

Frontend Proposal

1. Why We Chose the Web App + Slack Application Setup

2. Basic Functionalities

- Allow for translation of user inputted emoji filled text to pure text interpretation while maintaining intended meaning
 - Users who previously did not understand another's emoji filled language now can understand what they mean
 - Users who need to update their emoji-filled dialect to a different tone (potentially more professional)
- Allow for translation of user inputted pure text to text with emojis while maintaining intended meaning
 - Users who want to make their textual messages more "fun" or "expressive"
- RAIL Compliance
 - All user input acknowledged within 100ms
 - User knows that their request is received immediately and keeps them engaged
 - Any animations must have 60+ fps
 - Avoids any accessibility concerns from janky stop and start animations
 - Any translations/content must be delivered to the user in less than 5 seconds
 - People do not wish to wait
 - Keeps users engaged and does not create an expectation of the user to devote time to waiting for this service which limits this service's ability to serve them
- Feedback form easily available to user that can be digested by our backend to improve translations
 - Users should be able to give feedback to our backend so that if there are issues with translation, especially recurrent issues, these can be remedied to improve user experience. This may also be used to encourage good translations.

Web-app Specific

- RWD (responsive web design)
 - Promotes accessibility. Users should be able to use webapp regardless of device.
 - In particular, the phone is one of the most common devices used to communicate with others in a written form so our app should minimize friction that can come with the vastly different screen sizes between phones and computers.

Slack Application Specific

- Allow translation of existing messages via slack actions

- Allow user to get the translated interpretation of existing messages without the need to copy paste into separate translator therefore reducing number of clicks and friction
- Allow translation directly in messages via slack commands
 - Allow user to translate their message to chosen form in one message without the need to copy paste into separate translator, reducing the number of actions they would otherwise take to message in their chosen form

Optional/Low Priority Functionality

- Allow user to select style of translation (whether the translation should be more expressive or more literal or include more modern day slang)
 - User can better customize their translated response, increases likelihood that user gets translation result that they want and improves ability to communicate with different parties and specify tone
- Allow translator to take in more contextual information
 - In the form of allowing optional input of additional requests
 - Better user customization, increased likelihood of desired result, allows very specific requests from user and better understanding of desired tone
 - In the form of allowing user to input local context
 - Conversational context can help translator better understand the tone attempting to translate from or translate into
 - In the form of access to local conversation (Slack)
 - Major con: Privacy concerns especially because not all parties can provide consent for their messages being used
 - Would allow translator to better understand the tone attempting to translate from or translate into without user needing to manually input
 - In the form of country-based context
 - Increasing understanding of translation and unlock region specific meaning
- Allow user to see their translation history
 - Increase transparency and allow user to revisit old translation
 - Could also allow user to better learn how to translate themselves by seeing their translations (like flashcards)
- Allow user to specify their backend method in advanced settings
 - Increases user agency
 - If some translation types work better with some models than other or if response times are better on different backend methods allow user to decide
- Provide the user achievements and reveal their statistics of how they use the translator
 - Increases user engagement with service as they attempt to get achievements
 - Increases sociability as statistics and achievements can be compared between friends
- Translate modern day slang along with emojis

- Expands scope to similar use case of modern day slang which is used by similar parties as emojis
- Custom emoji support
 - Custom emojis are a part of slack workspace so attempt to provide context for these via their label. Better adaptation to the slack environment

3. Expected Tech Stack

Web-app

Framework: React.js

Slack Application

Our Slack App is divided into two main components: Frontend and Handler.

a. Slack Frontend

The “frontend” of a Slack App refers to the user-facing interface inside Slack, such as:

- Home Tab
- Modals (popup forms)
- Message Blocks (buttons, menus, interactive layouts)
- Slash Commands input box

Most of the frontend configuration, including app name, commands, permissions (OAuth scopes), and event subscriptions, can be managed directly through **Slack's official App Management UI**.

The screenshot shows the Slack API Basic Information page. The left sidebar has a dropdown menu with 'Test' selected, followed by sections for 'Settings' (with 'Basic Information' highlighted in blue) and 'Features'. Under 'Features', there are links for 'App Home', 'Agents & AI Apps (NEW)', 'Work Object Previews (NEW)', 'Workflow Steps (NEW)', 'Org Level Apps', 'Incoming Webhooks', 'Interactivity & Shortcuts', 'Slash Commands', 'Steps from Apps (LEGACY)', 'OAuth & Permissions', and 'Event Subscriptions'. The main content area is titled 'Basic Information' and contains a 'App Credentials' section. It explains that these credentials allow the app to access the Slack API and cautions against sharing them. It lists the 'App ID' (A09Q6DLR604), 'Date of App Creation' (October 31, 2025), 'Client ID' (9677578791664.9822462856004), and 'Client Secret' (redacted). Below the Client Secret, it says: 'You'll need to send this secret along with your client ID when making your oauth.v2.access request.' There is also a 'Signing Secret' section with a redacted value, a 'Show' button, and a 'Regenerate' button. At the bottom, it says: 'Slack signs the requests we send you using this secret. Confirm that each request comes from Slack by verifying its unique signature.'

For UI design, we use **Slack Block Kit**, which defines the visual layout of messages and modals in JSON format.

b. Slack Handler

The “Handler” is the backend logic that processes interaction and events from slack, it receives HTTP requests from Slack (for commands, button clicks, modal submissions, etc.) and sends responses back to Slack using its Web API.

We implement the handler using **Node.js**, **JavaScript**, and the official **@slack/bolt** framework.

Testing the feasibility of Slack Application

To validate the feasibility of adopting the Slack application, we conducted some basic experiments before the project officially started.

We’re currently planning to start from **Slack App Home**, **modals**, and **slash commands** as our main focus.

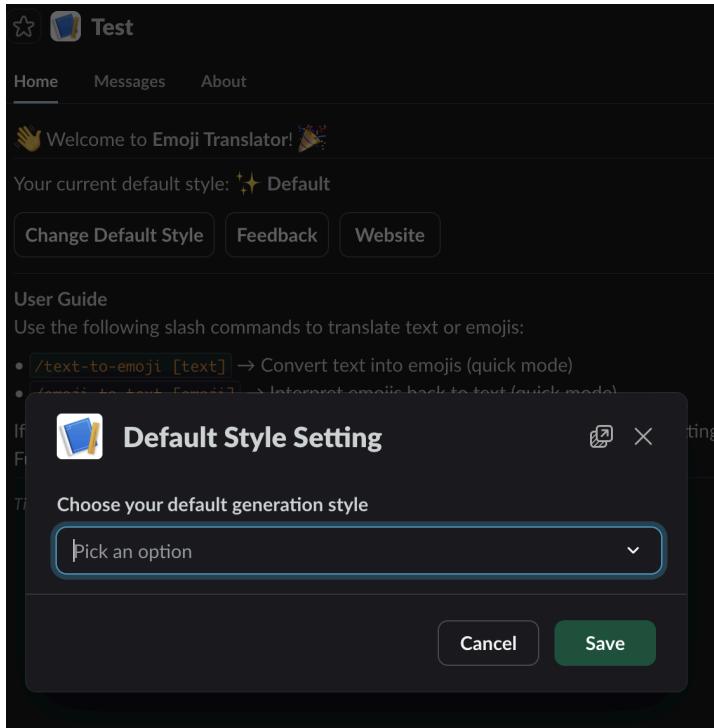
- Modals can be used for style configurations or any other features we may want to add in the future.
- The Slack App Home can serve as a mini web page to introduce how to use the Slack commands, it can also embed modals as default users settings or act as a user guide.
- As for Slack commands, they serve as the main way for users to interact with the backend.

Please note that everything below is just for testing purposes, it doesn’t represent the actual features of the final product.

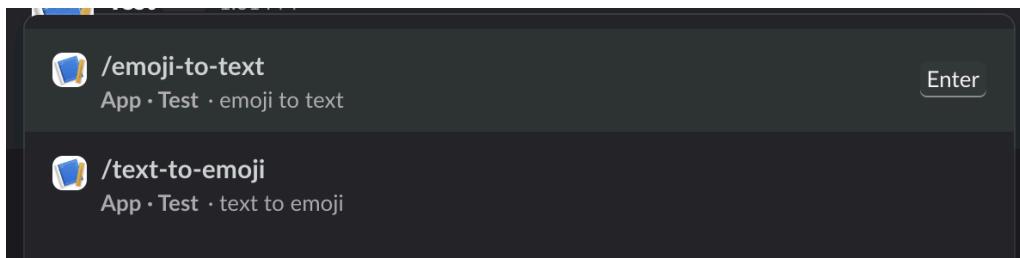
1. Slack App Home

The screenshot shows the Slack App Home interface for the "Emoji Translator" app. At the top, there are icons for a user, a file, and a gear, followed by the app name "Test". Below the header is a navigation bar with "Home" (which is underlined), "Messages", and "About". On the far right of the header is a bell icon. The main content area starts with a welcome message: "Welcome to Emoji Translator! 🎉". It says "Your current default style: 🌟 Default". There are three buttons at the bottom of this section: "Change Default Style", "Feedback", and "Website". Below this is a "User Guide" section with instructions on how to use slash commands to translate text or emojis. It lists two commands: "/text-to-emoji [text]" (Convert text into emojis) and "/emoji-to-text [emoji]" (Interpret emojis back into text). It also notes that if no arguments are provided, an interactive modal will appear to choose a style like "Default", "Funny", or "Formal". A tip at the bottom suggests returning to the App Home tab for settings and help.

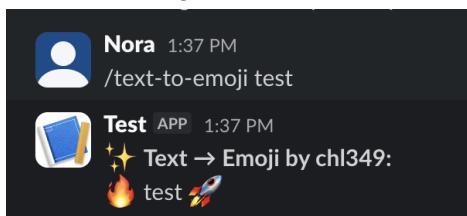
2. Modal in Slack App Home - can be used to configure certain default settings



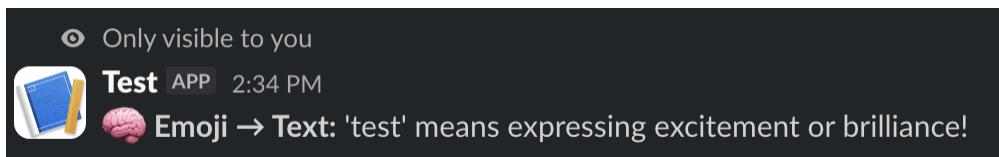
3. Slash Command



Public message in channel:

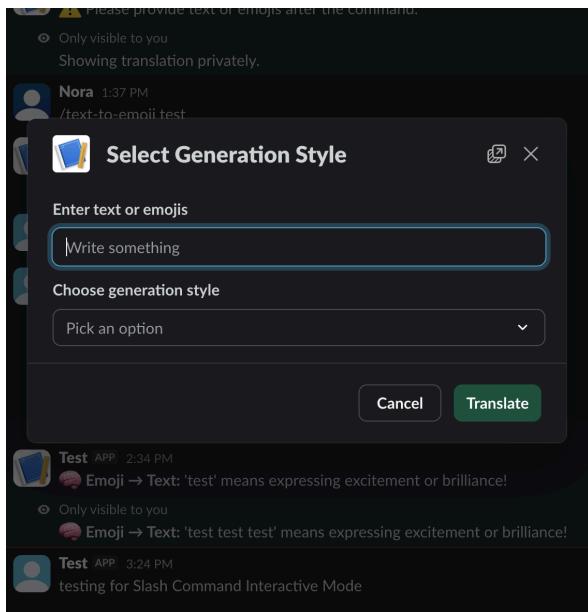


Private message in channel:

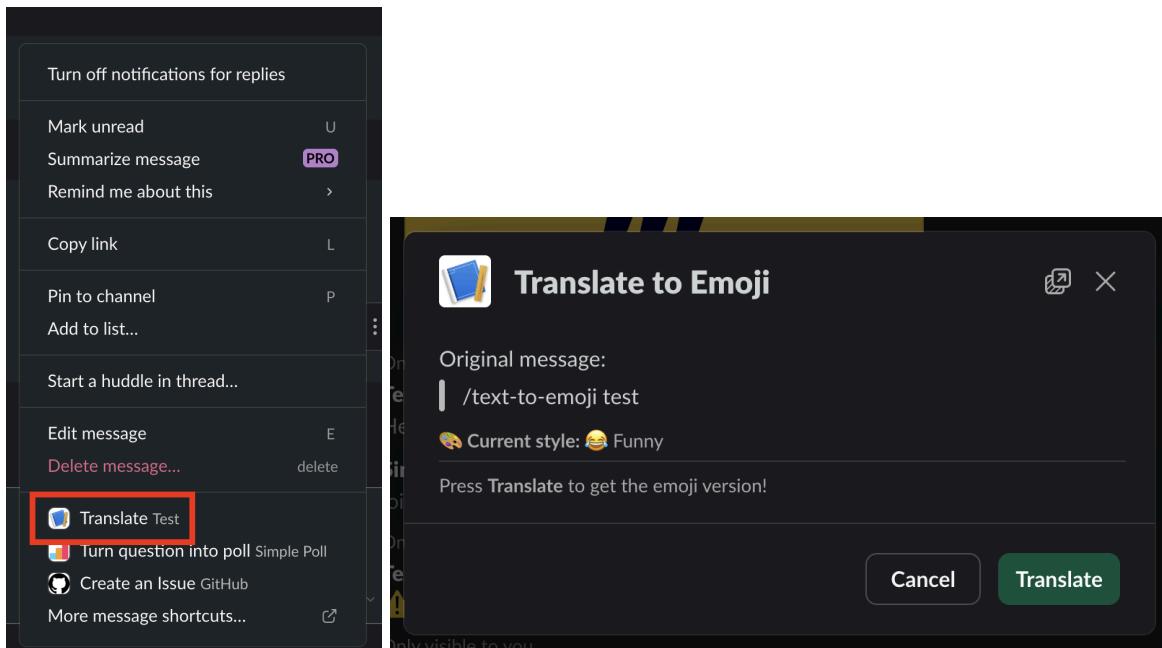


The message can be either visible to everyone or only to the user, so we need to clarify the user story and user flow first to determine the development details.

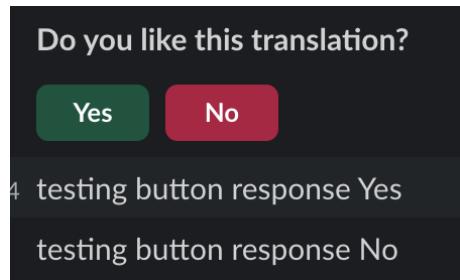
4. Modal in channel - If the user doesn't want to use the default values set in the App Home when entering a slash command, they can instead open a settings modal after typing the command.



5. Shortcut button for user message - this feature allows users to directly translate messages within a channel.



6. Something that might be useful - feedback interaction



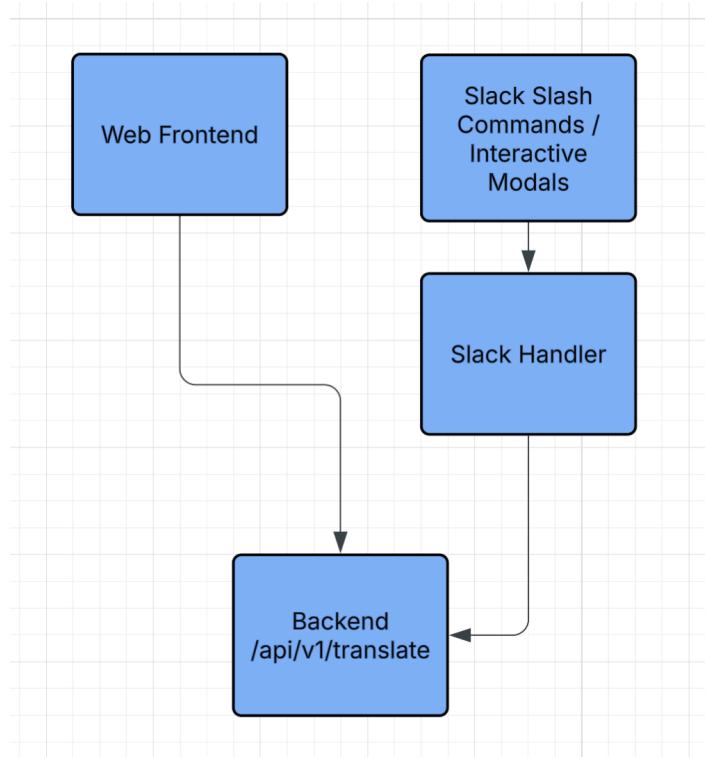
Official Slack Developer Documentation

For more details on Slack Application development, refer to the documentation below:

- [Slack Web API](#)
- [Slash Command](#)
- [Modals](#)
- [App Home](#)
- [Shortcuts](#)

Architecture Overview

Take the Backend API `/api/v1/translate` for example:



1. Frontend Layer
 - Include Web frontend and Slack frontend
2. Integration Layer
 - Acts as a bridge between the Slack platform and the backend translation service.

- Verifies Slack request signatures.
- Converts Slack command or modal input into the backend API request format.

3. Backend Layer

4. Data Layer

The frontend team will be responsible for maintaining the frontend layer and Integration layer. As for Backend and Data layers, they will not be covered in detail here.

In the current design, the web app is intended for general users, while the Slack app serves as an auxiliary tool for convenience.

Although both share the same backend server, there is no direct information exchange between them.

4. Potential Risks of the Current Approach

- API Latency:
 - Slack requires responses within **3 seconds** for slash commands; delays could cause timeout errors.
 - Webapp may need caching or asynchronous processing to ensure RAIL performance.
- Slack Platform Limitations:
 - Limited UI flexibility (cannot freely design interfaces).
 - Rate limits and permission scopes may restrict certain workflows.
 - We intend for users to add this application to their own channels, however this requires the app to be officially published on the Slack App Directory, and the review process may take several weeks. It's unlikely that the app will be approved and published before the project deadline.
 - For MVP, the app can only be installed and tested / used within the internal workspace.