

# Office Hour

CSCI 201

Final Game Project

Complete Documentation

Team Members:

Hudson-Han Nguyen

Jacob Cashman

John Meyering

Catherine Su

Denny Shen

Joseph Lee

## Table Of Contents

Project Proposal.....	3
High Level Specifications.....	4
Technical Specifications.....	6
Testing Document.....	9
Detailed Design Document.....	15
Deployment Document.....	22

# Project Proposal

Diner Dash is a popular ultra-casual game that serves people everywhere in their need to waste the precious time they have on this Earth. The gameplay involves controlling a waitress who sits and serves guests in a bustling restaurant. However, not everyone enjoys the aesthetic of Diner Dash, and therefore the need for 'Office Hour' arises. Our 'Office Hour' will be a reskin of Diner Dash, wherein the user controls a CP tending to the needs of the struggling students, a perfect game for students accustomed to wasting away their lives in SAL and find the activity comforting in times of boredom and procrastination.

## High-Level Requirements

Upon entering the website, the user is greeted with a login page, which will provide the options to continue as a guest, log-in, or create an account. Whether the user decides to login or continue as a guest will have little impact on the game experience itself; being a guest will simply mean the user's score will not contribute to the leaderboard.

The game area consists of a room with five chairs, a CP character that the player controls, a "lives counter" in the top right corner that starts at three lives, a score counter on the top left and a leaderboard with the top three players' scores in the bottom left.

A queue of students will form at the bottom of the screen and if there are any empty chairs, the next student in the queue will go to sit in it. When the user clicks on a student, the CP character will move to that student and an icon will appear above their head, indicating the type of problem they have.

There are two computers at the top of the room. The user may click on the computer and it will load the answer to the problem. Once it loads, the user may take (serve) the information to the student. This will increase the score in the top left by a number proportional to the student's remaining patience level and the student will leave.

Each student will have a timer corresponding to their patience. As they wait, their patience decreases and if they lose all their patience, they will leave SAL. If the user serves the information to the wrong student, that student will lose a quarter of their patience, which could result in them leaving right immediately. A student leaving will deduct a life from the user's "lives counter".

The queue will increase throughout the game as more students come for help. They will have a patience bar for waiting in the queue, and if they spend too long standing and waiting for help without a chair opening up for them to sit on, then they will leave without having sat down. This will also deduct a life.

Once the queue of kids is empty and all students on the map have been helped, the round will end, and the user's points total from that round will be displayed in the center so they can see how they did. A "next round" button will appear, which the user may press when they are ready to continue. Each successive round will be more difficult than the last, with more overall students in the round, and smaller patience timers on the students. Completing three levels will grant the user an extra life. Once the user runs out of lives, the score they had in that round up to that point will be halted and displayed as their final score. Their position on the overall leaderboard will be displayed if they are a logged-in user, then they will be prompted to select whether they would like to play again or return to the logged-out menu. A guest user will have the same experience but their position on the leaderboard will not be displayed since their score will not be added to it.

The leaderboard in the bottom left tracks the scores of all users who have logged in and will update in real time, as the user is playing the game.

Classes:

Login Page: Deals with everything regarding the functionality of the login page.

Game: The main game loop controller. This class organizes the rounds and the screen when the user loses the game which asks them if they would like to play again. It also keeps track of the player's score and lives.

Map: This class is connected with individual rounds in the game. It tracks the objects and players on the map during a round.

Actor: Abstract class that refers to the objects on the map. Sub-classes which inherit from this implement its functions.

Class:	Info:	Class:	Info:
Game	The main game loop controller	Student	Inherits from Actor
Actor	Objects on the map	Login Page	functionality of the login page
Player	Inherits from Actor	Map	Tracks the actors and components on the map
Chair	Inherits from Actor	Component	
Move Component	Inherits from Component, Dictates how an actor moves	Sprite Component	Inherits from Component, Dictates what sprite it has

# Technical Specifications

## **Main Menus (4 hours):**

- The title screen will contain the title of the game, and buttons for “Sign up”, “Login”, and “Continue as Guest”
- Pressing “Sign up” will lead to a different menu which has text fields for username and password as well as buttons for “Submit” and “Go Back”
  - If the information entered is empty or the username is repeated in the database, then pressing “Submit” will prompt the user to adjust the information; otherwise, it will add the user information to the database and continue to the game
  - Pressing “Go Back” will send the user back to the previous page
- Pressing “Login” will operate in the exact same way, but instead, it will prompt the user to reenter if the username doesn’t exist; it will not change the database in any way
- Pressing “Continue as Guest” will start the game in guest mode

## **Game (10 hours):**

- Will serve as the main game loop controller, and will keep track of all aspects of the game that carry over between rounds, including: lives counter, score, # of rounds completed/round difficulty
  - Lives counter: Starting with 3 lives, the lives counter will deduct a life as prompted by the map, and will add a life after the completion of every three rounds (as indicated by the # of rounds completed)
  - Score: At the end of a round, the game class will retrieve the updated score from the user, then update the database with that score, then retrieve the database (which may have updates from other players playing at that moment); it will then initialize the new round with the new leaderboard, and the updated score
- Will also be responsible for setting up each successive new round, providing inputs such as: # of overall students in that room, difficulty rating of the round which will correspond to the patience timers on the students and/or the number of students that enters the queue at the same time?
  - These will be based on the # of rounds completed/round difficulty: a simple tracker of how many rounds have been completed
- Control loop: When the user runs out of lives, their score will be updated in the database, a “game over” screen will appear asking if the user would like to play again; there will be a “Play Again” button and a “Quit” button
  - Pressing “Play Again” will restart the game and reinitialize the lives counter, # of rounds completed/round difficulty, and score
  - Pressing “Quit” will return the user to the title screen

## **Map (20 hours):**

- Will manage the individual rounds of the game
- Will track these objects on the screen and manage their processes: Player, Student, Chair, Computer, Queue (of students)
  - Player: able to retrieve questions from students at chairs, input questions into the computer, retrieve answers from the computer, and return answers to students
    - Can hold one piece of information (question or answer) at any given moment

- Student: patience meters will be based on a patience level which will be determined by a random probabilistic algorithm which uses the difficulty level of the round
  - Will walk to a chair whenever there is one open
  - Will have two separate patience meters: one while they are waiting in the queue, and one while they are at the desk waiting for help
  - Will have an indicator above their head which shows the type of question they had asked after the player retrieves a question. This will go away either when they are given an answer or runs out of patience
  - Patience Meters: Will have a set timer and countdown to zero. Will have a sprite that corresponds to that timer
- Chair: will have an open/closed condition that indicates whether it is an option to the student in the front of the queue
- Computer: will have 3 states; ready, active, waiting
  - Ready: available to retrieve a question from the player; player must be holding a question to interact with it
  - Active: working on retrieving an answer; player will not be able to interact with it in this state
  - Waiting: able to give answer to player; player must be holding no information to retrieve it
    - Will have a different sprite depending on the question it was given
- Queue (of students): there will be a fixed number of students that will appear in the overall round based on the # of rounds completed up to that point
  - Each student will enter the queue at a random time which will be determined at the beginning of the round; but it will be moderated to avoid too many students coming at once
    - The rate of students entering the queue will also increase as the difficulty rating of the round increases
  - If there is only one student sitting in a chair and the rest are empty, a student will enter the queue, no matter the interval
- When the queue is empty, all of the chairs are open, and there are no students left to arrive, that will signify the end of the round and the end of the instance of the map

**Database (4 hours):**

- The database will be made of two tables - the User table and the Score table
- The User table will be used for user authentication and will consist of userID, username, and password
- The Score table will be used for keeping of the scores in the game and will consist of userID, attemptID, score received, and timestamp of when the score was completed
  - userID will be used to cross reference the Score table with the User table
  - attemptID will be used to reference the score, having already been entered into the database, when it is updated at the end of rounds

# Testing Document

## Main Menus:

<b>Test #</b>	1
<b>Type</b>	Black box
Description	Clicking “Register” with correct fields filled in and a unique username will enter the user’s data in the database.
Input	Click “Register” button
Expected Result	The first screen of the game

<b>Test #</b>	2
<b>Type</b>	Black box
Description	Clicking “Register” with a field blank will refresh the page indicating to the user which field was blank.
Input	Click “Register” button
Expected Result	Page will refresh, displaying error message

<b>Test #</b>	3
<b>Type</b>	White box
Description	Clicking “Register” with a username already entered in the database.
Input	Click “Register” button
Expected Result	Should refresh the page telling the user that the Username already exists

<b>Test #</b>	4
<b>Type</b>	Black box
Description	Clicking “Continue as Guest”, this will allow the user to continue to play the game without saving its score onto the leaderboard. (Continues to show score, however, just doesn’t save it anywhere)
Input	Click “Continue as Guest” button.



Expected Result	The first screen of the game
-----------------	------------------------------

<b>Test #</b>	5
<b>Type</b>	Black box
Description	Clicking “Login” with the correct fields will grab the user’s information (username) from the database so that the user’s score can correctly display the right name on the leaderboards.
Input	Clicking login
Expected Result	Retrieves username from database and uses it to display high score

**Game:**

<b>Test #</b>	1
<b>Type</b>	White box
Description	At the end of a round, the game retrieves the current leaderboard from the database and displays it on the screen.
Input	Player finishes the round.
Expected Result	The correct current leaderboard according to the MySQL database is displayed on the screen.

<b>Test #</b>	2
<b>Type</b>	Black box
Description	Test the life total status for real-time updating functionality.
Input	A student has lost patience and leaves SAL.
Expected Result	The life total decrements by 1 and a heart container is lost.

<b>Test #</b>	3
<b>Type</b>	Black box

Description	User wants to Try Again after finishing one game.
Input	Clicking “Try Again”.
Expected Result	The first screen of the game.

<b>Test #</b>	4
<b>Type</b>	Black box
Description	User wants to Quit after finishing one game.
Input	Clicking “Quit”.
Expected Result	Goes back to the title screen.

**Map:**

<b>Test #</b>	1
<b>Type</b>	Black box
Description	When a chair is clicked, the CP moves to the chair and stands in front of it.
Input	Mouse click on chair.
Expected Result	CP moves in front of chair that was clicked on.

<b>Test #</b>	2
<b>Type</b>	Black box
Description	When a computer is clicked, the CP moves to the computer and stands in front of it.
Input	Mouse click on computer.
Expected Result	CP moves in front of computer that was clicked on.

<b>Test #</b>	3
<b>Type</b>	White box
Description	When a chair with a student is clicked and cp is holding nothing,

	isHolding becomes true and info = student's question.
Input	Mouse click on chair with student.
Expected Result	isHolding becomes true and info = student's question.

<b>Test #</b>	4
<b>Type</b>	White box
Description	When a chair with a student is clicked and cp is holding an answer, isHolding becomes false, info = null, student's question is removed.
Input	Mouse click on chair with student.
Expected Result	isHolding becomes false, info = null, student's question is removed.

**Student:**

<b>Test #</b>	1
<b>Type</b>	Black box
Description	During the time in which a student is seated, does their patience meter decrease over time?
Input	Mouse click on empty chair with unseated student selected, wait for specified time interval.
Expected Result	A student's patience meter should be decreasing over a constant time rate.

<b>Test #</b>	2
<b>Type</b>	Black box
Description	When a student leaves the map, depending on their patience meter and the state of which they left the map (unfinished or finished), does the player's score get subtracted/added according to the leaving student's state?
Input	Seat a student and observe the player's score according to the student's leaving status.
Expected Result	Player should be awarded with additional points for each student who leaves when their problem is solved, bonus points depending

	on how much patience they had left and the student's problem type. Player should be penalized their points when a student leaves with their problem unfinished depending on what state they were left in.
--	---

<b>Test #</b>	3
<b>Type</b>	White box
Description	When a student object's patience is set to 0, and solProblem = false, is scoreNum decreased?
Input	When the player seats a student, isSeated = true, then decreasePatience() is called over a constant rate of time.
Expected Result	When patience = 0 and solProblem = false, scoreNum from Game class should be decreased. Amount of scoreNum decreased depends on orderTaken and isSeated boolean flag.

**Database:**

<b>Test #</b>	1
<b>Type</b>	White box
Description	Testing that the users and scores tables are connected properly.
Input	An SQL Statement: SELECT * FROM users JOIN scores ON users.userID =scores.userID;
Expected Result	All rows from each table should be displayed and joined according to userID.

## Detailed Design

### *Hardware:*

Windows: Windows 7, Windows 10

Mac OS X: Intel-based Mac running Mac OS Mojave 10.14.6

### *Software:*

JavaScript (JS ES6)

HTML5

Java 8 (JDBC 4.3)

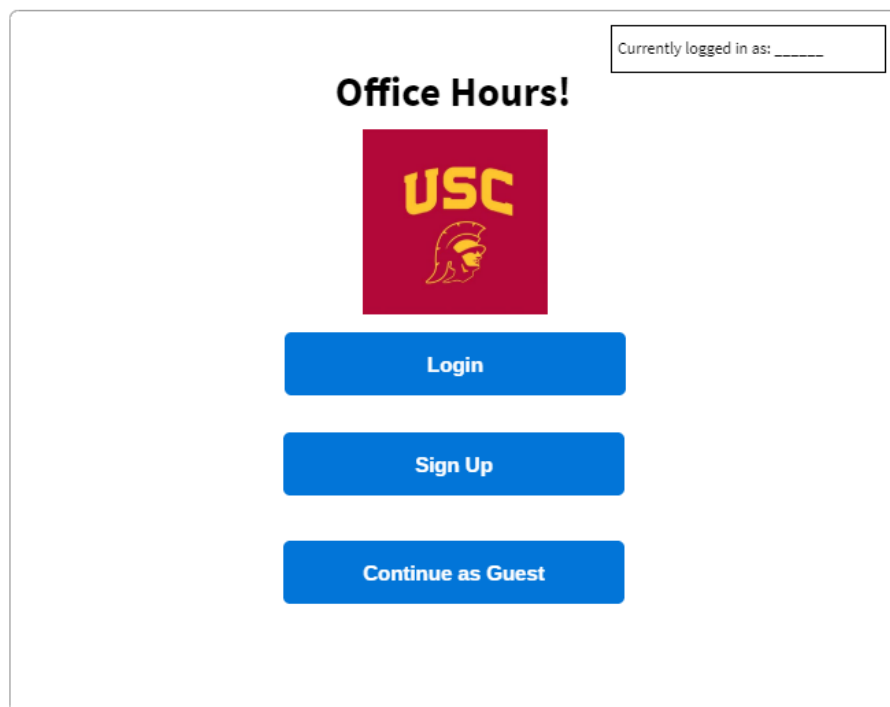
Eclipse IDE for Java EE Developers

MySQL

Chrome

### **Main Menus :**


- Title Screen:
  - The user is first directed to a sign-in page. They can choose to log-in, sign-up or continue as guest.




The mockup shows a title screen for a web application. At the top right, there is a box labeled "Currently logged in as: \_\_\_\_\_". In the center, the text "Office Hours!" is displayed above the USC logo, which features the letters "USC" in yellow on a red background with a Trojan helmet below. Below the logo are three blue buttons with white text: "Login", "Sign Up", and "Continue as Guest".

- Screen if user presses 'Sign Up':
  - The user is asked to input a unique username and password.

## Sign Up


 Username


 Password

Register

- Screen if user presses 'Login':
  - The user is asked to input username and password. If a valid match of username and password is entered, the user would be taken to the game page, otherwise an error is displayed. The backend should take care of the authentication.

## Login

 Username

 Password

☒ Remember me

Login

- If user presses 'Continue as Guest':
  - Game starts in guest mode

User	String username String password int currentScore int highestScore void setUsername(String username) void setPassword(String password) String getUsername() String getPassword() int getCurrentScore() int getHighestScore() void setCurrentScore() void setHighestScore()
MainMenu	void startMenu() void toSignUpPage() void toSignInPage() void toGuestPage() boolean checkAuthentication(User user)

**Game:**

Game	int lives int scoreNum int roundsDone Pair<String, int>[] leaderboard //keeps track of in-game live leaderboard in an array of pairs, in which there is a String for the username and int for the score  void newRound() //runs the next round, creating a new map void updateDatabase() //updating database (for end of round) void retrieveLeaderboard() //retrieving leaderboard from database (to start round)
Map	int roundScoreNum int roundNum Queue<Student> studentQueue Computer[] computers Chair[] chairs Game game  Map(Game game) //Map constructor, takes in the

	data from the game and uses that to initialize map void startRound()
Student	String problemType boolean isSeated boolean orderTaken boolean solProblem int patience int getPatience() void setPatience() void decreasePatience() boolean orderTaken() boolean solvedProblem()  void seated() //Student is seated, so the patience meter will be reset and the student waits for their problem to be solved  void leaveUnfinished() //Player did not help the student on time, student leaves and player's health is decreased  void leaveFinished() //Player was able to help the student completely, student leaves and player's score is added

- Screen between rounds:



**End of Round \_\_\_\_**

**Current Score:** \_\_\_\_\_



**Continue**

- Game over screen:

**Office Hours Closed!**

**Top Scores:**

1) Jeffrey Miller	999,999
2) Reem Alfayez	999,999
3) Jack Xu	999,999



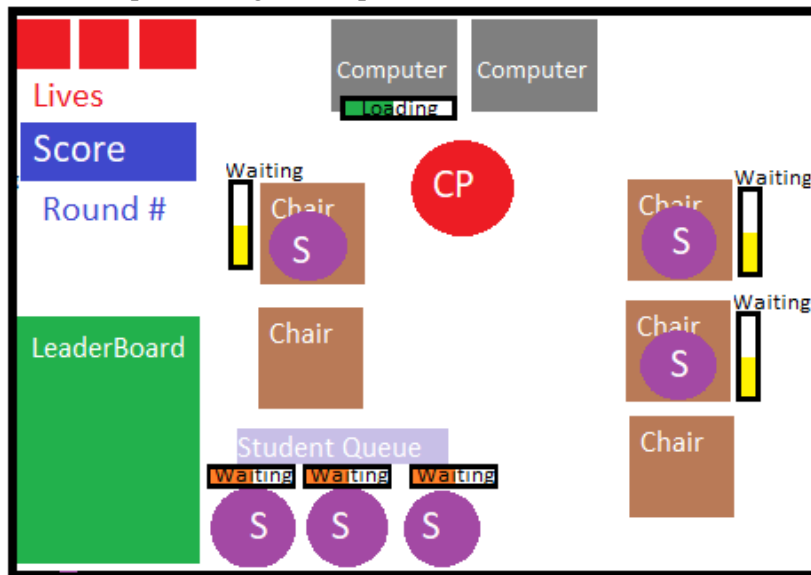
**Current Score:** \_\_\_\_\_

**Try Again**

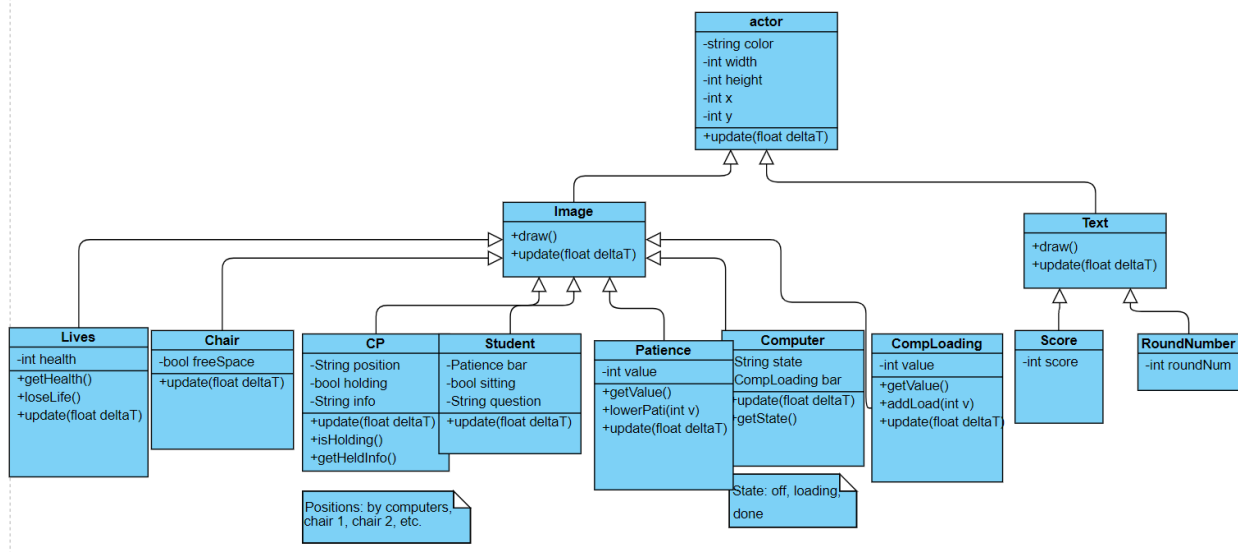
**Quit**

### Map:

- Depiction of game map:



Class Diagram of objects that could be allocated in Map:



### Database (schema):

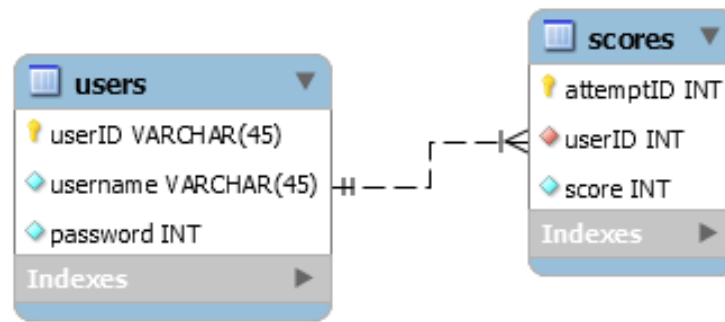
Users

- VARCHAR(45) userID (primary key)
- VARCHAR(45) username
- INT password

Scores

- INT attemptID (primary key)
- INT userID (foreign key)

- INT score



UserInfoJDBC	<pre> void addUser(String username, String password, int highestScore) boolean doesUserExist(String name) boolean checkAuthentication(String username, String password)  void sortUserScore() String[] topThreeUser() Int[] topThreeScore()  // use sortUserScore() to sort the database by score in descending order, and then topThreeUser() and topThreeScore() would return the top three users with the highest scores to fill the leaderboard </pre>
--------------	--

## Deployment Document

1. Open Eclipse
2. Import the project.zip "office\_hours.zip"
3. Open the project folder
4. Open the SQL script into your MySQL Workspace
5. Navigate to Server.java
6. Run Server.java on Tomcat
7. Open a proper web browser (e.g. Chrome)
8. Go to localhost:8080/mainmenu.html
9. Have fun!