CS3099 Deliverable 2 Report

Group 24
200007413, 200032853, 200007047, 200036815, 190006961

Implemented functionality of our system:

Required Functionality

Our website has currently fulfilled all the required functionality for the minimum viable product.

To create or play puzzles on our website, a user should login to an existing account by email and password. New users can first register an account with an email, password, a display name and a chosen role. The email should be unique and not used by another account before, and this would be checked in the backend database. After registration, the new user will be redirected to a login page where they can use their newly registered credentials to log in to their account. During login, false credentials will trigger an alert to inform the user of the non-existent account.

As roles are yet to be implemented, all registered users are allowed to play sudoku once they are logged in. At this stage, our website would choose a random sudoku created and stored in the database for users to play. Instructions for interacting with the sudoku board are also given on the playing page. The page auto-submits the sudoku to the backend for checking once all cells are filled. Corresponding alerts will inform the user on whether their answer is correct or not.

Registered users can also create their own sudoku, with instructions and rules specified on the page. They should press the submit button once the sudoku creation is complete, with a full solution and sufficient number of hint cells. They will be informed on whether their creation is successful or not by alert message. A successful creation will also redirect the user back to the home page.

Our user interface is simple and easy to operate. It contains a home page with a navigation bar at the top of the website. So far, guest users cannot submit created puzzles, but can play them. Whilst we have checked for users to be logged in, we have not yet implemented the permissions system to determine if a user is able to set or solve puzzles. After logging in, the navigation bar would show navigation to home, create page, playing page, a user profile page which is yet to be implemented, and an option to log out. The home page introduces the purpose and function of our website, while instructions for playing and creating sudoku are clearly stated in the corresponding pages.

For supergroup logins, it is done by redirection of web pages. Users are able to login the federation pages of all other groups of Twenties (20-29) using an account registered on the user24 page. Right after users select group 24 on another group's website login page, they will be redirected to our login page, and a successful login will lead the user back to the origin group website, such that our users can proceed to setting or solving puzzles from other groups. For users from other groups who intend to play our puzzles, they can choose their group number in the dropdown list on our login page. The selection would, similarly, redirect them to their own group login page then they will be able to create and play puzzles on our site after a successful login.

Other extra functionalities

Apart from the requirements, we also implemented a few extra functionality on our website.

For puzzle playing, we implemented a pencil noting function for each cell. This is quite an important function especially when it comes to difficult sudokus. It will always help if the users are able to mark down the potential numbers in a cell, for comparison with other cells before confirmation. We also allow the notes to reappear if the user is hesitant on confirming the number in a cell. For example, a player confirms the number after taking several notes of a cell value, but realizes their selection is wrong and decides to remove the confirmed value. In this case, their previously marked down numbers will reappear so that they can reconsider their choice based on the old notes.

For login sessions, users are neither supposed to create or play any puzzle when they are logged out, nor supposed to have access to registration or login page when they are logged in. Therefore, apart from the change in navigation bar content as mentioned above, redirection of pages are also implemented in case they intend to visit disabled page in their session by URL. Logged in users will be redirected to the home page, without losing their login session, if they access the login or registration page. Guest users will be redirected to the login page if they intend to enter play or create page.

For sudoku creation, we validate the sudokus in the backend by making sure that all cells are filled, that there are at least 17 hint cells and that the sudoku is new for the database. To prevent manual illegal submission in the frontend console, our backend also makes sure that the sudoku puzzle is compatible with the solution. Similar applies to sudoku playing, where we check if the answer submitted is actually from a sudoku present in our database, in case of illegal submission.

For registration, our website will store the user ID, group ID, user email, display name, creation date and roles into our users database. The password will be stored separately in another collection for security reasons. We have implemented the bcrypt library for hashing the password before storing it. The hashing will include 11 salt rounds as we believe that will be secure enough (recommended 10 rounds). Hashing the password with a salt makes the algorithm's output unpredictable.

For login, our website requires users to enter their email and password. The system will then find the user entry of the user and compare the input password with the one stored in our database using the compare method of the bcrypt library. The method hashes the password in the background and compares it with the one in our database. If the password and email are correct, the user will be redirected to the home page. If either of them is incorrect, we will prompt the user to try again.

Instructions of website usage:

Users must start by registering an account, and inputting an email, password, and display name. The email will be what is used to login, while the display name is what's shown on their profile. Once logged in, users can either select "Play" or "Create."

The Play page will offer a random puzzle. Users can select a number and then click on cells to input that number. To place notes, users can click the pencil, then select a number and click the desired cells. To remove a note, they just click the cell again while the pencil and the number they wish to remove are selected.

The Create page displays an empty sudoku board, which a user can populate with numbers to create a puzzle. Numbers are input the same way as in play mode, but users can click a cell twice to mark a number as a shown hint. Once the puzzle has been filled out, users can click the submit button to submit it.

Evidence of following Scrum/ Agile development model:

We began our first development sprint on Thursday of week 4, which ran into week 6 (as per the 2-week sprint length outlined in our plan). Along with our weekly supervisor meeting on Fridays, we mostly kept to the twice-weekly meetings on Tuesdays and Thursdays (mostly on Discord at first, with in-person meetings becoming more frequent as we progressed). Tuesday meetings mainly consisted of reporting or progress, and giving a space for group members to ask for help in any specific task they may be working on. Thursday meetings were used for updating the backlog where necessary - adding new tasks and user-stories where needed. Also, these meetings were used at the end of every sprint cycle to set up for the next sprint, where we would assign the product-owner and scrum-master roles, which rotated to ensure that everyone had a chance to be assigned one of these roles before this deadline.

For each sprint, as a group, we tried to determine the highest priority user stories to work on, and break them down into their constituent tasks, and distribute those tasks based on: estimated relative effort, estimated knowledge each member of the group had with that area of programming, and interest in the area of programming.

Along with this, we assigned most groups of tasks to pairs, who would work together in that sprint cycle because we are eager to learn throughout this project. These pairs usually consisted of someone with more experience (either from previous sprints, or from experience from other places), paired with someone who wanted to get involved in that area of the project. This helped each group member gain experience in different areas of the stack, whilst having someone to help prevent major mistakes, and to guide them through understanding. It also encourages group members to work and communicate more with each other in a Scrum team.

We also change our Scrum master and Product owner in every cycle, ensuring that every teammate has at least tried one of the roles. We made use of roles assigning function on discord to indicate the 2 main roles.

Sprint 1: 06/10/22 to 20/10/22

For our first sprint cycle, we assigned 200007047 as our Scrum master, and 200036815 as our product-owner.

- See appendix IV.I for user stories
- See appendix V.I for task backlog

Sprint 2: 20/10/22 to 03/11/22

For our second sprint, we decided that 200032853 would be the scrum-master, and 200007413 would be the product-owner.

- See appendix IV.II for user stories
- See appendix V.II for task backlog

Sprint 3: 03/11/22 to 10/11/22

For the third and final sprint before this deadline, 190006961 was the scrum-master, and 200007047 was the product owner. Here, we meet online once every 2 days to make sure everything is working on track as it was closer to the deadline, despite having 2 teammates catching covid the week before.

- See appendix IV.III for user stories
- See appendix V.III for task backlog

Significant changes we have made:

At the start of the project, we were not familiar with the school server so we chose MongoDB Atlas for storing our database. It has a lot of nice tools online showing your data and monitoring your database with a great user interface. However, we believe storing passwords on the cloud is not secure, so we end up storing the data in the school's server.

We had not particularly planned for the structure of our development stack. We decided that a thinner client would work best, as it allows for less powerful hardware to run the application.

Therefore, apart from basic checks when solving and creating puzzles, most validation is done in the backend. This applies to logging in, as it greatly improves security, as less knowledge about the inner workings of the program are known to users.

This applies to the changes we made to the design of the frontend from our plan, we decided to use a single page application model instead of the multi page application model. We were initially less familiar with VueJS, and the degree to which it is designed and optimized for single page applications. Given the functionality of our site is primarily server-heavy, we decided that a single page application structure would work well.

Supergroup interaction:

During the supergroup meetings, we proposed using a suffix instead of a prefix for user ID and using a username instead of an email for logging in because we think that is more sensible. However, the federations rejected both suggestions as they stood firm with the original method. They believe we can change them ourselves as we are from different schools of thought.

Apart from meetings and communications through different discord channels, we created a shared google document for the outline and specification of the supergroup authentication protocol (Appendix I) such that the format is unified to reduce coding workload in redirection.

Extraneous Issues Faced in Development:

During our second sprint cycle, two out of the five team members tested positive for COVID which has impacted the progress of development. We took steps to distribute their work amongst the remaining. Although it took us quite a lot of effort to pick up their parts, we managed to connect the parts together and fulfill the requirements.

Appendix

Appendix I: Twenties Supergroup Authentication Protocol:

Outline:

The general auth flow will run as follows. Assume there are sites A, B with backends A, B:

1. The user has an account registered at site B, and wishes to use this account to access site A
2. The user opens the login page of site A
3. The user selects that their account is hosted on site B
4. This selection redirects the user to www.siteB.com/auth/authorize?redirect=https://siteA.com/auth/retrieveToken/ or http://www.siteb.com/auth/authorize?redirect=https://siteA.com/auth/validateToken/ for example
5. At this page on site B, the user is prompted to log in with their credentials
6. Once the user successfully logs in on the site B page, this page will redirect the user to the URL specified by the redirect query param. **Appended** to this redirect will be the JWT string that was generated by backend B
7. Now that the user is redirected to www.siteA.com/auth/retrieveToken/<tokenstring>, site A will be able to set the JWT as a cookie/header/whatever the group decides and allow them to login

Note: There are two main security flaws with this implementation.

The first is that the redirect could be made to be anywhere, so any malicious users could make the redirect go to their own site and steal JWTs. To combat this, please ensure that your code checks the redirect specified, and only redirects to sites that are in the supergroup.

The second is that the JWT is placed in the URL, which could allow it to be stolen by a logger or a proxy server. For now this can be allowed for the MVP. However, in semester 2 we should augment this protocol to instead send back authorization codes that the backends can then use to retrieve the JWTs. Search “oauth implicit grant” and “oauth authorisation code” for details on our current protocol vs the more secure version.

Specification:

GET <groupdomain>/auth/authorize

Parameters:

Field	Type	Description
redirect	string (query param)	The redirect URL that will be returned to after a successful login. This will be specified by each individual group. For example the redirect could be www.siteA.com/storeToken/, or www.siteA/auth/retrieveToken, etc.

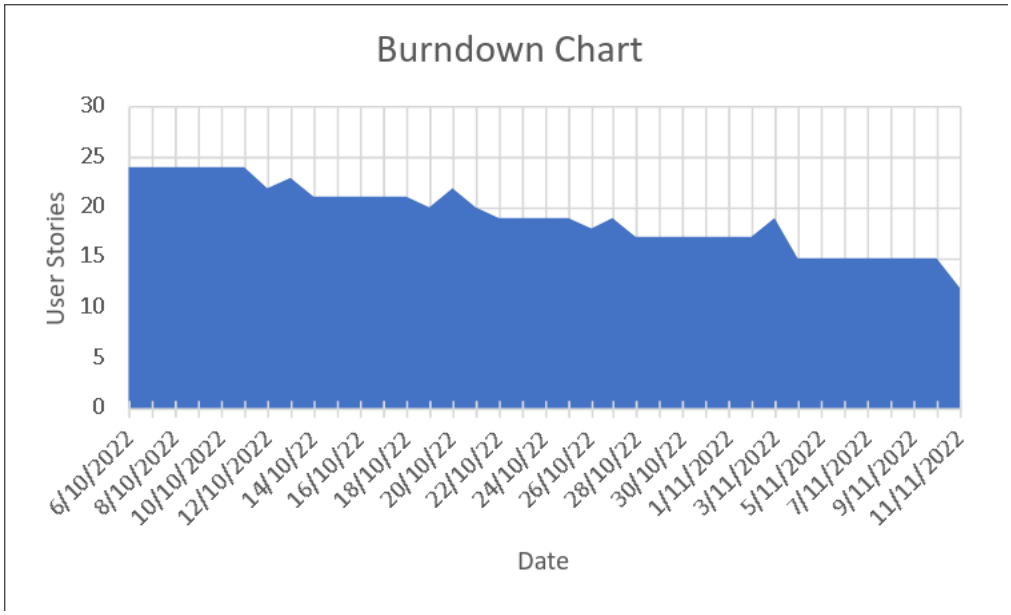
Behaviour:

When this route is navigated to, the user should be prompted to log in with their credentials. Upon a successful log in, this page will redirect the user to the URL specified in the “redirect” query parameter, with the generated JWT appended to this URL.

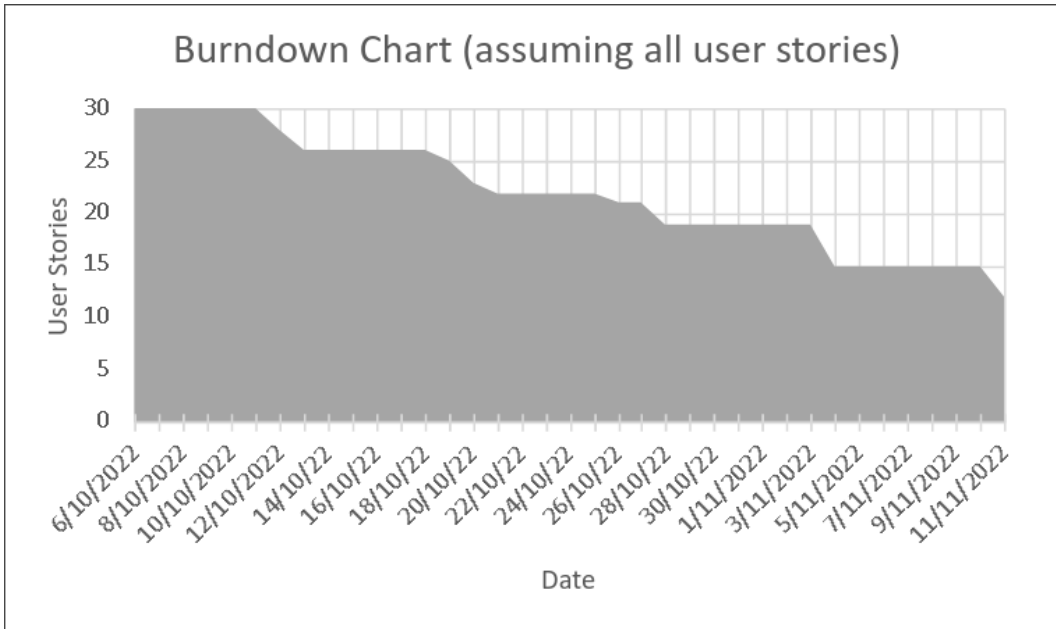
Example:

1. User is navigated to www.siteB.com/auth/authorize?redirect=www.siteA.com/storeToken
2. User now enters login credentials successfully
3. Site B generates the JWT, for example: v6s9rYfQK5
    a. For security, site B will ensure that the redirect string is a recognised URL from our supergroup
4. Site B will redirect the user to www.siteA.com/storeToken/v6s9rYfQK5

Appendix II. Burndown Chart with Progressive Updates

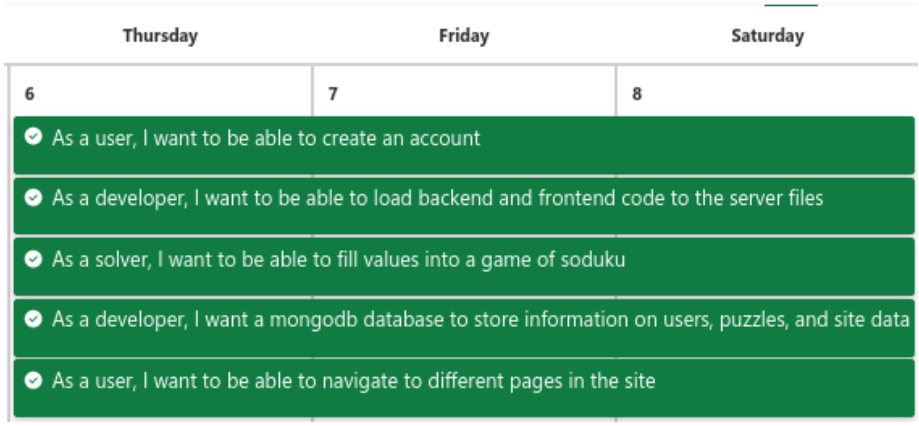


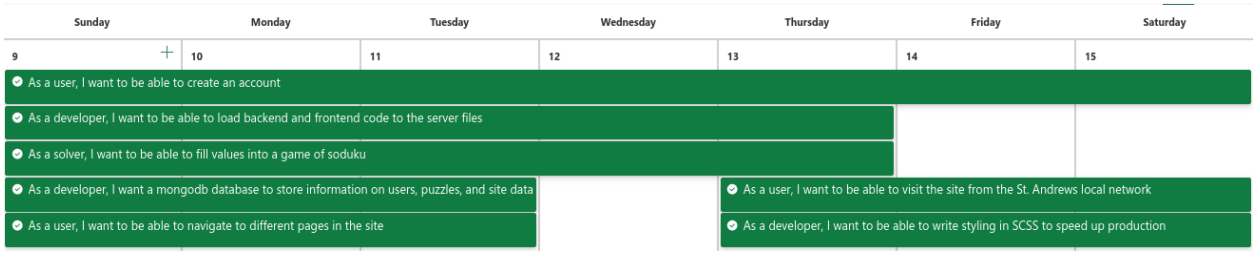
Appendix III. Burndown Chart with All User Stories Assumed At Start

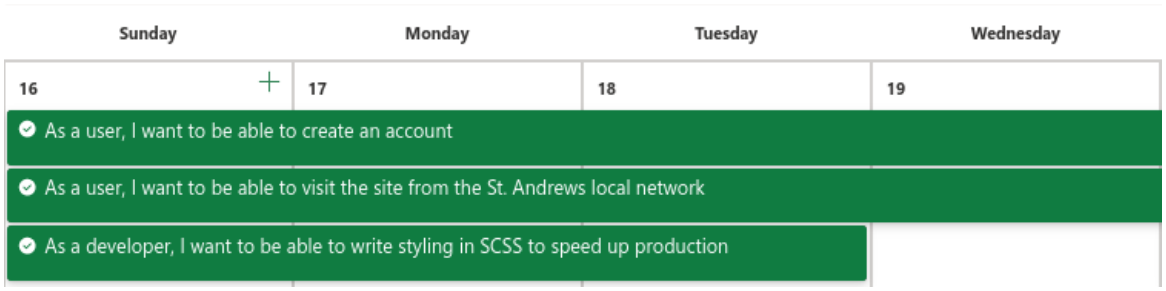


Appendix IV. Sprint Cycle User Stories

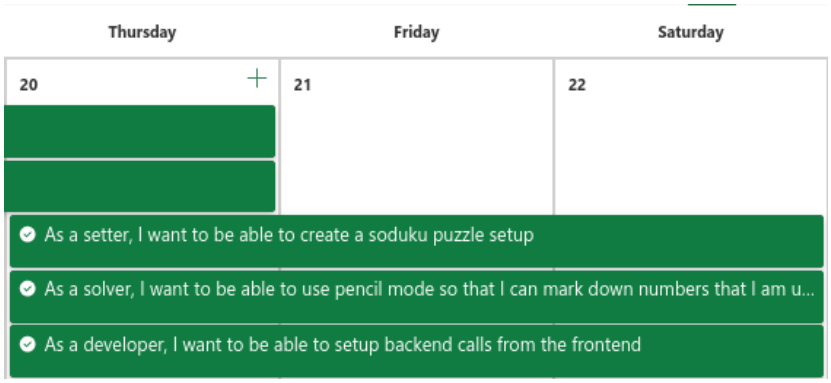
Appendix IV.I - Sprint 1 User Stories

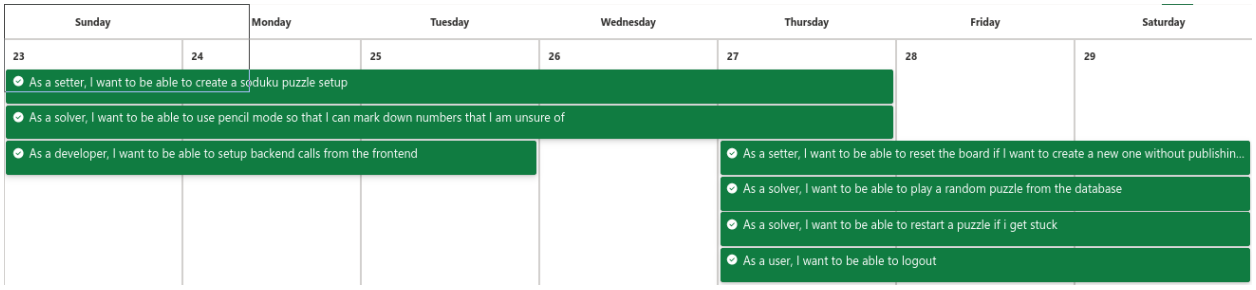


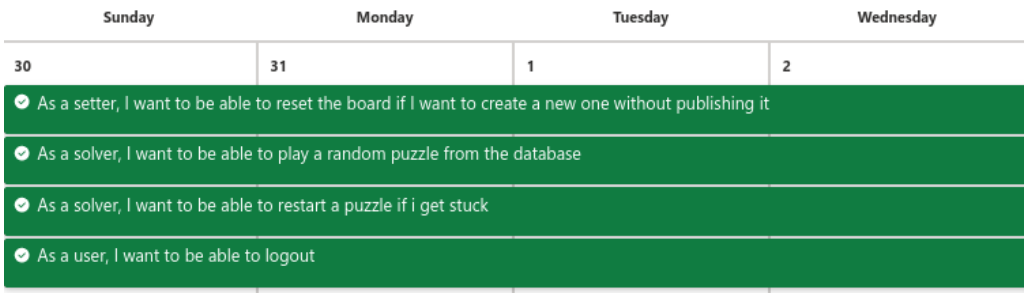




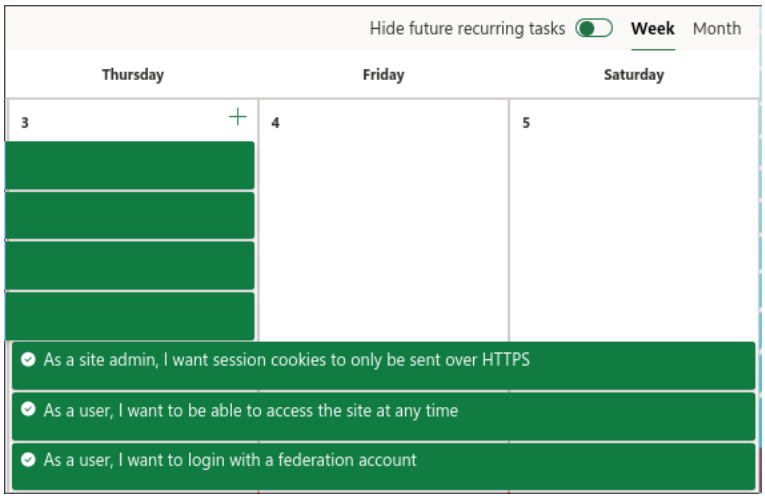
Appendix IV.II Sprint 2 User Stories

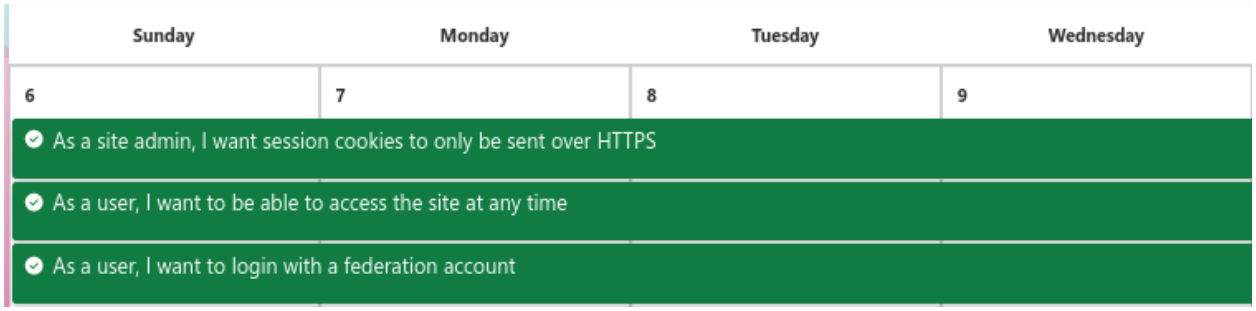






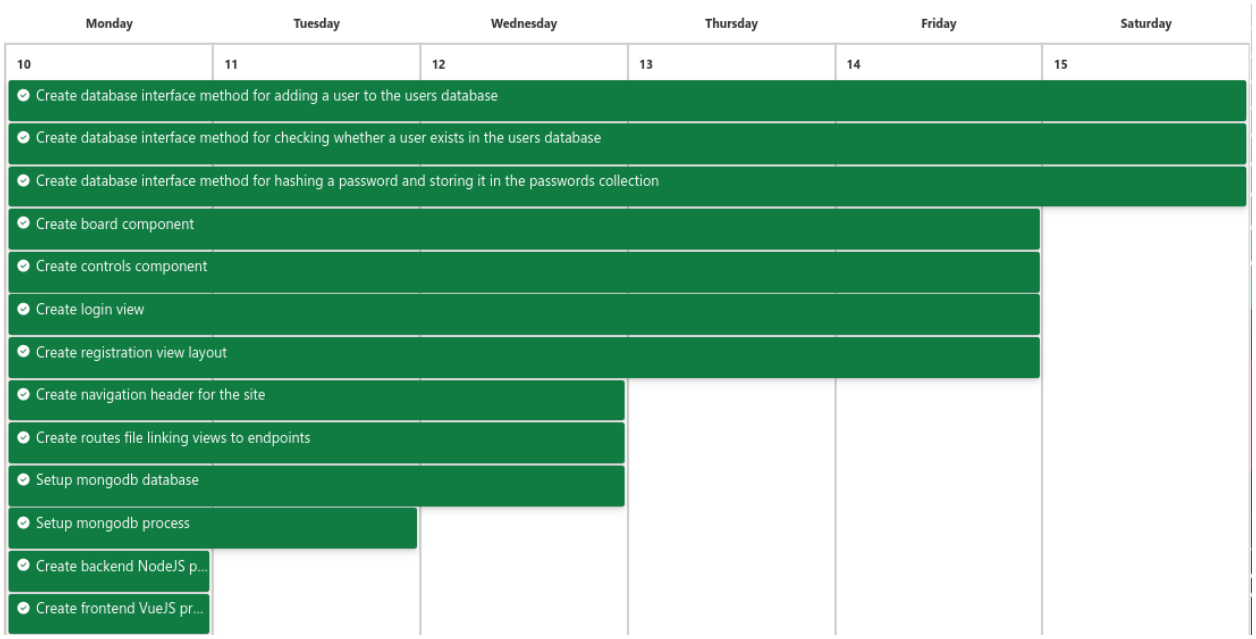
Appendix IV.III - Sprint 3 User Stories

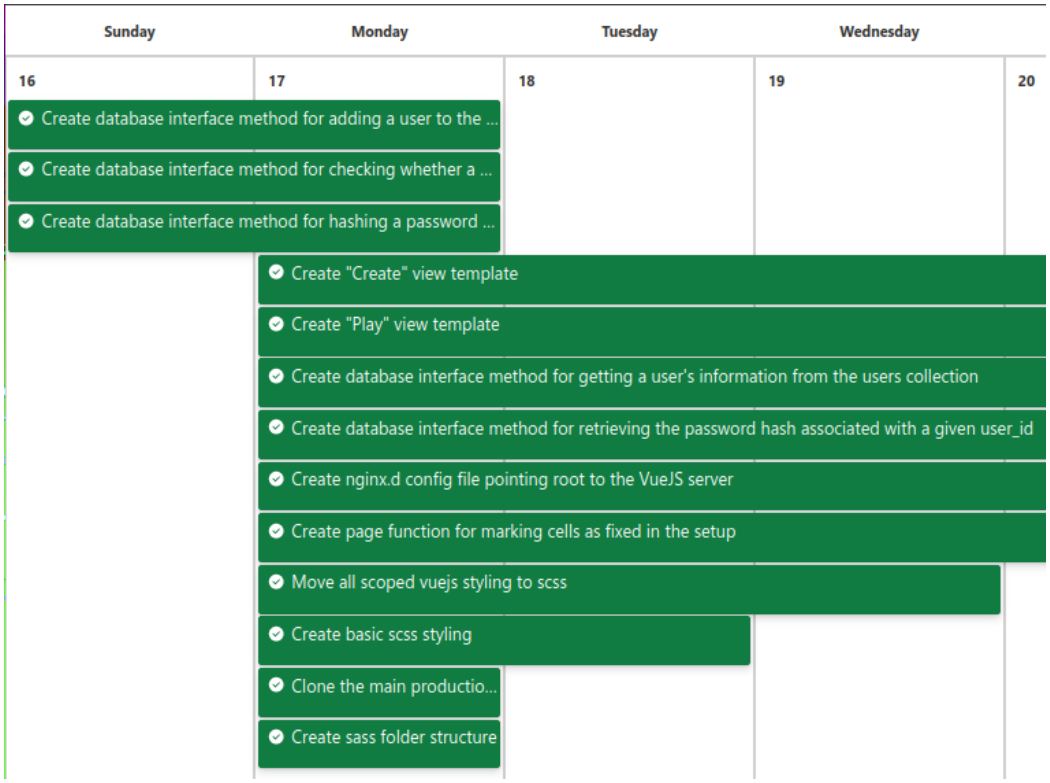




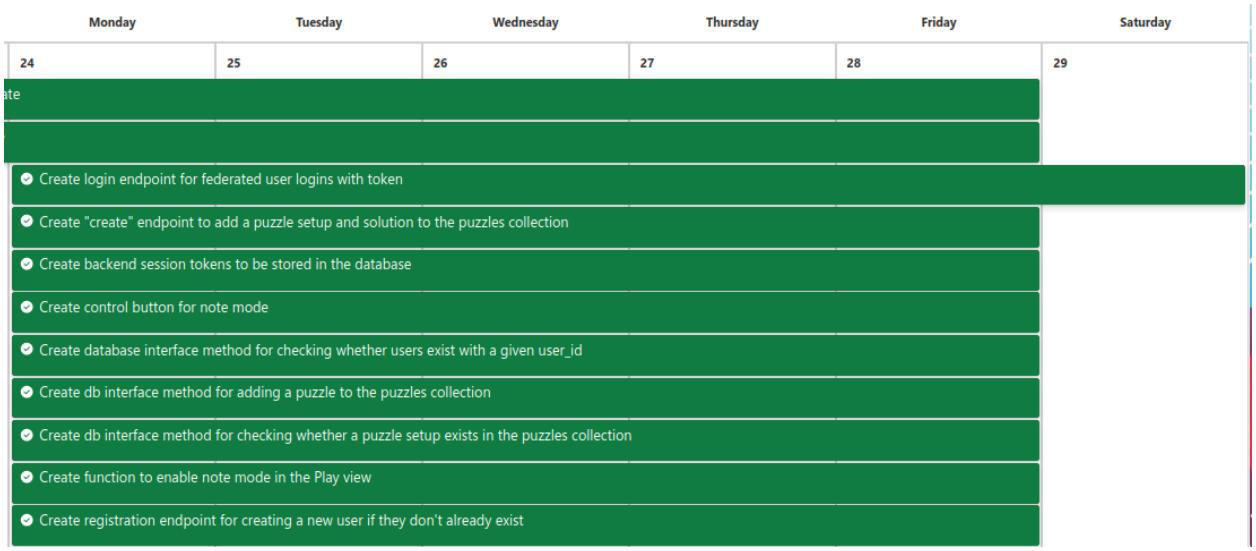
Appendix V. Sprint Task Backlogs

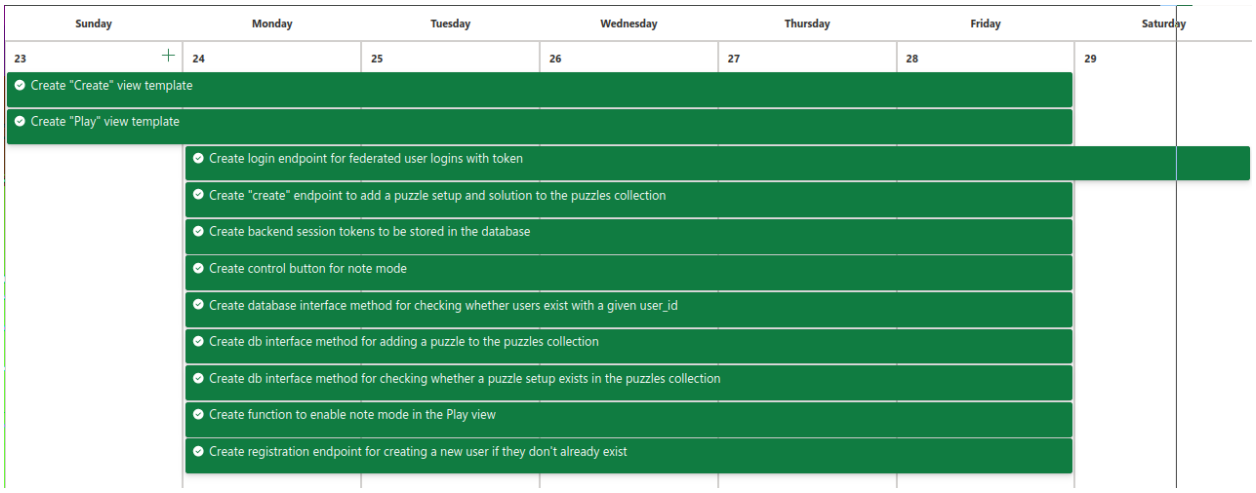
Appendix V.I. Sprint 1 Task Backlog

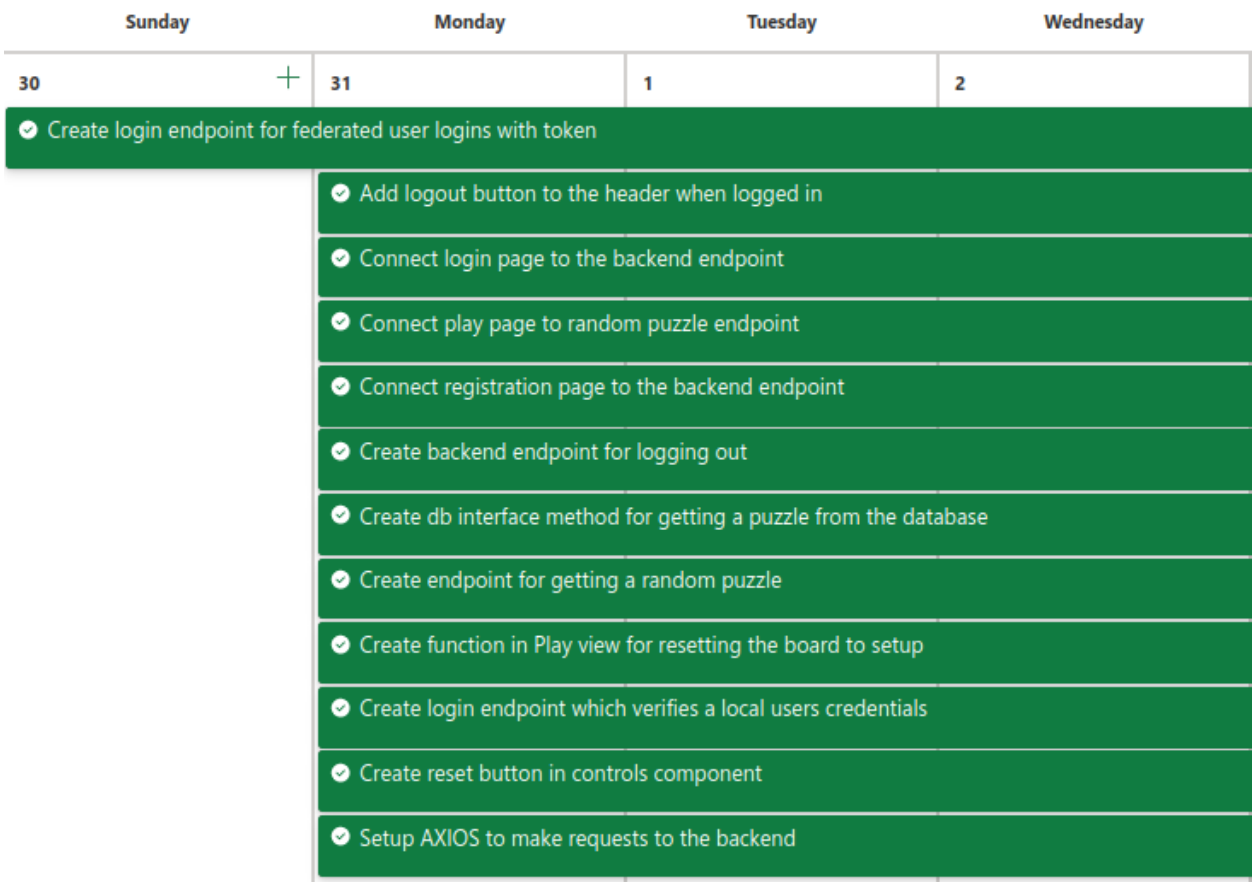




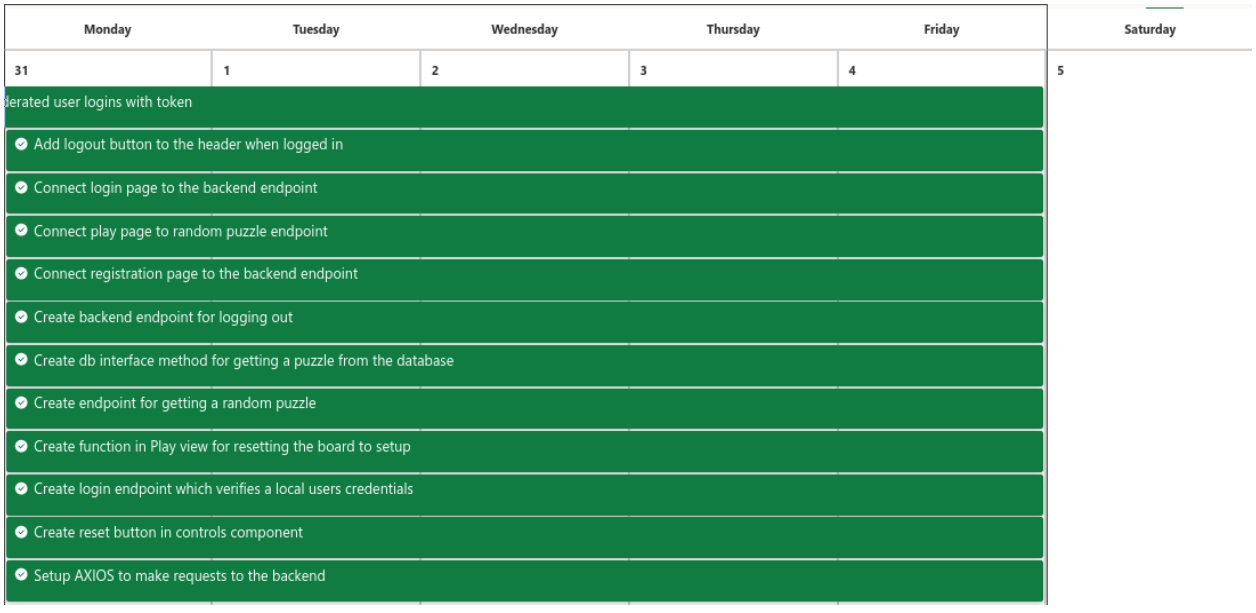
Appendix V.II - Sprint 2 Task Backlog

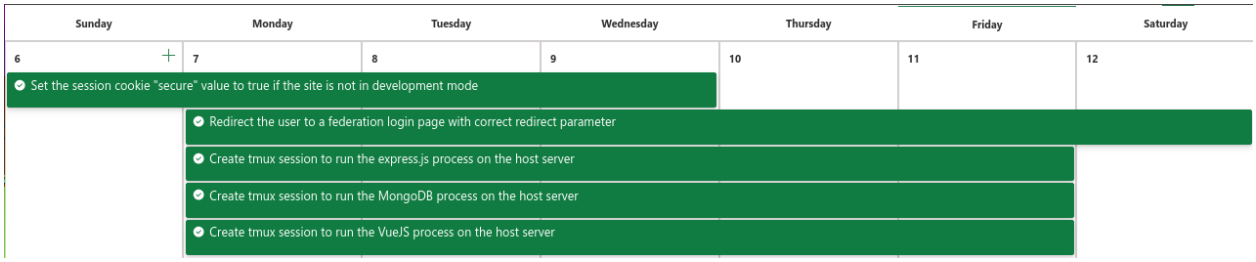






Appendix V.III - Sprint 3 Task Backlog




Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
6
7
8
9
10
11
12
Set the session cookie "secure" value to true if the site is not in development mode
Redirect the user to a federation login page with correct redirect parameter
Create tmux session to run the express.js process on the host server
Create tmux session to run the MongoDB process on the host server
Create tmux session to run the VueJS process on the host server