Team 7 - Sprint 1 Planning Document

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

Sprint Overview

During this sprint, we are going to focus on setting up the systems that will become the backbone of our application. Firstly, this involves setting up our development environments (Heroku, IDEs, Github, Trello, Python, and Node.js). Secondly, we have to set up our database schema in CassandraDB. We will also have to establish the models created in the database and corresponding models in the Flask backend. We will continue by creating actions and views for a properly created and authenticated user, as well as the authentication process. We will begin the preliminary steps of our major record, transcribe, and summarize algorithm by working with integrating the Google Speech to Text API into the functioning application for a user.

SCRUM Master: Jenna Ellis

Meeting Plan: Sundays 4:00-6:00PM, Tuesdays 4:15-5:30PM, Thursdays 4:15-5:30PM

Risks and Challenges

Some challenges we will face during this sprint will be oriented around initial development setup and group cohesiveness. A significant amount of time will likely be spent in establishing git protocols, code standards, and setting up development environments for each developer. Some risks we will face during this sprint will be with following through on the planned user stories and tasks. Given that this is the first step in actually coding and implementing the plans we have made so far, we anticipate running into some product structure issues that will require backtracking and significant group discussions; it is likely that a large amount of structural revisions will be necessary.

Current Sprint Detail

User Story 1:
As a user, I would like to create a Quillio account.

Task Table:

Task Number	Description	Time	Task Owner
1	Create User model in Flask Backend	2	Jenna
2	Create User tables in Database	2	Charlie
3	Create a signup page for to insert a new user into the system	3	Jenna
4	Tie the three above components together to complete user signup process (frontend to backend to DB)	3	Jenna
5	Unit test- correct and incorrect input, different types of login (Google, email, etc.)	2	Jenna

Acceptance Criteria:

- Given that the user creation process is functional, a user model can be instantiated with the desired information in the flask backend.
- Given the user creation process is functional, a user instance can successfully be inserted into the User table in the database.
- Given the user creation process is functional, a user can view a page to securely enter their information and save it in the Quillio system base.
- Given the user creation process is functional, a user’s information can be sent from the client, to the server, to the backend without data disruption or exploitation.
- Given the user creation process is functional, as a developer we should be able to test the input and different types of logins, and all unit tests should pass when expected.

User Story 2:
As a user, I would like to login to my Quillio account.

Task Table:

Task Number	Description	Time	Task Owner
1	Create a login page	2	Prashanth
2	Integrate flask authentication process	3	Charlie
3	Encrypt the password (highly sensitive information) before any transfers occur	2	Prashanth
4	Create and manage a user's session	2	Prashanth
5	Unit Test - correct and incorrect input, unknown email, incorrect password attempt limits	2	Prashanth

Acceptance Criteria:
- Given that user login and authentication is functional, a user can easily access the login page for quillio, and enter their login credentials.
- Given that there is integration with the flask is implemented, a user's login will be authenticated based on certain criteria like username/password.
- Given that user login and authentication is functional, a user's password will be encrypted before a login is considered a success or failure.
- Given that creating and managing a user's session is implemented, a user will be able to login to their dashboard.
- Given that a unit test has been created to test for incorrect input, then the user should be able to receive feedback

User Story 3:
As a user, I would like to have a clean user interface to see my organized meetings.

Task Table:

Task Number	Description	Time	Task Owner
1	Integrate decided color palette and follow UI mockups - in CSS global	2	Prashanth
2	Create index (home page) and navigation view for logged in user	3	Prashanth
3	Unit Test - Access to this page only after a user is logged in	2	Prashanth

Acceptance Criteria:

- Given that a logged in user interface is functional, a user should see a set group of colors and formatted
- Given that a logged in user interface is functional, a user should be able to see a navigation menu for their searching and informational needs and be able to easily find all components of the home page and easily understand functions.
- Given that a logged in user interface is functional, a user should only be able to access this page after logging in to our application

User Story 4:
As a user, I would like to create a new meeting directory.

Task Table:

Task Number	Description	Time	Task Owner
1	Create Meeting table in database	2	Charlie
2	Create Meeting model in flask backend	3	Jenna
3	Allow a logged in user to access meeting creation	2	Jenna
4	Create UI for meeting creation process	3	Jenna
5	Unit Test - valid and invalid input for meeting creation, authenticated and unauthenticated user access	2	Jenna

Acceptance Criteria:
- Given that meeting creation is functional, a meeting table should exist in the database, and new meeting information can be inserted in the table.
- Given that meeting creation is functional, a meeting object can be created and instantiated in the backend.
- Given that meeting creation is functional, a logged-in Quillio user can actively create a new meeting from the frontend.
- Given that the meeting creation is functional, a simple but robust form should appear to collect all data necessary to create a new meeting.
- Given that the meeting creation is functional, unit tests should be created to check limits and correctness to meeting creation, and all tests intended to succeed should be successful.

User Story 5:
As a user, I would like to add colleagues/academic peers to access a meeting directory.

Task Table:

Task Number	Description	Time	Task Owner
1	Search for other Quillio users via email or name	4	Daniel
2	Add ID of searched user to the meeting model being created/previously created	3	Daniel
3	Unit Tests - adding existing users, search for non existing users, limit number of members to a meeting	2	Daniel

Acceptance Criteria:

- Given that user database is functioning properly, searching for a user via email or name should return all users with specified email or name.
- Given that adding Quillio users to a meeting is functional, adding the found user should add them to the meeting directory.
- Given that adding Quillio users to a meeting is functional, only emails that are linked to a Quillio account can be added to a meeting (if the email is not found, it is invalid input).

User Story 6:
As a user, I would like my meetings to be recorded.

Task Table:

Task Number	Description	Time	Task Owner
1	Create Note table in database	2	Ammar
2	Create Note model in Flask backend	3	Ammar
3	Unit Test - single speaker recording, multiple, saving text with correct models, security access	2	Charlie

Acceptance Criteria:

- Given that early stage meeting recordings are functional, note information should be safely stored in the Note database table
- Given that early stage meeting recordings are functional, note objects can be created and instantiated in the flask backend.
- Given that early stage meeting recordings are functional, unit tests should be successful when testing the record action, the text save, and appropriate security access.

User Story 7:
As a user, I would like to be able to upload an audio file to be turned into a transcript.

Task Table:

Task Number	Description	Time	Task Owner
1	Design protocols to send audio file for processing	4	Anoop
2	Save data with corresponding model in database	4	Anoop
3	Integrate with meeting in progress (UI aspect)	4	Charlie
4	Unit Test - make sure audio file is readable from API output, securely saved in database with correct meeting	3	Anoop

Acceptance Criteria:

- Given that protocols to send audio file for processing exists, the Google Speech To Text (GST) API should receive an audio file through GST API commands and return a transcript.
- Given that an audio file or a meeting transcript are being inserted into the database, ensure that it is inserted into the database with the correct model.
- Uploaded files should be first processed (speech to text) and then summarized/analyzed with the same standards that a live meeting would.
- Given that audio files can be uploaded and turned into transcripts, unit tests should be able to detect any breakage in the code.

User Story 8:
As a user, I would like my meetings to be transcribed.

Task Table:

Task Number	Description	Time	Task Owner
1	Integrate Google Speech to Text API	4	Anoop
2	Connect GST-API to frontend	4	Ammar
3	Be able to input a sample audio file into GST to convert it to text	3	Anoop
4	Unit Test- uploading test files into GST to convert to text	2	Ammar

Acceptance Criteria:

- Given that the program has an audio file that it wants to convert into a text transcript, GST API should perform that conversion and return a transcript.
- Given that the program has an audio file to be converted to text, the user should be able to swiftly select this file or record their file and associate it with a meeting directory.
- Given that an audio file is sent to the GST API, the API should return a transcript of the audio file.
- Given that the transcription process is functional, all unit tests involving uploading files and analyzing the text output should be successful.

User Story 9:
As a user, I would like to be able to search for meetings.

Task Table:

Task Number	Description	Time	Task Owner
1	Create search bar on front end at user home	2	Charlie
2	Let user search through all accessed meetings by title	2	Charlie
3	Search all accessed meetings by meeting members	3	Charlie
4	Search all accessed meetings by date created and/or modified	3	Anoop
5	Unit Test - Expected input and output, search for meetings that don't exist , null input	2	Daniel

Acceptance Criteria:

- The user should be able to conveniently search through their meetings to find the desired one via the home page.
- The meetings should be searchable by title, and the results should be given within a reasonable time.
- Meetings should be queryable by the users who participated in the meeting.
- Given that meeting search and filter is functional, I should be able to search for meetings based on date of meeting or last time the transcription was modified.
- Given that meeting database is functional, I should not receive erroneous meetings for searches and filters, as confirmed by unit tests.

User Story 10:
As a user, I would like to create a reusable group for recurring meetings.

Task Table:

Task Number	Description	Time	Task Owner
1	Create Group database table, establish relations with other tables	3	Ammar
2	Create group model in flask backend	2	Ammar
3	Create front end option to create a group and add colleagues	3	Ammar
4	Unit test - create with correct input, incorrect input, restrict number of members in a group, creating without any input	3	Ammar

Acceptance Criteria:
- Given that groups can be created and reused, the database should take all pertinent information and insert it into the group table.
- Given that groups can be created and reused, the Flask backend should be able to instantiate group objects.
- Given that groups can be created and reused, the user should be able to easily follow a process of creating a group, which involves searching for other users.
- Given that groups can be created and reused, a group can only be created with appropriate input, can only be deleted by user permission, and is bug free, as verified by unit tests.

User Story 11:
As a member of a meeting group, I would like to be able to delegate an admin position to an individual in a given group.

Task Table:

Task Number	Description	Time	Task Owner
1	Update Meeting schema to include an owner attribute	2	Prashanth
2	Update model for meeting to designate an owner	3	Daniel
3	Add a field on front end to designate an owner	2	Daniel
4	Add checks to admin privileges to other portions in the UI	4	Daniel
5	Unit Test - valid and existing user, non existing user, unanswered field, restrictions on input	2	Prashanth

Acceptance Criteria:
- Given that the meeting schema is updated to include an owner and is functional, the front end should be able to access the owner of each meeting.
- Given that model for meeting in a group setting is functional, a user within a group should be able to be designated as an owner.
- Given that model for designating an owner is functional, a user can choose which user within the group will be designated as owner.
- Given that administrative delegation in a given group is functional, only admins should be able to access pages used for deleting, modifying a group.
- Given that the test is implemented correctly, it should be thorough and successful (pass). It should also indicate that the users are valid/invalid respectively.

User Story 12:
As a developer, I would like to produce thorough documentation at each step in the development process.

Task Table:

Task Number	Description	Time	Task Owner
1	Establish weekly code reviews and coding standards	12 (2 hours per person)	Anoop, Daniel, Jenna, Ammar, Prashanth, Charlie
2	Weekly Progress Reports	12 (2 hours per person)	Anoop, Daniel, Jenna, Ammar, Prashanth, Charlie
3	Planning Documents	12 (2 hours per person)	Anoop, Daniel, Jenna, Ammar, Prashanth, Charlie
4	Create thorough documentation as coding progresses	12 (2 hours per person)	Anoop, Daniel, Jenna, Ammar, Prashanth, Charlie

Acceptance Criteria:
- Given that each team member is assigned and delivers a reasonable amount of work, each member’s work should go through code review to ensure that coding standards are being kept.
- Given that each team member delivers work each week, each team member should also deliver a weekly progress report detailing the work they did that week.
- Given that each team member participates in each sprint, each team member should have a role in writing all sprint planning documents.

- Given that the code we write will eventually have to be read and understood by somebody other than the original author, each team member should write thorough documentation to help others understand their code.

User Story 13:
As a developer, I would like an organized work-flow that encourages team collaboration on all ends.

Task Table:

Task Number	Description	Time	Task Owner
1	Setup github account and git on your local machine	6 (1 hours per person)	Anoop, Daniel, Jenna, Ammar, Prashanth, Charlie
2	Setup key repos and add all members as admin	6 (1 hours per person)	Anoop, Daniel, Jenna, Ammar, Prashanth, Charlie
3	Set up github integrations for testing, trello, etc.	6 (1 hours per person)	Anoop, Daniel, Jenna, Ammar, Prashanth, Charlie
4	Set up deployment service, Heroku	6 (1 hours per person)	Anoop, Daniel, Jenna, Ammar, Prashanth, Charlie

Acceptance Criteria:
- Given that each team member has to have a development environment connected properly to the Quillio Github repository, ensure that each team member can push to and pull from the Github repository without error. Also, ensure that team members know how to use the Github to preclude merge conflicts and other Git-related disasters.
- Given that Quillio is split into many different parts, ensure that separate parts of the project have their own repositories.
- Given that our team will be using a variety of different programs to manage coding workflow (Github, Trello, PyUnit, etc.), ensure that each program is connected to each other correctly. For example, ensure that Github commits and pull requests are linked to their tickets in Trello.
- Given that different components of Quillio are going to be deployed separately, setup Heroku to correctly deploy each component of Quillio.

Remaining Product Backlog

As a user...

1. ~~I would like to be able to create an account in Quillio.~~
2. ~~I would like to login to my Quillio account.~~
3. ~~I would like to be able to create a new meeting directory~~
4. ~~I would like to add colleagues/academic peers to access a meeting directory.~~
5. I would like to be able to invite peers to use Quillio if they do not already have an account.
6. ~~I would like my meetings to be recorded~~
7. ~~I would like my meetings to be transcribed.~~
8. I would like my meeting transcriptions to be emailed to all members of the meeting.
9. I would like my meeting transcripts to be summarized with meeting highlights.
10. I would like my summaries to have rich text media and hyperlinks to team artifacts (if time allows).
11. I would like my meetings to be tagged based on meeting topics.
12. I would like to search meetings I have access to based on tags.
13. ~~I would like to search meetings based off of title, members, and date.~~
14. I would like to be able to access my meeting transcriptions and summaries in a web application.
15. I would like to be able to edit my meeting transcriptions and summaries.
16. I would like to share my meeting transcriptions and summaries with others.
17. I would like to view topics covered in the meeting based on predefined agenda.
18. ~~I would like to create a specialized group, for recurring meeting team figures.~~
19. I would like to provide some set objectives and agenda points before my meeting.
20. I would like said objectives to be analyzed with the transcripts after the meeting.
21. I would like to be notified if all objectives were covered in a given meeting.
22. I would like to specify a new meeting directory as professional or academic in nature (if time allows).
23. I would like to see my transcripts get better/more accurate the more I use the service (if time allows).
24. I would like to see my transcripts be developed as the meetings is happening (if time allows).
25. ~~I would like to be able to upload a pre-recorded audio file to be turned into a transcript.~~
26. I would like to have a clean user interface to see my organized meetings.
27. I would like all of my audio files and meeting transcripts to be stored in a database so that I can access them the next time I log in.

As a member of a meeting group...

1. I would like to be able to use Quillio for meetings for my company or my schoolwork.
2. In a corporate setting, I would like people in my company to be able to share meeting notes with their peers and superiors.
3. In a corporate setting, I would like supervisors to be able to access transcriptions and summaries of the supervisor’s employees.
4. I would like to track user contribution to each meeting.
5. I would like to view meeting productivity analytics.
6. ~~I would like to be able to delegate an admin position to an individual in a given group.~~
7. I would like to be able to reference meeting notes in presentations.
8. In an academic setting, I would like to be able to mark summary notes up based on their validity and relative importance after the automated summarization.
9. I would like to schedule recurring meetings to my preferred calendar service straight from Quillio.

As a developer…

1. ~~I would like an organized work-flow that encourages team collaboration on all ends.~~
2. ~~I would like to be able to track issues that are currently facing the development of the service.~~
3. I would like to see individual contributions to the project.
4. I would like to convert an audio file of speech to text
5. I would like to convert a stream of audio being sent to a program to text
6. I would like to groom converted speech to text and remove any unreadable language.
7. I would like to tag important keywords in a given transcript of text.
8. I would like to distinguish between speakers in a given audio file of speech.
9. I would like to train the NLP to get better the more recordings there are.
10. I would like to see certain usage/user statistics that help aid development.
11. I would like to securely store the user's’ information.
12. ~~I would like to produce thorough documentation at each step in the development process.~~