UNICEF Child-Friendly Cities Initiative

ANONYMOUS AUTHOR(S)

1 REQUIREMENTS SPECIFICATION

We are creating an app and companion website for UNICEF's Child-Friendly Cities Initiative (CFC) so that young people can find it easy to digest information about local issues and their council's Local Development Plans so that they can give feedback through the council's consultations and get more involved in local politics.

Our problem statement: Young people lack the information and awareness to be involved in local politics, and so are under-represented in their Council's plans. Our project aim is therefore to create a system that will make it easier for Young People to find information and be involved in local issues they care about.

1.1 Reviewed Literature

Due to the specificity of both our problem and solution, there is very little literature on the topic. Thus we have opted for building our requirements from stakeholder engagement. This is even more relevant as the solution we are designing must meet criteria that they have provided.

1.2 Current Solutions

1.2.1 Cardiff Website. The Cardiff Website serves many separate purposes; it is not particularly easy to navigate, and there is not a solution tailored specifically to keep up with local politics. When attempting to navigate to the local development plan, errors are encountered, and so clearly little effort is made to keep the information up to date[1].

1.2.2 Neighbourhood Apps. here are apps which cater specifically to keeping people up to date in their local neighbourhood, for example "Nextdoor", however these apps are more like messaging sites, which means that the information is not curated, unlike our proposed solution, in which the news will come from a central source, the council.

These two existing solutions require our solution to have consistent, accurate information directly from Cardiff Council. In addition, the young people in which we consulted expressed a desire for information to be flagged if the admins abuse their power and post malicious content. The flagged content would then go under a review, and, if deemed to be malicious, will be taken down. This was a requirement we had not previously considered.

1.3 Discussion with Stakeholders

On the 24th of November we conducted a meeting with Cardiff Youth Council (CYC) to gain a better insight on how they visualise the application running, and how people will interact with it. This meeting included three young people, and some members of the CYC to help facilitate. This was fewer people than we were expecting but this was due to the fact that many of the young people involved with the CFC had been inundated with meetings and so were unable to attend.

Having spoken to young stakeholders, we gained the following information:

- Currently, young stakeholders only discuss political topics that are relevant to them, and often only with their close friends or families.
- At the moment, information is hard to find with the Cardiff website not being user friendly, and consultations hard to find for those unfamiliar with the system.

- The preferred interface is something similar to Instagram, where images and text are blended together in a scrolling feed.
- The notification system should be able to be customisable to different time frames, and it should not be via email, as otherwise it will get ignored or put in spam. Ideally, notifications will contain a bulletin round-up of a certain time frame that the user can specify.
- Young stakeholders wish to be able to find out about events, campaigns, meet your MP events and more through the app.
- The information should not be delivered in a condescending way.
- Users of the app should be able to flag publishers if they act in a malicious way
- A private profiling system should be implemented to allow users to record their interests and get notified about them as well as make comments on posts.

On the 3rd of December, two of our members held a meeting with two of our stakeholders who work with the CFC. This meeting was to consider the side of the publishers and to discuss the capabilities of the CFC and the CYC to produce content for the app once we launched it.

Most importantly, we confirmed that the CYC are willing and able to produce content for the app. An administrative member of their staff will be given the task of creating both written summaries of relevant government plan changes and consultations, as well as creating ten second video summaries. Further, in order to ensure engaging and relevant content for our target users, the young people of the CYC will have the opportunity to help create these video summaries.

Our stakeholders assured us that they do have access to the necessary materials from the council, as well as the experience to create and publish these summaries.

Following our feedback from the young people, we presented the idea of putting local events onto the app. The CFC Workers agreed that this is a good idea but would require proper verification.

1.4 Environment of Operation

The role of workers at CYC and CFC is to effectively engage as many young people in local politics as possible, creating educational campaigns, and running programs that keep young people informed about the work of Cardiff Council. They will be administrating the app, moderating publishers and creating content based on government announcements. There is a concern that, as there are only a few of these workers that they may be biased, which is alleviated as all employees of CYC and CFC undergo anti-bias training. Many of Cardiff's young people struggle to find information about the council's work, and would be using our app as a first port-of-call when searching for updates about local developments.

1.5 Use Cases

The first use case is as an end user, who is trying to find information about their local council. As an end user, their goal will be to find more information about their council. Information can be found in one of two ways:

(1) **Search function**: if the user is searching for specific interests, they can use the search bar to find all the submissions containing their search using keywords.
(2) **Subscription based method**: if the user is planning to subscribe to certain issues/issue types, then they will be presented with all the submissions containing their subscribed keywords.

The information in each submission will contain a short video (10-15 seconds long) explaining the plan of action, and a short section of text summarising it, with a link to the source for those interested to read more.

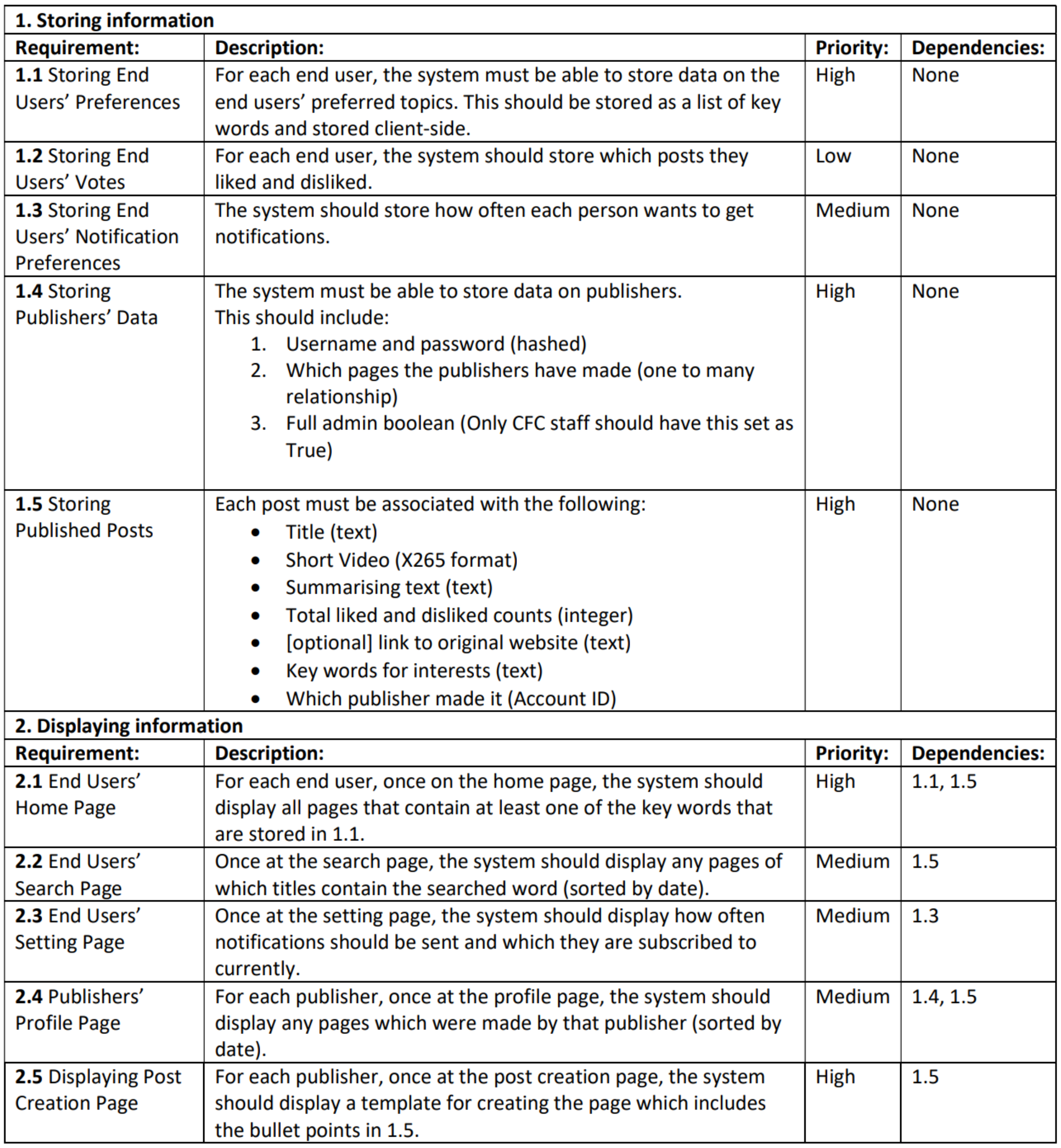
The second use case is a publisher, who is trying to publish information about their local council. As the publisher, their goal will be to engage young people in politics. Thus, they must be able to write up the information in an appealing way.

Given a website with information, the publisher must condense the information into a short section of text, and present it in a 10-15 second long video.

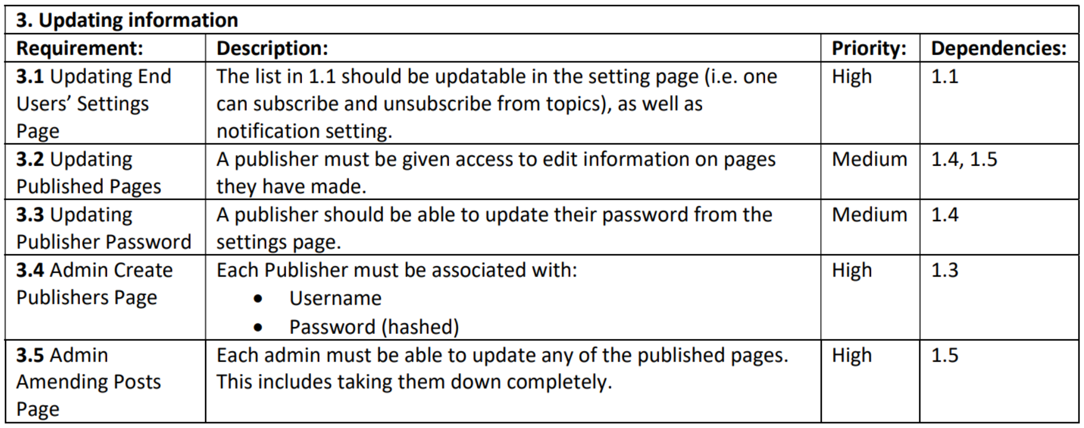
To ensure consistency, we will have a template which will include:

- Title
- A place to upload a video
- A place to post the summarised text
- An optional section for a poll for end users to vote
- A place at the end to post the link to the original website for those wanting to read more

1.6 Core Functional Requirements



1. Storing information			
Requirement:	**Description:**	**Priority:**	**Dependencies:**
1.1 Storing End Users' Preferences	For each end user, the system must be able to store data on the end users' preferred topics. This should be stored as a list of key words and stored client-side.	High	None
1.2 Storing End Users' Votes	For each end user, the system should store which posts they liked and disliked.	Low	None
1.3 Storing End Users' Notification Preferences	The system should store how often each person wants to get notifications.	Medium	None
1.4 Storing Publishers' Data	The system must be able to store data on publishers. This should include: 1. Username and password (hashed) 2. Which pages the publishers have made (one to many relationship) 3. Full admin boolean (Only CFC staff should have this set as True)	High	None
1.5 Storing Published Posts	Each post must be associated with the following: • Title (text) • Short Video (X265 format) • Summarising text (text) • Total liked and disliked counts (integer) • [optional] link to original website (text) • Key words for interests (text) • Which publisher made it (Account ID)	High	None
2. Displaying information			
Requirement:	**Description:**	**Priority:**	**Dependencies:**
2.1 End Users' Home Page	For each end user, once on the home page, the system should display all pages that contain at least one of the key words that are stored in 1.1.	High	1.1, 1.5
2.2 End Users' Search Page	Once at the search page, the system should display any pages of which titles contain the searched word (sorted by date).	Medium	1.5
2.3 End Users' Setting Page	Once at the setting page, the system should display how often notifications should be sent and which they are subscribed to currently.	Medium	1.3
2.4 Publishers' Profile Page	For each publisher, once at the profile page, the system should display any pages which were made by that publisher (sorted by date).	Medium	1.4, 1.5
2.5 Displaying Post Creation Page	For each publisher, once at the post creation page, the system should display a template for creating the page which includes the bullet points in 1.5.	High	1.5



3. Updating information			
Requirement:	**Description:**	**Priority:**	**Dependencies:**
3.1 Updating End Users' Settings Page	The list in 1.1 should be updatable in the setting page (i.e. one can subscribe and unsubscribe from topics), as well as notification setting.	High	1.1
3.2 Updating Published Pages	A publisher must be given access to edit information on pages they have made.	Medium	1.4, 1.5
3.3 Updating Publisher Password	A publisher should be able to update their password from the settings page.	Medium	1.4
3.4 Admin Create Publishers Page	Each Publisher must be associated with: • Username • Password (hashed)	High	1.3
3.5 Admin Amending Posts Page	Each admin must be able to update any of the published pages. This includes taking them down completely.	High	1.5

1.7 Core Non-functional Requirements

- **Usability**: To keep the user experience clean, the solution will not require logins to function. We will be using a library to provide screen-reader support on the website client. Many CSS libraries provide classes to designate text as screen-reader only. The W3 specification also provides ARIA attributes to enhance accessibility.
- **Reliability**: In this context, reliable can mean two separate things:
  - Reliable content. To ensure content is reliable and not manipulable, we need to prevent the ability for a man-in-the-middle attack to alter data in transmit. We will also have dedicated admins who can remove or hide incorrect or malicious posts.
  - Reliably accessible. The website will be run from a server with 24/7 uptime. Flask applications are very fast to reload for updates, so the solution will be very reliable in terms of its uptime.
- **Scalability**: Since in deployment our solution will only target Cardiff, there is an upper limit of about 1 million users. We can expect that these users will not be concurrent, and so will not need scaling. However, in the case of a larger scale deployment, the solution would need scaling. To accommodate this, a load balancer should be available optionally to divide the load across multiple servers.
- **Performance**: Performance targets two areas: the performance of the client, and the performance of the web server.
  - The client will naturally be thin, which means it should be performant. To ensure videos don't hurt performance too much, compression can be used and benchmarked and videos loaded lazily with a debounce to prevent unnecessary requests and decoding.
  - The two main performance effectors on the server will be database commits and static content serving. Caching of frequently updated values such as likes can be used to prevent high volumes of database commits, which are expensive. Storing of compressed versions of content can also be used to prevent frequent transcoding, which is a very expensive operation to perform.
- **Security**: Security has a wide range of potential issues:
  - The passwords of admins need to be secured in transmit and in storage.
  - The remote login to the hosting server needs to be sufficiently protected against malicious actors.
  - A web server gateway interface (WSGI) will be used to pass requests to Flask, as Flask's built-in WSGI server is not secure and not production-ready.

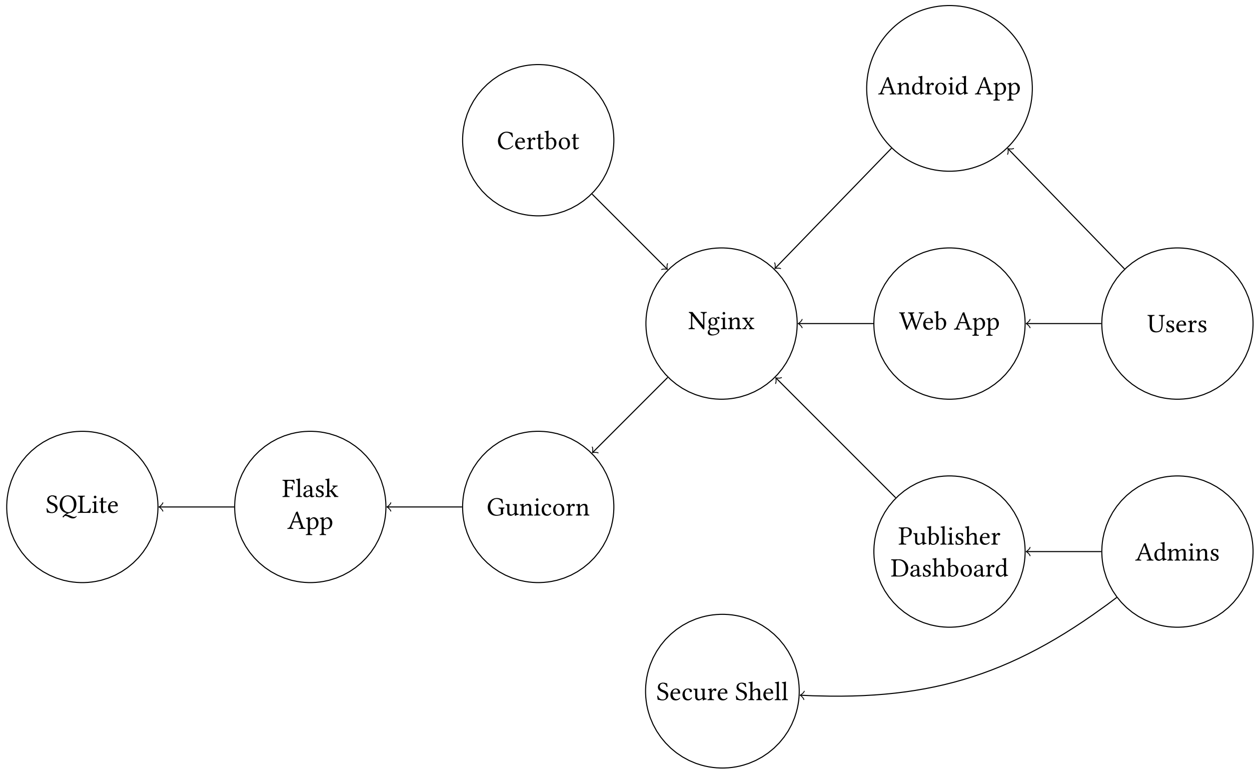
A conflict was reached concerning the storage of user data on the platform. Originally, an account system was discussed as a way for users to sync information between devices and to leave comments. However, there were multiple concerns brought up with this approach.

The young people from the CYC raised a concern for privacy regarding storing user data. This is particularly concerning as the system would be primarily storing user interests, which is data of high value to advertisers and other bad faith actors. Three potential alternatives were discussed before a decision was settled on.

- The first alternative considered was that interests could be stored as an encrypted BLOB. However, this requires significant trust that the system is acting as it claims, which would in any case be misplaced, as it relies on too many parties to act in accordance[4].
- The second alternative was to use 'share codes'. These would be QR codes generated by the app to share preferences between devices, similar to the Brave Browser's Brave Sync function. However, this system is overly complicated for sharing a limited amount of preferences: for contrast, Brave Sync syncs plugins, bookmarks and more, whilst our app only needs to sync filtering preferences. So overall, share codes would be more complicated than they're worth.
- The final alternative, and the idea that was settled on, is that the system will not use user accounts. Instead, preferences will be stored on the client side. This means that user data stays totally separate from the server. For features like 'liking', a global unique identifier can be generated and stored on the client. This connects a client to its likes.

2 ARCHITECTURE

2.1 High-Level Architecture



In our high-level architecture, we consider a number of important components of the full solution.

- **Nginx**: Nginx is a web server that communicates between the WSGI and the users. Nginx is important for a number of reasons:
  - Nginx can serve TLS certificates, to encrypt traffic and enable HTTPS. This ensures that transmitted data cannot be intercepted, and most importantly ensures the integrity of the transmitted data. This satisfies the requirements of reliability and security.
  - Nginx is high-performance and can be used for load-balancing. This means that, if necessary, more instances of the WSGI/Flask Program can be run to process requests in parallel, potentially across multiple servers. This increases the scalability.

  Nginx is free and open-source, and very popular[3]. Therefore, it is an ideal candidate for our web server. The alternative would be to use Apache. Whilst Apache is equally well-known, Nginx is more modern and easier to configure. Configuration files for Nginx can be shipped with the solution, which will make it very simple to set up.
- **Gunicorn**: Gunicorn is the WSGI, which passes requests from the web server to Flask to be processed. For security reasons, a WSGI is *essential*: Flask explicitly states that a WSGI must be used in live deployment[2]. This is primarily due to security, but also offers performance benefits. The WSGI can manage additional threads for the Flask application, and can also use event frameworks to process requests asynchronously or run background tasks.

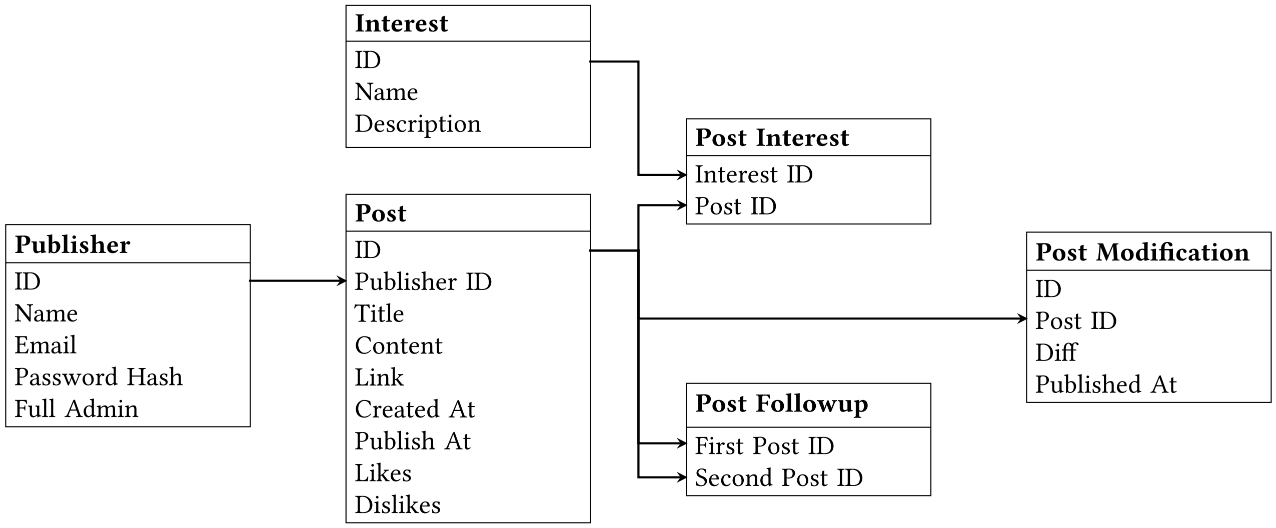
  Flask offers a number of suggestions for the WSGI. Gunicorn is very easy to install, as it can be installed via PIP (the Python package manager), and often also ships as binaries available in the Advanced Package Tool on Debian-based Linux distributions. This makes Gunicorn the best option.
- **Certbot**: Certbot is a program provided by LetsEncrypt for generating HTTPS certificates.

  HTTPS certificates are essential as stated previously. Certbot has the benefit of being free to use, and generating certificates that are signed by a recognised authority. This makes their certificates trusted in all modern browsers, enabling seamless access for end users.

  The disadvantage of Certbot over a premium certificate authority is that Certbot certificates have a very short expiration time of 3 months. This means that admin work would be frequently needed to refresh the certificates and prevent them expiring. To reduce this, a cronjob could be configured on the server to automatically refresh the certificates, but some admin may still be necessary to keep the connections secure in case the cronjob runs into an error.
- **Secure Shell**: Secure Shell (SSH) is a method for system administrators to maintain the system. This will be how updates are installed and reboots/patches applied. SSH comes as default with most headless Linux distributions, and SSH keys can be configured to ensure security.

  Other security measures for SSH include Fail2ban and UFW. UFW is a firewall that can be used to block connections to unnecessary ports. Fail2ban is a simple program that utilises iptables to block bruteforce attacks. This fulfils a non-functional requirement of system security.
- **SQLite**: SQLite, as discussed in deliverable 1, will be our DBMS.
- **Flask App/Android App/Web App/Publisher Dashboard**: these are the end products, as discussed throughout this document.

2.2 Database Schema

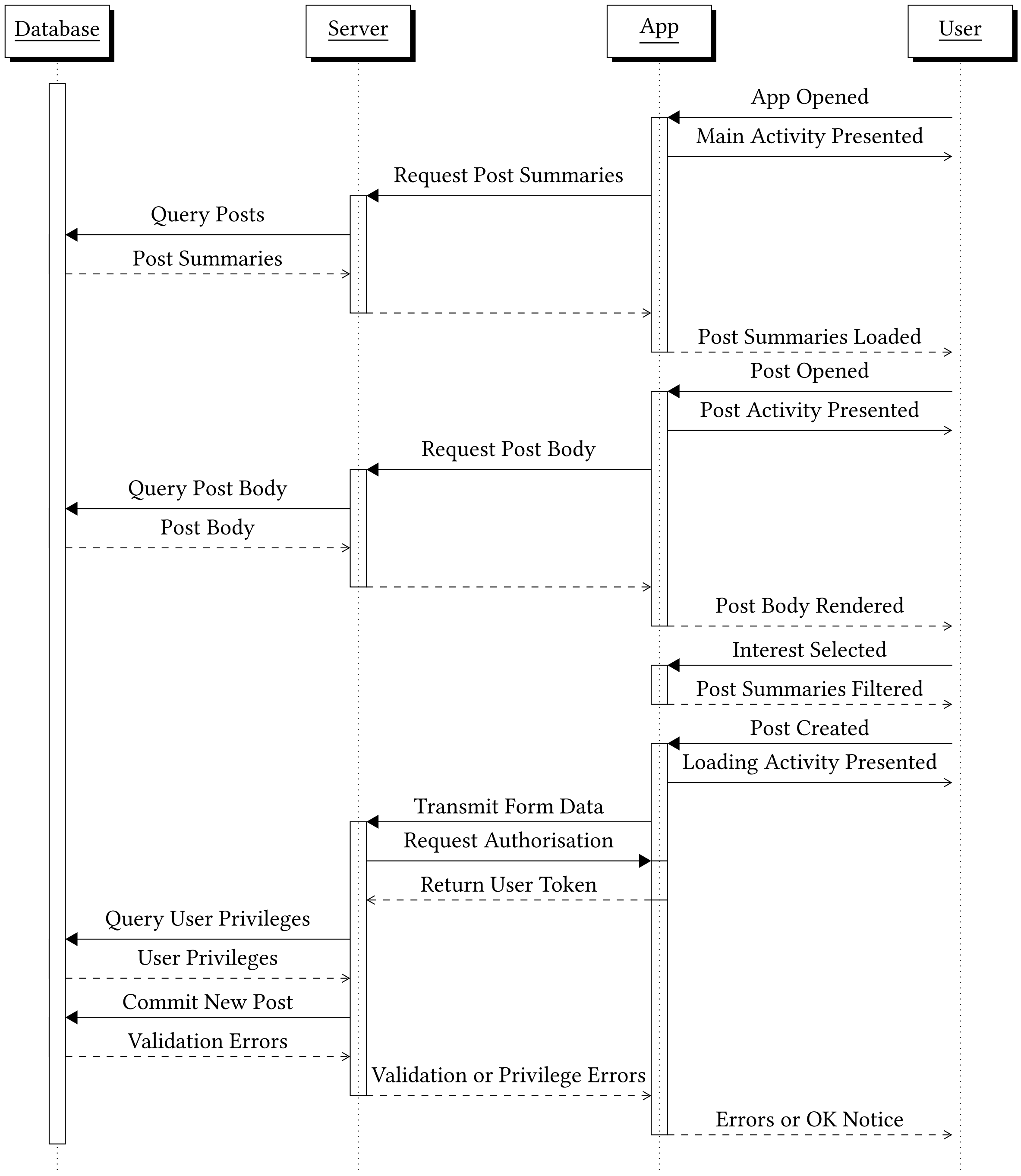


The first UML diagram is a relation diagram describing the structure of the relational database that will be used to store persistent data. Each arrow signifies a foreign key relationship between two tables on the connected fields. These relationships will behave in slightly different ways:

- All foreign keys on `Admin.ID` will **SET NULL**. This ensures that if an admin is deleted, posts created by them will not be removed.
- All foreign keys on `Interest.ID` will **CASCADE**. Hence, if an interest is removed, users will have the interest removed from their feed and posts will be untagged from the interest.
- `(Post Modification).ID`, `(Post Followup).(Second Post ID)`, `(Post Interest).(Post ID)` and *(Post Subscription).(Post ID)* will all **CASCADE**. This ensures that if a post is removed, no redundant data is left behind.
- `(Post Followup).(First Post ID)` will **RESTRICT**. This will prevent a post being deleted if it is has another post referencing it. This helps to maintain the integrity of the content delivered.

The `Post` relation has two timestamps: a `Created At` timestamp and a `Publish At` timestamp. The `Created At` timestamp will describe when the post was initially created, and the `Publish At` describes when the post will be available to the end-users.

The `Content` field of the `Post` relation is discussed in the *Core Functional Requirements* section.



The second UML diagram shows a sequence diagram of the primary interactions between the clients and server.

The first user interaction describes the opening of the app by the user. Opening the app will immediately display what Android describes as the **Main Activity**. A 'loading' message will be displayed, as the app sends requests to the server to fetch posts. Then, the post summaries will be displayed on the **Main Activity**.

On a post being opened, a very similar interaction will take place, where the body of the post will be fetched from the server.

For interest selection, no requests are sent to the server. These are stored on the client side, for privacy concerns.

The final block displays a sequence for users who are trying to create posts. There are more requests made here, to validate the user and check their authorisation. This involves querying the database for their privileges, and validating any input data. These errors will then need to be presented back to the user, if present.

This sequence diagram satisfies the primary functional requirements of the young people in a handful of app activities. Other functional requirements like voting will be provided via buttons on the interface.

3 PRELIMINARY DESIGN & PROTOTYPING

From our consultations, we ascertained that our users are likely to want regular notifications from the app (with the frequency of such notifications being customisable), however, they are not likely to want to receive regular emails, or join any mailing lists. They also liked our ideas about being able to select interests, thereby making the notifications more personal to them.

We also discussed the format of the content on the app. We decided that posts should be able to include text, images, links and videos. In our consultation with the CFC, they agreed that they would be able to regularly create small 10 second summary videos for each article; this was also something the young people were keen on.

Our next big focus in the conversations with our stakeholders was trust and privacy. The young people were indifferent to the concept of being able to set up accounts, but the CFC wishes to not store any user's personal data and so we will not go ahead with implementing user accounts into our solution. All user data is to be stored client side, such data will not be personal, and will only be used to better curate content for individuals.

Our solution is that there will be three levels of user:

- **CFC Admin**: they will be able to make and edit their own posts as well as delete or archive other posts for moderation purposes. They can also create new publisher accounts.
- **Publisher**: they have the power to create and edit their own posts. To be given to trusted sources by the CFC to keep local people in the loop about local events.
- **End user**: the majority of users fit into this category, they have no power to create or edit posts and can only view the content that has been uploaded, share content and like content. None of their data is stored server side.

The lack of user accounts removes the possibility for a comment system but this reduction in functionality will both mean that privacy of users is protected, as well as making the app easier to maintain.

The addition of an admin account will help to ensure that the information provided will be credible (an issue brought up in our consultations). The admin account will be controlled by members of the CFC, and be available through login on the website client. The admin account will have the functionality to create and delete publisher accounts which will allow only trustworthy sources to publish information on the app as well as provide a way to regulate publishers that may misuse the platform. The CFC, being an reputable charity, will moderate the platform.

Based on the young people's recommendations, we will add functionality to allow local events to be posted by trusted organisers. This has been translated into features by allowing the admin account to create new publisher accounts that can be given out, upon request, to trusted local events organisers so that they can post events to the app.

3.1 Interface Design

Aside from the discussions of the core functionality of the app, we also consulted with the stakeholders on what sort of UI would be best for our app. We presented the user stakeholders with three simple prototypes of potential interfaces:

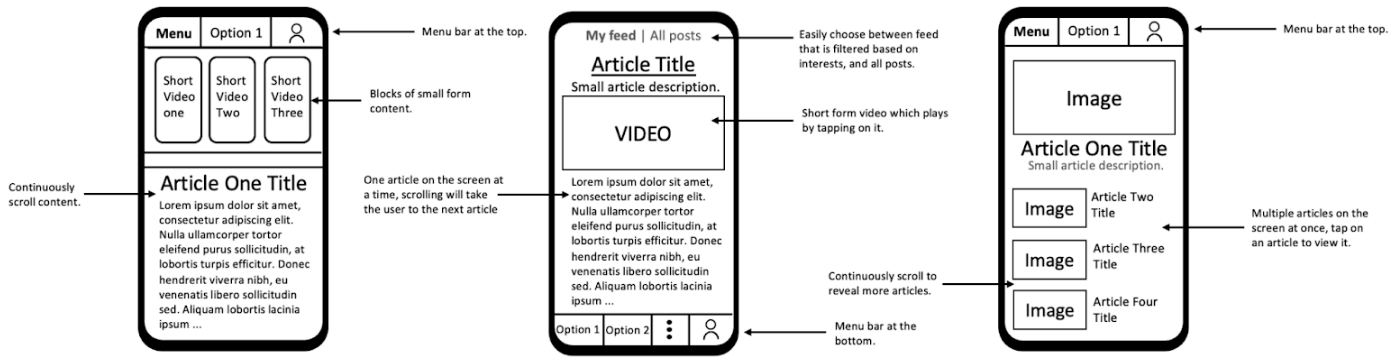

Menu
Option 1
Short Video one
Short Video Two
Short Video Three
Article One Title
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ullamcorper tortor eleifend purus sollicitudin, at lobortis turpis efficitur. Donec hendrerit viverra nibh, eu venenatis libero sollicitudin sed. Aliquam lobortis lacinia ipsum ...
Menu bar at the top.
Blocks of small form content.
Continuously scroll content.
One article on the screen at a time, scrolling will take the user to the next article
My feed | All posts
Article Title
Small article description.
VIDEO
Option 1
Option 2
Easily choose between feed that is filtered based on interests, and all posts.
Short form video which plays by tapping on it.
Menu bar at the bottom.
Continuously scroll to reveal more articles.
Image
Article One Title
Small article description.
Article Two Title
Article Three Title
Article Four Title
Menu bar at the top.
Multiple articles on the screen at once, tap on an article to view it.

Fig. 1. Left to right: Facebook, Instagram/TikTok, BBC news

The first design was based on the Facebook app. The design utilises mixed media blocks with scrollable content. This design was unpopular, as Facebook's interface is unfamiliar to many younger users, and mixed-media makes for a very cluttered experience.

The second design was based on Instagram, with large media content with text to caption it. This design was the most popular. The young people were familiar with this style of social media interface. Having one main article on the screen at a time is easy to navigate and understand.

The final design was a 'news app' style design, modelled after the BBC News app. The design was somewhat popular, though the young people preferred a social media style interface. Some elements of this design will be incorporated into the final design , such as the ease of navigability.

Therefore, going forward, the second design will be the basis for our UI.

4 PROGRESS REPORT

4.1 Challenges Encountered

As a result of our discussions with our stakeholders, we found the following challenges:

(1) The app will not have a broad appeal if it only contains purely authoritative information. They want to have a more centralised system to use for multiple purposes (e.g. events)

(2) Allowing users to comment on our posts means we will have to be able to moderate comments.
(3) Publishers who are allowed to create content will also have to be verified, otherwise, anyone could add potentially irrelevant or harmful content to our app.
(4) Requiring users to create profiles will require us to store personal information, which could cause issues regarding data protection, but without profiles, users would not be able to access their preferred content across multiple devices.

4.1.1 *Corrective Action Taken.*

(1) Following our discussion with the young people of the CYC, we decided to expand our scope to include local events of interest, such as music festivals, markets, and sports events, which would be removed once the event had occurred.
(2) After lengthy discussion, we have decided not to include commenting on posts as part of our app, as moderating these comments would require time and resources that would be strenuous to the administrator from CYC.
(3) During our discussions with our stakeholders from CFC and CYC we decided that we will have three levels of access: a basic user, publishers, and administrators. Only the administrators from CYC will have the administrator level of access, and they will be able to verify new publishers when they want to join the app, and possibly remove publishers' privileges if they post unacceptable content.
(4) We have decided not to require users to create profiles, as it should take less than a minute for users to input their interests when installing the app for the first time on a new device, and ensures their privacy is protected in the best way by simply not requiring their personal data. For publishers and administrators, however, accounts will be necessary to ensure security, so that no one unauthorised can add content to the app.

4.2 Progress & Departure From Targets

We have progressed strongly since November 15th, having refined our requirements, and developed and reviewed our front-end and back-end designs. However, due to our focus on design, we are yet to begin development of the app and website as expected by the Gantt chart.

As a result of this, we will have to make slight changes to our plans as we progress. We will be focusing heavily on the development of the app following the 6th of February, once University examinations end.

4.3 Approaches, Methodologies, and Risks

Our focus on group discussion rather than design and development has resulted in having to move our plans of initial development later into the year. Many members of the group spent much of their time uncertain about what areas of development they were working on.

The risks described in deliverable 1 (falling behind on work and failure to work in tangent) remain the same, although it is clear that, having already failed in the former, we shall have to be more vigilant in the future. We will do this by assigning tasks to individual members of the group earlier into the development phase so that each member can focus their efforts into achieving results rather than worrying about larger scale design ideas.

There is also a further risk that we have not previously considered: many members of the group will have to learn new technologies for this project, which will be time consuming, and possibly beyond the capacity of some of our members during the current time constraints. Additionally, working with new software means there is a higher risk that critical or non-critical mistakes may be made while programming. To mitigate this, we will attempt to learn these technologies before the programming phase, to reduce time pressure, and ensure that all code is checked by at least two members of the team, and assist each other with debugging new languages to reduce time spent solving simple syntax errors. This can be achieved by using the 'Request Review' feature of pull requests on GitHub. We have also for the most part chosen technologies that at least one member of the group is already fluent in, with the exception of Kotlin.

REFERENCES

[1] [n.d.]. https://www.cardiffldp.co.uk/2021/05/local-development-plan-review/. Accessed 24-Nov-2021.

[2] Flask. [n.d.]. Development Server - Flask Documentation. https://flask.palletsprojects.com/en/2.0.x/server/. Accessed 23-Nov-2021.

[3] Tonino Jankov. [n.d.]. Nginx vs Apache: Web Server Showdown. https://kinsta.com/blog/nginx-vs-apache/. Accessed 23-Nov-2021.

[4] Jim Salter. [n.d.]. https://arstechnica.com/information-technology/2021/09/privacy-focused-protonmail-provided-a-users-ip-address-to-authorities/. Accessed 13-Dec-2021.