Table of Contents

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

Work Flow

Profile Setup

An already registered user will setup profile with the requested information above. All data fields will be form validated.

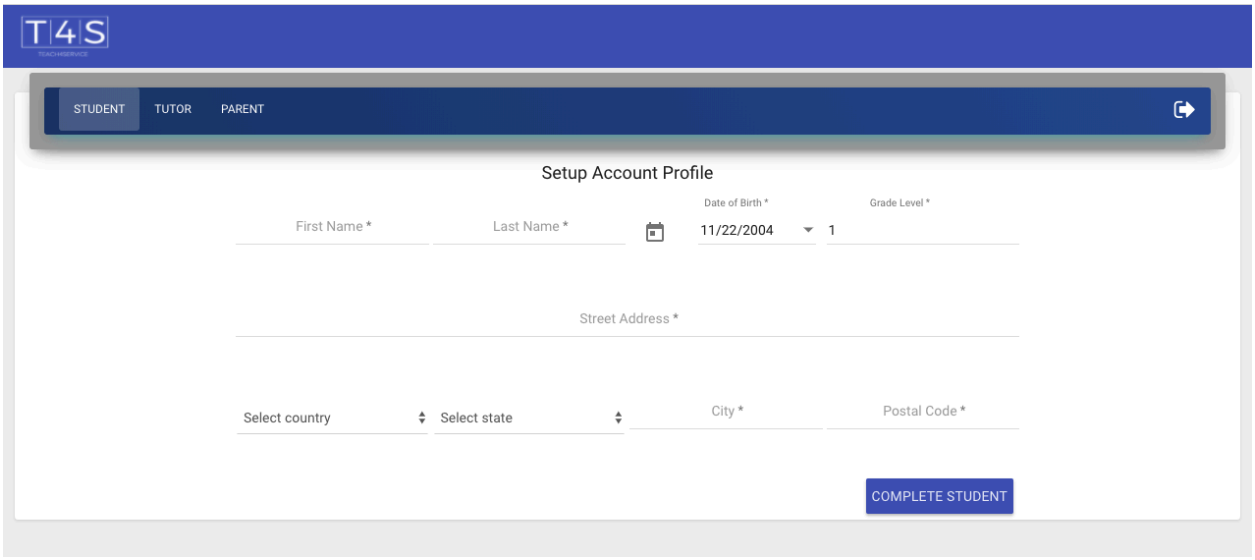


Figure 1.1 Student Profile Setup

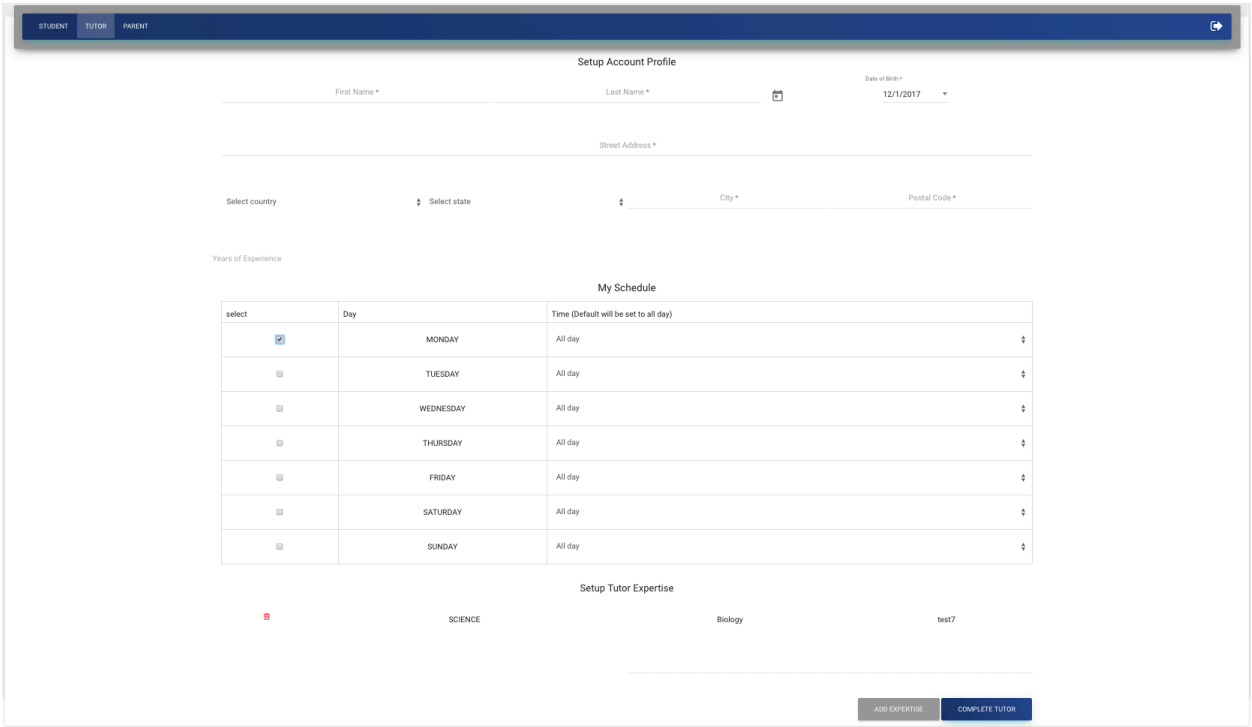


Figure 1.2 Tutor Profile Setup

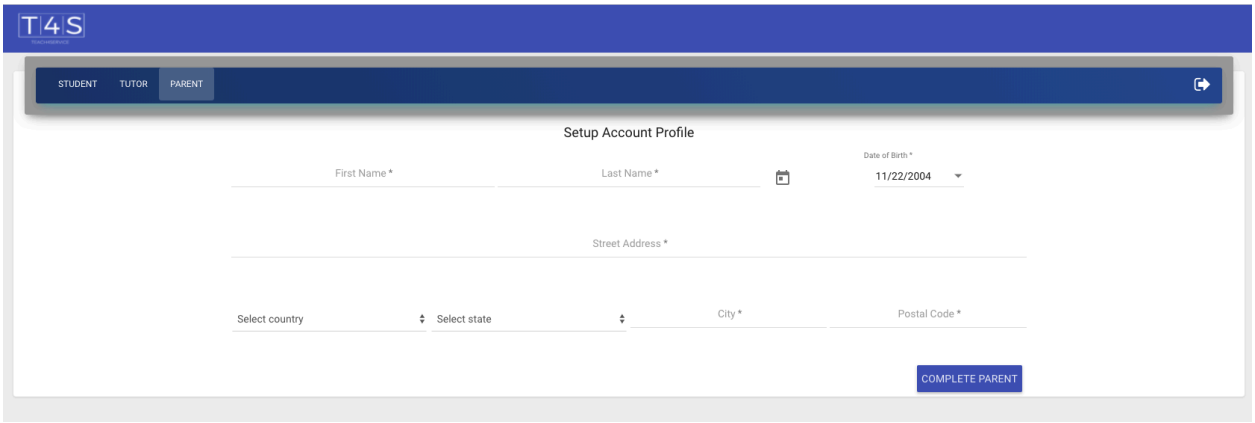


Figure 1.3 Parent Profile Setup

Profile Management

A logged in user with at least one profile setup will see their account profiles.

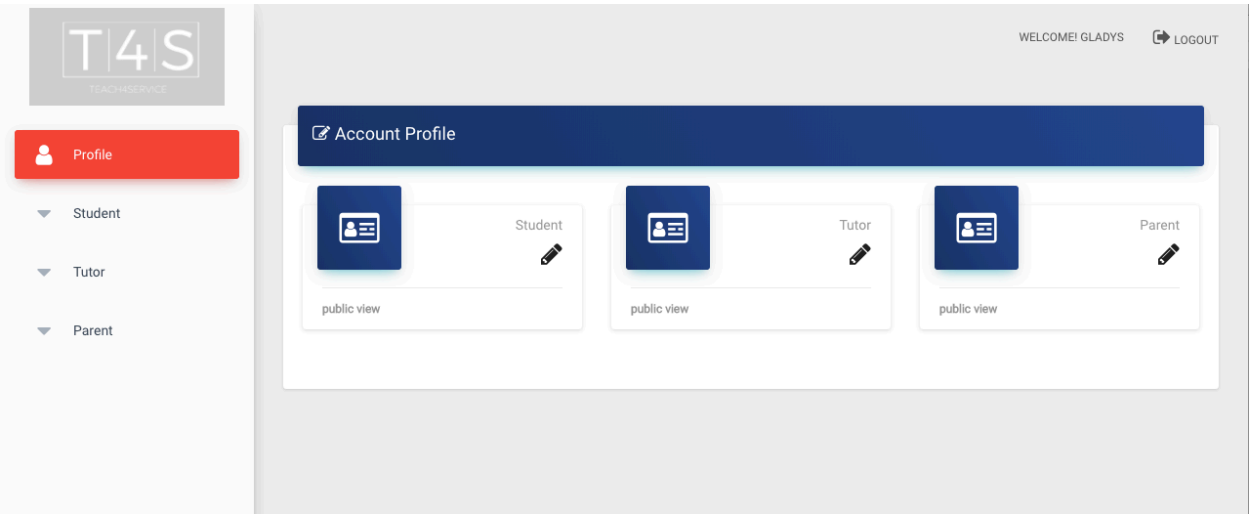


Figure 2. View User Profiles

Edit Profile

A logged in user with an existing student or parent profile will edit their profile information

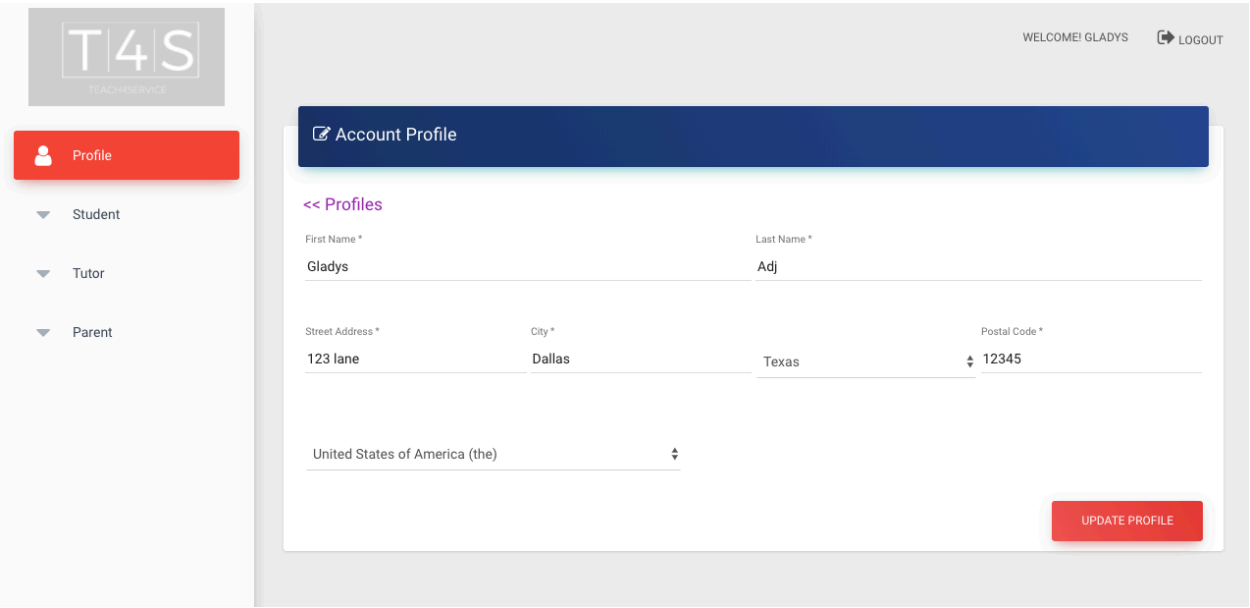


Figure 3. Edit student or parent profile

Public Profile

A logged in user with a student profile will see how their profile will appear publically

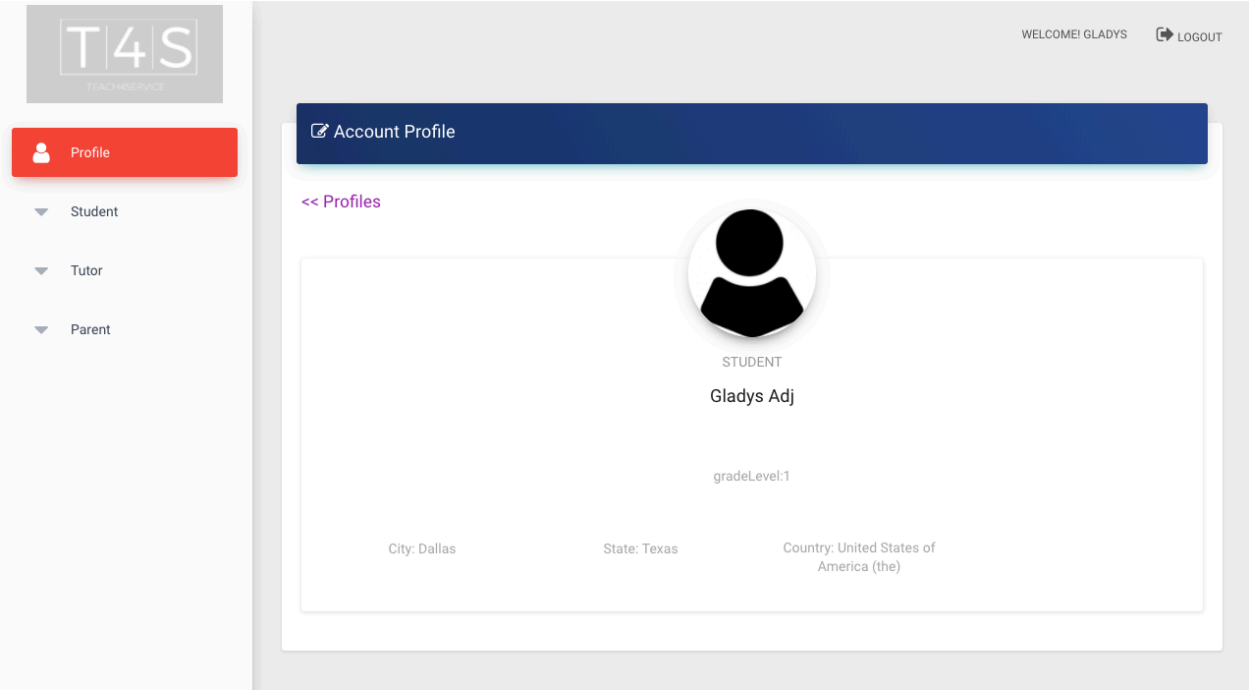


Figure 4 Public view of the student profile

Add Additional Profiles

A logged in user can add additional profiles to his/her account. He or she can add a student, tutor, or manager profile. Profile must not be one the user already has setup up

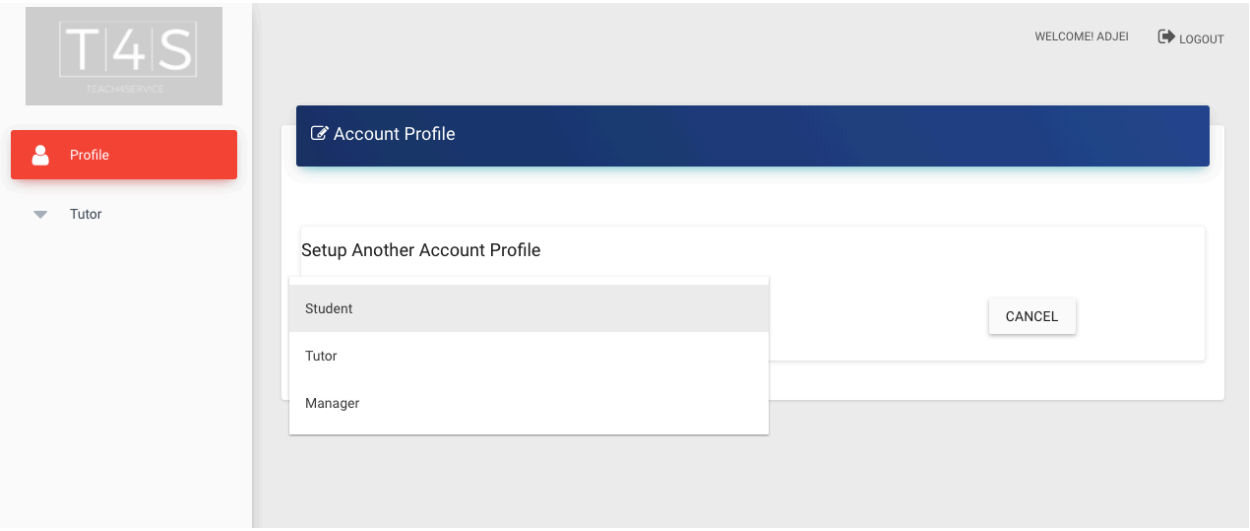


Figure 5.1 Options to add additional profiles

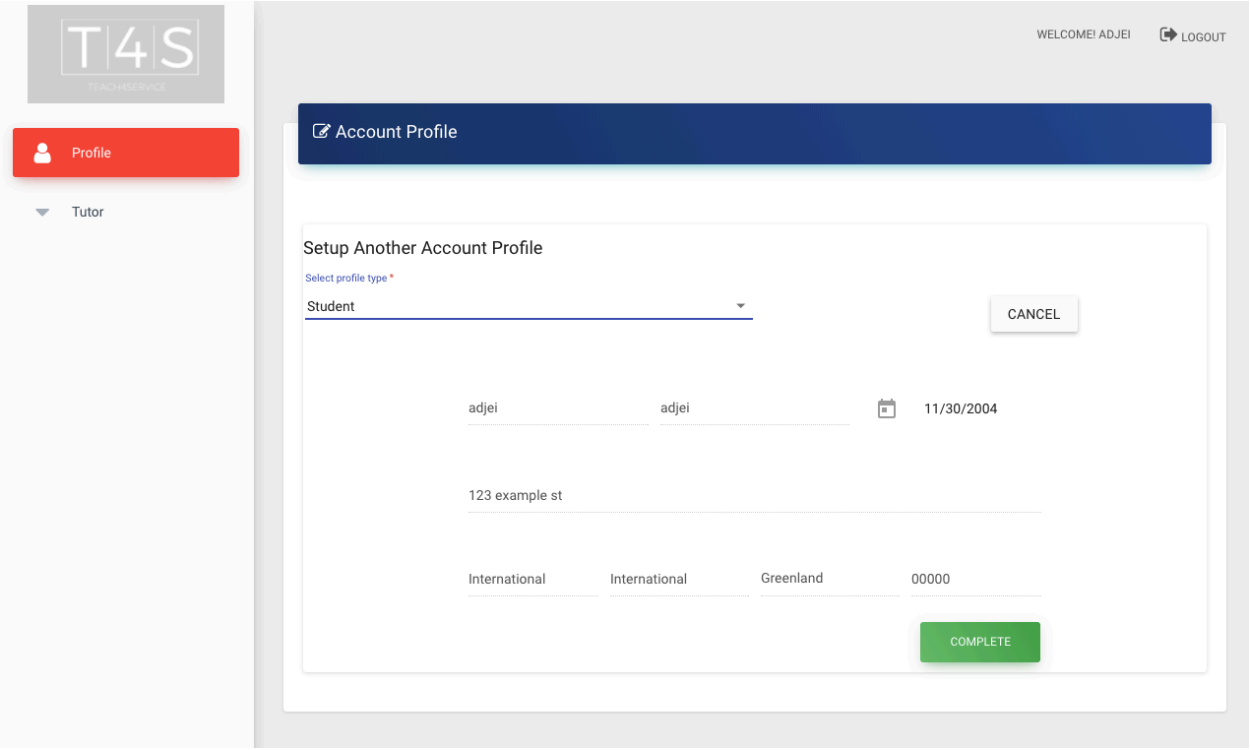


Figure 5.2 Adding a student profile

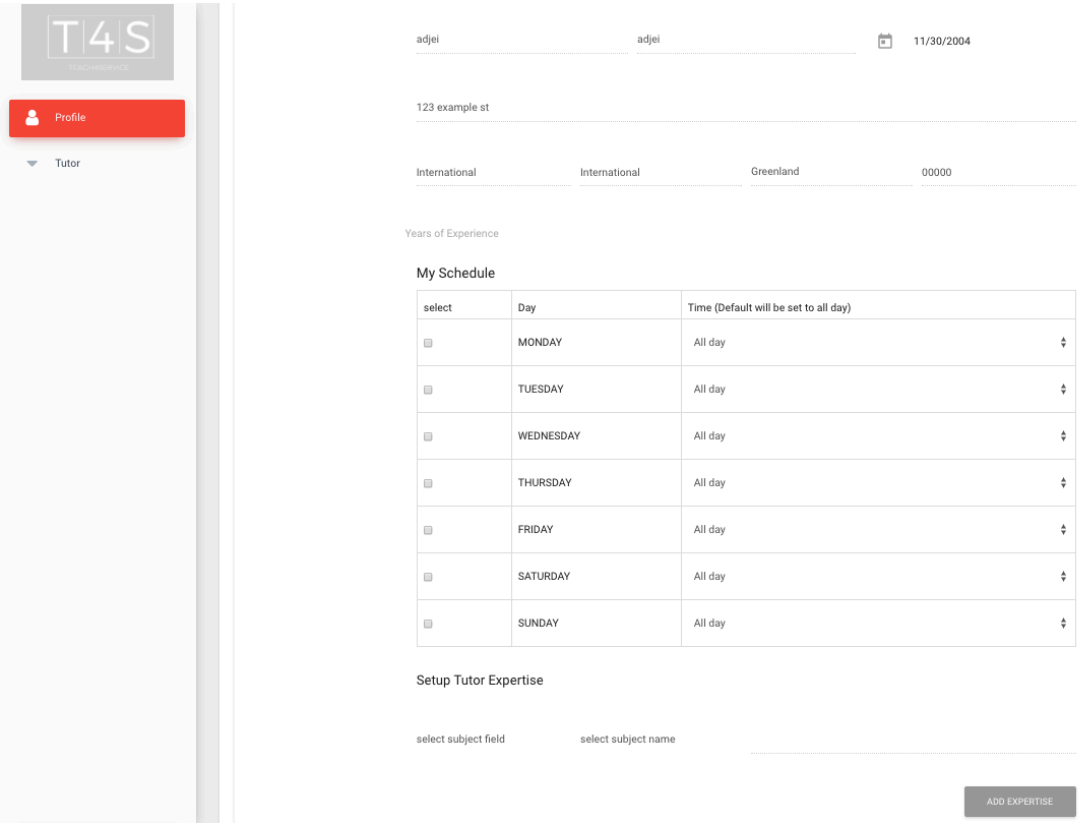


Figure 5.2 Adding a tutor profile

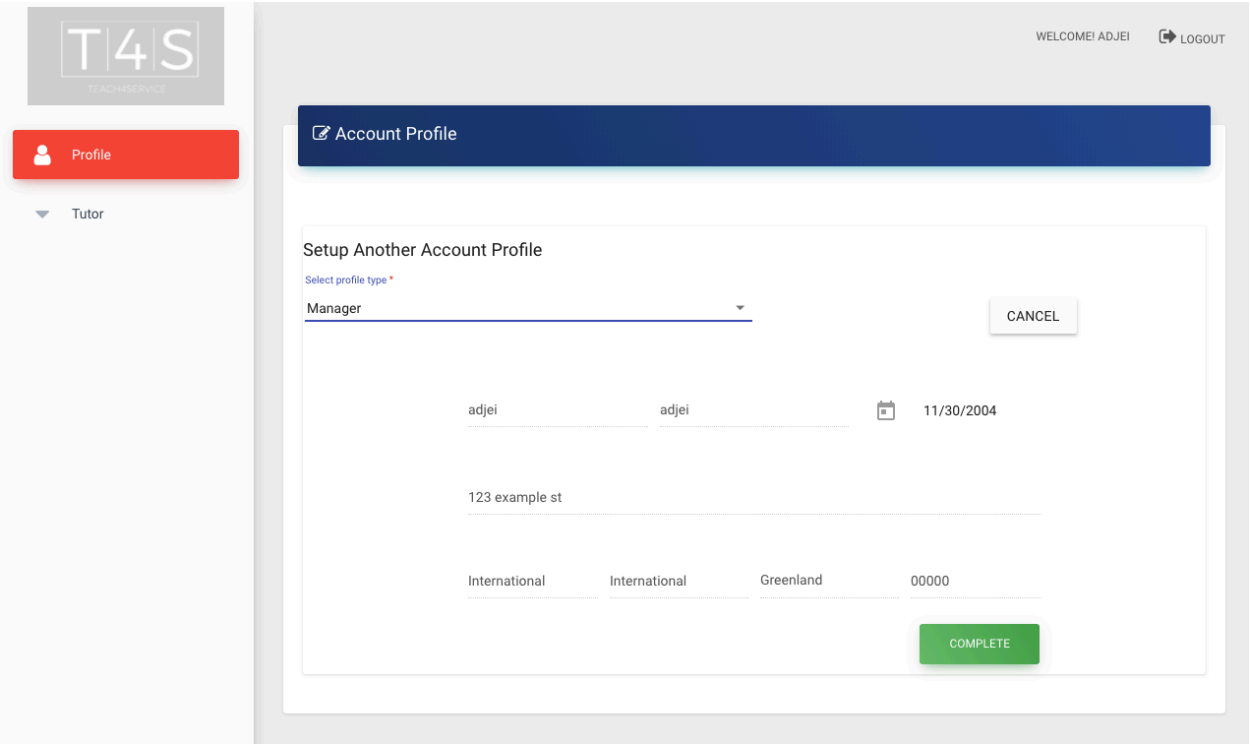


Figure 5.2 Adding a manager profile

User Dashboard

A logged in user with a student or parent profile will view their respective dashboard where if a student, user will see their sessions for the day and reviews. And if parent, will see the sessions for the day of the students he/she manages as well as be able to add students to manage

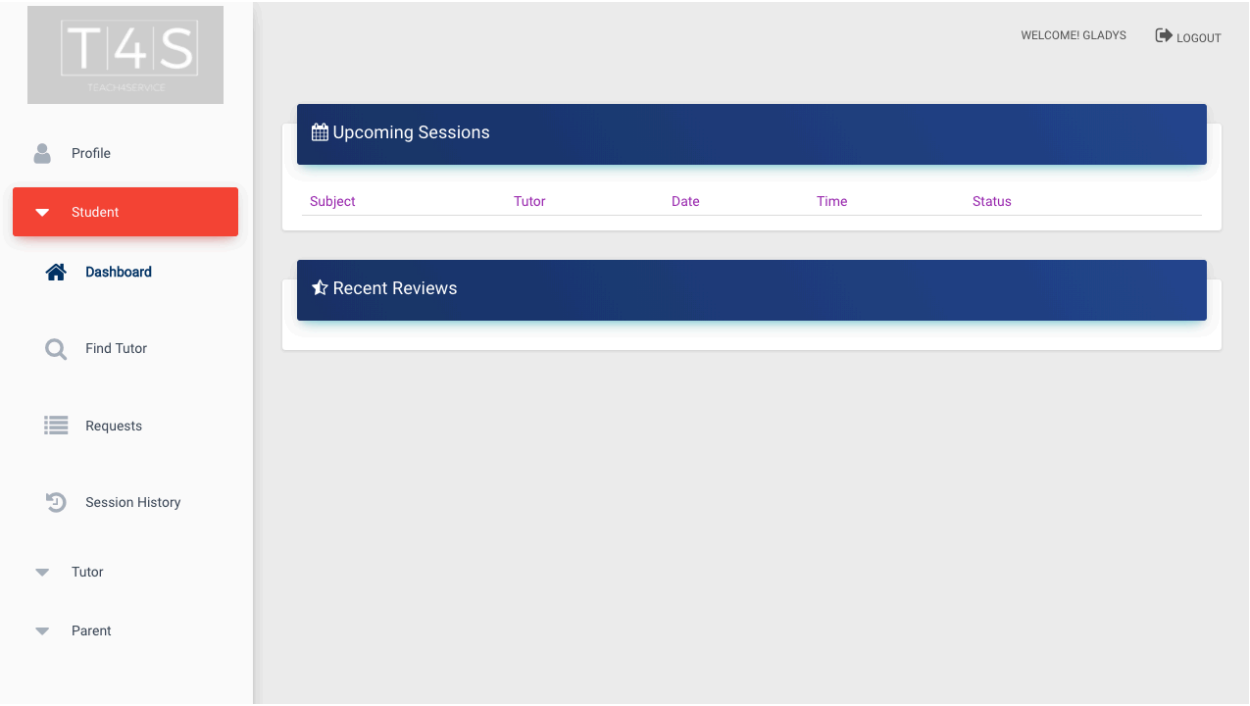


Figure 5.1 View of student dashboard

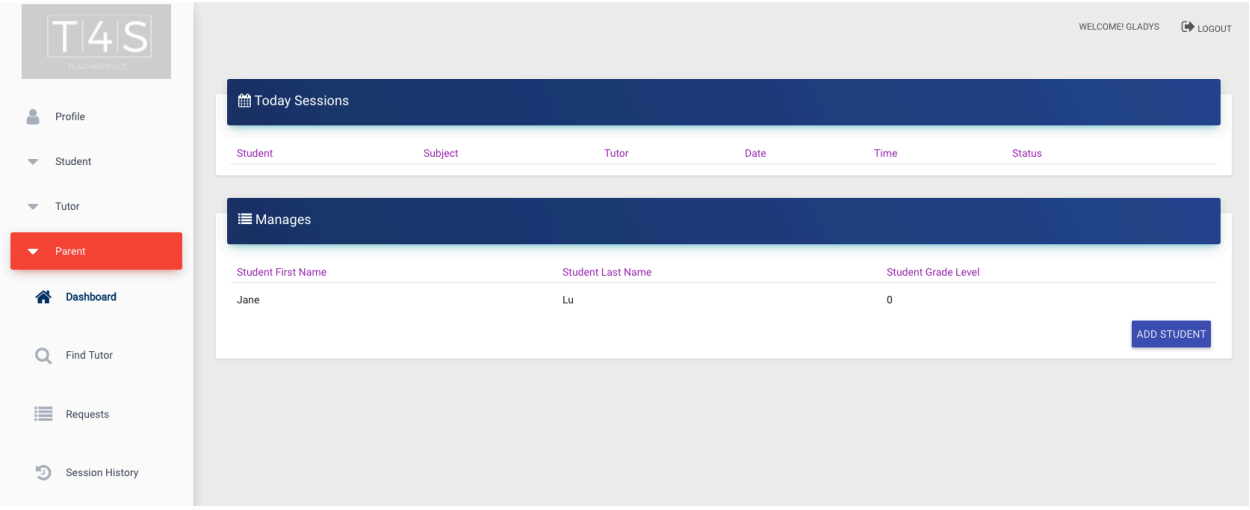


Figure 5.2 View of Parent dashboard

Search and Find Tutor

A logged in student or parent will search for tutor by subject, tutor name, or tutor username. The search will return results that are divided into tutors who are immediately available to tutor and other tutors who are not immediately available but matches the search criteria. User can filter search by text or by subject

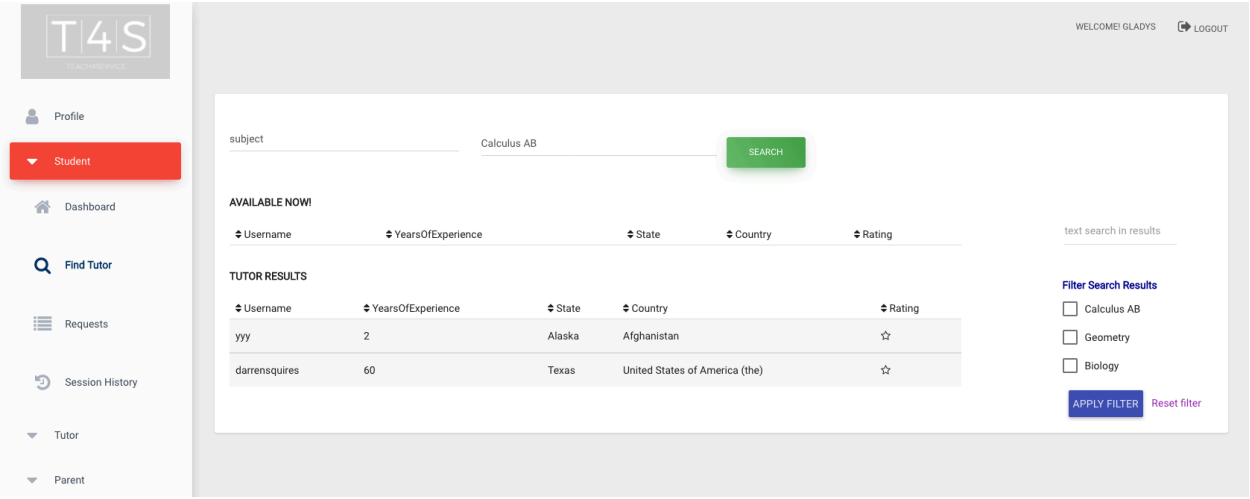


Figure 6 Search and find tutor

Register with Tutor

A logged in student or parent will register with a tutor. If parent, user will identify which of his/her students he/she is registering for

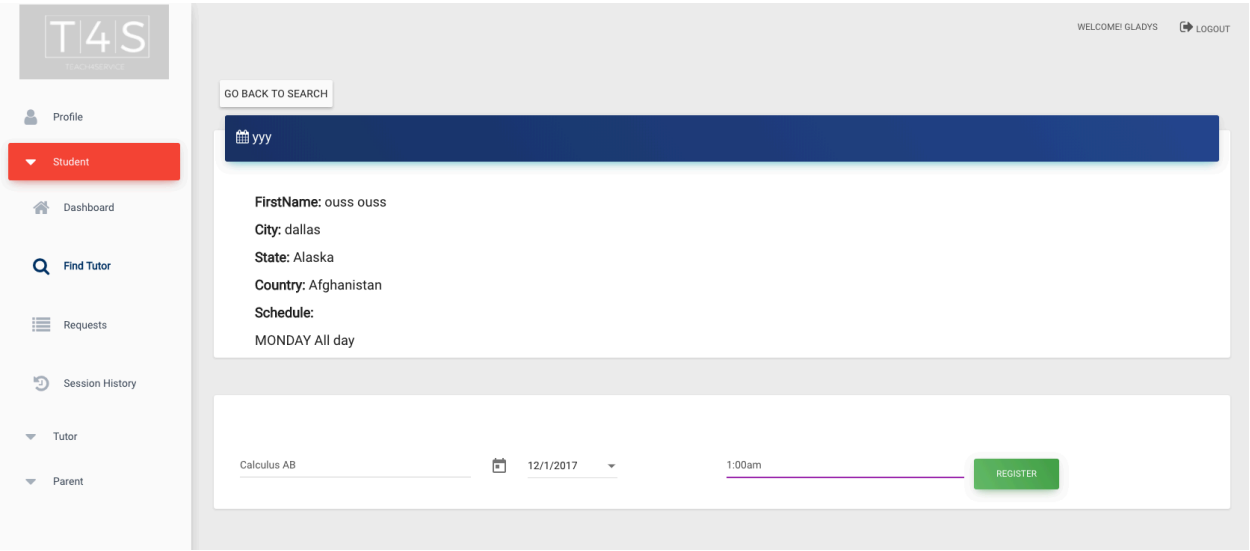


Figure 7.1 Student registering with tutor

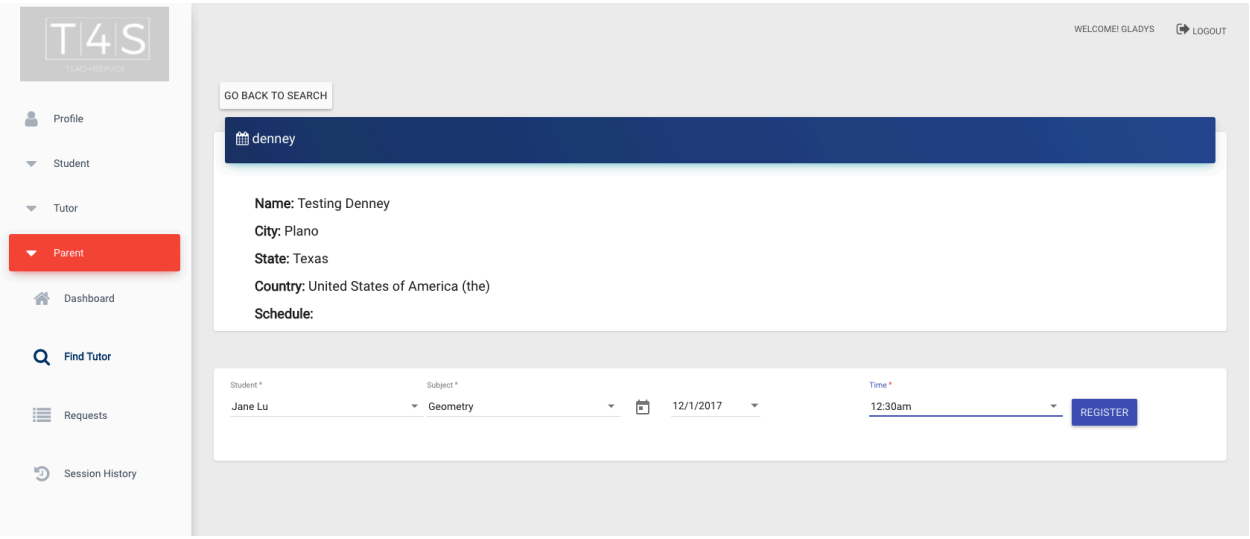


Figure 7.2 Parent registering one of his/her students with a tutor

Session Registering Requests

A logged in student or tutor will view the requests for sessions they have sent or received

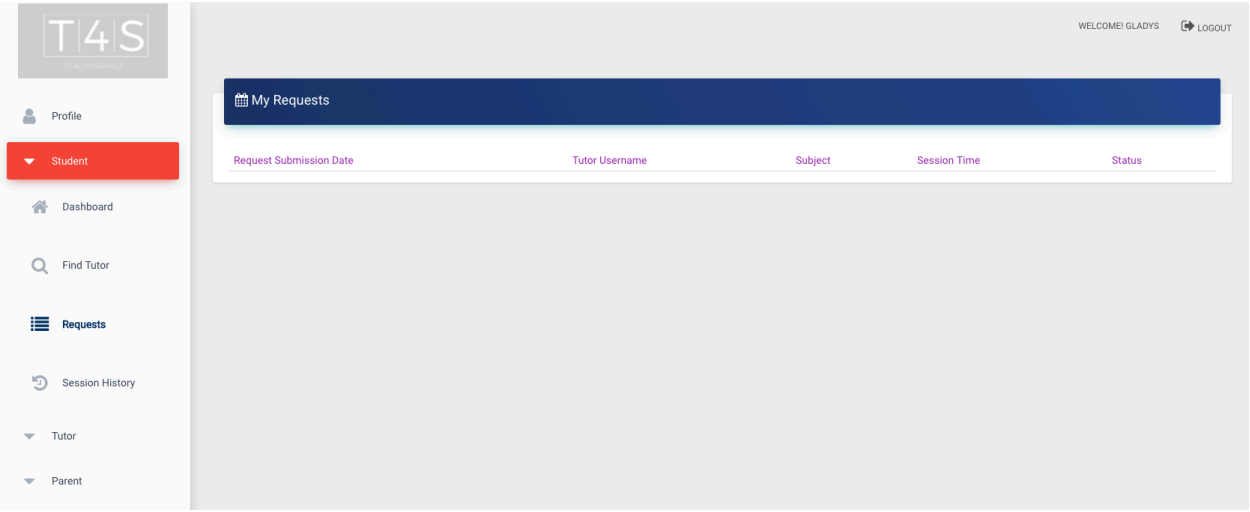


Figure 8.1 Student requests he/she has sent to a tutor

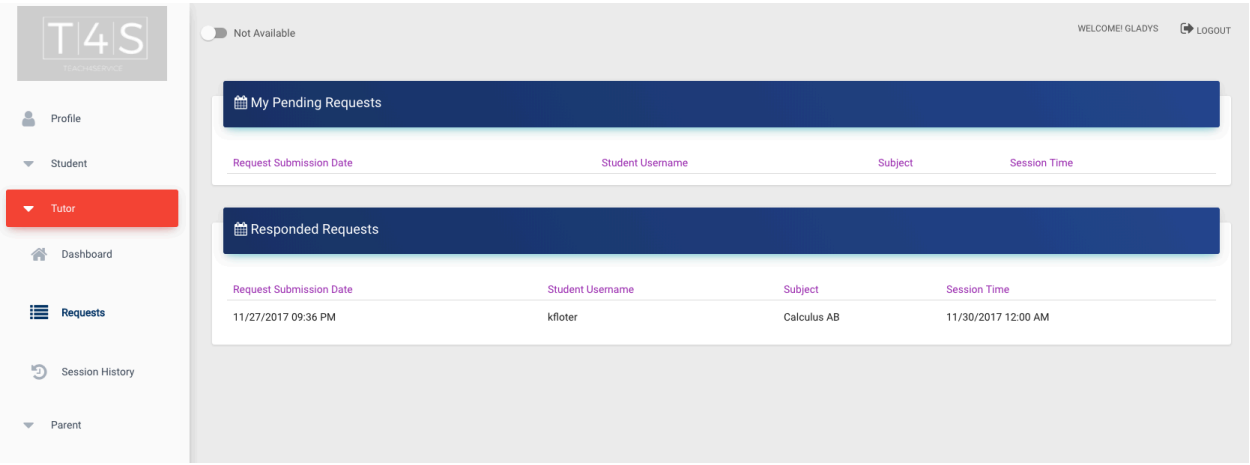


Figure 8.2 Tutor requests he/she has received and the requests he/she has responded to

Past Session History

A logged in student will view sessions he/she has had in the past

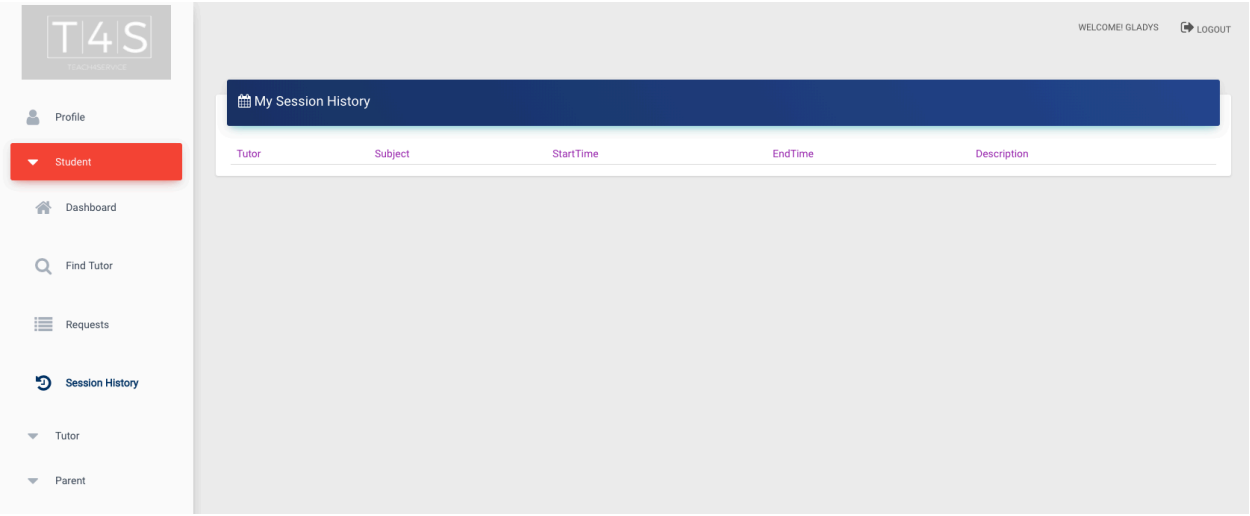


Figure 9 Student past session history

Admin Login

An added admin will login with their username and admin password

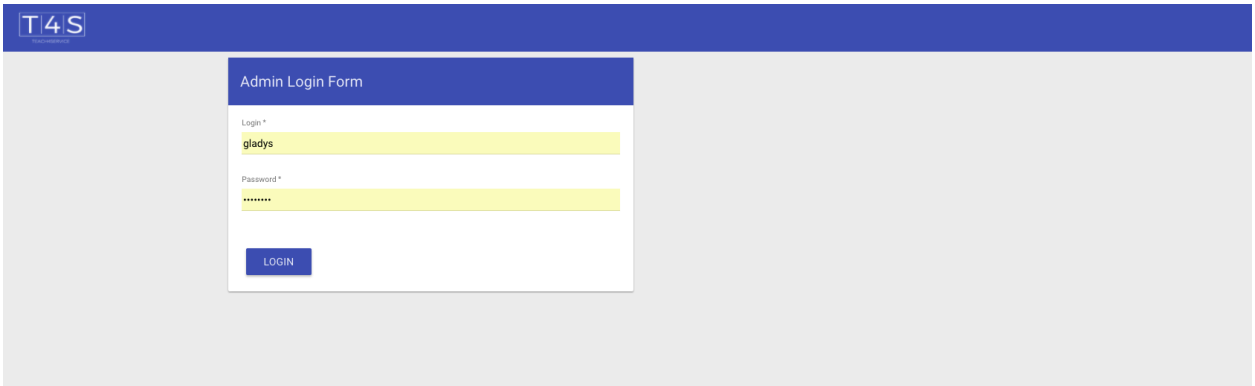


Figure 10 Admin login

Admin Functionalities

Figure 11 Admin manage subjects, tutors, and other admin accounts

Admin Manage Subjects

A logged in admin will create, edit, or delete subjects

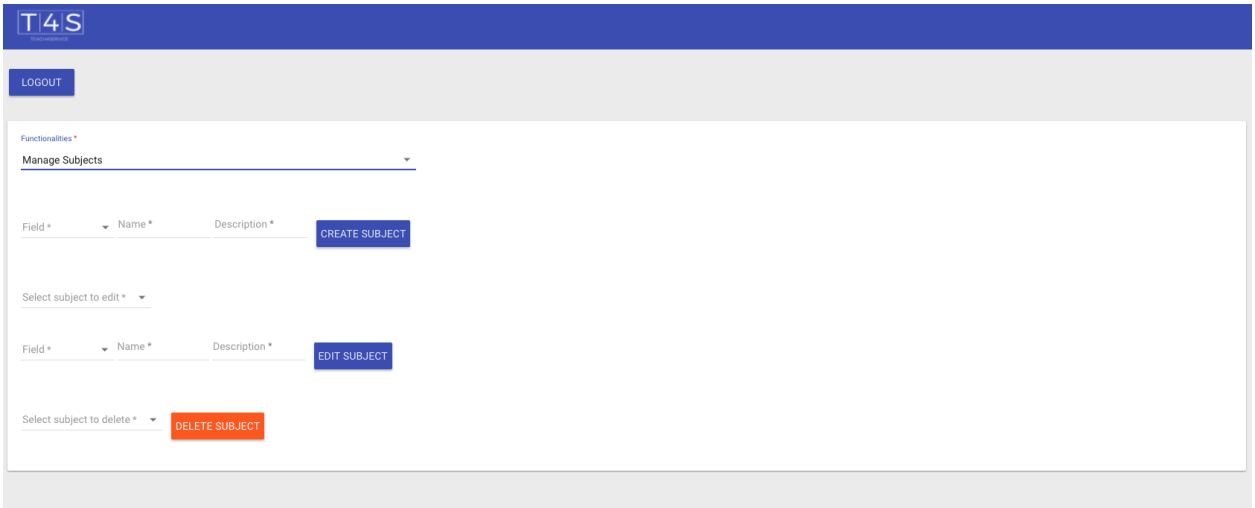


Figure 12 Admin manages subjects

Admin Manage Tutor

A logged in admin can approve or decline a tutor

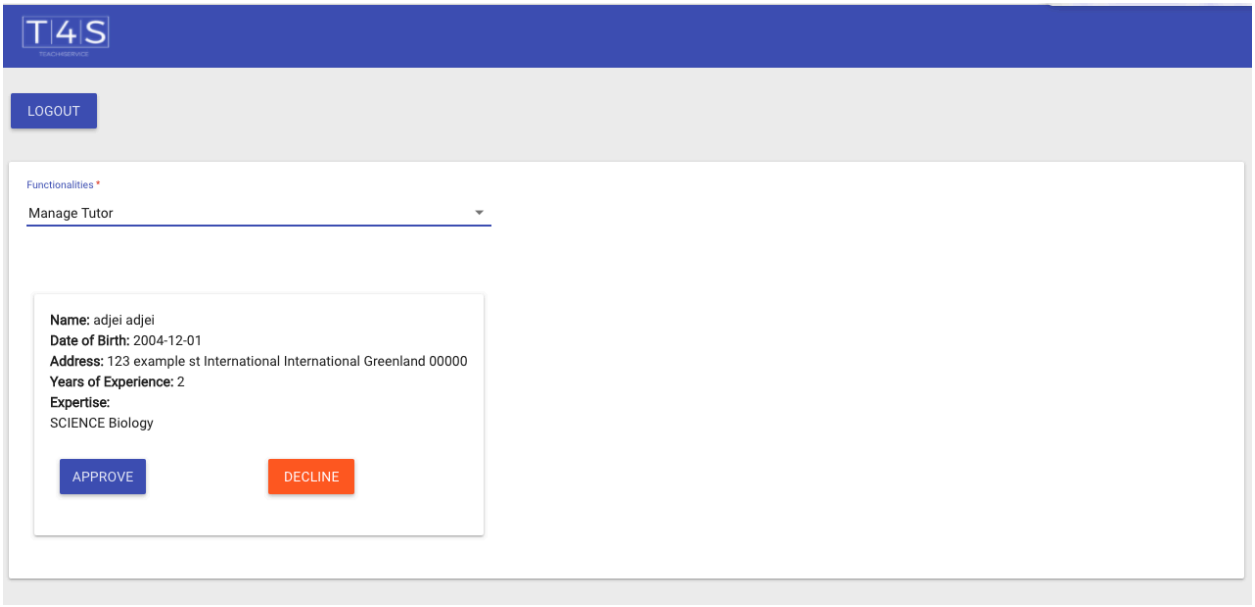


Figure 13 Admin manages tutor

Admin Manages Other Admin Accounts

A logged in admin can create or delete an admin

Figure 14 Admin manages admins

Implementation

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

HTML Code

This code is the layout for profile management

profileLayout.html

```
<div class="card">
    <div class="card-header" data-background-color="blue">
        <h4 class="title"><i class="fa fa-pencil-square-o" aria-hidden="true"></i> Account Profile</h4>
    </div>
    <div class="card-content">
        <ui-view></ui-view>
    </div>
</div>
```

initialProfileSetup.html

```
<div class="container-fluid">
    <div class="card text-center">
        <div class="card-header">
            <ul class="nav nav-tabs card-header-tabs" data-background-color="blue">
                <li ui-sref-active="active">
                    <a ui-sref="app.profileSetup.student">Student</a>
                </li>
                <li ui-sref-active="active">
                    <a ui-sref="app.profileSetup.tutor">Tutor</a>
                </li>
                <li ui-sref-active="active">
                    <a ui-sref="app.profileSetup.manager">Parent</a>
                </li>
                <li class="navbar-right" ng-click="logout()">
                    <a><i class="fa fa-sign-out fa-2x" aria-hidden="true"></i></a>
                </li>
            </ul>
        </div>
        <div class="card-block">
            <h4 class="card-title">Setup Account Profile</h4>
            <ui-view></ui-view>
        </div>
    </div>
</div>
```

This code is for student profile setup

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

initialProfileSetup.html | manager.findTutor.html | searchTutor.html | tutor.profileSetup.html | student.profileSetup.html

```
<div class="col-md-2"></div>

<div class="col-md-8">
    <form ng-submit="completeProfile()">
        <div layout="row">
            <md-input-container flex="25">
                <label>First Name</label>
                <input type="text" ng-model="student.firstName" required>
            </md-input-container>
            <md-input-container flex="25">
                <label>Last Name</label>
                <input type="text" ng-model="student.lastName" required>
            </md-input-container>
            <md-input-container flex="25">
                <label>Date of Birth</label>
                <md-datepicker md-placeholder="Date of Birth"
                               ng-model="student.dateOfBirth" md-min-date="minDate" md-max-date="maxDate"
                               required></md-datepicker>
            </md-input-container>
            <md-input-container flex="25">
                <label>Grade Level</label>
                <input type="number" ng-model="student.gradeLevel" required min="1">
            </md-input-container>
        </div>
        <div layout="row">
            <md-input-container flex="100">
                <label>Street Address</label>
                <input type="text" ng-model="student.streetAddress" required>
            </md-input-container>


        </div>
        <div layout="row">

            <md-input-container flex="25">
                <select class="form-control" ng-model="student.country" pvp-country-picker="name" required>
                    <option value="">Select country</option>
                </select>
            </md-input-container>
            <md-input-container flex="25">
                <select class="form-control" ng-model="student.state"  required>
                    <option value="">Select state</option>
                    <option value="{{s}}" ng-repeat="s in states">{{s}}</option>
                </select>
            </md-input-container>
            <md-input-container flex="25">
                <label>City</label>
                <input type="text" ng-model="student.city" required>
            </md-input-container>
            <md-input-container flex="25">
                <label>Postal Code</label>
                <input type="text" ng-model="student.postalCode" minlength="5" maxlength="10" required>
            </md-input-container>
        </div>
        <md-button type="submit" class="md-raised md-primary" style="...">complete student</md-button>
    </form>
</div>

<div class="col-md-2"></div>
```

This code is for parent profile setup

initialProfileSetup.html | manager.findTutor.html | searchTutor.html | tutor.profileSetup.html | student.profileSetup.html | manager.profileSetup.html

```
<div class="col-md-2"></div>

<div class="col-md-8">
    <form ng-submit="completeProfile()">
        <div layout="row">
            <md-input-container flex="33">
                <label>First Name</label>
                <input type="text" ng-model="manager.firstName" required>
            </md-input-container>
            <md-input-container flex="33">
                <label>Last Name</label>
                <input type="text" ng-model="manager.lastName" required>
            </md-input-container>
            <md-input-container flex="33">
                <label>Date of Birth</label>
                <md-datepicker md-placeholder="Date of Birth"
                               ng-model="manager.dateOfBirth" md-min-date="minDate" md-max-date="maxDate"
                               required></md-datepicker>
            </md-input-container>
        </div>
        <div layout="row">
            <md-input-container flex="100">
                <label>Street Address</label>
                <input type="text" ng-model="manager.streetAddress" required>
            </md-input-container>


        </div>
        <div layout="row">

            <md-input-container flex="25">
                <select class="form-control" ng-model="manager.country" pvp-country-picker="name" required>
                    <option value="">Select country</option>
                </select>
            </md-input-container>
            <md-input-container flex="25">
                <select class="form-control" ng-model="manager.state"  required>
                    <option value="">Select state</option>
                    <option value="{{s}}" ng-repeat="s in states">{{s}}</option>
                </select>
            </md-input-container>
            <md-input-container flex="25">
                <label>City</label>
                <input type="text" ng-model="manager.city" required>
            </md-input-container>
            <md-input-container flex="25">
                <label>Postal Code</label>
                <input type="text" ng-model="manager.postalCode" minlength="5" maxlength="10" required>
            </md-input-container>
        </div>
        <md-button type="submit" class="md-raised md-primary" style="...">complete parent</md-button>
    </form>
</div>

<div class="col-md-2"></div>
```

This code is for tutor profile setup

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

initialProfileSetup.html | manager.findTutor.html | searchTutor.html | tutor.profileSetup.html

```
<div class="col-md-2"></div>

<div class="col-md-8">
    <form ng-submit="complete()">
        <div layout="row">
            <md-input-container flex="33">
                <label>First Name</label>
                <input type="text" ng-model="tutor.firstName" required>
            </md-input-container>
            <md-input-container flex="33">
                <label>Last Name</label>
                <input type="text" ng-model="tutor.lastName" required>
            </md-input-container>
            <md-input-container flex="33">
                <label>Date of Birth</label>
                <md-datepicker md-placeholder="Date of Birth"
                               ng-model="tutor.dateOfBirth" md-min-date="minDate" md-max-date="maxDate"
                               required></md-datepicker>
            </md-input-container>
        </div>
        <div layout="row">
            <md-input-container flex="100">
                <label>Street Address</label>
                <input type="text" ng-model="tutor.streetAddress" required>
            </md-input-container>


        </div>
        <div layout="row">

            <md-input-container flex="25">
                <select class="form-control" ng-model="tutor.country" pvp-country-picker="name" required>
                    <option value="">Select country</option>
                </select>
            </md-input-container>
            <md-input-container flex="25">
                <select class="form-control" ng-model="tutor.state"  required>
                    <option value="">Select state</option>
                    <option value="{{s}}" ng-repeat="s in states">{{s}}</option>
                </select>
            </md-input-container>
            <md-input-container flex="25">
                <label>City</label>
                <input type="text" ng-model="tutor.city" required>
            </md-input-container>
            <md-input-container flex="25">
                <label>Postal Code</label>
                <input type="text" ng-model="tutor.postalCode" minlength="5" maxlength="10" required>
            </md-input-container>
        </div>

        <div class="row">
            <div>
                <div class="form-group label-floating">
                    <input type="number" class="form-control" placeholder="Years of Experience" min="0" max="70"
                           ng-model="tutor.expYears" required>
                </div>
            </div>
        </div>
```

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

initialProfileSetup.html | manager.findTutor.html | searchTutor.html | tutor.profileSetup.html

```
        <h4>My Schedule</h4>
        <table class="table table-bordered">
            <thead>
            <tr>
                <th>select</th>
                <th>Day</th>
                <th>Time </th>
            </tr>
            </thead>
            <tbody>
            <tr ng-repeat="x in schedules">
                <td ><input type="checkbox" ng-model="x.selected"></td>
                <td ng-model="x.day">{{x.day}}</td>
                <td class="form-inline"><input class="form-control" type="time" ng-model="x.start" >

                </input>
                    <input class="form-control" type="time" ng-model="x.end">
                    </input>
                </td>
            </tr>
            </tbody>
        </table>

        <br>
        <h5>Setup Tutor Expertise

        </h5>
        <div class="row">
            <table class="table">
                <tbody>
                <tr ng-repeat="e in tutor.expertise">

                    <td><p style="..."><i class="fa fa-trash-o" aria-hidden="true"
                                                                ng-click="remove(e)"></i></p></td>
                    <td>{{e.field}}</td>
                    <td>{{e.subject}}</td>
                    <td>{{e.description}}</td>
                </tr>
                </tbody>
            </table>
        </div>

        <div class="row">
            <div class="col-md-3 form-group">
                <select class="form-control" ng-model="selectedField" ng-change="getFieldSubjects(selectedField.field)"
                        ng-options="x.field for x in expertise.fields">
                    <option value="">select subject field</option>
                </select>
            </div>
            <div class="col-md-3 form-group">
                <select class="form-control" ng-model="selectedSubject" ng-options="x.name for x in expertise.subjects">
                    <option value="">select subject name</option>
                </select>
            </div>
            <div class="col-md-6 form-group">
                <input type="text" class="form-control" placeholder="{{selectedSubject.description}}" disabled>
            </div>
        </div>
        <div class="form-group">
```

This code if for editing student and parent profile

student.editProfile.html

```
    <a ui-sref="app.account.profile.list">
        <h5> &#60;&#60; Profiles</h5>
    </a>
</div>

<form>
    <div class="row">
        <div class="col-md-6">
            <div class="form-group label-floating">
                <input type="text" class="form-control" placeholder="First Name">
            </div>
        </div>
        <div class="col-md-6">
            <div class="form-group label-floating">
                <input type="text" class="form-control" placeholder="Last Name">
            </div>
        </div>
    </div>
    <div class="row">
        <div class="col-md-12">
            <div class="form-group label-floating">
                <input type="text" class="form-control" placeholder="Email">
            </div>
        </div>
    </div>
    <div class="row">
        <div class="col-md-12">
            <div class="form-group label-floating">
                <input type="text" class="form-control" placeholder="Address">
            </div>
        </div>
    </div>
    <div class="row">
        <div class="col-md-3">
            <div class="form-group label-floating">
                <input type="text" class="form-control" placeholder="City">
            </div>
        </div>
        <div class="col-md-3">
            <div class="form-group label-floating">
                <input type="text" class="form-control" placeholder="State/Province">
            </div>
        </div>
        <div class="col-md-3">
```

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

manager.editProfile.html

```
<div>
    <a ui-sref="app.account.profile.list">
        <h5> &#60;&#60; Profiles</h5>
    </a>
</div>

<form ng-submit="update()">
    <div layout="row">
        <md-input-container flex="50">
            <label>First Name</label>
            <input type="text" ng-model="manager.firstName" required>
        </md-input-container>
        <md-input-container flex="50">
        <md-input-container flex="50">
            <label>Last Name</label>
            <input type="text" ng-model="manager.lastName" required>
        </md-input-container>
    </div>
    <div layout="row">
        <md-input-container flex="25">
            <label>Street Address</label>
            <input type="text" ng-model="manager.streetAddress" required>
        </md-input-container>
        <md-input-container flex="25">
            <label>City</label>
            <input type="text" ng-model="manager.city" required>
        </md-input-container>
        <md-input-container flex="25">
            <select class="form-control" ng-model="manager.state" required>
                <option value="">Select state</option>
                <option value="{{s}}" ng-repeat="s in states">{{s}}</option>
            </select>
        </md-input-container>
        <md-input-container flex="25">
            <label>Postal Code</label>
            <input type="text" ng-model="manager.postalCode" minlength="5" maxlength="5" required>
        </md-input-container>
    </div>
    <div layout="row">

        <md-input-container>
            <select class="form-control" ng-model="manager.country" pvp-country-picker="name" required>
                <option value="">Select country</option>
            </select>
        </md-input-container>

    </div>
    <button type="submit" class="btn btn-secondary pull-right" data-background-color="red">Update Profile</button>
</form>
```

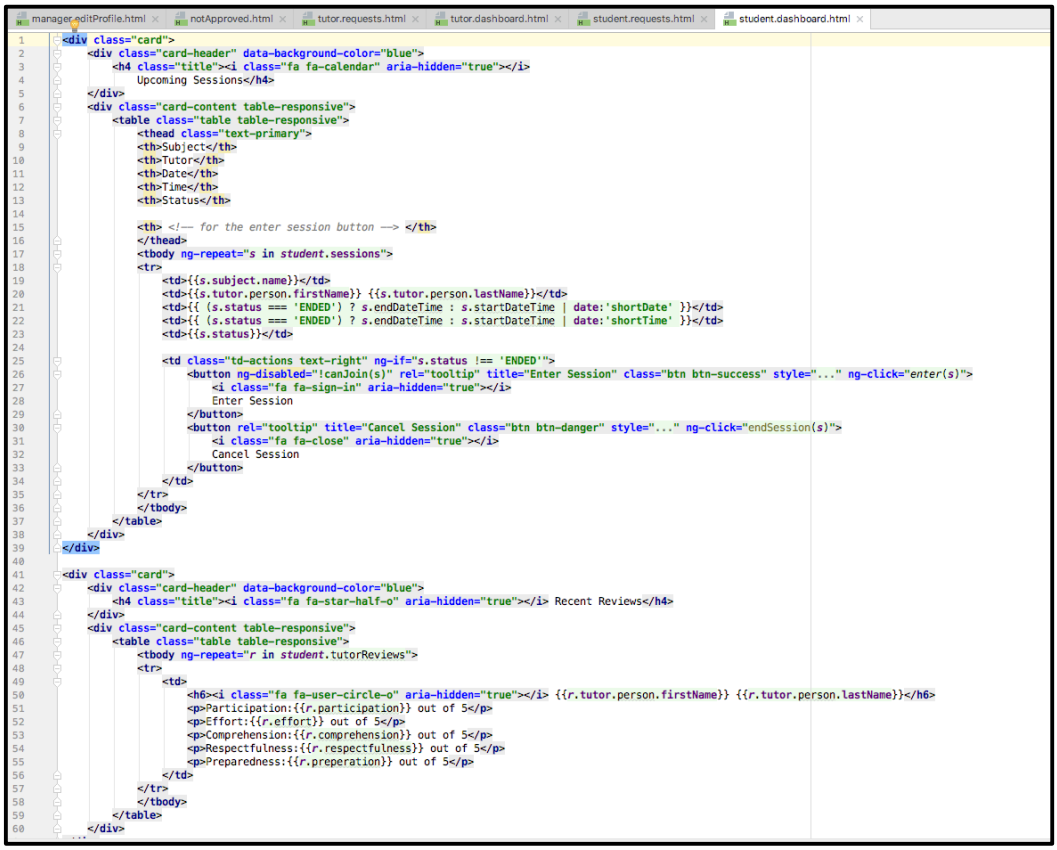
This code is for student public profile

student.publicProfile.html

```
<div>
    <a ui-sref="app.account.profile.list">
        <h5> &#60;&#60; Profiles</h5>
    </a>
</div>

<div class="card card-profile">
    <div class="card-avatar">
        <a href="#pablo">
            <img class="img" src="https://image.freepik.com/free-icon/user-profile-icon_318-33925.jpg"/>
        </a>
    </div>
    <div class="content">
        <h6 class="category text-gray">Student</h6>
        <h4 class="card-title">{{student.firstName}} {{student.lastName}} </h4>
        <div class="card-content">
            <form>
                <div class="row">
                    <div class="col-md-12">
                        <div class="form-group label-floating">
                            <label>gradeLevel:{{student.gradeLevel}}</label>
                        </div>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-3">
                        <div class="form-group label-floating">
                            <label>City: {{student.city}}</label>
                        </div>
                    </div>
                    <div class="col-md-3">
                        <div class="form-group label-floating">
                            <label>State: {{student.state}}</label>
                        </div>
                    </div>
                    <div class="col-md-3">
                        <div class="form-group label-floating">
                            <label>Country: {{student.country}}</label>
                        </div>
                    </div>
                </div>
            </form>
        </div>
        <div>
```

This code is for student dashboard

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design




```
<div class="card">
    <div class="card-header" data-background-color="blue">
        <h4 class="title"><i class="fa fa-calendar" aria-hidden="true"></i>
            Upcoming Sessions</h4>
    </div>
    <div class="card-content table-responsive">
        <table class="table table-responsive">
            <thead class="text-primary">
            <th>Subject</th>
            <th>Tutor</th>
            <th>Date</th>
            <th>Time</th>
            <th>Status</th>

            <th> <!-- for the enter session button --> </th>
            </thead>
            <tbody ng-repeat="s in student.sessions">
            <tr>
                <td>{{s.subject.name}}</td>
                <td>{{s.tutor.person.firstName}} {{s.tutor.person.lastName}}</td>
                <td>{{ (s.status === 'ENDED') ? s.endDateTime : s.startDateTime | date:'shortDate' }}</td>
                <td>{{ (s.status === 'ENDED') ? s.endDateTime : s.startDateTime | date:'shortTime' }}</td>
                <td>{{s.status}}</td>

                <td class="td-actions text-right" ng-if="s.status !== 'ENDED'">
                    <button ng-disabled="!canJoin(s)" rel="tooltip" title="Enter Session" class="btn btn-success" style="..." ng-click="enter(s)">
                        <i class="fa fa-sign-in" aria-hidden="true"></i>
                        Enter Session
                    </button>
                    <button rel="tooltip" title="Cancel Session" class="btn btn-danger" style="..." ng-click="endSession(s)">
                        <i class="fa fa-close" aria-hidden="true"></i>
                        Cancel Session
                    </button>
                </td>
            </tr>
            </tbody>
        </table>
    </div>
</div>

<div class="card">
    <div class="card-header" data-background-color="blue">
        <h4 class="title"><i class="fa fa-star-half-o" aria-hidden="true"></i> Recent Reviews</h4>
    </div>
    <div class="card-content table-responsive">
        <table class="table table-responsive">
            <tbody ng-repeat="r in student.tutorReviews">
            <tr>
                <td>
                    <h6><i class="fa fa-user-circle-o" aria-hidden="true"></i> {{r.tutor.person.firstName}} {{r.tutor.person.lastName}}</h6>
                    <p>Participation:{{r.participation}} out of 5</p>
                    <p>Effort:{{r.effort}} out of 5</p>
                    <p>Comprehension:{{r.comprehension}} out of 5</p>
                    <p>Respectfulness:{{r.respectfulness}} out of 5</p>
                    <p>Preparedness:{{r.preperation}} out of 5</p>
                </td>
            </tr>
            </tbody>
        </table>
    </div>
```

This code is for the review section of tutor dashboard

```
<div class="card">
    <div class="card-header" data-background-color="blue">
        <h4 class="title"><i class="fa fa-star-half-o" aria-hidden="true"></i> Recent Reviews</h4>
    </div>
    <div class="card-content table-responsive">
        <table class="table table-responsive">
            <tbody ng-repeat="r in tutor.tutorReviews">
            <tr>
                <td>
                    <h6><i class="fa fa-user-circle-o" aria-hidden="true"></i> {{r.student.person.firstName}} {{r.student.person.lastName}}</h6>
                    <p>Knowledge:{{r.knowledge}} out of 5</p>
                    <p>Helpfulness:{{r.helpful}} out of 5</p>
                    <p>Preparedness:{{r.prepared}} out of 5</p>
                </td>
            </tr>
            </tbody>
        </table>
    </div>
</div>
```

This code is for managing students on the manager dashboard

```
<!--Manager dashboard-->
<th> <!-- for the enter session button --> </th>
</thead>
<tbody ng-repeat="s in [].concat.apply([], manager.sessions)">
<tr>
    <td>{{s.student.person.firstName}} {{s.student.person.lastName}}</td>
    <td>{{s.subject.name}}</td>
    <td>{{s.tutor.person.firstName}} {{s.tutor.person.lastName}}</td>
    <td>{{ (s.status === 'ENDED') ? s.endDateTime : s.startDateTime | date:'shortDate' }}</td>
    <td>{{ (s.status === 'ENDED') ? s.endDateTime : s.startDateTime | date:'shortTime' }}</td>
    <td>{{s.status}}</td>
```

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

```
        <md-button type="button" class="md-raised md-primary" style="..." ng-click="addForm()">Add Student</md-button>
    </div>
<br><br>


        <div ng-if="!show">
            <div class="col-md-2"></div>
            <form ng-submit="completeProfile()" class="col-md-8">
                <div layout="row">
                    <md-input-container flex="33">
                        <label>First Name</label>
                        <input type="text" ng-model="student.firstName" required>
                    </md-input-container>
                    <md-input-container flex="33">
                        <label>Last Name</label>
                        <input type="text" ng-model="student.lastName" required>
                    </md-input-container>
                   <md-input-container flex="33">
                       <label>Date Of Birth</label>
                       <md-datepicker ng-model="student.dateOfBirth" md-min-date="minDate" md-max-date="maxDate" required></md-datepicker>
                   </md-input-container>
                </div>
                <div layout="row">
                    <md-input-container flex="33">
                        <label>Street Address</label>
                        <input type="text" ng-model="student.streetAddress" required>
                    </md-input-container>
                    <md-input-container flex="33">
                        <label>City</label>
                        <input type="text" ng-model="student.city" required>
                    </md-input-container>

                    <md-input-container flex="33">
                        <label>Postal Code</label>
                        <input type="text" ng-model="student.postalCode" minlength="5" maxlength="10" required>
                    </md-input-container>
                </div>
                <div layout="row">
                    <md-input-container flex="50">
                        <select class="form-control" ng-model="student.country" pvp-country-picker="name" required>
                            <option value="">Select country</option>
                        </select>
                    </md-input-container>
                    <md-input-container flex="50">
                        <select class="form-control" ng-model="student.state"  required>
                            <option value="">Select state</option>
                            <option value="{{s}}" ng-repeat="s in states">{{s}}</option>
                        </select>
                    </md-input-container>

                </div>
                <div layout="row">
                    <md-input-container>
                        <md-button type="submit" class="md-raised">add student</md-button>
                    </md-input-container>
                    <md-input-container>
                        <md-button type="button" class="md-raised" ng-click="cancel()">Cancel</md-button>
                    </md-input-container>
                </div>
```

This code is for displaying student session requests

```
<div class="card">
    <div class="card-header" data-background-color="blue">
        <h4 class="title"><i class="fa fa-calendar" aria-hidden="true"></i>
         My Requests</h4>
    </div>
    <div class="card-content table-responsive">
        <h4>{{errorMsg}}</h4>
        <table class="table table-responsive">
            <thead class="text-primary">
            <th>Request Submission Date</th>
            <th>Tutor Username</th>
            <th>Subject</th>
            <th>Session Time</th>
            <th>Status</th>
            </thead>
            <tbody ng-repeat="sr in studentSessionRequests">
            <tr>
                <td>{{sr.requestDate}}</td>
                <td>{{sr.tutorUsername}}</td>
                <td>{{sr.subject}}</td>
                <td>{{sr.time}}</td>
                <td>{{sr.status}}</td>
            </tr>
            </tbody>
        </table>
    </div>
</div>
```

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

This code is for displaying tutor session requests

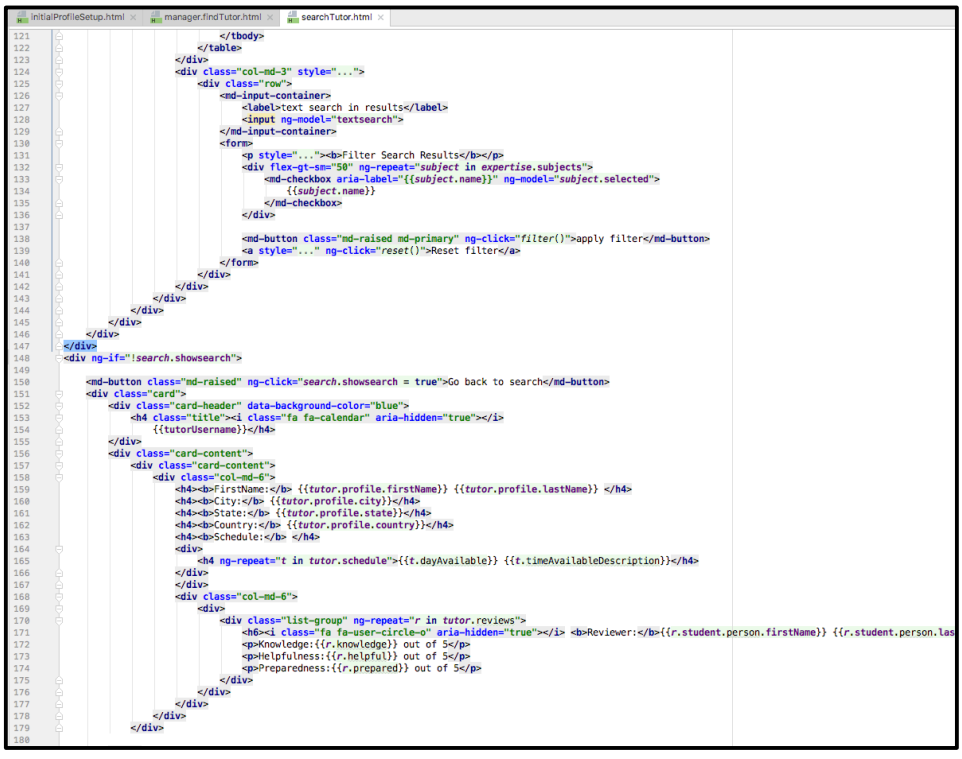
```
<div class="card">
    <div class="card-header" data-background-color="blue">
        <h4 class="title"><i class="fa fa-calendar" aria-hidden="true"></i>
            Responded Requests</h4>
    </div>
    <div class="card-content table-responsive">
        <h4>{{errorMsg}}</h4>
        <table class="table table-responsive">
            <thead class="text-primary">
                <th>Request Submission Date</th>
                <th>Student Username</th>
                <th>Subject</th>
                <th>Session Time</th>
            <th>Status</th>
            </thead>
            <tbody ng-repeat="sr in tutorSessionRequests.req">
                <tr>
                    <td>{{sr.requestDate}}</td>
                    <td>{{sr.studentUsername}}</td>
                    <td>{{sr.subject}}</td>
                    <td>{{sr.time}}</td>
                    <td>{{sr.status}}</td>
                </tr>
            </tbody>
        </table>
    </div>
</div>
```

This is code for searching and finding a tutor for both student and parent

initialProfileSetup.html | manager.findTutor.html | searchTutor.html

```
<div ng-show="search.showsearch">

    <div class="card">
        <div class="card-content">
            <div class="row">
                <div class="col-md-3">
                    <div class="form-group label-floating">
                        <select class="form-control" ng-model="searchBy" ng-options="x for x in searchCriteria"
                                required>
                            <option value="">search by</option>
                        </select>
                    </div>
                </div>
                <form ng-if="searchBy == 'subject'" ng-submit="searchBySubject()">
                    <div>
                        <div class="col-md-3 form-group label-floating">
                            <select class="form-control" ng-model="search.subject"
                                    ng-options="x.name for x in expertise.subjects" required>
                                <option></option>
                            </select>
                        </div>
                        <div class="col-md-3">
                            <div layout-gt-xs="row">
                                <div class="form-group">
                                    <button type="submit" class="btn btn-success" data-background-color="green">
                                        Search
                                    </button>
                                </div>
                            </div>
                        </div>
                    </div>
                </form>
                <form ng-if="searchBy == 'tutor name'" ng-submit="searchByName()">
                    <div class="col-md-3">
                        <div class="form-group label-floating">
                            <input type="text" class="form-control" placeholder="tutor first name"
                                   ng-model="search.tutorFirstName">
                        </div>
                    </div>
                    <div class="col-md-3">
                        <div class="form-group label-floating">
                            <input type="text" class="form-control" placeholder="tutor last name"
                                   ng-model="search.tutorLastName">
                        </div>
                    </div>
                    <div class="col-md-3">
                        <div layout-gt-xs="row">
                            <div class="form-group">
                                <button type="submit" class="btn btn-success" data-background-color="green">Search
                                </button>
                            </div>
                        </div>
                    </div>
                </form>
                <form ng-if="searchBy == 'tutor username'" ng-submit="searchByTutor()">
                    <div class="col-md-3">
                        <div class="form-group label-floating">
                            <input type="text" class="form-control" placeholder="tutor username"
                                   ng-model="search.tutorUsername" required>
                        </div>
```


```
                    <div class="col-md-3">
                        <div layout-gt-xs="row">
                            <div class="form-group">
                                <button type="submit" class="btn btn-success" data-background-color="green">Search
                                </button>
                            </div>
                        </div>
                    </div>
                </form>
            </div>

            <div ng-if="!hide">
                <div class="row">
                    <div class="col-md-9">
                        <h6><b>Available Now!</b></h6>
                        <table class="table table-striped table-responsive">
                            <thead>
                            <tr>
                                <th><i class="fa fa-sort" aria-hidden="true"></i> Username</th>
                                <th><i class="fa fa-sort" aria-hidden="true"></i> YearsOfExperience</th>
                                <th><i class="fa fa-sort" aria-hidden="true"></i> State</th>
                                <th><i class="fa fa-sort" aria-hidden="true"></i> Country</th>
                                <th><i class="fa fa-sort" aria-hidden="true"></i> Rating</th>

                            </tr>
                            </thead>
                            <tbody ng-repeat="r in filters.availables | filter:textsearch" ng-click="goToTutorPage(r)"
                                   style="...">
                            <tr>
                                <td>{{r.username}}</td>
                                <td>{{r.expYears}}</td>
                                <td>{{r.state}}</td>
                                <td>{{r.country}}</td>
                                <td><i class="fa fa-star-o" aria-hidden="true"></i></td>
                            </tr>
                            </tbody>
                        </table>
                        <br>
                        <h6><b>Tutor Results</b></h6>
                        <table class="table table-striped table-responsive">
                            <thead>
                            <tr>
                                <th><i class="fa fa-sort" aria-hidden="true"></i> Username</th>
                                <th><i class="fa fa-sort" aria-hidden="true"></i> YearsOfExperience</th>
                                <th><i class="fa fa-sort" aria-hidden="true"></i> State</th>
                                <th><i class="fa fa-sort" aria-hidden="true"></i> Country</th>
                                <th><i class="fa fa-sort" aria-hidden="true"></i> Rating</th>

                            </tr>
                            </thead>
                            <tbody ng-repeat="r in filters.results | filter:textsearch" ng-click="goToTutorPage(r)"
                                   style="...">
                            <tr>
                                <td>{{r.username}}</td>
                                <td>{{r.expYears}}</td>
                                <td>{{r.state}}</td>
                                <td>{{r.country}}</td>
                                <td><i class="fa fa-star-o" aria-hidden="true"></i></td>
                            </tr>
                            </tbody>
                        </table>
```




```
                            </tbody>
                        </table>
                    </div>
                    <div class="col-md-3" style="...">
                        <div class="row">
                            <md-input-container>
                                <label>text search in results</label>
                                <input ng-model="textsearch">
                            </md-input-container>
                            <form>
                                <p style="..."><b>Filter Search Results</b></p>
                                <div flex-gt-sm="50" ng-repeat="subject in expertise.subjects">
                                    <md-checkbox aria-label="{{subject.name}}" ng-model="subject.selected">
                                        {{subject.name}}
                                    </md-checkbox>
                                </div>

                                <md-button class="md-raised md-primary" ng-click="filter()">apply filter</md-button>
                                <a style="..." ng-click="reset()">Reset filter</a>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
<div ng-if="!search.showsearch">

    <md-button class="md-raised" ng-click="search.showsearch = true">Go back to search</md-button>
    <div class="card">
        <div class="card-header" data-background-color="blue">
            <h4 class="title"><i class="fa fa-calendar" aria-hidden="true"></i>
                {{tutorUsername}}</h4>
        </div>
        <div class="card-content">
            <div class="card-content">
                <div class="col-md-6">
                    <h4><b>FirstName:</b> {{tutor.profile.firstName}} {{tutor.profile.lastName}} </h4>
                    <h4><b>City:</b> {{tutor.profile.city}}</h4>
                    <h4><b>State:</b> {{tutor.profile.state}}</h4>
                    <h4><b>Country:</b> {{tutor.profile.country}}</h4>
                    <h4><b>Schedule:</b> </h4>
                    <div>
                        <h4 ng-repeat="t in tutor.schedule">{{t.dayAvailable}} {{t.timeAvailableDescription}}</h4>
                    </div>
                    </div>
                    <div class="col-md-6">
                        <div>
                            <div class="list-group" ng-repeat="r in tutor.reviews">
                                <h6><i class="fa fa-user-circle-o" aria-hidden="true"></i> <b>Reviewer:</b>{{r.student.person.firstName}} {{r.student.person.las
                                <p>Knowledge:{{r.knowledge}} out of 5</p>
                                <p>Helpfulness:{{r.helpful}} out of 5</p>
                                <p>Preparedness:{{r.prepared}} out of 5</p>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
```

initialProfileSetup.html | manager.findTutor.html | searchTutor.html

```
                    </div>
                </div>
            </div>

        </div>
        <div>
            <div class="card">
                <div class="card-content">
                    <form ng-submit="register()">
                        <div class="form-group label-floating">
                            <div class="col-md-3">
                                <div class="form-group label-floating">
                                    <div>
                                        <select class="form-control" ng-model="reg.subject"
                                                ng-options="x.name for x in tutor.subjects" required>
                                            <option value="">select subject</option>
                                        </select>
                                    </div>
                                </div>
                            </div>
                            <div class="col-md-3">
                                <div layout-gt-xs="row">
                                    <div class="form-group">
                                        <div flex-gt-xs>
                                            <md-datepicker md-placeholder="date"
                                                           ng-model="reg.date" required></md-datepicker>
                                        </div>
                                    </div>
                                </div>
                            </div>
                            <div class="col-md-3">
                                <div class="form-group label-floating">
                                    <div>
                                        <select class="form-control" ng-model="reg.time"
                                                ng-options="x.name for x in time" required>
                                            <option value="">select time</option>
                                        </select>
                                    </div>
                                </div>
                            </div>
                            <div class="col-md-3">
                                <div layout-gt-xs="row">
                                    <div class="form-group">
                                        <button type="submit" class="btn btn-success" data-background-color="green">
                                            Register
                                        </button>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
```

initialProfileSetup.html | manager.findTutor.html

```
<div ng-show="search.showsearch">

    <div class="card">
        <div class="card-content">
            <div class="row">
                <div class="col-md-3">
                    <div class="form-group label-floating">
                        <select class="form-control" ng-model="searchBy" ng-options="x for x in searchCriteria"
                                required>
                            <option value="">search by</option>
                        </select>
                    </div>
                </div>
                <form ng-if="searchBy == 'subject'" ng-submit="searchBySubject()">
                    <div>
                        <div class="col-md-3 form-group label-floating">
                            <select class="form-control" ng-model="search.subject"
                                    ng-options="x.name for x in expertise.subjects" required>
                                <option></option>
                            </select>
                        </div>
                        <div class="col-md-3">
                            <div layout-gt-xs="row">
                                <div class="form-group">
                                    <button type="submit" class="btn btn-success" data-background-color="green">
                                        Search
                                    </button>
                                </div>
                            </div>
                        </div>
                    </div>
                </form>
                <form ng-if="searchBy == 'tutor name'" ng-submit="searchByName()">
                    <div class="col-md-3">
                        <div class="form-group label-floating">
                            <input type="text" class="form-control" placeholder="tutor first name"
                                   ng-model="search.tutorFirstName">
                        </div>
                    </div>
                    <div class="col-md-3">
                        <div class="form-group label-floating">
                            <input type="text" class="form-control" placeholder="tutor last name"
                                   ng-model="search.tutorLastName">
                        </div>
                    </div>
                    <div class="col-md-3">
                        <div layout-gt-xs="row">
                            <div class="form-group">
                                <button type="submit" class="btn btn-success" data-background-color="green">Search
                                </button>
                            </div>
                        </div>
                    </div>
                </form>
                <form ng-if="searchBy == 'tutor username'" ng-submit="searchByTutor()">
                    <div class="col-md-3">
                        <div class="form-group label-floating">
                            <input type="text" class="form-control" placeholder="tutor username"
                                   ng-model="search.tutorUsername" required>
                        </div>
```

initialProfileSetup.html | manager.findTutor.html

```
<div ng-if="!hide">
    <div class="row">
        <div class="col-md-9">
            <h6><b>Available Now!</b></h6>
            <table class="table table-striped table-responsive">
                <thead>
                <tr>
                    <th><i class="fa fa-sort" aria-hidden="true"></i> Username</th>
                    <th><i class="fa fa-sort" aria-hidden="true"></i> YearsOfExperience</th>
                    <th><i class="fa fa-sort" aria-hidden="true"></i> State</th>
                    <th><i class="fa fa-sort" aria-hidden="true"></i> Country</th>
                    <th><i class="fa fa-sort" aria-hidden="true"></i> Rating</th>

                </tr>
                </thead>
                <tbody ng-repeat="r in filters.availables | filter:textsearch" ng-click="goToTutorPage(r)"
                       style="...">
                <tr>
                    <td>{{r.username}}</td>
                    <td>{{r.expYears}}</td>
                    <td>{{r.state}}</td>
                    <td>{{r.country}}</td>
                    <td><i class="fa fa-star-o" aria-hidden="true"></i></td>
                </tr>
                </tbody>
            </table>
            <br>
            <h6><b>Tutor Results</b></h6>
            <table class="table table-striped table-responsive">
                <thead>
                <tr>
                    <th><i class="fa fa-sort" aria-hidden="true"></i> Username</th>
                    <th><i class="fa fa-sort" aria-hidden="true"></i> YearsOfExperience</th>
                    <th><i class="fa fa-sort" aria-hidden="true"></i> State</th>
                    <th><i class="fa fa-sort" aria-hidden="true"></i> Country</th>
                    <th><i class="fa fa-sort" aria-hidden="true"></i> Rating</th>

                </tr>
                </thead>
                <tbody ng-repeat="r in filters.results | filter:textsearch" ng-click="goToTutorPage(r)"
                       style="...">
                <tr>
                    <td>{{r.username}}</td>
                    <td>{{r.expYears}}</td>
                    <td>{{r.state}}</td>
                    <td>{{r.country}}</td>
                    <td><i class="fa fa-star-o" aria-hidden="true"></i></td>
                </tr>
                </tbody>
            </table>
        </div>
        <div class="col-md-3" style="...">
            <div class="row">
                <md-input-container>
                    <label>text search in results</label>
                    <input ng-model="textsearch">
                </md-input-container>
                <form>
                    <p style="..."><b>Filter Search Results</b></p>
```

initialProfileSetup.html | manager.findTutor.html

```
                </form>
            </div>
        </div>
    </div>
</div>
</div>
</div>
</div>
<div ng-if="!search.showsearch">

    <md-button class="md-raised" ng-click="search.showsearch = true">Go back to search</md-button>
    <div class="card">
        <div class="card-header" data-background-color="blue">
            <h4 class="title"><i class="fa fa-calendar" aria-hidden="true"></i>
                {{tutorUsername}}</h4>
        </div>
        <div class="card-content">
            <div class="card-content">
                <div class="col-md-5">
                    <h4><b>Name:</b> {{tutor.profile.firstName}} {{tutor.profile.lastName}} </h4>
                    <h4><b>City:</b> {{tutor.profile.city}}</h4>
                    <h4><b>State:</b> {{tutor.profile.state}}</h4>
                    <h4><b>Country:</b> {{tutor.profile.country}}</h4>
                    <h4><b>Schedule:</b> </h4>
                    <div>
                        <h4 ng-repeat="t in tutor.schedule">{{t.dayAvailable}} {{t.timeAvailableDescription}}</h4>
                    </div>
                </div>
                <div class="col-md-7">
                    <div>
                        <div class="list-group">
                            <div>
                                <div class="list-group" ng-repeat="r in tutor.reviews">
                                    <h6><i class="fa fa-user-circle-o" aria-hidden="true"></i> <b>Reviewer:</b>{{r.student.person.firstName}} {{r.student.person.l
                                    <p>Knowledge:{{r.knowledge}} out of 5</p>
                                    <p>Helpfulness:{{r.helpful}} out of 5</p>
                                    <p>Preparedness:{{r.prepared}} out of 5</p>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>

    </div>
    <div>
        <div class="card">
            <div class="card-content">
                <form ng-submit="register()">
                    <div layout="row">
                        <md-input-container flex="20">
                            <label>Student</label>
                            <md-select ng-model="reg.student" required>
                                <md-option ng-repeat="s in students" ng-value="s.id">
                                    {{s.person.firstName}} {{s.person.lastName}}
                                </md-option>
                            </md-select>
                        </md-input-container>
                        <md-input-container flex="20">
```

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

initialProfileSetup.html | manager.findTutor.html

```
                        <div>
                            <div class="list-group" ng-repeat="r in tutor.reviews">
                                <h6><i class="fa fa-user-circle-o" aria-hidden="true"></i> <b>Reviewer:</b>{{r.student.person.firstName}} {{r.student.person.las
                                <p>Knowledge:{{r.knowledge}} out of 5</p>
                                <p>Helpfulness:{{r.helpful}} out of 5</p>
                                <p>Preparedness:{{r.prepared}} out of 5</p>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>

    </div>
    <div>
        <div class="card">
            <div class="card-content">
                <form ng-submit="register()">
                    <div layout="row">
                        <md-input-container flex="20">
                            <label>Student</label>
                            <md-select ng-model="reg.student" required>
                                <md-option ng-repeat="s in students" ng-value="s.id">
                                    {{s.person.firstName}} {{s.person.lastName}}
                                </md-option>
                            </md-select>
                        </md-input-container>
                        <md-input-container flex="20">
                            <label>Subject</label>
                            <md-select ng-model="reg.subject" required>
                                <md-option ng-repeat="x in tutor.subjects" ng-value="x.id">{{x.name}}</md-option>
                            </md-select>
                        </md-input-container>
                        <md-input-container flex="25">
                            <md-datepicker md-placeholder="date"
                                           ng-model="reg.date" required></md-datepicker>
                        </md-input-container>
                        <md-input-container flex="20">
                            <label>Time</label>
                            <md-select ng-model="reg.time" required>
                                <md-option ng-value="x" ng-repeat="x in time">{{x.name}}</md-option>
                            </md-select>
                        </md-input-container>
                        <md-input-container flex="15">
                            <md-button type="submit" class="md-raised md-primary">
                                Register
                            </md-button>
                        </md-input-container>
                    </div>
                </form>
            </div>
        </div>
    </div>
    iiv>
```

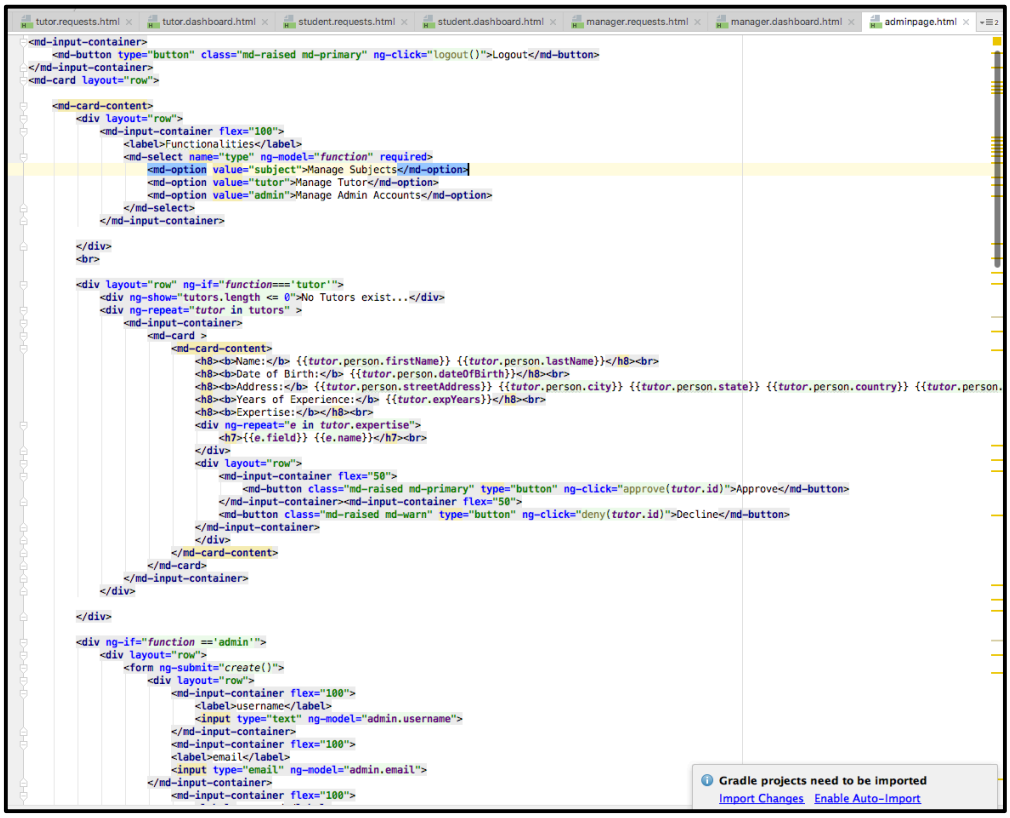
This code is to display when a tutor has not been approved

```
<div class="container-fluid">
    <div class="card text-center">
        <div class="card-block">
            <h3>Your account has not been approved yet. Please contact admin for further details.</h3>
        </div>
    </div>
</div>
```

This code is for Admin Login

```
<div class="col-md-2"></div>
<div class="col-md-8">
    <md-card flex="flex" flex-gt-sm="50" flex-gt-md="33">
        <md-toolbar>
            <div class="md-toolbar-tools">
                <h2><span>Admin Login Form</span></h2>
            </div>
        </md-toolbar>
        <md-card-content>
            <form name="Form" ng-submit="loginAdmin()">
                <md-input-container class="md-block">
                    <label>Login</label>
                    <input type="text" name="username" placeholder="login" ng-model="admin.username" required=""/>
                    <div ng-messages="Form.email.$error" role="alert" multiple="">
                        <div ng-message="required" class="my-message">Please enter your username.</div>
                    </div>
                </md-input-container>
                <md-input-container class="md-block">
                    <label>Password</label>
                    <input type="password" name="password" placeholder="password" ng-model="admin.password" required=""/>
                    <div ng-messages="Form.password.$error" role="alert" multiple="">
                        <div ng-message="required" class="my-message">Please enter your password.</div>
                    </div>
                </md-input-container>
                <md-button ng-disabled="!Form.$valid" type="submit" class="md-raised md-primary">Login</md-button>
            </form>
        </md-card-content>
    </md-card>
</div>
<div class="col-md-2"></div>
```

This code is for the functionalities of an Admin


tutor.requests.html
tutor.dashboard.html
student.requests.html
student.dashboard.html
manager.requests.html
manager.dashboard.html
adminpage.html
<md-input-container>
<md-button type="button" class="md-raised md-primary" ng-click="logout()">Logout</md-button>
</md-input-container>
<md-card layout="row">
<md-card-content>

<md-input-container flex="100">
<label>Functionalities</label>
<md-select name="type" ng-model="function" required>
<md-option value="subject">Manage Subjects</md-option>
<md-option value="tutor">Manage Tutor</md-option>
<md-option value="admin">Manage Admin Accounts</md-option>
</md-select>
</md-input-container>

<div ng-show="tutors.length <= 0">No Tutors exist...

<md-input-container>
<md-card >
<md-card-content>
<h8>Name: {{tutor.person.firstName}} {{tutor.person.lastName}}</h8>
<h8>Date of Birth: {{tutor.person.dateOfBirth}}</h8>
<h8>Address: {{tutor.person.streetAddress}} {{tutor.person.city}} {{tutor.person.state}} {{tutor.person.country}} {{tutor.person.
<h8>Years of Experience: {{tutor.expYears}}</h8>
<h8>Expertise:</h8>

<h7>{{e.field}} {{e.name}}</h7>

<md-input-container flex="50">
<md-button class="md-raised md-primary" type="button" ng-click="approve(tutor.id)">Approve</md-button>
</md-input-container><md-input-container flex="50">
<md-button class="md-raised md-warn" type="button" ng-click="deny(tutor.id)">Decline</md-button>
</md-input-container>

</md-card-content>
</md-card>
</md-input-container>

<form ng-submit="create()">

<md-input-container flex="100">
<label>username</label>
<input type="text" ng-model="admin.username">
</md-input-container>
<md-input-container flex="100">
<label>email</label>
<input type="email" ng-model="admin.email">
</md-input-container>
<md-input-container flex="100">
Gradle projects need to be imported
Import Changes Enable Auto-Import


tutor.requests.html
tutor.dashboard.html
student.requests.html
student.dashboard.html
manager.requests.html
manager.dashboard.html
adminpage.html
<label>password</label>
<input type="password" ng-model="admin.password"minlength="8">
</md-input-container>
<md-input-container flex="100">
<md-button class="md-raised md-primary" type="submit">create admin</md-button>
</md-input-container>

</form>

<form ng-submit="delete(selectedAdmin)">

<md-input-container>
<label>Select an admin to edit</label>
<md-select name="type" ng-model="selectedAdmin" required>
<md-option ng-value="a.id" ng-repeat="a in admins">{{a.username}}</md-option>
</md-select>
</md-input-container>
<md-input-container >
<md-button class="md-raised md-primary" type="submit">delete admin</md-button>
</md-input-container>

</form>

<form ng-submit="createSubject()">

<md-input-container flex="100">
<md-select ng-model="sub.field" required>
<label>Field</label>
<md-option ng-value="'MATH'">Math</md-option>
<md-option ng-value="'SCIENCE'">Science</md-option>
<md-option ng-value="'LANGUAGE'">Language</md-option>
<md-option ng-value="'ART'">Art</md-option>
<md-option ng-value="'SOCIAL'">Social</md-option>
<md-option ng-value="'OTHER'">Other</md-option>
</md-select>
</md-input-container>
<md-input-container flex="100">
<label>Name</label>
<input type="text" ng-model="sub.name" required>
</md-input-container>
<md-input-container flex="100">
<label>Description</label>
<input type="text" ng-model="sub.description" required>
</md-input-container>
<md-input-container flex="100">
<md-button class="md-raised md-primary" type="submit">create subject</md-button>
</md-input-container>

</form>
Gradle projects need to be imported
Import Changes Enable Auto-Import

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

tutor.requests.html | tutor.dashboard.html | student.requests.html | student.dashboard.html | manager.requests.html | manager.dashboard.html | adminpage.html

```
        </div>
        <div layout="row">
            <form ng-submit="editSubject(selectedSub.id)">
                <div layout="row">
                    <md-input-container>
                        <label>Select subject to edit</label>
                        <md-select name="type" ng-model="selectedSub" ng-change="toEditSubjectSelected(selectedSub)" required )>
                            <md-option ng-value="s" ng-repeat="s in subjects">{{s.name}}</md-option>
                        </md-select>
                    </md-input-container>
                </div>
                <div layout="row">
                    <md-input-container flex="100">
                        <md-select ng-model="editSub.field" required>
                            <label>Field</label>
                            <md-option ng-value="'MATH'">Math</md-option>
                            <md-option ng-value="'SCIENCE'">Science</md-option>
                            <md-option ng-value="'LANGUAGE'">Language</md-option>
                            <md-option ng-value="'ART'">Art</md-option>
                            <md-option ng-value="'SOCIAL'">Social</md-option>
                            <md-option ng-value="'OTHER'">Other</md-option>
                        </md-select>
                    </md-input-container>
                    <md-input-container flex="100">
                        <label>Name</label>
                        <input type="text" ng-model="editSub.name" required>
                    </md-input-container>
                    <md-input-container flex="100">
                        <label>Description</label>
                        <input type="text" ng-model="editSub.description" required>
                    </md-input-container>
                    <md-input-container flex="100">
                        <md-button class="md-raised md-primary" type="submit">edit subject</md-button>
                    </md-input-container>
                </div>

            </form>

        </div>
        <div layout="row">
            <form ng-submit="deleteSub(deletedSub)">
                <div layout="row">
                    <md-input-container>
                        <label>Select subject to delete</label>
                        <md-select name="type" ng-model="deletedSub" required>
                            <md-option ng-value="s" ng-repeat="s in subjects">{{s.name}}</md-option>
                        </md-select>
                    </md-input-container>
                    <md-input-container >
                        <md-button class="md-raised md-warn" type="submit">delete subject</md-button>
                    </md-input-container>
                </div>

            </form>

        </div>
    </div>
```

Angular JavaScript

This code is for user login. Taking the form data and sending to the authentication endpoint

```
import "./auth.scss";
import template from "./auth.view.pug";


export default {
    controller: function ($scope, authenticationService, $state) {
        "ngInject";

        $scope.login = {
            user: '',
            password: ''
        };

        $scope.signup = {
            user: '',
            email: '',
            confirmEmail: '',
            pass: '',
            confirmPass: ''
        };

        $scope.error = {
            email: '',
            password: ''
        };

        $scope.loginUser = function () {
            let userData = {
                username: $scope.login.user,
                password: $scope.login.password
            };
            login(userData);
        };

        function login(userData) {
            authenticationService.authenticateUser(userData).then(function (response) {
                localLogin(response);
            }, function (err) {
                $scope.login = {};
                alert(err.data.message);
            });
        }

        function localLogin(response) {
            authenticationService.setUserToken(response, false);
            authenticationService.setUserInfo(response.data);
            if (authenticationService.getUserInfo().isSetup) {
                $state.go('app.profileSetup.student');
            }
            else {
                $state.go('app.account.profile.list');
                console.log("go to profileList");
            }
        }

```

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

```
            if (authenticationService.getUserInfo().isSetup) {
                $state.go('app.profileSetup.student');
            }
            else {
                $state.go('app.account.profile.list');
                console.log("go to profileList");
            }
        }

        $scope.emailMsg = '';
        $scope.passMsg = '';
        $scope.signUp = function () {
            if ($scope.signup.email === $scope.signup.confirmEmail && $scope.signup.pass === $scope.signup.confirmPass) {
                let userData = {
                    username: $scope.signup.user,
                    email: $scope.signup.email,
                    password: $scope.signup.pass
                };

                authenticationService.signUp(userData).then(function (response) {
                    alert("You signed up successfully");
                    localLogin(response);
                    $scope.signup = {};
                }, function (response) {
                    if (response.status === 400) {
                        // bad signup
                        alert(response.data.message)
                    }
                    else if (response.status >= 500) {
                        alert("Server error! Try again later.")
                    }
                    else {
                        alert("Signup failed for an unknown reason. Contact an admin.")
                        console.log("error " + response);
                    }
                })
            }
            else {
                if ($scope.signup.email !== $scope.signup.confirmEmail) {
                    $scope.emailMsg = "email";
                    $scope.signup.confirmEmail = '';
                }
                if ($scope.signup.pass !== $scope.signup.confirmPass) {
                    $scope.passMsg = "password";
                    $scope.signup.confirmPass = '';
                }
                alert("The confirm fields for the following do not match: " + $scope.emailMsg + " " + $scope.passMsg);
                $scope.emailMsg = '';
                $scope.passMsg = '';
            }
        };
    },
    template: template
};
```

This code contains the angular configuration and the routes configuration for the web pages

app.js

```
// the global CSS
import ...
// angular
import ...
//js controllers
import ...
//services
import ...
//import layout htmls
import profileLayout from "./content/shared/profileLayout.html";

const app = angular.module("Teach4Service", ["ui.router", "ngMaterial", "ngMessages", "puigcerber.countryPicker"]);
authService.init(app);
app.config(($stateProvider, $urlRouterProvider) => {
    "ngInject";
    $urlRouterProvider.otherwise("/login");

    $stateProvider
        .state("app", {
            template: "<ui-view></ui-view>",
            reload: true,
            abstract: true
        })
        .state("app.auth", {
            url: "/login",
            reload: true,
            controller: authComponent.controller,
            template: authComponent.template,
            requireLogin: false
        })

        .state('app.account', {
            abstract: true,
            reload: true,
            template: mainLayoutComp.template,
            controller: mainLayoutComp.controller,
            requireLogin: true
        })

        .state('app.account.tutor.notapprove', {
            url: "/profile/notapproved",
            reload: true,
            template: notApproveComp.template,
            requireLogin: true
        })

        .state('app.profileSetup', {
            url: '/profilesetup',
            reload: true,
            controller: profileSetup.controller,
            template: profileSetup.template,
            abstract: true,
            requireLogin: true
        })

        .state('app.profileSetup.student', {
            url: '/student',
            reload: true,
            template: studentProfileSetup.template,
            controller: studentProfileSetup.controller,
```

app.js

```
    .state('app.profileSetup.tutor', {
        url: '/tutor',
        reload: true,
        template: tutorProfileSetup.template,
        controller: tutorProfileSetup.controller,
        requireLogin: true
    })

    .state('app.profileSetup.manager', {
        url: '/manager',
        reload: true,
        template: managerProfileSetup.template,
        controller: managerProfileSetup.controller,
        requireLogin: true
    })

    .state('app.account.profile', {
        abstract: true,
        template: profileLayout,
        reload: true,
        requireLogin: true
    })

    .state('app.account.profile.list', {
        url: '/profile',
        controller: profileComp.controller,
        template: profileComp.template,
        reload: true,
        requireLogin: true,
    })


    //STUDENT ROUTES
    .state('app.account.student', {
        url: '/student',
        template: '<ui-view></ui-view>',
        reload: true,
        abstract: true,
        requireLogin: true
    })
    .state('app.account.student.dashboard', {
        url: '/dashboard',
        controller: studentDashboardComp.controller,
        template: studentDashboardComp.template,
        reload: true,
        requireLogin: true
    })
    .state('app.account.profile.studentEdit', {
        url: '/profile/student/edit',
        template: studentEditProfileComp.template,
        controller: studentEditProfileComp.controller,
        reload: true,
        requireLogin: true
    })
    .state('app.account.profile.studentView', {
        url: '/profile/student/publicview',
        template: studentPublicProfile.template,
        controller: studentPublicProfile.controller,
        reload: true,
```

app.js

```
    })
    .state('app.account.student.findTutor', {
        url: '/findtutor',
        template: searchTutorComp.template,
        controller: searchTutorComp.controller,
        reload: true,
        requireLogin: true

    })


    .state('app.account.student.requests', {
        url: '/requests',
        template: studentRequests.template,
        controller: studentRequests.controller,
        reload: true,
        requireLogin: true

    })


    .state('app.account.student.sessionHistory', {
        url: '/sessionhistory',
        controller: studentSessionCtrl.controller,
        template: studentSessionCtrl.template,
        reload: true,
        requireLogin: true
    })

    //TUTOR ROUTES
    .state('app.account.tutor', {
        url: '/tutor',
        template: '<ui-view></ui-view>',
        reload: true,
        abstract: true,
        requireLogin: true
    })
    .state('app.account.tutor.dashboard', {
        url: '/dashboard',
        controller: tutorDashboardCtrl.controller,
        template: tutorDashboardCtrl.template,
        reload: true,
        requireLogin: true
    })
    .state('app.account.tutor.requests', {
        url: '/requests',
        template: tutorRequests.template,
        controller: tutorRequests.controller,
        reload: true,
        requireLogin: true
    })

    .state('app.account.profile.tutorEdit', {
        url: '/profile/tutor/edit',
        reload: true,
        template: tutorEditProfileComp.template,
        controller: tutorEditProfileComp.controller,
        requireLogin: true
    })
    .state('app.account.profile.tutorView', {
```

app.js

```
        })
        .state('app.account.tutor.sessionhistory', {
            url: '/session',
            controller: tutorSessionCtrl.controller,
            template: tutorSessionCtrl.template,
            reload: true,
            requireLogin: true
        })


        //MANAGER ROUTES
        .state('app.account.manager', {
            url: '/manager',
            template: '<ui-view></ui-view>',
            reload: true,
            abstract: true,
            requireLogin: true
        })
        .state('app.account.manager.dashboard', {
            url: '/dashboard',
            controller: managerDashboardComp.controller,
            template: managerDashboardComp.template,
            requireLogin: true
        })
        .state('app.account.profile.managerEdit', {
            url: '/profile/manager/edit',
            template: managerEditProfileComp.template,
            controller: managerEditProfileComp.controller,
            requireLogin: true
        })
        .state('app.account.profile.managerView', {
            url: '/profile/manager/publicview',
            template: managerPublicProfile.template,
            controller: managerPublicProfile.controller,
            requireLogin: true
        })
        .state('app.account.manager.requests', {
            url: '/studentrequests',
            template: managerStudentRequestComp.template,
            controller: managerStudentRequestComp.controller,
            reload: true,
            requireLogin: true
        })

        .state('app.account.manager.session', {
            url: '/session',
            controller: managerSessionCtrl.controller,
            template: managerSessionCtrl.template,
            reload: true,
            requireLogin: true
        })
        .state('app.account.manager.findTutor', {
            url: '/findtutor',
            template: managerFindTutorComp.template,
            controller: managerFindTutorComp.controller,
            requireLogin: true,
            reload: true
        })
```

app.js

```
        //Session Routes
        .state('app.session', {
            url: "/session/{sessionId:int}",
            template: videoChatPageComp.template,
            controller: videoChatPageComp.controller,
            requireLogin: true,
            reload: true
        })
        .state('app.session.tutor', {
            url: "/tutor",
            data: {isTutor: true},
            requireLogin: true,
            reload: true
        })
        .state('app.session.student', {
            url: "/student",
            data: {isTutor: false},
            requireLogin: true,
            reload: true
        })

        //ADMIN ROUTES
        .state('app.admin', {
            url: '/admin',
            template: "<ui-view></ui-view>",
            abstract: true,
            reload: true,
            requireLogin: false
        })

        .state('app.admin.login', {
            url: '/login',
            template: adminLogin.template,
            controller: adminLogin.controller,
            reload: true,
            requireLogin: false
        })

        .state('app.admin.home', {
            url: '/home',
            template: adminPage.template,
            controller: adminPage.controller,
            reload: true,
            requireLogin: true
        })


    /*.state('app.account.admin.admin', {
        url: '/session',
        controller: adminCtrl.controller,
        template: adminCtrl.template,
        reload: true,
        requireLogin: true
    })*/

});
```

app.js

```
            url: '/session',
            controller: adminCtrl.controller,
            template: adminCtrl.template,
            reload: true,
            requireLogin: true
        })*/

});

app.factory('authHttpResponseInterceptor', ['$q', '$location', function ($q, $location) {
    return {
        response: function (response) {
            if (response.status === 401) {
                //console.log("Response 401");
            }
            return response || $q.when(response);
        },
        responseError: function (rejection) {
            if (rejection.status === 401) {
                localStorage.removeItem("_userToken");
                $location.path('/login').search('returnTo', $location.path());
            }
            return $q.reject(rejection);
        }
    }
}])
```

app.js

```
app.run(function ($http, $rootScope, $state, authenticationService) {
    "ngInject"
    $rootScope.$on('$stateChangeStart', function (event, toState, fromState) {
        var requireLogin = toState.requireLogin;
        if (requireLogin) {
            if (authenticationService.getUserToken() == null) {
                event.preventDefault();
                $state.go('app.auth');
            }
            else {
                if (!authenticationService.getUserToken().systemUser) {

                    authenticationService.isApproved().then(function (response) {
                        authenticationService.setApproved(response.data);
                        authenticationService.getSelf().then(function (response) {
                            authenticationService.setUserInfo(response.data);
                            if (!authenticationService.getUserInfo().isApproved && toState.name.includes("tutor")) {
                                event.preventDefault();
                                $state.go("app.account.tutor.notapprove");
                            } else {
                                if (authenticationService.getUserInfo().isApproved && toState.name.includes("notapprove")) {
                                    event.preventDefault();
                                    $state.go("app.account.profile.list");
                                }
                            }
                            if (authenticationService.getUserInfo().isSetup && !toState.name.includes('profileSetup')) {
                                event.preventDefault();
                            }
                            else {

                                var userType = toState.name;
                                if (!authenticationService.isAuthorized(userType)) {
                                    event.preventDefault();
                                }
                            }

                        }, function () {
                            event.preventDefault();
                            authenticationService.logout();
                        })
                    })

                } else {
                    $state.go("app.admin.home");
                }


            }


        } else {
            if (authenticationService.getUserToken() != null) {
                event.preventDefault();
                if (authenticationService.getUserToken().systemUser) {
                    $state.go("app.admin.home");
                } else {
                    $state.go("app.account.profile.list");
                }
            }
        }
```

This is the code for the frontend endpoints which contains all the connection to the backend endpoints

authentication.service.js

```
"use strict";

export default {
    init: (app) => {
        app.factory('authenticationService', function authenticationService($http, $rootScope, $window,$state) {
            "ngInject";
            const userTokenKey = "_userToken";
            let vm = {
                isSetup: null,
                isStudent: null,
                isTutor: null,
                isManager: null,
                isApproved: false
            };

            return {
                isAuthorized: isAuthorized,
                authenticateUser: authenticate,
                setUserToken: setUserToken,
                getUserToken: getUserToken,
                clearAuthInfo: clearAuthInfo,
                logout: logout,
                getUserInfo: getUserInfo,
                signUp: signUp,
                checkUsername: checkUsername,
                studentProfileSetup: studentProfileSetup,
                tutorProfileSetup:tutorProfileSetup,
                managerProfileSetup:managerProfileSetup,
                getSelf: getSelf,
                setUserInfo: setUserInfo,
                getAllSubjects: getAllSubjects,
                getSubjects:getSubjects,
                getTutorRequests:getTutorRequests,
                getStudentRequests:getStudentRequests,
                respondToRequest:respondToRequest,
                findByTutor:findByTutor,
                findBySubject:findBySubject,
                findByTutorName:findByName,
                getTutorSubjects:getTutorSubjects,
                registerSession:register,
                isApproved: isApproved,
                setApproved: setApproved,
                getStudentHistory:getStudentHistory,
                getTutorHistory: getTutorHistory,
                getStudentProfile: getStudentProfile,
                getManagerProfile:getManagerProfile,
                getTutorProfile:getTutorProfile,
                getATutorProfile:getATutor,
                getTutorSchedule:getTutorSchedule,
                updateStudent:updateStudent,
                updateManager:updateManager,
                updateTutor:updateTutor,
                getManagerDashboard:getManagerDashboard,
                getTutorDashboard:getTutorDashboard,
                getStudentDashboard:getStudentDashboard,
                getManagerSessionHistory:getManagerSessionHistory,
                getManagerRequest:getManagerRequest,
                setUpStudent: setUpStudent,
                getProfile:getProfile,
                setAvailability:setAvailability,
```

authentication.service.js

```
                logoutAdmin: logoutAdmin,
                getAdminTutors:getAdminTutors,
                adminApprove:adminApprove,
                adminDeny:adminDeny,
                getAdmins:getAdmins,
                deleteAdmin:deleteAdmin,
                createAdmin:createAdmin,
                getAdminSubjects:getAdminSubjects,
                createSubject:createSubject,
                editSubject:editSubject,
                deleteSubject:deleteSubject,

                joinStudentSession: joinStudentSession,
                joinTutorSession: joinTutorSession,
                endSession: endSession

            };
            function deleteSubject(id) {
                return http("DELETE","./api/admin/subjects/"+id);
            }
            function editSubject(data,id) {
                return http("PUT","./api/admin/subjects/"+id,data);
            }
            function createSubject(data) {
                return http("POST","./api/admin/subjects",data);
            }
            function getAdminSubjects() {
                return http("GET","./api/admin/subjects");
            }
            function createAdmin(data) {
                return http("POST","./api/admin/accounts",data);
            }
            function deleteAdmin(id) {
                return http("DELETE","./api/admin/accounts/"+id);
            }
            function getAdmins() {
                return http("GET","./api/admin/accounts");
            }

            function adminApprove(id) {
                return http("GET","./api/admin/tutors/"+id+"/approve");
            }
            function adminDeny(id) {
                return http("DELETE","./api/admin/tutors/"+id, undefined);
            }

            function getAdminTutors() {
                return $http({
                    url: "./api/admin/tutors",
                    method: "GET",
                    params: {approved: false},
                    headers: {
                        'x-csrf': getUserToken().CSRF_TOKEN
                    }
                });
            }

            function logoutAdmin() {
                http('GET','./api/admin/auth/logout').then(function () {
                    clearAuthInfo();
                    $state.go('app.admin.login');
```

authentication.service.js

```
            console.log(err.data);
        });
    }

    function loginAdmin(data) {
        return $http.post("./api/admin/auth/login",data);
    }

    function setAvailability(data) {
        return http("POST","./api/dashboard/isavailable",data);
    }

    function getProfile() {
        return http("GET","./api/profiles/user");
    }

    function setUpStudent(data) {
        return http("POST","./api/dashboard/addstudent", data);
    }

    function getManagerRequest(id) {
        return http("GET","./api/sessions/students/requests/"+id);
    }

    function getManagerSessionHistory(id) {
        return http("GET","./api/sessions/students/"+id);
    }

    function joinStudentSession(sessionId) {
        return http("GET", `./api/opentok/${sessionId}/student`);
    }

    function joinTutorSession(sessionId) {
        return http("GET", `./api/opentok/${sessionId}/tutor`);
    }

    function endSession(sessionId) {
        return http("DELETE", `./api/opentok/${sessionId}`);
    }

    function getManagerDashboard() {
        return http("GET","./api/dashboard/Parent");
    }

    function getStudentDashboard() {
        return http("GET","./api/dashboard/student");
    }

    function getTutorDashboard() {
        return http("GET","./api/dashboard/Tutor");
    }

    function updateManager(data) {
        return http("PUT","./api/profiles/parent",data);
    }

    function updateTutor(data) {
        return http("PUT","./api/profiles/tutor",data);
    }
```

authentication.service.js

```
    }

    function getTutorSchedule(username) {
        return http("GET","./api/profiles/schedule/"+username);
    }

    function getTutorHistory() {
        return http("GET","./api/sessions/self/tutorhistory");
    }

    function getStudentHistory() {
        return http("GET","./api/sessions/self/studenthistory");
    }

    function setApproved(data) {
        vm.isApproved = data.approved;
    }
    function isApproved() {
        return http("GET","./api/profiles/isapproved");
    }

    function findByName(data) {
        return http("POST","./api/sessions/findtutor/bytutorname",data);
    }

    function register(data) {
        return http("POST","./api/sessions/register",data);
    }

    function getTutorSubjects(data) {
        return http("GET","./api/sessions/tutor/subjects/"+data);
    }

    function findBySubject(data) {
        return http("POST","./api/sessions/findtutor/bysubject",data);
    }

    function findByTutor(data) {
        return http("POST","./api/sessions/findtutor/bytutor",data);
    }

    function respondToRequest(data) {
        return http("POST","./api/sessions/requests/respond",data);
    }

    function getTutorRequests() {
        return http("GET","./api/sessions/tutors/requests/self");
    }

    function getStudentRequests() {
        return http("GET","./api/sessions/students/requests/self");
    }

    function getSubjects(field) {
        return http("GET","./api/profiles/subjects/"+field);
    }

    function getAllSubjects() {
        return http("GET",'./api/profiles/subjects');
    }
```

authentication.service.js

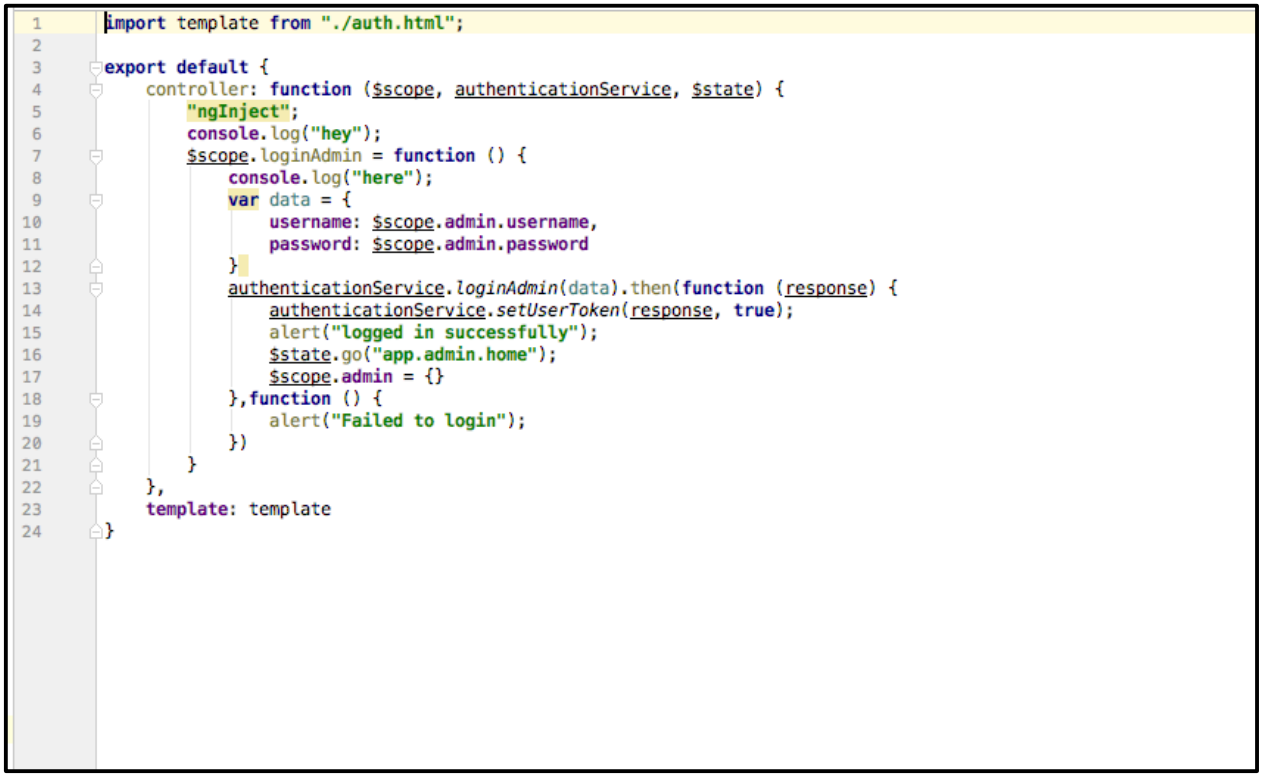
```
302
303             function logout() {
304                 http('GET','./api/users/logout').then(function () {
305                     clearAuthInfo();
306                     $state.go('app.auth');
307                 }, function (err) {
308                     console.log(err.data);
309                 });
310             }
311
312             function authenticate(userData) {
313                 return $http.post('./api/users/login', userData);
314             }
315
316             function getSelf() {
317                 return http('GET','./api/users/self');
318             }
319
320
321             function getUserInfo() {
322                 return vm;
323             }
324
325             function isAuthorized(userType) {
326                 if (userType.includes('profileSetup')) {
327                     return vm.isSetup;
328                 }
329                 else if (userType.includes('student')) {
330                     return vm.isStudent;
331                 }
332                 else if (userType.includes('tutor')) {
333                     return vm.isTutor;
334                 }
335                 else if (userType.includes('manager')) {
336                     return vm.isManager;
337                 }
338                 else if (userType.includes('profile')) {
339                     return !vm.isSetup;
340                 }
341                 else {
342                     return false;
343                 }
344             }
345
346             function setUserToken(token,bool) {
347                 clearAuthInfo();
348                 var userData = {
349                     systemUser: bool,
350                     CSRF_TOKEN: token.headers('x-csrf')
351                 }
352
353                 localStorage.setItem(userTokenKey, angular.toJson(userData));
354                 //sessionStorage.setItem(userTokenKey, angular.toJson(token));
355             };
356
357             function clearAuthInfo() {
358                 clearTokenInfo();
359             };
360
361             function clearTokenInfo() {
```

authentication.service.js

```
344             }
345
346             function setUserToken(token,bool) {
347                 clearAuthInfo();
348                 var userData = {
349                     systemUser: bool,
350                     CSRF_TOKEN: token.headers('x-csrf')
351                 }
352
353                 localStorage.setItem(userTokenKey, angular.toJson(userData));
354                 //sessionStorage.setItem(userTokenKey, angular.toJson(token));
355             };
356
357             function clearAuthInfo() {
358                 clearTokenInfo();
359             };
360
361             function clearTokenInfo() {
362                 localStorage.removeItem(userTokenKey);
363                 //sessionStorage.removeItem(userTokenKey);
364             };
365
366             function getUserToken() {
367                 let userToken = localStorage.getItem(userTokenKey);
368                 if (userToken) {
369                     return angular.fromJson(localStorage.getItem(userTokenKey));
370                 } else {
371                     return null;
372                 }
373             };
374
375             function signUp(userData) {
376                 return $http.post('./api/users/signup', userData);
377             }
378
379             function http(method, url, data){
380                 let req = {
381                     method: method,
382                     url: url,
383                     headers: {
384                         'x-csrf': getUserToken().CSRF_TOKEN
385                     }
386                 };
387
388                 if (method !== 'GET' && data) {
389                     req.data = data;
390                 }
391
392                 return $http(req);
393             }
394
395         })
396     }
397 }
398
```

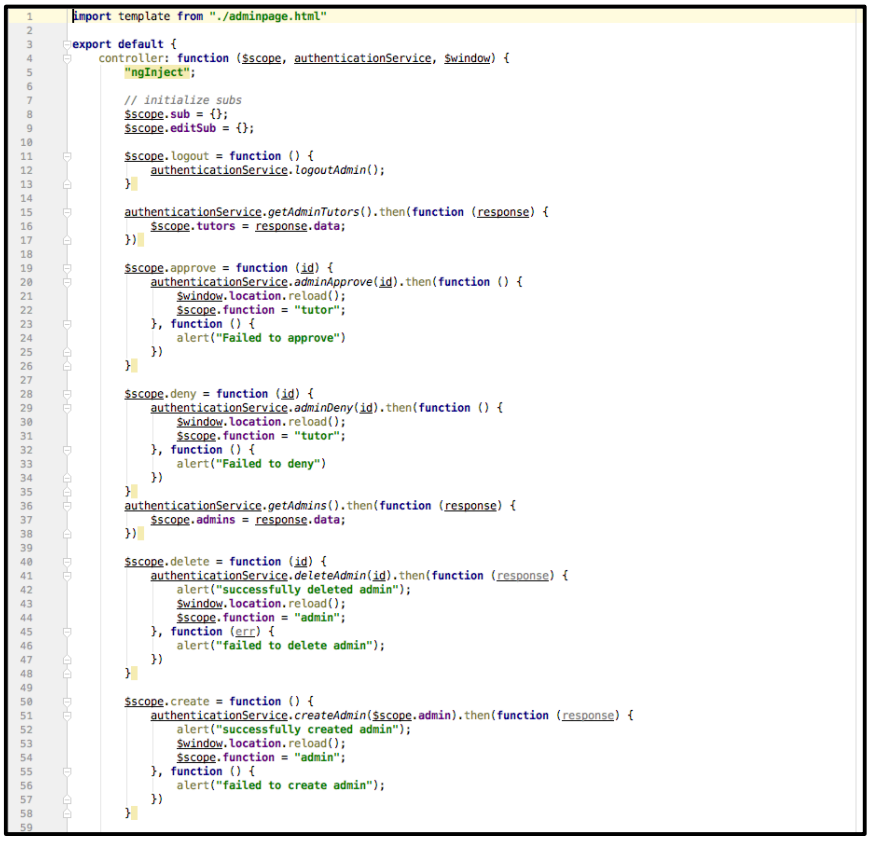
CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

This is the code for sending the admin login data to the backend endpoint



```
import template from "./auth.html";

export default {
    controller: function ($scope, authenticationService, $state) {
        "ngInject";
        console.log("hey");
        $scope.loginAdmin = function () {
            console.log("here");
            var data = {
                username: $scope.admin.username,
                password: $scope.admin.password
            }
            authenticationService.loginAdmin(data).then(function (response) {
                authenticationService.setUserToken(response, true);
                alert("logged in successfully");
                $state.go("app.admin.home");
                $scope.admin = {}
            },function () {
                alert("Failed to login");
            })
        }
    },
    template: template
}
```

This is the code for handling the functionalities that an Admin can perform



```
import template from "./adminpage.html"

export default {
    controller: function ($scope, authenticationService, $window) {
        "ngInject";

        // initialize subs
        $scope.sub = {};
        $scope.editSub = {};

        $scope.logout = function () {
            authenticationService.logoutAdmin();
        }

        authenticationService.getAdminTutors().then(function (response) {
            $scope.tutors = response.data;
        })

        $scope.approve = function (id) {
            authenticationService.adminApprove(id).then(function () {
                $window.location.reload();
                $scope.function = "tutor";
            }, function () {
                alert("Failed to approve")
            })
        }

        $scope.deny = function (id) {
            authenticationService.adminDeny(id).then(function () {
                $window.location.reload();
                $scope.function = "tutor";
            }, function () {
                alert("Failed to deny")
            })
        }
        authenticationService.getAdmins().then(function (response) {
            $scope.admins = response.data;
        })

        $scope.delete = function (id) {
            authenticationService.deleteAdmin(id).then(function (response) {
                alert("successfully deleted admin");
                $window.location.reload();
                $scope.function = "admin";
            }, function (err) {
                alert("failed to delete admin");
            })
        }

        $scope.create = function () {
            authenticationService.createAdmin($scope.admin).then(function (response) {
                alert("successfully created admin");
                $window.location.reload();
                $scope.function = "admin";
            }, function () {
                alert("failed to create admin");
            })
        }

```

```
        }

        $scope.createSubject = function () {
            authenticationService.createSubject($scope.sub).then(function (response) {
                alert("successfully created subject");
                $window.location.reload();
                $scope.sub = {};
                $scope.function = "subject";
            }, function () {
                alert("failed to create subject");
            })
        }

        authenticationService.getAdminSubjects().then(function (response) {
            $scope.subjects = response.data;
        })

        $scope.toEditSubjectSelected = function(selected)
        {
            $scope.editSub.field = selected.field;
            $scope.editSub.name = selected.name;
            $scope.editSub.description = selected.description;
        };

        $scope.editSubject = function (id) {
            authenticationService.editSubject($scope.editSub, id).then(function () {
                alert("successfully edited subject");
                $window.location.reload();
                $scope.editSub = {};
                $scope.selectedSub = "";
                $scope.function = "subject";
            }, function () {

                alert("failed to create subject");
            })
        }
        $scope.deleteSub = function (s) {
            var data = {
                "field": s.field,
                "name": s.name,
                "description": s.description
            }
            authenticationService.deleteSubject(s.id, data).then(function () {
                alert("successfully deleted subject");
                $window.location.reload();
                $scope.selectedSub = "";
                $scope.function = "subject";
            }, function () {

                alert("failed to delete subject");
            })
        }
    },
    template: template
}
```

This is the code for getting data from the backend endpoint to display on the parent dashboard

```
import template from './manager.dashboard.html';

export default {
    controller: function ($scope, authenticationService, $window, $state) {
        "ngInject";

        $scope.student = {
            firstName: '',
            lastName: '',
            dateOfBirth: new Date(),
            streetAddress: '',
            city: '',
            state: '',
            postalCode: ''
        };

        $scope.states = ['Alabama', 'Alaska', 'American Samoa', 'Arizona', 'Arkansas', 'California', 'Colorado', 'Connecticut', 'Delaware', 'District

        $scope.minDate = new Date(
            $scope.student.dateOfBirth.getFullYear() - 100,
            $scope.student.dateOfBirth.getMonth(),
            $scope.student.dateOfBirth.getDate()
        );

        $scope.maxDate = new Date(
            $scope.student.dateOfBirth.getFullYear(),
            $scope.student.dateOfBirth.getMonth(),
            $scope.student.dateOfBirth.getDate()
        );
        $scope.show = true;
        $scope.addForm = function () {
            $scope.show = false;
        }
        $scope.cancel = function () {
            $scope.show = true;
            $scope.student = {};
        }

        authenticationService.getManagerDashboard().then(function (response) {
            $scope.manager = response.data;
        })

        $scope.completeProfile = function () {
            authenticationService.setUpStudent($scope.student).then(function (response) {
                $scope.show = !$scope.show;
                alert("Successfully added student");
                $window.location.reload();
                $scope.student = {};
            }, function (err) {
                alert("Failed to add student");
            })
        }
```

This is the code for getting data from the backend endpoint to display on the student dashboard

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

```
import template from "./student.dashboard.html";

export default {
    controller: function ($scope, authenticationService, $state) {
        "ngInject";

        $scope.student = {};

        reloadData();

        function reloadData() {
            authenticationService.getStudentDashboard().then(function (response) {
                $scope.student = response.data;
            });
        }

```

This is the code for getting data from the backend endpoint to display on the tutor dashboard

```
import template from './tutor.dashboard.html';

export default {
    controller: function ($scope, authenticationService, $state) {
        "ngInject";

        $scope.tutor = {};

        reloadData();

        function reloadData() {
            authenticationService.getTutorDashboard().then(function (response) {
                $scope.tutor = response.data;
            });
        }

```

This is the code for getting data from the backend endpoint to display on the parent session request page

```
import template from "./manager.requests.html";

export default {
    controller: function ($scope, authenticationService) {
        "ngInject";

        $scope.parent={

        }
        authenticationService.getManagerDashboard().then(function (response) {
            $scope.students = response.data.managedStudents;
        })
        $scope.getRequest = function (id) {
            authenticationService.getManagerRequest(id).then(function (response) {
                $scope.errorMsg = '';
                angular.forEach(response.data, function (obj) {
                    obj.requestDate.toLocaleString();
                })
                $scope.studentRequests = response.data;
            }, function () {
                $scope.errorMsg = "Unable to retrieve your requests at this moment";
            })
        }

    },
    template: template
}
```

This is the code for getting data from the backend endpoint to display on the student session request page

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

```
import template from "./student.requests.html";

export default {
    controller: function ($scope, authenticationService) {
        "ngInject";
        authenticationService.getStudentRequests().then(function (response) {
            $scope.errorMsg = '';
            console.log(response.data);
            angular.forEach(response.data, function (obj) {
                obj.requestDate.toLocaleString();
            })
            console.log(response.data);
            $scope.studentSessionRequests = response.data;
        }, function () {
            $scope.errorMsg = "Unable to retrieve your requests at this moment";
        })

    },
    template: template
}
```

This is the code for getting data from the backend endpoint to display on the tutor session request page

```
import template from "./tutor.requests.html";

export default {
    controller: function ($scope, authenticationService) {
        "ngInject";

        $scope.tutorSessionRequests = {
            pending: [],
            req: []
        };

        function loadData() {
            authenticationService.getTutorRequests().then(function (response) {
                $scope.errorMsg = '';

                // clear them first.
                $scope.tutorSessionRequests.pending = [];
                $scope.tutorSessionRequests.req = [];

                for (let sr of response.data) {
                    if (sr.status === "PENDING") {
                        $scope.tutorSessionRequests.pending.push(sr);
                    } else {
                        $scope.tutorSessionRequests.req.push(sr);
                    }
                }
                console.log($scope.tutorSessionRequests)
            }, function () {
                $scope.errorMsg = "Unable to retrieve your requests at this moment";
            });
        }

        loadData();

        $scope.decline = function (id) {
            let data = {
                requestId: id,
                status: "DECLINED"
            };
            authenticationService.respondToRequest(data).then(function (response) {
                $scope.tutorSessionRequests = response.data;
                alert("successfully declined request!")
                loadData(); // again.
            }, function () {
                $scope.errorMsg = "Failed to decline request";
            })
        };

        $scope.approve = function (id) {
            let data = {
                requestId: id,
                status: "APPROVED"
            };
            authenticationService.respondToRequest(data).then(function (response) {
                $scope.tutorSessionRequests = response.data;
                alert("successfully approved request!")
                loadData(); // again.
            }, function () {
                $scope.errorMsg = "Failed to approve request";
            })
```

Database SQL

```
ALTER TABLE `Users`
ADD `managerPersonId` INT(11)        DEFAULT NULL;

ALTER TABLE `Users`
  ADD KEY `FK_Users_Managers` (`managerPersonId`);

ALTER TABLE `Users`
  ADD CONSTRAINT `FK_Users_Managers` FOREIGN KEY (`managerPersonId`) REFERENCES `Persons` (`id`)
  ON DELETE SET NULL;
```

```
CREATE TABLE `Sessions` (
  `id`            INT(11)                    NOT NULL,
  `studentId`   INT(11)  NOT NULL,
  `tutorId`   INT(11)  NOT NULL,
  `subjectId` INT(11)  NOT NULL,
   `description` VARCHAR(500)
              COLLATE `utf8mb4_unicode_ci`  NULL DEFAULT '',

   `startDateTime`   DATETIME                      NOT NULL,
   `endDateTime`   DATETIME                      NULL,
  `status` VARCHAR(100)                  NOT NULL,
  `createdOn`        DATETIME                    NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `lastModified`  TIMESTAMP                  NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)
  ENGINE = InnoDB
  DEFAULT CHARSET = `utf8mb4`
  COLLATE = `utf8mb4_unicode_ci`;


ALTER TABLE `Sessions`
  ADD PRIMARY KEY (`id`),
  ADD KEY `FK_Sessions_Student` (`studentId`),
  ADD KEY `FK_Sessions_Tutor` (`tutorId`),
  ADD KEY `FK_Sessions_Subject` (`subjectId`);

ALTER TABLE `Sessions`
  MODIFY `id` INT(11) NOT NULL AUTO_INCREMENT;

ALTER TABLE `Sessions`
  ADD CONSTRAINT `FK_Sessions_Student` FOREIGN KEY (`studentId`) REFERENCES `Students` (`id`),
  ADD CONSTRAINT `FK_Sessions_Tutor` FOREIGN KEY (`tutorId`) REFERENCES `Tutors` (`id`),
  ADD CONSTRAINT `FK_Sessions_Subject` FOREIGN KEY (`subjectId`) REFERENCES `Subjects` (`id`);
```

```
INSERT INTO `Subjects` (`id`, `name`, `field`, `description`) VALUES ('1', 'Painting', 'ART', 'test4');
INSERT INTO `Subjects` (`id`, `name`, `field`, `description`) VALUES ('2', 'Calculus AB', 'MATH', 'test3');
INSERT INTO `Subjects` (`id`, `name`, `field`, `description`) VALUES ('3', 'Geometry', 'MATH', 'test6');
INSERT INTO `Subjects` (`id`, `name`, `field`, `description`) VALUES ('4', 'Biology', 'SCIENCE', 'test7');
```

```
ALTER TABLE `Tutors`
ADD `availability` TINYINT(1)        NOT NULL DEFAULT '0';
```

```
CREATE TABLE `SessionRequests` (
  `id`            INT(11)                       NOT NULL,
  `requestedFor`   INT(11)  NOT NULL,
  `requestedTo`   INT(11)  NOT NULL,
  `subjectId` INT(11)  NOT NULL,

   `sessionDateTime`   DATETIME                          NOT NULL,
  `status`        VARCHAR(10)
                COLLATE `utf8mb4_unicode_ci` NOT NULL,
  `createdOn`    DATETIME  NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `lastModified`  TIMESTAMP                     NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)
  ENGINE = InnoDB
  DEFAULT CHARSET = `utf8mb4`
  COLLATE = `utf8mb4_unicode_ci`;


ALTER TABLE `SessionRequests`
  ADD PRIMARY KEY (`id`),
  ADD KEY `FK_SessionRequests_User` (`requestedFor`),
  ADD KEY `FK_SessionRequests_Tutor` (`requestedTo`),
  ADD KEY `FK_SessionRequests_Subject` (`subjectId`);

ALTER TABLE `SessionRequests`
  MODIFY `id` INT(11) NOT NULL AUTO_INCREMENT;

ALTER TABLE `SessionRequests`
  ADD CONSTRAINT `FK_SessionRequests_User` FOREIGN KEY (`requestedFor`) REFERENCES `Users` (`id`),
  ADD CONSTRAINT `FK_SessionRequests_Tutor` FOREIGN KEY (`requestedTo`) REFERENCES `Tutors` (`id`),
  ADD CONSTRAINT `FK_SessionRequests_Subject` FOREIGN KEY (`subjectId`) REFERENCES `Subjects` (`id`);
```

```
CREATE TABLE `Schedules` (
  `id`            INT(11)                       NOT NULL,
  `tutorId`   INT(11)  NOT NULL,
  `dayAvailable`           ENUM ('MONDAY', 'TUESDAY', 'WEDNESDAY','THURSDAY','FRIDAY','SATURDAY','SUNDAY')
                  COLLATE `utf8mb4_unicode_ci` NOT NULL,
                  `timeAvailableDescription` VARCHAR(100)
                COLLATE `utf8mb4_unicode_ci` NOT NULL DEFAULT 'All day',
  `createdOn`    DATETIME  NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `lastModified`  TIMESTAMP                     NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)
  ENGINE = InnoDB
  DEFAULT CHARSET = `utf8mb4`
  COLLATE = `utf8mb4_unicode_ci`;


ALTER TABLE `Schedules`
  ADD PRIMARY KEY (`id`),
  ADD KEY `FK_Schedules_Tutor` (`tutorId`);

ALTER TABLE `Schedules`
  MODIFY `id` INT(11) NOT NULL AUTO_INCREMENT;

ALTER TABLE `Schedules`
  ADD CONSTRAINT `FK_Schedules_Tutor` FOREIGN KEY (`tutorId`) REFERENCES `Tutors` (`id`);
```

Backend API Endpoints

Takes the value of the tutor availability and updates it in the database. It returns the updated tutor object

```
@POST
@UnitOfWork
@Path("/isavailable")
public Tutor setAvailability(@Auth SessionToken auth, @Valid AvailableRequest setup)
{
    if(auth.getTutorId() != 0)
    {
        Tutor t = tdao.get(auth.getTutorId());
        t.setAvailability(setup.isAvailable());
        return t;
    }
    throw new ForbiddenException("Unable to set availability");
}
```

Takes the new student information, creates a student object and enters the student information into the database. The general information such as name and address are saved in the Persons table. Therefore, this function also creates a new person record for the specified student. It returns the newly created student object

```
@POST
@UnitOfWork
@Path("/addstudent")
public Student addManagerStudent(@Auth SessionToken auth, @Valid ProfileSetupRequest setup)
{
    if(auth.isStudentManager())
    {
        Date dob = setup.getDateOfBirth();
        Person person = pdao.create(setup.getFirstName(), setup.getLastName(), dob, setup.getStreetAddress(), setup.getCity(), setup.getState(), setup.getCount
        Student student = sdao.create(person);
        student.setManagerId(auth.getUserId());
        return student;
    }
```

In the initial phase of account creation, these functions take the user data and for whichever user type the user indicated, creates the corresponding object and stores the data in the database. It updates the User table with a foreign key mapping to the indicated user type table. It returns the newly created student or tutor or parent object

```
@POST
@UnitOfWork
@Path("/studentsetup")
public Response studentProfileSetup(@Auth SessionToken auth, @Valid ProfileSetupRequest setup)
{
    if(auth.getStudentId() == 0)
    {
        User user = udao.get(auth.getUserId());
        Date dob = setup.getDateOfBirth();
        Person person = pdao.create(setup.getFirstName(), setup.getLastName(), dob, setup.getStreetAddress(), setup.getCity(), setup.getState(), setup.getCountr
        Student student = sdao.create(person);
        student.setGradeLevel(setup.getGradeLevel());
        user.setStudent(student);
        user.setStudentId(student.getId());
        NewAuthCreds creds = authHelper.buildCredentials(user);

        return Response.ok(user).cookie(creds.getCookie())
                .header(AuthHelper.CSRF_HEADER, creds.getHeader()).build();
    }
    throw new ForbiddenException("Student account is already setup");
}
```

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

```
@POST
@UnitOfWork
@Path("/tutorsetup")
public Response tutorProfileSetup(@Auth SessionToken auth, @Valid TutorProfileSetupRequest setup)
{
    if(auth.getTutorId() == 0)
    {
        User user = udao.get(auth.getUserId());
        Date dob = setup.getDateOfBirth();
        Person person = pdao.create(setup.getFirstName(), setup.getLastName(), dob, setup.getStreetAddress(), setup.getCity(), setup.getState(), setup.getCoun
        ArrayList<Long> userExpertise = setup.getExpertise();
        Tutor tutor = tdao.create(person, setup.getExpYears());
        for(Long anUserExpertise : userExpertise)
        {
            Subject subject = subdao.get(anUserExpertise);
            if(!tedao.exists(tutor, subject).isPresent())
            {
                tedao.create(tutor, subject);
            }
        }
        ArrayList<ScheduleSetupRequest> s = setup.getSchedule();
        for(ScheduleSetupRequest value : s)
        {
            scdao.create(tutor, value.getDay(), value.getTime());
        }
        user.setTutorId(tutor.getId());

        NewAuthCreds creds = authHelper.buildCredentials(user);

        return Response.ok(user)
                .cookie(creds.getCookie())
                .header(AuthHelper.CSRF_HEADER, creds.getHeader())
                .build();
    }
    throw new ForbiddenException("Tutor account is already setup");
}
```

```
@POST
@UnitOfWork
@Path("/managersetup")
public Response managerProfileSetup(@Auth SessionToken auth, @Valid ProfileSetupRequest setup)
{
    if(!auth.isStudentManager())
    {
        User user = udao.get(auth.getUserId());
        Date dob = setup.getDateOfBirth();
        Person person = pdao.create(setup.getFirstName(), setup.getLastName(), dob, setup.getStreetAddress(), setup.getCity(), setup.getState(), setup.getCount
        user.setManagerPersonId(person.getId());
        user.setStudentManager(true);
        NewAuthCreds creds = authHelper.buildCredentials(user);

        return Response.ok(user).cookie(creds.getCookie())
                .header(AuthHelper.CSRF_HEADER, creds.getHeader()).build();
    }
    throw new ForbiddenException("Manager account is already setup");
}
```

This takes the get parameter which is the name of a course field and the function returns all the records that fall under the specified field

```
@GET
@UnitOfWork
@Path("/subjects/{field}")
public Response getSubjects(@PathParam("field") @NotNull String field)
{
    return Response.ok(subdao.getSubjects(field)).build();
}
```

This function returns the entire list of records in Subjects table

```
@GET
@UnitOfWork
@Path("/subjects")
public Response getAllSubjects() { return Response.ok(subdao.getSubjects()).build(); }
```

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

These functions return the list of past sessions a student or tutor or student being managed by a parent has had

```
@GET
@UnitOfWork
@Path("/self/studenthistory")
public Response studentSessionHistory(@Auth SessionToken auth)
{
    if(auth.getStudentId() != 0)
    {
        List<Sessions> sessionsList = sdao.getStudentSessions(auth.getStudentId());
        List<HashMap<String, String>> studentSessions = new ArrayList();
        for(Sessions s : sessionsList)
        {
            HashMap<String, String> sessions = new HashMap();
            sessions.put("tutorUsername", udao.getByTutorId(s.getTutorId()).get().getUsername());
            sessions.put("subject", s.getSubject().getName());
            sessions.put("startDate", s.getStartDateTime().toString());
            sessions.put("endDate", s.getEndDateTime().toString());
            sessions.put("description", s.getDescription());
            studentSessions.add(sessions);
        }
        return Response.ok(studentSessions).build();
    }
    throw new ForbiddenException("You are not authorize to access this data");
}
```

```
@GET
@UnitOfWork
@Path("/self/tutorhistory")
public Response tutorSessionHistory(@Auth SessionToken auth)
{
    if(auth.getTutorId() != 0)
    {
        List<Sessions> sessionsList = sdao.getTutorSessions(auth.getTutorId());
        List<HashMap<String, String>> tutorSessions = new ArrayList();
        for(Sessions s : sessionsList)
        {
            HashMap<String, String> sessions = new HashMap();
            sessions.put("studentUsername", (udao.getByStudentId(s.getStudentId()).get().getUsername()));
            sessions.put("subject", s.getSubject().getName());
            sessions.put("startDate", s.getStartDateTime().toString());
            sessions.put("endDate", s.getEndDateTime().toString());
            sessions.put("description", s.getDescription());
            tutorSessions.add(sessions);
        }

        return Response.ok(tutorSessions).build();
    }
    throw new ForbiddenException("You are not authorize to access this data");
}
```

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

```
@GET
@UnitOfWork
@Path("/students/{studentId}")
public Response managerStudentHistory(@Auth SessionToken auth, @PathParam("studentId") @Min(1) long studentId)
{
    if(auth.isStudentManager() && udao.getManagedStudents(auth.getUserId()).contains(stdao.get(studentId)))
    {
        List<Sessions> sessionsList = sdao.getTutorSessions(studentId);
        List<HashMap<String, String>> studentSessions = new ArrayList();
        for(Sessions s : sessionsList)
        {
            HashMap<String, String> sessions = new HashMap();
            sessions.put("tutorUsername", udao.getByTutorId(s.getTutorId()).get().getUsername());
            sessions.put("subject", s.getSubject().getName());
            sessions.put("startDate", s.getStartDateTime().toString());
            sessions.put("endDate", s.getEndDateTime().toString());
            sessions.put("description", s.getDescription());
            studentSessions.add(sessions);
        }
        return Response.ok(studentSessions).build();
    }
    throw new ForbiddenException("You are not authorize to access this data");
}
```

These functions return the list of session requests of a student or tutor or a specific student being managed by a parent

```
@GET
@UnitOfWork
@Path("/students/requests/self")
public Response getStudentRequests(@Auth SessionToken auth)
{
    if(auth.getStudentId() != 0)
    {
        List<SessionRequest> srs = new ArrayList();
        srs = srdao.getStudentRequests(auth.getStudentId());
        List<HashMap<String, String>> studentRequests = new ArrayList();
        for(SessionRequest s : srs)
        {
            HashMap<String, String> request = new HashMap();
            request.put("requestId", Long.toString(s.getId()));
            request.put("requestDate", s.getCreatedOn().format(dateFormat));
            request.put("tutorUsername", udao.getByTutorId(s.getRequestedTo()).get().getUsername());
            request.put("subject", s.getSubject().getName());
            request.put("time", s.getSessionDateTime().format(dateFormat));
            request.put("status", s.getStatus().name());
            studentRequests.add(request);
        }
        return Response.ok(studentRequests).build();
    }
    throw new ForbiddenException("You are not authorize to access this data");
}
```

```java
@GET
@UnitOfWork
@Path("/students/requests/{studentId}")
public Response getAStudentRequests(@Auth SessionToken auth, @PathParam("studentId") @Min(1) long studentId)
{
    Student student = stdao.get(studentId);

    if(auth.isStudentManager() && student != null && auth.getUserId() == student.getManagerId())
    {
        List<SessionRequest> srs  = srdao.getStudentRequests(studentId);
        List<HashMap<String, String>> studentRequests = Lists.newArrayList();

        for(SessionRequest s : srs)
        {
            HashMap<String, String> request = new HashMap();
            request.put("requestId", Long.toString(s.getId()));
            request.put("requestDate", s.getCreatedOn().format(dateFormat));
            request.put("tutorUsername", udao.getByTutorId(s.getRequestedTo()).get().getUsername());
            request.put("subject", s.getSubject().getName());
            request.put("time", s.getSessionDateTime().format(dateFormat));
            request.put("status", s.getStatus().name());
            studentRequests.add(request);
        }
        return Response.ok(studentRequests).build();
    }
    throw new ForbiddenException("You are not authorize to access this data");
}
```

```java
@GET
@UnitOfWork
@Path("/tutors/requests/self")
public Response getTutorRequests(@Auth SessionToken auth)
{
    if(auth.getTutorId() != 0)
    {
        List<SessionRequest> srs = srdao.getTutorRequests(auth.getTutorId());
        List<HashMap<String, String>> tutorRequests = new ArrayList();
        for(SessionRequest s : srs)
        {
            HashMap<String, String> request = new HashMap();
            request.put("requestId", Long.toString(s.getId()));
            request.put("requestDate", s.getCreatedOn().format(dateFormat));

            Optional<User> studentUser = udao.getByStudentId(s.getRequestedFor());
            request.put("studentUsername", studentUser.isPresent() ? studentUser.get().getUsername() : "ManagedUser"+s.getRequestedFor());

            request.put("subject", s.getSubject().getName());
            request.put("time", s.getSessionDateTime().format(dateFormat));
            request.put("status", s.getStatus().name());
            tutorRequests.add(request);
        }
        return Response.ok(tutorRequests).build();
    }
    throw new ForbiddenException("You are not authorize to access this data");
}
```

This returns all the subjects that a specific tutor expertise in. it takes in the tutor username

```java
@GET
@UnitOfWork
@Path("/tutor/subjects/{tutorusername}")
public Response getTutorSubjects(@PathParam("tutorusername") @NotNull String username){
    if(udao.getByUsername(username).isPresent()){
        User user = udao.getByUsername(username).get();
        return Response.ok(user.getTutor().getExpertise()).build();
    }
    throw new ForbiddenException("You are not authorize to access this data");
}
```

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

This takes the response value a tutor made to a request and sets it as the status of that request in the database. It returns the updated session request data

```
@POST
@UnitOfWork
@Path("/requests/respond")
public Response responseToRequest(@Auth SessionToken auth, SessionResponseRequest response)
{
    if(auth.getTutorId() != 0 && srdao.get(response.getRequestId()).getRequestedTo() == auth.getTutorId())
    {
        SessionRequest sr = srdao.get(response.getRequestId());
        sr.setStatus(response.getStatus());
        if(response.getStatus().equals(SessionRequestStatus.APPROVED)){
            Sessions session = sdao.create(sr.getRequestedFor(),sr.getRequestedTo(),sr.getSubjectId(),sr.getSessionDateTime());
        }
        List<SessionRequest> srs = new ArrayList();
        srs = srdao.getTutorRequests(auth.getTutorId());
        List<HashMap<String, String>> tutorRequests = new ArrayList();
        for(SessionRequest s : srs)
        {
            HashMap<String, String> request = new HashMap();
            request.put("requestId", Long.toString(s.getId()));
            request.put("requestDate", s.getCreatedOn().format(dateFormat));

            Optional<User> studentUser = udao.getByStudentId(s.getRequestedFor());
            request.put("studentUsername", studentUser.isPresent() ? studentUser.get().getUsername() : "ManagedUser"+s.getRequestedFor());

            request.put("subject", s.getSubject().getName());
            request.put("time", s.getSessionDateTime().format(dateFormat));
            request.put("status", s.getStatus().name());
            tutorRequests.add(request);
        }
        return Response.ok(tutorRequests).build();
    }
    throw new ForbiddenException("You are not authorize to respond to this request");
}
```

These functions allow returns the information of all tutors that match the specified search criteria.

```
@POST
@UnitOfWork
@Path("/findtutor/bytutor")
public Response findByTutor(@Auth SessionToken auth, FindByTutorRequest req)
{
    if(auth.getStudentId() != 0 || auth.isStudentManager())
    {

        User user = udao.getByUsername(req.getUsername()).get();
        if(user.getTutorId() != 0)
        {
            Tutor tutor = user.getTutor();
          if(tutor.getId() != auth.getTutorId() && tutor.isApproved()){
              List <HashMap<String, String>> t = new ArrayList();
              HashMap<String, String> tutorInfo = new HashMap();
              tutorInfo.put("username", user.getUsername());
              tutorInfo.put("state", tutor.getPerson().getState());
              tutorInfo.put("firstName",tutor.getPerson().getFirstName());
              tutorInfo.put("lastName",tutor.getPerson().getLastName());
              tutorInfo.put("city",tutor.getPerson().getCity());
              tutorInfo.put("postalCode",tutor.getPerson().getPostalCode());
              tutorInfo.put("address",tutor.getPerson().getStreetAddress());
              tutorInfo.put("country", tutor.getPerson().getCountry());
              tutorInfo.put("expYears", Integer.toString(tutor.getExpYears()));
              tutorInfo.put("available", Boolean.toString(tutor.isAvailability()));
              Iterator<Subject> iterator = tutor.getExpertise().iterator();
              while(iterator.hasNext())
              {
                  Subject s = iterator.next();
                  tutorInfo.put(s.getName(), s.getDescription());
              }
              t.add(tutorInfo);
              return Response.ok(t).build();
          }else{
              return Response.ok().build();
          }
        }
    }
    throw new ForbiddenException("You are not authorize to access this data");
}
```

```
@POST
@UnitOfWork
@Path("/findtutor/bytutorname")
public Response findByTutorName(@Auth SessionToken auth, FindByTutorNameRequest req)
{
    if(auth.getStudentId() != 0 || auth.isStudentManager())
    {
        List<Tutor> te = tdao.getTutors(req.getFirstName(),req.getLastName());
        List<HashMap<String, String>> tutors = new ArrayList();
        for(int i = 0; i < te.size(); i++)
        {
            Tutor tutor = te.get(i);
            if(te.get(i).getId() == auth.getTutorId() || !te.get(i).isApproved()){
                ++i;
                if(i<te.size()){
                    tutor = te.get(i);
                }
                else {
                    break;
                }
            }
            HashMap<String, String> tutorInfo = new HashMap();
            tutorInfo.put("username", udao.getByTutorId(te.get(i).getId()).get().getUsername());
            tutorInfo.put("firstName",te.get(i).getPerson().getFirstName());
            tutorInfo.put("lastName",te.get(i).getPerson().getLastName());
            tutorInfo.put("city",te.get(i).getPerson().getCity());
            tutorInfo.put("postalCode",te.get(i).getPerson().getPostalCode());
            tutorInfo.put("address",te.get(i).getPerson().getStreetAddress());
            tutorInfo.put("state", te.get(i).getPerson().getState());
            tutorInfo.put("country", te.get(i).getPerson().getCountry());
            tutorInfo.put("expYears", Integer.toString(te.get(i).getExpYears()));
            tutorInfo.put("available", Boolean.toString(te.get(i).isAvailability()));
            for(Subject s : te.get(i).getExpertise())
            {
                tutorInfo.put(s.getName(), s.getDescription());
            }
            tutors.add(tutorInfo);
        }
        return Response.ok(tutors).build();
    }
    throw new ForbiddenException("You are not authorize to access this data");
}
```

```
@POST
@UnitOfWork
@Path("/findtutor/bysubject")
public Response findBySubject(@Auth SessionToken auth, FindBySubject req)
{
    if(auth.getStudentId() != 0 || auth.isStudentManager())
    {
        List<TutorExpertise> te = tdao.findBySubject(req.getSubjectId());
        List<HashMap<String, String>> tutors = new ArrayList();
        for(int i = 0; i < te.size(); i++)
        {
            Tutor tutor = te.get(i).getTutor();
            if(tutor.getId() == auth.getTutorId()|| !tutor.isApproved()){
                ++i;
                if(i<te.size()){
                    tutor = te.get(i).getTutor();
                }
                else {
                    break;
                }
            }

            HashMap<String, String> tutorInfo = new HashMap();
            tutorInfo.put("username", udao.getByTutorId(tutor.getId()).get().getUsername());
            tutorInfo.put("state", tutor.getPerson().getState());
            tutorInfo.put("firstName",tutor.getPerson().getFirstName());
            tutorInfo.put("lastName",tutor.getPerson().getLastName());
            tutorInfo.put("city",tutor.getPerson().getCity());
            tutorInfo.put("postalCode",tutor.getPerson().getPostalCode());
            tutorInfo.put("address",tutor.getPerson().getStreetAddress());
            tutorInfo.put("country", tutor.getPerson().getCountry());
            tutorInfo.put("expYears", Integer.toString(tutor.getExpYears()));
            tutorInfo.put("available", Boolean.toString(tutor.isAvailability()));
            for(Subject s : tutor.getExpertise())
            {
                tutorInfo.put(s.getName(), s.getDescription());
            }
            tutors.add(tutorInfo);
        }
        return Response.ok(tutors).build();
    }
    throw new ForbiddenException("You are not authorize to access this data");
}
```

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

This takes the session registration form data and creates a new record in the Session Request table. It returns the newly created session request

```
@POST
@UnitOfWork
@Path("/register")
public SessionRequest register(@Auth SessionToken auth, RegisterSessionRequest req){
    if(auth.getStudentId()!=0||auth.isStudentManager()){
        if(req.getStudentId() != 0){
            User user = udao.getByUsername(req.getUsername()).get();
            return srdao.create(req.getStudentId(),user.getTutorId(),req.getSubjectId(),req.getDate());
        }
        if(udao.getByUsername(req.getUsername()).isPresent()){
            User user = udao.getByUsername(req.getUsername()).get();
            return srdao.create(auth.getStudentId(),user.getTutorId(),req.getSubjectId(),req.getDate());
        }
    }
    throw new ForbiddenException("You are not authorize to access this data");
}
```

Endpoint Request Classes

This is the format in which the form data of student or parent profile setup need to be before being sent to its corresponding API endpoint

```
package edu.utdallas.utdesign.teach4service.resources.requests;

import ...

@Data
public class ProfileSetupRequest
{
    @NotBlank
    private String firstName;

    @NotBlank
    private String lastName;

    @NotNull
    private Date dateOfBirth;

    @NotBlank
    private String streetAddress;

    private int gradeLevel;

    @NotBlank
    private String city;

    @NotBlank
    private String state;

    @NotBlank
    private String country;

    @NotBlank
    private String postalCode;
}
```

This is the format in which the form data of tutor profile setup need to be before being sent to its corresponding API endpoint

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

```java
package edu.utdallas.utdesign.teach4service.resources.requests;

import ...

@Data
@EqualsAndHashCode(callSuper=false)
public class TutorProfileSetupRequest extends ProfileSetupRequest
{
    @NotNull
    private int expYears;

    @NotEmpty
    private ArrayList <Long> expertise;

    @NotEmpty
    private ArrayList<ScheduleSetupRequest> schedule;



}
```

This is the format in which the form data of creating subject need to be before being sent to its corresponding API endpoint

```java
package edu.utdallas.utdesign.teach4service.resources.requests;

import ...

@Data
public class SubjectRequest
{
    @NotBlank
    private String name;

    @NotBlank
    private String field;

    @NotBlank
    private String description;
}
```

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

This is the format in which the form data of responding to a request need to be before being sent to its corresponding API endpoint

```
package edu.utdallas.utdesign.teach4service.resources.requests;

import ...

@Data
public class SessionResponseRequest
{
    @Min(1)
    private long                requestId;
    private SessionRequestStatus status;
}
```

This is the format in which the form data of tutor setting up their schedule need to be before being sent to its corresponding API endpoint

```
package edu.utdallas.utdesign.teach4service.resources.requests;

import ...

@Data
public class ScheduleSetupRequest
{
    @NotNull
    private String day;

    @NotNull
    private String time;
}
```

This is the format in which the form data of registering a session with a tutor need to be before being sent to its corresponding API endpoint

```
package edu.utdallas.utdesign.teach4service.resources.requests;

import ...

@Data
public class RegisterSessionRequest
{
    @NotNull
    private String        username;
    private int studentId;
    private long          subjectId;
    private ZonedDateTime date;
}
```

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

This is the format in which the form data of searching for a tutor need to be before being sent to its corresponding API endpoint

```java
package edu.utdallas.utdesign.teach4service.resources.requests;

import ...

@Data
public class FindByTutorRequest
{
    @NotNull
    private String username;
}
```

```java
package edu.utdallas.utdesign.teach4service.resources.requests;

import lombok.Data;

@Data
public class FindByTutorNameRequest
{
    private String firstName;
    private String lastName;
}
```

```java
package edu.utdallas.utdesign.teach4service.resources.requests;

import lombok.Data;

@Data
public class FindBySubject
{
    private long subjectId;
}
```

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

This is the DAO class for the Persons entity and table in the DB. It includes functions that create a person record, a function that retrieves an existing record, and a function that retrieves the ID

```
package edu.utdallas.utdesign.teach4service.db;

import ...

public class PersonDAO extends AbstractDAO<Person>
{
    public PersonDAO(SessionFactory factory) { super(factory); }

    @Override
    public Person get(Serializable id) { return super.get(id); }

    public Optional<Person> getExistingPerson(String firstName, String lastName, Date dob, String address, String city, String state, String country, String postalCo
    {
        return currentSession().createNativeQuery( sqlString: "SELECT * FROM `Persons` WHERE `firstName` = :firstname and `lastName` = :lastname" +
                " and `dateOfBirth` = :dob and `streetAddress` = :address and `city` = :city " +
                "and `state` = :state and `country` = :country and `postalCode` = :postalCode", Person.class)
                .setParameter( name: "firstname", firstName)
                .setParameter( name: "lastname",lastName)
                .setParameter( name: "dob",dob)
                .setParameter( name: "address",address)
                .setParameter( name: "city",city)
                .setParameter( name: "state",state)
                .setParameter( name: "country",country)
                .setParameter( name: "postalCode",postalCode)
                .uniqueResultOptional();
    }



    public Person create(String firstName, String lastName, Date dob, String address, String city, String state, String country, String postalCode){
        if(!getExistingPerson(firstName,lastName,dob,address,city,state,country,postalCode).isPresent()){
            Person person = new Person();
            person.setFirstName(firstName);
            person.setLastName(lastName);

            person.setDateOfBirth(dob);
            person.setStreetAddress(address);
            person.setCity(city);
            person.setState(state);
            person.setCountry(country);
            person.setPostalCode(postalCode);
            return persist(person);
        }
        else{
            return getExistingPerson(firstName,lastName,dob,address,city,state,country,postalCode).get();
        }


    }
}
```

This function creates a new record in the Schedule table

```
public Schedule create(Tutor t, String day, String time){
    Schedule s = new Schedule();
    s.setTutorId(t.getId());
    s.setTutor(t);
    s.setTimeAvailableDescription(time);
    Days d = Days.valueOf(day);
    s.setDayAvailable(d);
    return persist(s);
}
```

This function creates a new record in the Sessions table

```
public Sessions create(long studentId, long tutorId, long subjectId, ZonedDateTime startTime)
{
    Sessions session = new Sessions();
    session.setStudentId(studentId);
    session.setTutorId(tutorId);
    session.setSubjectId(subjectId);
    session.setStartDateTime(startTime);
    return persist(session);
}
```

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

This is the DAO class for the Session Request entity and table in the DB. It includes functions that create a session request record, that retrieves the ID, that retrieves a student's request, and that retrieves a tutor's request

```
package edu.utdallas.utdesign.teach4service.db;

import ...

public class SessionRequestDAO extends AbstractDAO<SessionRequest>
{
    public SessionRequestDAO(SessionFactory factory) { super(factory); }

    @Override
    public SessionRequest get(Serializable id) { return super.get(id); }

    public List<SessionRequest> getStudentRequests(long sId){
        return currentSession().createNativeQuery( sqlString: "SELECT * FROM `SessionRequests` WHERE `requestedFor` = :sId", SessionRequest.class)
                .setParameter( name: "sId", sId)
                .getResultList();
    }

    public List<SessionRequest> getTutorRequests(long tId){
        return currentSession().createNativeQuery( sqlString: "SELECT * FROM `SessionRequests` WHERE `requestedTo` = :tId", SessionRequest.class)
                .setParameter( name: "tId", tId)
                .getResultList();
    }

    public SessionRequest create(long studentId, long tutorId, long subjectId, ZonedDateTime sessionDate){
        SessionRequest sr =  new SessionRequest();
        sr.setRequestedFor(studentId);
        sr.setRequestedTo(tutorId);
        sr.setStatus(SessionRequestStatus.PENDING);
        sr.setSubjectId(subjectId);
        sr.setSessionDateTime(sessionDate);

        return persist(sr);
    }
}
```

A function in User DAO that returns the user object that corresponds to a student ID

```
public Optional<User> getuser(long studentId){
    return currentSession().createNativeQuery( sqlString: "SELECT * FROM `Users` WHERE `studentId` = :studentId", User.class)
            .setParameter( name: "studentId", studentId)
            .uniqueResultOptional();
}
```

A function in Tutor Expertise DAO that returns the all the tutor expertise objects that corresponds to a given subject ID

```
public List<TutorExpertise> findBySubject(long id){
    return currentSession().createNativeQuery( sqlString: "SELECT * FROM `TutorExpertise` WHERE `subjectId` = :sId", TutorExpertise.class)
            .setParameter( name: "sId", id)
            .getResultList();
}
```

A function in Person DAO that returns all person objects that corresponds to a given first and last name

```
private List<Person> getByName(String firstName, String lastName){
    if(!firstName.isEmpty() && !lastName.isEmpty()){
        return currentSession().createNativeQuery( sqlString: "SELECT * FROM `Persons` WHERE `firstName` =:fname and `lastName` = :name", Person.class)
                .setParameter( name: "name", lastName)
                .setParameter( name: "fname",firstName)
                .getResultList();
    }else if(!firstName.isEmpty()){
        return currentSession().createNativeQuery( sqlString: "SELECT * FROM `Persons` WHERE `firstName` = :name", Person.class)
                .setParameter( name: "name", firstName)
                .getResultList();
    }
    return currentSession().createNativeQuery( sqlString: "SELECT * FROM `Persons` WHERE `lastName` = :name", Person.class)
            .setParameter( name: "name", lastName)
            .getResultList();
}
```

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

A function in Tutor DAO that returns all the tutor objects that corresponds to a given first and last name

```
public List<Tutor> getTutors(String firstName, String LastName){
    List<Person> p = getByName(firstName,LastName);
    List <Tutor> t = new ArrayList();
    for(int i=0;i<p.size();i++){
        if(getByPersonId(p.get(i).getId()).isPresent()){
            t.add(getByPersonId(p.get(i).getId()).get());
        }
    }
    return t;
}
```

This is the DAO class for the Subject entity and table in the DB. It includes functions that create a subject record, that retrieves the ID, that retrieves a subject by name, that retrieves a subject by field, and that retrieves all subject records in the DB

```
package edu.utdallas.utdesign.teach4service.db;

import ...

public class SubjectDAO extends AbstractDAO<Subject>
{
    public SubjectDAO(SessionFactory factory) { super(factory); }

    @Override
    public Subject get(Serializable id) { return super.get(id); }

    public Optional<Subject> getByName(String subName)
    {
        return currentSession().createNativeQuery( sqlString: "SELECT * FROM `Subjects` WHERE `name` = :name", Subject.class)
                .setParameter( name: "name", subName)
                .uniqueResultOptional();
    }

    public List<Subject> getSubjects(String field){
        return currentSession().createNativeQuery( sqlString: "SELECT * FROM `Subjects` WHERE `field` = :name", Subject.class)
                .setParameter( name: "name", field.toUpperCase())
                .getResultList();
    }

    public List<Subject> getSubjects(){
        return currentSession().createNativeQuery( sqlString: "SELECT * FROM `Subjects`", Subject.class)
                .getResultList();
    }

    public Subject create(String name, String field, String description){
        if(!getByName(name).isPresent()){
            Subject subject = new Subject();
            subject.setName(name);
            subject.setDescription(description);
            SubjectField s = SubjectField.valueOf(field.toUpperCase());

            subject.setField(SubjectField.valueOf(field.toUpperCase()));

            return persist(subject);
        }
        else{
            return getByName(name).get();
        }
        //TODO check if subject exists
    }
```

This is the DAO class for the Tutor Expertise entity and table in the DB. It includes functions that create a tutor expertise record, that retrieves the ID, that retrieves an existing tutor expertise object given a tutor and subject object

CS4485 Name: Gladys Adjei. Netid: gxa151130 About: Senior Design

```java
package edu.utdallas.utdesign.teach4service.db;

import ...

public class TutorExpertiseDAO extends AbstractDAO<TutorExpertise>
{
    public TutorExpertiseDAO(SessionFactory factory) { super(factory); }

    @Override
    public TutorExpertise get(Serializable id) { return super.get(id); }

    public Optional<TutorExpertise> exists(Tutor tutor, Subject subject)
    {
        return currentSession().createNativeQuery( sqlString: "SELECT * FROM `TutorExpertise` WHERE `tutorId` = :tutorId and `subjectId` =:subId", TutorExpertise.class)
                .setParameter( name: "tutorId",tutor.getId()).setParameter( name: "subId",subject.getId())
                .uniqueResultOptional();
    }

    public TutorExpertise create(Tutor tutor, Subject subject){
        TutorExpertise tutorExpertise = new TutorExpertise();
        tutorExpertise.setTutorId(tutor.getId());
        tutorExpertise.setTutor(tutor);
        tutorExpertise.setSubjectId(subject.getId());
        tutorExpertise.setSubject(subject);

        return persist(tutorExpertise);
    }
}
```

JAVA Entity Classes

Days class that contains the enumeration for days of the week

```java
package edu.utdallas.utdesign.teach4service.db.entities;

public enum Days
{
    MONDAY,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY,
    SUNDAY
}
```

Subject Field class that contains the enumeration for the list of course subject fields

```java
package edu.utdallas.utdesign.teach4service.db.entities;

public enum SubjectField
{
    MATH,
    SCIENCE,
    LANGUAGE,
    ART,
    SOCIAL,
    OTHER
}
```

Entity Class for Schedules table in the database

```java
package edu.utdallas.utdesign.teach4service.db.entities;

import ...
import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.Data;

import javax.persistence.*;
import javax.validation.constraints.NotNull;
import java.time.ZonedDateTime;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @Enumerated (EnumType.STRING)
    private Days dayAvailable;

    @NotNull
    private String timeAvailableDescription;

    private long tutorId;


    @JsonIgnore
    @OneToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "tutorId", insertable = false, updatable = false)
    private Tutor tutor;

    private ZonedDateTime createdOn    = ZonedDateTime.now();
    private  ZonedDateTime lastModified = ZonedDateTime.now();

    //scroll up lol

}
```

Entity Class for Subjects table in the database

```java
package edu.utdallas.utdesign.teach4service.db.entities;

import ...

@Data
@Entity
@Table(name = "Subjects",
        uniqueConstraints = {
                @UniqueConstraint(columnNames = {"name"})
        }
)
public class Subject
{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @Enumerated (EnumType.STRING)
    private SubjectField field;

    private String name;
    private String description;

    private ZonedDateTime createdOn = ZonedDateTime.now();
}
```

Entity Class for Session Requests table in the database

```java
package edu.utdallas.utdesign.teach4service.db.entities;

import ...

@Data
@Entity
@Table(name = "SessionRequests")
public class SessionRequest
{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    private long requestedFor;
    private long requestedTo;
    private long subjectId;

    @Enumerated(EnumType.STRING)
    private SessionRequestStatus status;
    private ZonedDateTime        sessionDateTime;

    @JsonIgnore
    @OneToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "subjectId", insertable = false, updatable = false)
    private Subject subject;

    @JsonIgnore
    @OneToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "requestedTo", insertable = false, updatable = false)
    private Tutor tutor;

    @JsonIgnore
    @OneToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "requestedFor", insertable = false, updatable = false)
    private Student student;

    private ZonedDateTime createdOn    = ZonedDateTime.now();
    private ZonedDateTime lastModified = ZonedDateTime.now();
}
```

Entity Class for Tutor Expertise table in the database

```java
package edu.utdallas.utdesign.teach4service.db.entities;

import ...

@Data
@Entity
@Table(name = "TutorExpertise")
public class TutorExpertise
{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "tutorId", insertable = false, updatable = false)
    private Tutor tutor;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "subjectId", insertable = false, updatable = false)
    private Subject subject;

    private Long tutorId;

    private Long subjectId;

    int rating = 0;

    private ZonedDateTime createdOn = ZonedDateTime.now();
}
```