EDUYA

Team 13 - Design Document

Tomasz Parzadka, Hans Allendorfer, Weiyi Zhou, Puja Mittal, Ashwin Prasad

TABLE OF CONTENTS

PURPOSE

Summary

A university student's academic success is dependent on their ability to synthesize and process information learned in lecture and lab. Tutors, professor office hours, study groups, and other academic resources can greatly assist a student in achieving this goal. However, students are often forced to comb through numerous websites and scattered posters in order to determine the location and times of these resources. Even then, it's not always easy to identify how much help they can be with the students' specific problems. Therefore, we aim to centralize this information on a single platform with an emphasis on convenience. Our project aims to connect students with tutors and academic resources that will help them in their academics endeavours.

Functional Requirements

Our functional requirements include the following items:

1. **Account**
    a. Register for an account
    b. Log into my account
    c. Reset my password if forgotten
    d. Edit my profile (e.g. add a picture, contact info, personal info, etc.)
    e. Opt to be a tutor and have a public profile with reviews and information
2. **Students**
    a. See a list of available tutors sorted by price, class, rating, etc.
    b. See the contact information for the tutors listed on the platform
    c. Leave reviews for tutors including ratings, pricing, and other comments
    d. View all tutors, office hours, or SI sessions
    e. Add, remove, and view my related courses
    f. Add, remove, and view my related professors

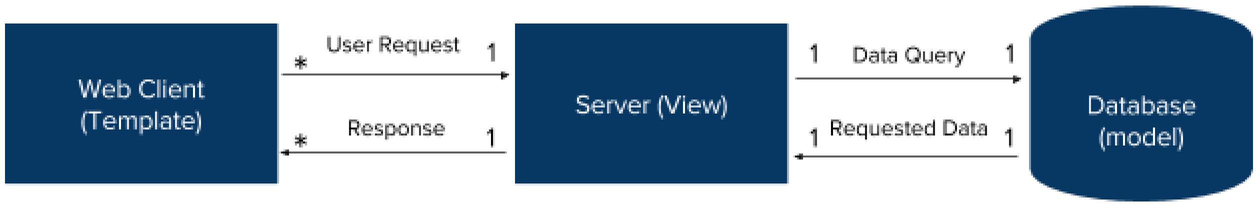
   g. Create a posting looking for a tutor with specified pricing and specified time slots

3. **Tutors**
   a. Create a posting looking to tutor others with specified pricings and specified available time slots
   b. Specify which classes/subjects I am willing to tutor
4. **Professors**
   a. View professor reviews and ratings from RateMyProfessor
   b. Comment on courses/professors, so future students enrolled in that class/with that professor can see a review
5. **Courses**
   a. View course related information (e.g. professor names)
   b. View available resources for the course
   c. View previous course exams and lecture notes if publicly available
   d. Allow students to see required materials for the course.

DESIGN OUTLINE

High Level Overview

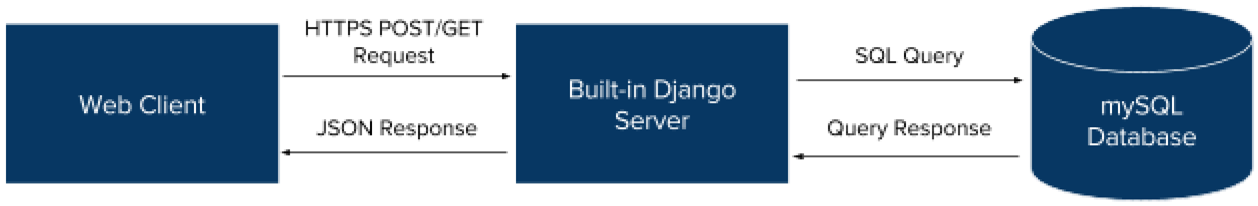
Our project utilizes the Model-View-Controller architectural model. We chose the Python-based Django to serve as our web framework, and thus we will use Django's terminology for the MVC model, Model-View-Template (MVT), which calls view "template" and calls controller "view". In order to allow many users to access our platform simultaneously, we use a many-to-one client-server model, with many web clients (templates) communicating with a single server (view). This clients send user requests and input to the server, which processes and validates the request before querying the database (model) to either store or retrieve data. This data is then sent back to the client, which renders it to be displayed to the user. This process is displayed in the diagram below.



Component Overview

Our system consists of three primary components: the web client, the server (built into the Django framework), and the database (mySQL). The web client will send the server a HTTPS GET/POST request, depending on whether or not the user is submitting information, such as a review. Any data will be included in a POST request in JSON format. The server will then receive and process the request, performing any requisite validation and parsing. If required, server will then construct a SQL statement to query the MySQL database for the required data. Using this data, the server will construct as response, including data in JSON format, and send it back to the requesting web client. Finally, the client displays and renders the new information to the user using the React framework.

1. Web Client (Template)
    a. Sends requests to server API using the JSON format
    b. Receives JSON responses
    c. Renders responses using the REACT framework
2. Built-in Django Server (View)
    a. Receives client JSON responses and checks for validation
    b. Processes responses and queries MySQL database if necessary
    c. Generates and sends JSON responses with requested information back to client
    d. Run a daily cron job that validates site models by scraping data from data sources used (e.g Purdue websites) and update models accordingly.
3. MySQL Database (Model)
    a. Maintains record of all data from users and scraping
    b. Responds to queries from server
    c. Stores/updates user submitted data



DESIGN ISSUES

Issue 1: What type of UI design scheme will be utilized?

a) Adaptive

b) Responsive

Discussion: We decided on responsive design because adaptive design would not give enough flexibility. Using adaptive design runs the risk of improper rendering of elements for users viewing on an irregularly size device. Although responsive design is more difficult to implement, we felt that the tradeoff was worth it for the assurance of a consistent viewing experience, regardless of device.

Issue 2: What language/framework will we use for our frontend?

a) React

b) Raw Javascript/jQuery

c) AngularJS

Discussion: We decided on React as our frontend framework as it has better performance, due to it's "virtual DOM" implementation. This allows React to only rerender DOM elements that have been changed instead of the entire DOM. Although AngularJS is more fully featured than React, many of its additional features are superfluous to our project, and thus we determined that it wasn't worth the tradeoff for decreased performance. We also decided against using raw JS/jQuery because it offers no advantages over the modularity offered by using a framework, allowing easier integration with other components.

Issue 3: What language/framework will we use for our backend?

a) Python (Django)

b) Java

c) Python (Pyramid)

Discussion: All of our team was pretty familiar with programming in Java through classwork, but few of us had worked with it in the context of web development. In addition, the two Python frameworks seemed easier to implement while still being fully fledged in terms of features, especially in terms of web-scraping, where Python far outclasses Java. Between the two Python frameworks, we decided on Django, because it is largely pre-configured, saving us time on decision that will ultimately have minimal impact on our project. In addition, Django had a much larger supporting community in terms of resources and documentation to help familiarize us with the framework.

Issue 4: What type of database should we use?

a) SQLite

b) mySQL

c) PostgreSQL

d) NoSQL (i.e. MongoDB)

Discussion: We briefly considered a NoSQL database, but quickly discarded the idea based on the fact that none of us had any experience with non-relational databases. We also discarded SQLite, because of its lack of support for multiple concurrent write operations, a fatal flaw for multi-user databases like ours. Between PostgreSQL and mySQL, we ultimately settled on mySQL for its easier use and faster speed. Although PostgreSQL has powerful analytical and data integrity functionality, ultimately, we decided these features were overkill for our needs.

Issue 5: Should account creation be required for users?

a) No account creation - Purdue credentials must be entered every time to access sensitive information

b) Account creation - Require a unique username and an accompanying password

c) Account creation & linking - Allow user to login using their Facebook, Google+, etc.

Discussion: Ultimately, we decided on option B. Option A required the least amount of work. However, it would be extremely inconvenient for users to have to enter their information every time they wanted to access personal information. In addition, it would be impossible to maintain reviews on tutors, and would decrease the barrier of entry against bots/fake users. On the other hand, we felt that the additional work we would need to put in to enable user to login using 3rd party accounts was not worth the minimal amount of convenience gain for our users. Thus, we decided on option B as a good compromise between the two extremes.

Issue 6: How will we connect users and tutors in a manner that ensures mutual satisfaction?

a) Allow tutors to create an offer post specifying subject/class, prices, availability, etc.

b) Allow students to create a post specifying what kind of tutor they are looking for

Discussion: We decided to include both features, because we want both tutors and tutees to engage with our platform. We want to provide a platform for tutors to offer their services to students. In addition, we also want to implement data scraping in order to include already existing tutoring services from other sources, such as school web pages. However, we felt that including a way for students to make request posts would encourage more user to take part in tutoring by providing a ready made offer that a potential tutor could simply accept if they felt that it fit their requirements.

Issue 7: Should we include an platform-native rating system for professors' ratings?

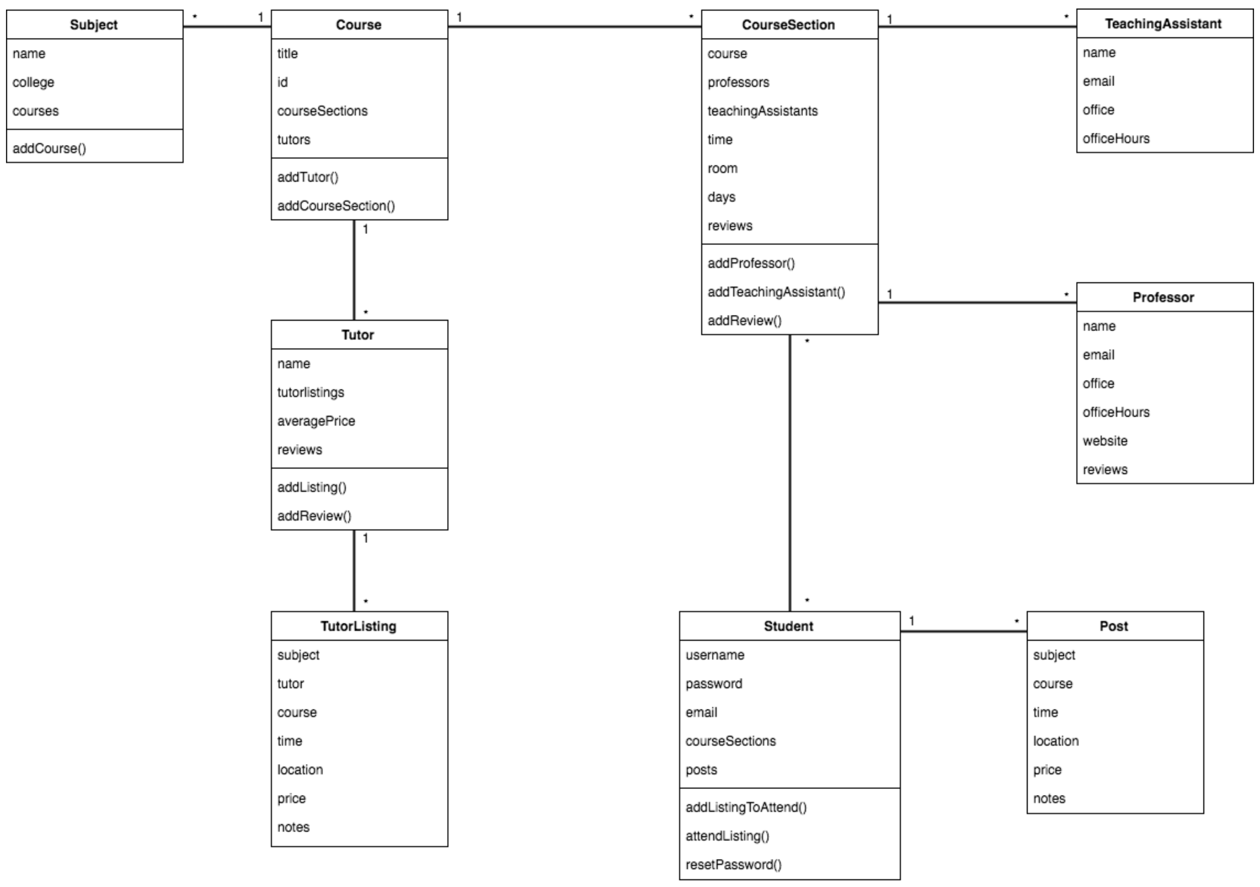
a) Allow users to rate a professor through the platform

b) Only display reviews, direct users to submit reviews through RateMyProfessor

Discussion: Although we could easily incorporate a rating system on our platform independent of RateMyProfessor, it would be redundant to do so. RateMyProfessor is an already established service focused on rating professors nationwide. We plan on allowing students comment on professors on our site, but we felt it would be more practical to utilize an already established service in this area.

DESIGN DETAILS

Class Diagram



Subject

- Contains attributes:
  - *name* - name of the subject (major)
  - *college* - college in which the major belongs
  - *courses* - list of Course objects

Course - Belongs to a Subject object

- Contains attributes:
    - *title* - specific course title
    - *Id* - course number
    - *courseSections* - list of CourseSection objects.

CourseSection - Belongs to a Course object.

- Contains attributes that characterize the course section, specifically:
    - *course* - course the course section belongs to
    - *professors* - list of professors that teach the course section
    - *teachingAssistants* - list of teaching assistants associated with the course section
    - *time* - time of the course section
    - *days* - list of days the course section is in session during the week
    - *reviews* attribute - list of *Review* objects related to the course section

TeachingAssistant - Belongs to a *CourseSection* object.

- Contains attributes that summarize teaching assistant information, specifically:
    - *name* - name of the teaching assistant
    - *email* - teaching assistant's purdue email
    - *office* - name and number of the teaching assistant's office if applicable
    - *officeHours* - list of their office hours

Professor - Belongs to a *CourseSection* object.

- Contains attributes that summarize professor information, specifically:
    - *name, email, office* - office location
    - *officeHours* - list of times of office hours
    - *website* - professor website, if applicable
    - *reviews* - list of text reviews retrieved from ratemyprofessor.

Student

- Contains attributes related to the student account, specifically *Username, password, and emailAddress*.
- Contains attributes related to the student itself, specifically:
    - *courseSections* - list of course sections the student is registered in
    - *posts* - list of *Post* objects posted by the student looking for tutors
- Contains a *tutor* attribute referring to a *Tutor* object if the student also decides to offer tutoring services.

Tutor

- Contains attributes that summarize tutor information, specifically:
    - *name* - name of the tutor
    - *listings* - list of *Listing* objects posted by the Tutor looking for students to tutor
    - *averagePrice* - price average of the tutor's listings
    - *reviews* - list of *Review* objects related to the tutor

TutorListing - Belongs to a Tutor Object (an attribute of Student)

- Contains attributes that summarize tutor listing information, specifically:
    - *subject* and *course* - course information
    - *tutor* - tutor associated with the listing
    - *location, time, price,* and *notes* - information related to the tutor session itself

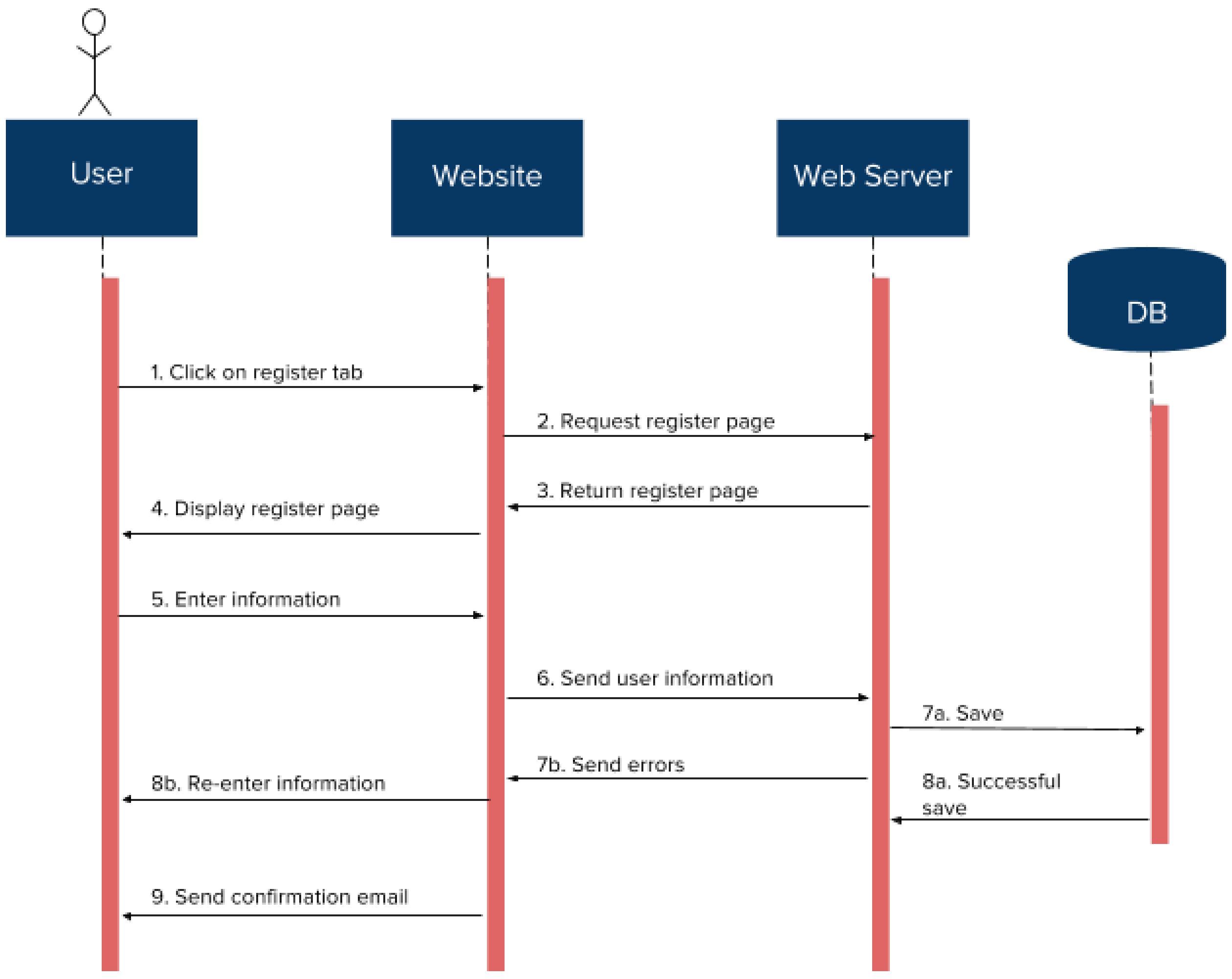
Post - Belongs to a Student object

- Contains attributes that summarize the student's tutor request, specifically:
    - *subject,* and *course* - course information
    - *price* - what a student is willing to pay for a tutor
    - *location*, *time and notes* - location and time preferences as well as additional preferences for the tutor session

Sequence Diagrams

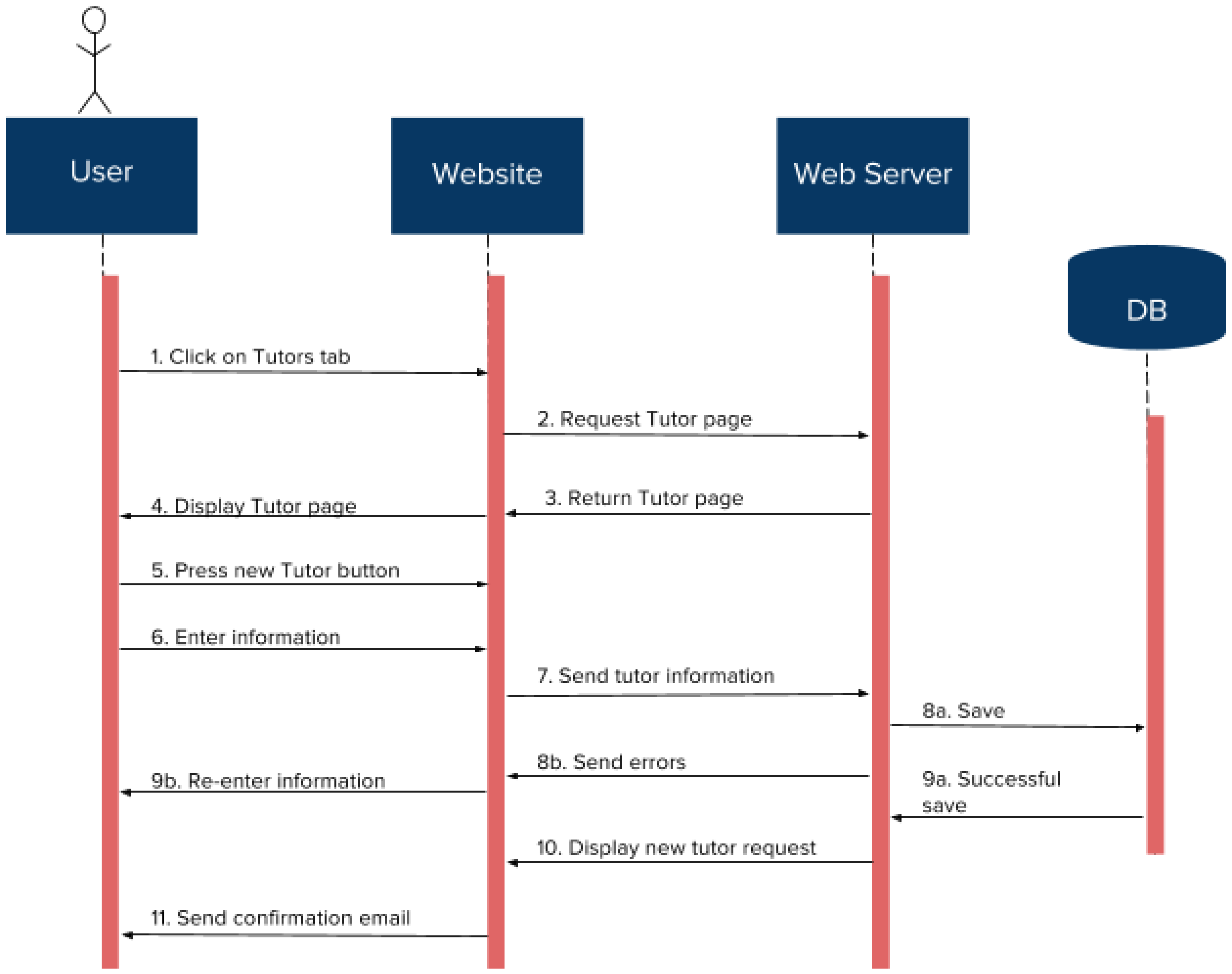
Registration Diagram

The registration sequence begins with the user entering the registration page of the website. The user clicks on the register tab and the website sends a request to the web server which will then return the registration page and display it to the user. The user is then fills out the page with basic account creation information. This would be sent to the web server which would either be saved or would return an error if the input is invalid. Once the inputted data is successfully saved to the database, the account will be created and a confirmation email will be sent to the user.



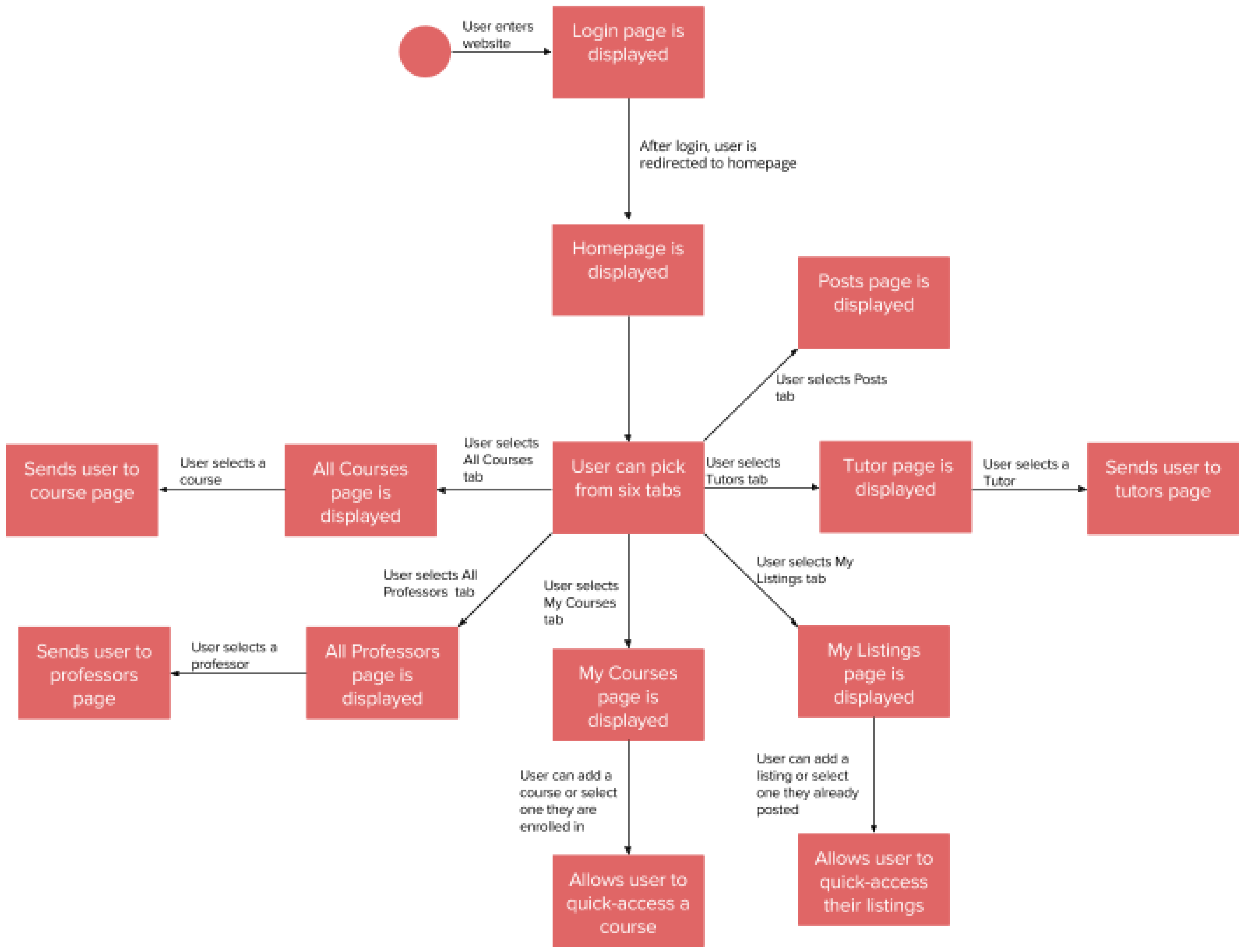
Tutor Request Diagram

The tutor request sequence begins with the user entering the tutor page of the website. The user will click on the tutors tab, sending a request to the website, which in turn will return the tutor page and display it to the user. The user can then select the option to make a new tutor request and fill out the requisite information. This will then be sent to the web server which will either save it to the database or return an error. The new tutor request will then be displayed on the website and a confirmation email will be sent to the user.

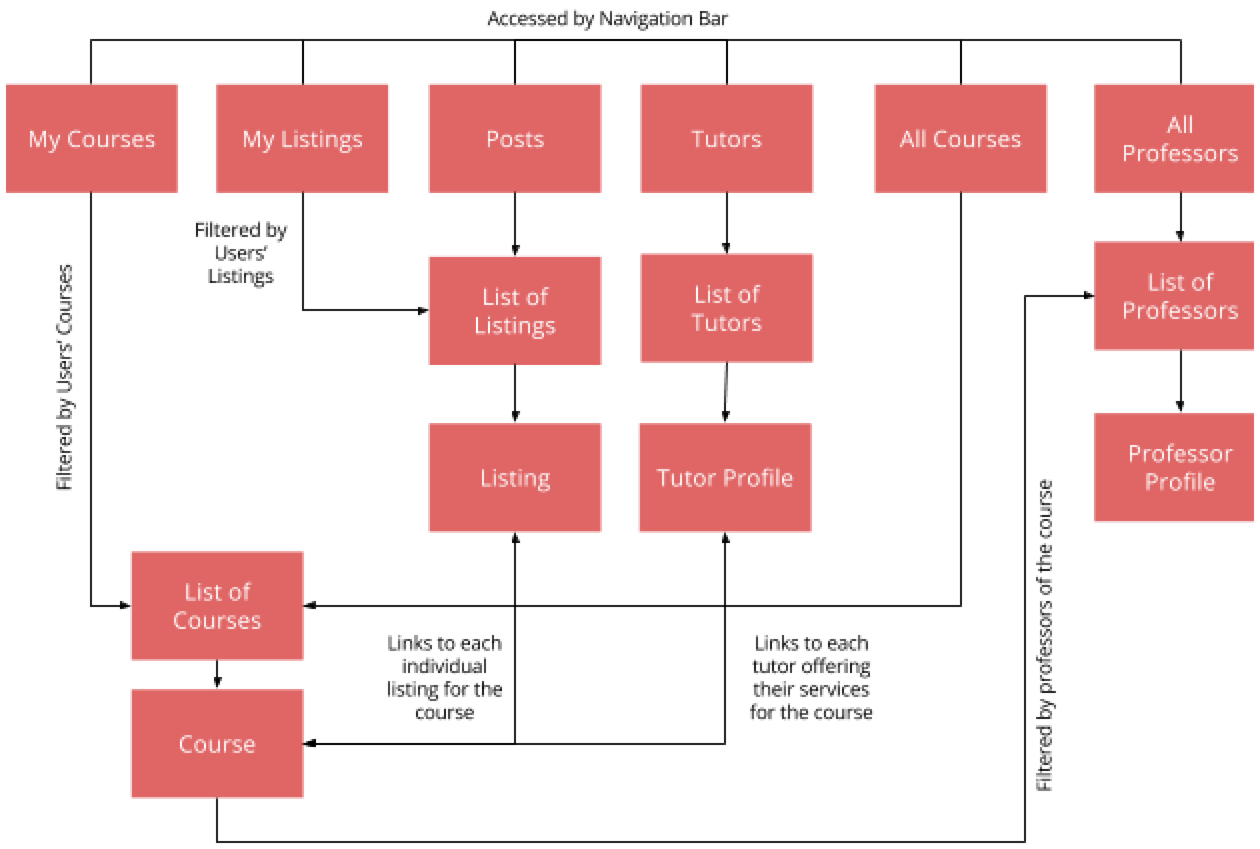


Activity/State Diagram

When the user enters the website, the login page will be displayed, which will then redirect to the homepage after login. On the homepage, six tabs are available: Posts, Tutors, My Listings, My Courses, All Professors, and All Courses.

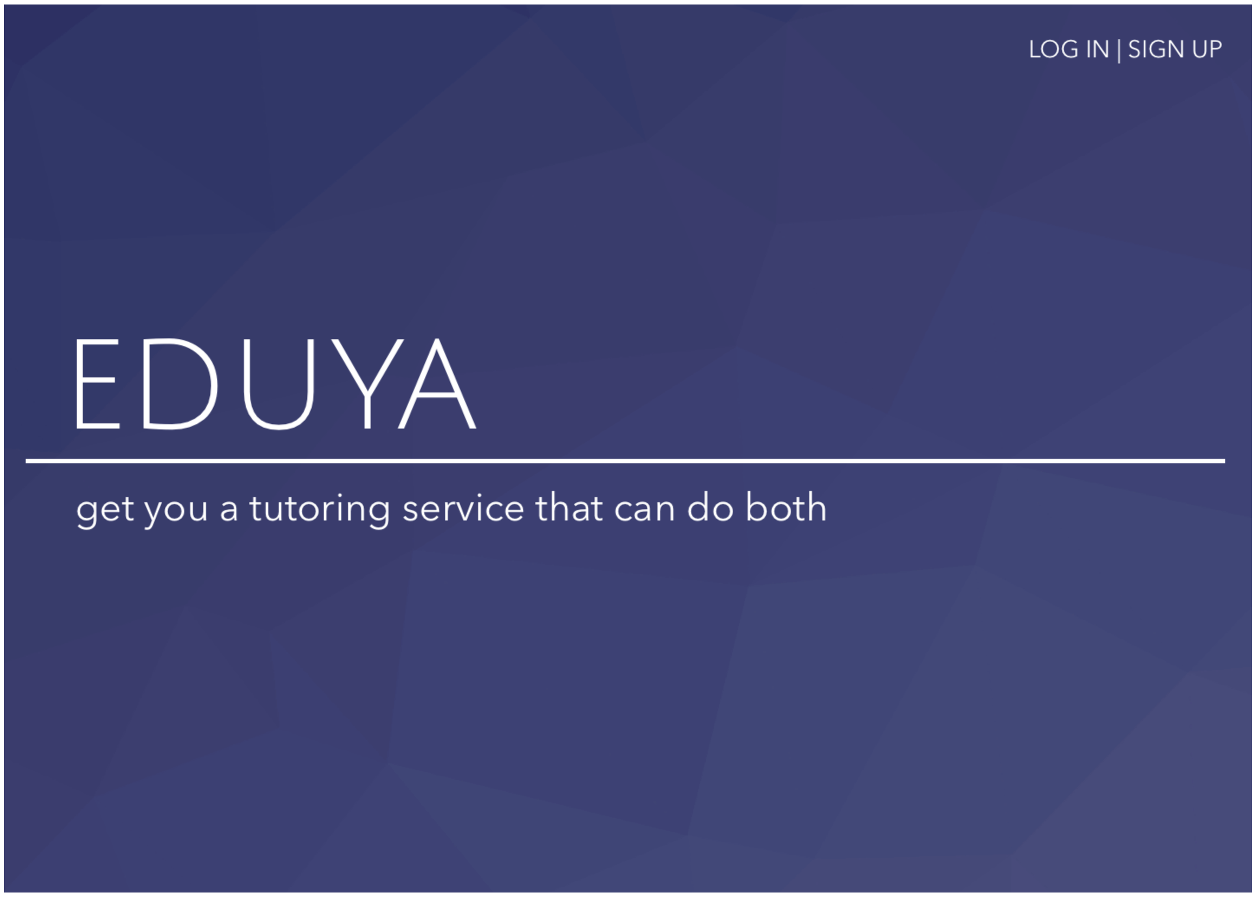

User enters website
Login page is displayed
After login, user is redirected to homepage
Homepage is displayed
Posts page is displayed
User selects Posts tab
User can pick from six tabs
User selects All Courses tab
All Courses page is displayed
User selects a course
Sends user to course page
User selects Tutors tab
Tutor page is displayed
User selects a Tutor
Sends user to tutors page
User selects All Professors tab
All Professors page is displayed
User selects a professor
Sends user to professors page
User selects My Courses tab
My Courses page is displayed
User can add a course or select one they are enrolled in
Allows user to quick-access a course
User selects My Listings tab
My Listings page is displayed
User can add a listing or select one they already posted
Allows user to quick-access their listings

Navigation Diagram

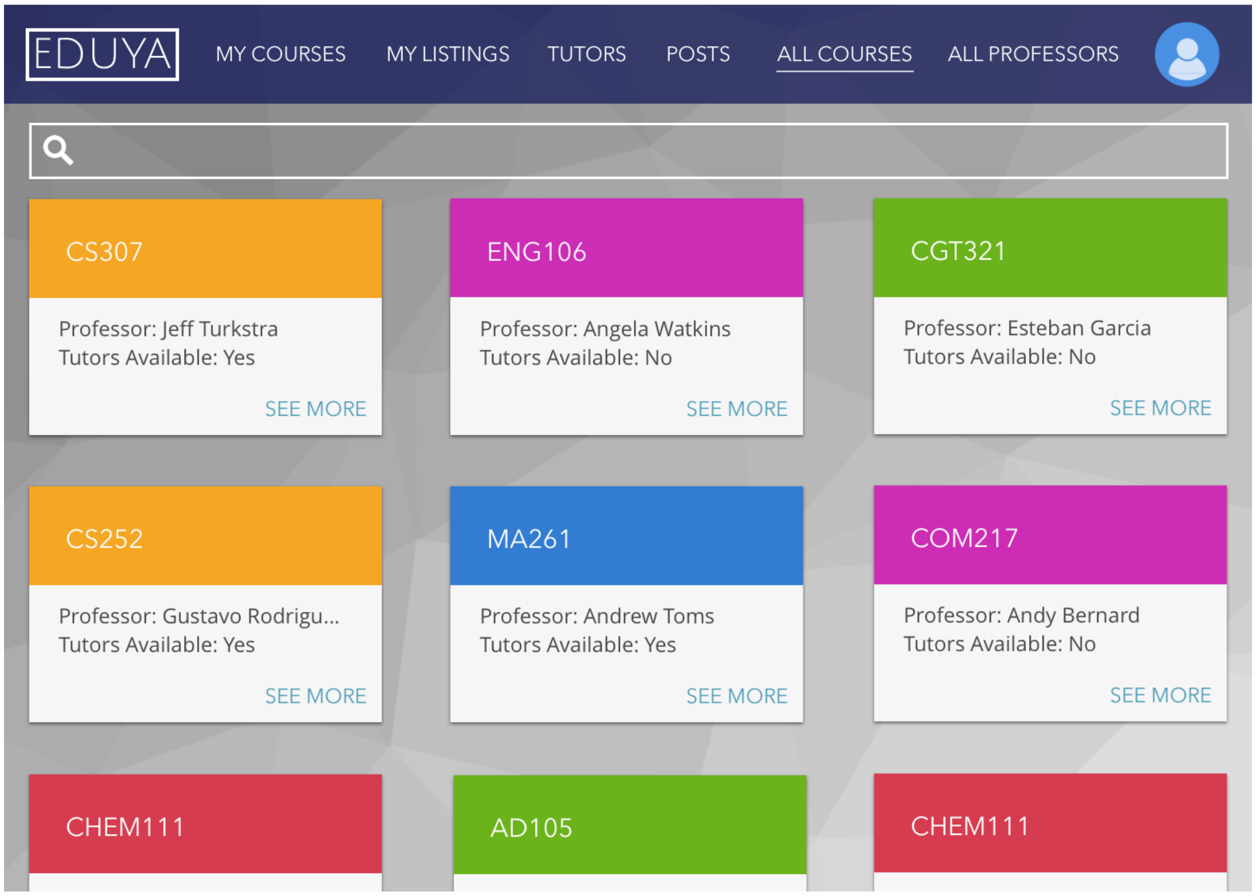

Accessed by Navigation Bar
My Courses
My Listings
Posts
Tutors
All Courses
All Professors
Filtered by Users' Listings
List of Listings
List of Tutors
List of Professors
Listing
Tutor Profile
Professor Profile
Filtered by Users' Courses
List of Courses
Course
Links to each individual listing for the course
Links to each tutor offering their services for the course
Filtered by professors of the course

UI Mockups

We designed our UI to be content focused and easily scalable for different screen sizes. The top navigation bar makes all the available content easy to access, and the cards modularize the content so it's easy to scan through. The cards also allow for scalability, showing one card at a time for smaller screen sizes. The prominent search bars allow the user to quickly find what they're looking for, and the colors help differentiate between different topics quickly.



Landing page



CS307
Professor: Jeff Turkstra
Tutors Available: Yes
SEE MORE

ENG106
Professor: Angela Watkins
Tutors Available: No
SEE MORE

CGT321
Professor: Esteban Garcia
Tutors Available: No
SEE MORE

CS252
Professor: Gustavo Rodrigu...
Tutors Available: Yes
SEE MORE

MA261
Professor: Andrew Toms
Tutors Available: Yes
SEE MORE

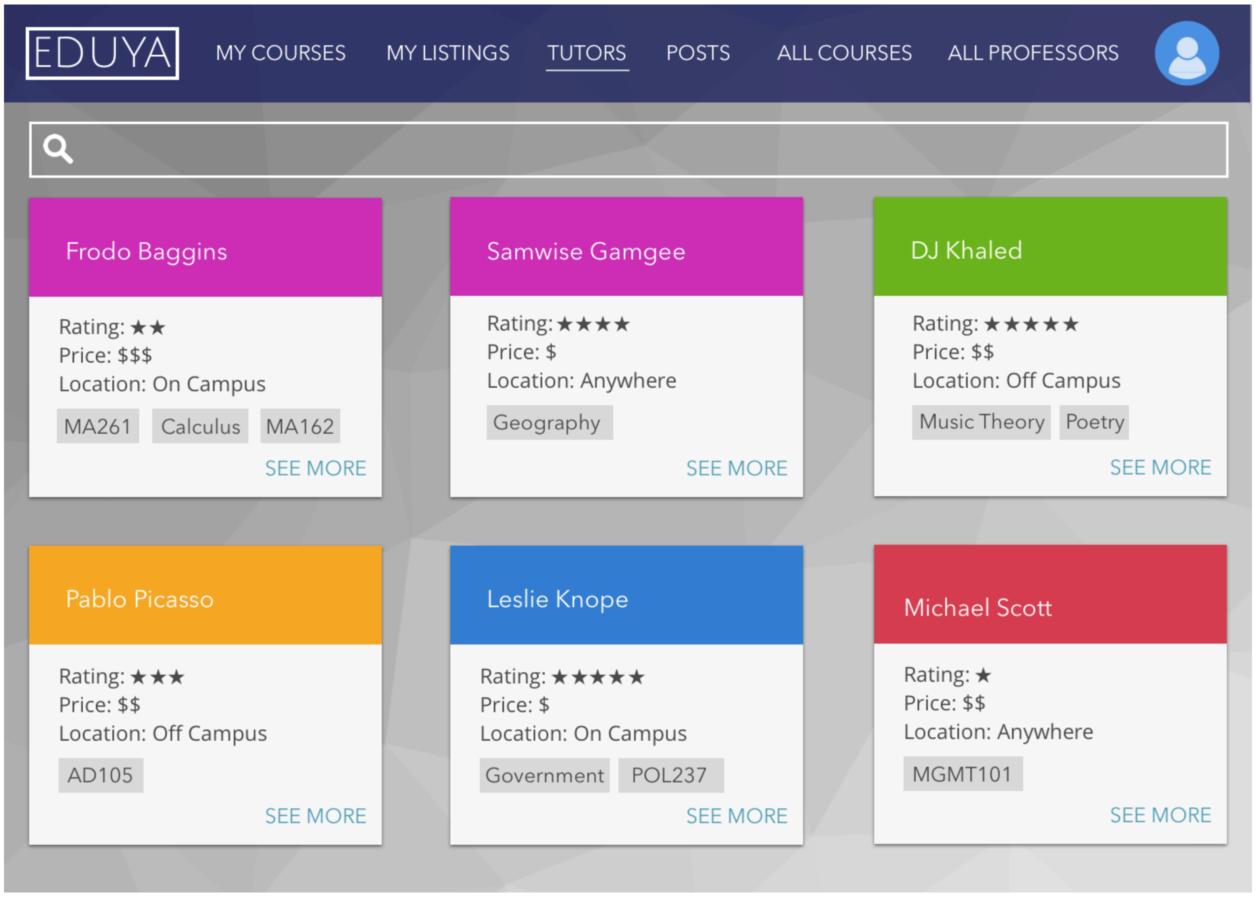
COM217
Professor: Andy Bernard
Tutors Available: No
SEE MORE

CHEM111

AD105

CHEM111

A view of all the courses available; gives brief but pertinent information and is color coded by related fields and disciplines



Frodo Baggins
Rating: ★★
Price: $$$
Location: On Campus
MA261 Calculus MA162
SEE MORE

Samwise Gamgee
Rating: ★★★★
Price: $
Location: Anywhere
Geography
SEE MORE

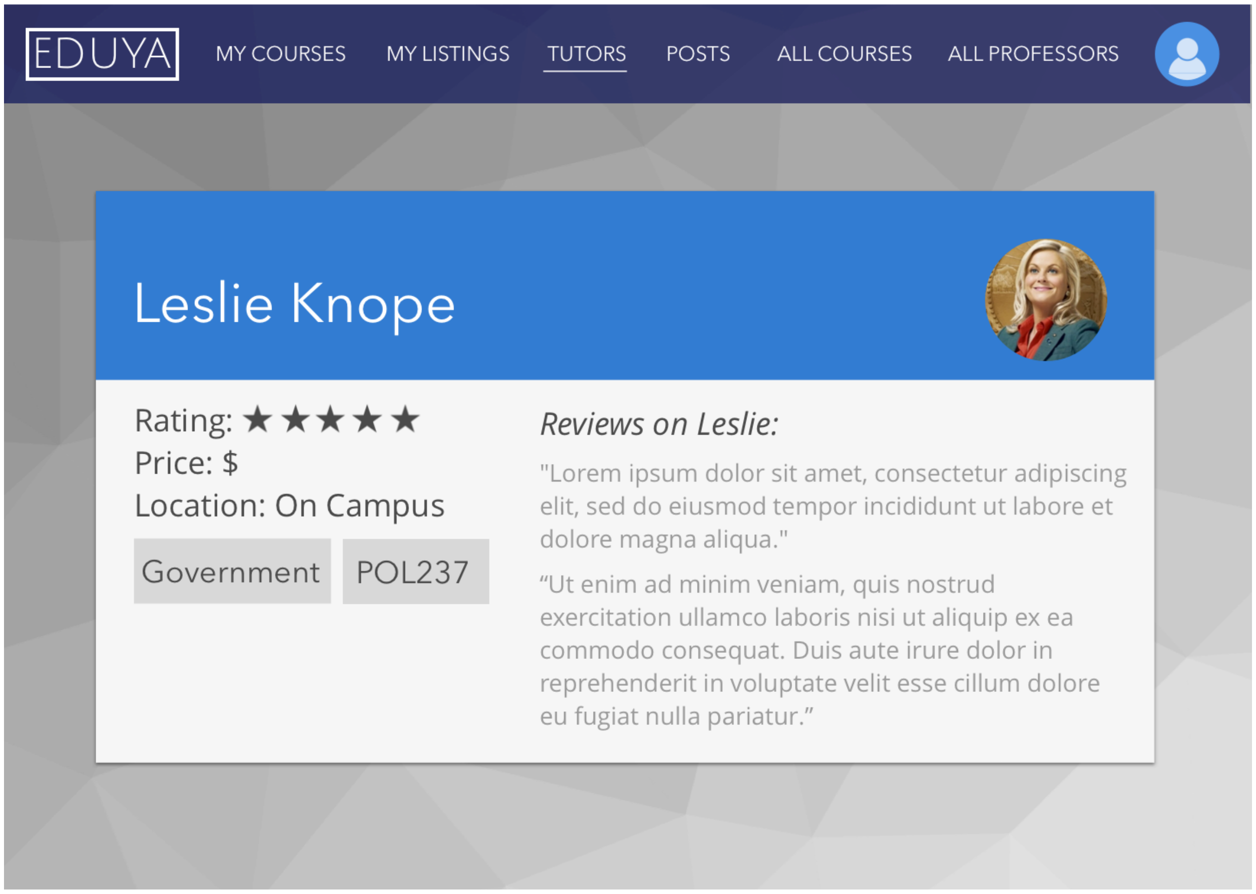
DJ Khaled
Rating: ★★★★★
Price: $$
Location: Off Campus
Music Theory Poetry
SEE MORE

Pablo Picasso
Rating: ★★★
Price: $$
Location: Off Campus
AD105
SEE MORE

Leslie Knope
Rating: ★★★★★
Price: $
Location: On Campus
Government POL237
SEE MORE

Michael Scott
Rating: ★
Price: $$
Location: Anywhere
MGMT101
SEE MORE

A view of the available tutors shown in a Yelp-style format, featuring aspects like pricing, offered subjects, location, and user-curated ratings



Leslie Knope

Rating: ★★★★★
Price: $
Location: On Campus

Government POL237

Reviews on Leslie:

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua."

“Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.”

A profile for a tutor which includes the same information as the thumbnails and includes further reviews given by users about the tutors