

Problem

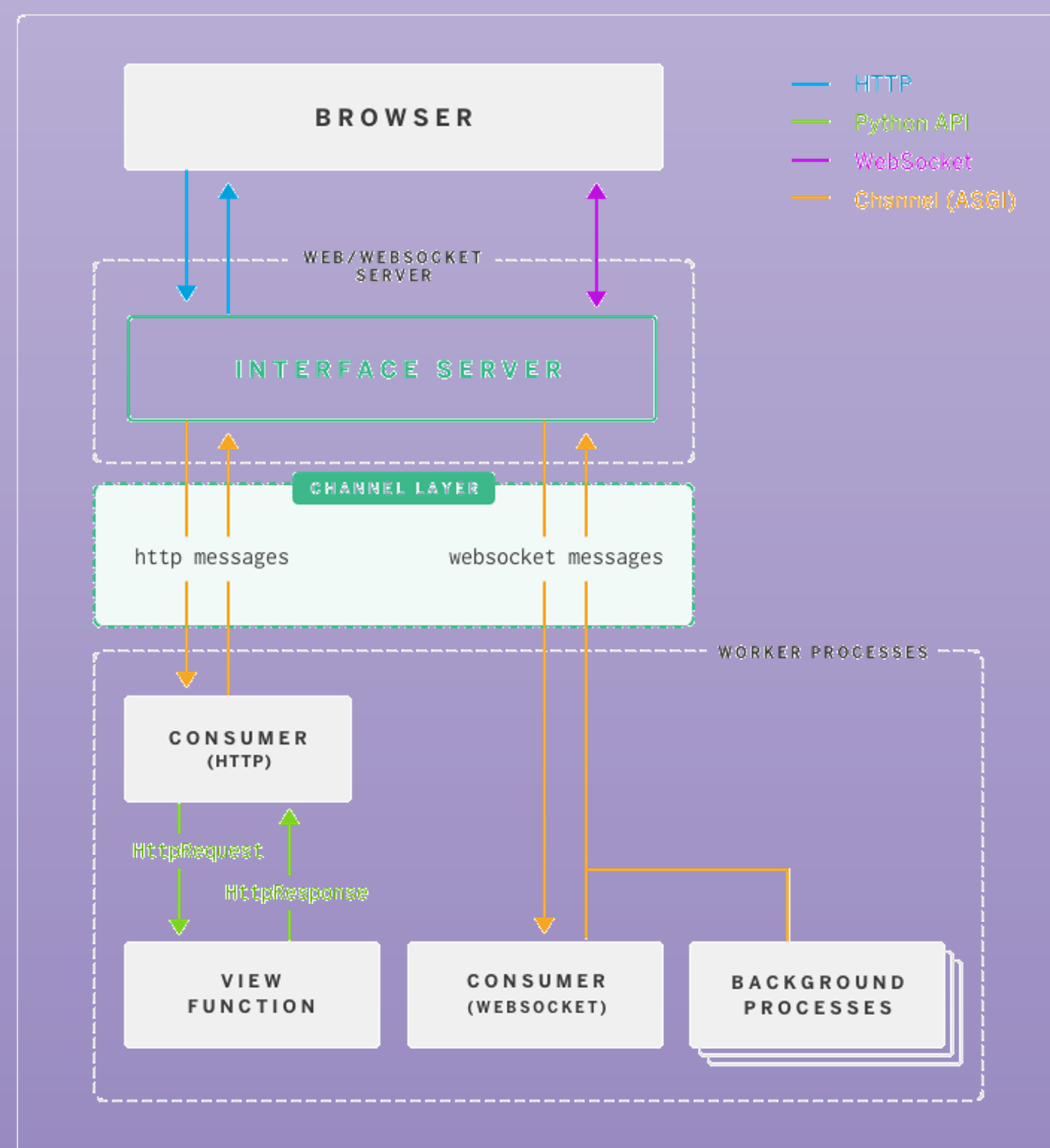
The creation of guacamole ID has implemented the technology for facial recognition and continuous authentication. Usually, authentication software only authenticates the users once at login, and if the system is left unattended, other unauthorized users could access sensitive information. As for a messaging application, if the system is left alone or others oversee your messages, there is no form of intelligent privacy and security system to detect them. This is the problem this software is trying to solve.

Solution

The solution to this problem is to design a messaging web application with continuous authentication (Guacamole ID) that can detect intruders and unauthorized users and stop them from accessing sensitive information, pretending to be the actual user and messaging on their behave and overseeing their private messages.

Implementation

- Django framework was used for backend with PostgreSQL database server.
- Data gets stored in the database and can be retrieved later.
- Ajax, Websockets, and Redis were used for communication, messaging system, data transmission, and caching messages for low latency.
- Html, Javascript, and Bootstrap CSS were used for the frontend.



Current System

GuacamoleID chat is a real-time messaging system that can be extended with a computer vision application called GuacamoleID that provides on-device continuous authentication technology. Primarily, it's a "FaceID" for computers. It authenticates users continuously or on-demand with the capability of detecting unauthorized users, locking, and protecting the system from intruders.

Requirements

Create a web chat application as follow:

- Register
- Login
- Logout
- Change password
- Go through facial recognition to access
- Real time messaging system.

Verification

Test Case1: User Registration, login, logout..

Purpose: Verify that database is responsive.

Prerequisites: PostgreSQL and Redis server running.

Input: Register with email and choose password.

Expected Output: Account created and can be modified.

Test Case2: Add friend, Receive request, accept, decline.

Purpose: Verify functionality.

Prerequisites: PostgreSQL and Redis server running.

Input: Search, send request, accept/decline request.

Expected Output: Friends added to friends list and can be removed.

Test Case3: Chat with friends, and switch to a different room.

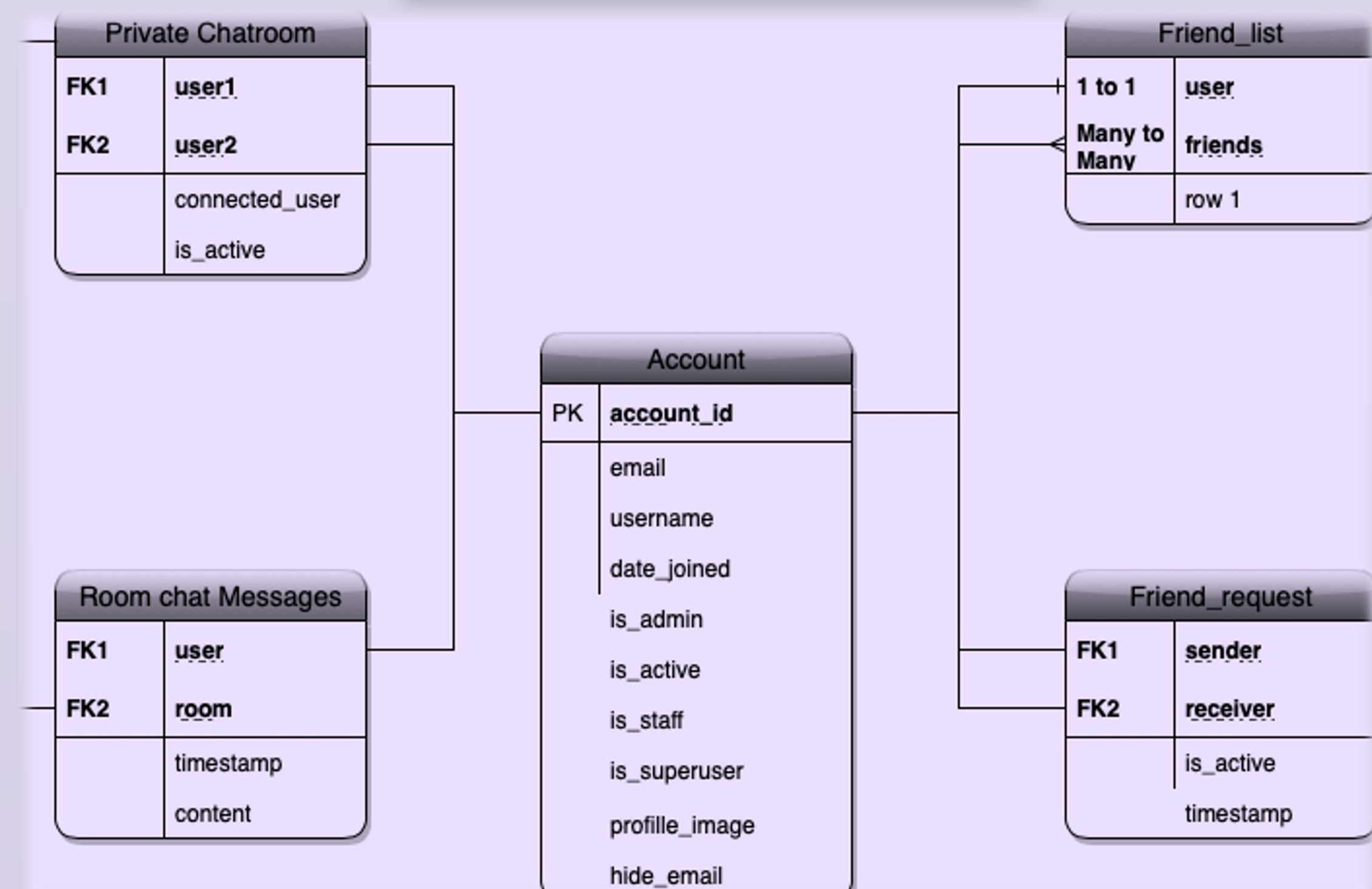
Purpose: Verify that WebSockets make connections.

Prerequisites: PostgreSQL and Redis server running.

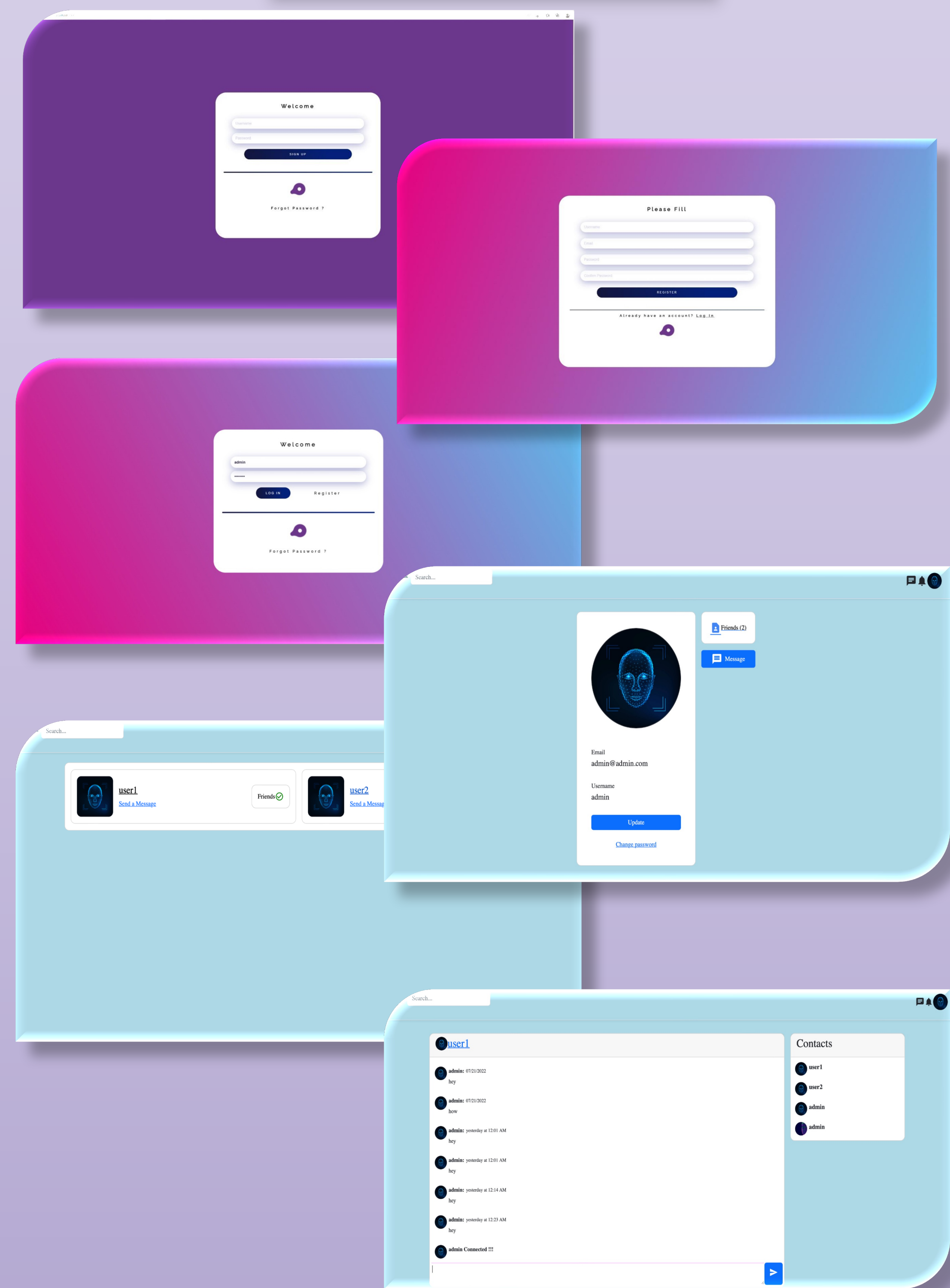
Input: Send message, Receive message.

Expected Output: Receive message with the date and can be revisited.

Database Design



Screenshots



Summary

- Web chat application with database capable of being extended with guacamoleID authentication system.
- Create an account, add friends, delete friends, search for friends, and have a real-time messaging system.
- Real-time messaging using WebSockets protocols and Redis as a cache system for low latency transmission.
- The front end creates a register, login, logout, and change password functional buttons.

Acknowledgments: