CS4825: MSci Team Project

Final Report

Carlos Matos

Giorgios Koutsoukos

2022/23

Team members: **Neka Toni-Uebari, *Christopher Buss (left)***

Introduction

The main objective mentioned in this document is performing the tasks of planning, designing, implementation, testing and deployment of a system to allow for Royal Holloway, University of London (RHUL), to organise and collaborate relevant information and events between the University and the Industry Advisory Board (IAB). It is also mentioned that this project went through unforeseen circumstances of losing a team member in the process, leaving only one active developer; thus workload and prioritisation of tasks had to be re-considered.

Problem

Currently, the organisation of an Annual General Meeting (AGM) between RHUL and the IAB is cumbersome. At least ten companies comprise this board at any given time, and therefore collating and organising all the relevant information required takes a lot of manual effort. This can cause problems as finding an available time for all relevant parties involved is a tricky effort, and any documents that are produced by the IAB can not be easily viewed by any interested parties. Additionally, the current approach of a singular three-hour meeting annually means that it is difficult to get feedback at any other point throughout the year where required. As a result, any concerns or extra information required by RHUL can end up being overlooked, creating a worse experience for the students at RHUL, and ultimately the skills developed by the students may not match the qualities that the wider industry desires.

Brief

The client wants a system that will allow for members of RHUL to easily obtain and request relevant information from representatives of the IAB, such as calendar information to arrange an AGM. The system should also allow for members to be able to easily view any relevant information that they have not viewed since their last visit to the site. Site managers should be able to handle administration of the system; the monitoring of the existing accounts and their uploaded/shared resources. Furthermore, the system should allow group leaders to handle groups and collaborative spaces where resources can be shared by the representatives who are part of this group.

User Characteristics

For all users in the system, there will be the basic requirements of user credentials to log into the system. The admin will be responsible for the creation of accounts for users and other staff within the system, rather than the new user creating the account themselves. There is also the assumption that the users of the system do not have extra expertise, thus the usability of the system is important for the application, alongside with documentation to guide the user on each feature the user can interact with in the system. It is also vital to mention that the role higher inherits permissions and actions that the role lower can do.

Four different types of users will be active in this system. These are:

- **Admin -** the highest role of the system; this user is expected to have access to all parts of the system and has the most control, in terms of the creation and deletion of users. Also, the user will be able to monitor actions within the system and initiate features that other users of lower levels can interact with.
- **Group Admin** - an admin that has full control over a subset of the system, e.g. an admin for a Computer Science group would have full control over any users and representatives added to the Computer Science group, but would be unable to make modifications, for example, the group designed for Social Sciences.
- **Representative -** the role of a representative of an IAB-affiliated company in the system; this user does not have permission to manipulate users i.e. creating new users and deleting users. The main focus of this user involves the uploading of particular resources into the portal.
- **User -** the lowest role of the system; this user will be very limited in terms of being able to manipulate data within the system. The main focus of this user is viewing resources provided by representatives and admins.

Functional Requirements

The requirements were obtained by a combination of team meetings and analysing existing systems: more specifically the functionalities which aligned with the client's needs and the current problem. The communication of team meetings with the RHUL staff helped provide clarity to the client's needs which would make the functional requirements more accurate. Listed below are the functional requirements of the system.The rightmost cells in the tables denote the importance of each requirement for implementation. As mentioned by Mike Holcombe[1], the following define those measures of importance:

M: a mandatory requirement (something the system must do)
D: a desirable requirement (something the system preferably should do)
O: an optional requirement (something the system may do)

The requirements are grouped by user type.

Admin

ID	Description	Priority
1	Admins can create new accounts for all other user types.	M
2	Admins can add/remove accounts to group collaboration spaces.	M
3	Admins can log into their account.	M
4	Admins can delete existing accounts.	D
5	Admins can schedule calendar events.	M
6	Admins can modify calendar events (change, delete, etc.).	M
7	Admins can create new group collaboration spaces and pages.	M
8	Admins can delete group collaboration spaces and pages.	M
9	Admins can create new surveys/polls.	M
10	Admins can edit their details (profile information, password, etc.).	D
11	Admins can host virtual meetings online.	D
12	Admins can set up specific integrations with different platforms.	D
13	Admins can view details of other users.	O
14	Admins can delete documents of other users.	O

[1] L., Holcombe.W.M. (2008) *Running an agile software development project*. Oxford: Wiley-Blackwell.

15	Admins can send notifications to users.	O
16	Admins can send direct messages to other users.	O

Group Admin

ID	Description	Priority
17	Group Admins can create new representatives and users.	M
18	Group Admins can add representatives and users to their group.	M
19	Group Admins can remove representatives and users from their group.	M
20	Group Admins can log into their account.	M
21	Group Admins can schedule calendar events.	M
22	Group Admins can edit calendar events (change, delete, etc.)	M
23	Group Admins can create new group pages.	M
24	Group Admins can delete group pages.	M
25	Group Admins can create new surveys/polls.	M
26	Group Admins can edit their details.	D
27	Group Admins can host virtual meetings online.	D
28	Group Admins can set up specific integrations with different platforms.	D
29	Group Admins can view details of other users.	O
30	Group Admins can delete documents of other users.	O
31	Group Admins can send notifications to users.	O

Representative

ID	Description	Priority
32	Representatives can log into their account.	M
33	Representatives can submit answer to surveys/polls.	M

34	Representatives can upload new documents into the portal.	M
35	Representatives can view calendar events.	M
36	Representatives can upload their availability for meetings.	M
37	Representatives can view pages/collaboration spaces shared to them.	M
38	Representatives can delete existing documents owned by them.	M
39	Representatives can edit their details.	D
40	Representatives can view the results of polls.	D
41	Representatives can view existing documents in the portal.	D
42	Representatives can view notifications from admins.	O

User

ID	Description	Priority
43	Users can log into their account.	M
44	Users can view calendar events.	M
45	Users can view pages/collaboration spaces shared to them.	M
46	Users can view existing documents in the portal.	D
47	Users can view notifications from admins (sent to users).	O
48	Users can join virtual meetings.	O
49	Users can view the results of polls.	O

User Stories

Listed below are the user stories of the system. The rightmost cells in the tables denote the importance of each user story for implementation. The following define those measures of importance:

M: a mandatory user story (something the system must do)
D: a desirable user story (something the system preferably should do)
O: an optional user story (something the system may do)

The user stories are grouped by user type.

Admin

ID	Description	Priority
1	As an admin, I would like to create new accounts for all other user types, so that a new user is added into the system through my control.	M
2	As an admin, I would like to add and remove accounts from group collaboration spaces, so that only the relevant accounts are involved in a group collaboration space.	M
3	As an admin, I would like to log into my account, so that I have my own personal access into the system with my own details and full permissions.	M
4	As an admin, I would like to delete existing accounts, so that only the relevant and active accounts are existing in the system.	D
5	As an admin, I would like to schedule calendar events, so that I can initiate meetings and notifications for other staff members.	M
6	As an admin, I would like to modify calendar events (change, delete, etc.), so that I can make any corrections that need to be made to an event.	M
7	As an admin, I would like to create new group collaboration spaces and pages, so that other staff members (if permitted) would have access to these spaces.	M
8	As an admin, I would like to delete group collaboration spaces and pages, so that only relevant and active spaces and pages are existing.	M
9	As an admin, I would like to create new surveys and polls, so that I can receive the opinion of the staff on a particular matter.	M
10	As an admin, I would like to edit my details (profile information, password, etc.), so that I have up-to-date and relevant information about myself.	D

11	As an admin, I would like to host virtual meetings online, so that I can initiate virtual communication with other staff members about a particular meeting topic.	D
12	As an admin, I would like to set up specific integrations with different platforms, so that I have more customizability of my own pages.	D
13	As an admin, I would like to view the details of other users, to monitor the users' information and ensure it is correct.	O
14	As an admin, I would like to delete the documents of other users, so that only relevant documents or documents that abide by the guidelines are the only documents existing in the system.	O
15	As an admin, I would like to send notifications to users, so that the users are informed about any important information or upcoming events.	O
16	As an admin, I would like to send direct messages to other users, so that there is a line of private online communication between myself and a particular user.	O

Group Admin

ID	Description	Priority
17	As a group admin, I would like to create new representatives and users, so that I have the permission to introduce new users into the system, if needed.	M
18	As a group admin, I would like to add representatives and users to their group, so that the groups have the correct, active and relevant representatives.	M
19	As a group admin, I would like to remove representatives and users from their group, so that the groups have the correct, active and relevant representatives.	M
20	As a group admin, I would like to log into my account, so that I have personal access into the system with my corresponding permissions.	M
21	As a group admin, I would like to schedule calendar events, so that I can initiate meetings and notifications for other staff members.	M
22	As a group admin, I would like to edit calendar events (change, delete, etc.), so that I can make any corrections that need to be made to an event.	M
23	As a group admin, I would like to create new group pages, so that I can provide the relevant information to those with access.	M

24	As a group admin, I would like to delete group pages, so that only active and relevant pages should be existing.	M
25	As a group admin, I would like to create new surveys/polls, so that I can receive the opinion of the staff on a particular matter.	M
26	As a group admin, I would like to edit my details, so that I have up-to-date and relevant information about myself.	D
27	As a group admin, I would like to host virtual meetings online, so that I can initiate virtual communication with other staff members about a particular meeting topic.	D
28	As a group admin, I would like to set up specific integrations with different platforms, so that I have more customizability of my own pages.	D
29	As a group admin, I would like to view details of other users, to monitor the users' information and ensure it is correct.	O
30	As a group admin, I would like to delete documents of other users, so that only relevant documents or documents that abide by the guidelines are the only documents existing in the system.	O
31	As a group admin, I would like to send notifications to users, so that the users are informed about any important information or upcoming events.	O

Representative

ID	**Description**	**Priority**
32	As a representative, I would like to log into their account, so that I have personal access into the system with my corresponding permissions.	M
33	As a representative, I would like to submit answer to surveys/polls, so that I have made a contribution into the opinion on a subject matter of the poll.	M
34	As a representative, I would like to upload new documents into the portal, so that it can be accessed and viewed by other users of the portal.	M
35	As a representative, I would like to view calendar events, so that I am informed about the upcoming meetings and other events.	M
36	As a representative, I would like to upload their availability for meetings, so that I can inform the host and ensure that they host a meeting at a time most suitable for the representatives involved.	M
37	As a representative, I would like to view pages/collaboration spaces	M

	shared to me so that I am informed about the existing pages and spaces that I have personal access to.	
38	As a representative, I would like to delete existing documents owned by me, so that only documents that I want (and are relevant) exist in the system.	M
39	As a representative, I would like to edit my details, so that I have up-to-date and relevant information about myself.	D
40	As a representative, I would like to view the results of polls, so that I am informed about the overall opinion on a particular subject matter of a poll.	D
41	As a representative, I would like to view existing documents in the portal, so that I can be fully informed about the information in these documents in the portal.	D
42	As a representative, I would like to view notifications from admins, so that I am informed about any important information or upcoming events.	O

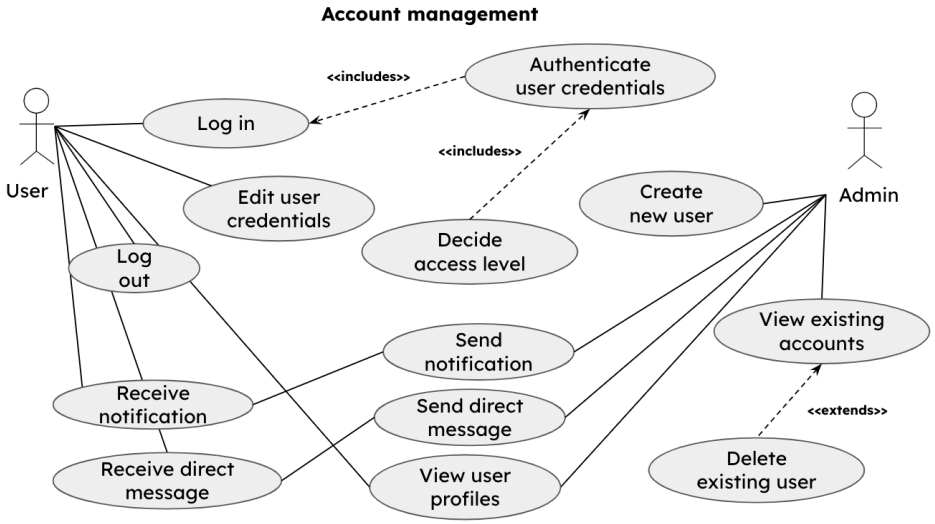
User

ID	Description	Priority
43	As a user, I would like to log into their account, so that I have personal access into the system with my corresponding permissions.	M
44	As a user, I would like to view calendar events, so that I am informed about upcoming meetings and other events.	M
45	As a user, I would like to view pages/collaboration spaces shared to me, so that I am informed about the existing pages and spaces that I have personal access to.	M
46	As a user, I would like to view existing documents in the portal, so that I am aware and informed of the existing documents in the system.	D
47	As a user, I would like to view notifications from admins (sent to users), so that I am informed about any important information or upcoming events.	O
48	As a user, I would like to join virtual meetings, so that I am a participant of virtual communication with other staff members about a particular meeting topic.	O
49	As a user, I would like to view the results of polls, so that I am informed about the opinion of those who participated in the poll on a particular matter.	O

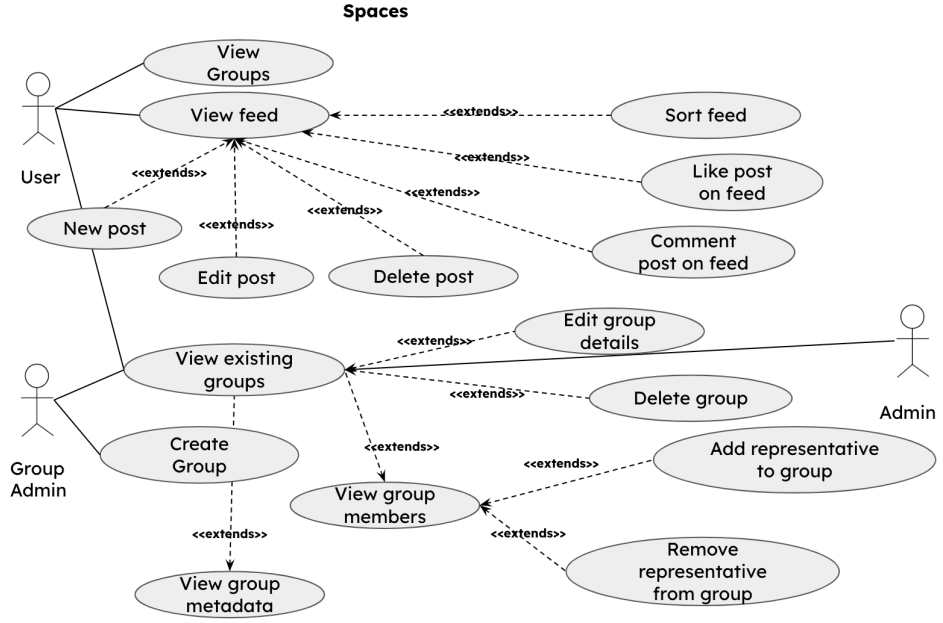
Use Case Analysis

The use case analysis shows the demonstration of the process of the requirements performed by users of different roles in the system. An important note in this diagram is that the higher role inherits permissions and accesses that the lower level can do e.g. a group admin would already be able to do actions a user can do. The rank of levels are (lowest to highest): user, representative, group admin, admin.

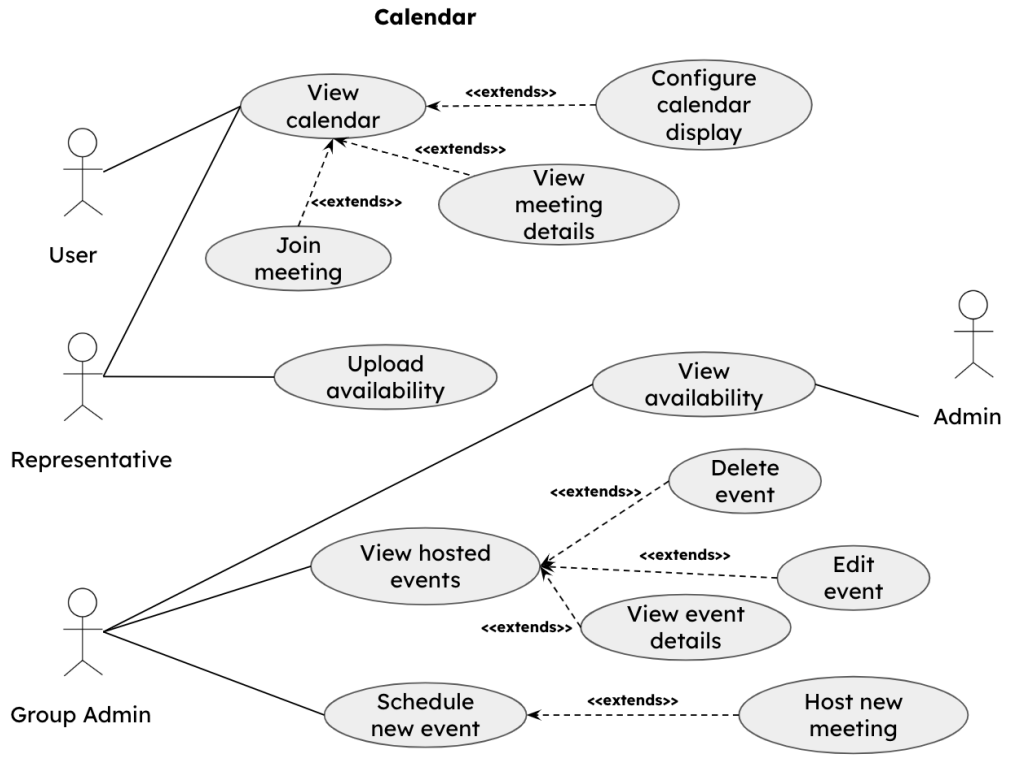
Account management



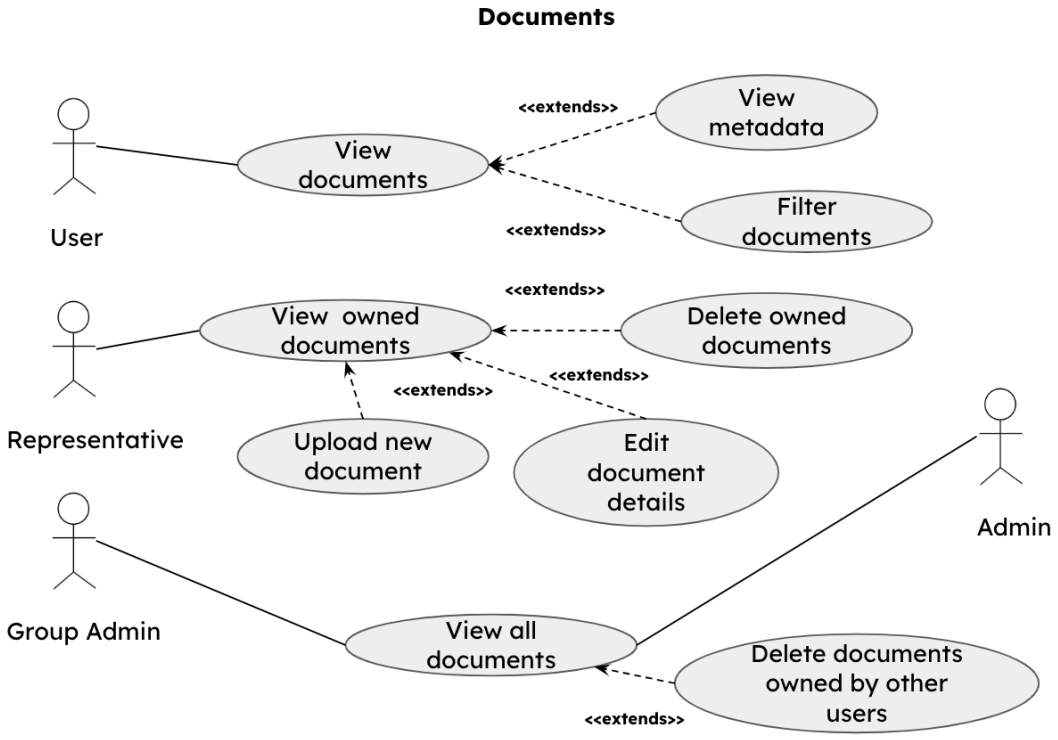
Spaces



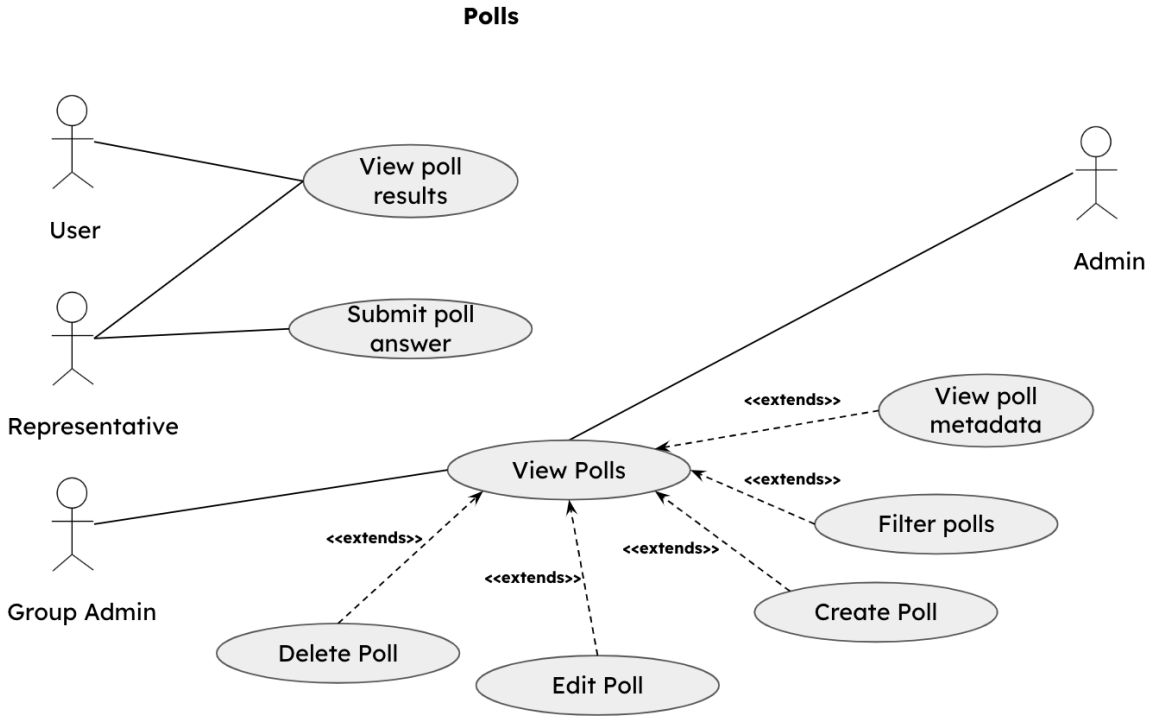
Calendar



Documents

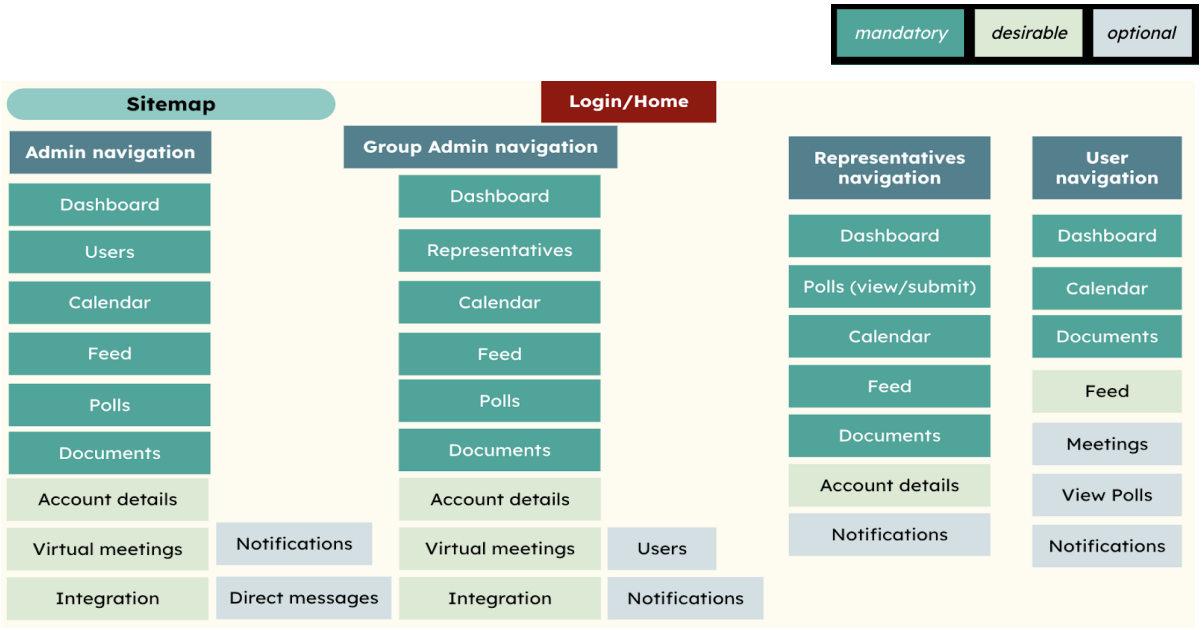


Polls


Polls
User
View poll results
Submit poll answer
Representative
Admin
View poll metadata
<<extends>>
View Polls
<<extends>>
Filter polls
<<extends>>
Group Admin
<<extends>>
<<extends>>
Create Poll
Delete Poll
Edit Poll

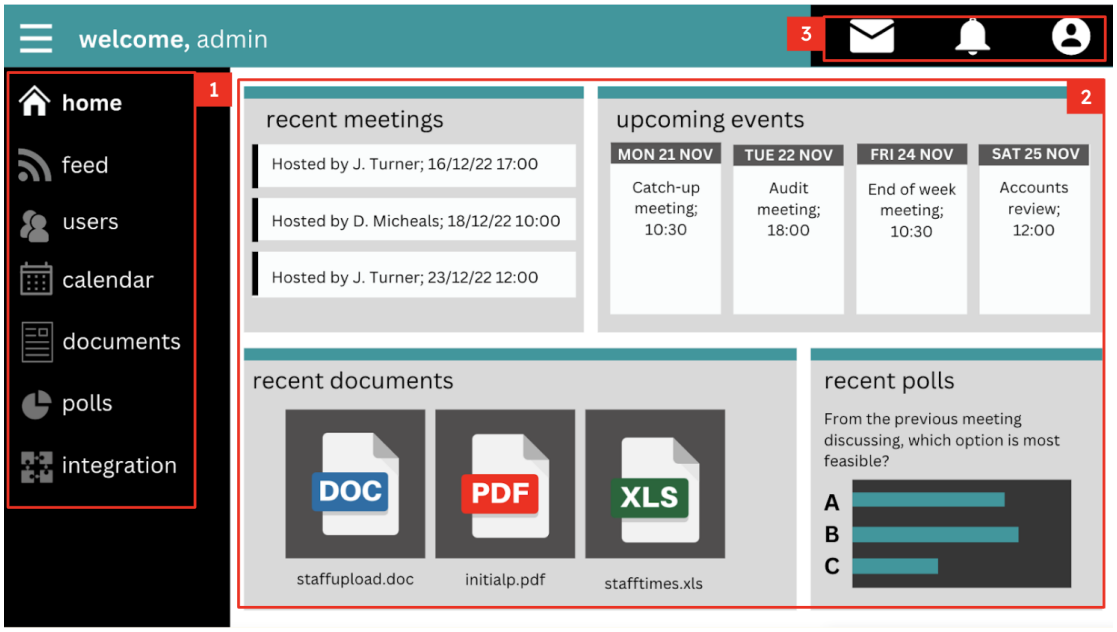
Design

The sitemap was designed on the basis of the user roles and considering the different permissions and access to particular pages. Each component of the navigation is also set by priority (based on the user requirements).



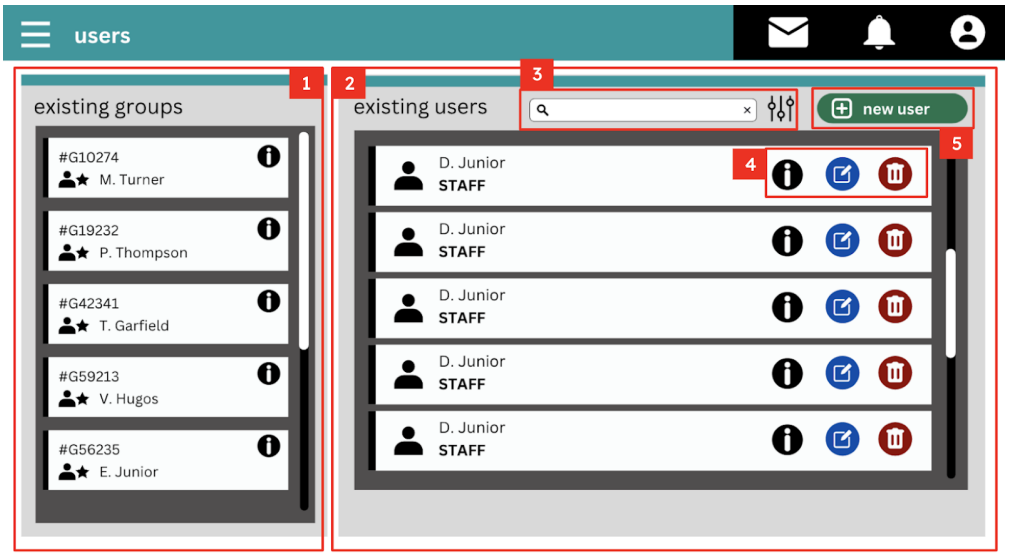
Storyboards

Dashboard - the home of the navigation which welcomes the users and presents an overview of the meetings, event, documents and polls **(admin perspective)**



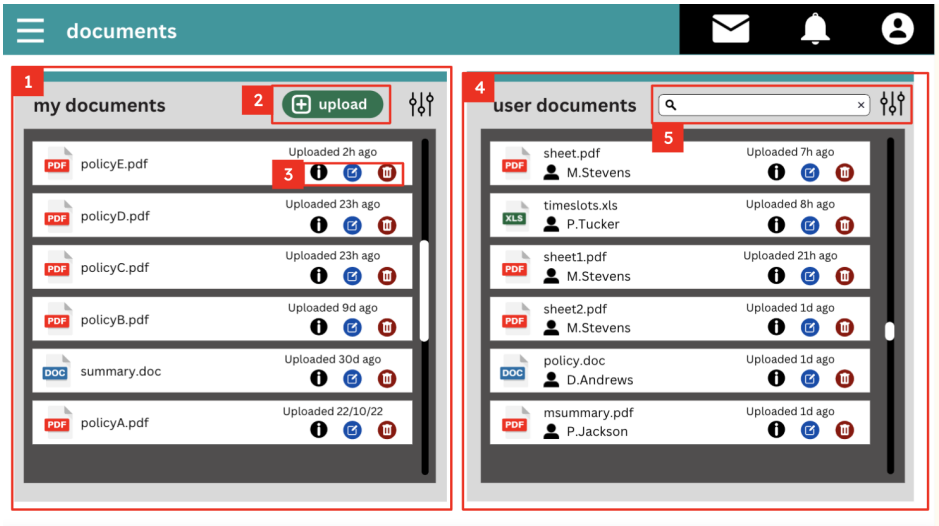
1. The menu
2. The overview of recent activity or upcoming events
3. The profile bar - direct messages, notifications and the user profile

Users - the page for managing users and groups within the system **(admin perspective)**



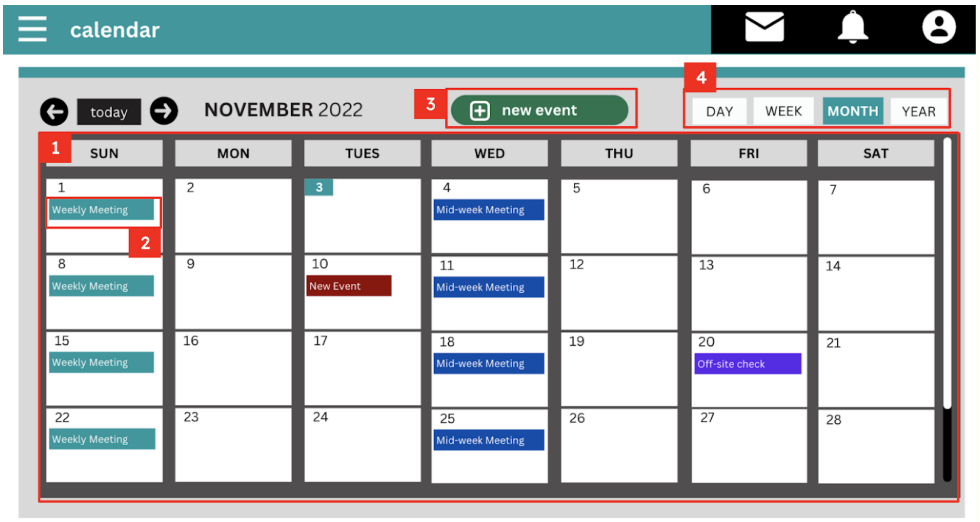
1. The existing groups in the system (including button for metadata about the group
2. The existing users of the system - their name and role
3. Searching and filter for users
4. Management of users - metadata of user, edit user or delete user from system
5. Create new user

Documents - the page for managing documents within the system **(admin perspective)**



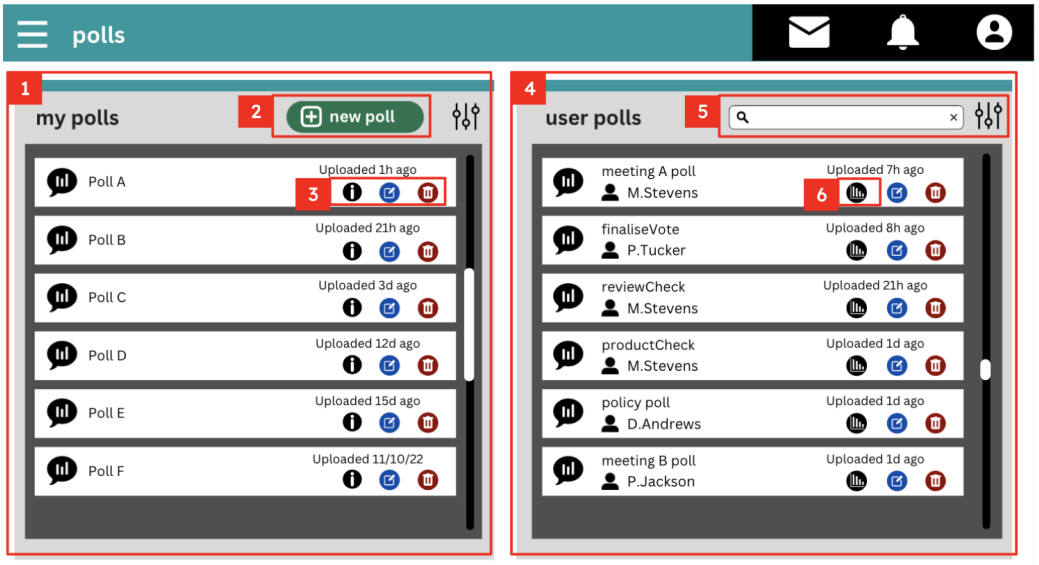
1. Documents owned by the user - shows document name, its format type, and date of upload
2. Uploading a new document
3. Management of document - metadata of document, edit document or delete document from system
4. All documents of the system
5. Search and filter for documents

Calendar - the page for viewing the events in the calendar (e.g. upcoming meetings)



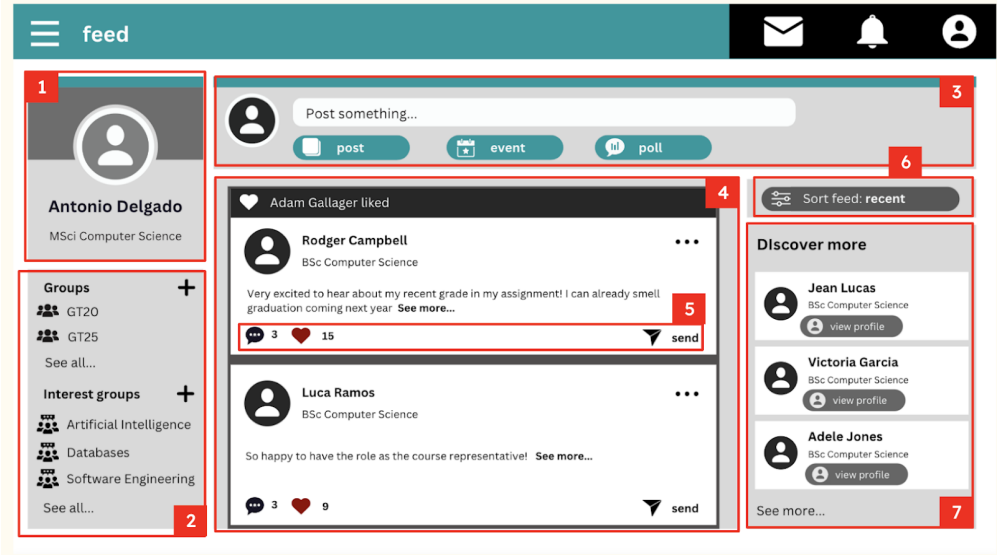
1. Calendar of events relevant to user (and any groups they are part of)
2. Select event - edit or delete event (only if they host the event); view details of event
3. Create event (hosting meetings limited to group admins)
4. Configuring calendar display - by day, week, month or year

Polls - the page for managing polls within the system **(admin perspective)**



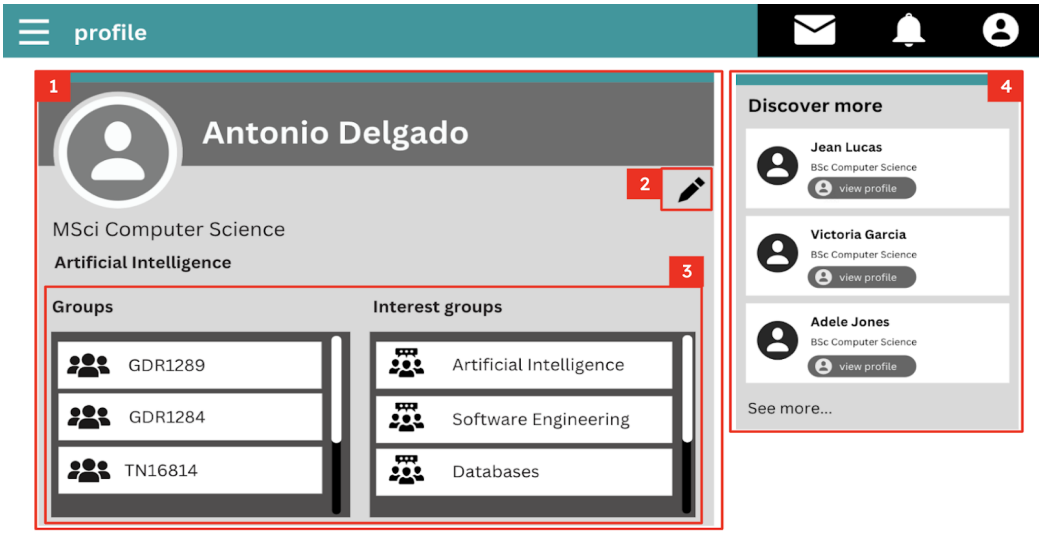
1. Polls owned by the user - shows poll name and date of upload
2. Creating a new poll
3. Management of poll - metadata of poll, edit poll or delete poll from system
4. All polls of the system
5. Search and filter for polls
6. View results of poll

Feed - the page for viewing and interacting with users' posts within a space of a group.



1. The user profile
2. The groups and interest groups they are part of and posts within that group(can add new groups if given permission e.g. is an admin or group admin)
3. Creating a new post (can be text, an event or a poll; depending on permissions)
4. The scrollable feed; shows posts relevant to the user and of the groups they are in (includes posts liked by other users they are in a group with as well)
5. Interaction with post (like, comment or share)
6. Sorting feed (e.g. recent, top posts)
7. Discovering more users (i.e. checking out their profiles)

Profile - the user's profile and other details



1. The overview of the user's profile (their name and their role in staff/university)
2. Edit profile details
3. Groups and interest groups the user is part of

4. Discovering more users (i.e. checking out their profiles)

The overall design of the pages was intended to follow a minimalist design with a consistent colour scheme and consistent uses of icons. It is essential for this design to offer accessibility, ensure usability and help the user to be able to memorise (without effort) or learn actions and interactions within the system with ease of use.

Timeline

*(The **gantt chart** can be found in the Appendices)*

The milestones in this project follow a Waterfall methodology, where the project tasks are broken down into activities which follow phases in a linear sequence. This allows for the total cost of the project to be accurately estimated as well as providing project management that is straight forward. The following phases in the Gantt chart were:

- **Initiation (requirements) -** the phase of gathering the requirements of the project. A thorough understanding and analysis of the client's requirements is gathered through the specification: the use case analysis, user requirements and user stories.
- **Content -** the "mini" phase which transitions between the requirements and the design. A section of the project was used on the comparison of different tools under a criteria of non-functional requirements. Thus a conclusion of the tool to use for the project and the environment to implement the project on.
- **Design -** the phase of the technical solution to the project, more specifically the user interface designs based on the project's requirements i.e. the product of storyboards for different components of the project. The UI design is reviewed to ensure it is accurate to the product's requirements as well as following accessibility.
- **Implementation / Testing -** the phase of implementation of the project. The coding of the application is done based on the requirements and specification. Each component of the system is implemented (in order of priority and starting from the foundation e.g. login system is required first, then a dashboard etc.)
- **Review (verification and deployment) -** the phase of verification and maintenance. The overall project is reviewed with any polished changes and adjustments that are required to be made before deployment.

Schedule and Milestones

No.	Task Title	Start Date	End Date	Duration (days)	Description
1	**Initiation (Requirements)**				
1.1	Introduction to project	10/10/2022	17/10/2022	7	Introduction meeting - team members informed about the project background and context and thus the client's needs.
1.2	User requirements	17/10/2022	21/10/2022	4	User requirements gathered from the client's needs, broken down by role and priority.
1.3	User stories and use case analysis	17/10/2022	24/10/2022	7	User stories (by priority) and use case analysis gathered from the client's needs.
2	**Content**				
2.1	Tool comparison	24/10/2022	4/11/2022	11	Comparison of tool options based on criteria of non-functional requirements
2.2	Presentation of tool comparison	31/10/2022	4/11/2022	4	Presentation evaluating each tool and its pros and cons. Final recommendation given in the presentation
2.3	Pending release of development environment for chosen tool	4/11/2022	18/11/2022	14	Release of the development environment of the chosen tool for implementation purposes.
3	**Design**				
3.1	Sitemap and UI design of storyboards	15/11/2022	26/11/2022	11	The user interface designs
3.2	Presentation on storyboards	21/11/2022	25/11/2022	4	Presentation of the proposed UI designs, including a sitemap which breaks down the different components/pages of the system (for navigation)
3.3	Review design and improve UI from review	25/11/2022	9/12/2022	14	Review of the UI designs to ensure it stays within scope and requirements and also follows accessibility.
4	**Implementation / Testing**				
4.1	Sharepoint playground - first interactions and	9/12/2022	27/12/2022	18	Introduction to the

	backend set-up				recommended tool (Sharepoint): first interactions (for learning purposes) and thus beginning implementation.
4.2	Login system / account management	13/12/2022	27/12/2022	14	Foundation of the implementation; a login system is required for account management and as a way to navigate the rest of the system.
4.3	Dashboard	27/12/2022	14/1/2023	18	Implementation of the dashboard to introduce user into the system
4.4	Spaces and groups	10/1/2023	24/1/2023	14	Implementation of the collaborative spaces of groups to share resources
4.5	Documents component	24/1/2023	7/2/2023	14	Implementation of the functional requirements uploading, editing and deleting of documents
4.6	Polls component	28/1/2023	11/2/2023	14	Implementations of the functional requirements of posting, viewing and deleting polls.
4.7	Calendar component	11/2/2023	4/3/2023	21	Implementation of the calendar and the handling of events and availability from representatives.
5	**Review**				
5.1	Review project	4/3/2023	11/3/2023	7	Review the scope of tasks and requirements succeeded or failed.
5.2	Make final updates	25/3/2023	29/3/2023	4	Any polished changes as a result from final testing for errors and overall user experience.
5.3	Official launch	25/3/2023	29/3/2023	4	Deployment of the system

Comparison

Below is a comprehensive overview of different approaches that could be taken to develop the system as specified by the client. This is approached by defining the notion of non-functional requirements into two categories: quality attributes, which determine how well the system should perform; and resource attributes, which constrain or limit the possible solutions to a business problem.

Confluence

Overview: A team workspace, developed by Atlassian, that allows for teams to create dynamic pages which allow for creating, capturing, and collaborating on project ideas. Confluence spaces typically allow for teams to organise, structure, and share work, so that team members have visibility of institutional knowledge and access to information that allow users to do their best work.

Confluence allows for users to build applications on top of it's platform using it's forge platform, using either UI kits (prebuilt kits by Atlassian to speed-up development), Custom UI (using HTML, CSS, JS, and images), or even add components from the Atlassian marketplace, created by others users (which can be free or paid for depending on the app).

Benefits:

- **Suitability** - due to a rich ecosystem of apps, and ability to create apps easily through pre-built components, it can be very quick to get a working application up and running quickly. If the pre-built components aren't suitable, then they can be easily modified, or there is even the possibility for custom apps to be built.
- **Interoperability** - the atlassian marketplace has many different integrations, with calendars, apps that can visualise tagged content, widgets, etc.
- **Security** - Atlassian uses modern security practices and are backed by industry-accepted certifications, such as ISO 27001, SOC, CSA, PCI DSS.

Drawbacks:

- **Cost** - As Atlassian[2] states for cloud hosting, it would cost "$11/month/user." For a self-hosted platform, for up to 500 years, there would be a $27000/year fee. There is an academic discount for this platform, which is advertised at being 50% off the advertised price ($5.50/month/user and $13,5000/year respectively). Additionally, any external integrations required may need to be priced on top.

Slack

Overview: a messaging app for business that connects people to the information that they need. Different information is split up into channels, and in these channels you can aggregate data, and then users can access this data when they have the time without having to coordinate schedules.

Slack also has an API that can help to speed up routine processes and eliminate mundane tasks. This is done by either integrating external/internal apps into slack, or through the use of "bots", which are applications that encourage the user to talk to them to relay information.

[2] Atlassian *Confluence - Pricing | Atlassian*. [Online] Atlassian. Available from : https://www.atlassian.com/software/confluence/pricing.

Benefits:

- **Interoperability** - despite being a messaging application first and foremost, slack could allow for data to be aggregated into channels, integrations could be found from other services, and applications could be built on top to streamline the process needed. Additionally, Slack already has built-in fine-grain access to different user types making access control easy, and it would also be simple to set up new spaces where required.
- **Security** - Slack has enterprise-grade data protection, backed up by industry-accepted certifications such as ISO 27001, SOC2, SOC3, CSA, APEC, PRP. Passwordless authentication can also be used through the use of OAuth logins.
- **Usability** - Due to being an pre-existing platform, users may already have used slack internally with their company or other endeavours and therefore onboarding would be straightforward. For those without experience, the UX is intuitive, but there is also pre-existing extensive documentation online to help users where stuck, along with tips and tricks built into the app.

Drawbacks:

- **Cost** - Slack advertises a 85% discount off the £9.75/month/user cost, making it £1.46/month/user.
- **Suitability** - although an application could be built on top of Slack, due to it being a messaging platform, the layout and the overall platform is designed with this in mind. This also means that certain core features, such as messaging, could not be turned off if desired. There is also no option to self-host a slack platform if desired.

Microsoft Sharepoint

Overview: A Microsoft platform which allows users to create collaborative websites. These websites can be used for sharing resources such as files and blogs, the assignment of tasks and also building new workflows

Benefits:

- **Suitability -** the customisation is one of the strong points of the application, as the tools are extensive and can be easily customised to fit the client's needs i.e. as Framework IT[3] states SharePoint can be used to *"add branding elements, easily create automated workflows using Power Automate [...] and customise a sharePoint form using Power Apps."*.
- **Interoperability -** able to integrate office and web-based documents. This allows for different technologies to be considered when building custom-based solutions.
- **Security -** As being part of Microsoft, the protection of data is advanced due to its encryption. Microsoft also provides two-factor authentication as an extra step of security which is heavily recommended for organisations in protecting their data.

Drawbacks:

- **Cost -** customisation of the platform depends on the budget of the organisation. Because the application is under Microsoft, Ungoti[4] states that it is likely that there are extra costs required for *"further customisation, implementation, training and administration"*.
- **Learning curve for integration -** training is mandatory to deal with the complexity of this application. As stated by Framework IT, *"Microsoft recommends that companies provide internal training so any intricate deployment of SharePoint doesn't have an adverse effect*

[3] Frameworkit.com. 2022. *Pros and Cons of SharePoint 2020: Is SharePoint Worth it?*. [online] Available at: <https://www.frameworkit.com/blog/pros-and-cons-of-sharepoint> [Accessed 17 October 2022].

[4] Ungoti. 2022. *You must Know these Pros and Cons of SharePoint Intranet*. [online] Available at: <https://ungoti.com/blog/sharepoint-intranet-pros-and-cons/> [Accessed 17 October 2022].

and cripple the productivity of your team". It could even take months for certain users to fully integrate and develop the site to their specific needs.

- **Maintainability -** despite the platform being extensive with its features, it is difficult to maintain. Inadequate user training for the complexity of this platform can heavily contribute to the overall cost of projects.

Custom development

Overview: The alternative option is undergoing the process of the development of software which aims at a specific set of requirements and users.

Benefits:

- **Suitability -** there is full freedom and control over the software thus reaching specific needs and requirements can be simpler.
- **Accuracy -** development of a project that is from scratch and heavily reliant on the expertise of users alone has a risk of failure. These risks can however be possibly reduced via the implementation and support of frameworks.
- **Usability -** in comparison to the "off-the-shelf" options, the implementation of features from the user perspective is usually less effective compared to applications that are custom-developed.

Drawbacks:

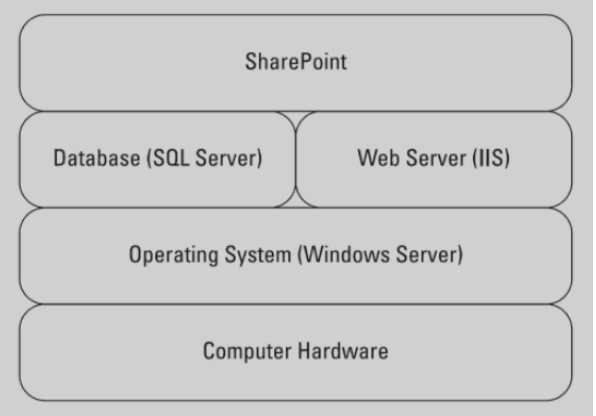
- **Cost -** the cost for custom development can be heavy. In terms of time, the project is running on a limited time schedule, thus starting from scratch can be a risky move as developing such a software can take months. In terms of cost, it is usually more expensive in comparison to software that is prepackaged; this cost is further increased with the consideration of constant maintenance depending on the complexity and quality of the code (highly dependent on the expertise of the coders in the team).

Final recommendation

With the evaluation of the comparison of the four different tools, It is concluded that the final recommendation on the tool to use is **Microsoft SharePoint**. This is because, while the cost in terms of time and money is essential for the platform's selection, the benefits of the platform as mentioned outweigh the cost. The suitability, interoperability and security of the platform gives confirmation that this tool is right to use and worth exploring. It is mentioned that Sharepoint has the drawback of the licensing cost, which would usually be the driving factor to not deciding to use this tool. However, the licensing cost has been solved due to the current circumstances. The circumstances are the licensing costs are already covered as the department has already bought and owned it. With this final decision, it is most appropriate to set up a development environment within this platform to implement the project in.

Background theory/technology: SharePoint

SharePoint is a web-based collaboration platform that has been developed by Microsoft. It is designed to support collaboration, document management and sharing of information within organisations. As visualised by *Rosemaire Withee*[5] (the diagram on the right), SharePoint is built on top of several existing Microsoft technologies, which can include the Microsoft SQL Server, Windows Server and Information Information Services (IIS). The focused technology for this project would be the Microsoft SharePoint generator which can create a web part template with the integration of React.



The platform can provide a wide variety of tools for managing documents, libraries, and lists, as well as advanced features including workflow automation, search capabilities and business intelligence. The deployment of SharePoint can be done on-premises or in the cloud, and it can also be customised to meet the specific needs of different organisations. The use of SharePoint within this project includes the feature to deploy custom web parts that will be implemented by the developers of the team.

Furthermore giving a theoretical perspective on the platform, SharePoint is based on several key principles of collaboration and management. These principles can include:

- **Content management:** First Rank Publishing states[6] that *"there are access controls that allow the organisation to centrally manage the diverse content that may be stored on the SharePoint platform. It contains rich metadata, workflow and access controls that will help you manage your content in your desired way"*. In context to the project, this would be within the group of admins and other users of the SharePoint website under the Royal Holloway tenant.
- **Collaboration:** With SharePoint being a collaboration platform, it means that teams are enabled to work together more effectively. It provides tools for creating and managing team sites, sharing documents and other content and communicating with other team members.
- **Workflow automation:** a powerful workflow engine that allows organisations to automate their business processes. This can include workflows for task management, document approval and other types of processes. This can be achieved by the integration of the tool, Power Automate, with SharePoint. Aaron Guilmette[7] further explains Power Automate as *"primarily a web-based tool designed to interface with a growing library of software from both Microsoft and other venders [...]. Due to its no-code/low-code design, Power Automate can be approached by individuals with any skill level".*

[5] Withee, R., Withee, K. (2021). SharePoint For Dummies. United Kingdom: Wiley.
[6] Sharepoint 2017: An Easy Guide to the Best Features. (2017). (n.p.): First Rank Publishing.
[7] Guilmette, A. (2022). Workflow Automation with Microsoft Power Automate: Use Business Process Automation to Achieve Digital Transformation with Minimal Code. United Kingdom: Packt Publishing.

The architecture of SharePoint

The overview of the architecture of SharePoint is that it is designed to support various components and features of the platform in a way that ensures flexibility and scalability for various deployment scenarios. The key components of SharePoint architecture (as with the explanations of some by Shannon Bray[8]) would be:

- **Web Front End (WFE):** The first point of contact for users accessing SharePoint with the role of receiving requests from users and communicating with various other components to fulfil those requests. It is also responsible for rendering the user interface and delivering content to the user's browser
- **Application Server:** hosts the Sharepoint services and provides backend processing for SharePoint features such as search, workflows and business intelligence.
- **Database Server:** a server that stores the content and configuration information for the SharePoint environment. It hosts the content databases, configuration databases and service application databases.
- **SharePoint Farm:** a group of SharePoint servers which work together to provide the functionality for the platform. It is noted that *"when building the farm, the first component that gets created is the SharePoint Configuration and Admin Content Databases"*. The benefit of the farm's architecture is that it provides scalability and redundancy to allow the environment to handle increased traffic and fault tolerance.
- **Service Applications:** a functionality in Sharepoint which deals with aspects such as the user profiles, search and the business connectivity. Each separate service application can be deployed on a separate server to ensure scalability and performance.
- **Web applications:** a set of web pages that are hosted on a SharePoint site. It is stated that *"in order for a SharePoint farm to exist, it must contain at least one SharePoint web application."*
- **Site collections:** a container of sites that share common settings, permission and other content. It is stated that *"every functional SharePoint web application is required to have at least one site collection"*.

[8] Bray, S., Wood, M., Curran, P. (2013). Microsoft SharePoint 2013 Designing and Architecting Solutions. United States: Pearson Education.

Technologies

Frontend technologies -

- Library: **React**
    - It is already an option to be integrated as a framework with the web part solution template (created by Yeoman generator)
    - Vipul Jain[9] mentions that the advantages of ReactJS is that the *"application performance increases; [it] can be used on the client as well as server-side; [and] code readability gets improved."*
- Configuration:
    - **Gulp**
        - Used as the build process task runner for Sharepoint client-side development
        - Used for building, testing and deployment
    - **Yeoman**
        - Useful in creating client-side solution project templates for creating new web parts including a React framework
- Integration: **SharePoint Framework (SPFx)**
    - Ease of integration between Sharepoint Online
    - SP HTTP Client used to handle API requests (GET, POST)
    - Used for the implementation of SPFx client-side solutions (the web parts) deployed and uploaded with the approval of tenant administrators of websites.
    - Nanddeep Nachan[10] mentions that the framework's key features which can be suggested as benefits include: *"all controls are rendered in the normal DOM enabling faster rendering on the browser; controls are responsive; [...] it is framework agnostic (any framework like React, Angular, Knockout, and so on can be used with SPFx; [and] it supports open source development tools (npm, TypeScript, Yeoman, webpack and Gulp"*

- CSS framework: **Tailwind**
    - It is a utility-first framework; thus does not require you to switch between different files; saves the cost of time and effort and improves productivity
    - The framework is easily customisable
    - Provides pre-built classes for responsive layouts
- Testing framework - **Jest**
    - Command line tool available to control tests; includes interactive mode that runs automatic and fast unit tests
    - The reason for choosing this testing framework for unit tests can be given by the features offered. Azat Mardan[11] lists down the features Jest offer, which include: *"powerful mocking of JavaScript/Node modules [which] makes it easier to isolate*

[9] Jain, V. (2020). Getting Started with SharePoint Framework (SPFx): Design and Build Engaging Intelligent Applications Using SharePoint Framework. India: BPB PUBN.
[10] Nachan, N. (2019). Mastering Sharepoint Framework: Master the SharePoint Framework Development with Easy-to-Follow Examples. India: BPB PUBN.
[11] Mardan, A. (2017). React Quickly: Painless Web Apps with React, JSX, Redux, and GraphQL. United States: Manning.

code in order to unit test it; less setup is required to get started than with other test runners, such as Mocha, where you need to import Chai or standalone Expect. Jest also finds tests in the __tests__ folder; tests can be sandboxed and executed in parallel to run them more quickly; you can perform static analysis with the support of Facebook's Flow [...], which is a static type checker for JS; and Jest provides modularity, configurability and adaptability (via the support of Jasmine assertions)."

Backend technologies

- (acts as a Database): **SharePoint Lists**
    - Site contents that exist within the SharePoint site which are lists that handle data
    - Use of CRUD (create, read, update, delete) operations to manage the lists within the web context of the Sharepoint server.
- Server: **SharePoint Server 2019** - the server that is used along with the SharePoint Framework (SPFx)

Professional Issues

Data Privacy

The professional issues related to data privacy that the team needed to consider when developing web parts in SharePoint included various aspects:

- **User data protection** - The issue of data privacy can fall under the necessary actions and implementation to protect any data, collected or processed by the web parts, against unauthorised access, modification, or the misuse of disclosure. This can include the implementation of securing encryption, authentication mechanisms and access controls to protect sensitive data. Jason Medero[12] explains an example of implementing SSL encryption for SharePoint where the *"proper use of SSL on your SharePoint sites offers the three benefits: confidentiality, integrity and authentication [...]*
    - *Confidentiality - 'employing SSL on a site ensures that if the network traffic is intercepted in transit, it is not legible or open to decryption by an unauthorised party'.*
    - *Integrity - 'when using SSL, you and the server are both assured that the network traffic received was not modified in transit'*
    - *Authentication - 'means you are assured of the identity of the remote server [...]. This is important because an attacker could redirect your users to another server and intercept the user's authentication credentials, among other possibilities'"*
- **Data storage -** Sharepoint web parts offer features which can store data in various locations such as Sharepoint lists, databases, or external systems. Developers of the team must ensure that the data is stored securely, with the appropriate access controls in place to prevent unauthorised access. The site contents (including the Sharepoint lists) of the Sharepoint pages must be away from anyone unauthorised; the sharepoint page is private and is only exclusive access to those part of the group. Any further permissions of accessing site owners must be implemented by the administrators of the group.
- **Data collection and transparency -** The principle of transparency is emphasised by the Data Protection Act. The particular pieces of information that the developers must be transparent with the users of the web parts are about what data is being collected and processed, how it will be used and who will have access to it.
- **Compliance with regulations -** complying with the relevant data privacy regulations is essential for developers of the team. Reza Alirezaei[13] emphasises this further when stating that *"from a SharePoint perspective, if you are architecting any solution that will maintain PII information, you should seek to ensure that your solution complies with the policies set out by the legal department of the organisation"*. This can include regulation such as the GDPR (General Data Protection Regulation) - obtaining consent from users before collecting or

[12] Medero, J., Fox, B., Prussak, P., Mehta, N., Poelmans, J., Buechler, C. M., Pragash, C., Lotter, M., Regan, C. J., Pyles, J., Gordon, M. (2008). SharePoint 2007: The Definitive Guide. United States: O'Reilly Media.
[13] Alirezaei, R., Baer, B., Wilson, B., Kearn, M. (2012). SharePoint 2010 Enterprise Architect's Guidebook. Germany: Wiley.

processing their personal data and ensuring the data is only used for the purposes for which it was collected.

By considering these professional issues regarding data privacy, developers can ensure web parts are developed in SharePoint with data privacy in mind, thus protecting user data and reducing the risk of any privacy related issues mentioned. The use of data in the project merely included the built-in SharePoint context to collect the users' information (i.e. their name, email, department etc.). The data was used strictly within the tenant it should be in and used service key to access the context of the SharePoint. The use of sensitive information such as passwords was not necessary for this web parts because it was not relevant and the user accounts and authentication is already handled by Microsoft.

Copyright

With the development of web parts which are deployed and uploaded within a Microsoft application, it is essential to abide by the copyright regulations to avoid legislation problems. The issues regarding the copyright that need to be considered include:

- **Ownership of the web parts** - the developers of the team must ensure that they claim ownership of the web parts they are developing. Otherwise, they must ensure they have necessary permissions from their employer or for clients to develop such web parts. The procedure for this usually includes ensuring that any non-disclosure agreements or contractual obligations are not violated.
- **Open source licences** - the development of Sharepoint web parts included familiarity with examples of web parts other developers have implemented. Microsoft[14] offers many example web parts to help Sharepoint developers with their own websites. The developers in the team that used open source code from this repository must comply with the terms of the open source licence stated in the documentation of the repositories. This can include releasing the code under the same licence or even making it freely available to others. The credit of work in the documentation will also be passed onto the developers of the original open source code.
- **The use of copyrighted material and copyright infringement** - the development of web parts can often include forms of multimedia e.g. images, videos or text. This multimedia could potentially contain copyrighted material, thus developers in the team must ensure that they have undergone the necessary permissions to use such material. The permission to material can be claimed by obtaining the necessary licences or taking the option of using material that is in the public domain. Microsoft[15] states in one of their answers in their copyright FAQ that it is *"generally required by law to disable access to copyrighted content (including videos, music, photographs, or other content you upload onto a Microsoft website) if the copyright holder claims that the use of the copyrighted work is infringing. [The user can] let us know if [they] believe that a copyright holder wrongly requested that we disable access to content*

[14] GitHub. (n.d.). sp-dev-fx-web parts/samples at main · pnp/sp-dev-fx-web parts. [online] Available at: https://github.com/pnp/sp-dev-fx-web parts/tree/main/samples [Accessed 30 Mar. 2023].
[15] Microsoft Legal. (n.d.). Copyrights. [online] Available at: https://www.microsoft.com/en-us/legal/intellectualproperty/copyright/ [Accessed 30 Mar. 2023].

[they] uploaded (e.g., [they] believe [they] have the rights to use that content or because [their] use is a fair use)". The team developers have the responsibility of ensuring that there is no infringement on the copyright of others. The procedure for this includes ensuring that code or other material is not entirely copied without permission or the use of copyrighted material without the necessary licences.

To summarise, it is essential for developers of the team to be aware of the implications of copyright in the Sharepoint web parts, thus making the necessary measures to ensure that there is no infringement on the rights of others. These measures can include obtaining the necessary licences and permissions, ensuring that they claim their own licence and/or permission to develop the web part and also complying with the open source licences if applicable.

Discussion about the web part software

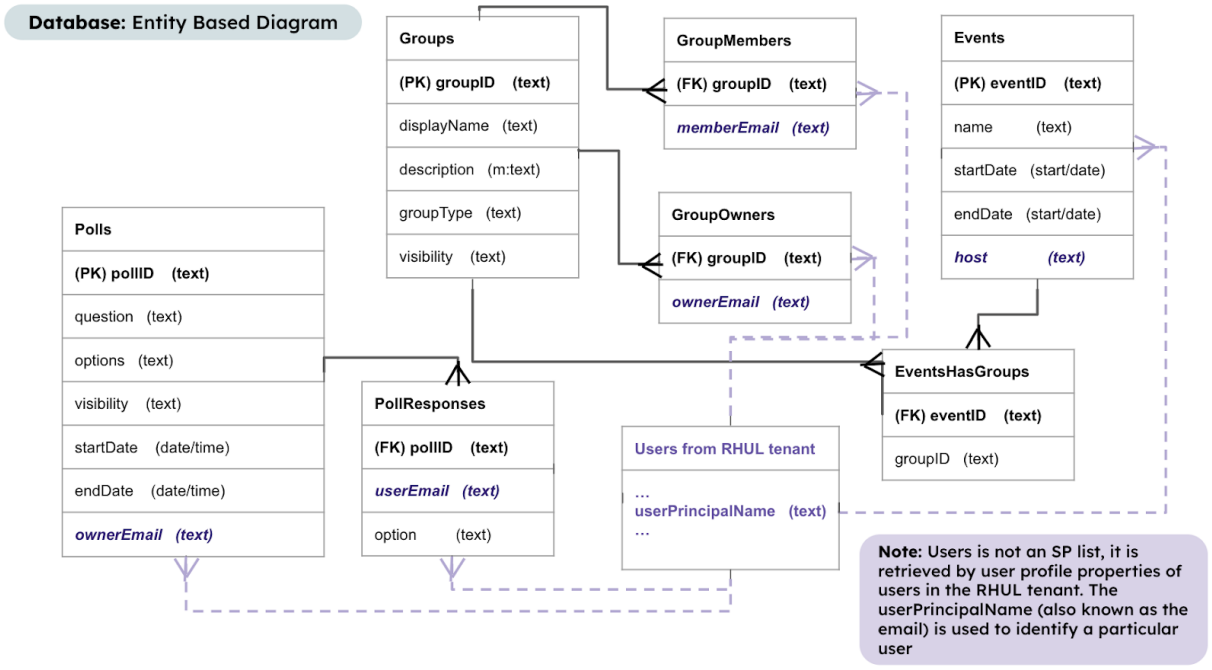
The similarity of the SharePoint web parts to alternatives is mainly the ability to create custom components to integrate into websites. However, the use of the React JavaScript library means the development is easier. The licensing arrangements of the web parts would be open-source whereas for alternatives, the licensing depends on the particular solution, but generally it is under a licence which requires permissions (e.g. Apache or MIT licence) or even a commercial licence.

The performance of the software is based on the configuration and implementation of the web parts. However, regarding the React framework which is highly optimised, the web parts are typically faster (e.g. in rendering performance) compared to other technologies. The distribution for SharePoint web parts would usually be deployed and uploaded into the SharePoint store, however this requires administration access into the SharePoint RHUL admin centre (that the developers of the team do not have). Thus for availability to develop with, web parts would be distributed via GitHub for developers with testing web parts under the workbench of the SharePoint 2019 Server.

The maintenance of the web parts depends on the implementation of each web part, but it is generally essential for developers to have ongoing maintenance to ensure the web parts are functional. This is also because there are dependencies that may require updating or may become deprecated over time.

Implementation

Database



The web part solutions used Sharepoint Lists as a way to structure collections of information. It is visually presented to look like a database table or Excel spreadsheet. Steve Wright[16] further explains that *"often SharePoint lists are chosen as the main storage source [...] [It is] very similar to a database table [...]. [The] small differences include the following: a key is automatically created and selected on every list (usually the column named Title); lists exist within a site and are not generally available to all sites; [and] lists can be throttled to manage large amounts of data and improve response times."* The column names were created to imitate the behaviour of relational databases (i.e. a column used as the identifier for the list as well as columns which act as 'foreign keys', even though foreign keys do not exist in Sharepoint Lists). It is essential for the system to keep track of data within the system that can be handled via CRUD operations (create, read, update, delete).

An important note to mention is that handling a list of **users** in the system was not necessary because user authentication is already handled by SharePoint, thus there are already functionalities within the SPFx framework and the SP HTTP client to retrieve details about the existing users within the Royal Holloway tenant. The references to a user in SP lists were their user principal name/email i.e. 'zhacXXX@live.rhul.ac.uk'.

Groups - the groups of the system

[16] Wright, S., Bishop, D., Malik, S., McDermott, M., Rais, R. b., Milner, D., Krause, J., LLC, W., Eddinger, M., Sethunarayanan, K., Orange, M., Naveed, T., Bakmand-Mikalski, D., Musters, E., Farnhill, B., Ralston, B., Richard, E., Hild, E., Loriot, C. R., Ortiz, D. (2011). Expert SharePoint 2010 Practices. Netherlands: Apress.

- **groupID** (p.k) - the identifier for the group
- displayName - the name of the group
- description - the description of the group
- groupType - the type of group
- visibility - the visibility of the group (is the group public or private?)

GroupMembers - the group members of a particular group
- **groupID** (f.k references Groups.groupID) - the group ID
- memberEmail - the email of the group member

GroupOwners - the group owners of a particular group
- **groupID** (f.k references Groups.groupID) - the group ID
- ownerEmail - the email of the group owner

Polls - the polls of the system
- **pollID (p.k) -** the identifier of the poll
- question - the question of the poll
- options - the list of options of the poll
- visibility - the visibility of the group (is the poll active or disabled?)
- startDate - the start date of the poll
- endDate - the end date of the poll
- ownerEmail - the owner's (of the poll) email

PollResponses - the responses from a particular poll; used for poll analytics
- pollID (f.k references Polls.PollID) - the poll ID
- userEmail - the user of the poll response
- option - the option chosen from the poll response

Events - the events of the system
- **eventID (p.k)** - the identifier of the event
- name - the name of the event
- startDate - the start date of the event
- endDate - the end date of the event
- host - the host of the event
- [...] (there are many other attributes but the ones above are the most relevant in terms of entity relationships)

EventHasGroups - the events available to a particular group
- eventID (f.k references Events.eventID) -
- groupID (f.k references Groups.groupID) -

Architectural and design patterns

Singleton Pattern: Addy Osmani[17] describes the singleton pattern as one that *"restricts instantiation of a class to a single object. Classically, the Singleton pattern can be implemented by creating a class with a method that creates a new instance of the class if one doesn't exist. In the event of an instance already existing, it simply returns a reference to that object"*. The singletons, in this case, are the 'services' for particular web parts e.g. the *'UserEventService'* class which is exported to be used for functionalities required by the web part for events. The reason for the use of this pattern is because only one instance of the services are needed and it also brings the benefit of saving a lot of memory space and avoids unnecessary instances of the same object which all perform the functionalities.

Example of implementation of this design pattern:

```
export class UserEventService {
   private static instance: UserEventService
   [...]

   static getInstance(): UserEventService {
       if (!UserEventService.instance) {
           UserEventService.instance = new UserEventService();
       }
       return UserEventService.instance
   }
   [...]
}
```

- The pattern forces only one instance of the class because when referencing the class (i.e. getInstance). There is either the case of:
    - No class exists: create a new instance and assign it
    - A class already exists: just refer to the instance that has already been created

Container-Presentational Pattern: A React pattern which intends to separate the logic of a component from its presentational component. This is achieved by the concept of a container component and a presentation component.

- The container component: provides the overall structure and functionality for the web part. It typically contains child components.
- The presentation component: Carl Rippon[18] describes this part of the pattern as the React component that is *"responsible for how things look. Presentational components receive data via their props, and also have property event handlers so that their containers can manage user interactions."*
- The reason for the use of this pattern is because *Patterns*[19] explains that the advantages of the pattern are how it *"makes it easy to enforce the separation of concerns. Presentational*

[17] Osmani, A. (2012). Learning JavaScript Design Patterns. United States: O'Reilly Media.
[18] Rippon, C. (2021). ASP.NET Core 5 and React: Full-stack Web Development Using .NET 5, React 17, and TypeScript 4, 2nd Edition. United Kingdom: Packt Publishing.
[19] *Container/Presentational Pattern*. [Online] Container/Presentational Pattern. Available from : https://www.patterns.dev/posts/presentational-container-pattern.

components are easily made reusable, as they simply display data without altering this data. We can reuse the presentational components throughout our application for different purposes. [Also] Testing presentational components is easy, as they are usually pure functions. We know what the components will render based on which data we pass, without having to mock a data store".

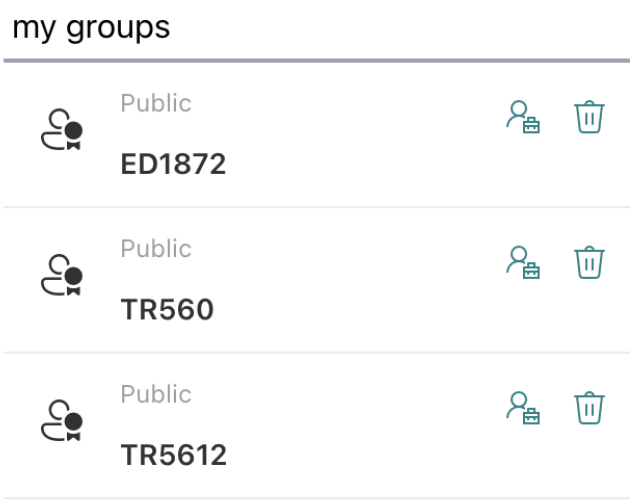
Example of implementation of this React pattern:

```
export default class GroupList extends React.Component<IGroupListProps,
IGroupListState> {
 private _originalItems: IGroup[];
[...]

 constructor(props: IGroupListProps) {
   super(props);

   props.items.map(group => {
     let myUserRole: string = "";

     if (props.ownerGroups.indexOf(group.id) > -1) {
       myUserRole = "Owner";
     }
     else if (props.memberGroups.indexOf(group.id) > -1) {
       myUserRole = "Member";
     }

     group.userRole = myUserRole;
   });

   this._originalItems = props.items;

   this.state = {
     [...]
     groups: this._originalItems
   };

   this._onRenderUserGroupCell = this._onRenderUserGroupCell.bind(this);
   [...]
 }

[...]

public render(): React.ReactElement<IGroupListProps> {
   const { groups = [] } = this.state;
```

```
    [...]

    return (
            [...]
                <h2 className="[...]"> my groups </h2>
                <List items={groups.filter(group => group.userRole === "Member"
|| group.userRole === "Owner")} onRenderCell={this._onRenderUserGroupCell} />
            [...]
    )}
}
```

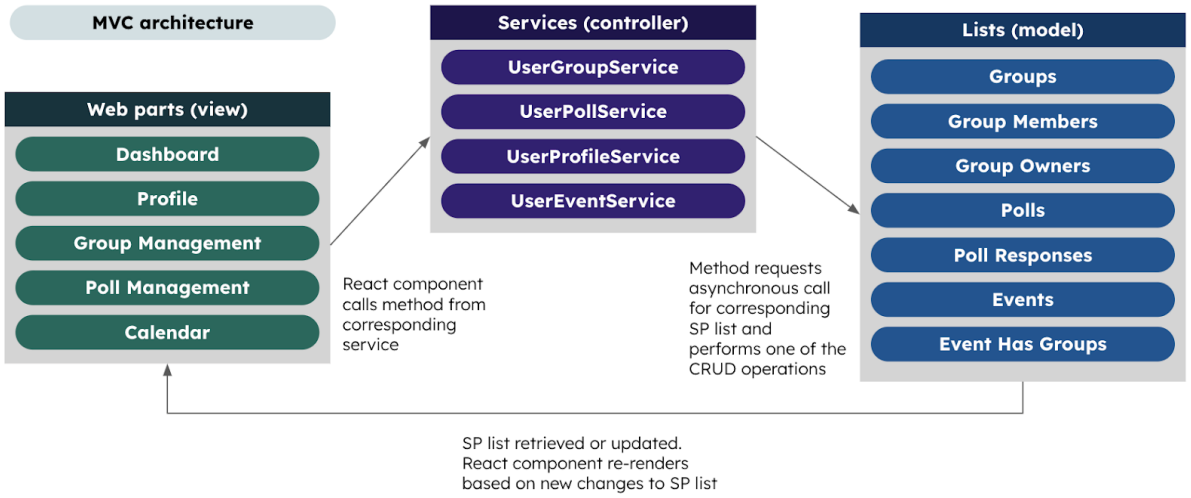
- **The container component:** The overall structure is the React component 'GroupList'. This is a component that handles the display of the user's groups. The constructor uses its props to assign the retrieved data of groups as the 'original items' and sets it as the initial state. It also assigns the appropriate user role for each group from the props (i.e. is the user a 'member' of the group, an 'owner' of the group or not in the group at all)
- **The presentation component:** The List component renders a list of items based on the given props. The props in this case are all the groups that the user is an owner or a member of.

The view output:



Model-View-Controller (MVC) architecture: Dipen Patel[20] explains that the key to implementing the MVC architecture is to "separate functionality, logic [model], and the interface [view] of an application to promote organised programming". It creates a separate component (controller) to handle the communication between the two. It organises the program into the functions, logic and the interface. The main benefit of this architectural approach is that it helps to keep code maintainable, organised and scalable.

Visual process of the implementation of this architectural design:



- **The model** represents the data logic of the application. In the context of this project, this will be data from the lists.
- **The view** is responsible for the rendering of the user interface based on the data from the model. The React framework uses a set of components to create the view and pass data and actions from the model and controller to the components as props.
- **The controller** is the link between the model and the view. It receives user input from the user and updates the model accordingly. It also sends updates to the view based on changes to the model

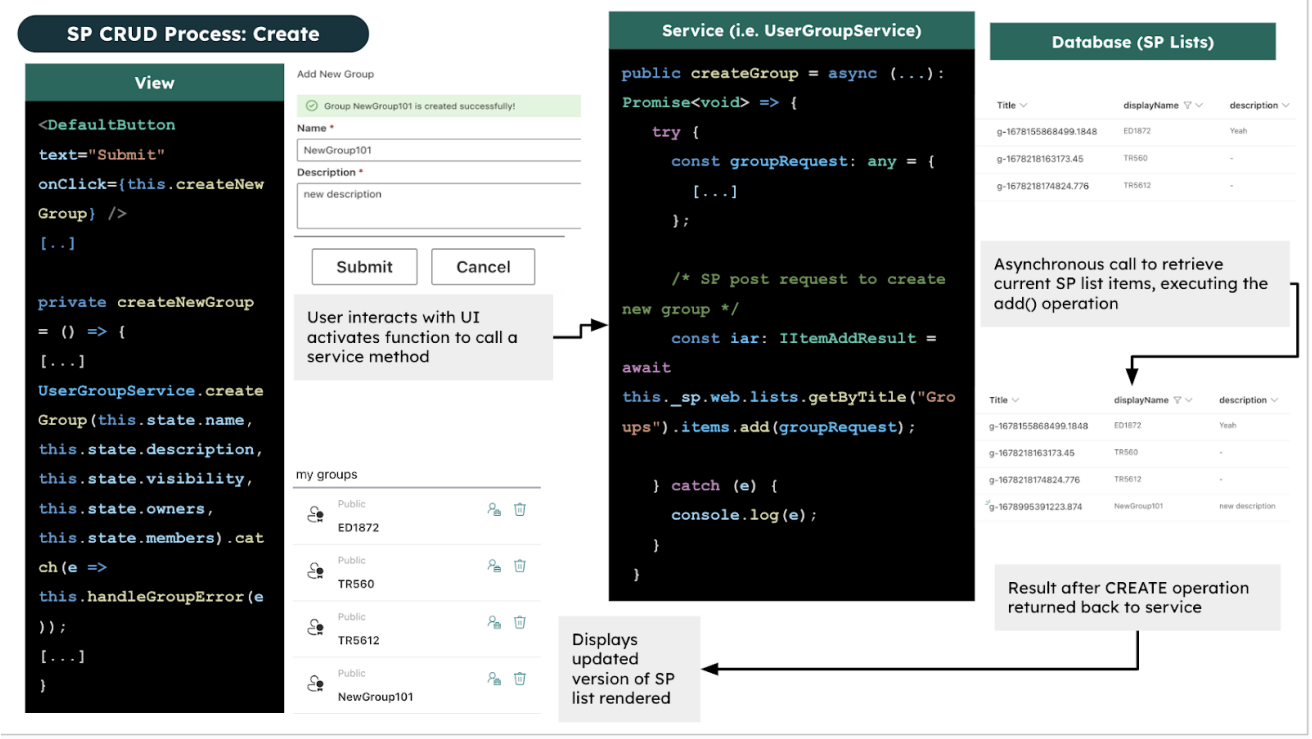
The process of steps in the project's implementation of MVC include:

- The user interacts with the view via browser; initiates a request via interaction e.g. clicking a button that performs an action.
- The action involves calling a method from one of the services.
- Method in service performs CRUD operation (create, read, update, delete) on the corresponding SharePoint list (the model).
- The list is updated or retrieved and is sent back to the service
- Service sends processed data back to the browser and React component
- User views new view with processed data via browser

[20] Patel, D. (2019) *An introduction to MVC architecture: A web developer's point of view - dzone*, *dzone.com*. DZone. Available at: https://dzone.com/articles/introduction-to-mvc-architecture-web-developer-poi (Accessed: March 29, 2023).

MVC Process of CRUD operations:

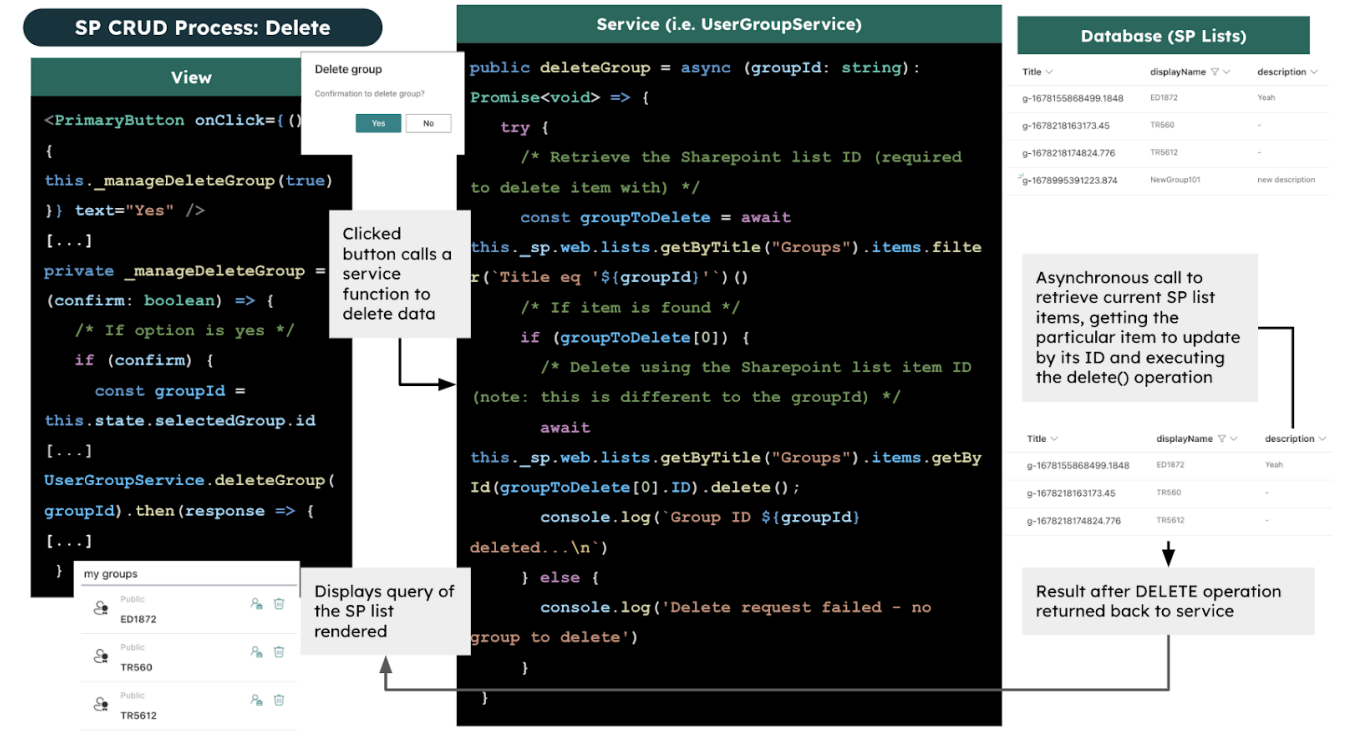
Create operation - create a new item in list



Read operation - read items from a list

Update operation - update an item in the list

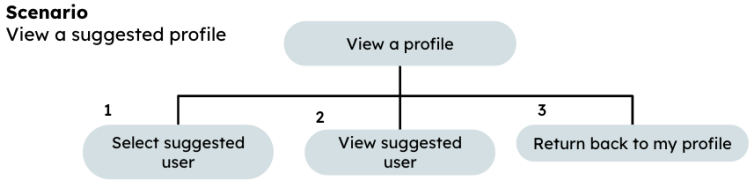
Delete operation - delete an item from the list



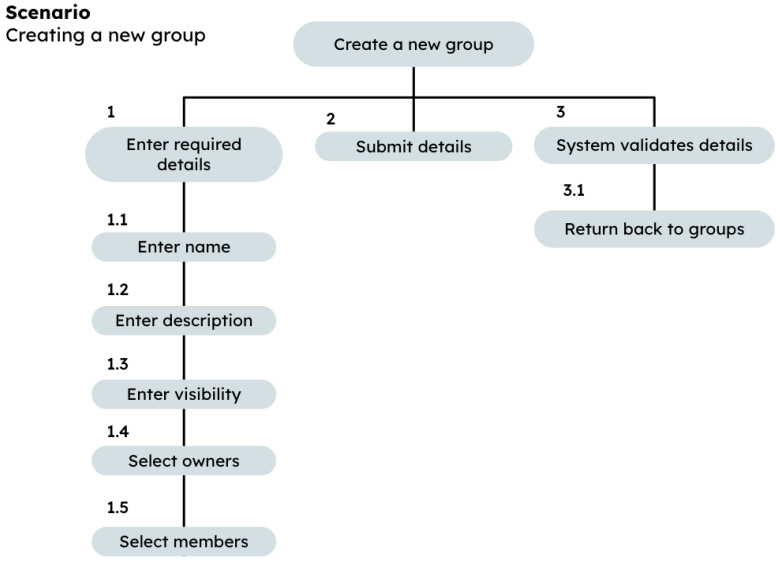
Task analysis

The breakdown of the tasks that are performed to achieve a particular goal in the system:

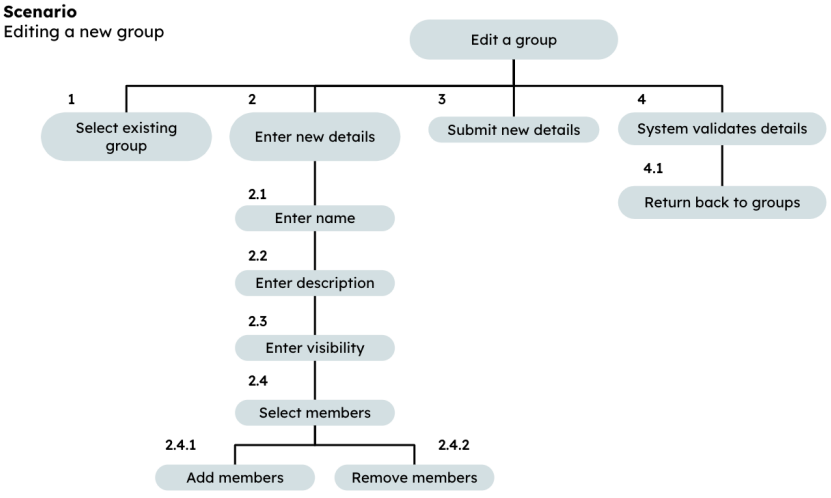
- **Profile -**
  - View a suggested profile



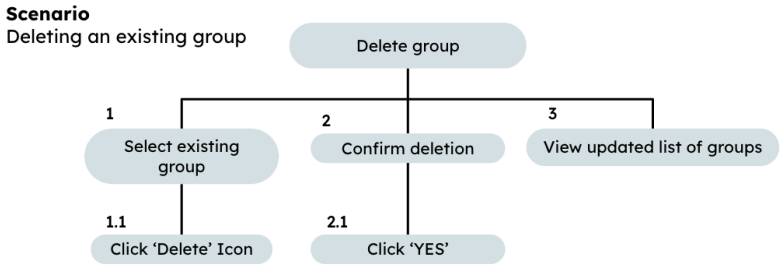
- **Group Management -**
  - Creating a group



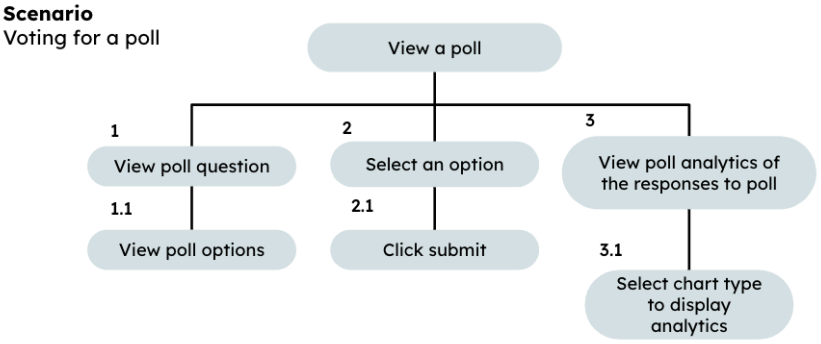
  - Editing a group



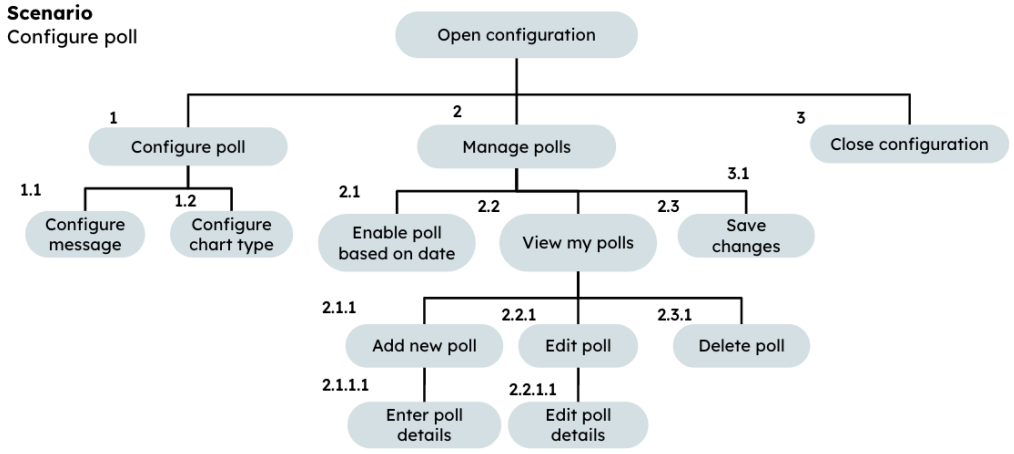
- Deleting a group



- **Poll Management -**
  - Voting for a poll

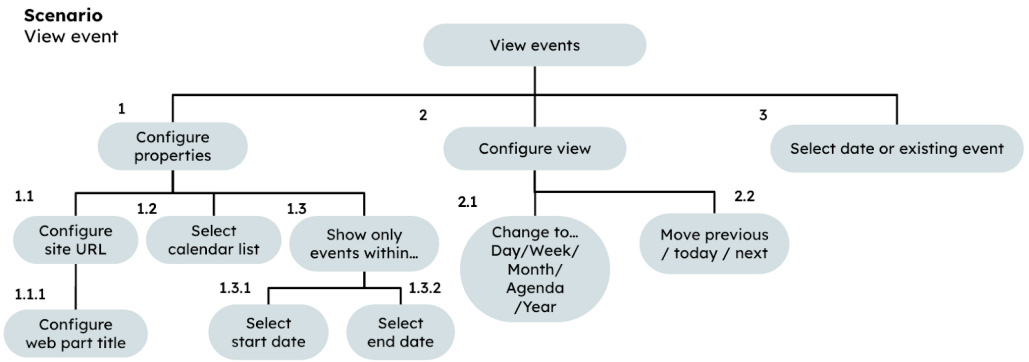


  - Configure polls

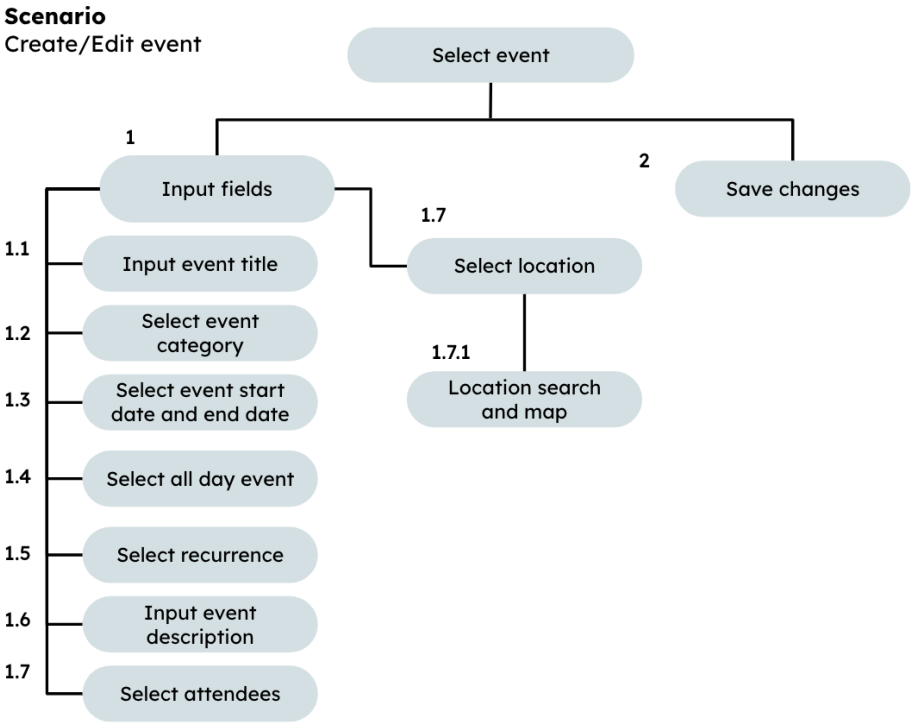


-

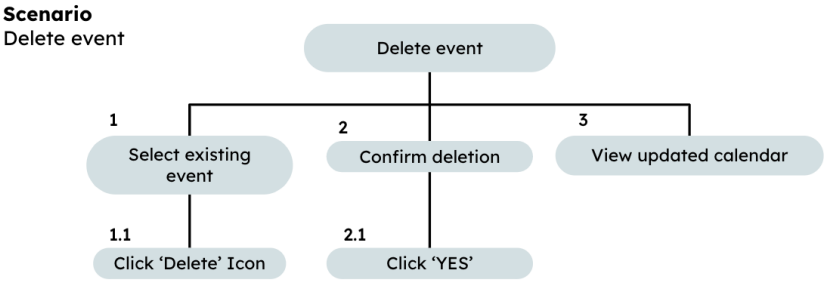
- **Events**
  - View events



  - Create/Edit event



  - Delete event



Web part solutions

With our investigation into the features that Sharepoint has to offer, it was concluded that there were already some planned features that were already built-in by Sharepoint. These following features included:

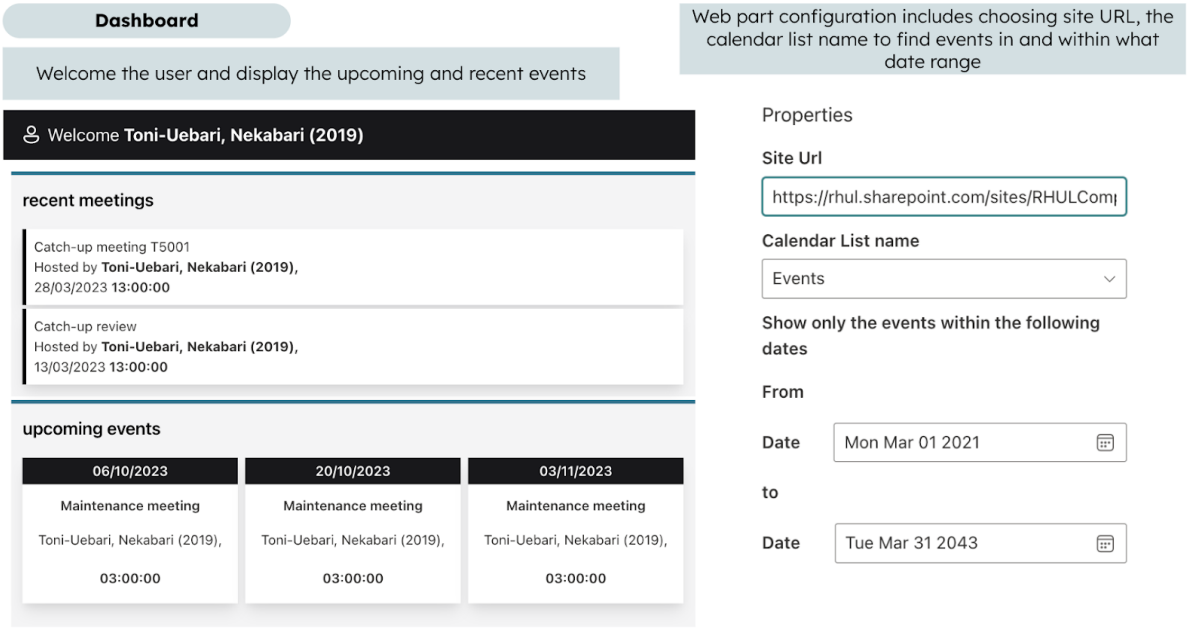
- User authentication - users already exist within the Royal Holloway University tenant and are already required to access the system via two-factor authentication handled by Microsoft.
- Documents - SharePoint already offers a web part to handle documents, including a web part that shows the existing documents uploaded by users within the website.
- Feed - Sharepoint already offers a web part that is similar to the functionality of a feed. The news web part allows users to post media where other users can comment on it.

Thus, this leaves the team with a refactored list of web parts to implement, deploy and integrate with the website and extend its functionalities.

- Dashboard
- Profile
- Group Management
- Poll management
- Calendar

Customer interaction (with source code and annotations)

Dashboard - a web part which offers the overview of the system e.g. upcoming events, recent events, recent polls.



- Render recent and upcoming events

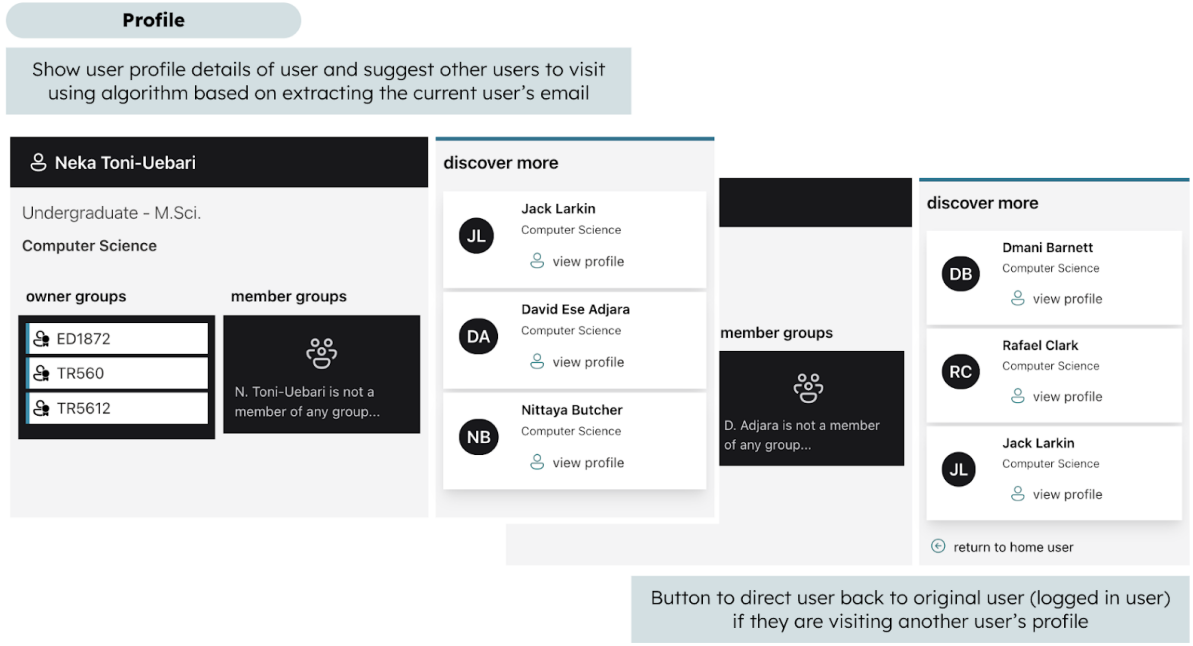
Render recent and upcoming events

```
private async loadEvents(): Promise<void> {
   try {
     /* Check if values for properties exist */
     if (!this.props.list || !this.props.siteUrl ||
!this.props.eventStartDate.value || !this.props.eventEndDate.value) return;

     const eventsData: IEventData[] = await
UserEventService.getEvents(escape(this.props.siteUrl), escape(this.props.list),
this.props.eventStartDate.value, this.props.eventEndDate.value);

     const today = new Date().getTime();
     let eventsBefore = eventsData.filter((event: any) =>
Date.parse(event.EventDate) < today);
     eventsBefore = eventsBefore.slice(0, 3)

     /* Filtered list of 3 events max. after today's date */
     let eventsAfter: any = eventsData.filter((event: any) =>
Date.parse(event.EventDate) >= today);
     eventsAfter = eventsAfter.slice(Math.max(eventsAfter.length - 3, 0))
```

```
    /* Use filtered lists to render recent meetings and upcoming meetings */
    this.recentEvents = this.renderRecentEvents(eventsBefore);
    this.upcomingEvents = this.renderUpcomingEvents(eventsAfter);
    /* Re-render based on recent and upcoming events gathered */
    this.forceUpdate()

    this.setState({ eventData: eventsData, hasError: false, recentEvents:
eventsBefore, upcomingEvents: eventsAfter });
  } catch (e) {
    this.setState({ eventData: [], hasError: true })
  }
}
```

- Retrieves all existing events
  - Filters the first 3 events BEFORE and AFTER today's date: this indicates the recent events and upcoming events
  - Recent and upcoming events passed into corresponding rendering functions; list rendering

Profile - a web part which presents the profile details of a selected user



- Get user properties

Get user properties

```
public _getUserProperties = (email: string, discover: boolean): IUserProfile => {
  /* Email is either 'me' (current user) or zhacXXX@live.rhul.ac.uk (another
user's profile) */
  const serviceScope: ServiceScope = this.props.serviceScope;
```

```
   this.dataCenterServiceInstance =
serviceScope.consume(UserProfileService.serviceKey);

   /* Get user profile properties of the current user chosen */

this.dataCenterServiceInstance.getUserProfileProperties(email).then((userProfileI
tems: IUserProfile) => {

     /* Only retrieve properties if the user actually exists */
     if (userProfileItems.UserProfileProperties !== null &&
userProfileItems.UserProfileProperties !== undefined) {

       for (let i: number = 0; i < userProfileItems.UserProfileProperties.length;
i++) {

         if (userProfileItems.UserProfileProperties[i].Key ===
"msOnline-ObjectId") {
           userProfileItems.Id = userProfileItems.UserProfileProperties[i].Value;
         }

         [...]
       }

       /* Don't display users with blank first names or last names in suggestion
box */
       if (userProfileItems.FirstName.trim().length > 0 &&
userProfileItems.LastName.trim().length > 0) {
         /* Cases: either getting current user's properties or properties of a
user to discover */
         if (!discover) {
           this.setState({ usersToDiscover: [] })

           /* Update state to have properties of current user */

           const newProfile = new UserProfile();
           [...]

           /* Call current user's groups */
           this._getGroups(newProfile.Email)
           /* Suggest some users to discover based on selected user's page */
           this._getUsersToDiscover(newProfile.Email)

           this.setState({ currentUser: newProfile.Email, userProfileItems:
newProfile })
```

```
          /* Ignore the discovered user if it is the logged in user themself */
        } else if (userProfileItems.Email !== this.state.loggedInUser) {
          /* Update state to have the properties of a user to discover */
          const currentUsersToDiscover = this.state.usersToDiscover
          if (currentUsersToDiscover.length < 3) {
            currentUsersToDiscover.push(userProfileItems)
            this.setState({ usersToDiscover: currentUsersToDiscover })
          }
        }
        this.forceUpdate()
      }
    }
  }).catch((e) => console.log(e));
  return
}
```

- Service scope: use the service key to access the data centre service
- Call the service from UserProfileService to assign the appropriate user profile items to the items of a user profile.
- Validation checks: do not allow user profiles that are not found or user profiles with blank first or last names
  - Case 1: retrieving user profile properties of current user; create a new user profile
  - Case 2: discovering new users to suggest: push a new user profile into the list of ‘users to discover’ (until we receive three users)

- Suggest related users

Suggest related users

```
public _getUsersToDiscover = (email: string): void => {
  /* Initialise array */
  this.setState({ usersToDiscover: [] })

  if (email === 'me') {
    email = this.state.loggedInUser
  }
  /* Extract ZHAC code */
  const zhacCodeString = email.replace('i:0#.f|membership|', '')
  const matches = zhacCodeString.match('[0-9]+').toString()
  const zhacNumber = parseInt(matches)

  /* Random shuffle of potential user profile codes */
```

```
    let userCodeRange = [-1, -2, -3, -4, -5, 1, 2, 3, 4, 5]
    function shuffleArray(array: any) {
      for (let i = array.length - 1; i > 0; i--) {
        const j = Math.floor(Math.random() * (i + 1));
        const temp = array[i];
        array[i] = array[j];
        array[j] = temp;
      }
      return array
    }

    /* Padding zeros to three - e.g. 5 to 005 */
    function pad(n: number, length: number) {
      let len = length - ('' + n).length;
      return (len > 0 ? new Array(++len).join('0') : '') + n
    }

    /* Shuffle the range of profiles and extract the number to use as email
parameter
    i.e. Current user zhac020@live.rhul.ac.uk -> Random user code {20 + 2} -> 021
    -> Find user properties of user 'zhac022@live.rhul.ac.uk' */

    userCodeRange = shuffleArray(userCodeRange)
    const zhacCodeRanges = userCodeRange.map(c => pad((c + zhacNumber), 3))

    /* Variables for iterator */
    let i = 0

    /* Some ZHAC-XXX codes might be out of range or invalid so check through the
range at least */
    while (this.state.usersToDiscover.length < 3 && i < zhacCodeRanges.length) {
      /* Three users to discover finally found */
this._getUserProperties(`i:0#.f|membership|zhac${zhacCodeRanges[i]}@live.rhul.ac.
uk`, true)
      /* Increment */
      i++;
    }
  }
```

- Extraction of email and string manipulation: Extracts a user's email and uses its numbers to find suggested users i.e. 'zhac030@live.rhul.ac.uk' → a number of 030 → a user code range of [025, 026, 027, 028, 029, 031, 032, 033, 034, 035] which is shuffled and the first three valid emails are selected i.e. zhac{XXX}@live.rhul.ac.uk.
- Re-use method that gets the user's properties

Getting user profile properties: UserProfileService

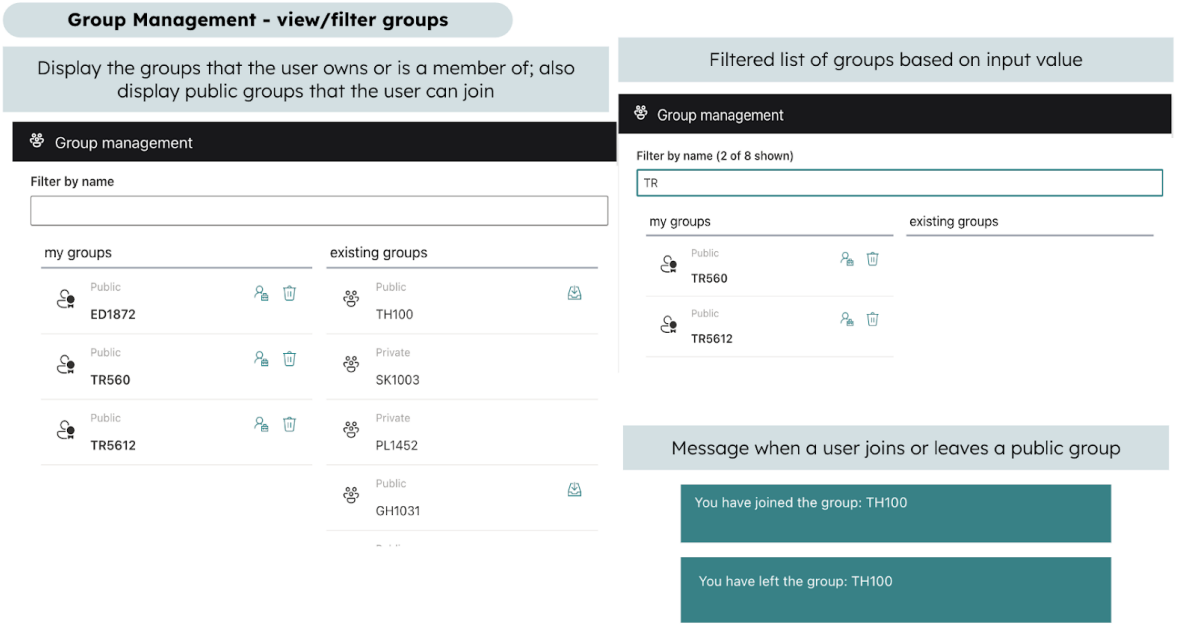
```
export class UserProfileService implements IDataService {
[...]
   public static readonly serviceKey: ServiceKey<IDataService> =
ServiceKey.create<IDataService>('userProfle:data-service', UserProfileService);
   private _spHttpClient: SPHttpClient;
   private _pageContext: PageContext;
   private _currentWebUrl: string;

[...]

   private readUserProfile(who: string): Promise<IUserProfile> {
       return new Promise<IUserProfile>((resolve: (itemId: IUserProfile) => void,
reject: (error: any) => void): void => {

           let targetURL = ''
           if (who === 'me') {
               /* Get profile properties of myself (the current user) */
               targetURL =
`${this._currentWebUrl}/_api/SP.UserProfiles.PeopleManager/GetMyProperties`
           } else {
               /* Get profile properties of another user (the URL would contain
the target e.g. 'i:0#.f|membership|zhacXXX@live.rhul.ac.uk' */
               targetURL =
`${this._currentWebUrl}/_api/SP.UserProfiles.PeopleManager/GetPropertiesFor(accou
ntName=@v)?@v='${encodeURIComponent(who)}'`
           }
           this._spHttpClient.get(targetURL,
               SPHttpClient.configurations.v1,
               {
                   headers: {
                       'Accept': 'application/json;odata=nometadata',
                       'odata-version': ''
                   }
               })
               .then((response: SPHttpClientResponse): Promise<{ value:
IUserProfile }> => {
                   return response.json();
               })
               .then((response: { value: IUserProfile }): void => {
                   //resolve(response.value);
                   const output: any = JSON.stringify(response);
                   resolve(output);
```

```
            }, (error: any): void => {
                reject(error);
            });
        });
    }

    private processUserProfile(orgChartItems: any): any {
        return JSON.parse(orgChartItems);
    }
}
```

- Service scope and context: the service scope requires the dependencies of the SP HTTP client and the page context using service keys.
- Get user profile properties: Reads and resolves new user profile
- Read user profile - uses SP HTTP client to receive a response from GET request
  - Case 1 (myself): retrieve the user profile properties of the current user
  - Case 2 (other users): retrieve the user profile properties of a specified user
- Process user profile - returns the JSON parsing of the profile properties items

Group Management - a web part which manages groups of the system

- View groups - display the user's groups



View groups: view

```
public _getGroups = (): void => {
    UserGroupService.getGroups().then(groups => {

      this.setState({
```

```
      isLoading: false,
      groups: groups,
      loadCount: this.state.loadCount + 1
    });

    UserGroupService.getMyOwnerGroups(groups, 'me').then(groups => {
      this.setState({
        ownerGroups: groups.map((item: { id: any; }) => item.id),
        loadCount: this.state.loadCount + 1
      });
    }).catch((e: any) => console.log(e));

    UserGroupService.getMyMemberGroups(groups, 'me').then(groups => {
      this.setState({
        memberGroups: groups.map(item => item.id),
        loadCount: this.state.loadCount + 1
      });
    }).catch((e: any) => console.log(e));
  }).catch((e: any) => console.log(e));
}
```

- **View:** Retrieve existing groups (including the groups that the user is an owner or a member of)
  - Sets the state based on what groups are found by the user
  - Exception handling: to handle asynchronous call errors

View groups: UserGroupService

```
public getGroups = async (): Promise<IGroup[]> => {
  /* Get items from SP list Groups */
  const groups = await this._sp.web.lists.getByTitle("Groups").items()

  return new Promise<IGroup[]>((resolve) => {
    try {
      const spGroups: Array<IGroup> = new Array<IGroup>();

      /* Push each group item into SP (Sharepoint Lists) Groups list */
      groups.map((item: any) => {
        spGroups.push({
          id: item.Title,
          displayName: item.field_1,
          description: item.field_2,
          visibility: item.field_7,
          SPId: item.ID
```

```
        });
      });

      resolve(spGroups);

    } catch (error) {
      console.error(error);
    }
  });
}
```

- Service: Populate data about groups into a model instance of group (IGroup)
  - Retrieves all the items of the SP List, Groups
- Exception handling: if no groups are found

- Filter groups - display the user's group based on filtered text

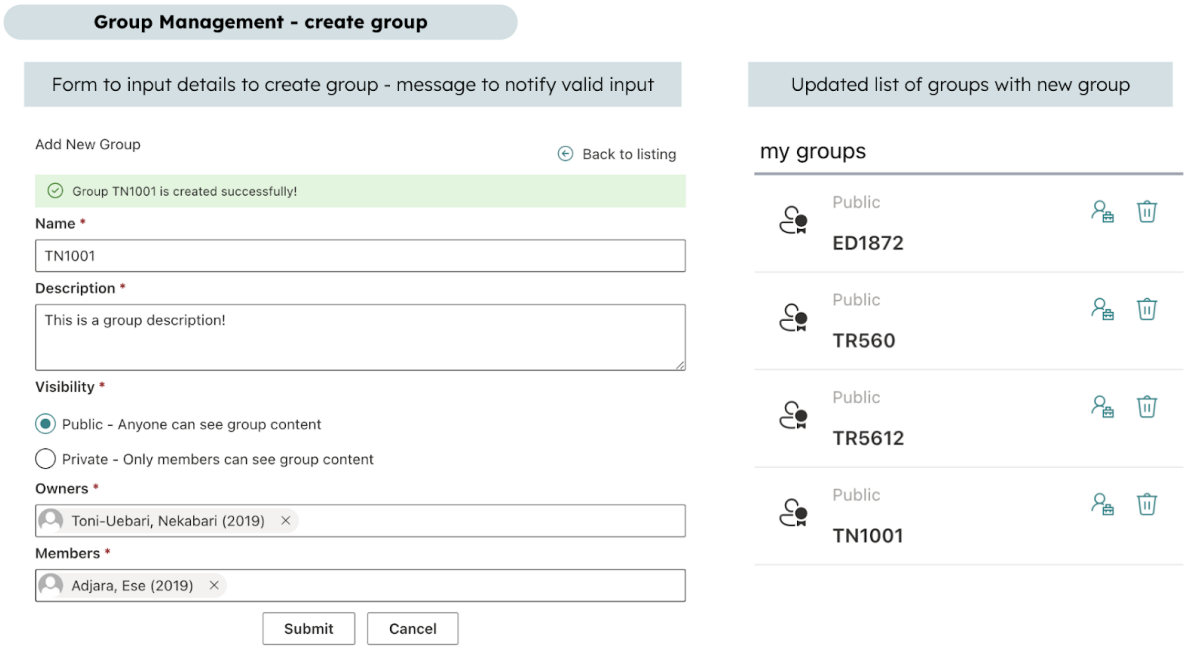
Filter groups

```
public render(): React.ReactElement<IGroupListProps> {
   const { groups = [] } = this.state;
   const resultCountText = groups.length === this._originalItems.length ? '' : `
(${groups.length} of ${this._originalItems.length} shown)`;

  return (
   [...]
   <TextField label={'Filter by name' + resultCountText}
onChange={this._onFilterChanged} />
   [...]
)}

[...]

private _onFilterChanged = (_: any, text: string): void => {
  this.setState({
    filterText: text,
    groups: text ? this._originalItems.filter(item =>
item.displayName.toLowerCase().indexOf(text.toLowerCase()) >= 0) :
this._originalItems
  });
}
```

- Event handler: event fired when value of text field is changed

        - Filter groups based on current value for filter text matching with the group display names
    - Render based on filtered item results

- Create group - create a new group



Create groups: view

```
private _getPeoplePickerOwners = (items: IPeoplePickerUserItem[]) => {
    this.setState(() => {
        return {
            ...this.state,
            owners: items.map(x => x.id.replace('i:0#.f|membership|', ''))
        };
    });
}

private _getPeoplePickerMembers = (items: IPeoplePickerUserItem[]) => {
    this.setState(() => {
        return {
            ...this.state,
            members: items.map(x => x.id.replace('i:0#.f|membership|', ''))
        };
    });
}

private onchangedName = (groupName: any) => {
    this.setState({ name: groupName.target.value });
}
```

```
   private onchangedDescription = (groupDescription: any) => {
       this.setState({ description: groupDescription.target.value });
   }

   private onChangeVisibility = (ev: React.FormEvent<HTMLInputElement>, option:
IChoiceGroupOption) => {
       this.setState({ visibility: option.key });
   }

   private createNewGroup = () => {
       try {
           UserGroupService.createGroup(this.state.name, this.state.description,
this.state.visibility, this.state.owners, this.state.members).catch(e =>
this.handleGroupError(e));

           this.setState({
               message: "Group " + this.state.name + " is created successfully!",
               showMessageBar: true,
               messageType: MessageBarType.success
           });
       } catch (error) {
           this.handleGroupError(error)
       }
   }
```

- Event handlers -
  - The people picker selects users from within the RHUL tenant. The owners and members are processed to only have the emails of the users
  - Input of the text fields and checkboxes for onChange[...] - detects any change of the input value (i.e. for the name, description and visibility of the group)
- Create new group -
  - Clicking 'submit' will call the service to create the group with the state's current values based on the input values
- Exception handling - to handle any invalid input or errors in asynchronous calls.

Create groups: UserGroupService

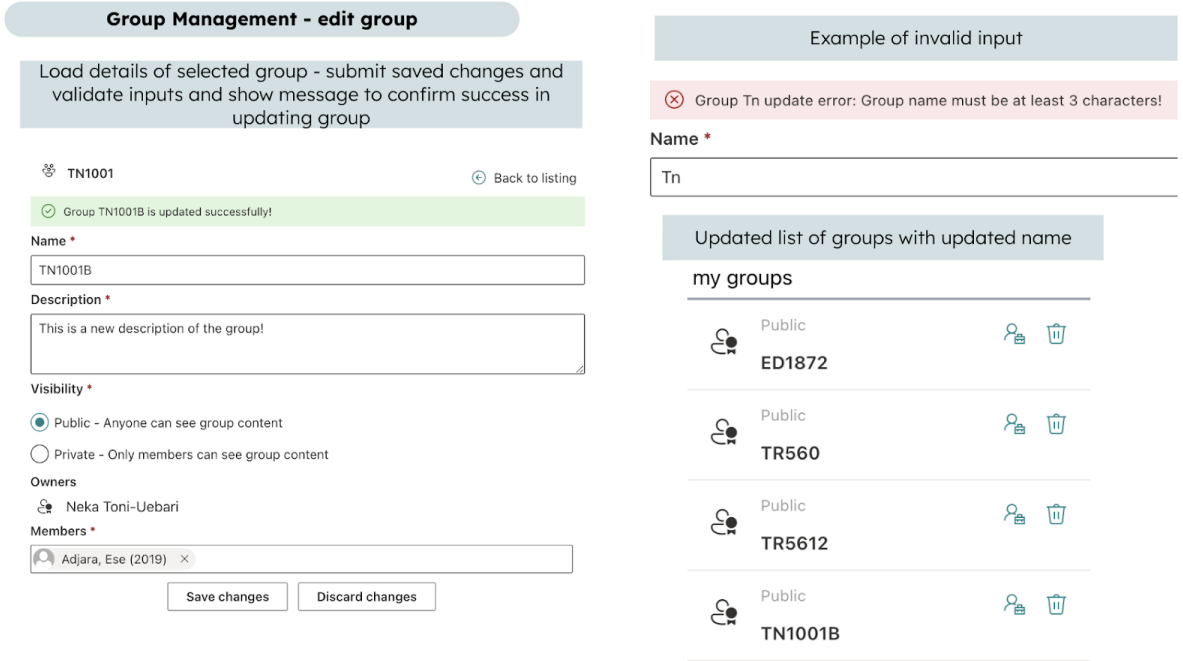
```
public createGroup = async (groupName: string, groupDescription: string,
groupVisibility: string, groupOwners: string[], groupMembers: string[]):
Promise<void> => {
   try {
     /* ID creation: generated using timestamp */
     const generatedGroupId = 'g-' + (Date.now() + Math.random()).toString()
```

```
const groupRequest: any = {
  Title: generatedGroupId,
  field_1: groupName,
  field_2: groupDescription,
  field_3: "Unified",
  field_4: true,
  field_5: groupName.replace(/\s/g, ""),
  field_6: false,
  field_7: groupVisibility,
};


/* SP post request to create new group */
this._sp.web.lists.getByTitle("Groups").items.add(groupRequest);

/*------ SP post request to add owners of the new group */
const [batchedSP, execute] = this._sp.batched();
const ownersList = batchedSP.web.lists.getByTitle("GroupOwners");
const membersList = batchedSP.web.lists.getByTitle("GroupMembers");
let res: any[] = [];
res = [];

/* Add SP batch for adding group owners */
for (let i = 0; i < groupOwners.length; i++) {
  /*
  GROUP OWNER REQUEST COLUMNS:
  Title: the generated group ID
  field_1: the email of group owner
  */
  ownersList.items.add({
    Title: generatedGroupId,
    field_1: groupOwners[i]
  }).then(r => res.push(r))
    .catch(e => console.log(e));
}

/* Add SP batch for adding group members */
for (let i = 0; i < groupMembers.length; i++) {
  /*
  GROUP MEMBER REQUEST COLUMNS:
  Title: the generated group ID
  field_1: the email of group member
  */
  membersList.items.add({
```

```
        Title: generatedGroupId,
        field_1: groupMembers[i]
      }).then(r => res.push(r))
        .catch(e => console.log(e));
    }

    /* Execute batch for owners and members */
    await execute();
  } catch (e) {
    console.log(e);
  }
}
```

- Generate group ID for new group - generated using randomised number and timestamp
- Create a group request to add into SP List “add” operation to the SP List, “Groups”
- SP Batch execution to add group owners
- SP Batch execution to add group members

- Edit group - edit an existing group



Edit group: view

```
private updateGroup = () => {

    try {
        let success = true;
        let errorMessage = '';
```

```
            /* Before and after member list of changes */
            const membersBefore = this.state.originalState.members
            const membersAfter = this.state.members

            /* The members to add and/or remove from the group based on saved
changes */
            const membersToAdd = membersAfter.filter((m: any) =>
membersBefore.indexOf(m) < 0)
            const membersToRemove = membersBefore.filter((m: any) =>
membersAfter.indexOf(m) < 0)

            /* Validation check */
            let newName = this.state.name
            let newDescription = this.state.description

            /* Validate name */
            if (this.state.name.trim().length === 0) {
                /* Keep name as the placeholder value */
                newName = this.state.originalState.name
            } else if (this.state.name.trim().length > 0 &&
this.state.name.trim().length < 3) {
                /* Error message: must be at least 3 characters */
                success = false
                errorMessage = 'Group name must be at least 3 characters!'
                [...]
            }

            /* Validate description */
            if (this.state.description.trim().length === 0) {
                /* Keep description as the placeholder value */
                newDescription = this.state.originalState.description
            } else if (this.state.description.trim().length > 0 &&
this.state.description.trim().length < 3) {
                /* Error message: must be at least 3 characters */
                success = false
                errorMessage = 'Description must be at least 3 characters!'
                [...]
            }

            /* Validate members */
            /* Check that there are no added members that are already 'owners' of
the group */
            const ownerEmails = this.state.originalState.owners.map((m: any) =>
```

```
m.email)
            const includesOwners = this.state.members.filter(m =>
ownerEmails.indexOf(m) >= 0)
            if (includesOwners.length > 0) {
                /* Error message: cannot add members that are already owners of
group */
                success = false
                errorMessage = 'Cannot have members that are already owners of the
group!'
                [...]
            }

            /* Valid changes */
            if (success) {
                UserGroupService.editGroupDetails(this.state.spId, this.state.id,
newName, newDescription, this.state.visibility, membersToAdd,
membersToRemove).catch(e => this.handleGroupError(e));
                [...]
            }
        } catch (error) {
            this.handleGroupError(error)
        }
    }
```

- The event handlers very similar to the view of 'Create Group'
- Compute members to add or remove based on state of the members before and after the changes
- Validation of the inputs:
  - Any unchanged field should be placeholder
  - Name: must be at least three characters
  - Description: must be at least three characters
  - Members: cannot have members that are already owners of the group
- Call service to edit the group if the details are valid (including the members to add or remove).
- Exception handling for the case that the update request fails.

Edit group: UserGroupService

```
public editGroupDetails = async (spGroupID: number, groupId: string, groupName:
string, groupDescription: string, groupVisibility: string, membersToAdd:
string[], membersToRemove: string[]): Promise<void> => {
    try {
      /* In context to internal fields,
      field_1: the group name
      field_2: the group description
```

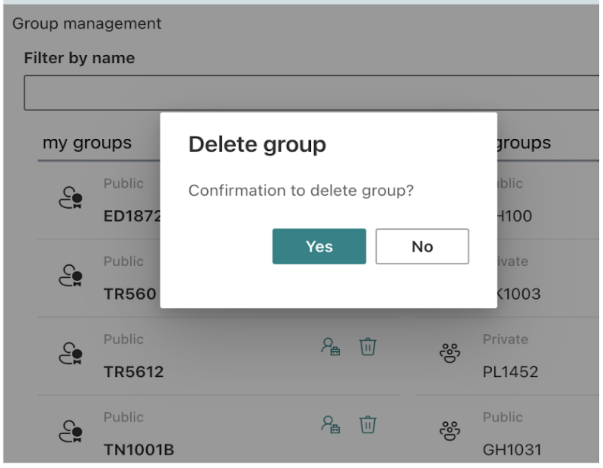
```
    field_7: the group visibility */
    const updateRequest: any = {
      field_1: groupName,
      field_2: groupDescription,
      field_7: groupVisibility,
    };

    /* Update group details */
    const groups = this._sp.web.lists.getByTitle("Groups");
    await groups.items.getById(spGroupID).update(updateRequest);

    /* Batch update for adding or removing members from group */
    if (membersToAdd.length > 0) {
      await this.addMembersToGroup(groupId, membersToAdd)
    }
    if (membersToRemove.length > 0) {
      await this.removeMembersFromGroup(groupId, membersToRemove)
    }

  } catch (e) {
    console.log(e)
  }
}
```

- Retrieve the SP ID of the item from SP List “Groups”
- Create a group request to update into SP List “update” operation to the SP List, “Groups”
- Further operations to add or remove any members required

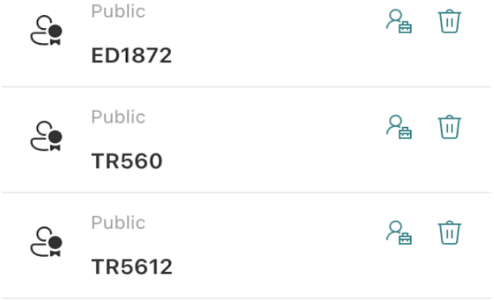
- Delete group - delete an existing group

Group Management - delete group

Confirmation box asking user to delete group



Updated list of groups - e.g. Group *‘TN100B’* is deleted



Delete group: view

```
private _manageDeleteGroup = (confirm: boolean) => {
  /* If option is yes */
  if (confirm) {
    const groupId = this.state.selectedGroup.id
    const groupName = this.state.selectedGroup.displayName
    UserGroupService.deleteGroup(groupId).then(response => {
      /* Filter out the deleted group */
      this._originalItems = this.state.groups.filter(group => group.id !==
groupId)
      this.setState({
        groups: this._originalItems
      })

      /* Re-check groups with map function and show confirmation message */
      this.setState(prevState => ({
        groups: prevState.groups.map(group => group.id === groupId ? { ...group,
userRole: "" } : group),
        isTeachingBubbleVisible: true,
        techingBubbleMessage: 'You have deleted group: ' + groupName
      }));

    }).catch(e => console.log(e));
  }

  /* Back to default: no group to delete now */
  this.setState({
    showDialog: false,
    selectedGroup: null
  });

  this.forceUpdate();
 }
```

- Check confirm is true - did the user click ‘YES’ in the confirm dialog when asking to delete the group
- If so, then call the service to delete the group and update the state of the view, including showing a message to the user to confirm that the group was deleted.

Delete group: UserGroupService

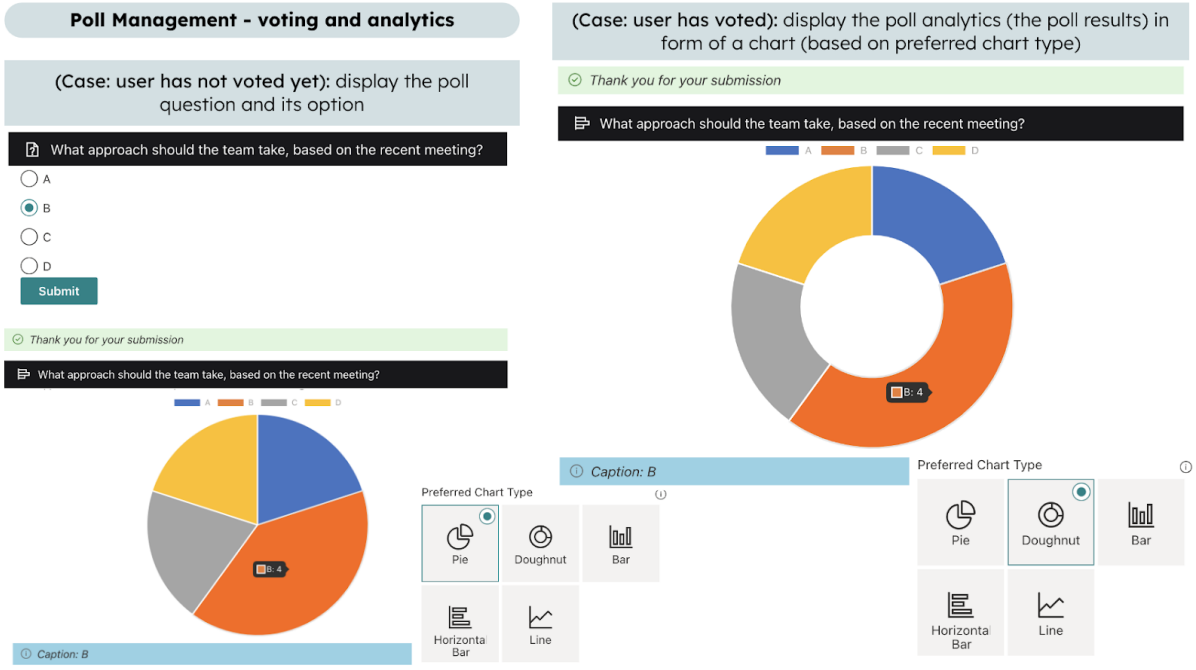
```
public deleteGroup = async (groupId: string): Promise<void> => {
  try {
    /* Retrieve the Sharepoint list ID (required to delete item with) */
```

```
      const groupToDelete = await
this._sp.web.lists.getByTitle("Groups").items.filter(`Title eq '${groupId}'`)()

      /* If item is found */
      if (groupToDelete[0]) {
        /* Delete using the Sharepoint list item ID (note: this is different to
the groupId) */
        await
this._sp.web.lists.getByTitle("Groups").items.getById(groupToDelete[0].ID).delete
();
      }

      /* SP List for group members and owners to delete */
      const [batchedSP, execute] = this._sp.batched();
      const ownersList = batchedSP.web.lists.getByTitle("GroupOwners");
      const membersList = batchedSP.web.lists.getByTitle("GroupMembers");

      const groupOwners = await ownersList.items.filter(`Title eq '${groupId}'`)()
      const groupMembers = await membersList.items.filter(`Title eq
'${groupId}'`)()

      let res: any[] = [];
      res = [];

      /* Add SP batch for deleting group owners of deleted group ID */
      for (let i = 0; i < groupOwners.length; i++) {
        ownersList.items.getById(groupOwners[i].ID).delete().then(r =>
res.push(r))
          .catch(e => console.log(e));
      }

      /* Add SP batch for deleting group members of deleted group ID */
      for (let i = 0; i < groupMembers.length; i++) {
        membersList.items.getById(groupMembers[i].ID).delete().then(r =>
res.push(r))
          .catch(e => console.log(e));
      }

      /* Execute batch for deleted owners and members */
      await execute();
    } catch (e) {
      console.log(e)
    }
  }
```

- Retrieve the SP ID of the item from SP List “Groups”
- Use ID of item to perform SP List “delete” operation from the SP List, “Groups”
- Further operations to remove all owners and members of the group, from SP Lists “GroupMembers” and “GroupOwners”

Poll Management - a web part which manages the polls of the system

- Submit poll - submit a vote for a poll



Submit poll: UserPollService

```
public isubmitResponseToPoll = async (userResponse: IResponseDetails):
Promise<void> => {
        try {
            /* SP List PollResponses fields:
            Title,
            field_1: Email
            field_2: Response */

            const responseRequest: any = {
                Title: userResponse.QuestionID,
                field_1: userResponse.UserEmail,
                field_2: userResponse.PollResponse
            }

            await this._pollResponses.items.add(responseRequest)
```

```
    } catch (err) {
        console.log(err);
    }
}
```

- Uses the model ‘IResponseDetails’ consisting the ID of the poll, the user’s email and the poll response
- Create request to add new poll response into SP List, “PollResponses”

- Poll analytics - the analytics of the poll (based on poll responses)

Poll analytics

```
public _bindResponseAnalytics = (): void => {
  const { currentPoll } = this.state;
  const tmpUserResponse: any = this.state.pollResponse;

  /* Check that response exists */
  if (tmpUserResponse && tmpUserResponse.length > 0) {
    const pChoices: string[] = currentPoll.Choices.split(',');
    const finalData: any[] = [];

    const tempData: any = _.countBy(tmpUserResponse, 'PollResponse')

    pChoices.map((label) => {
      if (tempData[label.trim()] === undefined) {
        finalData.push(0);
      } else finalData.push(tempData[label.trim()]);
    });

    const pollAnalytics: IPollAnalyticsInfo = {
      ChartType: this.props.chartType,
      Labels: pChoices,
      Question: currentPoll.DisplayName,
      PollResponse: finalData
    };

    this.setState({
      showProgress: false,
      showOptions: false,
      showChartProgress: false,
      showChart: true,
      PollAnalytics: pollAnalytics
```

```
    })
  }
}
```

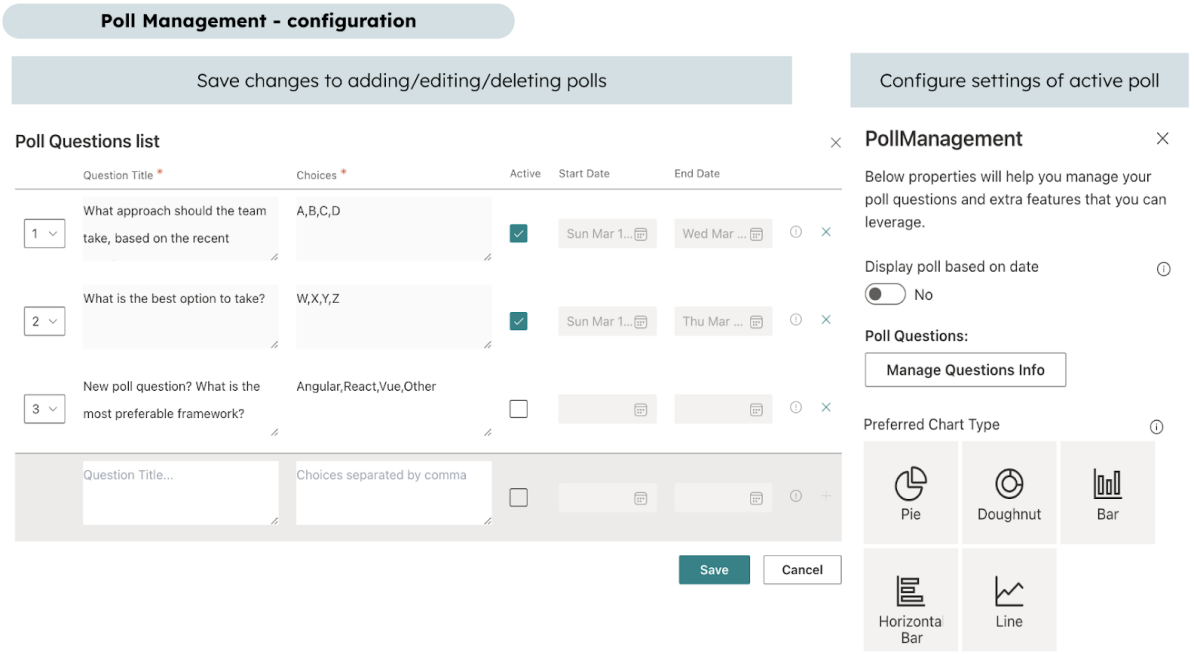
- Must check that the response from the poll actually exists - the chart should only be shown if the poll vote is submitted
- Count the votes for each option in the poll
- Display chart of vote results based on chart type configured (i.e. pie chart, bar chart, line graph, doughnut etc.).
  - Update state: currently has analytics for the poll, thus display the chart instead of the options

Get poll responses: UserPollService

```
public igetPollResponses = async (pollId: any): Promise<any> => {
      /* Get items from poll responses of specific poll */
      const pollResponses = await this._pollResponses.items.filter(`Title eq
'${pollId}'`)()

      return new Promise<any>((resolve) => {
          try {
              const spPollResponses: IResponseDetails[] = [];
              pollResponses.map((item: any) => {
                  spPollResponses.push({
                      QuestionID: item.Title,
                      UserEmail: item.field_1,
                      PollResponse: item.field_2
                  })
              })
              resolve(spPollResponses)
          } catch (e) {
              console.error(e);
          }
      });
  }
```

- Service: Populate data about poll responses into a model instance of details about the poll response (IResponseDetails)
  - Retrieves all the items of the SP List, Poll Responses, that are of a particular poll (hence the parameter of the poll ID to select with)
- Exception handling: if no poll responses are found or the poll ID is invalid

- Configure poll display details - management of polls (display polls based on date, add, edit, delete polls, set preferred chart type etc.)



Get polls: UserPollService

```
public igetPolls = async (questions?: any[]): Promise<any> => {
        /* Get items from polls */
        const polls = await this._polls.items()

        return new Promise<any>((resolve) => {
            try {
                const spPolls: IQuestionDetails[] = [];
                polls.map((item: any) => {
                    spPolls.push({
                        Id: item.Title,
                        DisplayName: item.field_1,
                        Choices: item.field_2,
                        Visibility: item.field_3,
                        UseDate: false,
                        StartDate: new Date(item.field_4),
                        EndDate: new Date(item.field_5),
                        Owner: item.field_6,
                        SortIdx: item.ID,
                        SPId: item.ID
                    })
                })

                resolve(spPolls)
            } catch (e) {
```

```
                console.error(e);
            }
        })

    }
```

- Service: Populate data about groups into a model instance of poll (IQuestionDetails)
  - Retrieves all the items of the SP List, Polls
- Exception handling: if no polls are found

Create new poll: UserPollService

```
public icreatePoll = async (pollUniqueID: string, pollQuestion: string, options:
string, visibility: string, startDate: any, endDate: any): Promise<void> => {
        /* Add poll into polls */
        const pollRequest: any = {
            Title: pollUniqueID,
            field_1: pollQuestion,
            field_2: options,
            field_3: visibility,
            field_4: startDate,
            field_5: endDate,
            field_6: this.currentUser.Email
        }

        /* SP post request to create new group */
        await this._polls.items.add(pollRequest);
    }
```

- Create a poll request from the input fields by the user for the attributes of the new poll.
- Use the poll request to add into SP List “add” operation to the SP List, “Polls”

Edit poll: UserPollService

```
public ieditPoll = async (pollId: string, visibility: string, startDate: any,
endDate: any): Promise<void> => {
        /* User only has the ability to change the visibility of poll, or the
start and end date */
        try {
            const pollToEdit = await this._polls.items.filter(`Title eq
```

```
'${pollId}'`)()
            const updateRequest: any = {
                field_3: visibility,
                field_4: startDate,
                field_5: endDate
            }

            /* Update poll details */
            if (pollToEdit[0]) {
                await
this._polls.items.getById(pollToEdit[0].ID).update(updateRequest);
            }

        } catch (e) {
            console.log(e)
        }
    }
```

- Check that the poll to edit actually exists; if so, then get its ID
- Update request to particular poll in SP list, Polls, based on user's input fields
- Exception handling: for any failed asynchronous calls

Delete poll: UserPollService

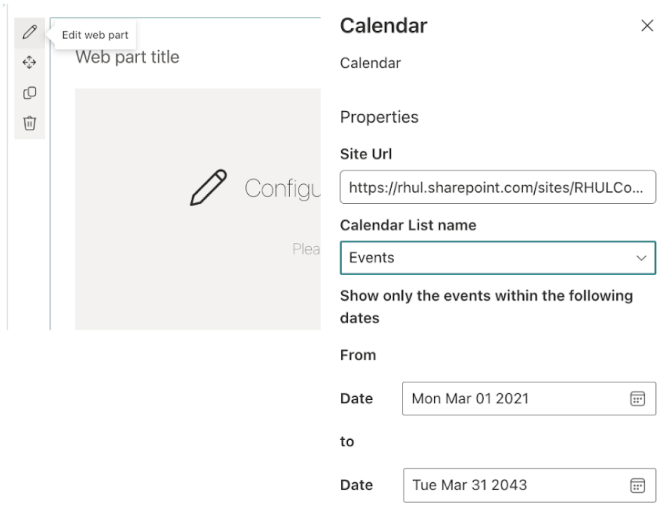
```
public ideletePoll = async (pollId: string): Promise<void> => {
        /* Retrieve the Sharepoint list ID (required to delete item with) */
      const pollToDelete = await this._polls.items.filter(`Title eq
'${pollId}'`)()

      /* If item is found */
      if (pollToDelete[0]) {
        /* Delete using the Sharepoint list item ID (note: this is different to
the groupId) */
        await this._polls.items.getById(pollToDelete[0].ID).delete();
      }

      /* SP List for poll responses to delete */
      let pollResponses = await this.igetPollResponses(pollId)
      pollResponses = pollResponses.map((p: any) => p.userEmail)

      /* Finally remove all responses from poll */
      await this.iremoveResponsesFromPoll(pollId, pollResponses)
    }
```

- Check if the poll to delete exists (by retrieving its SP ID)
- If so, then perform SP “delete” operation on SP List, Polls
  - Also, delete all responses from the deleted poll (SP List, PollResponses)

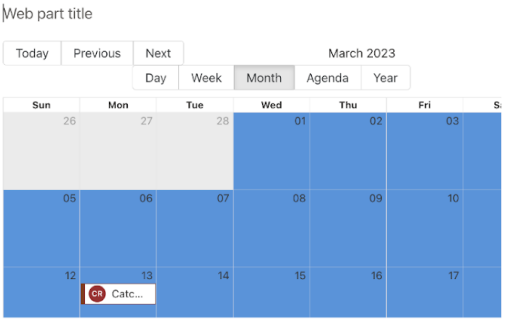
Calendar - a webpart which manages the events of the system; i.e. the calendar

Calendar - configuration

(Case: there is no list loaded): open web part configuration panel to select site URL, list and date range to get events within.

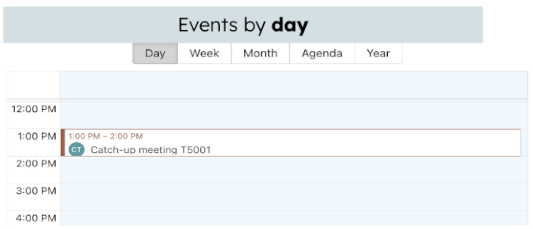


Edit web part title and change view of calendar

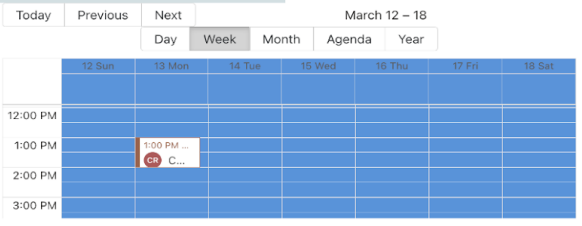


Calendar - views

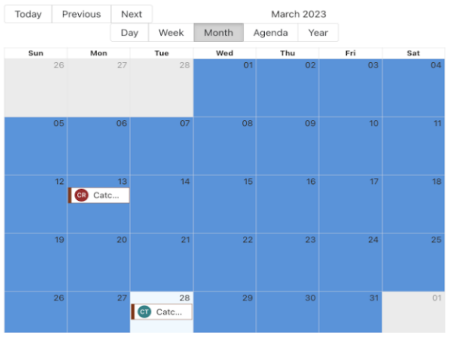
Events by **day**



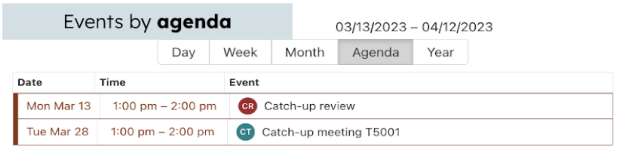
Events by **week**



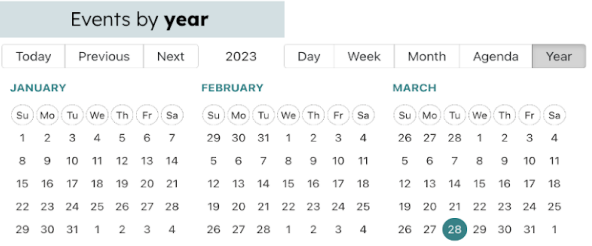
Events by **month**



Events by **agenda**



Events by **year**



- View events

View events

```
private async loadEvents(): Promise<void> {
   try {
     // Teste Properties
     if (!this.props.list || !this.props.siteUrl ||
!this.props.eventStartDate.value || !this.props.eventEndDate.value) return;

     this.userListPermissions = await
UserEventService.getUserPermissions(this.props.siteUrl, this.props.list);

     const eventsData: IEventData[] = await
UserEventService.getEvents(escape(this.props.siteUrl), escape(this.props.list),
this.props.eventStartDate.value, this.props.eventEndDate.value);

     this.setState({ eventData: eventsData, hasError: false, errorMessage: "" });
   } catch (error) {
     this.setState({ hasError: true, errorMessage: error.message, isloading:
false });
   }
 }
```

  - Load events: requires the SP list of events selected, Sharepoint site URL, the event start date, the event end date.
  - Get user permissions of the SP list
  - Populate events into the model, IEventData.

View events: UserEventService

```
public async getEvents(siteUrl: string, listId: string, eventStartDate: Date,
eventEndDate: Date): Promise<IEventData[]> {

       let events: IEventData[] = [];
       if (!siteUrl) {
           return [];
       }
       try {
           // Get Category Field Choices
           const categoryDropdownOption = await
this.getChoiceFieldOptions(siteUrl, listId, 'Category');
           const categoryColor: { category: string, color: string }[] = [];
           for (const cat of categoryDropdownOption) {
```

```
                categoryColor.push({ category: cat.text, color: await
this.colorGenerate() });
            }

            const results = await
this._sp.web.lists.getById(listId).renderListDataAsStream(
                {
                    DatesInUtc: true,
                    ViewXml: [...]
                }
            );

            if (results && results.Row.length > 0) {
                let event: any = '';
                const mapEvents = async (): Promise<boolean> => {
                    for (event of results.Row) {
                        [...]

                        for (const attendee of event.ParticipantsPicker) {
                            attendees.push(parseInt(attendee.id));
                        }

                        events.push({
                            Id: event.ID,
                            ID: event.ID,
                            EventType: event.EventType,
                            title: await this.deCodeHtmlEntities(event.Title),
                            [...]
                        });
                    }
                    return true;
                };
                [...]
            const parseEvt: parseRecurrentEvent = new parseRecurrentEvent();
            events = parseEvt.parseEvents(events, null, null);

            // Return Data
            return events;
        } catch (error) {
            console.dir(error);
            return Promise.reject(error);
        }
    }
```

- Check that URL site exists: otherwise empty list of events
- Get the category field choices
- Render the list data using XML (includes the event start date and end date in a particular data string format)
- Check that any results exists:
  - Map the events by processing each attribute of each event in the appropriate manner
  - Finally push valid events into a list that can be returned
- Check for any events saved in local storage (the cache)
- Finally parse the events and return the final result
- Exception handling: there are several methods thus there are many opportunities for processed data to be invalid. These exceptions must be caught and handled appropriately.

- Get event

Get event

```
- private async renderEventData(eventId?: number): Promise<void> {

   this.setState({ isloading: true });
   const event: IEventData = !eventId ? this.props.event : await
UserEventService.getEvent(this.props.siteUrl, this.props.listId, eventId);
[...]
}
```

- Get the selected event and render based on attributes e.g. get the event type, the geo location, the description etc.
- Also checks if the event should be in “edit mode” (render a panel to edit the details of the event and save changes)

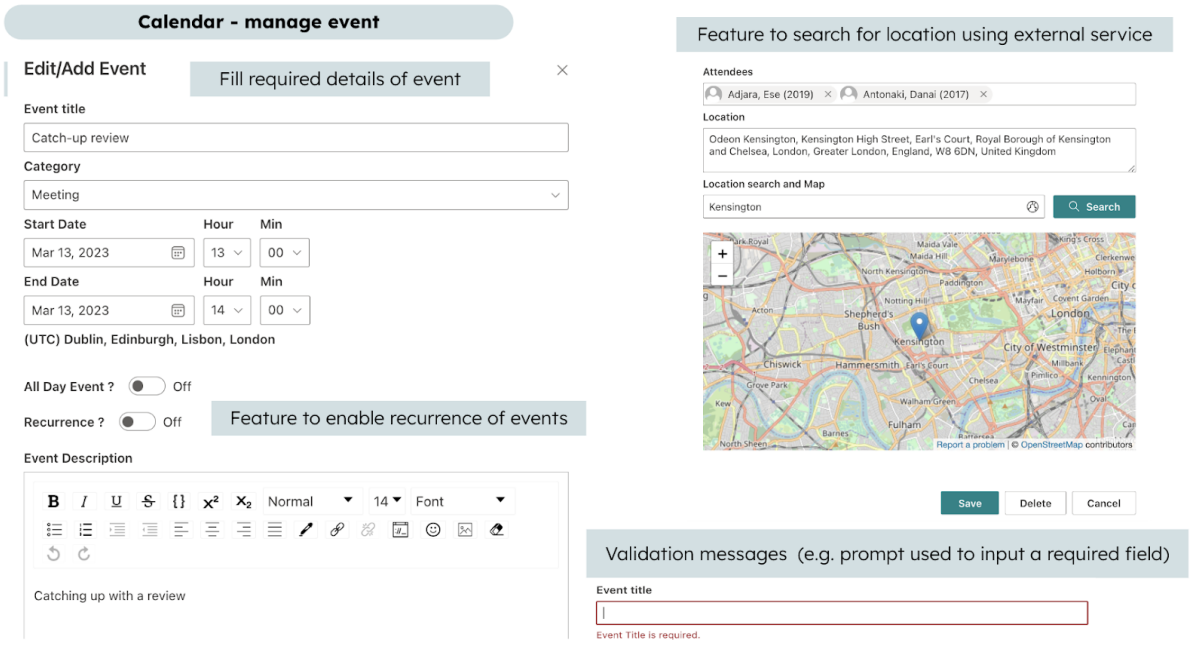
Get event: UserEventService

```
-public async getEvent(siteUrl: string, listId: string, eventId: number):
Promise<IEventData> {
       let returnEvent: IEventData = undefined;
       try {
           const event = await
this._sp.web.lists.getById(listId).items.getById(eventId)
               .select([...])
               .expand("Author")
               ();


           const eventDate = await this.getLocalTime(event.EventDate);
```

```
            const endDate = await this.getLocalTime(event.EndDate);

            returnEvent = {
                Id: event.ID,
                ID: event.ID,
                EventType: event.EventType,
                title: await this.deCodeHtmlEntities(event.Title),
                [...]
            };
        }
        catch (error) {
            return Promise.reject(error);
        }
        return returnEvent;
    }
```

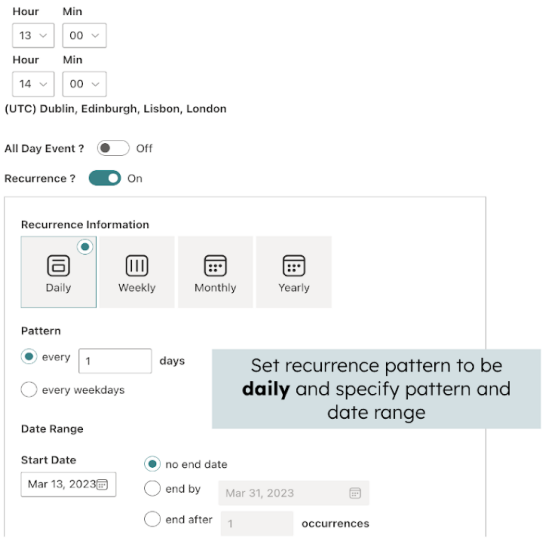
- Retrieve the SP List for events and select all of the specified attributes and expand to get the related fields from a lookup column.
- Get the local time of the event date and its end date
- Return an object of the event and all of its attributes
- Exception handling: reject any exception found while doing asynchronous calls for the SP list of events.

- Create and Edit event



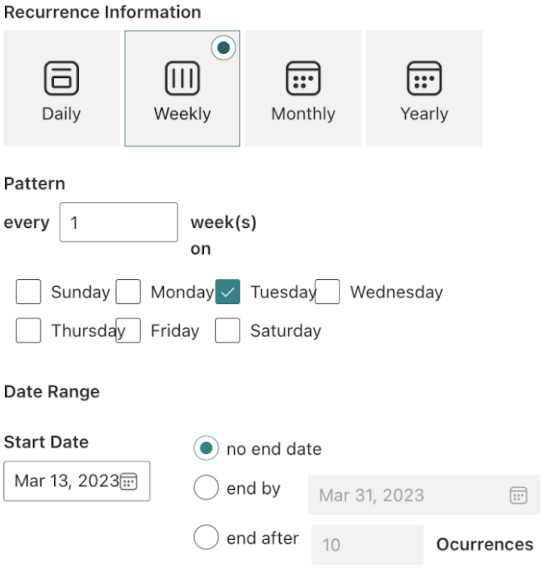
Calendar - manage event (recurrences)

Feature to include recurrence of events



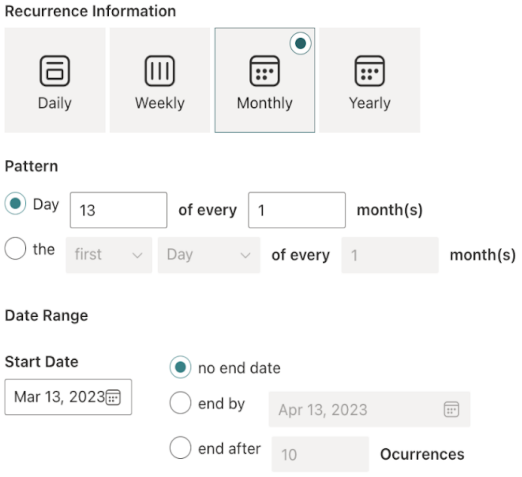
Set recurrence pattern to be **daily** and specify pattern and date range

Set recurrence pattern to be **weekly** and specify pattern and date range

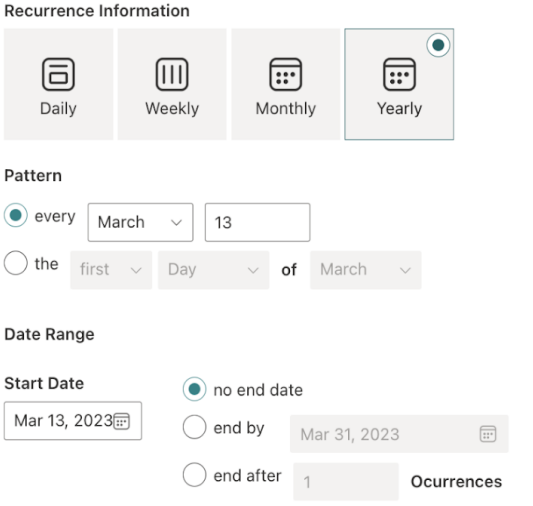


Calendar - manage event (recurrences)

Set recurrence pattern to be **monthly** and specify pattern and date range



Set recurrence pattern to be **yearly** and specify pattern and date range



Create event

```
private async onSave(): Promise<void> {
  const eventData: IEventData = this.state.eventData;
  let panelMode = this.props.panelMode;
  let startDate: string = null;
  let endDate: string = null;
  eventData.fRecurrence = false;

  [...]
    switch (panelMode) {
      case IPanelModelEnum.edit:
```

```
          await UserEventService.updateEvent(eventData, this.props.siteUrl,
this.props.listId);
          break;
        case IPanelModelEnum.add:
          await UserEventService.addEvent(eventData, this.props.siteUrl,
this.props.listId);
          break;
        default:
          break;
      }
[...]
 }
```

- Render the placeholder input fields based on the selected event (if there is any), otherwise this is blank
- The action that onSave should do is based on the panel mode: the mode is either set to update an event or create a new event.

Create event: UserEventService

```
public async addEvent(newEvent: IEventData, siteUrl: string, listId: string):
Promise<any> {
        let results = null;
        try {
            results = await this._sp.web.lists.getById(listId).items.add({
                Title: newEvent.title,
                Description: newEvent.Description,
                Geolocation: newEvent.geolocation,
                [...]
            });
        }
        catch (error) {
            return Promise.reject(error);
        }
        return results;
    }
```

- Await SP list operation to add a new event with the given attributes into the SP list of events. If the results are valid, then return the promise.
- Exception handling: reject any exception found while doing asynchronous calls for the SP list of events

Edit event: UserEventService

```
public async updateEvent(updateEvent: IEventData, siteUrl: string, listId:
string): Promise<any> {
        let results = null;
        try {
            // delete all recursive extensions before update recurrence event
            if (updateEvent.EventType.toString() === "1") await
this.deleteRecurrenceExceptions(updateEvent, siteUrl, listId);

            const eventDate = await this.getUtcTime(updateEvent.EventDate);
            const endDate = await this.getUtcTime(updateEvent.EndDate);

            const newItem: any = {
                Title: updateEvent.title,
                Description: updateEvent.Description,
                Geolocation: updateEvent.geolocation,
                [...]
            };

            if (updateEvent.UID) {
                newItem.UID = updateEvent.UID;
            }
            if (updateEvent.MasterSeriesItemID) {
                newItem.MasterSeriesItemID = updateEvent.MasterSeriesItemID;
            }

            results = await
this._sp.web.lists.getById(listId).items.getById(updateEvent.Id).update(newItem);
        }
        catch (error) {
            return Promise.reject(error);
        }
        return results;
    }
```

- Must delete all recursive extensions before updating any recurrence events (to refresh the recurrence)
- Retrieve the ID of the event to use to update the SP List of events with the saved changes to the attributes of the event.
- Exception handling: reject any exception found while doing asynchronous calls for the SP list of events

- Delete event

Delete event

```
private async confirmDelete(ev: React.MouseEvent<HTMLDivElement>): Promise<void>
{
   ev.preventDefault();
   try {
     this.setState({ isDeleting: true });

     switch (this.props.panelMode) {
       case IPanelModelEnum.edit:
         await UserEventService.deleteEvent(this.state.eventData,
this.props.siteUrl, this.props.listId, this.state.recurrenceSeriesEdited);
         break;
       default:
         break;
     }
     this.setState({ isDeleting: false });
     this.props.onDismissPanel(true);
   } catch (error) {
     this.setState({ hasError: true, errorMessage: error.message, isDeleting:
false, displayDialog: false });
   }
 }
```

- Confirmation to deletion of the selected event: also check that the panel mode is on edit mode. Call the service to delete the event by passing the selected event and its details as well as other relevant information such as the Sharepoint site URL, the list ID and if there are any recurrences in this event.
- Exception handling: to handle errors where the asynchronous call to delete the event failed

Delete event: UserEventService

```
public async deleteEvent(event: IEventData, siteUrl: string, listId: string,
recurrenceSeriesEdited: boolean): Promise<any> {
       let results = null;
       try {
           // Exception Recurrence eventtype = 4 ?  update to deleted Recurrence
eventtype=3
           switch (event.EventType.toString()) {
               case '4': // Exception Recurrence Event
                   results = await
this._sp.web.lists.getById(listId).items.getById(event.Id).update({
                       Title: `Deleted: ${event.title}`,
                       EventType: '3',
```

```
                    });

                    break;
                case '1': // recurrence Event
                    // if  delete is a main recurrence delete all recurrences and
main recurrence
                    if (recurrenceSeriesEdited) {
                        // delete exceptions if exists before delete recurrence
event
                        await this.deleteRecurrenceExceptions(event, siteUrl,
listId);
                        await
this._sp.web.lists.getById(listId).items.getById(event.Id).delete();
                    } else {
                        event.title = `Deleted: ${event.title}`;
                        [...]
                        event.fRecurrence = true;
                        event.EventType = '3';
                        await this.addEvent(event, siteUrl, listId);
                    }

                    break;
                case '0': // normal Event
                    await
this._sp.web.lists.getById(listId).items.getById(event.Id).delete();
                    break;
            }
        } catch (error) {
            return Promise.reject(error);
        }
        return;
    }
```

- Handling the different cases of events
  - Case of the event being an exception recurrence event:
    - Update the event's title instead as 'deleted'
  - Case of the event being a recurrence event
    - Delete all recurrences and main recurrence. Delete the exceptions, if any exist, after deleting recurrence event.
  - Case of the event being 'normal'
    - Simply delete the event from the SP list of events (using its ID)
- Exception handling: reject any exception found while doing asynchronous calls for the SP list of events

Version Management

The chosen source code management platform for version management was GitLabs. GitLabs is suitable for the collaboration of teams on software development projects. It offers the following functionalities:

- **Version control -** As Coding Club[21] states *"version control allows you to keep track of your work and helps you to easily explore the changes you have made, be it data, coding scripts, notes, etc. [...] With version control software such as Git, version control is much smoother and easier to implement. Using an online platform [such as **GitLabs**] to store your files means that you have an online backup of your work, which is beneficial for both you and your collaborators."*
- **Issue tracking -** GitLab[22] states that it uses the advanced issue tracker tool *"for collaboratively developing ideas, solving problems, and planning work."*. The common use cases of issues include: *"Discussing the implementation of a new idea; tracking tasks and work status; accepting feature proposals, questions, support requests, or bug reports; and elaborating on new code implementations"*
- **Merge conflicts -** merge conflicts occur when two branches in a merge request each have different changes and the team must communicate and decide on which change to accept. The two versions are compared line by line. In context to GitLabs, PRACE Repository[23] states that the merge commit strategy used is that *"GitLab resolves conflicts by creating a merge commit in the source branch, but does not merge it into the target branch. You can then review and test the merge commit. Verify it contains no unintended changes and doesn't break your build"*.
- **Rollback strategy -** Maintaining previous versions of the web part code in GitLab means that developers can simply roll back to a previous version. This is especially in the case of acknowledging issues caused by a newer version of the web part that require the software to be reverted back to a version where it was functional.

GitLabs is essential for teams in terms of versional management as it provides the benefits necessary for the transformation into effective software development. The benefits are mentioned by the GitLab website[24] itself:

- **Collaboration -** *"helps your development team collaborate and maximise productivity, sparking faster delivery and increased visibility"*. This is through the features of reviewing, commenting and improving code as well as file locking to prevent conflicts.
- **Acceleration -** *"asset version control, feedback loops, and powerful branching patterns to help your developers solve problems and ship value"*. Developers are able to work from a

---

[21] *Intro to github for version control*. Available at: https://ourcodingclub.github.io/tutorials/git/ (Accessed: March 29, 2023).
[22] *Issues: GitLab*. Available at: https://software.rcc.uchicago.edu/git/help/user/project/issues/index.md#:~:text=Overview,You%20and%20your%20team. (Accessed: March 29, 2023).
[23] *Merge requests: GitLab*. Available at: https://repository.prace-ri.eu/git/help/user/project/merge_requests/conflicts.md (Accessed: March 29, 2023).
[24] *Source code management* (no date) *GitLab*. Available at: https://about.gitlab.com/stages-devops-lifecycle/source-code-management/ (Accessed: March 29, 2023).

local copy and branch code, make changes and merge into the main branch; this is useful when different developers are assigned tasks to develop different features of the web parts.

- **Compliance and security -** *"enables teams to manage their work with a single source of truth"*. This means that code quality and security can be automatically scanned with each commit. The audit and compliance is also simplified with the access of controls and reporting.

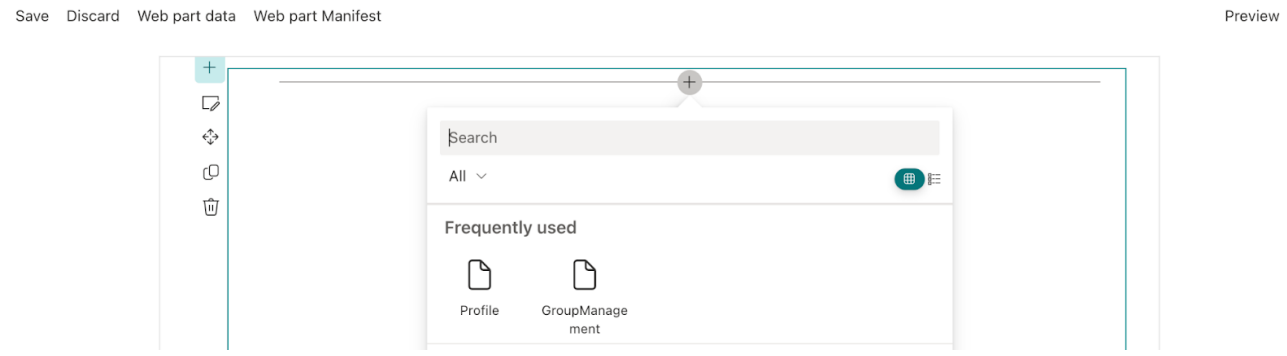
Test Driven Development

The implementation stage of this project required the set-up of the Sharepoint hosted workbench without the need to deploy the solution to Sharepoint. This was set-up by starting up a local web server that the hosted workbench is able to download files from, using the gulp task, 'serve'. The feature of hot-reload (the page automatically refreshes after each change on code) was vital for developers for faster development.

Terminal which displays compiling of the web parts (especially any live errors in the code):

```
[00:38:04] Project samples version:0.0.1
[00:38:04] Build tools version:3.17.19
[00:38:04] Node version:v16.19.0
[00:38:04] Total duration:53 s
[00:38:04] Task warnings:1
[00:38:08] [fast-serve] To load your scripts, use this query string: ?debug=true&noredir=true&debugMa
/temp/manifests.js
ℹ ｢wds｣: Project is running at https://localhost:4321/
ℹ ｢wds｣: webpack output is served from https://localhost:4321/dist/
ℹ ｢wds｣: Content not from webpack is served from /Users/neka/CS4825Repo/spfx-react
ℹ ｢wds｣: 404s will fallback to /index.html
ℹ ｢wdm｣: Time: 29387ms
Entrypoint poll-app-web-part = poll-app-web-part.js poll-app-web-part.js.map
Entrypoint profile-web-part = profile-web-part.js profile-web-part.js.map
Entrypoint feed-web-part = feed-web-part.js feed-web-part.js.map
Entrypoint dashboard-app-web-part = dashboard-app-web-part.js dashboard-app-web-part.js.map
Entrypoint group-management-web-part = group-management-web-part.js group-management-web-part.js.map
Entrypoint poll-management-web-part = poll-management-web-part.js poll-management-web-part.js.map
Entrypoint calendar-web-part = calendar-web-part.js calendar-web-part.js.map
ℹ ｢wdm｣: Compiled successfully.
No issues found.
```

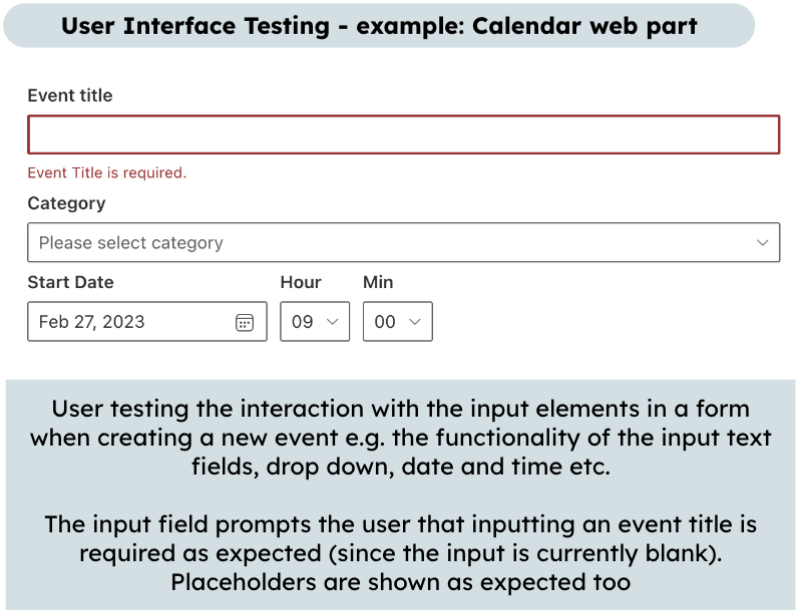
Workbench to preview web parts:



The testing for development of a web part also used a pyramid structure to break down the type of tests to apply to the components. It is typically broken down into the three layers of the user interface testing, integration (service) testing, and unit testing.

1. **User Interface testing -** tests which consist of the user's interaction with the view of the browser. The React testing Library is used to test the user interface of the web parts. Writing test cases can be used to simulate the user interaction with web parts e.g. clicking a button. This type of testing ensures web parts are responsive and user friendly.

Example of user interface testing in the project:



2. **Integration testing -** tests which include the combination of multiple components. Jonathan Rasmusson[25] further describes integration tests as *"the underlying services that power an application. For web applications, these are our services - programs that run on web servers and respond to HTTP requests"*. It is also mentioned that this testing does not have to *"deal with the fragility of the UI, while still retaining some of the ability to check that things are properly hooked up and connected"*.
    - The output of the console was used to test the SP HTTP requests and responses sent. For example, any asynchronous call which returns a promise object.

[25] Rasmusson, J. (2016). The Way of the Web Tester: A Beginner's Guide to Automating Tests. (n.p.): Pragmatic Bookshelf.

Example of integration testing in the project:

Integration Testing - example: Polls web part

View: from PollManagement component

```
UserPollService.igetPolls().then(polls => {
```

Service: from UserPollService (asynchronous call for the SP list's items)

```
public igetPolls = async (): Promise<any> => {
    /* Get items from polls */
    const polls = await this._polls.items()
    console.log('Polls, ', polls)
```

Output on client side (browser console)

```
Polls,                                                    UserPollService.ts:60
▼Array(3) i
  ▶0: {odata.type: 'SP.Data.PollsListItem', odata.id: 'd834e3ba-0962-4798-b0d0-69039bb317f6', od
  ▶1: {odata.type: 'SP.Data.PollsListItem', odata.id: '2c755dc0-34c7-432c-b7aa-f9483c594369', od
  ▶2: {odata.type: 'SP.Data.PollsListItem', odata.id: '3efffdfb-d0c0-4c53-8fbb-df19b976f206', od
    length: 3
  ▶[[Prototype]]: Array(0)
```

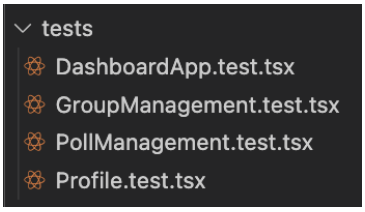
Console of the array of items from the polls SharePoint list. The console log outputs the three items of the list as expected (these were the exact items from the SharePoint list, 'Polls')

'Polls' - from SharePoint Lists

Polls ☆

Title	question	options
p-001-test	What approach should be?	A,B,C,D
p-002-test	What is the best option to take?	W,X,Y,Z
p-003-test	What framework are you most experienced with?	React,Vue,Angular,Other

3. **Unit testing -** tests which consist of individual components of a web part. For example, key methods of a particular web part. Unit tests are written using the dependencies of Jest.



Example of unit testing in the project:

Unit testing Testing - example: Group Management web part

Example of unit test e.g. validating the input details of a group (name must be at least 3 characters)

```
[...]
it('Validating details - validating name', () => {
      [...]

      /* Valid tests: must be at least 3 characters
*/
      groupToTest.name = 'TN001'
      expect(validateDetails(groupToTest)).toBe(true)
      [...]

      /* Invalid tests */
      groupToTest.name = 'T'
     expect(validateDetails(groupToTest)).toBe(false)
      [...]
   });

[...]
```

Snippet of terminal output after running 'npm test'

```
PASS src/tests/GroupManagement.test.tsx (15.298 s)
  ● Console

    console.log
      Invalid name:  T  - not at least 3 characters

      at validateDetails (src/tests/GroupManagement.test.tsx:11:19)
```

```
Test Suites: 5 passed, 5 total
Tests:       11 passed, 11 total
Snapshots:   0 total
Time:        16.026 s
```

Assessment and Reflection

Assessment of team:

The risk of team re-arrangement was very detrimental to the cost of time in the project. The circumstance of a team member being lost within the process, leaving only one developer in the team, led to the immediate mitigation action of re-allocation of tasks and an emphasised focus on the core requirements of the project. The allocation of tasks was appropriate, given the unforeseen circumstance, as it still achieved the core functionalities of the project. The completion of the following web parts: the dashboard, the profile, group management, poll management and the calendar; was sufficient thus achieving the most of the core user stories.

The overall communication of the team was a massive strength. The meetings were smooth and consistent and the team was able to thoroughly track progress and quickly resolve any issues encountered throughout the implementation. The feedback from the presenting progress in the web parts was clear and direct and the team was always provided with clarity with any questions asked about the project.

The main challenge was the learning curve for the project, for the developers of the project. This is because the developers on the team were stepping into a project of new technologies. The main technologies with heavy learning time were the SPFx framework and React Typescript framework. Thus, the progress for the implementation was initially very slow with many troubleshooting errors and required extensive research and tutorials from React regarding implementing web parts. Once the learning was achieved, the implementation was finally speeded up.

Tasks unable to complete:

- **Feed web part -** it was initially planned to have a web part where users within a space could create new posts or interact with them. The functionality of the web part was unable to be completed. However, this was overshadowed by the importance of other web parts thus the web part not being implemented was not detrimental to the project.
- **Notifications and virtual meetings -** the prioritisation of tasks led to avoiding the completion of these tasks. However, given the optional priority of these tasks, for being trivial features to the system, it was not a massive loss to the overall functionalities of the project.
- **Significant bug fixing -** there was a massive time constraint on the testing phase of the web parts thus, there is a high possibility of bugs in the implemented web parts.

Possible risks or risks found in the progress:

Risk	Cause of risk	Probability	Impact	Mitigating action
The risk in failure of the set-up of the technologies	Lack of knowledge of tool	**Moderate**	**High**, can prevent implementation progress of the project	Ensure to have thorough understanding of technologies before initialising progress on them.
The progress in the development phase is behind what was planned in the Gantt chart (can be found in Appendices).	Overestimation of the amount of tasks and requirements that can be completed by a specific deadline	**Moderate**	**Moderate**	Re-evaluate the requirements that are going to be core to the functionality of the system and prioritise on working on those tasks (without the need of changing the overall project duration)
Team re-arrangements; losing team members in the process of the project	No particular cause that can be controlled; unforeseen circumstances.	**Low**	**High**, can affect overall project progress (if workload is not changed; too much workload for too little number of team members)	Re-allocation of tasks in team project; focus on core requirements of the project to make best use of progress within the given time frame

Improvements and reflection:

- The selection of tools and technologies should be based on the expertise of the developers in the team. Despite the benefit of acquiring new skills with new technologies, the learning curve can be a time constraint that is detrimental to the time cost and schedule of the project. The gantt chart should have considered additional time to accommodate for the learning time that developers went through with new technologies.
- Further research into the tool before the implementation phase: it was later found out by the team that the planning of the features of the project would have benefited from knowing further details of the extensive features SharePoint has to offer. This is because some of the web parts or other features planned did not need implementation as SharePoint already offers it (e.g. there is already a web part for handling documents), thus affecting the user stories and overall timetable of tasks.
- Areas of improvement in testing: the time constraints reduced the planned time used for unit testing. Despite the existence of unit tests for the implemented web parts, there is still a possibility of bugs that would have been discovered given more time. In future projects,

importance should be placed on allocating more time for more thorough unit testing under Test Driven development (TDD).

- Alternative plans in deployment - there was insufficient discussion on the deployment of web parts to accommodate for different circumstances that could arise from the project, thus the web parts were unable to be deployed properly into the web site. This resulted in an alternate plan of developers requiring to sign up to Microsoft and create their own tenant with provided dummy data (as a way to claim administration access to upload web parts).

Overall, the team is very satisfied with what was accomplished with this project. With such detrimental factors of reduced number of team members and the learning curve of new technologies, the web parts implemented within the timeline was sufficient. The developers of the team earned the benefit of learning a new technology, i.e. React, which can be applied to further projects of web development. The team also learned the experience and importance of implementation that integrates into a platform which already provides some functionalities. This is a situation that is common to developers in the industry; it is often rare that the developer has to start completely from scratch with a problem.

Summary of the diary (full project diary in appendix) -

- **Meetings -** Weekly (or bi-weekly) 30-minute meetings held with the supervisors within the team to check on progress from developers of the planning and development phases of the system.
- **Tool recommendation -** the criteria for the non-functional requirements of the different approaches to the project. The final decision was Sharepoint which is a tool which must be immediately explored to speed up implementation progress.
- **User stories and use case analysis -** user stories derived from the user requirements. Requirements allocated by role of the user (user, representative, group admin or admin) and set by priority. The number of core requirements could be subject to change depending on whether implementation is behind schedule or not.
- **Sitemap & Wireframe -** the design phase included a sitemap.
    - **Mockup designs -** UI designs of the different pages of the system demonstrated to supervisors (with advice and criticism). The UI design seems achievable and looks suitable for the user. The accessibility of the UI design was also considered (e.g. colour contrasts, use of icons, memorisation).
- **SPFx Framework set-up -** the implementation phase finally begins.
- **Implementation of UI designs -** the implementation of UI design for the web parts without the functionalities.
- **Implementation of web parts -** the implementation of the components planned (i.e. Dashboard, Calendar, Group management, Poll management and Profile)
- **Test driven development (TDD) -** the testing phase within implementation, includes unit testing, integration testing and UI testing. The unit testing was specifically for key functions in the components in an isolated environment (without the use of a SharePoint server).
- **Documentation of the web parts -** the polishing of the documentation of the web parts.

Bibliography

[1] - L., Holcombe.W.M. (2008) *Running an agile software development project*. Oxford: Wiley-Blackwell.

[2] - Atlassian *Confluence - Pricing | Atlassian*. [Online] Atlassian. Available from https://www.atlassian.com/software/confluence/pricing.

[3] - Frameworkit.com. 2022. *Pros and Cons of SharePoint 2020: Is SharePoint Worth it?*. [online] Available at: <https://www.frameworkit.com/blog/pros-and-cons-of-sharepoint> [Accessed 17 October 2022].

[4] - Ungoti. 2022. *You must Know these Pros and Cons of SharePoint Intranet*. [online] Available at: <https://ungoti.com/blog/sharepoint-intranet-pros-and-cons/> [Accessed 17 October 2022].

[5] - Withee, R., Withee, K. (2021). SharePoint For Dummies. United Kingdom: Wiley.

[6] - Sharepoint 2017: An Easy Guide to the Best Features. (2017). (n.p.): First Rank Publishing.

[7] - Guilmette, A. (2022). Workflow Automation with Microsoft Power Automate: Use Business Process Automation to Achieve Digital Transformation with Minimal Code. United Kingdom: Packt Publishing.

[8] - Bray, S., Wood, M., Curran, P. (2013). Microsoft SharePoint 2013 Designing and Architecting Solutions. United States: Pearson Education.

[9] - Jain, V. (2020). Getting Started with SharePoint Framework (SPFx): Design and Build Engaging Intelligent Applications Using SharePoint Framework. India: BPB PUBN.

[10] - Nachan, N. (2019). Mastering Sharepoint Framework: Master the SharePoint Framework Development with Easy-to-Follow Examples. India: BPB PUBN.

[11] - Mardan, A. (2017). React Quickly: Painless Web Apps with React, JSX, Redux, and GraphQL. United States: Manning.

[12] - Medero, J., Fox, B., Prussak, P., Mehta, N., Poelmans, J., Buechler, C. M., Pragash, C., Lotter, M., Regan, C. J., Pyles, J., Gordon, M. (2008). SharePoint 2007: The Definitive Guide. United States: O'Reilly Media.

[13] - Alirezaei, R., Baer, B., Wilson, B., Kearn, M. (2012). SharePoint 2010 Enterprise Architect's Guidebook. Germany: Wiley.

[14] - GitHub. (n.d.). sp-dev-fx-web parts/samples at main · pnp/sp-dev-fx-web parts. [online] Available at: https://github.com/pnp/sp-dev-fx-web parts/tree/main/samples [Accessed 30 Mar. 2023].

[15] - Microsoft Legal. (n.d.). Copyrights. [online] Available at: https://www.microsoft.com/en-us/legal/intellectualproperty/copyright/ [Accessed 30 Mar. 2023].

[16] - Wright, S., Bishop, D., Malik, S., McDermott, M., Rais, R. b., Milner, D., Krause, J., LLC, W., Eddinger, M., Sethunarayanan, K., Orange, M., Naveed, T., Bakmand-Mikalski, D., Musters, E., Farnhill, B., Ralston, B., Richard, E., Hild, E., Loriot, C. R., Ortiz, D. (2011). Expert SharePoint 2010 Practices. Netherlands: Apress.

[17] - Osmani, A. (2012). Learning JavaScript Design Patterns. United States: O'Reilly Media.

[18] - Rippon, C. (2021). ASP.NET Core 5 and React: Full-stack Web Development Using .NET 5, React 17, and TypeScript 4, 2nd Edition. United Kingdom: Packt Publishing.

[19] - *Container/Presentational Pattern*. [Online] Container/Presentational Pattern. Available from : https://www.patterns.dev/posts/presentational-container-pattern.

[20] - Patel, D. (2019) An introduction to MVC architecture: A web developer's point of view - dzone, dzone.com. DZone. Available at: https://dzone.com/articles/introduction-to-mvc-architecture-web-developer-poi (Accessed: March 29, 2023).

[21] - Intro to github for version control. Available at: https://ourcodingclub.github.io/tutorials/git/ (Accessed: March 29, 2023).

[22] - Issues: GitLab. Available at: https://software.rcc.uchicago.edu/git/help/user/project/issues/index.md#:~:text=Overview,You%20and%20your%20team. (Accessed: March 29, 2023).

[23] - Merge requests: GitLab. Available at: https://repository.prace-ri.eu/git/help/user/project/merge_requests/conflicts.md (Accessed: March 29, 2023).

[24] - Source code management (no date) GitLab. Available at: https://about.gitlab.com/stages-devops-lifecycle/source-code-management/ (Accessed: March 29, 2023).

[25] - Rasmusson, J. (2016). The Way of the Web Tester: A Beginner's Guide to Automating Tests. (n.p.): Pragmatic Bookshelf.

Appendices

Diary

Meetings

Date	Progress update	Takeaways
10/10/22	● Introduction to the project and discussion about the report and booklet. Explanation of what the project is about. ● Meeting duration: 30 mins	● Examining and comparing the options available for the project (or even find any other platforms if possible) ● Criteria to compare the option (the functional criteria) ● Initial steps on the interim deliverable - the report outlining the problems and describes and compares the options and final recommendations
17/10/22	● Clarification of the project; requirements and priorities which can be translated into user stories. ● Meeting duration: 30 mins	● Presentation of comparison and final recommendation
24/10/22	● Presentation which concludes final decision on Sharepoint as the tool to use ● Discussion on separate portal for students and staff and searching for Sharepoint environment for the implementation to occur ● Feedback of tool comparison: include measure for the costs of the tools. ● Meeting duration: 30 mins	● Check for development environment for Sharepoint ● Start sketches of the UI ● Completion of user stories
14/11/22	● Still pending progress on a local instance of a Sharepoint environment ● Change of project strategy due to the circumstances of a team member dropping out. ● Meeting duration: 30 mins	● Show presentation of UI-mockups ● Take inspiration from how the tools are used and examples from the latest versions of SharePoint ● Look at handbook to think about the technical aspects of the deliverable (the report)
22/11/22	● Presentation of the sitemap	● Feedback of designing a page (i.e.

	of the project and the UI designs ● Feedback given about UI designs ● Meeting duration: 30 mins	spaces) to give users the ability to post something and comment (e.g. LinkedIn) ● Tag system for polls ● The creation of interest groups (e.g. machine learning, AI) for the user.
28/11/22	● Feedback from improved UI design ○ Feed - remove unnecessary features such as follow and reposting videos/images. ● Meeting duration: 30 mins	● UI design adjustments based on criticism (e.g. removal of unnecessary features such as the follow feature, images/videos on feed) ● Begin interactions and tests with Sharepoint environment (playground; before starting proper implementation) ○ Focus on core requirements ● Consistent commits on repository
20/01/23	● Discussion of the feedback - possibility of testers for the system ● Further adjustment of plans ● Plan for demo for later ● Reminder of documentation and user manual (update as you go)	● Initial start on implementation of the first web part of the system
24/02/23	● Demo of the UI designs of web parts with placeholder data ● Initial start at functional of CRUD operations and use of SharePoint lists ● Questions about what is classed as sufficient implementation of the project ● Change of focus on core features	● Present deployed web parts on SharePoint site to then upload into the site.
10/03/23	● Discussion about the report - question about the professional issues ● Update on deployment - unable to deploy web parts due to requiring admin access as a RHUL SharePoint user; discussion of an alternate plan if web parts are unable to be deployed to	● Check out Microsoft Sharepoint - sign up as a developer and play around with provided tenant and administration access ○ Includes dummy data of fake users

	site	

Progress

Date	Progress update	Plan for later
6/10/22	● Initial progress on interim report - gathering requirements ● Emailing advisors for clarity	● Receive clarity of the project from advisors to understand projects
8/10/22	● Written introduction of the report: the problem, the brief of the proposed solution and any other context relevant ● Christopher and Neka organising weekly meetups in Bedford to discuss and continue progress; task allocation of doing different aspects of the requirements section.	● Finalise understanding of the project's context and background
10/10/22	● The different phases of the project discussed ● The phases expressed as a timeline of Waterfall methodology: requirements, analysis, design, implementation, deployment (Gantt chart)	● Follow the waterfall model in linear order as decided. The next phase depends on the previous phase
12/10/22	● Email received; gathering user requirements; Christopher and Neka assigned two roles each to do user requirements on. ● Research on user requirements with the help of book on Agile Development [1]	● Finalise user requirements
17/10/22	● Translating the user requirements into user stories	● Finalise user stories
20/10/22	● Use case analysis diagram to	● Finalise use case analysis

	finalise requirements analysis	
24/10/22	• Requirements phase completed. Tool comparison begins - Christopher researches and writes about pros and cons of Slack and Confluence • Neka researches and writes about pros and cons of Sharepoint and Custom Development	• Finalise tool comparison • Presentation to show recommended tool
19/11/12	• Final recommendation for the tool: Sharepoint • Thus request for environment	• Researching Sharepoint (i.e. examples, tutorials) • Waiting for testing/development environment to begin implementation

20/11/22	• Content phase completed
	• The design phase begins - UI ideas being brainstorms
	• Dilemma of Christopher leaving the group
	  ○ Drop of communication
	  ○ Change of workload adjustments for Neka
		• Team dilemma communicated to advisors to re-adjust the timeline plan

21/11/22	• Team dilemma resolved by re-adjustment of project structure • Neka completes and presents sitemap and UI designs	• Present Ui designs and improve it based on review from advisors
28/11/22	• Neka improves UI designs based on advisors' reviews • UI design within scope of requirements	• Present improved version of UI designs to advisors
8/12/22	• Design phase completed • Working on Interim deliverable - report and presentation • No implementation done for repository yet due to time constraints from the unforeseen circumstances already mentioned before.	• Completion of Interim deliverable • Implementation phase begins for Term 2

19/12/22	• Commits on initial set-up for the implementation	• Research into SPFx framework (how to set it up)
26/12/22	• Initial SPFx framework set-up: testing, playing around (HelloWorld webpart example from tutorial)	• Attempt to set-up using SPFx and Vue technologies integrated together.
31/12/22	• Restarted framework set-up for framework using Vue and SPFx framework	• Research tutorials regarding Vue and SPFx set-up.
2/01/23	• Fixing errors regarding complications with the git submodules	• Diagnose the problem and restart the project and install the appropriate dependencies.
16/01/23	• Alternative plan for the set-up (*VueJS failed)* • SPFx framework integrated with React instead	• Research and tutorials on ReactJS and the set-up of SPFx framework with React.
20/01/23	• Restarted solution of initial SPFx react framework. • Playground of ReactJS controls e.g. accordion and placeholders (from tutorials)	• Integrate tailwind CSS for easier styling of pages.
23/01/23	• Tailwind CSS included into SPFx framework build	• Playground of the first web part of the system including set-up, error-fixing
31/01/23	• Playground of SPFx framework with the included tailwind CSS	• Playground with Microsoft web part samples (from GitHub)
03/02/23	• Webpart for the feed; error-fixing, playground of reactJS “News SPFx sample”	• Add another web part, of choice, from the system i.e. Groups • Research into Sharepoint Lists (an understanding on how to handle data in CRUD operations)
05/02/23	• Addition of Groups web part for handling groups and members of the system	• Focus on the HTML design of the UI (the functionality can come later) • Add more web parts i.e. calendar and poll web part, to work on
07/02/23	• Initial progress on Calendar and poll web parts (without the functionality aspects)	• Consider implementing design on the profile section
09/02/23	• Implementation of the design for the profile section	• Continue the UI designs for other aspects of the system

	(without the functionality)	
12/02/23	● Initial UI designs on the web part for the polls component	● Move onto the UI design for the dashboard
16/02/23	● Initial UI designs on the web part for the dashboard component	● Move onto the UI design for the feed webpart
20/02/23	● Initial UI designs on the web part for the feeds component	● Begin functionalities with any of the designed web parts (with the use of CRUD operations). ● Create Sharepoint Lists along with dummy data.
22/02/23	● Addition of CSVs for importing Sharepoint Lists; and also retrieving user profile properties with Profile webpart	● Continue implementation of CRUD operations using Sharepoint Lists, beginning with dashboard webpart
23/02/23	● Initial progress on CRUD operations on web parts using Sharepoint lists: using Dashboard	● Functionality on the READ operation of Sharepoint lists; retrieving the upcoming and recent events to render as overview on dashboard
24/02/23	● Able to display the data retrieved from Sharepoint lists and render (dummy data was used).	● Move onto the web part of user memberships; checking user groups and group members
25/02/23	● Initial progress on web part of group management; a lot of error fixing; token currently fails	● Investigate the problem with the token and any other required labels of access (i.e. Microsoft Azure Active Directory - AAD)
28/02/23	● Group management web part: able to create new groups with owners and members	● Find alternative solution to retrieve data on users (suggestion: SP HTTP Client); as Microsoft Graph API requires admin rights into the tenant's AAD.
01/03/23	● Progress on the services for group management; feature to retrieve user profile properties of current user and other users	● Work on functionality to view public/private groups AND join/leave public groups
02/03/23	● Implementation on functionality for user to view public/private groups and join/leave public groups	● Work on functionality to delete a group and to display/filter member and non-member groups

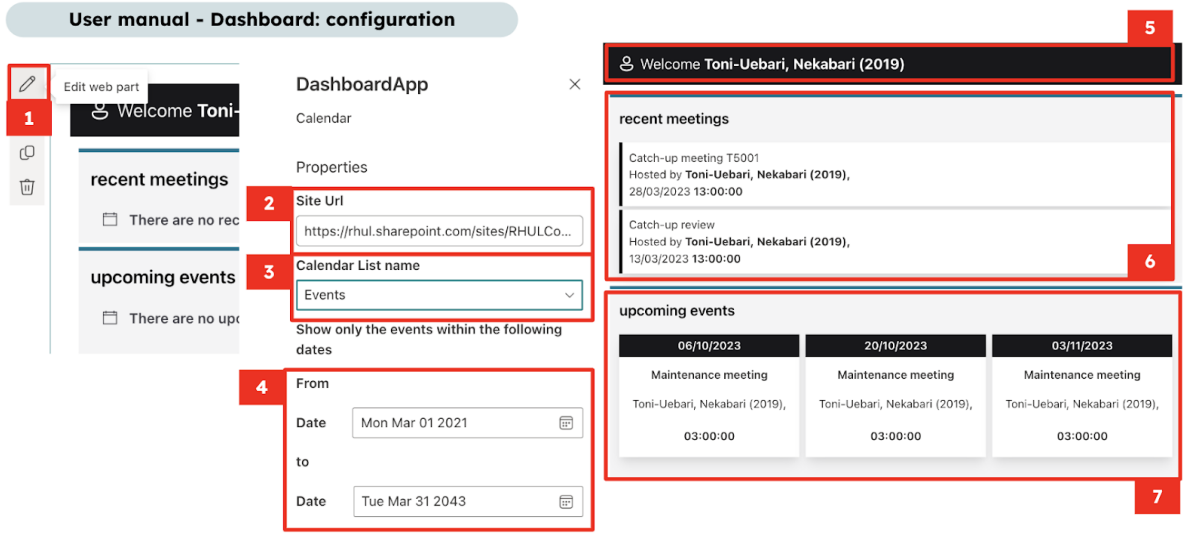
	(implementation of the service used to update SP Lists)	
03/03/23	● Completion of dialog box to confirm deletion of a group	● Work on functionality to edit/manage details of an existing group a user owns.
04/03/23	● Initial progress on edit/view mode of a group to manage	● Fix minor rendering and service issues with the requests to update group details (issues with batch execution)
07/03/23	● Completion of editing group details; minor error fixes on service calls	● Begin implementation of the profile component
08/03/23	● Completion of the profile component - renders chosen user; their groups and functionality to suggest new profiles to check out	● Minor fixes of bugs in the rendering ● Begin process on the poll management web part
09/03/23	● Initial progress on poll management web part configuration to create, schedule and view polls	● Fix major bugs of the components; the components and the service are not fully integrated yet.
14/03/23	● Functionality to successfully submit a poll and render its chart results ● Poll management web part functionalities to bind and render props and state; Options container displays successfully.	● Complete functionalities on the configuration of the web part - including the management of polls
15/03/23	● Completion of functionality to create, edit and delete polls using the poll web part configuration	● Implementation of the web part - calendar
20/03/23	● Initial progress of the calendar web part	● Major error fixes required - components are not fully integrated with the services yet.
21/03/23	● Error fixes on calendar web part: configuration able to function	● Polish unit testing of the existing web parts
26/03/23	● Unit testing of web parts, isolated key functions:	● Polish documentation of the web parts

	including group management, poll management and dashboard	
27/03/23	● Polished documentation of the web parts; including group management, poll management, dashboard, profile and calendar	● Polish dashboard web part
29/03/23	● Polish the UI design of the web parts - e.g. consistency in themes, icons and colours	● Polish the documentation of the repository (including the README)

User Manual

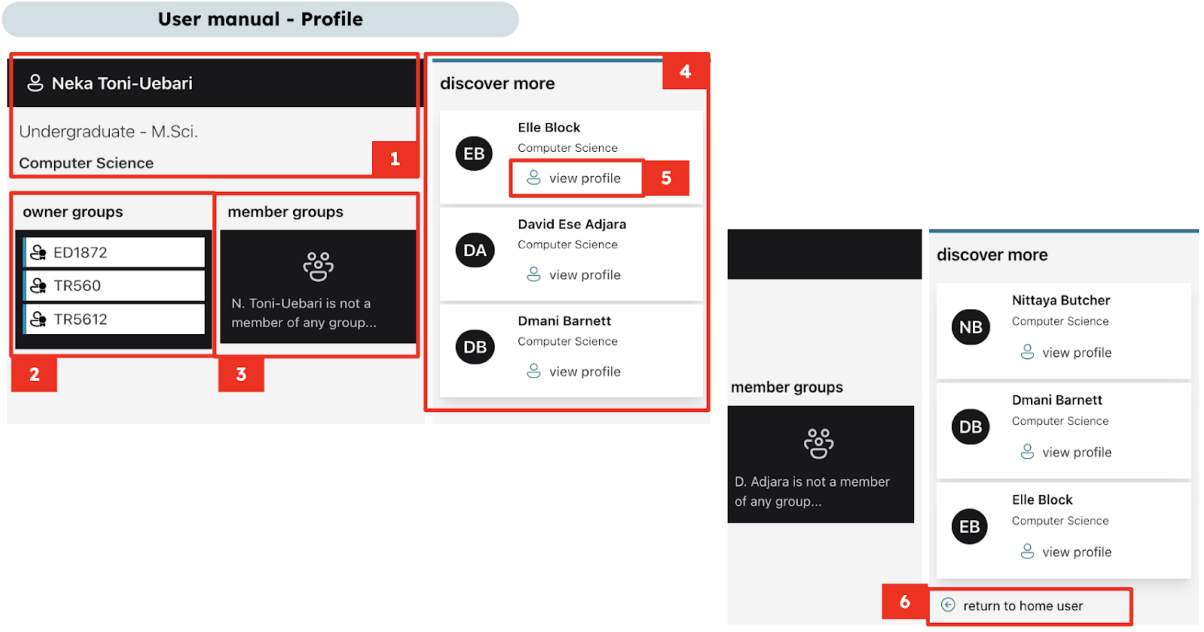
Videos/GIFs for the running application can be found in the repository

Dashboard



1. Open web part configuration panel
2. Set the site URL
3. Set the SharePoint ‘Event’ list
4. Set the date range to look for events between
5. Welcomes the user
6. Display of the recent meetings
7. Display of the upcoming events

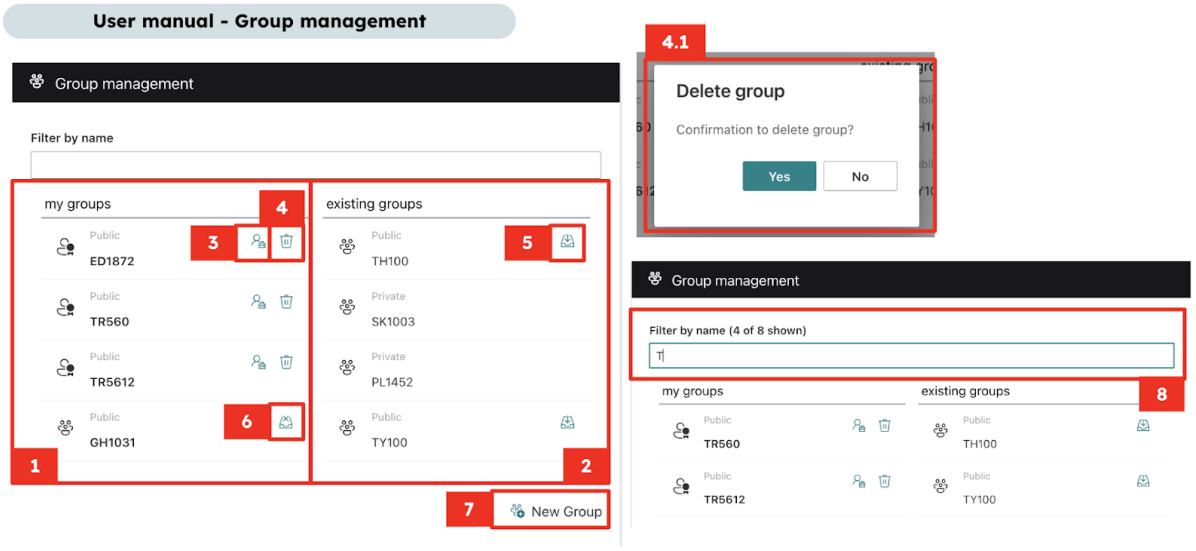
Profile



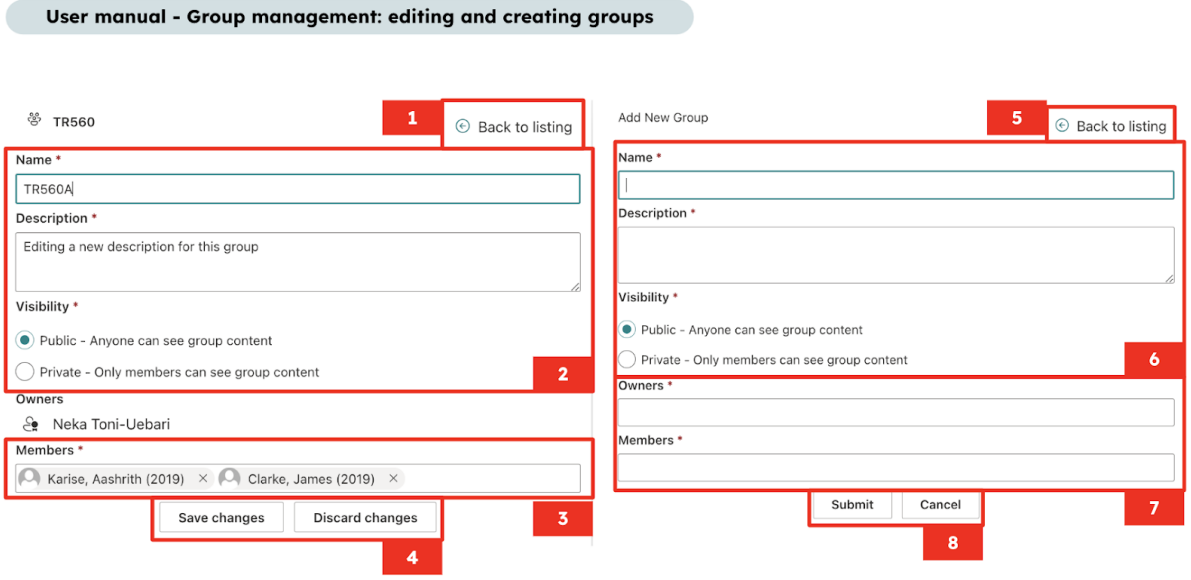
1. Display of the user’s name, title and department
2. Display of the groups the user owns
3. Display of the groups the user is a member of

4. The suggested users to view profiles
5. Opens profile of selected user
6. (when the profile is not the login user) - Returns back to the profile of login user

Groups



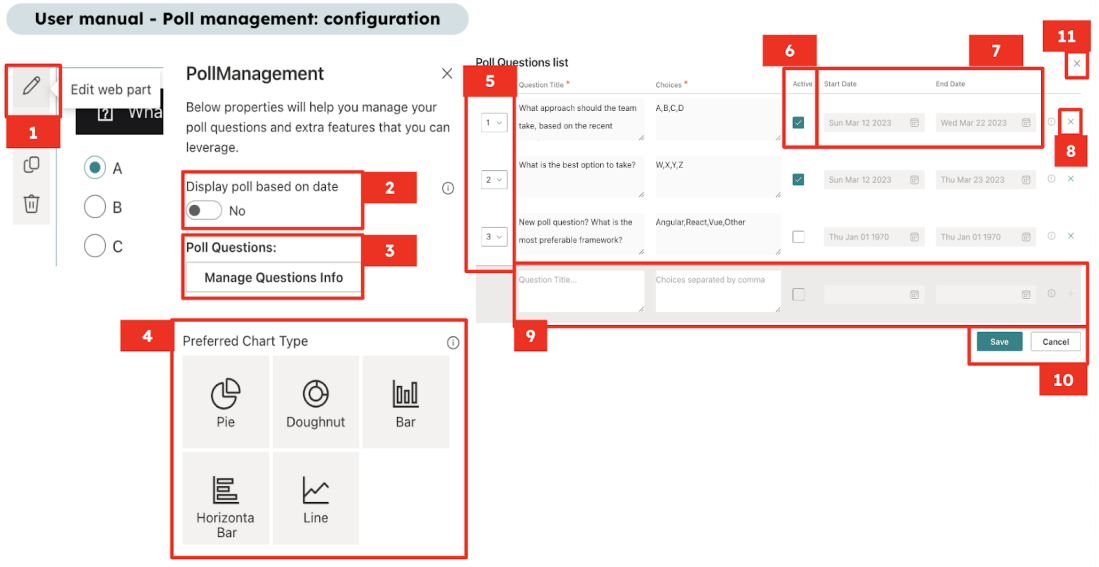
1. Display of the groups the user owns
2. Display of the groups the user is a member of
3. Manage a group that the user owns
4. Delete a group
    - 4.1 - confirm to delete group
5. Join public group
6. Leave public group
7. Opens form to create a new group
8. Filter the groups by name



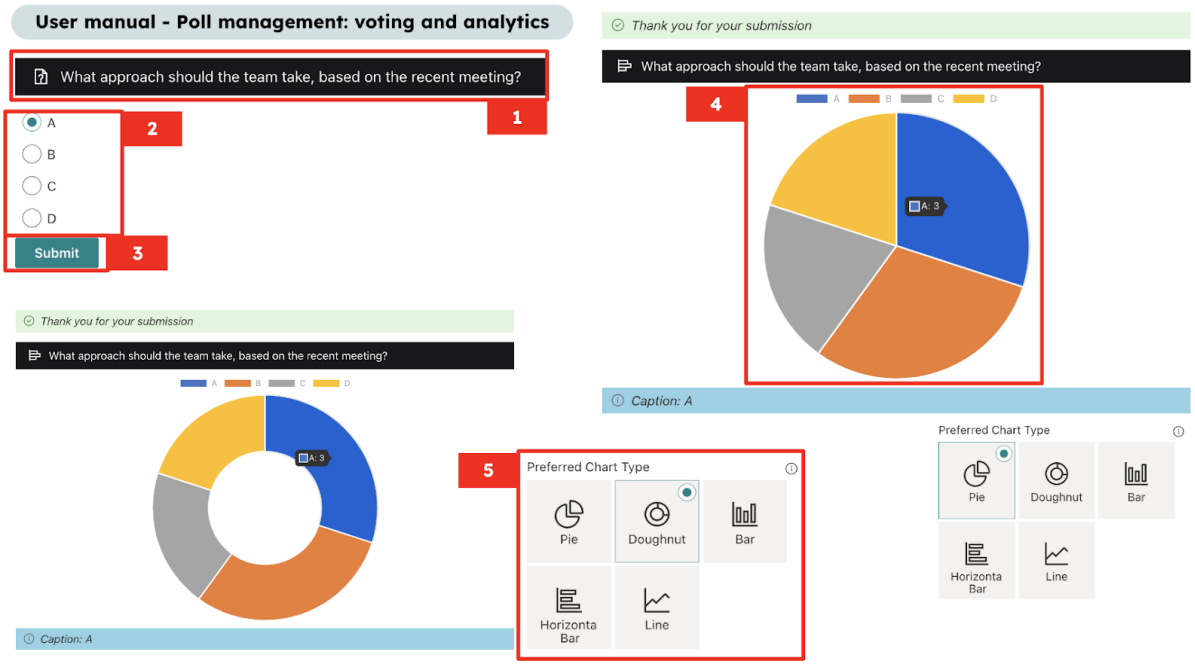
1. Exit edit mode and return back to listing of groups
2. Edit details of existing group
3. Add or remove members of existing group

4. Save or discard changes
5. Exit add mode and return back to listing of groups
6. Enter details for new group
7. Add owners and members of new group
8. Submit or cancel form to create new group

Polls

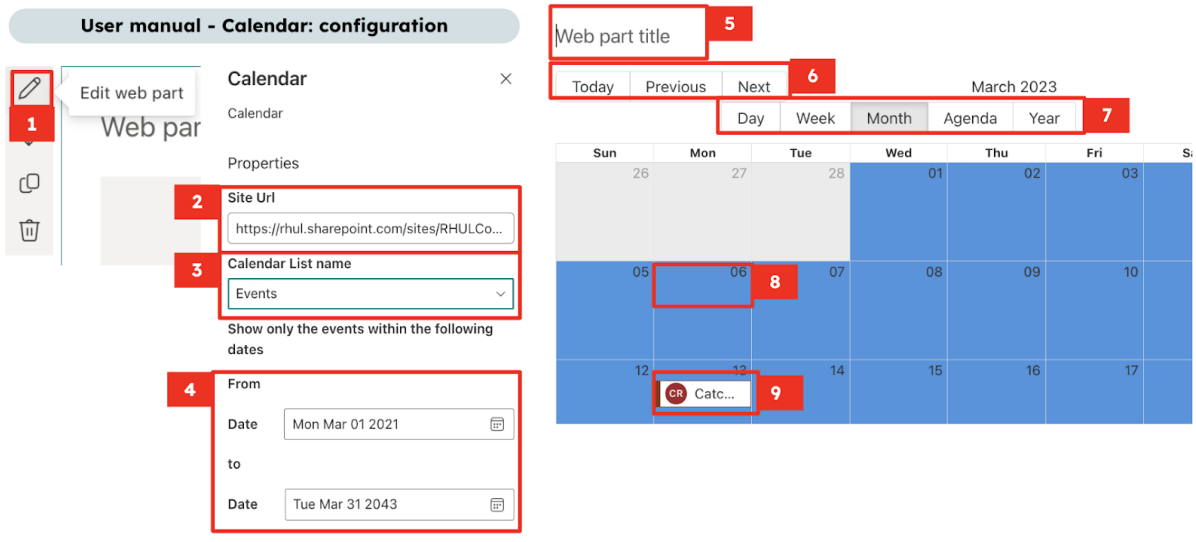


1. Open web part configuration panel
2. Enable/disable checkbox to display the polls based on date
3. Open panel to manage polls
4. Select the preferred chart type to display poll results
5. Sort the order of polls to display
6. Enable/disable checkbox on whether the poll is active or not
7. (If poll based on date is enabled) Set the date range that the poll will be displayed.
8. Remove poll
9. Add new poll
10. Save or discard changes
11. Exit web part configuration panel



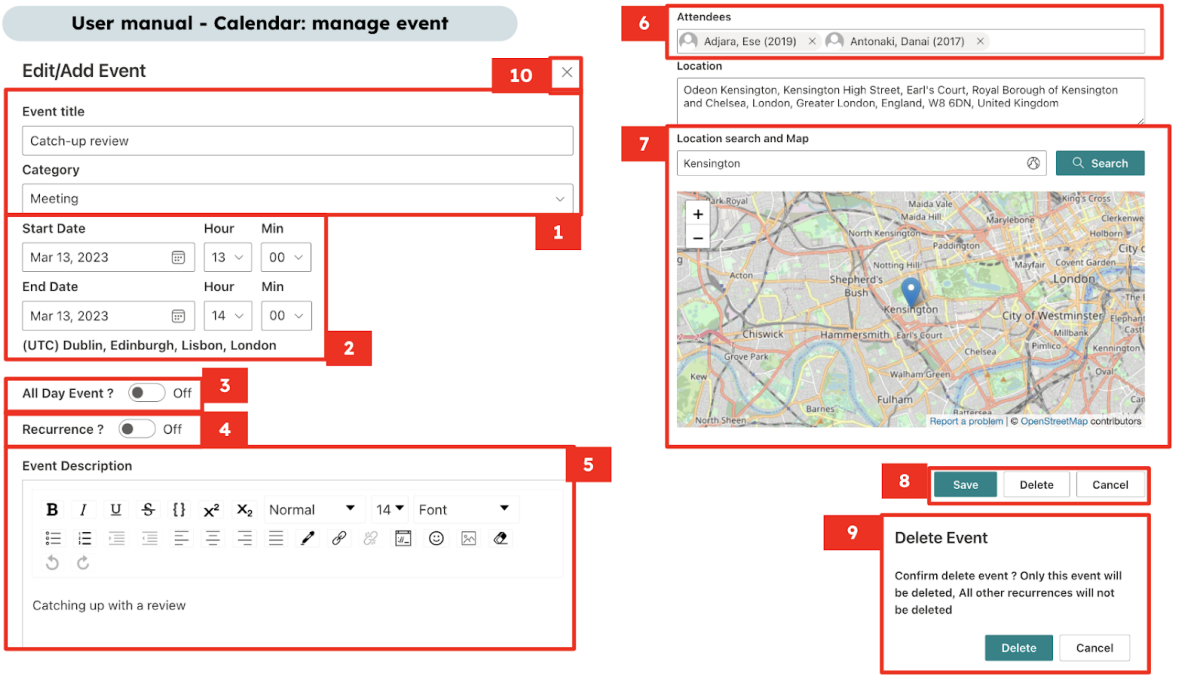
1. Display the question of poll
2. Select option of poll
3. Submit the poll option
4. (Preferred chart type is “Pie” by default) Display of chart results
5. (Open web part configuration panel) Change the preferred chart type to display poll results in

Calendar

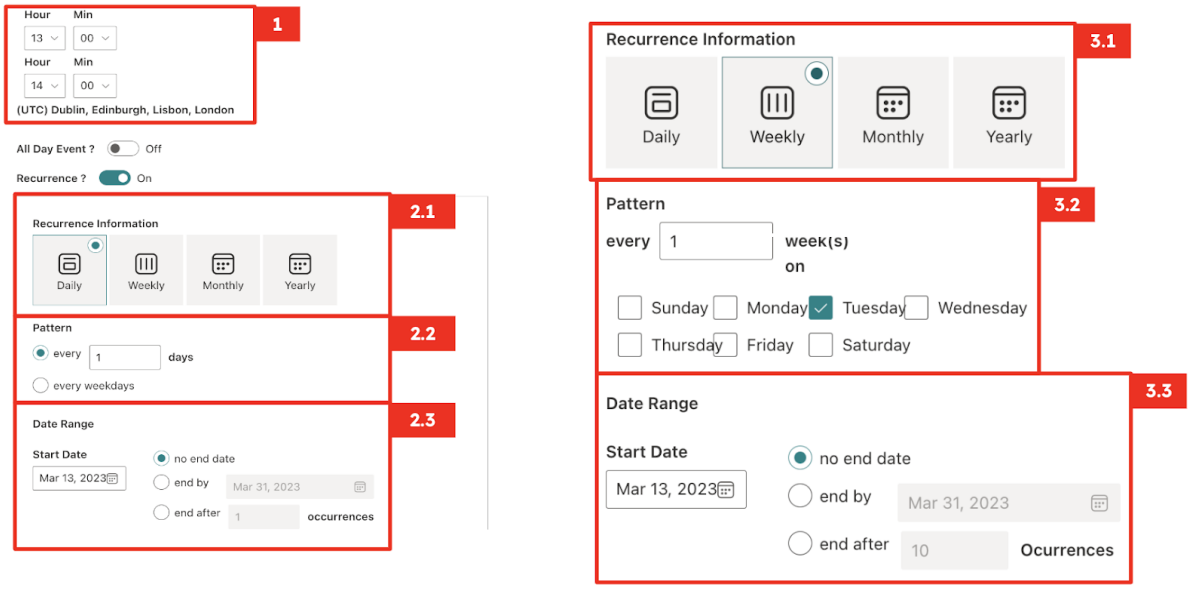


1. Open web part configuration panel
2. Set the site URL
3. Set the SharePoint ‘Event’ list
4. Set the date range to look for events between
5. Edit the web part title
6. Move forwards or backwards in the calendar navigator (or navigate to today’s date)
7. Choose view of calendar (by day, by week, by month, by agenda or by year)
8. Add new event

9. Edit selected event

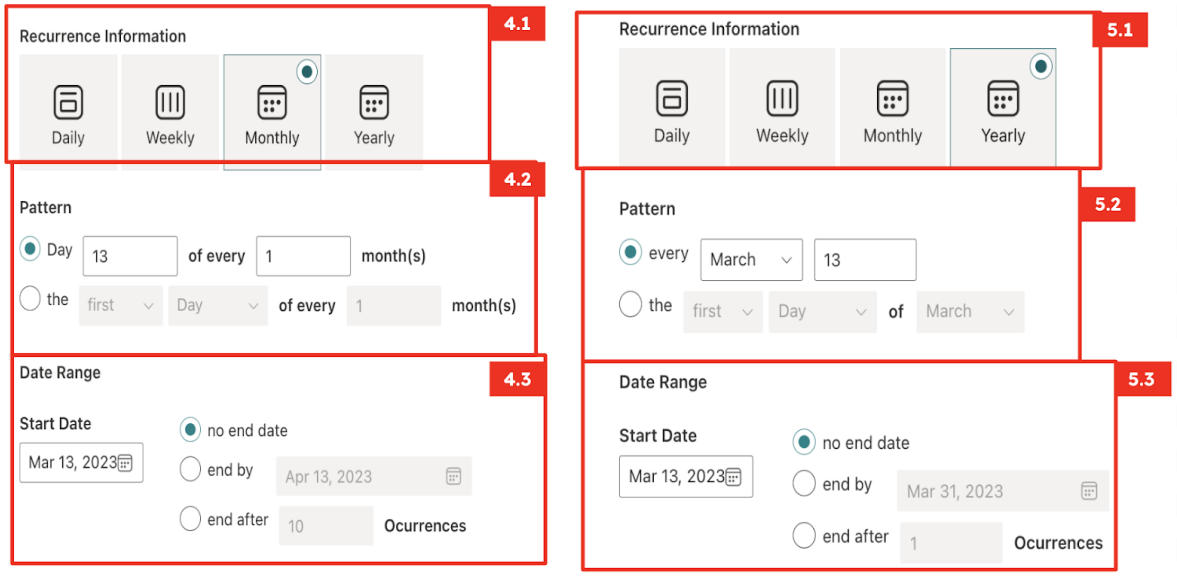


1. Enter event title and select event category (e.g. ‘Meeting’)
2. Enter start and end date of event (only the time is included if the event is ‘all-day’)
3. Enable/disable checkbox of whether the event is all-day
4. Enable/disable checkbox of whether the event should have recurrences
5. Enter the description of the event
6. Select the attendees for the event
7. Enter location to search for the event’s location (uses an external service to find the location on map)
8. Edit mode: save or discard changes or delete event; add mode: add new event or cancel event
9. Confirm to delete existing event



1. Enter the start and end time of recurrence event
2. Recurrence event **daily** - event repeats by days
    - 2.1 - Select 'Daily'
    - 2.2 - Specify pattern - every n days or every weekdays
    - 2.3 - Specify date range - choose start date; choose if there is an end date, or the end date, or the number of occurrences to end the event after
3. Recurrence event **weekly** - event repeats by weeks
    - 3.1 - Select 'Weekly'
    - 3.2 - Specify pattern - every n week or every certain day(s)
    - 3.3 - Specify date range - choose start date; choose if there is an end date, or the end date, or the number of occurrences to end the event after

User manual - Calendar: manage event: recurrence



4. Recurrence event **monthly** - event repeats by months
    - 4.1 - Select 'Monthly'
    - 4.2 - Specify pattern - day m of every n months OR the nth day of every n months

- 4.3 - Specify date range - choose start date; choose if there is an end date, or the end date, or the number of occurrences to end the event after

5. Recurrence event **yearly** - event repeats by years
    - 5.1 - Select 'Yearly'
      5.2 - Specify pattern - every specific date OR the nth day of specific month
    - 5.3 - Specify date range - choose start date; choose if there is an end date, or the end date, or the number of occurrences to end the event after

Installation Manual

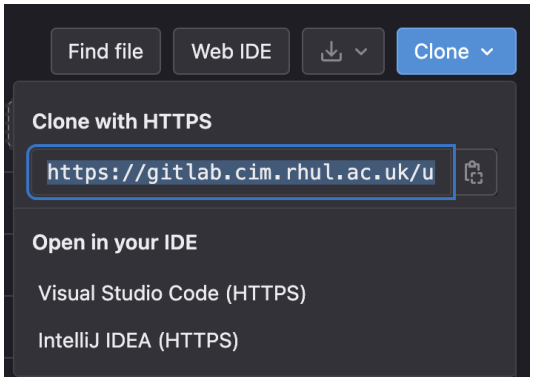
Dependencies:

- Node.js (https://nodejs.org/)
- NPM (https://www.npmjs.com/)
- Yeoman Generator for Microsoft
- Gulp

Installation (as a developer):

The open-source repository used for the development of this solution can be found in GitHub under: https://gitlab.cim.rhul.ac.uk/uyac001/CS4825_202223

- **Clone the repository into a new folder on your local development machine**



- **Terminal commands:**
    - **1.** Run this terminal command to install dependencies of the solution

```
npm install
```

- **Usage:**
    - **1.** Run this terminal command to build the project

```
gulp build
```

    - **2a.** Run this terminal command to run the SharePoint server

```
gulp serve
```

- **2b.** OR Run this terminal command to run the SharePoint server under hot-reload **(recommended for fast development)**

```
npm run serve
```

- **SharePoint lists configuration -** ensure that the 'Site contents' of the SharePoint site has the following SP lists and their attributes (There is a folder of CSVs that can be imported to create the SP lists if needed; the CSVs can also be used for dummy data):
  - Groups - List
    - Title - single line of text
    - displayName - single line of text
    - description - multiple line of text
    - groupType - single line of text
    - mailEnabled - single line of text
    - mailNickname - single line of text
    - securityEnabled - single line of text
    - visibility - single line of text
  - GroupMembers - List
    - Title - single line of text
    - memberEmail - single line of text
  - GroupOwners - List
    - Title - single line of text
    - ownerEmail - single line of text
  - Polls - List
    - Title - single line of text
    - question - single line of text
    - options - single line of text
    - visibility - single line of text
    - startDate - date/time
    - endDate - date/time
    - ownerEmail - single line of text
  - PollResponses - List
    - Title - single line of text
    - userEmail - single line of text
    - option single line of text
  - Events - Events List
    - (The list is defined using custom XML in the user event service)
  - EventHasGroups - List
    - Title - single line of text
    - groupID - single line of text

Prerequisites:

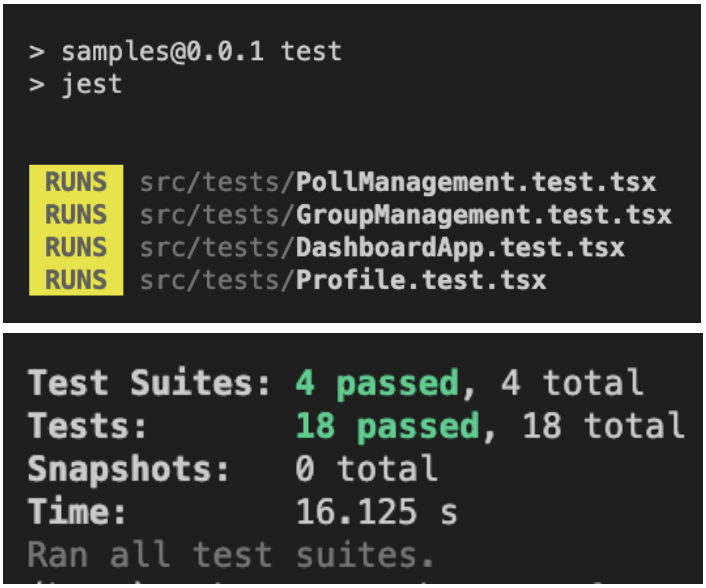
- PnPJS library
- Office UI Fabric React components
- React Big Calendar components
- Microsoft 365 tenant
- SharePoint Lists

Unit test and usage:

- The unit tests can be found under /spfx-react/src/tests (this can be modified to

- Run this terminal command to run the unit tests:

```
npm test
```



Deployment (as an admin of RHUL SharePoint admin centre): packaging the web parts

Steps based on Microsoft (if unable to follow steps below, then refer to link below):
https://learn.microsoft.com/en-us/sharepoint/dev/spfx/web-parts/get-started/serve-your-web-part-in-a-sharepoint-page

The deployment for the web part depends on the developer's access into the administration centre of RHUL SharePoint. However, the team did not have access into the administration centre. This would have been the steps to follow if given the administration access:

1. Go into the directory of the solution (spfx-react) and make sure **gulp serve** is not running (can be stopped by selecting CTRL + C)

```
cd spfx-react
```

2. Package the web parts - open the **package-solution.json** from the **config** folder. This file defines the metadata of the package
3. Run this terminal command to bundle the client-side solution

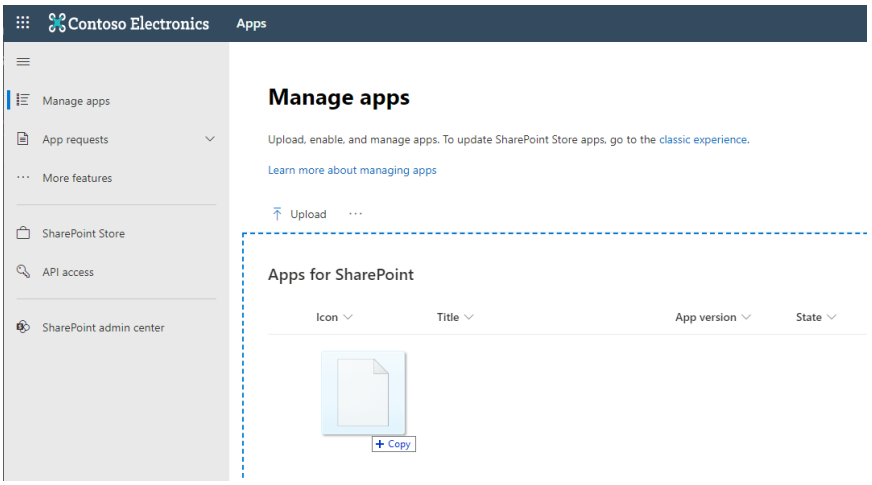
```
gulp bundle --ship
```

4. Run this terminal command to package the client-side solution that contains the web-part (this will create the following package of './sharepoint/solution/[name of solution]')

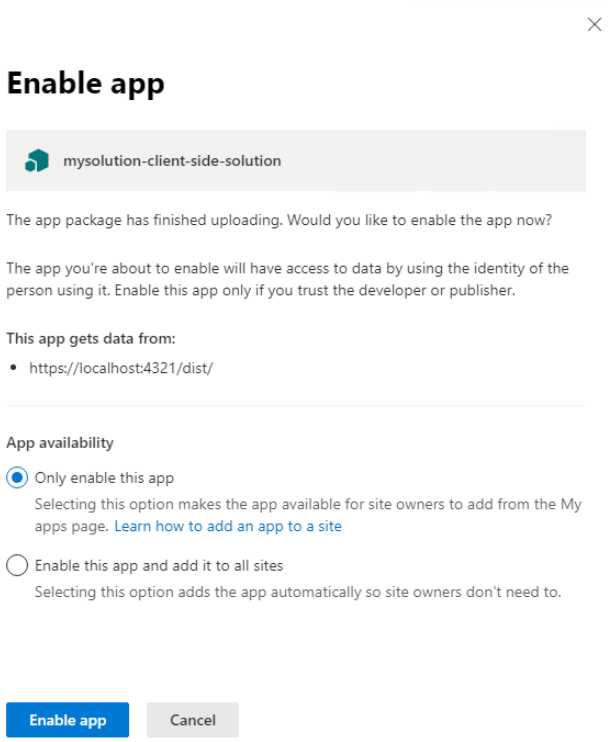
```
gulp package-solution --ship
```

Deployment of the package to the app catalogue:

5. Go to your SharePoint site app's catalogue
6. Go to 'Manage Apps' - upload and drag and drop the .sppkg to the app catalogue

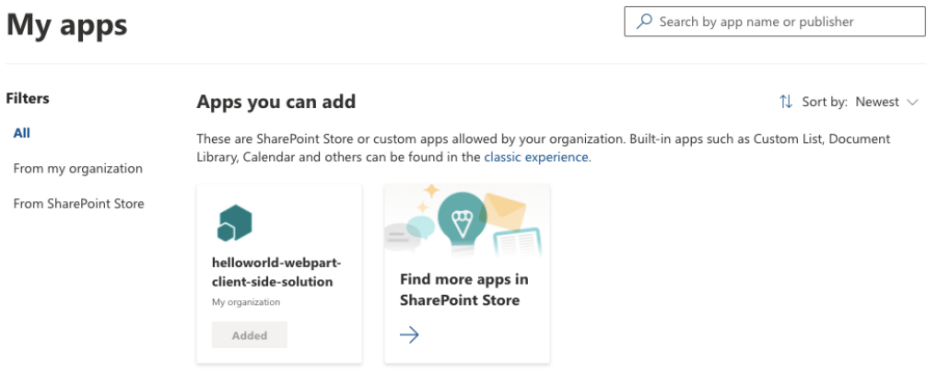


7. Answer dialog that is asking you to trust the client-side solution to deploy **- select enable app**

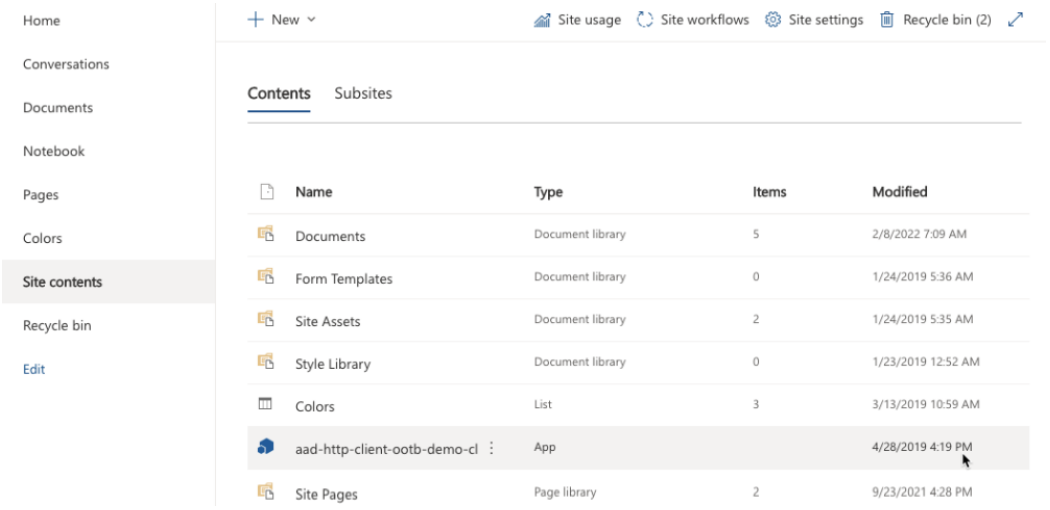


Install the client-side solution to your SharePoint site:

8. Go to your developer site collection for testing
9. Select the gear icon on top right of the navigation bar, then select 'Add an app' to go to the Apps page.
10. In the Search box, search for the name of your packaged solution and select ENTER to filter your apps



11. Select the [solution-name]-client-side-solution app to install the app on the site (the site contents page shows the installation status of the client-side solution; make sure this installation is complete before going to the next steps of previewing the web parts)



Preview the web-parts on SharePoint page:

12. Open the {{your-webpart-guid}}.manifest.json from the dist folder.

```
"internalModuleBaseUrls": [
  "https://localhost:4321/dist/"
],
```

13. Before adding the web parts to a SharePoint server-side page, run the local server.
14. Run the terminal command inside the console window that has the project directory (to serve from localhost):

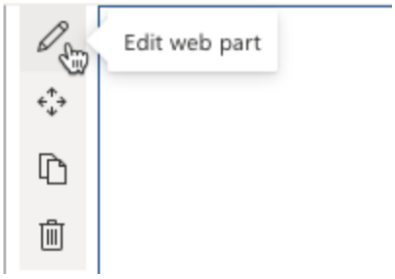
```
gulp serve --nobrowser
```

Add the web parts to the modern page:

15. In browser, go to the site where the solution was installed
16. Select the gears icon in the top right of the navigation bar, and then select ‘Add a page’
17. Edit the page
18. Open the web part picker and select the web parts you have recently uploaded

Edit the web parts:

19. Select the Configure element icon (pen) in the webpart to open the property pane of the web part (the user manual shows the web parts which require configuration)



Gantt Chart

	TASK	Start date	End date	D	TERM 1									BREAK				TERM 2									
					1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	**Initiation**																									
1.1	Introduction to project	10/10/2022	17/10/2022	7																						
1.2	User requirements	17/10/2022	21/10/2022	4																						
1.3	User stories and use case analysis	17/10/2022	24/10/2022	7																						
2	**Content**																									
2.1	Tool comparison	24/10/2022	4/11/2022	11																						
2.2	Presentation of tool comparison	31/10/2022	4/11/2022	4																						
2.3	Pending release of development environment for chosen tool	4/11/2022	18/11/2022	14																						
3	**Design**																									
3.1	UI design of storyboards	15/11/2022	26/11/2022	11																						
3.2	Presentation on storyboards	21/11/2022	25/11/2022	4																						
3.3	Review design and improve UI from review	25/11/2022	9/12/2022	14																						
4	**Implementation / Testing**																									
4.1	Sharepoint playground - first interactions and backend set-up	9/12/2022	27/12/2022	18																						
4.2	Login system / account management	13/12/2022	27/12/2022	14																						
4.3	Dashboard	27/12/2022	14/1/2023	18																						
4.4	Spaces and groups	10/1/2023	24/1/2023	14																						
4.5	Documents component	24/1/2023	7/2/2023	14																						
4.6	Polls component	28/1/2023	11/2/2023	14																						
4.7	Calendar component	11/2/2023	4/3/2023	21																						
5	**Review**																									
5.1	Review project	4/3/2023	11/3/2023	7																						
5.2	Make final updates	25/3/2023	29/3/2023	4																						
5.3	Official launch	25/3/2023	29/3/2023	4																						