David Goggins Society – Design Document

Group Members

- Tom Gehling – a1838949
- Julian Lovell – a1829617
- Tom Materne – a1825150
- Jack Scholten – a1826009

Contents

Project Specifications

A Vue3 website that can manage university clubs. This website will need to handle membership signup for students, club management and administration. The clubs should also be able to create and manage events. These events should then be on the member's home page. Their homepage should also include club updates. The members will have different roles within each club.

Members will create and authenticate their account with an email and password. This should be stored in a database. The account should be able to link social media accounts, so that they can login with those instead in the future. The members should be able to view event updates and see upcoming events in a calendar or list type view. The members should also have a page to view the club announcements/posts.

Clubs are managed by members with specific roles. A club manager can manage other users roles & other club information. Clubs can make posts to the members. Clubs can be open or invite only. Clubs can create events, and the RSVP status of the members should be visible.

A system admin is a privileged member who can manage/delete club and delete users. Can change other users to admins as well.

Application Name & Style

Plooto

This application name was inspired by the famous company Mars, which shares its name with a planet. Thus, we have decided to follow suit and name our website after the infamous non-planet, Pluto. Through this we expect it to inspire a sense of curiosity. When we brainstormed design ideas, we wanted to keep space as a key part of our identity, so the purple theme stems from our visions of space.

Figure 2 https://wallup.net/wp-content/uploads/2016/01/9387-space-planet-space_art-purple.jpg

Figure 1 https://wallpaper.dog/large/10701736.jpg

Features

There are several features which are required to be implemented within the website. These are as follows:

1. User Management:
    - Users can sign up and log in.
    - Users can manage their user information.
    - Users can link social media/email accounts for easier login.
2. Club Membership:
    - Users can join a club.
    - Users can view updates from clubs they're members of.
    - Users can see upcoming club events and RSVP.
3. Club Manager Functionality:
    - Club managers can log in and manage their user information.
    - Club managers can view their club members.
    - Club managers can post updates publicly or privately to club members.
    - Club managers can create and update club events.
    - Club managers can see who has RSVP'd for an event.
4. System Admin Functionality:
    - System admins can log in and manage their user information.
    - System admins can manage users (e.g., create, edit, delete user accounts).
    - System admins can manage clubs (e.g., create, edit, delete club information).
    - System admins can sign up other admins.
5. User Access and Public Information:
    - Users without accounts can view public information and updates for clubs.
6. Email Notifications:
    - Users can sign up to receive email notifications from clubs.
    - Users can choose the types of email notifications they receive for each club.

Additionally, there are other features that we seek to add to improve upon the functionality of the website and provide a better user experience:

1. Calendar:
    - Support for a calendar displaying past and upcoming events that the user has.
2. GPS Functionality:
    - Events have a GPS point displaying their location on a map.
3. Attendance Status Options:
    - Users can provide ‘Going’ or ‘Interested’ states under events for more options on event management.

Research & Design

Style

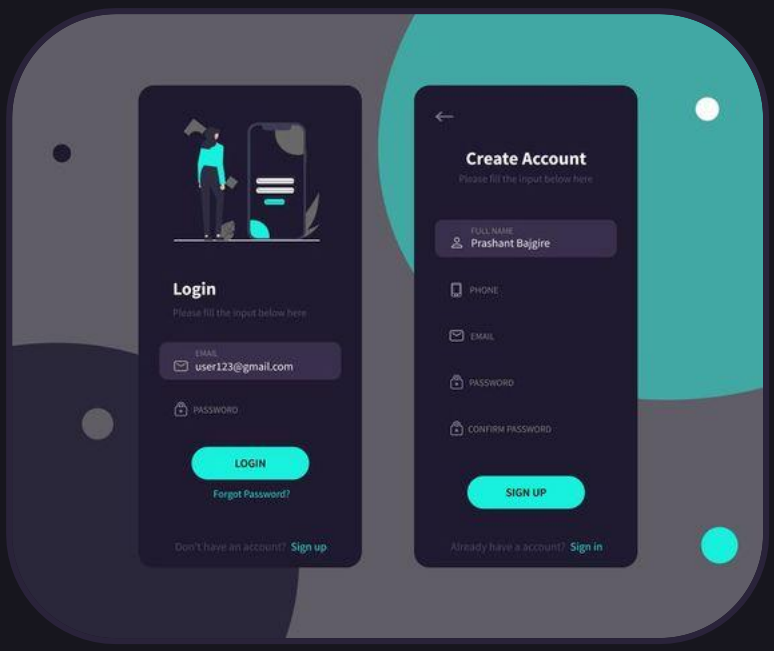


Figure 3 https://www.pinterest.com.au/pin/634092822542834963/

This mobile login/create account page has really nice buttons and input fields. Radius placed on all edges to maintain clean looking design.

Minimal colour scheme, fonts and overall design to make it simple and easy to use as well as being appealing to look at.

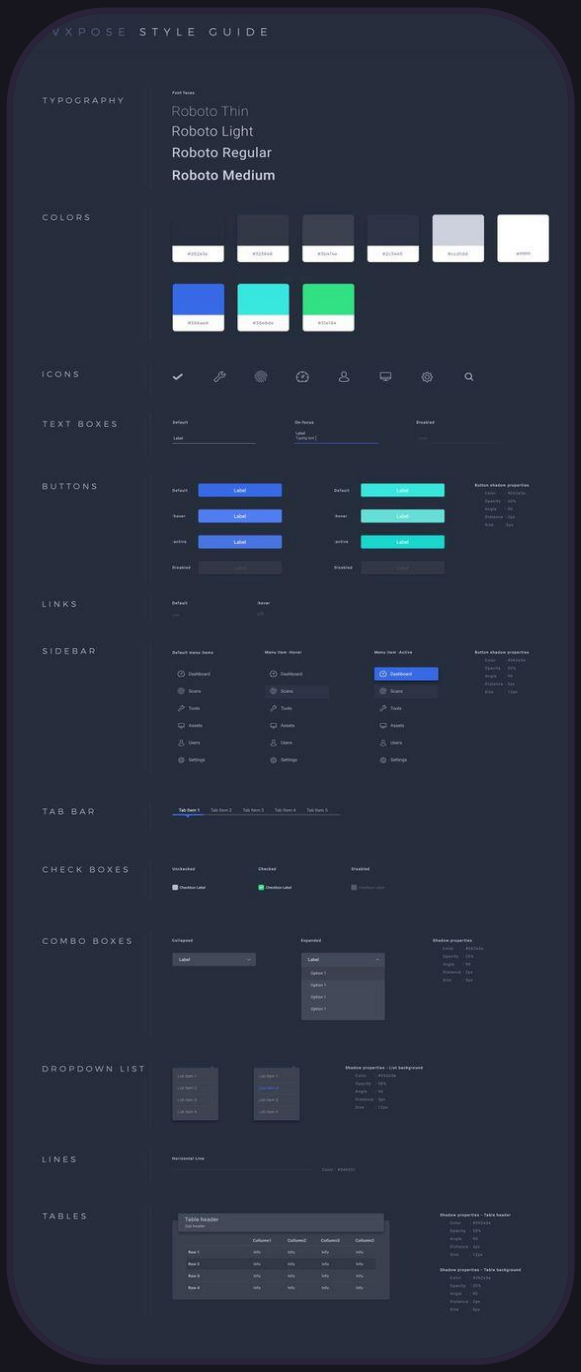


Figure 4 https://dribbble.com/shots/3456737-UI-Style-guide/attachments/760330

Figure 5 https://www.pinterest.com.au/pin/914862417927726/

This colour palette is a good foundation to base our colour scheme off for the website. We can utilise it as the base of a dark mode theme, with orange being our accent colour across both light and dark mode.

Another benefit of using purple, is that it will create a unique look. The goal of this is to separate our website from other similar social media sites which commonly use colours such as blue, pink and red.

Orange is a good choice as an accent colour because it is a contrasting colour to purple, so supports both colour theory and will stand out to the eye.

Therefore, by incorporating this colour palette into the dark mode design, we can create a visually appealing and user-friendly interface that embraces both good appearance and functionality.

The biggest takeaway from this image is the use of a radius on the corners on all elements across the page.

This creates a modern and appealing design which can be used to build upon the minimal appearance we want to create.

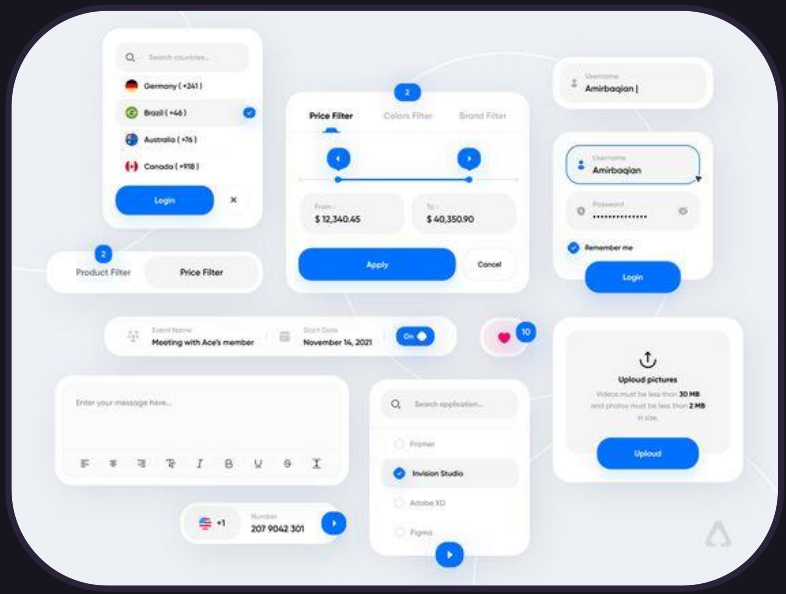


Figure 6 https://dribbble.com/shots/15469095-Dark-UI-Elements-Dropdowns-Calendar

Structure

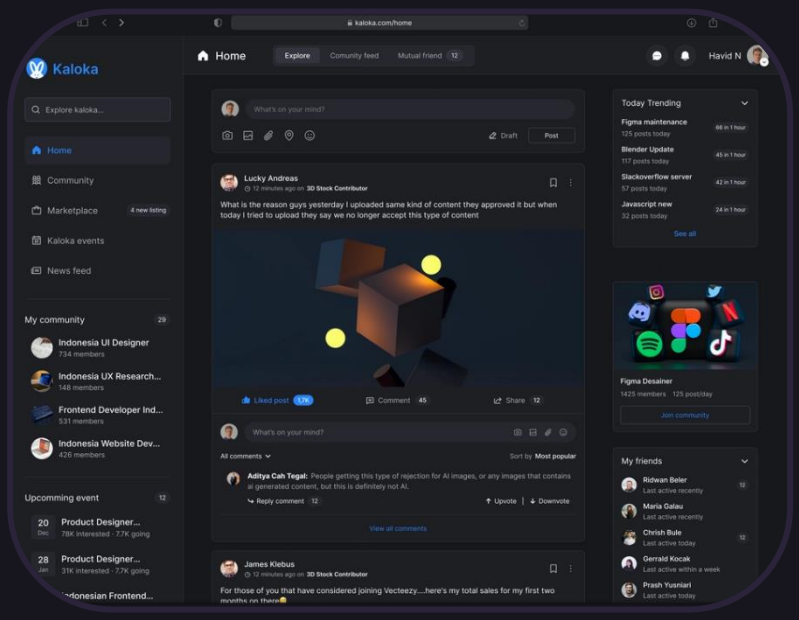


Figure 7 https://dribbble.com/shots/20053757-Kaloka-Social-Media-Platform

From this image, we like the structure of the navigation, with the sidebar that has links to the different pages, and the top bar with user image and notifications & settings buttons. The layout of the timeline was also very familiar, most modern social media sites have similar structure, with endless scrolling of content in their "feed". This is where we will display upcoming events and posts.

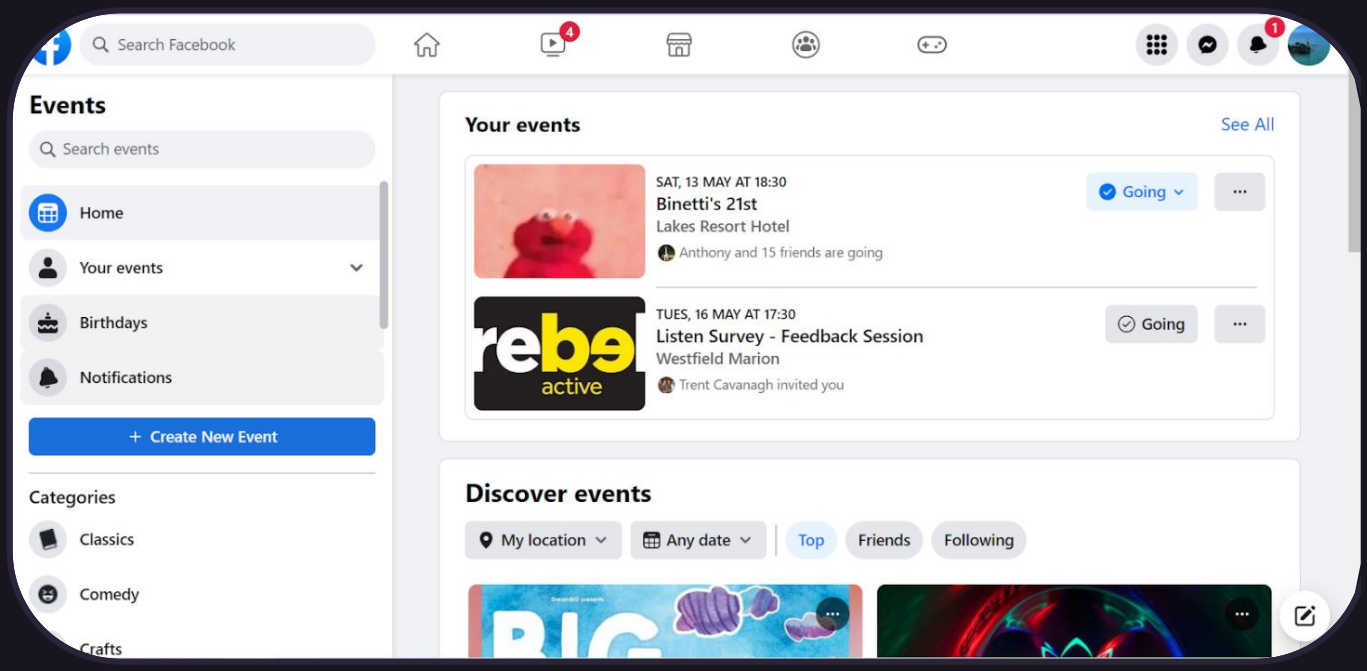


Figure 8 https://www.facebook.com/

The Facebook events page has a similar structure in regard to events, with the sidebar for navigation and the topbar for login information & notifications. We will adapt this layout with the "your events" and "discover events" into our own website.

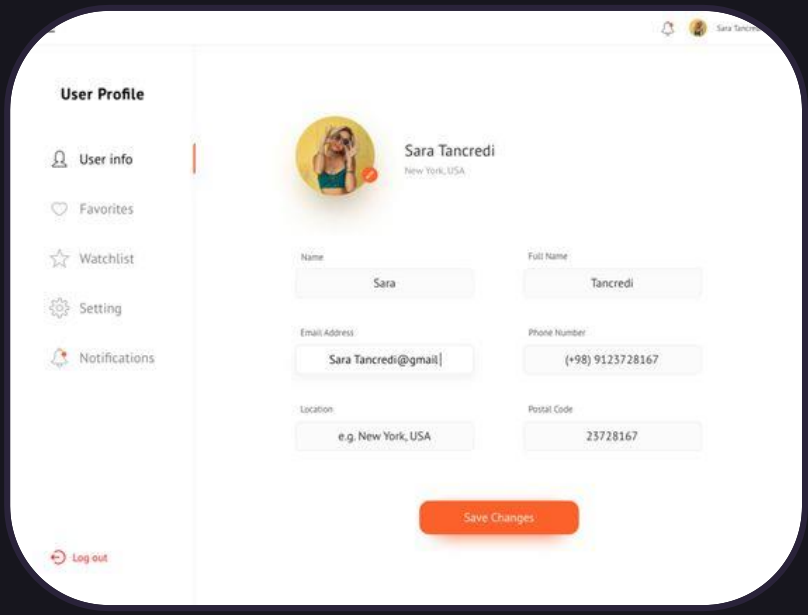


Figure 9 https://dribbble.com/shots/5923013-User-Profile

The layout of this settings page is clean and easy to use. This is a good layout that can be used as inspiration for our various settings and admin pages.

It also has the sidebar for navigation which we also plan on using as detailed in the other images.

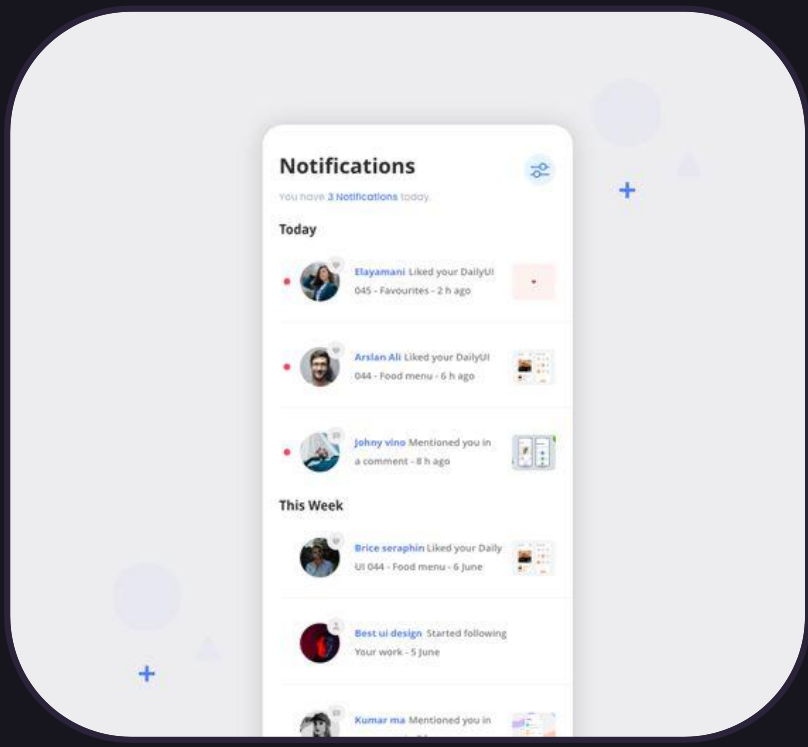


Figure 10 https://www.behance.net/gallery/81746037/DailyUI-047-Activity-feed?tracking_source=search%7Cnotification+design

The notification overlay can be set out like this example. Using this design as inspiration, we can display a user/club image to the left, with details such as post time and a brief description of the post to the right.

Additionally, having a status symbol to the left, where this image has the orange circle, can be used to show the user if the notification has been seen yet.

Features

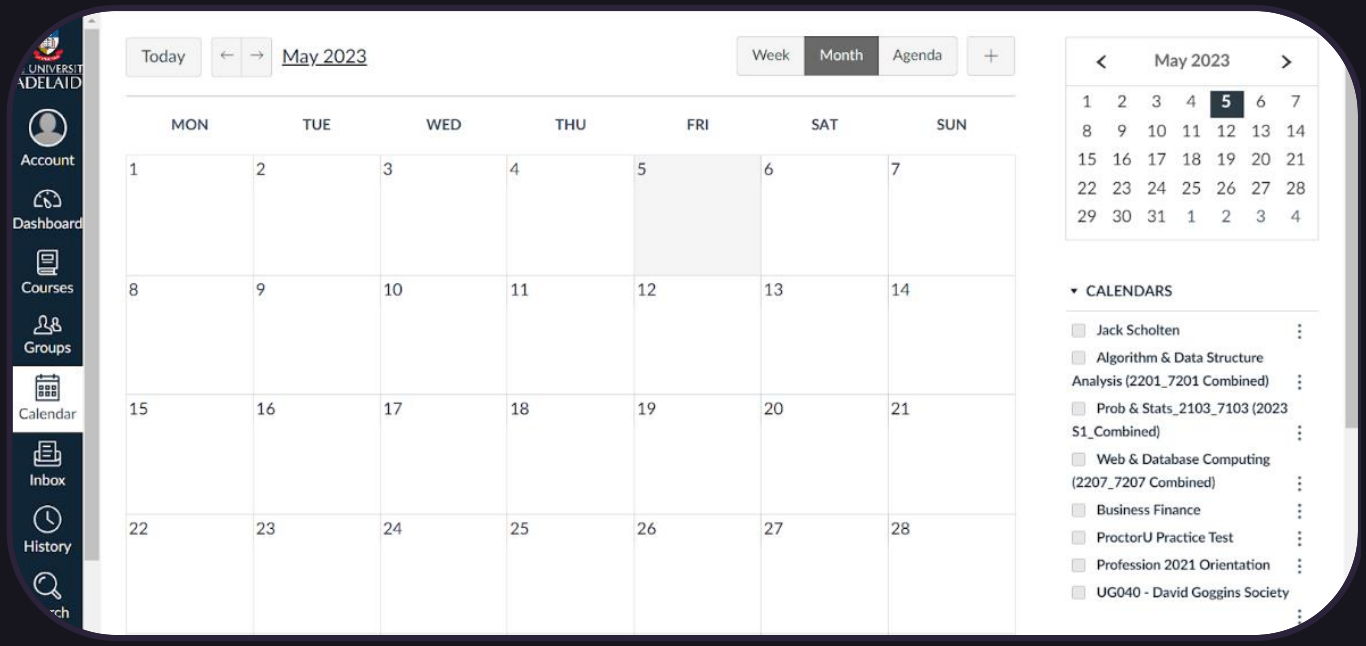


Figure 11 https://myuni.adelaide.edu.au/calendar

Website supports a calendar for scheduling and displaying. The user can access this calendar to view any of their own upcoming events.

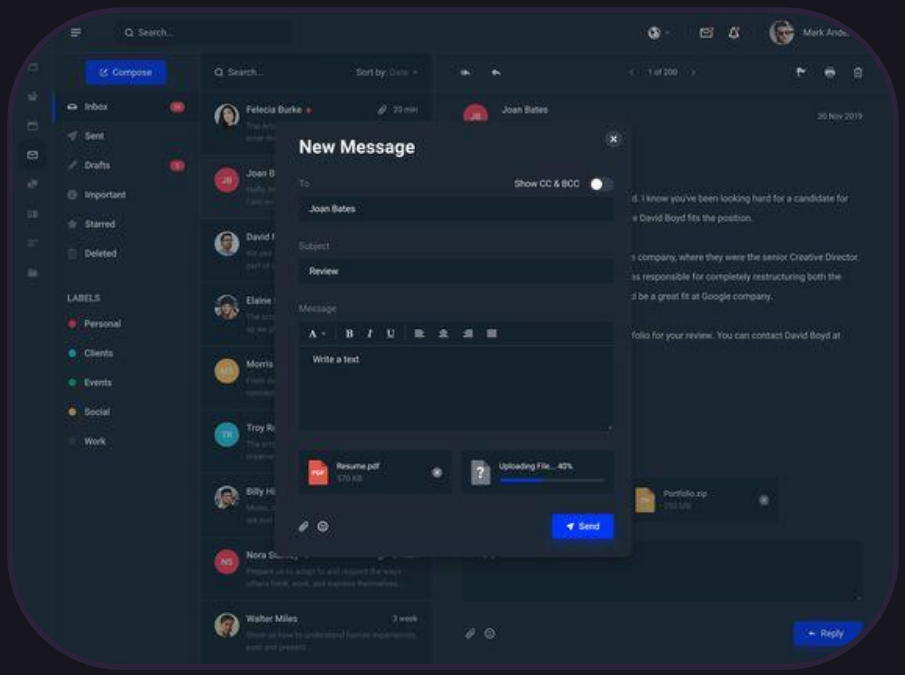


Figure 12 https://twitter.com/OpenPhone/status/1240759725509074949

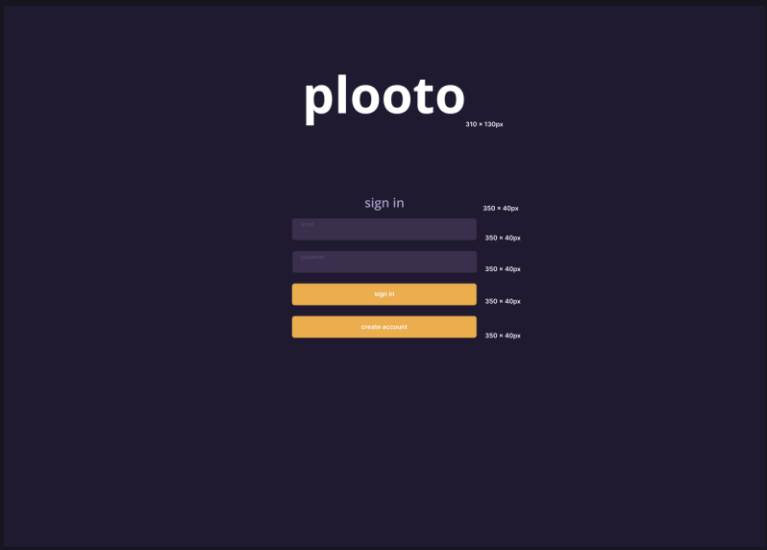
Plooto will also implement manageable notifications per club and event, that the user can opt in and out on. As is present on the figma design, we have administration for clubs, posts and events.

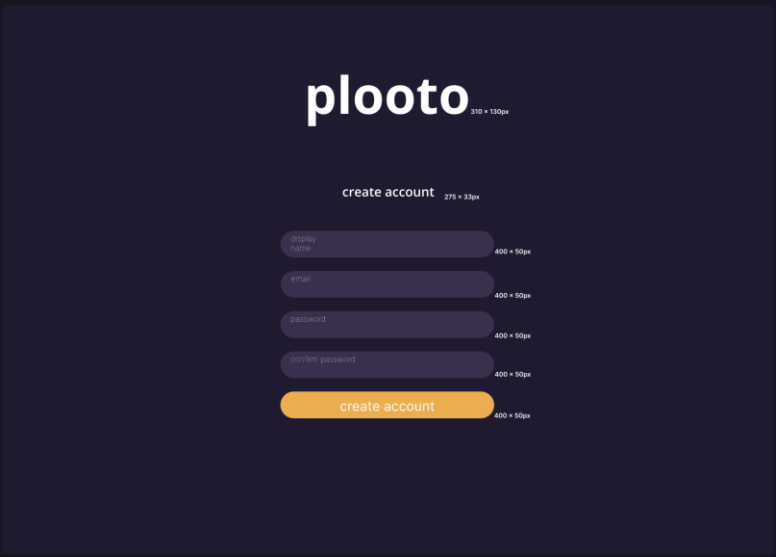
For editing controls, we will implement a similar sort of modal / popup, as it's unobtrusive and should be instant, without needing to leave the page. This will also preserve any page state and scrolling positions.

Page Mock-ups and User Interactions

Combing our research design with our decided application style of purple, in the theme of Plooto, we decided to make various page mock-ups for our website. These are not complete implementations; they exist as style guides and layout references for when we build our app. These designs were made using Figma.

Sign Up & Create Account

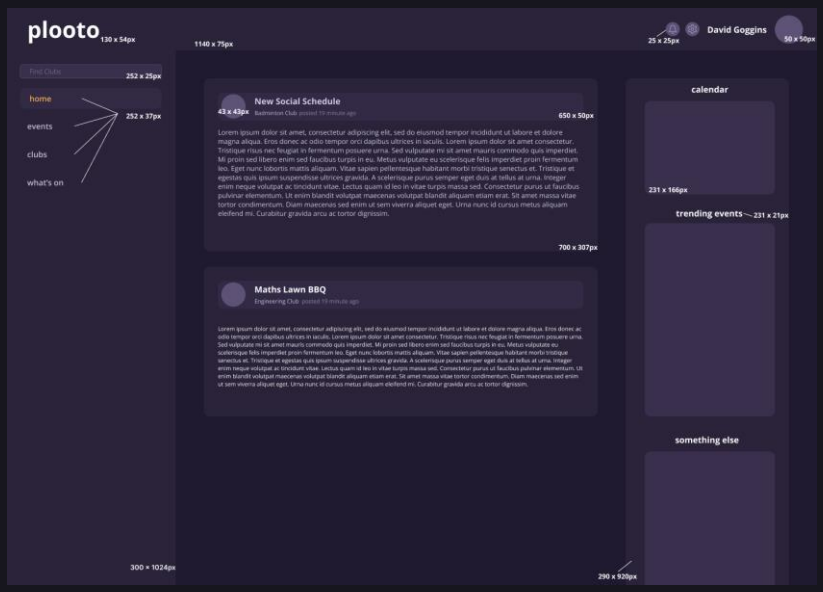


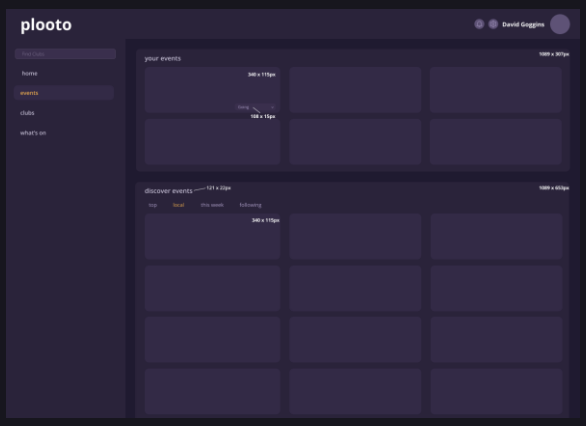


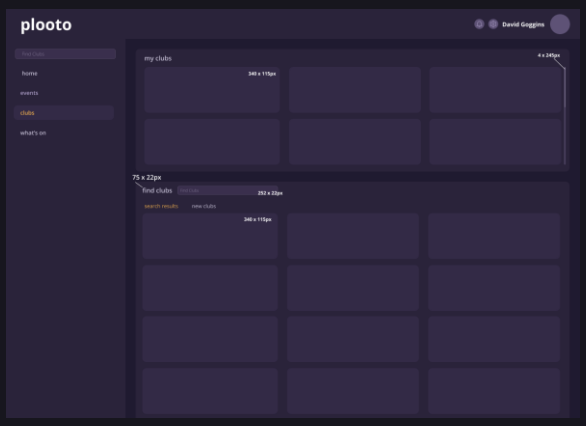
These pages were designed to be simplistic in nature, unobtrusive to the workflow and clear in what needs to be done. The interactions in this page are quite simple, the user would input their login information, and upon clicking login it would verify if their information were correct or not. If the login returns unsuccessful, then an error message would display. In the create account page, some basic validation would be necessary before creating the account, like confirming the password is the same. Potentially, this onboarding process could require a verified email, but for simplicity we will skip this step.

Clicking the login button will send a request to the server with the user input information. The response will either be a success or not, if it is not successful then the error will be displayed to the user, and upon success the user will be taken to the homepage. The create account button takes the user to the signup page. The signup button will send a request to the server to create a new user with the form input provided. Upon success it will return the new user ID and redirect the user to the homepage.

User Pages







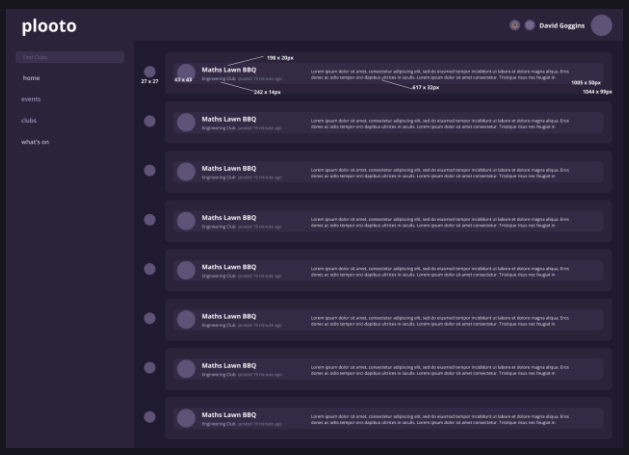
There is a consistent layout across these pages, with the navigation being present on the sidebar, and a settings, notification, and login information in the top bar on the right.

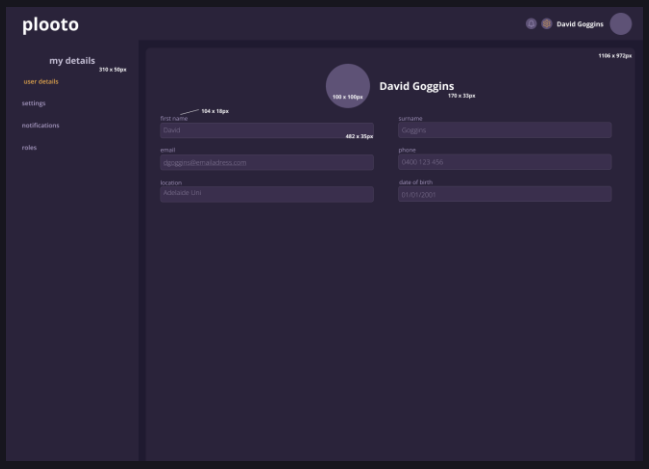
The home page is the page that the user is going to land on when they first visit the website and is likely the page that they will spend the most time on. The homepage features the familiar timeline/feed structure, where the main content is present in the middle and is scrollable. The aside in the homepage will hold a calendar, that will display upcoming events in a calendar format, and this will be directly interactable and could bring the user to a more complete, full-page calendar. There also exist other quick and easy information for the user like trending events, and room for more components that will be decided later.

The events and clubs page manages the user's RSVP'd events, and clubs that they are a part of. The idea of the clubs page is that in order to join a club, you would simply search for it, and on the same page you would join the club, or send a request to join the club if it's a private club. There would be multiple different ways to search for a club. A similar concept applies to events, where you would set your RSVP status to each event on this same page. Each club and event would have their own dedicated pages as well, and you could share this link to the event/club with someone else, and if they had an account, they would be able to see the same content.

The sidebar navigation will use vue-router to change the state of the application to render the correct view/page. The search buttons will use javascript filters to filter out the events/clubs and present the user with the results of the search query. The homepage will contain posts and upcoming events based on what they've responded interested too and what clubs they have joined.

Notifications & Settings



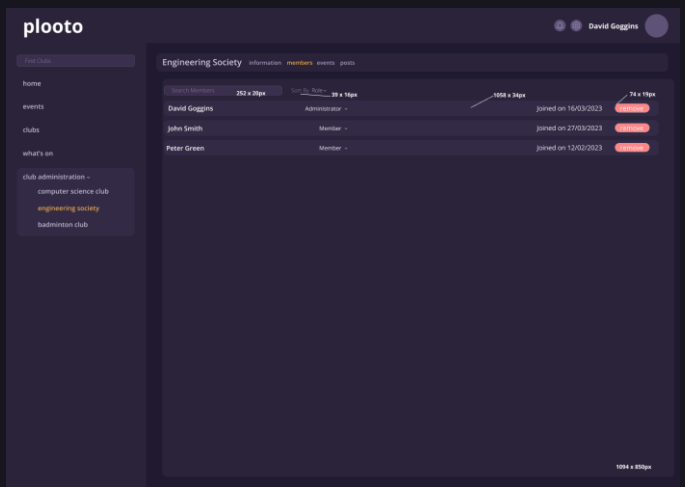


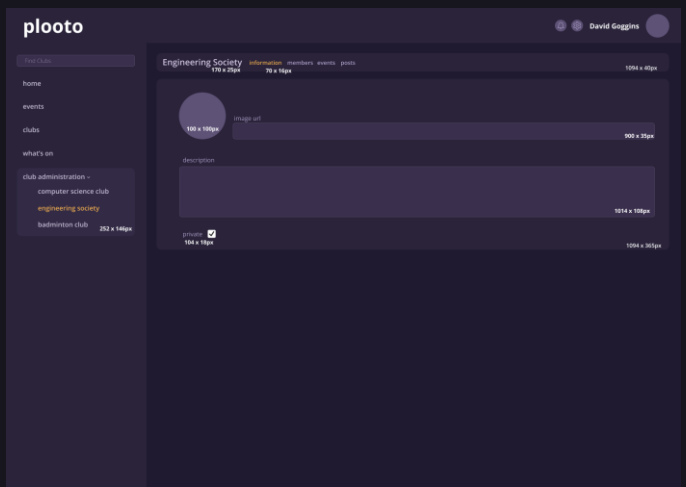
These two pages relate to each other because they are both the pages that exist in the top bar. In terms of the notifications page, this page would be present through a “show all” button in the notification popup, which is what you would first see when you clicked the notifications button. This would show read/unread status of each notification.

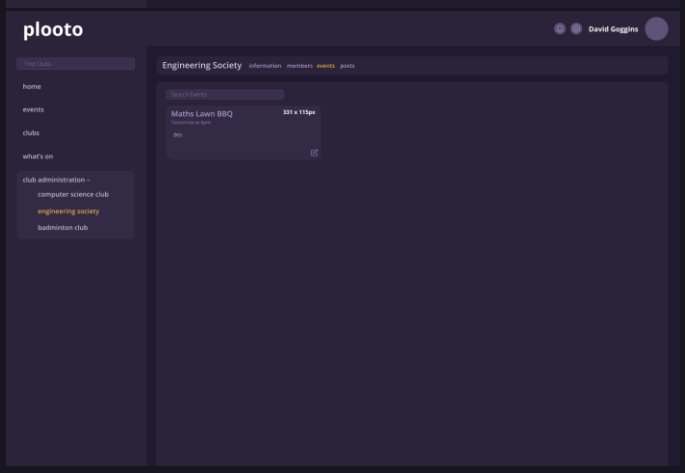
The settings page has multiple sub-sections and they’re all basic forms for user input. It is inside here that the user would set their email notification settings for each event, club, etc. In addition, if the user was designated as a system administrator, then the “admin” subsection is where they would perform these administrator tasks.

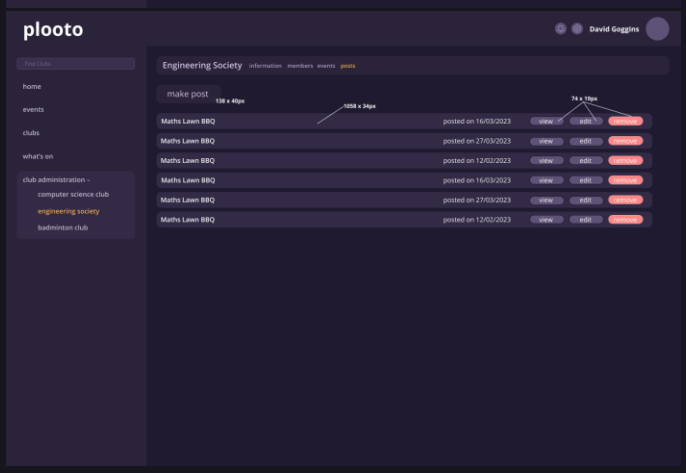
The settings page will have a “Save” button which will send an update request to the server, and the notifications will be loaded with a fetch request to the server.

Club Management

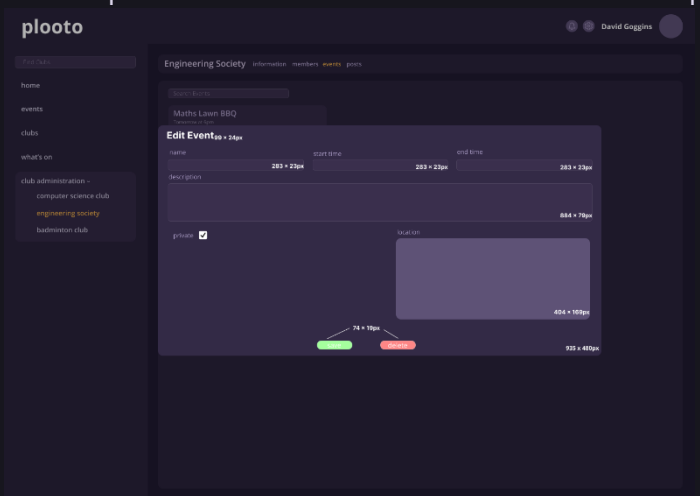


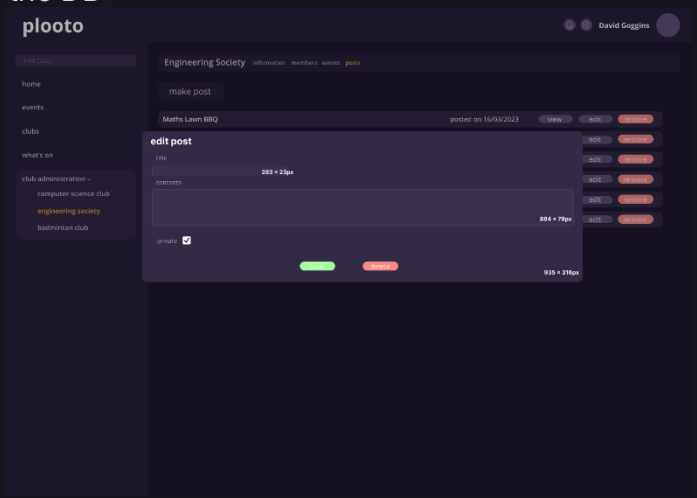






These pages offer management capabilities for the clubs. Each subsection has a different purpose. A club is able to create, edit, and delete events and posts. The club is also to have a look at who is a member of the club, change roles, and remove them. The editor controls for posts and events are popup modals, as demonstrated below. The button to remove a member or post will send an update request to the server and on success it will update the DB>





If these pages are too difficult to view in this document, they are available under /planning/pages/ in the GitHub repository.

Peer Feedback

Group 1

- Dark mode isn't the common choice among most people, need choice of light mode in settings
- Enjoyed the minimal design, made the page easy to scan and understand all the information
- Well made layout made it very user friendly to understand how to operate the website and how to navigate it

Key changes to investigate:

Add options for light and dark mode.

Group 2

- How to show if notifications have been read or not (Disappear or not), more memory if it doesn't disappear
- Colour pallet easy on the eyes
- minimalistic design is good
- make sure sign in pages have a ‘forgot password’ button
- Checking who can see what information (such as location and date of birth) about users
- Could add a page that shows where members are located to help plan events

Key changes to investigate:

Make notifications go away after being viewed, include forgot password button, ensure only certain people can see user data, location of club members viewable.

Group 3

- Do not like the design choice to make everything lowercase
- Should have a dedicated page for events/posts

Key changes to investigate:

Uppercase lettering, events/posts page

Review

Discussion of Feedback

From the group feedback we have identified some clear goals to improve our design:

- Correct capitalisation
- “Forgot Password” button
- Clearly show notifications being “read” or “unread”
- Events/posts page
- Light mode support
    - Light/dark mode toggle

There was feedback that was given that we have decided should not be a priority to resolve before what was listed above. These are:

- Admin ability to view location of users. We feel that this is not a critical feature in the operation of our website. Therefore, while it may be a useful feature to add, it is not seen as a high priority through the development process.
- Setting who can see user data. As it stands, nobody except the user will be able to see any data aside from their name. Similarly, to the last point, the implementation of this could open up opportunities for users such as club admin to better manage and assess the use of their clubs, however, as it stands is certainly not as important as other goals.

- Ensure that in the final design, there is “Undo” buttons for the actions, and that in each modal there is a clear “Close” or “Cancel” button.
- Add in shortcut buttons or shortcut keybinds to allow for quick and easy navigation.

Usability Heuristics

The usability heuristics provide a clear design outline. Following these leads to interfaces that are more user-centric, efficient, and enjoyable to use. It helps create positive user experiences will contribute to the overall success of the website.

The 10 usability heuristics and how we have sought to follow them are as follows.

1. Visibility of system status: Keep users informed about what's happening.
2. Match between system and the real world: Use familiar language and concepts.
3. User control and freedom: Allow users to undo actions and easily navigate.
4. Consistency and standards: Follow established conventions for a familiar experience.
5. Error prevention: Anticipate and prevent errors with helpful feedback.
6. Recognition rather than recall: Minimize the need for users to remember information.
7. Flexibility and efficiency of use: Provide shortcuts and customization options.
8. Aesthetic and minimalist design: Keep the interface simple and clutter-free.
9. Help users recognize, diagnose, and recover from errors: Clearly communicate errors and offer guidance.
10. Help and documentation: Provide accessible and comprehensive help resources.

In the following explanation of our design choices, decisions promoting these are covered.

Cognitive Load

Cognitive load focuses on designing the website with an understanding of human cognition and mental processes. Using this, it is important to consider how users perceive, process, and interact with information on the website. We aim to make our website intuitive, user-friendly, and supportive of various users' cognitive abilities.

Success in this will involve designing clear navigation, logical presentation of information, meaningful visual cues, and interfaces that minimize the cognitive load. Overall, this will be crucial in the overall design of the website, to ensure users can easily understand and navigate the website as well and make informed decisions with little required thought. Assessing this against the design and layout we have produced, we can make decisions on the performance of our website in this regard and potential adjustments. The key design strategies we developed to follow this are:

Minimal design - The minimal design reduces cognitive load by simplifying visual elements, decluttering the interface, and focusing on essential content, allowing users to navigate and process information more easily and efficiently. It eliminates distractions and provides a clear and intuitive user experience.

Information presented in a way users expect - presenting information in a way users are accustomed to, such as placing the like buttons underneath a post or menu similarly to how other social media layouts our, helps reduce cognitive load by leveraging users' familiarity with existing design patterns. By adopting familiar conventions, users can quickly understand and interact with the interface without needing to learn new behaviours or spend extra effort in navigating and using the platform.

Contrasting colours - contrasting colours for key buttons and information reduce cognitive load by providing clear visual cues, aiding in efficient scanning, and helping users quickly identify important elements and interact with the website.

Navigation sidebar – a navigation sidebar displaying all pages reduces cognitive load by providing easy and constant access to site navigation, allowing users to quickly find and switch between different sections without relying on memory or exploration.

Clear symbols and buttons - simple symbols on buttons reduce cognitive load by providing intuitive visual cues, enabling users to quickly understand button functions without relying on text or thinking, resulting in streamlined navigation and improved user experience. An example of this is a cog for a settings button.

Kinematic Load

Kinematic load relates to the physical effort required by users to interact with elements in a design, such as buttons or navigation menus. It is important to minimise kinematic load in web design to ensure ease of access and smooth user interaction. By making elements easily reachable and reducing physical effort, we can create a more user-friendly experience that promotes efficient navigation and interaction with the website. Assessing this against our proposed design, are strategies to succeed in this area include:

Close grouping of related objects – grouping of similar buttons, posts and other objects, provides users a straightforward process on carrying out actions on our website. Any buttons that may be used together or successively such as the filters for events are grouped together and away from other possibly conflicting processes. Similarly, information relating to specific events is contained only within that event and separated from others, makes it clear what information is related to each.

No 'extra' potential actions – buttons and information irrelevant to specific pages will be hidden when the user is on those pages. By doing this, we can minimise the potential actions in different areas, streamlining the user process and preventing an overload of decisions to make. This helps to direct users to key elements and information. A basic example of this is only showing account or notification settings when the user has clicked on 'settings' and keeping it off the screen at all other times.

Key controls always visible – key controls such as settings, account and page navigation will always be available to the user. These are buttons and information which we know will be frequently used or sought for. The sidebar and topbar will be used to contain these, so that their location is consistent, therefore making them easily accessible for the user.

Opportunities for Cognitive Load, Kinematic Load and Usability Heuristics

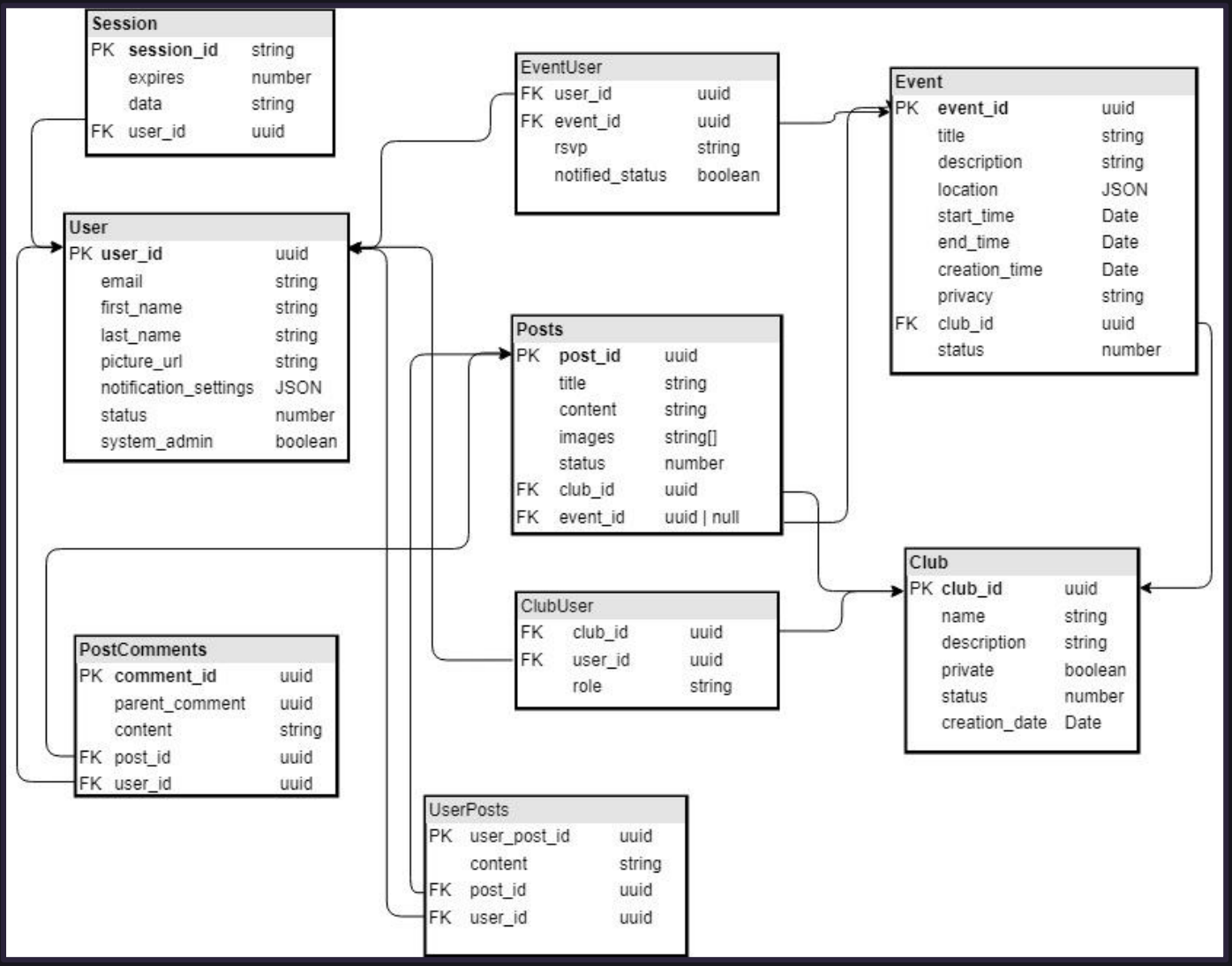
There are a lot of areas in our design which are successful in following these concepts. However, we have identified these as areas for more potential regarding cognitive and kinematic load:

- Make sure there is proper feedback on user actions, so when creating or updating posts/events it is clear to the user that the update has been successful.
- Keep the navigation consistent across the different pages, such that all navigation should only be done in the sidebar, and not in the top bar of a subpage. We currently have a sidebar, but it is important that this is properly used, and we look to keep key navigation on this sidebar and away from other areas on the page.

Finally, comparing our website to design to the usability heuristics, we've identified these areas for improvement:

- Need more statistics/information readily available to users/clubs, in accordance with "Visibility of system status".
- Perhaps change "administration" to "management", as some users may not understand that administration includes the functions for managing aspects of clubs ("Match between system and the real world").

Database Schema Design



This schema includes post comments, which was not included in the original design. These comments will be a feature that will only be implemented if we have enough time at the end, as they are not critical to our app's main functions.

Data Plan

The data plan exists in the repository under /planning/data_plan.md

As it is an interactive markdown document, it is best viewed inside a markdown renderer, however we've included screenshots from the document as rendered on GitHub.

Pages Plan

Any initial request will make a call to `api/user/data`. This will include any session data, if the user is not logged in this request will redirect them to `/login`. If the request is successful, then it will return all data about the user and all clubs in the system. Hopefully, there aren't too many clubs so these can be loaded into memory on the initial request.

Any event page will need to make a request to `/api/events` and this will load in all event data required into the events store in the client. The homepage `/` will load `/api/user/timeline`, which will contain events and posts relating to the user based on their profile.

The fetches to the `/update` api paths will come from their respective Editor components from the client. All requests will need to path an authentication check, so a middleware on server will be suitable, and this should include the user_id in the body, after verifying authentication. If they're not authentication, they should be redirected to `/login`, and ideally have a message saying "Invalid Session" or something of that sort.

Notes

All requests will be `POST` as every request will have a body or return a different response based on the user's session ID (which will be included in the session as is not a part of the request body).

The output is often described like this:

```
output {
        success: true,
        error: null
} | {
        success: false,
        error: string
}
```

As one object this would be

```
output {
        success: boolean,
        error: string | null,
}
```

User

▼ `POST` `` `/api/user/create` ``

Attempts to create a user with the given input body

Sources:

- `/views/Signup.vue`

```
body {
        email: string,
        first_name: string,
        last_name: string,
        display_picture: string
}
```

`display_picture` will be a URL of where the image is hosted. Will perform an `INSERT` query and return the ID, success will be false if the DB is down or some other IO operation cancels it.

```
response {
        success: true,
        ID: string
} | {
        success: false,
        error: string
}
```

▼ `POST` `` `/api/user/update` ``

Attempts to update a user update query given the user ID, and updates with the other data specified. This will also handle user deletions, in which case the `status` will be set to 0.

Sources:

- `/views/settings/Account.vue`

```
body {
        id: string,
        email: string,
        first_name: string,
        last_name: string,
        display_picture: string,
        notification_settings: any,
        status: 0 | 1
}
```

`display_picture` will be a URL of where the image is hosted.

```
response {
        success: true,
        error: null
} | {
        success: false,
        error: string
}
```

▼ `POST` `/api/user/timeline`

Loads relevant posts and events for a given user.
Sources:

- `/views/Home.vue`

```
body {
        user_id: string // this may also be gotten from session
}
```

```
response {
        events: Event[],
        posts: Post[],
} | {
        success: false,
        error: string
}
```

▼ `POST` `/api/user/data`

Loads relevant user information for a given user and all clubs for storage within local memory.
Sources:

- `/views/Home.vue`

```
body {
        user_id: string // this may also be gotten from session
}
```

UserInformation is the data form the `Users` table.

```
response {
        clubs: Club[],
        user: UserInformation
} | {
        success: false,
        error: string
}
```

▼ `POST` `/api/user/login`

Attempts to login with login data
Sources:

- `/views/Login.vue`

```
body {
        email: string,
        password: string
}
```

Will return success if the login was successful

```
response {
        success: true,
        error: null
} | {
        success: false,
        error: string
}
```

Club

▼ `POST` `/api/club/create`

Sources:

- club creation component

```
input {
        name: string,
        description: string,
        private: boolean,
}
```

Will perform an INSERT query and return the ID, success will be false if the DB is down or some other IO operation cancels it.

```
output {
        success: true,
        ID: string
} | {
        success: false,
        error: string
}
```

▼ `POST` `/api/club/update`

Sources:

- `/views/club-admin/Information.vue`

```
input {
        id: string,
        name: string,
        description: string,
        private: boolean,
        status: 0 | 1
}
```

```
output {
        success: true,
        error: null
} | {
        success: false,
        error: string
}
```

▼ `POST` `/api/club/join`

Sources:

- Join club component

```
input {
        club_id: string,
        user_id: string,
}
```

As this doesn't require a primary key, we can make use of an UPSERT query or MySQL's equivalent. This also means that the UI doesn't need to wait for a response to continue.

```
output {
        success: true,
        error: null
} | {
        success: false,
        error: string
}
```

▼ POST `/api/events`

Sources:

- event creation component

```
input {
        user_id: string
}
```

```
output {
        success: true,
        events: Event[]
} | {
        success: false,
        error: string
}
```

▼ POST `/api/event/create`

Sources:

- event creation component

```
input {
        title: string,
        description: string,
        location: {
                lat: number,
                lon: number
        },
        start_time: Date,
        end_time: Date,
        club_id: string,
        privacy: "MEMBER" | "OPEN" | "INVITE ONLY"
}
```

Will perform an INSERT query and return the ID, success will be false if the DB is down or some other IO operation cancels it.

```
output {
        success: true,
        ID: string
} | {
        success: false,
        error: string
}
```

▼ `POST` `/api/event/update`

Sources:

- `components/EventEditor.vue`

```
input {
        id: string,
        title: string,
        description: string,
        location: {
                lat: number,
                lon: number
        },
        start_time: Date,
        end_time: Date,
        club_id: string,
        privacy: "MEMBER" | "OPEN" | "INVITE ONLY",
        status: 0 | 1
}
```

Will perform an INSERT query and return the ID, success will be false if the DB is down or some other IO operation cancels it.

```
output {
        success: true,
        error: null,
} | {
        success: false,
        error: string
}
```

▼ `POST` `/api/attendance/upsert`

Sources:

- In the `EventCard` component, when the user changes their event status

```
input {
        event_id: string,
        user_id: string,
        rsvp: string,
}
```

As this doesn't require a primary key, we can just send the variables needed and let the backend decide whether it needs to do an insert or update. There might be a UPSERT type query in MySQL, or something similar that we can make us of.

```
output {
        success: true,
        error: null
} | {
        success: false,
        error: string
}
```

Posts

▼ `POST` `/api/post/create`

Sources:

- post creation component

```
input {
        title: string,
        content: string,
        images: string[],
        club_id: string,
        event_id: string | null
}
```

Will perform an INSERT query and return the ID, success will be false if the DB is down or some other IO operation cancels it.

```
output {
        success: true,
        ID: string
} | {
        success: false,
        error: string
}
```

▼ `POST` `/api/post/update`

Sources:

- post editor component

```
input {
        id: string,
        title: string,
        content: string,
        images: string[],
        club_id: string,
        event_id: string | null,
        status: 0 | 1
}
```

```
output {
        success: true,
        error: null
} | {
        success: false,
        error: string
}
```