPageCount(document, margin = 10){
  const PageCountText = localizePageCount();
  const pageCount = document.internal.getNumberOfPages();
  const documentWidth = document.internal.pageSize.getWidth();

  for(let i = 0; i < pageCount; i++) {
    document.setPage(i);
    let pageCurrent = document.internal.getCurrentPageInfo().pageNumber;
    document.setFontSize(7);
  }
}

```

```
        document.text(PageCountText[0] + pageCurrent + PageCountText[1] + pageCount,
documentWidth-margin, margin, "right");
    }
```

```
function localizePageCount(){
    const lang = navigator.language;
    if (lang == "nb-NO" || lang == "no-NO" || lang == "nb"){
        return ["Side: ", " av "];
    } else {
        return ["Page: ", " of "];
    }
}
}
```

```
function getCurrentTime(localization){
    return new Date().toLocaleString(localization);
}
```

```
function getReportType(){
    const reportTypeColumn = getReportTypeColumn();
    let type = arrayTable[0][reportTypeColumn];
    if(type == "Kundesøk"){
        type = "Kunde";
    }
    return type;
}
```

```
function getReportTypeColumn(){
    const reportTypeColumnTitles = ["Kundesøk", "Customer", "Leverandør", "Supplier"];
    return findIndexOfColumns(reportTypeColumnTitles);
}
}
```

```
function getReportSubject(){
    const subjectColumn = getSubjectColumn();
    return arrayTable[2][subjectColumn];
}
```

```
function getSubjectColumn(){
    const subjectColumnTitles = ["Kundesøk", "Customer", "Leverandør", "Supplier"];
    return findIndexOfColumns(subjectColumnTitles)[0];
}
}
```

```
function getReportPeriod(){
    const monthDictionary = localizeMonth();
    let month = getFirstMonthFromTable();
}
```

```

let year = getYearFromTable();

if (scope.selectedMonth === undefined){
    return "1 - 12/" + year;
}
return monthDictionary[month] + " (" + month.slice(-1) + ")/" + year;

function localizeMonth(){
    const lang = navigator.language;
    if (lang == "nb-NO" || lang == "no-NO" || lang == "nb"){
        return {
            "01": "Januar",
            "02": "Februar",
            "03": "Mars",
            "04": "April",
            "05": "Mai",
            "06": "Juni",
            "07": "Juli",
            "08": "August",
            "09": "September",
            "10": "Oktober",
            "11": "November",
            "12": "Desember",
        };
    } else {
        return {
            "01": "January",
            "02": "February",
            "03": "March",
            "04": "April",
            "05": "May",
            "06": "June",
            "07": "July",
            "08": "August",
            "09": "September",
            "10": "October",
            "11": "November",
            "12": "December",
        };
    }
}

function getYearFromTable(){
    const dateColumn = getDateColumnIndex();
    return arrayTable[2][dateColumn].slice(-4);
}

```

```

}

function getFirstMonthFromTable(){
  const dateColumn = getDateColumnIndex();
  return arrayTable[2][dateColumn].slice(3,5);
}

function getDateColumnIndex(){
  const dateColumnTitles = ["Bilagsdato", "Voucher Date"];
  return findIndexOfColumns(dateColumnTitles)[0];
}
}

function findIndexOfColumns(titleArray, table = arrayTable){
  let indexArray = [];
  for(let title = 0; title < titleArray.length; title++){
    if (!indexArray.includes(findColumnName(titleArray[title], table))){
      indexArray.push(findColumnName(titleArray[title], table));
    }
  }
  return removeUndefinedElementsFromArray(indexArray);

  function removeUndefinedElementsFromArray(array){
    return array.filter(function (element){
      return element !== undefined;
    })
  }
}

function findColumnName(columnName, table = arrayTable){
  const tableHeader = table[0];
  for(let cell = 0; cell < tableHeader.length; cell++){
    if (tableHeader[cell] == columnName){
      return cell;
    }
  }
}

function removeTableColumns(nameArray){
  const filterArray = findIndexOfColumns(nameArray);
  let tempArray = [];
  for (let row = 0; row < arrayTable.length; row++){
    try{
      let tempRow = [];
      for(let column = 0; column < arrayTable[row].length; column++){

```

```

        if(!filterArray.includes(column)){
            tempRow.push(arrayTable[row][column]);
        }
    }
    tempArray[row] = tempRow;
} catch(err) {
    console.error(err);
}
}
return tempArray;
}
}

```

```

function convertHtmlTableToArray(htmlTable){
    let arrayToReturn = [];
    for (let i = 0; i < htmlTable.rows.length; i++){
        let cells = [];
        for(let j = 0; j < htmlTable.rows[i].cells.length; j++){
            if (!htmlTable.rows[i].cells[j].outerHTML.includes("hide-download")){
                cells.push(htmlTable.rows[i].cells[j].innerText);
            }
        }
        arrayToReturn[i] = cells;
    }
    return arrayToReturn;
}

```

```

function generateTestArrayTable(numberOfRows, numberOfColumns){
    let testArray = generateRows(numberOfRows, numberOfColumns);

```

```

    function generateRows(numRows, numColumns){
        let generatedTable = [];
        for (let row = 0; row < numRows; row++){
            generatedTable[row] = generateCells(numColumns, row);
        }
        return generatedTable;
    }

```

```

    function generateCells(columnCount, rowNumber){
        let generatedRow = [];
        for (let cell = 0; cell < columnCount; cell++){
            generatedRow[cell] = "Row " + rowNumber + " Cell " + cell;
        }
        return generatedRow
    }
}
}

```

```

        return testArray;
    }
    });
}
});

```

## String utility service

```
'use strict';
```

```
app.factory('utilityStringService', function () {
    let utilityStringServiceFactory = {};

```

```
    let _sanitizeString = function (workerString, additionalTerms = {}){

```

```
        const replacementDictionary = {

```

```
            "&nbsp;": " ",

```

```
            "&amp;": "&",

```

```
            "&apos;": "'",

```

```
            "&quot;": "\"",

```

```
            "&gt;": ">",

```

```
            "&lt;": "<",

```

```
            "&reg;": "®",

```

```
            "&copy;": "©",

```

```
            "%": "% ",

```

```
            "_": "_ "

```

```
        }

```

```
        addToDictionary();

```

```
        replaceUsingDictionary();

```

```
        return workerString;

```

```
    function addToDictionary(){

```

```
        if (Object.keys(additionalTerms).length > 0){

```

```
            for (let key in additionalTerms){

```

```
                replacementDictionary[key] = additionalTerms[key];

```

```
            }

```

```
        }

```

```
    }

```

```
    function replaceUsingDictionary(){

```

```
        for (let key in replacementDictionary){

```

```
            workerString = workerString.replace(new RegExp(key, 'g'), replacementDictionary[key]);

```

```
        }

```

```
    }

```

```

}

```



```

let _btoaCleaner = function (btoaString){
  try{
    return btoa(btoaString)
  }catch(error){
    return btoa(removeBadSymbols(findBadBtoASymbols(_returnUniqueSymbols(btoaString)), btoaString));
  }

  function removeBadSymbols(badSymbolArray, stringToClean){
    let workerString = stringToClean;
    for (let badSymbol of badSymbolArray){
      workerString = workerString.replace(new RegExp(badSymbol, 'g'), "");
    }
    return workerString;
  }

  function findBadBtoASymbols(stringContainingBadSymbols){
    let badSymbols = [];
    for (let symbol of stringContainingBadSymbols){
      if(tryBtoAOnSymbol(symbol) == false && !badSymbols.includes(symbol)){
        badSymbols.push(symbol);
      }
    }
    return badSymbols;
  }

  function tryBtoAOnSymbol(symbol){
    try {
      btoa(symbol);
      return true;
    } catch (error) {
      return false;
    }
  }
}

let _returnUniqueSymbols = function (stringToParse){
  let workerString = "";
  for(let character of stringToParse){
    if (workerString.includes(character)){
      continue;
    }
    workerString += character;
  }
  return workerString;
}

```

```
}  
  
utilityStringServiceFactory.sanitizeString = _sanitizeString;  
utilityStringServiceFactory.btoaCleaner = _btoaCleaner;  
utilityStringServiceFactory.returnUniqueSymbols = _returnUniqueSymbols;  
return utilityStringServiceFactory;  
});
```