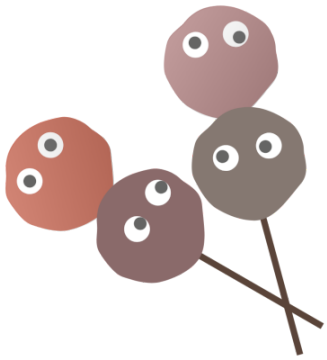
CS3216 Final Project
Final Report



MeetBalls

Get the ball rolling

https://meetballsapp.com/

Members:

Ang Chun Yang A0180311X

Chong Wen Hao A0217317J

Lim Jin Hao A0205878R

Ong Ying Gao A0201924N

Table of Contents

1. Application Description

MeetBalls is an all-in-one companion web application to Zoom that aims to help meeting secretaries manage meetings and improve productivity. It aims to automate or simplify traditional tasks for secretaries with features such as an agenda planner, automated attendance taking and timekeeping, mass emails, et cetera to participants in a bid to make meetings less painful to manage.

By offering the above services as a single intuitive package, it could also encourage more users to plan even less formal meetings with it, potentially keeping meetings focused and on time and allowing conference members to get more done with less.

2. Real-world problem

From our interviews with various parties who have experiences with meetings, we found that tools that secretaries use fall under 3 categories.

First are the tools that are used before the meeting. Each meeting usually has a person or a team responsible for planning a meeting's agenda and collating information from participants before sending it for approval. This is sometimes done with apps such as Microsoft Word and Outlook.

Next, we have tools that are used during meetings. One aspect of this would be taking minutes, but we found out that the secretaries may either choose to scribe by hand or type it out. Our interviews also showed us that timekeeping while taking notes and controlling slides is difficult, so a tool that can help to keep track of timing may help.

Lastly, after meetings, secretaries would tidy up their minutes and email it out to the participants. Some companies would even have two secretaries taking down minutes during each meeting, and they can compare and compile their minutes after the meeting. Hence, a tool for mass emailing of participants would be helpful.

3. Market Validation

There are plenty of other meeting productivity applications. However, most of these applications focus on duties from just one of the above categories and this does not help in lessening the number of separate tools that they already have to use to run a meeting. For example, if we refer to this blog, most of the applications listed offer services such as agenda sharing, issue tracking, note taking but hardly any attempts to consolidate those services together.

Our closest competitor would be Meeting Decisions, which aims to solve the same problem we laid out. However, their product is meant for use with Microsoft Teams while ours is meant for use with Zoom. We went for Zoom integration as Zoom is leading in the web conference market (with 48.7% of the market share against Teams' 14.5% as of 2021) and we can leverage on its extensive user base to reach more users.

4. Milestone & Timeline

Please refer to our milestones at the Appendix.

5. Contributions & Role

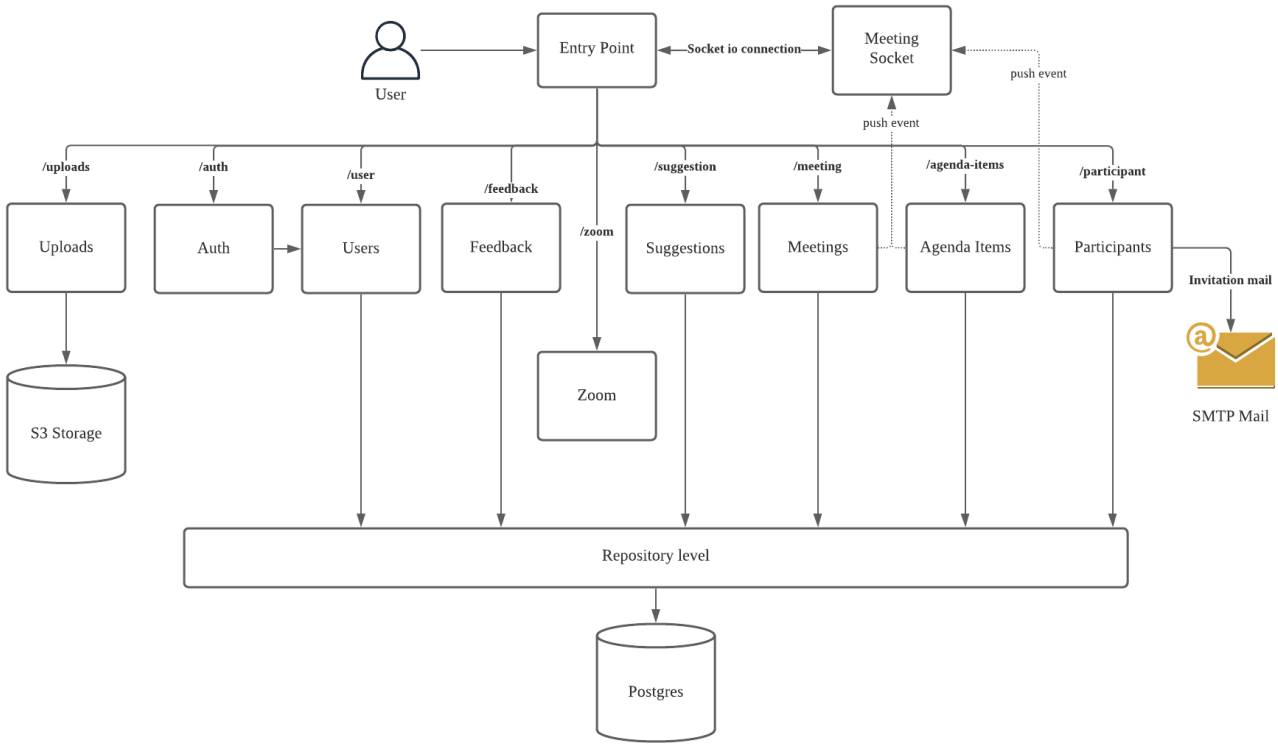
Team members	Role	Contributions
Ang Chun Yang	Backend	- Participants, Suggestions, Agenda Items endpoints - Marketing
Chong Wen Hao	Frontend	- Styling - Graphics design - UI - Video and animation
Lim Jin Hao	Backend	- Authentication, User and Meeting endpoints - Zoom integration - Socket integration
Ong Ying Gao	Frontend	- Business logic - Wrote presentation script - UI

6. Application Design

6.1 Backend

Our backend is built on the NestJS framework as it is a versatile, extensible and progressive Node.js framework. Furthermore, NestJS has a robust and well-maintained documentation and supports dozens of nest-specific modules, which enable us to use common techniques and tools such as TypeORM, Swagger, Validation and Logging.

6.1.1 Architecture Diagram

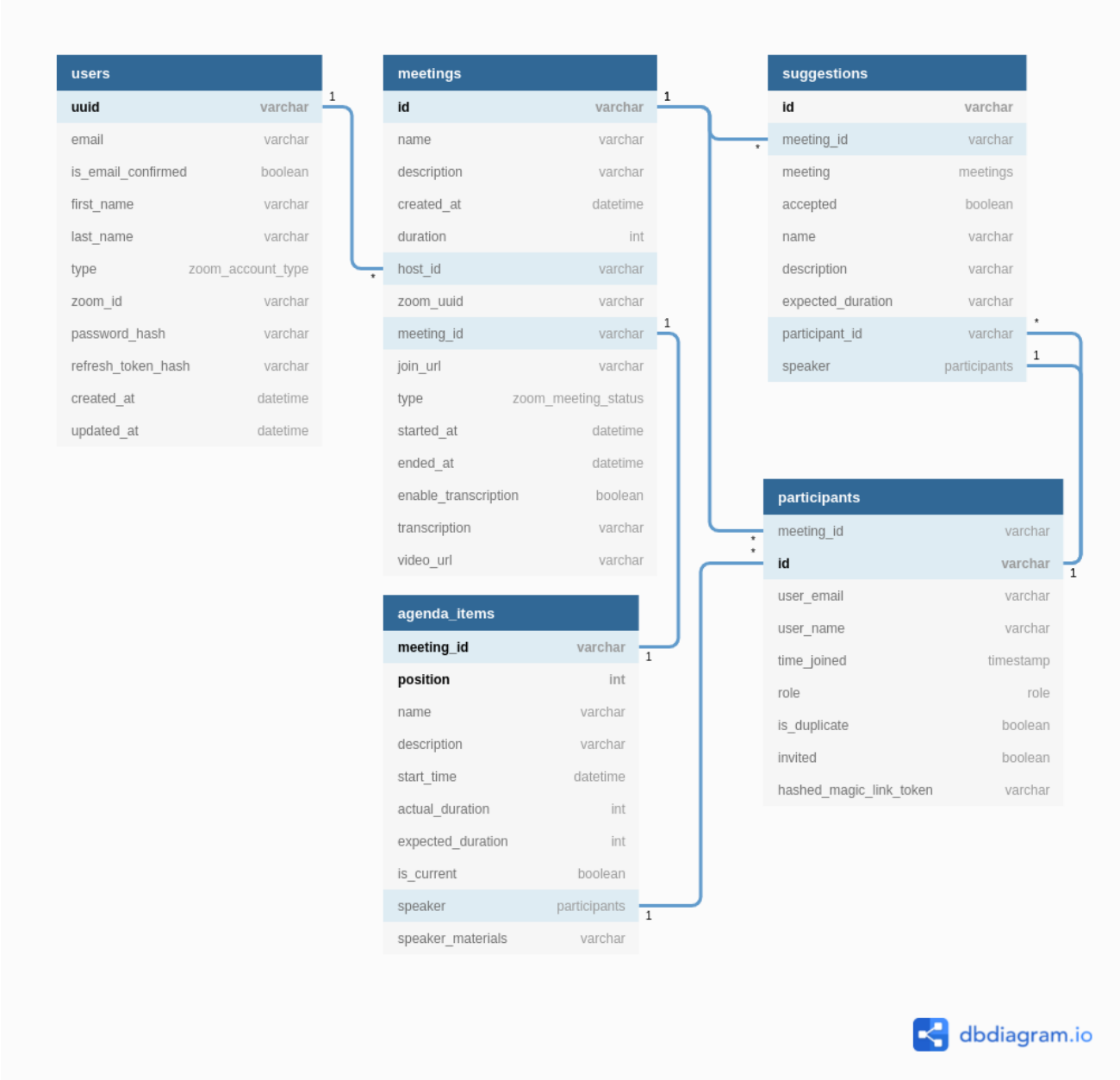


The above diagram briefly summarizes the structure of our backend and the relationship between components. The backend code follows an Angular-like structure where each logical component is grouped as modules and controllers defines the behaviour of the endpoint.

We deployed our backend on Heroku and used the Heroku Postgres database for our database service. We used Papertrail to log network requests for maintenance and debugging and flag any errors such as 500 error codes.

We also deploy 2 separate backend instances on Heroku, one for production and one for development. This workflow is used so that any changes we make to our staging branch do not break our production build instance.

6.1.2 Database Schema



The above Entity-Relationship diagram (link) captures the information stored in our Postgres database. Primary keys are shown in bold while the one-to-many and one-to-one relationships are specified by the connection between two entities.

Our *users* table contains all the necessary information to identify the user as well as authenticate the user. The user's *zoom_id* is used as a unique identifier tying the user to Zoom.

Our *meetings* table contains the details of the MeetBalls meeting created by the user. Each meeting has a one-to-many relationship with the *participants* table, *agenda_items* table and *suggestions* table.

The *participants* table holds all the participant details. The email in the *participants* table is used to send a curated invite link to all meeting participants. The *agenda_items* table holds the details of the meeting agenda and includes important information such as position of the item, the expected duration, speaker item and speaker materials.

Lastly, the *suggestions* table includes the suggestions created by a meeting participant. A participant can suggest an agenda item and also assign a speaker to it. The meeting host can then choose to either accept or reject the suggestion.

6.1.3 REST API Design

Link to API Swagger documentation: https://meetballs-dev.herokuapp.com/api/

** Note the above swagger endpoint is only used for our development build and not deployed on the production server for security reasons*

The REST API documentation is logically organised and titled according to their request method and relative url. For each API documentation, the schema of the request parameters, query and body are shown. The response success code, data and error codes for each REST API are also documented.

A summary of each section is shown below:

API Section	Summary
Server	Endpoints used to check the health of the server
User	Endpoints to get user information such as their name and Zoom id
Authentication	Endpoints that handle all authentication-related calls. Mainly logging in to Zoom using authorization code and refreshing Zoom's access token
Meeting	CRUD endpoints for authenticated users to create, update and delete meetings. Also contains endpoint for meeting management, such as starting, ending meeting and moving to next agenda item
Agenda Item	CRUD endpoints for meeting hosts or co-host to create, update and delete meeting agendas. Also contain endpoint for reordering the agenda items position for a specific meeting

Participant	CRUD endpoints for meeting hosts or co-host to create, update and delete meeting participants. Also contain endpoints to send meeting invitations and mark a participant as present, absent or duplicate during a meeting
Zoom Meetings	Endpoints that retrieves Zoom meeting information from the Zoom API. The sync endpoint ensures that Zoom meeting data are in-sync with MeetBalls meeting data so that our core Zoom features such as automated attendance taking can work.
Feedback	Endpoint to create user feedback once the meeting host completed his meeting
Uploads	Endpoints to generate a signed S3 url to upload or download meeting materials from/to the S3 server.
Suggestion	CRUD endpoints that allow participants of meetings to create, update and delete suggestions. It also contains an endpoint to allow the host of a meeting to accept or reject a suggestion.

6.1.4 Zoom Authentication

As our app uses Zoom's JWT authentication to authenticate users and communicate with Zoom's API, there are a few requirements that we have to comply with for the authentication flow.

For the frontend, the user will authorize the Zoom application and receive an authorization token from Zoom. The backend can then help the user login by using the authorization token and client secret to retrieve Zoom's access and refresh token. This access token has to be used for all subsequent communication with Zoom's API, which can only be called from the backend.

The backend is also required to create a deauthorization endpoint so that all of the user's private identifiable information will be deleted when a user chooses to deauthorize the Zoom app as per Zoom's privacy policy.

Furthermore, Zoom requires that all authorization, deauthorization and redirect endpoints come from the same domain as the frontend. This means that we cannot simply provide the backend endpoint as the deauthorization endpoint as our backend uses heroku and does not have the same domain as our frontend. To resolve this, we used Netlify functions to route deauthorization requests from Zoom to the backend.

6.1.5 Zoom Event Subscription

To allow for features such as automated attendance taking, we subscribed to Zoom's "user joined" event. Hence we have to create an endpoint that will be called by Zoom whenever a *user_join* event takes place. This endpoint will check if the meeting is managed by MeetBalls before creating the participant or marking the participant as present.

6.1.6 Live Socket Implementation

As one of the core features of MeetBalls is the management of a live meeting, we require that all of our Meeting participants get access to the same meeting information in real-time. To allow for real-time implementation we use Socket.io to transmit update events to all connected participants.

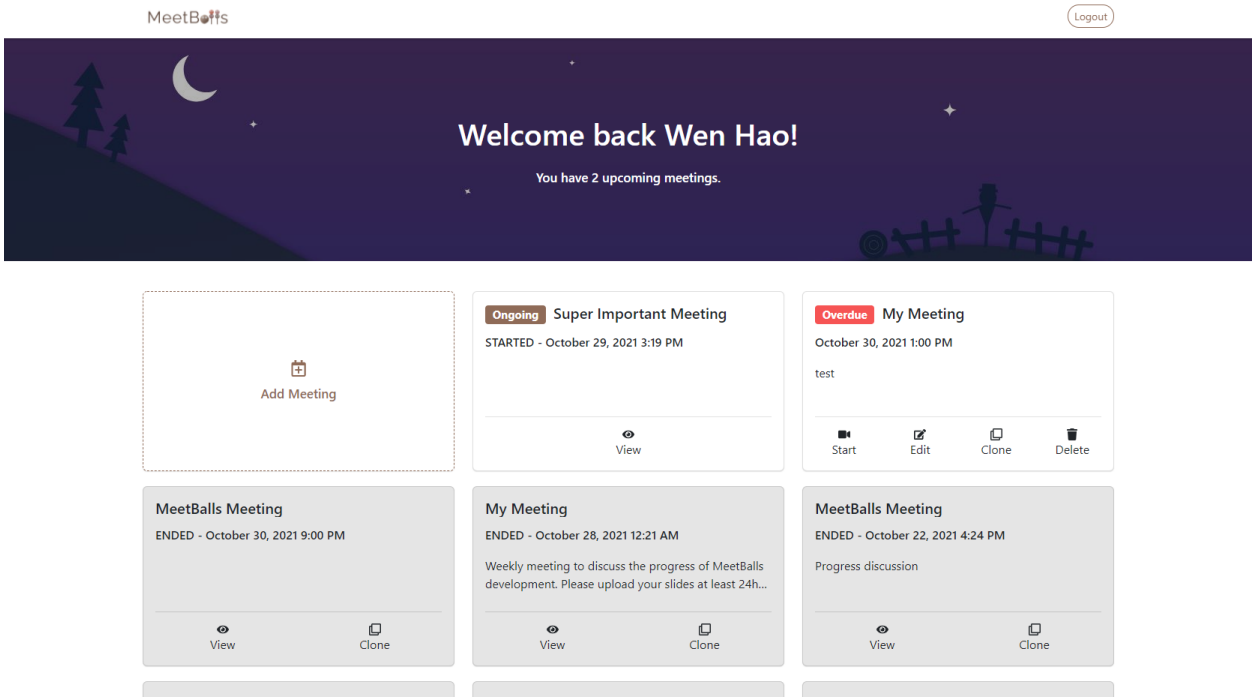
Whenever a change in meeting details (eg. agenda, participants) is detected, the meeting-socket module will be used to transmit the relevant socket.io event to all connected users. Each user will then conduct a merge of the edited item so that everyone has the same meeting information. In this workflow, the backend serves as a single source of truth that pushes change events to all subscribed users.

6.2 Frontend

Our frontend is built on the React framework. It allows us to design reusable UI elements and is also fast as a single page application. React also offers many third party libraries for various UI options such as React Bootstrap and Toastify which speeds up development.

6.2.1 Dashboard

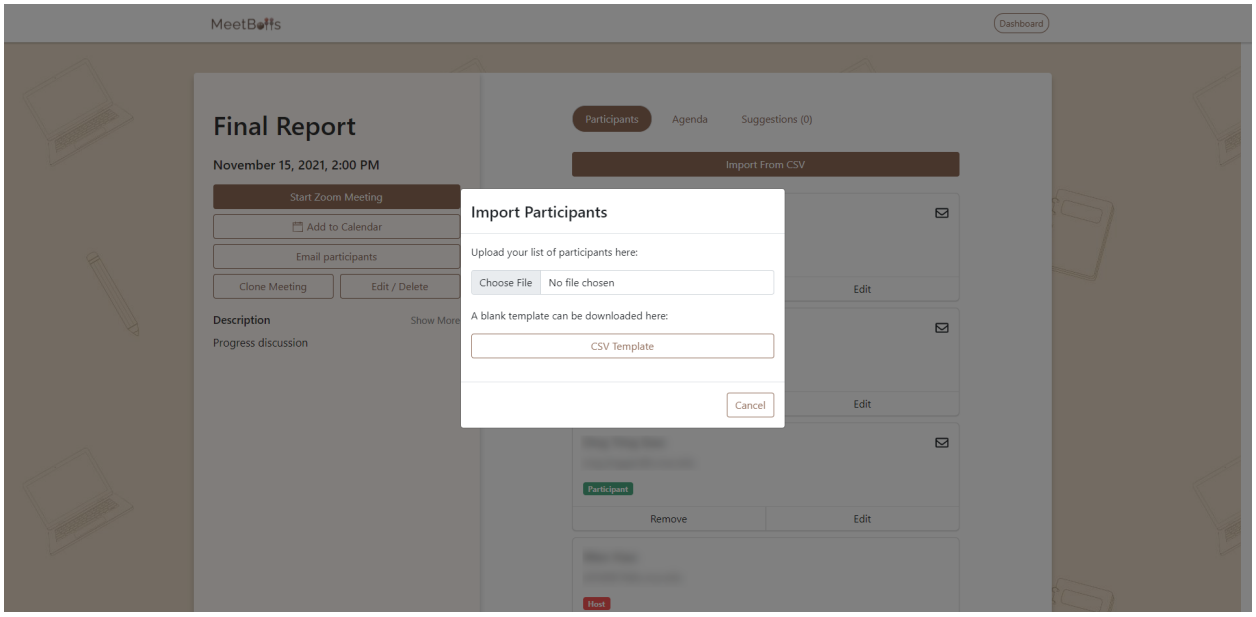
After logging in with their Zoom account, users will be redirected to the dashboard screen shown below. Here, users will be able to view all the meetings that they have created, both upcoming and those that have passed. Users will also be able to create a new meeting, or clone the details of an existing meeting to a new one.



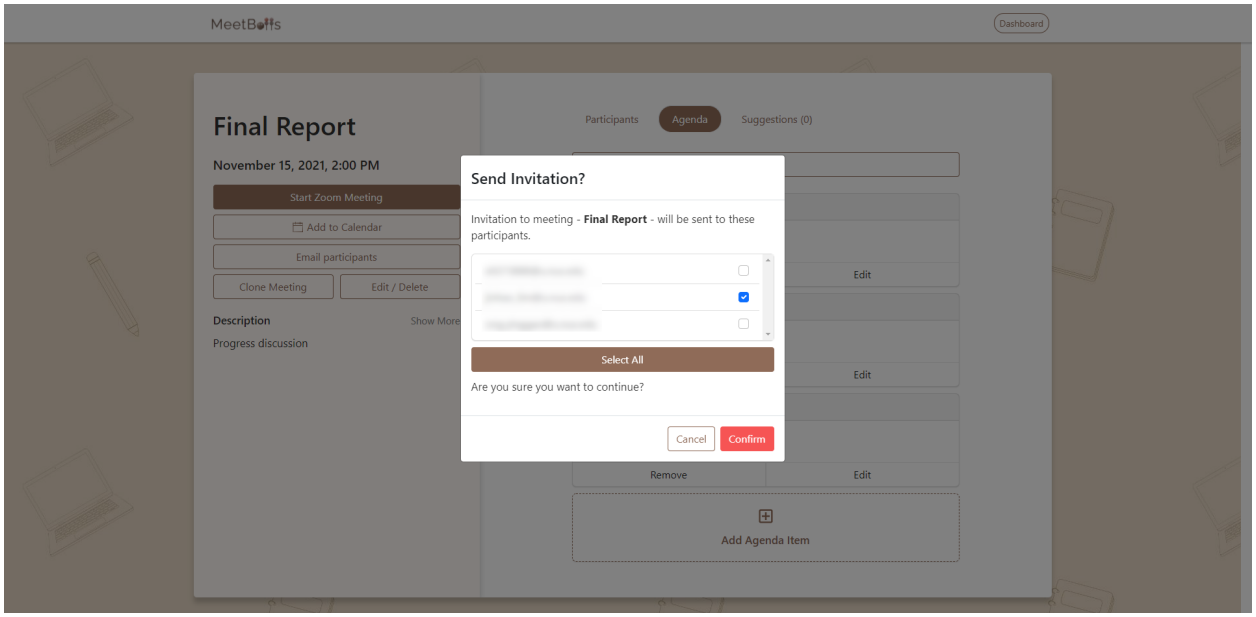
From our users' feedback, we also made the dashboard more personal by including the user's name in the banner. As we are using Zoom authentication, we are extracting user information directly from Zoom so we do not have an edit profile page.

6.2.2 Meeting Planner

After a user has created a meeting, they are brought straight to the upcoming meeting screen, where they can add participants and the agenda to the meeting. Users can either add participants manually by entering their name and email address or import them through a CSV file. Participants can also be chosen as co-host and given the ability to manage meetings when they start.



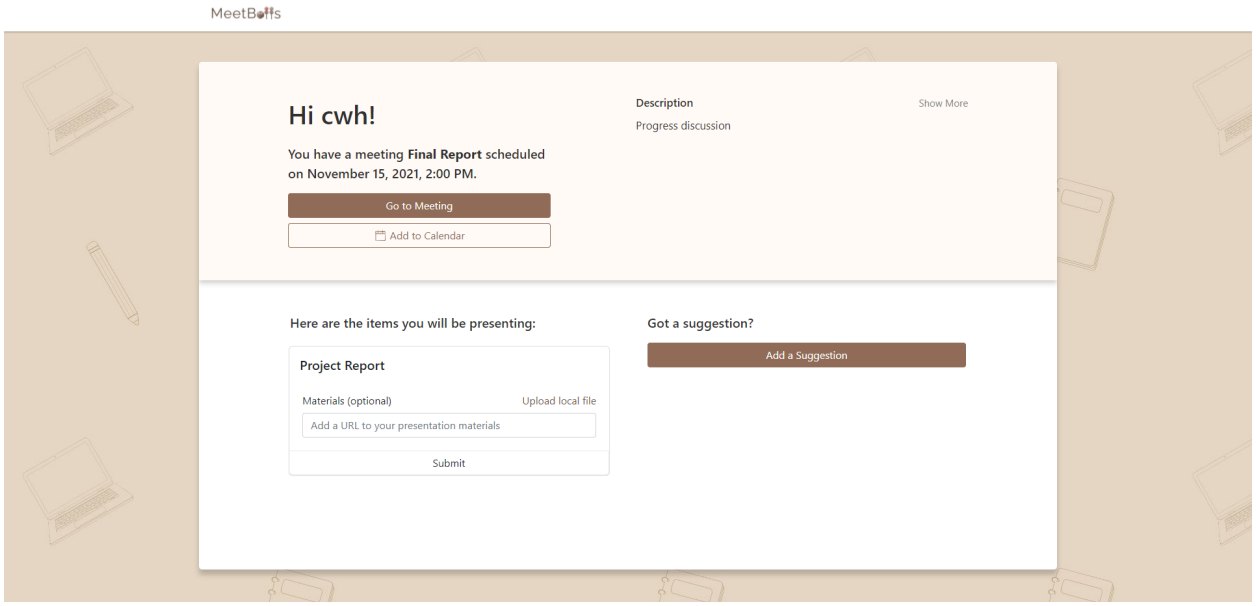
The user can also add items to the agenda, set the duration and assign speakers to them. Once the participants and agenda have all been added, users can click on the “Email participants” button to send mass invitations. By default, only uninvited participants will be selected, but the user can resend invitations if necessary.



Users can also review suggestions for new agenda items from their participants and either approve or reject them. We will cover more of this in the participants’ UI. Approved suggestions are automatically added to the back of the agenda.

6.2.3 Participant UI

Every participant will receive a unique link. The page the link takes them to displays their name (in the welcome message) and the agenda items that they have been chosen to present. If need be, participants may upload materials to those items.

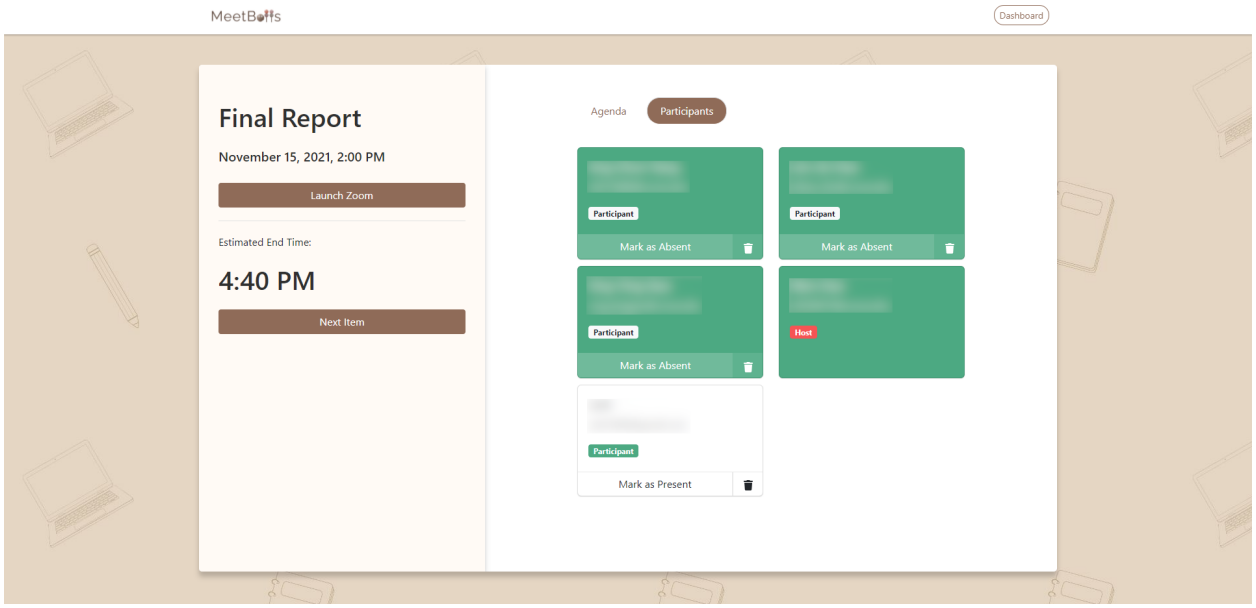


Participants can also suggest the addition of new agenda items. Hosts can view these in the meeting planner. This feature was inspired by a problem brought up by one of our interviewees, who complained of the need to add new agenda items manually.

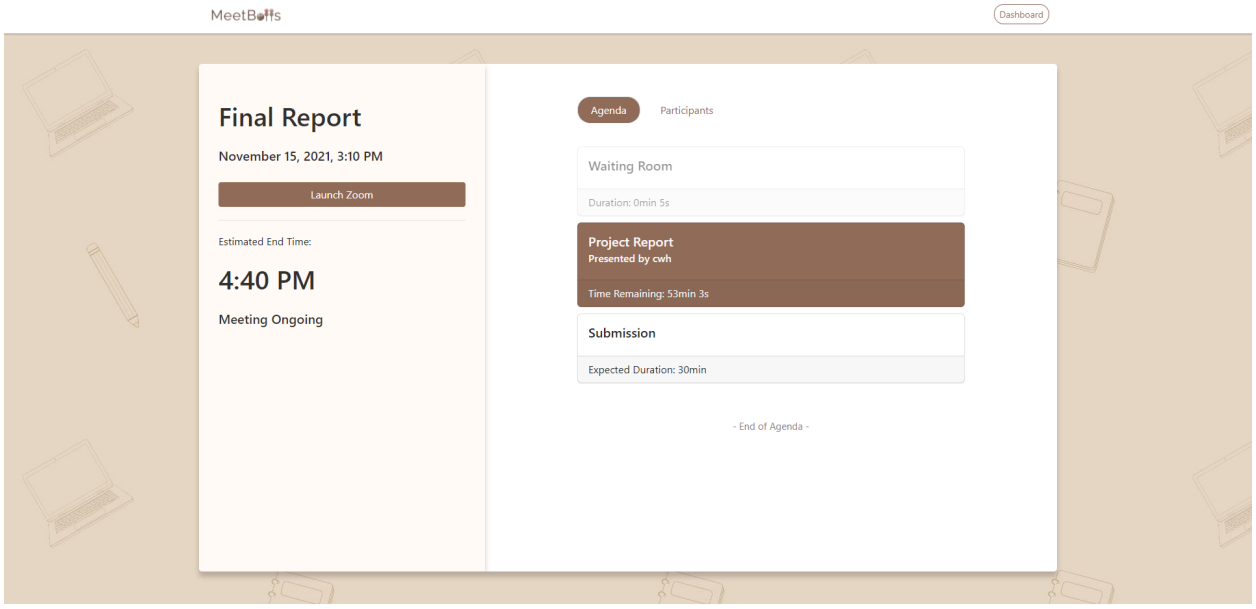
Participants can also navigate to the meeting page to view the full agenda, participants list and the meeting link.

6.2.4 Ongoing Meeting

When the meeting has started, MeetBalls automatically takes the attendance of participants who join the Zoom meeting. Zoom users whose emails are not in the invitation list are recorded as different users. Host can remove any duplicate participants that arise.

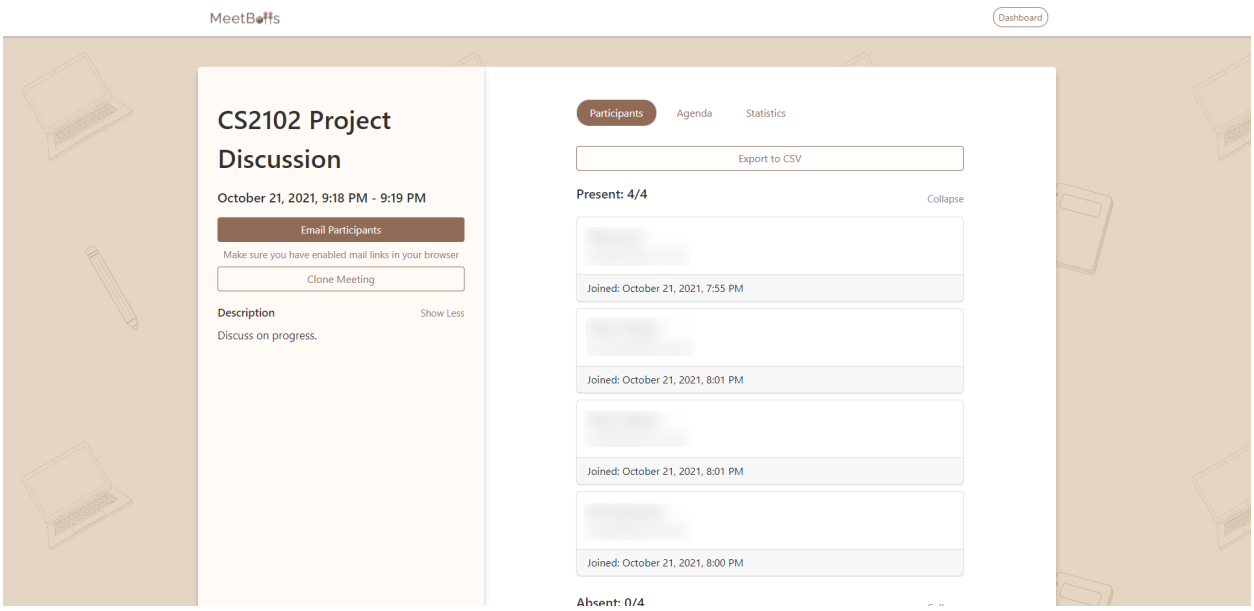


The host and his participants can view the agenda and estimated time remaining. This estimation is based on the current pace of the meeting. Exceeding the time limit on an agenda item will trigger a bell to inform the host to move on to the next item. Users will also be able to download any attachments for the agenda items.



6.2.5 Post Mortem

After a meeting has ended, the host can email all participants with a single button click. This would normally be used for sending minutes, but it can also be used for other matters if need be. The host can export the attendance as a CSV file.



7. Marketing Campaign

We started marketing MeetBalls through Instagram (@meetballsapp) and Facebook as soon as our MVP was ready on 19 October. We posted on our social media 2 times a week to showcase our app's features and how it helps to solve the issue of unproductive meetings. We also created a demo video and TikTok videos to showcase different features of MeetBalls.

Demo Video: https://youtu.be/Ru1LYvjYynk

Furthermore, we also did our marketing by sharing MeetBalls through Telegram channels such as NUS Computing Grads, and sharing it with our friends who need to conduct meetings. As we obtained more users progressively through our marketing campaign, we also started conducting user interviews and obtaining feedback from different users on what they like and didn't like about MeetBalls.

8. User Analytics

Report on the current number of users who have installed, active users, etc. Perhaps Google Analytics data and screenshots (or similar analytics tools) to support your claims

As of 15 November 2021, we have managed to obtain 29 users to create a MeetBalls account and 72 meetings were created by them. The screenshots below show the queries made in heroku to obtain the data mentioned. *(Actual output not shown for privacy purposes)*

```
1  SELECT * FROM users ORDER BY Created_at;
```

RESULTS
29 rows in 21 ms

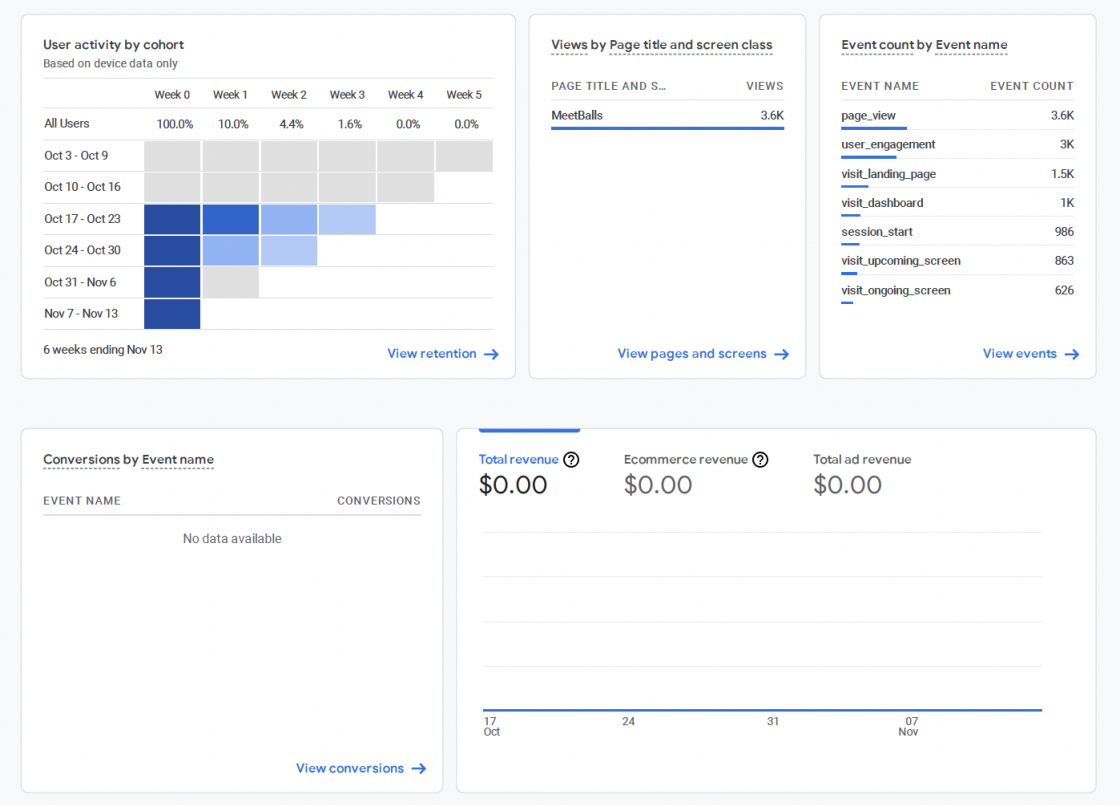
UPDATED
November 15, 2021 14:14

```
1  SELECT * FROM meetings
```

RESULTS
72 rows in 24 ms

UPDATED
November 15, 2021 14:13

The screenshots of the Google Analytics data are shown below, which shows that as of 15 November 2021, there were 361 unique users who visited MeetBalls.



9. Future plans & strategies

After the end of the semester, our plan for MeetBalls would be to firstly discuss and possibly redefine the scope of MeetBalls and the problem that we want to solve. One thing that we realised from user interviews with the staff from SOC is that there MeetBalls lacks two-way interaction features such as asking questions and voting. Thus, we plan to discuss this further and decide what are some two-way interaction features we should implement.

Next, we also plan to expand our target users. Currently, MeetBalls was built for secretaries to conduct more productive meetings. However, we received feedback from some professors that MeetBalls could possibly be useful when conducting lessons and even exams. For instance, Prof Damith asked if MeetBalls is able to detect if users have left a meeting. We felt that this could potentially be a useful feature for professors to keep track of who is still present during an online exam and reduce the possibility of cheating. Hence, in the future, we could consider offering different meeting types to fit this use case and possible ways of doing so.

During STePS, we also received some feedback regarding the UI. For example, Prof Ben Leong commented that our space usage is wasteful for displaying participants'

information in a large card. This is especially so in large meetings as the secretary would have to scroll far down to find a participant. We plan to redesign the UI to fit the recommendation.

Lastly, once MeetBalls finally gets approved by Zoom, we plan to finally implement our transcription feature that helps to produce a transcript for users after a meeting has been completed.

10. Learning & Summary

One key takeaway was that tasks should be allocated to match each team members' skills. Previously we simply took the list of features and distributed them which led to inconsistent work quality. We later redefined our roles by the first sprint. For instance, one of the frontend developers was not as good at UI programming as the other. As such, he was tasked with handling the business logic instead while the other would focus on UI as a whole. By leveraging off each members' proficiencies, we could increase the quality of work and production speed. Another side effect is that everyone now had a broader understanding of the whole application as we were involved in every feature in some form.

Another was learning how we can integrate the Zoom API, something completely new to the team. The main challenge was obtaining approval from Zoom to publish our app in the marketplace. We did not anticipate the approval process from Zoom's side to take such a long time. Hence, we learnt that in future, if we intend to build an application that requires an approval before it can be officially published, we should set aside more time for the approval process to be safe.

The team also picked up new skills along the way by working on stuff we have never done before. For some of us, this is the very first time we have used web sockets. This is also our first experience with React UI for some members too. As none of us have a strong art background, designing graphics and animation was also a novel challenge

Appendix: Milestones and Timeline

Legend: Brown is late, red is not done.

Week	Technical	Marketing	User Acceptance Testing
8	☑ ~~Designed UI/UX for MVP~~ ☑ ~~Designed Data Model~~ ☑ ~~Implemented MVP without Zoom Integration~~ ☑ ~~Authentication~~ ☑ ~~Meeting info~~ ☑ ~~Agenda planner~~ ☑ ~~Show agenda items in-app~~ ☑ ~~Full implementation of timekeeping~~ ☑ ~~Add attendance taking~~		☑ ~~MVP out by end of week (Late)~~ ☑ ~~Conducted user interviews to get a sense of experience and issues with online meetings and tools.~~
9 (MVP released)	☑ ~~Automate zoom attendance~~ ☑ ~~Get lists of zoom meetings~~ ☑ ~~Agenda items to participants~~ ☑ ~~Meeting minutes to participants~~ ☑ ~~Roles to indicate the speaker~~ ☑ ~~Meeting item file attachment/link~~ ☐ Add buttons to inform “nothing to add”, “don’t know what to say” (abandoned) ☑ ~~Add google analytics~~	☑ ~~Create posters required to be posted on social media/ sent out.~~ ☑ ~~Create an Instagram account and start hyping people up.~~ ☑ ~~Start creating Landing Page - complete at least the mockup~~ ☑ ~~Start small first and market the app to CS3216 students for them to use it during their meetings. (late)~~	☑ ~~Launch to NUS users (delayed due to bugs)~~ ☑ ~~Submission for Zoom approval (delayed due to bugs)~~

10	☑ ~~Allow participants to suggest new agenda items~~ ☑ ~~Display meeting post-mortem (e.g. overtime duration)~~ ☑ ~~Performance improvements (caching)~~	☑ ~~Finish creating the landing page~~ ☑ ~~Post app's features one day at a time on Instagram.~~	☐ Interview users for feedback (CS3216 peers) (Decided to focus on those who conducted more formal meetings) ☑ ~~Interview users for feedback on new build (NUS teaching staff)~~
11 (Report 2)	☑ ~~Bug fixes and tweaks~~ ☑ ~~Security enhancements~~ ☑ ~~Beautify UI~~ ☑ ~~Clean up code~~ ~~Prep for final presentation~~ ☑ ~~Performance improvements (pagination)~~ ☐ Transcription service (unable to implement due to waiting for Zoom publication approval)	☑ ~~Continue hyping on social media about our initial launch~~ ☑ ~~Create small snippets of tutorial videos to post on social media via TikTok and post them on stories on instagram.~~ ☑ ~~Initial poster design~~ ☐ Share about people's experience before using vs after using the app. ☐ Show data from google analytics to show the effectiveness of our product ☐ Time saved/exceeded on average for each meeting ☐ Number of active users who use the app ☐ Number of users who signed up ☐ Would have gotten some reviews and feedback, and	☑ ~~Interview users for feedback on final build (CS3216 peers and NUS teaching staff)~~

		post users' reviews too on top of the app's features. (Abandoned due to the low number of users, we decided it would be better not to share the figures)	
12 (Final project presentation)	☑ ~~Bug fixes and tweaks~~ ☐ Security enhancements (Received feedback from Zoom recently regarding security, will be working on it after the semester)	☑ ~~Initial launch to the public~~ ☑ ~~Create posters~~ ☑ ~~Do up a 1-2 min pitch~~	
13	STePs		
Reading week	Final report due		