TutorU

Dr. Miryung Kim

TA: Yun Lu

Kyoseong Ku (604-217-597)

Bolun Hu (104-056-403)

Zhiping Tan (204-021-272)

Youshan Zhuang (104-021-282)

CS 130: Software Engineering

University of California, Los Angeles

Spring 2015

I. **Introduction**

TutorU is a mobile application that resembles Uber in providing an efficient platform that connects students to nearby tutors in real-time. Our objective is to create a reliable and easy service for students to connect with the best-matching tutors available with minimum hassle, 24/7, in an efficient and comfortable way. This application includes easy-to-use, aesthetic, and intuitive features and user interface such as: simple registration and sign-in, stress-free billing, secure database management of user information, accurate suggestions of available tutors based on relevant data and user input, and text communication between the client and the tutor. In this document, our TutorU tech team presents the problem, user use cases, use features, user workflow, a mock up page, and a class diagram.

**II. Motivation**

Students’ needs arise at unpredictable times as many students study at irregular hours and may need a tutor’s assistance at times when professors, teaching assistants, and other resources are unavailable. TutorU grants students and tutors the freedom from fixed schedules and allows students to receive assistance whenever the need arises and wherever tutors are available. Tutors also gain the freedom to freelance their services during their time of choice. Furthermore, users are free from being confined to a fixed location with time constraints, such as business hours of a tutoring academy. The users arrange the time and location of meeting, and the tutoring ends when the student and tutor agrees. In addition, proximity-based tutor-matching means less time spent on commuting and more time spent on learning.

This idea is first brought forward and pitched by TutorU’s founder Tyler Phillips, who had an awful experience in finding a tutor a week before final. TutorU is aimed at providing a safe network platform within university for students to find a tutor who has taken the same class with the same professor, knows the course material and exam ranges extremely well, and is able to teach and meet up with a student at the hotspot of TutorU on campus at the time when the student is in need. TutorU mobile application serves the technology side of this platform, and hope to attract students with its simple but elegant design, few but reliable functions.

**III. User Benefits**

The concept of TutorU is similar to the idea behind Uber, which is to provide students the easiest access to assistance from the nearest tutors who have learned and understood the materials for the course they stuck on.

By quickly logging in and selecting the preferred location and course subjects, student user will be able to find the best available tutor and set up a tutorial session in less than a minute, rather than searching on the internet and waste hours of time for a simple problem which can be resolved in less than a few sentences.

On the other hand, TutorU also provides talented students the best opportunity to help others and earn quick revenues conveniently. The user interface for tutor user is simple and straightforward: click the availability button to declare that you want to help others, and we will assign student users to you.

Overall, the TutorU concept creates a win-win situation: it saves students’ time and effort when they need help on course materials, while also provides tutors the flexibility to quickly offer help to others whenever they are free. From the developer’s perspective, the TutorU platform is also a good platform for earning profits: every order associates with a transaction fee collected by the website. So as long as students are looking for help, and tutors are offering help on TutorU, everyone associated will receive benefits in someway. Therefore, TutorU is a platform that would bring benefits to users and developers.

**IV. Feature Description and Requirements**

In general, TutorU enables students to login and get help on their class materials from tutors. On the other hand, tutors are able to login and offer assistance to students needing help. We will start describing the application feature from the **perspective of a front-end client:**

Registration & Login

User registration and login process is simple. User simply needs to select the type of account he/she wants to create, email address, real name, phone number, password, and credit card number to finish the sign-up. To login, user simply needs to type in the correct email address and password combination and click the login button.

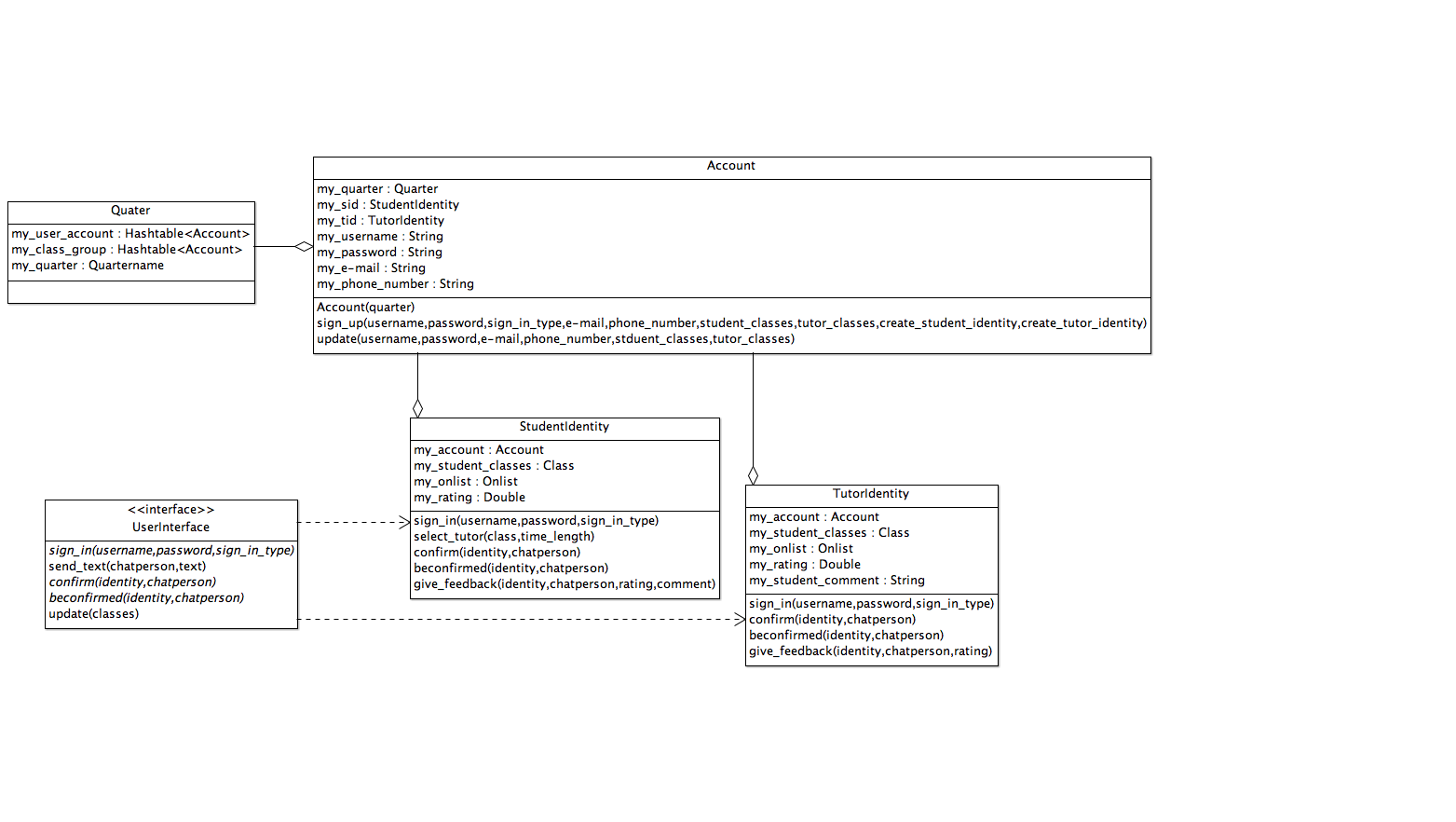
User Functionalities (Student & Tutor)

For the student user, the main two functionalities we provide are to update personal profile and find a tutor. Specifically, upon succesful registration and login, student user could update his/her personal profile, which includes account type, email address, real name, phone number, password, and credit/debit card number. Student user could find a tutor by first clicking the “find a tutor” button located on the homepage, and then selecting his/her preferences in the following pages, which include the course subject, location of the tutorial, duration. After filtering different conditions, student will either be provided with a tutor based on his preference, or notified that currently no tutor available for his selection. In the former scenario where tutor is available, student user then needs to confirm the tutor and proceed to the payment page. After sucessful payment student user will be directed to the confirmation page, in which there is a instant chat box enabling the student to further discuss the details of the tutorial session. If however, no tutor is currently available, the student will be given the option to be placed on a waitlist for that course. If student decides to stay on the waitlist, he/she will be notified once a tutor becomes available. After succesful payment, student user will be able to rate the tutor and write additional comments regarding the tutorial session.

For tutor user, the two main functionalities we provide are to update personal profile and declare availability. Upon succesful registration and login, tutor will be able to update his/her personal information which is identical to that of the student user. Tutor user is able to delcare his/her availability by clicking the sliding button, of which the default value is “Not Available”. After the button value becomes “Available”, tutor user will be able to further choose the length of his/her available time slot and confirm his/her choice by clicking “Set” button. If tutor user wants to change the length of the time slot, he/she can do it by clicking the “Reset” button and choose a new time length. After sucessful declaring availability, tutor user will receive a notification email containing the information regarding his next tutorial session, and instant chatbox set for the tutor and student.

**V. General Design and API**

**Back-end:** (Javascript)

Below is a UML class diagram that shows data structure in NoSQL database provided by firebase and basic object relations.

API:

//input: identity of either student or tutor, email, password, username, credit card number, //output: void

//function: generate a unique user id and create a user field in database.

function register

//input: email and password, output: void

//function: log user in across browser pages

function emailLogin

//input: unique user id, output: user name

//function: find user name by giving user id and output user name onto the webpage

function idToUsername

//input: location as a string, output: google map location

//function: transform a string location to a google map format location

function locateStudent

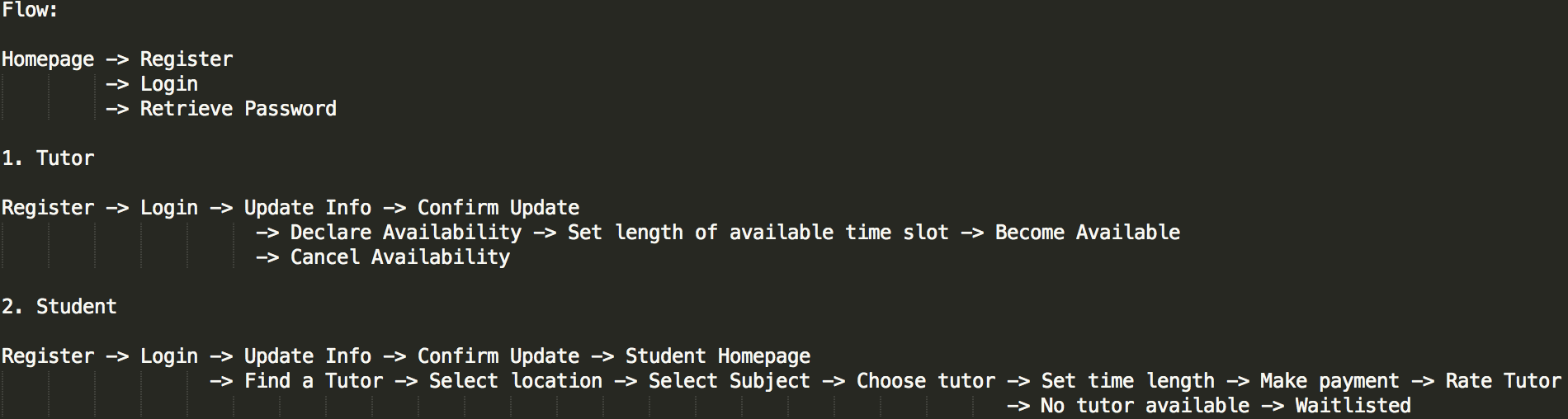
//input: message as a string, output: message and history message on html and in //database

//function: store chat history into database and only allow the two users who share this //chatbox add chat history to this chatbox data field.

function createChatbox

**Front-end:**

The flow of the application interface is as follows:

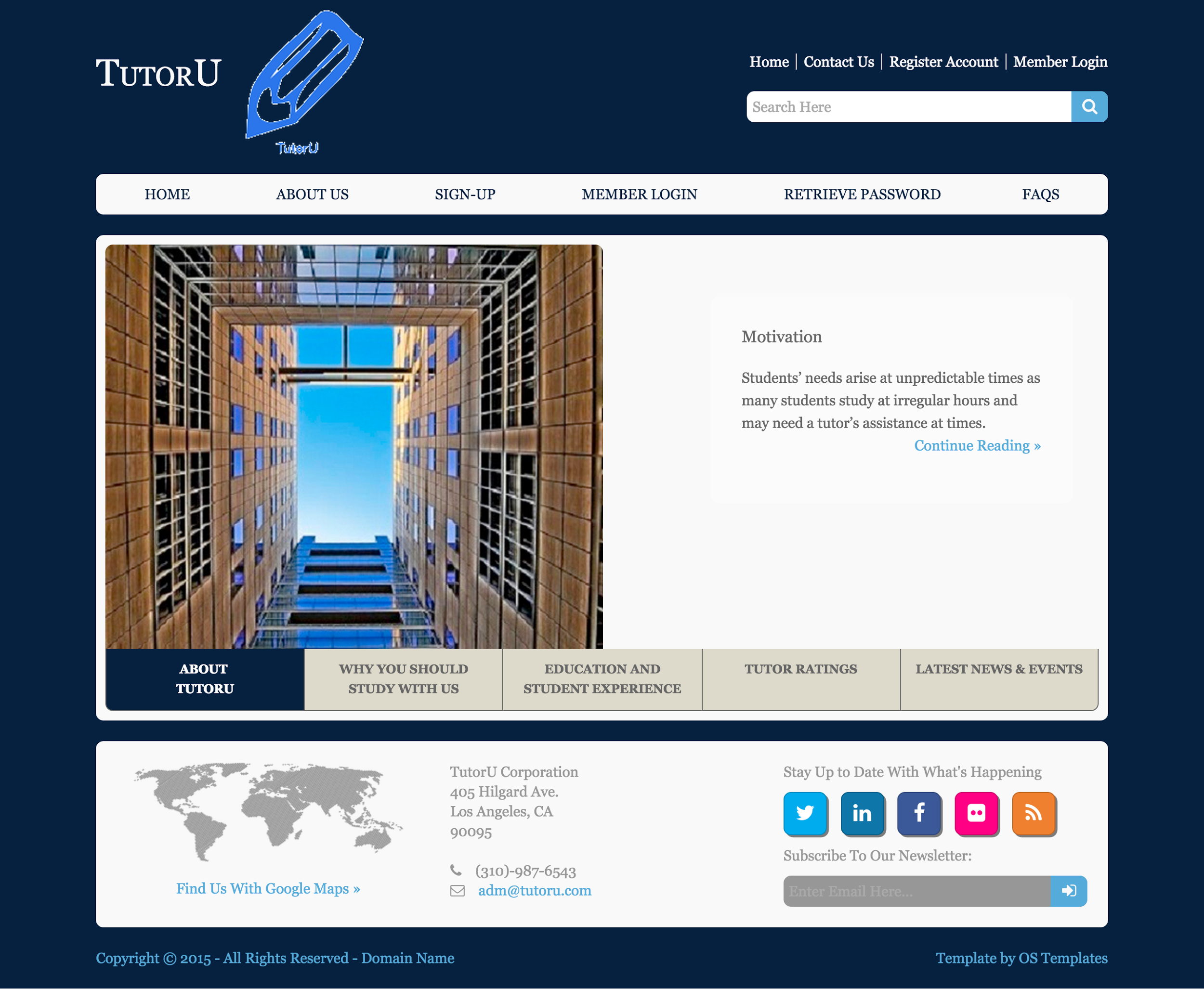


For the front-end design, we mainly rely on the existing html template and CSS to make the user interface simple and smooth. The full list of template and styles we used in this project is included in the end of the report. Two of the many APIs we used for the user interface are the BonBon buttons used in most pages, and Sliding button used in the tutor page’s availability button, which plays a iPhone-style sliding effect.

**User Interface Snapshot**

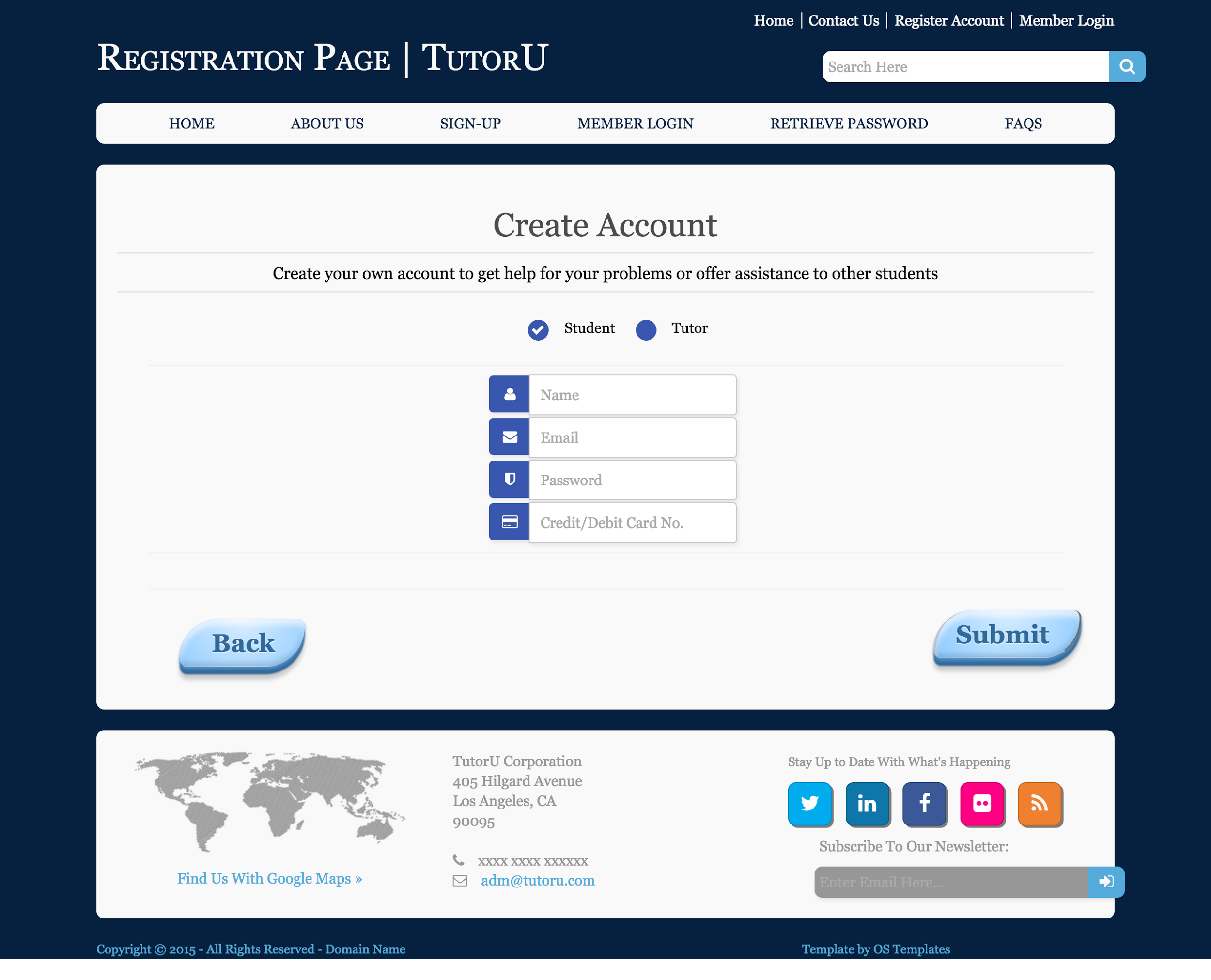
Homepage:

Homepage enables any incoming guests to create an account or sign in.



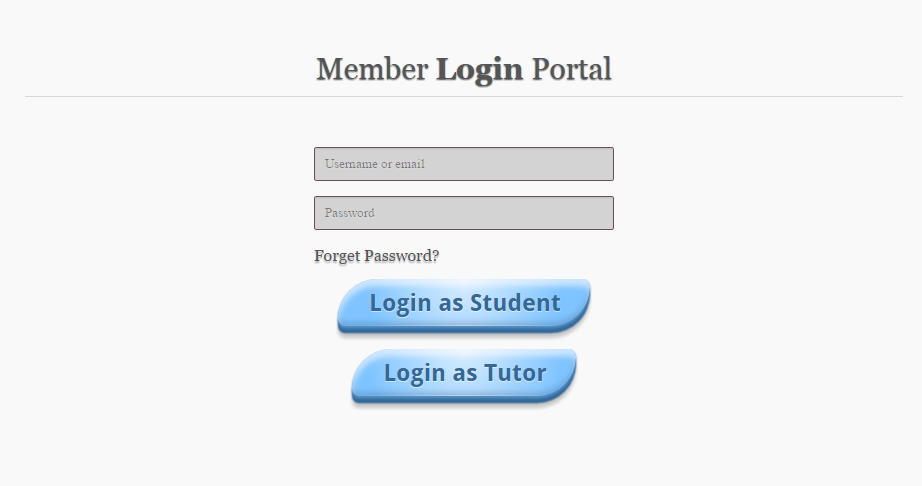
Register:

Registration page allows guests to create their account based on their needs. The student account requires basic information including name, email, password, and payment method. The tutor account requires extra information including the subjects user wants to teach.



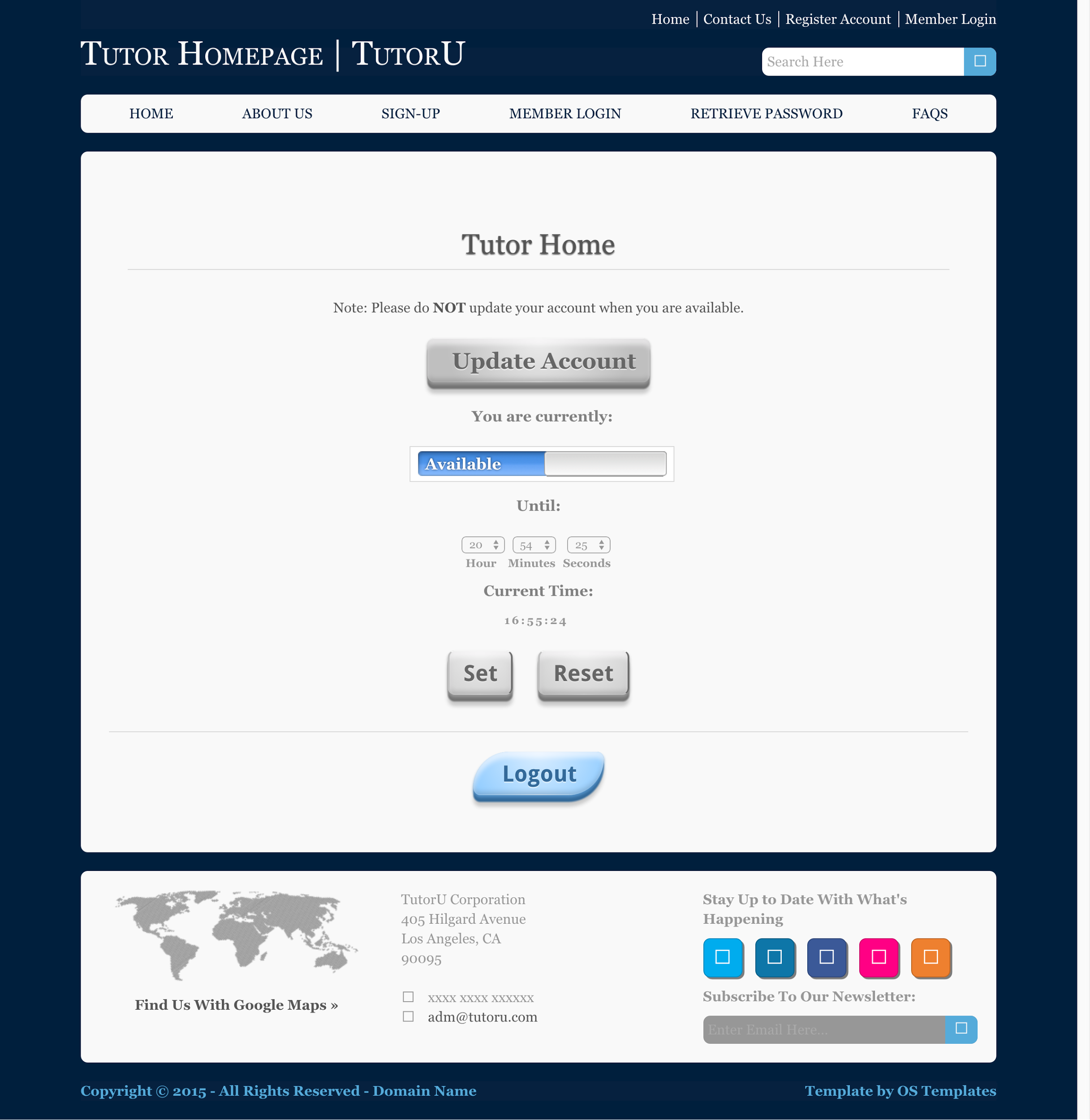
Login:

Login page enables both types of user to quick login by providing the correct email address and password.



Tutor’s portal:

Tutor’s portal is very simple. It allows users to update account information and set their availability. Tutor can set/reset the length of their available time slot using the clock below. After declaring availability, tutor will receive notification email if a student orders a tutorial session with him/her.



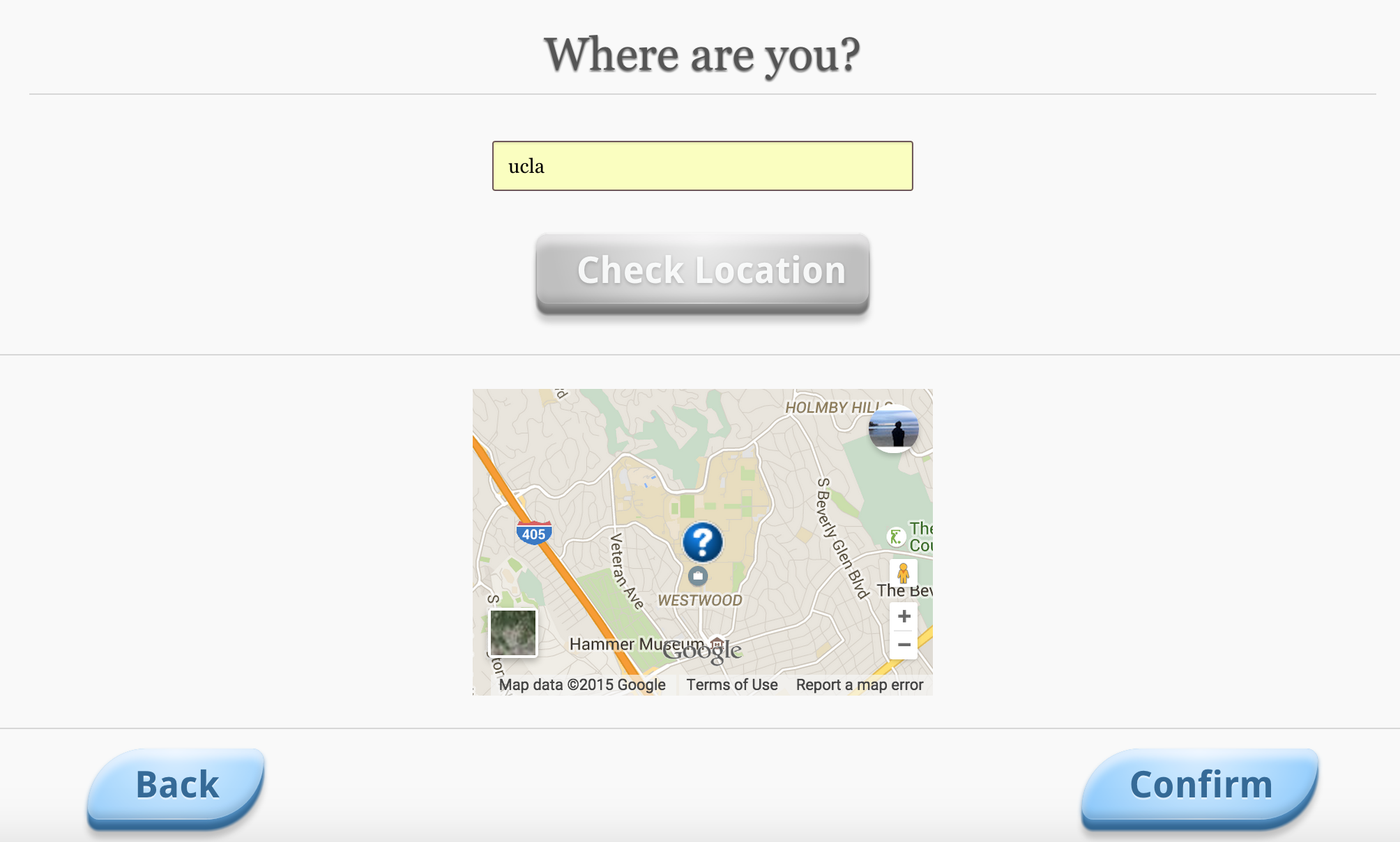
Student Portal:

Student user is allowed to update account information or click find tutor button, which will direct user to a new process including selecting tutor and making payment.



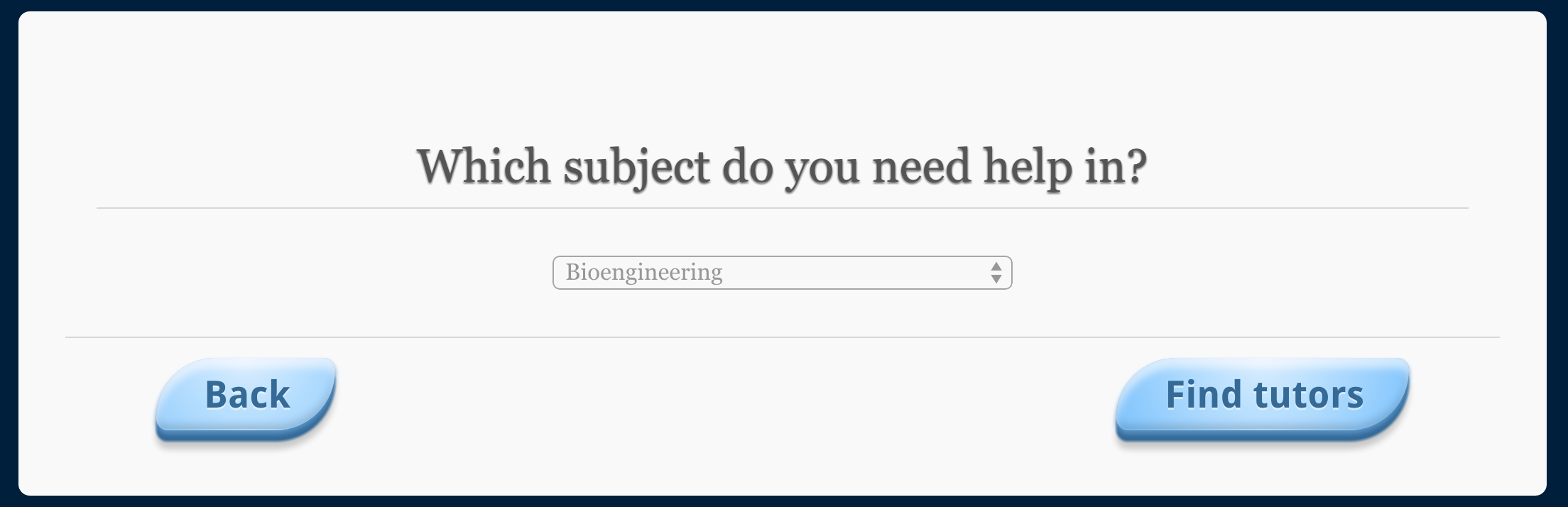
Select Location:

After clicking “find a tutor” button on the home portal, student user will then select the preferred location for the tutorial session. The Google Maps API is used in this page, helping user to better search and locate their preferred place.

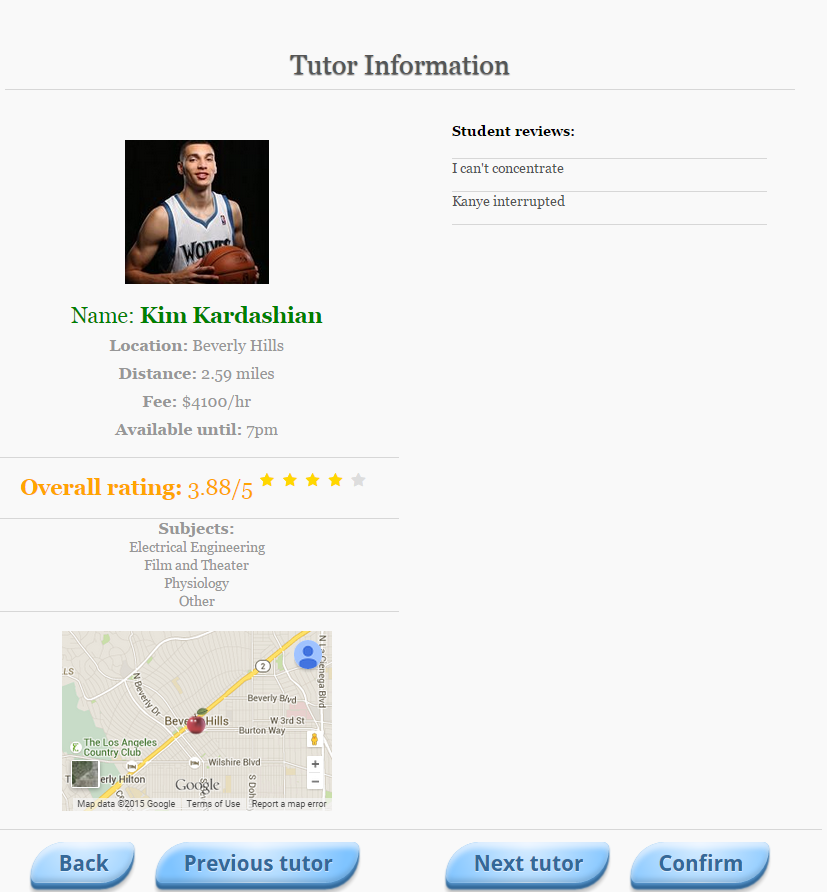


Select Subject:

After choosing location, student user then needs to select the course subject to be discussed/taught in the tutorial session.

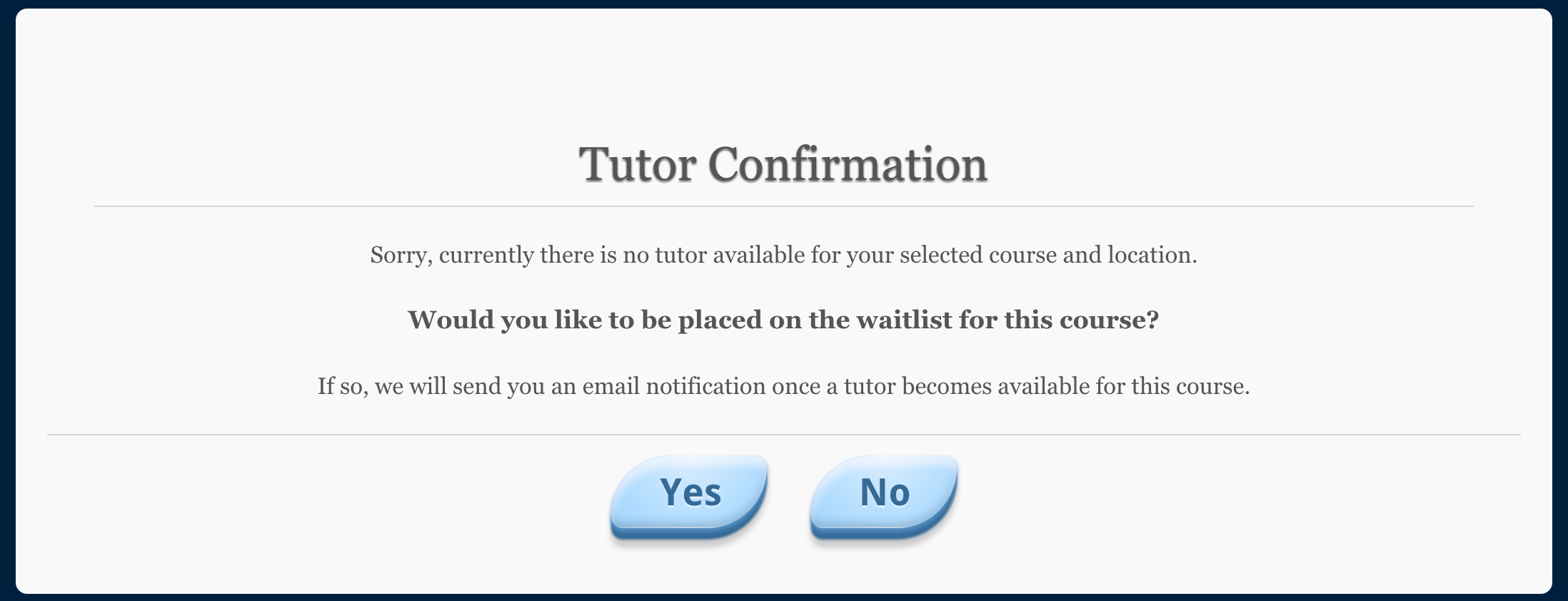


Confirm tutor:

After selecting appointment location and course subject, the student will then be able to select the tutor, if there is one or more available candidates currently declared as available to help. User can navigate through previous or next buttons. Tutor’s profile will be displayed as a combination of personal information, historical rating, and student comments. Student will then select the best fit based on the given information.

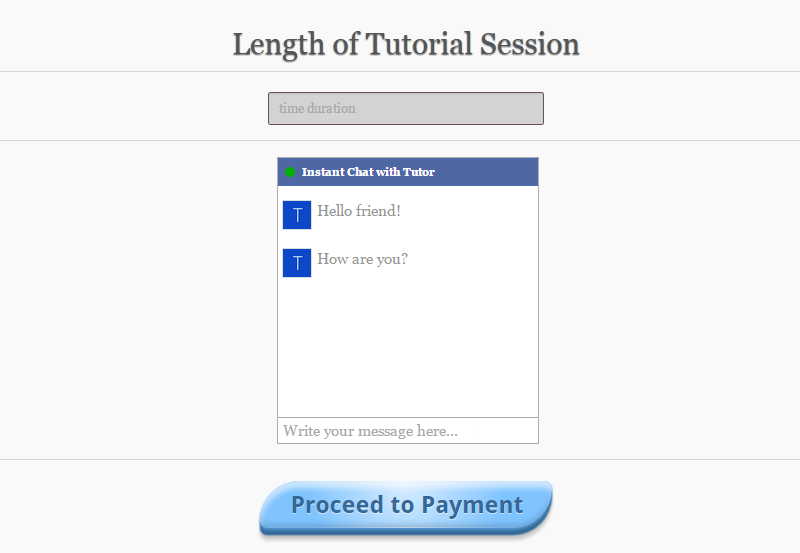
No tutor available:

If there is no tutor available, system will notify the user and offer the option of placing on the waitlist for that course. If student chooses so, he/she will be notified via email once a tutor becomes available for that course.



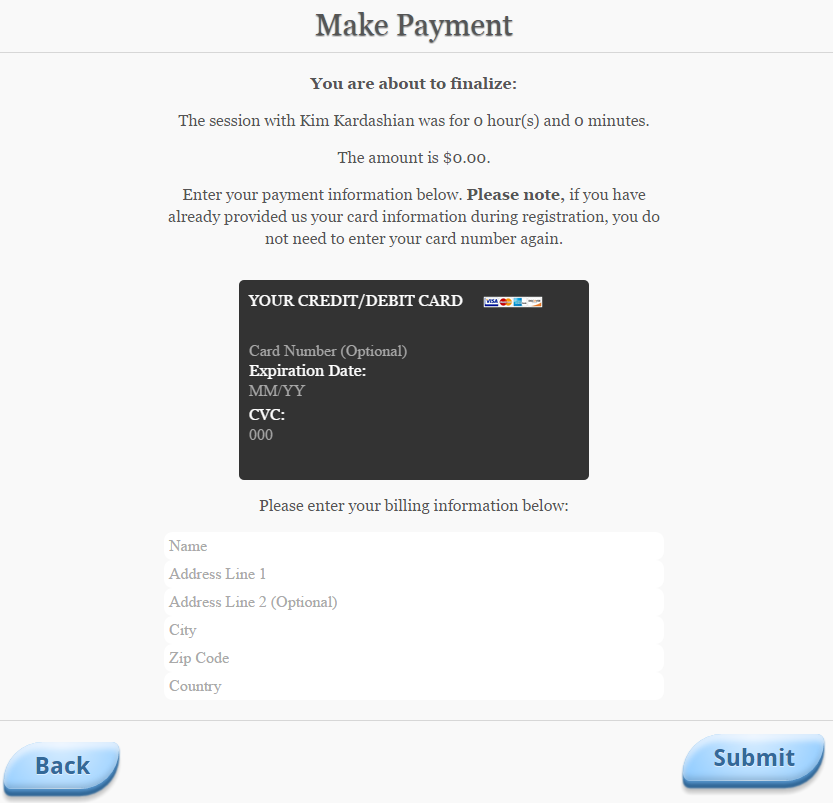
Session Duration and Chat:

After confirming the tutor, students will proceed to the chatting page. A student can input the duration of the seesion he/she desires and also chat with the tutor.



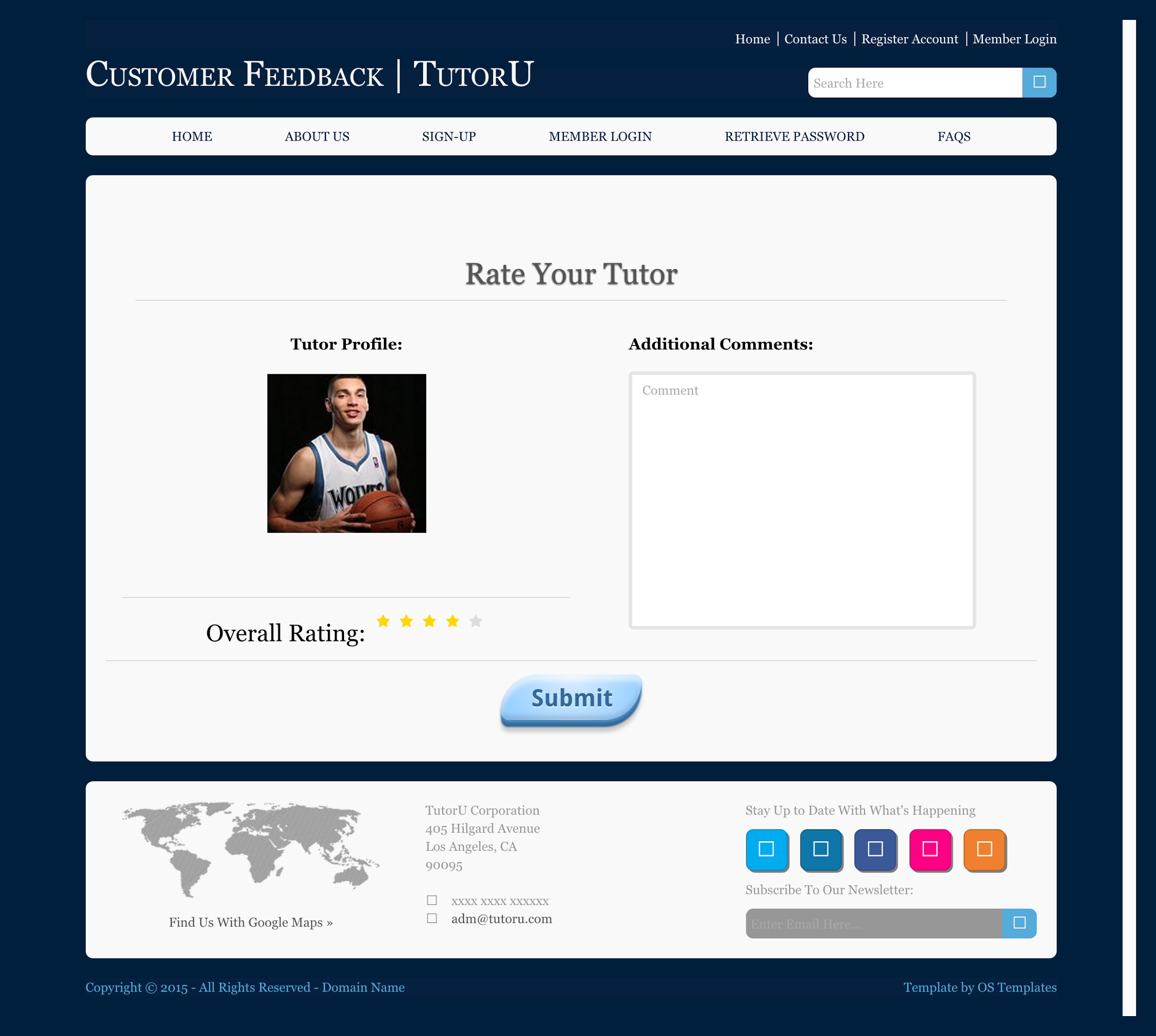
Payment:

After confirming the session duraiton, student will then proceed to the payment page. User needs to provide valid credit/debit card information to finish the payment.



Rate tutor:

After the payment, student will be given the chance to rate the tutor and add comments. Currently we only support overall rating.



The above process summarizes the flow for both tutor user and student user from front-end perspective.

**V. Test Scenarios (statement test)**

Register Page & Login Page:

Input: empty email address, email address with no @ and .com, empty password,

Output: Javascript alert on the problems.

Input: valid email address and password

Output: Register Page leads the user goes back to home page. The user can then go to Login page to login. Login Page leads the user to student/tutor portal page. If the user goes back to Login page again, there will be an alert telling the user that the user has already logged in, and a “Hello <username>” will be shown on the header of the page.

Select Location page:

Input: input empty location

Output: Javascript alert on the problems.

Input: nonempty location

Output: Direct to the SelectSubject, and save the user input data into users’ database fields.

Select Subject page:

Input: select a subject

Output: Save the subject data into the database’s tutor request field of this student. Match the availbe tutors who teach the student’s requrested subject and is currently within 15 miles of the location that the student specifies. Fetch the information of the tutor who has the highest rating and present’s tutor’s information in the confirmation page. If no tutor is currently available, direct to No tutor available page.

Chat page:

Input: messages

Output: new message and history messages of both students and tutors’ are in the chatbox chronically.

Select Payment page:

Input: null

Output: the page shows the amount of money to pay.

**VI. Design Patterns**

Mediator:

Each user has a unique uid shared by their identities, student identity, tutor identity, or both as userObject stored in the database. Only students authorized with the uid can ask chatbox object to create a chat session between themselves and their requested tutor. The chatbox object then modify the its value and thus grant the permission to the requested tutor to send message to the chatbox. Once the chatbox recieves the message from the tutor, the chatbox will modify its value and thus show tutor’s message to the students.

Facade:

For the tutor user, we took in the concept of Facade pattern, which improves the user experience from tutor’s perspective. In the tutor’s homepage, the only thing tutor needs to do when he/she wants to offer help to others is to click the “availablility” button, and the “subsystem” will then be activated and start to assign possible student users to the tutor user. The entire process is done by the backend code. The Facade pattern is extremely useful in the user account implementation, because it largely simplifies the interface design and delivers fluent user experience.

**VII. Team Contribution**

Austin and Youshan are responsible for the backend implementation. Austin implemented registration; pinpoint location on google map and caculate the distance between student and tutors; match student and tutors’ subject; sort available tutors by rating and output the tutors with the highest ratings; caculate payment by multiplying the input session time and tutor’s per hour wages; create dummy database. Youshan implemented login and show user name. Bolun is mainly responsible for the front-end design and implementation such as the chatbox and its message, sliding button, and retriving current time etc. , and part of the front-end back-end code integration. Zhiping is mainly responsible for testing and video presentation.

**Youtube:**

**(**[**https://youtu.be/dooUA-aCJOc**](https://youtu.be/dooUA-aCJOc)**)**

**Github:**

**(**[**https://github.com/HelenRouty/firebase\_function\_practice.git**](https://github.com/HelenRouty/firebase_function_practice.git)**)**

**Firebase:**

**(**[**https://tutoru-v1.firebaseapp.com**](https://tutoru-v1.firebaseapp.com)**)**

**Reference List**

http://www.dillerdesign.com/css/cookbook/component\_star\_rater.html

http://css3playground.com/flip-card.php http://www.tomasdostal.com/projects/css3-creditcard/

http://codepen.io/afirulaf/pen/djAen

http://codepen.io/jamesbarnett/pen/vlpkh

http://tympanus.net/codrops/2012/10/16/custom-login-form-styling/

http://cssdeck.com/labs/bonbon-buttons

http://cssdeck.com/labs/simple-css3-comment-form

http://jsfiddle.net/QAaHP/12/

http://ios-checkboxes.awardwinningfjords.com/