Design Document

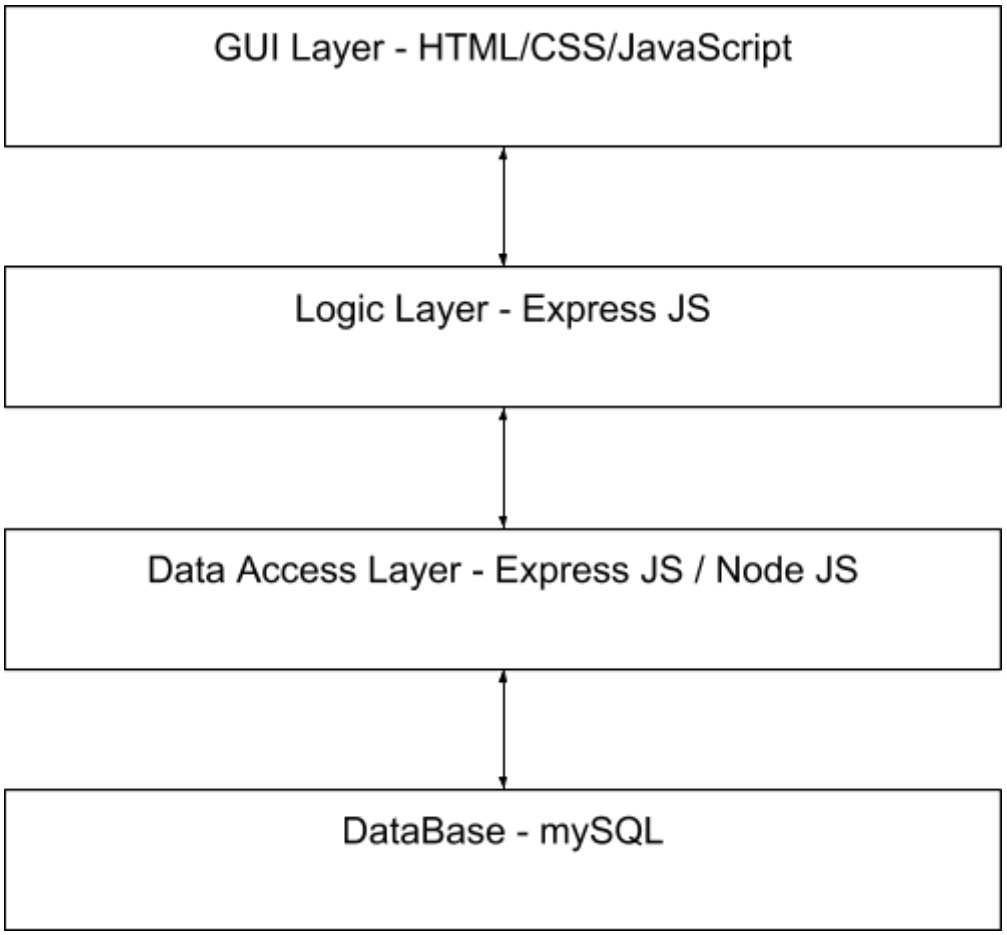
CSCE 361 - Spring 2018

1. **Introduction**

   The purpose of this design is to demonstrate the high level architecture and entity relations for the teaCHat application management system. This document shall include architecture and entity relation diagrams showing the relationship between different parts of the system, as well as the relationship between database tables. Readers of this document are system architects and software engineers who will be implementing and maintaining the described system.

2. **Architecture**

2.1 Introduction



The layered model will make up the architecture for the design of this system. At the lowest level of this layered model is the MySQL database, that will be used to store basic information about users, their passwords, and potentially the chat rooms associated with each user group. The data access layer will be able to read and write to the database, however, it will only be sparingly used as message sending and receiving will be handled by the real time javascript framework, Sockets IO. The data access layers primary job will be to store account information as it pertains to rooms joined, user type, and credentials. The logic layer will be purely written in JavaScript to handle the necessary functions on the data, forming the third lay.

2.2 Modules

2.2.1 Database layer

The database layer will store persistent data from the system, and define its relations. The system will be using a MySQL database, that will allow us to have simple yet easily accessible relations to other data records.

2.2.2 Data Access Layer

This module is responsible for selecting and saving to the database layer. All access to the database will go through Node that will communicate directly with the database layer. This layer is responsible for converting database records into JavaScript objects that can be manipulated by the Logic and GUI layers. It's primary function will be to query the database and translate the information into JavaScript objects. It will also be responsible for communicating changes to the JavaScript objects in any other layers to the database.

2.2.3 Logic Layer

In order to keep the layer hierarchy, this layer is supposed to communicate only data access layer and GUI layer if there is a request from it. As our goal is to show the questions from students in the chat room, we only retrieve the questions for respected course and class ime. Since the students identities are anonymous, the other students in the chat room don't see the other students identity but the professor. When the professor clicked on the close the chat room button, there will be a message appeared that asks whether they want to save the chat room, this allows the professor to export the chat to reference further after class. This is also the layer where we will implement our spam classifier for the second iteration to filter out messages with inappropriate or irrelevant content.

2.2.4 GUI layer

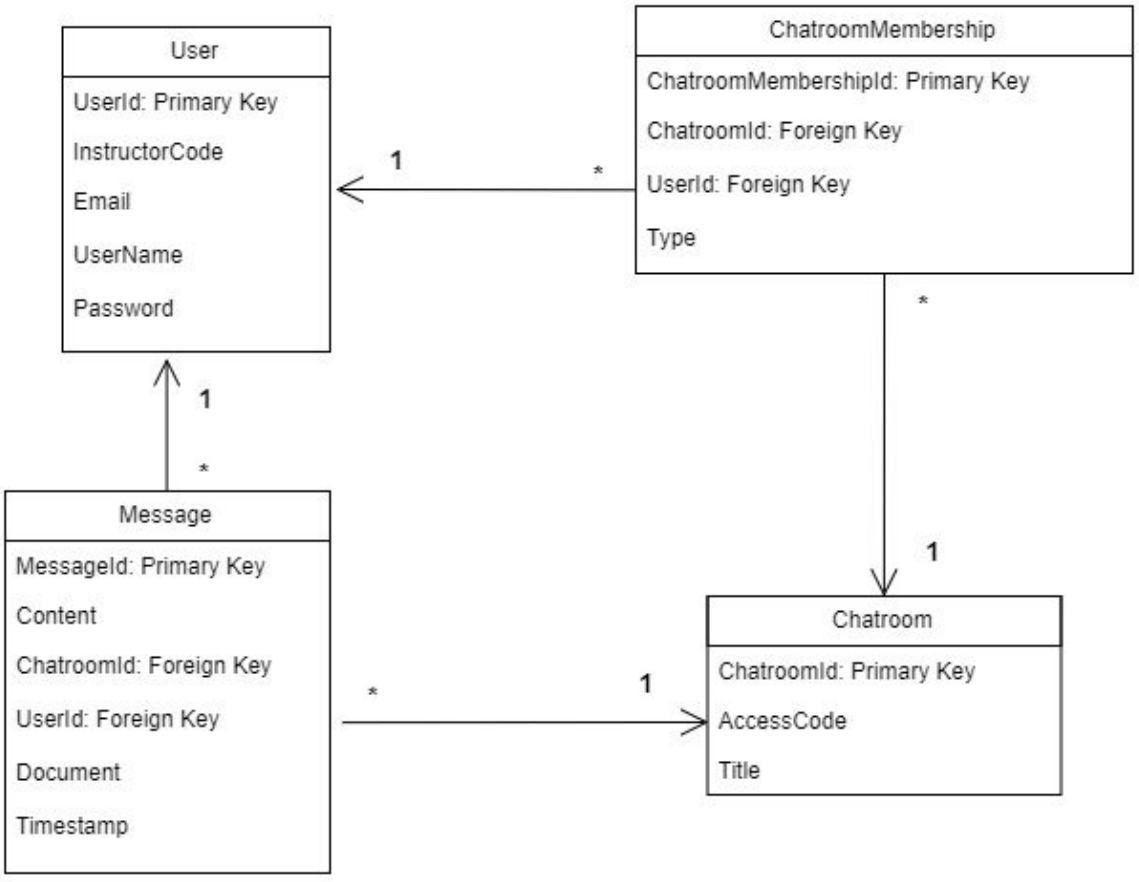
This layer will be responsible with communicating directly with the logic layer to make a responsive interface is possible. The GUI will be updated every time the logic layer receives new information from the data access layer, i.e., the flow of chat messages. If a new question is asked, there will be a notification message to the professor's window. Then professors can answer the questions directly in class or respond in chatroom. It is in this layer that both students and teachers will be able to login, create and select chat rooms, and interact in chat with the rest of the class. Grid layouts in the HTML/CSS will be used to design the GUI part of the software so that mobile devices are also supported.

3. Class Diagrams

3.1 Data Table Classes

The system will be using a MySQL database to hold all the persistent data necessary. This includes information on users, chat rooms, messages and the relationships between them. The figure below shows each table in the database with its respective columns as well as the relationships between the different tables

3.1.1 Schema



3.1.2 Schema Information

The data in the database shall be organized in the following tables and columns.

User: This table holds the data for a single user of the system including information such as their username and password, email, and instructor code (if the user is an instructor).

ChatroomMembership: This table is the junction table for the User and Chatroom tables to accurately represent the many to many relationship between the user and the chatroom. Each record will have a user identifier, the type of the user (student or teacher), and a chatroom identifier.

Chatroom: The table will represent a chatroom for a given class. Each chatroom will have a name, a unique access code, and a title.

Message: This table holds the messages sent by users to the chatroom. It will contain the actual text written by users, user identifier (to identify the sender), chatroom identifier (to identify the chatroom the message belongs to, document link (to keep track of user-uploaded attachments), and a timestamp of when the message was sent.

3.2 Class Information

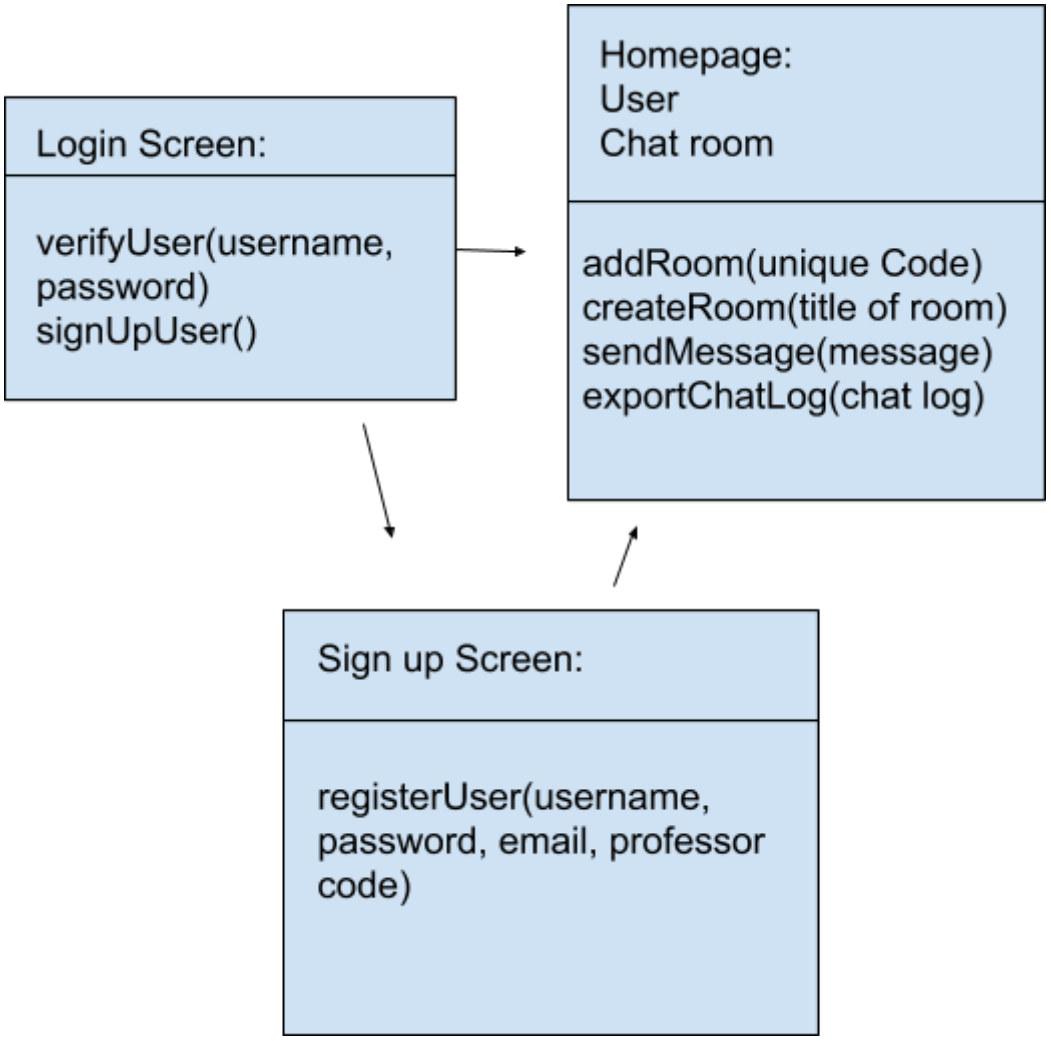
Classes will be implemented using Javascript by modelling the database schema in section 3.1.2 except the Chatroom Membership table which does not need to be an object as it only serves as a junction table to simulate the many to many relationship of Users and Chat rooms. For instance, a user class will be implemented with variables for UserName, Email, Password and InstructorCode. The logic layer will take these classes and pass the information up to the GUI layer where it is displayed to the user.

3.3 GUI Layer

The top layer of this system contains three pages : a login screen, a sign up page, and the homepage. The login screen of the application will be the landing page once the users enter the application. This page is where the users will enter their login credentials, specifically their username and password. The user's input will be verified by cross-referencing the input with the database information. If invalid, an error will be outputted to the user. If the information is valid, the user will be redirected to the homepage, and depending on the type of user, the system will load the appropriate existing chat rooms and functionalities. If the user does not have an account, there will be functionality that will allow the user to be redirected to the sign up page.

The sign up page will ask for three inputs from the user including email, username, and password. It will ask for the type of user which is either a professor or a student. Once the user inputs this information, there will be a sign up button that will create the account with the appropriate information. The user will then be brought to the homepage.

The homepage will display the chat rooms the user has access to. If the user is an instructor, he/she will have access to a 'create room' button to instantiate a new chat room. This will pop up a modal that will prompt the instructor to enter relevant information to create a chat room for their course. This information includes the chat room name and a chat room code to allow students to join. Both the professor and the student user will be given the functionality on the homepage to join a room through the "Add Room" button. This will load up an 'Add Room' modal that will ask for the input of the chat room code. When a student user enters a chat, their name will appear as an anonymous pseudonym to the other student users. The instructor user will be able to see the student user's name and all users can see the instructor user's name in the chat room.
Existing chat rooms can be found on the side of the screen, and once clicked, will open up that chat room. A text input box will appear towards the bottom of the screen for users to input their messages and send it to other users in the room to see on the screen. There will also be an export chat log button next to the send button of the text input box. The instructor user will have the ability to export chat logs or section of chat logs to their desktop using this button.


Homepage:
User
Chat room
addRoom(unique Code)
createRoom(title of room)
sendMessage(message)
exportChatLog(chat log)
Login Screen:
verifyUser(username, password)
signUpUser()
Sign up Screen:
registerUser(username, password, email, professor code)