Requirements 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

1 Introduction

This project has been introduced to us by the family business Pijper Media. Pijper Media is a family business located in Groningen. It functions as a printing company, a magazine publisher and a large online branch. Pijper Media employs a total of 100 people, with headquarters in Groningen and editorial offices in Amsterdam. It consists of Pijper Druk, Pijper Publishing and WeBLog media. This project will be created for WeBlog media. The editorial team of WeBlog media would like to produce content based on viral posts to uplift the brands they publish content for.
These days almost everyone is in some way involved in social media. It could either be by texting each other or posting pictures on Instagram. Certain topics or posts go viral, which means they get a huge amount of attention. Our client would like a quick and easy overview of such topics which are trending/viral. To do this we're going to build a dashboard on which all posts will be shown and categorized for easy access. This dashboard is going to be shown on a desktop and optionally on a (non-interactive) tv screen.

2 Important definitions

In this document we will use the terms viral and trending, but these terms have a vague definition. To combat this, we will define how we use these words.
Viral is when something becomes popular in a really short timeframe (e.g. within 30 minutes).
Trending is when something is popular over a longer period of time (e.g. 3 days).

3 Actors

- **Main user:** The main users of the application will be the editors that write articles for Pijper Media.
- **Secondary user:** The other users for the application are the admins.

4 Requirements

Requirements listed in bold are requirements yet to be implemented.

1. Must have

R1.1. As an editor I would like to receive a pop-up or an email notification when a post is going viral, so I will be one of the first people to write about it.

R1.2. As an editor I would like to be able to log into my own account, so I will be able to personalize my own dashboard and be able to identify myself to the system.

R1.3. As an editor I would like to be able to click on a post to go to the source page, so I can easily get more information about the article/post.

R1.4. As an editor I would like to be able to see (article) posts from Facebook, Instagram and Twitter, so I can see which topics are trending/going viral.

2. Should have

R2.1. As an editor I would like to be able to check off a post I have written (or am writing) an article about, so there will be no duplication of work and this will make room for other posts to be shown.

R2.2. As an editor I would like to be able to remove some posts from my personalized dashboard if they are irrelevant, so I have a better chance of seeing relevant information for my work.

R2.3. As an editor I would like to be able to choose the categories that are displayed to me on my own dashboard, so I can look at the categories that are important for the work I deliver.

R2.4. As an editor I would like to have a calendar on which I can add and remove important (birth) days and events.

R2.5. As a user I would like to add/remove/edit specific Facebook, Twitter and Instagram sources and have the option to connect it to a category, which can be added/deleted/edited in a section of the admin tools.

3. Could have

R3.1. As an editor I would like to be able to search for a post/topic by name, so I can easily find the post I am looking for.

4. Won’t have

R4.1. As an editor I would like to be able to see (article) posts from Tiktok, Reddit, and Youtube, so I can see which topics are trending/going viral.

5. Non-functional requirements

R5.1. The code should be implemented in such a way that it is easily modifiable, which is important for the algorithm that determines popularity/virality.

MVP Requirements

In our Minimum Viable Product (MVP), we will make sure that the following requirements are met. This is to make sure that the project is going towards the way the client wants.

- As an editor I would like to be able to log into my own account, so I will be able to personalize my own dashboard and be able to identify myself to the system.
- As an editor I would like to be able to see (article) posts from Facebook and Instagram, so I can see which topics are trending/going viral.
- As an editor I would like to be able to click on a post to go to the source page, so I can easily get more information about the article/post.

5 Use Cases

In this section, we will elaborate on the following requirements:

1. Receive notification (R1.1)
2. Logging in (R1.2)
3. Navigate to source page (R1.3)
4. Viewing social media posts (R1.4)
5. Personalize dashboard (R2.3)
6. Check off post (R2.1)

Use Case 1 (R1.1): Receive notification

User Story

As an editor, I would like to receive a notification when a post gets above average attention[1], such that I always know when a post is viral.

Rationale/Context

The editors want to receive a notification in case a post gets above average attention[1]. This could either be via a pop-up or an email. If the editor is logged in, the editor will get the notification via a pop-up. However, when the user is not using the system, the user will get the notification via an email.

Frequency of Occurrence

A few times per week. Depending on the amount of posts that are getting above average[1] attention.

Primary Actor

The editorial team.

Preconditions

The User is a member of the editorial team and is authorized to use this use case.

Postconditions

The editor is notified via an email that a post is getting above average attention.

Main Success Scenario

1. The System checks if a post is getting above average attention.
2. The System finds a new post that is viral.
3. The System sends an email to the users of the application.

[1] See section 2 of this document

Use Case 2 (R1.2): Logging in

User Story
As an editor, I would like to log into my own account.

Rationale/Context
Editors need to log into their own account so they can add posts from the dashboard to their activity list.

Frequency of Occurrence
Every time the user opens the application.

Primary Actor
The editorial team.

Preconditions
The user is a member of the editorial team and has already registered themselves.
The credentials of the user are stored inside the database.
The user is on the login page.

Postconditions
The dashboard is loaded for the user to view. The dashboard is the view that every logged in user uses to view and save posts.

Main Success Scenario
1. The user is connected to the system.
2. The user enters their credentials in the login form.
3. The system validates the credentials.
4. The system determines the user’s role.
5. The system loads the dashboard view.
6. The user gets redirected to the dashboard.

Alternate Scenario
If in step 3 of the main success scenario the credentials do not match, in other words the user enters unregistered credentials in the login form, the system will return to step 2 of the main success scenario.

Use Case 3 (R1.3): Navigate to source page

User Story

As an editor, I would like to be able to view the source page of an article/post, this way I can easily get more information about the article/post.

Rationale/Context

Before choosing an article to write about, the editor wants to get more information about it. If an article looks like something they would want to write about, they should be able to go to the source page of the article for more information. This way they can find an article which they like, and they can write about it.

Frequency of Occurrence

Every time a user wants to (possibly) write about an article, during the working hours of the editors this will be about 10 times (for selecting a good article) per 20 minutes for every editor at work.

Primary Actor

The editor/writer

Preconditions

User is logged in, and is on the dashboard page.
There exists a source page to the article in the database.

Postconditions

The user is redirected to the source page of the article.

Main Success Scenario

1. The system gives the user articles/posts from which he can choose one.
2. The user indicates to the system the wish to see more information about a certain article.
3. The system sends the user to the source page of that article.

Use Case 4 (R1.4): Viewing social media posts

User Story

As an editor, I would like to see the timeline posts from the accounts on the social media platforms that are registered in the system.

Rationale/Context

On the dashboard, every post from the social media accounts we gather data from has to be shown. This way the user can easily see what posts are getting a lot of attention/engagement.

Frequency of Occurrence

Every time the user is on the dashboard page.

Primary Actor

The editorial team.

Preconditions

The user is logged in and is on the dashboard page.

Postconditions

The user views the posts on the page.

Main Success Scenario

1. The system fetches the posts saved in the database.
2. The system creates the new dashboard page, with the fetched posts.
3. The system loads the new dashboard page.

Use Case 5 (R2.3): Personalize dashboard

User Story

As an editor, I would like to be able to change the categories that are displayed to me on my personal dashboard.

Rationale/Context

Articles/posts can be sorted into different categories. Editors have different preferences as to what topics they like writing about. To show the most relevant articles we want them to be able to choose the categories that are shown to them.

Frequency of Occurrence

Once at the registering process and later it will probably be once every week.

Primary Actor

The editor.

Preconditions

The user is logged in and is on the dashboard page.

Postconditions

The preferred categories of the user are stored in the categories table in the database.

Main Success Scenario

1. User indicates that he/she wants to change his preferences
2. The system gives the user a list of categories
3. The user indicates to the system which of the categories he/she wants to see.
4. The system will store these preferences in the database.

Use Case 6 (R2.1): Check off post

User Story

As an editor, I would like to be able to check off a post in order to avoid duplication in the stories which are being written about.

Rationale/Context

The feed contains a lot of posts. These posts need to have an option to be removed from the feed once an editor decides to write about it. This is in order to make sure no article is being written about more than once.

Frequency of Occurrence

Several times a day

Primary Actor

Editors

Preconditions

The user is a member of the editorial team.

Postconditions

The post is removed from the feed and added to the “written about” section.

Main Success Scenario

1. The User is on the dashboard page.
2. The User indicates to the system which posts he/she wants to write about.
3. The System removes the post from the feed.
4. The System marks the post as “being written about”.

6 Traceability matrix

Requirement	Module	Files affected	Test	Passed	Comment	Description
R1.1 Receive Notification	DB Front-End Backend	trendingCheck.php Viral.blade.php ViralController.php	Manual check	Yes	Apple software accepts Bootstrap, but, in browser email clients do not. (Security)	Email Trials on running program, scheduling every 10 minutes, trending posts are both the same on the platform and the email client.
R1.2 Logging in/ registering	Front End	LoginController.php login.blade.php	LoginTest.php RegisterTest.php	Yes	Some pages could be accessed at first without logging in. We took care of it.	Secure pages cannot be accessed without logging in. Users are logged in to their account on demand with correct credentials.
R1.3 Navigate to source page	Front End	home.blade.php	RedirectTest.php	Yes	No comments here	When something is clicked, you get the desired action. Meaning: when the user clicks on ‘source’ in a post he goes to the page correctly.
R1.4 Viewing social media posts	Back End	HomeController.php	Manual check	Yes	No comment here	Authenticated user gets all posts in his category list and nothing more/less. Authenticated user can add/remove

					As long as the instagram scraper is working, the user can see posts from all three platforms.	posts to his activity and others can see on their accounts. Authenticated user can see posts from: Facebook Twitter Instagram
R2.1 Check off post	Back End DB Front End	PostsController.php facebook/instagram/twitterData.php Activity.blade.php myactivity.blade.php	PostTest.php DBTest.php	Yes	Problems were occurring with a bug where a person always sees “You are writing about this post” instead of the name of the real writer, and the bug was fixed.	When a user adds a post to his activity it gets assigned to his user ID in the database with no overlaps.
R2.3 Personalize dashboard	Back End DB	CategoriesController.php UserModel.php	CategoriesTest.php DBTest.php	Yes	Categories can also be changed upon registration.	When a user registers and chooses categories, they are marked as TRUE in his database row
R2.4 Calendar of Birthdays with Add/Remove functionalities	Front End	calendar.blade.php	Manual check	Yes	Added birthdays are gone on reload.	Birthdays (chosen by client) can be found on the calendar. Can add/remove Birthdays
R3.1 Searching in the database	Back End Front End	SearchController.php search.blade.php	SearchTest.php	Yes	Search Engine is not a smart one but executes the queries as specified by the	When a phrase is searched, only exact matches are returned and they are only

					client.	searched for in the caption/message of the post.
Other Requirements						
R.Other.1 Scheduling of database updates	Back End DB	PostsModel.php Kernel.php twitterData.php instagramData.php facebookData.php	Manual check DBTest.php	Yes Yes	We have problems with the instagram scraper getting us blocked. Moreover, we're doing trial and error with facebook and twitter since the numbers they gave on their websites are not accurate.	Database is repopulated every 10 minutes. API calls are not surpassed.

7 Meeting log

Date	**Discussed**
Wednesday 10-02	Kick-off & introduction
Tuesday 16-02	Functional Requirements
Friday 19-03	Technical Meeting
Tuesday 23-03	Live Demo and Question/Answer session
Tuesday 01-06	Live Demo and Question/Answer session

8 Changes block 1 -> block 2

In block 1 we had a working website with a welcome page on which the user could register and login. After the user logged in, he got redirected to a page on which he selects his preferred categories. Then he gets redirected to the dashboard on which facebook posts can be seen. The activity section is also shown on the dashboard. The activity page and the my_activity page are both implemented as well. All necessary information about the posts could be seen on the card of each post and the user could click on the source button to be redirected to the source page of the post. The dropdown menu for the categories preference was also implemented at this time.

In block 2 we improved our application a lot. The first thing we decided to do was update our database, because our database had a lot of redundant data. After that, we made sure that we could gather posts from Instagram and Twitter aswell. Doing this for Twitter was easy, but Instagram gave us a lot of trouble because we use a scraper for gathering posts. Once we fixed this part of the application, we created a search bar so the user can easily find certain posts. After that we created an algorithm that checks if a post is trending and another one that checks if a post is viral. We also added a calendar page, on which the important (birth) days are shown.

9 Change log

Date	Changes
Friday 05-03	Cleared up points said in feedback.
Sunday 07-03	Specify which requirements are going to be featured in the MVP.
Sunday 07-03	Continue: Clear up points said in feedback.
Saturday 27-03	Change some keywords according to the meetings and put the unimplemented requirements in bold.
Tuesday 18-05	Added Use Cases. Cleared up points given in feedback.
Friday 21-05	Added Use Cases.
Monday 07-06	Cleared up some points of feedback
Friday 11-06	Added traceability matrix
Saturday 12-06	Added changes block 1 -> block 2
Sunday 13-06	Finalized traceability matrix