Office Hour Queue
Team 8

Team Members

Mengfei Zhang, mengfeiz@usc.edu
Chengxi Xu, chengxix@usc.edu
Jiaxi Yang, jiaxiyan@usc.edu
Enyan Xia, enyanxia@usc.edu
Zhewen Yu, zhewenyu@usc.edu
Xiang Li, limike@usc.edu

Table of contents

Project Proposal

- Weekly Meeting Time
    - Thursday 6:00PM PDT

- Description:
  We would like to create an office hour and lab queue web application, so students in courses such as CS201L can know how many people are in the queue and the approximate time when they are going to be checked off. This app will help students manage their time more effectively. It would be desirable if this app is going to be a web app. All users can see how many students are in the queue and the information about the TAs. Registered users can add themselves to the queue. Verified admins(TAs) can add/remove office hours and check students off.

High-level Requirements

Objectives

Create a website

- To allow students to visualize their position in the office hour queue and know the estimated time they have to wait.
- To allow faculty to make modify the queue status after students get checked off

User Access

- Users will be split into 3 categories: Student, Faculty(Admin), Guest.
- Students can request to enter the queue
- Students queueing have the option to leave the queue
- Faculty(Admins) can add/remove office hours and check students off
- Guest accounts (aka public accounts) can only view queue status

Functionality

- The web application will support login functionality to distinguish between students, faculty, and public
- On the student page, the application will
  - Have a button for students to add themselves to the queue
  - Have a button for students to leave the queue for office hours
  - Display the total number of students waiting
  - Display students' current status in the queue(Waiting, Being Answered, Completed)
- On the faculty page, the application will
  - display the current students in office hours as well as the list of students currently in the queue.
  - Have a button that checks one student off and changes that student status.
  - Have a button to start queue and end queue
- The application will allow unregistered users to view how many students are in the queue and information about the TAs.

Security

- All users must login or sign up to the system to have full functionality
- User data will need to be secure during login.

Technical Specifications

Web interface (40 hours) -Chengxi, Jiaxi

› The web interface needs to have a login page with a username field, a password field, and a login button

› Before verification, the guest page should show the queue name and the number of students waiting.

› Upon verification, for students, another page should be displayed with the student's information, the information of the faculty currently available, the current status of the queue, number of people in front of him or her in the queue, and the estimated time to wait before getting checked off. There should be a form for the student to choose the type of help he or she request(ie. office hour, checkoff, or question). There should also be a button for the student to leave the queue.

› The faculty page should show the faculty the current students in office hours as well as the list of names of students currently in the queue. There should be a button for the faculty to change the status of the queue (running or paused). There should be a button that checks one student off and updates that student's status (Waiting, Being Answered, Completed). In addition, the page should display all the office hours information, with a form and a button function for the faculty to add/remove his or her office hours.

Backend (40 hours) -Mengfei, Mike

› Server has a Queue in the database that multiple clients can get access

› Process requests and display changes when students are added to the queue and faculties remove students from the queue

› Student can see the broadcast messages from faculty

› Students will see other students ahead of them in the queue.

› Student names will be returned to the faculty for them to check off insequence

Database (20 hours) - Zhewen, Enyan

› The database will consist of four tables – a registered user table, a queue table to store current status of the queue, a StudentInQueue table to store current students in queue, and a faculty table to store current on-duty faculty.

› The user table will be used for validating users and consist of userID, username, password, fname, lname, user identity(student/faculty), and email

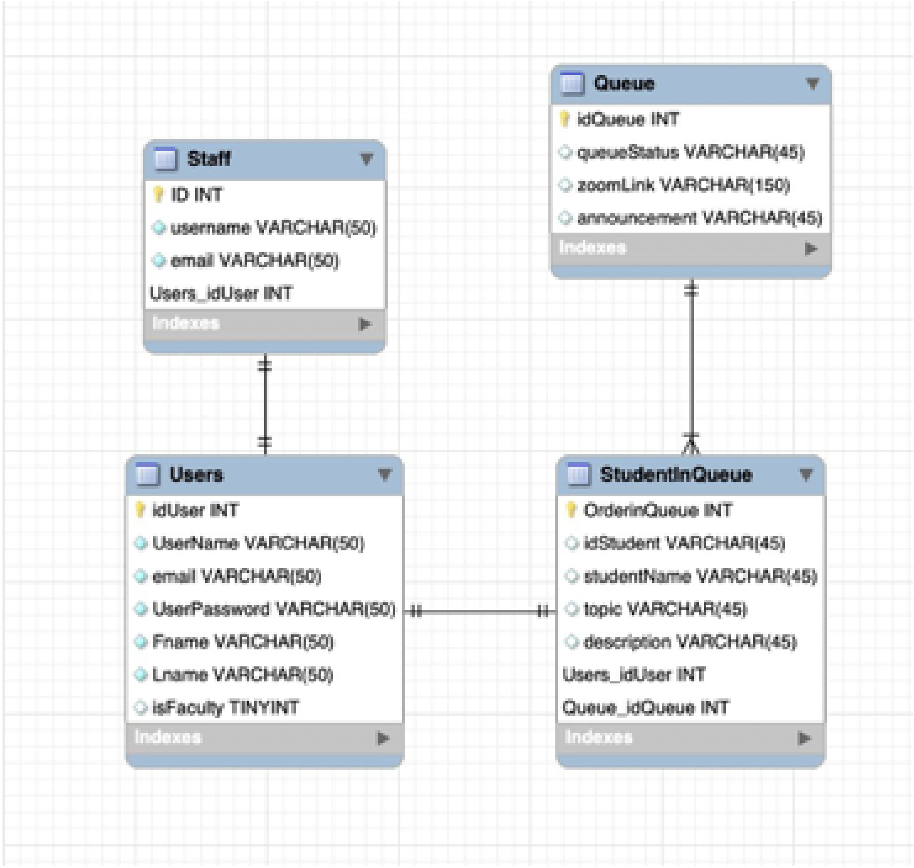
› The queue table will consist of idQueue,queueStatus,zoomLink,announcement

› The StudentInQueue table will consist of Order in Queue, idStudent, studentName, topic, description

› The Staff table will consist of username and email

Full Design

Database schema



Hardware and software requirements

Environment

- Front end will be done using JS, HTML, CSS
- Back end
  - MySQL server for user data and events
  - Java for multi-threading
- Tomcat to deploy and host the project, which can be accessed by all devices in LAN

Class diagram and inheritance hierarchy

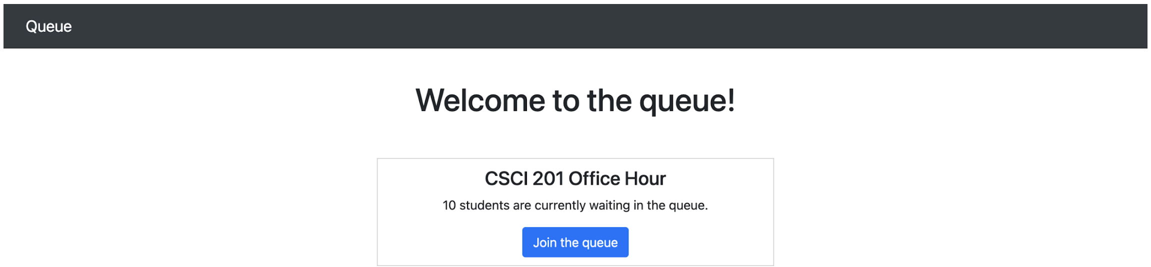
Server Main Class

- Guest Page Socket
  - Show total number of student in queue to all guest users
- LoginServlet
  - Receive login in email and password from user
  - Check with the database, authenticate the user, direct to student page if the user is a student and direct to faculty page if the user is a faculty
- Sign Up Servlet
  - Receive sign up email and password from user
  - Check with the database, authenticate the user, direct to student page if the user is a student and direct to faculty page if the user is a faculty
- Student Page Socket
  - Add the student to queue if student requested
  - Delete the student from queue if the student decides to leave
  - Pass information to the frontend with -queueStatus, studentCount = -1; zoomLink, staffs, announcement, students = "["
- Student Class
  - Student object to encapsulate data together
  - Has studentName, student ID, topic, description, and leave status
- Server Socket Faculty
  - Board cast message
  - Check student off
  - Send data to frontend

Web page Design

- We will be using Javascript mainly for the front-end development creating transparent user-interface and HTML/CSS for page layout and decoration purposes.
- Using RegEx for the validation of username and password, Form for users to enter their info, Buttons for users to either join or leave the queue, DOM Storage to store client-side data and jQuery&Animation for interaction.
- Document Object Model(Login):Document—>Body—>h1—>Form
- Document Object Model(Student/Faculty):Document—>Body—>Class(FrontNote)—>h1—>Class(notification)—>Text—>Class(student queue)—>Text—>Form

Guest Page

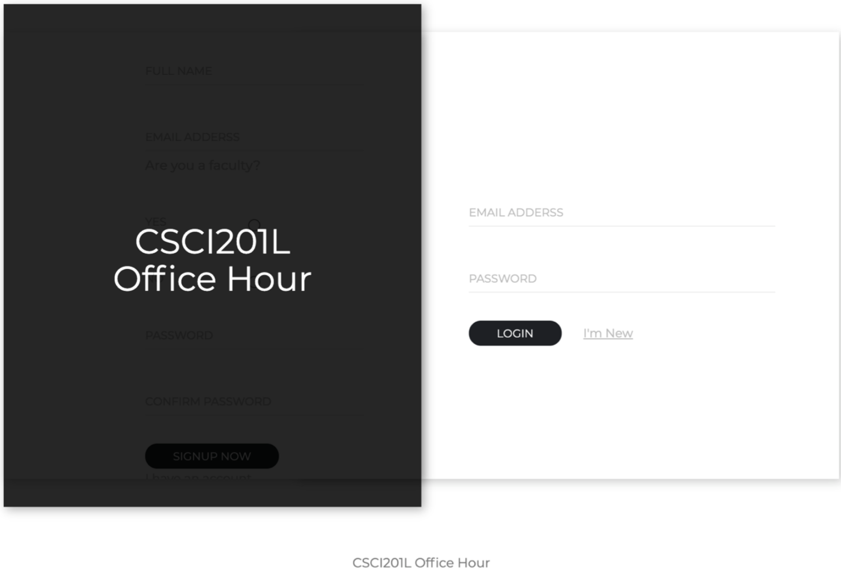


Welcome to the queue!

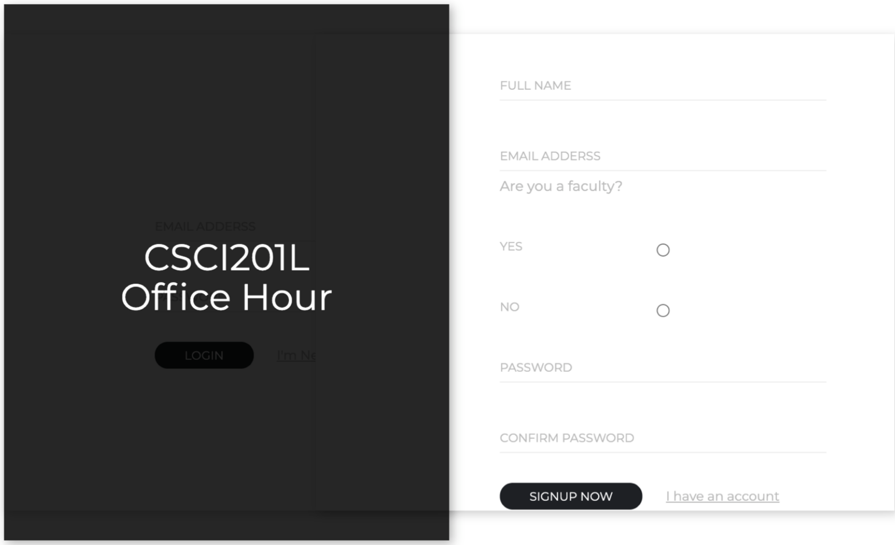
CSCI 201 Office Hour

10 students are currently waiting in the queue.

Login Page



Signup Page



CSCI201L
Office Hour

FULL NAME

EMAIL ADDERSS

Are you a faculty?

YES ○

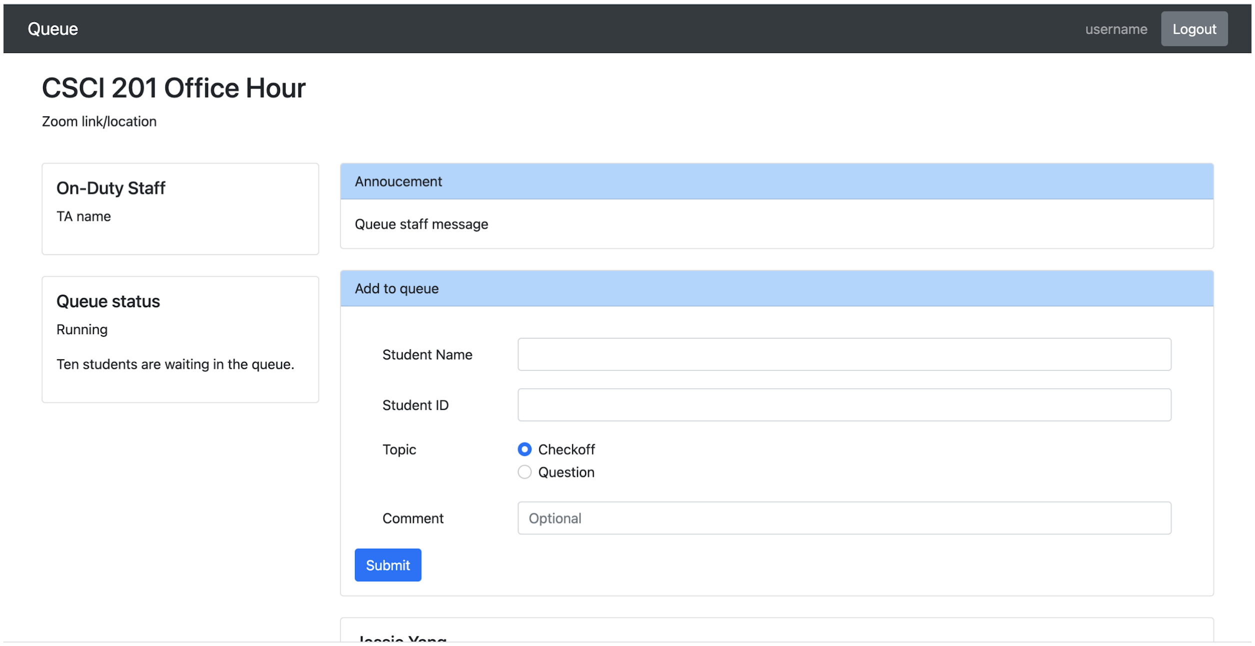
NO ○

PASSWORD

CONFIRM PASSWORD

CSCI201L Office Hour

Student Page



CSCI 201 Office Hour

Zoom link/location

On-Duty Staff

TA name

Queue status

Running

Ten students are waiting in the queue.

Annoucement

Queue staff message

Add to queue

Student Name

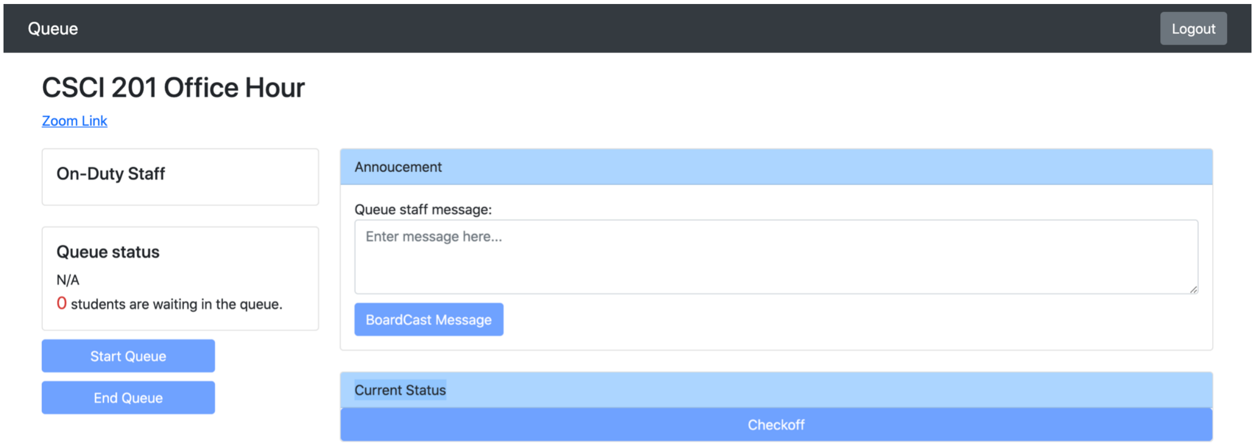
Student ID

Topic ● Checkoff ○ Question

Comment Optional

Jessie Yang

Faculty Page



CSCI 201 Office Hour

On-Duty Staff

Queue status

N/A

0 students are waiting in the queue.

Annoucement

Queue staff message:

Enter message here...

Current Status

Testing Plan

a. Login Function Test
   i. Test Case 1
      1. White Box Test - test the login functionality by specifying a username that does not exist. The user should be taken back to the login page with a message "Wrong username or password, please try again".
   ii. Test Case 2
      1. White Box Test - test the login functionality by specifying a username that exists and a password does not match. The user should be taken back to the login page with a message "Wrong username or password, please try again".
   iii. Test Case 3
      1. White Box Test - test the login functionality by specifying a username that exists and a password that matches. The user should be taken to the student page or faculty page based on *isFaculty* in the database.
   iv. Test Case 4
      1. Unit Test - insert SQL code in the username variable of the UserAuthentication.authenticate method. Verify that the SQL code specified is not executed against the database.

b. Guest Page Function Test
   i. Test Case 1
      1. Black box Test - Guest page should correctly display the queue status (either the number of students waiting in the queue, or the queue is closed)
   ii. Test Case 2
      1. Black box Test - Join the queue button will redirect the user to the login page

c. Student Page Function Test
   i. Test Case 1
      1. Black Box Test - test the on-duty staff is correct according to the current time
   ii. Test Case 2
      1. Black Box Test - test the announcement is the same as the latest announcement the faculty posted
   iii. Test Case 3
      1. Black Box Test - test the queue status changes as the faculty changes the queue status

iv. Test Case 4
   1. Black Box Test - test queue status showing the number of students waiting in the queue will increase or decrease when changes are made.

v. Test Case 5
   1. Black Box Test - test that students could either join or leave the queue and the status of queue changes as expected;

vi. Test Case 6
   1. Black Box Test - test that one student cannot add to queue again once entered, unless leave the queue first

vii. Test Case 7
   1. Black Box Test - test that current status shows all students currently waiting in the queue

d. Queue function test

i. Test Case 1 - Student test
   1. Unit Test - test the queue functionality by having a number of students login and join the queue. Check if the position of the student in the queue and the order they are checked off is correct.
   2. Unit Test - test the leave queue functionality by having a number of students join the queue, and some of them leave the queue. Check if the queue still functions correctly.

ii. Test Case 2 - faculty test
   1. Unit Test - The faculty dequeue the queue by checking them off one by one. Check if the position of the student in the queue and the order they are checked off is correct.
   2. White Box Test - test the edge case when faculty want to pop students from an empty queue.

iii. Test Case 3 - edge case
   1. Blackbox Test - let a number of students join the queue, some of them leave the queue at random time while the faculty checks students off.
   2. White Box Test - Test when a faculty check the student off and student leave the queue at the same time.

Deployment Documentation

- Deploy web server to Tomcat server, accessible through LAN. Note: Change the ip address to your home ip.
- Create sql tables with .sql files in the util folder and connect to the databases. Our current database password is ‘vincent0124!’, you can either change your database password to this or change the password in the code.
- You can run the code from any device in LAN. The landing page is guestPage.html.