

# Assignment 2: Project Plan

Team 8C: Cristian Ciacu, Diana Micloiu, Matei Mirica, Vlad Nitu, Rares Toader

May 4, 2023

## Contents

<b>1</b>	<b>Problem Analysis</b>	<b>3</b>
1.1	Problem statement . . . . .	3
1.2	Stakeholders . . . . .	3
1.3	Use cases . . . . .	3
1.4	Already existing products . . . . .	4
<b>2</b>	<b>Feasibility Study &amp; Risk Analysis</b>	<b>5</b>
2.1	Operational feasibility . . . . .	5
2.2	Technical feasibility . . . . .	5
2.3	Economic feasibility . . . . .	6
2.4	Risk analysis . . . . .	6
<b>3</b>	<b>Requirements</b>	<b>7</b>
3.1	Requirements elicitation . . . . .	7
3.2	Requirements prioritization - The MoSCoW Method . . . . .	7
<b>4</b>	<b>Project approach</b>	<b>9</b>
4.1	Software Architecture . . . . .	9
4.2	Technology chosen . . . . .	9
4.2.1	Overview . . . . .	9
4.2.2	Motivation . . . . .	10
4.3	Obtaining news articles . . . . .	10
4.4	Testing for desired behavior . . . . .	10
4.4.1	Overview . . . . .	10
4.4.2	Pipeline . . . . .	11
<b>5</b>	<b>Development methodology</b>	<b>11</b>
5.1	Scrum . . . . .	11
5.2	Feature Branches and Merge Reviews . . . . .	12
5.3	The Definition of Done . . . . .	12
5.4	Tools . . . . .	12
5.5	Communication . . . . .	13
<b>6</b>	<b>Appendix</b>	<b>14</b>
6.1	Merge Request Template . . . . .	14

6.2	Issue Template . . . . .	15
6.3	UI Figma diagram . . . . .	15

# 1 Problem Analysis

In the current times, plagiarism has gained a greater significance in the context of academic papers or news articles, due to the major advancements in the field of artificial intelligence and the launch of various chatbots. Plagiarism can be expressed as the action of incorporating someone else's effort or ideas into your own, without proper acknowledgment, regardless of whether you have obtained the original author's consent or not. It can take different forms, for example, verbatim quotation without proper and explicit attribution or acknowledgment, paraphrasing, or even inaccurate citation.

## 1.1 Problem statement

For the project "News article overlap detection platform", we were asked to come up with an approach to combat the rapidly expanding issue of plagiarism in the news industry. The core points the client wants us to deal with consist in finding the overlap between articles.

From these base points, we would need to gain several insights into the problem. One important understanding we need to acquire in order to solve this problem is: what confidence level is required to determine whether an article is part of ethical journalism or not. Moreover, we can gain information about text ownership, to find out who plagiarised from who.

## 1.2 Stakeholders

As in any project, stakeholders play a key role in developing a project, as they can get involved in the decision-making process and influence the organisation's actions in a way that is helpful to the project management team. In the context of our project, we have identified six main stakeholders: the client, readers, journalists, and last but not least, researchers.

1. **Client** - our main stakeholder is the client, as they have commissioned us to develop the "News article overlap detection platform." By meeting the needs of the client, we can ensure the success of the project and deliver a valuable service to the media industry.
2. **Readers** - are considered to be the ultimate beneficiaries of our project, since it can help them to trust the media and stay informed about important events that happen around them, by ensuring that online news articles are original and accurate.
3. **Journalists** - are responsible for researching, and creating news stories that are original and engaging, however, it can be challenging to avoid overlaps with other articles, especially when working under tight deadlines. And here comes our tool for help, allowing them to check their work for overlaps and improve the quality of their articles.
4. **Researchers** - are interested in advancing the state of knowledge in the media industry. Our platform can contribute to this goal by providing a valuable dataset for researchers to analyze and identify trends in news coverage.

## 1.3 Use cases

The main goal of this project is to create an approach to combat the issue of plagiarism in the context of news articles. That being noted, below, there are some use cases for the platform we are developing.

1. **News Aggregation** - the platform can be used by a search engine to ensure that when a user searches for an event, it will not display plagiarised articles. This can improve the user experience by providing only relevant and original information.
2. **Plagiarism detection** - it can help identify instances of plagiarism in news articles by efficiently comparing them with other published articles from the web. It can be useful in the field of media, where publishers can ensure the integrity of their content and avoid legal issues related to copyright infringement.
3. **Article ownership** - the tool can be exploited to determine the ownership of a given article, by comparing it with other similar candidates with regards to the publish date of each article.

## 1.4 Already existing products

There are already many other tools available online that can help identify plagiarism given a specific piece of text. However, the issue of plagiarism in news articles is not widely understood, and this is where our platform aims to provide assistance. In the following section, we will enumerate several well-known tools that can be used for detecting overlap between documents. For each such tool, we will provide a description of its features, as well as an analysis of its advantages and disadvantages.

1. **Turnitin** - is a web-based software application used to detect plagiarism in academic writings. It is primarily used by educational institutions to help teachers identify instances of plagiarism in student work. When a student submits a paper to Turnitin, the software compares the text against a vast database of published works, as well as papers previously submitted to the system, to identify any matching text. Turnitin then generates a “Similarity index”, which indicates the percentage of matching between the submitted paper and other sources, and also an “Originality report”, which demonstrates all these matches in detail, including the sources found.

While Turnitin has many benefits, it also has an important downside in the context of privacy concerns, that is, students might have concerns about the privacy of the submitted documents, as Turnitin saves documents in its database and shares them with other institutions that use the software.

2. **Grammarly** - is a popular writing assistant tool that provides a plagiarism checker feature. The plagiarism checker scans text for any instances of plagiarism or potential copyright violation by comparing it to a database of online sources. Although Turnitin offers the same feature, Grammarly is intended to be used mostly by individuals.

One important advantage of using Grammarly is that it offers a user-friendly interface, as well as it generates quick and accurate plagiarism reports, which can be understood by any type of user. On the other hand, the main disadvantage of Grammarly is the lack of access to gated or academic content, making it not a viable solution for researchers or teachers.

After carefully examining the functionality of each of the above-mentioned plagiarism checkers, we concluded that we are going to consider them as a model in terms of user experience. Each of those tools displays a very intuitive interface where users of all skill levels can easily gain insight into the plagiarism level from their submitted document. On top of that, the interfaces are visually appealing, with clean layouts that are easy to navigate. To make it more concrete,

Grammarly's plagiarism checker is designed with a user-centric approach, with a focus on making it straightforward for users to access and use its features and this represents a key attribute we are going to follow when designing the user interface for our platform.

## 2 Feasibility Study & Risk Analysis

### 2.1 Operational feasibility

After carefully assessing the necessities of the client against the given timeline and resource constraints, we strongly believe that given the expertise of the 5 members of the team, completing the requirements we have come up with is achievable.

By conducting a thorough requirements elicitation process and prioritizing them using the **MoSCoW Method** (see Section 3), we consider that the Must-haves will be ideally finished within  $\frac{3}{4}$  of the project leaving room for the Should-haves to be implemented as the additional functionality provided by them will enhance the project's overall usability.

Nonetheless, given the time constraints and the way the project is structured, a handful of the issues regarded as Could-haves will most likely be implemented. They are seen as stretch goals by the team and have low priority in the client's perspective and therefore are set to be completed within the time frame only if it is possible.

Therefore, we believe that despite the significant expected codebase and thorough implementation details, this project has a suitable complexity to be approached during an academic quarter. Coming up with functional software that meets the basic and most important requirements of the client is within reach. We are also well-positioned to expand and improve the product until the end.

### 2.2 Technical feasibility

Given the context of the software project, the technical feasibility study is strongly related to the technical environment provided by the company and also by TU Delft University.

With that, the entire development process will be recorded and sustained by using the GitLab Faculty service. This will help the team coordinate, stay on track and organise better. Yet, it is also a great way for the supervisors to see the progress of the whole project throughout the quarter.

The company will also have a large input regarding the software we are provided with for better development of the given project. Therefore, the resource constraints will not be an issue, as the client will provide us with a web crawler for extracting the content of the articles given their URLs.

Also, the project is not expected to be computationally intensive, besides the large size of the database which will be handled by choosing an appropriate architecture and DBMS. Thus, there is no need for high-performing hardware to back up our implementation.

Our team members also have experience in the programming languages that are going to be used: Python for the backend and JavaScript for the front end. This will facilitate the development of the software within the timeframe. Additionally, our team has experience in

some of the frameworks and tools that we are going to use, such as React, allowing us to come up with a reliable and efficient product.

Lastly, we have accounted for the extra time which will be required to research plagiarism-detection algorithms and to get accustomed to the frameworks. This will enable us to develop highly efficient and accurate copy-detection software.

## 2.3 Economic feasibility

By analyzing the overall cost in comparison to the benefits that our product yields we came up with the following feasibility observations. Since developing the software only makes use of open-source resources, the overall “production” cost will be zero. However, if we take into consideration the time and effort put into the project, it is clear that there will exist an overall “cost”. That being said, we strongly believe that the cost of creating such software will pay off in its usability. The trade-off makes sense since the current society confronts daily the problem of misinformation in the media. Since news articles are part of every citizen’s daily life and many times also influence local movements or life decisions of the individuals, it is clear that our product will make a significant impact in society and, therefore, it is worth the investment.

## 2.4 Risk analysis

We have analyzed all the risk factors we were able to come up with and tried to reduce them to the minimum:

**Client involvement:** We have agreed with our client to host weekly online meetings each Friday at 4 PM in order to keep him on track with our progress and discuss any feedback and suggestions they may have. Therefore, we do not anticipate the need for the continuous availability of the client during the 10 weeks. Also, if necessary, they may address questions we have via email, and we expect a response within one week.

**Hardware requirements:** We do not expect the need for any dedicated hardware, as given the size and complexity of the project, we expect it to be deployed on our own machines. However, in the unfortunate situation of any of the team members facing any difficulties (i.e.: lost/damaged laptop), we are going to rely on the PCs that are already available in campus buildings such as Echo or EEMCS. Therefore, the hardware risk is minimal, foreseeable, and solvable on short notice.

**Legal issues:** The only potential legal issues we may encounter are privacy constraints due to ownership/copyright of data. Fortunately, this can be mitigated by storing a fingerprint of the articles, not the plain text version. We will not face any data protection / personal data issues, as users will not need an account to use the application. For computing the fingerprints, we will need to crawl a news website. Our crawler will also try to obtain the data via RSS feed. Another problem that might occur would be accidentally causing DDOS attacks, which we will try to remedy by only allowing an URL to be crawled every 30 seconds.

**Integration:** The application will be isolated from the client’s codebase, so integrating our code into their already existing one is not an aspect of concern. Nonetheless, we have opted to use some of the technologies used by the client. This will facilitate future integration of our software into their codebase if they choose to after we manage to deliver a successful product.

**Background knowledge:** We are aware that not all of us are familiar with all the technologies which will be used during the project, but we are confident that this will not impose any obstacles in our way. Firstly, we can rely on our internal team for any questions that may arise. Secondly, we are continuously researching the tools required to build the article overlap detection application, which gives us confidence in our ability to successfully complete the project. The selected tools are well-documented and have a large and supportive developer community behind them, assuring us that we can overcome any challenge we may face during development.

To conclude, we are confident that the presented risks will not affect us in any way during these weeks and we will be able to deliver a successful project.

## 3 Requirements

### 3.1 Requirements elicitation

The requirement elicitation process we conducted had as its primary focus the description of the project given by the client. Thus, at first, we individually analysed the project overview and formed a general understanding of the important aspects of the project. Afterward, we had a brainstorming session in which we gathered all of our ideas and created an overall concept of the software in question. This step was followed by two interview meetings with the client in which we presented the requirements we came up with, got feedback from the client, asked questions and, lastly, improved our skeleton of the project so that we ended up on the same page and with a deeper understanding of how the prototype should look like.

### 3.2 Requirements prioritization - The MoSCoW Method

We have used the MoSCoW model to derive the requirements and prioritise them from the most to the least important ones, allowing us to come up with the following:

#### **Must have requirements**

1. As an admin, I want to make use of a plagiarism detection algorithm in order to detect overlapping news articles.
2. As a user, I want to have a main page that includes a menu, so that I will have the ability to access the different features of the web application.
3. As a user, I want to have a special option in the menu so that I will have the ability to insert a URL of a news article in a given text box.
4. As a user, I want to enter the link of a news article, so that I will be given a plagiarism warning if the article was plagiarized.
5. As a user, I want to enter the link of a news article, so that I will get back similar articles, together with their respective URLs.
6. As a user, I want to enter the link of a news article so that I will get back similar articles, together with their plagiarism similarity ratio.
7. As an admin, I want each URL that is inputted by the user to be processed and persisted in the database, in order to be used in future plagiarism checks.

### Should have requirements

1. As a user, I want to have a special option in the menu so that I will have the ability to insert a piece of text from a news article in a given text box.
2. As a user, I want to enter a piece of text of a news article, so that I will be given a plagiarism warning if the article was plagiarized.
3. As a user, I want to have a special option in the menu, so that I will have the ability to insert two URLs of news articles in two different text boxes.
4. As a user, I want to have a special option in the menu, so that I will have the ability to insert two pieces of text of news articles in two different text boxes.
5. As a user, I want to check whether the two URLs that I have inputted are plagiarised.
6. As a user, I want to check whether the two pieces of text that I have inputted are plagiarised.

### Could have requirements

1. As a user, I want to enter the URL of a news article, so that I will be given at most 10 articles in decreasing order of their plagiarism ratio.
2. As a user, I desire a system that checks the originality of an article based on its publication date, with the aim of obtaining ownership of the article in question.
3. As a user, I want to be able to input two news article URLs in the given boxes and run the program to see highlighted parts of the news that have high plagiarism similarity.
4. As a user, I want to provide a similarity threshold to set a plagiarism sensitivity, and in turn be shown all news articles' URLs in the database that have similarities above that threshold.
5. As a user, I want to enter the URL of two news articles, so that I can be able to see a rendered side-by-side comparison of the two web pages.

### Won't have requirements

1. As a user, I want to input a URL of a news article and be shown a sentiment analysis of it.
2. As a user, I want audio files in news articles to be included in plagiarism detection.
3. As a user, I want the images to be considered in the check for plagiarism.
4. As a user, I want to sign-up on the platform in order to see my previous checks.
5. As a user, I want to be provided with a web extension of the software compatible with the search engine that I am using.

### Non-functional requirements

1. As an admin, I will implement the system's backend using Python's Django 4.2 framework.
2. As an admin, I will implement the system's frontend using JavaScript's React v16.0 framework.



3. As an admin, I want the repository to be hosted on GitLab and a GitLab CI/CD pipeline should be set up to run every time a team member creates a merge request on the platform.
4. As an admin, I want to augment the development process using GitLab dedicated features, such as Issues, Milestone boards, etc.
5. As an admin, I want the system to be pre-tuned against a database containing 100k articles.
6. As an admin, I want to extract all relevant information from a news article using an open-source web crawler (news-please), so that I can accurately inspect a news article for plagiarism.

## 4 Project approach

In order to solve our client's problem, we are going to develop a full-stack application that heads toward article overlapping, which uses a state-of-the-art Data Mining algorithm for detecting plagiarism in the context of news articles.

As the amount of data we are targeting our application to work with is expected to be massive, it would be infeasible to store all the news articles in plain text. To combat this issue, we will use a cutting-edge *document fingerprinting algorithm* called *winnowing* [1], which satisfies the three properties every copy-detection algorithm should fulfill:

1. Whitespace insensitivity
2. Noise suppression
3. Position independence.

To minimize the number of comparisons so that we can keep our approach scalable and fast, we will drop irrelevant articles in a *pre-filtering* step similar to the one explained in "3.3.4. Pre-filtering" [2] by using a *Signature Tree* [3] data structure.

### 4.1 Software Architecture

We consider that the most suitable architecture for our application would be the "Client-server architecture" model, which implies structuring our project in two modules: the client (frontend submodule) and the server (backend submodule)

The backend of our software system will provide access to its functionality and data through a RESTful API that the frontend side will consume via HTTP requests.

### 4.2 Technology chosen

#### 4.2.1 Overview

We are going to use Django [4] for implementing the backend server and also make use of Django REST framework (DRF) [5], as it provides a robust API layer built on top of Django (meaning that it integrates well with any other Django libraries and packages). On the frontend side, we agreed on using React [6] for styling our UI (User Interface), and also to consume the APIs on the backend server via HTTP requests. This communication will

be established via the Fetch API, as it is a simple (but less robust) way of executing HTTP requests, while also offering asynchronous functionality handled through promises. (Note that we also considered using the Axios third-party library, but anticipated that we won't need all of its complex features)

#### 4.2.2 Motivation

We decided to implement our system using this specific tech stack configuration as both React and Django are highly favored in their respective domains. Consequently, they have large community support and provide immediate assistance. When it comes to DjangoREST, it makes serialization (i.e: convert complex data to native Python data types, which will be then rendered to JSON that will be used on client-side by React) easier; removes the extra work on the developer side to write all the boilerplate code to create HTTP requests by abstracting this in its "APIView" library, thus the developer gets to dedicate more of his/her time crafting actual logic/functionality of the application.

### 4.3 Obtaining news articles

As our project is intensively data-driven, we need to carefully consider how we are going to obtain this data and make sure that the dataset we are going to work with is not only qualitative but also quantitative (as our algorithm is designed for production usage, that is: it should process billions of news articles and compare some of them pair-wise). Thus, we will use "news-please" [7], an open-source web crawler we opted for that was initially recommended by our client, which also used it in building previous products. Moreover, it is capable of both crawling news data and also extracting information from it.

Furthermore, the API integrates with [commoncrawl](#) workflow, which provides convenient methods to filter the results such that we download only news articles we are interested in (and hence reducing the number of disk accesses).

While researching the alternative web crawlers, we considered [Trafilatura](#), which only provided a CLI, so it did not satisfy our main needs: a library that integrates with our Django backend. Another ubiquitous alternative was "[newspaper3k](#)", which is not maintained anymore, so this may have harmed our development process if we found ourselves stuck on some bug related to the library. Moreover, Hamborg et al. (2017): elicit the following drawbacks related to "newspaper3k": "lacks full website extraction, auto-extraction of new articles, and news content verification, i.e. determining whether a page contains a news article".

In our research on what open-sourced web crawler to use, we also analyzed this "[Evaluation and alternatives](#)" section, which emphasized that the framework we chose yields one of the best performances.

### 4.4 Testing for desired behavior

#### 4.4.1 Overview

We will conduct our application's testing process by following the "Testing Pyramid" studied in CSE1110: Software Quality and Testing course, which means: prioritizing unit tests (by validating the behavior of individual components on our frontend and functions from our backend), next moving on to integration tests (for example: validating the interaction between different parts of Django backend, or checking if the communication between frontend and

backend is indeed returning expected results). Lastly, we will write some end-to-end (E2E) tests that will simulate the user's interaction with our software.

#### 4.4.2 Pipeline

The CI/CD pipeline script will automatically run all these tests for each merge request created. We decided on configuring a single GitLab CI/CD pipeline for both the backend and frontend of our application (see `.gitlab-ci.yml` file in the root directory), as it is easy to set up and eliminates the burden of setting up a different system for the Continuous Integration process (i.e: setting up a CircleCI pipeline), easy-to-learn, scalable and has high compatibility with GitLab (i.e: GitLab has already implemented features that allows you to Lint, Edit and Validate your CI/CD document directly in their web browser, you can simulate pushes on different branches to observe how the pipeline will perform if you were to push the changes, etc.).

## 5 Development methodology

### 5.1 Scrum

There are several development methodologies that have been described over time by developer teams and we chose to stick with the Scrum methodology [8], for the main reason being that we have already been using it for two projects, and we felt like it lead to very good results. To further describe how we integrated Scrum within our team, here is a short list of the concepts [9] we have implemented in our workflow:

1. **Sprint planning** - we organise ourselves in a meeting, with the entire team, in order to distribute tasks that need to be solved by the end of the upcoming iteration. The tasks can range from specific items in the backlog, or to one of our issues in the issue board. This meeting is led by one of us, who has been appointed as the scrum master. By the end of the meeting, each team member should have a clear understanding of what needs to be delivered by the end of the sprint.
2. **Sprint goal** - during the sprint planning meeting, the team agrees on a goal for the end of the iteration. This goal is closely related to the tasks that have been distributed for the upcoming sprint. An important thing to note is that the sprint goal should be feasible to reach by the end of the iteration.
3. **Sprint** - is one of the core concepts surrounding our development methodology. We have short, iterative sprints that last around one week, and this is where the main bulk of the work happens. During this period, each team member works on the tasks they have been assigned, and hopefully, by the end of the sprint, the sprint goal is met.
4. **Sprint retrospective** - is a short meeting where the team gathers to discuss the challenges they were faced with during the sprint, related to tools, coding, or other problems. The whole point of this is to know what went well so that the team keeps doing it, and to also acknowledge the hurdles encountered, in order to avoid them in the future.
5. **The daily scrum** - is a moment where each one of us shares what we are working on or what we have accomplished so far during the sprint. One of the main goals for this is to keep each other in the loop, in order to make sure the team is focused on the sprint goal. This is also a good occasion to raise an alarm if something is not going well, so that it can be fixed by the team and that the sprint productivity is not hurt.

## 5.2 Feature Branches and Merge Reviews

Working with feature branches [10] enables the members of the team to work on different features at the same time, avoiding the problem of conflicting with what others are changing. Feature branches also allow for isolating new features, making it also easier to test and validate them before merging them into the main development branch.

Whenever new functionality is added, a merge request from the feature branch to the main development branch needs to be opened. We have agreed that in order to accept the new functionality, at least two team members need to review that merge request. For reviewing, we have chosen a merge review template, that we have included in the Appendix. We wanted to follow some of the best practices related to code reviews [11]. To approve a merge request, we have agreed on a few acceptance criteria, inspired from [12]:

- the changes in the MR should be as small as possible;
- all changes made in the MR should be properly tested and documented;
- the MR should pass the CI/CD pipeline that we have set up;
- the MR contains functionality that other users will benefit from;
- only one specific issue is fixed or one specific feature is implemented in the MR;
- the MR respects the definition of done explained below.

## 5.3 The Definition of Done

The definition of done represents the criteria that must be met for a task, issue, or feature to be considered complete. It usually comprises certain expectations and quality standards that must be checked before a change is released.

The following list displays the checklist for our definition of done:

- The code is properly tested and documented
- The code has been thoroughly reviewed.
- The changes pass our CI/CD pipeline.
- The implementation is manually validated to ensure that it fulfills what the feature asked for.
- The changes do not introduce regression bugs (errors to previously deployed functionality).

## 5.4 Tools

We are using several tools for the development of our project. We are using Git [13] as our version control system, on GitLab, which will be our DevOps platform where we will deploy our code. We are going to be using *pip* and *npm* as package managers. For CI/CD are going to be using GitLab's CI/CD, as it is very convenient because our project will also be deployed on GitLab. To plan and organise we are going to be using mainly Google Docs/Sheets and GitLab boards. For creating some preliminary wireframes of our web application, we will use Figma, PlantUML and Lucidchart. We will likely use Heroku for hosting our web application.

## 5.5 Communication

Our strong belief is that good communication within a team is one of the key factors in the success of a project. Besides the daily scrum, we steer away from having too many meetings, and when we need to, we like to keep them short so that we do not lose productivity. As main communication channels, we are using Mattermost to keep in touch with our TA, and Discord for our daily matters.

For making the communication within the team consistent, we have a few documents where we track important information related to the project. For instance, we have a Google Sheet, where we include all future meetings (with dates) that we will have. Another example would be a shared Google Doc where we prepare questions for a future meeting with the coach, client, or TA. This ensures that the whole team is on the same page related to organisation.

## References

- [1] Saul Schleimer, Daniel Wilkerson, and Alex Aiken. “Winnowing: Local Algorithms for Document Fingerprinting.” In: *Proceedings of the ACM SIGMOD International Conference on Management of Data* 10 (Apr. 2003).
- [2] R. Schellenberger. “Plagiarism detection by similarity join.” MA thesis. Aug. 2009.
- [3] Yangjun Chen and Yibin Chen. “On the Signature Tree Construction and Analysis.” In: *Knowledge and Data Engineering, IEEE Transactions on* 18 (Oct. 2006), pp. 1207–1224.
- [4] *Django Web Framework*. <https://www.djangoproject.com/>.
- [5] *Django REST Framework*. <https://www.django-rest-framework.org/>.
- [6] *React framework*. <https://react.dev/>.
- [7] Felix Hamborg et al. “news-please: A Generic News Crawler and Extractor.” In: (Mar. 2017).
- [8] *Scrum Methodology*. <https://www.atlassian.com/agile/scrum>.
- [9] Jeff Sutherland. “Business Object Design and Implementation IV: From Business Objects to Complex Adaptive Systems OOPSLA’98 Workshop Report.” In: (Jan. 1998). DOI: 10.1145/346852.346953.
- [10] Earl T. Barr et al. “Cohesive and Isolated Development with Branches.” In: *Fundamental Approaches to Software Engineering*. Ed. by Juan de Lara and Andrea Zisman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 316–331. ISBN: 978-3-642-28872-2.
- [11] *Code reviews*. [https://docs.gitlab.com/ee/development/code\\_review.html](https://docs.gitlab.com/ee/development/code_review.html).
- [12] *Merge request workflow*. [https://docs.gitlab.com/ee/development/contributing/merge\\_request\\_workflow.html](https://docs.gitlab.com/ee/development/contributing/merge_request_workflow.html).
- [13] *Git*. <https://git-scm.com/>.

## 6 Appendix

### 6.1 Merge Request Template

#### Merge Request

---

- This merge request closes/relates *specify issue number* which refers to *description of the merge request + include definition of done*

#### Checklist

---

- ☐ Do you have sufficient test coverage (if applicable)?
- ☐ Did you fix all style errors?
- ☐ Did you achieve the definition of done?
- ☐ Are you merging to the right branch?

#### Details


---

*Please specify any details regarding your merge request*

#### How to test

---

*Please specify how reviewers should test your functionality*

 [image](#) (if you just CTRL+V an image here it automatically uploads it and gets the link)

## 6.2 Issue Template

### Details

---

- *Description*
- *Definition of done*
- *REMINDER: Include related / blocking issues*

### Definition of Done

---

- ☐ TODO list item 1

## 6.3 UI Figma diagram

Follow this [link](#) to see the full Figma diagram.

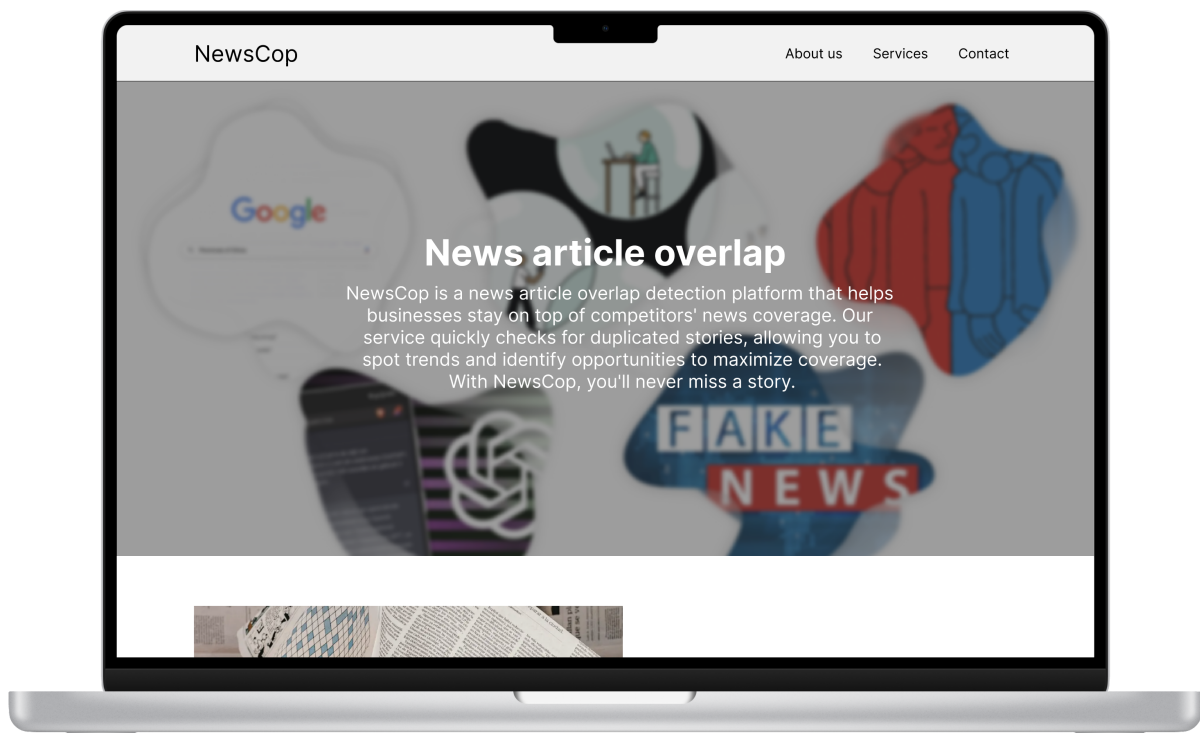


Figure 1: UI Figma diagram