Introduction

This is the report for COMP 1004. The document outlines the planning and execution for the project. The project itself it to create a SPA. The project - as all projects do - start with an idea. The idea of this SPA is dependant on the criteria though, so I will start from there. The requirements following the spec are as follows;

- Contain an index.html file
- Be an SPA with some form of interactivity
- Be able to read a flat JSON file
- Be able to export a flat JSON file

Given these requirements I have come up with the idea of a calendar. This isn't just because it fits the criteria but because I want to build one. I use my own calendar religiously and would love to be able to create my own version of it. Having the control to make it do what I want and look like what I want is something that makes me look forward to coding it and will push me to do it to the best of my ability.

Project Vision

My vision for this SPA is for it to be the simplest, easiest to use, quickest calendar. My SPA is designed for people who appreciate the right balance between minimalism and customisation. I want to deliver an at-a-glance dashboard for events of the date the user wants.

Research

Researching into calendars there are an infinite amount of options to choose from. The 3 top used calendars are;

- cal.com
- calendar.google.com
- icloud.com/calendar

Looking into these they all are actually very different given they all aim to do the exact same thing. Cal aims to be a one stop solution, saying that it can connect all your calendars, have custom alerts and show when your busy to other people. Google aims to be a simple usable calendar with information quickly available at a glance and my favourite; Apple. They seem to have a minimalist design while also retaining import information that is most likely to be needed. They all have there strong points and niches which helps separate them. I aim to create a simple calendar inspired by all the above as well as my own ideas. Simple, clean, customisable, concise. What they all have in common is having a month calendar visible at all times, as well as the current date at the top. New event buttons are always visible and generally there is a lack of colors on the page.

Functionality

Initially there are functions the SPA must include. I will later go on to create a kanban board that gets updated every time a function changes state (e.g. goes from backlog to in progress) but for now I will put my initial requirements below;

- Display the current month with an option to change the month (e.g. go to 2021 October). This is self explanatory, a calendar must be able to see past and future months to be usable.
- Have a persistent banner for todays events. If there are no events on the current day do not show the banner. The calendar must be able to show upcoming events with a different level or urgency compared to other days events
- Be able to export and import your own calendar using JSON. Having a calendar that can go cross device is a very clear must and since I'm not willing to create a cloud infrastructure for this project this is the next best thing.

- Be persistent using local storage and proxies. The calendar would be fairly useless if every time you refresh the page your events disappeared.
- Options for each event. Each event must be able to have options that get reflected on the UI like colors and urgency.
- Live date/time. Having a library to update the time/date into whatever element I want just by in the element tag adding an attribute (e.g.)
- Create a fake calendar (the data for it)

SDLC

The approach I took to creating this project was the agile approach. Because I have a small amount of pre existing knowledge I know certain "challenges" will not be difficult or take time for me. So to combat this I will use the agile method; figuring out what I want to do, designing it, coding it, testing it, maintaining it. This works best for me in this project. Each part (from the kanban board) will be what you could call a sprint (details later about sprints). Going through each process for each feature of the project. These will all be documented in the meetings files on the GitHub repo.

Designs

I like to start my designs off on paper instead of going straight to Figma. I tried for over 30 minutes but images were not getting along with me so I have them all at the bottom of the page. They are all labelled. I went through 3 versions of the designs. First was on paper as stated and the others digital - using Figma. I went through multiple different iterations for different parts, not everything needed to be redesigned so I have dont some parts separate like the single event. I do prefer designing on paper (don't have an tablet or anything digital to draw with) but know that there are tools online that can be really effective. I make the most of both worlds and until I have a initial idea on paper I wont go onto digital. Once ive got digital its fairly easy to update and change - even make components - so I do

understand the appeal. Its also better if I need to collaborate with someone as its easier to share.

User Story & UML Models

Below are the UML models for the calendar, highlighting key events and naming the procedures that should follow the events taking place. Also below is my user story graph, This helps me visualise what I should prioritise. Using this graph you can figure out what the user is most likely going to do so you can make sure those areas are better and focus on making them the star of the show. User story graphs are really important for not just reading what the user will do but it will help with what to code and where to start. Using user stories and UML models together can really improve the speed of how I program. If I know what I want the user to do I can split that into how we can do that. From that ill be able to put together a plan of how to tackle each problem. I don't normally start with the smallest (in size) but instead the problem with the least dependencies. What I mean by this is working on functions or parts of the code that require the least or no other parts to work. Take a part I mention at the end of this report. I create a function to make a fake calendar and apply it, this uses a function called 'createFakeEvent'. The create fake event function doesn't require any other functions so I would work on that first, I hope that's a good example.

Sprints

There were quite a few sprints and if you would like to read more on each sprint you can do so by reading the meetings in the GitHub repo. Each one of the below is a sprint I did. But when I say sprint I do not mean 2 weeks. There wasn't anything that could've taken that long so I mean sprint as in each feature worth mentioning.

Feature	Status
Persistence using local storage/proxies	

Feature	Status
Day (Tuesday, Monday) on each date in calendar view	
Ability to change selected month in calendar view	
Create, update, delete events	
Go to current month button if current month != selected month	
Dark mode/light mode based on color scheme	
Import calendar JSON	
Export calendar JSON	
Banner for todays events (hides if 0 events toady)	
Events have custom colors and priority	
Live date/time with html attributes	
Define calendar data/calendar event type	
Make calendar data	
Simpler UI Mode	

Here is a demo of one of my meetings/sprints

> # Meeting 15/02/2024
>
> This sprint was based off the remaining tasks to do. There are 2; add days of the week to each date on the calendar and use local storage to save the events locally.
>
> Having each date on the calendar display the day of the week was a simple task. I used the 'Date' object (which is already formatted) to get the day of the week and then displayed it on the calendar with a bit of style.
>
> The local storage was a bit more complicated. It has to be simple, readable and understandable. Another big thing is having the code component-ised so that it can be reused and if your debugging something it is much easier to figure out where a problem could be and fix it. With the local storage it is there to make sure that the events are saved when the page was closed and when its opened again the events are still there.
>
> As i am all done now here is a link to the final live [Website](https://comp-1004-spa.vercel.app/)

Issues & Challenges

As stated above because of my small but yet pre existing knowledge or web dev I didn't seem to struggle with many things, I did in fact enjoy this project. The most difficult things I faced were;

- The calendar data being persistent. This posed quite a challenge as you have to first check if it exists, then deal with getting it. Then you have to save to local storage every time the calendar changes, even the smallest change. Luckily something called proxies exist in JavaScript making this easier to implement.
- Live date/time was also fairly difficult as I had to have a constantly updated and formatted time/date and then on each element that requires it insert the correct information where it should be. Again JS luckily has a built in date formatter then can be used to neatly format not only today and now but any date.

Having things I found difficult was good as it made me think about how to disect a problem. Coming up with solutions isn't always as easy as just one thing, it might be a collection of different things which is a good way to look at it anyway. I find for me breaking a problem down into smaller problems/sections/functions makes it not only manageable but allows me to see the solution clearly. If any problem is small enough I can see clearly a way to fix it. Some problems are not problems as well, they are rather just things that have not yet been done. For instance live date and time. Break it down; get the current date -> format it -> display it where needed -> repeat every 1000ms. It doesn't seem so bad now. And now they're small enough I wouldn't call them problems, more in progress functions.

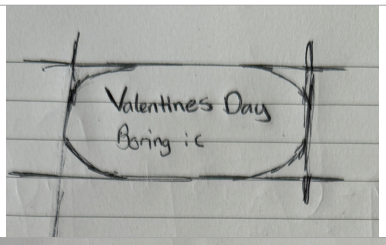
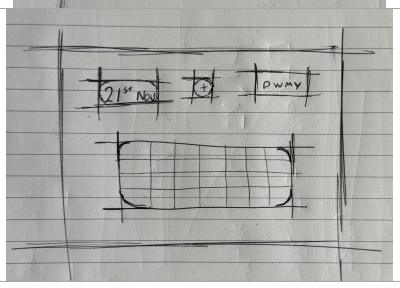
Fake Data/Calendar

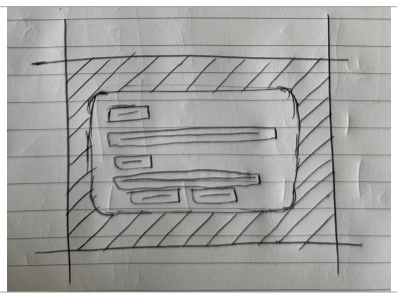
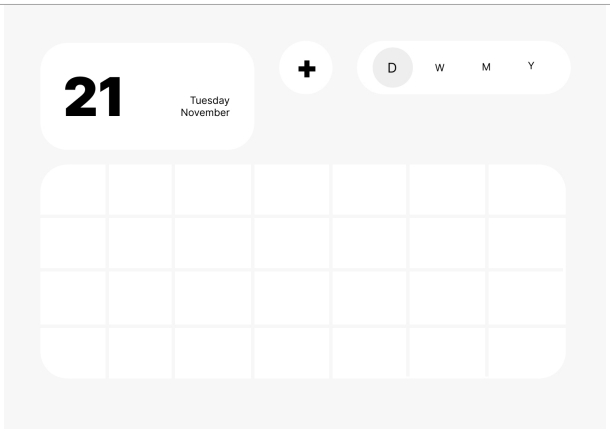
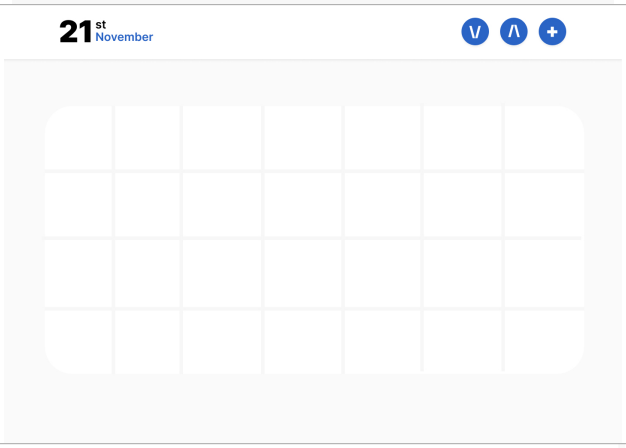
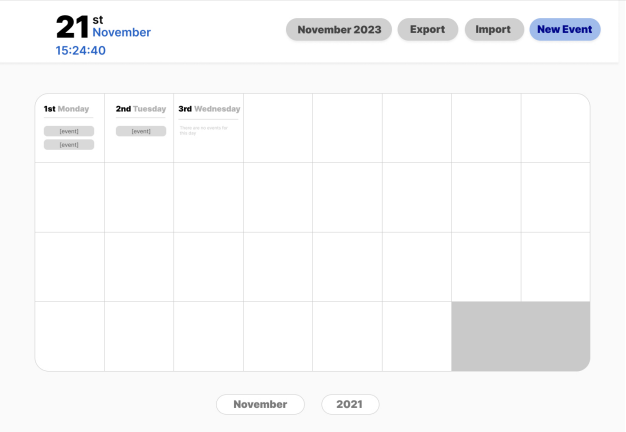
So another issue I encountered was actually testing things. I cannot see the changes I am making to the ui if I cannot see events and stuff. Before I dive into how I fixed that there were a couple other hot fixes I managed to get in. Firstly instead of having the year picker to be a select (so limited amount of years) I

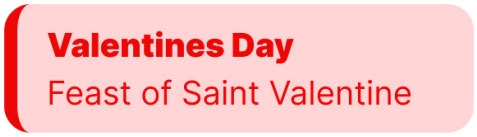
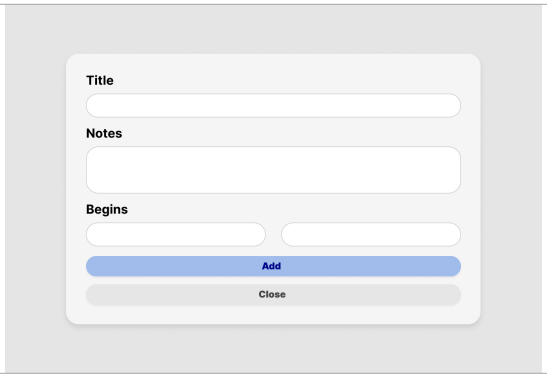
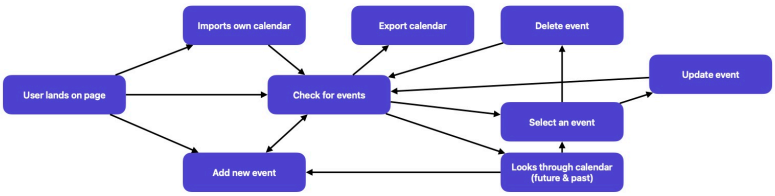
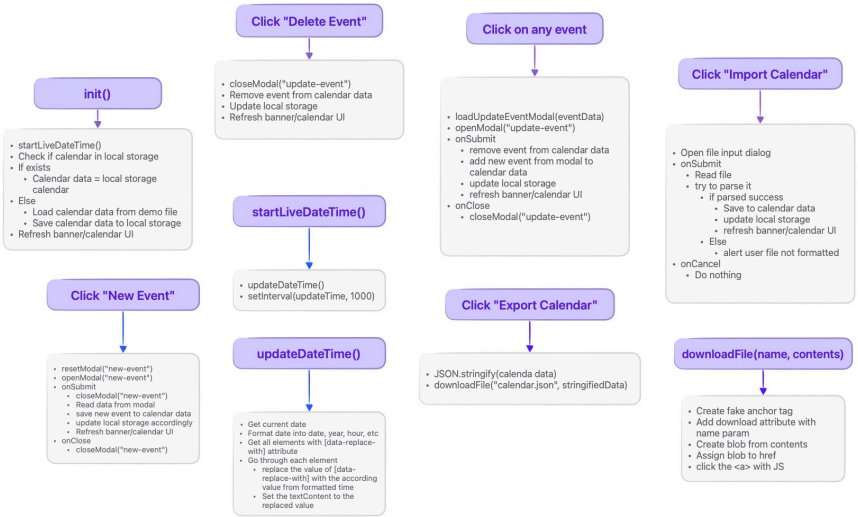
changed it to a number input, that way you can plan without limits. Another thing I changed was having a cleaner ui mode, so just removing padding in some areas, limiting font size, removing event descriptions. Just something to make the ui a tiny bit cleaner. Now back to the fake data. This is something that is needed to actually test the data. I cannot just keep manually adding and changing things so I needed to come up with a way to create and fill a fake calendar. I did this by first creating a function. This functions job is to create a fake event and return it, this is fairly simple, using lorem ipsum and then for the date getting a number in ms +- 1 month from the current date as the being and then + 1-3 hours for the end date. We utilise this function in a separate function that will create an array of empty elements and then populate each one with the functions return. After a couple mishaps with my math, I didn't see that 26e9 should of been 26e8 for hours and it make the demo start date be +- 10 months instead of one. Working with e numbers is nice and makes you feel cool but sometimes it can cause difficulty reading. To battle this I created a few constants called [time]InMs like hourInMs and so on to make it easier to read.

Poster

My poster is below with the other images

Paper Design - V1 - Single Event			
Paper Design - V1 - Page			

Paper Design - V1 - Modal	
Digital Design - V1 - Single Event	**Valentines Day** Feast of Saint Valentine
Digital Design - V1 - Page	
Digital Design - V2 - Page	
Digital Design - V3 - Page	

Digital Design - V3 - Single Event	
Digital Design - V3 - Modal	
User Story Graph	
UML Models	

Poster	

GitHub Repo Link: https://www.github.com/bobbymannino/comp-1004-spa