SPE ePortfolio
Project Documentation Portfolio Part A

Joel Grimmer
Pragya Gurung
Linda Lomencikova
Yuxiang Hu
Disen Hu

Overview

Client

Our client, Dr Paul Harper, is a lecturer in the department of Civil Engineering at the University of Bristol. As a lecturer, he's very passionate about encouraging students to document their growth and learning over time while in the process, evaluate their academic work on a critical level. Paul would like the ePortfolios to be easily accessible to potential employers and tutors to facilitate assessing work.

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. This new tool would enable students to catalogue and reflect on their academic and extra-curricular work overtime. Whilst also allowing them to critically evaluate their achievements and gain deeper understanding from project work by connecting it to other previous work. In addition, the portfolio will have a feature that allows it to be shared among peers and potentially prospective employers via a link that can be opened on any standard browser.

Mobile app or Web app?

Our client stressed the importance of making the ePortfolio tool very simple to use and easily accessible. Paul also mentioned that he'd like the web app to potentially be a part of the university's blackboard front page, so that all students can access it just like they would their units. Therefore, 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 design 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 upload various file formats about any project, work or industrial experience they may have. Each post can be put into categories such as "Projects" and "Internships" which the user can customise. For example, a photography student may add a category called "Gallery" to display all their photos. There will also be an option to save posts as drafts and these will not be displayed on their published portfolios. Aesthetic wise, it will be minimalistic so there will only be a couple of themes the user can choose, as the main focus of the portfolio will be the contents.

First, we will be using Spring-Boot, React.js with RESTful API to develop a fully functioning ePortfolio web app. Then we will add features to make the interface accessible and streamlined.

Requirements

Stakeholders

The stakeholders are all the individuals or groups directly affected or associated by our web app.

1. ***Web Application Users*** - *Students who will use our web application to create their own portfolio.*

    - As a student, I want to make a portfolio to record all of my achievements and projects so that I can showcase my abilities.
    - I want to be able to upload my projects/essays and categorise them so that it's easy to access.
    - I want to be able to save drafts of my unfinished posts, so that I can continue to improve my ePortfolio before I publish it.
    - I need to be able to share a link to my ePortfolio, so that I can show it to people easily.
    - I want an accessible platform that I can sign into and use with ease on whatever device is available.

2. ***Potential Employers*** - *Possible interviewers that want to read the ePortfolio of a student they are interested in.*

    - As an employer, I want to see the interviewee's ePortfolio so that I can better understand their work.
    - I want the ePortfolio to be easy to navigate so that I'm able to clearly see exactly what the student has achieved/worked on.
    - I want an accessible way to interact with a prospective employee's information without having to create an account.

3. ***The Development Team (Us)*** - *The group of students who will develop this the tool.*

    - As a member of the development team, I want to make an easy to use tool that allows students to document their learning over time.
    - I also want to gain experience working on a group project whilst using the agile methodology as this it's common practice in the industry.
    - I want to gain skills and knowledge on web app development, including how to design and implement the front and back ends.
    - I would like to include this web app in my CV; therefore, I want the open source code for this project to be a positive representation of my skills.

Primary User Requirements

"As a student, I want to make a portfolio to record all of my achievements and projects so that I can showcase my abilities."

Basic flow

1. The student opens the web app on their laptop and accepts the cookies.
2. The student creates an account by registering their email address and a password.
3. The student writes in all the information they want to share in their portfolio.
4. After the filling in the details, the student will publish their portfolio.
5. Once published, the student can share their link.

Alternative Flow

1. The student enters the web app on their phone.
2. The app asks for consent to use cookies.
3. The student accepts the cookie consent.
4. The student can continue to make their portfolio.

Exceptional Flow

1. The student enters the web app.
2. The app asks for cookie consent.
3. The student rejects the request.
4. The student won't be able to use the ePortfolio tool.

Other Core User Requirements

"*As an employer I want to view a candidates ePortfolio to see their achievements/past works.***"**

Basic flow

1. Employers receive the link to the students ePortfolio via an email.
2. Employers open the link and view the ePortfolio.

Alternative Flow

1. Employer sees the link somewhere else. For example, on the student's LinkedIn profile.
2. Employer opens the link and views the ePortfolio.

Exceptional Flow

1. The employer opens the link to a student's ePortfolio.
2. The students ePortfolio does not load correctly, just a simple message stating what happened.
3. The employer can close the window without any problems.

Primary User Story

Web application User: *"As a student, I want to make a portfolio to record all of my achievements and projects so that I can showcase my abilities."*

Step 1 - Create an account

The student opens the web app on any standard browser and they're prompted to consent to cookies. If they accept, they can continue. If they reject, they will not be able to proceed further into the web app. Once the student accepts, they can click on the "register" button. Then the registration page appears for them to enter their username, email address, password and confirm it. The student will have to confirm their email and then the account registration is complete.

Step 2 - Write the ePortfolio

When the student logs in for the first time, they will be given a tutorial/guide about how to use the tool. There is an option to skip it. After viewing the guide, a new ePortfolio template with the basic theme and layout will appear. Next, the student can fill in their ePortfolio by clicking on the box that says "Add text here". For example, if they are a Computer Science student, they could upload a post about their most recent coding project and tag it with the category "Projects". The post could include screenshots of the working product as well as a link to a GitHub repository. If they want to save the post but not publish it, they can choose to draft it. The student could also customize their ePortfolio's appearance; for example, they may choose a different theme or add more categories to make it more personalised. On the top left corner, there is an option for the student to preview their ePortfolio and if they are happy, they can publish it.

Step 3 - Send Portfolio to other people

Once published, the student can copy the link to their portfolio and then send it to anyone.

Primary User Story Diagram

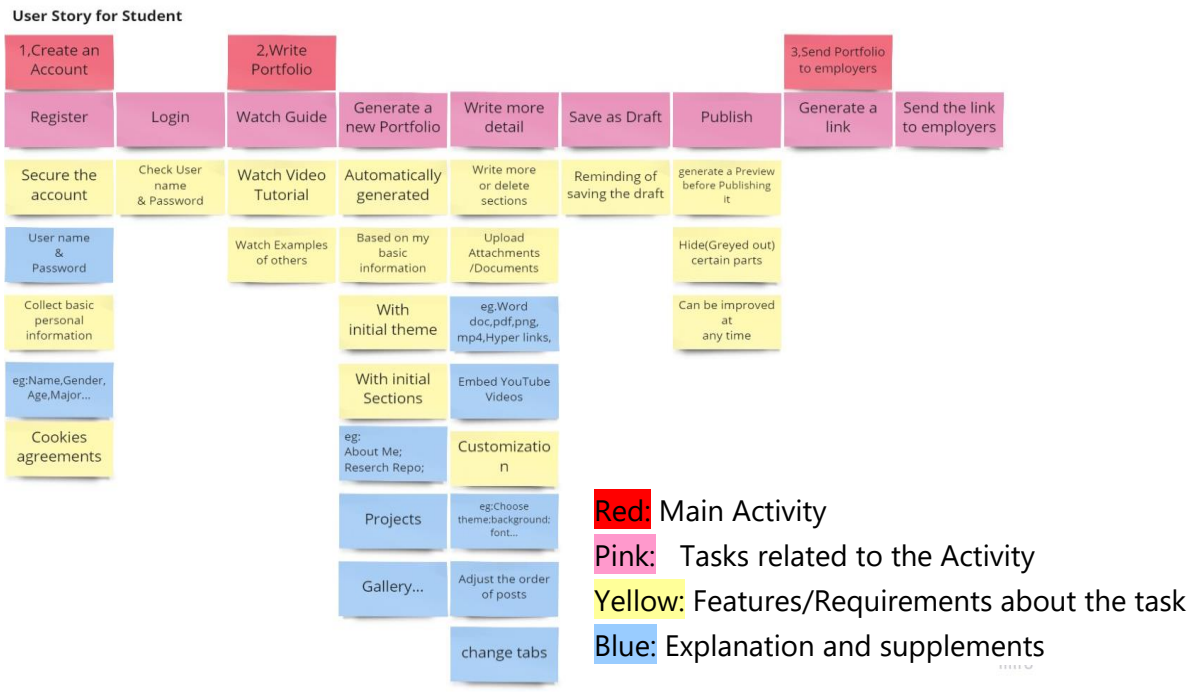
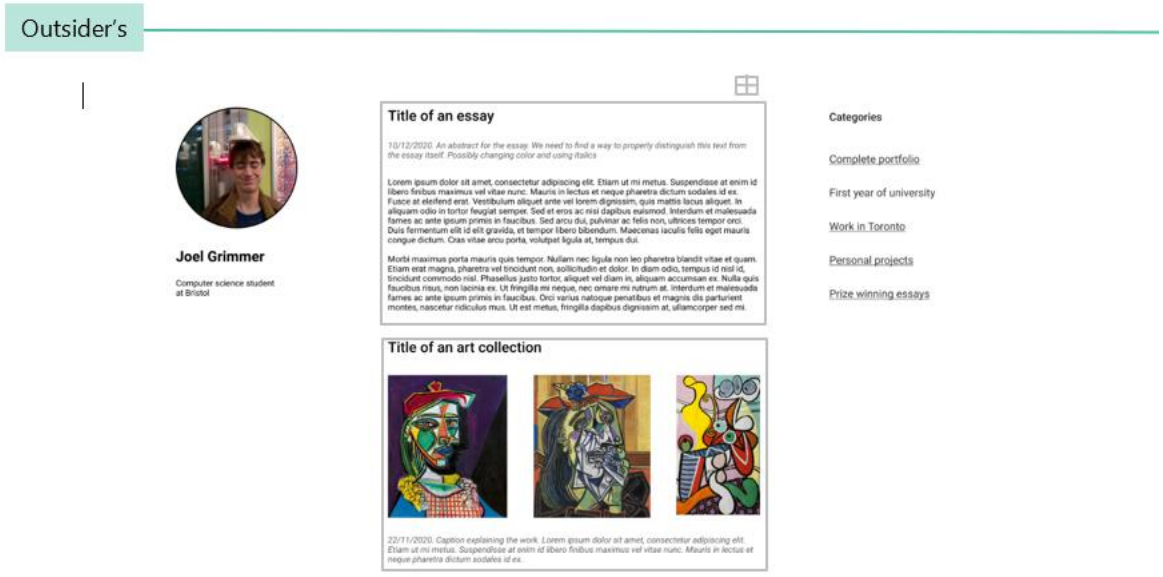
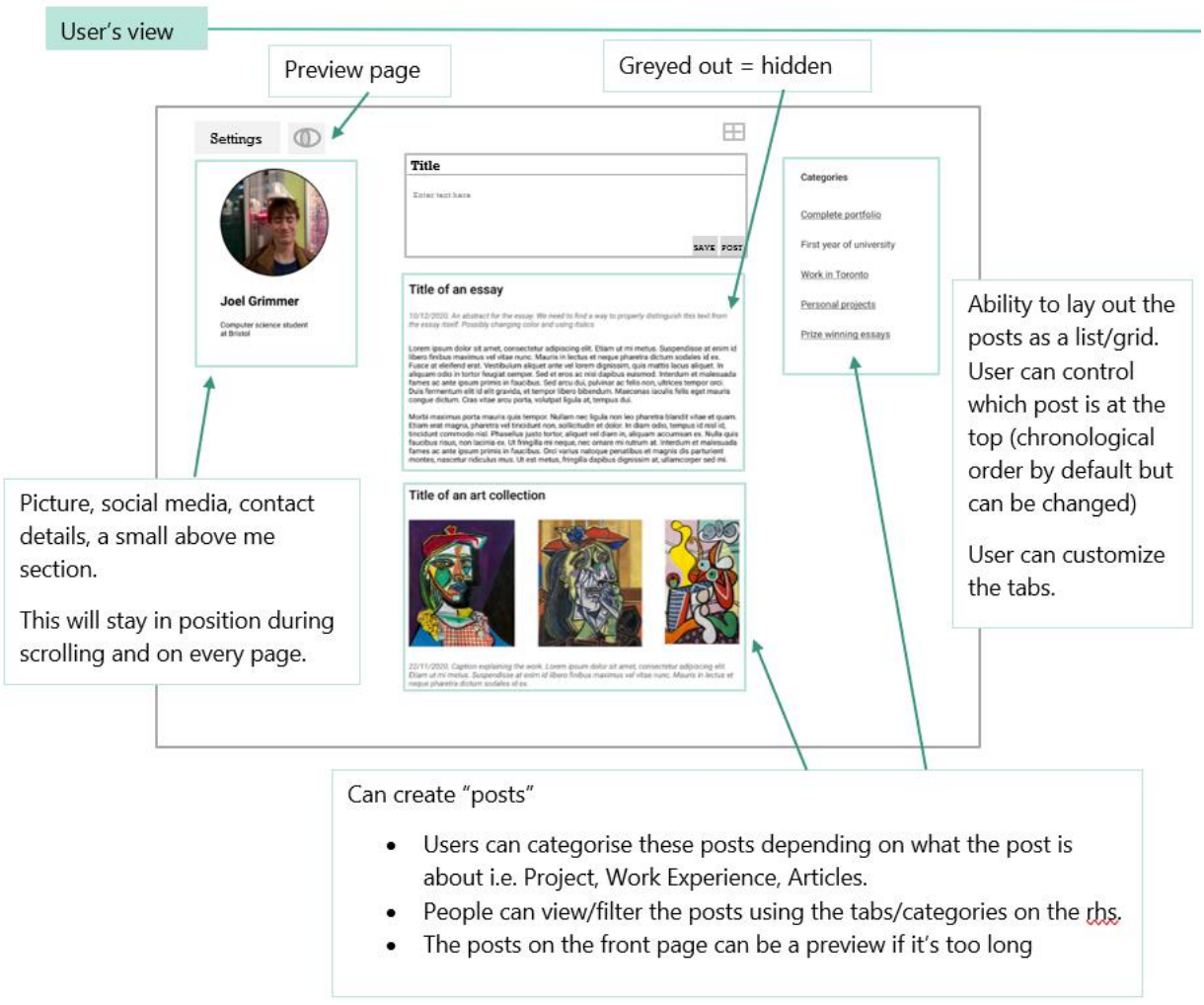


Diagram of the user's layout





Personal Data, Privacy, Security and Ethics Management

To access the ePortfolio tool, the user must create an account by providing their email address and creating a password. The account holder will be the only one who can access the portfolio associated with that account. The only data collected and stored is the data given by the users themselves upon making their account. We will store this information in a database.

To process and collect user data lawfully, we will adhere to the General Data Protection Regulation (GDPR). The laws require us to get consent from our user to collect and store the person's data. We must also explain how the data will be used. Our registration form will have separate consent confirmation from Terms and Conditions. It will be opt-in, meaning the form will be set to "no" or blank as default when registering. They will have to agree to the T&S to be able to register. Users will also be asked for permissions to use cookies. If they accept, they can continue to use the web app. If they reject, they will not be able to use it.

However, the user can choose to include other personal data in their portfolio. They can choose to share their name, address, telephone number or date of birth and anything else. We will only store this information in our database with all the other information the user includes in their portfolio and we will not use the data in any other way.

The account holder is the only person who can access the portfolio associated with the account and modify what is written in the portfolio. They choose what will be visible to anyone with a link and what will be stored in their drafts. They will have the ability to send a link to any individual they choose. Anyone with the link to their portfolio can view it. We will include all of these specifications in the consent and Terms and Conditions form, so the user is aware of all the facts.

Our database will be protected by a password and it will be only accessible to the Development Team. We will identify sensitive and critical data and encrypt them; for example, passwords of all the accounts will be hashed/encoded. We will audit and monitor the database to keep track of all the changes and movement of information in it.

Ethics pre-approval was applied for on the 22nd of November 2020 at 17:18 pm.

Architecture

The ePortfolio will be delivered as a web application on request of the client, as opposed to a native mobile application. We will be using a number of libraries and frameworks to achieve a fully working web application.

We will host the application in a serverless manner, with the front-end, business logic, and data stores made available through AWS Elastic Beanstalk. This will manage load-balancing, scaling, and application-health monitoring amongst many other useful dev-ops services.

We use a combination of GitHub Actions and CircleCI to check Maven build and test status, and maintain test coverage.

We will be using Spring Boot to create a RESTful API where HTTP requests will be sent to our model, authenticated, and then structured data (in our case JSON for ease of parsing with our front-end) is returned. 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 that take place in our database in a secure manner. We are using @RESTController for our response controllers to make sure they are fully REST compliant.

We will use Spring Security to manage access levels to our API and to enforce data protection standards.

We achieve Hypermedia as the Engine of Application State (HATEOAS) through Spring's HATEOAS library, which allows us to insert links between different requests to the rest API. This allows us to access Level-Three Richardson Maturity.

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 saves us from ever having to write raw SQL requests to the server. This improves security as it prevents us from spending a considerable time preventing SQL injection in all our form fields, and improves the reliability of the requests that we do send to our database by preventing them from being entirely a string.

We will be using React.js to drive the front-end of our web application. We will use client-side rendering to prevent too much work 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 will use to access our REST API, and the application will update the page on receipt of the relevant data when loaded.

We will be using React Router to maintain a clean SPA to prevent all server-side rendering, in a similar manner to how Facebook and Twitter are managing web page state. We will be using JS query strings to allow access to custom rendered components, such as an individual user's portfolio.

Development Testing

For development testing, we will use a software called Selenium to test the functionality of our web app. As Selenium scripts are compatible with multiple browsers, it allows us to easily automate testing across various browsers like Chrome, Mozilla and Safari. With Selenium, we could carry out UI tests in an automated way where user actions/inputs can be simulated, such as uploading files, clicking tabs and signing up. Specifically, for the sign-up forms, we can test it to ensure all mandatory fields are filled in, and once submitted, the data in the forms is correctly sent to a database. We can also test that all links (outgoing and internal) in our pages are working correctly to ensure nothing is broken. Unit tests can also be written with Junit, a Java unit testing framework, which is pre-installed in IntelliJ by default. CircleCI and GitHub Actions are both Continuous Integration platforms that we will be using to automate, build, and test our code.

To ensure that front-end development is consistent with back-end development and that correct information is displayed to the users, we will use a software called Postman to test HTTP requests. It would test if the JSON body returned included the correct content. Postman allows us to create and send any HTTP request that can be saved in history and executed again. Once we verify that all the HTTP requests are fine and the content received is correct, it will be ready to use in our web app. For testing our front-end (REACT), we will use a unit testing framework called Jest. It comes with its own test runner, so we can simply call Jest from our terminal to run all the tests. Jest also has a tool called Snapshot that essentially creates a 'snapshot' of your system and stores it in a separate file. Once you make a change and run the tests again, Jest will compare your new snapshot with the old one. If they are different, the Snapshot test fails, shows you what's different and gives you a chance to either accept or deny the changes. This will keep our tests lightweight and prevent regression.

Test	Expected Results
Test login page renders.	When the button register is clicked, the login page should appear correctly along with the appropriate input fields and submit button. There should be a terms and conditions check box just above the submit button
A new category can be added to the tabs section.	Once logged in, the default tabs will be About Me, Project and Work Experience. The user can customise them by adding and rearranging the order of the tabs.
The user can upload posts to the correct category.	A text box should appear in the middle of the screen for the user to write anything. There will be an option to upload files, and to save the post as a draft. There will be a drop-down box that allows the user to tag the post appropriately.
A guide/tutorial button	This button will appear on the top left of the page and when clicked, a tutorial explaining how to use the tool will appear.
User can edit and delete posts easily	Next to each post there will be a bin and pencil icon. The bin icon will delete the post whilst the pencil will allow the user to edit the post. The user should be able to freely edit the post and that includes being able to change the category it belongs to, adding more files and drafting the post.

Release Testing

Selenium will also be used to test scenarios. For example, a scenario could be a new user wanting to make a post about their project. The test code should accomplish the following: Create an account with username X and password Y, verify that an error will occur if registration isn't filled properly (e.g. the password must be 8 characters long and includes a number), make a post about a project Z and finally verify that the post is visible on the accounts portfolio.

The user story we will be manually testing:

"As a student, I want to make a portfolio so that I can document my growth and learning over time"

We choose this particular story because the main demographic for our ePortfolio tool is students. They are the one that will interact the most with the web app, therefore, at some point they will encounter all the features. It also includes the ease of use and accessibility requirements that our client emphasised.

Test	Expected Result
A student has opened web app on chrome. They are will be a welcome page with the option to register or login.	Web app should be successfully loaded. The ePortfolio logo should appear along with a short description of what the tool is about. There will be a simple slideshow presenting the different features in the tool and also examples of finished portfolios. Underneath should be 2 buttons, login and register.
Sign up as a new user.	Once they click on the register button, they should be directed to a registration page. They will need to fill in the field's username, email and password. There will also be a terms and condition check box above the submit button that they will need to agree to before submitting. Once the form has been submitted, the user will have to confirm their email address.
The student will upload a recent project they've completed to the project category.	The student should be able to write a post by clicking on the text box in the middle of the page. Along with a description, the user can attach any relevant files. Once submitted, the post should appear on front page of their portfolio and in the correct category
If the student doesn't want to publish a post but also doesn't want to delete it, they can draft it.	Next to the "Post" button, there will be a "Draft" button. If clicked, the post will not be visible on the students published portfolio. On the user's side, it will appear greyed out.
The student can preview their portfolio and share the published version using link	To the top left there will be a preview button that will allow the students to view their portfolio as an "outsider". Then there will also be a publish button to the top right. This will open a new tab showing the users published portfolio. The user will be able to copy the url and send it to whoever they want. Anyone with the link will be able to view the students e-portfolio.