

**Table of Contents**

[1.1 – Intoduction 1](#__RefHeading___Toc600_1387563260)

[1.3 – Operating systems, Software and end-user enviroment 2](#__RefHeading___Toc602_1387563260)

[1.4 – System architecture 3](#__RefHeading___Toc604_1387563260)

[1.5.3 Protocol 6](#__RefHeading___Toc606_1387563260)

[AI 9](#__RefHeading___Toc608_1387563260)

|  |  |
| --- | --- |
| date | update |
| 30.10.2024 | First version |
| 04.11.2024 | Tables added |
| 01.11.2024 | Reworked |
| 11.12.2024 | Added languages |
| 06.01.2025 | Major bugfix |
| 08.01.2025 | Passwords added |
| 20.01.2025 | Mistake detection added |
| 01.02.2025 | Network protocol update |
| 04.02.2025 | Encryption update |
| 11.02.2025 | POS color scheme update |
| 26.02.2025 | Documentation update |
| 04.03.2025 | UI update |
| 16.03.2025 | Major bugfix |
| 24.03.2025 | Coloring rework update |

# 1.1 – Intoduction

The purpose of this document is to present the concept of my project, a chatting platform for rare languages leaners called "InterHolon".

InterHolon is a chat-platform. It supports simple messenger functions such as group chats, personal chats and creating your own groups. The key feature is POS enlightening, which gives different colors for different parts of speech (nouns, verbs, e.t.c) and so provides you with easier understanding of the grammar.

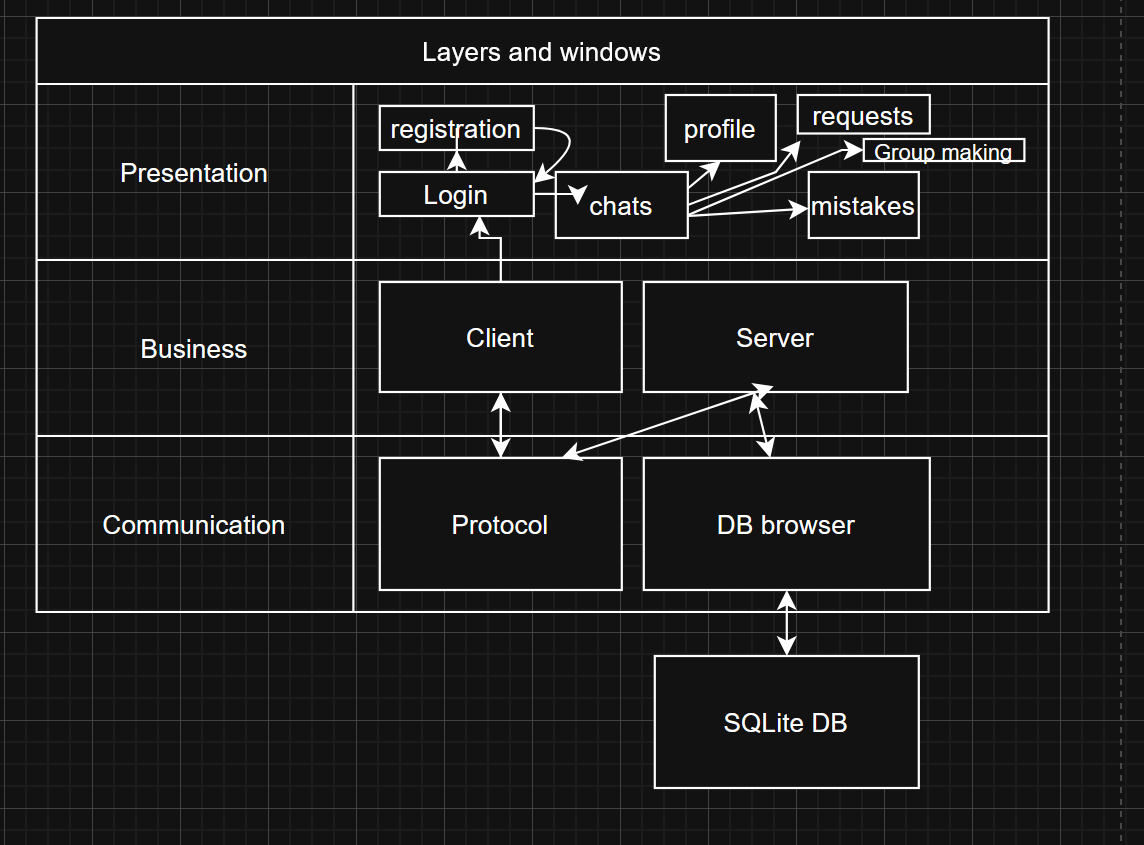
The app is a chatting platform for native speakers of none common languages and linguistics involved people. It’s design is inspired by modern chatting platforms such as telegram, whats app and signal. The application includes screens of mistakes, requests, group creation and chatting as well as registration, login and profile pages.

# 1.3 – Operating systems, Software and end-user enviroment

My app is ran on windows operated systems.

It needs no special knowledge but reading this documentation, installing python and the following libraries:

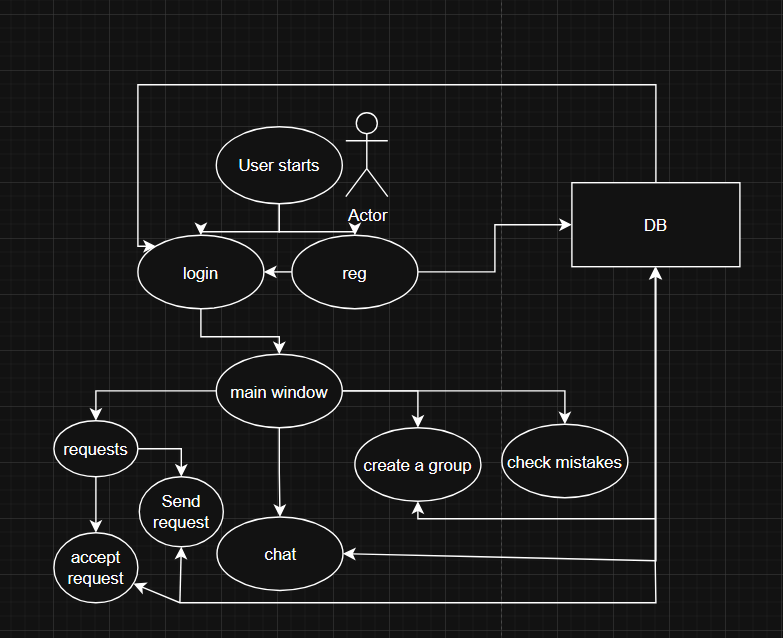
cryptography



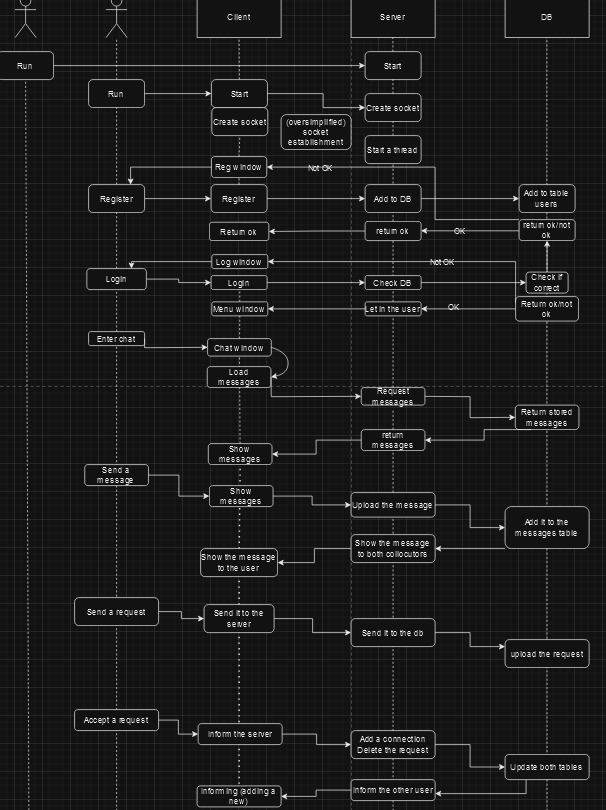
# 1.4 – System architecture

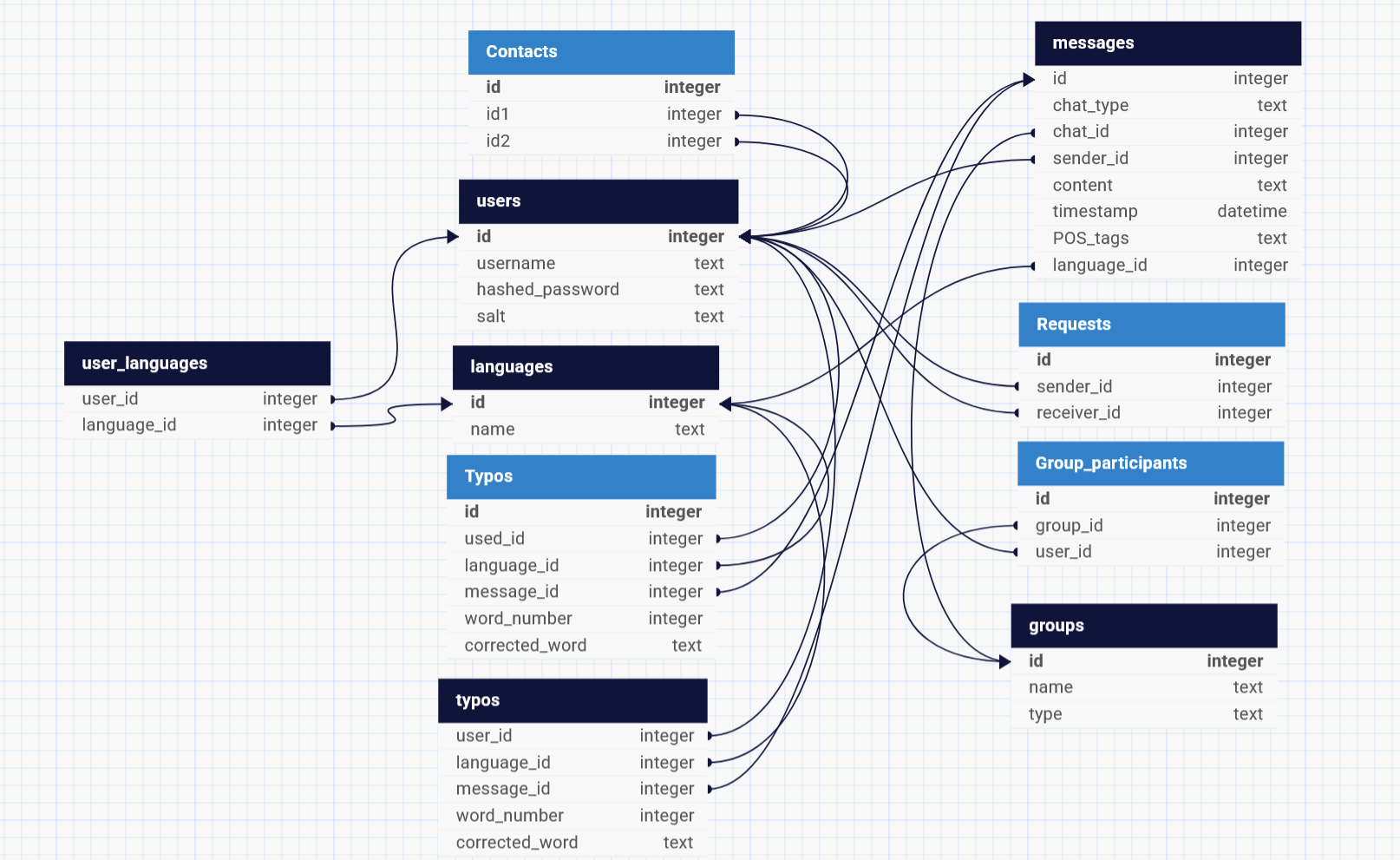
My system contains a bunch of windows separated into three layers

Also I prepared a user-case diagram showing the expected experience of a user



A sequence diagram showing the correlation between layers as well



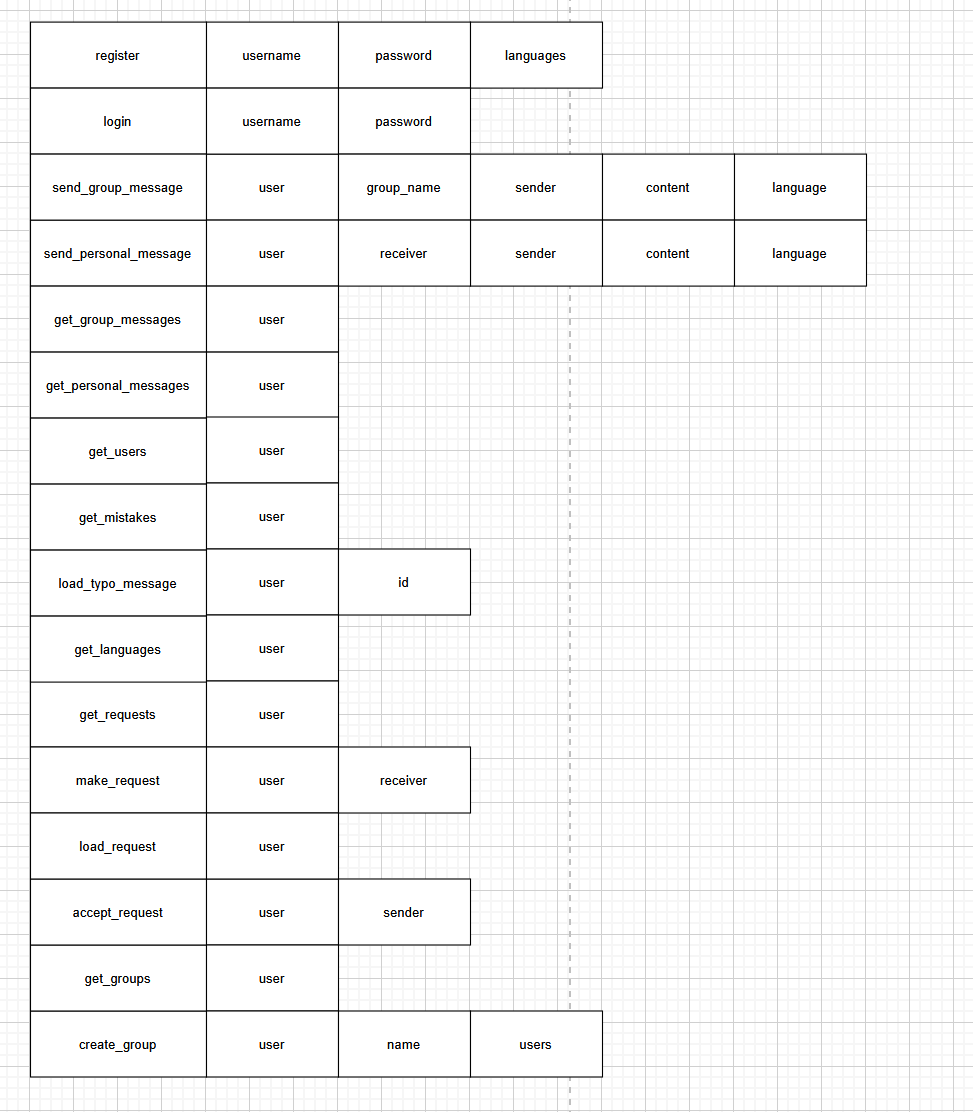
Also I have a scheme of all the data tables I need and their connections

# 1.5.3 Protocol

Also I prepared a blank version of a protocol for sockets

Protocol:I use JSONs (json.dumps(dict), to be exact) to transport data between server and client. The dict always includes “action” key, value of which provides type of the query. Other keys of the dict are specified for each request separately

For each possible request I listed needed arguments and expected responds:



The current scheme looks this way

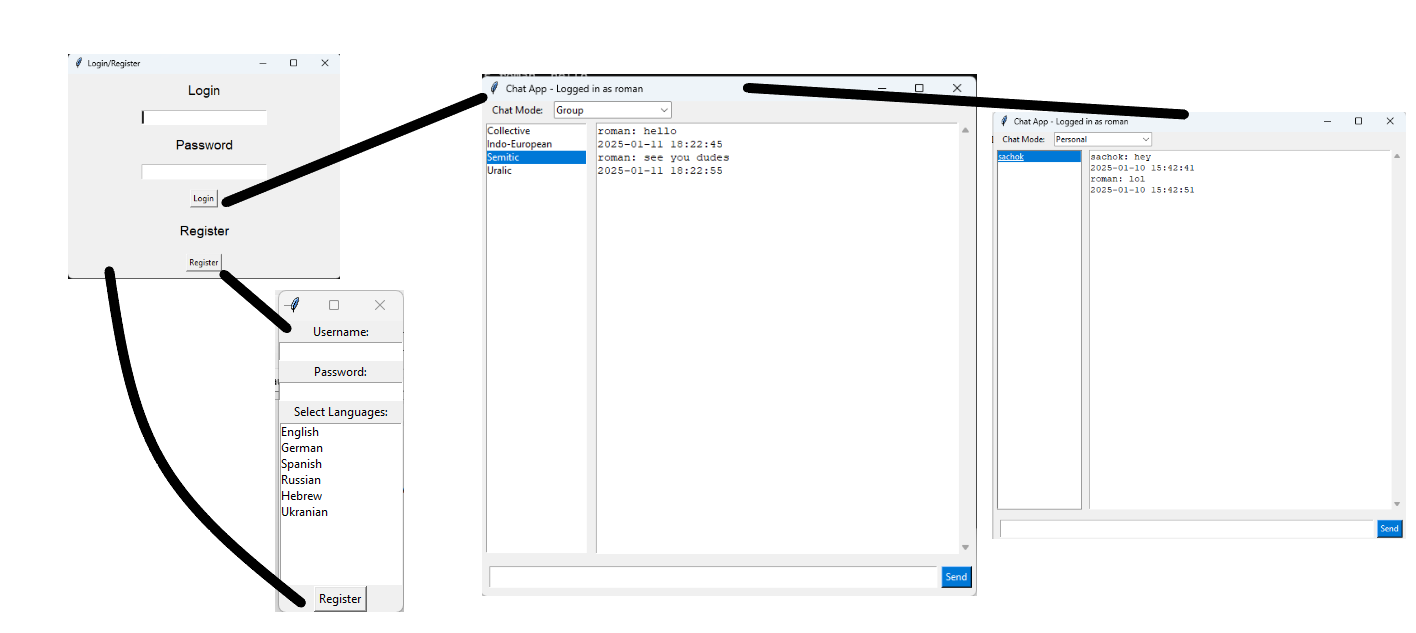
1.5.4 passwords are stored only hashed with scrypt algorithm and salted

Every message transported is encrypted by my special algorithm:

In the beginning of a session server and user exchange asymmetrical encryption keys. During each following sending we generate a new AES key, encrypt the message with it, then encrypt the AES key RSA key and sends the encrypted RSA key and the data encrypted with it.

1.5.5 classes are Server and Client so far. In the next update gonna separate into BL and GUI both of them

Workflow



Classes:

I have following classes

ChatAppLogic – responds for the inner logic of the client, ChatApp’s parent

ChatApp – responds for connecting the inner logic of the client to the UI

ChatServer – responds for the server handling clients

ChatServerUtilities – contains all the inner server utilities and logic, ChatServer’s parent

Message – responds for analyzing messages

ChatServer class methods:

initialize\_db – initializes the database

handle\_client – the function that’s called to handle each client

process\_request – performs the orders from a request. Calls one of the following funtions:

register\_user

login\_user

send\_group\_message

get\_group\_message

send\_personal\_message

get\_personal\_message

get\_users

get\_mistakes

load\_typo\_message

get\_languages

get\_requests

make\_request

accept\_request

get\_groups

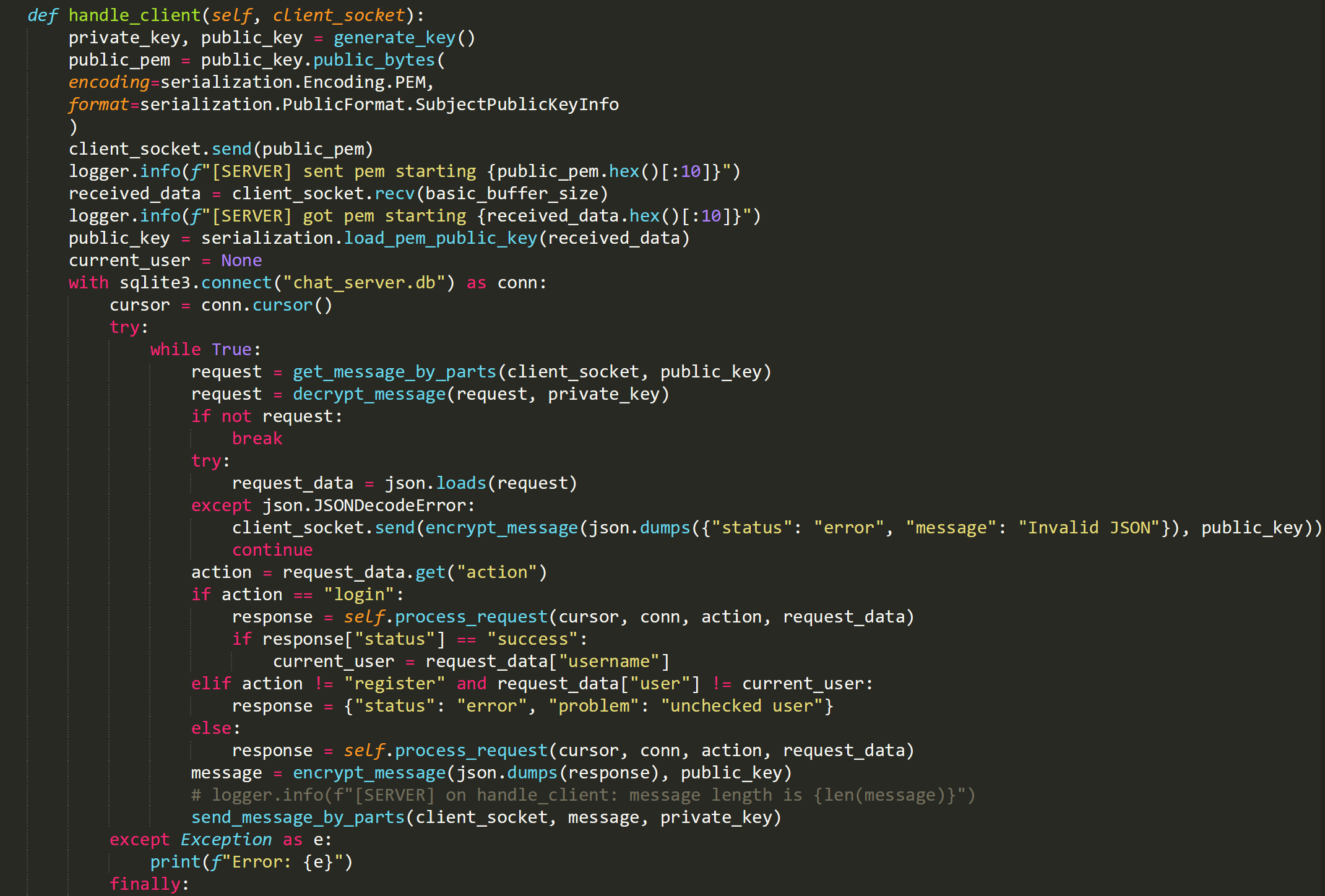
create\_group

start\_server – starts server

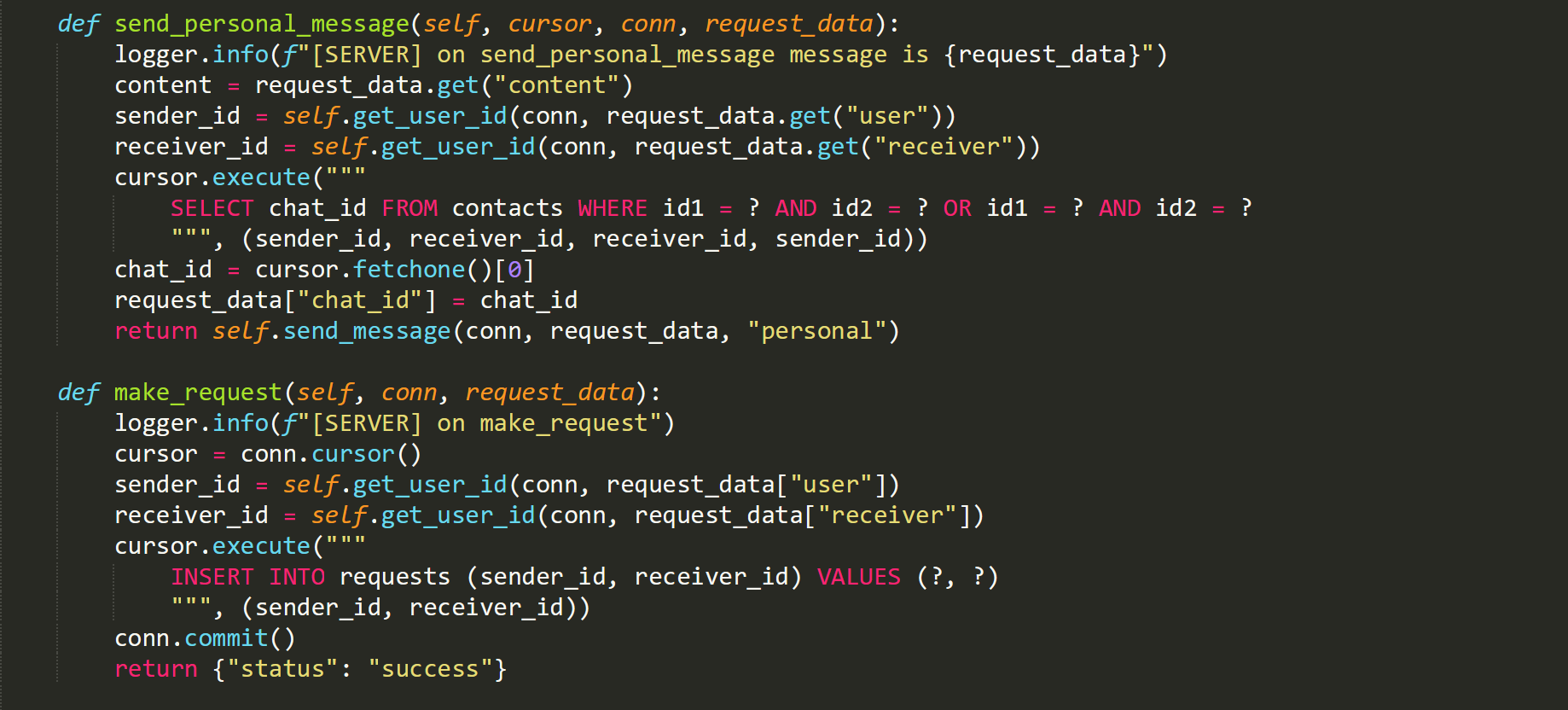
# AI

Stanza and Phunspell libraries, which I use for POS tagging and finding typos are made using AI. Also I used AI a lot during learning about the project. Copilot, chatGPT and deepseek provided me with lots of knowledge that’d be far harder to find without them

Code exmples:









**tests**

1. Registration test:

1.1 Registration existing users

1.2 Registration no languages

1.3 Registration

1.4 Login wrong username

1.5 Login wrong password

1.6 Login

2. Message test

2.1 SQL injection

2.2 message in Hebrew

2.3 message in English

2.4 message in Russian

2.5 message in Ukranian

2.6 message in Spanish

2.7 message in German