ePortfolio

Project Documentation Portfolio B

Joel Grimmer, Pragya Gurung, Linda Lomenčíková, Yuxiang Hu, Disen Hu

Overview

Client

The ePortoflio web app is to be developed for our client, Dr Paul Harper, who is a lecturer in the Department of Civil Engineering at the University of Bristol. As a lecturer, he is concerned about how students document their achievements over time in a way that will aid them when looking for future employment. Dr Harper hopes to help students who do not have prior experience making a portfolio, create their own personalised ePortfolios.

The Domain

Dr Harper has spoken to PhD Computer Science students who have coded their own online portfolios and he noted how effective they were at showcasing the students' skills and experience. However, this is not possible for the average student to accomplish and it can be daunting to start a portfolio if it is the students' first time. Dr Harper requested that the ePortoflio tool should be simple to use and minimalist in design.

The online portfolio tool should enable students to catalogue and reflect on their academic and extra-curricular work over time. Whilst also allowing them to critically evaluate their achievements and gain a deeper understanding of project work by connecting it to their previous work. Finally, Dr Harper wishes for the ePortfolios to be easily shareable with just a link.

The Problem

The University of Bristol has a tool on Blackboard called Personal Development Planning (PDP) that guides students on how to make their own portfolios. However, our client has stated that this tool is both inaccessible and grossly underdeveloped to the point of near uselessness. Our client believes there is a need for a more ergonomic and streamlined portfolio creation tool.

The main issue with the tool lies in ensuring that it caters to the majority of students' needs. What a photography student will want in a portfolio may be completely different to a computer science student. Our solution was to make it universal, there are 4 main categories that all students would need: Projects, Work experience, Qualifications and Achievements.

We opted for a web-based app for increased accessibility as it operates from within a browser. It can also adapt to whichever device it's being viewed on and can be easily updated by the developer.

Our Vision

We plan to create a web-based experience that students can log onto and access via any standard web browser, according to our client's parameters.

- The user can create "posts" and categorise them as project work, work experience or achievements etc.
- Users can attach files to posts and customise the post using different font styles and headings.
- The posts will be able to be drafted and so will not appear in the published portfolio. This gives the student the ability to continue to work on their portfolio whilst having a finished version that they can share with potential prospective employers.

We used Spring-Boot and React.js with RESTful API to develop a fully functioning ePortfolio web app. Afterwards, we streamlined features to make the interface more user-friendly by completing testing, doing surveys and receiving client feedback.

Requirements

Primary Stakeholders and User Stories

Dr Paul Harper (Client) - Lecturer at the Department of Civil Engineering at the University of Bristol and he is a primary stakeholder. He initiated this project, put forward the basic specifications, and gave feedback throughout the process of this project. His goal is the ability to:

- Create a tool that will help students easily make ePortfolios.
- Continue the development of the web app to add more features.

Web App Users (students)* - Students are the main target demographic for ePortfolio. They will use the web app to create their portfolios to showcase their achievements and experiences, and then share them with potential employers when applying for jobs.

- ➔ A student who wants to use ePortfolio to make a portfolio to record all of their achievements and projects to showcase their abilities.
- ➔ They would need to be able to upload "posts" about their projects/work-experience and categorise them so that it's easy to access.
- ➔ If they want to work on the "post" later, they should be able to save it as a draft.
- ➔ A student should be able to share their published portfolio with a link. With this link, anyone can view the students' work.

Potential Employers - An employer who received an ePortfolio link sent by a student. They will be able to look at the portfolio to understand more about the student. An employer could use the information in the portfolio to determine whether to invite the student in for an interview.

- ➔ An employer may view an interviewee's ePortfolio so that they can better understand their skill level and past experiences.

- ➔ They would want the portfolio to be easy to navigate so that they're able to clearly see exactly what the student has achieved/worked on.
- ➔ The employer should have an accessible way to interact with a prospective employee's portfolio without having to create an account.

Additional Stakeholders

Tutors - Students can give a link to their portfolio to their tutors so that they can get feedback. It would allow their tutors to easily keep track of what the student has done and give suggestions to improve. Similar to the Potential Employer, they would require the user's portfolio to be easy to navigate.

University of Bristol - Our client mentioned that the tool could be adopted by Uob and be integrated with Blackboard. This positively impacts UoB as it would make the tool easily accessible to all students including first years, thus encouraging them to start developing a portfolio early.

Primary User Requirements

Our primary user story* can be broken down into three sequences of steps as shown below:

Basic flow

1. The student creates an account and logs in with their credentials on their laptop.
2. The student adds a new post to their ePortfolio
    2.1. Start a new post by clicking on the "⊕" button on the top of the post section or the "Post" button in the navbar.
    2.2. The text editor will be open.
    2.3. Write the content, add files and categorise the post e.g. "Projects".
    2.4. The student finishes writing the post
        2.4.1. The submit button is clicked and the post will appear in the published portfolio.
        2.4.2. The draft button is clicked and the post is drafted. It will not appear in the published portfolio.
        2.4.3. The cancel button is clicked and the post is deleted.
    2.5. Another new post can now be added.
3. Share the Portfolio with other people.
    3.1. Click on the "Publish" button and copy the URL to the published portfolio.
    3.2. The student can share the link with others.

Alternative Flow

1. The student enters the web app on their phone.
2. The student writes a post but does not finish it.
3. The student saves the post as a draft, which will not be shown in the published version Portfolio.
4. The student continues to add to their portfolio.

Exceptional Flow

1. The student attempts to log in.
2. The credentials are not correct, so an error message appears.
3. The student is asked again to log in.

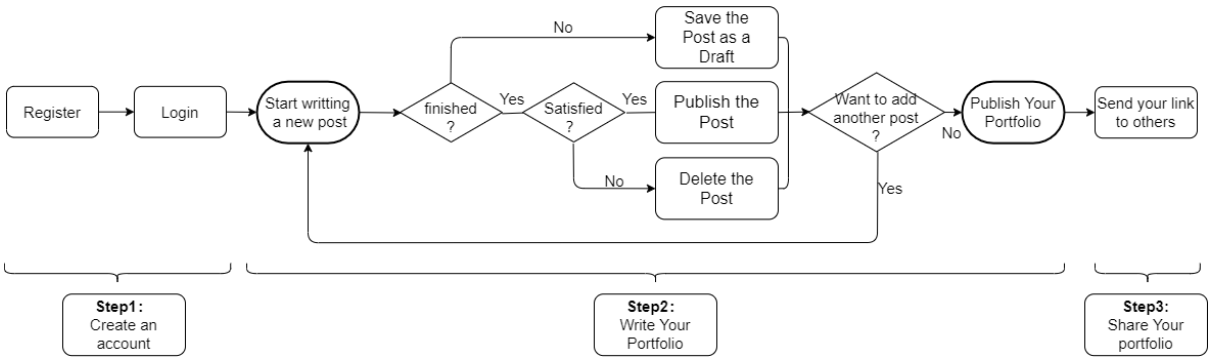

Figure 1: Flow Map for the Primary User Story

Atomic Requirement

Table 1: Functional Requirements

Version	Description
1	**Users can log in with their account with the correct credentials**
1.1	Ensure only the user with the correct credentials/access token associated with that account can edit the portfolio
2	**Users can edit/add to their ePortfolio**
2.1	Clicking on the “post” or “⊕” button should take you to a text editor page
2.2	Must be able to attach any type of file to the post.
2.3	Must be able to customize font styles and headings.
2.4	Each post must have an option to categorize.
2.5	Users must be able to edit their personal information and about me.
3	**Must allow the user to publish their portfolio**
3.1	Posts that are drafted, do not appear in the published portfolio
3.2	Must have a “publish” button that takes the user to their published portfolio which they can show others with a link
4	**Comply with the GDPR**
4.1	The terms & condition checkbox must be checked when registering an account
4.2	Setting page added where users can re-read the T&C and privacy policy

5	**The web app must be widely accessible**
5.1	The web app must run well on any standard browser.
5.2	Ability to use it on mobile phones by making the web app fully responsive.
6	**Must pass all Front-end tests & Back-end tests**

Personal Data, Privacy, Security and Ethics Management

Collecting Personal Data

To access the ePortfolio tool, the user must create an account by providing their full name and an email address and creating a username and a password. We store this information in a database.

The user can only successfully register after checking two checkboxes and therefore agreeing with the two statements associated with them. The first statement is ”I agree to the Terms and Conditions”. Here, by clicking on the words “Terms and Conditions” the user is directed to a separate page where they can read in detail the terms and conditions they are agreeing to. Here it is explained how we use and process the user’s data. The second statement the user checks is “I consent to the processing of personal data”.

The only personal data collected and stored is the data given by the users themselves upon making their account - their first and last name and their email address. The portfolio is referenced by the username created, the user does not have to share their name nor their email address on the portfolio for other people to see. However, the user can choose to share it along with any other personal information.

We make sure the user is aware that everything written in their portfolio (apart from drafts) can be seen by other parties with a link. We warn them that we store all the information they further provide, even personal information, in our database. This is also detailed in Terms and Conditions.

The person with the correct credentials will be the only one who can modify what is written in the portfolio associated with that account. The user chooses what will be visible in the published portfolio and what will be stored in their drafts. They then have the ability to share a link to their published portfolio with any individual they choose.

GDPR

To process and collect user data lawfully, we adhere to the General Data Protection Regulation (GDPR). The laws require us to get consent from our users to collect and store their data, which we do upon registration. We also explain how the data will be used. Our registration form has a separate “I consent” confirmation. It is opt-in, meaning the box is unchecked by default when registering.

Privacy
The ePortfolio website does not disclose, use, copy, publish or modify in any way the stored user data. The data is solely available to the Development team which has to guarantee further privacy of it.

Cookies
Cookies maintain data that pertains to the user, and it resides on the user's computer so that it gets loaded whenever they come back to the site. In our case, the user must be logged in to use the website. After they successfully log in all the information they saved is rendered from the database. The user stays logged in until they log out or until the login token expires. This means we do not require the use of cookies after the user logs in.

We opted to use token-based authentication, which is stored in *localStorage*. We chose tokens instead of cookies, as token authentication is stateless. Meaning the backend does not need to keep a record of each token, because they are self-contained, containing all the data required to check its validity.

Security
Spring Security was used to manage access levels to our API and to enforce data protection standards. Upon signup, we hash user passwords using our `BCryptPasswordEncoder`, which uses the BCrypt strong hashing function, with a strength value of 10. Only hashed passwords are ever stored in our DB, to prevent compromising our users' plaintext passwords in the event of a database breach.

We implement the `WebSecurityConfigurerAdapter` interface to manage our web security chain. We use a cors configuration to allow interfacing between our front and back ends as they reside in different domains. We disable CSRF as our backend does not serve any web pages itself. We permit non authorized users to access public paths such as "`/signup`" and "`/users/**`" so that people can view the user's portfolio.

We use JWT for session management, and implement Spring Security's `UsernamePasswordAuthenticationFilter` and `BasicAuthenticationFilter` to use JWTs for authentication and authorization. We use these filters to associate an incoming JWT Bearer Token in the authentication header with a username and password and to generate a new JWT when provided with valid credentials. We use Auth0's algorithm package to generate JWTs with the HMAC512 algorithm. The JWT authentication and authorization filters are added to our web security chain to authenticate appropriate requests.

Ethics
Ethics pre-approval was applied for on the 22nd of November 2020 at 5:18 pm

Architecture

The ePortfolio will be delivered as a web application at the request of the client, as opposed to a native mobile application. We will be using several libraries and frameworks to achieve a fully working web application. Figure 2 shows a high-level diagram of the structure of our web app.

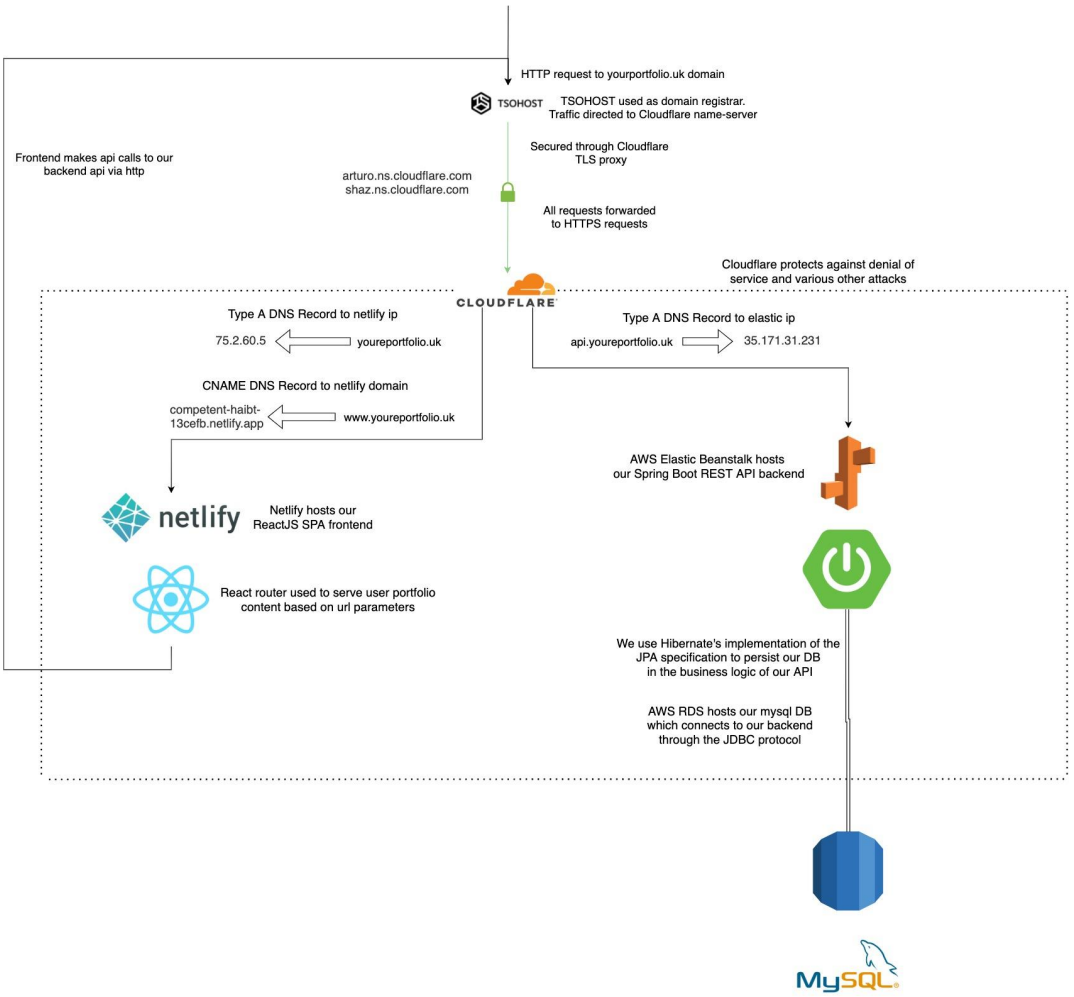

Figure 2: High-level diagram of web application

Domain and Deployment

To deploy our web app, we purchased our own domain name through the registrar TSOHOST due to time constraints. Then we pointed it to our Cloudflare nameservers and used Cloudflare proxy-ing to protect our front-end and back-end. Cloudflare's proxied TLS encryption allows us to encrypt our front-end and backend automatically without the need to have custom cron jobs working on both ends to maintain our certificates. TLS is a necessity as it prevents user credentials from being exposed when our front-end sends a login request to our backend. Unfortunately, AWS Educate blocks AWS certificate manager, which is the only meaningful way to set up TLS on an EB instance as the contents of the instance are wiped with every restart. This makes a local let's encrypt difficult to manage. As a result, making the use of Cloudflare proxy TLS is a necessity.

We point the (sub)domains youreportfolio.uk and www.youreportfolio.uk to their respective netlify addresses, and the subdomain api.youreportfolio.uk to our backend's elastic IP.

Back-end

We use Spring Boot to create a RESTful API where HTTP. Requests will be sent to our model, authenticated, and then structured data is returned. In our case, the structured data is JSON/HAL for ease of parsing with our front-end. This allows for separation of concerns, with our spring boot application providing a secure wrapper for our database that manages all CRUD (Create, Read, Update, Delete) actions. Additionally, we are using `@RESTController` for our response controllers to make sure they are fully REST compliant.

We achieve Hypermedia as the Engine of Application State (HATEOAS) through the use of Spring's HATEOAS library. The library allows us to insert links between different requests to the REST API. As a result, it allows us to access Level-Three Richardson Maturity. We use the HAL format to return both the request body and the associated links for the request. For example, also returning a link to all posts by a certain user when requesting said user's details.

Database

Our database will be hosted as a mySQL database. We have opted for a relational database due to Spring's strong integration with the relational model, which ensures we don't have to write raw SQL requests to the server. This improves security as it prevents us from spending a considerable amount of time making sure SQL injection does not take place. Furthermore, we use Hibernate to persist our database, and JPA repositories to model our tables.

Front-end

We used React.js to drive the front-end of our web application. We used client-side rendering to prevent too much work from being done on the server end of the application and to make sure our application is interactive locally. JavaScript provides the *fetch()* function which we used to access our REST API, and the application updates the page on receipt of the relevant data when loaded.

We used React Router to maintain a seamless user experience and a clean SPA to prevent all server-side rendering, similar to how Facebook and Twitter manage web page state. To allow access to custom rendered components, such as an individual user's portfolio, we used JS query strings.

Additionally, we used Github Actions to check our back-end's maven build and test status, and our front-end's node.js build and test status. This allowed us to automatically test and build our project very efficiently. We used Github actions as opposed to other CI/CD pipelines such as CircleCI as one member of our group had to experience using it, thus making the setup easy.

OO Design & UML

Static UML

We wish to represent the relationship between our backend components through a series of static UML diagrams. An overall diagram showing all connections between all classes becomes large and unwieldy, and hard to read due to its size, so instead, we have opted to illustrate some common user actions through UML. These actions include getting a user's details and uploading a post with an associated asset file.

When modelling our backend structure, the spring framework's heavy use of dependency injection makes it important to represent object instantiation at runtime. Dashed lines represent dependency injection (at runtime), while solid lines represent the structure and interaction between objects at compile time.

Get User Details

Getting user details is the most important API endpoint in our application and as such, we decided to create a static-UML diagram (Figure 3). Getting these details allows us to both display a user's profile information but also provides the links from which we can get the user's posts. We expose two endpoints for getting user details, "`/users/{id}`", and "`/u/{username}`" which gives the front-end team two easy way of getting what they need depending on what data they already have. These endpoints are associated with the methods `userById` and `userByUsername` respectively. We have omitted security associated classes from our diagram as we wish to maintain clarity of our concepts.

We implement the `RepresentationModelAssembler` interface with our `UserModelAssembler`. The latter uses the `toModel` method to transform a `UserModel` JPA representation into an `EntityModel` which can display HATEOAS links to the client. This entity model is the data structure that is returned from our backend on request.

Our `UserModelRepository` interface extends the `JpaRepository` interface. The associated methods from each interface are used to persist our `UserModel` objects in our database. Our `UserModel` is a JPA `@Entity` object meaning we can define SQL related parameters such as the names of its parameters, in the table.

The `UserController` instantiates a `UserModelAssembler` and a `UserModelRepository` implementation at runtime using dependency injection, as represented by the dashed line between the objects.

Figure 3: Get user details static-UML

Uploading a Post with Assets

We chose to create another static diagram (Figure 4) for this specific component because it is a crucial part of our web app that enables users to start building their portfolio. We have omitted the parent interfaces of our repository and assembler entities for clarity's sake.

`PostController` uses the method `addPost` to consume a POST request to the "`/post`" endpoint and receives a `Post` object in JSON format. This uses the post model assembler to produce an entity model of the post. The asset model controller contains the method `getUpload`, which takes the id of the relevant post. This method makes use of the S3Utils wrapper to interact with the AWS S3 API, a cloud object storage for our assets.
The method `getUploadLink` takes a string and returns a signed link for the browser to upload to. This link is then saved as an `AssetModel` in the `AssetModelRepository`, with a reference to the link and the post it is associated with. The link is returned as a response to the query, ready for the client to upload any type of file.

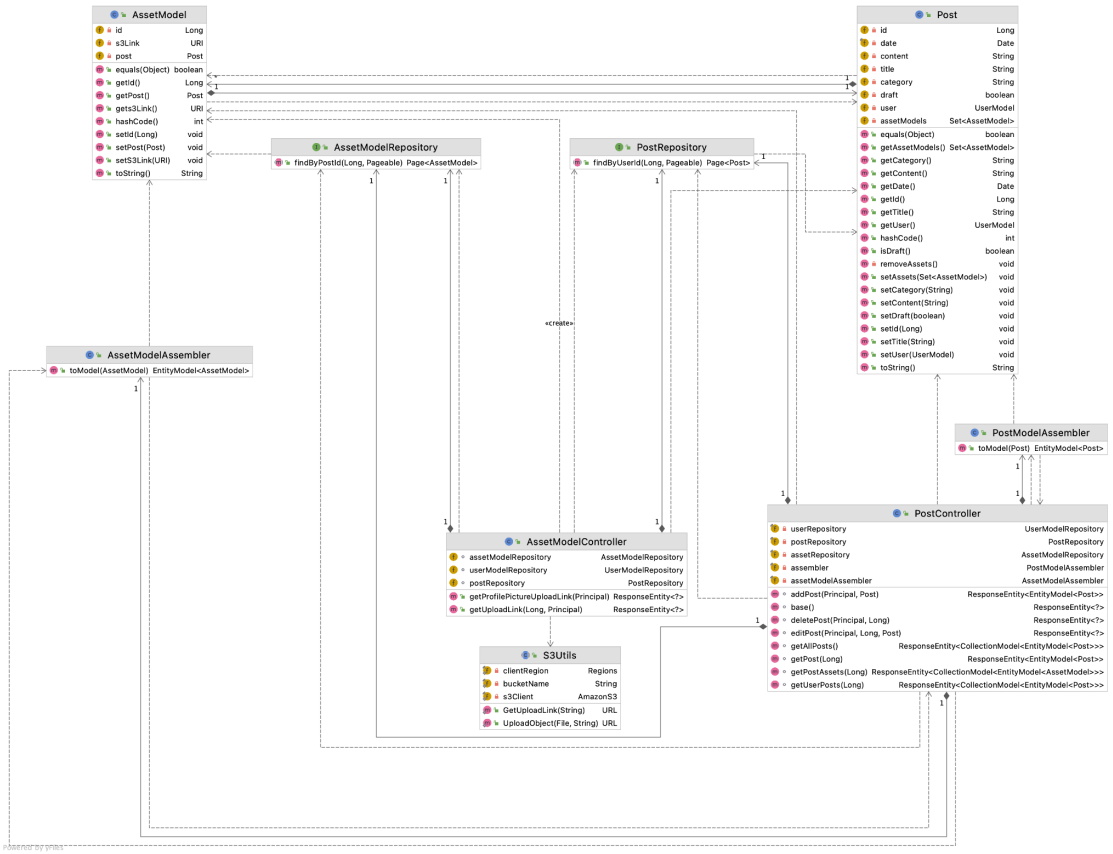

Figure 4: Post and asset upload static-UML

Dynamic UML

We demonstrate a common user action, adding a post to a portfolio(Figure 5). The flowchart below shows the various decisions that can be made by the user when making a post. We chose to represent publishing a post as it is the most common action taken by a user when interacting with the web app and is tied to our primary user story.

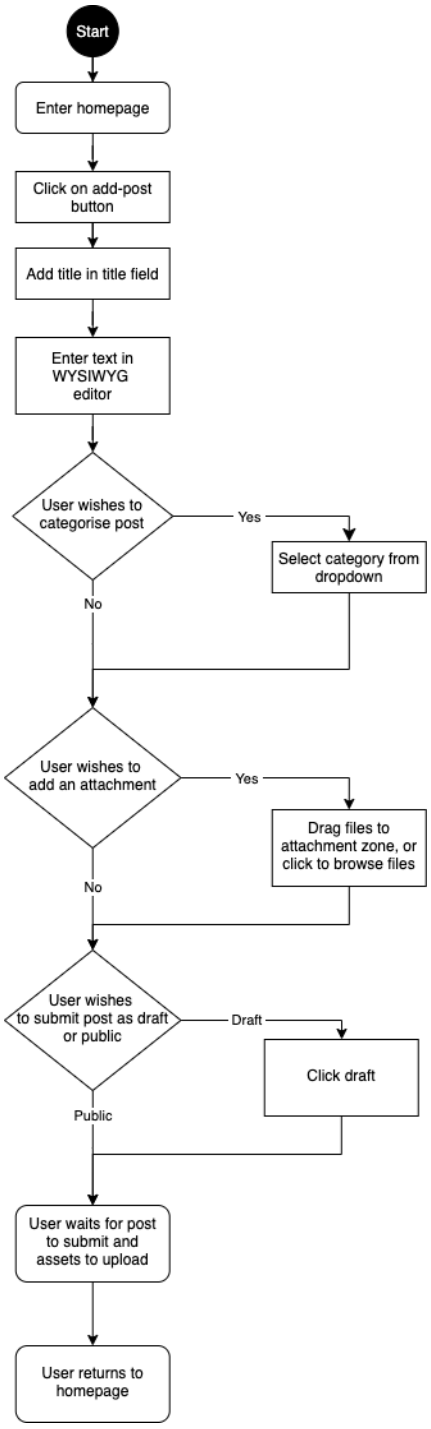

Figure 5: dynamic-UML diagram

Development Testing

The TDD (Test-Driven Development) strategy was used throughout development. We used Postman, which is a powerful HTTP request debugging tool, to debug our API. As for continuous integration, Github Action is used to ensure any new changes of code do not break existing functions and to identify any defects in the code efficiently.

For testing our front-end (REACT), we will use a unit testing framework called Jest. Jest has a tool called Snapshot that essentially creates a 'snapshot' of your system and stores it in a separate file. This will keep our tests lightweight and prevent regression. We have five snapshots in total, they are to ensure the pages render correctly so that the UI is in the right place.

Unit tests for the backend are written with Junit, a Java unit testing framework, which is pre-installed in IntelliJ. We tested "Find the user by username" and the REST controllers to make sure that the correct portfolio and user details are retrieved.

Table 2: Development Testing

Test Case	Input	Expected Results
Pages render correctly	"npm run test"	Pass - The new snapshots match the previous snapshot.
Find user from database by username	**userModelRepository**.findByUsername("joelgg");	Pass - The user found from the repository is equal to the one added at the start of the test, under the same username
User, Post and Asset REST controllers load properly	n/a	Assert that `userController`, `postController` and `assetModelController` are not null when autowire-loaded

Release Testing

On release, we will need to test to see if the intended features are present and are functioning correctly. To do this we focused on one user story and broke down the steps and manually checked each component. This ensures the developers' prior knowledge about the system does not affect the testing. In addition, we will invite third parties to also carry out the release testing to help us identify issues we may have overlooked.

The release tests were based on our primary user story. We choose this particular story because the main target demographic for our tool are students. They are the ones that will interact the most with the web app, therefore, at some point, they will encounter all the features. As a result, one of our key aims is to ensure the web app is easy to use for students.

Table 3: Release Testing

Test	Action	Expected Result
Able to use the webapp on any standard browser & device	Open the web app on any standard browser on any device.	Web app landing page should successfully load. The student should be prompted to log in or sign up.
Can add a post	Click on the plus button or the “post” button and fill in the text editor. Then submit.	When the buttons are clicked, a text editor should appear. Once filled in and the user clicks “submit”, the post should appear on the front page of their portfolio and in the correct category
Can draft a post in text editor	Instead of clicking “submit”, click “draft”.	The drafted post should appear with a grey left border. In the published portfolio, it should not appear at all.
Can edit/draft/delete posts using the 3 buttons on top of each post.	Click on the “edit” icon. Click on the “draft” icon. Click on the “delete” icon.	“Edit” should take you to the text editor with the post’s details filled in. “Draft” should grey out the post and draft it. “Delete” should remove the post.
Can publish portfolio and share it with a link	Click on the “publish” button. Then share portfolio using URL link.	Clicking on the “publish” button, it should open a new tab with the published portfolio. Anyone with a link should be able to view the students' published ePortfolio on any standard browser.

Acceptance Testing

After reaching a certain point in the development of our website, we decided it was necessary to start getting feedback from potential users. There were two main stages of our acceptance testing. We conducted an online survey just after our Beta release. After deployment, we sent a link to the website to a set of selected users. These were students from different courses, countries and degrees we know. After they tested the tool, we interviewed them with a prepared set of questions.

Client Feedback

Throughout the whole development process, we continuously received feedback from our client, as it was important to us for him to be involved throughout the development stages. On several occasions, it proved to be very beneficial, since our client had great ideas we agreed to implement. For example, our client was the one who suggested doing a survey when the team was disagreeing on the design. Furthermore, he suggested the idea to have the posts filtered on separate pages and to have an overview page.

Survey

The site was not usable yet and the team was disagreeing on certain aspects of the design of the "About Me" section and the navigation bar categories. We couldn't have users test the version yet and we had additional questions about the design; therefore, an online survey was the best way to come to a conclusion.

The online survey we sent out was receiving responses from the 10th of March 2021 until the 18th of March. Altogether we had 40 respondents, out of which 25 were students and 15 employers. Most of our respondents reached at least a bachelor's degree, are in the IT industry and are open to using portfolios as a tool to hire people. After we closed the survey, we analyzed the responses to each question. The three most important questions, and the reason we conducted the survey, gave us a clear idea of how to proceed in the design. They are presented in the figures below.

From Question 1 *(Figure 6)* we were able to select which sections to use as the different categories for posts. In Questions 2 and Question 3 we asked about specific design options for the webs app. The respondents could choose between two options for both questions. From Question 2 *(Figure 7),* we concluded that the "About Me" section would remain small and to the side of the posts, as opposed to being its own separate page. Additionally, the responses to Question 3 *(Figure 8)* clearly showed that the respondents preferred having the different categories be in the navigation as tabs. The other option was to have a single page for all the posts with a filtering tool on the RHS.

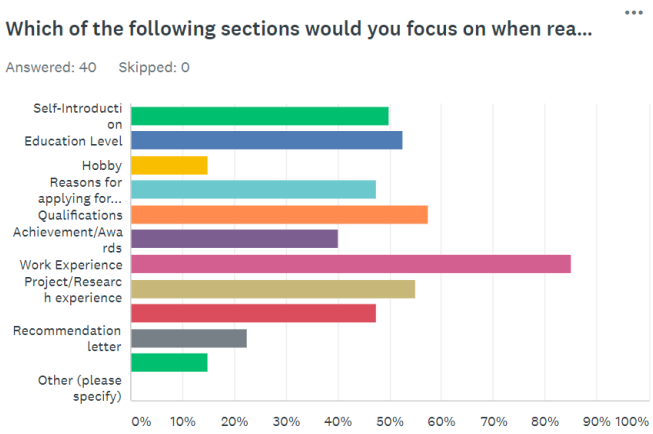

Figure 6: Which of the following sections would you focus on when reading/writing a CV?

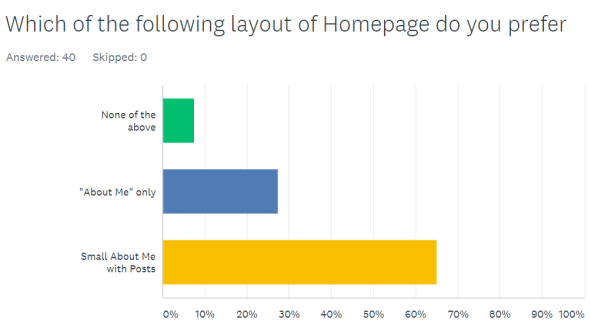

Figure 7: Homepage layout preference

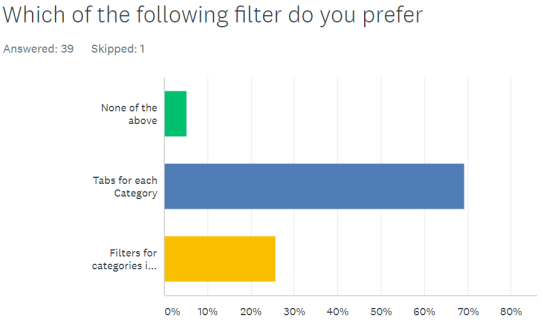

Figure 8: filters layout preference

User Testing

Even after finishing our coding tasks and accomplishing all of our initial goals, we did not feel ready for the final release. We realized how beneficial it would be to have potential users try our website and give us more feedback. Therefore, we reached out to 6 students from different courses and gave them 2 days to test our website.

After they made their portfolios, we prepared a few leading questions about their experience and conducted an interview via Teams. During the interviews, we carefully documented the user feedback. Afterwards, we analysed the findings in a meeting and made a list of all the changes the users suggested. From that list, we decided which ones we would be able to accomplish before the final release.

Unfortunately, due to time constraints, we could not pursue more complex ideas such as adding colour schemes and allowing the user to create their own custom categories. On the other hand, we successfully implemented several other changes raised in the interview. Table 4 details the suggestions and our response.

Table 4: The successful results of user testing

Suggestions	**What we changed**
Have the “About Me” on each category page.	The “About Me” renders on each page in the published portfolio.
Make the site more cohesive. The login page and the landing page look like different websites.	Unified the colour scheme, changed the background of the login/register page, changed the shape of the profile picture and removed rounded edges.
Very blank if there are no posts.	Added text that shows up when there are no posts on a page.
When pictures are attached to a post, it isn’t obvious they are pictures until you click on the link.	Previews of the pictures now show up when the post is opened in a grid format.

Reflection

The finished product has been handed over to our client, Dr Paul Harper, and he is very pleased with it as we implemented all of the essential features as well as some additional ones.

Problems Encountered

In the beginning, we had trouble developing a weekly routine. We were following the agile methodology and we had three sprints that aligned with the release deadlines set for the unit. However, we found that this wasn't enough and we struggled as a group to focus on one common goal. Eventually, we decided to assign our own soft deadlines for certain features. For example, we had a soft deadline for the category feature that allowed users to categorize their posts and ensured they appeared in the right place. Doing so helped us, as a group, focus on one core feature at a time.

We also utilised the kanban board to assign tasks each week for each member. Then in our Friday sessions, we would raise any issues we encountered and confirm the changes worked properly. We found this to be an efficient way of organising and distributing work. It ensured everyone was given an equal amount of work and it allowed us to identify any tasks that may have taken longer to accomplish. There was also a list of unassigned tasks that anyone could pick up if they had finished their own. Most of these included minor issues like css or bug fixes. Doing this helped reduce the number of small issues that needed to be fixed towards the end of the project.

At the end of the week, we would each push to our respective branch and attempt to merge it with main. However, we faced some issues with this. Most members in our group were not aware of how to properly use github and git. In the beginning, we designated one person to copy and paste the other members' code and merge it with main. However, this removed all of the commit histories of the other members. We eventually figured out how to push and merge correctly just after our Beta submission by talking to our mentor. If we were to do this project again, we would ensure everyone understood how to use github and git properly.

Ethical Issues

An ethical issue we face right now is potentially inappropriate/harmful content being uploaded for users. Currently, there are no regulations in place to control what can and cannot be posted to a portfolio. This could result in inappropriate content being uploaded and shared with no moderation. In future development, this issue can be tackled by adding a feature to report content that goes against our policy.

Another ethical issue is users not being able to withdraw their consent if they no longer wish for their personal data to be stored. Currently, users can only view the ToS and Privacy policy again in the settings page. An option could be added to allow the user to withdraw their consent anytime, thereby deleting their account.

Conclusion

We believe our completed system will have a positive impact as students will be able to easily build their own portfolios, even if it's their first time doing so. They can then continue to use the web app to develop their portfolio over time whilst easily sharing it. The web app also caters to most students' needs as they can upload any type of file. However, it can also be

developed further to give users even more customization tools to personalize their portfolios. This is something our client, Dr Harper, is looking into developing in the future.