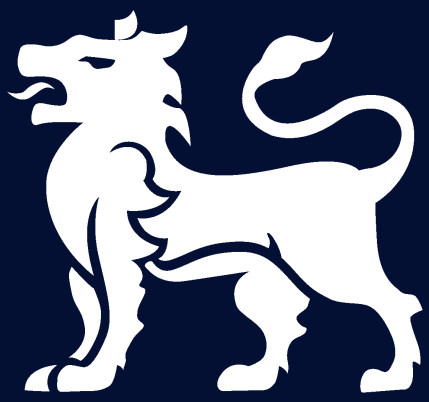
BBC Feed

A Novel Interface for News Consumption Inspired by Social Media

Owen Tourlamain

Submitted in partial fulfilment of the requirements of Birmingham City University for the degree of Master of Science



CEBE

Birmingham City University

United Kingdom

BBC News Labs

BBC

United Kingdom

May 26, 2021- 5512 words

Contents

1 Abstract

2 Research Question

How could social media user and application interface elements be adapted for and applied to a digital news consumption platform?

3 Aims and Objectives

3.1 Project Overview

This project aims to improve upon the application and user interface of traditional online news platforms, focusing specifically those used by BBC News. The digital news landscape is changing rapidly; a report from Ofcom found that people in the UK are increasingly using digital platforms to consume news, often through the use of social media (Ofcom, 2018). This project proposes that adapting user interface and data structure techniques from social media and applying them to a news delivery system could increase both user engagement and digital journalism efficiency. As the world of online news is constantly growing and evolving, the system created will also aim to be flexible so that it can adapt to future changes in this space.

The project will be content-agnostic allowing video, audio and written content to co-exist on a single platform. The system will not be limited to those three types of media however, it will be designed so that it can work with any future media that may be devised. To achieve this, the project will be divided into three parts: the feed, the catalogue and the scrapers. The feed will be an app that users can interact with. It will need to intelligently select content from the catalogue and present it to the user for consumption. The catalogue will store content that can be displayed, this will aim to be fast to access, and well organised to ensure a smooth flow of content to the users. The scrapers

will create the content for the catalogue by connecting to various BBC news data sources. They will read this content, translate it into a form suitable for the catalogue and store it. Creating these scrapers will involve writing specific interfaces to many different sources of news content; as this will take a lot of time they will not be produced as part of this project however they will be considered when developing the API for the catalogue portion.

Social media often faces criticism for being addictive and of wasting users time (Neyman, 2017). This raises ethical considerations with this project as it has the potential to also create these same issues. This will be mediated by the lack of user created content, meaning that the users time will be spent browsing informative news rather than wasting it. The addictive nature of social media is actually a benefit to this project as it is trying to increase user engagement on news platforms. As this project is focusing on the interface implementation the user experience is out of scope here, and will not be measured in any way. This is an area that a future study may wish to explore however as the ethical considerations around increasing user engagement time are both interesting and important to understand.

3.2 Key Terminology

Infinite scrolling: A user interface paradigm that presents content in a scrolling window. Before the end of the content is reached more content is loaded, resulting in the content appearing infinite to the user. This is often seen in social media, for example Facebook's news feed, and Twitter's timeline.

News Content: For the purposes of this study news content will be defined as text, audio, images or videos that are created or distributed by a news agency. These can come from a variety of sources ranging from broadcast news programmes to stock photos used within an article.

3.3 Deliverables and Goals

The deliverables for this project will be two of the three components highlighted above, as well as all documentation required to understand, maintain, deploy and use the system. The documentation will be provided as readme files alongside documentation within the code base itself. The majority code produced will be written in Python, with HTML, CSS and JavaScript used for the web interface. Also included will be a set of test data that can be used to demonstrate the system in a working state.

3.3.1 The Feed

This deliverable will be a Python Flask application that can either be run locally or deployed to a web server. It will have accompanying scripts that install any prerequisites for running the application. This deliverable will depend on having a valid catalogue to read from.

3.3.2 The Catalogue

This will consist of a Python module that will provide an interface to a database holding test data. In a production environment this database would be hosted on a saleable cloud system, such as AWS RDS. For this project however, a local SQLite database will be used.

3.3.3 Test Data

This will take the form of a single file containing SQL data that can be loaded by the catalogue to initialise its database with test data. The contents of this data will be sourced from the BBC News website.

3.4 Out of Scope Issues

This project is primarily focused on the engineering challenges of developing this system, as such it will not be attempting to measure user experiences in any way. The project will also not be attempting to judge the performance of any content selection algorithms that are used. These topics are highly important to the problem space being explored, however they each could constitute a project in their own right and would spread the focus of this project too thin. These two issues would make for excellent follow up projects however so some comments may be made where relevant to future research.

4 Background and Rationale

4.1 Overview of the Problem Space

Modern news consumption started with the newspaper. Despite moving to online platforms these roots can still be seen in the user interfaces of digital news platforms such as the BBC News website and mobile app (Ofcom, 2018). These interfaces broadly function by providing categories of content to browse, this makes these interfaces good for researching topics and for getting an overview of recent news within a topic (Yalanska, 2021). Social media instead often delivers content to users through an infinitely scrolling feed. This style of interface fits well with short bursts of interaction, such as while waiting for a kettle to boil or waiting for a bus (Yalanska, 2020). In these scenarios users often want to consume content without having to choose a category or risk running out of content.

This style of interface removes the need for the end user to decide what content they consume at the point of consumption, instead they control what content is presented to them by following, liking, subscribing to or otherwise choosing to receive content from a number of sources. From this user input

a social media platform will choose exactly which content to provide to a user, and in which order. The specifics of how these decisions are made are closely guarded secrets, and as such will not be investigated here.

While social media allows users to access news content in this way, it is often preferential to deliver news to users on a first party platform, such as the app or website of the news agency. This gives the producers of the content more control over how it is presented and consumed, as well as what other content may be presented to the user alongside the current content. This project aims to create an interface inspired by social media, but completely controlled by the news agency.

This also gives the creators of this content more flexibility to experiment with new content types, they are not limited to the narrow selection of media that the various social platforms offer. This feed can also offer features such as A/B testing that would enable the content producers to test the effectiveness of new media types (Gallo, 2017).

When creating printed media, such as a newspaper, each article or photograph that is included takes up space that could have been used for something else. Because of this it is key to only select content that will have the greatest impact, and testing new content is risky. In an infinite scrolling style of interface however there is no limit on the amount of content that can be shown to a user, as such inserting new media forms to test them becomes more viable as if the user is not interested they can scroll past and move on to the next piece of content.

4.2 Motivation for the Project

The BBC runs a news innovation team, called News Labs, which develops novel technologies for news creation, consumption and delivery. Two of their recent projects, which formed the inspiration for this project, are SlicerAV and Live Segment Notifications (LSN) projects. SlicerAV takes broadcast TV and

radio news programmes and automatically breaks them up into 'slices' of short form media (BBC News Labs, 2020c). Once this project was functional the next question was how to deliver this content to end users. For testing purposes they decided to tweet the slices as it was identified that the content fit well on to a social media platform, and twitter worked best for this use-case.

The LSN project aimed to inspect news broadcasts just before they go live and notify users about upcoming content, in the form of 'slices', that may be interesting to them (BBC News Labs, 2020b). This was successful, however it did not fit well with the current BBC online news delivery systems. This raised the possibility of a feed that would hold all the slices that a user had been notified about for them to browse. This idea was considered and built upon, eventually leading to the idea of a unified news homepage, where sliced content could be surfaced alongside regular articles and videos.

News Labs is focused on innovating news creation, consumption and delivery, as such they have several projects that could benefit from a flexible user interface to surface their content. One such project is Graphical Story Telling (GST) which aims to take articles and turn them into a series of graphics with overlaid text that can be swiped through (BBC News Labs, 2020a). This content is inspired by social media and would fit perfectly into this project. This provided the idea to make a flexible feed that can present a wide variety of content in one place, with functionality for personalisation and testing.

5 Literature Review

5.1 Interfaces for Online News Consumption

Since the popularity of online news has taken off, there has been a lot of innovation and experimentation into the interfaces we use to consume this media. Most news agencies have tried to move away from simply displaying walls of text to users, in the work of Braun (2014) we see a case study into how one large

news agency re-designed their user interface, and the software stack that runs it. This study highlights how important flexibility was in the system. They split the system between the front end and the back end with the goal that ”User interfaces and databases would become isolated components to be updated and swapped out” (Braun, 2014). This closely mirrors the model proposed in this project. The case study also made note of the new system allowing content to reach audiences that hadn't been considered targets when the content was produced:

> The same trading zone-style compromise also made it possible to take advantage of moments of alignment between editorial staffs and products whose tenor and target audiences might be viewed as incompatible on a larger scale.

This shows that the connections to the SlicerAV project should prove useful, as that will allow segments of broadcast programmes to reach audiences that the whole programme may not.

Some projects have taken this redesign in a different direction, in the work of Teitler et al. (2008) we see an attempt to display news content based on it's geographical location. Localised news has always been popular, with the BBC running separate news programmes for different UK regions. This approach would allow a user to browse content purely by location which may have advantages to both end users and content producers. A map interface will not be a part of this project, however it is worth keeping in mind when building the data structures so that out of the box ideas like this can be built in the future.

One key element to all major user interface redesigns is that they should be an improvement in some regard to the old system. Whether that is measured in loading speed, running costs, or any other metric, the improvement should be measurable in someway. Testing the proposed system against the current BBC News user interface would be a monumental task, requiring data that cannot be easily accessed. As such the measurement is out of scope for this project, however the means to perform those tests should be built into the

system. In order to identify which metrics are important to track we can look at the published work of Obrien & Lebow (2013) and Obrien (2011). Obrien & Lebow (2013) highlight that simple metrics, such as time spent browsing or number of articles viewed, are not sufficient on their own as both high and low figures for these can indicate good and bad user experiences. The authors also found that the best way to measure user engagement was through interviews, this would be impractical to use all the time. Instead this project will aim to include some form of user feedback into the system, this will likely be modelled on a favourites system or a positive/negative indicator.

5.2 Infinite Scrolling User Interfaces

(New Target Inc, 2020) defines infinite scrolling as:

> a technique that allows users to scroll through a massive chunk of content with no finish line in sight. This technique simply keeps refreshing a page when you scroll down it.

This description highlights how simple this design pattern is from a users point of view, but goes on to show how much of an impact it can have on the users. It states that users often feel that they ”might be missing out on information” New Target Inc (2020) as there is always more content being added to their screen. One possible solution it offers for this is some form of marker to show users how far they have gotten in the list and allow them to return later, this will be considered as a feature for this project.

Neyman (2017) and Karlsson & Larsson (2016) have looked into the negatives of infinite scrolling interfaces, highlighting the addictive nature and that users often waste time while using them. For the purposes of this project however these features are actually helpful as there is a desire to increase engagement on the BBC News platforms, and the users time will not be wasted as they will be browsing informative and well written news articles. Both studies do however note that too much choice can be a bad thing, Karlsson & Larsson

(2016) state:

> while people may be attracted by a large variety of options, they are more likely to act when given fewer choices.

This is something to keep in mind when designing this system, the project should aim to not overload the user with too many options at once. Techniques such as allowing users to save articles for later, return to their position in the list and allowing them to filter content to different categories will help to overcome this.

Another aspect to consider is the implementation, as mentioned by New Target Inc (2020) infinite scrolling has the potential to reduce the responsiveness of the interface as more content is loaded. Tajima et al. (2017) used SuperSQL in order to work around some of these issues, this may be a fitting technique for this project as it will be using a relational database. As a lot of content will have to be loaded, small delays between system components will quickly add up resulting in a poor user experience. To combat this each element of the system will need to be tested thoroughly and UI tasks such as rendering should be separated from other logic through threading.

5.3 Social Media Design for Increasing User Engagement

Social media is not a new concept to news distribution. Standley (2013) found in a survey that social media is now the most commonly used communication channel on the internet. Boukes (2019) found that social media ranked highly as an important source for news, especially among younger audiences. This has naturally caused many news outlets to start publishing content on social media, this however has not come without any downsides. Boukes (2019) showed that ”Twitter usage positively influenced knowledge acquisition [and] Facebook had a negative effect”, they suggest that this is co-related to the target users of the platforms, stating that twitter users are often motivated to use the platform to gain knowledge and information. This means that using these

techniques on a news app or website should work well, as the audience will have visited the service with the purpose of gaining information.

This project has one key difference from social media, in that it will not contain any user made content. Zhang & Liu (2013) found that an infinite feed increased appreciation of content produced by others, but decreased users desire to create their own content. Since there will be no ability to create user content this will not be an issue for this project.

There are a lot of negative effects often associated with social media. As this project is taking inspiration from this area it is worth taking steps to ensure that these negatives are not a part of this new system. Lupinacci (2020) suggests that a lot of these issues stem from a state of ”continuous connectedness” to other people. Users will not be connected to each other in this system, however they may still experience these issues by being constantly connected to the world through the news. The best way to avoid this is a topic for a study in it’s own right, for this project it is sufficient to be aware of the issue if it appears.

6 Methodology and Planning

6.1 Development Methodology

The development of this project will follow a Kanban-style methodology (Gupta, 2021). Since there is only one developer on this project this will not be used for collaborative purposes, instead it will be used to manage and monitor tasks. As tasks arise they will be added to a To-Do list. These will me moved to an In Progress list when they are started and then a Done list when finished. Tasks will be added, edited and split into sub-tasks as needed throughout the project. This methodology will allow a high degree of flexibility which fits the project well as the specifics of the project will likely evolve as the project progresses.

Test driven development was considered for this project. It would help to increase system reliability and enable the testing of the system to be efficiently

tracked and repeated when necessary. These features however aren't key to this project as the focus is around prototyping and rapid development. Using a methodology such as test driven development would add more overhead to the project and slow down progress. The risk of reducing the robustness of the system has little impact as this project will never be deployed in a live setting.

6.2 Project Timeline

Time management for this project will be monitored through the use of a Gantt chart (see fig. 1). Contingency time takes the form of two week long blocks where no development is planned. Depending on how the development progresses this time will either be used for further research and writing, or more development time. This chart will be updated as the project develops and objectives are added or removed, however individual daily tasks will not be added. These will be tracked through the Kanban board mentioned later in the methodology.

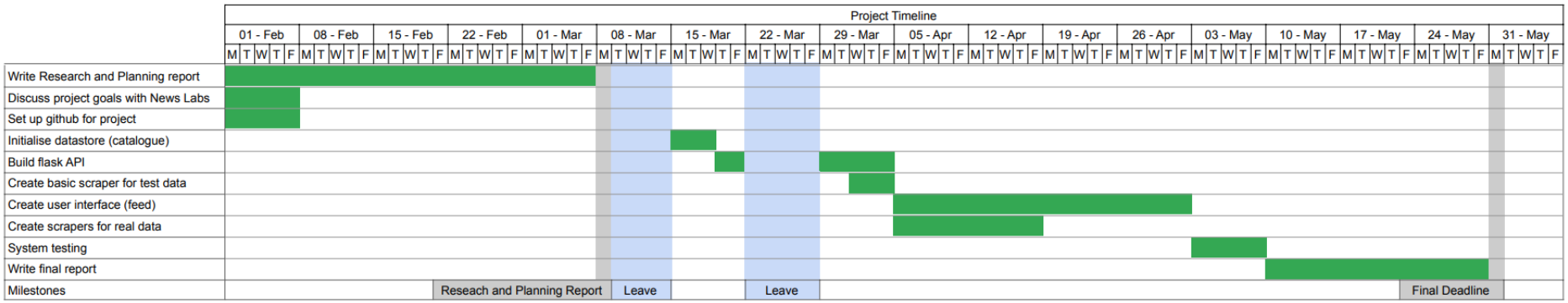


Figure 1: Project timeline gantt chart

7 System Development

7.1 Development Methodology

Throughout this project an iterative development style was used. This took the form of a 'design-implement-test' loop that was repeated for each step of the project. Tasks were taken from the Kanban board, implemented and tested before moving on to the next task. This was used to provide compartmentalisation to the system; each element performs a simple function that could be tested easily. When each of these elements are connected they create a more complex system.

A good example of this is the API side of the catalogue module. This uses SQL commands to retrieve information from the database and convert this information into Python objects. As part of this it is necessary to link some objects together, every *post* belongs to a *story* for example. Rather than writing specialised code to create this link each time, the existing *get_story()* method was used.

This compartmentalisation also streamlined development significantly as each element could be re-written independently of the rest of the system. This flexibility was utilised heavily when the catalogue module had to be re-written to support large changes to the database structure. It was simple to change the code within the module without affecting the rest of the system.

Using test driven development was considered for this project; it would have provided structure and repeatability to the testing procedure. However this would also have added significant overhead to the development of the project. As the project was heavily time restricted, and the output was focused around the user interface rather than the technical implementation, it was decided to use a methodology with less process overhead.

7.2 Language and Technology Choices

Language and technology decisions for this project were guided by the need for rapid development and the prototype nature of the system. As long term maintainability and reliability were not priorities for this project, the focus was instead on choosing technologies and languages that would facilitate rapid development and innovation. As such, the chosen platforms are all ones that the author has a lot of experience with.

The core of the project was implemented in Python 3.7, this was chosen as it is a recent version of Python that runs well on the hardware used for developing the system. To create the API and back-end portions of the system Flask was used as this fit well with the rapid prototyping nature of the project. Flask also provides the Jinja template rendering engine which made producing web pages fast and efficient. These pages were built using pure HTML, CSS and JavaScript as this avoided having to learn new user interface frameworks.

7.3 Development Progress

The first section that was implemented was the initial catalogue and API. These needed to be developed concurrently as they are very tightly coupled. It would also have been difficult to test the catalogue without the API present, and vice versa. The structure for developing these elements was to select a table from the database and implement it in catalogue with *add* and both singular and multi-item *get* methods, these were accompanied by matching API endpoints.

Once all of the tables were represented in this fashion, work was started on the user interface. To develop this, static web pages were first written. These had no connection to the data stored in the catalogue and were used to develop the interface without worrying about the data interface working correctly. Once a page was finalised the static elements were replaced by data read from the catalogue. The use of Jinja as a template rendering engine made this step

simple and greatly streamlined this area of development.

At this point a basic web interface had been developed and the next step was to add functionality to the site. Each feature was roughly prioritised based on time to implement and benefit to the system. Adding a functional log in system was quite time consuming, however it was vital for other elements of the system to function at all so this was given a high priority. As part of this process several new methods needed to be written in order to support new features. For example, it was required that users could be retrieved by name rather than by ID for the log in system to function correctly.

7.4 Issues Faced

The main issues faced during the project were around the need to balance the benefit of adding a feature with the time it would take to implement. The best example of this is when it was realised that a ground up re-write of the catalogue module would be highly beneficial to the project, however it would also take several days of development time. It was decided instead to find a work around that would take much less time to develop. This method of task prioritisation was vital to ensure the project was completed within the time frame required.

The time constraints of the project also required that some desired features be cut from the system. It was decided early on in the project timeline to remove the data gathering portions of the system. This allowed the project to become more focused around the user interface implementation, as creating placeholder data was significantly faster.

8 Final System Overview

The final system consists of two main components that provide the functionality and user interface of the system respectively which are detailed bellow. In

addition to these components there is a database schema and prototype data set which have been produced.

8.1 Catalogue

The catalogue portion of the system has been built as an independent Python module, the aim of this was to encourage de-coupling the data model implementation from the rest of the system. This allows the data source to be swapped out very easily and would allow the use of saleable cloud-based databases such as AWS RDS.

The main functionality of the module is provided by *catalogue.py*, in this file is a single class that handles all the interaction with the SQL database. When this class is initialised it attempts to access a database file, creating it if the file doesn't exist. If the database is empty it initialises it with a pre-written database schema and test data. This was done to reduce the development time needed to maintain the database as it could be quickly wiped and rebuilt with minimal effort.

All database tables have an *add* method, as well as *get* methods for both single items or all items in the table. In a full production system it would be necessary for this module to provide *create*, *read*, *update* and *delete* methods for all data. It was decided that this would take a long time to develop and would add minimal functionality to the project so the update and delete methods were not implemented. In the final system the *add* methods were hardly used due to the data scraping part of the project being removed, however implementing these methods has provided an API for automated data entry.

Some of the *get* methods have to link multiple items together, for example every *post* belongs to a *story*. This was achieved by having the relevant methods call other methods from the same class, for example *get_post* calls *get_story* using the *story_id* from the post. This method in turn calls *get_topic* which calls *get_brand* to complete the post item. With a longer development

time frame a better solution to this problem could have been found by utilising a pre-written Python object relational mapper such as SQLAlchemy. This would have made the data management side of the project much simpler, however it would have also imposed limits on which data technologies could be used which would have reduced the independence between the database and data management system.

The rest of the methods in the class provide specific data access as needed by other elements of the system. Methods such as *get_user_by_name* are required to fulfil a specific purpose and were added as and when the need arose during the project development. For example *get_user_by_name* was added to improve efficiency when checking if a user exists in the database. Only one update method was required during the project, this was to allow users to change their topic preferences through the web user interface.

As it stands the project implements zero authentication or security to any of the endpoints or database. Currently if the project was hosted online anyone who knew the endpoints could inject data into the catalogue easily, and anyone with access to the host machine could connect directly to the database to manipulate the data. In a production environment these issues would need to be addressed. The API would implement authentication where allowed applications could receive a token that would allow them to access the API endpoints. Additionally security certificates could be used to ensure that requests are originating from valid systems. The database wold also have to be secured by creating users with limited permissions and passwords, in addition to this the host machine would need to be secure to ensure attackers cannot access the system this way. These issues were not resolved in this project as it will never be hosted or used in a production environment, however they would need to be addressed if the project was developed any further.

8.2 Feed

The feed portion of the system provides a web based user interface as well as a RESTful API, using Flask to process these requests. Both of these are contained in *app.py*, using different endpoints to differentiate the two interfaces. The endpoint structure has been designed so that all the API endpoints end in */json/*; this means that for most user interface pages */json/* can be appended to the URL to retrieve the data in JSON format.

The API side of the Flask app provides an endpoint for each of the basic add and get methods that the catalogue supports. These endpoints can be used to add data to the system from external sources. As these external services would not need to modify or remove data so these endpoints were not added.

The rest of the endpoints provide the various user interface pages. Each is written as a Jinja template using plain HTML, CSS and JavaScript. The templates were designed to be modular wherever possible, this allows common UI elements such as the navigation pane to be reused between pages. The UI design follows BBC guidelines wherever possible, Including making use of the BBC font Reith Sans, and using a header and footer designed by BBC News Labs. These elements were included to help the UI feel more authentic, which makes the final system easier to compare to current news platforms such as the BBC News website. This was reinforced by the use of the official BBC UI and branding colours.

The novel elements of the UI were inspired by social media feeds such as Facebook's news feed and Twitter's timeline. This lead to the vertically scrolling list of posts, which allows users to easily consume news articles or scroll past them if they are not interested. One interesting UI development was the expanding posts, this allows for users to consume more content they are interested in without interrupting their scrolling. This keeps the UI focused on a single page experience, rather than directing users to multiple pages which could lead to navigational issues.

The feed implements very basic authentication, however this is only based on the user name and no password or other form of secret is used to secure these accounts. In a production environment this would cause major issues, however implementing these features would have added significant development time without contributing to the user interface or other features of the project. As this project is only a prototype and will not be deployed anywhere it was decided that not implementing this was acceptable

8.3 Database

The database has been designed to show how the use of intelligently structured data can effect the delivery of online news. The core data structure in the system is the relation between *posts*, *stories* and *topics*. Using this structure posts can be collected under their stories, which in turn can be grouped and filtered by topic. This allows for user interfaces to display content in a form which reflects the dynamic way that news stories develop and grow. This structure allows data to be inserted into the system from many sources all contributing to the same set of stories.

To achieve this structure the database had to be built to be designed following industrial best practices. As such it was normalised to the third normal form, this ensures that the data is stored logically and to eliminate redundant data. The best way to achieve this was to make sure that each record in the database was atomic, meaning that it only contains one item. For example each *user* can have multiple *topics* that they are following. To represent this 'one-to-many' relationship in the database an extra table was created that holds pairs of *(user_id, topic_id)* that can easily be queried to extract the followed topics for a given user.

To make the database easier to maintain, the schema has been defined in a separate file that can be loaded on start up. This allows the structure of the database to be changed with minimal effort, and even swapped out for a

completely different one if desired. In a similar manner the test data is stored in plain text as a sequence of SQL *INSERT* commands, Keeping these two files in plain text allows them to be version controlled along with the rest of the system, this helps streamline development and track changes throughout the process.

9 Informal Evaluation

10 Discussion

11 Conclusion

12 Recommendations

References

BBC News Labs (2020a) *Graphical Storytelling*. Available at: `https://bbcnewslabs.co.uk/projects/graphical-storytelling/` [Accessed 2021-03-06].

BBC News Labs (2020b) *Live Segment Notifications*. Available at: `https://bbcnewslabs.co.uk/projects/live-segment-notifications/` [Accessed 2021-03-06].

BBC News Labs (2020c) *Segmenting broadcast content with The News Slicer*. Available at: `https://bbcnewslabs.co.uk/projects/news-slicer/` [Accessed 2021-03-06].

Boukes, M. (2019) *Social network sites and acquiring current affairs knowledge: The impact of Twitter and Facebook usage on learning about the news*.

Journal of Information Technology & Politics, vol. 16(1):pp. 36–51. Available at: https://doi.org/10.1080/19331681.2019.1572568.

Braun, J. A. (2014) *News programs: Designing MSNBC.com's online interfaces*. *Journalism: Theory, Practice & Criticism*, vol. 16(1):p. 27–43. Available at: https://doi.org/10.1177/1464884914545730.

Gallo, A. (2017) *A Refresher on A/B Testing*. Available at: https://hbr.org/2017/06/a-refresher-on-ab-testing [Accessed 2021-03-06].

Gupta, D. (2021) *Kanban Methodology: The Japanese Method To Streamline Your Business*. Available at: https://techshali.com/kanban-methodology/ [Accessed 2021-03-06].

Karlsson, J. and Larsson, M. (2016) *Adapting infinite-scroll with the user experience in mind*.

Lupinacci, L. (2020) *'Absentmindedly scrolling through nothing': liveness and compulsory continuous connectedness in social media*. *Media, Culture & Society*, p. 016344372093945. Available at: https://doi.org/10.1177/0163443720939454.

New Target Inc (2020) *What Is Infinite Scrolling?* Available at: https://www.newtarget.com/about/web-insights-blog/what-infinite-scrolling [Accessed 2021-03-06].

Neyman, C. (2017) *A Survey of Addictive Software Design*. *California Polytechnic State University*, vol. 1(1).

Obrien, H. L. (2011) *Exploring user engagement in online news interactions*. *Proceedings of the American Society for Information Science and Technology*, vol. 48(1):p. 1–10. Available at: https://doi.org/10.1002/meet.2011.14504801088.

Obrien, H. L. and Lebow, M. (2013) *Mixed-methods approach to measuring user experience in online news interactions*. *Journal of the American Society for Information Science and Technology*, vol. 64(8):p. 1543–1556. Available at: https://doi.org/10.1002/asi.22871.

Ofcom (2018) *Scrolling news: The changing face of online news consumption*. Available at: https://www.ofcom.org.uk/__data/assets/pdf_file/0022/115915/Scrolling-News.pdf [Accessed 2021-03-06].

Standley, T. (2013) *Traditional News Media's Use Of Social Media*. *The Social Media Industries*, p. 154–167. Available at: https://doi.org/10.4324/9780203121054-17.

Tajima, M., Goto, K. and Toyama, M. (2017) *Non-procedural generation of web pages with nested infinite-scrolls in superSQL*. *Proceedings of the 19th International Conference on Information Integration and Web-based Applications & Services*. Available at: https://doi.org/10.1145/3151759.3151806.

Teitler, B. E., Lieberman, M. D., Panozzo, D., Sankaranarayanan, J., Samet, H. and Sperling, J. (2008) *NewsStand*. *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems - GIS 08*. Available at: https://doi.org/10.1145/1463434.1463458.

Yalanska, M. (2020) *Social Network Design. UX for Communication*. Available at: https://blog.tubikstudio.com/social-network-design-ux-for-communication/ [Accessed 2021-03-06].

Yalanska, M. (2021) *Web Design: The Big Guide into Different Types of Websites*. Available at: https://blog.tubikstudio.com/web-design-the-big-guide-into-different-types-of-websites/ [Accessed 2021-03-06].

Zhang, S. and Liu, I. (2013) *Attention trade-off between two types of user contributions: Effects of pinterest-style infinite scroll layouts on creating original sharing and appreciating others' sharing*. *International Conference on*

Information Systems (ICIS 2013): Reshaping Society Through Information Systems Design, vol. 1:pp. 538–556.