Task

* [Direction](https://careerfoundry.com/en/course/full-stack-immersion/exercise/communication-features#directions)
* [Submission History](https://careerfoundry.com/en/course/full-stack-immersion/exercise/communication-features#step_submission_history)

 Estimated Task Time: 1 - 3 hours.

In the first part of the task, you’ll add two new communication features to your chat app, giving users the ability to send images (chosen from their device’s image library or taken with the camera) and share their current location. You’re also going to store users’ images in Google Cloud Storage so that other chat participants can view them.

**Part 1: Communication Features**

1. Open up the file for your chat screen and add a button into the chat’s input field that opens up an ActionSheet.
2. Create storage in Google Firebase for your media files and connect your app to this storage using getStorage().
3. Create a custom ActionSheet:
   * Create a new CustomActions.js component.
   * in Chat.js, create a new function returning the CustomActions component with all necessary props, then use the function in Gifted Chat’s renderActions prop.
   * In CustomActions.js, create and initialize a new ActionSheet that includes four options: “Select an image from library,” “Take a photo,” “Share location,” and “Cancel.”
4. Create a function that lets the user pick an image from the library:
   * Ask the user for permission.
   * Use Expo’s ImagePicker.launchImageLibraryAsync API.
   * Generate a unique reference string for the picked image.
   * Upload and send the picked image to the Firebase Storage using the reference you’ve just created.
   * Use Gifted Chat’s onSend() method to send a message containing the picked image.
   * Integrate the function into the custom ActionSheet.
5. Create a function that lets users take a picture with the camera:
   * Ask for permission.
   * Use Expo’s ImagePicker.launchCameraAsync API.
   * Generate a unique reference string for the photo.
   * Upload and send the photo to the Firebase Storage using the reference you’ve just created.
   * Use Gifted Chat’s onSend() method to send a message containing the taken photo.
   * Integrate the function into the Custom ActionSheet.
6. Create a function for retrieving the device’s geolocation.
   * Ask the user for permission.
   * Use Location.getCurrentPositionAsync to read the location data.
   * Use Gifted Chat’s onSend() method to send a message containing the location data.
   * Integrate the function into the custom ActionSheet.
   * Create a custom message bubble to render a MapView (hint: customize Gifted Chat’s renderCustomView prop).
7. Make the button in the input field (that opens the ActionSheet) accessible:
   * Use the React Native Accessibility features you first learned in [Exercise 5.2](https://careerfoundry.com/en/course/full-stack-immersion/exercise/chat-ui-accessibility#accessibility-for-native-mobile-applications) on the CustomActions’s TouchableOpacity.
8. Update the GitHub repository of your chat app.
9. Using Android Studio, create a short recording and share a link or upload the file here. You’ll need your physical device and the Android emulator to record the following actions:
   * Entering the chat screen in your app;
   * Picking an image from the library and sending it;
   * Taking a photo with the device’s camera and sending it (your Android emulator will fake a room or something similar to take a picture of);
   * Sharing your geolocation;
   * Receiving an image from another user (take a photo with your physical device and send it to yourself);
   * Showing that the three messages you’ve just sent (one with the picked image, one with the taken photo, and one with the shared location) are present in the Firestore database;
   * Showing that the two images (the picked image and the taken photo) are present in the Firebase Storage.

Part 1 Bonus: Audio Features  
If you’d like to provide even more communications features to your users, why not add an audio feature? To achieve this, you’ll need to implement the following into your project:

* The ability to record and play sounds.
* The ability to upload the audio file.
* The ability to send the audio file.

You’ll need to integrate the audio feature with Gifted Chat by adding a new option in the ActionSheet.

You can find instructions to help you get started in the [5.5 Bonus: Audio Communication Feature PDF](https://coach-courses-us.s3.amazonaws.com/public/courses/fullstack-immersion/A5/5.5-bonus/5.5-Bonus_Audio_Communication_Feature.pdf)

Be sure to update your GitHub repo and submit your Bonus task along with your main submission for this Exercise.

**Part 2: Documentation**

1. Double-check that your code is properly formatted and indented, and make sure that your code comments are effective and efficient. There’s no need to comment on everything—just focus on the parts where you think it could be cumbersome to figure out what the code is doing. As you know by now, well-commented code is impressive to employers as it demonstrates that you’re mindful of others, which is a key characteristic for good collaborators!
2. Check and update (if necessary) the README.md file in your chat repository in GitHub.
3. Make sure you’ve included all the necessary steps someone would have to follow to set up your chat app:
   * Setting up the development environment (Expo, Android Studio, etc.);
   * Database configuration (which one, where to put database credentials, etc.);
   * Necessary libraries to install.
4. Test that you’ve included all the necessary information. To do so, start by cloning the repo into an entirely new folder on your computer, then:
   * Follow your setup description;
   * Start your project and see if everything’s working.
   * If not, find the missing step and add it to your documentation.
5. Be sure to commit the final README.md file to your chat app GitHub repo and submit the link here for your mentor to review.

Prepare Your Portfolio  
At the end of the program, you’ll prepare and submit your final portfolio, so collecting your project deliverables and reflecting on your work **as you go** will save a ton of time. Remember: you don’t have to submit this to your mentor until the **end of the program**.

If you haven’t already, start a running text document and add reflections on your project for this Achievement, including the following:

1) A brief description of the project. You can use [the project brief](https://images.careerfoundry.com/public/courses/fullstack-immersion/full-stack-project-briefs/A5-Project-Brief-Mar2023.pdf) or your README.md file as inspiration.  
In your description, try to reflect on your work by answering the following questions (you don’t need too much detail; the key points are fine):

* What was your role in this project, and what tasks did you face?
* What decisions did you take and why? What were the consequences?
* If you could, what would you do differently?
* What lessons did you learn during this project?

2) A screenshot to represent the project.

3) A link to the project’s GitHub repository.

4) A link to the live, hosted version of your app (if possible). If you don’t have a live version, include screenshots that show the app’s functionality or a recording of your app in use.

5) A list of the technologies used for each project (React, CSS, etc.). You can pull this information from your README.md file.

6) Any other relevant materials you created for the project; for example, user flows, user stories, or a Kanban board. Be sure to explain how you worked with these materials during the project.