Introduction to Software Engineering

Project Proposal

Software Engineering Department
Faculty of Information and Technology
University of Science

Table of Contents

Project Proposal

Objectives

This document focus on the following topics:

- ✓ Completing the Project Proposal document with the following sections:
  - Preliminary Problem Statement
  - Proposed Solution
  - Development Plan
  - Human Resources & Costing Plan
- ✓ Understanding the Project Proposal document.

1 Member Contribution Assessment

ID	Name	Contribution (%)	Signature
23127241	Đoàn Thành Phát	25%	
23127089	Nguyễn Quang Minh	25%	
23127085	Phạm Phát Lộc	25%	
23127102	Lê Quang Phúc	25%	

2 Preliminary Problem Statement

The persistent global demand for practical English proficiency, particularly spoken fluency and accuracy, faces a bottleneck in accessible, quality training. In many markets, traditional learning methods and even modern mobile applications fall short in providing the precise, corrective feedback necessary for mastering challenging English phonology.

The proposed system, online English learning system focusing on the Speaking skill, aims to help learners practice pronunciation, speaking fluency, and conversational skills with automatic feedback, similar to how the ELSA application operates. All learning activities - including lessons, speech recording, pronunciation analysis, and performance tracking - are conducted entirely online through a web interface.

When a user first accesses the system, they must register and log in to create a personal learning account. Each learner has a profile that stores their name, current level, completed lessons, pronunciation scores, and progress data. After logging in, learners can choose from a list of speaking topics, such as *Greetings*, *Self-introduction*, *Work and Career*, *Travel*, or *Shopping*.

Each topic contains multiple lessons. In each lesson, the system displays an English sentence and provides an audio sample for the learner to listen to. The learner then clicks a record button to speak the same sentence aloud using their microphone. Once finished, the system uses speech recognition technology to convert the recorded voice into text, compare it with the target sentence, and calculate a pronunciation score based on similarity.

In addition to the overall score, the system performs detailed phoneme-level analysis, highlighting mispronounced sounds and giving suggestions for improvement. Results are presented visually so learners can easily identify which parts of the sentence were pronounced incorrectly.

After each practice session, learners can view a performance report that includes their score, pronunciation accuracy, list of errors, and specific feedback for each mistake. The system also maintains a learning history, allowing users to monitor their progress and improvement over time.

The system provides an optional random practice mode, where learners can be given random sentences at their level to improve their speaking reflexes.

To increase engagement, learners can earn badges or rankings based on their speaking performance and consistency. This gamification element motivates users to practice regularly.

From the administration side, system administrators can add, delete, or update lessons, manage user accounts, monitor learner statistics, and adjust lesson difficulty.

All voice data, learning records, and user information are securely encrypted to ensure privacy and data protection.

Operating Environment

The English Speaking Learning System will be developed as a web-based application. The system operates under the following environment:

- **Client-side (Frontend):**
  - Web browser supporting HTML5, CSS3, and JavaScript (e.g., Google Chrome, Microsoft Edge).
  - Audio recording support through Web Speech API or compatible browser audio APIs.
  - Responsive design compatible with desktop and mobile devices.
- **Server-side (Backend):**
  - Web server: Apache or Node.js-based server (Express framework).
  - Operating System: Windows
  - Database server: MySQL or MongoDB.
  - Speech processing services: Google Speech-to-Text API or OpenAI Whisper API for speech recognition.

The system requires a stable Internet connection to handle real-time voice data and cloud-based speech processing.

Design and Implementation Constraints

- **Programming Languages:**
  - Frontend: HTML5, CSS3, JavaScript, Vite, Tailwind, ReactJS,.
  - Backend: Node.js (Express) or Python (Flask/Django).
- **Database:**
  - MySQL (relational model) or MongoDB (NoSQL document model).
- **Speech Recognition Engine:**
  - Web Speech API, Google Speech-to-Text, or OpenAI Whisper for pronunciation scoring and text conversion.
- **Development Tools:**
  - Visual Studio Code (IDE), Postman for API testing, GitHub for version control.
- **Documentation Standards:**

  - Use of IEEE standard documentation format for software requirement and design specifications.
  - All code should follow clean coding standards and include inline comments for maintainability.
- **Security Constraints:**
  - All user credentials and learning data must be encrypted using HTTPS/TLS.
  - Input validation and sanitization must be applied to prevent injection attacks.
- **Performance Constraints:**
  - Voice analysis must be processed and returned within 3–5 seconds for smooth user experience.
  - Server must handle multiple concurrent users with scalable architecture.

English Speaking Learning System provides an interactive, AI-assisted platform for English learners to practice and improve their speaking skills. By combining modern web technologies, speech recognition, and real-time feedback, the system aims to enhance pronunciation accuracy, communication confidence, and learning motivation for users worldwide.

3 Proposed Solution

3.1 Software

3.1.1. Features

Need	*Requirement & Constraints*
As a learner, I want a personal account so the system can track my progress.	**Requirement:** The system must allow users to register and log in. Each user must have a personal profile to store their information, level, and progress data.
As a learner, I want to practice specific communication topics.	**Requirement:** The system must provide a list of speaking topics (e.g., Greetings, Travel, Work). Each topic contains multiple lessons.
As a learner, I want to know how to pronounce a sentence correctly before I try it myself.	**Requirement:** In each lesson, the system must display a sample sentence and provide a sample audio file for the learner to listen to.
As a learner, I want to be able to easily record my voice for practice.	**Requirement:** The system must feature a "record button" that allows the learner to speak into their microphone.
As a learner, I want to be graded immediately after I speak.	**Requirement:** The system must use speech recognition technology to convert the audio into text. **Requirement:** The system must compare the transcribed text with the sample sentence and calculate a pronunciation score.
As a learner, I don't just want an overall score; I want to know exactly what I did wrong.	**Requirement:** The system must perform detailed phoneme-level analysis. The system must highlight mispronounced sounds and provide suggestions for improvement.

As a learner, I want to review a summary after each session and track my improvement over time.	**Requirement:** The system must provide a performance report after each session (including score, errors, and feedback).The system must maintain a learning history.
As a learner, I want to feel motivated to practice regularly.	**Requirement:** The system should include Gamification elements, allowing learners to earn badges or rankings.
As an administrator, I want to be able to manage the lesson content.	**Requirement:** The system must allow administrators to add, delete, or update lessons and topics.
As an administrator, I want to manage the user community and monitor system activity.	**Requirement:** The system must allow administrators to manage user accounts and monitor learner statistics.
As a user, I want my personal information and recordings to be secure.	**Constraint:** All voice data, learning records, and user information must be securely encrypted.
As a learner, I want to receive feedback quickly after I speak.	**Constraint:** Voice analysis must be processed and returned within 3-5 seconds for a smooth user experience.

3.1.2. Software Architecture

- **Frontend (Client-side):**
  - A web-based application built with ReactJS, Vite, and Tailwind CSS.
  - Runs on modern web browsers (Google Chrome, Microsoft Edge) supporting HTML5, CSS3, and JavaScript.
  - Responsible for the user interface, lesson display, and capturing voice input via the Web Speech API.
  - Features a responsive design for compatibility with desktop and mobile devices.
- **Backend (Server-side):**
  - Built using Node.js and the Express.js framework.
  - Responsible for handling business logic, user authentication, data processing, and communication with the database and external APIs.
- **Database:**
  - Utilizes MySQL (relational model).
  - Stores user information, profiles, lesson content, and progress data.
- **Speech Processing Service (AI Service):**
  - The system will integrate with a cloud-based speech recognition API.
  - Specifically, it will use the Google Speech-to-Text API (or OpenAI Whisper API).
  - This service is used to convert speech to text and provide the data for pronunciation scoring and analysis.

3.2 Hardware:

Hardware requirements are divided into two environments:

- **Client-side (User):**
  - Any device (desktop or mobile) capable of running a modern web browser (Chrome, Edge).
  - **Mandatory:** A working microphone to record voice input.
  - **Mandatory:** A stable Internet connection to handle real-time voice data processing.
- **Server-side (Production Environment):**
  - The application will be deployed on cloud infrastructure for scalability.
  - **Frontend Hosting:** A service such as **Firebase Hosting** will be used.
  - **Backend Hosting:** The Node.js backend will be deployed on a cloud platform like **Google Cloud Platform**.
  - **Database Server:** Requires a server running **MySQL** (e.g., Google's Cloud SQL service)

4 Development Plan

4.1 Requirements Analysis

This phase aims to identify the needs of English learners and transform them into well-defined system requirements. The team will analyze the challenges learners face in speaking practice—such as lack of feedback and limited opportunities to converse in English—and determine how technology can address them.
The system targets students, professionals, and ESL learners who want to improve their speaking fluency, pronunciation, and confidence through AI-based interactions.

Key Objectives:

- Understand user needs and learning behaviors.
- Define system goals and scope.
- Specify functional and non-functional requirements.

Functional Requirements:

- User registration and authentication.
- Voice recording and recognition.
- Pronunciation scoring and feedback.
- Learning progress tracking and analytics.

Non-Functional Requirements:

- Speech recognition accuracy above 85%.
- Responsive design for mobile and desktop.
- Support at least 1,000 concurrent users.

Deliverable: System Requirement Specification (SRS) document.

4.2 Software Design

In this phase, the team will create the architectural blueprint of the system, ensuring that all components work together smoothly. The system will adopt a **client–server architecture**, where the frontend handles user interaction and the backend manages data processing and speech analysis.

System Overview:
The frontend will be built using **React.js**, providing an interactive interface for lessons and voice practice. The backend will use **Node.js** and **Express.js** to process user data, communicate with APIs, and store information in a **MySQL** database. The **Google Speech-to-Text API** will be integrated to handle voice recognition and pronunciation scoring.

Main Modules:

1. **User Module:** Handles login, registration, and profile management.
2. **Lesson Module:** Provides topic-based speaking exercises and simulations.
3. **Speech Analysis Module:** Evaluates pronunciation and fluency in real time.
4. **Progress Module:** Displays statistics and personalized learning paths.

Deliverable: Software Design Document (SDD) including system architecture diagram and database schema.

4.3 Implementation

The implementation stage involves translating the design into working code and integrating all system components. Development will follow an **iterative approach**, allowing feedback and improvement at each stage.

Implementation Phases:

- **Phase 1:** Environment setup and database creation.
- **Phase 2:** Backend development (authentication, API integration, scoring logic).
- **Phase 3:** Frontend development (UI screens, voice input, and feedback interface).
- **Phase 4:** Integration of all modules and preliminary testing.

Technologies Used:
React.js – Node.js – Express – MySQL – Google Speech API – Firebase

Deliverable: Working system prototype with core functionalities implemented.

4.4 Testing

The purpose of this phase is to verify that the system works correctly, efficiently, and meets user expectations. Testing will be performed continuously during and after development to ensure system stability.

Testing Types:

- **Unit Testing:** To check individual modules and components.
- **Integration Testing:** To verify communication between frontend, backend, and database.
- **User Acceptance Testing (UAT):** Conducted with real learners to assess usability and accuracy.
- **Performance Testing:** To ensure system responsiveness under load.

Testing Tools: Jest, Postman, and Lighthouse.
Success Criteria:

- No major bugs or crashes.
- Speech recognition accuracy ≥ 85%.
- Average response time < 2 seconds.

Deliverable: Testing Report and Debugged System.

4.5 Deployment and Maintainance

The final phase focuses on deploying the system to a live environment and maintaining its long-term operation. The application will be hosted on cloud infrastructure for scalability and accessibility.

Deployment Plan:

- Frontend hosted on **Firebase Hosting**.

- Backend deployed on **Google Cloud Platform**.
- Continuous Integration/Continuous Deployment (CI/CD) via GitHub Actions.

Maintenance Strategy:

- Regular updates and bug fixes.
- Monitoring system logs and user feedback.
- Adding new lessons, AI improvements, and performance optimization.

Future Enhancements:

- Real-time AI conversations.
- Support for multiple English accents.
- Mobile app version for Android/iOS.

Deliverable: Fully deployed and maintained system ready for user access.

5 Human Resources & Costing Plan

5.1. *Project Organization*

Technical role and responsibility table:

Role	Assigned to	Core focus	Key technical skills	Primary responsibilities	Collaborators
Project lead & backend lead	Đoàn Thành Phát	System architecture, core backend	-Node.js, Express.js -MySQL, Database Design -System Architecture -DevOps (CI/CD, GCP) -Git	-Design the overall system architecture and database schema. -Develop the core User & Authentication modules (Login/Register APIs, JWT). -Lead the deployment process to Google Cloud Platform and Cloud SQL. -Manage the project's Git repository and branching strategy.	-Phạm Phát Lộc: For AI Module integration -Front-end team: to define and provide required APIs
Backend & AI Specialis	Phạm Phát Lộc	AI Integration, Business Logic, Data Processing	-Node.js -External API Integration -Algorithm Design -Unit Testing	-Integrate Google Speech-to-Text API into the backend. -Develop the Speech Analysis module (scoring	-Đoàn Thành Phát: To ensure code aligns with the main architecture -Nguyễn Quang Minh: to align on

				algorithm, phoneme analysis). -Build APIs for Lesson Management and Progress Tracking. -Write unit tests for complex business logic (e.g., scoring).	the audio data format for analysis
Frontend Lead	Nguyễn Quang Minh	Frontend Architecture, State Management, Core Interactivity	-ReactJS -State Management (Context API/Redux) -Web APIs (Web Speech API) -API Consumption	-Set up the frontend project structure, routing, and state management. -Develop the core voice recording feature using the Web Speech API. -Build the main interactive components, especially the practice screen. -Manage API calls to the backend and handle application state.	-Lê Quang Phúc: To ensure component consistency -Back-end team: to consume APIs and handle data
Frontend & UI/UX Specialist	Lê Quang Phúc	UI Implementation, Component Library, Data Visualization	-ReactJS -HTML/CSS, Tailwind CSS -Responsive Design -Data Visualization (e.g., Chart.js)	-Translate Figma mockups into a library of reusable React components. -Build data-	-Nguyễn Quang MInh: To use the shared architecture and state -Phạm Phát Lộc: To

				display pages like the Dashboard, Lesson Lists, and Reports. -Implement data visualizations for user progress. -Ensure the application UI is fully responsive and user-friendly.	accurately display data from the progress APIs

5.2. *Project Development Plan*

Timeline table:

Week	Phase	Key activities	Deliverables
1	Foundation & Design	- Project Kick-off & Scope Definition: Rapidly define MVP (Minimum viable product) features. - High-Level Design: Sketch key user flows, design the database schema, and create a preliminary API contract. - UI/UX: Create simple wireframes for essential screens (Login, Practice, Report). - Environment Setup: Initialize Git repository and all development environments (Node.js, React).	- Lean Requirements & Design Document (including DB Schema and API contract). - Wireframes for core screens. - Configured Git repository.
2	Implementation – Core Systems	- Backend: Develop the User Module with APIs for registration, login, and JWT authentication. - Frontend: Build the UI	- Functional User Authentication System (Frontend + Backend). - Basic frontend application with

		and logic for the Login/Registration pages. - Integration: Connect the frontend authentication forms to the backend APIs. - State Management: Implement global state for handling user authentication.	protected routing.
3	Implementation – Core Feature	- Backend: Set up and integrate the Google Speech-to-Text API. Create the primary endpoint to receive audio and return a score/transcription. - Frontend: Integrate the Web Speech API to capture user voice input on the main practice screen. - Integration: Connect the frontend recording module to the backend speech analysis service.	- A functional end-to-end loop: user can speak into the app and receive a basic score and transcription.
4	Implementation – Enhancements	- Backend: Develop APIs to save practice results to the database. Refine the scoring logic to provide more detailed feedback. - Frontend: Build the UI to display detailed feedback (e.g., highlighting mispronounced words) and a simple practice history page. - Integration: Fetch and display user's past performance data.	-Enhanced scoring and feedback mechanism. - A functional Practice History feature. - Feature-Complete MVP.
5	Testing & Refinement	- Integration Testing: Thoroughly test all API endpoints and the	- Testing Report (including UAT feedback).

		communication between frontend and backend. - User Acceptance Testing (UAT): Conduct testing with a small group to identify usability issues and bugs. - Bug Fixing & Polishing: Address all critical issues found. Refine the UI/UX and ensure the application is responsive.	- A stable, polished, and debugged system.
6	Deployment & Documentation	- Deployment: Deploy the frontend (e.g., Firebase Hosting), backend (e.g., Google Cloud Platform), and database (e.g., Cloud SQL). - Final Testing: Perform smoke tests on the live production environment. - Documentation: Finalize the project report, create a user guide, and prepare the final presentation.	- Live URL for the fully deployed application. - Final Project Report and Presentation.

5.3. Cost Management Plan

Detailed Cost Breakdown

Human Resource Costs

This is the largest cost item, quantifying the value of the team's time and effort.

Total Estimated Human Resource Cost: 30,000,000 VND

+Details: This figure is based on a symbolic salary of 5,000,000 VND/month per student, multiplied by four team members over the 1.5-month project duration.

Infrastructure & Technology Costs

These are the necessary costs to deploy and run the application on cloud platforms.

Total Estimated Infrastructure Cost: 775,000 VND

+Backend Hosting (Google Cloud Platform): An estimated 525,000 VND for 1.5 months. This cost is primarily for usage that exceeds the Free Tier limits.

+Frontend Hosting (Firebase Hosting): 0 VND. The free Spark Plan is sufficient for the project's needs.

+Domain Name: 250,000 VND. This is a one-time fee for a one-year domain registration.

Third-Party API Costs
This covers the cost of the speech recognition service, which is a core technology for the application.

Total Estimated API Cost: 210,000 VND

+Details: This is based on using the Google Speech-to-Text service at a rate of 140,000 VND/month, calculated for the 1.5-month duration. This estimate accounts for the 60 free minutes provided by Google each month.

Tools & Software Costs
These are the tools used during the development process.

Total Tools Cost: 0 VND

+Details: The team will use the free versions of GitHub, Trello, and Figma, as well as open-source tools like Visual Studio Code, resulting in no costs for this category.

Project Budget Summary

Based on the detailed analysis above, the overall project budget is summarized as follows:

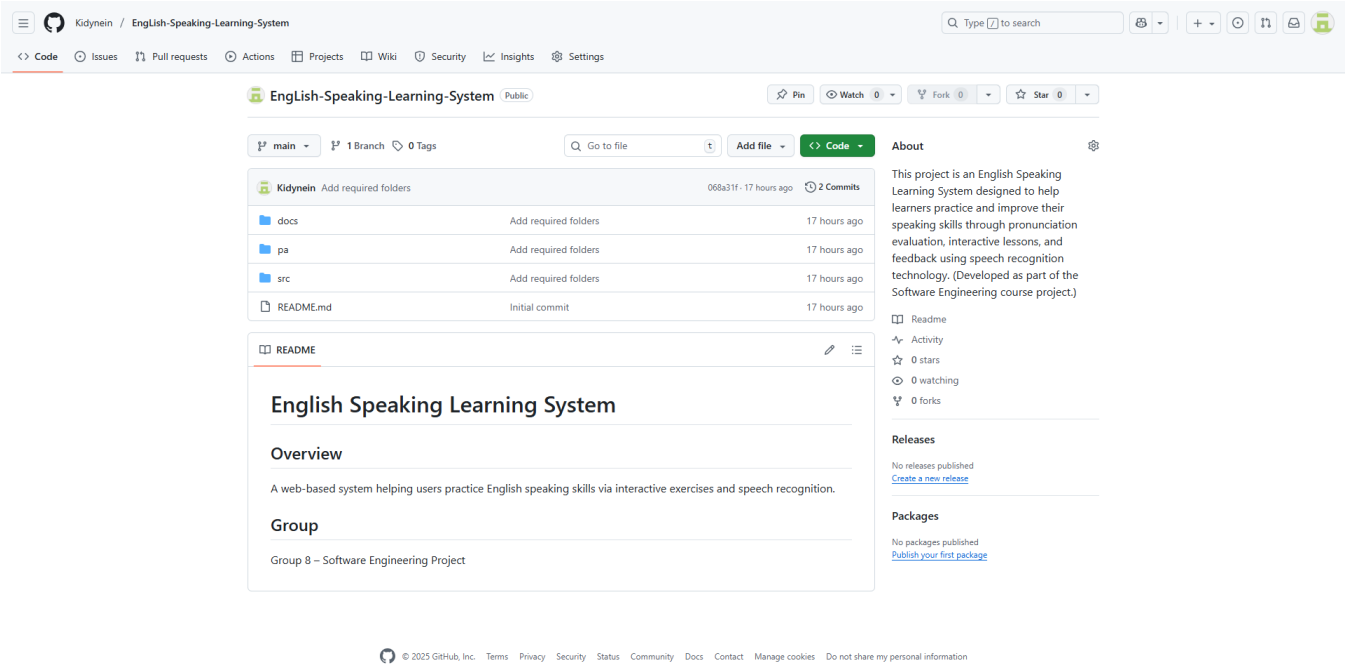
The total estimated cost from all categories is **30,985,000 VND**.
To handle unforeseen expenses, a contingency fund is established at 15% of the total operating costs (infrastructure and API). This contingency fund amounts to **147,750 VND**.

Therefore, the **TOTAL PROPOSED PROJECT BUDGET is 31,132,750 VND**.

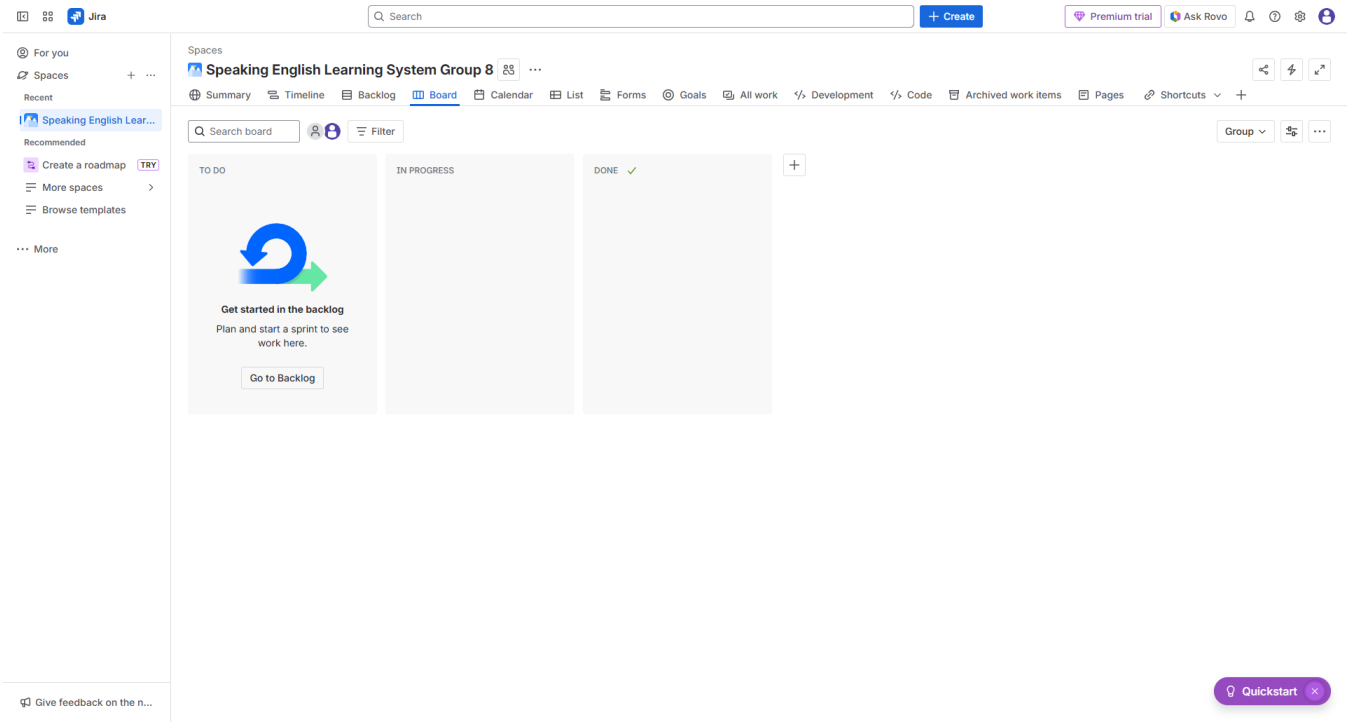
6 Tools setup

6.1. Github

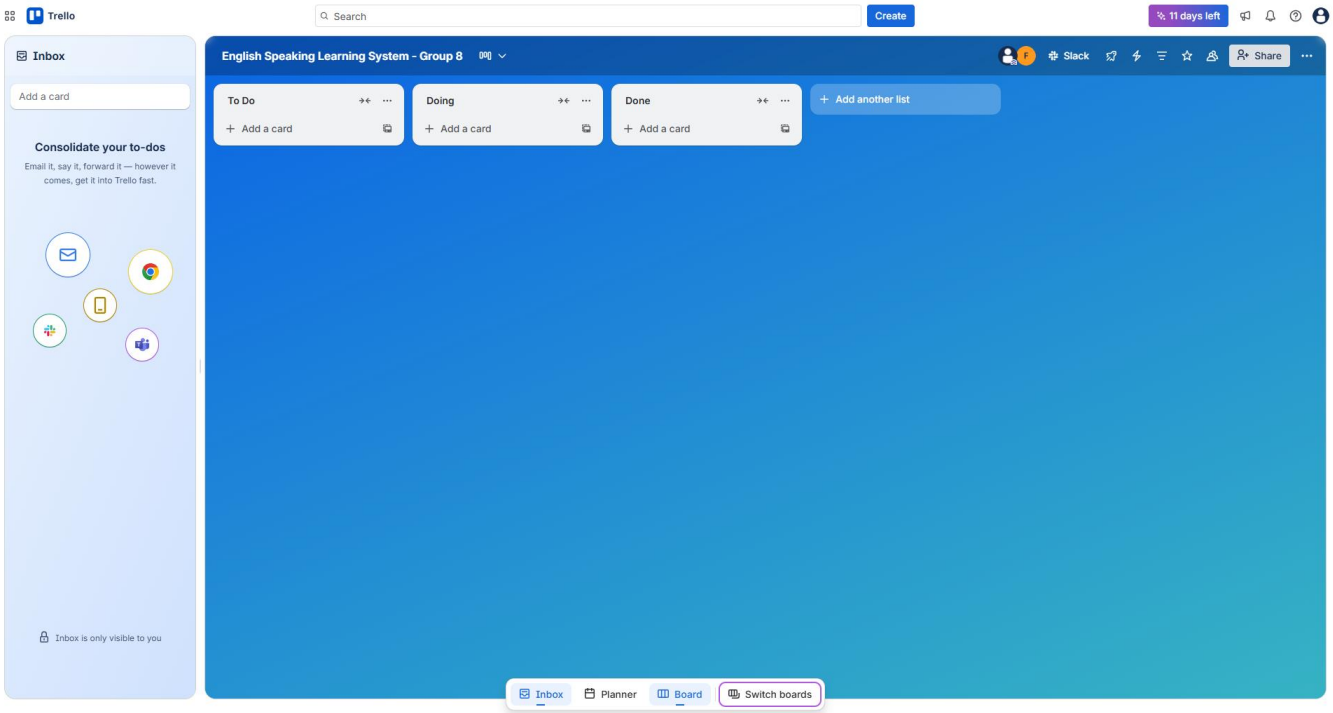
- **Github Repository: https://github.com/Kidynein/EngLish-Speaking-Learning-System**



6.2. Jira


Jira
Search
Create
Premium trial
Ask Rovo
For you
Spaces
Recent
Speaking English Lear...
Recommended
Create a roadmap
TRY
More spaces
Browse templates
More
Spaces
Speaking English Learning System Group 8
Summary
Timeline
Backlog
Board
Calendar
List
Forms
Goals
All work
Development
Code
Archived work items
Pages
Shortcuts
Search board
Filter
Group
TO DO
IN PROGRESS
DONE
Get started in the backlog
Plan and start a sprint to see work here.
Go to Backlog
Give feedback on the n...
Quickstart

6.3. Trello


Trello
Search
Create
11 days left
Inbox
Add a card
Consolidate your to-dos
Email it, say it, forward it — however it comes, get it into Trello fast.
Inbox is only visible to you
English Speaking Learning System - Group 8
Slack
Share
To Do
Add a card
Doing
Add a card
Done
Add a card
Add another list
Inbox
Planner
Board
Switch boards