Architecture Document

Project Pijper Media

Version 2.0

Client: Kevin Klomp, Eelco Luurtsema, Sanne Vast

Team names:
Daniël Scheepstra
Julian Pasveer
Medhat Kandil
Dilan Adel
Jeroen Klooster
TA:
Alina Matei

Table of contents

Introduction

Pijper Media is a family business in Groningen, with a printing company, a magazine publisher and a large online branch (WeBlog Media). For our project the important part of pijper media is the online branch, they have an editorial team that writes articles about trending and viral posts/posts. We have been asked to create a new and improved version of their “social wall”, this is a web page that shows the trending and viral posts of a given set of accounts on different social media platforms. This application will show the posts from Facebook, Twitter and Instagram.

To develop this application we will use the laravel framework for the backend and we will use bootstrap frontend. We will create a MYSQL database to store all the necessary data.

Architectural overview

API Design and Endpoints

1. Retrieve home/social wall page
2. Retrieve select category preferences page
3. Store category preferences
4. Retrieve activity page
5. Retrieve my activity page
6. Retrieve calendar
7. Retrieve search function
8. Retrieve viral posts

Retrieve home page/social wall

Description
Retrieves the home page view of the application. This is the social wall which shows all posts in the database gathered from different sources from facebook, twitter and instagram. It also shows the recent activity of users that are writing about a post.

URL

```
Get /home
```

Response format
Code 200

```
return view('home', ['user' => $user, 'posts' => $posts, 'accounts' =>
$accounts], ['categories' => $categories]);
```

Code 200
Retrieving of user model

```
{
  "id": {
    "Type": "Integer"
  },
  "name": {
    "Type": "string"
  },
  "email": {
    "Type": "string"
  },
```

```
  "email_verified_at": {
    "Type": "timestamp"
  },
  "password": {
    "Type": "string"
  },
  "remember_token": {
    "Type": "string"
  },
  "created_at": {
    "Type": "timestamp"
  },
  "updated_at": {
    "Type": "timestamp"
  }
}
```

Code 200
Retrieving of post model

```
{
  "Id": {
    "type": "integer"
  },
  "post_id": {
    "type": "string"
  },
  "caption": {
    "type": "binary"
  },
  "post_url": {
    "type": "text"
  },
  "image_url": {
    "type": "mediumText"
  },
  "is_trending": {
    "type": "boolean"
  },
  "is_viral": {
    "type": "boolean"
  },
  "engagement": {
```

```
    "type": "unsignedBigInteger"
  },
  "old_engagement": {
    "type": "unsignedBigInteger"
  },
  "writer_id": {
    "type": "unsignedBigInteger"
  },
  "posted_at": {
    "type": "dateTime"
  },
  "account_id": {
    "type": "unsignedBigInteger"
  },
  "created_at": {
    "type": "dateTime"
  },
  "updated_at": {
    "type": "dateTime"
  }
}
```

Code 200
Retrieving of account model

```
{
  "id": {
    "Type": "Integer"
  },
  "category": {
    "Type": "string"
  },
  "platform": {
    "Type": "string"
  },
  "data_source": {
    "Type": "string"
  },
  "followers_count": {
    "Type": "unsignedBigInteger"
  },
  "created_at": {
    "Type": "timestamp"
```

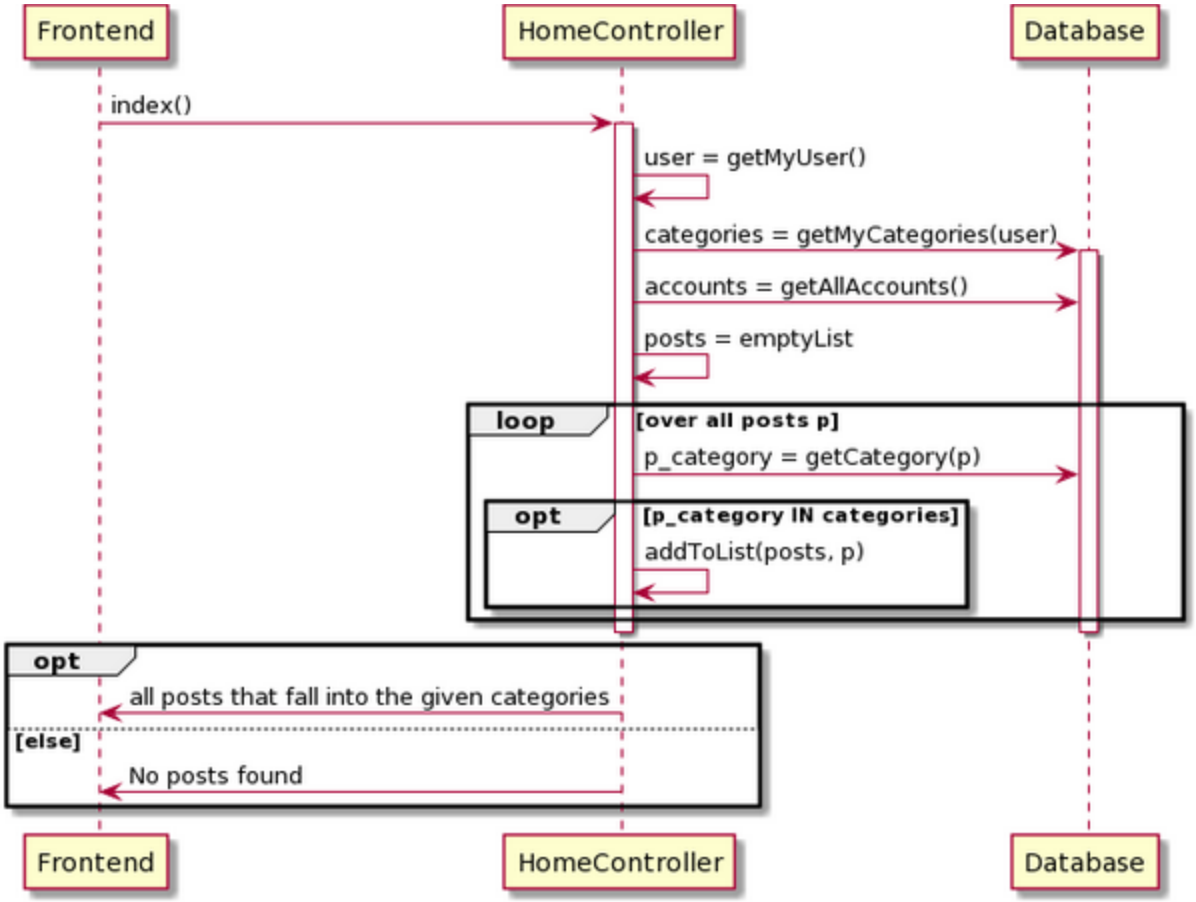
```
  },
  "updated_at": {
    "Type": "timestamp"
  }
}
```

Code 200
Retrieving of category model

```
{
  "user_id": {
    "Type": "unsignedBigInteger"
  },
  "News": {
    "Type": "tinyInteger"
  },
  "Showbizz/Entertainment": {
    "Type": "tinyInteger"
  },
  "Royals": {
    "Type": "tinyInteger"
  },
  "Food/Recipes": {
    "Type": "tinyInteger"
  },
  "Lifehacks": {
    "Type": "tinyInteger"
  },
  "Fashion": {
    "Type": "tinyInteger"
  },
  "Beauty": {
    "Type": "tinyInteger"
  },
  "Health": {
    "Type": "tinyInteger"
  },
  "Family": {
    "Type": "tinyInteger"
  },
  "House and garden": {
    "Type": "tinyInteger"
  },
```

```
  "Cleaning": {
    "Type": "tinyInteger"
  },
  "Lifestyle": {
    "Type": "tinyInteger"
  },
  "Cars": {
    "Type": "tinyInteger"
  },
  "Crime": {
    "Type": "tinyInteger"
  }
  "created_at": {
    "Type": "timestamp"
  },
  "updated_at": {
    "Type": "timestamp"
  }
}
```

Flow of Endpoint



Retrieve Select category preferences page

Description

Retrieves the page in which the user can select the categories he/she wants to base their feed on.

URL

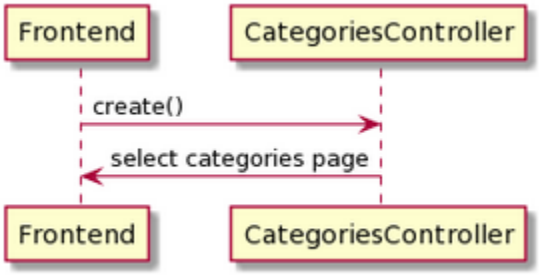
```
Get /categories
```

Response format

Code 200

```
return view('categories');
```

Flow of Endpoint



Store category preferences

Description

Once the user registers, the categories need to be stored in order to base the feed on those categories.

URL

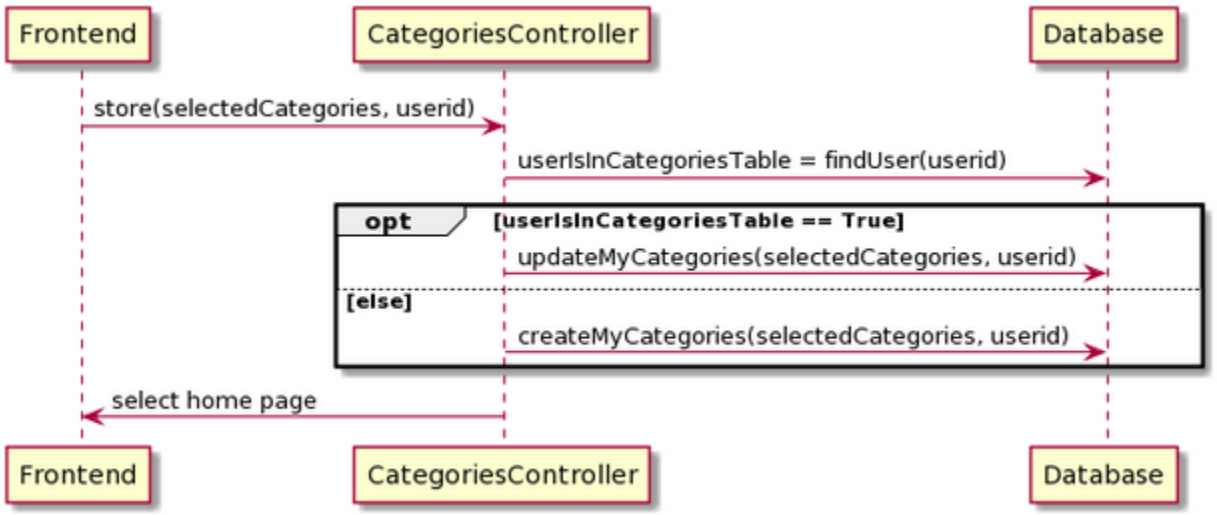
```
Post /categories
```

Response format

Code 200

```
return redirect('/home');
```

Flow of Endpoint



Retrieve activity page

Description
Once the user marks a post being written about, the post will be marked as such and needs to be shown in the activity page.

URL

```
Get /activity
```

Response format
Code 200

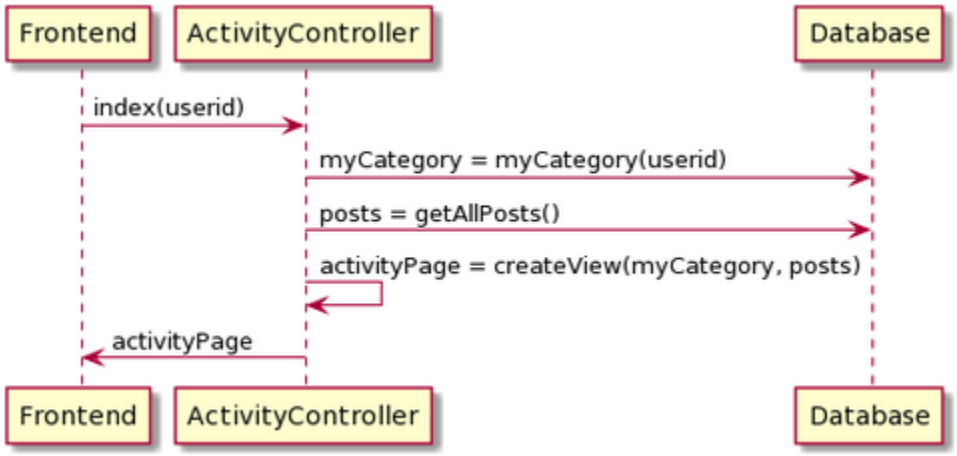
```
return view('activity', ['user' => $user, 'posts' => $posts], ['categories'
=> $categories]);
```

Code 200
Retrieving of user model as seen in a previous endpoint

Code 200
Retrieving of post model as seen in a previous endpoint

Code 200
Retrieving of categories model as seen in a previous endpoint

Flow of Endpoint



Retrieve my activity page

Description
Once the user marked some posts

URL

```
Get my_activity
```

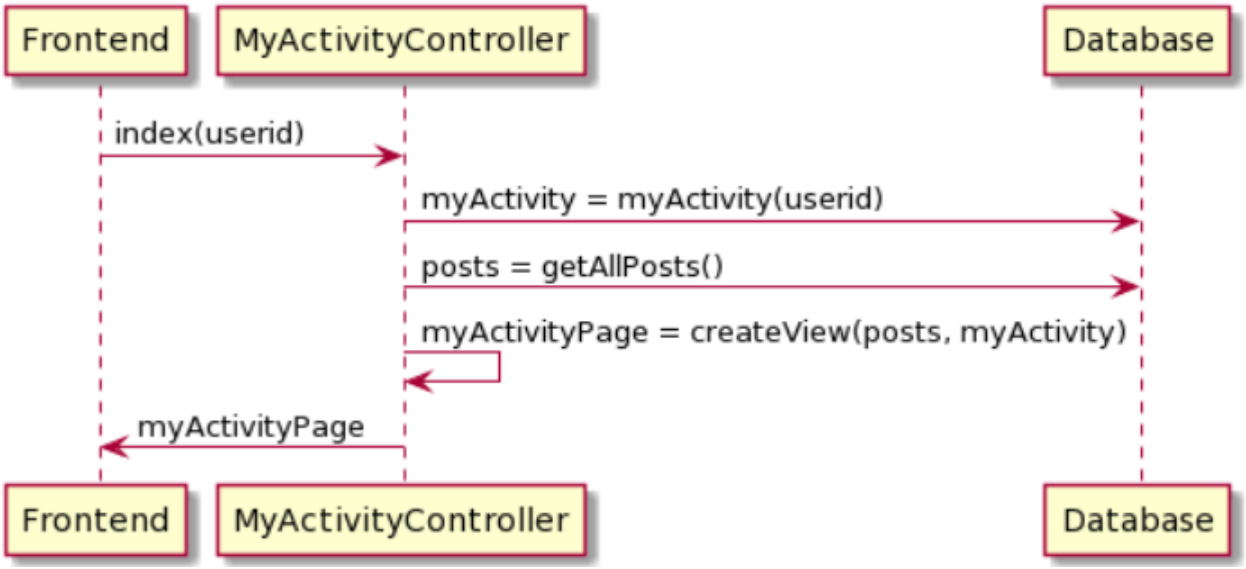
Response format
Code 200

```
return view('my_activity', ['user' => $user, 'posts' => $posts]);
```

Code 200
Retrieving of user model as seen in a previous endpoint

Code 200
Retrieving of post model as seen in a previous endpoint

Flow of Endpoint



Retrieve calendar page

Description
The calendar is a separate page with a calendar that's mostly filled and used for with birthdays of celebrities.

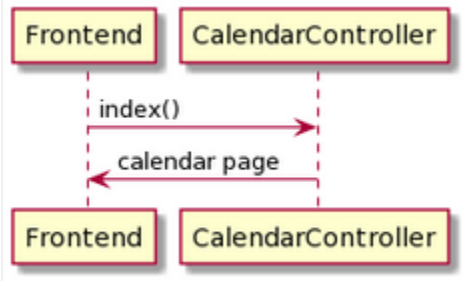
URL

```
Get /calendar
```

Response format
Code 200

```
return view('calendar');
```

Flow of Endpoint



Retrieve search function

Description
Retrieves the posts found when using the search function, in other words, retrieves the posts that contain the given text.

URL

```
Get /search
```

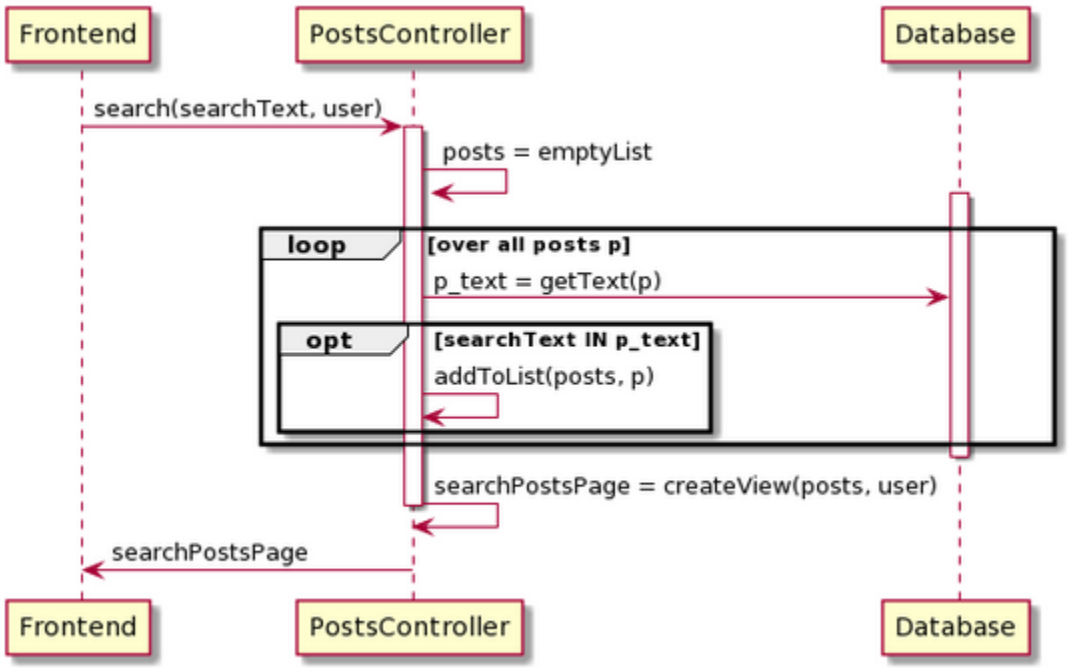
Response format
Code 200

```
return view('search', compact('posts', 'user'));
```

Code 200
Retrieving of user model as seen in a previous endpoint

Code 200
Retrieving of post model as seen in a previous endpoint

Flow of Endpoint



Retrieve viral posts

Description
Retrieves the posts found when using the search function, in other words, retrieves the posts that contain the given text.

URL

```
Get /viral
```

Response format
Code 200

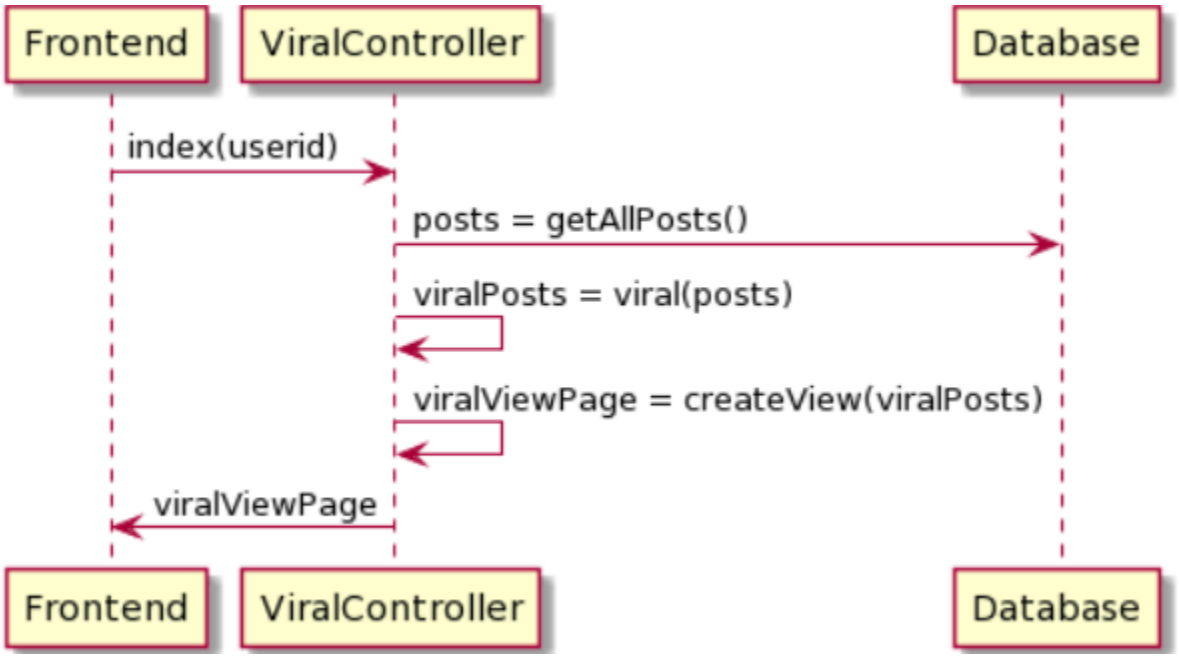
```
return view('viral', ['user' => $user, 'accounts' => $accounts, 'posts' =>
$posts]);
```

Code 200
Retrieving of user model as seen in a previous endpoint

Code 200
Retrieving of account model as seen in a previous endpoint

Code 200
Retrieving of post model as seen in a previous endpoint

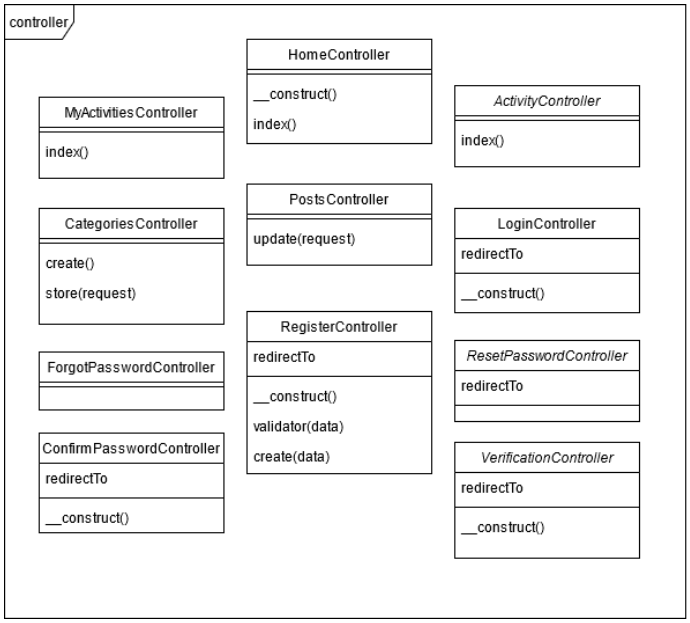
Flow of Endpoint

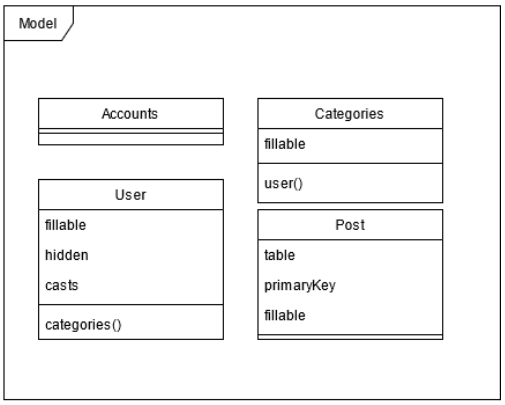


Overall Design Decisions

We are working with Laravel, which is a PHP-web framework. It is meant to be used for web applications that are based on the Model - View - Controller architecture. We are using this architecture.

Class Diagram

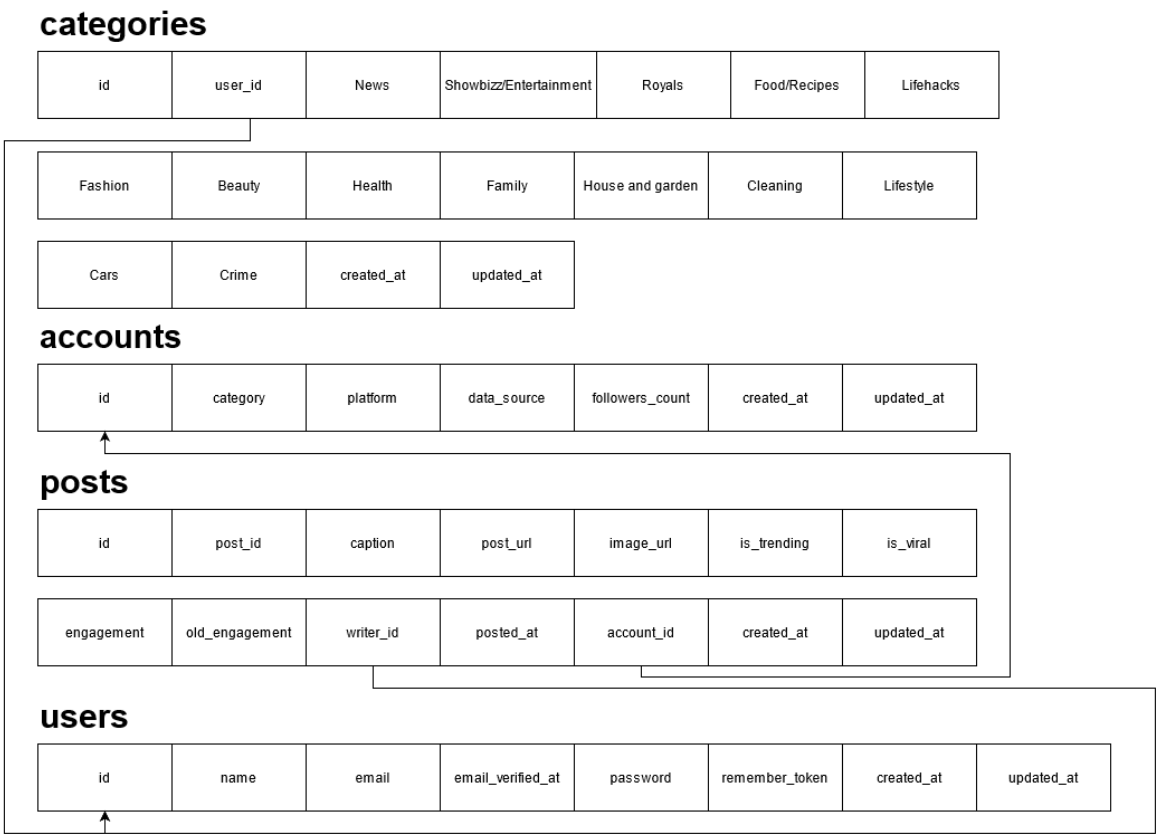




Back-end Design

Database Design

The main tables in our database are depicted in the following diagrams. We have decided to cut our longer tables in smaller bits and put them under each other so their text is readable, in our database they are of course one table.
In the categories table, we store the categories that the user would like to be shown in their social wall.
The accounts table contains information about the different accounts/sources from which we gather posts. For clarification purposes, one company (for example, Tasty) might have an account on each platform, in this case a company will have one account for each platform.
In the posts table, we store the necessary information to show the user of the social wall what the post is about and how much engagement it has.
The users table is a table generated by Laravel and it takes care of the authentication of users when they login.



The following tables in our database are necessary for using laravel, but we do not really affect our application.

password_reset

email	token	created_at

migrations

id	migration	batch

failed_jobs

id	uuid	connection	queue	payload	exception	failed_at

Data gathering

The data gathering for our application is done with the use of the Facebook Graph API, Twitter API and an Instagram Scraper. With the use of these API's and scraper we gather posts from a set of different sources/accounts. The data we gather from these posts is then stored in our database.

To make sure that the posts are kept up to date, we make use of a task scheduler in Laravel. This scheduler will run the data gathering process every ten minutes. When this scheduler runs every ten minutes, there will of course be posts that are already in the database. In this case we just update the engagement, old_engagement and updated_at columns of the post. The engagement column will contain the newly acquired engagement value, and the old_engagement column will contain the previous engagement value.

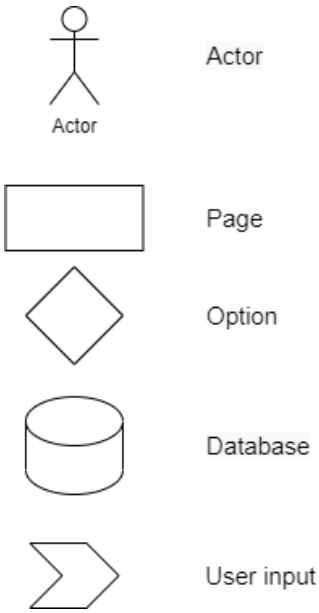
Front-end Design

The frontend consists of a desktop web application that the user can use. We use Laravel in order to create the view of this desktop web application. This is done via the blade template, in which the view of the page can be easily split up in sections.

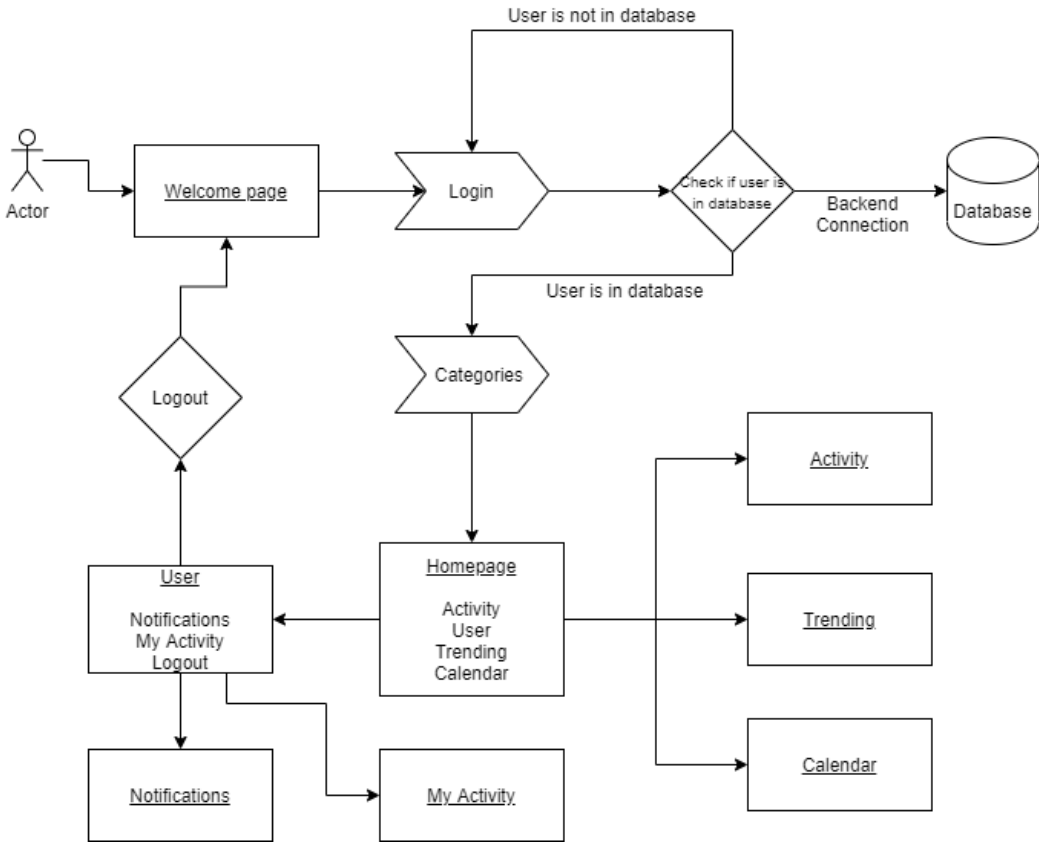
The first page to which the user is brought to is the welcome page. Next, the user has the option to log in to the application and it will bring the user to the login page. In case the user enters his/her correct credentials (checked by the backend), he/she will be redirected to the categories page, otherwise the user will get notified that he/she entered incorrect credentials. Once the categories have been checked off, the user will be brought to the homepage. The homepage consists of several sections and options. When the user is on the homepage, there is also the option to go to the trending page, the activity page, changing the categories and a dropdown section in which the user can either see his/her checked off posts or log out of the application.

In order to make this all a bit clearer, we will show a diagram corresponding to the above:

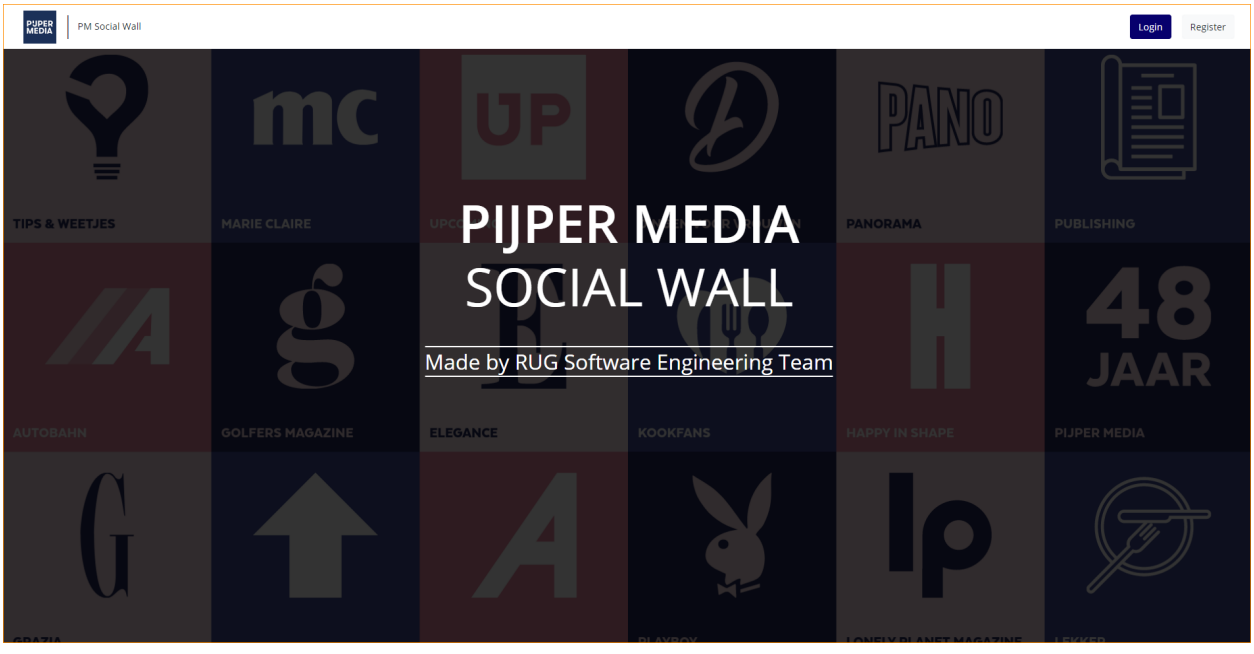
Attributes:



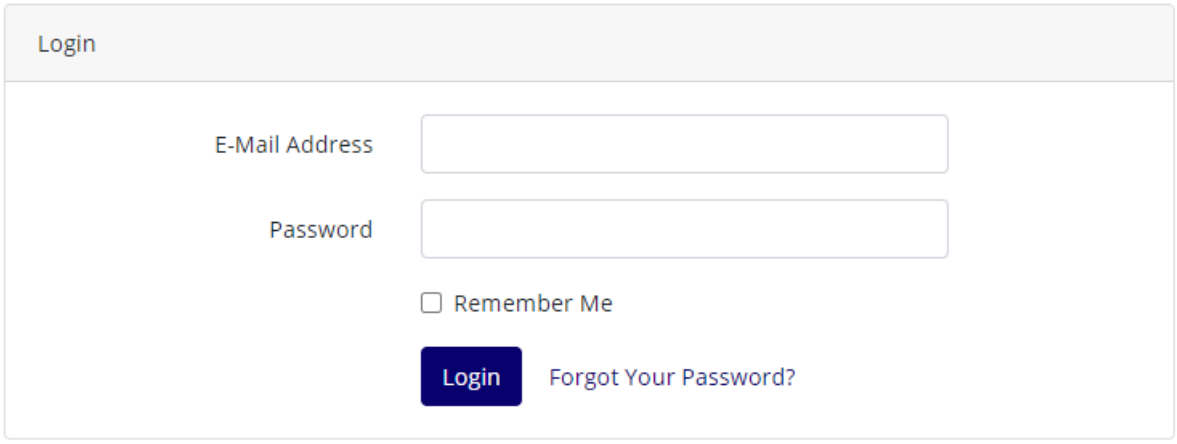
The diagram:



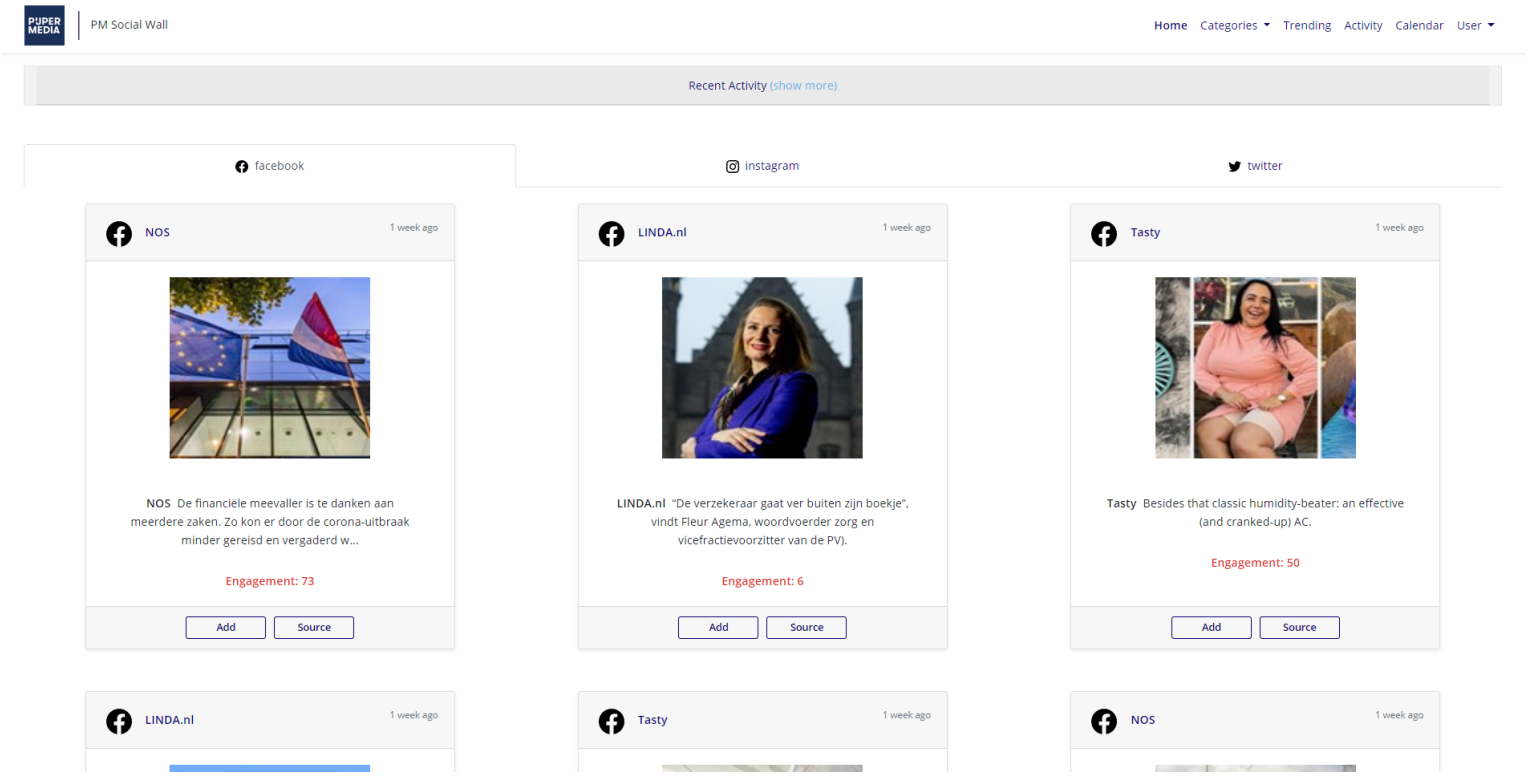
Visualisation of the most important pages:
Welcome page:



Login page:



Homepage:



Technology Stack

The programming languages that we used in this project are the following; PHP, Javascript, CSS, HTML and we used a MySQL database.

We will use the Laravel framework for the backend and the frontend. The reason why we use this technology stack is because our client works with the same technology stack and so it is their preference. Laravel is an MVC based framework and hence we use this framework as well.

MySQL is used as our database. Every post with its corresponding data is stored in the database and can be retrieved.

For the frontend, we decided to use Laravel as well, since the view can be easily created in Laravel. This is done via using HTML, CSS and Javascript.

Team organization

We have split the team up in the frontend, backend and the inbetween/connector. The frontend will be done by Julian Pasveer and Dilan Adel. The backend will be done by Daniël Scheepstra and Jeroen Klooster, everything will be connected by Medhat Kandil.

Change log

Date	**Changes**
Friday 05-03	First draft of the document
Sunday 07-03	Added team organization
Sunday 07-03	Added technology stack
Monday 29-03	Updated architectural overview (everything)
Monday 31-05	Rewrote a lot of unclear text and

	took the feedback into account
Tuesday 01-06	Added API Design and Endpoints