DISTRIBUTED SOCIAL NEWS

by

STEFANOS CHATZAKIS
URN: 6481123

A dissertation submitted in partial fulfilment of the
requirements for the award of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

May 2020

Department of Computing
University of Surrey
Guildford GU2 7XH

Supervised by: Sotiris Moschoyiannis

I declare that this dissertation is my own work and that the work of others is acknowledged and indicated by explicit references.

Stefanos Chatzakis
May 2020

Abstract

This report describes the design and implementation process for a web-based system that provides its users with a personalised view of the top social news depending on their interests. Moreover, in order to achieve this, the report will also focus on implementing some of the most popular ranking algorithm which will then be compared with each other in order to select the most appropriate for the project.

This report will loosely be split into two parts. The first being, understanding the strengths and weaknesses of each ranking algorithm and comparing them to find the most suitable. The second part will focus entirely on the implementation using an agile development methodology.

All in all the aim of this report will be to explain the challenges I face during the comparison process as well as the development process.

Acknowledgements

First of all, I would like to thank my parents Fedon and Katerina for all the sacrifices they have made for me to have this opportunity today, but also their continuous support and motivation. Furthermore, I would like to thank my sister and my close friends who have helped me cope with the difficulties I have faced during the past three years.

Contents

1 Introduction **12**

1.1 Background 12

1.2 Aims & Objectives 12

1.2.1 Project Limitations 13

1.3 Software Development Life Cycle 13

1.4 Chosen Methodology 14

1.5 Report Structure 15

1.6 Summary 15

2 Literature Review **16**

2.1 Technologies Used 16

2.1.1 Django Framework 16

2.1.2 Atom 17

2.2 Existing Solutions 17

2.2.1 Reddit Rank (Hot Sort) 17

2.2.1.1 Weaknesses 18

2.2.2 Hacker News Rank 18

2.2.2.1 Weaknesses 19

2.2.2.2 Conclusion 19

2.2.3 Wilson Confidence Sort 19
2.2.3.1 Conclusion 20
2.2.4 Summary 20

3 The Algorithm **21**
3.1 Planning 21
3.2 Designing 22
3.3 Creating 23

4 System Analysis **25**
4.1 Feasibility and System Limitations 25
4.2 User Experience 26
4.3 System Requirements 27
4.3.1 Hosting Requirements 27
4.3.2 System Functionality 28

5 System Design **29**
5.1 Prototype 29
5.2 User Interface Design and Usability 30
5.2.1 Home Page 31
5.2.2 Account Page 31
5.3 Difficulties faced 32
5.3.1 Pre-development 32
5.3.2 Development 32

6 Statement of Ethics **33**
6.1 Leagal stantpoint 33

7 Conclusion **34**

List of Figures

2.1 Reddit Ranking Algorithm in mathematical notation 17

2.2 Hacker News Algorithm in mathematical notation 18

2.3 Edwin B. Wilson confidence sort in mathematical notation 19

3.1 The stages of the final ranking algorithm. 22

4.1 Bootstrap colours 26

4.2 An example post 26

5.1 Django app file layout 29

5.2 Account view methods 29

5.3 Home page from mobile device 30

5.4 Home page from desktop 30

5.5 Account page pt. 1 31

5.6 Account page pt. 2 31

List of Tables

Glossary

AI	Artificial Intelligence.
NN	Neural Network.

Abbreviations

BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
BSC	Binary Symmetric Channel
DCT	Discrete Cosine Transform
ECC	Error Correcting Codes
FEC	Forward Error Correction
JPEG	Joint Photographic Experts Group
MPEG	Moving Pictures Experts Group
SER	Symbol Error Rate
SNR	Signal to Noise Ratio

Chapter 1

Introduction

1.1 Background

Current social news sites try to produce a list of current important news items by applying algorithms and voting by the community. While more decentralized than a traditional publication, the system still relies on a select few(moderators, administrators) to safeguard it from manipulation (spam, vote-rigging) while maintaining the culture of the website. Further, by providing a centralized view of the current news, they necessarily average the preferences of their members instead of addressing the needs of each one based on their voting patterns. This project aims to develop a new type of social news aggregator that allows each user to see a personalized view of current events while allowing them to customize the algorithm by which the items get sorted.

1.2 Aims & Objectives

The solution, which will first be theorized and made into an algorithm, will first be comprised of the combination of the most critical aspects of the most popular sorting and ranking solutions out there while tailoring the final solution to one that fits a social news aggregator.

1. Develop a Functioning website with users that can create, edit their posts and view all of the users post ranked by timestamp.

2. Compare the most commonly used algorithms, critique the positive and negative attributes and theorize, which parts of each algorithm should be implemented in the final implemen-

tation.

3. Compare the relevancy of the results after the final version of the algorithm has been implemented.

4. Create a modern and responsive website that is easy to use and compatible with a variety of screen resolutions, in order for the functionality of the program to be more clear.

1.2.1 Project Limitations

Since the development of this project was initiated, the goal was to create a testing platform on which to compare the results of the implementation of the final algorithm in a real-world environment. Thus, the scalability of the project as a whole was not taken into consideration. Having that said, the issue of scalability would occur not because of the algorithm itself, but due to the reason that the software developed does not address many security issues as well as correct optimisation so that the system would be able to hold on with more users.

1.3 Software Development Life Cycle

Software Development Life Cycle (SDLC) is the process that a software project follows to develop and test the system. A Software Development Life Cycle can be thought of as a master plan that describes how the the development, testing, enhancement and maintaining processes should occur during the implementation of the software (Stackify 2017).

The goal of the SDLC is to produce a software with the highest quality at the lowest possible cost. In order to achieve this, SDLC provides six phases that are consistent with the most popular methodologys such as waterfall, spiral and Agile (Stackify 2017).

The SDLC phases include:

- Requirement analysis
- Planning
- Software design such as architectural design
- Software development

- Testing
- Deployment

The most traditional method for software development is the Waterfall methodology. Waterfall, delivers the above phases in a linear fashion, meaning that each phase can begin only after the previous has been completed. Moreover, thanks to the implementation of the phases, the waterfall methodology becomes a fairly straightforward process. However, a process can never be repeated unless a critical error occurs that requires the revision of a previous phase. This makes it incredibly risky to use in a final year project development environment since fully following the methodology, could prove difficult to address the arising errors that are bound to occur.

A more common methodology in the recent years is the Agile development approach. This approach takes the phases previously mentioned and creates the so called sprints. The sprint's duration usually is a couple of weeks long with pre-specified goals set at the beginning of each sprint. Each sprint can be thought of as a many smaller implementation of the waterfall approach, since the phases are consistent with both methodologies but since agile is comprised of sprints, during each sprint each phase will re-occur.

1.4 Chosen Methodology

After considering most of the commonly used methodologies, and due to time limitations, I decided that I was not going to use a methodology that was too complicated or had too many restrictions. Having that said, I chose the Agile development methodology using the Kanban framework. The goal of using this framework is to aid the developer in the categorisation and prioritisation of the work and tasks. Furthermore, the agile methodology was chosen since before the development; the technologies used were previously known. Additionally, the development will be broken down into small increments which will encourage the mastery of the technologies used after each sprint. Lastly, the sprints will allow for overall better testing and feedback since they will be received after each sprint.

1.5 Report Structure

The report structure will mirror the planning and development process as it occurred. First, by reviewing and constructing the ideal ranking algorithm for a social news aggregator following the development of the web application which will implement the constructed algorithm.

1.6 Summary

Overall the introduction chapter has portrayed an overview of the project's definition, flow and goals as well as addressing how the project will deal with the different phases of development through defining the methodology.

Chapter 2

Literature Review

2.1 Technologies Used

During the entire design and implementation process, the decision was made to use the Python programming language. This choice was made originally for the ease of implementing algorithms in a scripting language like python. However, while progressing from the testing phase to the development phase, a dilemma was faced. Either to re-implement most of the progress I had made to a language that was more web and browser-based like JavaScript or, to create a hybrid of both front-end and back-end schemes. After some research online, Django was used, a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

2.1.1 Django Framework

According to the official Django website, "Django is a framework that was built as a tool for front-end developers that needed a simple way to bring their ideas to life without the need of a back-end developer that handles processes such as creating and connecting the server-side with the client-side as well as the creation and handling of the system's database (Editorial-Team 2016). Furthermore, Django is a high-level Python framework that utilizes SQLite as a relational database. Moreover, Django uses an MVT pattern similar to the more widely used MVC that frameworks like Ruby and Ruby on Rails uses. Lastly, Django is secure. It has one of the best out-of-the-box security systems out there, and it helps developers avoid common security issues, including clickjacking, cross-site scripting and SQL injection.

2.1.2 Atom

Since the whole project was entirely in Django, it meant that all the code was compiled through the console. For the text and file editor, Atom was used for its familiarity and simplicity. Moreover, for testing and displaying the application, Google Chrome was used.

2.2 Existing Solutions

2.2.1 Reddit Rank (Hot Sort)

After conducting research, it was clear that one of the most popular social news aggregators currently is without a doubt, Reddit. Reddit, although it has now switched to a different ranking algorithm, since the end of 2010 but have since made their old ranking system available to the public. Their algorithm explained in the purest form takes many parameters such as the timestamp of a post, the difference between a post's upvotes (likes) and downvotes (dislikes) and inputs those parameters in a formula that outputs a final rating that dictates a post's position compared to others.

Given the time the entry was posted A and the time of 7:46:43 a.m. December 8, 2005 B, we have t_s as their difference in seconds

$$t_s = A - B$$

and x as the difference between the number of up votes U and the number of down votes D

$$x = U - D$$

where $y \in \{-1, 0, 1\}$

$$y = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

and z as the maximal value, of the absolute value of x and 1

$$z = \begin{cases} |x| & \text{if } |x| \geq 1 \\ 1 & \text{if } |x| < 1 \end{cases}$$

we have the rating as a function $f(t_s, y, z)$

$$f(t_s, y, z) = \log_{10} z + \frac{yt_s}{45000}$$

Figure 2.1: Reddit Ranking Algorithm in mathematical notation

2.2.1.1 Weaknesses

An issue which is absent currently Reddit, but present in a lot of websites that take average rating as a rating attribute, is the following: Average rating works fine if you always have a ton of ratings, but suppose item 1 has 2 positive ratings and 0 negative ratings. Suppose item 2 has 100 positive ratings and 1 negative rating. This algorithm puts item two (tons of positive ratings) below item one (very few positive ratings).

2.2.2 Hacker News Rank

Hacker News is one of the most popular social news aggregator targeted towards developers and provides its users with mostly technology related news. Their ranking consists of three parameters, penalties, votes and age. The most impactful parameter on the formula is the penalty. The penalty's value is determined by the use of blacklisted words such as "NSA" which drops the story rapidly in the ranking. In order to keep the top stories fresh, Hacker News also issues a severe penalty on stories that reach 40 comments. The impact of a penalty can be calculated with the scoring formula since if an article gets penalty factor of 0.4, each vote will now count as 0.3. However, a factor of 0.1 is equivalent to each vote, counting 0.05. Meaning that although a penalty factor of 0.4 would drop an article 66% faster than usual, a factor of 0.1 would drop an article by 3.6 times than normal. In outline, each item is given a ranking, and the articles are sorted according to the ranking. The simplistic way to think about ranking is the number of votes is divided by time, so more votes results in a higher ranking, but the ranking also drops over time. The votes are raised to a powerless than one, while the time is raised to a power greater than one, so time has more effect than votes. Some additional penalties also may be applied to the ranking.

$$rank = \frac{(score - 1)^{.8}}{(age_{hours} + 2)^{1.8}} * penalties$$

Figure 2.2: Hacker News Algorithm in mathematical notation

2.2.2.1 Weaknesses

1. Wall-clock hours penalize an article even if no one is reading (overnight, for example). A time denominated in ticks of actual activity (such as views of the 'new' page, or even upvotes-to-all-submissions) might address this.

2. An article that misses it's audience first time through, perhaps due to the aforementioned reason or a bad headline may never recover, even with a later flurry of votes far beyond what the new submissions are getting.

2.2.2.2 Conclusion

Overall, Hacker News's algorithm is quite simple, thus making the implementation of something similar not that difficult, especially since it takes into account many factors. However, there are drawbacks with using timestamps as addressed above, and a solution to this would be to use time denominated in ticks of actual activity (such as views of the 'new' page, or even upvotes-to-all-submissions) which might address this issue. The use of penalty is an interesting concept that would make sense in a user generated content system in which there are no administrators to regulate the content.

2.2.3 Wilson Confidence Sort

The problem that the Wilson confidence score tried to resolve was, developers who need their application's users to sort things and to balance the proportion of positive ratings with the uncertainty of a small number of observations. The math to solve this problem was worked out in 1927 by Edwin B. Wilson (Miller 2009). Considering only positive and negative ratings (i.e. not a 5-star scale), the lower bound on the proportion of positive ratings is given by:

$$\left(\hat{p} + \frac{z^2_{\alpha/2}}{2n} \pm z_{\alpha/2}\sqrt{[\hat{p}(1-\hat{p}) + z^2_{\alpha/2}/4n]/n}\right)/(1 + z^2_{\alpha/2}/n).$$

Figure 2.3: Edwin B. Wilson confidence sort in mathematical notation

2.2.3.1 Conclusion

Despite a very sought after solution to modern-day sorting, the Wilson Confidence score is to some degree not oftenly used. This very complicated solution only increases the efficiency of the results by a small margin. Furthermore, many problems occur in the voting system that the Wilson Confidence sort is based on. A simple example is that many people who find something mediocre will not bother to rate it at all; viewing or purchasing something and declining to rate; it contains useful information about that item's quality (Miller, Feb. 2009).

2.2.4 Summary

To summarise, most of these algorithms mentioned are tailored for their particular use or website. However, most of the existing solutions are bound by the user's interaction with the content to improve or worsen the overall score of a post. What this project is trying to propose, is a system where the only interaction that is needed by the reader is their interests, after which the algorithm takes care of the rankings. Nonetheless, we can assimilate parts of existing systems that can help polish the final version of the algorithm.

Chapter 3

The Algorithm

3.1 Planning

Most ranking algorithms as have been discussed, mainly revolve around some combination of voting and time. While some have more complicated parameters, they are usually tailored to the website's philosophy, such as the use of the penalty attribute in Hacker Rank. However, since this project aims to provide the user with an automated display of the most relevant news tailored to the user's interests, voting will be replaced with the user's interests.

The user should be able to select interests in their account page, which have a score associated with them and that score should be a significant contributor in the score of a post. When a post gets created is also be a valuable attribute, and it should be beneficial to the algorithm since time can never be the same even if the rest of the attributes are the same, differentiating in that way two potentially similar posts. Another vital aspect to consider is how many interests does the post have, in which the user is not interested. However, the impact of this attribute should not be as negatively equal to a relevant interest in a post. For example, if the score of the common interest between user and post is equal to 10, a non-common interest should impact the overall score by -5 or lower. Lastly, the concept of banning or negatively reducing a ranking score is an essential attribute and can be detrimental in the image and environment that a system portrays.

3.2 Designing

After a couple of failed attempts, the final algorithm was a combination time, common interests, non-common interests and penalties. The algorithm relies on multiplication and division to increase or decrease a rating for the sake of simplicity bundled with the fact that time which when denoted in timestamps usually tends to be a couple of digits long in the first place. The algorithm can be seen below:

$$score = t_u ,$$

where t_u is the timestamp that the post was updated.

$$score = \left(\sum_{i=0}^{n-1} a_i * 5 \right) * score ,$$

where a_i are the common interests of the post and the user.

$$score = \frac{score}{\left(\sum_{i=0}^{n-1} b_i * 2 \right)} ,$$

where b_i are the non - common interests of the post and the user.

$$score = \frac{score}{\left(\sum_{i=0}^{n-1} c_i * 3 \right)} ,$$

where c_i are the number of unique penalty words in the post's content.

Figure 3.1: The stages of the final ranking algorithm.

Since time was going to be used as timestamps which usually tend to be a couple of digits long, I decided to use multiplication and division to increase or decrease a rating was used. As portrayed in 3.1, if the user is not logged in or, they have no interest selected, the score of each post is set equal to the time that the post was updated. However, if the user is logged in and has selected interests, then the existing score is multiplied by 5 for each common interest of the user and the post.

For the deduction part of the algorithm, the score is halved for each non-common interest. This decision makes sense since, if a potential user is interested in news, a post with only the "news"

interest will rank higher than a post with the news and music interest unless the first post is much older than the second one. The last deduction part of the algorithm is penalties. Penalties, that at the moment are profanity words, essentially counter a common interest, since the score of a post is divided by 5 for each penalized word the post contains.

3.3 Creating

Assigning each post with a score based on the parameters aforementioned happens when the "get_blog_queryset" function is called in "home_screen_view", and the code featured below explains is part of the "get_blog_queryset" function.

```
# Checks if the user is authenticated.
if user.is_authenticated:
for post in queryset:

  # A list of common interests.
  common_result = []
  # A list of non-common interests.
  non_common_result = []
  # Set's the score to the time a post is updated.
  post.post_score = post.date_updated.timestamp()

  for element in post.interests:
    # Updates based on common interests.
    if element in user.interests:
      common_result.append(element)
      post.post_score = post.post_score * (len(common_result) * 5)

    # Updates based on non-common interests.
    if element not in user.interests:
      non_common_result.append(element)
      post.post_score = post.post_score / (len(non_common_result) * 2)
```

```
# Updates based on the penalty words used.
if post.title in penalty_list or post.body in profanity_list:
  for penalty in penalty_list:
    post.post_score = post.post_score / (len(penalty_list) * 5)
```

Chapter 4

System Analysis

4.1 Feasibility and System Limitations

The project was designed as a demonstration to show how a let-alone, but personalized social news aggregator would work. The system flagship features of competing systems, all while mitigating issues such as the need of moderators to manage common issues such as vote-rigging or out-of-place content that disrupt the culture of the website.

Even though the system achieves the project's scope, that is not to say the system has reached its potential. There are many solutions that could perhaps increase the project's quality. For example, in order to improve sorting over time, the user could set their original interests but, over time a NN could monitor the user's actions and alter their interests judging from the way they interact with posts, such as time spent reading on a post that had other interests that the user did not initially mark as desirable. That way, the AI would create some hidden ranking that would sort the posts by that rank. It would not come with complications; nevertheless, building and running such a network would require not only immense dedicated computing power but also, periodic monitoring and safety regulation.

Changing the project's objectives and looking into lesser-known existing websites and their solutions in the literature review could likely provide more information on how these other systems handle sorting and what intricacies are involved when sorting information. Making the project more complex would defeat the goal of the project and could result in an overwhelming experience for the user. Moreover, in order for this project to yield meaningful results and an enjoyable experience results, responsiveness is key, which by overloading the system could result

in the lack of those aspects.

4.2 User Experience

In order to create a user-friendly interface and experience, the front end of the website is going to use Boostrap. Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components (Contributors 2019). This allowed me to make a very clean and friendly user interface.

Bootstrap uses default colours for different types of importance as visible:

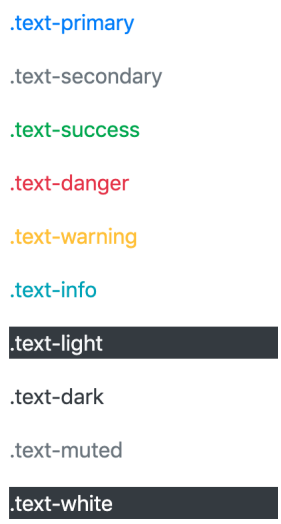

Figure 4.1: Bootstrap colours

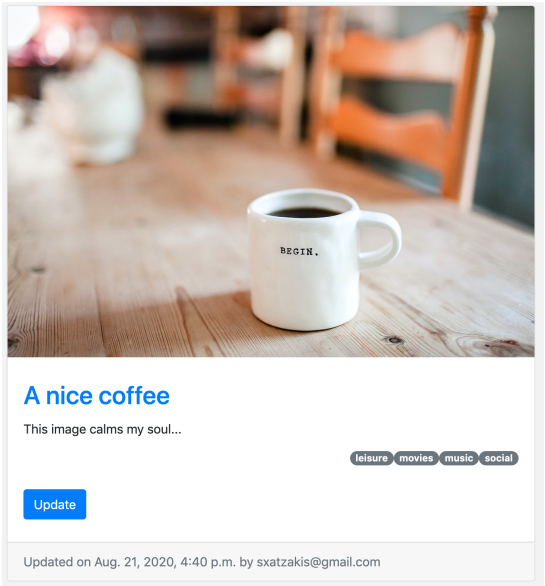

Figure 4.2: An example post

After taking a few approaches, the final design I came up with was as shown in 4.2. Even though this is just a design, it should be easily linked to the back-end system. It consists of an image, a title, a body, an update button (if the post is by the user viewing it), several tags for the interests, and mutable text at the bottom with information on the post. The title is an important aspect of the post, meaning the blue colour was used to show importance. Next, the body text is "plain text" meaning it was left as is. The interest tags are not interactable meaning, that the secondary colour of Bootstrap was used. The update button is a call to action; thus, used the primary colour. Lastly, the information below the post should be informative, and

thus the muted colour was used. This style should be kept consistent throughout the website, giving the user a clear picture of what actions they can take at any point.

4.3 System Requirements

The project will be entirely written in Django; the back-end consists of Python and SQL, but the front-end is HTML meaning that the website should be available by any browser regardless of system specifications. Yet, in order to host the project, the host is required to meet a couple of specifications.

4.3.1 Hosting Requirements

Django includes requirements.txt files that includes the versions of the modules that it uses. In this case:

1. Django==2.2.2
2. Pillow==7.1.1
3. pytz==2019.3
4. sqlparse==0.3.1

Specifications

1. A modern operating system (Windows 7 or 10, Mac OS X 10.11 or higher 64-bit, Linux: RHEL 6/7 64-bit)
2. x86 64-bit CPU (Intel/AMD architecture)
3. 4-GB RAM
4. 5GB free disk space

This list includes the requirements in order for the project to run smoothly. The most important requirement for a system to host this project is for Python to be runnable; once this requirement is met, the project will execute regardless of the specifications. Another issue to consider is that the Disk space requirement for this project will scale depending on the number of posts.

4.3.2 System Functionality

The user should be able to view the home page with all the blog posts sorted by time. The user can choose to register an account with an email, username and password. If they are logged in, a personalised view of the home page is displayed to them. Furthermore, the user has the option to view the account details, where they can edit their personal information, and interests, and view all the posts they have made. Also, when creating a post, the user should be able to enter a title, body content, choose an image as well as the post interests. Moreover, the user should be able to view other user's posts but only be able to edit posts made by them. Lastly, the user should be able to log out at any point.

Chapter 5

System Design

5.1 Prototype

The prototype mostly consisted of the user being able to create an account, and create posts that simply included a text. As stated before, since I decided to use Django to develop this app, I had to work with apps. This means that every major aspect of the project such as blog posts and account of is implemented as an app.

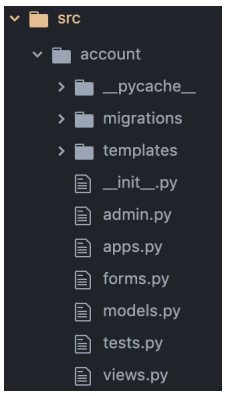

Figure 5.1: Django app file layout

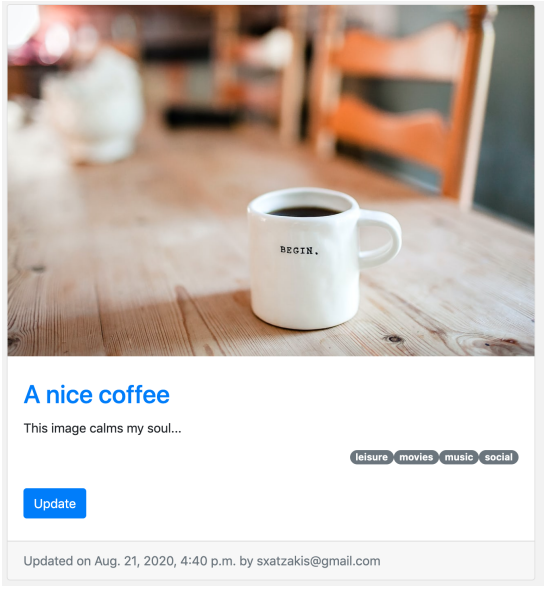

Figure 5.2: Account view methods

In Figure 5.2 are the methods implemented during the prototype process that handled the account authentication.

5.2 User Interface Design and Usability

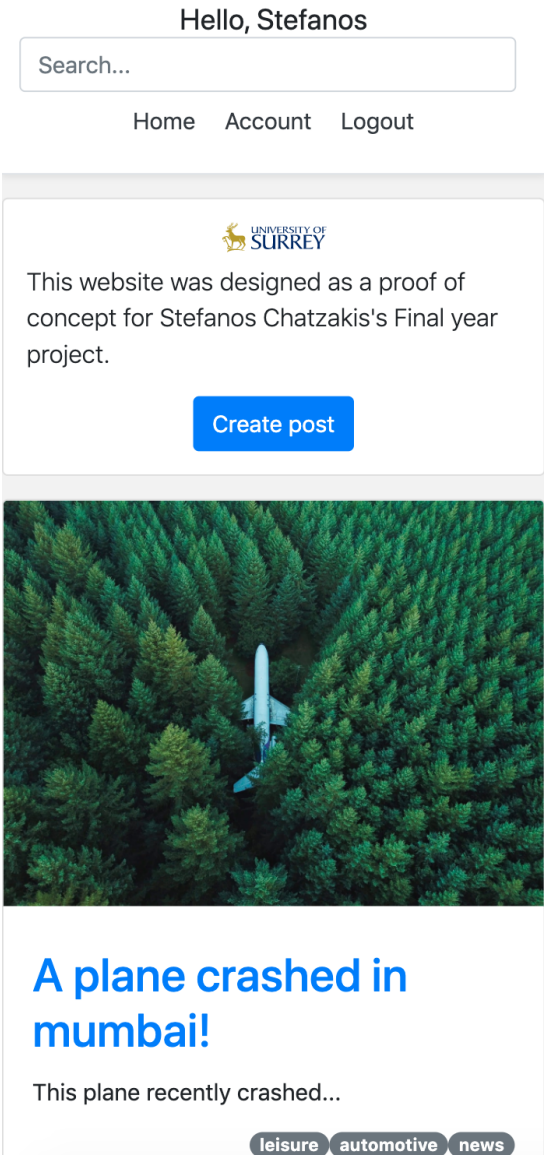

Figure 5.3: Home page from mobile device

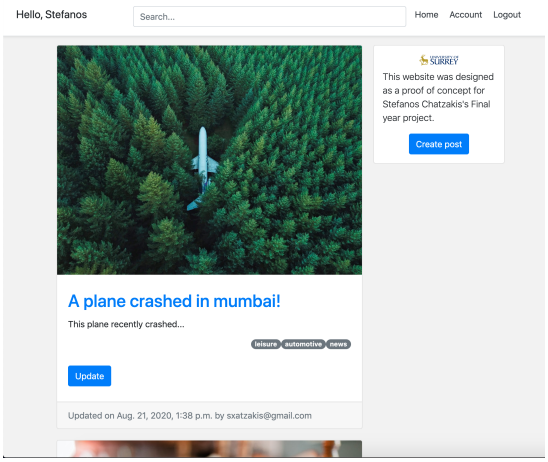

Figure 5.4: Home page from desktop

During the system analysis, I touched upon the user experience, both in how the user should interact with the colours and buttons and when a certain colour should be used. This section explains how those principles where implemented in the final version of the project. The project is designed to work in a plethora of screen sizes as seen on 5.3 and 5.4 making the project responsive giving the algorithm the chance to stand out, while providing an enjoyable experience to the user.

5.2.1 Home Page

5.4 shows the home page view of the website. It consists of the template mentioned in the system analytics, and the rest of the website follows along with those characteristics. The page header, includes a welcoming message, a search bar and three buttons which vary, depending on if the user is logged in or not. These buttons include: "Home", "Account" or "Login" and "Logout" or "Register". Then we can see the main content, which includes the blog post that is implemented as mentioned before and the create post section which again, follows the user experience philosophy mentioned before.

5.2.2 Account Page

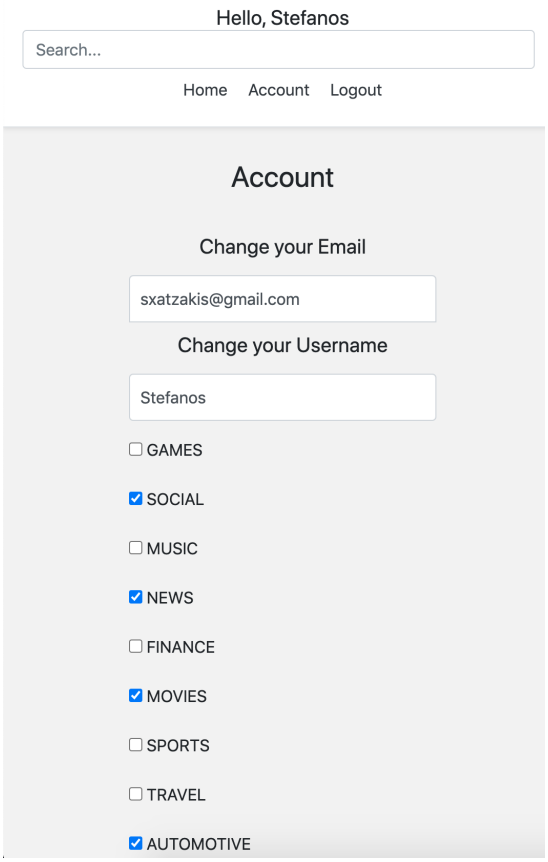

Figure 5.5: Account page pt. 1

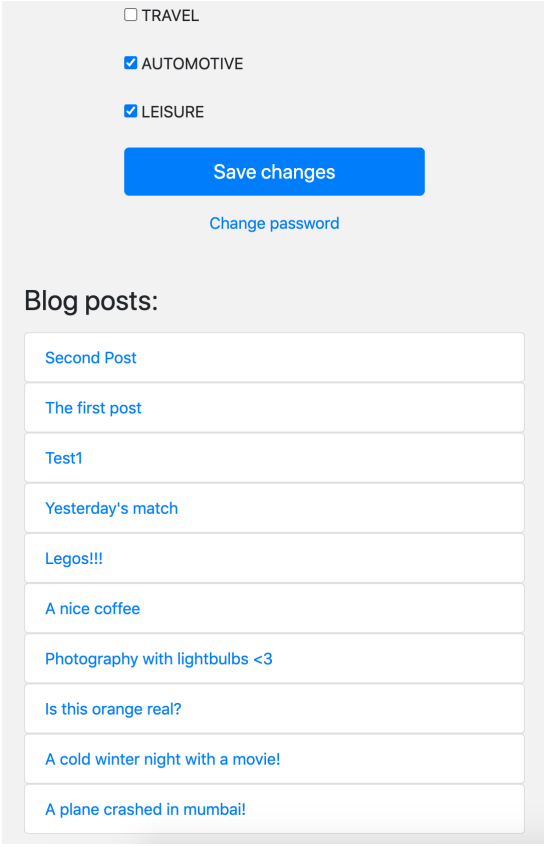

Figure 5.6: Account page pt. 2

5.5 and 5.6 display the account page. Apart from the header which hasn't changed, the main page content has changed, and several fields are visible. In this page, the user can edit their personal information (which is securely handled by Django's authentication system); as well

as configure their interests and view all the posts they have made. This page also follows the consistent design decisions mentioned before.

5.3 Difficulties faced

This part will mainly touch upon difficulties faced during the development process of designing and implementing the program.

5.3.1 Pre-development

During the planning process, the choice to use Django was certainly not one without issues. For example, there isn't much information online to create a form that handles the information the way I needed them to and how to render them effectively. These issues mostly have to be learned through trial and error with the Django documentation being the sole guide. This resulted in spending a lot of time to learn all the Django components before the implementation could begin. Still, since many big organisations such as Reddit run their website entirely on the Django framework, it made me confident that the learning curve was worth it.

5.3.2 Development

During the development process, I did come across a few issues, mostly regarding keeping track of versions from different dependencies and programs which sometimes resulted in restarting the project from the start. Moreover, I had to re-implement different variants of the algorithm and test them. This, however, became more difficult once I presented to the people that tested the application with two different algorithms. During the designing process, they provided me with information that I had to take into consideration and improve the final algorithm. Yet, now that I can reflect upon this issue, it is more apparent that the reimplementation process of reimplementing different parts of the algorithm, provided a valuable learning opportunity which led to me writing more efficient code overall.

Chapter 6

Statement of Ethics

During the creation of this project, all the legal, ethical and social aspects have been considered and analyzed below. Since one objective of this project is to eliminate the existence of administrators and moderators, the user's information itself should not ever be intervened by anyone.

6.1 Leagal stantpoint

In the process of creating this project I was extra careful that the information and software I used were under open-source or similar licenses. This is important since many relevant to this project technologies are licensed so that they can not be used in any shape or form. Furthermore, most of the software I used had technologies that handled user information securely by hashing or other ways. Specifically this what Django's authentication system consists of:

1. Users
2. Permissions: Binary (yes/no) flags designating whether a user may perform a certain task.
3. Groups: A generic way of applying labels and permissions to more than one user.
4. A configurable password hashing system
5. Forms and view tools for logging in users, or restricting content
6. A pluggable backend system

Chapter 7

Conclusion

Bibliography

Billard, M. (2019), 'An introduction to ranking algorithms seen on social news aggregators (example)'.
URL: https://coderwall.com/p/cacyhw/an-introduction-to-ranking-algorithms-seen-on-social

Contributors, W. (2019), 'Bootstrap (front-end framework)'.
URL: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

DoorDash (2017), 'Tips for building high-quality django apps at scale'.
URL: https://medium.com/@DoorDash/tips-for-building-high-quality-django-apps-at-scale-a5

Editorial-Team (2016), 'Why we use django framework and what is django best used for'.
URL: https://djangostars.com/blog/why-we-use-django-framework/

Gabriel, R. J. (2020), 'Robertjgabriel/google-profanity-words'.
URL: https://github.com/RobertJGabriel/Google-profanity-words

Laramee, R. S. (2011), 'Bob's project guidelines: Writing a dissertation for a bsc. in computer science', *Innovation in Teaching and Learning in Information and Computer Sciences* **10**, 43–54.

Majewski, M. (2019), 'Top 6 software development methodologies - blog | planview'.
URL: http://leankit.com/blog/2019/03/top-6-software-development-methodologies/

Miller, E. (2009), 'How not to sort by average rating'.
URL: https://www.evanmiller.org/how-not-to-sort-by-average-rating.html

Salihefendic, A. (2015*a*), 'How hacker news ranking algorithm works'.
URL: https://medium.com/hacking-and-gonzo/how-hacker-news-ranking-algorithm-works-1d9b0c

Salihefendic, A. (2015*b*), 'How reddit ranking algorithms work'.

URL: https://medium.com/hacking-and-gonzo/how-reddit-ranking-algorithms-work-ef111e33d0d9

Shirriff, K. S. (2013), 'How hacker news ranking really works: scoring, controversy, and penalties'.

URL: http://www.righto.com/2013/11/how-hacker-news-ranking-really-works.html

Stackify (2017), 'What is sdlc? understand the software development life cycle'.

URL: https://stackify.com/what-is-sdlc/

Willee, H. (2016), 'Django introduction'.

URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction