Sprint retrospective, Milestone #6

User story	Task	Assignee/s	Estimated effort	Actual effort	Done	Notes
Populate the database until 10-100k and test whether the algorithm is feasible on such amount of data	#68	Vlad Nitu & Rares Toader	5h	2h30m (Vlad) + 2h30m (Rares) = 5h	Completed	Stopped from populating the database from 10k articles in previous MongoDB storage. Implemented #80 and then populated a local PostgreSQL database up to 52.5k documents (at the moment of writing this document). Continuously profiled both memory and efficiency of queries. For all this information, see "Benchmarking" category from Migrate MongoDB -> PosgreSQL doc: Reorganised the structure of our data (as now we use 3 tables, URL, Fingerprints and a relatonal database that defines the relation between the first two). Other alternatives were considered, but were not implemented as those were estimated to be memory infeasible (See #80 "Strategy 2" section for full calculations)

Migrate database (previously MongoDB) to PostgreSQL	#80	Vlad Nitu & Rares Toader	35h	15h (Rares) + 15h (Vlad) = 30h	In-review	Researched PostgreSQL, psycopg, and how to setup PostgreSQL database, schema and table structures. Organised "Migrate MongoDB to PostgreSQL" doc where we've listed several resources that we used, details about implementation, diff. Commands run, analysed expected memory consumption of two different strategies of structuring the tables, and thus mitigated the need of trial and errro for Strategy 1 (with 1 table), as it proved to be memory infeasible. Moreover, benchmarked the database by taking snapshots in different states of the app (i.e: 2k docs, 26kdocs, 40k docs, 52k docs) This doc. Was also used to split some work, such as fixing tests, writting docs and deciding who writes what in the MR description. Implemented basic functionality with PostgreSQL driver, initially persisted ~3k documents and ran several queries to

						benchmark performance. Fixed all the tests, fixed all related bugs (i.e: forgetting to `conn.commit()` after an INSERT, before closing the collection). Refactor setUp and tearDown to reduce code duplication (by creating a `BaseTest` class that inherits django.test.TestClass) and document `views.py` to explain the 4 types of queries computed there.
Persist articles in background task	#52	Matei Mirica & Cristian Ciacu	3h	5h (Matei) + 3h (Cristian)= 8h	Dropped	Issue was functional from the previous week, however we needed to make some pipeline configuration changes because the feature was implemented by running a RabbitMQ server and connecting a Celery worker to it. Locally everything worked fine, however, when trying to configure the server on the pipeline, the server could not start because a username and a password were required. After managing to remove the need of a password, the server managed to run but the worker couldn't connect to it, as the server didn't have

						required accessibility which re-opened the need of a password. Together with the team, we decided to drop this performance boost issue as we had already spent much time on it and it seemed that the improvements it brought were not that significant.
Display statistics section on the main page	#67	Matei Mirica & Crisitan Ciacu	24h	Matei (14h) + Cristian (12h) = 26h	Completed	Implemented new section on the home page which displays some statistics about the application: a first part displays 3 cards, showing the number of articles stored in the database, the number of users that had used the app and the number of Check URL queries that have been performed. The second subsection displays 5 bars, showing overlap statistics obtained from the Check URL queries. The issue required implementing new frontend components and backend endpoint to update and retrieve statistics.

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.	#18	Diana Micloiu	5h	4h	Completed	Created a special component taht would encapsulate the functionality of both the Could haves I've implemented (#18 and #5). For this specific issue I created a button for displaying up to 10 articles similar with the one provided by the user. Note that the buttom appearsonly if there are more than 5 such articles.
As a user, I want to provide a similarity threshold to set a overlap sensitivity, and in turn be shown all news articles' URLs in the database that have similarity above that threshold.	#5	Diana Micloiu	5h	4h	Completed	Created a marked slider which can be used by the user to set the minimum required similarity ratio for the articles to be displayed. In this way the number of articles shown adapts dynamically changing the aspect of the whole component.