



University of Brighton

Constructing a Functional Web Application Educational Tool

By Jake Ward

CI601 The Computing Project
06/05/2023

| | |
|--|----|
| Aims and Objectives..... | 4 |
| Project Management | 4 |
| Requirements..... | 6 |
| Business Requirements..... | 6 |
| User Requirements | 7 |
| System Requirements | 7 |
| Deliverables..... | 9 |
| Approach | 9 |
| Methodologies | 10 |
| Waterfall | 10 |
| Agile | 12 |
| Agile v Waterfall..... | 13 |
| Tools and Techniques..... | 14 |
| Planning | 16 |
| Schedule..... | 16 |
| Risk Analysis | 16 |
| Research..... | 18 |
| Management of Legal & Ethical Issues | 18 |
| Intellectual Properties..... | 18 |
| Copyrighted Content..... | 19 |
| Patents | 19 |
| Research & GDPR | 19 |
| Design & GDPR..... | 21 |
| Subject-matter | 22 |
| Participants | 22 |
| Informed Consent | 22 |
| Structure & Resources | 22 |
| Architecture | 23 |
| Peer-to-Peer..... | 23 |
| Client-Server..... | 24 |
| Peer-to-Peer vs Client-Server..... | 26 |
| Server-Side Rendering vs Client-Side Rendering | 27 |
| Environments | 28 |
| Node.js | 28 |
| Brighton Domains | 28 |
| Frameworks | 30 |
| React | 31 |

| | |
|--------------------------------------|----|
| Angular | 32 |
| React vs Angular | 32 |
| Express.js | 33 |
| Django | 34 |
| Express vs Django | 34 |
| Toolkits | 34 |
| Gulp | 34 |
| Webpack | 35 |
| Gulp vs Webpack | 35 |
| Database Management | 36 |
| MySQL/phpMyAdmin | 36 |
| Middleware & Packages | 37 |
| cors | 37 |
| mysql2 | 37 |
| body-parser | 37 |
| express-fileupload | 37 |
| ejs | 37 |
| Testing | 38 |
| Postman | 38 |
| Browser Dev Tools | 38 |
| W3C Markup Validation Service | 39 |
| Primary Research | 40 |
| Secondary Research | 47 |
| Design | 61 |
| Implementation | 62 |
| Set-up | 62 |
| Testing | 65 |
| Research Material & References | 65 |

Aims and Objectives

My project is the development of a functional web application that functions as an educational tool. Specifically, this software artefact will serve as a web application to potentially be used by a company for training employees through hosted content such as training videos and articles. The system will allow the given management to assign courses to employees as well as track the progress of employees on said courses. This project serves as a template, not for commercial use but to emulate a web application of this kind.

I knew that my objectives were achievable because I designed the objectives based on previous successful experiences at the University of Brighton. I also knew these objectives would be achievable as I studied the potential various resources at length over the course of the year, to not only prove the objectives could be met, but also to find the most efficient method to meet these objectives. I knew these objectives would be achievable as the key objectives wouldn't require great lengths of development time. I knew this due to my experience in the area of web development. This meant I knew I could spend the remaining development time increasing the complexity of the product without the danger of not meeting the deadline. So, to summarise, I designed the objectives to be achievable. I knew they were achievable because of experience in the area, experience with chosen resources, research into resources capabilities and simply key requirements ensuring a wealth of development time.

Project Management

As I am the sole developer, I am also the project manager. In this way, I was responsible for deciding the requirements, deliverables, approach and resources to be used. I took immense time researching each aspect of the project to decide what would be the most effective and efficient methods to meet my set objectives. The justifications for each of my design and management choices can be seen at length in the research section of the report.

The project started via refinement through notation format. By mind mapping ideas/potential requirements, the various theoretical goals of the project would become solidified requirements allowing me to organise and carry out each task efficiently.

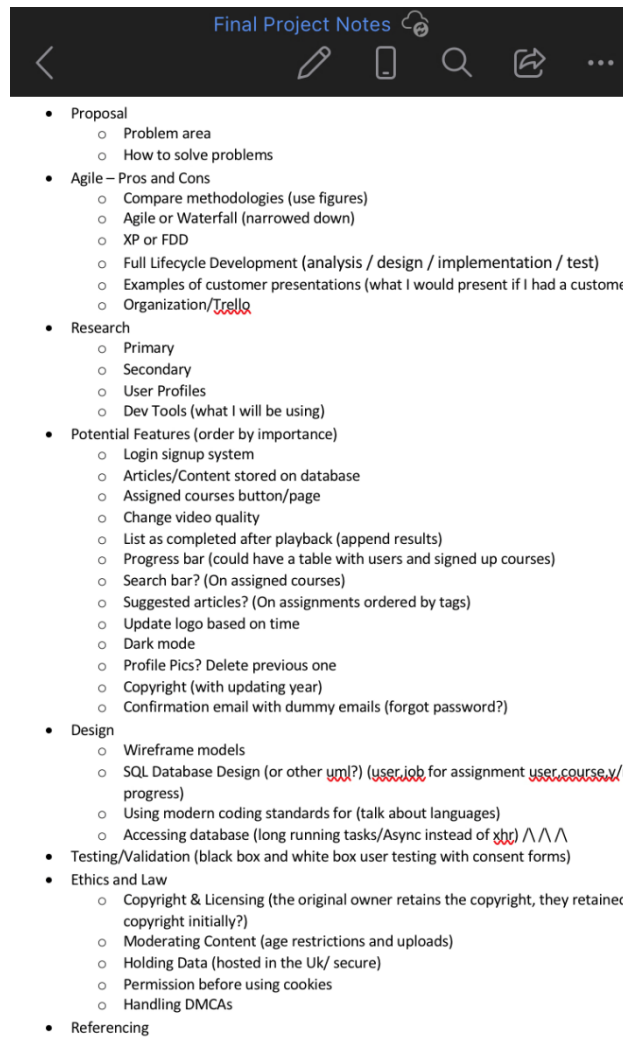


Figure 1: Project Outline Notes

This project outline (once refined enough to meet the correct standard) would then be transferred to a more professional web space. In a professional environment, it is common in this scenario to take advantage of a project management tool. I chose to use a Trello workspace to host the project requirements and potential features to take advantage of their user-friendly project management tools, as well as to help clearly document the process of the project's development.

Trello is a free-to-use project management tool developed by Trello Enterprise. It can be used to manage personal and organisational projects via an online Kanban board. This is used to visually depict work at various stages of development. Over the course of development, the requirements will have moved from the first to the final stage while the potential features could progress or be dropped from the table. It also supports integration with other useful services like Dropbox and Google Drive as well as being available on mobile and pc. These facts make Trello a perfect tool to use for documentation.

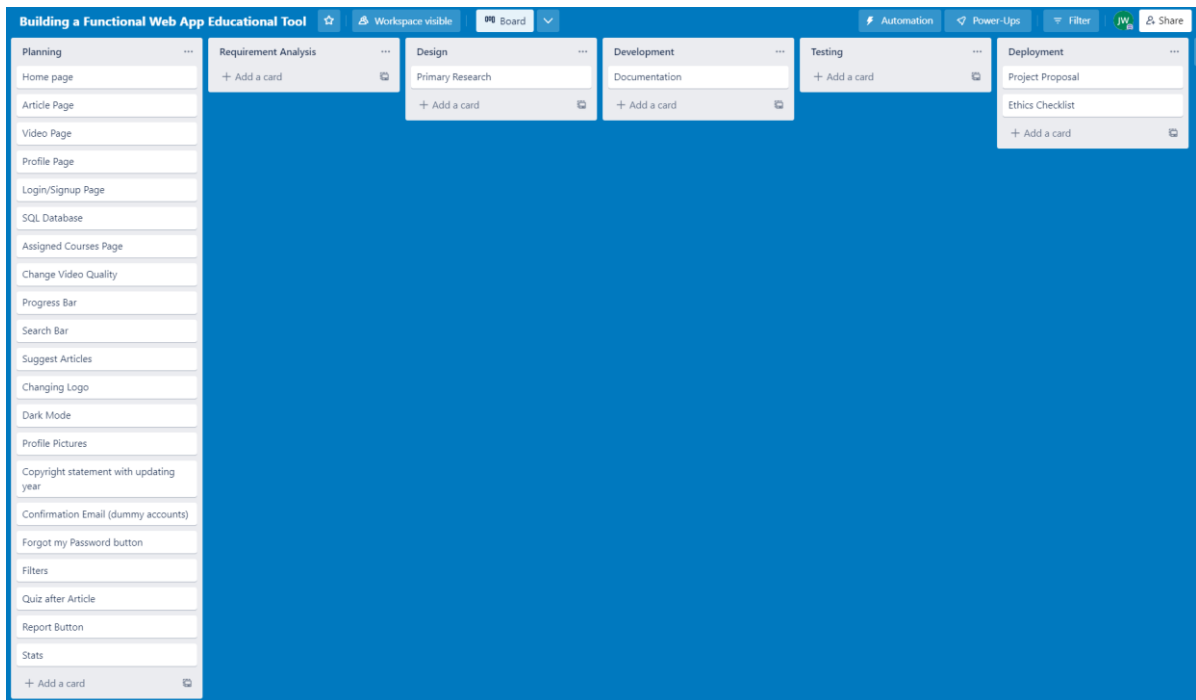


Figure 2: Trello Board based on Agile SDLC, 12/10/22

I designed my Trello board according to the Agile model of the SDLC (software development lifecycle). Due to this, the stages of development are as follows; Planning, Requirement Analysis, Design, Development, Testing and Deployment. Latest ideas and adjustments could be added to the table instantly via Trello's mobile support and peers could be invited to view the table to get an understanding of development or give critiques if peer review were allowed.

I decided to document the development of the project in a word document saved on Microsoft's OneDrive as well as regularly saving copies on my mobile device and personal computer. These various copies would serve as backups for each other to not lose the contents, it was the most effective way as it made use of both cloud and physical storage as well as automatic saving.

The structure of the documentation is to be separated into the stages of my chosen SDLC alongside other requirements such as Methodologies and References, while this may not display feature additions in chronological order it displays the information in an easier-to-understand format.

Requirements

In this section, I have listed the different types of key functional and non-functional requirements.

Business Requirements

Business requirements are high-level requirements describing what the application will do from a business standpoint without considering the objectives on a technical level. These requirements are ordered by priority and separated into key features and non-key features.

Must:

1. Avoid legal and ethical issues (e.g. keep data secure)
2. Validate employees before allowing them access
3. Allow employees to access training resources
4. Allow an administrator to manage training resources

5. Consider accessibility

Should:

6. Be usable from mobile devices
7. Test employees after training
8. Track employees' progress over courses

Could:

9. Add quality-of-life updates

User Requirements

User characteristics were defined in the section of the report labelled “primary research”. User requirements refer to the users' needs and represent the actions the user will be expected to take once the application is complete. These are ordered by priority and separated into key features and non-key features.

Must:

1. Allow the user to signup/login
2. Allow the user to visit training resources
3. Allow the use of a screen reader

Should:

4. Allow users to use the site on a mobile device
5. Allow users to see all their assigned courses
6. Allow users to see their progress in training
7. Allow users to take a test to complete training

Could:

8. Be able to switch between a dark and light theme
9. Send confirmation emails to users
10. Allow users to upload a profile picture
11. Allow users to search and filter through their training
12. Allow users to report a user or video
13. Allow users to view stats
14. Allow users to change the quality of training videos
15. Be notified when a due date approaches

System Requirements

System requirements are representative of the literal tasks that I as the developer will have to carry out to build the web application. Below I have listed the system requirements at the start of my project by order of importance. This was designed so that the bare minimum requirements being met would result in a successful software artefact. Any requirements listed as “must-have” are key functional requirements and are required to meet the project's objectives. Any requirements listed as “should have” or “could have” are non-key requirements and serve only to improve quality and increase complexity.

Must-Have:

1. Home Page
2. Login Page
3. Database
4. Login/Signup functionality
5. Article page outline
6. Content stored on the database
7. API for communication with the database
8. Dynamic article pages with content from the database
9. HTML validation

Should Have:

10. Responsive pages (WIA standard)
11. Ability to assign courses to users
12. Assigned courses page
13. Progress bar
14. Quiz after articles to complete course (results stored on the database)

Could Have:

15. Update logo, light/dark mode and copyright based on time (additional small visual elements)
16. Confirmation emails (using dummy accounts)
17. Profile pictures
18. Search bar (on suggested articles page)
19. Suggested articles
20. Report button
21. Stats (courses complete etc)
22. Change video quality
23. List a video-based course as complete after playback
24. Notify users when a due date is approaching
25. Saving states of a test/page when applicable

I must use HTML validation to ensure screen readers are compatible with the site to maintain accessibility. The web application must feature responsive pages also to ensure accessibility in case a user can only access the site via a mobile device. The pages are to be separated into a home page, login/signup page, dynamic article page and assigned courses page to maintain user-friendliness. It does this by assigning different functionality to each page to be conventional and understood by most users.

There is a login/signup system in place to ensure no unauthorised access to the application or any of the hosted content. All content is stored securely on the domain or in a database to ensure the safety of sensitive information.

An API must be created to send data between the database and the client while using restrictions to allow functionality but without potential unauthorised access.

There would be a progress bar/counter to help users see how much training they have left. As well as an assigned courses page with a search/filtering function to aid should there be an overwhelming

amount of training to sift through. This ensures a user-friendly design which can be improved via other quality-of-life updates like the use of profile pictures, dark mode and statistics.

A report button could be added should a piece of uploaded content violate any terms of service and copyright statements could automatically update to prevent copyright issues.

Allowing changing the quality of a video could be a worthwhile addition to accommodate users with low-speed internet connections, making the application more accessible.

Tests and Logging when there's a video playback ensure that the user has successfully completed their training. This is important for the user experience as it enables a user to keep track of which training they have yet to complete. It would also be important for the company as it enables them to check if users are completing their assigned training.

New content should be able to be added to simulate how a real company would update their content. Content should be able to be assigned to a user by the administrator so the user can easily navigate to their relevant training via the appropriate page, this also makes the site user-friendly by conforming to convention.

Overall, these requirements should ensure an inclusive, user-friendly, fast and efficient experience for the user, devoid of legal or ethical issues. They should be able to log in; be assigned training and prove they have completed it.

Deliverables

The chosen deliverables were impacted by research detailed later in the Research section of the report. The deliverables are the items that are evidence of the work detailed in the report as well as the final product.

The deliverables for the CI601 project include:

- The source code for the Client-side of the Web Application
- The source code for the Server-side REST API of the Web Application
- Code displaying the structure of the Database/Tables
- A link to the live Web Application
- The API support documentation
- A Report on the entire project (including project designs, implementation documentation and testing plans and results)

Approach

I am making a web application because I am confident in my capabilities in the area of web development. It is because of my experience in the area at the University of Brighton that I am confident that I can create a high-quality web application. The area of web development is also greatly ranging, allowing me to learn many new techniques and resources over the course of the

project. Because the area of Web Development is vast it allows me to showcase competence in various different programming languages and areas.

The project is a web application from a logical standpoint because web applications are amongst the most accessible types of software. As any device with the potential to access the internet can reach a web application, they are one of the most relevant areas to explore in the modern day.

As well as this, the setting in which employees would be required to complete online training is using a desktop. This is often an automated process hosted on a web application and so it stands to reason that if I were to create a software artefact designed to train employees, then the most effective and conventional solution would be to design a web application to host the training content.

A mobile application could have been designed for training employees but this would not be as accessible or relevant in a professional environment as desktops are the conventional professional computer.

Alternatively, a different type of web application could have been selected. However, I chose to create a web application designed to train employees because it allowed me to explore and showcase my knowledge in many varying areas. This choice would allow me to create a functional client, create a REST API, store and access information in a database and explore new development tools like modern frameworks and middleware. Overall, I chose this approach because it would allow me to showcase my skill as a developer in multiple areas and would be useful to potential clients in a real-world scenario.

I have detailed the methodologies, tools and techniques I decided to use in my approach to reach my set objectives below.

Methodologies

I researched the various methodologies for approaches to project development. There are ranging types of methodologies for project development and some methods may not fit into one simple category. I researched the following SDLC models

- Spiral
- Prototype
- Iterative
- RAD
- JAD
- V
- Big Bang
- Agile
- Waterfall
- DevOps

After researching these different methods on a surface level, I decided to explore a few of the potentially applicable models on a deeper level to find the model most appropriate for my project's development. I concluded that I would be progressing with either the Waterfall or Agile method of development.

Waterfall

The waterfall model, also known as the linear-sequential model, is an SDLC model developed in 1970 by Winston Royce. Its name is derived from its chronological and linear structure. Almost all old

software was based upon this model, yet it has fallen from its longstanding popularity due to the rise of the Agile development method. Development is segregated into multiple phases, each focusing on distinct goals. Each phase must be completed entirely and successfully to move on to the next phase and there is often no opportunity to modify the project's previous phase.

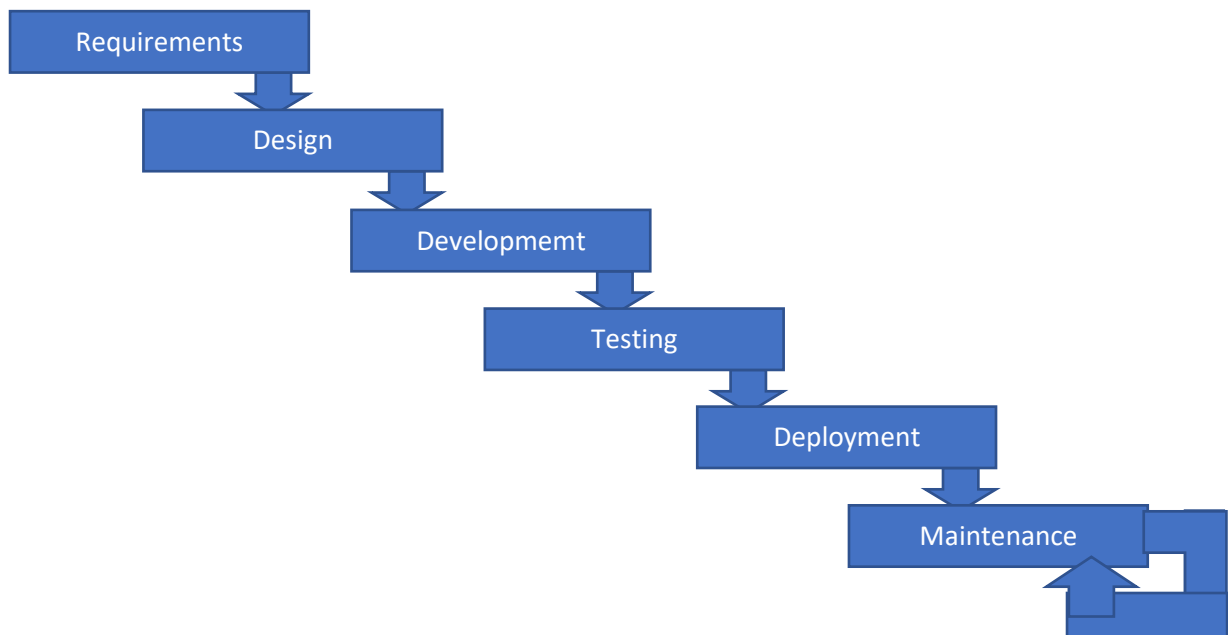


Figure 3: Diagram of a Waterfall model

Projects with stable objectives best use the waterfall method as it makes the development easy to understand and track progress. However, this would mostly benefit teams whose composition changes frequently. This applies a lot less to my individual project's criteria as my team would consist of myself and myself alone. The method can be slow and obstructive due to its aversion to modifying previous work making the process significantly less flexible. It is because of the methods' lack of flexibility in comparison to other methods that lost its status quo as the conventional SDLC model.

Advantages of using waterfall include:

- Easy to manage with its clear requirements in each phase of development
- Documentation is easy to manage
- Projects are delivered fast so long as they have stable criteria, and no unforeseen errors occur
- Simple dependency management
- Beneficial for teams with fluctuating composition

Disadvantages of using waterfall include:

- Not ideal for large-scale projects
- Lacks flexibility for problem-solving or changing the project's direction
- Unclear requirements become a detriment later
- Testing is only carried out toward the end of a project lifecycle, making bugs likely, potentially endangering the development of the project

It was because of these disadvantages I felt compelled to explore the Agile development method.

Agile

The agile development life-cycle model is often used by teams to significantly reduce risk when adding new functionality. It does this via its iterative nature and flexibility, the opposite nature of its counterpart the waterfall model.

The agile model spawned from the iterative waterfall model when developers found difficulty in handling changes or updates requested by clients. The new model was first incorporated in 1990 to combat the potential change in direction of a project and the agile methodology was officially created in 2001 by a 17-person team convening specifically to define the new model.

To combat changing requirements or updates the development method was designed for easy and rapid project achievement. In a team, developers would work simultaneously on the various stages of project development and potentially different iterations. However, in a one-man team, I can only utilize the flexibility of switching between the stages or iterations when appropriate. In both scenarios, the requirements are ordered by priority and then incrementally added via iterations similar to the waterfall model and its stages. However, the agile method is flexible and so older implementations and stages can be changed without the red tape. Once all iterations are complete then the project is successful. At the end of certain iterations, the project would be presented to clients for approval.

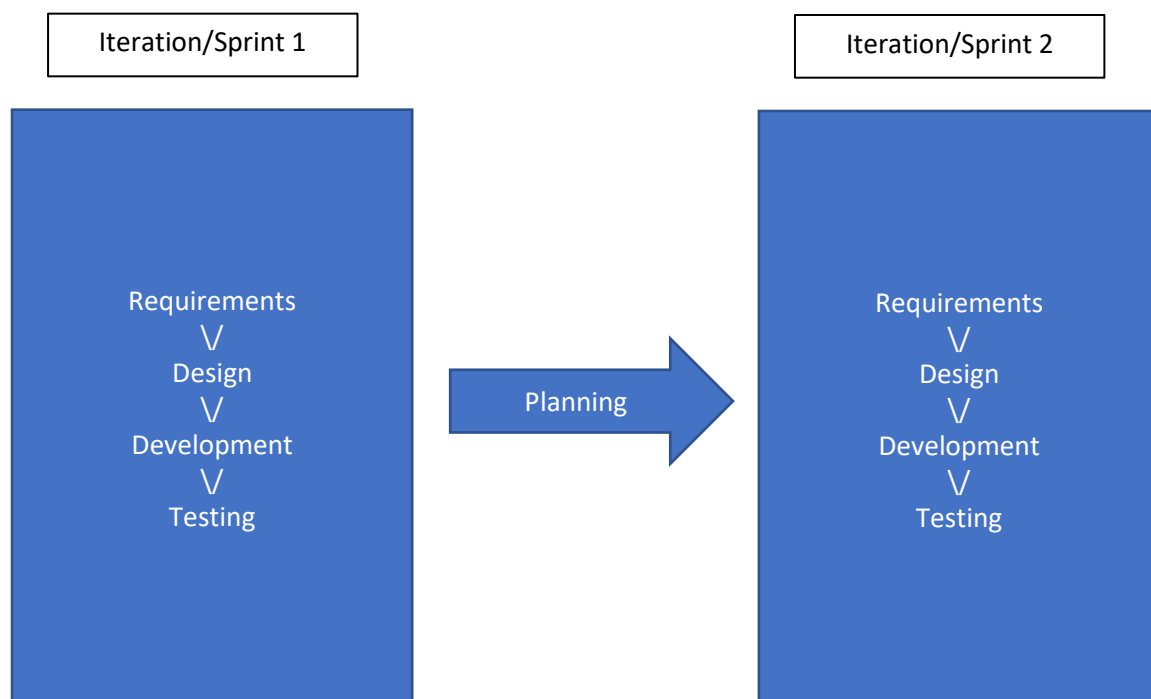


Figure 4: Diagram of an Agile model

The model relies on software deployment as opposed to comprehensive documentation and often provides clients with incremental versions at the end of each sprint/iterative cycle in intervals of 2 weeks. System requirements can be changed at any stage of development and in a team, a representative would be elected to showcase the latest version. While I have no client nor intended to get one involved in my project, I note that I could simulate it through vivid documentation. The agile method requires great efficiency between team members, something that is of no concern in an individual project.

The strategy of adding new functionality incrementally has been utilised in various other methodologies considered to be different forms of agile.

- Crystal
- Scrum
- Atern
- XP
- Lean
- Unified Process
- Feature-driven

These models are all attributed to being agile development methods with XP, Scrum and Lean being the most widely used. Utilising elements of these sub-genres of agile allows for the use of greater control and documentation. An example of this is the use of sprints and burndown charts in Scrum typically motivate and document the time and work left in a project or iteration.

Advantages of using agile include:

- Significantly reduces development time
- Clients would have real-time updates and the ability to change the project's direction at any point in the SDLC
- Iterative releases allow for bug fixes earlier in development
- Incremental additions allow for faulty plans to be changed if an unforeseen logic error arises
- Teams using agile are often more motivated and self-organised
- Development quality is maintained throughout
- Overall risk reduction

Disadvantages of using agile include:

- Agile is not particularly useful for small projects as many iterations/sprints wouldn't be required
- If you have a client an expert must be present to control meetings
- In a team, the cost of using the agile method is more than average
- If the project manager is unclear about what they want, then the project can get off track
- Heavy reliance on real-time communication
- Large commitments of time and labour are required to complete each feature within its designated iteration

I was more inclined to follow the Agile SDLC do to the lack of disadvantages applicable to an individual project without a client such as the one I would be undertaking.

Agile v Waterfall

I listed comparisons between the Agile and Waterfall SDLCs alongside my aforementioned advantages and disadvantages to make my final decision and to document the conclusions of my research.

| | |
|---------------------|----------------------|
| Waterfall | Agile |
| Sequential approach | Incremental approach |
| Rigid structure | Flexible structure |

| | |
|---|--|
| Development is divided into phases | Development is divided into iterations/sprints with phases within them |
| Development is completed as one project | Development is completed as incremental versions of the final product |
| Testing takes place at the end of development | Testing occurs frequently and simultaneously with development |
| Changes to requirements cannot be changed during implementation | Changes to requirements can be made at anytime |
| Phases like design, development and testing appear once | Phases like design, development and testing appear multiple times |
| Fixed rates in real-world scenarios | Non-fixed rates in real-world scenarios |
| Coordination is limited | Coordination is required if in a team |
| Developers are dedicated to their assigned roles | Developer's roles are interchangeable |
| Developers are interchangeable | Developers are difficult to change after a project has started |

Due to the disadvantages of the agile approach having a lack of application on my project outline I decided to move forward with the agile approach.

Any inconvenience caused by bad communication when using the agile model does not apply as I am to work on my project as an individual, and any potential teamwork is forbidden from this project. This means unforeseen errors from bad communication on the project manager's or the employees' part is null and void. As there is no client there is no one to hold meetings with at the end of a sprint and so no experts would need to be present. Instead of holding meetings, I would document my progress in this report to simulate any updates I was required to give the client. Any pricing issues are equally of no consequence as there is no budget and I am the only developer. Finally, as the project outline was designed as a potentially large project depending on how many features could be added bore the deadline, the incremental additions hold great validity.

I found the most attractive feature of the Agile model was the use of incremental development. This is because it almost guarantees that a functional product would be completed before the deadline. The high-priority requirements would outline a basic web application and then further additions would only serve to add greater complexity. This allows for safety in development, as at a bare minimum the success criteria will be met in terms of software artefact creation.

As the Agile methodology often involves the use of sprints, I've documented my progress at the end of each sprint via burn-down charts while my predicted schedule is represented by a Gantt chart.

Tools and Techniques

The web application would follow the client-server model. The client UI and REST API handling all the server-side background services would both be stored on Brighton domains. While a training site could be hosted on an intranet closed from the public I decided to host mine on the internet because it will be more accessible for grading and demonstration of this project as well as being more convenient for employees in a real-world scenario.

I considered two approaches for dedicating the tasks between the server and the client. One approach focused on server-side rendering while the other focused on client-side rendering.

In the first solution, The API would be built in Node.js using Express. It would handle server-side rendering for dynamic pages to lessen the number of pages needing to be stored on the server and reduce strain on the client. The API would handle get requests via different routes for returning dynamic pages as well as JSON responses from communication with the database. The API could also handle post requests to insert new information into the database. Various plugins listed in the research section would be used to render-pages, handle files and contact the database.

The UI elements would be written using HTML, CSS and JavaScript and potentially concatenated using Gulp or Webpack as a secondary option. There would exist a template for the dynamic pages that would be populated/rendered by the API. The UI would contact the API if a dynamic page was requested or if another request needed to be handled by the back end (e.g. posting login information or getting stats).

The second solution involved building the client-side of the application in the React framework and rendering the dynamic pages via dynamic routing. This would mean rendering the pages on the client side but would be utilising the conventional and more documented approach. In this approach, the API could still be created with express but would not hold responsibility for rendering the dynamic pages and would instead focus solely on handling get and post requests.

I decided to use my first approach as it would greatly add to my development time should I have to learn to use another new framework. As well as this, the concept of rendering pages on the server would benefit the client in terms of speed and computing power. The API would be created as a Node.js application hosted on Brighton domains. It would feature different routes for requesting different types of dynamic pages or uploading files.

All code for the Client and Server would have copies stored on a personal computer, Brighton domains and a GitHub repository. All code would be written in Visual Studio Code while attached to the repository to ensure frequent backups would be a simple task.

The database would be designed and managed using MySQL in phpMyAdmin and contacted via mysql2. It would use constraints to ensure unique entries and would have relations between tables. It would be used for storing login information, page content, quiz results and user statistics.

Middleware and packages installed on the server would include:

- cors to mitigate the risks of cross-origin HTTP requests causing errors
- mysql2 to query the database
- express-fileupload to handle files uploaded by users
- body-parser to read various types of request bodies
- ejs to render dynamic pages with templates

All plugins for node.js would be installed via the npm registry.

The testing of the database would be carried out through the use and monitoring of tables on phpMyAdmin. The testing of the REST API would be carried out via Postman. The testing of the client would be carried out via manual testing through Edge, Chrome and Firefox's developer tools. The client would be validated by the w3c markup validation service.

The entire report would be written in Microsoft word and stored in multiple separate OneDrives.

Additional reasoning for the selection of these resources can be found in the research section of the report. Overall, I chose these resources because I either had experience with them or had

researched them enough to know they were the most effective approach and would be capable of meeting the success criteria.

Planning

Schedule

The report was started immediately alongside the planning of the project. This was done so that the project would be documented from start to finish. This was the exception as all other tasks would have to be planned in the schedule. I decided to demonstrate my predicted schedule in the form of a Gantt chart to present it clearly and concisely.

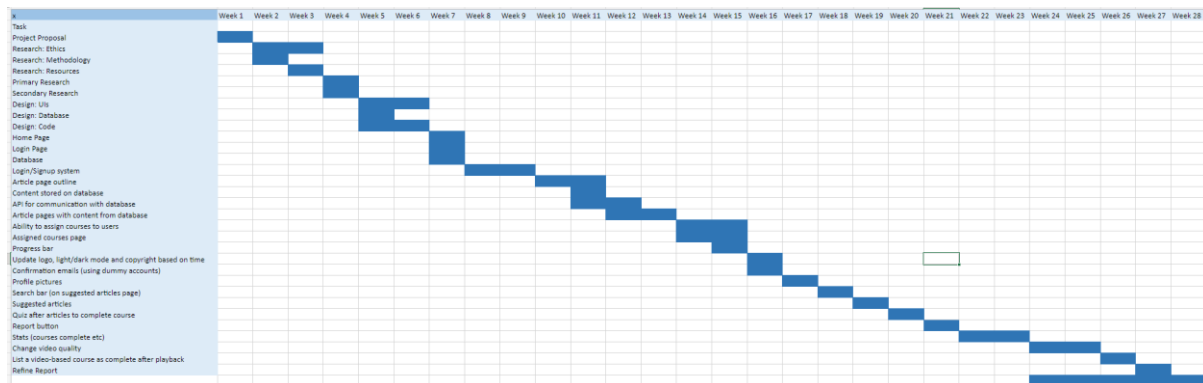


Figure 5: Gantt chart at the start of production

The schedule was subject to change if another more efficient plan was discovered over the course of additional research, or if any setbacks presented themselves. I also acknowledged that the schedule would potentially have occasional slowed development. This is due to other coursework designated to me by the University of Brighton, with more pressing deadlines, requiring my full attention. To make up for this potential setback, I always planned to work throughout the break between semesters one and two. I also planned to provide the project with as much overtime as necessary, to catch up with the schedule should I fall behind.

While the deadline for the project was the 6th of May 2023, I structured the schedule around a finished product having been developed by the end of April 2023. This was to reduce the risk of not meeting the deadline should the schedule be extended for any unforeseen reasons.

As the agile model was chosen as my methodology for the project, a burndown chart was to be created at the end of each sprint. The first sprint would begin upon the start of implementation. These sprints would last 2 weeks each.

The Gantt chart can be viewed in full by following this URL:

https://unibrightonac-my.sharepoint.com/:x/g/personal/j_ward26_uni_brighton_ac_uk/Ec4dW_-S27VJuiwTY8-ai5wBnLAFhD54khuMr4eSxwpenw?e=mbnDUN

Risk Analysis

| Cause | Effect | Mitigation |
|-------------------------------------|----------------------------------|------------------------|
| Hard-disk failure | Loss of progress (code & report) | Backups (GitHub) |
| Account hacked | Loss of progress (report) | Backups (OneDrive) |
| Erroneous code uploaded to the repo | Code breaks or loss of progress | GitHub Version Control |

| | | |
|---|---|--|
| CORS policy violated | UI cannot contact API/database | Use 'cors' middleware to enable CORS |
| SQL Injection | Loss or alteration of database | Input restrictions |
| The same named file is uploaded to API | File is overwritten | Check if the file exists/ name each image uniquely |
| A user tries to sign up with the same username or email as another user | Only the first user will be fetched | Use unique constraints on email and usernames |
| HTML markup is incorrect | Screen readers and extensions dependant on conventional markup don't function | Use the W3C validator to fix markup |
| A user attempts to access a page for employees without logging in | Unauthorised access | Redirect the user if not logged in |
| Potential loss of private/sensitive information | Unethical | Use a secure database |
| A user who is not an employee makes an account | Unauthorised access | Require an admin to signup users manually or require a code to create an account |
| Potentially real emails are used in the automated system | Emails are spammed | Use dummy email addresses or personally owned email addresses |

To mitigate the risk of permanently losing source code should a hard drive failure or similar event result in the loss of code, the entire project must have backups in multiple places. This was done by holding the source code on GitHub, a personal computer and Brighton Domain. On top of these being backup locations GitHub would also hold previous versions of the project should erroneous code be backed up accidentally. The code would be written in a VS Code window linked directly to the GitHub repository allowing for quick backup and rollback if necessary. Similarly, the report has multiple versions stored online on OneDrives across multiple accounts. Due to Microsoft word automatically saving regularly there is little risk of losing work due to a crash or network error. In this way, the loss of any work is highly improbable.

Depending on the location of the server and client the requests sent between the client and API may violate CORS policies. This risk is easily remedied as the solution simply involves utilising the cors plugin in the express application/API.

There is always a risk of SQL injection when a SQL-based database is concerned. To avoid this risk, there must be length restrictions in the database to disallow complex code from altering tables. The same restrictions can be added to the API contacting the database as an extra precaution.

To prevent users from potentially overwriting other users' files like profile pictures, I can have the images uploaded be named after the current date and time in milliseconds. This ensures uniqueness in the naming schemes, preventing overwriting via uploads through the client. To prevent overwriting via direct requests to the API I can make the API only allow the upload of an image if its path/name is present in the users' table and its name is not already present in the destination directory.

To prevent errors regarding users attempting to name themselves the same as others I can program the API to only add a new user to the database if the username and/or email address are not present in the database. I can also give certain elements in the database a unique constraint to prevent the same user from being added twice.

If a user attempts to access a page they are not supposed to have access to, then they will be redirected to the login/signup page so they can get clearance.

There is normally a risk of losing someone's personal data with a signup/login system. However, this web application is to be used inside of a company and so requires very little sensitive information about its users. Even so, any information about the users will not be disclosed by the API because it will only create and check the validity of accounts. While the training resources hosted on the domain and database could be sensitive to a company, there will be no way to access them unless you are logged in as a valid employee.

To prevent an outsider from creating an account, the company the application is given to could either: remove the signup feature and manually create the accounts for employees or have a verification code required to set up an account that goes on rotation to prevent attackers from gaining entry.

To prevent emails from getting spammed or ethical issues, the only emails held in the user table are either my own account or a dummy account.

The security of all resources is also enforced by their location on Brighton domains and in the MySQL database. Both of these resources come with their own security measures in place to prevent potential attacks.

Research

Research into every area of the project's planning is listed in this section. This includes technical research regarding which resources to use, user research in the form of primary research, research into accessibility, legal and ethical issues. This research was gathered via the study of resource documentation, articles, published books, query's from other develops and questionnaires. Any external research where a significant amount of knowledge was gained was cited in the references section.

Management of Legal & Ethical Issues

It is important that as the project manager I consider the potential legal, ethical, social, and professional issues relevant to the project. Some issues such as social issues will be minimised by the nature of an individual project in general, however Legal and Ethical requirements range widely with all issues needing to be considered in detail.

The Legal requirements my project must meet include: the Data Protection Act, copyright law and intellectual property law, among others. While the ethical concerns I need to acquaint myself with include: research on human subjects, ethics concerning holding personal data and ethics concerning public sites.

Intellectual Properties

As a project made under the tutorage of and to be marked by the University of Brighton, the intellectual property rights are as follows. *"The university requires access to intellectual property*

generated by students. As a condition of joining the university, students grant the university the right to use their work for academic purposes, including assessment and research, and for purposes relating to the administration of the university, including quality assurance and publicity.” In simpler terms, this means that the university holds the right to view the work in its entirety, but the intellectual property will remain with the students who created it, which in this case would be myself alone. This information would be an important detail to bring to the attention of any third party involved in supporting the project, however as this is a purely academic product rather than a commercial one and I hold no plans to involve a third party over the course of this project, no action is required on my part. If I were to involve a third party, I would need to write them a letter explicitly detailing that the IP involves the University of Brighton's critique.

Copyrighted Content

As this web application is not for actual commercial use and is only an emulation of a commercial product, it is instead classified as academic use. This means I am allowed to use copyrighted material, so long as permission is gained, and reference is given. If it were for commercial use explicit permission would be needed with evidence to host copywritten content. Copyright law also dictates that materials included in the project or project report taken from the internet are subject to the same proprietary rights as those originating from more conventional sources such as written/published materials. For insurance, any permission needed would be better off kept in written form to function as undeniable evidence.

Patents

A patent is a type of intellectual property licence that excludes others from making, using, or selling an invention. As the project presents pre-existing ideas and is not currently to be used for any industrial application it lacks patentability.

Research & GDPR

Any research involving human subjects must meet the standards of good ethical practice and relevant legislation such as the GDPR (General Data Protection Regulation). The GDPR is an EU regulation on data protection and privacy that went into full effect in 2018 (Data Protection, 2021). The GDPR was designed to allow consumers more control over how their data is handled by companies. The GDPR dictates that all websites must follow their rules regardless of where they are based and that if breached, must inform the consumers of the relevant information. As an important privacy law, my project must take care not to intrude upon the GDPR, as such the following precautions must be taken. Any user testing or primary research involving participants must:

- gather informed consent for all data to be processed
- only store needed information
- all stored data must be secure/confidential
- users/participants must have control over their data

(Boyd, 2021).

The GDPR dedicates its 2nd chapter to informed consent and states in its 4th article that “‘personal data’ means any information relating to an identified or identifiable natural person (‘data subject’); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person;” and “‘processing’ means any operation or set of operations which is performed on personal data or on sets of personal data, whether or not by automated

means, such as collection, recording, organisation, structuring, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, restriction, erasure or destruction.” (GDPR Archives - GDPR.EU, 2016) This means that any data that could potentially be used to find a person involved in the user testing or research is classified as personal data and any viewing of this data is classified as processing.

The GDPR states in Article 4 that you are held responsible for the collection of your data but also the passing of collected data to third parties. These are separated as “Controllers” and “Processors.” However, because I plan not to involve third parties in my project, I function without a processor and simply need to focus on data security.

In terms of participants having control over their data, the GDPR dictates that;

- “You must tell your users why you are collecting their data, what you are doing with it and how long you are keeping it.
- If a user requests it, you must give them a copy of all data you have collected about them.
- If a user says that their data is inaccurate, you must correct it.
- If a user requests it, you must remove all their data.
- If a user requests it, you must stop processing their data.
- If the user wants to move from your service to another service, you must allow them to transfer their data out of your service in a machine-readable format.
- A user has a special right to object to their data being used for certain purposes, like direct marketing.
- If you are using personal data for automated decision-making or profiling (like feeding customer data into a machine learning model to approve a home loan), then you are exposed to a bunch of extra requirements around explaining how your model works, having an appeal system and so on.”

(Geitgey, 2018). If the user/participant in any of my research or testing makes any of these demands I must honour them.

Taking the above facts into consideration, to ensure user testing/participation in primary research remains ethical and meets these regulations I decided upon these requirements for my research:

1. Consent forms must be filled out before the disclosure of any potentially sensitive information, this will walk the user through how, why, where and what data will be collected.
2. Alternatively, surveys can be filled out anonymously via platforms such as Google Forms so long as the information gathered holds no revealing information about the users. If the anonymous method is used, then the questions must be designed specifically to avoid gathering sensitive/revealing information.
3. All information gathered is to be stored in a protected location (such as a password-protected location on Brighton University’s secure OneDrive) and only the needed information will be stored.
4. As previously stated, the users must have control over their data, this will also be covered using the above criteria in the consent form and the fact I will allow the participants to stay in contact with myself as the project manager so that they can make any relevant demands concerning their data.

All data collected for research purposes will remain only used for that purpose. All data collected will be anonymised by separating data from identifying features but allowing users access to their data by assigning them codes in place of these features, thus conforming to the GDPR (this method will be used at all times anonymisation is required). This way if any unauthorised access were to happen then the data would be useless.

Design & GDPR

While the research of my project mostly concerns the collection of data, the design will have storage and usage of data on the pedestal of the highest priority. To avoid unwanted issues, only dummy accounts created by me, the project manager, will be stored in the database of the project for testing. This will maintain the functionality of the site and allow user testing without concern for ethical or legal issues. Any feedback from user testing will only be stored in the documentation which itself is stored securely on Brighton University's OneDrive and backed up on my password-protected PC running antivirus and encryption.

Cookies are also a potential dilemma but if permission is gained on the site before cookies are used then it is perfectly ethical and lawful. Although, because a user could refuse cookies the application cannot be reliant on them or must warn the user that the site may not work without permission. The former is preferable as it could be viewed as unethical if you are pressuring users to accept cookies.

However, if the account information were real then any sensitive information (such as emails, passwords, real names, stats etc..) would have to be:

- definitively secure
- retained only for as long as necessary
- disposed of definitively after use
- completely confidential
- Anonymous
- Correctly used

Despite using dummy accounts and anonymous feedback, any potentially sensitive information will be stored on Brighton University's OneDrive or Brighton Domains (where I will be securely running my database for holding accounts, courses, and other data integral to functionality). These are secure locations provided by the university due to their password protection, encryption services and other security measures used to avoid unauthorised access. No moveable storage will be used for this information and so there is no risk of it leaking in that way. There are no co-researchers to share the information with and the data will not be shared otherwise. Data will not be carried outside the EEA. Transfer of data will be done conventionally and securely. Accidental loss of data is the last potential remaining issue concerning how the system is designed but can be avoided through regular backups of important or sensitive data in an alternative but equally secure location (such as a private GitHub repository). Data will be disposed of securely (whether physically or electronically).

I must also note that the GDPR legally ensures that participants' or users' rights include...

- Subject access
- To have inaccuracies corrected
- To have their information erased
- To prevent direct marketing
- To prevent automated decision-making and profiling
- Data portability

All data will remain used only for its advertised purpose, whether it be feedback with user testing or account information with the login and signup information. The dummy accounts stored for the testing of the application will eliminate the need for anonymity on the database, however, the feedback will remain anonymous.

Subject-matter

While considering ethical issues in projects, any content found to be Controversial, Contentious, Sensitive, Embarrassing or Upsetting is considered an issue. As such the chief concern would be the management of content placed upon the web application, as at base it would hold no relevance with any of these concerns, being as inoffensive as it is. These potential issues would be fixed by either the administrator of the site once it is in the hands of a company using the site or could be managed by a report button if the site allowed the upload of user-created content. However, due to the site potentially allowing for a small amount of user-created content in the ways of usernames and profile pictures, a functioning or planned-to-function report feature would have to be implemented besides these additions to remain ethical.

Participants

Regarding participants in terms of ethics, these are the criteria I would use:

1. no participant will be younger than 18
2. no potentially vulnerable participants will be used
3. they will not be my colleagues or over-researched
4. They will speak/understand English
5. participants will not be recruited in public spaces
6. the positions will be voluntary
7. participants may remove themselves at any time
8. no one will be deceived.

I, as the head researcher and project manager, hold no financial interest in the project. The Health and safety of participants are of no concern as the project involves no physically harmful material whatsoever. No confidentiality issues exist so long as the participant's data is stored on the secure service provided by the university (the Brighton University OneDrive) and dummy accounts are used for the signup/login system of the project.

Informed Consent

As mentioned earlier, when talking about conforming to the GDPR, a consent form must be created to classify the project as legal. However, consent also affects whether a project is classified as ethical as well. To be ethical, informed consent must be given on the part of the participant and that can only be achieved if my consent form provides:

- in lay terms, the information for the participant to understand what the project is about and what is needed from them
- Who they can contact for more information (business contact details) and who is the organisation overseeing the research
- A date by which participants can withdraw their data from the study
- Assurances that their data will be held securely and treated correctly

A draft consent form provided by the university can be produced for this purpose, alongside any other relevant material required to support it.

Structure & Resources

I recorded all research into potential resources and structures in this section. Some knowledge was gained via research and experience from previous and current relevant modules, while other

knowledge was gained through research into how to best meet the objectives of this specific project. While I already had an ideal solution for how the objectives should be met, I sought to back up my decisions with evidence gathered through research to ensure/demonstrate that I had chosen the most efficient approach. Research into these technical issues and solutions was carried out by studying: the documentation of resources, articles discussing structures and resources, other developers' query's like those seen on stack overflow and published books discussing the relevant areas.

Architecture

I only investigated the two most popular architectural models at length despite acknowledging models such as the Space-based and Monolithic models. I did this because the community of developers agree that they are the most useful as well as the most documented. I decided that the peer-to-peer and client-server models were the architectural models best suited for my project. I researched, compared and contrasted them to decide which model to move forward with.

Peer-to-Peer

P2P models are a type of architecture where the devices involved are not separated into categories such as clients or servers. In P2P models, each device is defined as a node in the system with the same responsibilities and restrictions as one another. This hierarchical approach allows the efficient distribution of resources between nodes, making on-demand facilitation of these resources the model's greatest strength. Equally, any service using the model will be reliable because of the lack of reliance on one centralised server. There are no administrative issues as each user manages their own system. There is no server to setup making it inexpensive and easier to implement. Because all devices are able to connect to each other without security measures it allows malware to be shared and spread fast. Due to content being held in multiple different locations any form of backup becomes a complex process. The peer-to-peer model is highly scalable, with extra clients not degrading performance, the larger the network gets, the more performance will degrade. BitTorrent, LimeWire and Apple Cash are examples of widely known/used P2P systems.

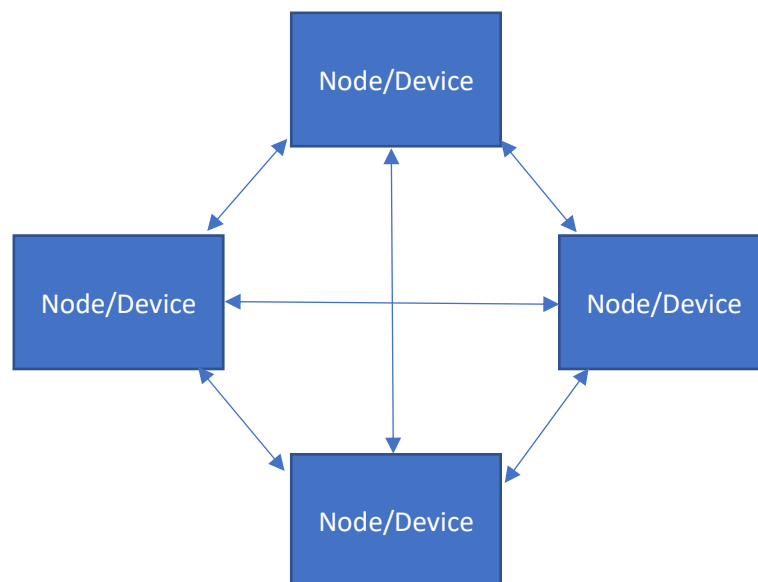


Figure 6: Diagram of a Peer-to-Peer architectural model

There are four main structures of the P2P model:

1. Unstructured P2P – has the advantage of easy connection between devices but resources become hard to find due to the lack of structure
2. Structured P2P – hard to set up but resources are easy to find

3. Pure P2P – each node in the architecture plays the same role due to the lack of a centralized server/device
4. Hybrid P2P – a device acts as a centralized server allowing for some advantages seen in the client-server model

| Advantages | Disadvantages |
|--|---|
| Simple Implementation | The non-centralised approach makes resources hard to find |
| No administrative issues | Performance degrades with larger networks |
| Non-centralized approach makes it reliable | Backups become difficult |
| Inexpensive maintenance | P2P is often used to share illegal or harmful content |
| High scalability | Assigning permissions is the only security |
| Real-time resource sharing | P2P networks can be compromised with remote access |
| P2P is always faster than client-server | Malware can easily spread between devices |

Client-Server

Client-server is an old form of architecture, being introduced in the early 1980s before peer-to-peer in the late 1990s, however, this doesn't discredit the model altogether as it is still widely used. The client-server model is the most popular architectural design, in which devices are segregated between clients and servers. A client is defined as a device with the purpose of sending requests while a server is defined as a device acting as a medium between clients and resources, by processing the requests and returning the appropriate resources. The client-server model can be set up on a singular device, a small network or the internet and is compatible with the concept of Cloud computing. Servers can have more clients or servers embedded in them for greater flexibility.

Conventionally the client-server model involves the client sending a request by entering the URL of the server into the browser. The DNS then searches for the requested resource and responds with its IP address. The client's browser then sends an HTTP request to the IP and obtains the resource.

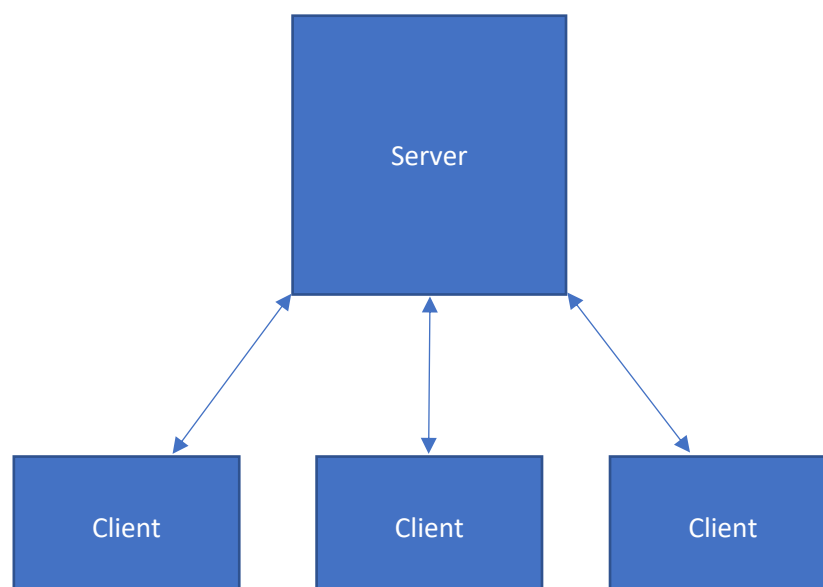


Figure 7: Diagram of a Client-Server architectural model

The client-server model can be separated into four different iterations of itself when concerning web development. These iterations are known as 1-Tier, 2-Tier, 3-Tier and N-Tier. The client-server model can also be separated into three distinct layers:

- The User Interface/Presentation Layer – the visual components the user will interact with, use to send requests and receive responses
- The Service/Application Layer – the intermediary between the UI and database used to provide functionality to the application, handles most application processes
- The Database/Backend Layer – the data tier, storing all data/resources to be managed by the application

The 1-Tier model defines the type of client-server architecture where the UI, Service layer and Database are all embedded in the same system. MS Office is an example of 1-Tier architecture. The 1-Tier architecture is not suitable for my application or web applications in general as the server can only be accessed within a single device.

The 2-Tier Model involves the UI and Database being separated by having locations on separate devices. There exists no separate intermediary in this model as the Service layers' responsibilities are handled on either the server alongside the database or on the client behind the UI. The client and server communicate with each other via the internet protocol. The direct connection between client and server ensures fast results and easy design, maintenance and modification. The disadvantage of this model is that the more clients connect to the server, the greater the performance suffers. Equally, if the Service layer resides on the client then the client will have to be updated every time the server is.

The 3-Tier architecture is the most common type of multitier architecture. This is when the UI, application processes and data management are all hosted on separate devices. The 3-Tier architecture disallows the client from directly interacting with the server containing the database. Instead, the client interacts with intermediary middleware which then passes requests and resources between the UI and database. Using middleware increases flexibility, security, data control and integrity by having the database be invisible to the client. The client is responsible for the presentation layer, the middleware server is responsible for the service layer and the database server is responsible for the database layer. N-Tier architecture is the descriptor for all other Multitier models. In an N-Tier model, the N simply stands for the amount of tiers/layers/middleware used as intermediaries between the client and the database.

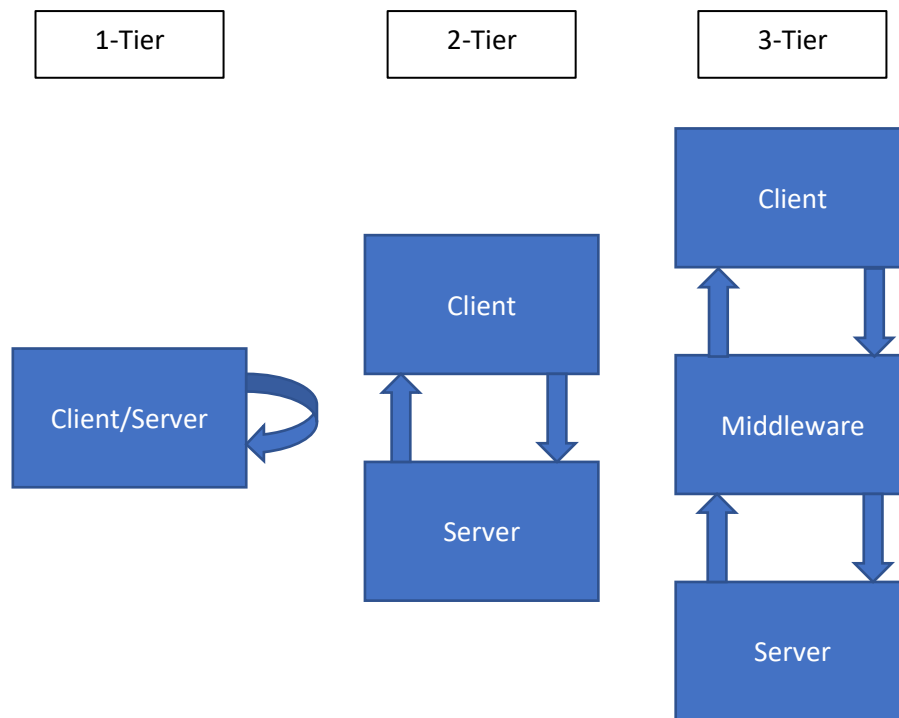


Figure 8: Diagram displaying differences between iterations of the Client-Server model

| Advantages | Disadvantages |
|---|---|
| Good flexibility | The whole architecture is disrupted if the server is |
| Connected devices can be controlled centrally | Security must be added to prevent unauthorised access |
| Efficient at managing requests and responses | Malware can be contracted from the server |
| Centralised data is more secure | Requires an OS capable of network access |
| Inexpensive maintenance | Can be expensive to set up |
| Centralised data is easier to manage | Excess traffic can cause congestion |
| Greater control over processes | Errors on the server must be remedied fast |
| High scalability | |

As a subset of the client-server model, servers can host REST APIs. REST APIs can be provided by middleware and are stateless. Stateless means that the client or server need not know the current state of the other. REST APIs separate the client and server meaning that the client doesn't need to know about the server and vice versa but can still communicate with each other. The advantage of this is that the client or server can be altered without harmfully affecting the other. The server and client communicate through the API meaning that they must only know what format to send or receive data or requests in. REST APIs ensure reliability, scalability and fast performance.

Peer-to-Peer vs Client-Server

I compared and contrasted the P2P and Client-Server architecture. By doing this I would be able to showcase and define the reasons for why I chose to use one over the other.

| Peer-to-Peer | Client-Server |
|--------------------------------|--|
| Devices have unique data | Centralisation for simple data-management |
| No segregation between devices | Devices are segregated between clients and servers |

| | |
|---|---|
| Peers can both request and provide services | Data is only requested by clients and provided by servers |
| Useful for small networks | Useful for large networks (e.g. the internet) |
| Designed for connecting peers | Designed for sharing data |

Since the P2P architecture is designed for the design of small local networks and the Client-server architecture is better-suited for web applications available through the internet, I would justify that the client-server architecture would be the best route to take the project in. It makes sense for a company's training videos to be available via client-server as many users would find it an inconvenience to connect to a peer-to-peer network to access data. As the project is geared towards the sharing of data the client-server is still the more appropriate of the two as no two users are supposed to be connecting with each other, but they do need easy access to the resources which the client-server architecture is more suited to provide. The centralisation of data allows the simple addition of new data to be accessed by users as well as greater control overall. The client-server approach was selected because of all the advantages listed above as well as its compatibility with the project objectives. I decided on using either a 2-Tier or 3-Tier architecture with the use of an API to separate the client and server. While a training site could be hosted on an intranet closed from the public I decided to host mine on the internet because it will be more accessible for grading and demonstration of this project as well as being more convenient for employees in a real-world scenario.

Server-Side Rendering vs Client-Side Rendering

Amongst developers in the web development community, there is an ongoing argument over whether rendering is better done on the client or server side of the client-server model. This argument has valid critiques on both sides and whichever method I chose to adopt would greatly change the architecture of the application.

To understand the advantages of either argument I had to research SEO (search engine optimisation). SEO is the process of optimising web applications' content and configurations to appear more relevant to search queries. Web applications with good optimisation will be ranked higher by search engines, this means they will appear in more users' queries and get more traffic as a consequence. An SEO-friendly site will ensure popular search engines rank it high in the search results during the rendering process. So, the more SEO-friendly the rendering method is then the easier to find the application would be.

Google has ensured that most client-side rendering frameworks used to create user interfaces are SEO-friendly. Client-side rendering functions by redirecting requests to an HTML file, then fetching the JavaScript and compiling it before rendering. As long as the device has a good internet connection then this method is fast, reliable and SEO friendly. However, many now argue that this is not the optimal solution despite its conventional use.

The use of server-side rendering was the original solution for rendering pages which were mostly forgotten in favour of the modern client-side solution. Server-side rendering used to involve using a language like PHP to render the page on the server and send the populated HTML page to the requester with little to no use for rendering on the client's part. While this alleviated stress on the client, this process would have to constantly be repeated from the beginning and so was retired due to its inefficiency in comparison to client-server rendering. The argument was made that most modern clients are powerful enough that rendering on their part is not a very laborious task, and so is not worth attempting to avoid. Rendering dynamic pages on the server side make this method not SEO-friendly for any dynamically rendered pages.

The reason the debate is reinvigorated is because of a new solution popularized by the React framework. The react framework allows only necessary data to be fetched via dynamic routing. This is a hybrid between server-side rendering and client-side rendering known as an isomorphic or universal app. The requested pages are still pre-populated putting less stress on the client. While this new method seems efficient it is still new and so lacks a lot of documentation and community support.

Because the client-side rendering solution is the conventional approach in the modern day it holds a lot of detailed documentation as well as community support. But I do think it is worth considering the use of server-side rendering in places as it would alleviate strain on the client and would allow the easy creation of dynamic pages based on database content. Overall, I decided to pursue the conventional method while potentially using a backend framework like express for server-side rendering.

Environments

Node.js

Node.js is an open-source, cross-platform server environment allowing developers to use JavaScript with the additional functionality of a server scripting language. While JavaScript usually runs via browsers' JS engines like V8 and SpiderMonkey, Node.js allows JavaScript to run outside of browsers by using chromium's V8 engine.

Node is event-driven allowing it to develop web applications that can react to requests through open connections. Web servers for back-end processes are simple to write in node but most developers use pre-made packages for that purpose like Express and Fastify. Node packages are widely ranging community-made libraries and tools to be used by other developers. The ecosystem of over 1.3 million packages is managed and accessed by the node package manager. NPM is installed automatically alongside Node and allows for the fast installation of many different packages. NPM lacks quality control and moderation, so only community/widely approved packages should be used to avoid malware or erroneous packages. Node had to remove all malware in 2021 using WhiteSource Diffend to remove 1300 packages, further proving that the registry must be navigated carefully. Packages range from frameworks to toolkits and serve various purposes. Packages can be installed locally or globally through the npm commands in any command line interface.

Overall Node can be used for servers, clients and tooling while npm is used as a command line tool for installing and managing node packages. I decided I would use node to create an API with the use of a framework and would explore the ecosystem for the relevant plugins required to meet my objectives. The use of Node would help speed up development time as well as enable the designing and testing of back-end processes locally. Node was required to install the packages I would be making use of in my API to allow communication between the UI and database. It would allow the API to be created in JavaScript with the use of premade frameworks and middleware to improve development time.

Brighton Domains

Brighton Domains was the web service provided by the University of Brighton to host our various resources on. I decided to use domains as not only was I familiar with the resource, but it required no cost while allowing me to host all my deliverables in the form of APIs, UIs and Databases. Domains appears to use a WordPress interface used to manage various content on a section of one

of the University's servers. This section is dedicated to each student with a gigabyte of disk space which is fit for purpose when storing mostly text and code. While it can link to various other types of UI, they all have the authority to access and change content stored on the system.

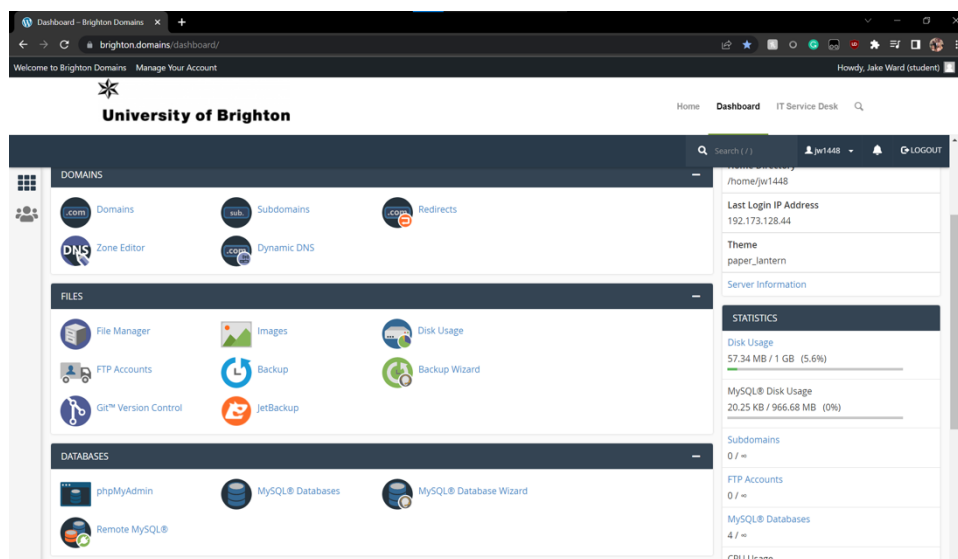


Figure 9: Brighton Domains dashboard and supported applications

There is support for automatically creating node applications that I would be able to create an API to communicate between the client and database. This service also allowed the various required packages to automatically be installed and was compatible with all packages I would be requiring.

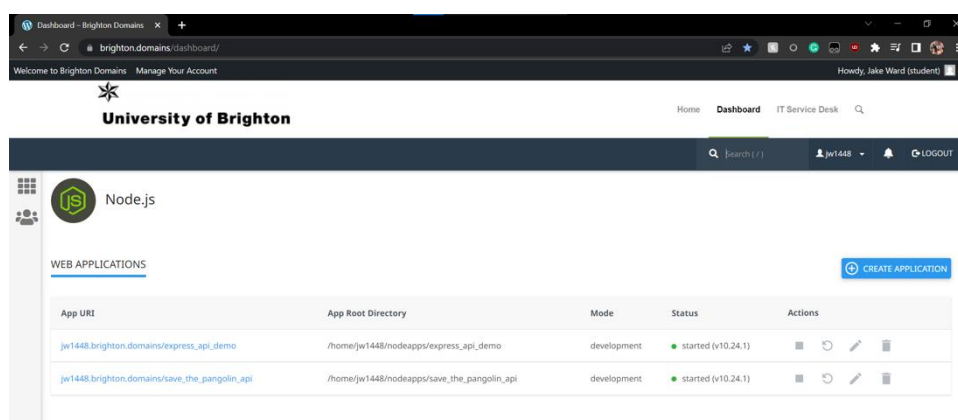


Figure 10: Brighton Domains automated Node application creation UI

Domains allowed a user interface to be publicly accessible through a repository labelled "public_html". This would be the location I would upload the final version of the user interface to. Once the final version was uploaded it would be delivered as a functional web application via a link to this public directory. While files could be uploaded to the domain via the FTP protocol through applications such as Filezilla, Brighton Domains also hosted a file manager that would allow the fast upload of all files as well as the ability to edit documents already hosted on the server.

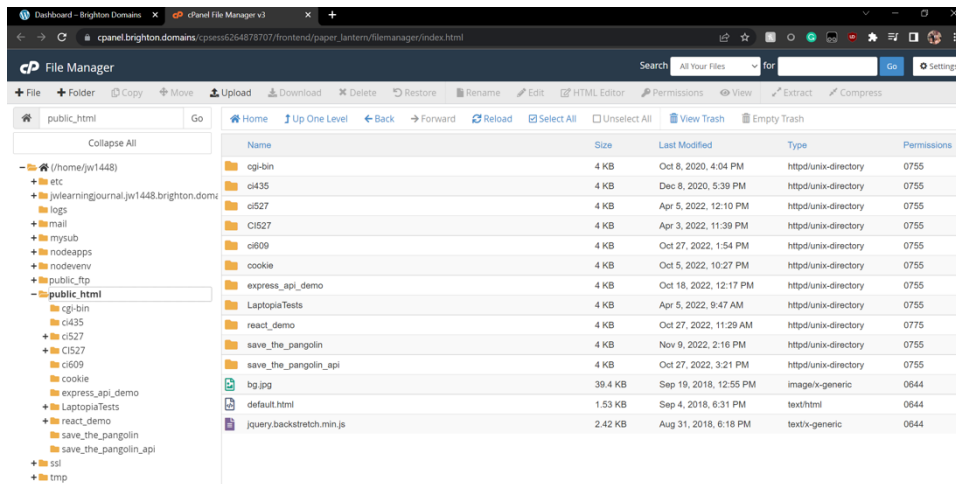


Figure 11: Brighton Domains file management UI

Brighton Domains allowed the automatic creation of databases and users with access to them. These databases could then be accessed and managed in phpMyAdmin. This would be where I would design and manage my database.

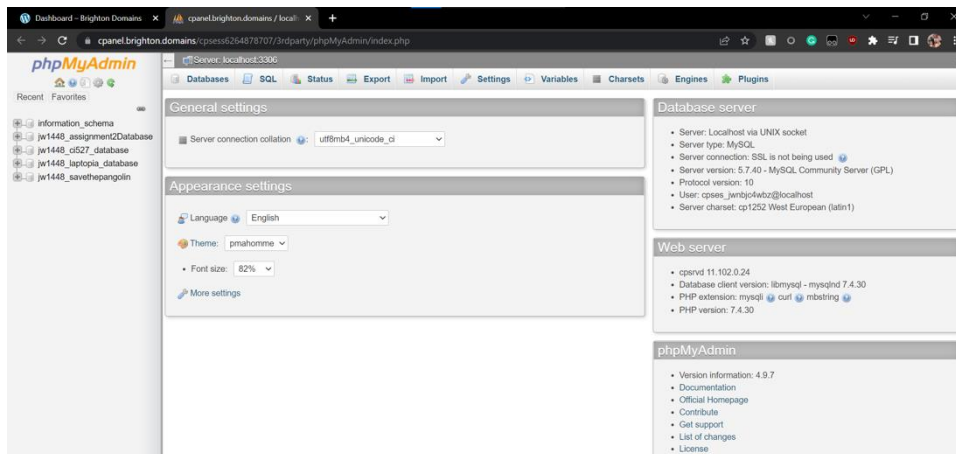


Figure 12: phpMyAdmin UI featured on Brighton Domains

Brighton domains also supported the creation of subdomains, but I decided not to use this feature due to a lack of necessity. Domains also supported the use of git and automatic creation of other types of applications such as python. I decided not to make use of git via domains because I would already be programming everything through VS Code which was already linked to GitHub.

By setting up back-end services on Brighton Domains it is inexpensive to me and I need not concern myself with running my own server constantly once the product is finished.

Frameworks

A framework is defined as a platform for developing software applications. They are used as foundations for developers to help kickstart projects and increase development time. There are different frameworks for different languages and purposes, but I decided to investigate frameworks compatible with Node in compliance with the “JavaScript Everywhere” mentality. I chose to use investigate Node frameworks because of their wide variety, accessibility and large community support. Frameworks are often tested and updated frequently to ensure they are stable and up-to-date for modern developers. Frameworks are not all community-made as some frameworks are

produced by corporations, like how Angular is built and supported by Google. Node frameworks are installed via NPM and included as dependencies in the package file.

Frameworks and libraries are not the same things despite many using the terms interchangeably. Libraries are sets of code to be used wherever a developer feels appropriate, in whatever way they want. Frameworks require a pattern to be followed in order to function. In this way, the framework holds control over an application's functionality and the developer just adds to correct parameters. Frameworks are written for front-end services, back-end services, tooling and many other purposes. A framework can be installed locally or globally alike any other packages seen in the npm repository.

Frameworks are used to reduce any potential errors in code as the pre-built code will have already been tested. Because of this, code is often more secure and efficient than anything written from scratch. Developers can focus on writing code specific to their application if all generic code has been written as part of the framework.

I decided I would use frameworks for the back-end processes and tooling, but I documented the research I did into all types of frameworks nonetheless.

React

The React framework is a free-to-use front-end open-source library used for building interfaces based on UI components. React was released on the 29th of May and is maintained by Meta (formerly Facebook) as well as a community of individual developers. Jordan Walke created the JavaScript component-based library after being inspired by XHP (PHP's HTML component library). While React is not technically a framework as it is only concerned with UI rendering, it is still treated like one and named as one by many. React is one of the most popular front-end frameworks/libraries with more than three million active users and more than one million web applications have been made with it. React is used for the development of both the Facebook and Instagram UI.

A proprietary language called JSX is the language used as the primary language in React. It incorporates elements of XML and HTML into base JavaScript. Toolchain is utilised to transpile JSX code and virtual DOM is utilised to improve rendering. Babel is the JavaScript compiler/transpiler used in React to transpile JSX to JavaScript. Conditional rendering is a useful feature of React that allows a component to render or not depending on the application's state. React is a large package with over 200MB of code that most developers will barely use, this is a lot of wasted space for development but only the relevant content will be featured in the deployed application. The React developer tools are also available as browser extensions.

It can also be noted that React is the base used for the development of the popular framework Next.js. However, Next.js has a very lacking ecosystem which would mostly hinder development rather than help it.

| Advantages | Disadvantages |
|---------------------------------------|-----------------------------|
| SEO friendly | Lack of documentation |
| Useful for creating dynamic pages | Constantly moving ecosystem |
| Simple testing | JSX must be learnt |
| Reusable UI components | Only concerns the UI |
| Easy to learn | |
| Rendering enhancements | |
| Developer tools available in browsers | |

Angular

Angular is a free TypeScript-based open-source web application framework maintained by Google. Angular is a rewrite of Angular.js which was made by the same team and recently discontinued. Many developers have expressed a lack of interest or aggravation in Angular due to its major differences from the previous deprecated version.

Angular's rewrite consists of four major differences. Angular introduced a command line interface that wasn't present in Angular.js. Angular.js used JavaScript whereas Angular uses TypeScript. The use of TypeScript allows for scalability, clean code and better tooling. Angular.js used directives whereas Angular has component-based architecture, increasing development time and code quality. Angular is designed to include mobile-friendly features whereas Angular.js was less concerned. It can also be noted that up until the previous versions' deprecation Angular was known as Angular 2. Despite controversy amongst the community Angular is currently the fifth most used JavaScript framework worldwide with 20.39% of web developers using the framework.

Angular allows two-way data-driven access to the DOM meaning there is no need to manually access it and manipulation of the DOM is easier. Angular is said to be useful in creating aesthetically pleasing UIs with a powerful ecosystem at its disposal. It makes use of the RxJS library to allow asynchronous programming. The framework comes equipped with CPU performance-enhancers ensuring faster performance than native frameworks. This fast performance allows tests and changes to be seen in real-time to ensure continuous development. It uses plain HTML templates instead of custom versions of them to help ease the task of development.

Angular is part of the MEAN stack framework, an acronym representing the technology used for each major area of web development: MongoDB for databases, Express for the back end, Angular for the front end and Node for the web-server. MEAN is not a concrete industry standard as variations such as MERN exist, wherein React is used in place of angular. While Angular was designed for the MEAN stack and performs well within one, it was not designed to be integrated with other resources like MySQL.

| Advantages | Disadvantages |
|------------------------------------|---|
| Supported by Google | The ecosystem is divided between versions |
| Two-way data binding | Complexity makes it hard-to-learn |
| Component-based | Migration from legacy is time-consuming |
| TypeScript | Databinding decreases performance |
| Asynchronous programming supported | Heavy framework |
| Creates attractive UIs | Designed for MEAN stack |
| Powerful ecosystem | Both TypeScript and RxJS must be learnt |
| Fast (Server) Performance | |
| Plain HTML | |

React vs Angular

| React | Angular |
|--|--|
| JavaScript Library | Full framework |
| Supported by Meta/Facebook | Supported by Google |
| Uses JSX | Uses TypeScript |
| Used for creating UI with varying data | Used for creating progressive and single-page web applications |

| | |
|--|---|
| Moderate learning curve | Steep learning curve |
| Used to create reusable front-end elements | Used to create dynamic sites compatible with the MEAN stack |

Overall, I decided that it would be best to create the UI without the use of a front-end framework or library. This is because I didn't want the client to be reliant on many libraries as well as the fact that learning to use another new framework would dramatically increase development time when I only had a limited amount of time to complete the application. UIs are easy to design in base HTML, CSS and JavaScript anyway and overcomplicating the process needlessly wouldn't be a good management decision. While I acknowledge the potential usefulness of having components premade for my UI, I would sooner design elements myself to show my competency in the area. It should be noted that my decision was made by weighing all advantages, disadvantages and features listed above.

I knew that I would be using a framework for back-end processes as PHP is an outdated method of handling these processes and the newer JavaScript frameworks are the most efficient modern equivalents. Node is also designed to allow server processes and so would be made better use of on the back end. So, I decided to research the most popular back-end-oriented frameworks next.

Express.js

Express is a free back-end web application framework for building REST APIs within Node.js. It can be installed like all other Node packages through the NPM registry. Express is so conventional that it is named by many developers as the de facto standard server framework for Node. Express is part of the MEAN stack and often its alternative counterparts like the MERN stack. Its conventional and widespread nature ensures great documentation and tutorials to help learn the framework. Express is designed to not obstruct any functionality from base Node. Express isn't tied to any one specific front-end or middleware allowing for greater freedom in the choice of tools when designing a web application. Feathers, ItemsAPI and KeystoneJS are just three examples of frameworks built upon Express.

Express advertises the use of its application as opposed to being installed as a dependency/package. However, the application is designed to create web applications as opposed to simply APIs. Because I wish to design only the back-end API with express the package is the more relevant resource to install. Express allows access to all elements of a given request allowing for greater control over the application and its chosen responses. Express makes use of routing both static and dynamic, this enables the modularization of code as well as adding or adjusting the default behaviour of the API. Express allows for the communication between API and database that was to be used in my project, but only supports communication with SQL and NoSQL databases. For an Express server to communicate with a MySQL database a middleware like mysql2 would have to be used.

Callback hell can be an issue but is easily avoided with the use of Nodes' asynchronous functions. While middleware is a very useful tool to be made use of in Express, some middleware can result in client request issues and so the relevant handling must be put in place. Unlike I/O operations, CPU tasks are not executed in parallel and so if run on the server will result in a blockage, it is for this reason express is highly recommended to not be used for CPU applications.

| Advantages | Disadvantages |
|---|---|
| Decreases development time | Callback hell |
| Enables simple server scripting functionality | Middleware integration can cause issues |
| Efficient handling of I/Q requests | Not efficient in running CPU tasks |

| | |
|--|--|
| Simple learning curve | |
| Large ecosystem | |
| Large community and useful documentation | |
| Simple middleware integration | |
| No extra cost | |

Django

Django is a free open-source web framework using a mixture of Python and HTML. It follows the model-template-views architectural pattern and was released on 21st July 2005 by the non-profit Django Software Foundation. Its objective is to ease the formation of multifaceted database-oriented websites. It has a pragmatic and clean design which is also made use of with machine-learning. It is preloaded to care for content management, RSS feeds, site maps and user authentication. It is versatile enough to be used for both social networks and calculators and scalable enough to handle large traffic. It allows simple integration with MongoDB databases and takes security very seriously.

| Advantages | Disadvantages |
|------------------------|---------------------------------------|
| Active Community | Not suitable for small projects |
| Flexible | Lacks conventions |
| Speeds up development | Difficult to learn |
| Versatile and scalable | Cannot simultaneously handle requests |
| Secure | Not JavaScript-based |

Express vs Django

Node/Express is faster than Django and is more compatible with the other resources I was likely to use. It was for this reason that Express would be the obvious choice of framework to design my middleware/API. JavaScript being Express's language is an advantage as I would have to be competent in the language to use the other resources. On top of this Django's functionality is available as a package in npm to be used in conjunction with Express. Express is the conventional standard and so holds a large supportive community and easy-to-use documentation.

Toolkits

Gulp

Gulp is another popular package available through the NPM registry. Gulp is a free JavaScript open-source toolkit used as a streaming build system in the area of front-end web development on Node.js. It was released on the 26th of September 2013 and was created by Eric Schoffstall. Gulp is based on Node streams and isn't tied to any one framework or application type, making it a very flexible and versatile toolkit. Gulp is still a popular JavaScript toolkit despite the rising popularity of webpack.

Gulp is used for:

- Automating the workflow and build processes
- Transpiling
- The concatenation of library files
- Compiling SASS
- Bundling JavaScript and CSS
- HTML, CSS, JavaScript and image minification

Gulp is useful because of its wide ecosystem of plugins as well as its easy installation and use. Gulp is conventionally installed via NPM globally but can be installed locally. To function for my purposes, a file named 'gulpfile.js' houses all tasks to be executed and are executed upon the 'gulp' commands use in the relevant directory. This would be a useful toolkit to optimise the front-end UI upon completion. Gulp's use of back-to-back processing of multiple files ensures fast processing. The automatic configuration of files allows developers to focus on functionality. The debugging process is reported to be unreliable on occasion and so additional validation maybe be required as well as manual debugging. Gulp is currently on version 4.0.2 with an unpacked size of 20kb and nearly 1.5 million weekly downloads.

| Advantages | Disadvantages |
|---|---|
| Versatile/Flexible | Unreliable debugging |
| Vast Ecosystem | Limited documentation for plugins |
| Able to process multiple files back-to-back | Concepts can be confusing |
| Configures code for the developer | Third-party libraries can cause dependency issues |
| Only four functions | |

Webpack

Webpack is currently the most popular JavaScript toolkit as it took the title in 2020 by overtaking Gulp. It is classified as a module bundler with its main purpose being to bundle JavaScript files for usage in a browser. It is also able to transform, package or bundle a wide range of other assets and resources. Its sponsored by large companies like Discord and Trivago a indirectly supported by Google through the development of the Google Web Toolkit. Webpack is highly popular with over thirty million weekly downloads. It is currently running on version 5.75 and has an unpacked size of 4MB (a heavy size when compared to Gulp).

Webpack has an entry point in the form of a dependency graph, it will follow this graph to know what to bundle. Its output is defined as the location and format of the desired bundle. Loaders are used to convert different types of files into modules before adding them to the dependency graph. Plugins are used for functionality like modification and optimizing bundles.

Webpack has to bundle all modules when starting a development server. This means it typically takes 30 – 150 seconds to finish booting a web server. It is also infamously hard to learn how to configure. Pages in webpack do not need to fully reload when small changes are made. It also saves time by reducing the load time of websites during debugging. Webpack provides a module system based on ECMAScript. Files are treated as modules and the features of one file can be used in another. Minification of assets are available like in Gulp and it also supports feature flagging.

| Advantages | Disadvantages |
|----------------------------|---------------------------------------|
| Decreases development time | Heavy for a toolkit |
| Minification | Confusing documentation for newcomers |
| Feature flagging | Hard to configure |
| Modularised code | Slow developer mode speeds |

Gulp vs Webpack

| Gulp | Webpack |
|------------------------------|---------------------------------|
| Easy configuration | Difficult configuration |
| Efficient and fast execution | Slow compared to other toolkits |
| Difficult to rerun tasks | Easy to rerun tasks |

| | |
|------------------------------------|------------------------------|
| Stable due to less memory required | Tendency to crash |
| Less popular with less support | Good community support |
| Lightweight and simple | Heavy and complex |
| Controlled process flow | Less controlled process flow |

While webpack is designed specifically for bundling and configuration, its harsh learning curve leaves Gulp as the more viable choice due to its ease of use. Because the development time is limited the easier the tooling process is, the more features I can focus on developing. However, due to gulps' difficulty with rerunning tasks if an error occurs, it would be best practice to only initiate a build when a backup has been created via GitHub or Similar. I decided not to completely discount webpack and would consider using should I become adept at handling the difficult configuration.

Database Management

MySQL/phpMyAdmin

While SQL can be written in many different iterations of the language such as SQL server and NoSQL, I decided it would be best to design, create and maintain my database backend with MySQL. MySQL is a free open-source relational database management system released on the 23rd of May 1995. I already have free access to the automatic creation and maintenance of a MySQL database via Brighton Domains phpMyAdmin UI. By making use of this free resource available to me I can host a database constantly with no budget. It is important that the MySQL database can be run constantly on the University provided server because my web application would need constant access to a database in a real-world scenario. I had also learnt MySQL over the course of my two years at the university and so was confident in how to use the resources with relations, constraints and all other functionalities. While I was committed to using MySQL because of these advantages unique to me, I decided to justify my decision further with research.

While I can write and design tables using MySQL script I can also design and alter tables with the tools provided in the phpMyAdmin UI to increase development time and productivity. MySQL boasts strong security, being used by companies such as Meta and Twitter. MySQL supports continuous uptime and on-demand availability which is functionality the design of my application would be reliant on. There is no cost for the software and for me, there is no cost to run it on a server as it is already covered by the University of Brighton as one of their resources available to me. MySQL is advertised as efficient in its execution and handling of queries so much so that it is reported that a database can handle millions of requests per day. phpMyAdmin allows the import and export of SQL in the form of script, this allows me to present the code for my designed table with ease.

Disadvantages can present themselves in certain scenarios. Certain SQL commands are not present in older versions of MySQL, however, I am aware of all commands and constraints that I will need to use to successfully design my database. Some functionality of MySQL can be dependent on certain add-ons but the University will not be removing any of the add-ons their database is reliant on any time soon. MySQL can have trouble supporting large databases with efficiency but the database I would be designing would not be complex or large enough for this to be a legitimate concern. MySQL, while widely supported, isn't community driven and so can become outdated while relying only on the developers for updates. This shouldn't be a problem as the functionality of the database is simply to insert, alter and query results in tables. This basic functionality will not require any new updates so long as I make use of the relevant middleware to communicate between the client and the database.

Middleware & Packages

[cors](#)

The cors package is a piece of middleware required to allow Express to enable CORS with various options. This package's inclusion in an API prevents most errors which can happen concerning cross-origin requests. It is because I have encountered these types of errors on multiple occasions that I am aware that its inclusion will be useful. The other method for avoiding CORS errors is by using a proxy which could cost money, so the use of the cors package is a much more efficient workaround. It can be installed via the NPM registry, consists of 20kb and is currently on version 2.8.5. To prove the integral use of this package it can be noted that it has over nine million weekly downloads.

[mysql2](#)

'mysql2' is a middleware package available for installation through the npm registry. It is a database driver that can be used for communication between Node servers and MySQL databases. It is simple to use with extensive and understandable documentation. It is a continuation of MySQL-Native and was rewritten to be in its current state. It is free from native bindings so can be installed on windows, mac and Linux. It supports prepared statements, compression, SSL and non-utf8 encodings. In order to communicate with my database from my API I will need to make use of this package. The current version and version I will be using is 2.3.3. It has an unpacked size of 454kb and over one million weekly downloads.

[body-parser](#)

The body parsing middleware is used to parse incoming request bodies into a single type of parameter saved under a singular property. The developers note in the documentation that because the body parsing of the package implies that it is used accepting for user-controlled inputs. As such any inputs should not be trusted and should be validated before use in an API or Node application. The body-parser middleware available via NPM includes parsers for Raw, JSON, Text and URL-encoded request bodies. It is currently on version 1.20.1 and has an unpacked size of 60kb. It is a highly popular and useful package with almost thirty million weekly downloads. The use of this package would allow me to accept varying types of request bodies and well as simplify the process of accepting and validating request bodies.

[express-fileupload](#)

The Express middleware 'express-fileupload' allows for the easy upload and handling of files on an Express server. I investigated the use of this package in case I decide to accommodate profile pictures but the use extends to all types of files. The package can be installed via NPM and wraps around the Busboy middleware. It has a small size of 1.22MB and is currently on version 1.4 with an average of two hundred thousand weekly downloads due to its specific nature.

[ejs](#)

Allowing for the creation of dynamic pages to be returned by an Express API, ejs is a node package as well as a simple templating language. It is very popular with average weekly downloads of above eleven million. It has an unpacked size of 140kb and is currently on version 3.1.8. EJS is a language that allows you to generate HTML markup with plain JavaScript. EJS stands for embedded JavaScript templates and is used to create templates to be populated/rendered by a Node application (namely using express). EJS and its corresponding package allow an Express server to generate and respond with dynamic pages created via templates written in EJS and populated by Express. It has a large and active community and is easy to debug. This simple SSR solution allows for my Express API to create dynamic pages created from MySQL database responses e.g. article pages made dynamically using an EJS template. EJS can be installed through npm like any other Node package.

Testing

Postman

Postman is a free API platform built in the Electron.js framework designed for developers to help build, test and iterate their APIs. Initially released in 2012, the application has reported that it remains the world's largest API hub with over twenty million active users and seventy-five thousand APIs. I decided to use the service for testing my Express API as well as helping generate and learn new types of request-sending techniques.

The UI is incredibly user-friendly with a simple interface using templates to create requests. Postman is highly accessible with easy installation via the application or browser extension. It makes use of basic request tracking abilities to notify users about the responses and status codes as well as supporting all HTTP methods. A specifically useful function of Postman is the ability to generate code based on a template, allowing for faster development.

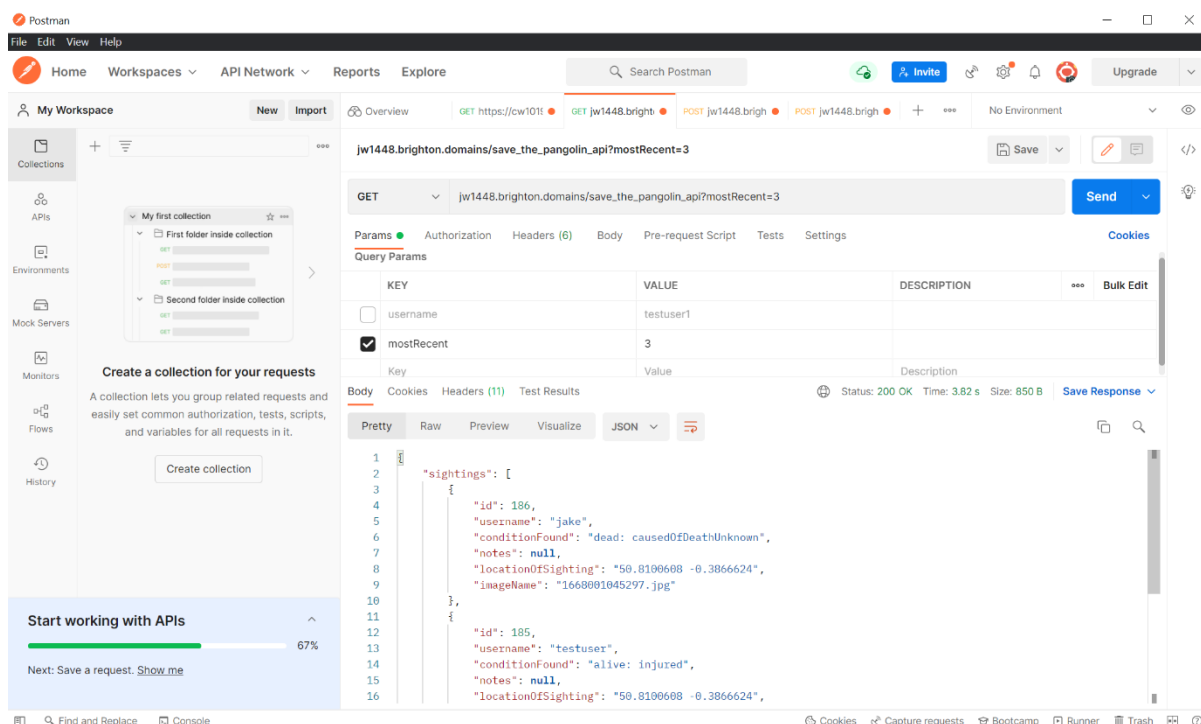


Figure 13: Successful postman GET request made with a template

While postman is highly suited to my needs in this project there are limitations that I must acknowledge. Postman is designed for testing REST APIs but is ineffective in testing other types such as SOAP APIs. Postman does not have much in the way of integration capabilities and users are unable to reuse their pre-written scripts. Alternatives for if the postman application were less applicable to my plan would include Katalon Studio and SoapUI.

I would be using the postman application but would avoid using the chrome extension due to its recent deprecation. I would use it due to its help in speeding up development time as well as simplistic testing with the use of templates. Postman would only be used for testing my Express REST API due to its applicability and its generation of code to speed development rather than writing fetch requests and the like entirely manually.

Browser Dev Tools

Google Chrome, Microsoft Edge and Mozilla Firefox are all highly popular and supported web browsers with industry-standard developer tools. I would mostly be testing my applications' UI in

Chrome with its developer tools as Chrome is the leading internet browser with 65.52% of the global market share as of August 2022. Apple's Safari is the second most popular with 18.78% however has little support for developers on windows. Because of this, I would only be able to test manually via a small apple device for safari's compatibility. Edge is the third most popular with a market share of 4.29%, it does support the majority of industry-standard developer tools and would demonstrate the UIs compatibility with the browser when used for testing. Firefox is the fourth most popular browser with a 3.15% global market share and would be the final browser I would be using to test compatibility with. Firefox supports all industry-standard development tools and would be used to check the UI's compatibility with the browser. The compatibility of the web application would be prioritised according to the global market share.

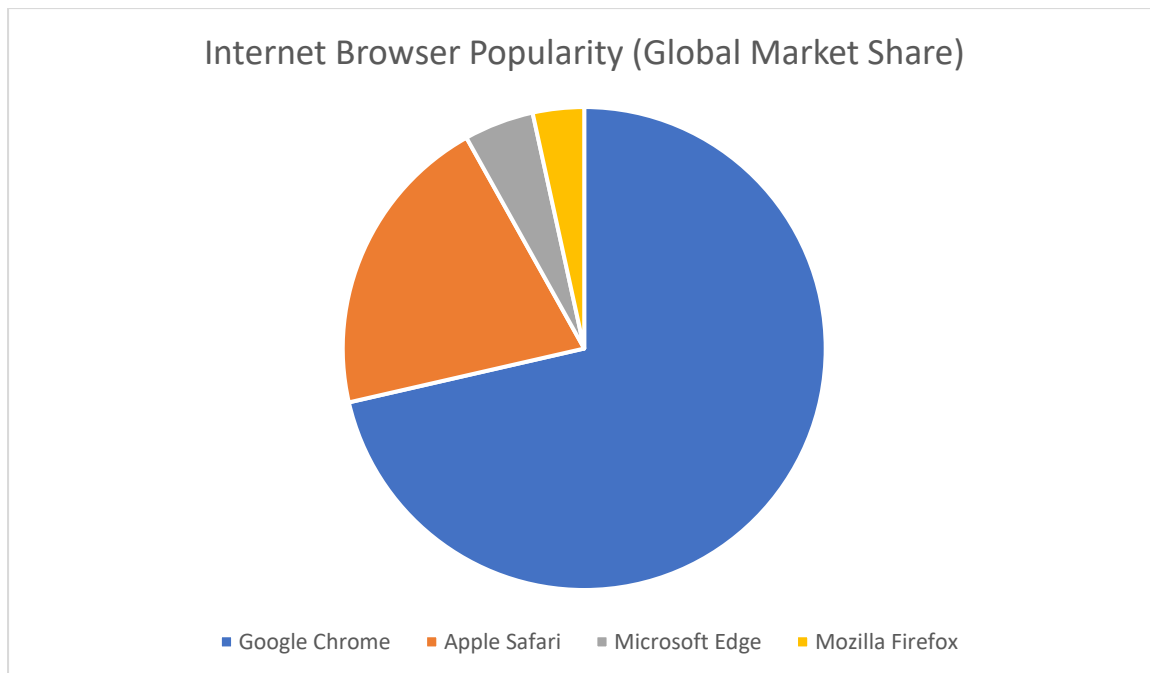


Figure 14: Global market share of Internet Browsers

As well as checking for various browser compatibility, the use of browser developer tools allows me to:

- Test compiled JavaScript and view errors in the console
- View network activity such as requests and responses
- View and edit some source files
- Limit and view performance for testing purposes
- View, create and edit assets in the application settings like cookies and session variables
- Customise keyboard shortcuts for faster testing
- View the browsers analysis of the application's security
- Record performance
- Generate lighthouse reports

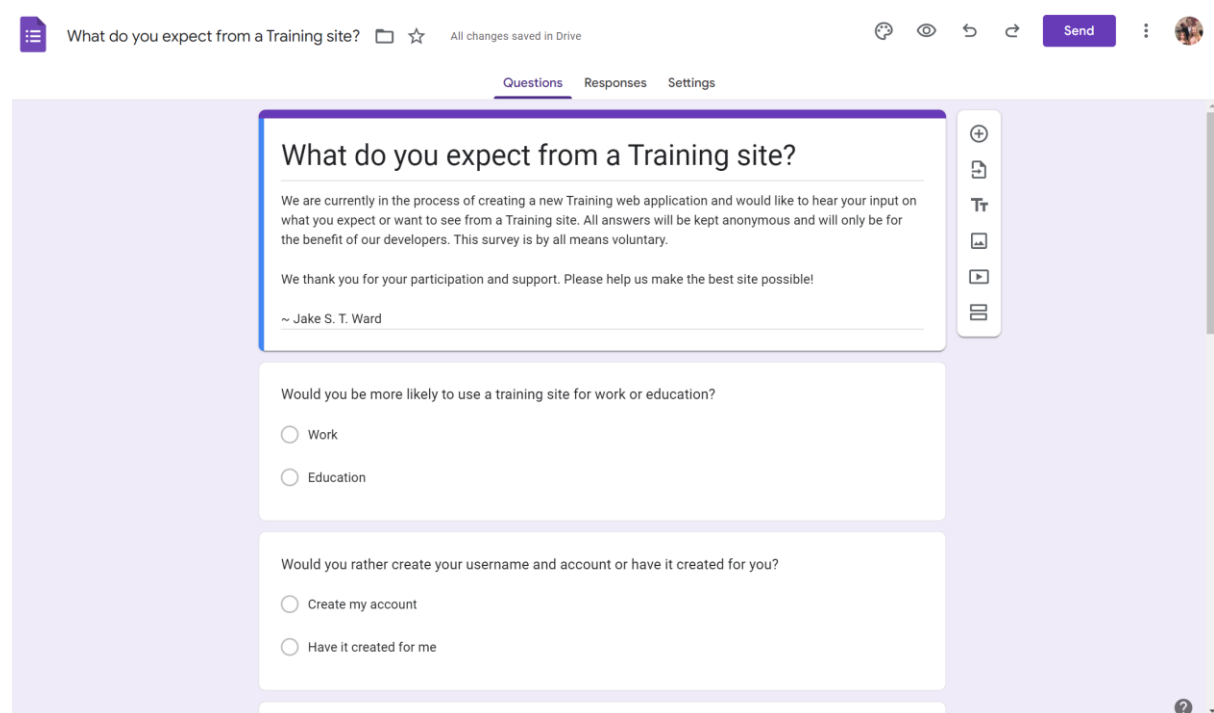
[W3C Markup Validation Service](#)

To maintain accessibility with screen readers and other technology reliant on HTML standards I would be making use of the W3C Markup Validation Service for testing the UI. It is a validator made by the World Wide Web Consortium allowing users to check HTML/XHTML documents for markup against the document type definition by entering its URL, file or script. W3C also provide markup for CSS documents as well as mobile-optimized validation. This would be used for basic error checking in the final stages of the UIs development and would ensure accessibility. By fitting all conventional

standards any extensions reliant on these standards would also be supported by the site without error.

Primary Research

The goal of primary research is to define user characteristics and/or user requirements which will help define other requirements later on. For primary research to be ethically sound there must be anonymity or written consent. While I could have used the consent forms provided by the University I decided that I would reach more potential users and thus get more accurate and varying answers if I instead used an online platform emphasising anonymity like Google Forms. The disadvantage of this method is that anonymity makes the potential to be spammed dramatically increase, this is a manageable concern as spam can simply be removed by the creator upon reflection and reduced by linking the form in more professional circles. If the form became the subject of spam Google Forms also supports the rejection of responses should it be enabled. Another advantage of Google Forms is that questions could be slightly altered or added by myself later on should additional information be required/desired. The questions in the questionnaire would have to be designed so as to make the potential users unidentifiable, because of this there would be a lack of long open-ended questions. The form would be submitted via a link upon permission from an administrator or linked on social media. For ethical reasons, the potential users were allowed to edit or rescind their responses via the same link. I used Google Forms as it is secure, efficient and was used for my group project in 2021-2022 for the CI536 module.



The screenshot shows the Google Forms editor interface. At the top, the title 'What do you expect from a Training site?' is displayed. Below the title, there is a description: 'We are currently in the process of creating a new Training web application and would like to hear your input on what you expect or want to see from a Training site. All answers will be kept anonymous and will only be for the benefit of our developers. This survey is by all means voluntary.' followed by a thank you message: 'We thank you for your participation and support. Please help us make the best site possible!' and the creator's name '~ Jake S. T. Ward'. The main content area contains two multiple-choice questions. The first question is 'Would you be more likely to use a training site for work or education?' with options 'Work' and 'Education'. The second question is 'Would you rather create your username and account or have it created for you?' with options 'Create my account' and 'Have it created for me'. The interface includes a top navigation bar with 'Questions', 'Responses', and 'Settings' tabs, and a right-hand sidebar with various form editing tools.

Figure 15: Google Forms UI creating the survey

Responses

Manage how responses are collected and protected

Collect email addresses



Send responders a copy of their response

Requires **Collect email addresses**

Off



Allow response editing

Responses can be changed after being submitted



Figure 16: Google Forms survey settings

The form is no longer available for ethical reasons as I cannot guarantee the safe storage of the data recorded if it was stored on my Google account. The data was then removed from the form and stored securely on the university OneDrive.

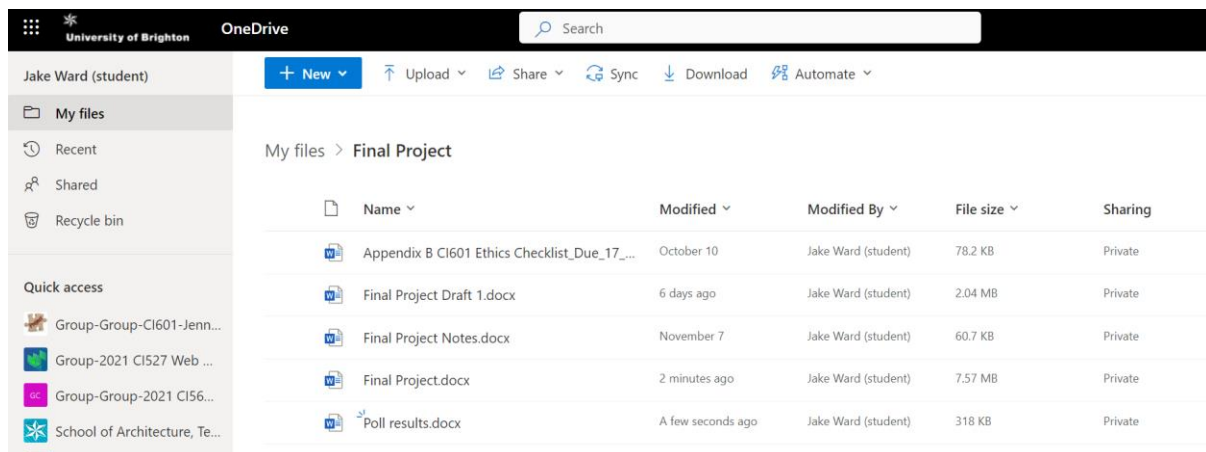


Figure 17: Storage location of gathered research data

The results of the questionnaires multiple-choice questions are as follows:

Would you be more likely to use a training site for work or education?

41 responses

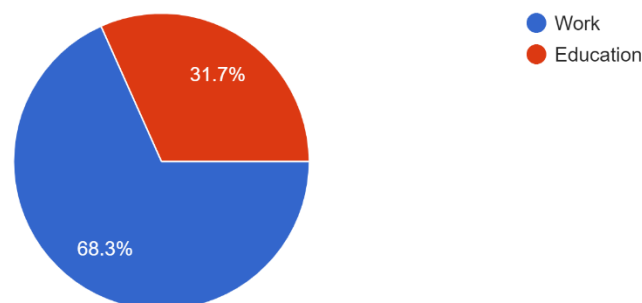


Figure 18: Survey question 1 results

Would you rather create your username and account or have it created for you?

41 responses

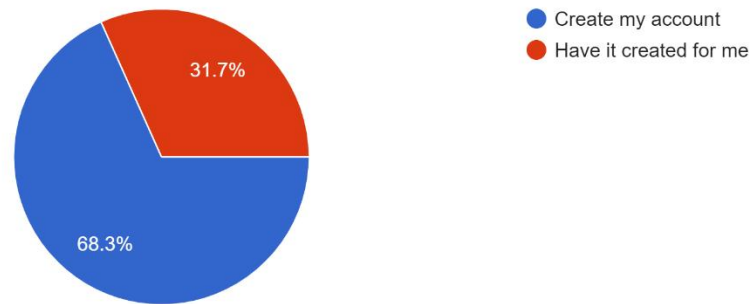


Figure 19: Survey question 2 results

Do you prefer watching training videos or reading articles?

41 responses

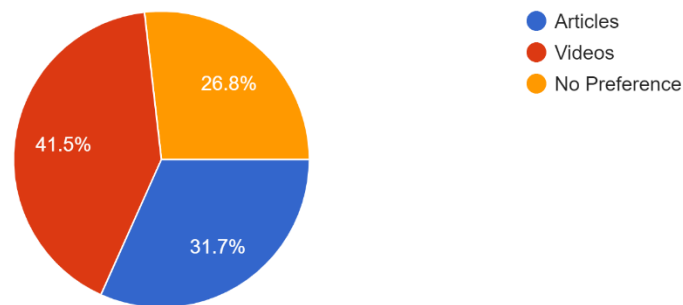


Figure 20: Survey question 3 results

Do you prefer a casual vibrant interface or a professional monotone interface? If neither please explain your preferred design.

41 responses

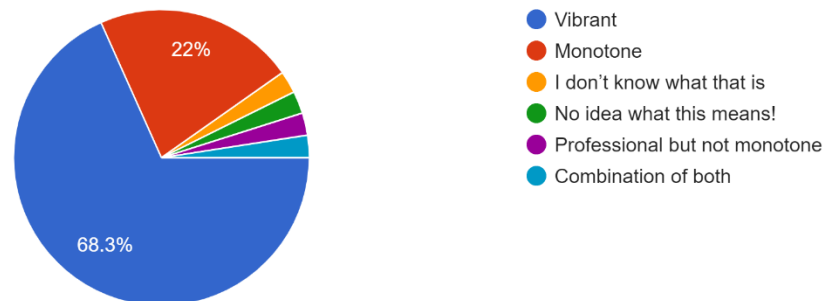


Figure 21: Survey question 4 results

Would you appreciate seeing your progress visually represented?

41 responses

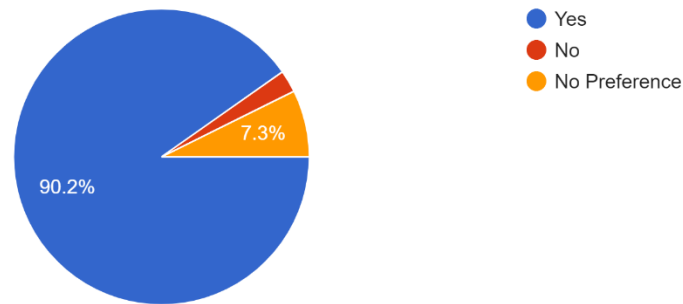


Figure 22: Survey question 5 results

Would you like to be emailed by the site automatically?

41 responses

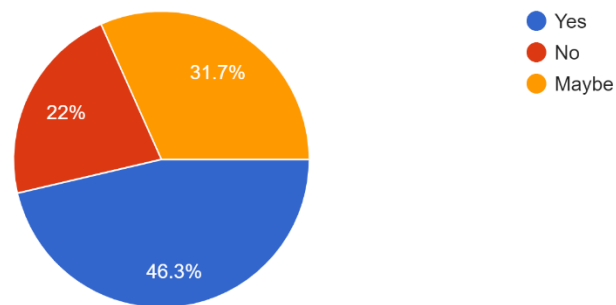


Figure 23: Survey question 6 results

Would you prefer the availability of profile pictures or a lack-there-of

41 responses

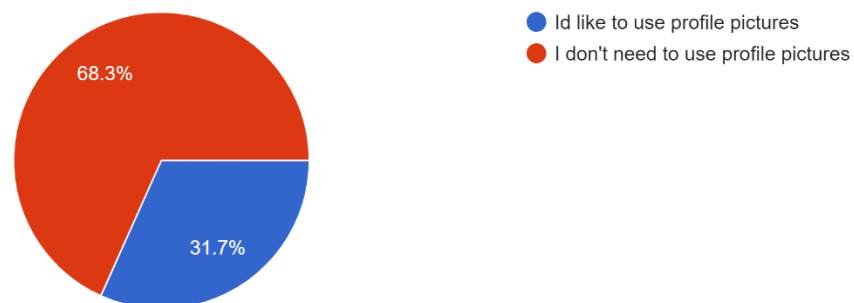


Figure 24: Survey question 7 results

Would you like to be notified about new, upcoming or training due via Email or on the website?

36 responses

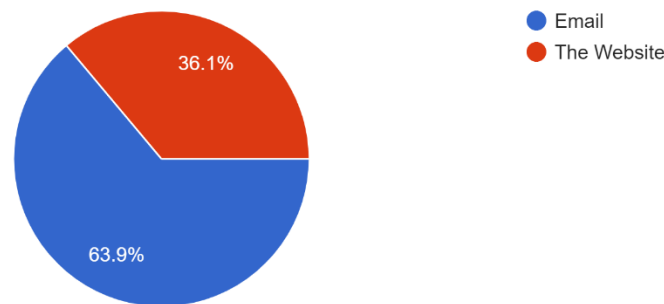


Figure 25: Survey question 8 results

The final optional open-ended questions results are as follows:

- Id like a site of this kind to notify me if some training is due soon
- Nah
- Helpful stuff
- Different colour backgrounds as a setting for people who prefer to read on a for example green background if dyslexic and dark mode . Also option to change the size of font
- Be able to record it for later reference
- No
- Make it easy/inexperienced pc user friendly The only reason I would be using it is because it was a mandatory work requirement.
- If it's a subject that I'm genuinely interested in further reading would be good but appreciate there are rules around sharing other organisations info.
- Dates and duration of training and who aimed at, email contact if unable to make.
- I like interactive graphics. I learn better by engaging with something that I have to read or short animated clips diagraming a process. People talking is the worst for online training.
- Saving part way through and a test at the end
- If appropriate for the type of system you were training on it would be good to be able to key in sample data so it can be followed through the system, including creating deliberate misposts and errors in order to learn how to correct them. If it was not that type of training then it needs to be engaging and progress through the training needs to be recordable.
- No thanks
- Yes, testimonials about the courses and trainer profiles of online training.
- Email confirmation of completed training in order to save
- A sandbox-type section for applying learning. For example, if learning about for loops in coding, having somewhere to create a for loop as part of your progress (see codewars or pluralsight for examples)
- Score table

- People learn in different ways - reading, and from videos with a practical representation. Using a combination of the two would be great - maybe videos with subtitles and a transcript to refer back to, and a summary. Colouration has an impact as to how much information we absorb, so worth looking in to that.

There was no initial desired target audience as training web applications can be used by varying groups of people from students to employees. It was by publishing the form that I would be able to refine my user characteristics. Any potentially identifying features about participants I was aware of cannot be featured in the report due to ethical issues. Overall, I can say that the target audience varies highly in age but consists mostly of employees rather than students in full-time education.

The main potential additional requirement gained from primary research was the notification recommendation. Potential users who used the open-ended question to make suggestions first suggested that they would appreciate being notified about training. Because potential users suggested they had no preference on whether they were emailed by a training site I added an additional question to clarify which approach to take. Once the wireframe models were finished the form or a new form would have the addition of these potential UIs and allow potential users to pick a preferred one.

After the form was taken down and its results stored securely, I analysed the final results to decide on requirements and their priority. The appropriate listed changes were made as a result.

- The first conclusion I drew was that the train site's audience consisted mostly of employed adults. It was for this reason that I would design the application to be specific to a company. I would not attach myself to a given company but would instead design the template/architecture for this type of training application.
- The second conclusion was that users would prefer to set up accounts themselves to allow more control over their details. While some preferred the ease of access of having details made for them by an administrator, I would now be supporting a signup feature as well as a login system.
- Overall, there was a lack of consensus on whether Articles or Videos would be better for training, so both would be supported equally and videos would potentially have extra features like the change in video quality found on YouTube.
- I found that the majority of users preferred to have an eye-catching/colourful design as opposed to a conventionally professional design. This means when I came to designing the UI I would use a more laid-back modern design as opposed to a rigid more conventionally professional design. The corresponding question had an alternative option to leave another type of design but few users used this feature.
- The overwhelming majority of users favoured having a visual representation of their progress leading to a progress bar or the like being given high priority in the requirements.
- The majority of people stated that they were not opposed to being emailed by the site after they had been signed up so the use of email would be listed in the requirements and emails would be stored in the user's table of the back-end database.

- Most people were surprisingly uninterested in the use of profile pictures. This was surprising due to their conventional use online. This result didn't lead to the delisting of the profile picture requirement but did significantly reduce its priority.
- Many potential users stated that they would rather be notified by the website via email as opposed to on the UI. This meant that if a notification system were to be put in place because of due dates then the notifications would be handled over email.
- The use of themes like dark mode was suggested for accessibility as well as change in font size, this was considered and added as a high-priority non-key requirement as it would require little effort in the form of CSS changes.
- User-friendliness was mentioned which would have always been a priority for design.
- Further reading was suggested as a potential feature and so this would be a feature potentially added to the articles. This would not require the changing of requirements but would factor into how the example articles were designed. Contact information for training would also come under this category.
- Interactive graphics were another CSS-based element that was suggested and would be included alongside the other small visual elements. Alongside this suggestion was the emphasis on reading being better for learning which factored into both videos and articles being supported.
- Saving states of a page were a good suggestion which would be added potentially due to my understanding of saving information like this in internal storage or as cookies. This was added as a low-priority requirement due to its lack of suggestion but useful nature.
- Tests at the end of articles were suggested which reinforced their inclusion in the project and raised their priority.
- Testimonials and author profiles were suggested which would be a feature potentially added to articles upon their design.
- Testing was brought up which would be followed up by using the resources listed in the testing section of the research.
- A very advanced feature was suggested in the form of a sandbox section for testing coding should the training involve that. This would not be added to requirements yet but would be added if more features were required to add complexity. It was not added as a requirement as after researching how to accomplish this goal it seemed to be too time-consuming to add it in time for the due date.
- Results were suggested to be visible which would be covered by the "stats" requirement.
- It was reinforced by the final suggestion that the combination of both language and video would be the most effective tool for learning/teaching.

Secondary Research

For the most accurate secondary research, I studied the layout of my employer's training site however because of legal and ethical reasons I cannot display any screenshots that disclose any identifying information in my report.

As a substitute, I also analysed and annotated a similar type of web application which is public. While the type of training I was going to make would be exclusive to one company there are sites that allow the purchase of training for education purposes. This is the closest type of site I can display analysis of in the report without obscuring features as any site exclusive to a company is said company's intellectual property. I analysed the UIs of all my chosen sites in desktop and mobile views for a full grasp of the designs and features.

First, I analysed the signup method for new employees. The method involved emailing the employee with a temporary password to be used alongside their email to log in. This implies that accounts were already made for employees. This is a good security measure to ensure only employees have access to the site. This is an approach I could have taken but due to my survey finding that most users would prefer to make their own accounts, I decided that a similar method would be to have them use a separate code similar to a temporary password that would be used as validation for account setups. All defining features that could disclose the site's identity have been obscured in the examples below.

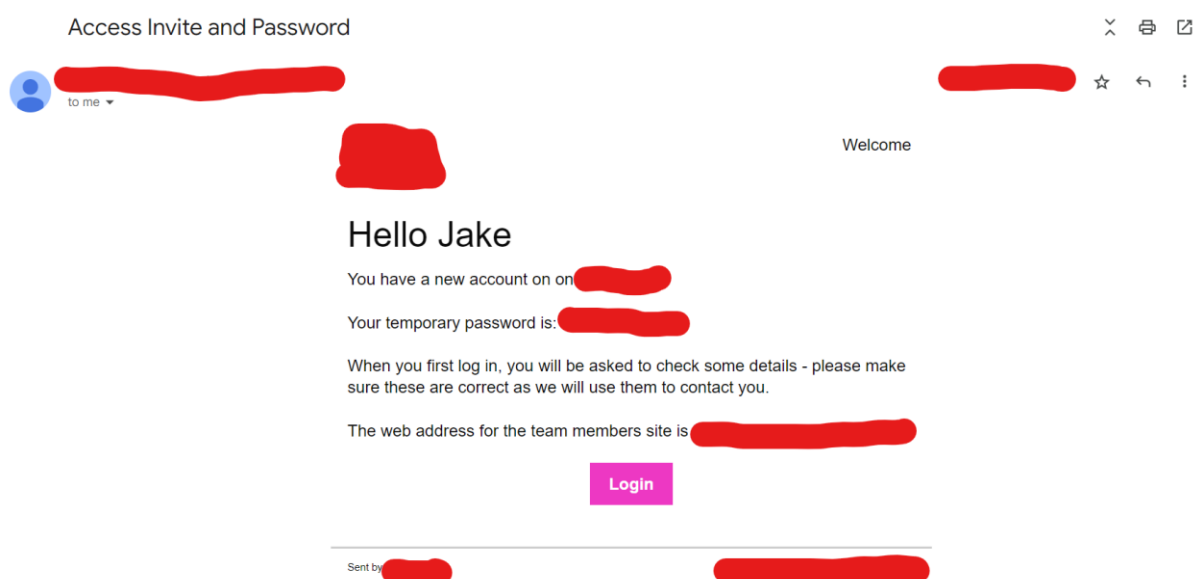


Figure 26: Invite to analysed site

The login page of the site resizes to look the same on both desktop and mobile views. It gives errors when a failed login occurs to help the user understand why they haven't been given access, this would be adopted by my site as it is a convention of login pages integral to the user experience. The user logs in with their email and password, but I decided I could make use of usernames for login as well. Upon an unsuccessful login attempt, the error message allows the user to reset their password, in this case, their password is set to a temporary password again and emailed to the user, allowing them to log in and change their password. In my case, I would address this feature similarly by sending the current validation code to their email and redirecting them to a part of the site allowing them to directly change their password. If an email doesn't suffice a second option could be the use of texting, however, this would involve the use of a budget to which I do not have access. The site has a feature to remember login details which could be saved in local storage or as a session or

cookie value. The page is colourful with a large logo which could be a style adopted in line with the results from the survey. Passwords are hidden upon typing them in but a button to reveal what's been written is missing, this is a feature I would use to raise security but mitigate the amount of accidentally failed logins

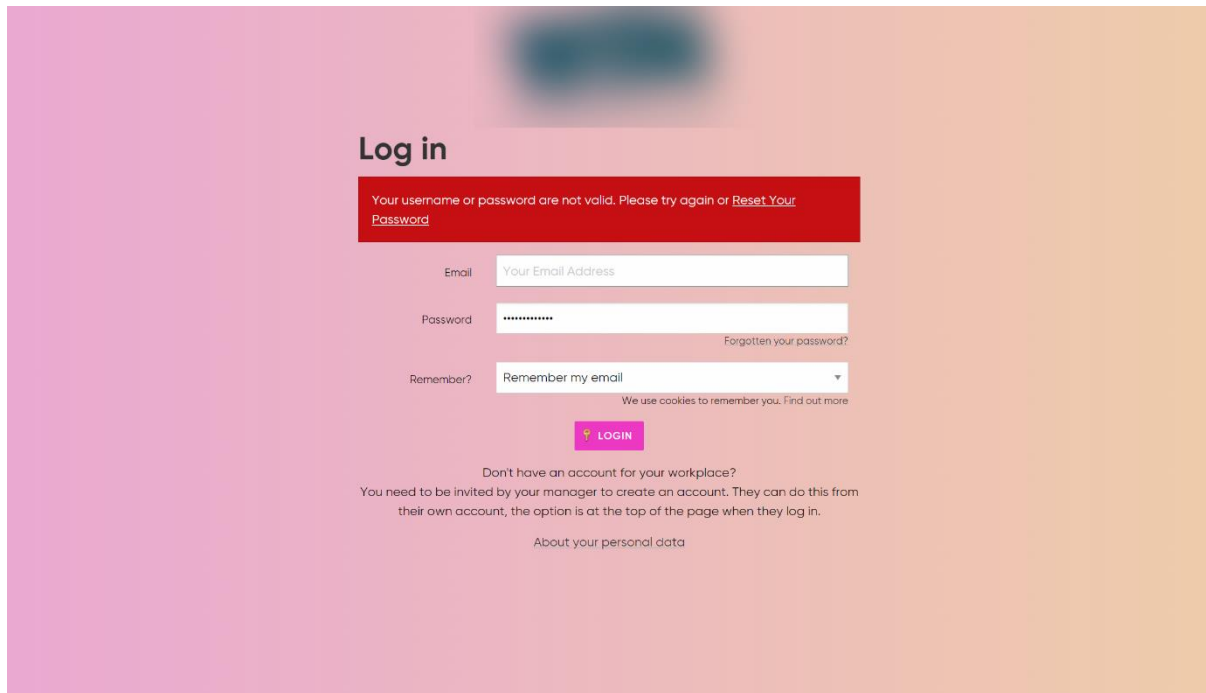


Figure 27: Login page of the analysed site

Upon successful login, the page you are directed to is the dashboard. It features a dropdown top navigation bar on desktop views and a side navigation bar on mobile views. The user's profile picture and username are featured in the top left corner of the page above the navigation beside the logo. The navigation also features a search bar (which I would integrate to search through assigned training) as well as the employee's job title. The page is a PHP page meaning that it is a dynamic page rendered with details from the database, this is an alternate option for dynamic pages on my own site but I will attempt to integrate it with my API so users won't have access to the database credentials.

Notifications are kept as separate boxes/elements below the page title, these are notifications for mandatory training and missing profile information. I will make use of these elements but will only notify users about training with upcoming due dates as I would assume that all training assigned would be mandatory and so would overwhelm the dashboard. The dashboard's two columns of elements merge into one on mobile views. News is featured on the dashboard which I could integrate but I decided that it'd be better to introduce a similar window with the due dates of assigned courses instead. Information about shifts is present on the dashboard but this isn't relevant to my application as it is designed for training only. Current courses you are signed up for as well as certificates earned or progress towards them are all featured on the dashboard, these could be inclusions to my homepage/dashboard but would be more integral to a page dedicated to assigned training. The site maintains its colourful design to attempt to remain visually interesting. There is only one CSS animation which is the extension of the sidebar on mobile view. All of these elements would be considered when designing the homepage of my application.

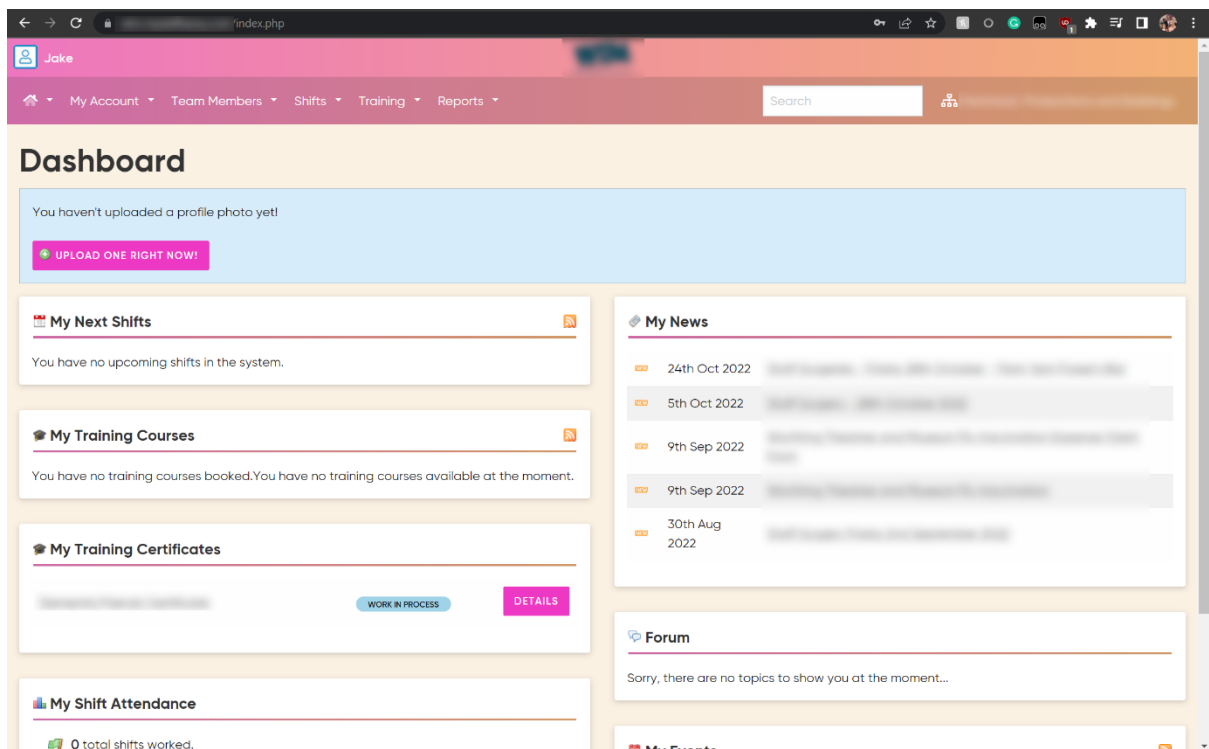


Figure 28: Dashboard/Homepage of the analysed site (desktop view)

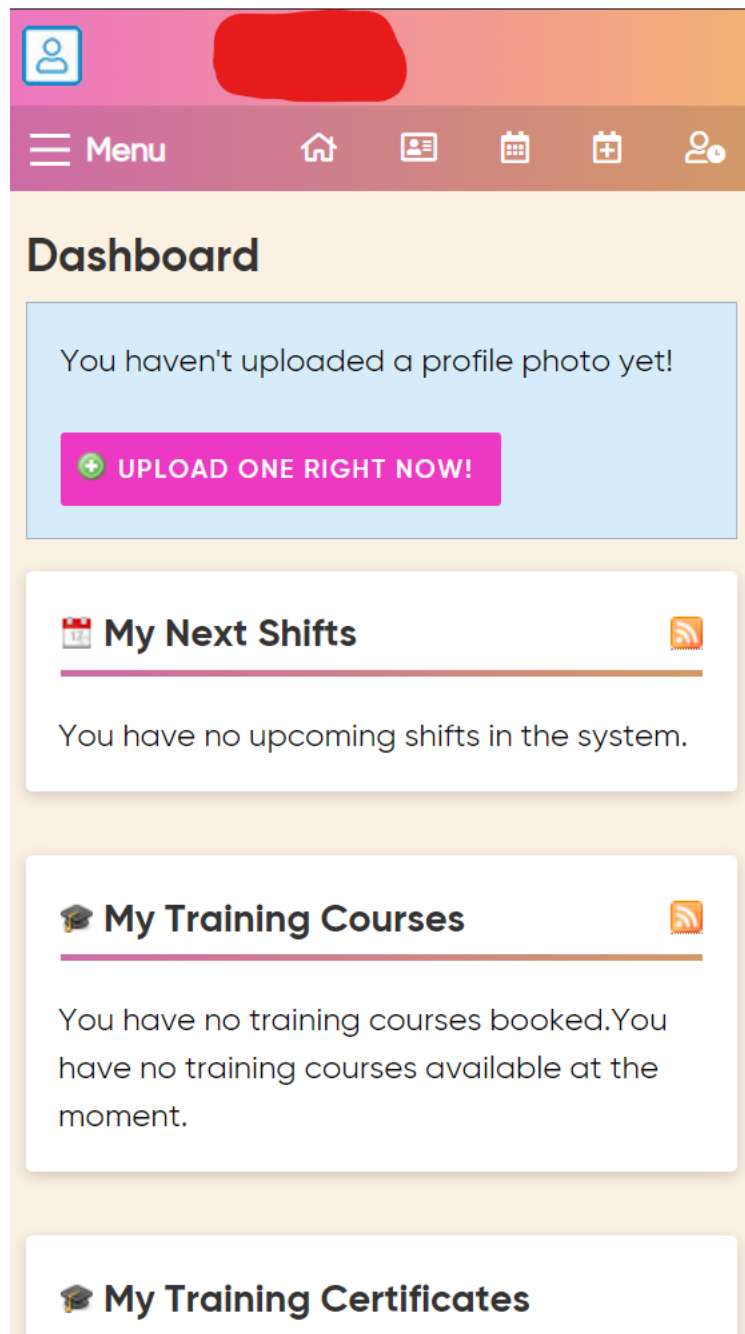


Figure 29: Dashboard/Homepage of the analysed site (mobile view)

The training pages were separated into the employee's status, assigned courses, resources and forums. While I wouldn't be planning on adding any forms a resource library would be a possible section to consider. The resource library isn't unique to any one user but is instead a list of all resources used for training, this could be implemented along with a search bar for navigation to aid those who want further reading. The employee status could be where statistics are displayed as well as results from training. The courses page itself would feature all assigned courses as well as past courses, this can be used to navigate to the training in need of completion and revision of old work. There is a copyright statement at the bottom of the page which I would integrate with an automatically updating year.

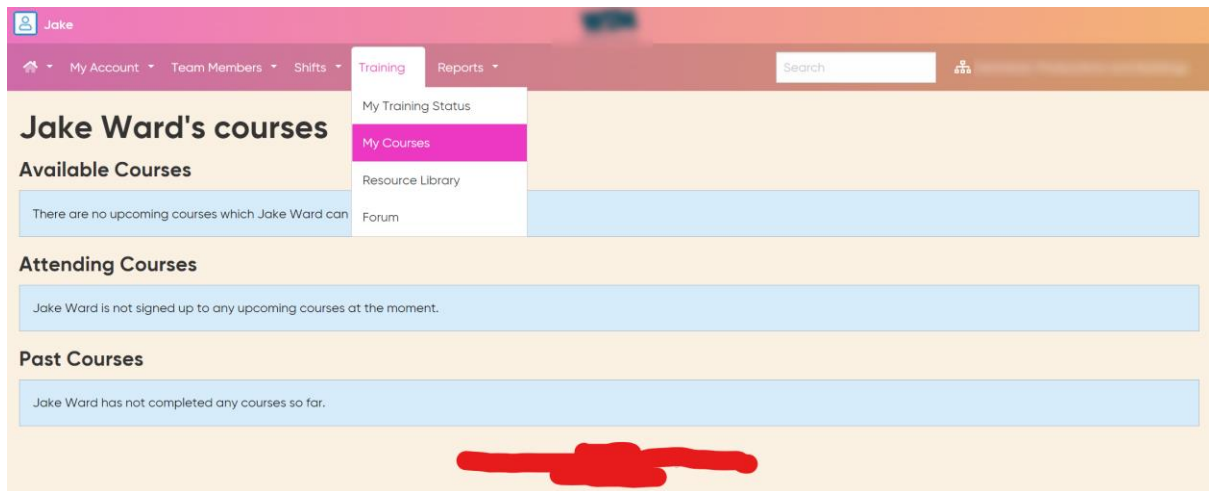


Figure 30: Training section of the analysed site (desktop view)

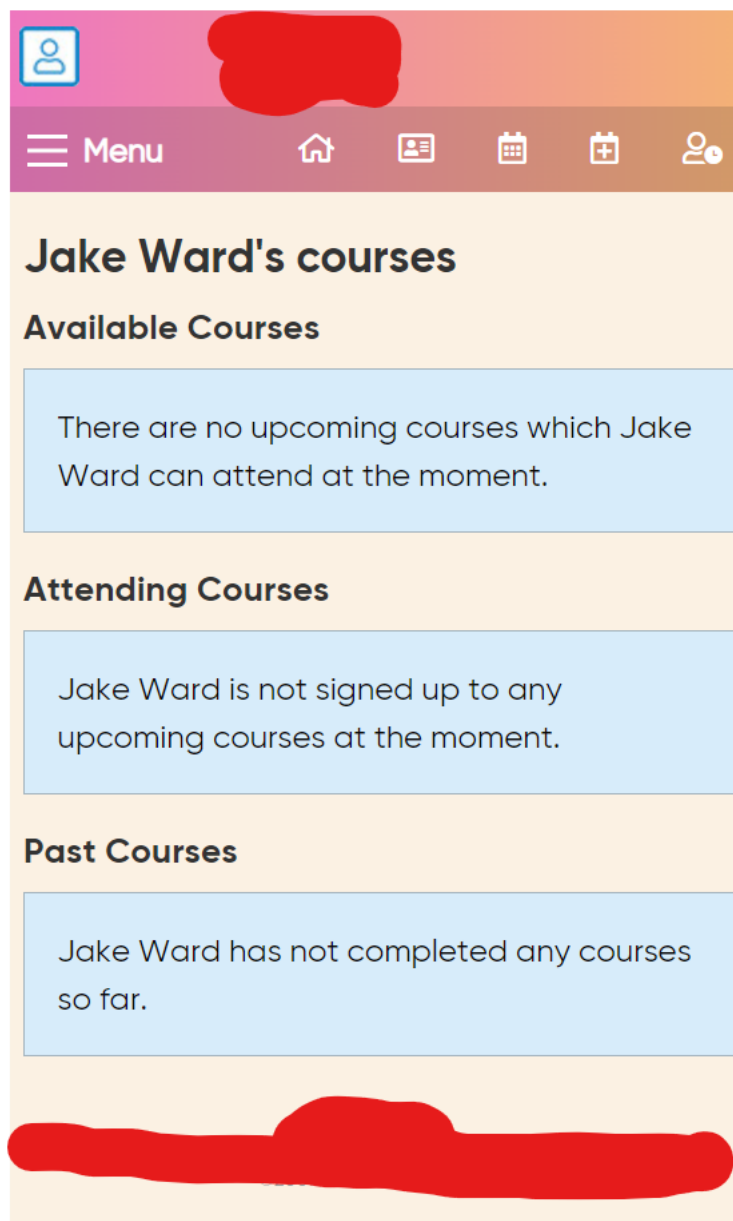


Figure 31: Training section of the analysed site (mobile view)

The navigation opens and extends sideways in both views. There are options to update information about the profile from the navigation as opposed to navigation via the profile page. The use of quick links for every part of the application would be a good inclusion to enhance the user experience. The mobile view introduced a phone-oriented element on mobile view in the form of the lock screen button, this could be another simple extra inclusion in my application. The mobile view also incorporates icons in place of links to take up less space and be more readable on smaller devices.

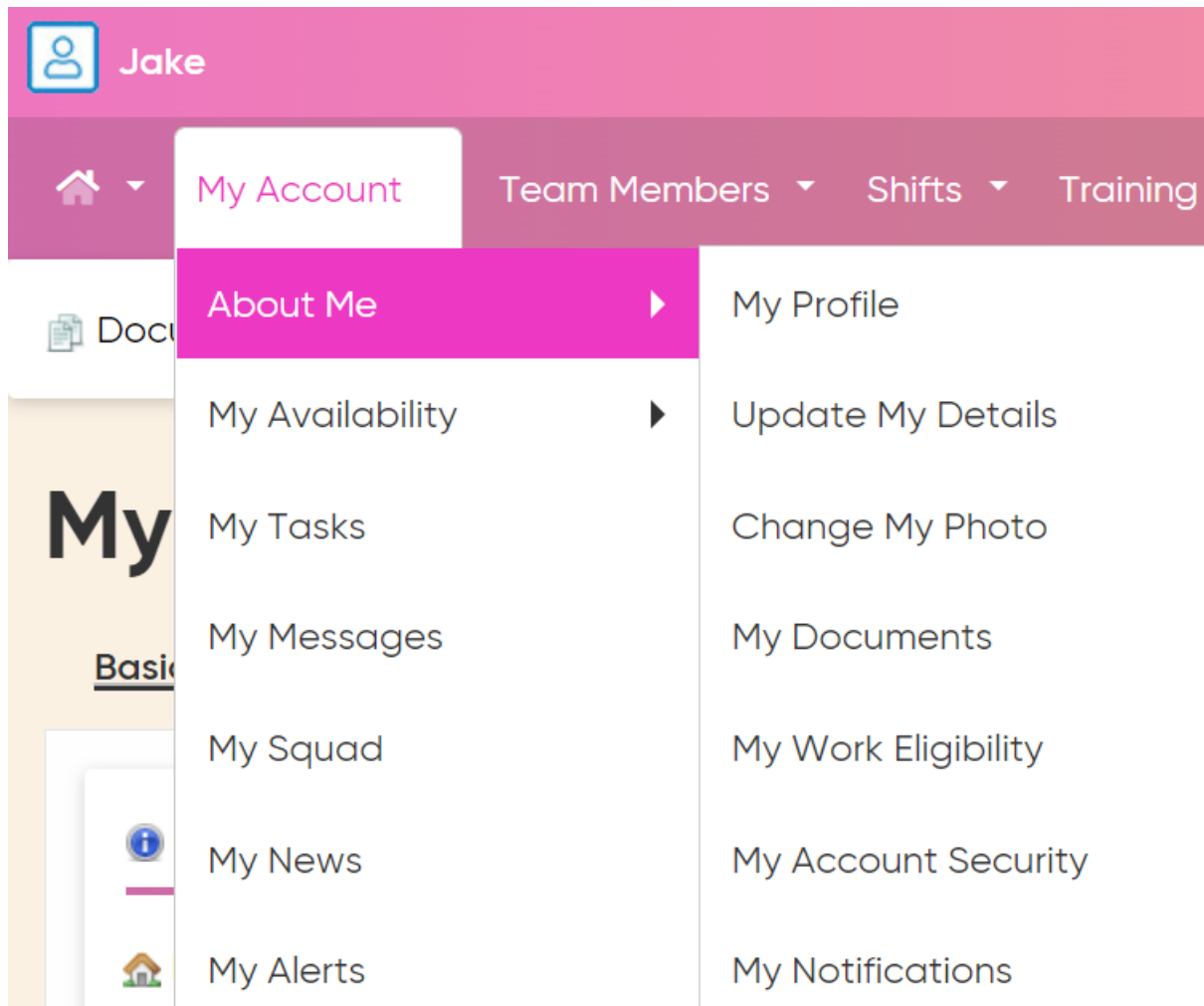


Figure 32: Navigation bar of the analysed site (desktop view)

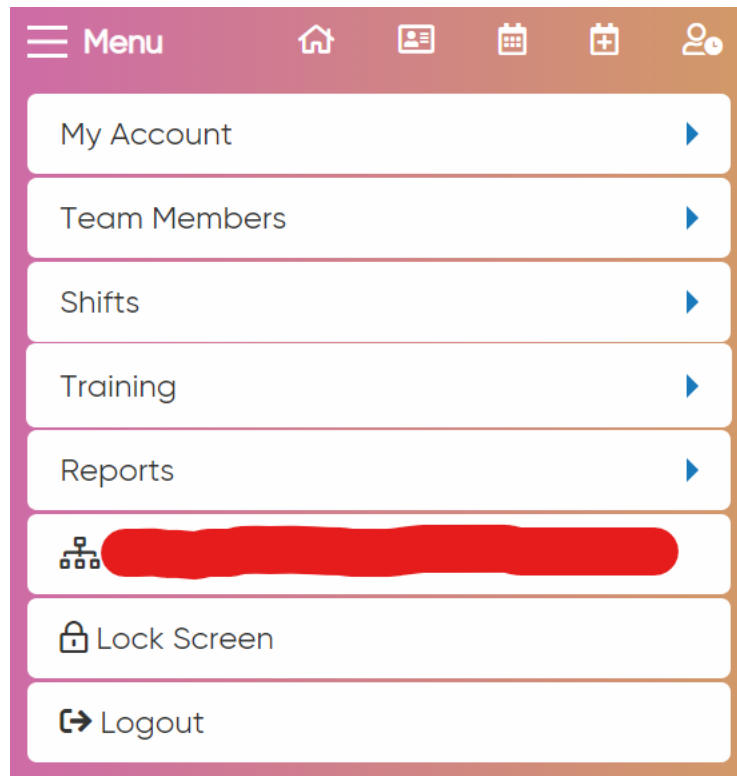


Figure 33: Navigation bar of the analysed site (mobile view)

The profile page itself is also a dynamic page generated with the use of PHP. It houses information about the user taken from the database separated into three sections: basic information about the employee, contact information including an email and phone number and a customizable 'About Me' section. All of these can be edited but some fields are mandatory. Profile pictures and the 'About Me' section are optional. All profile elements are edited from a separate page and are altered upon clicking a button labelled 'save'. There are again other extra details exclusive to work that I would not be incorporating into my site due to lack of applicability, like the ability to see documents submitted for work and reporting an absence. The extra navigation on mobile view takes up an unnecessary amount of space which I would avoid by using a side-scrolling element to ensure they only took up one line of space.

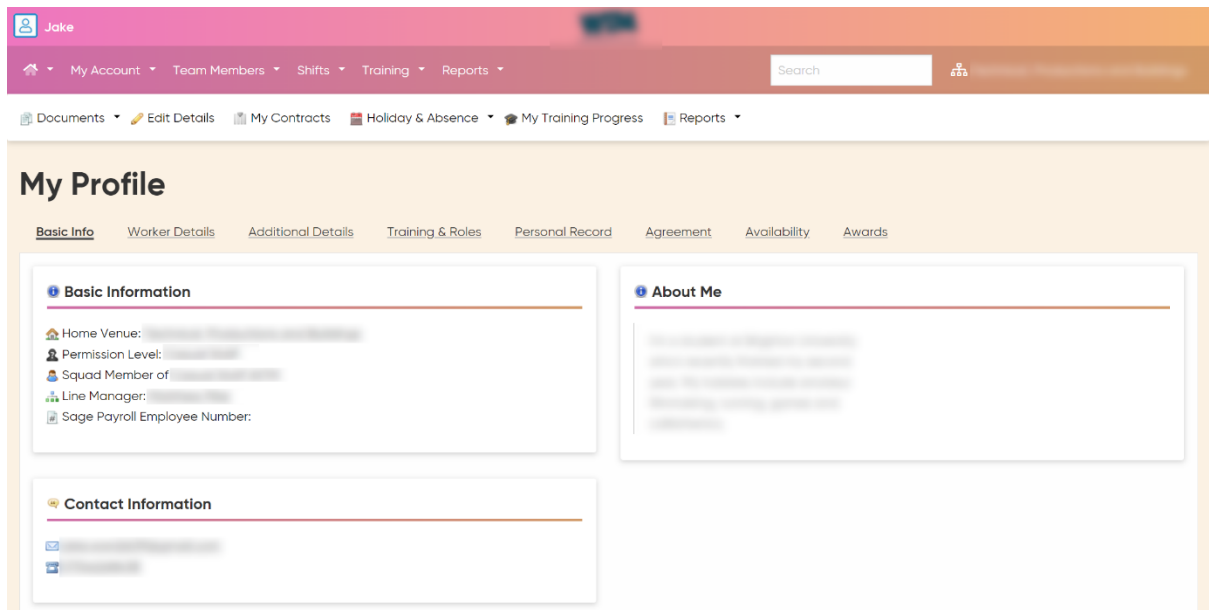


Figure 34: Profile page of the analysed site (desktop view)

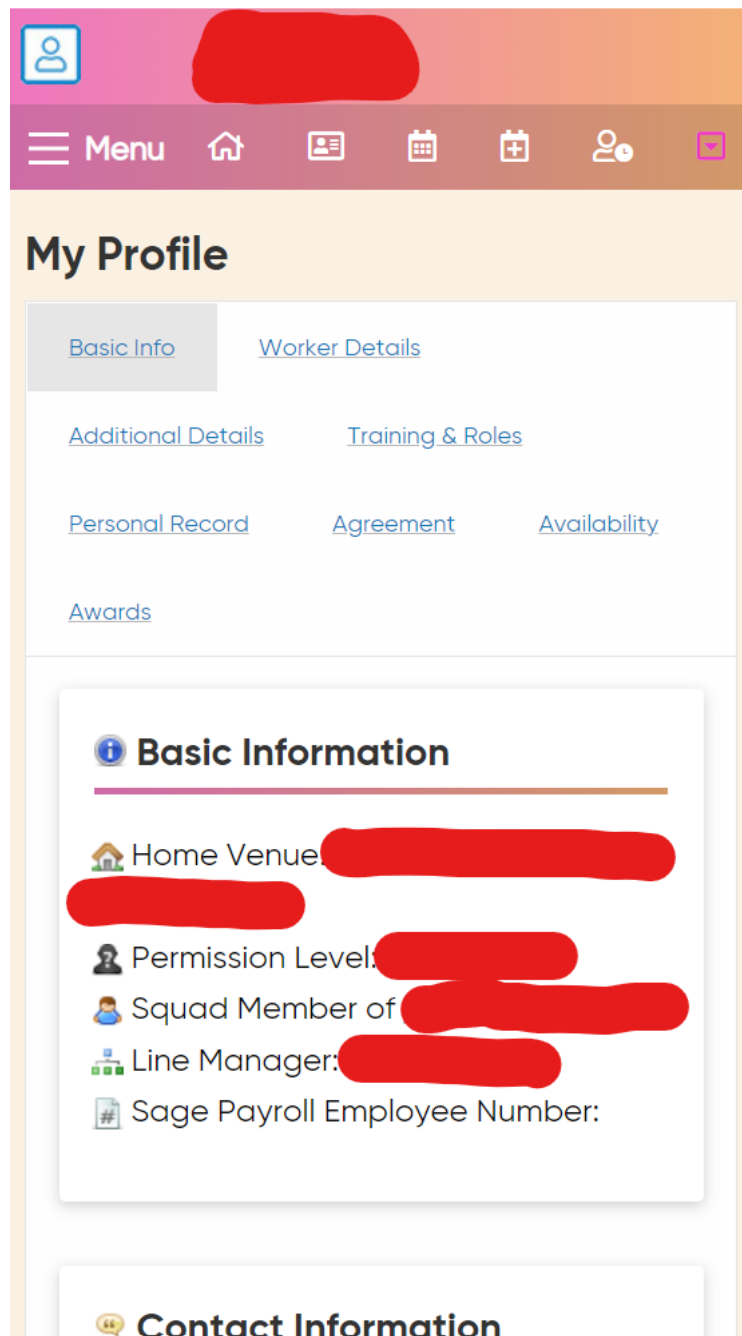


Figure 35: Profile page of the analysed site (mobile view)

All elements discussed would be considered upon the design of the site via wireframe models. Below can be found examples of the other sites which I've analyzed and annotated. These sites were researched for possible alternate designs. No extensive revisions were made due to these analyses, but these inclusions are evidence of the time put into research as well as the conventional design I would be adopting.

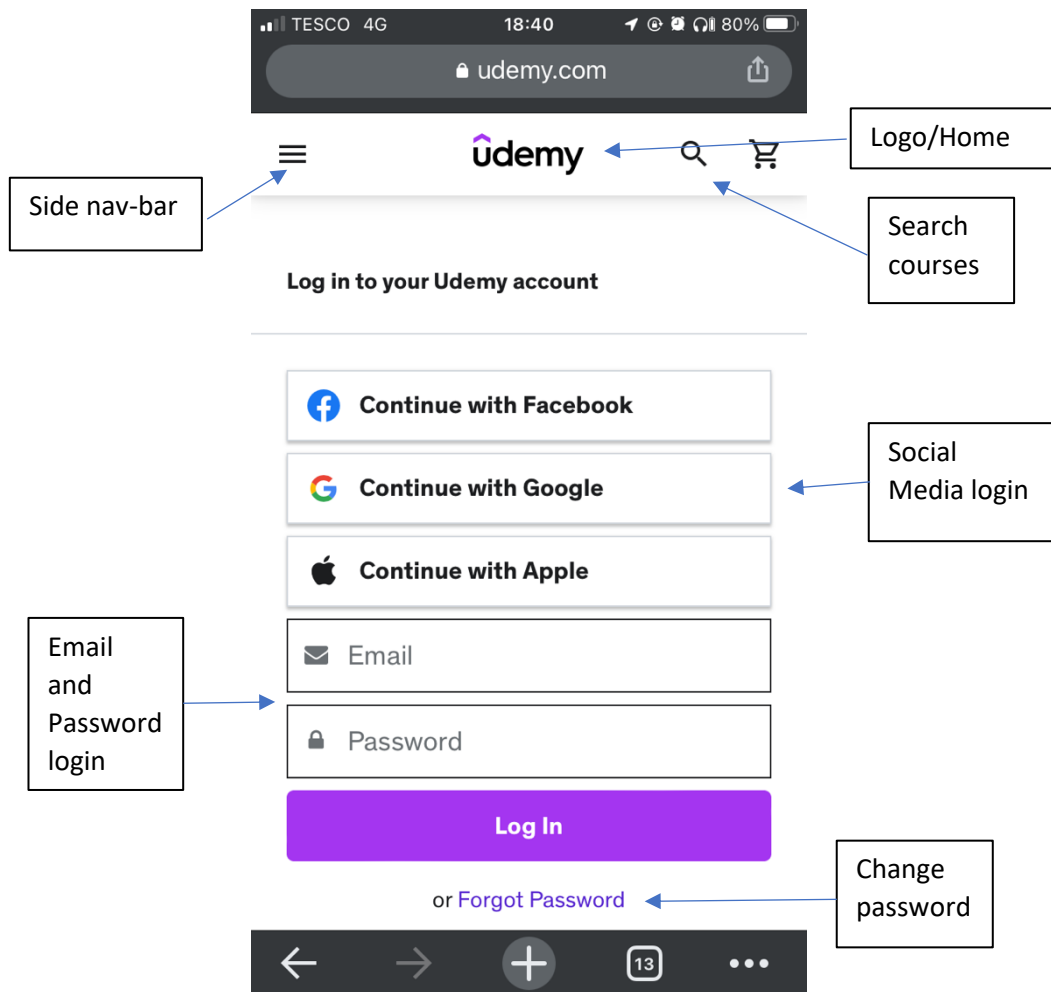


Figure 36: Annotation of Udemy login page

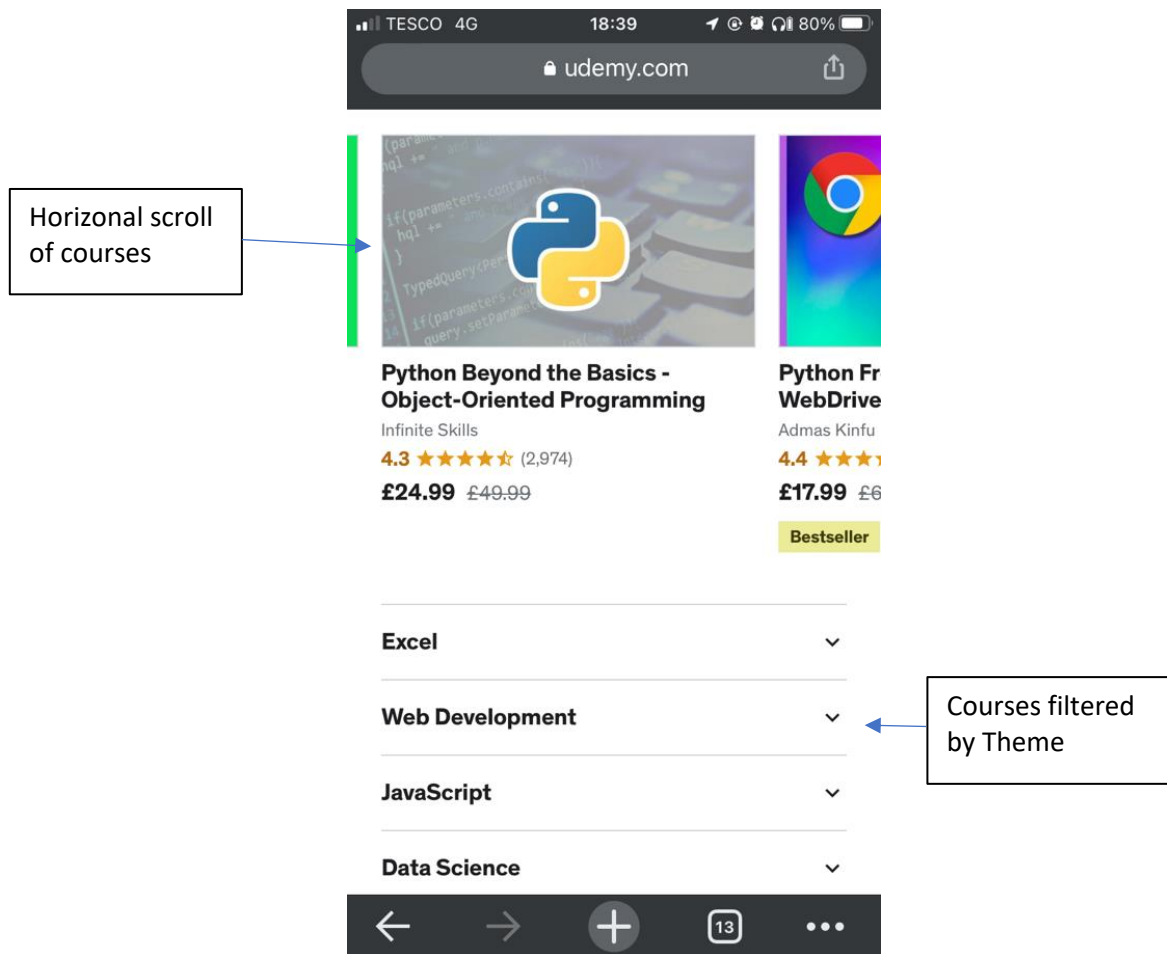


Figure 37: Annotation of Udemy homepage

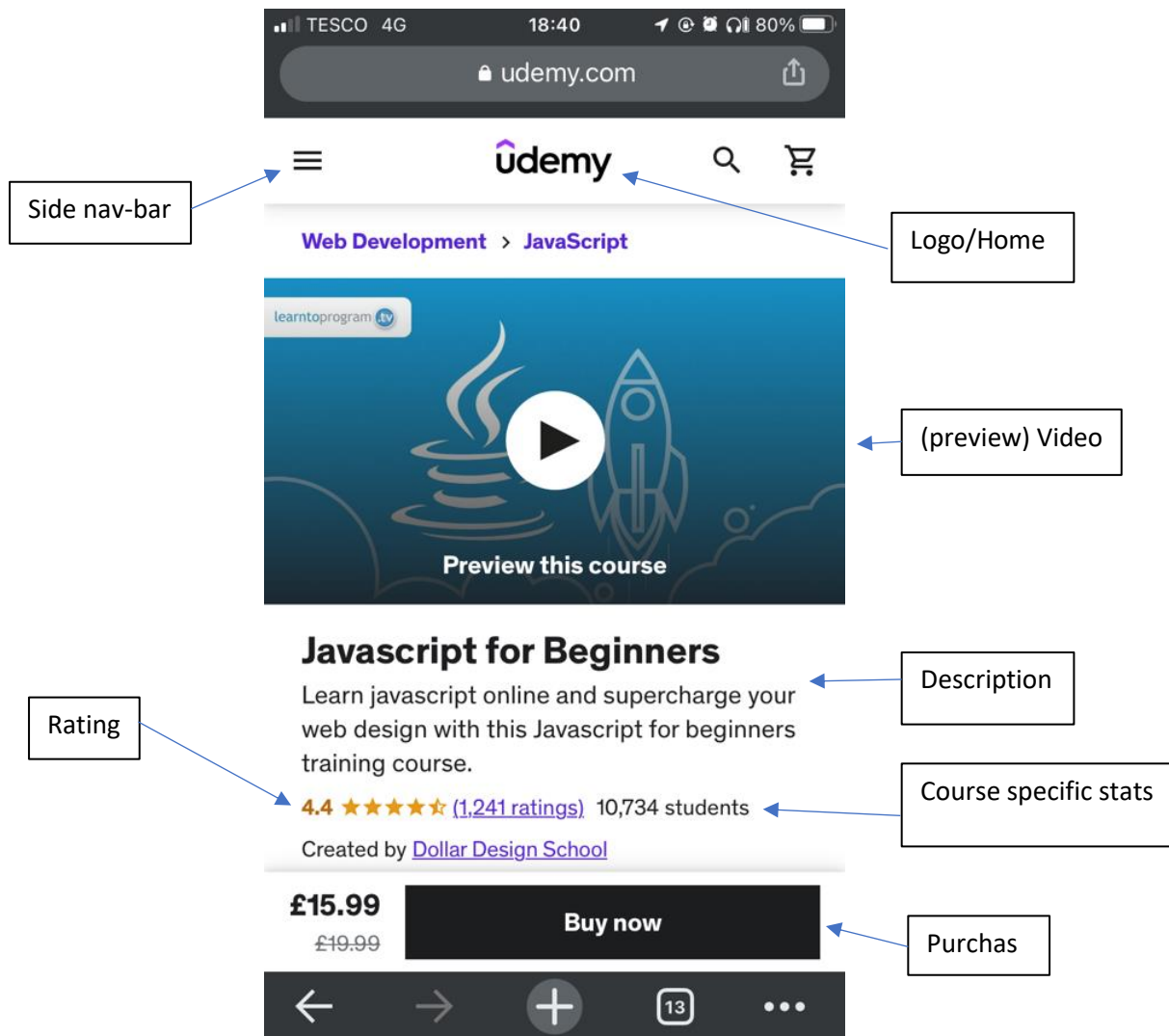


Figure 38: Annotation of Udemy course-specific page

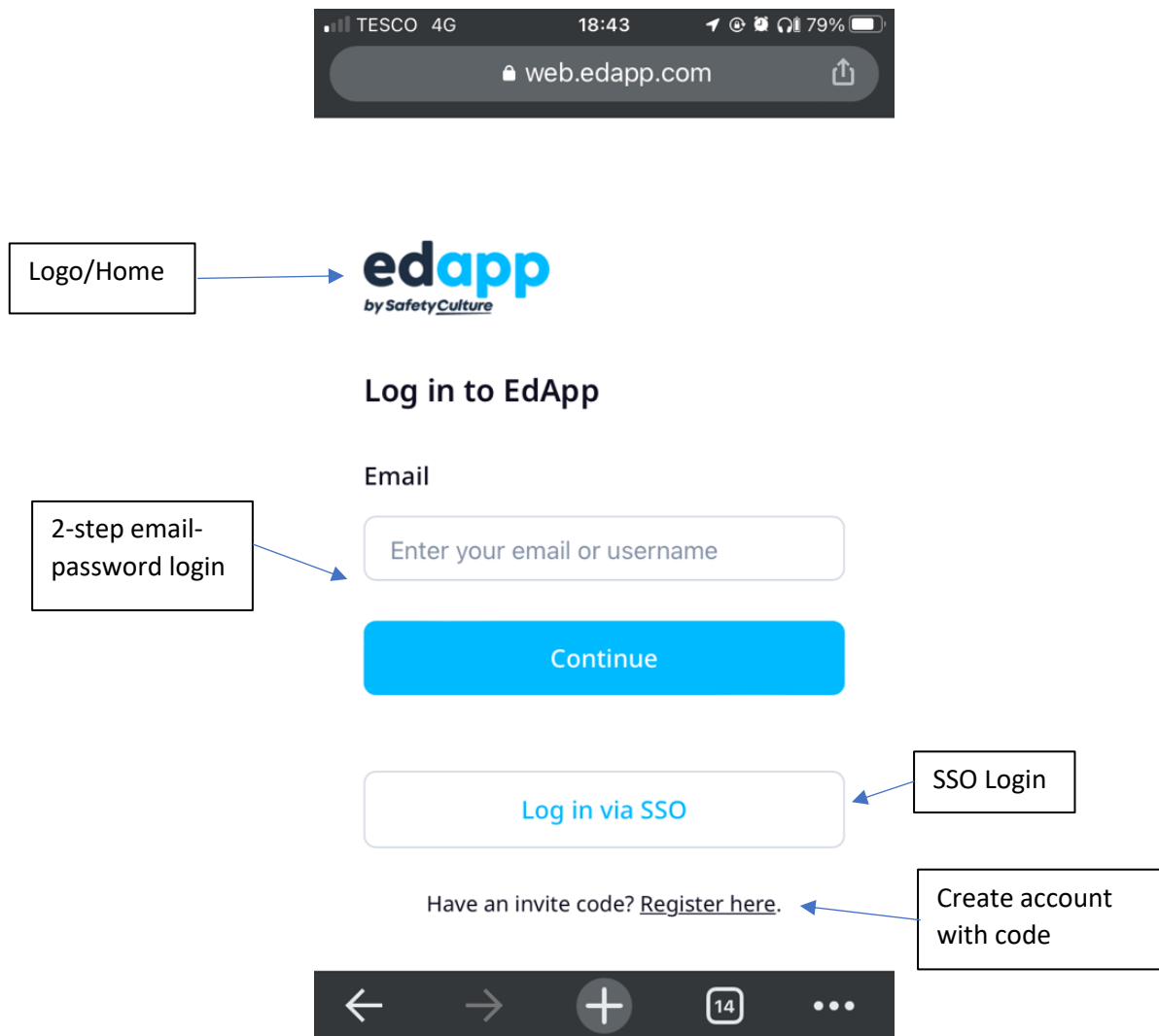


Figure 39: Annotation of Edapp login page

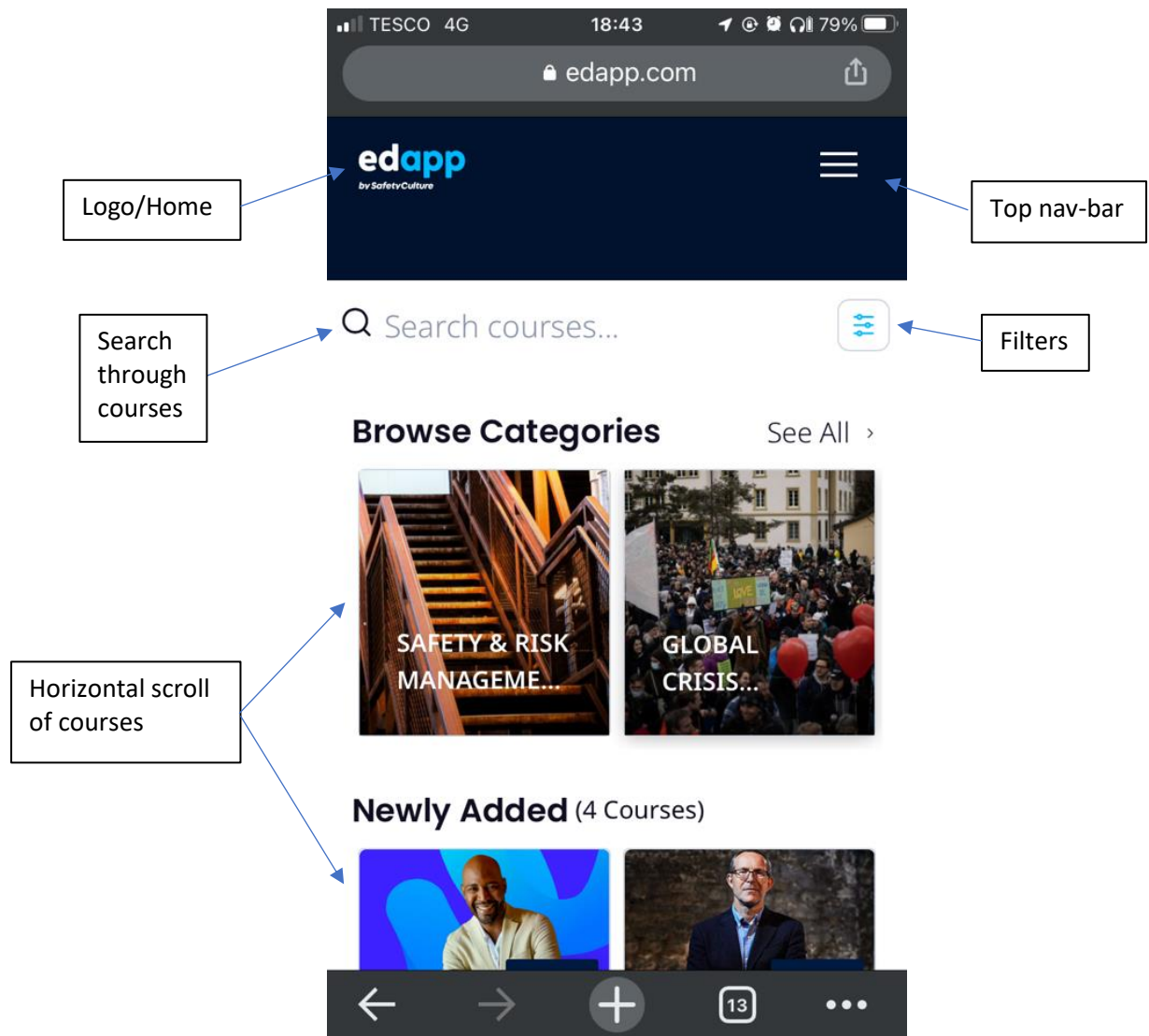


Figure 40: Annotation of Edapp homepage

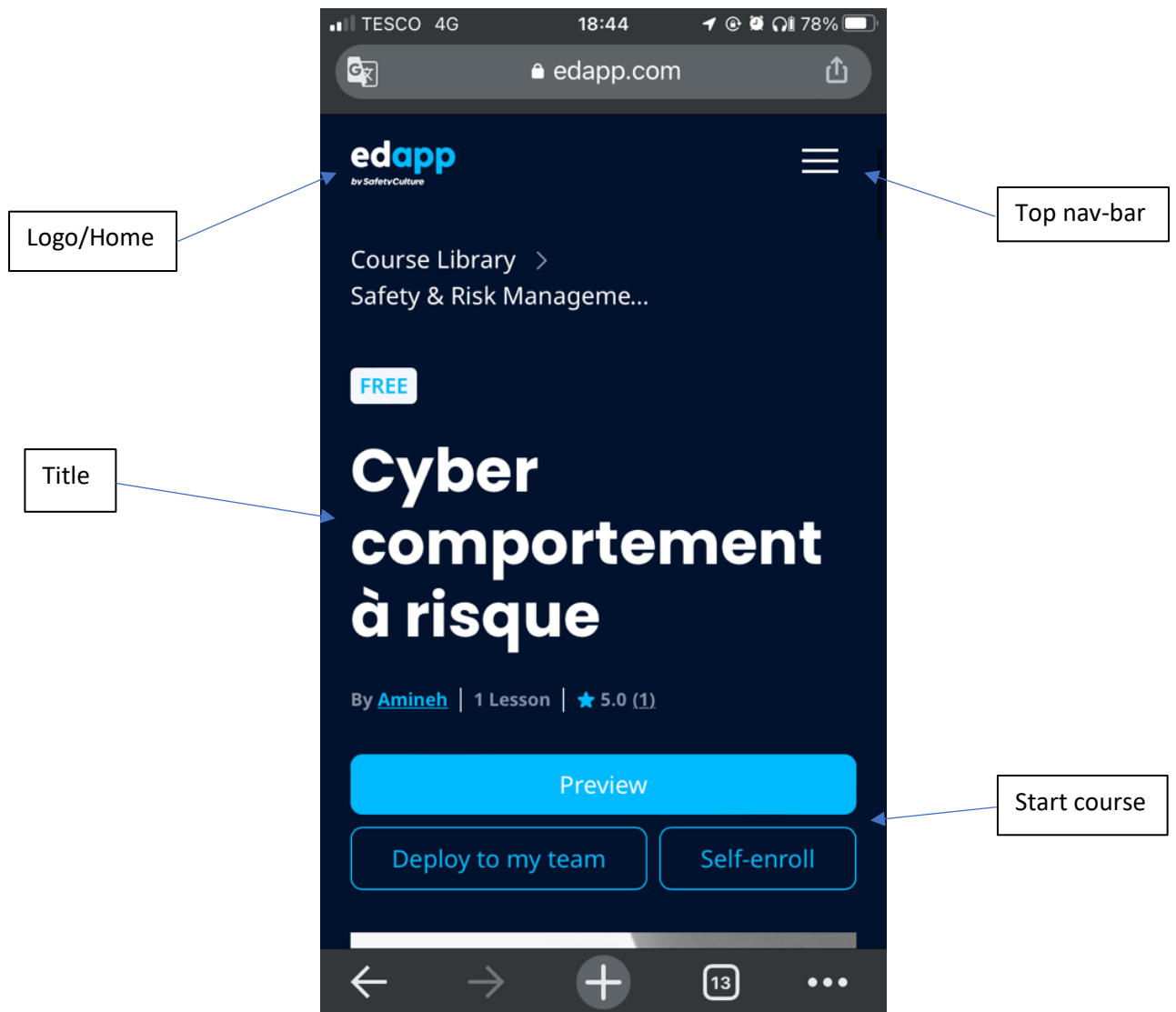


Figure 41: Annotation of Edapp course-specific page

Design

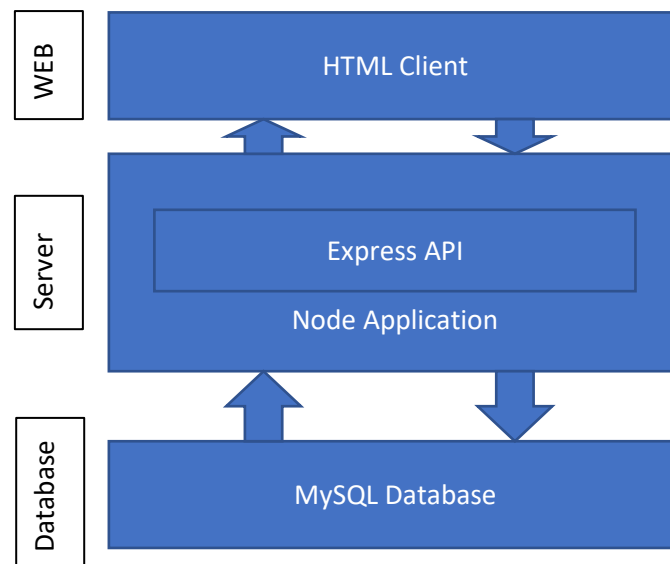


Figure 42: Diagram of project architectural design

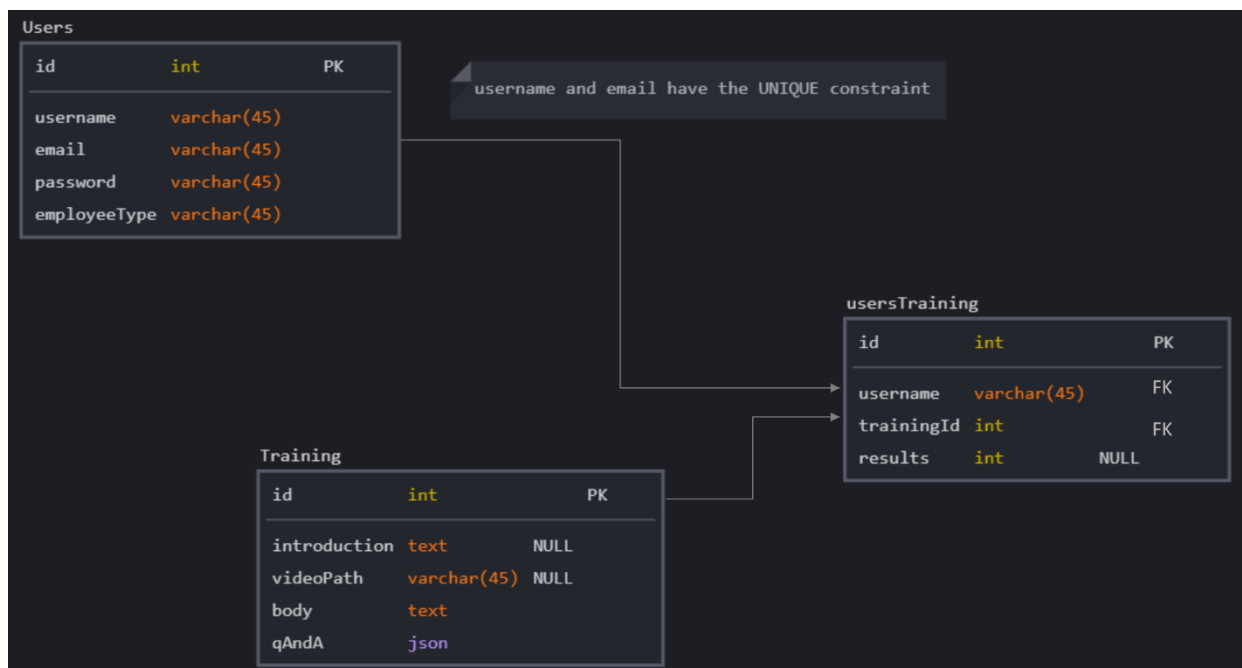


Figure 43: Diagram of project MySQL database design

Implementation

Set-up

Welcome to Brighton Domains Manage Your Account Howdy, Jake Ward (student)

Home **Dashboard** IT Service Desk

Search (/) jw1448 LOGOUT

MySQL® Databases

Manage large amounts of information over the web easily. MySQL databases are necessary to run many web-based applications, such as bulletin boards, content management systems, and online shopping carts. For more information, read the [documentation](#).

[Jump to MySQL Users](#)

Create New Database

New Database:

jw1448_

[Create Database](#)

Figure 44: Creation of database via Brighton Domains UI

MySQL Users

Add New User

Username

jw1448_

Password

Password (Again)

Strength ⓘ

Very Strong (100/100)

[Password Generator](#)

[Create User](#)

Figure 45: Creation of MySQL user via Brighton Domains

Add User To Database

User

Database

[Add](#)

Figure 46: Assignment of MySQL privileges via Brighton Domains

MySQL® Databases

Manage User Privileges

User: **jw1448_jwTrainingAdmin**

Database: **jw1448_jwTrainingDatabase**

☒ ALL PRIVILEGES

Figure 47: Assignment of all privileges to MySQL user via Brighton Domains

Current Databases

Search

Go

| Database | Size | Privileged Users | Actions |
|----------------------------|---------|------------------------|---|
| jw1448_assignment2Database | 3.28 KB | jw1448_admin2 | Rename Delete |
| jw1448_ci527_database | 5.16 KB | jw1448_jakeward2639 | Rename Delete |
| jw1448_jwTrainingDatabase | 0 bytes | jw1448_jwTrainingAdmin | Rename Delete |

Figure 48: Evidence of successful database creation



Node.js

WEB APPLICATIONS

[+ CREATE APPLICATION](#)

CANCEL

CREATE

Node.js version

10.24.1

Application mode

Development

Adds value for NODE_ENV variable

Application root

nodeapps/jwTrainingAPI

It is a physical address to your application on a server that corresponds with its URI. Upload your application files here.

Application URL

jw1448.brighton.domains/jwTrainingAPI

It is an HTTP/HTTPS link to your application

Application startup file

app.js

Passenger log file

/home/jw1448/nodeapps/jwTrainingAPI/passenger.log

You can define the path along with the filename (e.g. /home/jw1448/logs/passenger.log)

Figure 49: Creation of Node API via Brighton Domains

← → ↺ jw1448.brighton.domains/jwTrainingAPI

It works!

NodeJS 10.24.1

Figure 50: Evidence of successful Node application creation

Testing

To be done at a later date.

Research Material & References

(no date) *Software development life cycle (SDLC)*. Available at: <https://www.w3schools.in/sdlc/software-development-life-cycle-sdlc> (Accessed: October 11, 2022).

Hamilton, T. (2022) *Agile vs waterfall – difference between methodologies*, Guru99. Available at: <https://www.guru99.com/waterfall-vs-agile.html> (Accessed: October 11, 2022).

Corporate Finance Institute. 2021. *Data Protection*. [online] Available at: <https://corporatefinanceinstitute.com/resources/knowledge/data-analysis/data-protection/> [Accessed 12 October 2022].

GDPR.eu. 2016. *GDPR Archives - GDPR.eu*. [online] Available at: <https://gdpr.eu/tag/GDPR/> [Accessed 12 October 2022].

Boyd, C., 2021. *The User Researcher's Guide to GDPR*. [online] Userinterviews.com. Available at: <https://www.userinterviews.com/blog/the-user-researchers-guide-to-GDPR> [Accessed 12 October 2022].

Geitgey, A., 2018. *Understand the GDPR in 10 minutes*. [online] Medium. Available at: <https://medium.com/@ageitgey/understand-the-gdpr-in-10-minutes-407f4b54111f> [Accessed 12 October 2022].

Gaurav, S. (2022) *Difference between peer to peer and client server*, Scaler Topics. Available at: <https://www.scaler.com/topics/computer-network/difference-between-peer-to-peer-and-client-server/#:~:text=the%20computer%20systems,-,The%20main%20focus%20of%20the%20client%2Dserver%20network%20model%20is,stores%20individual%20files%20and%20data.> (Accessed: October 23, 2022).

Breux, G. (2018) *Client-side vs. server-side vs. pre-rendering for web apps*, Toptal Engineering Blog. Toptal. Available at: <https://www.toptal.com/front-end/client-side-vs-server-side-pre-rendering/#:~:text=Client%2Dside%20rendering%20manages%20the,displays%20a%20blank%20page%20first.> (Accessed: October 23, 2022).

Low, A., Siu, J., Ho, I. and Liu, G., 2014, November. Introduction to Node. js. In *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering* (pp. 283-284).

Editor (2020) *The good and the bad of Node.js Web App Programming*, AltexSoft. AltexSoft. Available at: <https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-node-js-web-app-development/> (Accessed: October 23, 2022).

(no date) *Brighton Domains Support Documentation*. Available at: <http://brighton.domains/docs/> (Accessed: October 23, 2022).

Pros and cons of reactjs - javatpoint (no date) www.javatpoint.com. Available at: <https://www.javatpoint.com/pros-and-cons-of-react> (Accessed: October 24, 2022).

React (no date) *npm*. Available at: <https://www.npmjs.com/package/react> (Accessed: October 24, 2022).

Good and bad of angular development (no date) *OdinSchool*. Available at: <https://www.odinschool.com/blog/programming/good-and-bad-of-angular-development> (Accessed: October 25, 2022).

Raval, N. (2022) *React vs angular: Which one is best for your next front-end project?*, *Radixweb*. Radixweb. Available at: <https://radixweb.com/blog/react-vs-angular> (Accessed: October 25, 2022).

Express.js mobile app development: Pros and cons for developers (no date) *Binariiks*. Available at: <https://binariiks.com/blog/express-js-mobile-app-development-pros-cons-developers/> (Accessed: October 26, 2022).

EJS (no date) *npm*. Available at: <https://www.npmjs.com/package/ejs> (Accessed: October 26, 2022).

Express-fileupload (no date) *npm*. Available at: <https://www.npmjs.com/package/express-fileupload> (Accessed: October 26, 2022).

Body-parser (no date) *npm*. Available at: <https://www.npmjs.com/package/body-parser> (Accessed: October 26, 2022).

Cors (no date) *npm*. Available at: <https://www.npmjs.com/package/cors> (Accessed: October 26, 2022).

MYSQL2 (no date) *npm*. Available at: <https://www.npmjs.com/package/mysql2> (Accessed: October 26, 2022).

Patel, J. (2022) *Django vs express: 2 best web frameworks compared in 2022*, *Monocubed*. Available at: <https://www.monocubed.com/blog/django-vs-express/#:~:text=Django%20is%20an%20advanced%20Python,js>. (Accessed: October 27, 2022).

Why developers like gulp (no date) *StackShare*. Available at: <https://stackshare.io/gulp> (Accessed: November 12, 2022).

Reeves, S. (2021) *Webpack vs gulp – the best build tools for Javascript projects*, *GoodCore Blog*. Available at: <https://www.goodcore.co.uk/blog/webpack-vs-gulp/> (Accessed: November 12, 2022).

Gulp vs Webpack: Top 8 differences between gulp vs Webpack (2021) *EDUCBA*. Available at: <https://www.educba.com/gulp-vs-webpack/> (Accessed: November 12, 2022).

Webpack (no date) *npm*. Available at: <https://www.npmjs.com/package/webpack> (Accessed: November 12, 2022).

Gulp (no date) *npm*. Available at: <https://www.npmjs.com/package/gulp> (Accessed: November 12, 2022).