DebateBot

The Intelligent AI Debate Platform

Project Documentation

DebateBot Team

January 23, 2026

Contents

1 Introduction **2**

2 Features Guide **3**
- 2.1 Dual-Sided AI Debates . . . 3
- 2.2 Live Debate Arena (User vs AI) . . . 3
- 2.3 Comprehensive Scoring System . . . 3
- 2.4 AI Coach & Feedback . . . 4

3 Architecture **5**
- 3.1 System Overview . . . 5
- 3.2 Component Details . . . 5
  - 3.2.1 Frontend (Client-Side) . . . 5
  - 3.2.2 Backend (Server-Side) . . . 5
  - 3.2.3 AI Logic (Intelligence Layer) . . . 6
- 3.3 Data Flow . . . 6

4 Technology Stack **7**
- 4.1 Backend . . . 7
- 4.2 Frontend . . . 7

5 API Reference **8**
- 5.1 Debate Generation . . . 8
- 5.2 Live Debate Counter . . . 8
- 5.3 Score Argument . . . 8

6 Deployment Guide **9**
- 6.1 Backend Deployment (Render.com) . . . 9
- 6.2 Frontend Deployment (Vercel/Netlify) . . . 9
- 6.3 Local Production Build . . . 9

Chapter 1

Introduction

DebateBot is a next-generation debating platform powered by **LLaMA 3.3 70B** and **LangGraph**. It goes beyond simple chat by orchestrating complex, multi-stage debates between AI agents or conducting live debates where users can challenge the AI directly.

Key Features

- **Dual-AI Debates**: Watch comprehensive debates unfold (Opening → Rebuttal → Closing).
- **Live Arena**: Step into the ring and debate the AI yourself.
- **Smart Scoring**: Get detailed feedback on coherence, evidence, and logical fallacies.
- **Real-Time Streaming**: Experience fluid, animated arguments as they are generated.

Chapter 2

Features Guide

DebateBot takes you beyond simple chat interfaces into a structured, educational, and engaging competitive debating environment.

2.1 Dual-Sided AI Debates

Watch two AI agents debate any topic in real-time.

- **Structure**: Follows a formal flow: Opening Arguments → Rebuttals → Closing Statements.
- **Fairness**: The AI is instructed to be objective and forceful on *both* sides, regardless of the topic.
- **Education**: Great for understanding multiple perspectives on complex issues.

2.2 Live Debate Arena (User vs AI)

Step into the ring and test your skills against our AI.

- **Interactive**: You choose a side (Proposition or Opposition).
- **Round-by-Round**:
  1. **Opening**: You state your case; the AI responds.
  2. **Rebuttal**: You counter the AI; the AI counters you.
  3. **Closing**: Final summaries.
- **Real-time Counter-arguments**: The AI "listens" to your text and generates specific refutations, not generic responses.

2.3 Comprehensive Scoring System

We don't just say "good job"; we tell you *why*.

- **Metrics**:
  - **Coherence**: Flow, logical connection between sentences.
  - **Relevance**: Adherence to the topic.
  - **Evidence Strength**: Use of facts, data, or strong reasoning.
  - **Fallacy Check**: Penalties for ad hominem, strawman, or circular reasoning.
- **Visual Feedback**: See your score break down in a beautiful radar chart or progress bars.

2.4 AI Coach & Feedback

Get actionable advice to become a better speaker.

- **Targeted Tips**: If your "Evidence" score is low, the AI suggests adding statistics.
- **Gap Analysis**: Shows exactly how far you are from your target score.
- **Specific Improvements**: "Try connecting your second point with 'Furthermore' to improve coherence."

Chapter 3

Architecture

3.1 System Overview

DebateBot is built on a modern, decoupled architecture separating the user interface from the complex logic of debate generation.

```
User -> [React Frontend] -> HTTP/JSON -> [FastAPI Backend]
                                                |
                                          [LangGraph]
                                                |
                                        [Groq LLaMA 3.3]
```

3.2 Component Details

3.2.1 Frontend (Client-Side)

- **Technology**: React 18, Vite
- **Responsibility**:
  - Renders the interactive UI.
  - Manages local state (user inputs, current debate stage).
  - Visualizes arguments with animations.
  - Handles API communication via proxy.
- **Key Components**:
  - `DebateArena`: The main container for the debate view.
  - `Scoring`: Visualizes the argument scores.
  - `LiveDebateArena`: Manages the interactive user-vs-AI mode.

3.2.2 Backend (Server-Side)

- **Technology**: Python 3.12, FastAPI
- **Responsibility**:
  - Exposes RESTful API endpoints.
  - Validates incoming requests using Pydantic models.
  - Manages CORS and security.
  - **LangGraph Integration**: The core logic resides here. It defines the "nodes" (Opening, Rebuttal, Closing) and "edges" (transitions) of the debate flow.

3.2.3 AI Logic (Intelligence Layer)

- **Technology**: LangChain, Groq API (LLaMA 3.3 70B)
- **Responsibility**:
  - **Contextual Understanding**: Analyzes the topic and previous arguments.
  - **Argument Generation**: Produces structured text for specific debate stages.
  - **Scoring & Feedback**: Evaluates arguments based on rubrics (coherence, evidence, etc.).

3.3 Data Flow

1. **Initiation**: User submits a topic (e.g., "AI is dangerous").
2. **Orchestration**:
   - `POST /api/debate` triggers the `graph.py` workflow.
   - **Node 1 (Opening)**: LLM generates opening statements for Prop and Opp.
   - **Node 2 (Rebuttal)**: LLM reads the opponent's opening and generates a counter.
   - **Node 3 (Closing)**: LLM synthesizes everything into a final statement.
3. **Delivery**: The full structured JSON object is returned to the frontend.

Chapter 4

Technology Stack

4.1 Backend

Component	Technology	Description
Language	Python 3.12	Modern, high-performance scripting.
Framework	FastAPI	High-performance web framework.
Orchestration	LangGraph	State machine library for AI agents.
State Mgmt	LangChain	LLM application framework.
AI Model	Groq (LLaMA 3.3)	Ultra-fast inference API.
Validation	Pydantic	Data validation.

4.2 Frontend

Component	Technology	Description
Library	React 18	UI Library.
Build Tool	Vite	Next Gen Frontend Tooling.
Routing	React Router 6	Declarative routing.
Styling	CSS3	Custom CSS with variables.
Icons	Lucide React	Icon toolkit.
Typography	Google Fonts	Space Grotesk & Plus Jakarta Sans.

Chapter 5

API Reference

Base URL: `http://127.0.0.1:8000` (Local)

5.1 Debate Generation

Generate a full debate flow.

- **Endpoint**: `POST /api/debate`

Request Body:

```
{
  "topic": "Social media does more harm than good"
}
```

5.2 Live Debate Counter

Generate a counter-argument for a specific round.

- **Endpoint**: `POST /api/live-counter`

Request Body:

```
{
  "topic": "string",
  "user_argument": "string",
  "round": "opening",
  "argument_history": [
    { "type": "user", "text": "..." },
    { "type": "ai", "text": "..." }
  ]
}
```

5.3 Score Argument

Analyze an argument and provide detailed scoring metrics.

- **Endpoint**: `POST /api/score-argument`

Request Body:

```
{
  "argument": "The user's argument text...",
  "topic": "The debate topic..."
}
```

Chapter 6

Deployment Guide

6.1 Backend Deployment (Render.com)

The backend is configured for deployment on Render using the `render.yaml` specification.

Steps:

1. Connect your GitHub repository to Render.
2. Select "Blueprints" and point it to `render.yaml`.
3. Render will automatically detect the service `debatebot-backend`.
4. **Environment Variables**:
    - Add `GROQ_API_KEY` in the Rende dashboard.

Start Command:

```
cd backend && uvicorn main:app --host 0.0.0.0 --port $PORT
```

6.2 Frontend Deployment (Vercel/Netlify)

The frontend is a static React application built with Vite.

Steps:

1. Connect your repository to Vercel or Netlify.
2. **Build Settings**:
    - **Root Directory**: `frontend`
    - **Build Command**: `npm run build`
    - **Output Directory**: `dist`

6.3 Local Production Build

```
cd frontend
npm run build
npm run preview
```