teaChat

Group #6

Vinay Singh, Min Jung Kim, Luis Ramirez, On Saint Tiong, Tuyet Tran

Software Requirements Specification

Document

Version: (2.0) **Date: (02/09/2018)**

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to give a detailed description of the requirements for the teaCHat application. It will explain the purpose and the features of the application. This document is intended for both the stakeholders and the potential developers of the system.

1.2 Scope

This system, teaCHat, will be a web-based chat room application used for student and instructor interactions to solve two major problems: students during classroom lectures that may not feel comfortable asking questions out loud and students who are in the classroom that may not hear the other students' questions clearly.

The application will serve as a student platform to communicate their questions and/or concerns and give instructors a way to analyze their students' feedback. The goal of this application is to enhance student interaction with the classroom lectures and to collect data about classroom interactions that could be used for potential research studies.

1.3 Definitions, Acronyms, and Abbreviations.

Chat Room: A service that allows multiple users to simultaneously send and view messages.
Socket.IO: A JavaScript library for realtime web applications.
MD5: The MD5 algorithm is a widely used hash function producing a 128-bit hash value.
iClicker**:** device by which most students at UNL answer in class questions, and is registrable through Blackboard
Spam: irrelevant or inappropriate messages sent on the Internet to a large number of recipients.
Express JS: Web and server application framework for Node Js.
Blackboard: Legacy student portal that stores relevant information for courses that students are enrolled in. Supports clicker registrations and responses.
Canvas: New student portal that stores relevant information for courses that students are enrolled in. Supports clicker registrations and responses, forum posting functionality, and messaging other members of the class.

SQL: Structured Query Language is used to communicate with databases. According to the American National Standards Institute (ANSI), it is the standard for relational database management.
Node JS: Backend technology that allows for server-side scripting in JavaScript. Runs script server-side to prepare content for users before sending the response.
Amazon Web Services: A safe and secure cloud services platform that host a variety of cloud operations such as server side hosting, data warehousing, and cloud computing.

1.4 References

[1] IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications”, October 20, 1998.

1.5 Overview

The remainder of this document includes two chapters and appendix. The second chapter provides an overview of the system requirement and description of the different system interfaces. The third chapter provides the requirements specification in detailed terms and overview of the system functionality. The Appendixes in the end of the document includes mockups of the entire system to better understand the functionality of the system.

2. The Overall Description

2.1 Product Perspective

2.1.1 Software Scope and Perspective

2.1.1.1 The product is a software that allows students to create online chat rooms for their concurrent classes. It is here where the instructor of a class can see the questions and concerns of students update in real time.

2.1.1.2 This product as it exists currently is the first of its class. We believe that this product is a strong contender to replace services such as Blackboard, as well as Canvas, that share similar, but weaker functionality.

2.1.1.2.1 The software, unlike its competitor Canvas, updates in real time as questions and conversation occurs in the chat. Canvas has a forum function that allows users to post to, but is often slow, and unresponsive.

2.1.1.2.2 The software, unlike Blackboard, provides a more intuitive way of gauging audience understanding of a certain topic. Blackboard’s clicker questions can only have at most 5 responses. A chat based questioning system such as the product allows students to ask specific questions over the material being covered.

2.1.1.3 The product will be self contained, as the system itself only needs to communicate to itself.

2.1.2 System Interfaces

2.1.2.1 The system will not need any exterior systems to operate.

2.1.3 User Interfaces

2.1.3.1 The system will be written for the web, and so users will interact with a web browser to chat with others and use the application.

2.1.3.2 Sign Up Interface

2.1.3.2.1 The software will have an administrative interface to handle student and professor creation of accounts.

2.1.3.2.2 If the user is new to the application, the system shall allow the user to click the Sign Up link seen in appendix 4.2 . The system shall prompt the user to input information for registration credentials stated in section 3.2.1.1.2 .

2.1.3.2.3 If the user is new to the system and is a professor, they will be able to specify that in the user creation interface. Students will also be able to specify that they are creating student accounts

2.1.3.3 Login Interface

2.1.3.3.1 If a user has already created an account, they will be able to navigate to the login.

2.1.3.4 Chat Room Creation

2.1.3.4.1 Once signed in, a user will be able to create a chat room.

2.1.3.4.2 Chat room creation only needs the chat room name. This will create a unique room, with a unique ID. This ID can be distributed, and used to join an existing room.

2.1.3.5 Chat Room Joining

2.1.3.5.1 Chat rooms will generate unique IDs that may be distributed for students to join an existing chat room

2.1.3.6 Student Chat Room Selection

2.1.3.6.1 Students can see the chat rooms that they have previously joined in the sidebar. To join one, the user simply has to select the room that they desire to join. No join code is necessary here as they have already been added to the room previously.

2.1.4 Hardware Interfaces

2.1.4.1 All devices that have an internet connection and a web browser will be able to use the system.

2.1.5 Software Interfaces.

2.1.5.4 Software Interfaces

2.1.5.4.1 Since the application will be openly hosted on the web, the system will be able to be accessed on the most updated web browsers.

2.1.6 Communications Interfaces

2.1.6.5 Communications Interfaces

2.1.6.5.1 This web application main function is to provide better communication between professors and students. The system will operate by sending messages between Socket.io and the individual users chat rooms

2.1.7 Memory Constraints

2.1.7.1 This is a very simple chat application, that will not need very much memory. The memory usage will depend on the browser the chat is being run in, max of 2MB.

2.1.8 Operations

2.1.8.1 There is only one mode of operations for this application, which is running the chat in a web browser and joining a chat room in the chat room interface.

2.1.9 Site Adaptation Requirements

2.1.9.1 There are no site adaptation requirements.

2.2 Product Functions

This application only function is to provide an interactive environment for students to participate in peer to peer discussion via chat room. The professor will moderate the room and answer questions that students will pose in the room.

2.3 User Characteristics

This application aims to accommodate two types of users: students and lecturers. Student users are assumed to be those pursuing or have a post-secondary education, thus should have at least a basic understanding a chatroom. Student users can also be seen as observers of the lecture/classroom and are assumed to be the audience that is interacting with the lectures. Lecturers or professors are the second type of users. They are the ones that present to the student users, thus will be using the application to interact with their audience. These users are assumed to have higher education or at least advanced knowledge in their lectures to strengthen their interactions with student users.

The UI design plans to only include simple functionalities for both users. Although these users are assumed to be applying this system in an environment of higher learning, the design aims to be able basic enough for those with little to no experience in web technology to use it.

2.4 Constraints

The system will be designed for laptops of various resolutions, as CSS is vastly scalable and will be able to detect differences in users screen size. There are few resources required by the system, but since web applications are at mercy of the users CPU, performance may vary user to user.

2.5 Assumptions and Dependencies

The application relies on Express, Node, and Socket.io dependencies. As with most web softwares, it takes many months, or potentially years, before large updates, improvements, or changes are made.

2.6 Apportioning of Requirements.

The basic functions like login, register as different users, and/or creating chat rooms should be implemented first in order to construct the general structure of the application. Future versions of the system will include machine learning filtering, to prevent inappropriate messages from being displayed to the users in the chat room. Along with this functionality the professor in future versions will be able to save the chat room, and address students individually to further assist them answering more specific questions that they might have missed during lecture. The user finds value in this as it helps them feel like the professor actually cares about the students concerns and commentary in class.

3. Specific Requirements

3.1 External Interfaces

The teaCHat web application system's purpose is to be a communication platform used by multiple users to do instant messaging. The system will run on any type of internet browser. The inputs are made by the users through button clicking and/or messaging in the text box. The system will take the input and depending on its type, will either immediately output it to other users or store it in the database.

3.2 Functions

3.2.1 First Release

3.2.1.1 User Authentication

3.2.1.1.1 Upon loading the application, the system shall load a user login/registration page. Mockups of this functionality are shown in the appendix sections 4.1 and 4.2.

3.2.1.1.2 The system shall prompt the user for their login credentials, or will allow them to create a new account. Specifically, the system shall prompt new users for their email, username and password, whereas existing users will be prompted for their usernames and passwords. Upon completion, the system shall redirect to the user home page (appendix sections 4.3 and 4.4).

3.2.1.1.3 For instructor roles, the system shall only allow users with unique identifiers, which are assigned to them on another website (company website that is yet to be developed), to sign up.

3.2.1.2 Chat Room Creation

3.2.1.2.1 The system shall allow instructor type users to create chat rooms for other users to join and interact in. A mockups of this functionality is shown in the appendix section 4.5 .

3.2.1.2.2 The system shall prompt the instructor to insert necessary information (Course Number, Student Access Code) for the chat room creation, at the click of a ‘New Chat room’ button.

3.2.1.3 Join (Add) Chat Room

3.2.1.2.1 The system shall allow users of any type to join chat rooms created by the instructor. A mockup of this page is shown in appendix 4.6.

3.2.1.2.2 Upon clicking the ‘Add room’ button, users will be prompted for a code that should be provided by the instructor.

3.2.1.4 Instant Messaging

3.2.1.3.1 The system shall provide an interface for a real-time multi-user messaging platform. (appendix sections 4.7 and 4.8).

3.2.1.3.2 The system shall receive a message from the user as input (attachments are allowed) and that message will be uploaded to the chat room for others to view.

3.2.1.3.3 The system shall anonymize the message senders’ identities if they are students. Instructor identities will be visible.

3.2.2 Subsequent Releases

3.2.2.1 Spam Filter

3.2.2.1.1 The system shall filter student messages before they appear on the chat room to remove inappropriate content.

3.2.2.1.2 The system shall handle this capability internally and hence, there are no inputs and outputs to implement this function.

3.2.2.2 Chat Room Transcript Storage

3.2.2.2.1 The system shall allow instructors to save entire chat room transcripts for future reference.

3.2.2.2.2 Within the instructor's view of the chat room of interest, the system shall provide a button for the instructor to export the transcript to an appropriate format (pdf, word, or csv document).

3.2.2.3 Student Response Questions

3.2.2.3.1 The system shall allow instructors to launch question polls on the chat room for students to respond to.

3.2.2.3.2 Within the instructor's view of the chat room, the system shall provide a button for the instructor to post a question on the chat room (open-ended or multiple choice) which will then be posted for students to click to respond to. Students may respond by clicking an answer option (multiple choice) or by entering their own text (open-ended), depending on the question type.

3.2.2.3.3 The system shall display a summary of student responses as a statistical chart once the instructor closes the question poll if a question is multiple choice.

3.3 Performance Requirements

The system should be able to run instantly without noticeable lag in receiving and sending messages. The system should take a maximum time of 3 seconds for a message to be received and displayed by the chatroom (this time is calculated as soon as the sender clicks 'send').

3.4 Logical Database Requirements

3.4.1 The system shall utilize a relational database to store the following relevant information to the application.

3.4.2 The database shall include a table to store user information (login credentials, and other relevant information) called the ‘Users’ table. Passwords will be stored in an MD5 encrypted format to protect integrity. The primary key of this table (User Id) will be used as a foreign key on the ‘Chatrooms’ table.

3.4.3 Each chat room shall be stored as its own entry on a database table called ‘Chatrooms’. It will be connected to appropriate users by the ‘User Id’. Its own primary key (Chatroom Id) will be used as a foreign key on the ‘Messages’ table.

3.4.4 Every message sent on the platform will be stored on a table called ‘Messages’. It will be connected to a chatroom by the ‘Chatroom Id’.

3.5 Design Constraints

The only constraint that this system will have is that it may only run on web browsers. Mobile phones would need to access the service through any mobile web browser as we are currently not developing this service to be a mobile application.

3.6 Software System Attributes

3.6.1 Reliability

This system should keep the database information consistent. This part of the system should not fail for 99% of production time. In the case of potential database failure, there should be a system to backup our data tables in a third party cloud service, such as Amazon Web Services. Our response time should be immediate, as we aim to keep mean time between failures (MTBF) as low as possible. In the user side, the system should be error free and display alerts if the system does not work properly. On the database side, if errors occur, it can be closed and opened again and should recover any information for the user and the rest of the system.

3.6.2 Availability

Our goal for the system availability is at least 98% uptime. Scheduled downtime will occur during times of little to no user traffic. In this case, it would mostly occur during evenings and/or weekends when lectures are not occuring. There is no reason to restart the system because it’s a real time chat service. Maintaining the hardware interface will not interfere with the availability since most of the hardware will be the database which would only need downtime if at risk of compromising data. Software failures may occur if bugs occur in the system, thus would be treated as discussed in the reliability section. If the system were to fail and be recovered, logs of chats should also be recovered from the last checkpoint of login which should be from the start of the creation of the chatroom.

3.6.3 Security

The user information such as passwords with the user IDs will be stored in an encrypted format (MD5) to protect its integrity. The system must not request unnecessary permissions from the user in order to prevent unwanted attacks. Finally, every user-prompted information which includes usernames, passwords, and email addresses shall be sanitized to prevent SQL-injection.

3.6.4 Maintainability

The system design elements should be documented well and all parts of the code should be easy to read. Also, the application should be easy to extend in order for future functions to be added easily to the application without changing any of the existing features.

3.6.5 Portability

Since the program is web-based application, it has to be easily portable in the any software development process. To improve portability, the software should run on a variety of platforms such as, Linux, Microsoft Windows and mac OS and any type of internet browser and variety of connection speeds. The software should be lightweight so that it can run on a machine with slow internet connection. Also, to run on most number of different platforms without any additional effort, it should be developed by using the common technologies and tools which are provided by all common web browsers and operating systems such as HTML.

4. Change Management Process

When a request for a requirement change comes in, the change management process has to be executed. The customer and the development team will have a meeting either through phone or in person to make sure everyone is on the same page in this process.

The steps of this process are:

Log changes
Estimate time needed for the change request
Obtain customer sign off
If needed to shutdown system for change, schedule a downtime
Make changes to the product as specified

A change request log is maintained to keep track of the change requests which will be incorporated in the appendix section. Each entry in the log contains a change request number, a brief description of the change and the status of the change request and important dates.

5. Document Approvals

Dr. Hamid Bagheri

Signature: ____________________ Date:________________________

M. Parvez Rashid

Signature: ____________________ Date:________________________

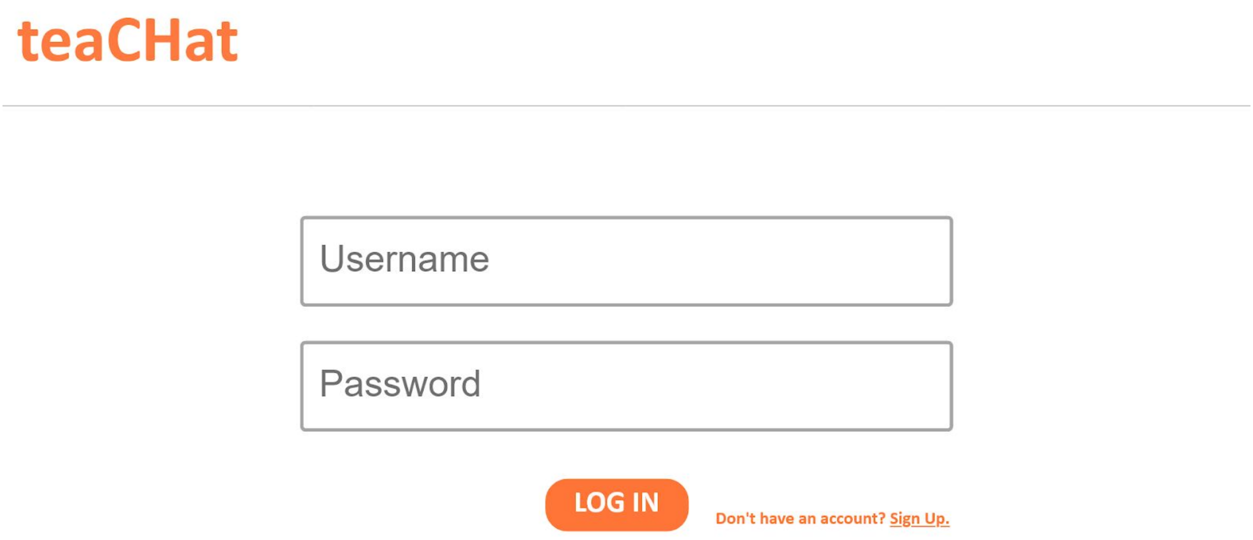
Qi Xia

Signature: ____________________ Date:________________________

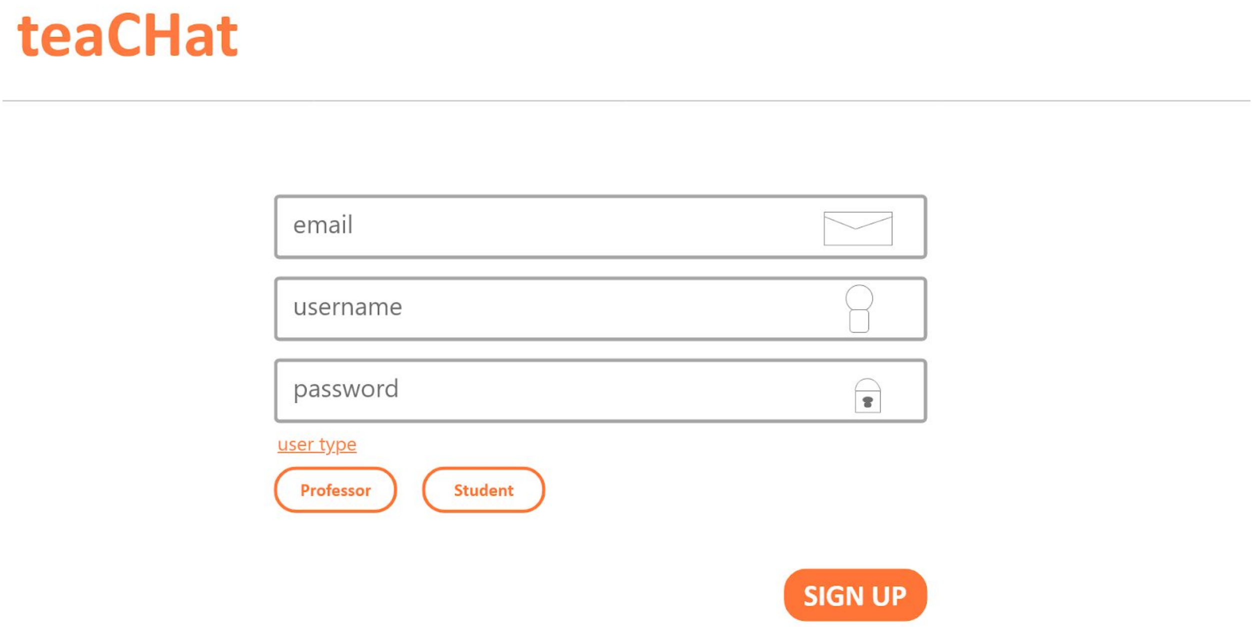
6. Appendix

Mockups below are the first prototypes created. It is expected these prototypes will change as the project goes on.

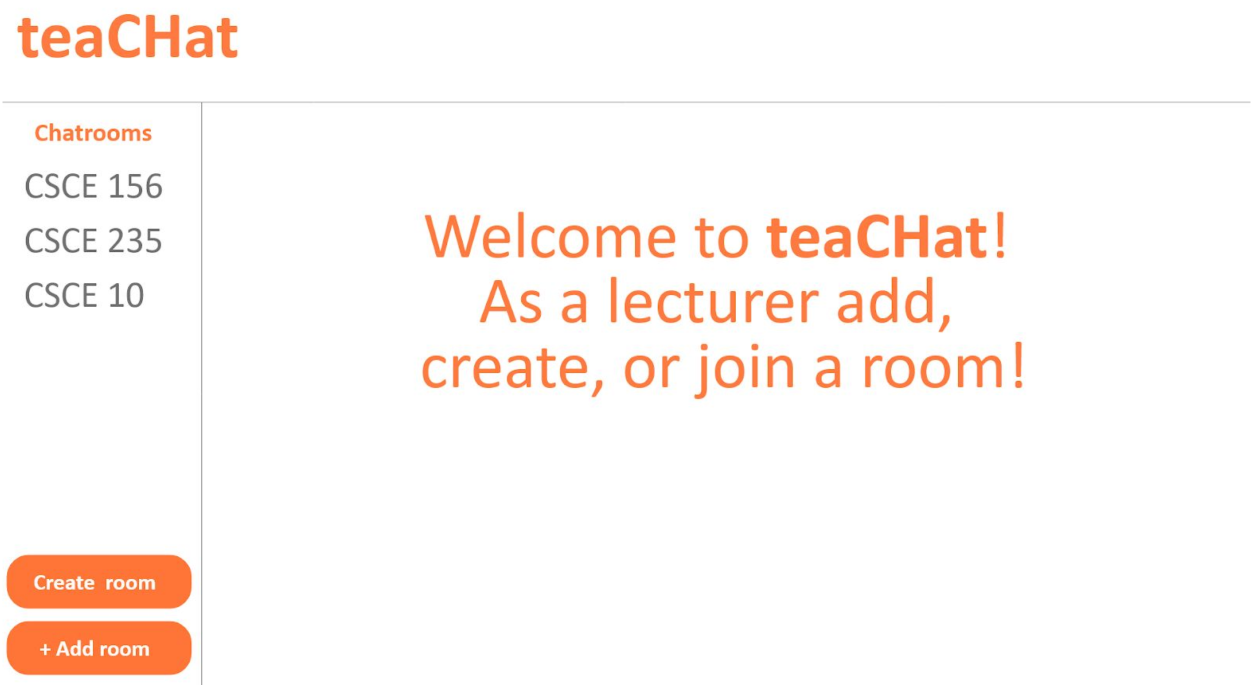
6.1 Login Interface



6.2 Registration Interface

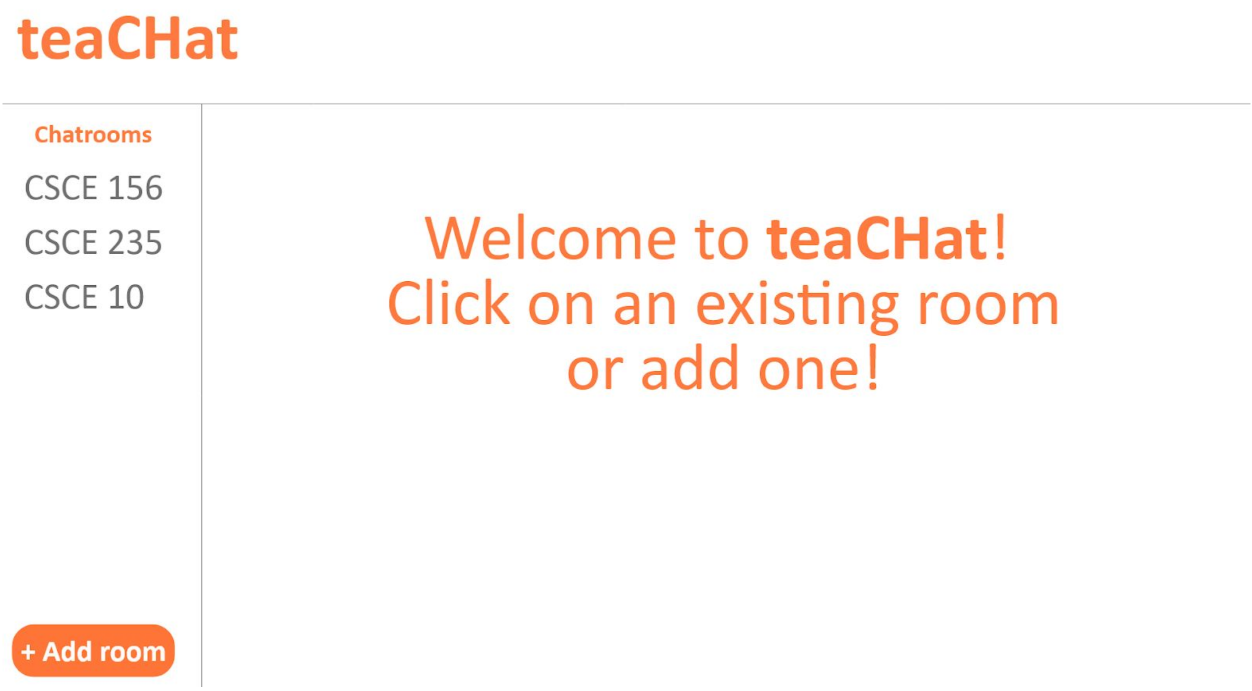


6.3 Instructor Home



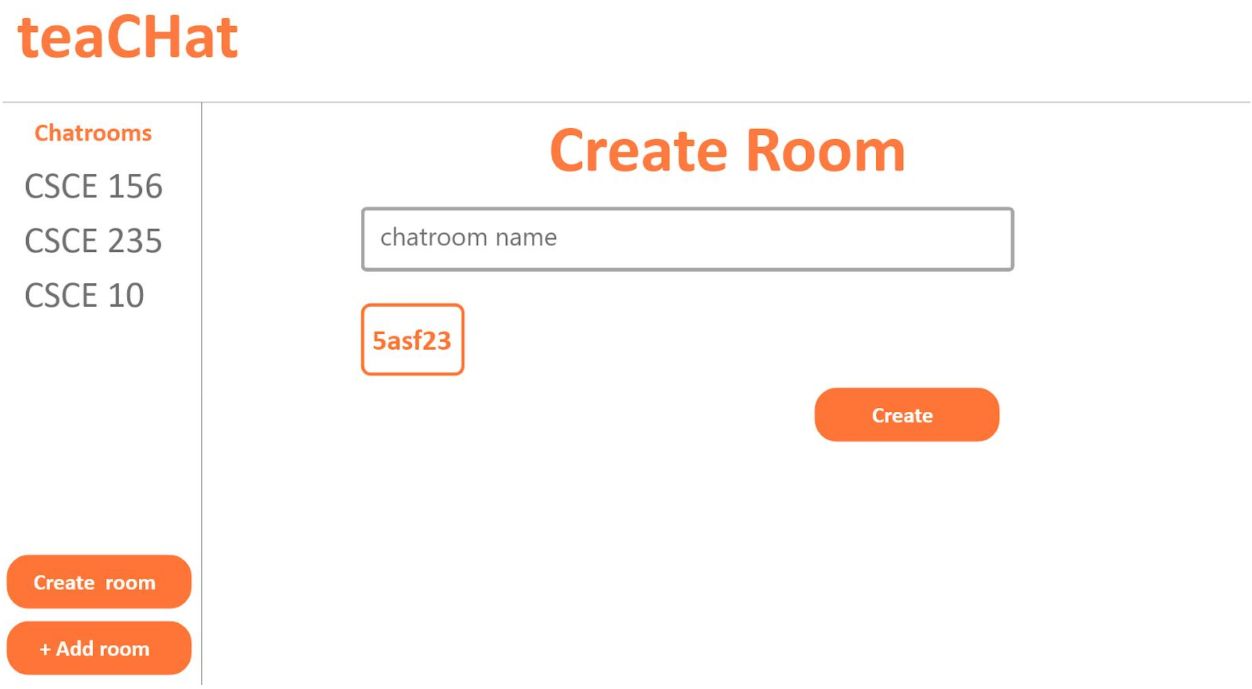
Welcome to **teaCHat**!
As a lecturer add,
create, or join a room!

6.4 Student Home

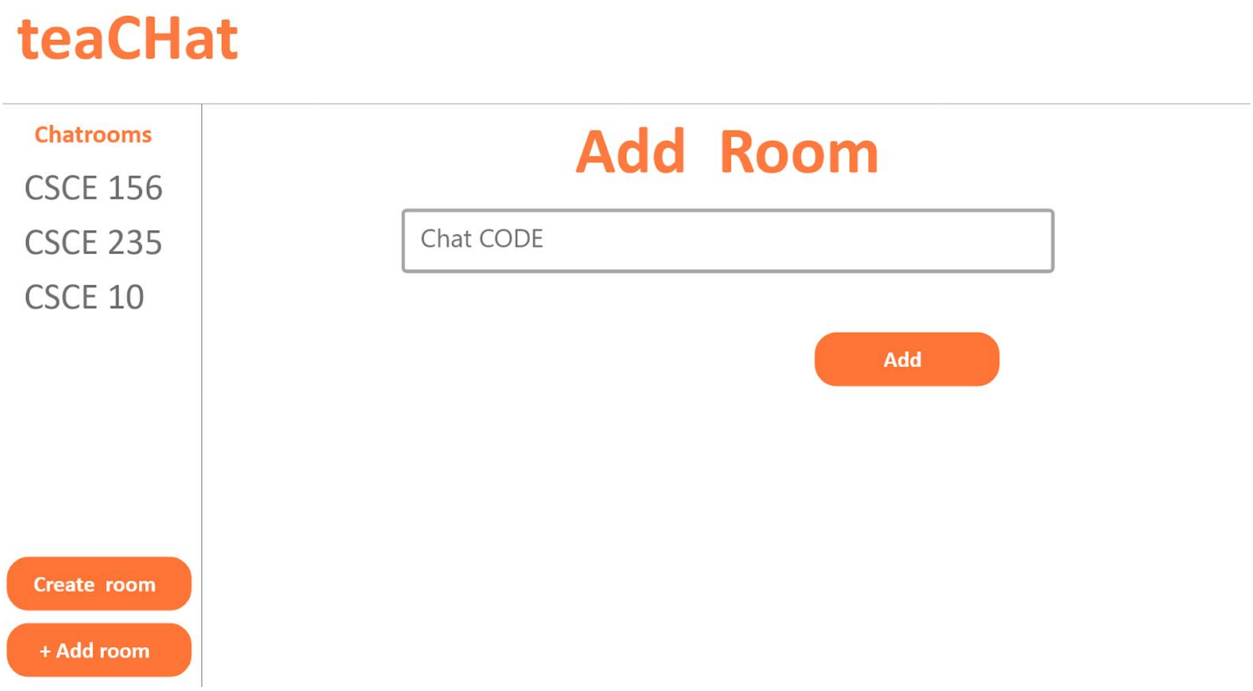


Welcome to **teaCHat**!
Click on an existing room
or add one!

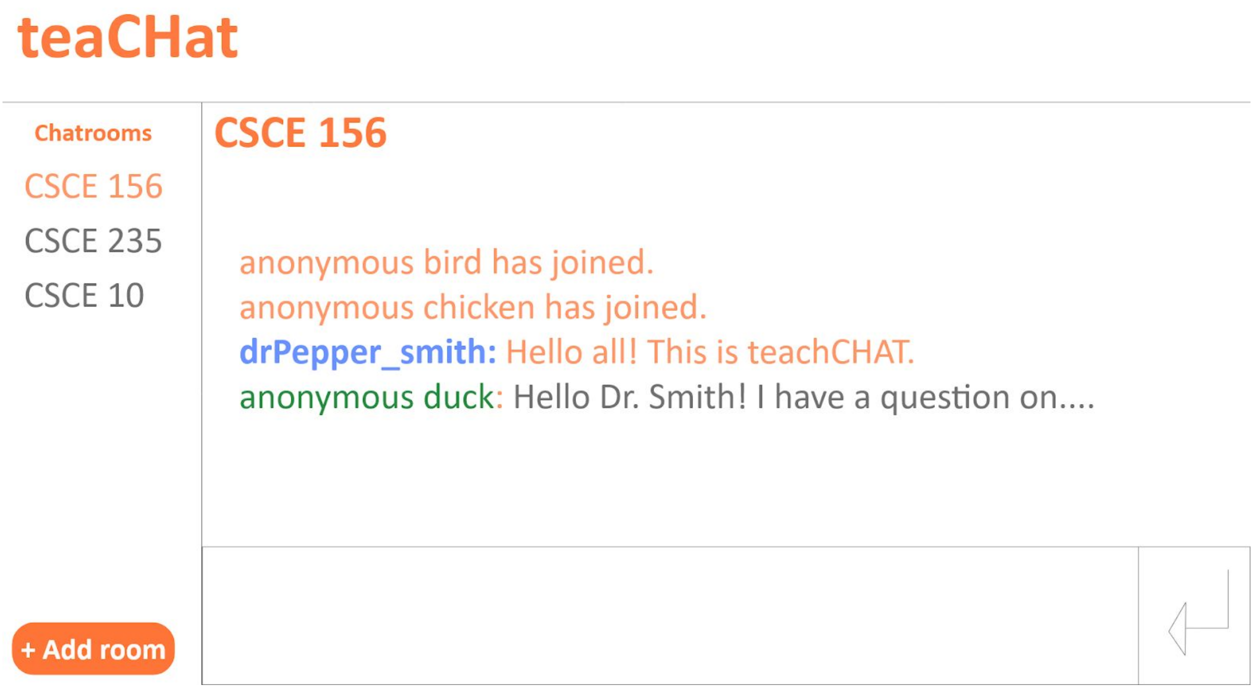
6.5 Create Room



6.6 Add Room

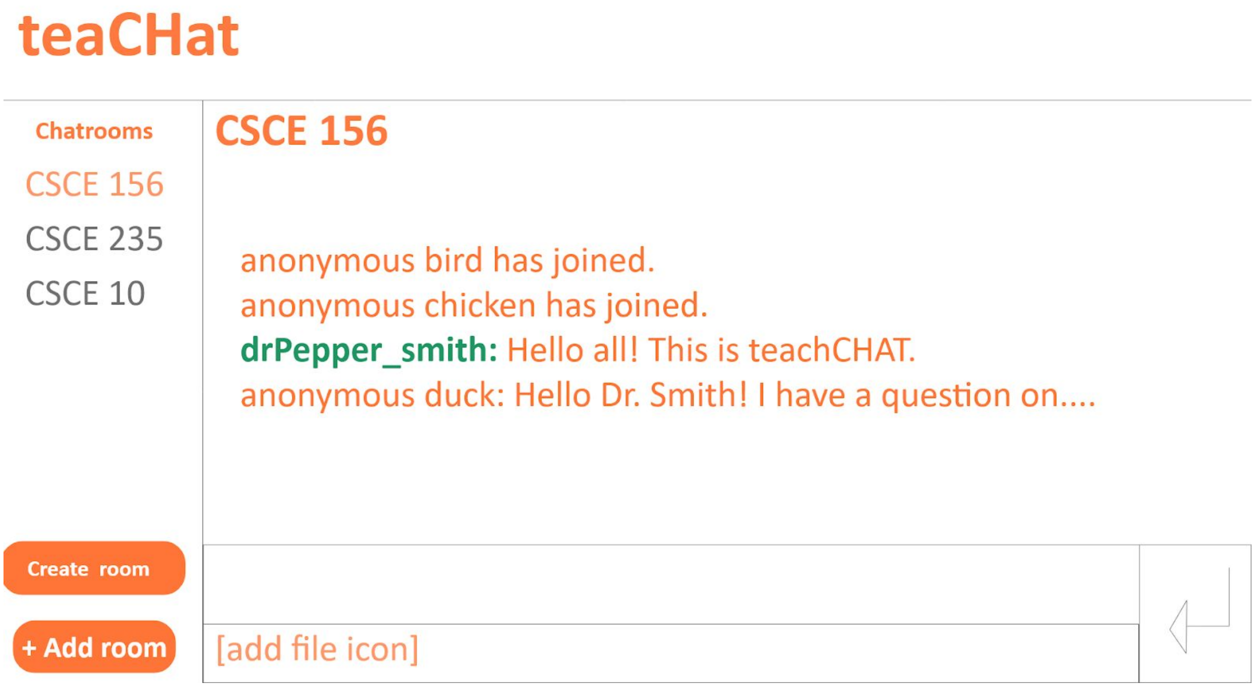


6.7 Student Chat Room



anonymous bird has joined.
anonymous chicken has joined.
drPepper_smith: Hello all! This is teachCHAT.
anonymous duck: Hello Dr. Smith! I have a question on....

6.8 Instructor Chat Room



anonymous bird has joined.
anonymous chicken has joined.
drPepper_smith: Hello all! This is teachCHAT.
anonymous duck: Hello Dr. Smith! I have a question on....