**Maintenance Guide**

**System Dependencies:**

We will start by discussing the crucial dependencies required for both the client-side (PReact web application) and server-side components of our project, outlining the key libraries and tools used in each environment.

**Frontend:**

The client-side of our application relies on the following essential libraries:

* **React**: A JavaScript library for building user interfaces. (Version: ^18.2.0)
* **React DOM**: Provides DOM-specific methods for React. (Version: ^18.2.0)
* **React Router DOM**: Declarative routing for React. (Version: ^6.22.0)
* **Tailwind CSS**: A utility-first CSS framework for rapid UI development. (Version: ^3.4.1)
* **SweetAlert2**: JavaScript library for creating customizable elegant and responsive alerts. (Version: ^11.10.5)
* **heroicons/react**: A set of free MIT-licensed high-quality SVG icons for React. (Version: ^2.1.4)
* **preact/compat**: Preact's compatibility layer that allows you to use React code with Preact. (Version: ^17.1.2)
* **Axios**: Promise-based HTTP client for making requests to APIs. (Version: ^1.7.2)
* **PeerJS**: A simple peer-to-peer connection library for video, audio, and data. (Version: ^1.3.2)
* **Preact**: A fast, 3kB alternative to React with the same modern API. (Version: ^10.23.1)
* **Cobe**: A lightweight library for creating animated 3D graphics in React. (Version: ^0.6.3)
* **Web Vitals**: A set of utilities for measuring the performance of your web application. (Version: ^2.1.4)

**Backend:**

The following libraries, along with Node.js, are essential for running the server-side of our application:

* **Node.js**: A JavaScript runtime environment that executes server-side code, allowing for building scalable network applications.
* **google-cloud/speech**: A client library for Google Cloud's Speech-to-Text API, enabling speech recognition capabilities. (Version: ^6.6.1)
* **Axios**: A promise-based HTTP client for making API requests, facilitating communication between the server and other services. (Version: ^1.7.2)
* **Body-Parser**: Middleware that parses incoming request bodies before handlers, making data available under the req.body property. (Version: ^1.20.2)
* **Dotenv**: Loads environment variables from a .env file into process.env, ensuring secure management of sensitive configuration data. (Version: ^16.4.5)
* **Express**: A fast, unopinionated web framework for Node.js used to build and manage the server, handling HTTP requests and responses. (Version: ^4.19.2)
* **Peer**: A simple server for managing peer-to-peer connections using WebRTC. (Version: ^1.0.2)
* **WS (WebSocket)**: A simple WebSocket implementation for Node.js, enabling real-time, bi-directional communication between the server and clients. (Version: ^8.18.0)
* **Nodemon**: A development tool that automatically restarts the server when file changes are detected, improving development efficiency. (Version: ^3.1.4)

The complete list of dependencies can be located within the package.json file of each environment (frontend, backend).

**Managing Tools and Subscriptions of the Project**

In this section, we provide an overview of the tools, APIs, and services used in our project, highlighting their functionalities, usage costs, and maintenance requirements.

**Google Speech-to-Text API**  
The Google Speech-to-Text API is crucial for converting spoken language into text in real time. It supports a wide range of languages, enabling accurate transcriptions that form the foundation of our translation process. The API operates on a pay-per-use model, charging based on the amount of audio processed. Regular monitoring of usage is necessary to manage costs and maintain access to this essential service.

**Google Translate API**  
The Google Translate API provides fast and reliable text translation between the input and output languages selected by users. It supports numerous languages, facilitating communication across different languages. The service incurs costs based on the number of characters translated, making it important to track usage to control expenses as the project scales.

**Google Text-to-Speech API**  
The Google Text-to-Speech API is used to convert translated text back into spoken audio, enhancing the interactivity and accessibility of the application. This API allows users to hear translations in a natural-sounding voice, making communication smoother. Usage costs are incurred based on the number of characters converted to speech, so it is important to monitor this to avoid unexpected charges.

**OpenAI API (GPT-4 Model)**  
Our project leverages the OpenAI GPT-4 model to enhance translation quality with advanced natural language processing. GPT-4 offers context-aware and nuanced translations, improving the overall user experience beyond simple text conversion. The API operates on a token-based pricing model, where each request consumes a number of tokens based on the length and complexity of the input and output. Monitoring token usage is essential to manage costs effectively, as excessive usage can lead to increased expenses.

**PeerJS (WebRTC)**  
PeerJS facilitates real-time, peer-to-peer communication between users during live video and voice translation sessions. This technology enhances privacy by establishing direct connections without relying on centralized servers. While PeerJS itself is free, maintaining and managing server connections efficiently is critical, especially as the number of users grows.

**Gmail Account**  
Our project uses the Gmail account sonicciphercontact@gmail.com for receiving user messages and feedback via the 'Contact Us' option. It is important to keep this account active and regularly check notifications to ensure effective communication with users. This account can also be used in the future to send updates, announcements, and feature information to enhance user engagement.