Hello Friend!

Group “Hello Friend!”
Ken Xu; Gene Yu; Tiange Wang; Derek Bang; Yifan Yu
What’s the Problem?
Problem Overview

Consider the following situation:

Given a foreign environment where people speak language that you don’t understand, you are supposed to complete an important mission: making friends with locals, and chatting with them using social applications everyday.

Prove that if the number of new friends you make is an Exponential random variable with mean not less than 5, then you will end up with being added to the blacklist of Google Translate.
Problem Overview

OK, just kidding.

Suppose that you are a DOTA 2 player. DOTA 2 community consists of players all over the world, speaking various languages. Each region has drastically different abbreviations and conventions from others.

Prove… Eh, imagine that, if none of you speak the same language, the situation is embarrassing: team working becomes impossible to achieve.

*Not even Paxos can help you reach a consensus.*
Another situation…

You are given a penpal who speak Italian. He only reads and writes in Italian and can’t stand waiting for a reply for more than five seconds.

You really don’t want to annoy your penpal, but the time needed for you to copy his message to the Google Translate or your translated message from it is three seconds. With the theory of special relativity, there is no way for you to make him happy.
Core Problem

Thanks to the internet, the world has been fully connected.

People are now free to share information with the entire world.

However, the largest barrier that blocks communications among people from different countries/regions is **language**.
Core Problem

Internet communities are usually unilingual. It’s far from enough.

The language barrier is so subtle that all of us, being affected, do not even realized that our browsing ranges have been severely limited by the language.

Our goal is to break the barrier, make the fragmented world-wide internet becomes a whole.

The major task of our approach is to present everything in user-preferred language.
Inspiration: Chat Wheel

Users press a certain button to send a specific message. But the message will be displayed in the language matched to receiver's account.
But we are not satisfied...

With the wheel, only a few sentences are translated.

We want to translate **EVERYTHING!!!**
Our Solution
Our Idea

Build a network environment where people can browse the entire internet space with their preferred language.

Only then we can say, “Internet truly connects the entire world.”

I am excited to tell you...
Our Idea

Only then we can say, “Internet truly connects the entire world.”

I am excited to tell you...

The first step in the road map is our chatting web-app:
Our Idea

Only then we can say, “Internet truly connects the entire world.”

I am excited to tell you...

The first step in the road map is our chatting web-app:

Hello, Friend!

It’s just so amazing...
What the ... is Hello, Friend!? 
Key Concepts

“Hello, Friend!” is a chatting web-app where you are free to chat with people who don’t speak the same languages with you.

You are free to use your preferred languages.

Key technologies:

- Spring Boot
- Google Cloud
Database? We don’t store any user data.

Too simple? It’s just a beginning!
Messy? No worry!
MVC Design Pattern

Idea: Start from user’s point of view, imagine how users would like the app to perform, and design the controller to satisfy these requirements.

There are four parts in the server:

- Controller
- Templates
- Operators
- Translator
The Main Controller

The entry point of user messages.

Although the functionality of our app is complicated in details…

We end up with **only three** functions in the main controller!!!
Templates

JoinRequest: chat-room registration interface

AdvancedMessage: communication interface between client and server

Room, User, Pair: backend administration tools
Operators

Manager: as its name implies, it is the administrator of the backend system. Upon receiving user request/message, the controller only needs to call methods in Manager to make progress.

Console: helper class for backend maintainers.
Translator

API_access: Google Translate and Speech to Text API. Easy to use, but painstakingly implemented.

```
+translate(text : String, source_code : String, target_code : String) : String
+speech(file_path : String, speech_code : String) : ArrayList<String>
+speech_translate(file_path : String, speech_code : String, source_code : String, target_code : String) : ArrayList<String>
```
Features of Our APIs

HIGHLY ENCAPSULATED: logics behind them are messy and painstaking to understand, but using these API is a piece of cake.
Features of Our APIs

**HIGHLY ENCAPSULATED**: logics behind them are messy and painstaking to understand, but using these API is a piece of cake.

**HIGHLY MODULARIZED**: functionalities are well-categorized and grouped into classes, making the system easy to modify/debug.
Features of Our APIs

**HIGHLY ENCAPSULATED**: logics behind them are messy and painstaking to understand, but using these API is a piece of cake.

**HIGHLY MODULARIZED**: functionalities are well-categorized and grouped into classes, making the system easy to modify/debug.

**HIGHLY CONSISTENT**: interfaces are strictly designed to handle any possible situations, modules can work with compatibility and consistency.
Features of Our APIs

HIGHLY ENCAPSULATED: logics behind them are messy and painstaking to understand, but using these API is a piece of cake.

HIGHLY MODULARIZED: functionalities are well-categorized and grouped into classes, making the system easy to modify/debug.

HIGHLY CONSISTENT: interfaces are strictly designed to handle any possible situations, modules can work with compatibility and consistency.

HIGHLY SECURED: our app does not, and will not, collect, save, or distribute any user information.
Demo with Explanations

3: channelMessage()

3.1: getMessageList()

3.1.1: Pair<User,AdvMessage>[

3.1.1.1: original message

3.1.1.2: translated message