Chat App Analysis

# Problem Area

Other similar chat systems, such as WhatsApp and Telegram, are blocked on my school’s internet filtering rules due to privacy concerns and safeguarding issues. There is strong demand within my school for a chat system for the exclusive use by the Sixth Form that can be used to on the school’s network. The aim of my project is to create a chat system for Sixth Formers for this purpose. Core goals behind the app will be privacy and equal permissions (all students will have the same level of access to the app).

# Prospective users

The chat system is intended for the use of Sixth Formers in my school. There are around 20 classes (of around 10 people each) in Lower Sixth and Upper Sixth each, and the chat system will allow people to sign up with their name and form group – this will make it easier for other users to find each other on the system and chat with each other.

# Research of existing systems

## WhatsApp

WhatsApp is an instant messaging mobile, desktop and web app that also supports VoIP calling. It was created by WhatsApp Inc. in California and was first released on the App Store for iPhone in August 2009 yet was later acquired by Facebook (now Meta Platforms) in 2014. As of June 2024, WhatsApp has almost 3 billion unique users worldwide.

**High-level feature overview** (*at time of writing, not exhaustive*)**:**

* Users can sign up with a cellular mobile telephone number on the mobile app and are then able to set a username and profile picture. On mobile, users can set up biometric authentication to secure the app on devices that have the necessary hardware.
* The desktop and web apps mirror your phone’s chat and call functionality, although WhatsApp must first be set up on the mobile app.
* The app allows users to send text messages, voice messages, images, videos, documents, locations, emojis and stickers within chats, while users are also able to react to messages with emojis. Messages can be edited and deleted.
* Group chats can be created with up to 1024 members. Chat members with administrator permissions can manage the chat, control who can join, moderate the group and kick people from the chat. A name, description and image can be set for the group.
* The app also supports voice and video calling between individuals and group calls with up to 32 participants.
* All messages and calls are end-to-end encrypted to ensure user privacy and security.
* WhatsApp also provides several privacy settings giving users the ability to restrict sees their online status and last seen, limit who can view their picture and status, disable read receipts, send disappearing messages and more.
* The WhatsApp status lets users share photo, text, video and GIF updates that disappear after 24 hours and reply to others’ status updates through private messages.
* Users can back up settings, chats and media to cloud storage providers Google Drive or iCloud (if they already use them).

## Telegram

Telegraph is another popular instant messaging platform, which launched for iOS in August 2013 and Android in October that year. The company behind it was founded by Nikolai and Pavel Durov. Company data indicates that 950 million use the software across the world.

**High-level feature overview** (*at time of writing, not exhaustive*)**:**

* Registration requires a smartphone or one of a few NFTs issued in December 2022.
* Similar messaging features to WhatsApp, alongside ‘silent messages’ where users can send messages that do not trigger a notification.
* A single group chat can support up to 200,000 members, with similar features to WhatsApp. Group analytics provide statistics for group administrators to monitor activity and engagement.
* Voice and video call support similar to WhatsApp. A single voice chat can accommodate thousands of listeners, while a group video call can support up to a thousand video viewers.
* Telegram channels (which can private, with restricted access, or public, allowing anyone to join) support sending messages to an unlimited number of subscribers.
* Web, mobile and desktop apps.
* Telegram stands out for its privacy and security features including end-to-end encryption, secret chats (messages that can self-destruct and cannot be forwarded), self-destructing media, two-step verification, username-based communication and anonymous admins.
* Personalisation including custom themes, notifications and chat folders.
* Developers can create custom chatbots.
* Cloud backup to Telegram’s cloud, except for secret chats.

## Proposed Features of My Chat App

I intend to a make a relatively simple chat app that will allow users to sign up with a unique username and password, choose their class and set a display name with their actual name (this will not have to be unique). Users will be able to change their username, display name, class and password even after signing up. Users will need to login to authenticate themselves and will be able to logout. For the sake of convenience, session cookies will be used to ‘remember’ users and authenticate them if they have logged in recently. Users will be able to able to view sign in, sign up and personal profile pages (to edit their own details and delete their account).

Group chats will be the core of messaging functionality and will support up to 500 users each (this is in case the school expands in the future). Users will be able to create, join and leave group chats. To ensure privacy, to join a group chat, users must first receive an invite request from an existing member of the group chat and can then accept or reject the request. Within the group chat, users will be able to send text messages of up to 2000 characters at a time. To keep things fair, there will be no administrator functionality as one of the key principles behind this application is all students should have equal permissions, although users must first be invited to a chat for privacy. This will also allow students to run chats themselves. Groups will also have group information pages, detailing group name, members, the number of members other information and will display the option for users to leave the group.

There will also be a notifications pane for each user to view any invite requests they have received and who has left or joined any groups the user is part of.

Users will also be to view profile pages of other users, which will list user details.

# Final Objective List

1. Users should be able to sign up with details including a unique username, a display name, their class name and a password.
2. Users should be able to login and logout.
3. Users should be able to delete their account.
4. Users should be able to change their username, display name, class and password.
5. Users should be able to view the profile pages of other users which should list username, display name, class and number of mutual groups they have in common.
6. Users should be able to create group chats.
7. Users should be able to invite other users to join group chats (as long as they are already a member of the chat).
8. Users should be able to leave group chats.
9. Users should be able to send text messages within the group chat.
10. Users should be able to view all the text messages within the group chat they are part of and who sent them.
11. Users should be able to accept or reject group chat invitations.
12. Users should be able to receive and view notifications of group chat invitations and when someone has joined or left a group chat they are part of.
13. There should be persistent storage of user and chat data, in a database.

# Proposed solution

## Client-side

Given the ubiquity of modern smartphones among teenagers, I first considered a mobile app. However, not everyone is able to download the certificate necessary to connect to the school’s internet, meaning many are dependent on mobile data for an Internet connection within school. However, there are several areas within school with very weak signal (there is a courtyard with high walls around them, in which download speeds are extremely slow) meaning a mobile app would be constrained by this issue.

The other device that every pupil in my school has access to is a school-issue laptop. A desktop application would not be ideal as school device management software restricts the installation of apps not approved by the school’s IT support team.

These issues make a **web-based application** (optimised for usage on desktop computers and laptops) the ideal solution. The server’s static IP address could then be regularly changed to get around the school’s internet filtering system.

## Server-side

The chat app will use a server rather than letting users message each other directly, as this will allow me to centralise the storage of all data and make it easier to implement group chat functionality.

Given my experience with it, I decided to choose **Python** as the server-side programming language. Although there are other popular web frameworks for Python, the ones most widely used and with the most documentation (important for a beginner such as me) are Flask and Django. While I have some experience with Django, although its ‘batteries-included’ approach to web development means that development is fast and easily scalable, ‘opting out’ of some of those built-in features such as the database object-relational mapper (ORM) can unnecessarily complicate your project. In essence, the fact that Django comes with a lot of ‘built-in’ features means that using all of these features together streamlines the development timeline. However, some of the features abstract much of the complexity away, meaning using some features such as the ORM would not allow you to implement the complex algorithms (such as cross-table parameterised SQL in group A) necessary for scoring high marks. Trying to use Django without the ORM requires altering writing the rest of the project in an unorthodox and what I believe is an overcomplicated format that removes the benefits of fast development that Django’s ‘batteries-included’ approach normally provides.

Instead, I chose **Flask**. Flask is a lightweight and flexible micro web framework for Python. While it lacks built-in features such as a database abstraction layer (such as Django’s ORM), Flask supports ‘extensions’ that add features as if they were implemented in Flask itself. The beauty of this approach is that the developer can choose where to use external libraries and where they would prefer to implement the logic themselves. This will allow me to write the complex algorithms necessary to score a high mark myself.

## Database Management

For persistent storage of user and chat data, I will use a relational database management system as this provides several advantages over a file management system including a locking system for concurrency and consistency control, much faster read/write and search times and built-in security.

Given my prior experience with SQL and that many of the example Group A algorithms mention SQL, a SQL relational database management system (RDBMS) is the obvious solution to handle my database needs. In the context of a chat system for a Sixth Form of around 400 pupils, most SQL RDBMS should be able to support a load of this level. Another constraint was my lack of a budget – this meant I was restricted to open-source relational database management systems.

The first option I considered was SQLite, a simple and lightweight RDBMS. A major advantage was its small footprint – SQLite takes up very little disk space and memory, potentially enabling me to choose cheaper hardware for the server later on. It also requires little setup time as it is serverless. On the other hand, one limitation with SQLite is its scalability, and may be less suitable for applications with high concurrency and many users reading and writing from the database at the same time. Another issue is that SQLite lacks a built-in user authentication system, although you can set file permissions to the .db file in which the database is stored.

Instead, I chose **MySQL**, a RDBMS that is very fast and scalable. In addition, MySQL supports user management for direct access to the database, making the server itself more secure, while it does not take very long to setup (although still longer than it takes to set up SQLite).

I also considered PostgreSQL, another fast and scalable RDBMS. However, I felt that it had poorer memory performance than MySQL, although this difference would largely be unnoticeable at the level of a small project such as mine. I decided to use MySQL as I had more experience with it.

# Hardware

Although I have chosen my programming language, web framework and RDBMS, the question of where and how to host my server remains.

I chose to host my server and database server on Microsoft Azure cloud, as it provides generous free monthly limits on many services for secondary school students, while the Azure cloud portal is also relatively simple to use. I decided on an Azure MySQL flexible server and to deploy my server as an Azure App Service, which allows you to host web applications without dealing with the infrastructure directly (both of which are forms of PaaS).