FOR ENGLISH TEACHERS

AI ディベート練習アプリ

設置マニュアル

2 つの作り方 —

「配布版を使う」／「AI とゼロから作る」

生徒が AI を相手に、英語ディベートの「反論・応答・ジャッジ・要約」をくり返し練習できる Web アプリ。このマニュアルは、そのアプリを先生ご自身の環境に設置し、生徒に配れるようにするまでを、一歩ずつ案内します。

> **このマニュアルの約束**
>
> むずかしい専門用語には、その場で短い説明を付けました。それでも詰まったときのために、各所に**「AI にこう聞いてみよう」という質問例**を用意しています。画面の前に ChatGPT・Claude・Gemini などを一つ開き、**わからないところはコピーして貼りながら**読み進めてください。それが、いちばん速くて確実です。
>
> 利用上のお願い：このサンプルは現状のまま提供されます。公開・授業利用・費用・安全・校内手続は、各自の判断と責任で確認してください。

〈発行〉debate-class.com

改訂版 v0.4（2026 年 7 月）

作成: 小林良裕

0　はじめに ― 全体像と、2 つの作り方

細かい手順に入る前に、「何を・なぜ・どの順で」やるのかを先につかんでおくと、途中で迷いにくくなります。最初の数ページは、その地図です。

このアプリでできること

- **AI が相手・コーチになる**ディベート練習アプリです。生徒はブラウザで開いて練習します。
- 4 つの練習モード ― **Attack（反論する）／Defend（質疑に応答する）／Judge（試合を判定する）／Summary（要約して反論する）**。
- マイクに英語で話すと文字に起こされ、AI が応答し、最後に**日本語の講評と A〜E の評価**を返します。
- 先生が決めた合言葉をサーバーが確認し、一致した生徒だけが練習画面へ進めます。

> **ゴールは「1 本の URL」を作ること**
>
> 設置が終わると、たとえば https://あなたのアプリ名.onrender.com のような URL が手に入ります。これを生徒に配れば、その日から練習に使えます。

作り方は 2 通り。どちらもゴールは同じ

作り方	内容	向いている人
その 1 配布版を使う	完成済みのコードを入手して設置する。プログラムは書かない。最短で「動く URL」に到達。	まずはこちら（おすすめ）
その 2 AI とゼロから作る	AI にコードを書いてもらい自作する。仕様を伝え、対話しながら組み立てる。自由に改造できる。	時間と根気がある人（中〜上級）

迷ったら、まず**その 1** で「動く形」を体験し、仕組みが分かってから**その 2** で自分好みに作り直す、という順がおすすめです。どちらも、後半の「Render で公開する」以降は共通の手順に合流します。

登場するのは「3 人の協力者」だけ

GitHub（ギットハブ）	Render（レンダー）	生徒のブラウザ
アプリの設計図（プログラム）を保管する「倉庫」	設計図から実際にアプリを動かし、公開する「工場兼お店」	配られた URL を開いて英語で練習する

さらに Render には、AI の「頭脳」として Anthropic（Claude） と、音声入力・読み上げ用に OpenAI の力を、鍵（API キー）でつなぎます。

用語メモ

API（エーピーアイ）：ソフト同士が「お願い」をやりとりするための窓口。ここでは、あなたのアプリが AI 会社に「この英語に反論して」と頼むための接続口のこと。

API キー：その窓口を使うための**あなた専用の鍵（パスワードのような長い文字列）**。使った分だけ料金がかかるので、他人に見せてはいけません。

画面の言葉について（日本語と英語の対応）

GitHub・Render・各社の管理画面は、**英語で表示されることが多い**です。本書では、画面に出るボタンや項目を **「日本語（English）」** の形で示します（例：環境変数（Environment Variables）、公開／非公開（Public／Private）、デプロイ（Deploy））。画面で英語を探すときの手がかりにしてください。

このあとの目次

まず AI に聞いてみよう（ウォームアップ）

Q：私はプログラミングの知識がない英語教員です。GitHub に置いた Web アプリを Render で公開し、Anthropic（Claude）の API を使えるようにしたいです。全体の流れを、専門用語をかみくだいて、小学生に説明するくらいやさしく 5 ステップで教えてください。

1　〔共通〕3 つの必須アカウント＋OpenAI（任意）を作る

どちらの作り方でも、最初に 3 つの必須サービスと、音声を使う場合だけ必要な OpenAI に登録します。どれも「メールアドレスとパスワードを決めて登録するだけ」で、特別な知識は要りません。すべて同じメールアドレスで登録しておくと管理が楽です。

サービス	役割	登録先と費用
GitHub（ギットハブ）	設計図を保管する倉庫	github.com／**無料**
Render（レンダー）	アプリを動かし公開する場所	render.com／**無料プランあり**
Anthropic（アンソロピック）	AI（Claude）の頭脳＝**必須**	console.anthropic.com／**使った分だけ課金**
OpenAI（オープンエーアイ）	音声入力・読み上げ＝任意	platform.openai.com／**使った分だけ課金**

「使った分だけ」の費用を安全に管理する

利用量に応じて費用が変わります。固定額や「必ずここで止まる」とは考えず、Anthropic のプリペイド残高・自動追加設定、OpenAI の予算通知・利用状況、各社の最新料金を確認してください（→ 2 章・7 章）。

⚠ パスワードは使い回さない

それぞれ別のパスワードにして、メモを安全な場所に控えておきましょう。できれば **2 段階認証**（ログイン時にスマホの確認も求める仕組み）もオンにすると安心です。

2　〔共通〕AI の鍵（API キー）を用意する

アプリが AI を呼び出すための**鍵**を用意します。鍵は「一度しか全体が表示されないことが多い」ので、コピーしたら**すぐ安全な場所に保存**してください。

2-1. Anthropic のキー（必須・Claude 用）

1　支払い方法を登録し、必要な範囲のクレジットを購入する

console.anthropic.com にログインし、Billing（請求）で支払い方法を登録します。API はプリペイドの利用クレジットから差し引かれます。

2　残高と自動追加（auto-reload）を確認する

自動追加を使わない場合はオフのままにします。使う場合は、追加の条件と金額を小さく設定し、残高と Usage／Cost を定期的に確認します。

3　キーを発行する

「API Keys」で新しいキーを作成し、`sk-ant-` で始まる長い文字列をコピーして安全に保存します（あとで 5 章・Render に貼ります）。

2-2. OpenAI のキー（任意・音声入力用）

マイク入力と読み上げを使う場合だけ必要です。文字入力だけで運用するなら、この手順と OPENAI_API_KEY は省略できます。platform.openai.com でプロジェクト用 API キーを作成し、安全に保存します。プロジェクトの月間予算は通知用のソフト上限で、超過時に API を必ず停止する仕組みではありません。利用状況と請求を定期的に確認してください。

⚠ 鍵は絶対に人前・ネットに出さない

API キーは銀行の暗証番号のようなものです。メール・SNS・配布物・スクリーンショットに写り込ませないでください。漏れたと思ったら、その画面ですぐ「Revoke（無効化）」して作り直せます。

請求画面が見つからないとき

Q：Anthropic Console でプリペイド残高と auto-reload、Usage／Cost を確認したいです。いま見えているメニュー名は〔　〕です。最新の公式画面に沿って、開く場所を順番に教えてください。

3　〔その 1〕配布版のコードを入手する

完成済みの修正版を入手し、GitHub から Render へつなぎます。プログラムを書く必要はありません。配布元は、公開 GitHub と教室サイトの ZIP の両方を用意しています。GitHub に不慣れな先生は ZIP から始められます。

用語メモ

リポジトリ：GitHub 上の、1 アプリ分の「フォルダ＋履歴」のまとまり。

公開（Public）／非公開（Private）：GitHub のリポジトリはこの 2 種類だけ。公開は誰でも検索で見つけられ、非公開は招待した人だけが見られます。「URL を知っている人だけ」という中間設定はありません。

commit（コミット）：変更を「ここまでの状態」として記録・保存する操作。

方法 A：教室サイトの ZIP を使う（GitHub に不慣れな先生向け）

1　教室サイトから ZIP を入手する

英語ディベートの教室のダウンロードページから修正版 ZIP を保存し、パソコン上で展開（解凍）します。

中に次の 6 ファイルがあることを確認します：.env.example／.gitignore／index.html／package.json／README.md／server.js

2　自分の GitHub にリポジトリを作る

GitHub で「New repository」を選び、名前（例：my-debate-app）を付けます。自分だけで管理したい場合は Private（非公開）で構いません。

3　6 ファイルをアップロードする

「Add file」→「Upload files」で、展開したフォルダの中身をまとめてアップロードし、「Commit changes」を押します。フォルダそのものではなく、中の 6 ファイルをリポジトリ直下に置きます。

.env.example は設定例です。本物の API キーや合言葉は書き込まないでください。

別案：公開 GitHub から始める

https://github.com/r214003/debate-trainer-sample01 を開き、「Fork」で自分のコピーを作るか、「Code」→「Download ZIP」で入手します。公開リポジトリは検索・閲覧できますが、秘密情報は Render 環境変数にだけ置きます。

方法 B：公開 GitHub を「Fork」する（更新を追いやすい）

公開 GitHub は https://github.com/r214003/debate-trainer-sample01 です。「Fork」で自分のコピーを作れます。公開リポジトリとその Fork は誰でも閲覧できますが、API キーや合言葉はコードに入れず、各自の Render 環境変数にだけ設定するため、秘密情報は公開されません。

> 💬 **アップロードでつまずいたら**
>
> Q：GitHub で、パソコンにある複数のファイルを新しいリポジトリにアップロードしたいです。ブラウザだけで（コマンドを使わずに）行う方法を、初心者向けに 1 ステップずつ教えてください。

入手が終わったら、**5 章「Render に設置して公開する」**へ進みます。

4　〔その 2〕AI と一緒に、自分だけのアプリをゼロから作る

配布版を使わず、「こんな受け答えをするアプリがほしい」と AI に伝えて、コードを書いてもらう作り方です。土台の技術（GitHub・Render・Claude Haiku・OpenAI の音声 API）はその 1 と同じなので、1〜2 章と 5 章はそのまま使えます。ディベートの中身だけ、あなたが自由に決めます。

4-1. 考え方：「土台」は共通、「中身」だけ自分で決める

アプリは、**いつも同じ「土台」**と、**先生ごとに違う「中身」**の 2 層でできています。この切り分けが分かると、毎回ゼロから悩まずに済みます。

土台（毎回共通）	中身（先生ごと）
・Node.js のサーバーが Claude Haiku を中継 ・API キーは環境変数（Environment Variables）で保護 ・合言葉（APP_PASSWORD）で入室制限 ・記録は保存しない・Render で公開	・AI はどんな役回りか（相手／コーチ） ・1 回のやりとりの流れ（ターン構成） ・評価の観点と甘辛・英語レベル ・論題の決め方、日本語の助言の出し方

そこで、AI に渡す指示も 2 つに分けます。**「土台プロンプト」（コピーしてそのまま使う固定文）**と、**「中身プロンプト」（あなたの設計を書き込む部分）**です。

4-2. まず「中身」を日本語で書き出す（設計メモ）

AI に触る前に、作りたいアプリの動きを**日本語で箇条書き**にします。ここがいちばん大切で、ディベート指導の腕の見せどころです。次の項目を埋めてみてください。

設計メモ（この項目を日本語で書く）

1. **目的**：生徒に何を練習させたいか。
2. **AI の役割**：相手（肯定側／否定側）か、質問役か、コーチか。
3. **1 回の流れ**：だれが先に話し、何ターンやりとりして終わるか。
4. **評価**：何を見て評価するか（例：理由の明確さ、相手への応答）。A〜E を出すか。
5. **英語レベル**：対象学年と、AI が使う英語の難しさ。
6. **論題**：あらかじめ用意するか、生徒／AI が決めるか。
7. **音声**：話して練習させるか、文字入力だけにするか。

書き方のイメージが湧くよう、3 つの例を挙げます。どれもあくまで一例で、自由に変えて構いません。

例	1 回の流れ（設計メモの中心部分）
例 A 反論練習	AI が論題について短い主張を 1 つ述べる → 生徒が英語で反論する → AI が再反論と一言の助言を返す → 最後に日本語の講評と A〜E 評価。
例 B 質疑応答	AI が生徒の立場に対して質問を 1 つする → 生徒が英語で答える → AI が答えを踏まえて追加質問 → これを 3 往復 → 講評。
例 C 立論フィードバック	生徒が自分の立論を英語で入力（または話す） → AI が「良い点・改善点・お手本の一文」を日本語＋英語で返す。対戦はしない。

4-3. AI に渡す「土台プロンプト」（コピーして固定）

技術の枠組みを決める部分です。**基本そのまま**使い、〔 〕だけ整えます。

土台プロンプト（固定・コピー用）

あなたは、プログラミング未経験の英語教員を助ける開発アシスタントです。次の条件を満たす、安全なゲスト専用 Web アプリを作ってください。

- 構成：index.html（素の JavaScript） + server.js（Node.js 20 以上） + package.json＋README.md ＋.env.example＋.gitignore。

- Claude API は必ず server.js から中継し、ANTHROPIC_API_KEY をブラウザへ送らない。モデルは ANTHROPIC_MODEL（既定 claude-haiku-4-5-20251001）。

- Render では ANTHROPIC_API_KEY と APP_PASSWORD を必須にし、未設定なら安全のため起動を停止する。

- 最初に POST /api/auth で合言葉をサーバー照合し、AI・音声の各 API ルートでも x-app-password を確認する。合言葉は同じタブの sessionStorage にだけ置き、終了またはタブを閉じたら消す。

- 任意の OPENAI_API_KEY がある場合だけ、音声入力（Whisper）と読み上げ（TTS）を使う。キーは必ずサーバー側で保持する。

- 練習内容・評価・完了回数をファイルやデータベースへ保存しない。個別 ID や個別ログインを作らない。

- 入力サイズ、API 待ち時間、出力トークン、短時間の連続リクエストに上限を設け、エラーは秘密情報を含めず日本語で返す。

- Render の PORT を環境変数から読み、0.0.0.0 で待ち受ける。PORT は利用者に手動設定させない。

- README に、API 事業者へのデータ送信、記録非保存の範囲、費用管理、合言葉の限界、各自の責任で利用する旨を明記する。

必要ファイルをファイル名ごとに全文で示し、最後に構文確認、起動、主要 API ルート、合言葉認証、Render 環境変数の確認手順を初心者向けに説明してください。

4-4. AI に渡す「中身プロンプト」（自分の設計を書く）

4-2 の設計メモを、そのまま文章にして貼ります。〔　〕を自分の内容に置き換えてください。

📋 中身プロンプト（テンプレート・書き換えて使う）

先ほどの土台の上に、次の「中身」で英語ディベート練習アプリを作ってください。

- 目的：〔生徒に練習させたいこと〕
- 対象：〔学年・英語レベル。例：中 3〜高 1〕。AI が使う英語もこのレベルに合わせる。
- AI の役割：〔相手（肯定側／否定側）／質問役／コーチ など〕
- 1 回の流れ：〔だれが先に話し、何ターンで、どう終わるか。4-2 の例 A〜C を参考に〕
- 評価：〔何を見るか。A〜E 評価を出すか。日本語の講評を付けるか〕
- 論題：〔あらかじめ用意する／生徒が入力する／AI が提案する〕
  用意する場合の例：〔論題を 2〜3 個〕
- 音声：〔使う／使わない〕
- 画面の言葉：ボタンや説明は〔日本語〕で表示する。

まず、この設計で不足している点や、初心者がつまずきやすい点があれば
先に指摘してから、コードを書き始めてください。

4-5. 小さく作って、動かしながら育てる

いちどに全部を作らせないのがコツです。次の順で、少しずつ確かめながら進めます。

1 まず最小版（テキストのみ・1 モード）

音声や複数モードは後回し。**文字入力だけで 1 つの流れが動く**ところまでを AI に作らせます。

2 GitHub に置いて、Render で公開（5 章）

出てきたファイルを自分のリポジトリ（repository）に置き、5 章の手順で公開します。まず「動く」ことを確認します。

3 動いたら、機能を 1 つずつ足す

「次に、音声入力（Whisper）を足して」「A〜E 評価を足して」「モードをもう 1 つ足して」と、**1 回に 1 つだけ**お願いして、そのつど公開して確かめます。

4 エラーはそのまま AI に戻す

Render の赤いログや画面のエラー文を**同じ会話に貼って**「直して」と伝えます。直ったらまた公開します。

⚠ 作成までは何度も修正を入れることが基本

一発で完璧に動くことは、ふつうありません。**数回〜十数回の往復**が当たり前です。うまくいかない時間も含めて楽しめる方向けの道です。急ぐときや確実性を優先するなら、その 1 をおすすめします。

4-6. ディベートの中身を詰める（工夫どころ）

動く土台ができたら、あとは AI と相談しながら「中身」を磨きます。たとえば次のような調整です。

- **役割と難易度**：AI の反論を易しめ／手強めに。「中学生にも分かる短い英語で反論して」など。
- **進行**：ターン数、先攻・後攻、時間の目安。
- **評価の観点**：内容重視か、応答重視か。甘め／辛めの基準。日本語の助言をどの場面で出すか。
- **論題**：学校生活・時事など、生徒に身近なものを用意。AI に候補を出させてもよい。
- **安全**：個人情報を聞かない、不適切な話題を避ける、と土台プロンプトに一文加える。

💬 設計を一緒に固めたいとき

Q：英語ディベートの練習アプリを作りたいです。私の生徒は〔学年・レベル〕、授業では〔使いたい場面〕に使います。まず、生徒がやる気になり、短時間で回せる「1 回の流れ」を 3 案、日本語で提案してください。コードはまだ書かなくて大丈夫です。

💬 一気に作られて分からなくなったとき

Q：いま渡されたコードが多くて、どのファイルを何のために GitHub に置くのか分かりません。ファイルごとの役割と、置く順番を、初心者向けに一覧で説明してください。

コードがそろったら、**5 章「Render に設置して公開する」**へ進みます。

5　〔共通〕Render に設置して公開する

ここで**その 1・その 2 の両方が合流**します。自分の GitHub リポジトリに置いたコードから、実際に動くアプリを作ります。この作業を**デプロイ（Deploy）**（＝プログラムを公開・稼働させること）と呼びます。やることは「元を選ぶ→鍵を入れる→ボタンを押す」の 3 つです。

1　新しい Web サービスを作り始める

render.com にログインし、「New +」→「Web Service」を選びます。

2　自分のリポジトリを選ぶ

GitHub との連携を許可し（非公開リポジトリも選べるようになります）、一覧から **3 章または 4 章で用意したリポジトリ**を選んで「Connect」します。

3　基本設定を確認する

項目	入れる値（初期設定として既に入力されていることも）
Runtime（実行環境）	Node
Build Command（準備の命令）	npm install（または空欄）
Start Command（起動の命令）	npm start

4　料金プランを選ぶ

まずは **Free（無料）** で構いません。本格運用時の注意は 7 章にまとめました。

5　環境変数（Environment Variables）を登録する ― ここが核心

Environment Variables（＝アプリに渡す秘密の設定値。鍵をここに入れておけば、コードの中に鍵を書かずに済み安全です）に、次を「名前＝値」で追加します。

名前（Key）	値（Value）
ANTHROPIC_API_KEY	2 章の Claude の鍵（sk-ant-…）必須
APP_PASSWORD	生徒に伝える、長く推測されにくい合言葉（必須）
OPENAI_API_KEY	音声入力・読み上げを使う場合のみ（任意）
ANTHROPIC_MODEL	claude-haiku-4-5-20251001（任意。指定なしでも動きます）

重要：Render では ANTHROPIC_API_KEY または APP_PASSWORD が未設定だと、安全のため起動しません。OPENAI_API_KEY を省略した場合も文字入力は使えますが、音声入力・読み上げは使えません。

6 デプロイ（公開）ボタンを押す

「Create Web Service」を押すと、Render が自動で準備を始めます。ログが流れ、数分で **Live** と表示されれば成功です。発行された `https://…onrender.com` があなたのアプリの URL です。

⚠ `PORT` は自分で追加しない

ポート番号は Render が自動で決めます。手動で `PORT` を足すと、かえって動かなくなることがあります。

💬 デプロイが赤字（失敗）で止まったら

Q：Render で Node.js のアプリをデプロイしたら失敗しました。ログの最後のほうにこう出ています：〔赤い行を中心に、エラー文をそのまま貼る〕。考えられる原因と、初心者向けの直し方を教えてください。

6　〔共通〕動作を確認し、生徒に配る

1 自分で開いてみる

発行 URL をブラウザで開きます。**合言葉**を求められたら、5 章で決めた APP_PASSWORD を入れます。練習画面に進めれば成功です。

2 1 モードだけ試す

どれか 1 モードで短く練習し、AI の応答と最後の**講評・A〜E 評価**まで出るか確認します。

3 生徒に「URL」と「合言葉」を配る

この 2 つだけ伝えれば生徒は練習できます。合言葉は**紙やクラス内掲示など閉じた場で**共有し、SNS 等には載せません。

授業前・部活で使う前に「1 回開いて起こす」

無料プランのアプリは、しばらく使われないと眠り、次に開くとき復帰に約 1 分かかることがあります。授業の数分前に先生が一度 URL を開いておくと、生徒がスムーズに始められます。

⚠ マイク（音声）は「https」のページでだけ動く

ブラウザの仕様で、マイク入力は安全な接続（https）でしか使えません。Render の URL は **https** なので問題ありません。

7　〔共通〕費用と、使いすぎを防ぐ設定

ポイントは「**無料でどこまで**」と「**従量課金は上限で囲う**」の 2 点です。

	内容
無料	GitHub（保管）。Render も無料プランで開始できます。
従量課金（使った分だけ）	Anthropic の AI 利用。音声を使う場合は OpenAI も。

費用は利用量と最新単価で変わる

「1 回数円」などの固定額は保証できません。授業前に少人数で試し、Anthropic・OpenAI の公式料金ページと利用画面で実測してください。

費用管理はサービスごとに確認

Anthropic はプリペイド残高と自動追加（auto-reload）を確認します。OpenAI のプロジェクト月間予算は主に通知用のソフト上限で、超過しても処理が続く場合があります。予算通知だけに頼らず、利用状況・残高・請求を定期確認し、不要なキーは無効化してください。

Render 無料インスタンスの注意

無料 Web サービスは、現在は 15 分間アクセスがないと停止し、次のアクセス時の復帰に約 1 分かかることがあります。月間利用枠などの制限もあるため、授業前に起動確認し、本格運用では最新の有料インスタンス料金を検討してください。

💬 自分のケースで試算したい

Q：Anthropic の〔モデル名〕を使い、1 回の練習でおよそ〔やりとりの回数や長さ〕のやりとりをします。生徒〔人数〕人が各 1 回使うと、1 コマの費用はおよそいくらですか。最新の料金を調べたうえで計算過程も見せてください。

8　〔共通〕困ったとき ― AI への聞き方

大事なのは**「何が起きたか」を正確に AI へ伝える**こと。コツは、**画面の文字（エラー文）をそのままコピーして貼る**こと。要約するより原文のほうが正しく直せます。

AI に伝える「3 点セット」

1. **やろうとしたこと**（例：Render でデプロイした）
2. **起きたこと**（画面の文・エラー文をそのまま）
3. **自分の状況**（例：プログラミングは未経験）

よくある症状と、まず確認

症状	まず確認
合言葉を入れても進めない	Render の `APP_PASSWORD` の値と入力が完全一致か（前後の空白に注意）
AI が応答しない／エラー	`ANTHROPIC_API_KEY` が正しいか。上限額に達していないか。
マイクが反応しない	URL が `https://` か。ブラウザでマイクを許可したか。`OPENAI_API_KEY` があるか。
開くのがとても遅い	無料インスタンスの復帰待ち（約 1 分のことがあります）。授業前に一度開いて起こす。

💬 万能テンプレ（何が起きても、まずこれ）

Q：私はプログラミング未経験の英語教員です。GitHub＋Render＋Anthropic API で作った Web アプリを設置しています。〔やろうとしたこと〕をしたら、〔画面に出た文・エラー文をそのまま〕となりました。原因の候補と、確認すべき場所・直す手順を、初心者向けに順番に教えてください。

直したあとの反映

GitHub 上でファイルを直して保存（commit）すると、Render が自動で作り直します。環境変数（Environment Variables）を変えたときは、Render 側で再デプロイ（Manual Deploy → Deploy latest commit（または環境変数の「Save, rebuild, and deploy」））を押すと確実です。

Render が起動直後に停止する	ANTHROPIC_API_KEY と APP_PASSWORD の両方が設定されているか確認
「利用が集中」「入力が長すぎる」と表示	1 分ほど待つ／文章を短くする。修正版の安全制限が働いています

9　〔共通〕安全とプライバシー

生徒に使ってもらうものだからこそ、次の点をおさえておきましょう。

- 合言葉は簡易的な「入口の鍵」で、個人別のログインではありません。修正版は合言葉をサーバーで照合し、短時間の連続利用を抑える制限も設けていますが、完全な不正利用防止や課金上限を保証するものではありません。
- **生徒に個人情報を入力させない。**氏名・住所・連絡先などを打ち込ませない運用にします。
- **記録の保存について確認する。**本書の基本構成では**練習内容や評価をサーバーに保存しません。**記録を残す版を使う場合は、保存先・内容・保護者への説明の要否を事前に確認してください。
- API キーは校内でも共有せず、Render の環境変数（Environment Variables）だけに置きます。GitHub、README、.env.example、画面、配布 ZIP に本物のキーや合言葉を書かないでください。
- AI の応答・講評・評価には誤りが混じることがあります。公開、授業での使用、API キーと費用の管理、生徒や保護者への説明を含め、各自の判断と責任で利用してください。

補足：合言葉はブラウザの同じタブ内の sessionStorage に保持されます。アプリは練習本文や評価を永続保存しませんが、短時間の連続利用を抑えるため接続元ごとの回数をサーバーの一時メモリで数えます。サーバー再起動で消え、練習本文は含みません。

> **校内で使う前に**
>
> 勤務校によっては、外部サービスや生成 AI の利用に関する規定・申請があります。公開・運用の前に、情報担当や管理職に一声かけておくと安心です。

付　付録：用語集と設置チェックリスト

用語集

用語	やさしい説明
API	ソフト同士がお願いをやりとりする窓口。アプリが AI に処理を頼む接続口。
API キー	その窓口を使うあなた専用の鍵。使った分だけ課金され、他人に見せない。
GitHub	プログラム（設計図）を保管・共有する倉庫のようなサービス。
リポジトリ	GitHub 上の、1 アプリ分のフォルダと変更履歴のまとまり。
公開／非公開	リポジトリの種類。公開＝誰でも検索可、非公開＝招待した人だけ。中間（URL 限定）は無い。
Fork	公開リポジトリを、自分の側にまるごと複製すること（ボタン 1 つ）。
commit	変更を「ここまでの状態」として記録・保存する操作。
Render	設計図から実際にアプリを動かし、URL で公開してくれるサービス。
デプロイ	プログラムを公開して、使える状態にすること。
環境変数	アプリに外から渡す秘密の設定値。鍵や合言葉をここに入れて安全に保つ。
Node.js	今回のアプリの土台となる実行環境。Render で自動的に用意される。
https	暗号化された安全な通信。マイク入力などはこの接続でのみ動く。
従量課金	使った分だけ料金が発生する仕組み。上限額の設定で囲える。
合言葉（APP_PASSWORD）	アプリを使える人を限定する入口の鍵。先生が自由に決める。

設置チェックリスト

- ☐ GitHub・Render・Anthropic のアカウントを作った（音声を使う場合は OpenAI も）
- ☐ Anthropic の残高・自動追加設定と、OpenAI の予算通知・利用状況を確認した
- ☐ Claude の鍵（sk-ant-…）を取得し、安全に保存した

- ☐ 〔その 1〕配布リンクから入手→自分のリポジトリに置いた／〔その 2〕AI にコードを書いてもらい GitHub に置いた
- ☐ Render で Web Service を作り、自分のリポジトリを連携した
- ☐ Render の環境変数に ANTHROPIC_API_KEY と APP_PASSWORD を入れた（音声を使う場合は OPENAI_API_KEY も）
- ☐ デプロイが成功し、`https://...onrender.com` の URL が出た
- ☐ 自分で開き、合言葉→練習→講評まで動くことを確認した
- ☐ 生徒に「URL」と「合言葉」を安全な方法で配った

このマニュアルは、プログラミング未経験の先生が、AI の助けを借りながら自力で設置できることを目指した改訂版です。サービスの画面、料金、名称は変わることがあります。表示が本書と異なるときは公式ヘルプを確認し、画面の文言やエラー文をそのまま AI に貼って尋ねてください。

〈発行〉debate-class.com　〈作成〉小林良裕　〈版〉改訂版 v0.4（2026 年 7 月）