CS3099: Junior Honours Project Final Report

PuzzleFlix: An Online Platform For Distributed Logic-puzzle Solving and Publishing With Support For Interaction Within A Federation

200006295, 200006387, 190015412
200014911, 200018293

University of
St Andrews

Date: 24/03/2023

Abstract

For this project, we were tasked to develop an online platform for distributed logic-puzzle solving and publishing with support for interaction within a federated community of other puzzle websites. As a team of five, we employed Agile methodologies to adopt an incremental development approach for creating not only the platform itself, but also a federation interaction protocol to allow our platform to interface with websites built by other groups within our federated community.

With the advent of user-friendly digital-content delivery platforms such as Netflix and Youtube, the demand for streamlined platforms for the delivery of various media is ever greater. Our goal for this project is to mimic the services and functionalities of these various pre-existing platforms with a focus on Sudoku and other similar puzzles as our content of choice. Our site supports industry-standard user registration and authentication, providing users access to our suite of puzzle-related functionality including, but not limited to, puzzle creation and solving, comment forums and puzzle ratings, XP systems & levels, federation Open Authorisation (OAuth), and much more.

This report highlights the goals and objectives for this project, including technologies we used, important design decisions, our incremental development system, and more broadly how we worked as a group.

Declaration

We declare that the material submitted for assessment is our own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated. The main text of this project report is 14,043 words long, including project specification and plan.

In submitting this project report to the University of St Andrews, we give permission for it to be made available for use in accordance with the regulations of the University Library. We also give permission for the report to be made available on the Web, for this work to be used in research within the University of St Andrews, and for any software to be released on an open source basis.

We retain the copyright in this work, and ownership of any resulting intellectual property.

Contents

Chapter 1

Introduction

1.1 Overview of Task

The basic specified requirements of this project are to allow users to register with the platform to solve, create, and publish Sudokus and other similar puzzles. Once registered with the site, users must be able to assume different roles within the site with varying levels of privileges (such as solvers, puzzle creators, moderators, and site administrators). Along with these basic features, it is necessary to include appropriate security measures and integrity reinforcement to keep the site robust and secure against any bad actors or any potentially harmful user actions. As a federated site, our implementation must incorporate the ability to interface with websites constructed by other groups. This is crucial to allow users to access all websites in our federated community and to retrieve puzzles created from these sites. After completing the basic features, we were encouraged to extend our implementation to include additional features of our choosing. These features ranged from attaching forums/comment sections to each puzzle to including automated solvers (to name a few).

1.2 Our Goals

As a website for hosting puzzles, our target user-base is rather vast and diverse. To ensure the best user experience for everyone, we placed a large focus on the ergonomics and the overall human-computer interaction (HCI) aspect of our website. In particular, we wanted to optimise our platform's usability for all different types of users regardless of technological proficiency. We decided to take Netflix (and other similar streaming platforms) as a model to emulate since it similarly has a massively diverse target user-base and is generally considered to be successful in catering to it. As a result, the layout of our website is quite similar to Netflix and other similar websites. However, we also wanted our platform to promote a sense of community and to facilitate discourse between users rather than provide an insular experience similar to that of Netflix. Creating a sense of community is vital for optimising user retention and engagement. Additionally it provides a medium for providing feedback to both the content creators and site creators to bolster the quality of improvements. Additionally, since it is a requirement to allow users to interact with each other's puzzles, it naturally follows that users should be able to directly interact with other users. To integrate this goal in our platform, we used Youtube as a model to emulate. In particular, we took influence from Youtube's features in allowing users to comment on content published by other users and in providing accessible and informative profile pages.

In regards to implementation, to date we believe we have been successful in implementing the core required functionality as well as a suite of additional features of our own choosing. We first focused on completing all of the basic requirements of the minimum viable product (MVP) before focusing on these HCI related aspects and the additional non-MVP features. We aimed to first allow

users to register and login into the site to both play and create sudoku puzzles. After implementing this core logic, we set out to create profile pages to, in a way, give users a viewable identity within the platform. Next we focused on the presentation and usability of our implemented functionality (the HCI aspect). Finally we implemented our additional features. This allowed us to work within an incremental development structure since we could divide our development into stages which are iteratively improved upon and extended.

Throughout our development, we concurrently worked with our supergroup (our federation) to design and develop a supergroup interaction protocol to allow different groups' platforms to interact and share content with other platforms within our federated community. This process involved outlining the extent of features to which our inter-group communication should encompass. At the base level, we are required to allow users to sign in to any website in our federation using credentials from other sites in the federation. Additionally each website must be able to send and retrieve puzzles to and from other websites in the federation. A part of this development process was focused on deliberating how to achieve these two requirements at the most basic level. The next step was to consider any additional features we wanted to include to both increase the robustness of our protocol and to extend our protocol to support additional non-required features. Ultimately, together with our supergroup, we were able to develop an interaction protocol that goes beyond the required basic specifications.

Overall, we are proud of the platform we have produced and feel that we have consistently made great progress throughout the past academic year. We have all learned a lot from the process both in regards to technologies but more importantly the team software development structure. While we have a finished product that goes beyond the required specifications, we do not believe we are finished and have plans to extend our platform to encompass more features which are outlined in 4.

1.3 Technologoes Used

1.3.1 Frontend

The front-end for this project is written in ReactJS using ES6 JavaScript syntax. Using a declarative front-end for a project like this was a trivial decision, as an interactive web app like this is exactly what declarative front-end frameworks are made for. As opposed to writing this project in vanilla JavaScript, which would have required us to explicitly bind certain actions to specific events for each user-interface (abbreviated as UI) update we would've wanted to make, React allows us to store and update the state of our app on certain event and render the UI as a function of the state. This allows changes in state to trigger a re-rendering of the UI, which removes a lot of potential state related bugs we would have had doing it in vanilla JavaScript, and makes it easier to reason about the logic of the state and the UI. We opted to use vanilla cascading style sheets (abbreviated as CSS) for our application, instead of a CSS superset like SCSS or Less. This decision was motivated by simplicity and also the fact that we didn't foresee the need for any extra features in our CSS. We also opted not to use a CSS UI library like Bootstrap or Material UI as we already had a specific design identity for our project and didn't want to be constrained by the generic looking components provided by UI library, not to mention the additional headache in modifying parts of the CSS for components in external libraries for any level of customisation. We also opted not to use a CSS-in-JS solution. This unfortunately means that styles are not locally scoped and exist in the global scope, however this is common in many front-end applications and in any case it's not considered best practice to use the same class names for unrelated elements. There are many reasons not to use CSS-in-JS[4], but the main reason we opted not to use it was to avoid inflating our bundle size and adding runtime overhead to our already browser heavy single page app. The build tool we are using to bundle all our JavaScript files is Vite. We choose to use Vite as it's currently the fastest build tool, not only when it comes to the initial build but also dynamic hot-reloading of any changed code

during development, and also the best maintained and most popular at the moment. When we initialised the project, the official React documentation recommended using create-react-app, React's default initialisation script which uses Webpack as its bundler. Create-react-app and Webpack are both particularly slow compared to Vite and even other front-end bundlers but as of last week, the newly updated React documentation recommends using Vite as a bundler if you're not using React with a larger meta-framework[10].

SSR vs SPA vs SSG

Another factor to consider when writing a front-end application in a framework like ReactJS, is which paradigm to use, the options of which are: server side rendering, static site generation, and a single page app. All three of these approaches allow us to write the code in largely the same manner but all three of them render the HTML and run the JavaScript differently. All three of these approaches also convert the JSX from React into separate, browser-compatible HTML and JavaScript as of course the browser does not understand anything other than that.

Static site generation can be ruled out as an approach pretty early on, as this paradigm is typically used on static sites without any dynamic data that don't have to update very often, or only need to be updated by a handful of users and every attempt to change the data on a static site requires the site to be rebuilt. As such this approach is not suitable for a dynamic application with frequent data changes.

Server side rendering and a single page app were both suitable approaches for this project, however some of the tradeoffs with server side rendering led us to choose using a single page app. Server side rendering essentially converts the React code into HTML and CSS on the server, before sending it to the client. While this allows for a quicker time to first contentful paint (i.e. how quick the code actually renders and displays on the browser), it makes the page transitions slower as the server has to render the UI before even sending it. A single page app on the other hand has instant page transitions as it essentially preloads all the data necessary before the entirety of the page loads. While this means our page loads slightly slower when the user first visits it, it also means every subsequent interaction is faster and overall when the main bottleneck we're optimising for is network latency and not device performance, using a single page approach is appropriate. This also aids us in providing a smoother and better user experience, something we tried to emphasise on with this project.

Choice of Frontend Frameworks

Now that we've concluded that using a single page app is the best approach for this project, we still have a world of front-end frameworks to choose from. A big part of using a framework, unfamiliar or otherwise, is the documentation and community which directly affects how easy it is to debug and find answers to problems online. As a result, new and niche frameworks like Svelte and SolidJS are already ruled out. While they're both very unique and useful frameworks with a significant performance advantage to React, the small community and lack of resources makes them an unsuitable choice for a production application, especially considering the fact that our team had limited experience with JavaScript and front-end programming. Factoring this in, we're left with three choices. Angular, React, and Vue. All three of the frameworks provide roughly the same utility, so the main things to consider are performance and developer experience. From the following graph 1.1 we can see the performances differences between the three frameworks.

We can see that Angular is significantly slower than React and Vue when it comes to the first and largest contentful paints, and also has a higher cumulative layout shift. Angular does have a slightly better first input delay than React, however the performance difference at initial page load is significant enough to remove Angular from being a contender. Angular is also infamous for its poor developer experience and given that we're a novice team, we opted not to use it.

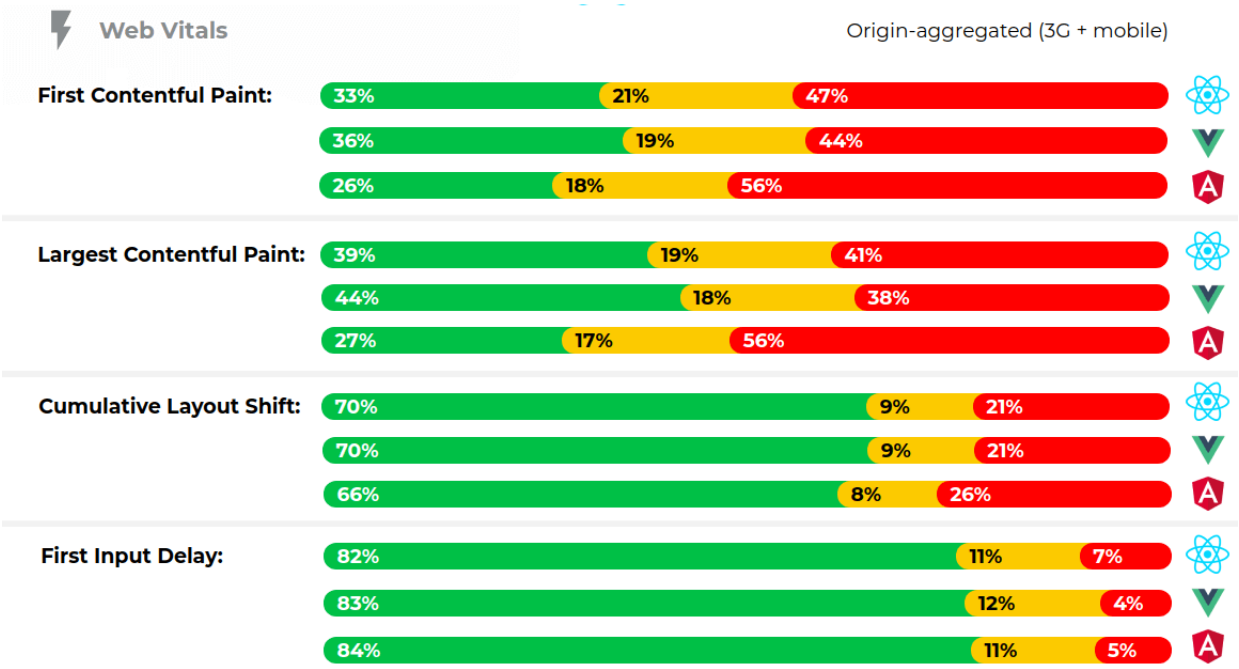


Figure 1.1: Source: [7]

Vue does have advantages over React, however the learning process of Vue includes a lot of Vue-specific patterns and functionality, something that doesn't necessarily translate to JavaScript ability in general and doesn't really translate well from JavaScript ability either. Vue also hides a lot of the declarative functionality and does a lot of "magic" when it comes to reactivity, which while being ideal for beginners in terms of ease, is definitely not ideal for beginners looking to learn and understand reactivity in a granular sense. As a result, we decided that using React would be the most sensible choice for this project.

1.3.2 Backend

Our server is written in ExpressJS, a Node.js framework for easily creating REST APIs. Express is an un-opinionated and lightweight framework, which does not provide much other functionality other than HTTP requests. It's very flexible and doesn't force any specific paradigm on the developer, and as such was an ideal choice for this project. Our server and client lie in different directories and the server essentially acts as a standalone API which the client can access. This approach is similar to the "JAMStack Architecture" and gives us faster performance. We elected to use Node.js on our backend as JavaScript across our app makes for a much more consistent developer experience, and also doesn't require the developers to learn an entirely new programming language. Express is also ideal for event-driven applications that receive a lot of requests. Since our application is essentially event-driven and has a lot of user input, the performance of Express with lots of requests ensures that our app can handle a large amount of traffic before server performance begins to be a bottleneck. Compared to other popular frameworks like Django (Python) and Spring (Java), Express can handle more requests per minute and its latency degrades slower when the number of concurrent users is increased. The database we are using on the backend is MySQL, using the object-relational-mapping (ORM) library "node-pg". This ORM is typically used with PostgreSQL, which was our database initially, but we decided to stick with the ORM when we switched to MySQL as it provides better security than using raw SQL queries.

1.4 Usage

Our app can be run by doing the following steps

1. ssh into your host server: `ssh <user@user.host.cs.st-andrews.ac.uk>`
2. ssh directly into cs3099user15: `ssh cs3099user15@cs3099user.host.cs.st-andrews.ac.uk`
3. Navigate to the project which can be found at: `~/Documents/project-code`
4. cd into `/puzzleflix-server` and run `./build.sh`
5. Enter git credentials when prompted
6. cd out to the parents and then into `/puzzleflix-client` and run `./build.sh`
7. Enter git credentials when prompted

Chapter 2

Project Details

2.1 Frontend Design Decisions

2.1.1 Main Goals

As with any front-end work, the design of the interface must come first and serve to guide the development. As mentioned before, as a platform with a wide and diverse user base, designing an ergonomic and universal user-interface with consideration of HCI aspects is imperative. We knew from the start that we wanted to mimic the general layout of Netflix's web client and tailor the smaller design aspects to be suitable for our content of choice, puzzles. This layout heavily utilises content belts (AKA carousels) to organise its content which allows a high level of modularity and flexibility. Using thumbnails, links to puzzle pages can be placed anywhere in the site in an intuitive and human-readable manner. Utilising content-belts allows groupings of thumbnails, belonging to various categories of content, to be grouped together in an intuitive way. This additionally improves our code development since it allows our code to be organised in a more modular and reusable way. With this we can take advantage of ReactJS's component-based structure to encapsulate the functionality of the elements of our site, such as thumbnails or content-belts, into separate components which can be used in a variety of scenarios. This helps reduce code redundancy as we can reuse components for different situations rather than have to re-define a new piece of functionality for each differing item. For example, our site has multiple content belts for the multiple types of puzzles that we have hosted on our site. Each content belt utilises the same component in our front-end codebase where the behaviour of each belt is tailored to its specific purpose through the use of parameters (in the same way as a function). By optimising re-usability, our code is more readable and more easily extendable which fits well with our incremental development approach.

Another aspect of great focus for our group was ensuring a fast, responsive, and dynamic experience for the user. This meant carefully balancing the execution of operations between the front-end and back-end. In general, delegating more work to the front-end provides a faster and more responsive experience for the user by reducing the amount of network traffic created from sending requests to the backend - but this can potentially sacrifice a level of integrity through lack of communication and validation of actions with the backend. So we put a lot of consideration into determining how our front-end HTTP communication endpoints were configured and the balance of their usage.

Our front-end design decisions can be split into five sections:

- Design Identity
- User Experience (UX) & User Interface Decisions
- HTTP Communication
- Puzzle Implementation

- Integrity Reinforcement

2.1.2 Design Identity

From the start of this project we knew we wanted to have a somewhat cohesive identity, and we didn't want to start programming the frontend blind, without any idea of what it should look like. Since we had already decided to follow a similar layout to streaming platforms like Netflix, we decided to name our application "PuzzleFlix". After doing this, we came up with a concept logo to help start distilling the design identity. The primary colours for the website we decided to use were red, and black or white depending on whether the website was in dark or light mode. We came up with a logo using these colours which signifies the fact that this is a website for solving puzzles, which is why the text "Puzzle" has been drawn with joining pieces, and also generally conveys the playful theme of the website. The logo was made using Adobe Illustrator and was exported as a vector to be used as an SVG.

Figure 2.1: PuzzleFlix Logo

After this, we made a very basic mockup of the signup page.

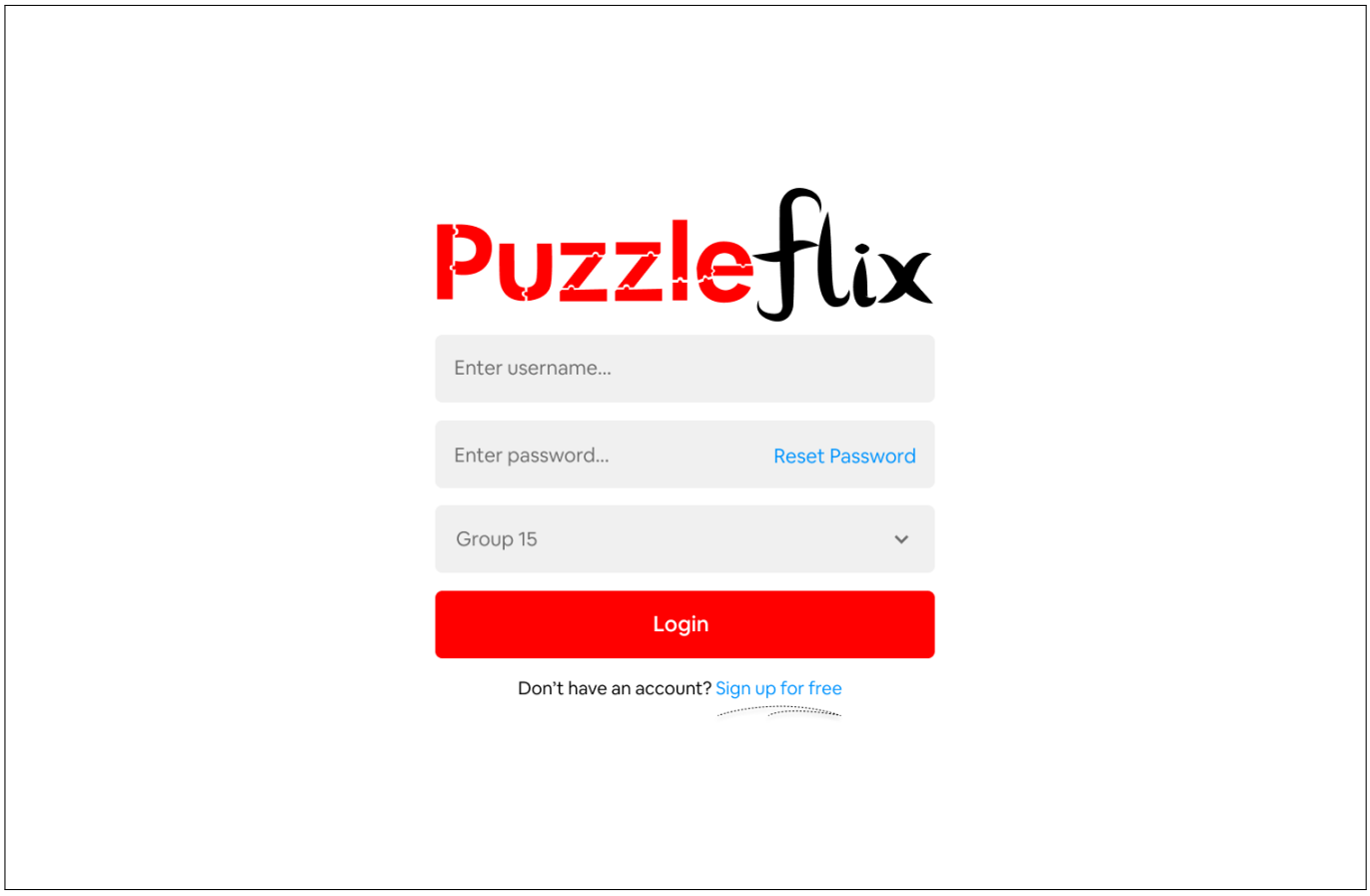


Figure 2.2: Login Page Mockup

This mockup was intended to solidify the colours of the UI, with red being the primary colour and blue being the accent colour, and some other design elements, like the rounded boxes and the

generally minimal and clean UI. The font we chose for our website is called Gibson and is licensed for commercial use from Adobe's TypeKit service. We also made a mockup of the same signup page in dark mode to solidify the theme there as well.

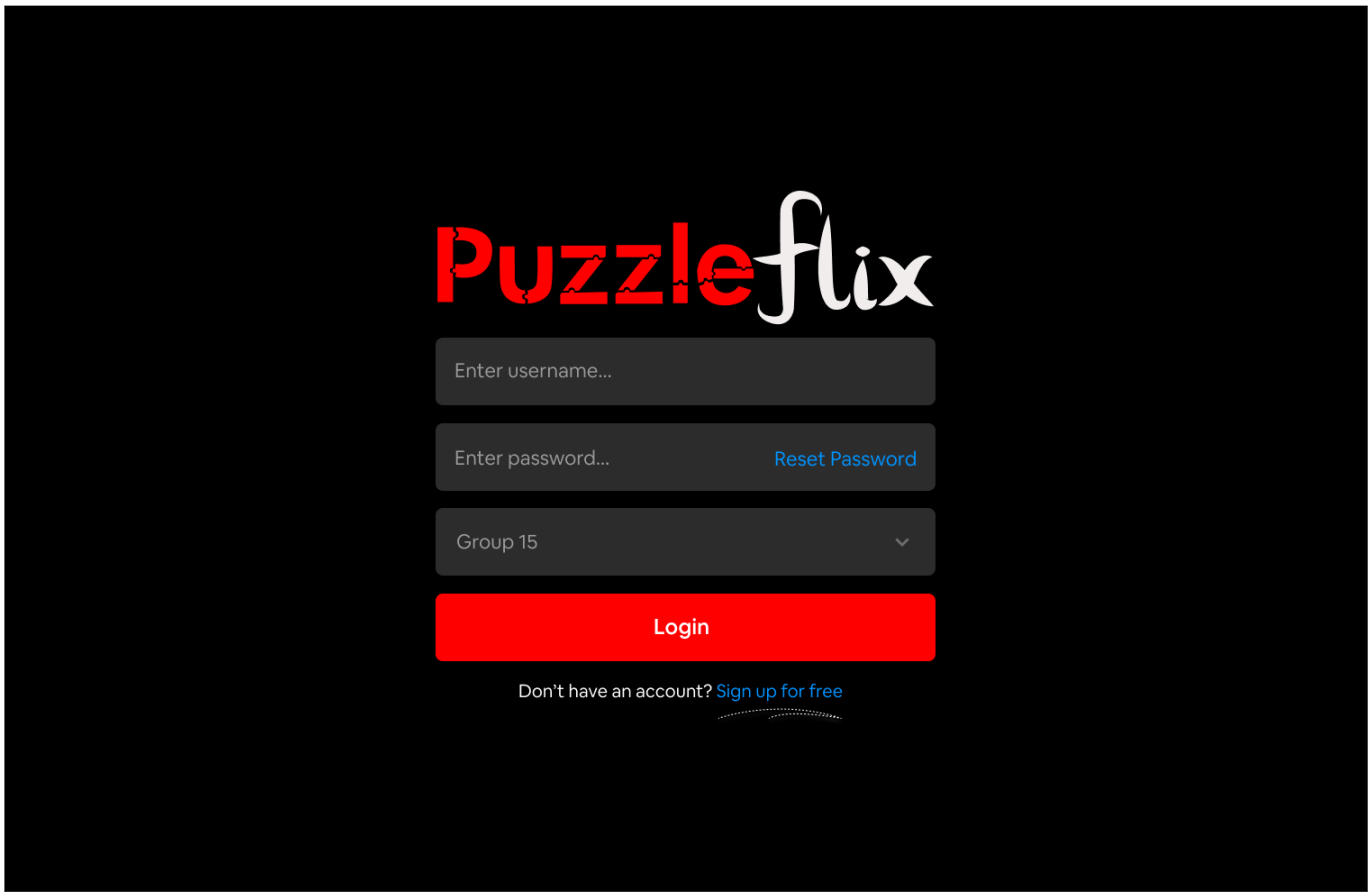


Figure 2.3: Login Page Mockup in Dark Mode

2.1.3 User Experience and User Interface Decisions

In this section I will discuss and describe some of the user experience and user interface decisions. I will also compare our website with Nielsen's usability heuristics[6] and highlight where we do a good job of complying with the heuristics.

Since the advent of smartphones and powerful mobile content, more and more internet traffic has been driven by mobile users. "As of February 2023, 52.08% of the total web visits are currently mobile, compared to 47.92% coming from desktops"[13]. With a majority of internet traffic coming from mobile, we wanted to ensure our site was not only performant on mobile devices but also responsive. As such, the responsivity of each page has been meticulously ensured and tested on multiple screen sizes.

We decided to utilise content belts (aka carousels) on the home page to display the puzzles. We have different content belts to display all the sudoku puzzles, all the eights puzzles, the puzzles created by the user, the puzzles the user is in the process of solving, and the puzzles the user has already solved. This vertical scroll layout is inspired by content streaming services like Netflix and is very familiar to most users. This familiarity is an important usability heuristic (match between system and real world) and the notion of "continue solving" is also sensible and familiar. Having pagination with a button click but also allowing users to scroll the content-belt with either their input device (mouse or trackpad) or with a scroll bar is also ideal as this optimised the layout for both desktop(Figure 2.4) and mobile(Figure 2.5a).

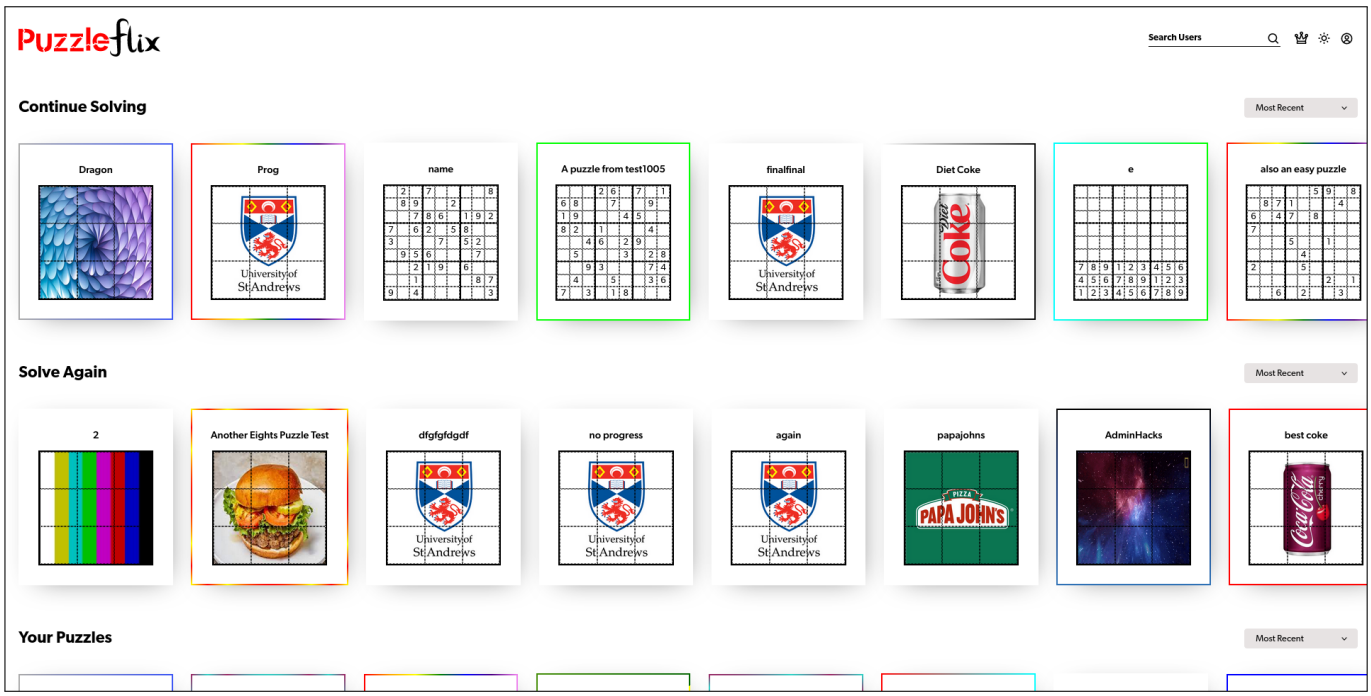


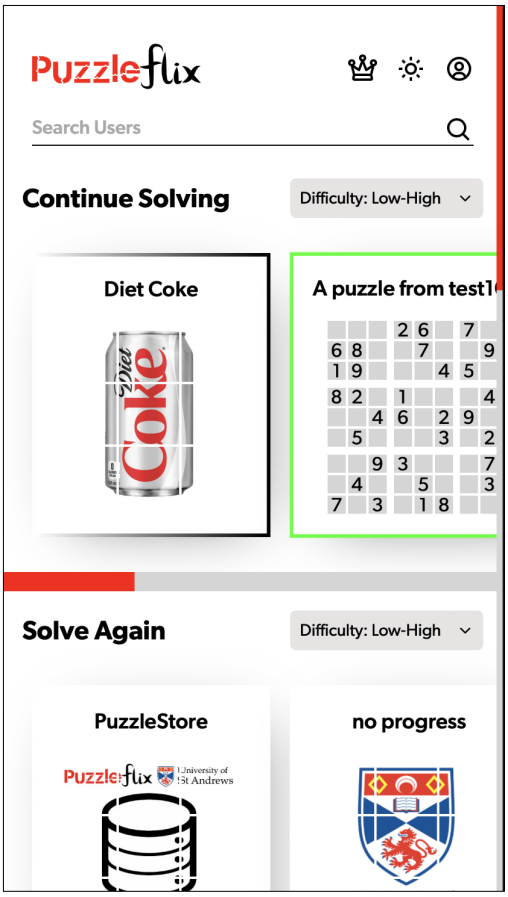
Figure 2.4: Desktop layout, where users scroll with the scroll bar underneath each content belt

The sudoku playing modal is probably the place where our design decisions made the most impact, as this directly affects gameplay. We decided to go with a slightly nicer looking grid than a traditional sudoku board, while not compromising on familiarity or playability. When a user inputs an invalid number, instead of highlighting the row, column, and box, we only show the user that they've made a mistake on input(Figure 2.6a). This is preferable compared to how most sudoku websites handle erroneous input, which shows the user exactly where they've gone wrong and makes gameplay essentially trivial(Figure 2.6b).

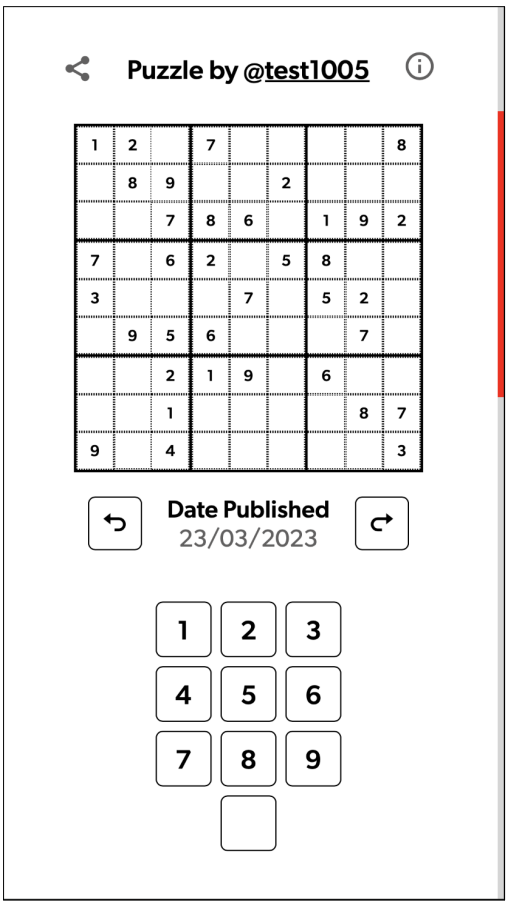
We also allow users to input digits by clicking on a cell and then a number in the control row at the bottom, or by navigating with their arrow keys and using the number row on their keyboard. Again, this optimises gameplay for mobile (by using on screen buttons, Figure 2.5b) and also for desktop (by using keyboard input, Figure 2.7). We also format the control row on mobile like a typical 3x3 phone input as this is the format most users are familiar with when inputting digits on a phone. We also have undo and redo buttons which allow the users to undo or redo all changes made in their current browser session by keeping track of all the changes. This complies with usability heuristics on user control and freedom, as users can easily undo and redo changes. Our website is also responsive and optimised for all screen sizes across the board.

The sudoku modal also shows an error message saying "invalid solution" when the user tries to submit an invalid solution and displays a modal with a success message and confetti when the user solves the puzzle. This complies with the usability heuristic "help users recognise, diagnose, and avoid errors". This page also has an intuitive and simple comments system, which is also mobile responsive, as well as a rating system for puzzles.

Our other primary puzzle is the eights puzzle. The page for this is very similar to the sudoku page(Figure 2.8), however the game play is simpler and only requires one click from the user on the square they want to move.

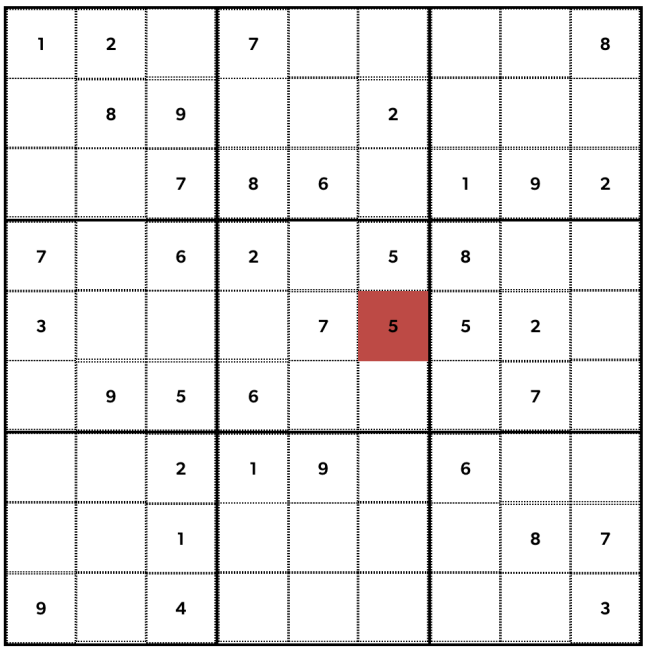


(a) Mobile layout, where users can scroll simply by dragging the content belt

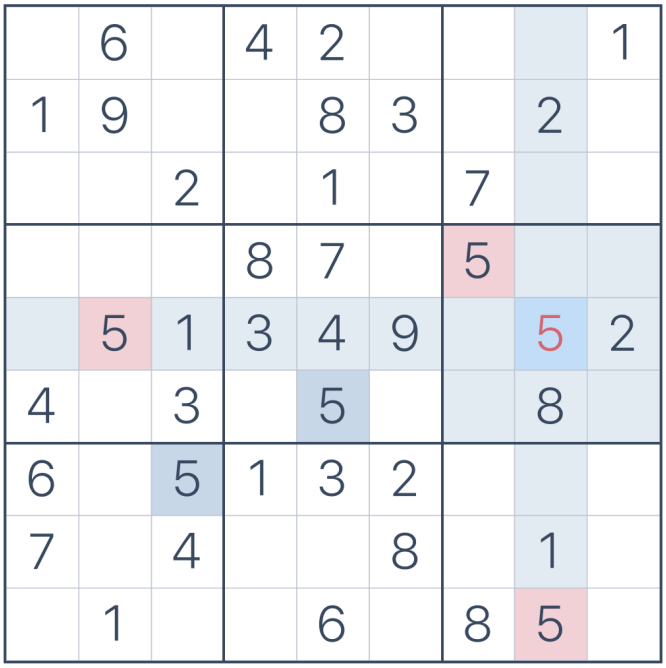


(b) Mobile layout while playing sudoku, with mobile appropriate input buttons

Figure 2.5



(a) Our sudoku modal only showing the erroneous cell



(b) The modal on Sudoku.com (as an example of a typical online sudoku grid)

Figure 2.6: Comparison of our and other website's sudoku playing modal

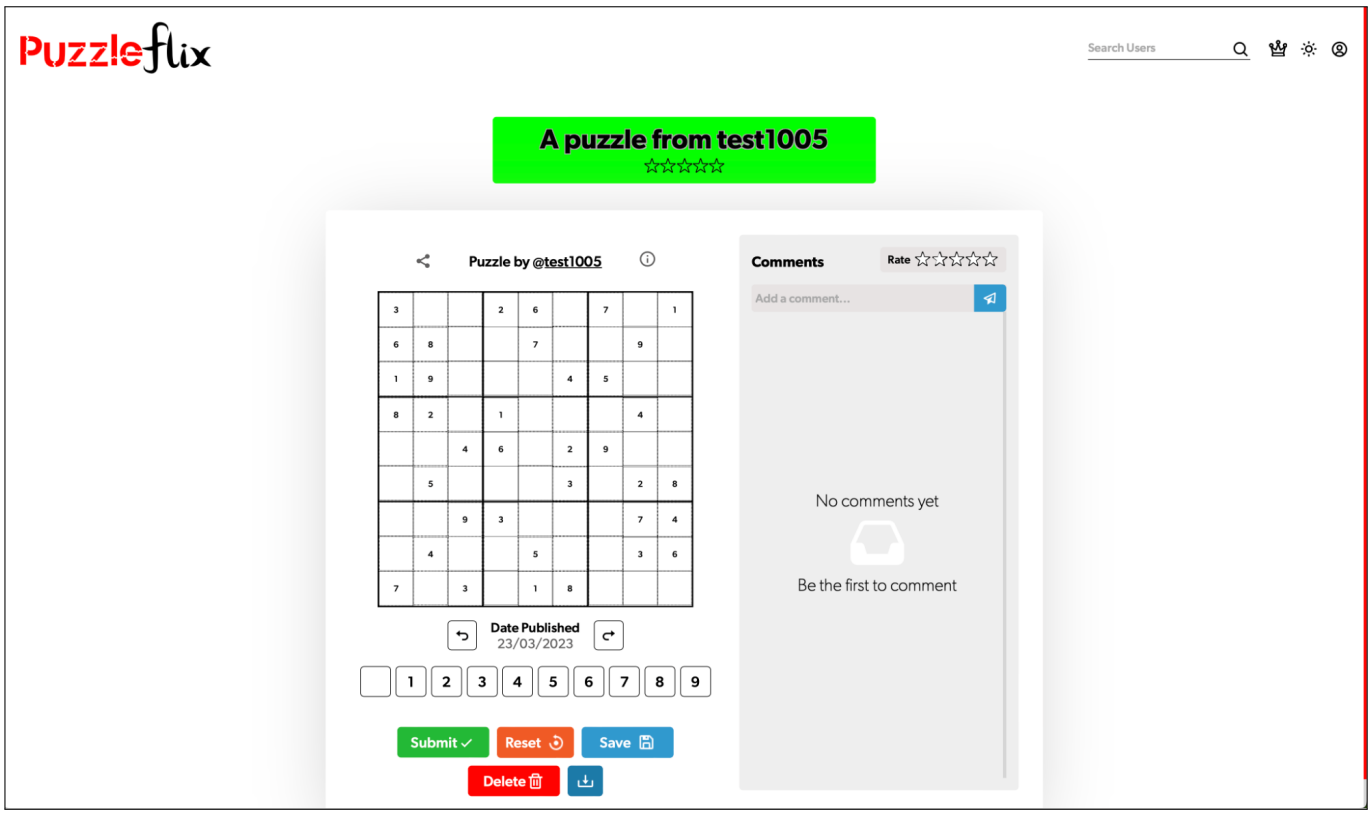


Figure 2.7: Desktop layout, where users scroll with the scroll bar underneath each content belt

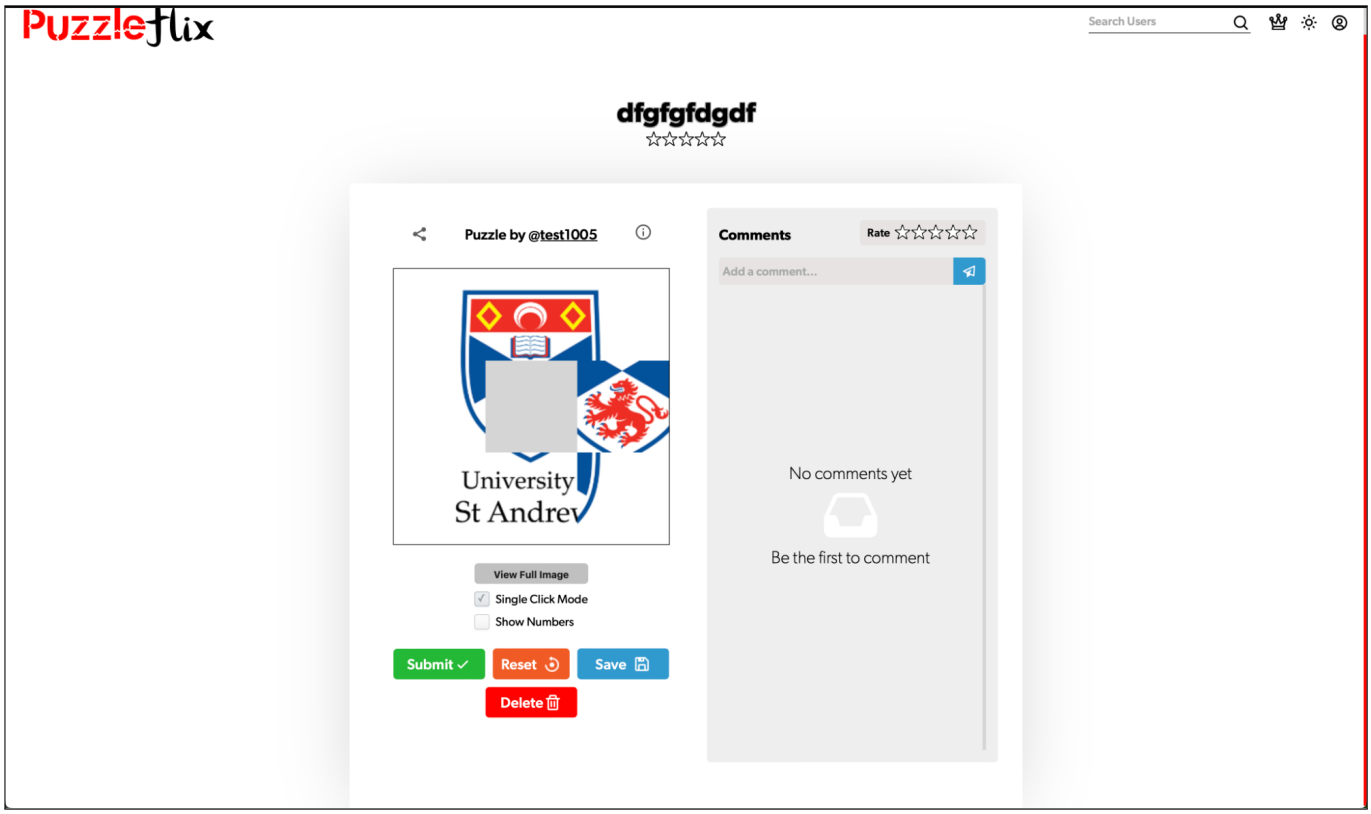


Figure 2.8: Eights puzzle page on desktop

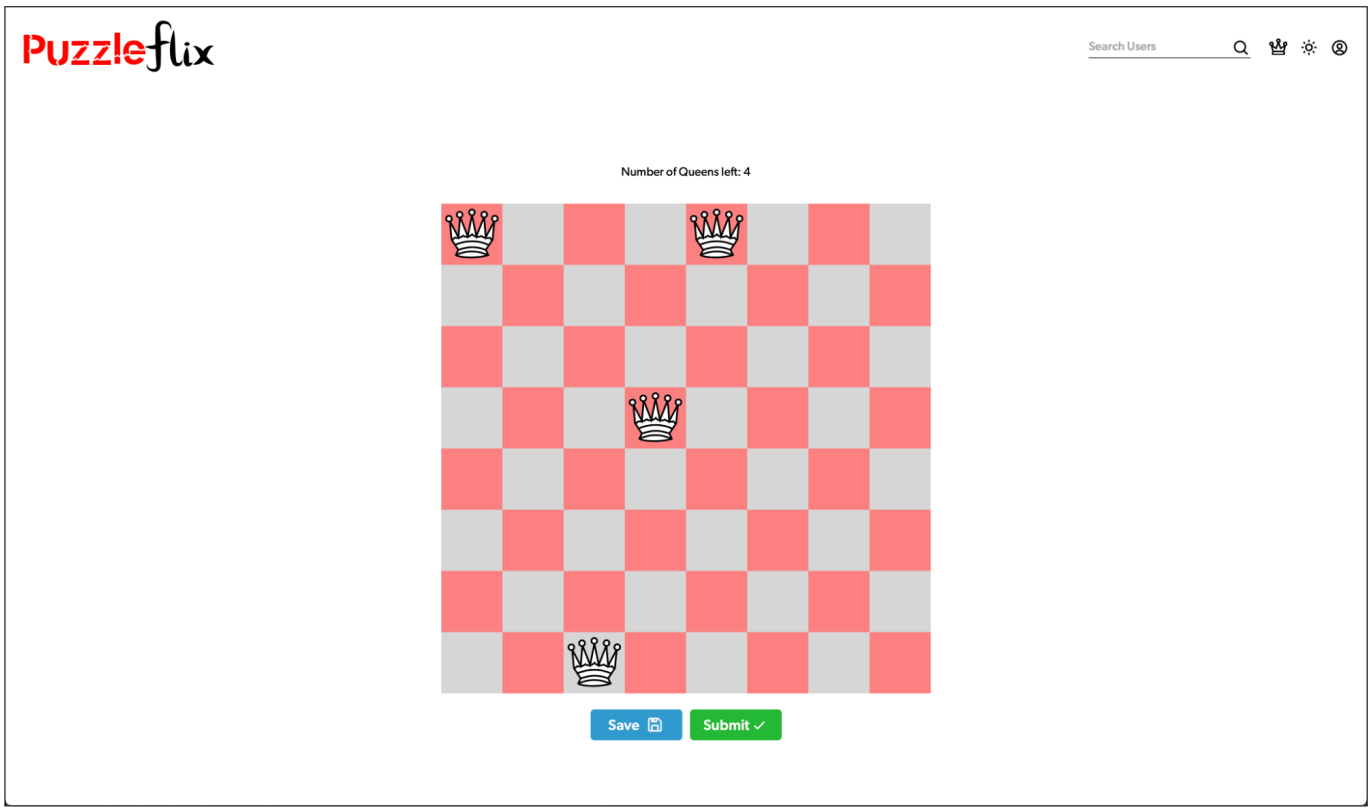


Figure 2.9: Eight queens puzzle page on desktop

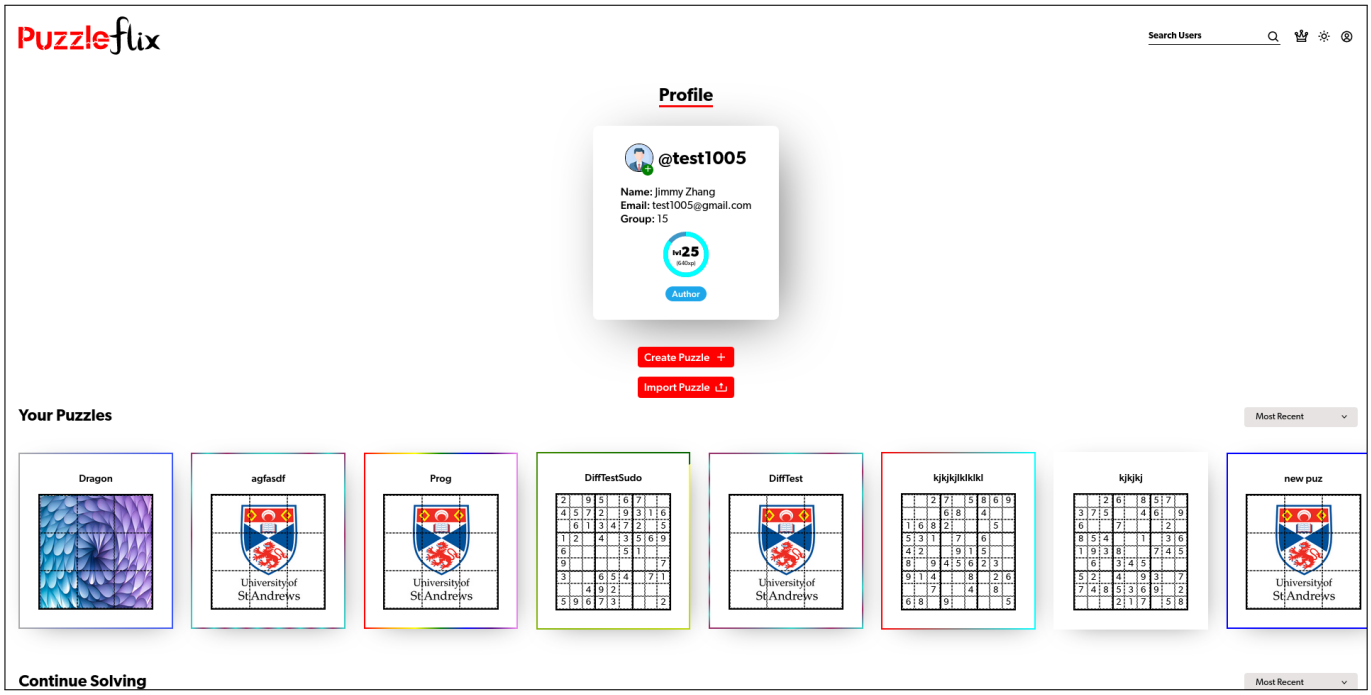


Figure 2.10: Profile page on desktop

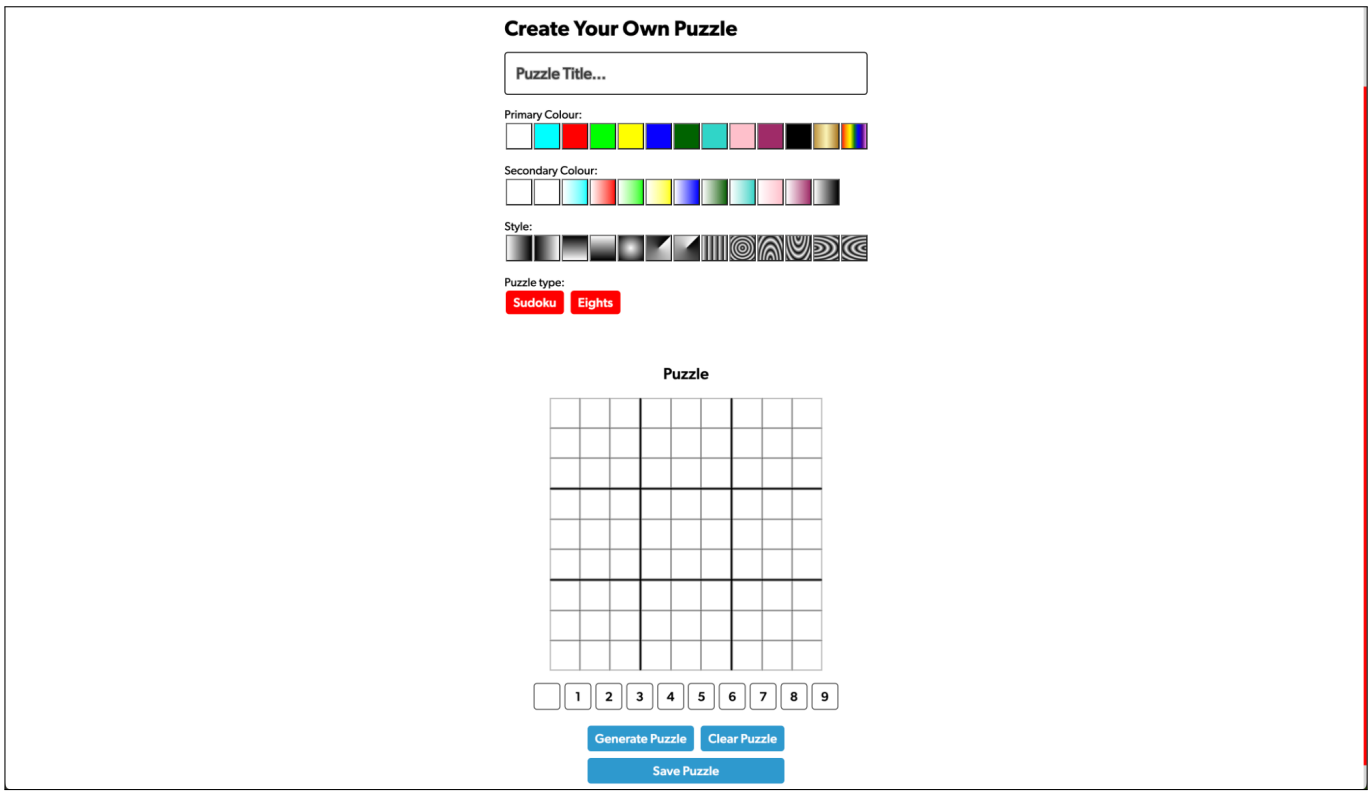


Figure 2.11: Create puzzle page on desktop

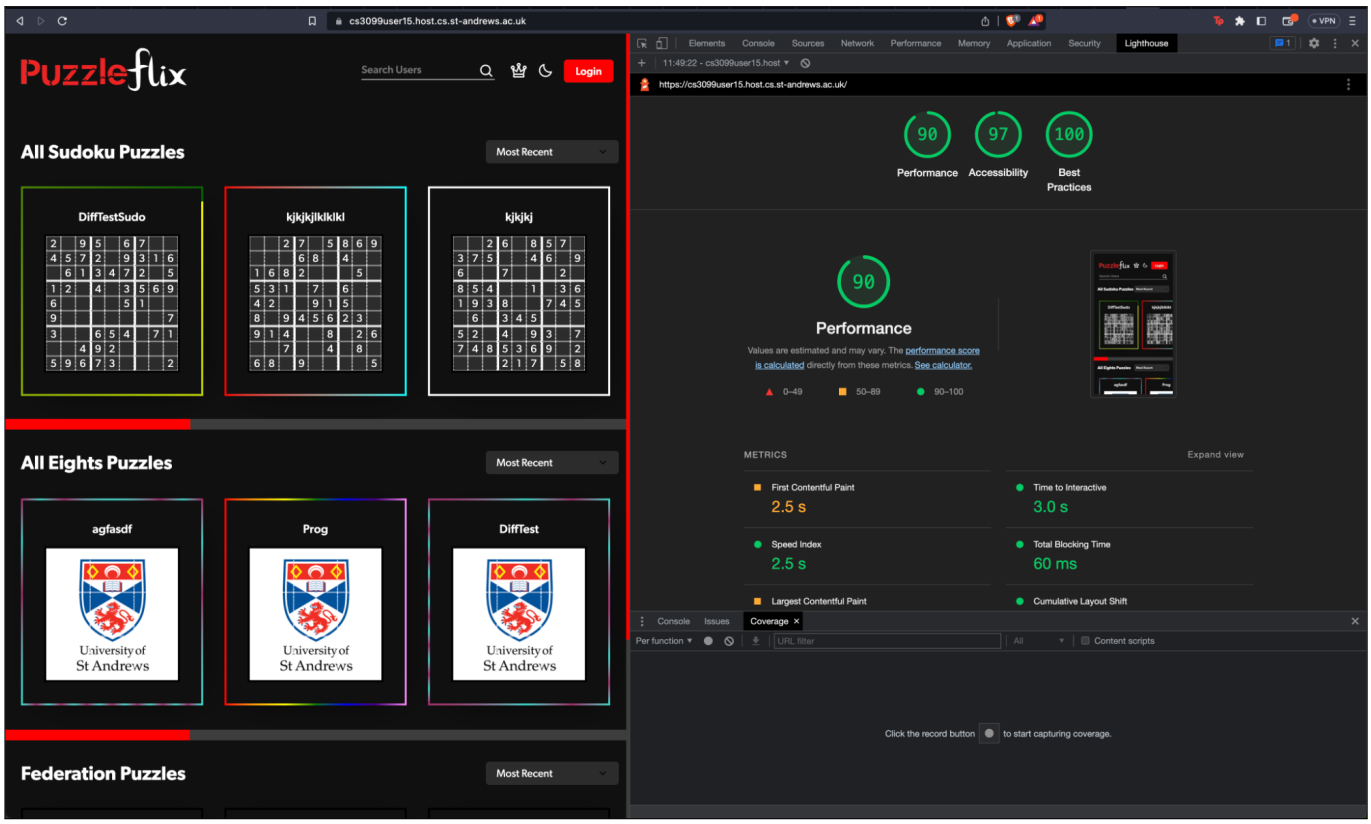


Figure 2.12: Brief overview of the report generated by auditing our site with lighthouse

The rest of our UI also keeps in theme with accessibility, responsiveness, and minimal design. We have an additional puzzle called eight queens(Figure 2.9), a profile page for each user(Figure 2.10), and a create puzzle page(Figure 2.11).

Note on accessibility and performance

In general our site performs very well on lighthouse audits(Figure 2.12). We consistently see a performance school of 90+ and at a minimum 80+, even on throttled mobile devices, and our accessibility score is consistently near 100%. Lighthouse does flag a few issues with accessibility, mainly the contrast ratio for the red button. However the WCAG guidelines for accessibility "don't always account for the high luminance contrast of white text. White is pure luminance with no hue or saturation, which is the strongest form for contrast". As a result, our read buttons are still readable and accessible even though they don't match AAA contrast standards.

2.1.4 HTTP Communication

The front end essentially has two roles:

1. To provide a visual interface for the user to interact with
2. To send and retrieve resources to and from the backend in response to various interactions performed by the user. This communication with the backend is essential for supporting the site

Our communication scheme utilises a REST API (Representational State Transfer API) as a protocol. This means that our application utilises HTTPs (Secure Hypertext Transfer Protocol) requests, such as GET and POST requests, to access data and functionality from the server. Using a REST API allows us to package necessary information into various components of HTTPs requests such as authentication tokens or JSON objects. In our case, a wide variety of data must be communicated to and from the frontend client. This data includes puzzle data , authentication tokens for protected resource access, user information, comments, and more. Additionally this data can take various forms from array structures to simple strings and numeric values. This made HTTPS the obvious choice for method communication for our frontend since it provides a secure channel of communication over the internet with encryption at the transport layer. Once an HTTPS connection is established, all data transmitted between the client and server is encrypted and can only be decrypted by the client and server which is necessary to prevent eavesdropping of sensitive data, such as user information, and prevents tampering of data which may affect the integrity of our website.

As mentioned before, we used the ReactJS Javascript framework to implement our frontend. This allows us to utilise the Axios library which is a powerful tool for making HTTP requests in web-application. It is a relatively easy-to-use API for sending asynchronous HTTPS requests to a server and provides native functionality for handling server responses. It supports an extensive range of HTTP requests including but not limited to GET, POST, PUT, and DELETE. For our application, we primarily used a combination of GET and POST requests. Axios gives more granular and detailed control over our HTTPS requests, allowing us to package various types of data in components such as HTTP headers, the request body, form data, or even encoded into the URL of the request.

So setting up communication endpoints in the frontend is relatively quick and simple with Axios. However, this raises the dilemma of how to balance our usage of this form of communication. From a networking perspective, while HTTPS requests are relatively cheap, they still incur an overhead. This overhead can both harm user experience by creating latency and can also overwhelm the server, potentially causing it to crash. To make our application as robust as possible, we had to consider how to limit the amount of network traffic between our client and server. On the other hand, a base level of communication with the backend is still necessary to ensure the validity and security of information and to improve performance for operations that require more computational resources and heavy-lifting. So a large aspect of our design work was focused around finding the right amount of communication between the client and server to balance these aspects.

With HCI being our overarching focus for our frontend, we leaned more towards a balance which prioritises user experience. So we tried to limit the amount of communication with the backend and left more computational work to be done in the front end to minimise latency and maximise responsiveness for the user. Additionally, we wanted to minimise the risk of overwhelming our backend since, due to time constraints, we were unable to dedicate as much time and work towards reinforcing the infrastructure of our backend to handle and recover from errors and deal with high network traffic. In general, we tried to use HTTP requests as sparingly as possible and only when necessary.

For example, our frontend structure involves the use of a lot of parent and child component schemes. Our content belts serve as a parent component to our thumbnail components. Each thumbnail must contain the puzzle data for the puzzle it has to display. Each content belt must contain the puzzle data for all puzzles it must display as thumbnails in order to both display the correct number of puzzle thumbnails and to ensure it displays the correct category of puzzles. To minimise network traffic, we only make a single HTTPS request to the backend from the content-belt component to retrieve a list of all puzzle objects the content belt must display using thumbnails. With this list of puzzle objects, the content-belt can then pass each puzzle object to a thumbnail component where each thumbnail component renders the puzzle object it gets passed respectively. With this, the content-belt and all of its thumbnails can obtain the necessary information it needs to render through the use of a single HTTPS request rather than having to make separate HTTPS requests for each component. So in this example, we can use a single HTTPS request to get all necessary information for a component and its children by propagating the data retrieved with the request down from the parent component to the children. Figure 2.13 below contains pseudocode to depict this scheme:

```
// Content-belt Component
function ContentBelt(props):
    // HTTPS API call to retrieve puzzles to display as thumbnails
    puzzle_obj_array = axios( get list of puzzle objects to display )

    render (
        // Pass each puzzle object to a thumbnail component to rendered
        for puzzle_obj in puzzle_obj_array:
            <Thumbnail puzzle_obj = puzzle_obj />
    )


// Thumbnail Component
function Thumbnail(props):
    // The puzzle object to render as a thumbnail
    puzzle_obj = props.puzzle_obj

    render(
       // Some HTML code to render
       // a thumbnail display for the puzzle object
    )
```

Figure 2.13: Pseudocode of content-belt and thumbnail scheme

In other situations, it may be inappropriate to use this propagation scheme. For example, when clicking on a thumbnail to access a puzzle page, it is necessary to ensure that the puzzle presented to the user in the puzzle page is the most up-to-date version. While we have the option of passing the puzzle data from the thumbnail component to the puzzle page component, and therefore be able to render the content belt, thumbnail, and puzzle page through the use of a single API call to the backend, this would be bad practice. A puzzle's data can change between the time the content-belt

and thumbnail are rendered and when a user clicks on a puzzle's thumbnail to visit the puzzle's page. So in this case, it is necessary to make another API call to the backend upon accessing the puzzle page to retrieve the most up-to-date version of the puzzle to display.

In other cases, the computational demand of a user action may be a guide for how we balance our use of API calls to the backend. For example, when playing a Sudoku puzzle, if a user makes an invalid move, our front end carries out the computation work of detecting a conflict in the user's game state and alerting the user. This requires relatively low computational resources and is algorithmically a cheap operation that can be done in negligible time in the front end. In this situation, it doesn't make sense to make an API call to the backend for every move a user makes to check whether the move was valid or not as that would flood the backend with HTTPS requests and also produce latency for an operation that should otherwise be instant for the user. Our site also includes an automated solution finder for Sudoku. This requires a lot more code and is much more computationally demanding. The automated solution finder is used much less frequently than Sudoku gameplay validation. Because it is computationally more expensive and less frequently used, it is a task that can be delegated to the backend to reduce the amount of code in the frontend and to take advantage of the backend's larger access to computational resources.

In general, these thought processes were used when designing our use of API calls to communicate with the backend server. We believe that although our communication scheme is not perfect, it is rather efficient and robust for our use case.

2.1.5 Puzzle Implementation

Puzzle Playing

We ended up deciding on using a thumbnails system for finding and opening puzzles. We believe that this is a very intuitive and simple way to find puzzles where the thumbnails simply display the puzzle. Once a user finds the puzzle that they want to play, they are sent to a page that shows the puzzle (the puzzle is based on the link which allows them to be shared with other people too). The puzzle page itself shows a variety of information. One thing that this includes is the title banner, with the title of the puzzle and a custom background that the creator can create themselves based on their level (as discussed in the level system section) which adds some entertainment value to our site and encourages people to solve more puzzles and increase their level to show off to other users. This is then followed by a secondary information bar that consists of a share button as mentioned, the author of the puzzle (which also serves as a hyperlink to visit the creator's page) and an information button which opens up a tooltip detailing how the puzzle should be played and any rules that the user may want to know. This is then followed by a puzzle player which is loaded based on the type of puzzle - these will be discussed below. Underneath the puzzle player is a row of buttons consisting of the submit, reset and save functions for all puzzles; a delete button if you are the creator or an administrator (admin) and an export button on sudokus that - on press - prompts the user to download the puzzle object locally in a format decided by our supergroup so that we can export and import puzzles from the other sites. Finally, to the right of the puzzle player (or below on mobile) , if you are logged in, you can see the comment box along with the rating bar so that users can interact with each other and give feedback on puzzles. Each comment in the scrollable comment box contains the user's profile picture (which on press redirects you to their profile page), the user's name, message contents and date posted. Similarly to the puzzle itself, admins and comment authors can see a delete button which allows admins to delete inappropriate comments or for people to delete their own posts that they may come to regret.

Sudoku Player

The SudokuModal inside PuzzlePage houses most of the sudoku logic, but also has some interactive elements and contains logic which helps implement the playing functionality. Above the modal is a share button which copies the unique URL of the puzzle to the clipboard and a information tooltip that can be clicked to display instruction, as well as the name of the author of the puzzle. Below the modal is a button to save progress or reset the puzzle to its initial state. The SudokuModal itself renders each element in the 2D of the puzzle as an html div with a number in it, each div has an "onclick" handler that saves its index in state and uses that index to edit the value of the cell not only in the UI but in the puzzle itself. The indices of the cells start from 00 on the top left, with the top right being 08 (so the second digit increments horizontally), 80 on the bottom left and 88 on the bottom right (so the first digit increments) vertically. The puzzle is also represented as a 2d array of objects, where each object contains the cell's value, whether or not it was pre-filled (i.e. a clue and not a user used filled digit) and its index. Once the index of the currently highlighted cell is in state, clicking on of the red input buttons below changes the value of the cell. Every time a value is changed, there is some functionality to loop through the row of the cell, the column, and the box and see whether that digit is already present, in which case it turns the cell red and tells the user they cannot input that there.

All puzzles on the site are accessed through clicking a puzzle thumbnail which links to a separate page for solving each particular puzzle. Depending on whether a user has logged in or not, the puzzle page will take two different appearances:

- When not logged in the page will only display the puzzle title, the puzzle grid itself, a share button for copying the link to the particular puzzle page, a refresh button for resetting the puzzle to its default state, and a submit button for checking the validity of the solution filled out by the user.
- When a user logs in, the puzzle page contains an additional button for saving the user's progress in solving the sudoku puzzle.
- When a user is logged in, upon clicking on a thumbnail to visit a puzzle's page, a check is made to determine whether the user has previous progress saved for the puzzle or not. If the user does have progress in the puzzle, the progress state of the puzzle will be rendered. Otherwise the default state of the puzzle will be rendered (by default, when a user is not logged in, the puzzle page will always show the default puzzle state).
- The reset button simply makes a request to the backend for the default puzzle data and replaces the puzzle grid on the puzzle page with the default puzzle values. The save button saves a user's progress by sending the current state of the puzzle along with the user's authentication token to the backend where the backend will first verify the token's validity, and then make the appropriate updates to the database to save the user's progress.

The function checkConflict in SudokuModal performs the validation when the user is inputting digits. This function loops through the 2D array that represents the sudoku, and takes spliced arrays of the row, column, and box corresponding to the cell that has just been modified. If the function finds any duplicates, it returns true and the user is allowed to continue playing but they are notified of their error by the highlighted cell turning red. This is displayed unobtrusively, while not showing the player exactly where their input is conflicting, and complies and complies with Nielsen's usability heuristic on error prevention. If the function finds no conflict, it returns false.

Eights Player

The eights puzzle player simply gives you a 3x3 grid that includes 8 total tiles and an empty tile that each contain a specific part of a broken down image. This player has two input modes, single

click and double click. When you click a piece, by default, it will slide into the empty tile if it is possible to do so. If the user disables the single click mode then they get to select a tile first and then click the empty space to send the tile to - which was the original way to control the puzzle and some of us preferred to play it this way (hence it is now a selectable play mode) but it is arguably inconvenient as the default setting as it just adds unnecessary complication to playing the puzzle. To aid in the solving, the user can enable a number guide which overlays the tile IDs over the image in case they are unsure on the order of pieces or if someone were to upload a puzzle of a solid colour. These numbers, for accessibility are black with a white outline and can be seen no matter what the image is. Furthermore, there is also a toggle (which also activates on hover) that lets you see what the final puzzle should look like. The reason for this is that it allows them to see how it should go together in cases where it may be not understandable at first. The user is able to submit the puzzle when the pieces are all in order.

Eight Queens Player

Since the eight queens puzzle requires entirely different gameplay logic than sudoku or eights, we decided to put it on a separate page. Additionally, due to the nature of the puzzle itself, there can only be one initial state, and since there can't be multiple variants of the puzzle, it felt out of place on a content belt. The EightsQueenModal represents the board as a 2D array, with 0s denoting blank spaces and a 1 to denote a queen. Since the queens can only be placed and removed, there's an "onclick" handler on each cell that simply adds or removes a queen if there already is one. This also keeps track of how many queens the user has placed and doesn't allow them to place more than eight. When the user clicks submit, the function hasConflicts checks that there is only one queen in every row, column, and diagonal. Checking the rows and columns is easy as the function simply loops through the array for rows, and takes a splice of the 2D array as a column with the helper function getColumn which simply returns a column from the 2D array as an array. While looping the function calls the helper function getAllIndexes for every row and column with the value "1". If the length of the returned array is more than 1 (i.e. there is more than one queen in some row or column) then the user has not won the game. The helper function getDiagonalsOfMatrix returns a 2D array of all the diagonals of a passed in 2D array by performing 6 separate loops. getAllIndexes is called again for each array in this 2D array and if there are still no duplicates then the user has submitted a valid puzzle. The save button on this page simply stores the users progress in the database so they can return and continue solving the puzzle later.

Saving puzzles

When a user wants to stop solving the puzzle for their current section but does not want to lose all of their progress so far, they are able to press the save button. This save button sends a request to the database to store their current state of play. Next time they open that puzzle, the website will load the puzzle as exactly how they last left off - and if they decide that they no longer want to keep this progress then they can hit the reset button to start over. However, the reset button does NOT force a save which means they can refresh the page to get their progress back in the event of an accidental press - they can manually save over their progress if they truly want to start over. Whenever the user has puzzle progress, the puzzle will be shown in the continue solving content belt - where they can see all of the puzzle that they have been solving.

Submitting puzzles

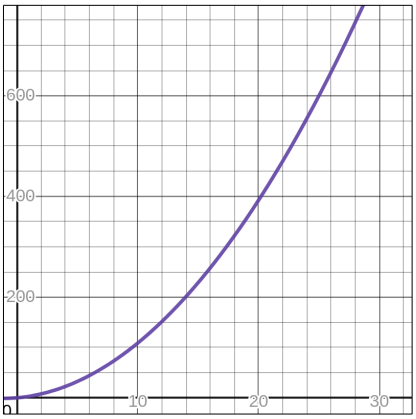
When the user presses the submit button, the front end will perform some checks to check if the puzzle is correctly solved or not. If the puzzle is incorrectly solved then a red error message pops up to tell the user that it is incorrect. If the puzzle is actually solved then they are greeted with a

congratulations popup which - on closure sends them back to the puzzle page. The database stores this victory and awards them with the correct amount of experience points (see level system). If the user resubmits a puzzle then they are shown the message again but they are not awarded more XP to prevent people from exploiting a puzzle that they know the solution for. Upon a puzzle completion, the puzzle will be shown in the "Solve again" content belt and not shown in the "Continue solving" belt.

Level System

Our website includes a player levelling system. This system works by allowing users to gain experience points (XP) which reflect the number of puzzles that a user has solved. Each time that a player submits a puzzle that they have never solved before, they gain experience points which are based on the puzzle difficulty (1xp for an easy puzzle, 2 XP for a medium puzzle and 3 XP for a hard puzzle). We decided that seeing huge numbers such as "633 XP" can get a little bit complicated to keep track of and does not really convey the user's efforts (for example, gaining 1 XP when you already have 600 XP does not look impressive but levelling up every so often really helps you feel like you are gaining progress) so we have a formula that converts the experience points into a level number - with each level requiring more XP to reach than the last.

Here is the Level (x-axis) to XP (y-axis) required graph:



XP to reach a certain level:

$$XP = \lfloor \frac{7}{8} \times Level^2 + \frac{17}{8} \times Level \rfloor \tag{2.1}$$

Get level from total XP:

$$Level = \lfloor \frac{-17 + \sqrt{224 \times XP + 289}}{14} \rfloor \tag{2.2}$$

We decided on these formulas because it ends up giving us a really nice level progression curve (the aim was finding a smooth curve that would give us a level of 25 (our decided max level for unlocks) which requires 200 normal difficulty puzzle completions - a substantial amount but not impossible). These curves allow new players to really feel like they are improving quickly while longer term players do not find it too easy to level up.

In addition to the ability to earn levels, the users can unlock features as they level up. Here are the unlocks we have at the moment:

Lvl 1	Ability to create puzzles unlocked
Lvl 1-5	Individual colour unlocks in the puzzle creator
Lvl 5	Unlock the ability to choose secondary colours and gradients in the creator
Lvl 5-10	More colour unlocks in the creator
Lvl 15	Custom gradient styles unlocked in the creator
Lvl 20	Gold banner unlocked
Lvl 25(600 xp)	Rainbow banner unlocked

These features are spread out throughout the different levels so that users can really feel like they are earning things by playing puzzles and they are able to customise their puzzles to attract more players and show off their own achievements. They will be explained in the puzzle creating section.

Puzzle Creating

There are two ways to create a puzzle on our website. The first way is to use our puzzle creator which can only be accessed when you reach level 1 (i.e. playing a single hard puzzle) and the second way is by uploading a puzzle from a file which allows users to submit puzzles from other websites to play on our own site. The user can access both options from their profile. If the user selects the import puzzle button, then they are greeted with a prompt to choose a file from their computer. Our site will then add this to the puzzle database and if the particular group did not follow the specification correctly (which a few groups did) and gave us a puzzle without a solution then we generate a solution for it to store too. On the other hand, if they select the create puzzle button (which is only active if they are level 1 or higher) then they are redirected to the create puzzle page. Whenever someone submits a new puzzle, they are automatically redirected to the page that displays the puzzle that they just published.

Create puzzle page

On this page, we have a number of accessibility features and some customization options that boost the overall user experience. At the top of the page, they can choose a puzzle title followed by the banner/border styles for their particular puzzle. The available options for primary colours, secondary colours and styles are dependent on the user's level. Any unlocked colours are shown as a button with their colour and style and any locked puzzles are displayed as grey and - on hover - show the level required to unlock them. Finally, the page gives them the option to choose either to create a sudoku puzzle or create eights puzzle - which the respective modal for is switched on depending on the selected puzzle as seen below.

Sudoku Creator

The sudoku created works much like the normal sudoku modal, except the sudoku creator modal does not allow users to input multiple conflicting cells. The creator asks the user to fix conflicts before moving to the next cell and blocks inputs until all conflicts are fixed. Before submitting, the page ensures that 17 or more cells are filled, the puzzle has no conflicting cells, and the puzzle has a name. If any of these fail an error message is displayed to the user and they can try again. Once a valid sudoku is submitted, the auto solver on the backend generates a solution. If there is no valid solution, it returns that to the user and does not store the puzzle. If there is exactly one valid solution the puzzle gets stored, however if there are multiple valid solutions the puzzle is still not stored and the user is told they cannot create sudokus with multiple solutions. The page also has a "generate" button that allows a user to generate a random valid sudoku with only one valid solution.

For a more detailed explanation of the automated solver and sudoku generator, refer to section 2.2.4.

Eights Creator

The eights modal simply includes a piece of text that informs the user what to do, the same toggles as the player so that they can play how they want and an upload from file button. The goal of the creator is to scramble the puzzle however they see fit and to choose an image to be displayed on the puzzle. If they try to submit the puzzle without a title or without scrambling it at all then an error message pops up and prevents submission. If the user chooses an image then an api request is sent to the backend where the image is stored and hosted so that any users on the website can see it when they play the puzzle - the backend uses a library to convert their input into a jpg file and the result URL is sent back to the client and shown on the puzzle. If the user does not add an image then the default image (the St Andrews logo) is used instead.

Submit button

The submit button exists on all puzzle pages. It posts to the backend and checks whether the submitted solution is correct or not. If it's not correct, the page displays an error message, and in the case of sudokus or the eights queen puzzle, it tells the user what the error was (like conflicting cells, board not entirely filled out) and if it correct, the SuccessModal gets triggered which shows the user a win message and displays some confetti using the library react-confetti. If it is a sudoku, it gets cleared and placed in the users "solved again" modal and if it's the eight queen puzzle it simply gets cleared.

2.1.6 Integrity Reinforcement

Integrity reinforcement pertains to the protection of resources that need to be protected in the frontend client to prevent unwanted actions or to avert bad actors from corrupting the site performance and/or data stored in the platform. Essentially we do not want users to gain access to privileges that they do not have which can include gaining access to higher level moderator privileges or gaining access to protected information of another user.

One of the main areas of focus for integrity reinforcement is ensuring a level of validation when retrieving sensitive resources from the backend. Essentially, when the backend receives an HTTPS request for a protected resource, it must ensure that the user requesting the resource (from the client) is in fact who they say they are and is allowed to retrieve said resource. When logging in and signing in, our backend provides the client with an access token which is generated through hashing the user's credentials into a unique key. This key is then sent to the frontend to be stored in local storage. In order to retrieve a protected resource from the backend, this key must be included in the API call to the backend where the backend verifies the validity of the token by decrypting it. This allows the backend to verify that the user requesting the protected resource is in fact who they say they are and is allowed to retrieve the resource.

On the backend, we ensured that all HTTPS route handlers utilise a provided access token for identifying the users who sent the request. The alternative to this would be to include the user identifier (user_id) explicitly in each request (such as in the request body), however, with this method, users would be able to modify the user_id they send to the backend. By strictly relying on using access tokens as a means of identifying users, we can ensure that the users cannot tamper with their identity and potentially modify data pertaining to other users - and the only possible way to obtain an access token is through login and signup.

This also raises the issue of what type of data can be visible to the user and what data should be hidden. Our frontend framework ReactJS gives us a variety of ways to pass information throughout the site. Information can be stored and passed from one component to another through appending the information in the URL (which is visible and modifiable by the user), or through other methods that are invisible to the user such as through component "props". For example, the user account

elevation is a piece of data which should be hidden from the user at all times since modifying its value would allow users to gain privileges they should not have. So our components utilise the account elevation variable in a way that is invisible to the user through a combination of using local variables that only exist in code or through other similar means.

For example, when visiting the profile pages of other users, certain sensitive information is hidden such as the user's email - this information is however visible when viewing your own profile page. To determine whether or not a user is viewing the profile page of another user, and not their own profile page, we pass a type variable to the profile page component to indicate what type of profile page visit the user is carrying out. Originally we set up a profile page to store this type variable and its value in the URL of the profile page. This would allow the user to modify the URL to be able to view extra information when visiting another user's profile. To fix this, we changed our implementation to pass the type variable through a different means which is hidden from the user. But the main point is that design changes such as this example were necessary throughout our development to reinforce the integrity of our site.

2.2 Backend Design Decisions

2.2.1 Main Goals

There were essentially four properties we considered to guide our design and development of our backend server :

1. Performance : The server should be optimized for speed and efficiency to minimize latency for resource retrieval

2. Maintainability : Code should be organized to allow developers to easily identify portions of functionality to main and update various aspects of the server.

3. Extendability : The code should be designed to optimize modularity and reusability to allow coders to easily extend the implementation over time.

4. Security : The server should include a form of resource protection and user authentication to ensure a level of access control for sensitive resources.

As mentioned before, we use the ExpressJS framework which is well suited to support these properties. ExpressJS allows us to set up a modular routing configuration where we can group various categories of HTTP route handlers together in a single file. In our current configuration, we have an App.js file which serves to handle the routing of HTTPS requests to their proper route handler. We have three routing handling files in total :

1. Index.JS : handles all routing for our local (non-federation) instance's HTTPS requests. All requests sent from our own frontend are handled by this routing file

2. EightQueens.js : an extension for our Index.js file which handles all HTTPS requests sent from playing the Eights Queens game on our website.

3. Fedapi.JS : handles all routing for HTTPS requests used for federation interaction.

With this scheme we can redirect all HTTPS requests with a particular URL prefix to its appropriate routing file and separate the routing handling for various requests into their own isolated environment. For example, all routes with "/fedpi" appended to the HTTPS URL are directed to the Fedapi.js routing file for handling, This provides a basic structure to organize our routing. This allows for greater extendability and maintainability since our route handling functionality can be

more easily navigated and located through finding the appropriate route handling file for which the functionality belongs to. We set up each routing handling file to rather one dimensional as well to again improve extensibility and maintainability. Each route handling file essentially consists of a list of route handler functions for various types of HTTPS requests in a flat layout. Each HTTPS request is handled by its own function which keeps code modular the insular. This makes the logic of our code more easily understandable. Ideally there is a balance that must be obtained when it comes to reusability of the route handling code. Making the code too reusable makes the maintainability and extendability of the code more difficult since the code logic ends up being more complex to understand. Keeping all routes separate and atomic allows for developers to easily identify elements of functionality which in turn allows for easier debugging. For example, if there is an issue with our login HTTPS communication, a developer can identify all pieces of backend logic that pertain to login functionality by locating the individual routing function in the index.js file.

In regards to optimizing performance, our main point of focus surrounded balancing computation work between the backend server and the database. In general databases are very powerful for querying and filtering data due to the inherent nature of their reasoning engines. So for any resource retrieval involving retrieving sets of data objects with a particular filter applied (such as retrieving all puzzle objects that are sudoku), our backend logic utilizes MySQL database queries to structure and filter data as opposed to using javascript functionality. We in general tried to limit as much as possible the amount of computational work done in the backend server itself to optimize performance and to keep code clean. In this way, our code overall is more organized since we know that all route handling logic resides in the backend server files (the route handling files in particular such as index.js or fedapi.js) and that all data processing and filtering logic resides in the database.js file which is responsible for executing queries to the database.

Our design choices in regards to security are outlined below in the section 2.2.2 below.

2.2.2 Resource Authentication & Protection

Our website uses a JSON Web Tokens (JWT) system that generates a unique token to each user by hashing the users credentials with a secret key which is stored in a .env file that remains only on local repositories for security. This token - along with a matching refresh token - can only be retrieved by logging in to an account and copying it from local storage which means there is no way for someone else to pretend to be another user without logging in to their account and taking the token themselves. These tokens are used for all privileged interactions with the database, such as: saving progress, creating puzzles, undergoing administrator only actions like deleting other people's comments and interacting with puzzles. This overall means that all user information is completely safe from other users finding it or accessing resources and functionalities that they should not be able to. In all cases where local storage is used - other than the tokens - it is only for the purpose of displaying information so even if someone were to edit the username or email in localstorage then they still will not be able to undergo any tasks under the guise of being a different user.

With our current token authentication scheme, we can append token validation functions to all route handling functions in our routing files which essentially protect the routes against unauthorized access. With this scheme, all HTTPS requests sent to our server for privileged resource retrieval must first be validated by the token validation function in order to access the routing handling functions core logic.

2.2.3 Database interaction

All interactions with the database are handled by the server to prevent users from accessing sensitive information or modifying the database in unintended ways with a modified client. Each time the client needs something from the database, it sends a POST request to the server which then,

based on the route, will deal with the request and send the client everything it needs. See beneath each route and how the database is used and how each route works.

Route	Explanation
/signup	First begins by using a database query that checks if the username and email are valid and have not been used before. If it is valid then it adds the new user to the database and returns the new user.
/login	Attempts to query the database to receive a user and returns the user if they exist, otherwise it returns an error message.
/userdata	Queries the database for a user by username and returns the user object.
/searchusers	Queries the database to find usernames of users matching a regular expression. Returns a list of user names and avatars
/paginatedpuzzle	Queries the database to find puzzles based on a pagination filter. Returns all of the puzzles.
/puzzleprogress	Queries the database to get either the puzzle or the user's progress on the puzzle if they have any. Returns a puzzle object.
/puzzle	Queries the database to get a fresh puzzle and ignore the user's progress.
/savepuzzleprogress	Queries the database to create or change a user's save state on a particular puzzle.
/addPuzzle	Queries the database to create a new puzzle based on a given puzzle object and returns the puzzle id so that the user can be redirected to the new puzzle page.
/deletePuzzle	Queries the database to delete a puzzle.
/getElevation	Queries the database to retrieve the elevation of a user $\sim$ useful for preventing local storage tampering.
/getLevel	Queries the database to retrieve the experience points that a user has $\sim$ useful for preventing local storage tampering.
/addRating	Queries the database to add a puzzle rating.
/getRating	Queries the database to get the rating and returns it.
/addComment	Queries the database to add a comment.
/removeComment	Queries the database to remove a comment.
/getcomments	Queries the database to get all comments belonging to a particular puzzle.
/submitPuzzle	Queries the database to check if a solution is valid and handles updating their save data to tell the system that they completed the puzzle and also awards experience points.
/solvepuzzle	Takes in a sudoku puzzle and generates a solution for it using the auto solver, this solution is returned or an error message is given if there is more than one solution or no solutions.
/changePfp	Queries the database to update the avatar field in the users table to a given URL.
/upload	Prompts a series of functions to prepare an image file, store it in the uploads directory and return the URL of the new image.
/generateRandomSudoku	Runs the sudoku generator and returns a valid puzzle.

2.2.4 Automated Sudoku Generator & Solver

To ensure that every sudoku published on the website has exactly one solution, we developed an automated sudoku solver. The solver uses backtracking to read in a puzzle and fill in all the cells without breaking any rules. For every cell that needs to be filled, there are nine possibilities (numbers 1-9). The solver starts filling in numbers from the element in the first row and the first column to the element of the last row and last column. If a conflict arises after a cell is filled, the

solver redoes the cell and tries another number. If the element in the last row and last column is filled, the sudoku is solved. However, to determine if a sudoku has a unique solution, the solver counts the number of solutions instead of returning the solved sudoku. If the solver finds a second solution, it stops solving the puzzle and returns a message to the frontend indicating that the puzzle has multiple solutions. If no solution is found after trying all possible combinations of numbers in all cells, the solver returns a message to the frontend that the puzzle has no solution. If the puzzle has a unique solution, the solver stores the solution into the database and sends a message that the puzzle has been created successfully.

Since creating sudoku puzzles with only one solution is not an easy task for humans, we also developed an automated sudoku generator to assist in creating valid sudokus. The generator first randomly fills in 10-20 cells and checks for any conflicts. Then, it uses the automated solver to generate a complete sudoku board. Afterward, the generator creates a random array consisting of all the indices and removes 30-50 cells from the complete board following the order of indices in the array. After removing each cell, the generator checks that the sudoku still has a unique solution. Finally, the generator returns a random sudoku puzzle that has a unique solution.

2.3 Database Design Decisions

2.3.1 Schema

For storing our database, we decided to use MySQL which can interact with the university's instance of MariaDB (a platform that allows the execution of MySQL code and also gives you command line access to the stored tables) which allows the host server to have access to the database. The schema we ended up with, which has one less table than our MVP due to puzzle_creation being unnecessary, includes 4 tables which you can see in the Relational Diagram in figure X. In our database, we are storing: the user data, puzzle data, the user's save progress (user_puzzle), and comment data.

Upon server startup, the tables are generated (if they are missing) and the connection is set up to the MariaDB instance that is hosted by the university. These tables are then used in future interactions that require access to the database (see 2.3.3). One thing to note is that user generated images are not stored in the same way as the rest of the data. The user images are stored directly on the server in a folder ("/uploads") which has access open to the client so that users can quickly load images that they need to see on profile pages and puzzle pages which is more convenient for users and prevents the need to encode images in a format suitable for the database. The database itself - under the avatar column in users and puzzle image in puzzles, simply stored the url of the image which the client can use to load the full image itself. This also allows moderators to delete any images that may be inappropriate without interacting with the database and can just delete it visually from the uploads file - which would not be possible if the image were to be encoded in a format unreadable to humans.

The way the database interactions work has been discussed in the backend design decisions section under 2.2.3.

2.3.2 Normal Form

Overall, our database is arguably in third normal form (3NF). Third normal form is when the database is when all non-key attributes are non-transitively dependent on the primary key (meaning that there are no non-key attributes that depend on other non-key attributes) and that the database is already in second normal form (2NF). Second normal form is when all non-key attributes depend on the whole of the primary key (i.e. a column in the comments table that depends on the user but not the comment would not be in 2NF because it does not rely on the whole key) and that the

database is already in first normal form (1NF). Finally, the first normal form is when all attributes are atomic (meaning no multi-valued fields). While our puzzle table stores an array, we can argue that it is still atomic because the puzzle itself is a single entity and not a list of separate entities - there is also no other reasonable way of representing this information.

Table	Primary Key(s)	Explanation for 3NF
users	userID	All attributes depend on the primary key and no partial dependencies.
puzzles	puzzleID	All attributes depend on the primary key and no partial dependencies. As mentioned earlier, the puzzles field is technically one single object so it is arguably still atomic.
user_puzzle	(userID, puzzleID)	All attributes depend on the whole of the composite primary key and no partial dependencies.
comments	commentID	All attributes depend on the primary key and no partial dependencies.

2.4 Supergroup Interaction

We implemented the supergroup protocol following the standard of OAuth2. In the following diagram, the protocol flow is explained given the scenario that a user who is registered with group B wants to access group A. The user interacts in this flow via their web browser, and the protocol itself makes extensive use of HTTP redirections. When accessing the login page on the website of group A, the user indicates login via group B. The web application of group A redirects the user to the authorization server of group B such that the user can login using the credentials form group B. If the user logs in successfully, the authorization server of group B redirects the user to group A and sends an authorization code to the backend of group A indicating that the user has been authorised. The backend of the group A application uses the authorization code together with the client secret of group A, i.e. the identifier of group A to exchange the access token from the backend of group B application. Upon receiving the correct authorization code and client secret from group A, group B issues the access token to the backend of group A. Group A then uses the access token to request the basic info of the user, such that it can create a profile for the user. Note that this process is mainly done on backend apart from authenticating the user, which maximises the security of access tokens.

The supergroup protocol, which we implemented following the standard of OAuth2, allows users registered with one group to access another group's website. The following diagram explains the protocol flow for a user registered with group B who wants to access group A:

1. The user accesses the login page on group A's website and selects "login via group B."

2. Group A's web application redirects the user to the authorization server of group B, where the user logs in using their credentials for group B.

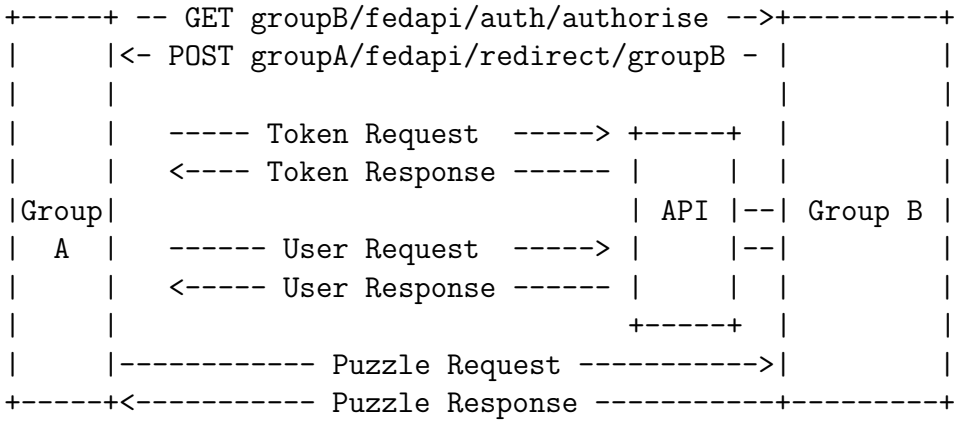
3. If the user logs in successfully, the authorization server of group B redirects the user back to group A and sends an authorization code to the backend of group A indicating that the user has been authorized.

4. Group A's backend uses the authorization code and the client secret (i.e., the identifier) of group A to request an access token from the backend of group B's application.

5. Upon receiving the correct authorization code and client secret from group A, group B issues the access token to the backend of group A.

6. Group A's backend uses the access token to request basic information about the user from group B's backend, enabling group A to create a profile for the user.

Note that most of this process is done on the backend, ensuring maximum security for access tokens.



To access the sudoku puzzles from group B, group A sends a request to group B for the required puzzle data. As puzzle data is not sensitive or related to privacy, there is no need for authentication within the federation. Group B processes the request and responds to group A with the requested sudoku puzzles.

The supergroup has established a standard schema for exporting/importing sudokus in JSON format. The required attributes that must be included are:

1. sudoku_id (unique identifier within the group)
2. author_id (unique identifier within the group)
3. puzzle (a 2-layer matrix, where each element is a string consisting of one number)
4. solution (same format as puzzle)

There are also optional attributes that can be included:

1. difficulty (an integer from 1 to 3 indicating easy to hard)
2. character_set (the set of all the characters used in the puzzle)
3. rating (an integer from 1 to 5)
4. date_created
5. title
6. description

All participating groups in the supergroup must adhere to this schema to ensure compatibility and ease of data exchange. For a detailed demonstration of the protocol and sudoku schema, please refer to [12].

2.5 Infrastructure Design Decisions

We are deploying and serving our application user docker. Utilizing docker allows us to turn our applications into "containers" that can be run anywhere, so we don't need to have our build configured to run on a specific operating system or hardware, and also allows us to run our application persistently on the school servers. After building our server, we run it using which allows us to deploy changes without having to restart the server. After building our client, we use the serve library from node package manager (NPM) to serve the HTML and CSS.

Chapter 3

Agile & Scrum

Scrum introduction:
"The Agile Manifesto:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan" [2].

Scrum is an Agile project management framework emphasizing iterative development, continuous improvement, and team collaboration. It is commonly used in software development but can be applied to any project or organization requiring a flexible project management approach. According to [9], Scrum is a framework that enables people to work productively and creatively to produce products while addressing complex adaptive issues. Scrum is a straightforward structure for successful teamwork on complicated products. Agile methodologies advocate for team members to be largely in charge of managing their own processes, choosing how project goals will be met, and assigning tasks to various team members [5]. A scrum sprint is a time-constrained iteration of development work where a cross-functional team attempts to deliver a releasable product increment. Sprints typically last between 1 and 4 weeks. During the sprint, the team collaborates to plan, execute, and review their work. Sprints are the backbone of the Scrum framework, and they provide a regular rhythm for teams to plan, implement, and inspect their work. By working in short iterations, teams can adapt to changing requirements, improve their process, and deliver value to the project's stakeholders more frequently.

The Product Owner, the Scrum Master, and the Development Team are the other three crucial responsibilities that Scrum outlines. Within the scrum structure, each of these positions serves a particular function.

1. Product Owner: The Product Owner oversees increasing the worth of the final product and the development team's work. They have the power to decide what work the Development Team will do because they are the main stakeholders in the product. The Product owner oversees keeping the product backlog current, setting work priorities, and ensuring the team focuses on the most critical tasks[9].

2. Scrum Master: The Scrum Master oversees the team's adherence to the Scrum structure and facilitates the Scrum process. They said the team understands and embraces the Agile mentality, eliminates any hindrances to their progress, and shields them from outside distractions. Additionally, the Scrum Master monitors the smooth operation of the Scrum activities (such as Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective) [9].

3. Development Team: After each Sprint, the Development Team delivers a possibly marketable product update. They are a self-organizing, cross-functional team collaborating to produce the product increment. The Development Team manages their work, including planning, carrying out, and reviewing the work's quality [9].

Pros and Cons of Scrum and Agile methodologies:

- Pros:

    1. Scrum collaboration is more effective than "divide and conquer." Scrum might seem time-consuming at first, but Scrum produces a better product in the end [8].
    2. Agile methodologies allow software development teams to quickly adapt to any needs or changes that may arise throughout a project [5].
    3. Agile methodologies facilitate interactions among group members and encourage self-managing teams [11].

- Cons:

    1. Agile methodologies pressure software development teams to produce functional software units in brief iterations because they have short cycle times.
    2. If given complete freedom, various developers' objectives may be very divergent, preventing development efforts from converging to improve the quality of software projects [5].

3.1 Scrum in the first semester

In the first semester, we established that we assigned story points based on one hour per story point. Here is how we set our story points in the first semester:

User Stories	Story Points	Estimated Time (hours)	Actual Time (hours)	Responsibility
LOGIN FUNCTIONALITY				
setup login token authentication back-end	2	2	8	190015412
add hook to determine if user is logged in at home page - request resources from back end if logged in	1	1	1	190015412
Setup up login token authentication Back-end	2	2	4	190015412
Create React Registration page	2	2	4	200006295, 190015412
Create React Login Page	1	1	3	200006295, 190015412
Receive user data from the front end and store them into the database created	1	1	2	200018293, 200014911
Create DB Table for Users	0.5	0.5	0.5	200018293, 200014911
Configure DB to run faster				200014911
Backend User Login & Authentication				190015412, 200006387
Backend User Registration				190015412
Set up front end to store user data and token in local storage after successful login	1	1	1	190015412
SIGNUP FUNCTIONALITY				
Create Signup page layout	1	1	2	200006295, 190015412
Add signup validation (all fields filled out, valid email, secure password, Username does not exist)	1	1	1	200006295, 190015412
Set up back end route for validating and storing user credentials	3	3	3	190015412
Set up token generation	1	1	1	190015412
Set up database query function for validating and checking users	2	2	2	200014911
Set redirect to home page after signup	0.5	0.5	0.5	190015412
set up signup page to put user data in local storage after sign up	1	1	1	190015412
DEFAULT PUZZLE				
Retrieve and produce puzzles from the database by difficulty in a descending order	2	2	2	200014911, 200018293
Backend puzzle create route	1	1	2	190015412
Create React Puzzle Solve page	2	2	4	190015412 200006295
Create DB Table for Default Puzzle	2	2	3	200014911, 200018293
FRONT END : implement content belt for puzzles (puzzle thumbnails)	2	2	3	200006295 190015412
FRONT END : Create puzzle thumbnail objects (to be displayed in content belt etc.)	1	1	1	200006295 190015412
FRONT END : add reset button for resetting entire puzzle and resetting an individual cell	1	1	1	190015412
Add ability save puzzle creation and solving progress	1	1	3	190015412
Add finish solving functionality (once a solution has been	2	2	3	190015412

filled and submitted, record the puzzle as a solved puzzle for the user)				200014911, 200018293
Setup puzzle to get data from params(url) instead of props	1	1	1	190015412
Add the function to check the solution validity from the backend	1	1	3	200018293
Add a button to copy puzzle URLs	0.1	0.1	0.2	200014911
PROFILE PAGE				
Create Front End Layout And Styling	2	2	5	200006295 190015412
Add Content Belt for continue solving	1	1	1.5	200006295
Add access token expiry check				190015412
add create puzzle button				200006295
Add solved puzzles content belt				190015412, 200006295
Add logout button and implement logout functionality	1	1	1	190015412
FED OAUTH HOME LOGIN (logging in from other sites with our credential)				
Create Fed Login Page	1	1	4	190015412
Set up auth/authorise route in back end	2	2	4	190015412 200006387
Set up auth/token route in backend	1	1	3.5	190015412 200006387
Set up /user route in backend	1	1	2	190015412 200006387 200014911
Set up redirect in fedlogin page	0.5	0.5	0.5	190015412
FED OAUTH TARGET LOGIN (logging into our site with credentials from other sites)				
Set up /auth/redirect route in backend	2	2	3	190015412 200006387 200014911
Setup Login page to redirect member to fed group login page with foreign group number specified (with necessary params)	1	1	1	190015412
Add handling in home page to detect fed group redirect after fed group authentication (importing federation user details from params)	1	1	1	190015412

Figure 3.1: Source: [1]

In the first semester, we intended to structure the development process and schedule the project using the Scrum and Agile frameworks. However, putting this plan into action confronted us with numerous challenges. Due to the lengthy planning process, the project's pace accelerated only two weeks before the deadline. Due to the strain of the approaching deadline, it took a lot of work to hold Scrum meetings before each sprint session. We soon exclusively focused on coding sessions and ceased having Scrum meetings[3].

One of the challenges we faced was with the Scrum meetings. Because of the delays in planning, we frequently needed to remember to hold the discussions that were supposed to be held periodically to track the project's progress. This led to a breakdown in team collaboration, slowing the project's advancement. Furthermore, there needed to be more accountability among team members because there needed to be a clearer understanding of who oversaw what duties because there were no regular scrum sessions [3].

The other challenge was accurately estimating the story points for each user story. This was partially due to inexperience, but it was also a result of difficulties establishing the fixed time for sprint sessions. Setting aside some user stories for later sprints was difficult because many were requirements to be fulfilled. This resulted in rigidity, which made it difficult for us to adapt to the changes in our plans [3].

Also, we distributed our user stories unevenly. We assigned two people to work on the front-end side of the user stories, one on OAuth user stories and two on the database side of user stories. However, this work distribution resulted in lower efficiency due to the unexpected difficulty of OAuth. One of the two group members from the front end eventually moved to OAuth user stories to help balance the workload. One person could do the database, but we decided to have two people because it is one of the essential aspects of the program. In hindsight, allocating two people to OAuth from the start might be beneficial.

Regarding the sprint sessions, we had to skip the meeting at the beginning of each sprint session because we thought it would be faster if we jumped into coding immediately. Although intuitively, it seemed like it would be more efficient, we encountered some problems where two group members would be working on the same user stories or the same functions, which dragged down our efficiency.

So, in essence, our Scrum scheme in semester one was not strict enough to keep us informed on progress, but also our usage of our Scrum system was not quite flexible enough. We need a consistent structure to keep all members informed on aspects such as progress, time estimates, and responsibilities while also not adding too much to our Scrum structure where adhering to the process creates unnecessary overhead. The Scrum structure we designed in semester one was far too complex for our use case, and all of the bureaucratic Scrum 'ceremonies' discouraged us from consistently sticking to the organizational scheme. Finding a balance of including enough aspects to our Scrum scheme to keep us informed on our development while not adding too much minutia is essential. This also raises the question of when it is best to use SCRUM. Scrum.org argues there is a middle ground between simple and chaotic situations where SCRUM is ideal. At one end, if the situation is too simple, the SCRUM processes and ceremonies are not necessary and add excessive overhead to the workflow. At the other end, when the situation gets too chaotic, where short-timed act-and-respond approaches are necessary to adapt to an inflow of unexpected circumstances, Scrum is not as beneficial as it's focused on planning around fixed sprints [14]. I believe these aspects affected our group and can help explain why we felt and acted in the way that we did. The next section outlines the changes we made to our Scrum scheme and its usage to improve our development.

3.2 Scrum in semester 2

Comments & Ratings: priority MED

Task	Difficulty	Completed?	Responsibility
Add frontend implementations for puzzle ratings	**MEDIUM**	**yes**	**200018293, 190015412**
Add server implementation for updating comments and puzzle ratings	**EASY**	**yes**	**200018293 20001491**
Add DB implementation for puzzle ratings	**MEDIUM**	**yes**	**200018293**
Add comments/forums front end modal	**MEDIUM**	**yes**	**20001491**

Import/ Export: priority **HIGH**			
Task	**Difficulty**	**Completed?**	**Responsibility**
Implement Import and Export functions	**MEDIUM**	**yes**	**20001491**

Design changes: priority **HIGH**

Task	Difficulty	Completed?	Responsibility
Add pagination for retrieving puzzles	**HARD**	**yes**	**190015412**
Change to more conventional sudoku board	**MEDIUM**	**yes**	**190015412 20001491 200006295**
Fix comments input - atm the puzzle focus prevents input in the comments box such as putting spaces	**EASY**	**yes**	**200006295 20001491**
Persistent light or dark mode	**MEDIUM**	**yes**	**200006295**
Add sorting for content belt	**MEDIUM**	**yes**	**190015412**
Fix colour theme styling bugs (dark/light mode)	**MEDIUM**	**yes**	**200006295**
Add colour theme preference to local storage for user	**MEDIUM**	**yes**	**200006295**

Adding border styling	**MEDIUM**	**yes**	**20001491**

User XP: priority **Medium**

Task	Difficulty	Completed?	Responsibility
Backend xp calculation and support	**MEDIUM**	**yes**	**200018293 20001491**
XP front end features and unlocks	**MEDIUM**	**yes**	**20001491**

Puzzle timer: priority **Low**

Task	Difficulty	Completed?	Responsibility
Front-end timer	**MEDIUM**	**no**	**n/a**
Leaderboard system	**MEDIUM**	**no**	**n/a**

MVP bug fixes: Priority **High**

Task	Difficulty	Completed?	Responsibility
Fix retrieving puzzles on other sites	**MEDIUM**	**yes**	**200006387 190015412**
Fix server crashing when other sites retrieve our puzzles	**MEDIUM**	**yes**	**200006387 190015412**
Add content belt for puzzles from other sites	**MEDIUM**	**yes**	**190015412**
Move solution checking to end of puzzle solving	**EASY**	**yes**	**200006295**
Fix the bugs in save buttons and submit buttons and collaborate them with the additional puzzles	**MEDIUM**	**yes**	**200018293**
Fix access token expiry detection	**MEDIUM**	**yes**	**190015412**
Fix incorrect indicators when inputting solution twice	**MEDIUM**	**yes**	**200006295**
Fix bug where front end recognizes saved user inputs as part of the default puzzle	**MEDIUM**	**yes**	**200006295**

Additional Puzzles: Priority **High**

Task	Difficulty	Completed?	Responsibility

Add the show/hide numbers to eights button	**MEDIUM**	**yes**	**20001491**
Front end game page EIGHTS	**MEDIUM**	**yes**	**20001491**
Front end game page QUEENS	**MEDIUM**	**yes**	**200006295**
Backend queens	**MEDIUM**	**yes**	**200006387**
Add DB implementation	**MEDIUM**	**yes**	**200018293 20001491**
Puzzle Thumbnail	**EASY**	**yes**	**20001491 190015412**
Puzzle modal	**HARD**	**yes**	**20001491**
Add image upload functionality for eights puzzle	**MEDIUM**	**yes**	**20001491**
Add content belt on home page and profile	**MEDIUM**	**yes**	**190015412**

Gameplay features: (includes undo redo and keyboard controls)
Priority **High**

Task	Difficulty	Completed?	Responsibility
Keyboard controls	**MEDIUM**	**yes**	**200006295**
Undo / Redo buttons	**MEDIUM**	**yes**	**200006295**
Auto solver	**HARD**	**yes**	**200006387**
Random puzzle generator	**HARD**	**yes**	**200006387**
Eights toggles	**EASY**	**yes**	**20001491**

Search bar:
Priority **Low**

Task	Difficulty	Completed?	Responsibility
Implement front end endpoint for communication search input to backend	**MED**	**yes**	**190015412**
Implement search bar front end input	**EASY**	**yes**	**190015412**
Implement Back end search route handling and database querying	**MED**	**yes**	**190015412**

Account Admins:
Priority **High**

Task	Difficulty	Completed?	Responsibility
Add moderator accounts that allows puzzle and comment deletion	**MEDIUM**	**yes**	**200018293**
Allow users become author	**MEDIUM**	**yes**	**200018293**

In semester 2, we made sure that we followed the Scrum framework as closely as possible. One of our main goals was to find a scheme that included all of the basic and required ceremonies of Scrum to keep us informed on our development while ensuring that the scheme was easy to stick to. We scheduled five Sprint sessions, each lasting for a week. Also, we had weekly Scrum meetings to discuss how we would proceed with the project. Before each sprint session, we briefly discussed the tasks each member was responsible for in the sprint session. With the sprint meetings in each sprint session, we guaranteed each member knew exactly what they were doing and clarified the work allocation. Because of that, we did not make the same mistake as the first semester, where two members worked on the same task and implemented redundant functions.

We decided to ditch the 10-point user story system in favour of a 3-tier easy, medium, and hard difficulty rating system. With this new rating system, multiple easy tasks can be completed within a single coding session, medium tasks can require multiple days/coding sessions to complete, and hard tasks can take up to a whole sprint to complete. We additionally implemented a triaging system where we assigned tasks a priority rating of low, med, and high. With this priority rating system, the high-priority user stories were the compulsory features of the project requirement, such as additional puzzles. The medium-priority user stories were the other features that were not compulsory but additional features we decided were essential for our project, such as user administration and the user experience (XP) system. The low-priority user stories were the user stories that would be great if they were in the project, but should only be attempted after the medium-priority tasks are finished and we deem that there is sufficient time remaining. This new difficulty and priority rating system was easier to visualize and make estimates for since we are all still not experienced enough to correctly estimate accurately the amount of time it takes to complete a given task - which we learned from semester one. Each user story consisted of multiple tasks required to achieve the user story.

In every Scrum meeting, we followed up with each group member about the task they oversaw and checked on their progress or if they needed any assistance. Like last semester, we also utilised a Trello board to keep track of the user stories and the members responsible. However, this semester, we switched people each week to update the Trello board so that everyone got to voice their opinions and concerns.

We split every sprint session down into multiple coding sessions. In each session, we had our entire group meet in the computer lab to simplify communication. Although members sometimes had other commitments that kept them from attending every coding session, we made sure to keep each member updated on each sprint session through WhatsApp. We planned for five or six sprint sessions but ended up only doing four sprint sessions, with the pressure of the practicals from other compulsory computer science modules. Yet, we made these four sprint sessions count and finished the project efficiently.

In conclusion, we were unfamiliar with the Scrum Framework in the first semester and struggled to adhere to it closely. In contrast, in the second semester, we were more comfortable with the structure, making it easier for us to put the Scrum framework to practise. Even though we did change some of the practices of Scrum, such as the user story points, we adhered to the Scrum practice closely to facilitate group communications and increase our efficiency.

Chapter 4

Evaluation and Critical Appraisal

Overall we are pleased with the state of our project. When it comes to front-end, we've managed to produce an aesthetic and intuitive application, with complete screen-size responsiveness and cross-browser compatibility. The design of our front-end is cohesive and while there is still some room for improvement when it comes to frontend performance, the overall performance of the site is more than satisfactory and performs fine on even low power mobile devices. We are similarly pleased with the state of our server, which is robust with good code quality. Deciding not to use a library for authentication and implementing it manually instead was definitely an added overhead when it came to work, however not using an authentication library (like PassportJS) gave us more granular control and this approach ensures any updates made to the library won't break our authentication system. The only known bug on the frontend of the application is the downward facing chevron in the ¡select¿ not switching colours appropriately in dark mode.

The following table will detail each feature and appraise our implementation.

Feature	Status	Appraisal
Login/Signup		
Users must be able to sign up for a new account.	Complete	Users are able to successfully sign up when entering valid credentials. On signup, if their credentials are valid, they are sent to the backend for database verification and redirected to the home page.
Users must be able to log in with a created account.	Complete	Users are able to log in with a valid account and are redirected to the home page.
Users should not be able to log in with a username that doesn't exist	Complete	The login form displays an appropriate error when the user inputs a username that doesn't exist.
Users should not be able to log in with the incorrect password	Complete	The login form displays an appropriate error when the user inputs the incorrect password.
Users must be able to show/hide their password when signing up or logging in.	Complete	There is a familiar eye icon that can be clicked to toggle whether or not the password is hidden. The toggle also changes visually to indicate its state.
Users should not be allowed to create accounts where the name has anything other than characters A-Z (irrespective of case).	Complete	The sign up form displays an appropriate message if the user tries to input illegal characters in the name field.
Users should not be allowed to create accounts where the email is not a properly formatted email	Complete	The sign up form displays an appropriate message if the user tries to input an email in an invalid format
Users should not be able to create a password less than 8 or greater than 32 characters, and passwords must have at least one number and one special character	Complete	The sign up form displays an appropriate message if the user tries to input a password that isn't strong enough
Users should not be able to create an account with a taken username	Complete	The sign up form displays an appropriate message if the user tries to sign up with a taken username
Users should be able to log out	Complete	The user is logged out and redirected to the home when they click the logout button.
Sudoku/Eights Playing		
Users must be able to input digits into the sudoku board with their	Complete	The user can click a cell to highlight it, and then click a digit

mouse.		in the input row to input a digit into a particular cell. They can erase this by inputting a blank character
Users must be able to input digits into the sudoku board with the number row on their keyboard, and navigate with the arrow keys.	Complete	The user can use their arrow keys to navigate the board and use the number row on the keyboard to input a digit. They can enter a blank space by pressing 0
User should be able to undo and redo using the buttons		The user can undo and redo all changes made in the current browser session by clicking undo or redo
Users should be displayed an error message upon submitting an incorrect solution.	Complete	The puzzle page displays “invalid solution” in red upon submitting an incorrect solution.
Users should be displayed a success message upon submitting a correct solution.	Complete	The puzzle page renders a modal which blurs the background and displays a success message, along with rendering some confetti to give the user a sense of achievement.
Puzzle should be cleared and moved into “solve again” upon submitting a correct solution.	Complete	The puzzle gets cleared and upon navigating to the home page, appears in the “solve again” content belt.
Users should be able to save progress when they click save.	Complete	The puzzle progress is saved when the user presses the save button and persists after the user reloads the page or navigates elsewhere
Puzzle should be moved into “continue solving” when progress is saved.	Complete	The puzzle appears in the “continue solving” content belt upon navigating to the home page if the user has saved any progress.
Eight tile should move when clicked by user if spot next to it is empty	Complete	When clicking on a tile, the tile slides into the correct place if it is allowed to do so. They can also toggle this to use a second click if they feel like it is more intuitive.
Eight’s page should display solution when view full image is clicked	Complete	Upon hover or click (to lock it) of the view full image button, the solution image is shown so that the user can see the goal.
Comments		
Users should be able to submit comments by pressing either the send button or the enter key on their keyboard.	Complete	Users can submit comments using either the enter key on their keyboard or clicking the button. The comments are displayed instantly with a human readable time format and the comment input does not allow users to input malicious code (XSS safe).
Users should be able to delete only their own comments with the delete button.	Complete	Users are only allowed to delete their own comments, this is authorised so users cannot delete other users’ comments regardless of what they attempt to do with local storage. Comments delete instantly from the comments column upon pressing the delete button.
Ratings		
Users should be able to rate puzzles	Complete	Users can rate the puzzle evne though they have not attempted the puzzle.
User Adminstrations		
Users can become moderators to delete puzzles and comments	Complete	Users can be assigned as moderators by the administrators and moderate the contents of the website.
Eight Queens		
Users should be able to click an empty cell to place a queen.	Complete	Clicking an empty cell places a queen in it.
Users should be able to click a cell with a queen to remove it.	Complete	Click a cell with a queen in it makes it empty.
Users should not be allowed to place more than 8 queens	Complete	Every subsequent click on an empty cell after the user has placed 8 queens does nothing. There is a counter above the board that displays the number of queens the user has left.
Users should be displayed an error message upon submitting an incorrect solution (conflicting queens or board not filled out).	Complete	The eight queens page displays “Board must have 8 queens to be solved” or “No queen must be able to kill another for the board to be solved” in red upon submitting an incorrect solution.
Users should be displayed a success message upon submitting a correct solution.	Complete	The eight queen page renders a modal which blurs the background and displays a success message, along with rendering some confetti to give the user a sense of achievement.
Users should be able to save progress when they click save.	Complete	When the user presses the save button, the state of the board is stored and allows them to resume whenever they want by opening the page again.

Users should be able to reset button with the solve again button	Complete	Solve again/ reset clears the board and clears the solved state in storage so that they can happily play again.
Create Puzzle		
Users should not be allowed to submit a puzzle without a name.	Complete	If the user tries to submit a puzzle without a name then they are prompted with an error message telling them what they did wrong.
Users should not be allowed to submit a puzzle with less than 17 clues	Complete	If the user tries to submit a puzzle with less than 17 clues then they are prompted with an error message telling them what they did wrong.
Users should not be allowed to submit a puzzle with no valid solution.	Complete	If the user tries to submit a puzzle with no valid solution then they are prompted with an error message telling them what they did wrong.
Users should not be allowed to submit a sudoku with more than one valid solution	Complete	If the user tries to submit a puzzle with more than one solution then they are prompted with an error message telling them what they did wrong.
Users should be allowed to generate a random valid sudoku with the generate button	Complete	Users can press the generate button and it will automatically fill in the board with a completely valid puzzle.
Users should be able to select what type of puzzle to create by choosing the respective option in the create puzzle page settings.		Users can select the type of puzzle that they want and the respective modal is shown to them so that they can make that kind of puzzle.
Users of certain levels should be allowed to choose from colours and gradients to style their sudoku titles	Complete	Users can unlock styling features as they level up and they can view how to unlock each item by hovering their mouses over the style they want.
User Generated Content		
Users should be able to upload a png or jpg formatted image for profile pictures and the eights puzzles	Complete	Users can easily change profile pictures and upload custom images when they make an eights puzzle.
Users should be able to upload a png with a blank background and have it automatically be filled in and converted to a jpg	Complete	Users can upload any image in the png format and it is converted to a jpg and stored in the server. They can also access the original png version if they so please.
Admins should be able to delete puzzles with inappropriate images	Complete	Moderators and admins can see a delete puzzle button on the puzzle page which allows them to moderate them as appropriate.

Chapter 5

Conclusions

In summary, this project has been largely successful both in implementing the assigned task and in providing a learning experience in employing Agile methodologies to develop a large-scale application in a team-based setting. This project has shed light on the processes of how to organise the development of an application between team members in an incremental development progression. In general, we ran into both technical and organisational obstacles throughout our development, and each obstacle has provided a valuable learning experience to guide us towards improvement. While our project has iteratively improved through an incremental development structure, so has our own understanding of software development and its methodologies.

Ultimately we have implemented a robust and intuitive platform for supporting the specified task for the project. Through the use of containers, our application is portable and can be deployed anywhere. Our emphasis on ergonomics and human computer interactivity ensures a smooth user experience on all devices which maximises our ability to cater to a broad user base. In terms of technologies, this project has served as formative experience into the dynamic of learning and picking up languages and technologies quickly on the go. So not only have we improved our proficiency with the languages we opted to use, but additionally we have gained insight into how to learn and improve our understanding of these technologies efficiently.

Moving forward, we would not only like to improve the core functionality that our platform offers, but to also improve other infrastructure related aspects. This includes setting up a continuous deployment pipeline of some sort, implementing an automated testing protocol with our continuous deployment pipeline, and reinforcing our docker usage to be more robust against errors and changes.

Bibliography

[1] CS3099 Group 15. *CS3099 Deliverable 2 Report*. Nov. 2022.

[2] Agile Alliance. *Agile Manifesto for Software Development — Agile Alliance*. Dec. 2018. URL: https://www.agilealliance.org/agile101/the-agile-manifesto/.

[3] Binyi Li. *CS3099 Deliverable 3 Reflective Essay*. Feb. 2023.

[4] Sam Magura. *Why we're breaking up with CSS-in-JS*. Oct. 2022. URL: https://dev.to/srmagura/why-were-breaking-up-wiht-css-in-js-4g9b.

[5] Likoebe M. Maruping, Viswanath Venkatesh, and Ritu Agarwal. "A Control Theory Perspective on Agile Methodology Use and Changing User Requirements". In: *Information Systems Research* 20.3 (2009), pp. 377–399. URL: https://www.jstor.org/stable/pdf/23015471.pdf?refreqid=excelsior%5C%3A6a4c26be4d6cc32d65e3a4b5a383ac3f&ab_segments=&origin=&initiator=.

[6] *Nielsen's Heuristics: 10 Usability Principles To Improve UI Design*. URL: https://aelaschool.com/en/interactiondesign/10-usability-heuristics-ui-design/.

[7] Aris Pattakos. *Angular vs react vs Vue: Which framework is better? 2023*. Jan. 2023. URL: https://athemes.com/guides/angular-vs-react-vs-vue/.

[8] Rebecca Pope-Ruark. "We Scrum Every Day: Using Scrum Project Management Framework for Group Projects". In: *College Teaching* 60.4 (2012), pp. 164–169. URL: https://www.jstor.org/stable/pdf/23525113.pdf?refreqid=excelsior%5C%3Ade0ff11b9d0a570d4623816d5c75c42e&ab_segments=&origin=&initiator=&acceptTC=1.

[9] Ken Schwaber and Jeff Sutherland. *The Scrum Guide the Definitive Guide to Scrum: the Rules of the Game*. 2020. URL: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf.

[10] *Start a new react project*. URL: https://react.dev/learn/start-a-new-react-project#can-i-use-react-without-a-framework.

[11] David S. Sweeney and Lauren Cifuentes. "Using Agile Project Management to Enhance the Performance of Instructional Design Teams". In: *Educational Technology* 50.4 (2010), pp. 34–41. URL: https://www.jstor.org/stable/pdf/44429839.pdf?refreqid=excelsior%5C%3A5b93d1d54b7affeacbcde5fc326c2a0a&ab_segments=&origin=&initiator=.

[12] *Tens Supergroup API*. URL: https://cs3099user11.host.cs.st-andrews.ac.uk/swagger/api-docs/#/.

[13] *What Percentage of Internet Traffic is Mobole?* URL: https://www.oberlo.com/statistics/mobile-internet-traffic.

[14] Stefan Wolpers. *When Is It Time to Stop Using Scrum?* Oct. 2022. URL: https://www.scrum.org/resources/blog/when-it-time-stop-using-scrum.

Appendix A

Scrum Sprint Notes

This section outlines the notes from each of the sprint meetings we carried out throughout semester two of our development.

Date	Notes
18/01/2023	Overview: New Stories: 1) Bug Fixes 2) UI Changes 3) Design Changes 4) Game Play 5) User Data 6) Additional Puzzle What was completed: Next Sprint
25/01/2023	Overview: - Still in pre-development planning phase New Stories/Tasks: What was completed - planning This Sprint - Bug Fixes - Design Changes Next Sprint
30/01/2023	Overview: - Started First Sprint New Stories/Tasks: - [Task] Add pagination and sorting to content belts for puzzles (Bug Fixes) What was completed: - Nothing yet

	This Sprint:
	- Bug Fixes
	  - [MED] Fix retrieving puzzles from other sites
	  - [MED] Fix server crashing when other sites retrieve our puzzles
	- Design Changes
	  - [MED] Add pagination for retrieving puzzles
	  - [Easy] Change to more conventional sudoku board
	- User Data Changes
	  - [EASY] Add user bio
	
	Next Sprint:
	- User Data Changes

05/02/2023	Overview:
	
	New Stories/Tasks
	
	What was completed:
	- [Easy] User data changes
	
	This Sprint:
	- Bug Fixes
	  - [MED] Fix retrieving puzzles from other sites
	  - [MED] Fix server crashing when other sites retrieve our puzzles
	- Design Changes
	  - [MED] Add pagination for retrieving puzzles
	  - [Easy] Change to more conventional sudoku board
	- User Data Changes
	  - [EASY] Add user bio
	Next Sprint:

12/02/2023	New Stories/Tasks
	
	What was completed:
	- Design Changes
	  - [Easy] Adding Border Styling
	  - [Easy] Fix Colour theme styling bugs
	  - [MED] Add sorting for content belt
	- Bug Fixes
	  - [MED] Incorrect indicator when inputting solution twice
	  - [MED] Fix bug where front end recognizes saved user inputs as part of the default puzzle
	
	Next Sprint:
	- History mode
	- Arrow navigation for sudoku
	- Update XP

19/02/2023	What was completed: - [MED] History mode: undo, redo - Arrow navigation for sudoku half done - Updating XP to local storage in profile page Next Sprint: - Additional Puzzle - Bug Fixes - User Administration
26/02/2023	All members entered hiatus due to a large number of deadlines from other modules.
02/03/2023	Reading week - we did not meet this week and took some time to rest.
09/03/2023	What was completed: - Additional puzzles - User Administration - Bug fixes Next Sprint: - Puzzle Pagination - Comments - Persistent Colour Theme - Keyboard inputs - Eights added - Eights features

15/03/2023	What was completed : - Puzzle Pagination - Comments - Persistent Colour Theme - Keyboard inputs - Eights added - Eights features Next Sprint: - Additional Puzzle - Moderator delete puzzles and comments - Logout on idle - Content belt for puzzles from other sites
19/03/2023	What was completed : - Additional Puzzle - Moderator delete puzzles and comments - Logout on idle - Content belt for puzzles from other sites Next Sprint: - Report - Final testing and quality checks
24/03/2023	What was completed : - [Easy] Change to more conventional sudoku board - Checking over all of our work - Testing - Quality checks - Commenting - FEDAPI finalised - Report Next Sprint: - N/A

Appendix B

Testing Summary

Test Case ID	user1	**Test Case Description**	Test if the user can create an account
Created By	200018293		

QA Tester's Log

Tester's Name	200018293	**Date Tested**	March 23, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme
4	

S #	Test Data Requirement
1	username = testme01
2	password = password
3	user email = testme@gmail.com
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	navigate to http://localhost:24600/Signup	site should open	As Expected	Pass
2	Enter the username and all the required fields	credentials can be entered	As Expected	Pass
3	click signup	should be able to sign up	As Expected	Pass

Test Case ID	user2	**Test Case Description**	Test if the user can log in
Created By	200018293		

QA Tester's Log

Tester's Name	200018293	**Date Tested**	March 23, 2023	**Test Case (Pass/Fail/Not Executed)**	

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme
4	

S #	Test Data Requirement
1	username = testme01
2	password = password
3	user email = testme@gmail.com
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	navigate to http://localhost:24600/Login	site should open	As Expected	Pass
2	Enter the username and password	credentials can be entered	As Expected	Pass
3	click login	should be able to sign in	As Expected	Pass

Test Case ID	user3	**Test Case Description**	Test if the user can log out
Created By	200018293		

QA Tester's Log

Tester's Name	200018293	**Date Tested**	March 23, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme
4	log in into one of the accounts

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:24600/Profile/?user=testme01	user can see the log out button	As Expected	Pass
2	click the log out button	user can log out	As Expected	Pass

Test Case ID	user4	**Test Case Description**	Test if the user can see the puzzles
Created By	200018293		

QA Tester's Log

Tester's Name	200018293	**Date Tested**	March 23, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme
4	log in into one of the accounts

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:24600/	user can see the puzzles	As Expected	Pass

Test Case ID	user5	**Test Case Description**	Test if the user can access the puzzles
Created By	200018293		

QA Tester's Log

Tester's Name	200018293	**Date Tested**	March 23, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme
4	log in into one of the accounts

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:24600/	user can see the puzzles	As Expected	Pass
2	click on one of the puzzles	user gets redirected to the puzzle page	As Expected	Pass

Test Case ID	user6	**Test Case Description**	Test if the user can save their progress
Created By	200018293		

QA Tester's Log

Tester's Name	200018293	**Date Tested**	March 23, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme
4	log in into one of the accounts

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:24600/	user can see the puzzles	As Expected	Pass
2	click on one of the puzzles	user gets redirected to the puzzle page	As Expected	Pass
3	start to solve the puzzle	user can solve the puzzle	As Expected	Pass
4	click on the save button	user can save	As Expected	Pass

Test Case ID	user7	**Test Case Description**	Test if the user can reset puzzle
Created By	200018293		

QA Tester's Log

Tester's Name	200018293	**Date Tested**	March 23, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme
4	log in into one of the accounts

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:24600/	user can see the puzzles	As Expected	Pass
2	click on one of the puzzles	user gets redirected to the puzzle page	As Expected	Pass
3	start to solve the puzzle	user can solve the puzzle	As Expected	Pass
4	click on the reset button	user can reset the puzzle	As Expected	Pass

Test Case ID	user8	**Test Case Description**	Test if the user can submit puzzle
Created By	200018293		

QA Tester's Log

Tester's Name	200018293	**Date Tested**	March 23, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme
4	log in into one of the accounts

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:24600/	user can see the puzzles	As Expected	Pass
2	click on one of the puzzles	user gets redirected to the puzzle page	As Expected	Pass
3	start to solve the puzzle	user can solve the puzzle	As Expected	Pass
4	click on the submit button	user can submit the puzzle	As Expected	Pass

Test Case ID	user9	**Test Case Description**	Test if the user can leave a comment
Created By	200018293		

QA Tester's Log

Tester's Name	200018293	**Date Tested**	March 23, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme
4	log in into one of the accounts

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:24600/	user can see the puzzles	As Expected	Pass
2	click on one of the puzzles	user gets redirected to the puzzle page	As Expected	Pass
3	comment on the puzzle	user can enter the comment	As Expected	Pass
4	click on the send button	user can send the comment	As Expected	Pass

Test Case ID	user10	**Test Case Description**	Test if the user can leave a rating
Created By	200018293		

QA Tester's Log

Tester's Name	200018293	**Date Tested**	March 23, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme
4	log in into one of the accounts

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:24600/	user can see the puzzles	As Expected	Pass
2	click on one of the puzzles	user gets redirected to the puzzle page	As Expected	Pass
3	rate the puzzle	user can enter the rating	As Expected	Pass

Test Case ID	puzzle1	**Test Case Description**	Test if the user can generate a puzzle once they get the access
Created By	200018293		

QA Tester's Log

Tester's Name	200018293	**Date Tested**	March 23, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme
4	log in into one of the accounts

S #	Test Data Requirement
1	username = test1005
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:24600/Profile/?user=test1005	user can see create puzzle button	As Expected	Pass
2	click on the create puzzle button	user gets redirected to the create puzzle page	As Expected	Pass
3	click on the generate puzzle button	user can generate puzzles	As Expected	Pass

Test Case ID	puzzle2	**Test Case Description**	Test if the user can generate a puzzle once they get the access
Created By	200018293		

QA Tester's Log

Tester's Name	200018293	**Date Tested**	March 23, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme
4	log in into one of the accounts

S #	Test Data Requirement
1	username = test1005
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:24600/Profile/?user=test1005	user can see create puzzle button	As Expected	Pass
2	click on the create puzzle button	user gets redirected to the create puzzle page	As Expected	Pass
3	click on the generate puzzle button	user can generate puzzles	As Expected	Pass
4	click on the save puzzle button	user can save and upload the puzzle	As Expected	Pass

Test Case ID	puzzle2	**Test Case Description**	Test if the user can access the federation puzzles
Created By	200018293		

QA Tester's Log

Tester's Name	200018293	**Date Tested**	March 23, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme
4	log in into one of the accounts

S #	Test Data Requirement
1	username = test1005
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:24600/	user can see create puzzle button	As Expected	Pass
2	scroll down to federation puzzles content belt	user can see the federation puzzles	As Expected	Pass
3	click on one of the puzzles	user gets redirected to the federation puzzle page	As Expected	Pass
4	start solving the puzzle	user can work on the puzzle	As Expected	Pass

Test Case ID	mod1	**Test Case Description**	Test if the moderator can delete puzzles (moderator's username and password are not shared here)
Created By	200018293		

QA Tester's Log

Tester's Name	200018293	**Date Tested**	March 23, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme
4	log in into a moderator account

S #	Test Data Requirement
1	username = moderator's username
2	password = moderator's password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:24600/	user can see the puzzles	As Expected	Pass
2	click on one of the puzzles	user gets redirected to the puzzle page	As Expected	Pass
3	click on the delete button	user can delete the puzzle	As Expected	Pass

Test Case ID	mod2	**Test Case Description**	Test if the moderator can delete comments (moderator's username and password are not shared here)
Created By	200018293		

QA Tester's Log

Tester's Name	200018293	**Date Tested**	March 23, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme
4	log in into a moderator account

S #	Test Data Requirement
1	username = moderator's username
2	password = moderator's password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:24600/	user can see the puzzles	As Expected	Pass
2	click on one of the puzzles	user gets redirected to the puzzle page	As Expected	Pass
3	click on the delete comment button (a trash can picture) next to a comment	user can delete the comment	As Expected	Pass

Test Case ID		Test Case Description	
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	Date Tested	March 24, 2023	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme
4	log in into a moderator account

S #	Test Data Requirement
1	username = test1005
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2009user15-host.cs	user can see the puzzles	As Expected	Pass
2	click on one of the puzzles	user gets redirected to the puzzle page	As Expected	Pass
3	click on the delete comment button (a trash can picture)	user can delete the comment	As Expected	Pass

Test Case ID	no log in 1	**Test Case Description**	User can solve sudoku puzzles without logging in but cannot submit the invalid solution
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2999user15.host.cs	user can see the puzzles	As Expected	Pass
2	click on one of the puzzles	user gets redirected to the puzzle page	As Expected	Pass
3	submit an invalid solution	prompts "invalid solution"	As Expected	Pass

Test Case ID	no log in 2	**Test Case Description**	User can solve sudoku puzzles without logging in and can submit a valid solution
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2999user15.host.cs	user can see the puzzles	As Expected	Pass
2	click on one of the puzzles	user gets redirected to the puzzle page	As Expected	Pass
3	submit a valid solution	show the user has successfully solve a puzzle	As Expected	Pass

Test Case ID	no log in 3	**Test Case Description**	User can solve eight puzzles without logging in and can submit a valid solution
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2999user15.host.cs	user can see the puzzles	As Expected	Pass
2	click on one of the eight puzzles	user gets redirected to the puzzle page	As Expected	Pass
3	submit a valid solution	show the user has successfully solve a puzzle	As Expected	Pass

Test Case ID	no log in 4	**Test Case Description**	User can solve eight puzzles without logging in but cannot submit the invalid solution
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2009user15-host.cs	user can see the puzzles	As Expected	Pass
2	click on one of the eight puzzles	user gets redirected to the puzzle page	As Expected	Pass
3	submit an invalid solution	prompts "invalid solution"	As Expected	Pass

Test Case ID	no log in 5	**Test Case Description**	User can solve eight queens puzzles without logging in but cannot submit the invalid solution 1
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2999user15.host.cs	user can see the puzzles	As Expected	Pass
2	click on the queen symbol at top right	user gets redirected to the puzzle page	As Expected	Pass
3	submit an invalid solution	prompts "No queen must be able to kill another for the	As Expected	Pass

Test Case ID	no log in 6	**Test Case Description**	User can solve eight queens puzzles without logging in but cannot submit a solution with less than 8 queens		
Created By	200006387				

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2099user15-host.cs	user can see the puzzles	As Expected	Pass
2	click on the queen symbol at top right	user gets redirected to the puzzle page	As Expected	Pass
3	submit a solution with less than 8 queens	prompts "Board must have 8 queens to be solved."	As Expected	Pass

Congratulations! You solved the Eight Queens puzzle.

Test Case ID	no log in 6	**Test Case Description**	User can solve eight queens puzzles without logging in and can submit a valid solution
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2999user15.host.cs	user can see the puzzles	As Expected	Pass
2	click on the queen symbol at top right	user gets redirected to the puzzle page	As Expected	Pass
3	submit a valid solution	prompts "Congratulations! You solved the Eight Queens	As Expected	Pass

Test Case ID	dark mode 1	**Test Case Description**	User can use the dark mode on main page
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs3099user15-host.cs	user can see the puzzles	As Expected	Pass
2	click on the moon symbol at top right	the page display dark mode	As Expected	Pass
3				

Test Case ID	dark mode 2	**Test Case Description**	User can use the dark mode on sudoku puzzle page
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2999user15.host.cs	user can see the puzzles	As Expected	Pass
2	click on the moon symbol at top right	the main page display dark mode	As Expected	Pass
3	click on a sudoku puzzle	the puzzle page display dark mode	As Expected	Pass

Test Case ID	dark mode 3	**Test Case Description**	User can use the dark mode on eight puzzle page
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2009user15-host.cs	user can see the puzzles	As Expected	Pass
2	click on the moon symbol at top right	the page display dark mode	As Expected	Pass
3	click on a eight puzzle	the eight puzzle page display dark mode	As Expected	Pass

Test Case ID	dark mode 4	**Test Case Description**	User can use the dark mode on eight queens puzzle page
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2999user15.host.cs	user can see the puzzles	As Expected	Pass
2	click on the moon symbol at top right	the page display dark mode	As Expected	Pass
3	click on a eight queens puzzle	the eight queens puzzle page display dark mode	As Expected	Pass

Test Case ID	fed login 1	**Test Case Description**	User can log in to the website using credentials from group 14
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	an account on group 14 website
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2999user15.host.cs	user can see the puzzles	As Expected	Pass
2	click login on top right	redirects to log in page	As Expected	Pass
3	select group 14 from a drop list	redirects to fed log in page of group 14	As Expected	Pass
4	user log in group 14	redirects back to our website where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 2	**Test Case Description**	User can log in to the website using credentials from group 13
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	an account on group 13 website
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2999user15.host.cs	user can see the puzzles	As Expected	Pass
2	click login on top right	redirects to log in page	As Expected	Pass
3	select group 13 from a drop list	redirects to fed log in page of group 13	As Expected	Pass
4	user log in group 13	redirects back to our website where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 3	**Test Case Description**	User can log in to the website using credentials from group 12
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	an account on group 12 website
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2999user15.host.cs	user can see the puzzles	As Expected	Pass
2	click login on top right	redirects to log in page	As Expected	Pass
3	select group 12 from a drop list	redirects to fed log in page of group 12	As Expected	Pass
4	user log in group 12	redirects back to our website where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 4	**Test Case Description**	User can log in to the website using credentials from group 11
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	an account on group 11 website
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2999user15.host.cs	user can see the puzzles	As Expected	Pass
2	click login on top right	redirects to log in page	As Expected	Pass
3	select group 11 from a drop list	redirects to fed log in page of group 11	As Expected	Pass
4	user log in group 11	redirects back to our website where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 5	**Test Case Description**	User can log in to the website using credentials from group 10
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	an account on group 10 website
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2999user15.host.cs	user can see the puzzles	As Expected	Pass
2	click login on top right	redirects to log in page	As Expected	Pass
3	select group 10 from a drop list	redirects to fed log in page of group 10	As Expected	Pass
4	user log in group 10	redirects back to our website where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 6	**Test Case Description**	User can log in to the website using credentials from group 16
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	an account on group 16 website
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2999user15.host.cs	user can see the puzzles	As Expected	Pass
2	click login on top right	redirects to log in page	As Expected	Pass
3	select group 16 from a drop list	redirects to fed log in page of group 16	As Expected	Pass
4	user log in group 16	redirects back to our website where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 7	**Test Case Description**	User can log in to the website using credentials from group 17
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	an account on group 17 website
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2099user15.host.cs	user can see the puzzles	As Expected	Pass
2	click login on top right	redirects to log in page	As Expected	Pass
3	select group 17 from a drop list	redirects to fed log in page of group 17	As Expected	Pass
4	user log in group 17	redirects back to our website where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 8	**Test Case Description**	User can log in to the website using credentials from group 18
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	an account on group 18 website
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2999user15.host.cs	user can see the puzzles	As Expected	Pass
2	click login on top right	redirects to log in page	As Expected	Pass
3	select group 18 from a drop list	redirects to fed log in page of group 18	As Expected	Pass
4	user log in group 18	redirects back to our website where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 9	**Test Case Description**	User can log in to the website using credentials from group 19
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	an account on group 19 website
2	
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs2999user15.host.cs	user can see the puzzles	As Expected	Pass
2	click login on top right	redirects to log in page	As Expected	Pass
3	select group 19 from a drop list	redirects to fed log in page of group 19	As Expected	Pass
4	user log in group 19	redirects back to our website where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 10	**Test Case Description**	User can log in to group 10 using credentials from our website
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs3099user10.host.cs	user can see group 10 main page	As Expected	Pass
2	click login on the website	redirects to log in page	As Expected	Pass
3	select group 15 from a drop list	redirects to our fed log in page	As Expected	Pass
4	user log in our website	redirects back to group 10 where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 11	**Test Case Description**	User can log in to group 11 using credentials from our website
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs3099user11.host.cs	user can see group 11 main page	As Expected	Pass
2	click login on the website	redirects to log in page	As Expected	Pass
3	select group 15 from a drop list	redirects to our fed log in page	As Expected	Pass
4	user log in our website	redirects back to group 11 where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 12	**Test Case Description**	User can log in to group 12 using credentials from our website
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs3099user12.host.cs	user can see group 12 main page	As Expected	Pass
2	click login on the website	redirects to log in page	As Expected	Pass
3	select group 15 from a drop list	redirects to our fed log in page	As Expected	Pass
4	user log in our website	redirects back to group 12 where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 13	**Test Case Description**	User can log in to group 13 using credentials from our website
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs3099user13.host.cs	user can see group 13 main page	As Expected	Pass
2	click login on the website	redirects to log in page	As Expected	Pass
3	select group 15 from a drop list	redirects to our fed log in page	As Expected	Pass
4	user log in our website	redirects back to group 13 where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 14	**Test Case Description**	User can log in to group 14 using credentials from our website
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs3099user14.host.cs	user can see group 14 main page	As Expected	Pass
2	click login on the website	redirects to log in page	As Expected	Pass
3	select group 15 from a drop list	redirects to our fed log in page	As Expected	Pass
4	user log in our website	redirects back to group 14 where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 16	**Test Case Description**	User can log in to group 16 using credentials from our website
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs3099user16-host.cs	user can see group 16 main page	As Expected	Pass
2	click login on the website	redirects to log in page	As Expected	Pass
3	select group 15 from a drop list	redirects to our fed log in page	As Expected	Pass
4	user log in our website	redirects back to group 16 where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 17	**Test Case Description**	User can log in to group 17 using credentials from our website
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs3099user17.host.cs	user can see group 17 main page	As Expected	Pass
2	click login on the website	redirects to log in page	As Expected	Pass
3	select group 15 from a drop list	redirects to our fed log in page	As Expected	Pass
4	user log in our website	redirects back to group 17 where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 18	**Test Case Description**	User can log in to group 18 using credentials from our website
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs3099user18.host.cs	user can see group 18 main page	As Expected	Pass
2	click login on the website	redirects to log in page	As Expected	Pass
3	select group 15 from a drop list	redirects to our fed log in page	As Expected	Pass
4	user log in our website	redirects back to group 18 where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass

Test Case ID	fed login 19	**Test Case Description**	User can log in to group 19 using credentials from our website
Created By	200006387		

QA Tester's Log

Tester's Name	200006387	**Date Tested**	March 24, 2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Connection to the school server
2	Browser
3	set up commands from Readme

S #	Test Data Requirement
1	username = testme01
2	password = password
3	
4	

Test Conditions

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://cs3099user19.host.cs	user can see group 19 main page	As Expected	Pass
2	click login on the website	redirects to log in page	As Expected	Pass
3	select group 15 from a drop list	redirects to our fed log in page	As Expected	Pass
4	user log in our website	redirects back to group 19 where the user is logged in	As Expected	Pass
5	clicks on profile	user has a solver profile with name and email address	As Expected	Pass