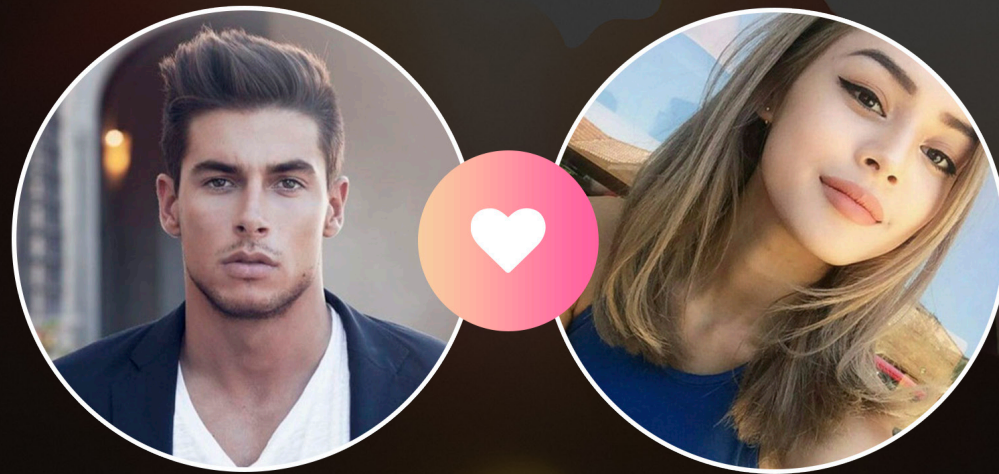


# Web Development Exam Project

Pair - Dating Web Application



**By Nikola Wulf-Andersen**

1. semester KEA 16-03-18

## TABLE OF CONTENTS

Introduction	3	<b>Execution</b>	13-15
<b>Structure</b>	4-5	jQuery	13
Sitemap	4	AJAX	13
Text files	5	LocalStorage	13
<b>Design</b>	6-7	For loop vs. Foreach	14
Design system	6	Upload to server	15
Libraries	7	<b>Test</b>	16
Bootstrap	7	API testing	16
Font Awsome	7	Usability testing	16
Sweet alerts	7	Testing system and features	16
<b>The system</b>	8-10	<b>Conclusion</b>	17
Login and signup	8		
Validation	8		
Verification email	8		
Find match	9		
Match with a user	10		
Benefits as a VIP user	11		
Administartor of the system	12		

### INTRODUCTION

This exam project is about doing a full stack system, in this case a dating web application like Tinder. My concept is called Pair and the main functionalities are very similar to Tinder. On the web application the user should be able to signup, login, edit their profile, like/dislike other users, see their matches, chat and so on. Besides these main functionalities for a normal user there should be an option to upgrade to VIP user and get more benefits. There will also be an administrator of the site who can CRUD any user of the system.

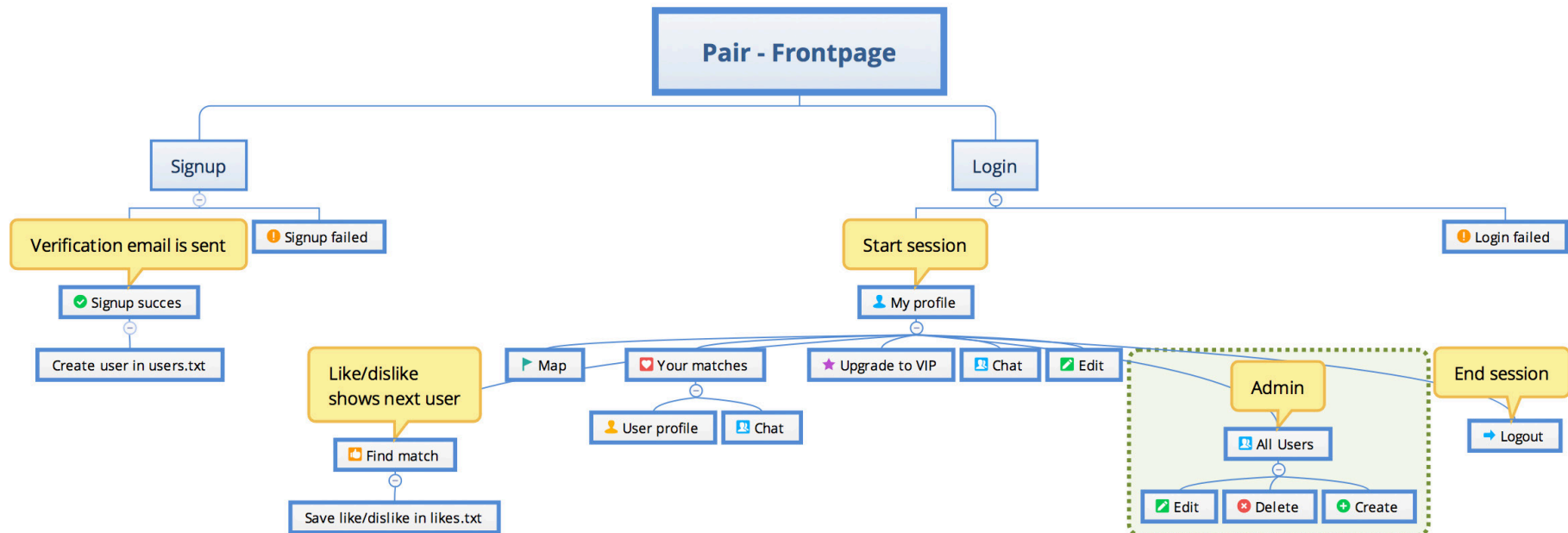
I have decided to divide the report into five sections: Structure, Design, The system, Execution and Test. Within these four sections I will explain how I got from A to B and which considerations and thoughts I have had during this project.

# STRUCTURE

## SITEMAP

Before I began coding anything on the project I started out creating a sitemap to get an overview of the different pages the web application should consist of. When working on a relatively large project like this it is important to understand the structure and planning of it before starting to write the code. In this way it was easier for me to take all the different components into account and get them to work together in the bigger picture. I think it also limited the number of mistakes I made in the beginning. When doing the sitemap, I tried to think ahead and foresee what information and things I needed later on in the process, so I didn't have to go back and forth and correct files all the time.

For example, when the user is created in the beginning after signup they only have to pass their name, email and password. Of course, a user needs more information, so instead of only creating those three keys I also created most of the other keys like id, gender, images, age, description, if the user is verified and so on. This is just a small example and not one that would take a long time to correct if I forgot a key or two, but some decisions can take significant time to correct and have an impact on the final result, so therefore it helped me to think about some of these decisions beforehand.



## TEXT FILES

For storage of my data I chose to use three separate text files. One text file with all my users (users.txt), one for the messages (chat.txt) and one to store all the likes/dislikes (likes.txt). The structure of the three text files are different which is why I chose to have it in separate files.

### users.txt

The users.txt file consists of an array with JSON users. Every user has 13 keys as you can see in the picture below. The key “images” is an array inside the JSON object and inside that array there can be stored up to four images.

```
{
  "name": "Anna",
  "lastName": "Smith",
  "email": "nikola@wulfandersen.dk",
  "password": "nnnnnn",
  "age": "25",
  "gender": "Woman",
  "interests": "Reading, going out",
  "description": "Hi I'm Anna! Looking for love",
  "status": "standard",
  "images": [
    "5a9ac0d5426d2.jpg",
    "5a9add370ab23.jpg"
  ],
  "id": "5a9803ca0c190",
  "verificationToken": "5a9803ca0c197",
  "verified": true
},
```

### likes.txt

The likes.txt file is the file I use to store likes and dislikes. This is also an array containing JSON objects. Inside every object are four keys: From, To, Like and Notified. The first key “From” stores the id of the user which the like or dislike is from and the key “To” stores the id of the user which the like is directed to. The key “Like” is a Boolean which is either set to true or false depending on whether it is a like or dislike. The last key “Notified” is also a Boolean which is either true or false depending on whether the user has been notified via a desktop notification or not.

```
{
  "from": "5aa242c026017",
  "to": "5aa23fa36a447",
  "like": true,
  "notified": false
},
```

### chat.txt

The chat.txt file stores all the messages from the chat. This file also has four keys: From, To, Message and Sent. “From” and “To” contains the id’s of the two-people chatting. The “Message” key contains the message and the last key “Sent” checks whether the message has been received or not. Default setting is false and then, when the message is being appended, it is set to true. To check if there is any new messages I used the method “setInterval()” to check every second if there is any messages in the array where the “Sent” key is set to false.

```
{
  "from": "5aa23f0458f2a",
  "to": "5aa23fa36a447",
  "message": "I'm good!",
  "sent": true
},
```

# DESIGN

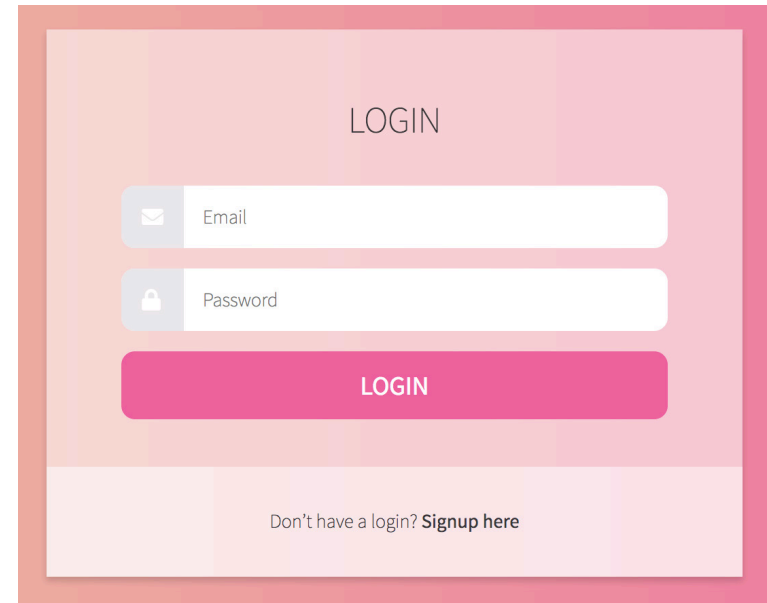
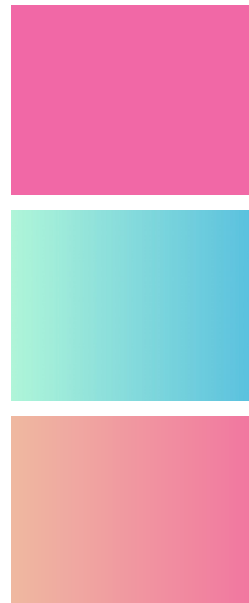
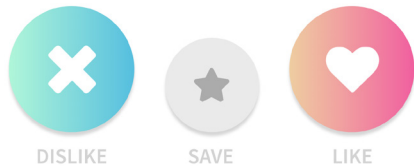
## DESIGN SYSTEM

After getting an initial overview of the project, I started thinking about the design and which external libraries to use. The first thing I did was to design the different components for my design system. I chose the color scheme, button design, forms, icons, fonts and so on.

For the colors I decided to use two gradients, one primary gradient that goes from yellow to pink and one secondary gradient that goes from green to blue. The primary gradient is used for the background color and sometimes also for buttons. The secondary gradient is mainly used for buttons. I also picked one primary solid color: pink.

I decided to use one primary font for my web application called "Source Sans Pro" which is embedded as a google font. This font comes in a variety of font-weights and goes from extra light to black.

It was my intention to make an appealing and consistent design throughout the whole dating web application. I think it is important that the user feels inspired and get the feeling that it is a professional web application. This will make the user feel more secure and not hesitate to give up their personal information.



### LIBRARIES

#### Bootstrap

Most of my design is grid based because there is more flexibility compared to external design systems like bootstrap. When that is said I did use a couple of things from bootstrap, like for instance their table. If I had had more time I would have created the whole web application using grid, so I could customize down to the last detail, but because of the time constraint I used bootstrap a couple of places to save time.

#### Font Awesome

Another external library I used for my web application was Font Awesome. All the icons used in the web application originate from this. Instead of having to draw all the icons myself I could just copy the code provided on their website, and in no time, I could get vector-based icons on my site.

#### *SweetAlert*

To give the users feedback on my site I chose to use something called Sweet Alerts. To improve the user friendliness on my site I embedded them several places in order to give the user either a success or error message. This will help the user understand what's going on and know if the action they are trying to perform is ok or not.

# THE SYSTEM

## SIGNUP AND LOGIN

The first thing the user will see when they visit my dating site is a landing-page with two options: they can either sign up or login. For the signup form I chose to require very little information because it should not be too comprehensive for the user to sign up. Then after they have signed up and logged in, they are asked to fill in the rest of the information needed to complete the profile like the description of themselves, age, gender, pictures etc.

SIGN UP NOW

✓

✓

✗

```
function onBlurValidate(e) {
  validateInput(e.target);
}

function validateInput(input) {
  var value = $(input).val()
  if (value.length >= $(input).attr('data-min') && value.length <= $(input).attr('data-max')) {
    $(input).parent().append('<i class="fas fa-check-circle validIcon"></i>');
    $(input).siblings('.notValidIcon').hide();
    console.log('Valid')
    return true;
  } else {
    $(input).parent().append('<i class="fas fa-times-circle notValidIcon"></i>');
    $(input).siblings('.validIcon').hide();
    console.log('Not Valid')
    return false;
  }
}

$('#txtSignupName').blur(onBlurValidate);
```

### Validation

When the user signs up for the dating web application all the information the user enters are validated both front-end and back-end. In the front-end, the length of each input (number of characters) are validated using a “data-min” and “data-max” for each input. To improve the user friendliness I validate the input boxes on the event “blur()”, so the user immediately get feedback if the input is valid or not. In the back-end I use to parameters to validate the input. First, I check if the value is entered or not with the “isset()” and afterwards I check if the min and max length is okay as well.

### Verification email

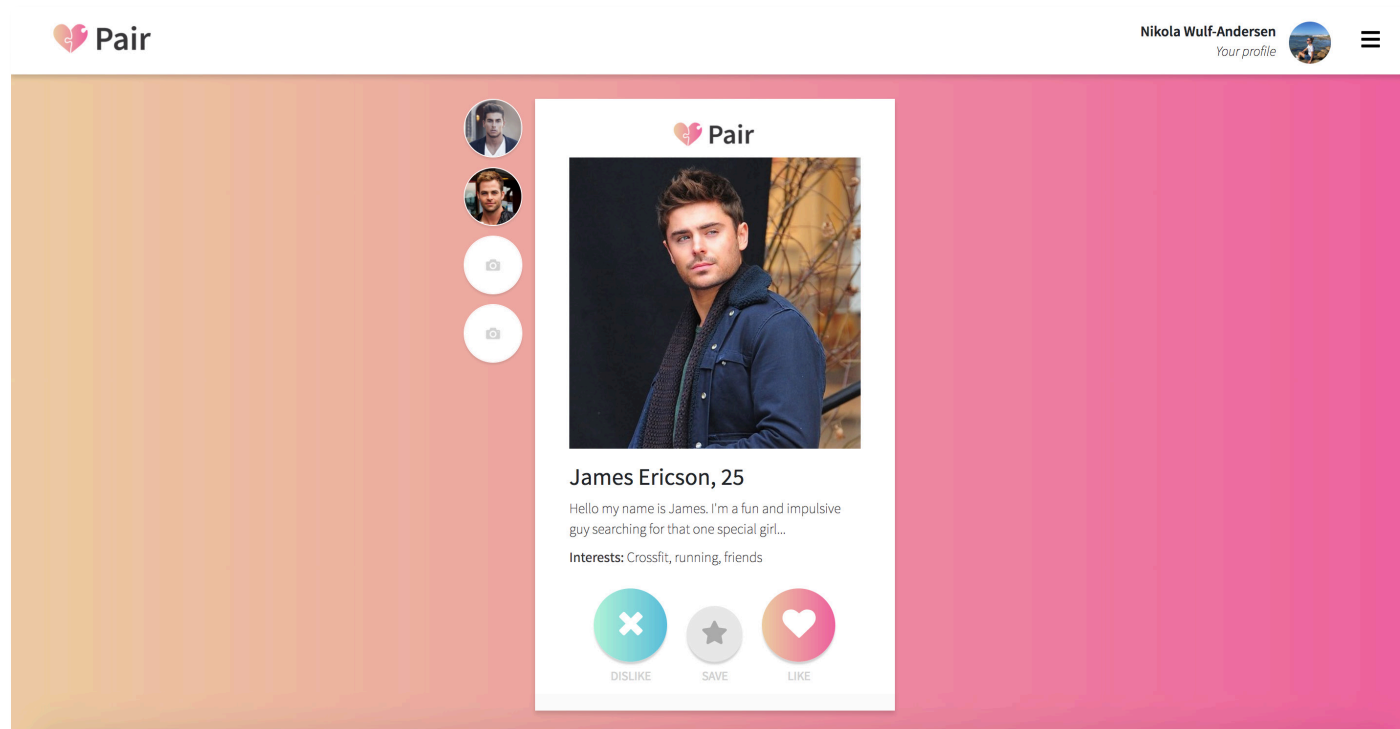
To make sure that the email address entered by the user in the signup form is correct, a verification email is sent to that email address, so it can be verified. If the email is not verified the user cannot continue to login.



## FIND MATCH

After the user has signed up and completed their profile they can click on “Find match”, and if a man is logged in it will only show female profiles and if a woman is logged in it will only show male profiles. The first user profile shown is displayed in the center of the page. The user profile has a profile picture, name, age, interests and a short description. The user has the opportunity to choose either “like”, “dislike”, or “save for later” if they cannot decide. A profile description can be up to 500 characters long, which would be too much for this layout. Therefore I created a short description using the method “Substr()” to limit the amount of characters, so it fit inside my layout.

The user has the ability to save up to four profiles. If a profile is saved for later their profile picture will pop up on the left side in one of the four thumbnail placeholders. If the user clicks on the thumbnail that particular user will be displayed and then the user can either like or dislike that person.

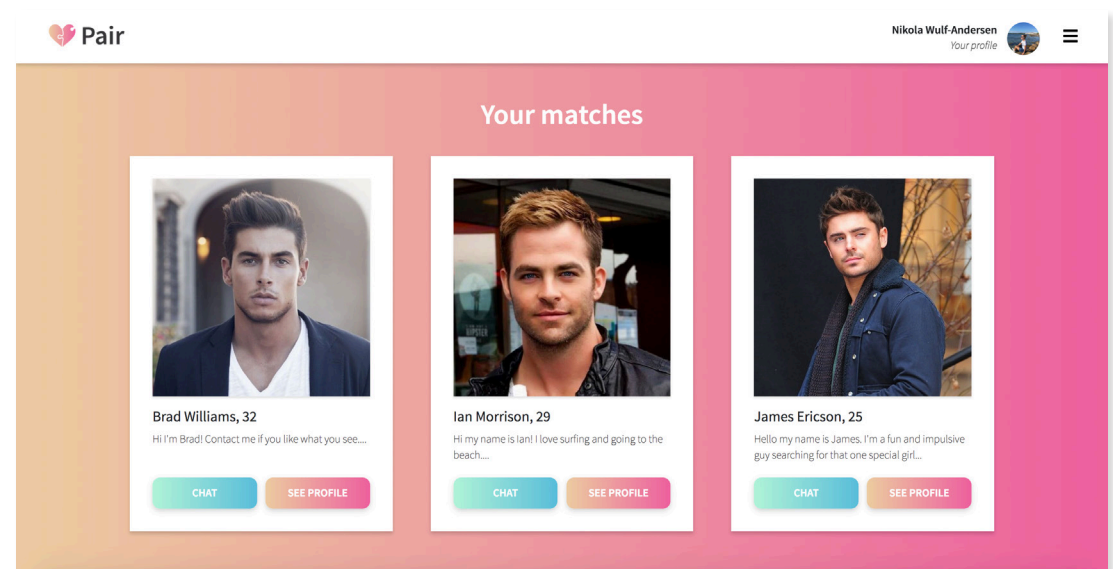


## MATCH WITH A USER

To match with another user both users have to “like” each other. To check if two people have matched I use the method “array\_intersect()” which checks if there are any matches between two arrays or more. Before I use “array\_intersect()” I start out by creating two new arrays one containing the id’s of users who have liked me, and one containing the id’s of users who I have liked. To extract these id’s I use an “if” statement to compare it against the “\$sId” which is the session id meaning the user logged in. If either the like from or to is equal to the session id I will push the opposite id to the arrays.

If you match with someone you have the possibility to chat with that person and see a more detailed profile. All matches will be displayed in the page “Your matches”. From here you can click on either “Chat” or “See Profile” and then via a GET I pass the users id to the next page.

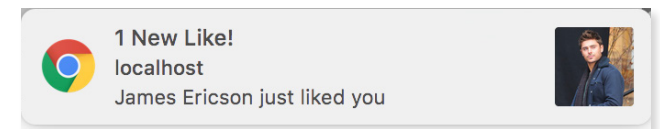
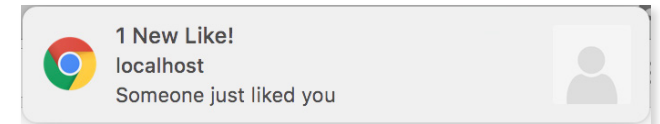
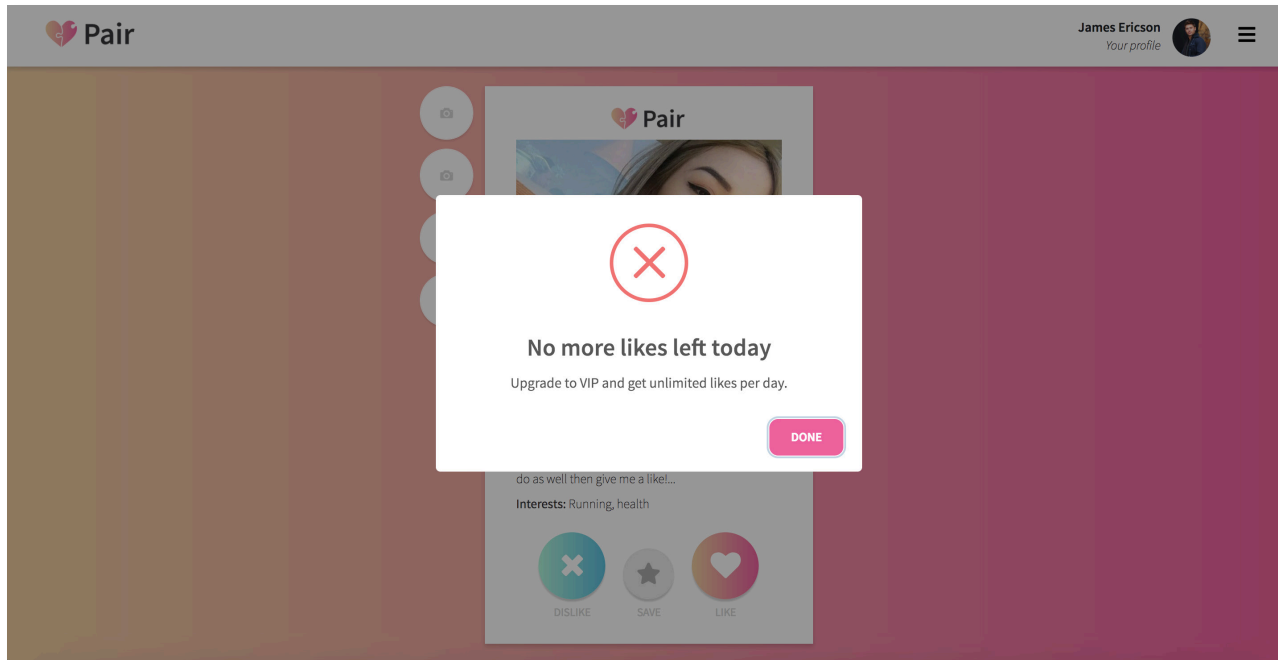
```
$aLikesTo = [];  
$aLikesFrom = [];  
  
foreach( $ajLikes as $jLike ){  
    if( $jLike->from == $sId && $jLike->like == true ){  
        array_push( $aLikesTo, $jLike->to );  
    }  
  
    if( $jLike->to == $sId && $jLike->like == true ){  
        array_push( $aLikesFrom, $jLike->from );  
    }  
}  
  
$matches = array_intersect( $aLikesTo, $aLikesFrom );
```



## BENEFITS AS A VIP USER

As a VIP user of the system the user will get more benefits than the normal users. Normal user of the system have a limited amount of likes and cannot see who likes them. The desktop notification will still show when you get a like, but it will be

anonymous. If you instead choose to upgrade to a VIP user you will see the picture and the name of the person who liked you in the desktop notification. Also, on the find match page you will as VIP have the ability to like as many people as you want.



## ADMINISTRATOR OF THE SYSTEM

For every user of the system I created a key called “status”. This key can either be set to “standard”, “VIP” or “admin”. Since I created the system, I named myself administrator and I set my status to “admin”. As the administrator I have the option to CRUD any user of the system. As soon as the status is set to “admin” another menu item called “All user” will show in the menu. The page consist of a table where all

the users are displayed. There is also an option to delete or edit each user. In the top right corner of the window there is a button called “Add user” where it is possible to add a new user to the system. All though all users can be given administrator rights it is only meant to be given to the real administrators.

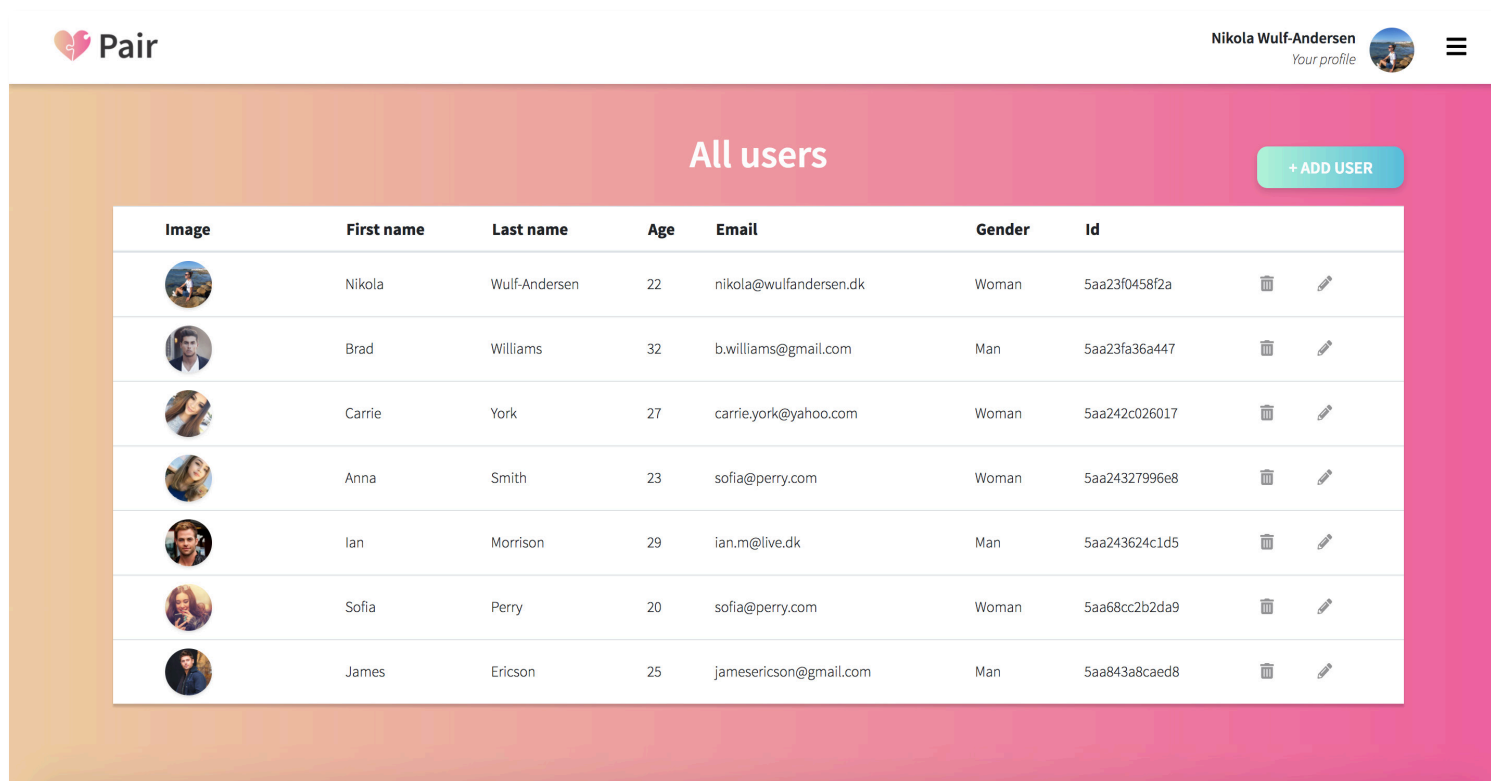























Image	First name	Last name	Age	Email	Gender	Id		
	Nikola	Wulf-Andersen	22	nikola@wulfandersen.dk	Woman	5aa23f0458f2a		
	Brad	Williams	32	b.williams@gmail.com	Man	5aa23fa36a447		
	Carrie	York	27	carrie.york@yahoo.com	Woman	5aa242c026017		
	Anna	Smith	23	sofia@perry.com	Woman	5aa24327996e8		
	Ian	Morrison	29	ian.m@live.dk	Man	5aa243624c1d5		
	Sofia	Perry	20	sofia@perry.com	Woman	5aa68cc2b2da9		
	James	Ericson	25	jamesericson@gmail.com	Man	5aa843a8caed8		

# EXECUTION

## JQUERY

jQuery is a framework for JavaScript. Throughout the project I have primarily been using jQuery instead of Query Selectors because in most cases the syntax is shorter and more manageable. It also make event handling, AJAX and the process of selecting and manipulating DOM elements much easier. Most of my web application is AJAX based, so it was obvious for me to use jQuery, because it simplified the process significantly.

## AJAX

I used AJAX a lot throughout the system for the following reasons. First it would result in a lot of pages and redirects if I did not use AJAX. Second, I think it is bad for the user experience if the user is redirected all the time, and especially if it is just to see a success or an error page. In some cases, it makes sense to redirect the user and in some cases it don't. An example where it don't make sense is on the user's profile, where the user have the opportunity to edit their personal information. When they click save it would be very frustrating if they were redirected to a page saying "Successfully updated" or "Something went wrong" and then afterwards had to go click back to see their profile again. Here it makes much more sense to show the success or error message directly on the user's profile page so they would not leave their profile when it is updated.

When using AJAX, you get more control of what will happen. If you just use the regular "submit" you are forced to be redirected where if you use AJAX you have the choice to redirect or stay on the same page. This is the primary reason why I chose to use AJAX because it gave me more control and AJAX always returns some data to the page, and with that data it is possible to give the user feedback on the action they are trying to perform.

## LOCALSTORAGE

I did not use a lot of "localStorage" throughout my system because I thought it would be less secure to store for instance password, email address, messages and other sensitive data in local storage. I only use "localStorage" to store "timestamps" which I use to check if a user has been inactive for more than 10 minutes.

```
// Timestamp - logout if inactive for 10 minutes

$(document).ready(function () {

    var date = Date.now();

    // Check if there is something in localStorage
    if( !localStorage.timeStamp){
        localStorage.setItem("timeStamp", date);
    }

    var timeStamp = localStorage.getItem("timeStamp");

    // Check if the time has exceeded 10 mintes (600000)
    if( Date.now() - timeStamp > 600000 ) {
        localStorage.removeItem("timeStamp")
        return window.location = "logout.php"
    }

    localStorage.setItem("timeStamp", date);

});
```

### FOR LOOP VS. FOREACH LOOP

I have used both the “for” loop and the “foreach” in the system. My rule of thumb was that if I could use the “foreach” loop I would prefer that and if it wasn’t sufficient in the specific case I would use the “for” loop. In most cases the “foreach” loop was adequate, but if it had to do with a user with a specific index it was necessary to use the “for” loop.

An example of that is on the “Find Match” page. When the user either clicks on the like or dislike button the next profile will show. There should of course be a counter

in the front end and that number should be sent via AJAX to the backend, where it can be used to get that exact profile index number. Furthermore, the next user is only allowed to be of opposite gender. So sometimes the index number will not just increase by one, but instead go from for instance 2 to 5 because the two profiles in-between was not of the opposite gender. In this particular case a “for” loop is therefore necessary.

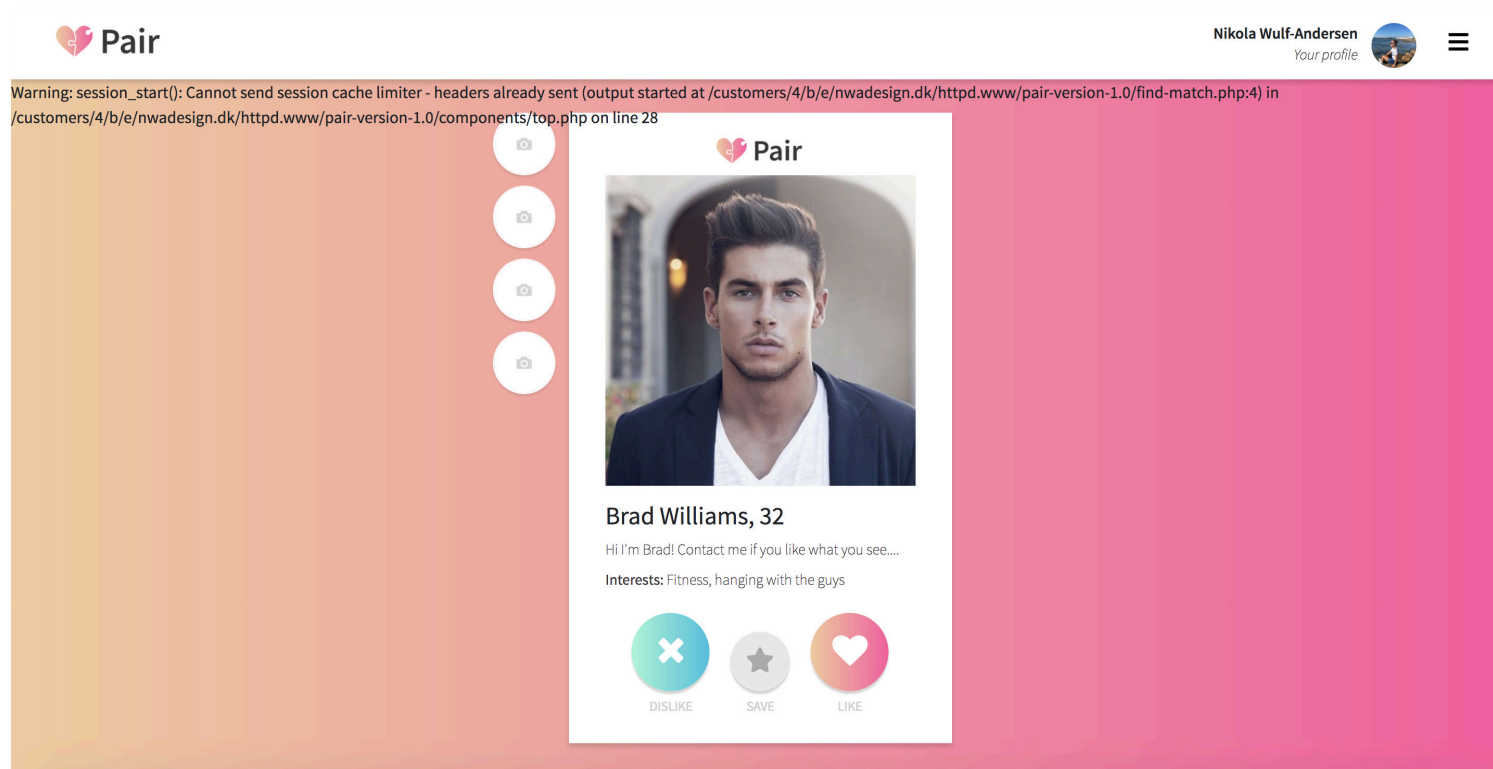
```
for($i = $iNextTemptation; $i < count($ajUsers); $i++){
    $jUser = $ajUsers[$i];
    if( $jUser->gender != $sCurrentUserGender ){
        echo json_encode($jUser);
        break;
    }
}
```

```
foreach( $ajUsers as $jUser ){
    if( $_SESSION['id'] == $jUser->id ){
        $jUser->status = $sStatus;
        break;
    }
}
```

## UPLOAD TO SERVER

I didn't have a lot of complications when I uploaded the project to my server. There was only one place where I needed to update a link, because it was referring to a "localhost" address. The only other issue I experienced was that it displayed a

couple of warnings on some pages because the session was already started in the header and it didn't show this in the "localhost". All in all these minor issues could easily be resolved.

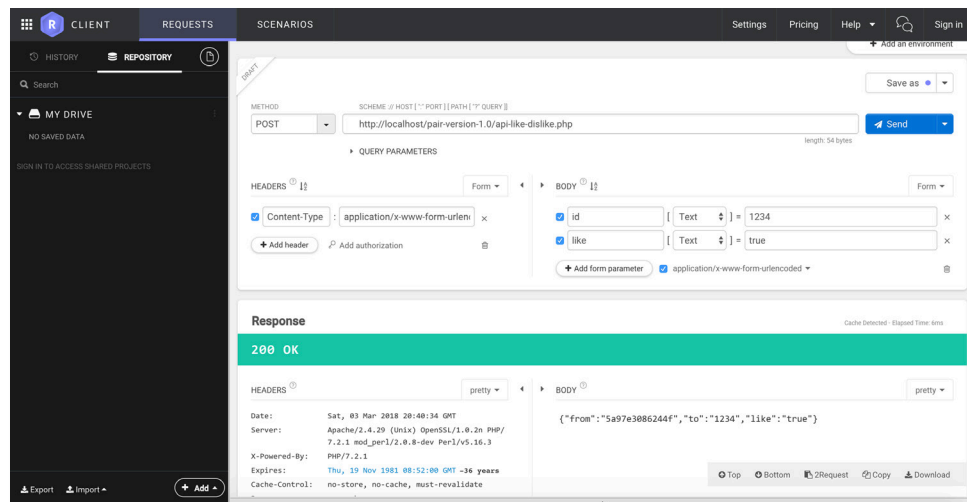


## TEST

### API TESTING

To test my APIs, I used the tool “Restlet Client”, because APIs cannot be tested very well in the browser. With “Restlet Client” I had the ability to test both “post” and “get” methods. If using a “post” method, you need to define some variables in the system, but if you use “get” the variables should be defined in the URL. The tool is very helpful because it will show if there are any errors and in which line number. If there are no errors it will show the output that you echo out in the API. This is very useful because you know if your API is doing what it is supposed to do and if it is returning the correct data.

In the beginning I didn’t test my API’s because I didn’t know about the tool “Restlet Client”, so I had no idea if my API were working correctly. I just ran my JavaScript and by accident I overwrote my whole database with an empty array. After that I started using this tool almost every time, especially if I needed to insert something or update my database.



### USABILITY TESTING

To test the usability of the system I had a couple of people sign up to the system and go through it step by step. One of the things I learned from this test was that the users wanted even more feedback than I thought. I did not have time to implement all the wishes from the user test, but I would have liked to implement more specific error messages like “Password is too short” or “All fields must be field filled in” instead of the more general error messages I have implemented.

### TESTING SYSTEM AND FEATURES

To test the system and its features I went through the process of signing up, logging in, editing, finding match, chatting and so on several times to make sure that everything was working as it should. I also tested for the success and error messages and tried on purpose to put in invalid data to see which message would pop up and if my validation was working, or if it just accepted all data.



### CONCLUSION

During this project I have learned a lot and also become more aware of the correct approaches related to both front-end and back-end. One of the things I have become better at is to debug my code. When I started out with this project I had a hard time detecting where the errors came from, so if my code wasn't working I was sometimes a bit lost. Within this project I have learned certain procedures and ways to detect these errors, which will help me a lot further on.

As I mentioned in the beginning of the report it is important to structure and plan your project before you start coding. I tried to live up to that, and it helped me a lot of the way, but I also learned that it is hard to foresee everything and there will always be some things that are unexpected. When I look back at my project now I can see that there are certain things that I would have changed if I were to do this project again.

Although I can see that the code isn't perfect, I'm still very satisfied with the end product and I also know that it is a part of a learning process and sometimes you learn from your mistakes. Often it is actually the mistakes you will remember more. For example in the beginning I didn't really understand the importance of testing an API until I accidentally deleted all content in my database. Even though it was a mistake I will remember this in the future and hopefully never make that mistake again, because I now understand the importance of testing an API.

If I had had more time to do this project I would have also liked to look more at my validation and the security of my system. When I tested it I noticed that it was possible to inject a script into the message area in the chat, because the message is appended directly. I'm not totally sure of the correct way to prevent this, but I could have appended the html first and then replaced the user created content afterwards using the method "text()". This is a thing I want to be more aware of in the future because it is so important and the system otherwise can be hacked.

In general I will say that it has been a very interesting and challenging project with a positive outcome. When you are doing a full stack system it is easy to see progress which felt very motivating for me. Every time I finished one part or feature of the system I was eager to do more and explore new features I could implement.