Pangaea

Jonathan Batscha
Eben Bitonte
Faruk Parhat
Guillermo Vargas

Uniting the world through language

Purpose

Pangaea aims to provide a platform to learn new languages through casual, one-on-one conversation with native speakers around the world.

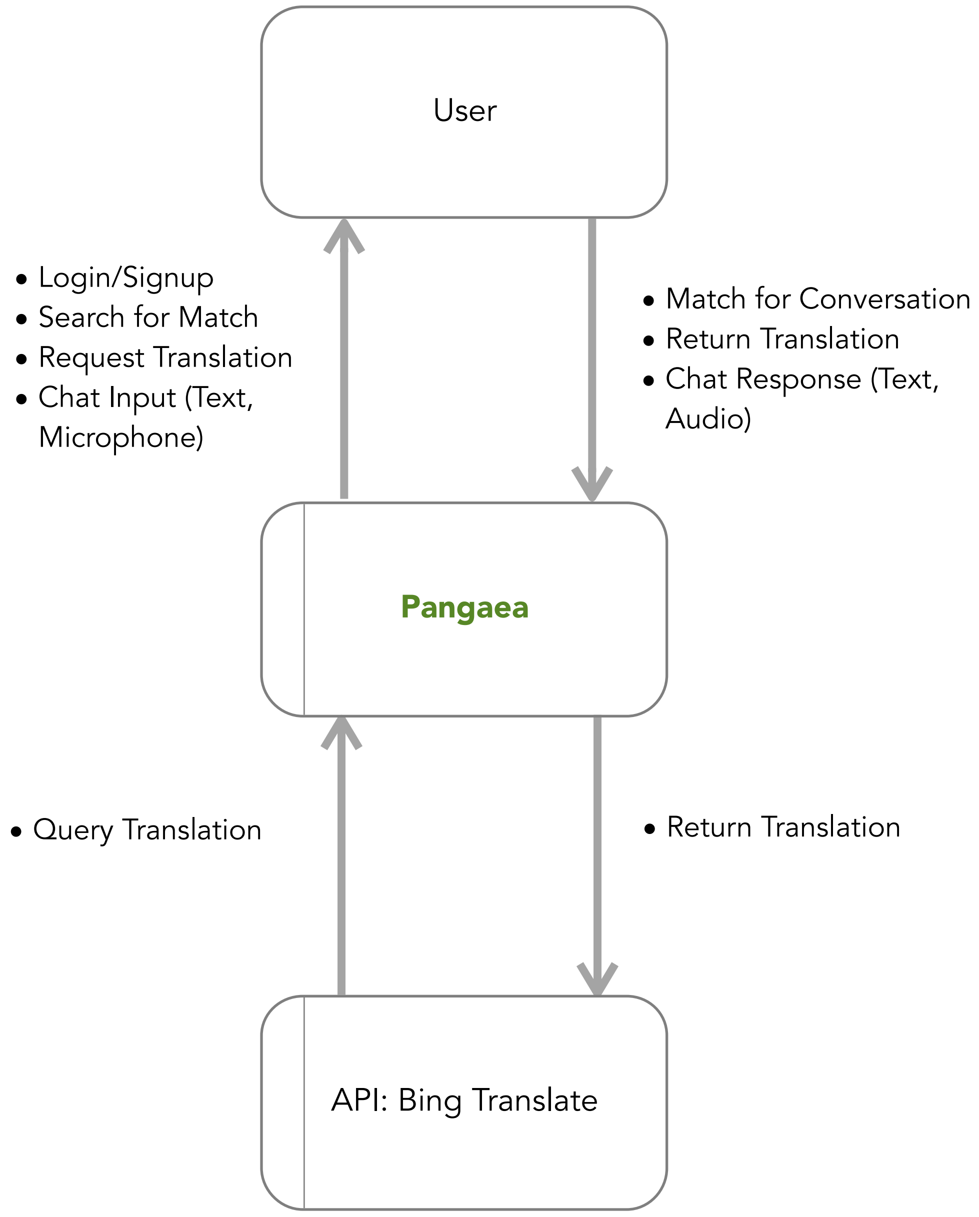
Casual Learning Environment

The casual nature of a chat client conversation should allow users to learn enough of the language to communicate effectively with native speakers, without focusing on the technical details associated with learning a new language.

Existing Solutions

The existing solutions for learning a language are expensive (classes), boring (Rosetta Stone), or focus on technicalities like grammar (textbooks). Pangaea addresses these issues by being a free service with a casual, colloquial conversation with another human being.

Context Diagram



Concepts

A Citizen

A **citizen** is a user of Pangaea. Each citizen has a list of **proficiencies**, the languages in which they are fluent. A citizen also has a username and a password. A citizen can request a chat to practice a given language. Two citizens will then be matched up according to their proficiencies and studies, thus initiating an **exchange**.

An Exchange

An **exchange** is the current chat that is in session. A purpose of Pangaea is to provide the platform for immersion into learning a language with another human being, and for this reason only supports one chat being open at a time - the exchange.

A Proficiency

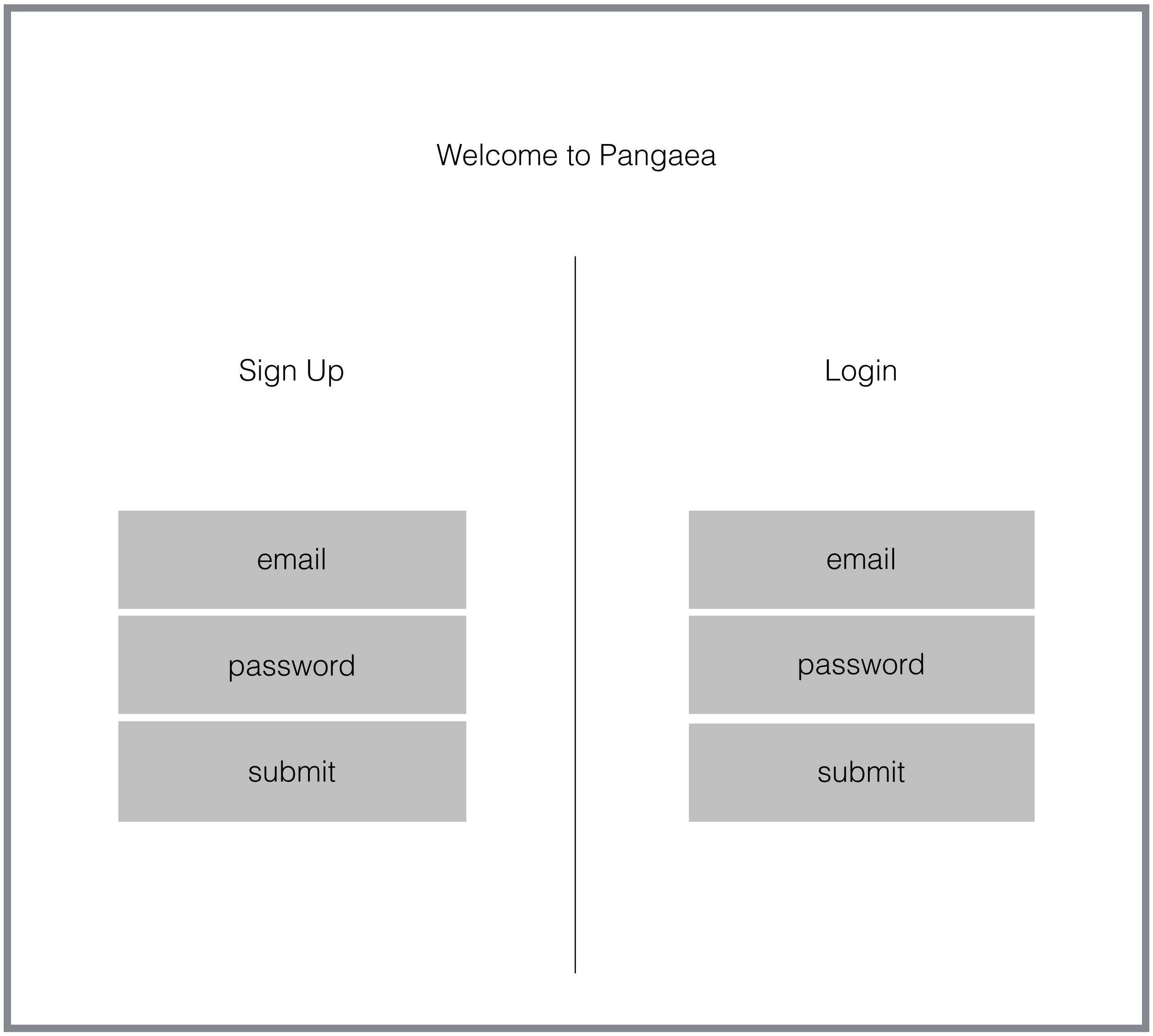
A **proficiency** is a language in which a user is fluent. The proficiency list can contain any number of languages, but it is required for the users to pass a small quiz in order to add a language to their list.

A Study

A **study** is the language that a user is hoping to learn or use during the next **exchange**. A citizen chooses a language as his or her study before requesting a chat and therefore establishing an exchange, allowing for the ability to choose a different study before any exchange. This allows for the flexibility to change what language is currently being learned every time a user starts a new session.

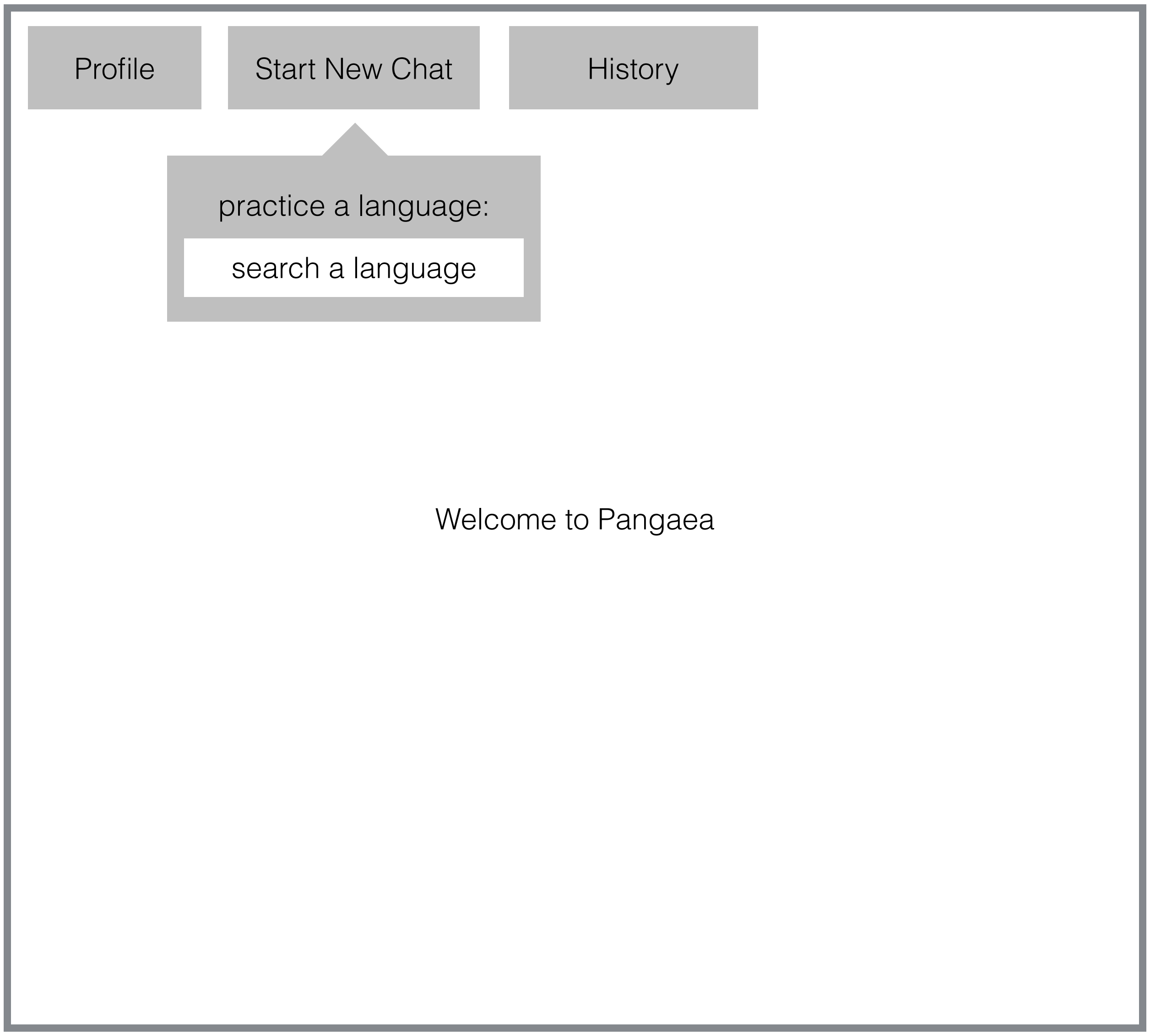
User Interface Wireframes

Login Screen



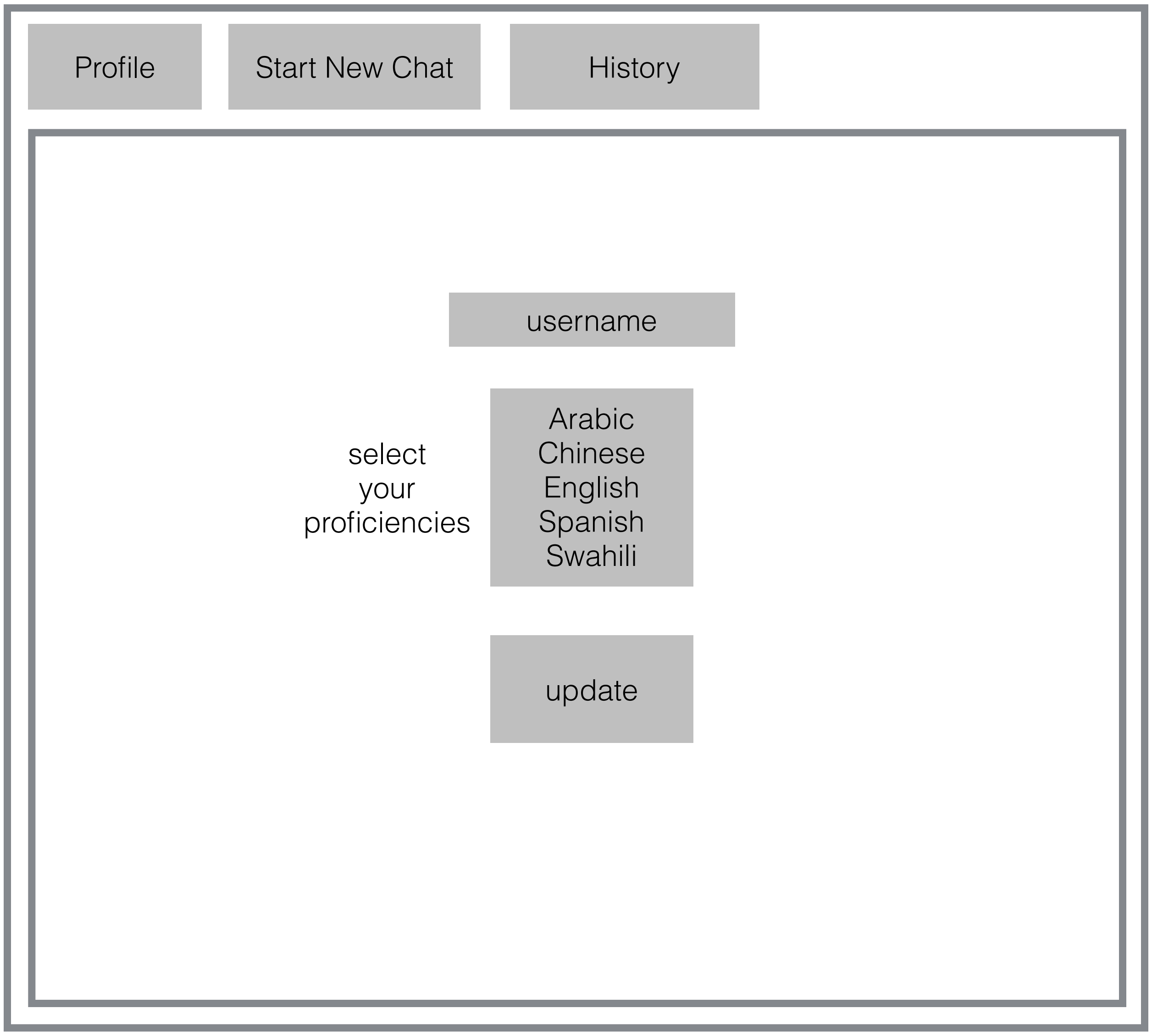
User Interface Wireframes (cont'd)

Welcome Screen



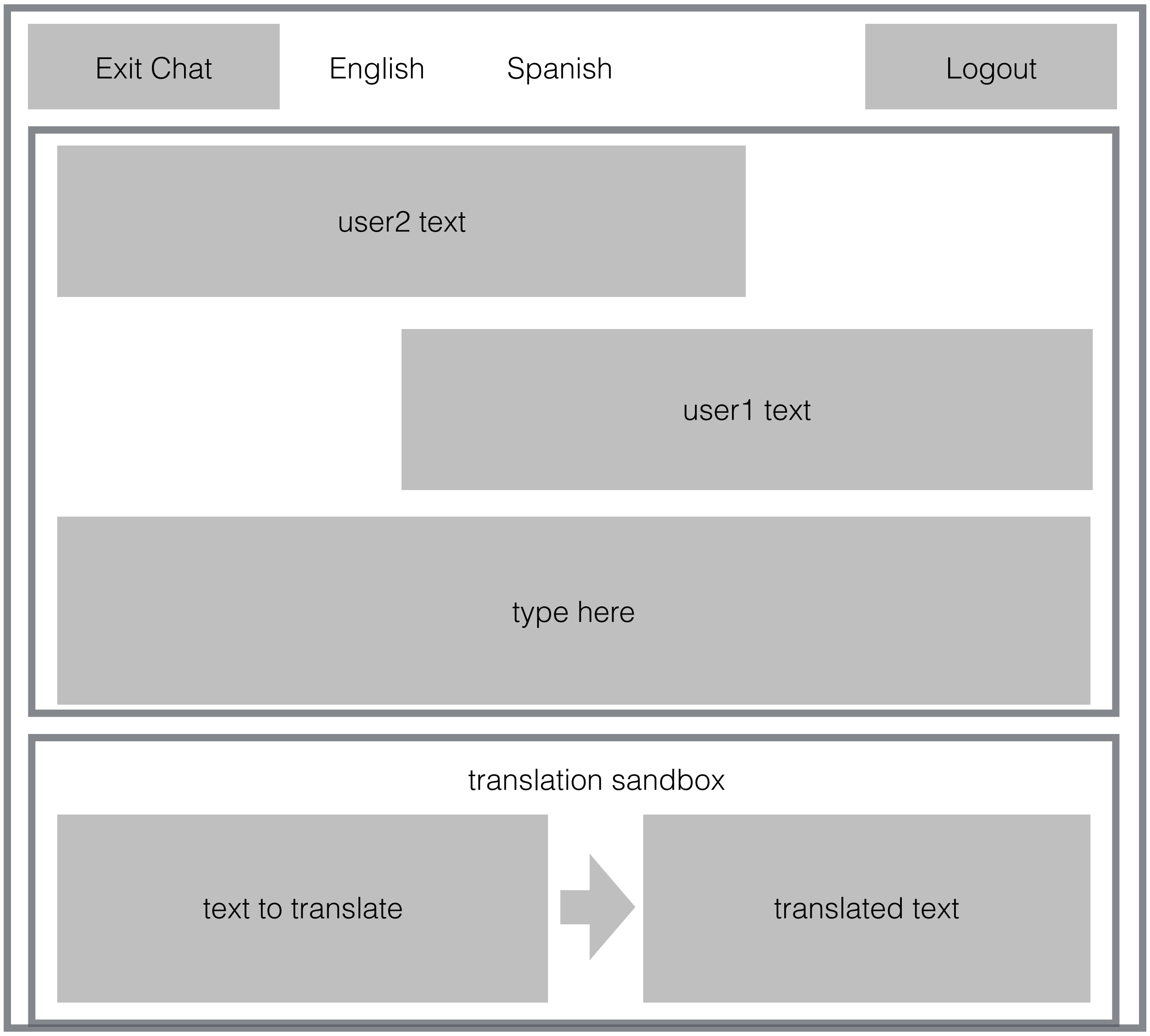
User Interface Wireframes (cont'd)

Profile Screen



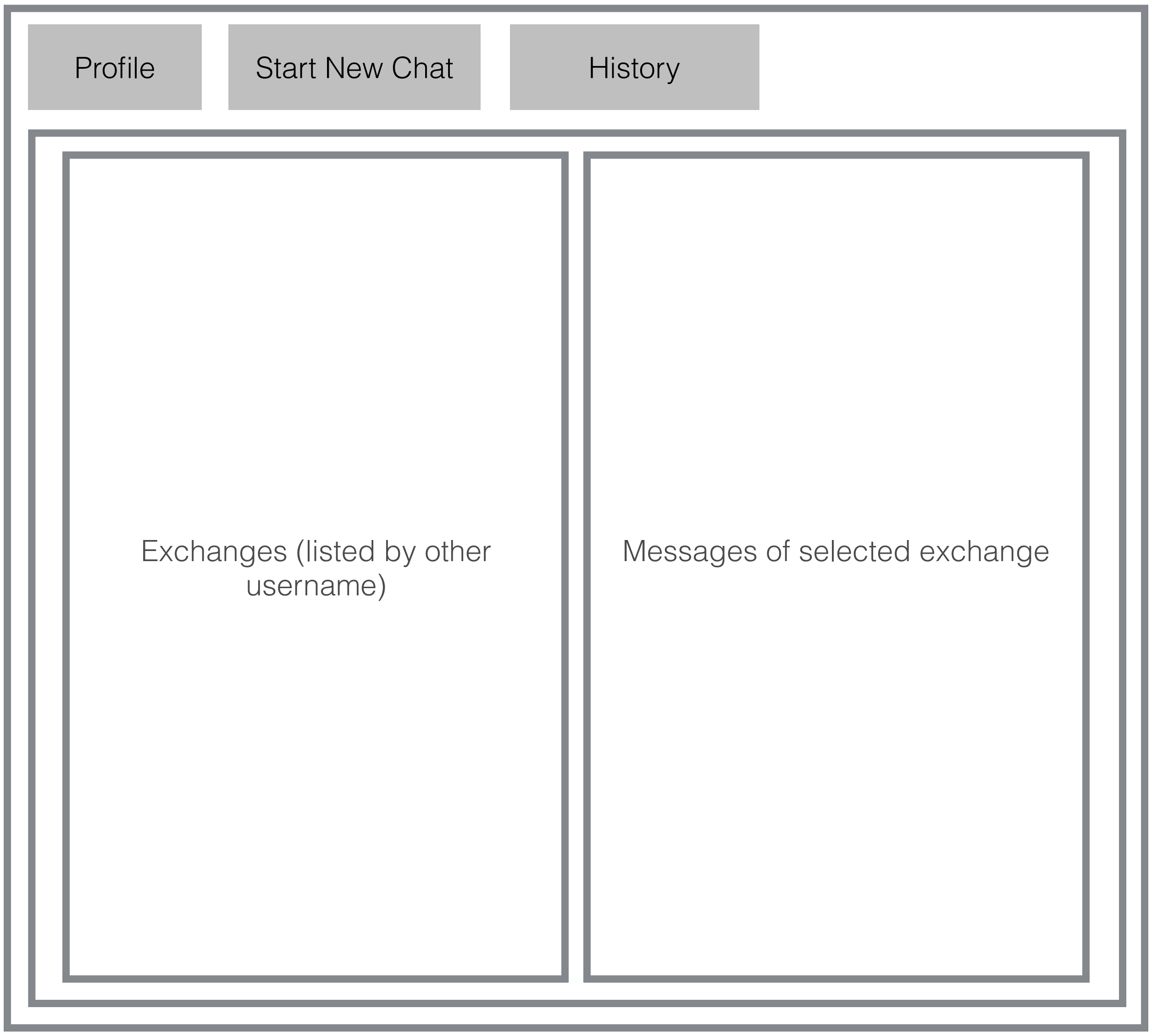
User Interface Wireframes (cont'd)

Chat Screen

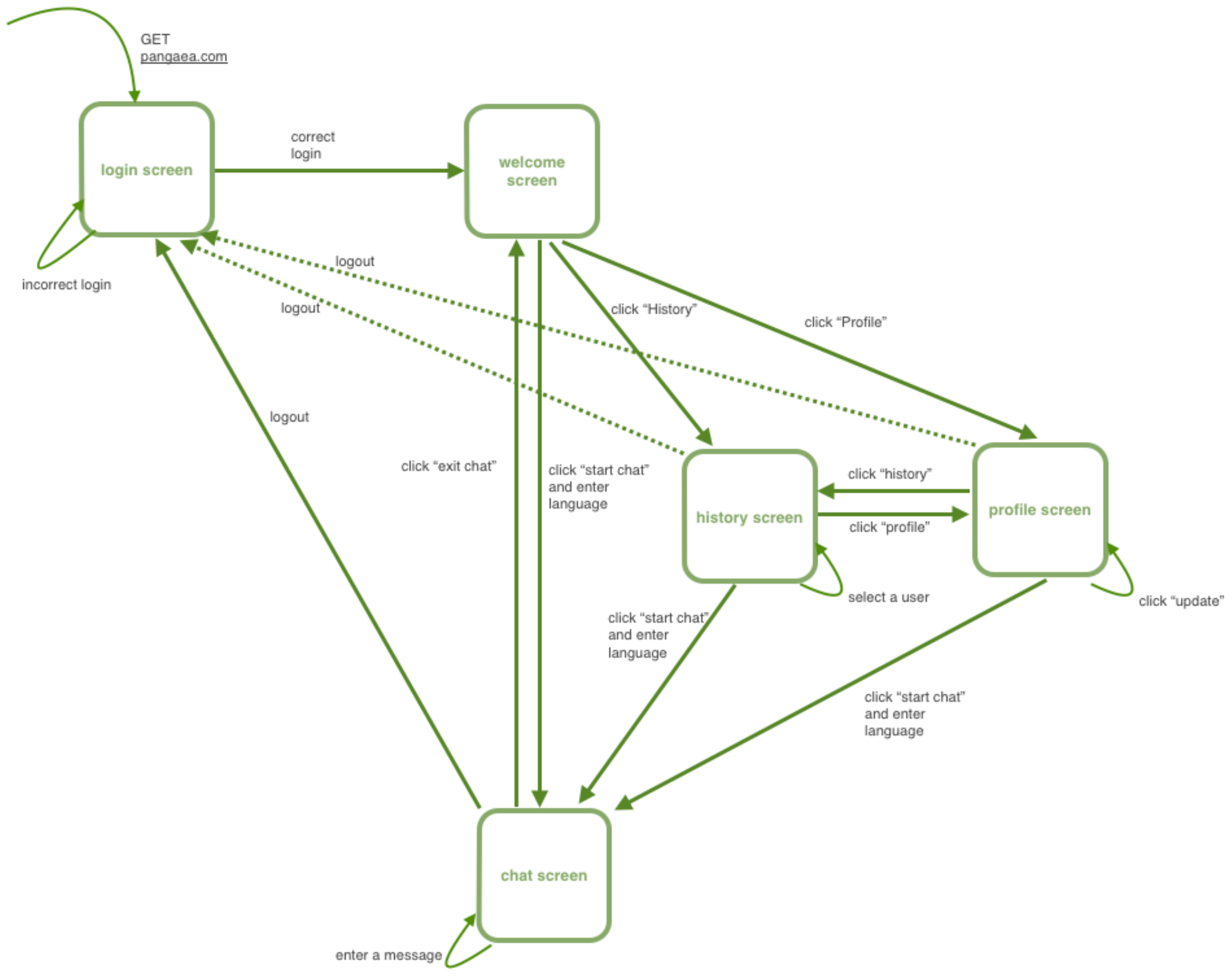


User Interface Wireframes (cont'd)

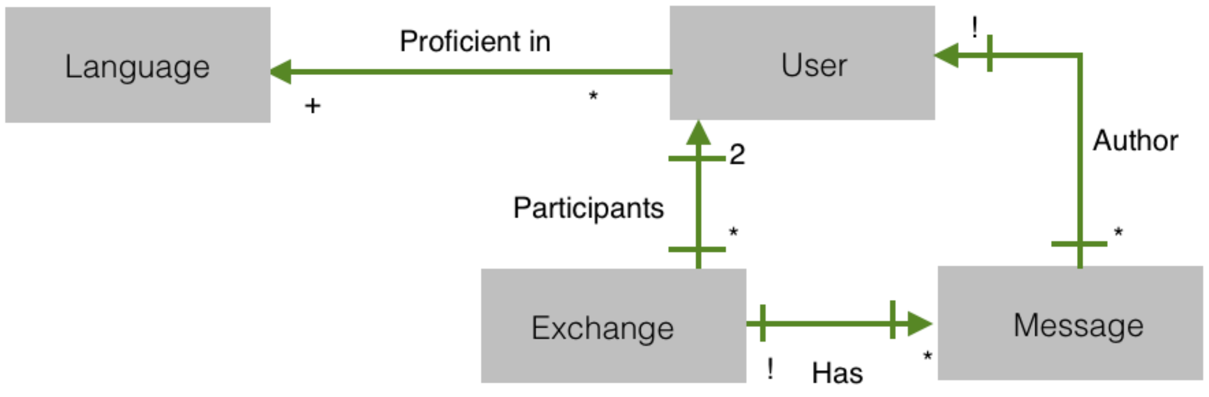
History Screen


Profile
Start New Chat
History
Exchanges (listed by other username)
Messages of selected exchange

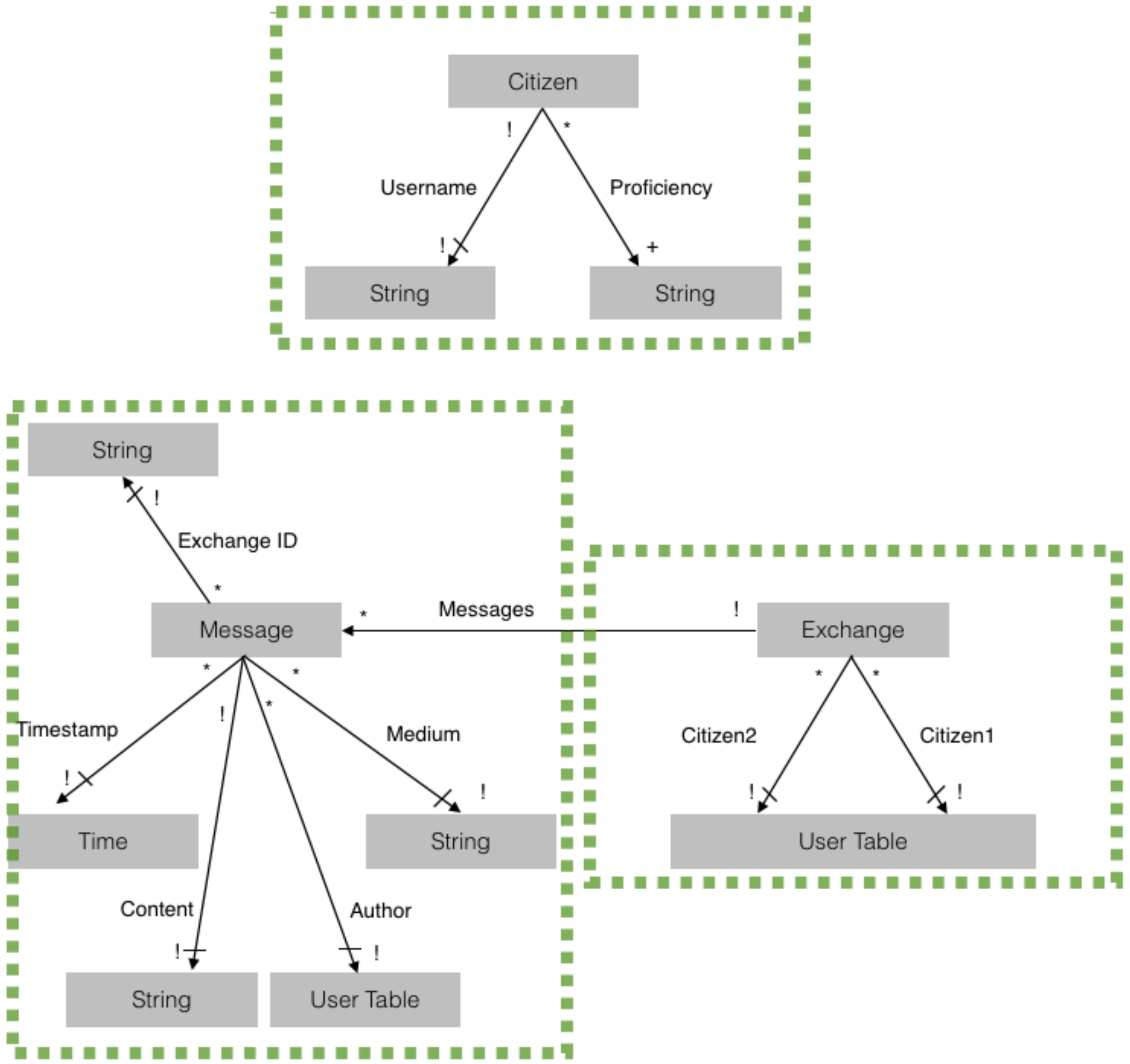
Flow Wireframe



Data Model


Language
Proficient in
User
+
*
!
Author
2
Participants
*
*
Exchange
Message
!
Has
*

Data Design


Citizen
Username
Proficiency
String
String
String
Exchange ID
Message
Messages
Exchange
Timestamp
Medium
Citizen2
Citizen1
Time
String
User Table
Content
Author
String
User Table

Data Design Example

```
Citizen
{
Username: “Joe”,
 Proficiency: [“Spanish”,”English”]
}

Exchange
{
User1ID: “123”,
User2ID: “456”
Messages:[
    {
    Medium: “Text”,
    Author: “Bob”,
    Content: “Hola!”,
    Timestamp: “1416369281003”,
    ExchangeID: “507f1f77bcf86cd799439011”
    },
    {
    Medium: “Text”,
    Author: “Joe”,
    Content: “Muy bien”,
    Timestamp: “1416369281004”,
    ExchangeID: “507f1f77bcf86cd799439011”
    }
}
```

Design Challenges

- Number of Concurrent Exchanges per User
  - This proved to be difficult because we wanted users to be fully involved in their conversations, without being slowed down by the user they are conversing with.
  - We decided to allow only one exchange at a time because a user can leave their exchange at any time if they are not satisfied with the pace of the exchange. Also, it proved to be easier to implement.

- Requirements for Matching Users
  - Ideally, matched users would teach each other their respective native tongues. This made implementation difficult though, since many conditions would have to be satisfied at once for any user to find a match.
  - We circumvented this problem by allowing users to find people who are fluent in the language they want to learn, and giving them the option to invite those people to chat.

- Real-Time Socket Chat vs Messaging
  - Initially we wanted our application to be a real time socket chat. This added complications though since two users would have to search each other's languages simultaneously for a match to ever occur.
  - We wanted to allow users to invite each other to chat as well as communicate in a real time exchange. For this reason we implemented an offline messaging system, for users to coordinate times to talk; and an online real time chat, for users to get a conversation-like experience. We plan to combine these into a single user interface, but keep the implementations separate.

- Logging Chats
  - We want to allow users to review their conversations, while keeping them secure and private.
  - We currently don't log live chats because doing so would run the risk of having one user publicize a conversation that the other user wanted to be private. Also, this method was easier to implement. We are considering logging chats in the future, but we will have to build a mechanism to ensure security of user data if we do so.