**CS673 Software Engineering** 

**Team 1 - Chit Chat**

**Software Design Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Masih Vahida | Team Leader |  | 09/07/2025 |
| Robin Roeoesli | Requirement Leader | *Robin Roeoesli* | 09/07/2025 |
| Jordyn Lipsey | Configuration Leader | *Jordyn Lipsey* | 09/07/2025 |
| Deasia Little | QA Leader |  | 09/07/2025 |
| Ardit Briskaj | Design and Implementation Leader | *Ardit Briskaj* | 09/07/2025 |
| All | Security Leader |  | 09/07/2025 |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

[Introduction](#_87t9hln2vjz0)

[Software Architecture](#_3ipvmjgn6clp)

[Class Diagram](#_ky60nv8suxxm)

[UI Design (if applicable)](#_7ucksmkf6rzx)

[Database Design (if applicable)](#_tcmuor4nl1kz)

[Security Design](#_x18fj36s1121)

[Business Logic and/or Key Algorithms](#_mtfbusfb0eq3)

[Design Patterns](#_9zvwkmc4luo5)

[Any Additional Topics you would like to include.](#_15tmymhipvdv)

[References](#_50ojo9i46ytq)

[Glossary](#_8n34lvocupub)

# Introduction

The purpose of this document is to describe the architecture and design of Chit Chat, a real-time messaging application that allows users to interact with a community or engage in a private conversation with an AI. This document serves as both a high-level overview of the system and a reference for the design of its architecture, database, business logic, and supporting components. It is intended to guide development, testing, and future enhancements of the Chit Chat application.

# Software Architecture

In this section, you will describe the decomposition of your software system, which includes each component (which may be in terms of package or folder) and the relationship between components. You shall have at least one diagram to show the whole architecture of . The interface of each component and dependency between components should also be described. If any framework is used, it shall be defined here too.

**Frontend Layer**

1. Message
2. Message container
3. Sidebar
4. Login
5. Message Input

**Backend Layer**

1. SocketIO Controller
2. DB Controller

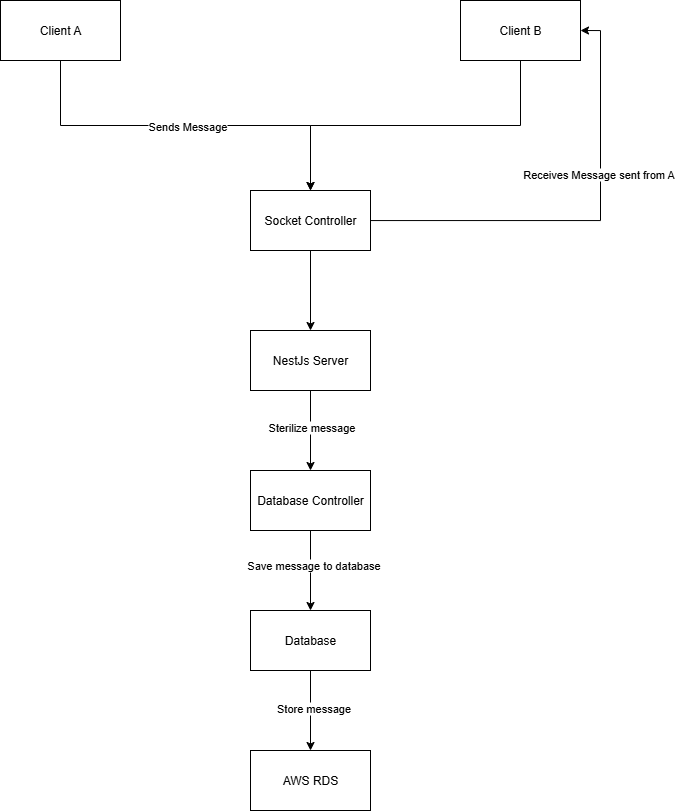
**Database Layer**

1. Accounts -username, password, full name.
2. Messages -> id, message
3. Reactions -> will store reactions by users on a specific message.

**SocketIO**

1. Socket On
2. Socket Emit

**PassportJs**

****

# Class Diagram

Message -> Text sent or received by user.

Message container -> Contains messages from user and other users

Sidebar -> Allows user to switch between community or ai chat

Login -> Allows user to log-in and retrieve their past chat history

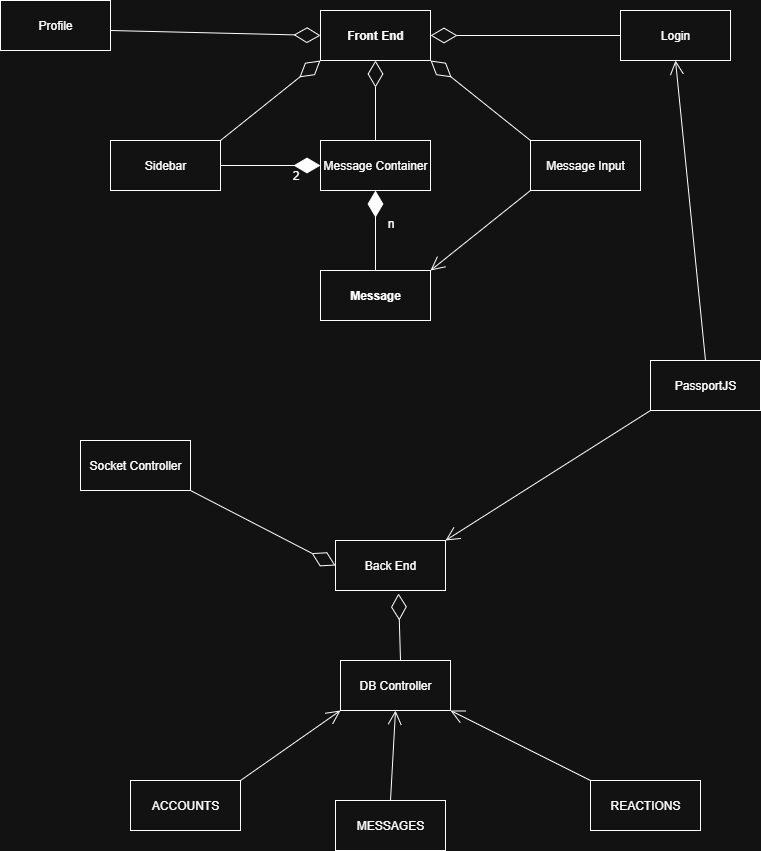
Profile -> Allows user to customize their profile and preferences

Message Input -> Allows user to type and review a message before submitting

SocketIO Controller -> Handles events between ui and server. Defines event listeners and event emitters and handles interaction appropriately on each event received.

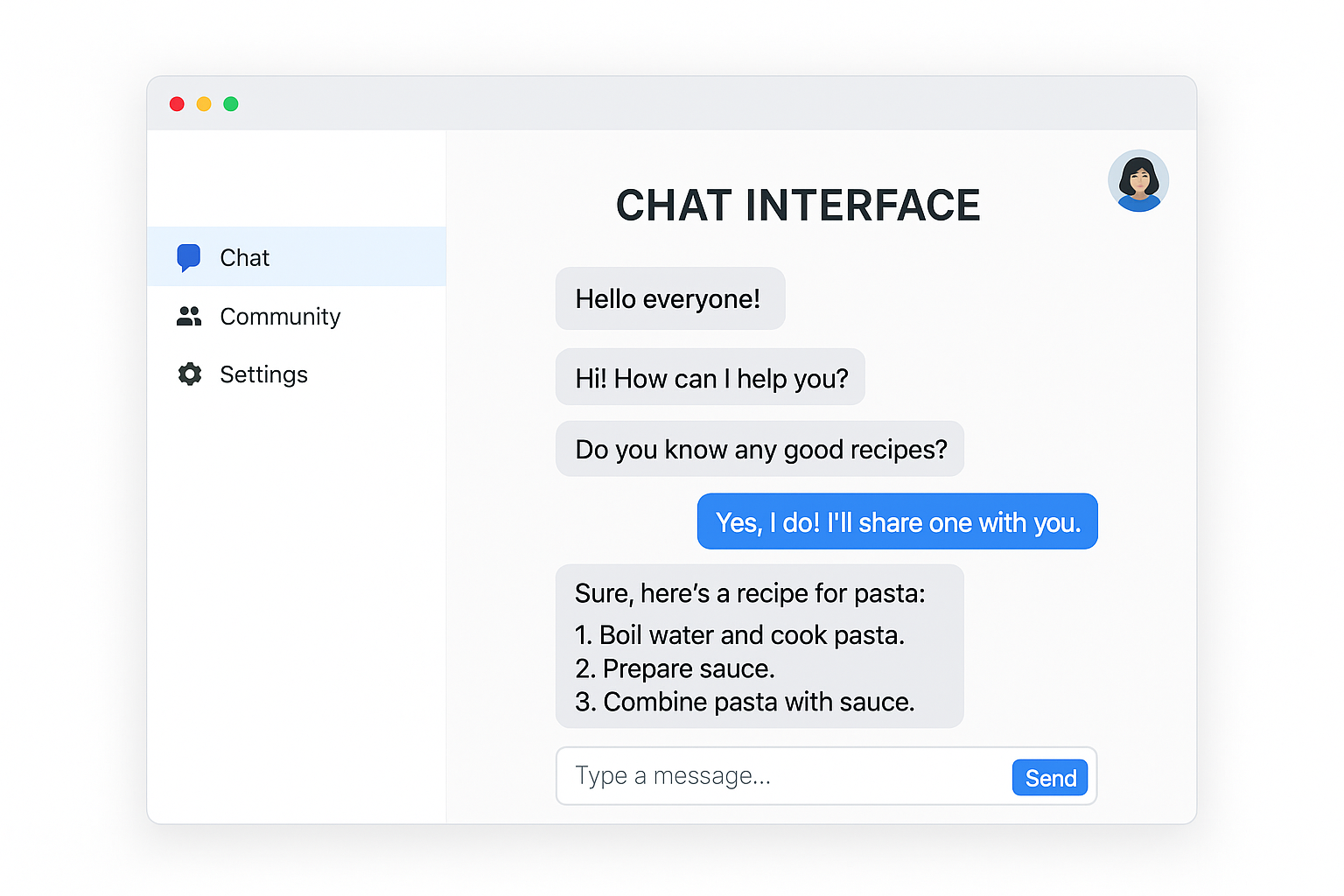
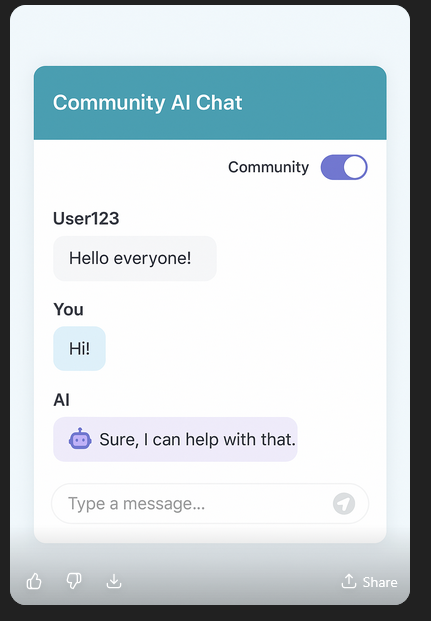
Database Controller -> Connects to application database and performs INSERT, UPDATE and DELETE operations.

PassportJs -> Handles the login authentication



# UI Design (if applicable)

We are following 2 UI designs as of now and make our own tweaks as we progress with our stories.

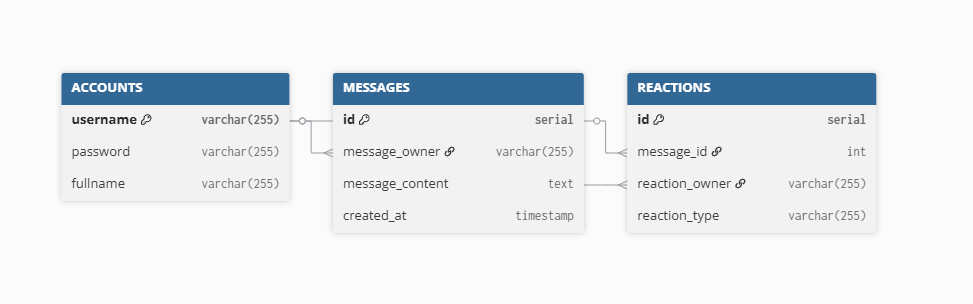


The main component of the application is the chat which as most chats contains all current and previous chat history. There are different bubble colors for the chats the user has sent and others have sent. In addition we have a side bar with currently 3 options. One is community which switches our main application to allow the users to chat with other users in real time.

The second option is the agent mode which allows the user to ask questions directly to an LLM.

The 3rd option is simply a settings item. There will be a profile on the top right corner with more options as we work through the application. There will also be support for message reactions.

# Database Design (if applicable)



For our Chit Chat application we have decided to go with PostgreSQL hosted in AWS. Our initial db design consists of the schema above which consists of 3 main tables. Accounts - stores user information such as username, password and full name.

Messages -> will store all messages sent by users.

Reactions -> will store reactions by users on a specific message.

Messages table has a foreign key constraint on accounts username on message\_owner.

Reactions table has a foreign key constraint on messages id on message\_id.

As the application grows we will introduce more tables, possibly some audit tables that keep track of LLM answers, as well as a way to store further information about the user like profile picture, settings etc.

# Security Design

Our UI will be using PassportJS to handle user authentication for our login system. If a user creates a new account we will store users data in a secure AWS hosted database. We will hash the user's password before we store it in our database. Returning users will verify users information against database credentials.

We will also sensitize all user’s messages before we store any of it in our database to prevent db injections. We will introduce a rate limit to control message frequency so it doesn’t crash our application but it still allows a flawless experience.

# Business Logic and/or Key Algorithms

Business Logic:

Community Chat: users send and receive messages from every other user within the application. Users can react to each others messages

AI Chat: User sends message directly to LLM and AI will process and respond privately to the user that messages the private chat AI mode.

Key Algorithms within the application are:

User Authentication using PassportJs and credential validation against database values.

Community Message Handling: This will handle messages sent and received separately between the type chat modes. Community messages will be kept and tracked separately from AI private messages.

# Design Patterns

Client-Server: Our application has a client which is the chat UI that serves as the interactive layer for our users, it also has a single server which hosts our [socket.io](http://socket.io) and serves as the layer which allows all of our users to communicate with each other and the interaction to our database.

Event Driven Architecture: The core system that allows real time interaction between our chat ui and our backend runs on [socket.io](http://socket.io). This publisher/consumer contract allows us to enable real time interaction between all of our users at the same time.

# Rest APIs

Chit Chat will not be using Rest APs instead it implements an Event Streamer design pattern using socket.io.

# Any Additional Topics you would like to include.

# AI usage Log

You are allowed and even encouraged to use AI tools to help you generate the project idea, plan it and build it, but you need to clearly describe 1) What tools were used? 2) for what specific tasks and 3) Is it helpful? 4) how did you evaluate or modify AI-generated content? Additionally, you should submit the exported AI chat history as an appendix or share that with the instructor and facilitators.

| Tools | Who | Tasks | helpful | Evaluation/modification | links |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# 

# References

[dbdiagram.io](http://dbdiagram.io) -> Used for db schema design

[drawio.com](http://drawio.com) -> Used for class diagram

# Glossary