WhenToMeet

Project Backlog

Team 17
Ben Crumbacher, Seerat Dheer, Nicholas Fordyce, Nicholas Franz, Mihir Somani

Problem Statement

When working in a group project, it can often be challenging to coordinate what time works best for a meeting. Group members can have conflicting schedules, making it a pain to find one specific time that works best. Our application, WhenToMeet, will allow individual users to specify when they are available and easily see what times work for the entire group.

Background Information

Working in a team can be tedious and overwhelming when it comes to scheduling a meeting time that is suitable to everyone. Currently, platforms like GroupMe, Slack or Discord are being used to get consensus on mutual meeting times. These are awesome applications for exchanging texts but have their own limitations. With WhenToMeet, which is particularly suited for organizing meetings with consensus, teams can overcome these limitations and effectively plan when to meet next! Features like polling on location and time, invite links, ability to select and view times, and notifications make WhenToMeet unique and a perfect tool to accommodate the needs of every member. The targeted audience for the app are mostly college students who are working in teams or leading student organizations.

Environment

The frontend of the application will be written using ReactJS. We will use the React Material UI framework and use some of its premade components to style the majority of the application, and we will use CSS to style any components that need specific styles. For the backend we will create a RESTful API using NodeJS, with a serverless architecture hosted using AWS Lambda. The API routing will use Amazon API Gateway, and the database will be hosted with Amazon DynamoDB, a noSQL, non-relational, fully managed database service.

Functional Requirements

Backlog Id	Functional Requirement	Hours	Status
1	As a group leader, I would like to create a new meeting	6	Planned for Sprint 1
2	As a group leader, I would like to add a title to a meeting	4	Planned for Sprint 1
3	As a group leader, I would like to add a description to the meeting	4	Planned for Sprint 1
4	As a group leader, I would like to set the location of the meeting	4	Planned for Sprint 1
5	As a group leader, I would like to set up a poll to decide the location of the meeting	15	Planned for Sprint 1
6	As a group member, I would like to vote on the location of the meeting	10	Planned for Sprint 1
7	As a group leader, I would like to edit the details of the meeting retroactively	6	Planned for Sprint 1
8	As a user, I would like to leave comments about the meeting	12	Planned for Sprint 2
9	As a user, I would like to see the comments that other users post about the meeting	10	Planned for Sprint 2
10	As a user, I would like to have a url generated to invite other users to the meeting	15	Planned for Sprint 1
11	As a user, I would like to enter my name before selecting my times	4	Planned for Sprint 1
12	As a user, I would like to set the hours that I am available for a meeting in a timetable	12	Planned for Sprint 1
13	As a user, I would like to see what percentage of group members have picked certain times	15	Planned for Sprint 1
14	As a user, I would like to see a tooltip that tells me the names of the group members that picked certain times when I hover over the group timetable	10	Planned for Sprint 1
15	As a user, I would like for the times that work for everyone to be distinct from other times	4	Planned for Sprint 1

16	As a user, I would like for the days that have no times selected to be hidden from the group view	4	Planned for Sprint 1
17	As a group leader, I would like to select the time for the meeting once everyone has put in their hours	10	Planned for Sprint 1
18	As a group member, I would like to see what time is selected by the group leader	10	Planned for Sprint 1
19	As a group leader, I would like to change the selected meeting time if I select the wrong time	8	Planned for Sprint 1
20	As a user, I would like to upload files for the meeting	15	Planned for Sprint 2
21	As a user, I would like to remove files I uploaded for the meeting.	15	Planned for Sprint 2
22	As a user, I would like to see a panel for all the files for the meeting.	15	Planned for Sprint 2
23	As a user, I would like to access files other users uploaded for the meeting	15	Planned for Sprint 2
24	As a user, I would like to see the changes that other people make to their hours make in real time	15	Planned for Sprint 2
25	As a user, I would like for the website to be functional for different screen sizes	10	Planned for Sprint 2

Non-Functional Requirements

Architecture

The project will be hosted in AWS. The frontend code will be hosted in a web-enabled Amazon S3 bucket. The backend will use a serverless architecture. API requests from the frontend will route through Amazon API Gateway then sent to AWS Lambda for serverless compute and we will use Amazon DynamoDB for our database.

Security

The meeting id for the url as well as user ids will be hashed so that attackers are unable to guess a meeting url. Our app won't have a login or ask users to store personal data in order to mitigate the effects of a successful attack. We will also be using https to make sure we have a secure connection.

Usability

The user interface will be designed so that it is intuitive for users to use. It will also be appealing to look at. In order to do this, we will use the React Material UI framework to import styled components. Our application will also be able to adapt to different sized screens and resolutions.

Performance

The backend will be designed for fast response times on API requests, even at a large scale. These are the advantages of using serverless compute, and a noSQL database. At any scale, our backend will respond to API calls in under 300ms.

Use Cases

Case 1: Create a new meeting

	Action		System Response
1.	On the main page, click on ‘Create Meeting’	2.	Displays a page to enter the meeting details
3.	Enter meeting details	4.	Client sends an API request to the backend to store the details in database

Case 2: Add a title to a meeting

	Action		System Response
1.	On creating a new meeting’, click on the first input box with the label ‘Add Title’	2.	The input box is highlighted if selected
3.	Enter Title	4.	Client sends an API request to the backend to store the title in database
		5.	The input box displays the title

Case 3: Add a description to a meeting

	Action		System Response
1.	On creating a new meeting, click on the second input box with the label ‘Add Description’.’	2.	The input box is highlighted if selected
3.	Enter Description	4.	Client sends an API request to the backend to store the description in database
		5.	The input box displays the description

Case 4: I would like to set the location of the meeting

	Action		System Response
1.	On creating a meeting, click on the input box labelled ‘Location’	2.	The input box is highlighted if selected
3.	Enter a location	4.	Client sends an API request to the backend to store the location.
		5.	The input box displays the location

Case 5: I would like to set up a poll to decide the location of the meeting

	Action		System Response
1.	After adding the members, click on ‘Poll’ button to create a poll	2.	A dialog box appears containing a form.
3.	Enter the choices of the poll in input boxes (eg. Different locations)	4.	Client sends an API request to the backend to store the choices and track and store the number of times it is chosen
5.	Click ‘Create’	5.	A question with different clickable answers/choices is displayed next to the location

Case 6: As a group member, I would like to vote on the location of the meeting

	Action		System Response
1.	Select desired meeting location and click ‘vote’	2.	Client sends API request to the backend to record the vote in the database
		3.	Backend responds with a list of votes
4.	Client displays a summary of votes		

Case 7: As a group leader, I would like to edit the details of the meeting retroactively

	Action		System Response
1.	Click on ‘edit’ button to edit the details of the meeting	2.	The details input page appears
3.	Edit the details of the meeting in the input boxes		

4.	Click on ‘save’ or ‘cancel’ button	5.	If ‘save’ button is clicked, send update API call to update data in the database

Case 8: As a user, I would like to leave comments about the meeting

	Action		System Response
1.	Enter comment into text input box and click ‘send’	2.	Client sends API request to the backend to store the comment in the database
		3.	Client clears text input box

Case 9: As a user, I would like to see the comments that other users post about the meeting

	Action		System Response
1.	User loads meeting page	2.	Client sends API request to the backend to retrieve comments
		3.	Client displays comments, sorting by time sent.

Case 10: As a user, I would like to have a url generated to invite other users to the meeting

	Action		System Response
1.	User creates new meeting	2.	Client sends API request to the backend to generate unique url
		3.	Backend responds with unique url
		4.	Client displays url on meeting page

Case 11: Enter a name before selecting times

	Action		System Response
1.	User clicks on invite link	2.	User is redirected to the meeting page and prompted to enter their name
3.	User fills out a valid name		
4.	User presses “Enter”	5.	Client sends an API request to an endpoint that adds the user to the

			meeting
		6.	A new user is added to the meeting in the database
		7.	A user id is stored in the browser's local storage for future authentication

Case 12: Set the hours for a meeting in a timetable

	Action		System Response
1.	A user clicks on an unselected square on their timetable	2.	The square turns from empty to filled to show that it is selected
		3.	Client sends an API request to update that user's times
		4.	The timetable showing all available user's times is updated to show the new times

Case 13: See what percentage of group members have picked certain times

	Action		System Response
1.	A user loads the meeting page	2.	Client sends an API request to get the available times other users have entered for that meeting
		3.	Backend sends an API response with all of the group member's available times for that meeting
		4.	For each square in the group timetable, the client calculates the percentage of group members who selected that time, and fills in the square an amount equal to the percentage

Case 14: See a tooltip that shows the names of the group members that picked certain times when the group timetable is hovered over

	Action		System Response
1.	A user hovers over a square on the group timetable	2.	Client finds which users selected the time interval the mouse is hovering over

		3.	Client displays a tooltip showing what percentage of users selected that time, and who selected it

Case 15: The times that work for everyone will be distinct from other times

	Action		System Response
1.	A user loads the meeting page	2.	Client sends an API request to get the available times other users have entered for that meeting
		3.	Backend sends an API response with all of the group member's available times for that meeting
		4.	For each square on the timetable that 100% of group members selected, that square will be distinctly styled

Case 16: The time table will hide days that have no selected times.

	Action		System Response
1.	A user loads the meeting page	2.	Client sends an API request to get the available times
		3.	Client loads the data for each day, and checks if a day does not have data. Client hides days with no times

Case 17: Selecting an official time of the meeting

	Action		System Response
1.	User loads the meeting page	2.	Client sends an API request to check if the user is the group leader
3.	User clicks a time and confirms it as the selected time	4.	Client sends the time to the backend.
		5.	Backend confirms the time and returns the data necessary for the meeting. Client page changes so that the information shown is the time of the

			meeting and the details of the meeting.

Case 18: Seeing the official time for the meeting

	Action		System Response
1.	User loads the meeting page	2.	Client sends the request to the backend, where the backend will check to see if the meeting has been confirmed for a time
		3.	If the meeting has been confirmed, the backend will send a response to the client to display a page that will show the time of the meeting and the details of the meeting. If the meeting is not confirmed, it will display the time table.

Case 19: Changing the selected time of the meeting

	Action		System Response
1.	User loads the meeting page	2.	Client sends an API request to check if the user is the group leader.
4.	User clicks the edit meeting button	3.	If the meeting time is confirmed and the user is the group leader, the response will display an edit meeting button
5.	User chooses new time for meeting	6.	Client submits the time to the backend, where the confirmed time is changed. The backend responds with the new confirmed time being displayed on the page.

Case 20: As a user, I would like to upload files for the meeting

	Action		System Response
1.	User loads the meeting page.	2.	Client sends an API request to get the details of the meeting
3.	User clicks “Files” tab in the header	4.	Client sends an API request to get any existing files associated with the meeting and displays it. Backend responds with a response to display the

			files page.
5.	User clicks “Upload file” button on the page. A file selector tool will display, where upon the user will select the file. The user will then click the “Upload” button.	6.	Client sends the file and it’s data to an S3 bucket. Client also sends the information to the backend, where the backend maps it to a meeting. Backend sends a response to the client to notify that the file has been added to the meeting, and the file is then displayed on the “files” page.

Case 21: Remove uploaded files for the meeting

	Action		System Response
1.	User loads the meeting page.	2.	Client sends an API request to get the details of the meeting
3.	User clicks “Files” tab in the header	4.	Client sends an API request to get any existing files associated with the meeting and displays it. Backend responds with a response to display the files page.
5.	User clicks “Remove file” button on the page for a given file. A confirmation dialog box will display, where the user can either confirm or deny the action.	6.	Client sends an API request to remove the selected file to remove the file from S3 bucket. Backend also removes the file from display on the “files” page.

Case 22: Display uploaded files for the meeting

	Action		System Response
1.	User loads the meeting page.	2.	Client sends an API request to get the details of the meeting
3.	User clicks “Files” tab in the header	4.	Client sends an API request to get any existing files associated with the meeting and displays it. Backend responds with a response to display the files page.

Case 23: Access uploaded files for the meeting

	Action		System Response

1.	User loads the meeting page.	2.	Client sends an API request to get the details of the meeting
3.	User clicks “Files” tab in the header	4.	Client sends an API request to get any existing files associated with the meeting and displays it. Backend responds with a response to display the files page.
5.	User clicks “Download file” button on the page for a given file.	6.	Client sends an API request to download the associated file. Backend responds with a response with the data of the file to the browser.

Case 24: Hours updated in real-time for team members

	Action		System Response
1.	User loads the meeting page.	2.	Client sends an API request to get the details of the meeting
3.	User updates the selected meeting time.	4.	Client sends an API request to the server which triggers all the members of the meeting to update their DOM tree.

Case 25: As a user, I would like for the website to be functional for different screen sizes

	Action		System Response
1.	User loads the meeting page.	2.	Client sends an API request to get the details of the meeting
3.	User resizes the window of the browser.	4.	Layout of the components are rearranged accordingly.