CS 307 -- Software Engineering

ApptiTutor

Design Document

Team 12

Daniel Cruz, Himal King, Nick Schumacher, Josh Stuckey, Michelle Young

Table of Contents

|  |  |
| --- | --- |
| Topic | Page |
| Purpose | 3 |
| Design Outline | 3 |
| Design Issues | 5 |
| Design Details | 8 |
| Sequence Diagrams | 9 |

1. **Purpose**

While going to school full time, classes can get tough and some students require additional tutoring. Purdue University does not offer a system dedicated to students finding tutors and tutors offering their services for specific classes. While bulletin boards with flyers offering tutoring services exist throughout almost every building, many students ignore or do not notice these postings. Our system will offer a quick and reliable way for students and tutors to connect via an intuitive web application from anywhere, at anytime.

1. **Design Outline**

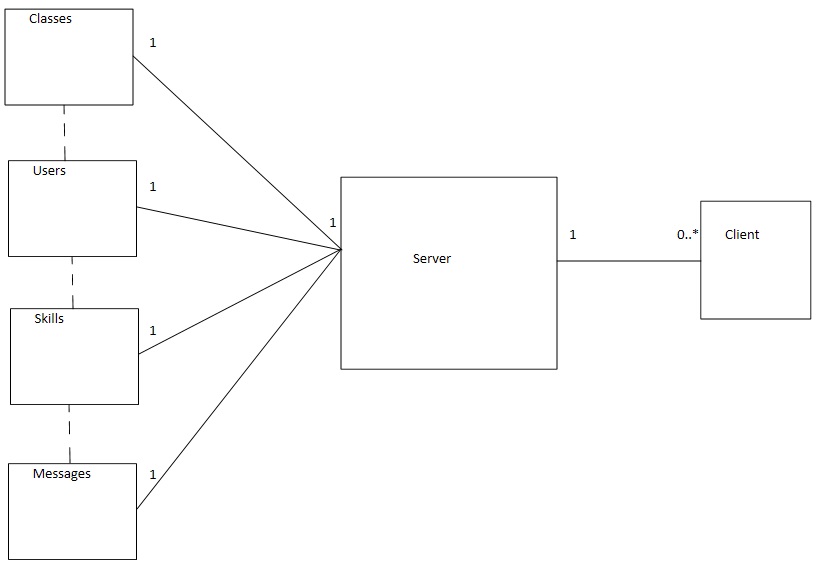
Our product will be a web based application that will allow tutors to create a profile and then students can search for tutors based off of skills and classes they tutor. We will be implementing the Client-Server model along with a database where Users will be either making their tutor profile, talking to students/tutors or looking up tutors. We will use a Relational database to link up the three tables in the database that we will need. We will also have a database to track messages sent between tutors and students.

Components

1. Client
   1. The web page will allow students to choose between whether they want to be a student or a tutor.
   2. The web page will have a calendar to look at when the user is free and when they have meetings.
   3. The web page will have a search bar with the ability to search for tutors.
   4. The Client will make calls to the server to get information it wants.
2. Server
   1. The server will process and return values to the Client as it asks for information from the database.
   2. The server will create the code for the website to show as the user interface.
   3. The server will take the Client requests and create database calls from them.
   4. All sorting and calculating will be done on the server.
3. Database
   1. The database will store the data for Users (username, password, email, etc.), classes (class name, tutors for classes), skills (skill name) for the project and for the messaging service.
   2. The database will be updated as needed. For example, every semester the class catalog changes, so the database will need to be updated with the latest class catalog. Other components of the database will not need to be updated frequently such as user names and email (unless there is an email change request). The messaging database will be updated with messages as they are sent.

Figure 2.1 - Overarching System Design

We will have many browser windows that are connected to our server. They will make requests to our server to either send or receive information that they would need. The server will then interpret these requests and call the necessary functions to get the data. This will require making calls to the database to get information. Once the server gets the information it needs together, it will send the data back to the browser.



1. **Design Issues**

Issue 1: Platform

i. Android Application

ii. iOS Application

iii. Web Application

**Choice:** Web Application

Android and iOS applications were appealing because of their popularity with the general public for mobile apps on smartphones. However, we decided to go with a web application instead because it will be able to be loaded in any web browser, including those on smart phones of any operating system, so we wouldn’t have to pick and choose between Apple and Android phones.

Issue 2: Language and Framework

i. Angular.js (Javascript)

ii. Ruby on Rails (Ruby)

iii. Django (Python)

**Choice:** Ruby on Rails

When we were looking at frameworks we knew that there were many potential frameworks that we could choose to implement. Angular.js and Ruby on Rails seemed to us to be the two most popular ones at the moment so we were able to settle it down to those two. We ultimately decided to go with Ruby on Rails because the language was simpler and many big companies today use it as well.

Issue 3: Database

i. mySQL

ii. PostgreSQL

iii. MongoDB

**Choice:** mySQL

We chose to go with mySQL over the other two because no one on our team had experience in the other two languages. Since we were learning Ruby on Rails with some HTML/CSS for the front end we decided it would be best to go with a database language that was at least somewhat familiar to us.

Issue 4: Designing the User Class

i. Have different classes for Tutors and Students so each is its own login

ii.  Have a User class with a “choose one or more” option with Tutors and Students

iii. Make everyone default to Student and they can add Tutor if they want

**Choice:** Have a User class with a “choose one or more” option with Tutors and

Students.

We looked at having different classes and logins but we decided that would be too complicated for the user to have two logins if they were a student and a tutor. We also looked at having everyone be a student and they could be a tutor as well if they wanted. We did not like that because maybe a tutor just wants to be a tutor. It would add more complications for them to see the student tab when they don’t want to be considered a student. It also would have caused confusion within our code because we would have code that both students and tutors would have along with code for just students in the User and then the Tutor code would be by itself. So we decided to go with the main User class with there being both a Student and Tutor class within it. This would allow Users to be one or both of the sub classes we have and the code will be more organized.

Issue 5: How do we incorporate the functionality for a student and a tutor in one account?

i. A tab for students and a tab for tutors

ii. A switch that changes between tutor and students

**Choice:** A tab for students and a tab for tutors

We decided to have a tab for the student side of the application and a separate tab for the tutor side because it will allow users to switch between the functionalities of both quickly. Also, a student that isn’t a tutor would have the tutor tab hidden.

Issue 6: How do we implement messaging and what kind of messaging?

i. Instant Messenger Built in

ii. Email

iii. Built in Email like messaging service

**Choice:** Built in Email like messaging service

We chose to go with a Built in email like messaging service. We thought that implementing an IM service would be difficult and we did not have the skills and time to do that this semester. We also thought that Email would be weird because we want the students and tutors to be able to connect together on our website. Also tutors or students may be weary about just handing out their contact information to people they do not know well. We wanted to offer a more secure option for students and tutors. That is why we decided on having an email like messaging service built into our app, it is not as complicated technically for us and it is more secure than just handing out your contact information to people you do not know well.

Issue 7: Threads vs. Processes

i. Use Threads to run every part of the system

ii. Use Processes to run every part of the system

iii. Use Both Processes and Threads

**Choice:** We will be using both threads and processes.

We want to use processes for each user that connects to the server for the sake of robustness. This will prevent our whole server from crashing if there is an error for one user. We will still use threading inside each process to be able to handle multiple tasks at a time such as rating a tutor and receiving notifications about messages.

Issue 8: Skill option list design

i. Use a database of predefined skills

ii. Allow tutors to write custom skills

iii. Do not allow tutors to add skills to their profile

**Choice:** Use a database of predefined skills

We decided to allow tutors to display their skills to add depth to their profile so that students can see what the tutor is exceptional at. We will be using predefined skills for our database so that there is uniformity across the app and it will allow for an easily searchable index.

**4. Design Details**

Figure 4.1 Class Diagram

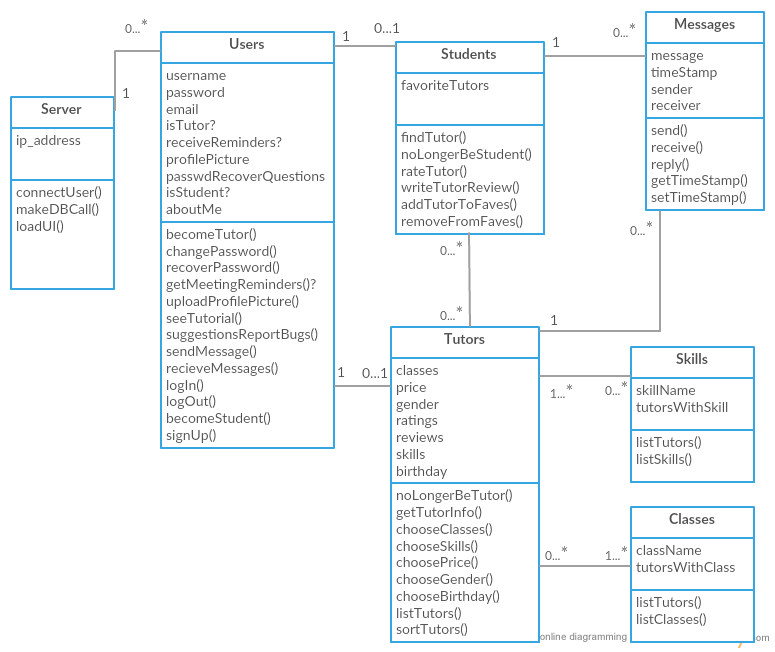


Figure 4.2 Message Diagram - How a basic messaging conversation will interact with our system.

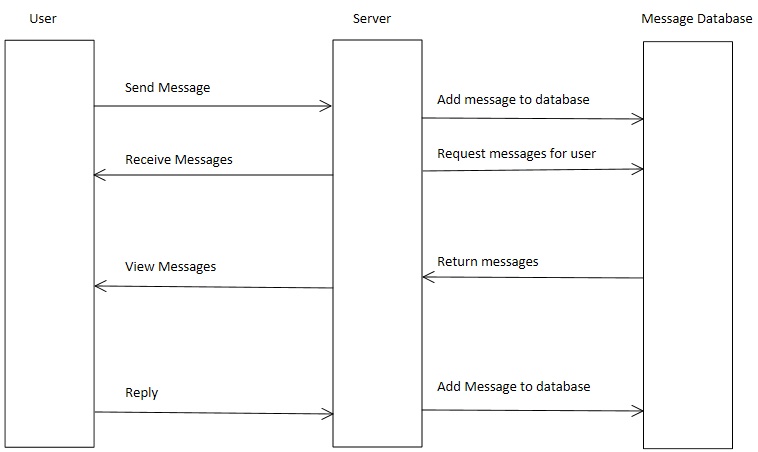


Figure 4.3 Tutor Sequence - The sequence of events that a user will go through to become a tutor.

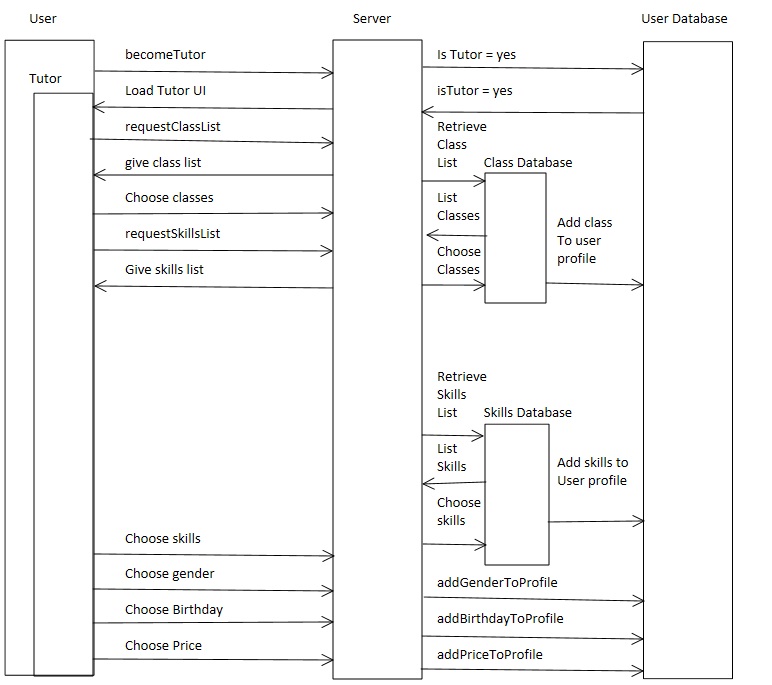


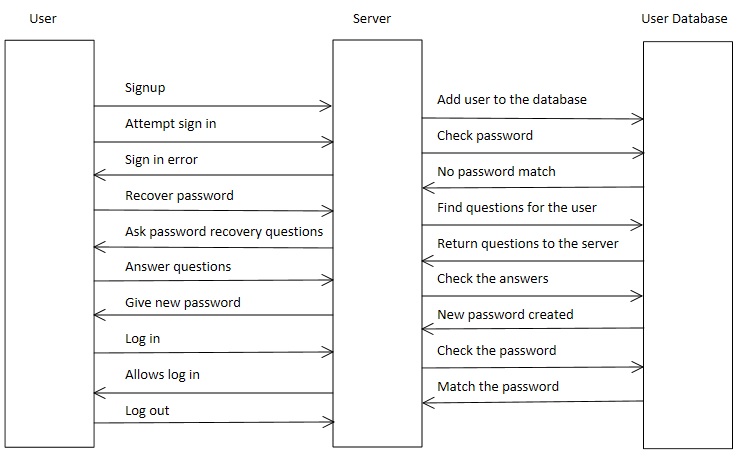
Figure 4.4 Password Recovery Sequence - This is the sequence of events that’ll take place when the user has forgotten his/her password.

Figure 4.5 Student Diagram - A sequence of events that a student goes through to become a student and search for tutors.

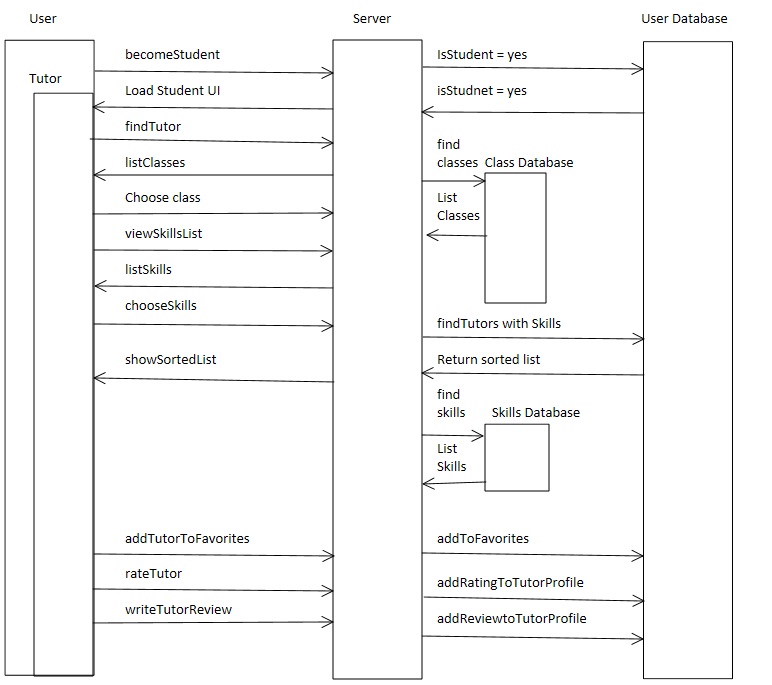
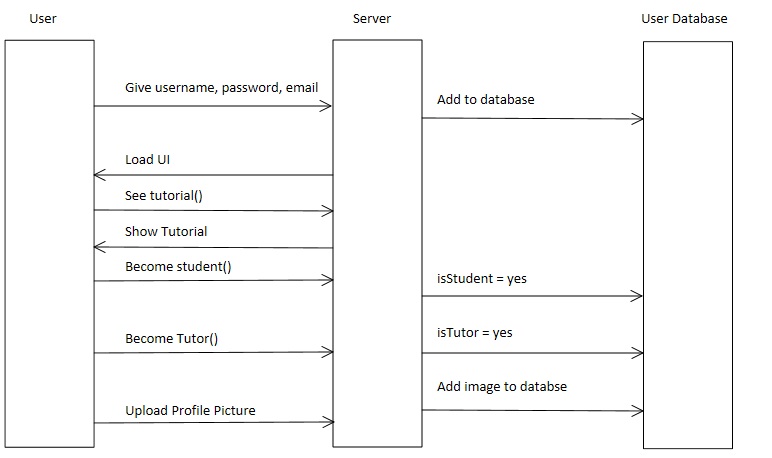


Figure 4.6 Signup Sequence - This is the sequence the user will go through when creating an account. Once the username and password are chosen, they are added to a database. A tutorial will then be loaded to assist the user with navigating the site.

User Interface - Basic UI Mockups

