Complete Documentation

----------Code Racer

Team# 3

Team Members: Jiawei Wang, Mingxuan Li, Zhonghang Huang, Andrew Valle

Table of Contents

Project Goal:

The goal of our project is to create a website for typing challenge game, specifically typing code. We aim to get current programmers, future programmers, and even potential programmers to be more familiar to programming through competing over a great variety of coding examples. We also aim to make ourselves better programmers by practicing front-end as well as back-end development skills learned in CSCI-201.

Project Description:

For our project, we are choosing to make a create a typing challenge game. This game will measure the speed and accuracy of a user's typing skills against his or hers other challengers. Since all programming languages frequently use special characters, not memorizing where specific keys are might slow a programmer down by having to look down at the keyboard. There is no current programs out there that help people get faster at typing code, just normal paragraphs. We will also add other features discussed in the project requirements section.

Project Requirement:

- User Login Functionality:
    - Guest users will be provided a generic username
    - Not be able to have updated average scores (only scores from individual played games will be displayed)
    - Will not be put on the high scores list.
    - Only authenticated users will have these features.
- Aesthetically-pleasing: No command line programs
- Multi-threading and networking functionality
    - Check typing progress for each player
    - Timely update the time record

Timeline:

Date	Objectives by the date
2019/05/29 - 2019/05/31	Design the outlook of the game and figure out how to implement the game by the servlet and front-end.
2019/06/01-2019/06/03	Finish the database as needed
2019/06/04-2019/06/06	Finish the GUI page part(front-end)
2019/06/07-2019/06/09	Finish the servlet part(back-end)
2019/06/10	Test and debug
2019/06/11	Present

Group Discipline:

Communication Methods:

- We will communicate via face-to-face or online meeting. Group members should respond to any messages within 1 day.
- All documents will be shared between the group members in Google Docs.
- In-person meetings will be held for important decisions and compiling documents.
- There will be a review announcement before every due date, so everyone can check it and re-work the wrong parts.
- Scheduling will be coordinated between group members to avoid any conflicts with a student's commitments.
- All weekly meetings are required. If anyone cannot attend the meeting please let everyone know in advance via email.

Group Ethics Statement

- All assignments will be of high-quality work and will be agreed upon by the majority of the group members by approval within the group discussion boards.
- We will not allow any behavior that will interfere with the group's progress towards their goal, and not allow any attitude that causes discomfort or intimidation within the group. If these behaviors are presented, the instructor will be notified.
- Distribute work evenly among the group, and strive to complete the task in an efficient manner as well as achieving a result above the expectations of the instructor.
- No plagiarism in the teamwork. Once such behaviors have occurred, the report will be immediately sent to the instructor.

Group Discipline Procedures

A member can be removed from the group for any of the following conditions:

- Failure to attend more than 3 group meetings.
- Failure to communicate or respond in a timely manner as set in Communication Rules.
- Failure to complete assignments within a timely manner.
- Failure to put forth their best effort.
- Any plagiarism behavior.

Contact:

Name	Email	Phone Number
Jiawei Wang	wangjiaw@usc.edu	8148269059
Mingxuan Li	mingxual@usc.edu	2137091158
Andrew Valle	Andrewjv@usc.edu	3107485453
Zhonghang Huang	zhonghah@usc.edu	3234018588

Help Sources:

TA:

Adelayde Rome adelaydr@usc.edu

Professor:

Jeffrey Miller jeffrey.miller@usc.edu

High-Level Requirements:

- Front-End
    - Login Feature
        - Users are categorized into two groups: guests and members
            - Guest: limited access to website functionalities (records in ranking)
                - Guests also are given a generic username
            - Member: full access to website functionalities
    - Page Design
            - Aesthetic layout of the content to make it appealing for users
            - Effective links to the various functionalities provided
            - Create a login homepage that uses a form to submit users information
    - Typing Game
        - Creating a visually pleasing, fun, interactive and educational game
        - A player can choose either to play alone or to compete with others.
        - Use simple graphics to represent the progress of typing
        - Show the top ten player results on the page
- Back-End
    - Multi-thread Time Record
        - Timely check and update the players' progress on the screen
        - The time used for each player will be recorded individually
        - Collect the result and update the database timely
    - Servlets to verify login information as well as when creating new accounts, checking if a username and password combination is valid
        - Guests cannot participate in the game but only to check the information given in the website
- Database
    - Initial Database
        - Create tables to record the username, userID, user password, user icon and user ranking data.
            - Insert new user into database if people register a new account. Or people need username and password to login their private account which records their personal information. And all the username, password and other information are acquired from database.
            - There should be a table that could indicate the current ranking for all users. So we could post the top 10 from the database.
        - Create tables to store the codes in use of
            - Normal typing game which players will compete on.
            - Error code which players will need to figure out and solve
                - In the meantime, store the cases that the code will be tested and graded on

Technical Specifications:

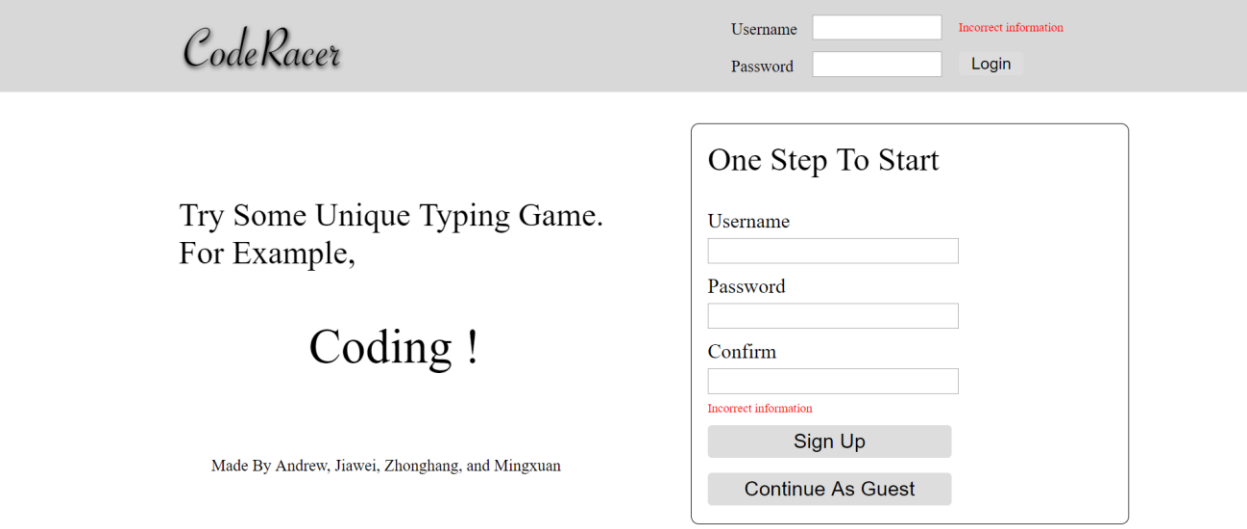
- Web interface (7 hours)
    - Starting from the homepage, the web interface needs to have a form of letting the user choose to login, register, or sign in as a guest. When clicking on a choice the user will be directed to the correct page.
    - The login page will have a username field, a password field, and a Login button.
    - Registration page will have a username field, a password field, a confirm password field, and a Register button.
    - Clicking on sign in as a guest will direct the user to the correct page and will have limited capabilities/access.
    - Upon verification, the user will be directed to the page where to play the game, guest users will only have access to typing game, not able to choose their name, not be able to be uploaded to high scores list.
    - There will be a view ranking button for both guests and users on the home page. When clicking on it, a table of the five highest scores will be shown.

- Database (5 hours)
    - The database will consist of two tables – a User table, a High Scores table.
    - The User table will be used for validating users and consist of userID, username, password as well as possibly keeping track of the personal best score.
    - The High Scores table will consist of the top ten players which would consist of userID and scores. This table needs to be sorted from Highest to lowest.

- Speed Typing Code Game (7 hours)
    - When the user clicks on the button to play the game, the user will be directed to page that the game does not start yet. The sample will not be shown at the time. The user can invite their friends to play, up to four people in total. The usernames will be shown on the left box, from the first to the last. The first person will have the right to start the game.
    - Once the game starts, there will be two big sections on the page. In the top section, a sample will be randomly picked from the database and shown to the user. In the bottom section, there will be a blank area for the user to type.
    - When the user finishes typing, there is a button on the bottom to submit the code, we will compare it to the sample we give and check if they match. If not correct, there will be an error message shown just on top of the submit button. If correct, the time will be recorded, and the rank will be updated.

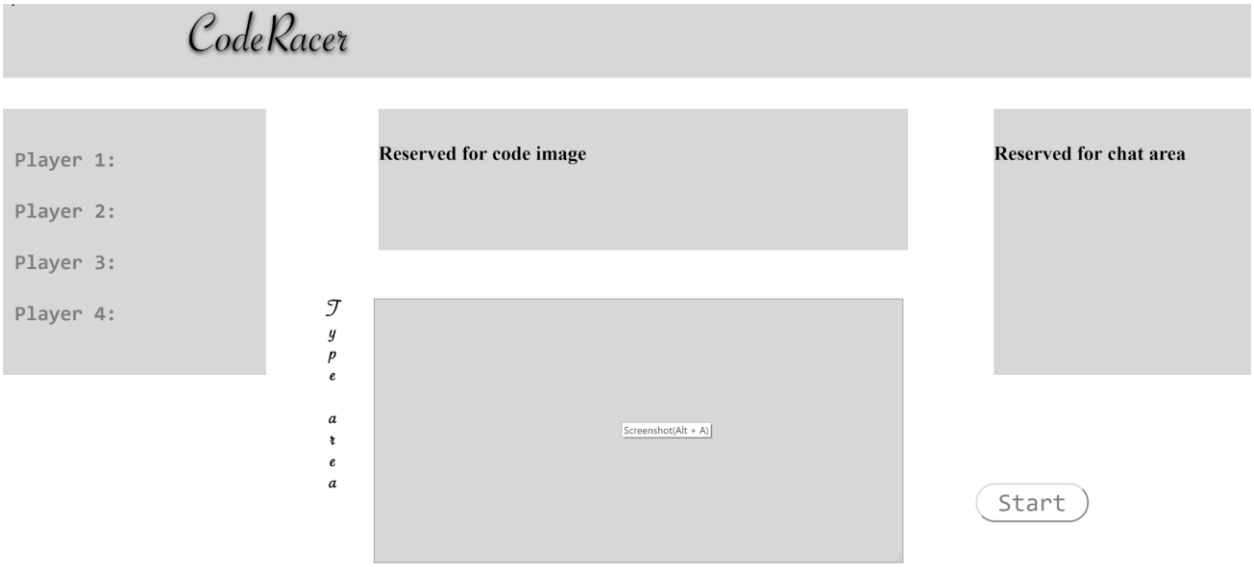
Detailed Design:

- Homepage
    - The homepage looks like this:



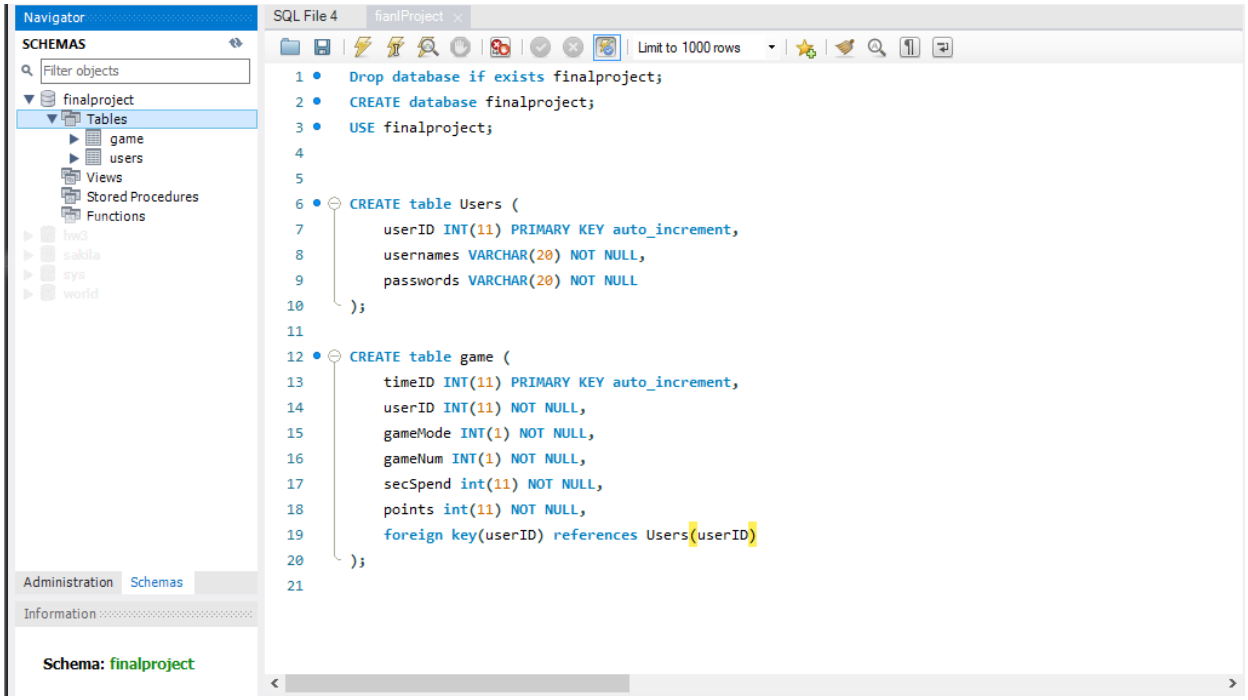
- There are three main functionalities that the homepage provides: login, register, and continue as guest.
- The Validation class will access the Users table in a sql database called finalproject using the jdbc:mysql driver. If the user has already logged in, the current request to login will be denied.
- The Register class will access the Users table in the same way as the validation class. It will check if the username exists and properly insert new user into the table once all the information passes.
- For these two classes, any error message will show up on the red area. If there is any error, the user will not be directed to next page. An AJAX method will be called when the user click the submit button.
- The Continue As Guest option will allow the user to access the next page without login. Anyone choose this option will be given a generic name. At the same time, although these people can play the game, the scores they get will not be recorded into the database.
- The button CodeRace on the top left will direct the user to the home page, if the user is already signed in, the form in this picture will be changed to buttons Sign Out and Records Check, which allows the user to log out and see their game history.

- Gamepage
    - The play page looks like this:



- The left area shows who is currently in the room by taking parameters passed in from the home page
- The middle-top area will become an image of given text or java code for players to type once the players click “start”, calling JavaScript function to change the src of the image and start the timer.
- The middle-bottom area is for the players to type. It will give instant feedback if players type something that does not match the given text. This is achieved by taking advantage of onkeyup() event, calling AJAX every time when something is entered and comparing it with the text stored in the database.
- The right area allows players to chat by establishing a multi-threaded network.
- Next to the names of players will be runways, where the position of the car represents how much the player has completed. The movement of the car is achieved by calling a backend function.

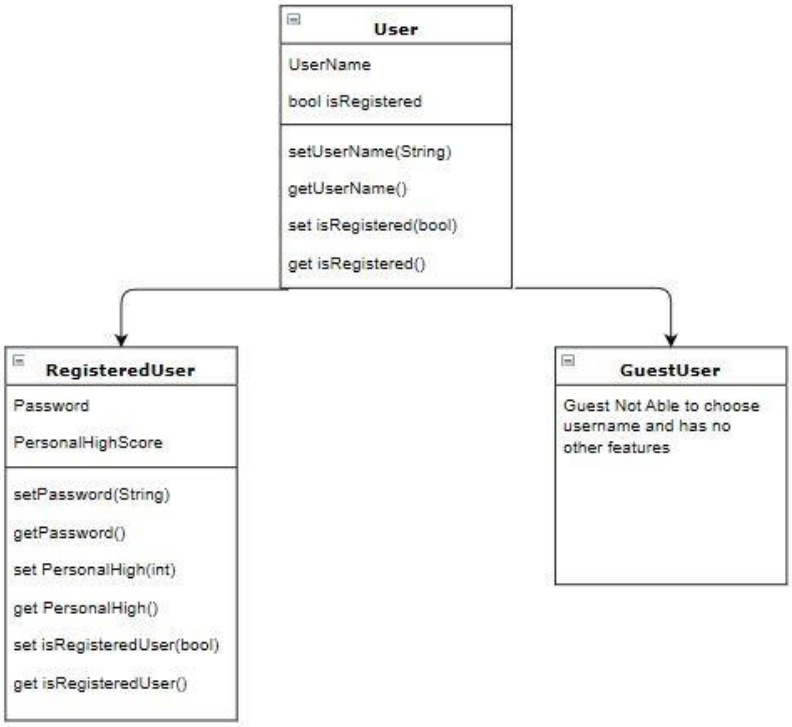
- Database Schema
    - Here is the SQL for the database:



    - We will have a database with two tables. One table is for users and the other for the game.
        - The user table stores the username and password for each of the users, so there will not be any duplicated username and the system and check the user based on the correct combination of username and password.
        - The game table stores the statistics for users' scores and times in different level games. So the system can make a rank to show the top 3 in each game.
            - The userID in the game table should be a foreign key that is linked to the userID in the Users table.
    - Whenever a user finishes a game, the game table will record the game and check with the previous record, if the user played the same game before, the database will only record the best score.
    - Whenever a user registers as a new player, we will check the username to avoid the same username.

- Hardware and software requirements
    - For the back end, we will use the java as our servlet's language.
    - For the front end, we will use html and jsp to form the page, and use css to decorate the page. We may use some javascript and AJAX to add functions inside the page.
    - For the database, we will use MySQL Workbench as our data storage. So we will use MySQL to query and insert the data into the database.
    - For hardware, we will use multiple laptops to play the game at the same time so that it could be a multi-player game.

- Class diagram
    - Here is the class diagram for our users:



    - Users will be split up into two categories:
        - All users
            - Have a username and a boolean letting us know if they are a registered user or not
            - Have methods to set/get username and set/get isRegistered

- Registered Users
    - Are actually able to choose usernames
    - Have capabilities of storing their personal best score
    - Have methods to set/get all data members
- Guest Users
    - Only have a username (which guest cannot choose)
    - And a boolean letting us know if they are a registered user or not.

Testing Document:

- Test Case 1
    - White Box Test – test the login functionality by confirming that a username and password combinations match. If there is a match, the user will proceed to the typing game page logged in. Else, the user should remain in the login page with a that displays a message “Invalid login.”

- Test Case 2
    - White Box Test – test the register functionality by checking if the inputted username already exists or not. If the username is already taken, the user should be notified that ”username already taken.” The user must then keep attempting to try until a valid username is typed.

- Test Case 3
    - White Box Test – test the continue as guest functionality. If the user chooses to continue as a guest, they should be taken to the typing game page with a generic username provided and scores will not be saved as well as not being able to have friends.

- Test Case 4
    - Unit testing – test individual functionality of the code by writing customized programming. For example, test the insert and query SQL code in the register and login method for the User table, and test the insert and query SQL code in the game ranking method for the Game table. And after executing the SQL code, the database should be able to change or provide data to the system.

- Test Case 5
    - Regression Testing - When a game is finished, if any player in the current game beats any score in the high scores list, the page should automatically be updated with the current information.

- Test Case 6
    - Regression testing – When one of the players exits in the game, the game page should be able to ignore the data from this player.

- Test Case 7
    - White Box Test — test the login functionality. If the user logins in and proceeds to the game page, the username should appear at the correct location.

- Test Case 8
    - White Box Test — test the start button in the game page. Once the first user that enters the room clicks start button, the game should begin. If other users click the button, nothing should happen.

- Test Case 9
    - White Box Test — test the functionality to show given text. The area for the given text should be blank before the game. Once the game starts, this area should be replaced with given image.

- Test Case 10
    - Stress testing - since the maximum players that could play in any game is four players, test that the game could handle all four players at once and track all if the user input while updating their scores.

- Test Case 11
    - Black box testing – test the entire application without looking at the Code. Find four computers and players, and ask them to test the whole game. For example, some of them should test the register and login functions on the home page, and the others should test the game by the guest mode. Then they will play the game together and check the changing of ranking data base on their scores in the game.

- Test Case 12

    - Unit testing — test the functionality of textarea in game page. Print the content in the textarea to the console in the backend every time the onkeup is triggered, which sends the content in the textarea to the backend, to see if this functionality works correctly.

Deployment:

- Step 1 - check all codes run well in one local PC.
- Step 2 - push the application to a live server.
- Step 3 - ensure all of the data, especially database, has been migrated to the live server from the existing system.
- Step 4 - perform a full regression test of our type game, ensuring all functionalities work properly.
- Step 5 - introduce our application to our school's CS department, and negotiate with E-Week's e-boards to add our typing game in E-week.
- Step 6 - at launch, get user feedback to get ideas for future updates.
- Step 7 - notify the users of our updated features, such as different typing levels and new designs.
- Step 8 - post the game in GitHub and make it public, but declare that no one should use the code in commerce.
- Step 9 - first invite and welcome the classmates and friends to play the game. If they like the game, ask them to play it with their friends, so more and more people will know the game.
- Step 10 - make some advertisements on social media like facebook and twitter.
- Step 11 - make our finding bugs in code game available for mobile phones on both Android and iOS devices while also promoting our website.