# “Transparency”

## Group members

Oleksii Avdieiev 191ADB057
Rustam Adilov 191ADB071
Shokhzhakhon Tursumamatov 191AIB143
Aakarsh Grandhi 191ADB108
Eldor Rakhmonaliev 191ADB052
Fatima Karimova 201ADB002
Sándor Burian 220ADM028
Kanan Sharifov 191ADB053
Ali Zikry 191ADB065

## Concept proposal

“Transparency” is an experimental messenger that breaks the conventions of messengers by eliminating the idea of privacy.

“Transparency” is meant to be an example of what would happen if everyone could access anyone’s personal messages. This experimental app would make people think more about messages and info that could be shared. Moreover, it could be used as a tool to conduct interviews that could be publically accessible without censorship. For the politicians and CEOs usage of “Transparency” could earn trust, by being transparent in their affairs and negotiations. Every word and statement made during negotiations in “Transparency” would be seen by people and that would serve as an additional stimulus to fulfill promises and to keep their word. At the same time, businessmen and politicians that do not use “Transparency” would lose trust, as they would oppose the idea of transparent communications. “Transparency” could be used as an instrument for entertainment, as the dialogue between media stars, characters, or even companies could be possible.

It could not replace, but could be an interesting addition to such messengers as “Telegram”, “WhatsApp” and “Viber”. There are no apparent alternatives or solutions in this niche, therefore to realize the idea, new software for it should be created.

## Requirements

### Functional Requirements

1. It should be possible to register in the system.
2. It should be possible to log in using the chosen login and given access key.
3. It should be possible to save the login information for a device.
4. It should be possible to add users to the “Friend list”.
5. Only users in each other's “Friend list” should be allowed to message each other.

6. It should be possible to find any registered user.
7. It should be possible to see all content that a user was sent or has sent to others.
8. It should be possible to create group chats.
9. It should be possible to share interesting accounts or chats with other users, or in other apps.
10. There should be a “Featured” page, which features the most popular chats in the last 24 hours.
11. It should be impossible to delete messages.
12. It should be possible to reply to and forward messages.
13. It should be possible to send images, videos, and audio recordings as well as video and audio messages.
14. It should be possible to remove users from the friend list, however, the chat is not deleted and is still accessible to all users.
15. It should be possible to hide specific chats, they are hidden only for the user, and all other users can still access them.

### Non-functional Requirements

1. Messages should be delivered in a span of 1 second.
2. Servers should allow up to 100000 active users, this number is scalable and depends on the number of registered users.
3. Servers should be backed up every 24 hours to the AWS cloud servers.
4. Access key consists of 32 randomly chosen UTF-8 symbols (such passwords are needed to strengthen the security of accounts, otherwise people using the app can be easily compromised).
5. The messenger should be available on all iOS and Android systems, as well as on macOS and Windows.
6. Group chats should be limited to 500 users.
7. The registration procedure should allow users to register using only their email and by making up a username.
8. All logins to accounts should be logged, to provide accurate information to the public, in case of unauthorized access.

## Development Process

We are planning on using Rapid Application Development or RAD for short. It is divided into 4 steps:

1. Requirements planning where our team discusses and agrees on business needs, project scope, constraints, and system requirements.

2. User design goes next where we are trying to make our design of a working model. During

this stage, there are several iterations of testing, refining, and prototyping, until all of the requirements conceived in the first stage are fulfilled.

3. Construction. It often goes together with User Design where we focus more on program and application development. We, as a team, can still suggest changes or improvements. Its tasks are programming, coding, unit integration, and system testing.

4. Cutover. The final phase of this development process is where we do data conversation, testing, the changeover to the new system, and user training.

Compared to other development process methodologies it has better quality, good risk control, and more projects completed on time and within budget. The entire process is compressed and thus the new system is built and tested much sooner. Moreover, this method is more suitable for smaller developer teams.

## Tools

We are going to use GitHub for version control, JetBrains ecosystem, which has a large variety of tools for development, will be used for the desktop version of the app, and Android Studio as IDE for mobile development (iOS and Android). Speaking of coding, we will use the Flutter framework for mobile app development, as well as C# for the desktop version. For macOS, Objective C is a great option, following the example of Telegram.

GitHub is a software development platform where users can host repositories of their code projects, it supports version control using Git.

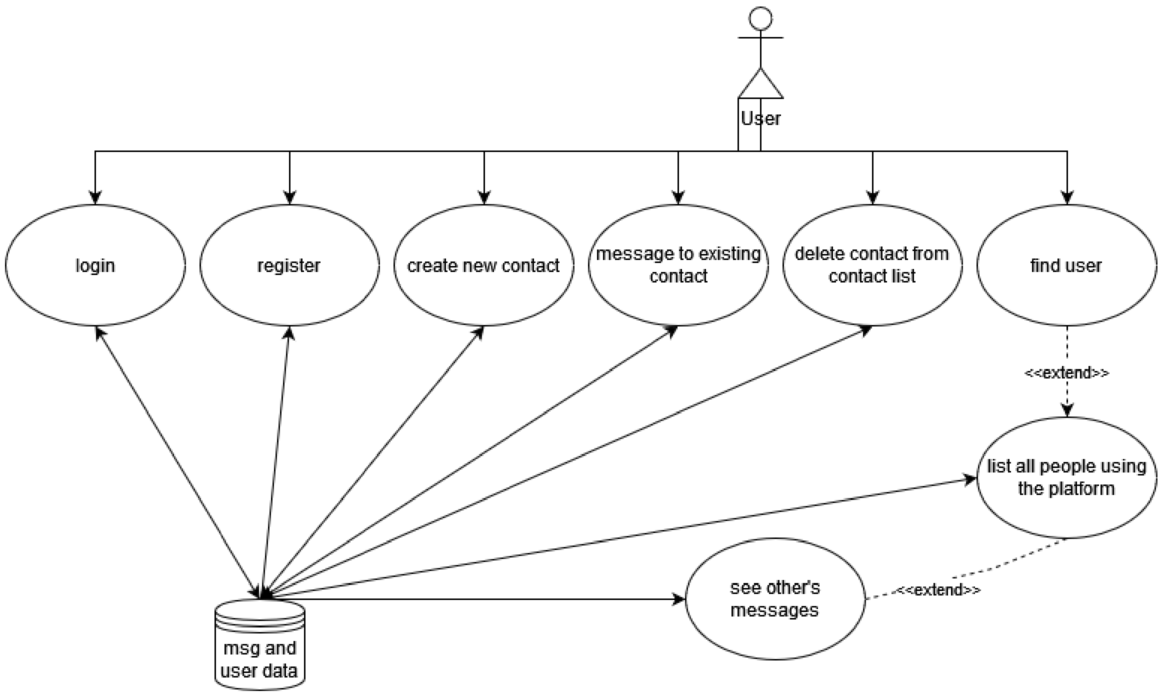
JetBrains ecosystem IDEs (integrated development environments) will be used to code. Jetbrains IDEs Rider will be used for C# language and AppCode for macOS app. Android Studio will be used for mobile development (Android and iOS) allowing cross-platform application development.

We chose AWS cloud storage services such as Amazon Elastic File System, Amazon Elastic Book Store, and Amazon FSx for data storage because they are simple to use, serverless, and offer high-performance block storage services. AWS Data Sync and AWS Snow Family for data migration as they provide online and offline storing facilities and AWS Elastic Disaster Recovery for backup as it may help in case massive data loss occurs (recovery to point-in-time).

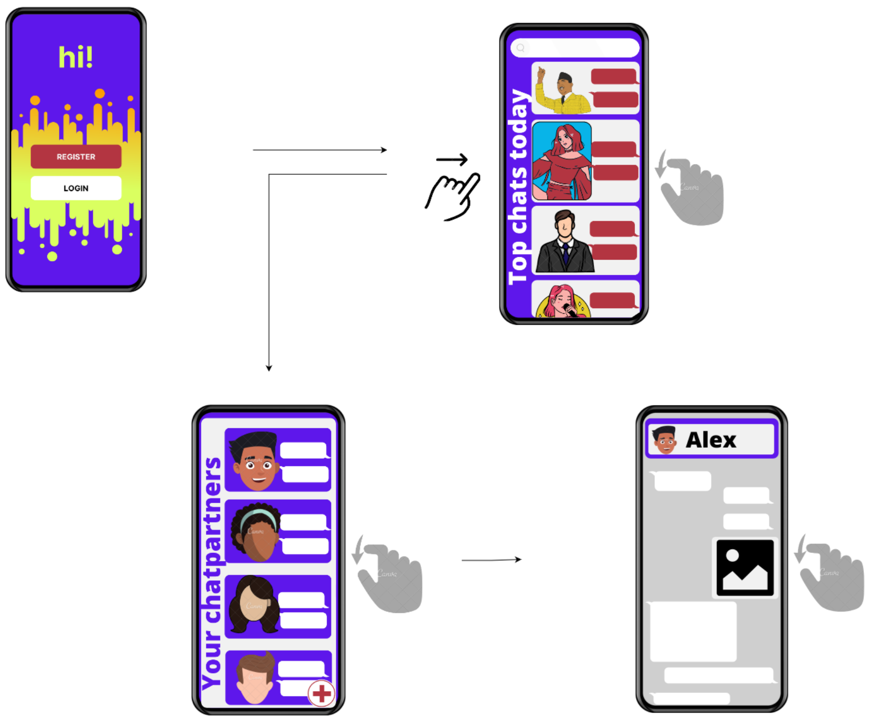
Jenkins continuous integration platform will be used to easily test changes and integrate them into the app during the development and for updates. It optimizes the development process by automating testing and merging the codes. Possibilities of customization by plugins in Jenkins also make it a valuable asset during the development.

## Appendix

### Use case diagram



# Wireframe


hi!
REGISTER
LOGIN
Top chats today
Your chatpartners
Alex