Sprint 2 Retrospective

Team 7 - Quillio

Ammar Husain, Anoop Jain, Charlie Crouse, Jenna Ellis, Jiwon “Daniel” Kim, Prashanth Koushik

What went well?

In general, we built out the major features of our project and integrated all of the important speech to text and transcription processes we planned to incorporate. We integrated in-meeting recording and transcription as well as editing and updating of transcripts after meetings have ended.

User Story 1: I would like to be able to access my meeting transcriptions (and summaries) in a web application.

1	Integrate meeting recordings and GST with core app	15	Anoop, Ammar, Charlie
2	Store returned transcript in note model	4	Charlie
3	Create UI for displaying the transcript text	4	Ammar
4	Add edit and update feature	6	Charlie

Completed:

The transcript is stored in the database after being generated, but we broke it up into a vectorized transcript by speaker and then a separate raw text document. The transcript is editable and savable using the quill text editor.

User Story 2: I would like my meetings to be tagged based on meeting topics.

1	Research LDA API	4	Anoop and Jenna and Ammar
2	Get basic demo running with sample text	6	Anoop and Jenna and Ammar
3	Run barebones tagging on some sample transcripts	2	Anoop and Jenna and Ammar
4	Create route in meeting processing module that returns array of tags	4	Anoop and Jenna and Ammar
5	Integrate with UI	6	Anoop and Jenna and Ammar

Completed:

At meeting end, topics are automatically created based on the transcript generated during the meeting. These topics are then inserted into the database as a field in the meeting model. Additionally, search query tokens beginning with '#' will search through topics in the database and return all meeting containing that topic.

User Story 3: I would like to search for meetings I have access to based on tags.

1	Integrate tags from meeting into meeting model in the database	4	Prashanth and Daniel
2	Include a filtering character (‘#’) to specify that search token is for tags	2	Prashanth and Daniel
3	Modify searching function to filter for tags	4	Prashanth and Daniel

Completed:

We have successfully implemented tags in for each meeting in our backend meeting model. We also used our search algorithm to apply for these tags that are stored in our database with ‘#’ as the activating character. URL is also now encoded to support ‘#’ in the URL.

User Story 4: I would like to integrate emailing services into Quillio, in which activation emails, password reset, and other email notifications are available.

1	Research Celery/SendGrid mailing services integration with flask and with mongoengine.	5	Jenna and Charlie
2	Integrate mailing service of choice into Quillio (create a general module for emailing)	3	Jenna and Charlie
3	Create email templates for user activation, password reset, and inviting users to Quillio.	4	Jenna and Charlie
4	Integrate email notification into existing code base to automatically execute when appropriate.	3	Jenna and Charlie
5	Add front end features for users to invite others to Quillio.	3	Jenna and Charlie

Completed:

After signing up, an email is sent with a unique link for the new user to activate their account. Password reset option is now available for users that forgot their password: an email is sent to the entered address (if that address exists in the Quillio database) with a unique link to access password reset. Upon resetting the password, the user is redirected to login. Once logged in, a user has the ability to send an invite to an email address that is not associated to a Quillio account.

User Story 5: I would like to tag important keywords in a given transcript of text. (Analyze for keywords and topics discussed).

1	The meeting transcript should be fed to a service that determines keywords.	6	Anoop and Charlie and Ammar

2	The generated Keywords should be associated with the meeting.	5	Anoop and Charlie and Ammar

Completed:

Integrated the RAKE-NLTK library as a route in the backend. When called and provided a meeting ID, the function associated with the route will generate tags from the transcript associated with that meeting and insert them back into that meeting document in the database.

User Story 6: I would like to be able to edit my meeting transcriptions.

1	Display transcripts in a meeting directory	8	Daniel and Prashanth
2	Allow users to edit transcript	8	Daniel and Prashanth
3	Save edits made by users to database	6	Daniel and Prashanth
4	Update tags based on edits made by users	8	Daniel and Prashanth

Completed:

We have successfully implemented a way to display our transcripts from our meetings using an editor named Quill. This editor allows to us to enable or disable the edit mode for the transcript. We can then save the new transcript and send it back to our tagging algorithm to update the tags.

User Story 7: I would like to have a user-friendly design that will let me efficiently navigate my meetings/workspace.

1	Update current meeting/group dashboards to link individual cards to their respective meetings (right now it does this, but it's really hacky).	10	Jenna and Prashanth

2	Create view that corresponds to a unique meeting URL.	5	Jenna and Prashanth

3	Clean up UI during navigation throughout the entire app.	6	Jenna and Prashanth

Completed:

During UI navigation, especially with user login and logout, the transitions between pages are more uniform in nature (no black screens during navigation as seen at the end of sprint one). A unique meeting url corresponds to a clean page in which transcripts are developed, as well as the transcription editing page.

User Story 8: I would like the user to provide keywords before each meeting.

1	Add parameter to GST call with keywords	4	Jenna and Anoop

2	Design UI element for user to specify keywords	6	Jenna and Anoop

3	Integrate with Meeting-In-Progress UI screen	5	Jenna and Anoop

Completed:

When a user joins a meeting, they are greeted with a modal to enter the keywords that will be used to guide GST while it processes the meeting speech. When the keywords are submitted, we send a request with the given keywords that will configure GST.

User Story 9: I would like to research how my meeting transcripts can be summarized with meeting highlights.

1	Research different summarization techniques	6	Ammar
2	Get article summarization working	8	Ammar
3	Try it with conversations	3	Ammar
4	Group portions of convo by topic then summarize	13	Ammar

Completed:

We did some research on different types of algorithms for summarization, and decided to go with the extraction based summary which uses a bunch of text heuristics extract sentences from a given text. We were able to use the Rake NLTK library to locate the keywords and phrases in a given text. We used this to rank sentences in the transcript and then returned the top two sentences.

Developer Story 10: I would like to clean up and refactor my code base.

1	Cleanup Code from Sprint 1	20	All team members

2	Sector code into readable functions.	20	All team members

3	Unification of fractured functions.	20	All team members

Completed:

We cleaned up the code from sprint one to be pep8 compliant and modularized.

What did not go well?

In general, we were unable to fully incorporate tests for every feature what was created this sprint. We did not try to test the various edge cases that may be present. However, we were able to do manual testing, but realized that it is not sufficient in testing all edge cases that our project may encounter.

Developer Story 11: I would like to run manual (or unit tests where applicable) on all appropriate features created during this sprint.

1	General Testing	10	All team members

Not Completed:

Tests for authentication were done, however, those were features implemented last sprint. Our finals tasks will be difficult to test with unit test, so we will have to manually test the edge cases that are present.

Developer Story 5:

3	Meeting model should be updated to include keywords (accessible by search algorithms)	4	Anoop and Charlie and Ammar

Not Completed:

We decided not to implement this feature because this would store redundant information in the database. The meeting keywords are only relevant for the speech to text API.

How should you improve?

Over the course of the second sprint, as a team, we did a good job improving our communication skills in general. We mainly solved this by increasing the frequency of meeting that we would have per week. However, during these meetings, we get distracted much too easily. This is a problem that is harder to address with our team than most other teams because we are very close to one another. We all have very similar schedules and interests, so we often end up talking about those interests or assignments for other classes during these meetings. This is definitely something that will be addressed.

Another issue that we had this sprint was that our standup posts have been very lacking in comparison to the first sprint. We generally decided that they weren't worth our time because we had a lot of other assignments and exams going on during the time that we had all decided to originally do daily standup (such as CS252 lecture, which all of us are currently enrolled in). To solve this problem, we should reschedule standup everyday to a time that works better for all of us.

During the sprint, we need to follow our sprint planning document more precisely. This means, people will complete stories that are not theirs and this will end up disrupting the evenly distributed workload. This is a problem that we had during the last sprint as well, through the problem got much worse this sprint. This also disrupted the trello board for the sprint which was designed around the sprint planning document. We can solve this by following and enforcing the sprint planning document more strictly.

On paper, our tasks were evenly distributed, but in practice, some members of the group had more difficult and time intensive tasks than others. Another issue with our sprint planning is the organization of tasks within a story. We plan to improve this by only allocating time for testing with certain user stories where it makes sense to test. We also plan to allocate sufficient time for front end development, which has routinely taken up a significant amount of time in comparison to the rest of the work we needed to do for this sprint.

Additionally, we can improve our time management skills as a group. Some of the research and set up overhead took up a significant amount of time during the first sprint, and it took away from the time and development efforts needed for the user stories we defined for the sprint. We will better manage and consider research and setup time in the future to avoid heavy workloads at the end of the sprint.