Sprint 2 Retrospective Document

Eric Aguilera, Justin Bonner, Carson Harmon, Dan Zheng

What went well?

During sprint 2 we finished the primary features of our application: messaging and meetings. Throughout our sprint we became more flexible with our work roles, allowing group members to reach maximum productivity. Communication outside meetings has improved greatly assisting group members as they work remotely.

#	Task description	Estimated Time	Owner
1	Create notification model and associate with user model	2 hrs	Justin
2	Add back-end controller for adding user notifications	2 hrs	Justin
3	Add front-end view for notifications	2 hrs	Justin
4	Add logic to existing actions that require notifications	2 hrs	Justin

Completed. Users can view their notifications on their home page. We added code to various back-end controllers so that certain actions (such as chat invitations and meetings deletions) send notifications.

#	Task description	Estimated Time	Owner
1	Add front-end view for viewing public profiles.	5 hrs	Justin
2	Add back-end controller for generating a public profile.	5 hrs	Justin
3	Incorporate privacy settings into public profiles.	5 hrs	Justin

Completed. Users have public profiles, which list their information and a chat invitation button. Public profiles integrates with privacy settings so that only public information is displayed.

#	Task description	Estimated Time	Owner
1	Add a front-end interface for privacy settings in the user profile view.	5 hrs	Eric
2	Add back-end controller for updating a users privacy settings.	5 hrs	Eric

Completed. Users have a privacy settings panel on their profile page. They can view and update their privacy settings using toggle buttons.

#	**Task description**	**Estimated Time**	**Owner**
1	Add messages model and associate with group model	5 hrs	Eric
2	Create prototype front-end view for chats	5 hrs	Carson
3	Add back-end logic using Socket.io	5 hrs	Eric
4	Add front-end logic using Socket.io	5 hrs	Eric

Completed. Messaging has been completed. We implemented messaging using the Socket.io library, so users can send and receive messages from their chats in real-time.

#	**Task description**	**Estimated Time**	**Owner**
1	Create meetings model and associate with user model	3 hrs	Dan
2	Create back-end controller for creating meetings and inviting users	3 hrs	Dan
3	Create front-end view for meetings page and inviting users	3 hrs	Dan

Completed. Users can create a meeting by specifying a meeting name, dates, times, and optionally a location and description. We created a robust database schema for Meetings which stores all of the information for a meetings, including the dates/times and user responses.

#	**Task description**	**Estimated Time**	**Owner**
1	Create back-end controller for meetings RSVP	5 hrs	Dan
2	Create front-end view for meetings RSVP	4 hrs	Dan
3	Add data visualization of available times for meetings	5 hrs	Dan

Completed. Users can RSVP to a meeting by selecting the times that they are available. We used jQuery and custom client-side JavaScript to implement click and drag functionality so that users can click and drag to select times, instead of clicking times individually.

#	**Task description**	**Estimated Time**	**Owner**
1	Create back-end controller for finalizing time and location	3 hrs	Carson
2	Create front-end view for finalizing time and location	3 hrs	Carson
3	Add logic for notifying meeting attendees	3 hrs	Carson

Completed. A meeting creator can finalize a time and location for a meeting, upon which all meeting participants will be notified. We implemented data visualization using d3 and jQuery so users can easily see which meeting times are best for everyone.

#	Task description	Estimated Time	Owner
1	Add back-end controller for handling cancellation of meetings.	3 hrs	Carson
2	Add back-end controller for sending notifications upon meeting cancellation.	3 hrs	Carson

Completed. Meeting creators can cancel meetings and meeting members will be notified.

What didn't go well?

During this sprint, it was difficult at times for some team members to find work. Since parts of a feature were assigned to different team members, at times one person had to wait for others' work to be done before writing code. Although we also faced this issue in sprint 1, we improved our task assignment strategy and communication, so it was not as severe during this sprint.

Also, we encountered issues when concurrently prototyping our code on the same database, which caused several database conflicts. Ultimately, we could resolve this problem by connecting to a local database so that the work of other team members is not affected.

How should we improve?

Although we made well-defined task assignments for sprint 2, as the sprint progressed, we naturally became more flexible with our work roles. As our project comes to a close, we could improve having more fluid work roles since most of the main features have been completed. This will allow group members to freely work on the final tasks without needing to wait for other work to be done.