ECHERA - Technical Project Report

1. PROJECT OBJECTIVE

ECHERA (English Conversation Helper with Evaluation & Real-time Analysis) is an AI-powered English language learning platform designed to provide real-time conversation practice with intelligent feedback. The system bridges the gap between theoretical knowledge and practical usage by offering an interactive conversational partner that evaluates user performance across three key dimensions:

- Fluency: Evaluates sentence structure, coherence, and natural flow of speech
- Word Choice: Analyzes vocabulary sophistication and contextual appropriateness
- Grammar: Identifies grammatical errors and provides corrective feedback

Unlike traditional learning tools focused on vocabulary drills or grammar exercises, ECHERA creates a judgment-free environment where learners can practice real-time spoken English. The platform leverages local AI deployment (Ollama with LLaMA model) to ensure data privacy while providing high-quality conversational responses. Real-time NLP analysis using spaCy delivers instant, actionable feedback, enabling users to identify and correct mistakes immediately within conversational context.

2. SYSTEM WORKFLOW

Authentication: Secure login and session management using bcrypt for password hashing.

Speech Processing: Real-time transcription via Web Speech API with a 2-second silence detection trigger for automatic processing.

AI Interaction: Response generation via local Ollama instance using the LLAMA model to ensure data privacy.

NLP Analysis: Synchronous evaluation using the spaCy engine to provide performance scores and personalized feedback tips.

Data Storage: User message, AI response, NLP scores, and feedback tips are stored in PostgreSQL database. Database tracks conversation history, maintains user profiles, and calculates aggregate statistics (average scores across all conversations).

Response Delivery: Backend returns JSON response containing AI message, scores (0-100 scale), and feedback tips. ChatInterface displays the conversation thread. FeedbackPanel shows color-coded scores and actionable suggestions. Text-to-Speech (Web Speech API) synthesizes audio playback of AI response using user-selected voice.

3. DESIGN RATIONALE

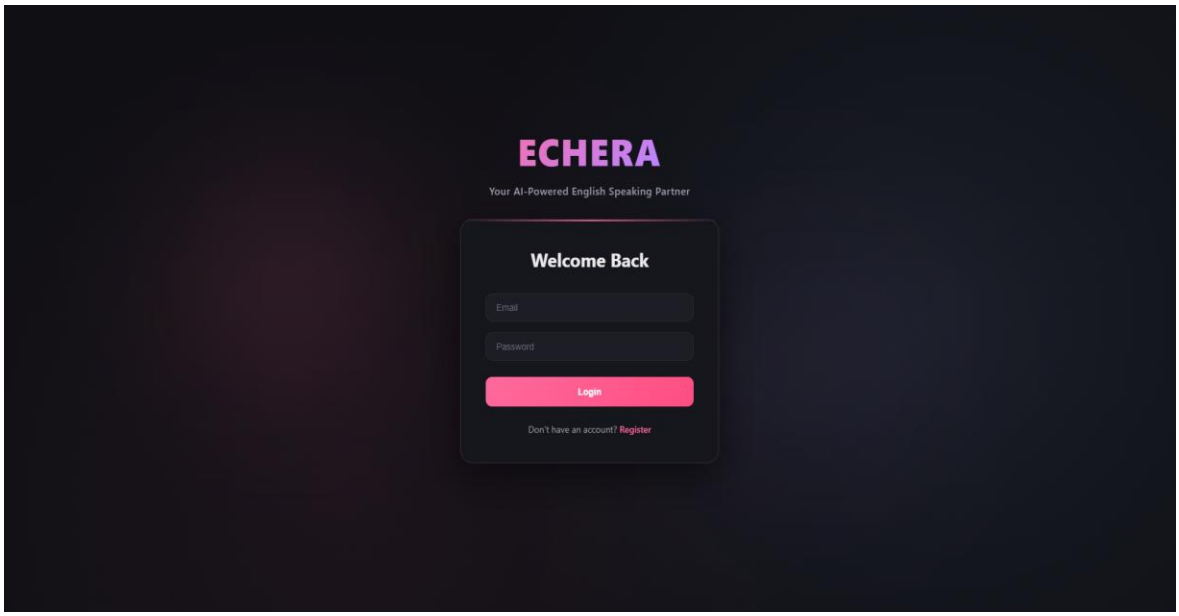
MVC Architecture: ECHERA implements the Model-View-Controller pattern to ensure clear separation of concerns. Models (PostgreSQL database) handle data persistence, Views (Vanilla JavaScript frontend) manage user interface, and Controllers (Flask backend) process business logic. This architecture provides scalability, maintainability, and testability.

Local AI Integration: Hosting Ollama locally eliminates external API dependencies, ensuring complete data privacy and zero per-request costs. Users' conversations never leave their local machine, addressing privacy concerns common in cloud-based AI services. Local deployment also enables offline functionality once models are downloaded.

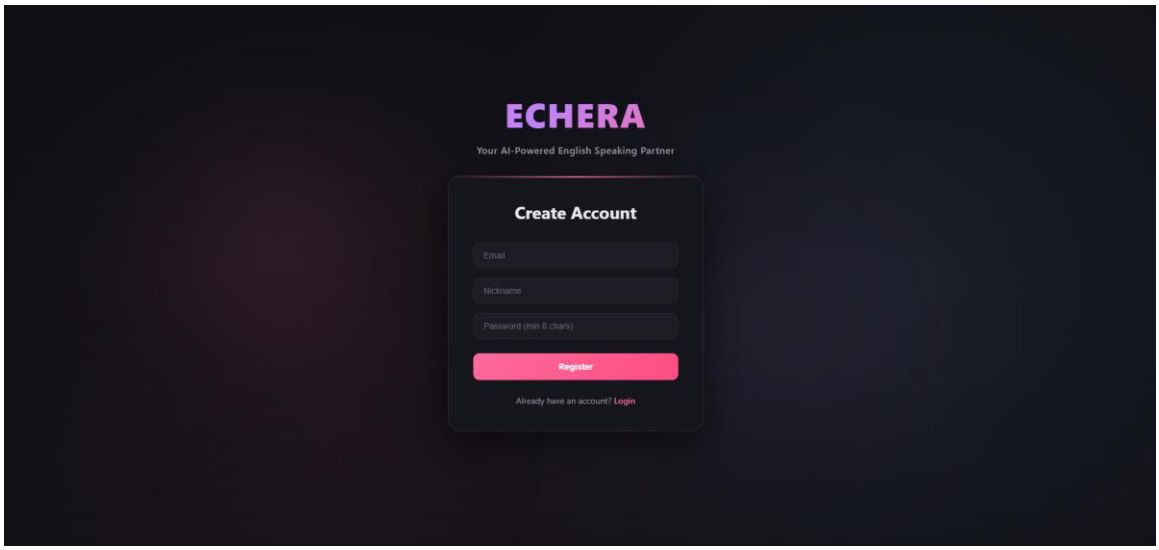
Deterministic Scoring: spaCy provides fast (<50ms), consistent, rule-based linguistic analysis. Unlike probabilistic models that may vary across runs, spaCy's deterministic approach ensures users receive identical scores for identical inputs. This consistency is crucial for tracking learning progress over time.

4. WEB UI DOCUMENTATION

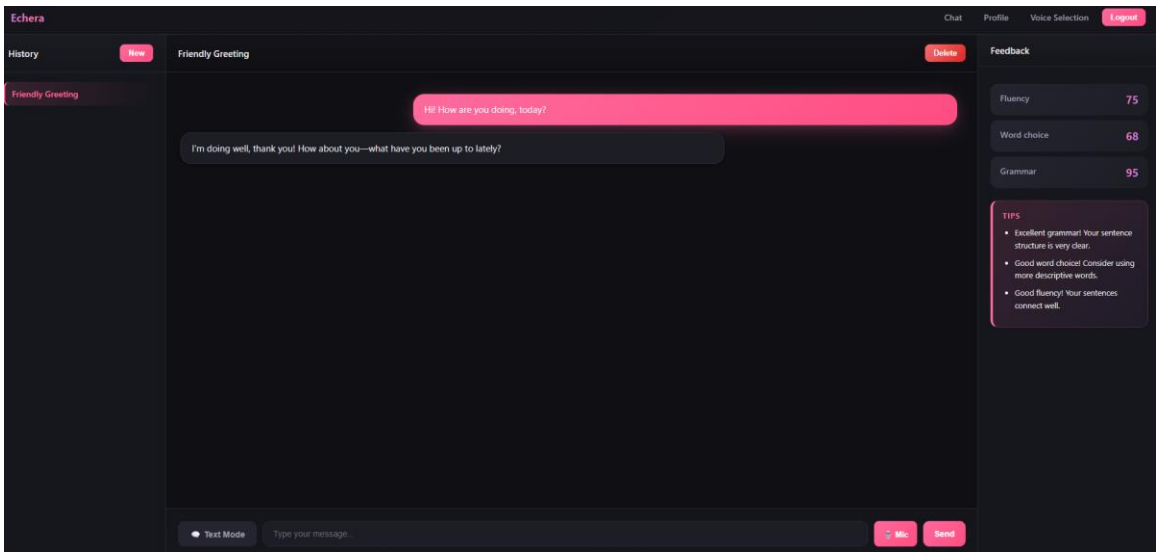
4.1 Login Screen



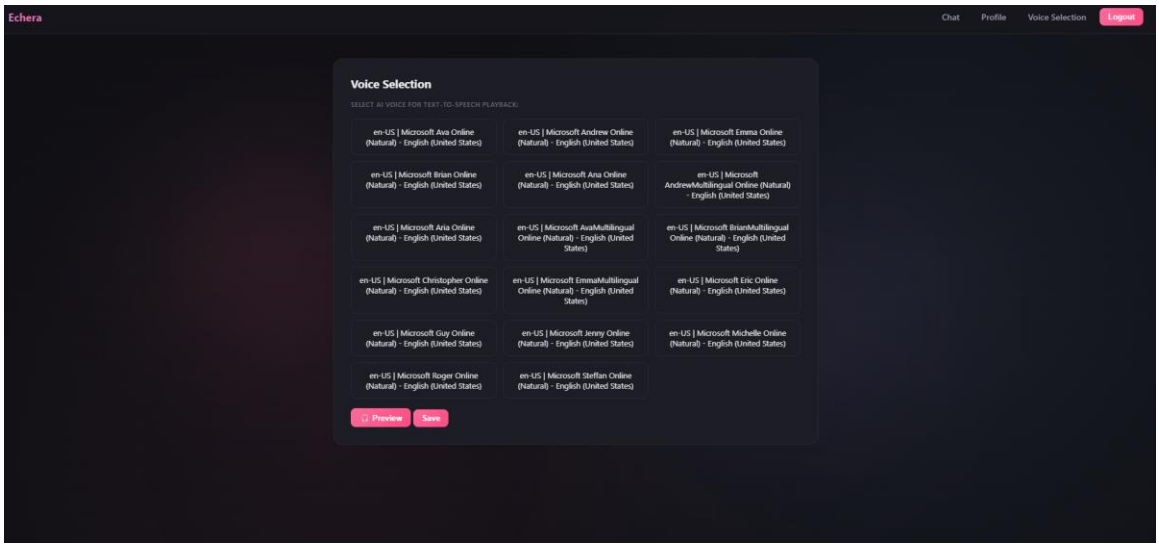
4.2 Register Screen



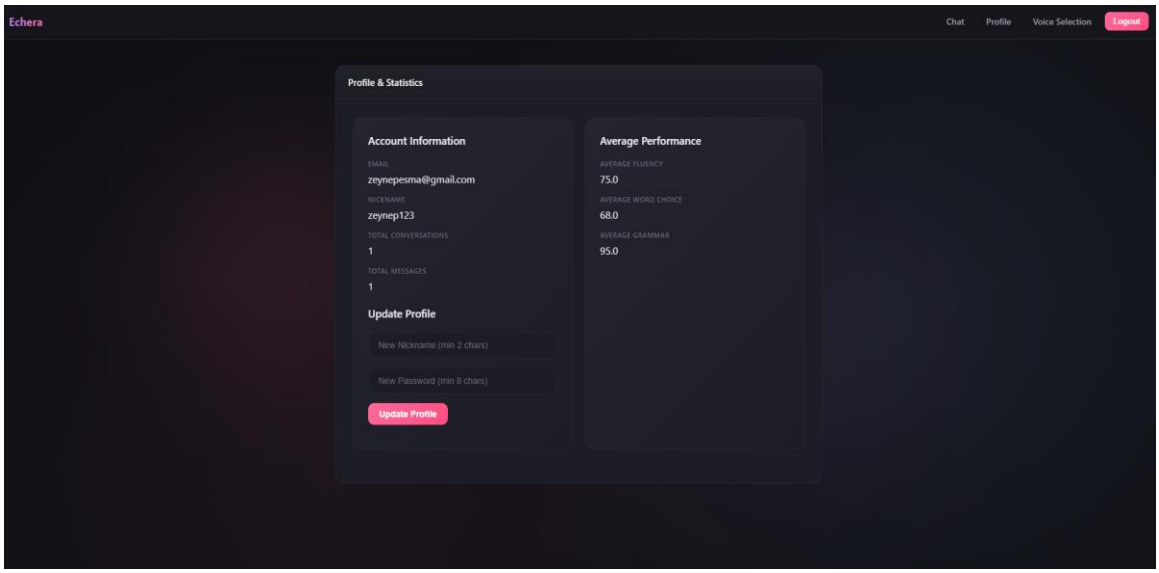
4.3 Chat and Panels



4.4 Voice Selection



4.5 Profile



5. INSTALLATION AND EXECUTION (COMPREHENSIVE GUIDE)

This section provides a comprehensive step-by-step guide for installing and running ECHERA. Follow each step carefully to ensure proper setup.

Step 1: Install Required Software

1. **Python 3.10 or higher: Download from python.org. During installation, check "Add Python to PATH".**

   Verify installation by opening Command Prompt and typing: python --version

   You should see something like: Python 3.10.5

2. **PostgreSQL 14 or higher: Download from postgresql.org and install.**

   Remember the password you set for the "postgres" user during installation - you'll need it later.

   Default port is 5432 - keep this setting.

3. **Ollama: Download from ollama.ai and install. This is the AI engine that powers conversations.**

   After installation, Ollama will run in the background automatically.

Step 2: Set Up the Database

4. **Open Command Prompt and type: psql -U postgres**

   Enter the postgres password you set during PostgreSQL installation.

5. **Create the database by typing: CREATE DATABASE seng321;**

   You should see: CREATE DATABASE

6. **Exit psql by typing: \q**

7. **Navigate to your ECHERA project folder in Command Prompt.**

   Example:*cd C:\Users\YourName\Documents\ECHERA*

8. **Run the schema file to create tables:**

   psql -U postgres -d seng321 -f backend/schema.sql

   You should see a series of CREATE TABLE messages. This creates all necessary database tables.

Step 3: Set Up Python Environment

9. **In Command Prompt (still in ECHERA folder), create a virtual environment:**

python -m venv venv

This creates a folder named "venv" - this isolates project dependencies.

10. Activate the virtual environment:

venv\Scripts\activate

Your command prompt should now show (venv) at the beginning.

11. Install required Python packages (this may take 2-3 minutes):

pip install flask psycopg2-binary bcrypt requests spacy

12. Download the spaCy English language model (this downloads ~50MB):

python -m spacy download en_core_web_sm

Step 4: Set Up Ollama Cloud AI (If not working, local model can be used too by changing the model in ai_engine.py)

13. Open Ollama application on your computer:

After installation, Ollama should be running in the background.

14. Log into your Ollama account:

Click on the Ollama icon and select “Sign in”.

15. Select the gpt-oss:120b-cloud model:

In the Ollama interface, find and select the “gpt-oss:120b-cloud” model.

16. Verify the model is active:

Open Command Prompt and type:

ollama list

You should see “gpt-oss:120b-cloud” listed as the active model.

Step 5: Configure and Launch

17. **Open backend/app.py in a text editor (Notepad works fine).**
18. **Find this line (around line 15):**

password='your_password_here'

19. **Replace 'your_password_here' with your actual PostgreSQL password (the one you set during installation).**
20. **Save and close the file.**
21. **Start the Flask server on the command line while still in project folder:**

python backend/app.py

You should see messages like:

** Running on http://127.0.0.1:5000*

Database connected successfully

22. **Open Google Chrome/Edge and navigate to: http://127.0.0.1:5000**

The ECHERA login page should appear.

6. USAGE SCENARIO

This section illustrates a complete user journey through ECHERA, demonstrating how the system facilitates English language practice with real-time feedback.

Scenario: Job Interview Preparation

Meet Sarah, a Turkish engineering student preparing for English-language job interviews at international tech companies. She uses ECHERA to practice answering common interview questions and improve her spoken English.

1. **Account Creation and Login**

   Sarah opens ECHERA in Chrome and clicks the registration link. She enters her email (sarah@gmail.com), creates a username (sarah_eng), and sets a secure password. After successful registration, she's automatically logged in and redirected to the main chat interface.

2. **Starting a New Conversation**

Sarah clicks the "New Conversation" button in the History Panel. The system creates a new conversation and displays an empty chat interface ready for interaction.

3. **Setting the Context**

   Sarah types her first message: "I want to practice answering interview questions for a software engineer position." She clicks Send. Within 3-4 seconds, the AI responds: "Great! Let's practice. Tell me about yourself and your background in software engineering."

4. **Using Speaking Mode**

   Sarah clicks the "Speaking Mode" button and then she clicks the microphone button. She speaks her response: "I am computer engineering student from METU. I have experience with Python and Flask. I building many projects during university." After 2 seconds of silence, the system automatically processes her speech.

5. **Receiving Real-Time Feedback**

   The transcribed text appears in the chat. Immediately, the Feedback Panel updates with her scores:

- Fluency: 72/100 - Good sentence flow but some choppiness
- Word Choice: 68/100 - Basic vocabulary, could be more sophisticated
- Grammar: 65/100 - Several errors detected

6. **AI Response and Audio Playback**

   The AI responds with encouragement and a follow-up question: "That's great background! Can you tell me about a challenging project you worked on and how you solved technical problems?" The text appears in the chat, and simultaneously, the system uses Text-to-Speech to read the AI's message aloud using Sarah's selected voice (If she is in the speaking mode).

7. **Continued Practice and Improvement**

   Sarah continues the conversation, incorporating the feedback tips. On her next response, she says: "I have built a challenging Flask application..." Her grammar score improves to 78/100. Over 15 minutes, Sarah answers 8 interview questions, and her average scores steadily increase as she applies the real-time feedback.

8. **Viewing Progress**

After the practice session, Sarah navigates to her Profile page. She sees her aggregate statistics across all conversations: Average Fluency 76/100, Average Word Choice 72/100, Average Grammar 74/100. She can track her improvement over time by comparing these metrics to her previous sessions.

9. **Reviewing Conversation History**

   Sarah's "Job Interview Practice" conversation is saved in the History Panel. She can return to it anytime to review past questions and answers, or start a new conversation on a different topic.

7. ASSUMPTIONS AND CONSTRAINTS

System Requirements

- **Operating System:** Windows 10 or higher. The application is optimized and tested exclusively for Windows platforms. MacOS and Linux support is not currently provided.
- **Browser:** Google Chrome (latest version recommended). Web Speech API requires Chrome for full functionality including voice selection and microphone access. Firefox and Safari have limited Web Speech API support.
- **Hardware:** Minimum 8GB RAM recommended for running Ollama AI model. SSD storage preferred for faster model loading. Microphone required for Speaking Mode.
- **Internet:** Required for initial setup and during active usage. Since the AI model (gpt-oss:120b-cloud) runs on cloud servers, a stable internet connection is necessary for AI response generation. All other features work offline except AI conversation.

Usage Limitations

- **Conversation Limit:** Each user account is limited to 50 conversations maximum to maintain optimal database performance and query speed.
- **Message Limit:** Each conversation supports up to 100 message exchanges (50 user messages + 50 AI responses). This prevents excessive memory usage during long sessions.
- **Response Time:** AI response generation typically takes 2-5 seconds depending on hardware capabilities. Users with slower CPUs may experience longer wait times.

- **Language Support:** Currently supports English language learning only. NLP analysis is optimized for English text using spaCy's en_core_web_sm model.

Known Limitations

- **Speech Recognition Accuracy:** Web Speech API accuracy depends on microphone quality, ambient noise, and accent. Users may need to repeat unclear phrases.
- **AI Response Quality:** LLaMA model responses are generally coherent but may occasionally produce nonsensical or off-topic replies. Users should review AI responses critically.
- **NLP Scoring:** Scoring algorithms are rule-based approximations and may not capture nuanced language mastery. Scores provide general guidance but should not be considered definitive assessments.