

## Table of Contents

<b>Work Flow .....</b>	<b>2</b>
<b>Profile Setup.....</b>	<b>2</b>
<b>Profile Management.....</b>	<b>3</b>
<b>Edit Profile.....</b>	<b>3</b>
<b>Public Profile .....</b>	<b>4</b>
<b>Add Additional Profiles .....</b>	<b>5</b>
<b>User Dashboard .....</b>	<b>7</b>
<b>Search and Find Tutor.....</b>	<b>8</b>
<b>Register with Tutor .....</b>	<b>8</b>
<b>Session Registering Requests .....</b>	<b>9</b>
<b>Past Session History .....</b>	<b>10</b>
<b>Admin Login .....</b>	<b>10</b>
<b>Admin Functionalities.....</b>	<b>11</b>
<b>Admin Manage Subjects .....</b>	<b>11</b>
<b>Admin Manage Tutor .....</b>	<b>12</b>
<b>Admin Manages Other Admin Accounts .....</b>	<b>12</b>
<b>Implementation.....</b>	<b>12</b>
<b>HTML Code.....</b>	<b>13</b>
<b>Angular JavaScript .....</b>	<b>26</b>
<b>Database SQL .....</b>	<b>37</b>
<b>Backend API Endpoints.....</b>	<b>40</b>
<b>Endpoint Request Classes .....</b>	<b>47</b>
<b>JAVA Entity Classes .....</b>	<b>54</b>

## Work Flow

### Profile Setup

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

The screenshot shows the 'Setup Account Profile' section of the T4S Teach4Success website. At the top, there is a navigation bar with tabs for STUDENT (which is selected), TUTOR, and PARENT. Below the navigation bar, the title 'Setup Account Profile' is centered. The form fields include:

- First Name \* (text input)
- Last Name \* (text input)
- Date of Birth \* (date input, showing 11/22/2004)
- Grade Level \* (dropdown menu, showing 1)
- Street Address \* (text input)
- Select country (dropdown menu)
- Select state (dropdown menu)
- City \* (text input)
- Postal Code \* (text input)

At the bottom right of the form area is a blue button labeled 'COMPLETE STUDENT'.

Figure 1.1 Student Profile Setup

The screenshot shows the 'Setup Account Profile' section of the T4S Teach4Success website for tutors. The interface is similar to the student setup, with tabs for STUDENT, TUTOR (selected), and PARENT at the top. The 'Setup Account Profile' title is centered. The form fields are identical to Figure 1.1, including First Name, Last Name, Date of Birth, Grade Level, Street Address, and location selection. Below the address fields, there is a section titled 'Years of Experience' which is currently empty. Underneath the address section is a table titled 'My Schedule' with columns for 'select', 'Day', and 'Time (Default will be set to all day)'. The days listed are MONDAY through SUNDAY, each with 'All day' selected. At the bottom of the page, there is a section titled 'Setup Tutor Expertise' with three entries: SCIENCE, Biology, and test7. There are 'ADD EXPERTISE' and 'COMPLETE TUTOR' buttons at the bottom right.

Figure 1.2 Tutor Profile Setup

The screenshot shows the 'Setup Account Profile' form. At the top, there are tabs for 'STUDENT', 'TUTOR', and 'PARENT'. The 'PARENT' tab is selected. Below the tabs, the form fields include 'First Name \*' (empty), 'Last Name \*' (empty), 'Date of Birth \*' (set to '11/22/2004'), 'Street Address \*' (empty), 'Select country' (empty), 'Select state' (empty), 'City \*' (empty), and 'Postal Code \*' (empty). A blue button at the bottom right says 'COMPLETE PARENT'.

Figure 1.3 Parent Profile Setup

## Profile Management

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

The screenshot shows the 'Account Profile' management interface. On the left, a sidebar has a red 'Profile' button and dropdown menus for 'Student', 'Tutor', and 'Parent'. The main area displays three profile cards: 'Student' (public view), 'Tutor' (public view), and 'Parent' (public view). Each card has a pencil icon for editing. At the top right, it says 'WELCOME! GLADYS' and 'LOGOUT'.

Figure 2. View User Profiles

## Edit Profile

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

The screenshot shows the 'Account Profile' section of the T4S application. At the top right, it says 'WELCOME! GLADYS' and has a 'LOGOUT' link. On the left, there's a sidebar with a red 'Profile' button and dropdown menus for 'Student', 'Tutor', and 'Parent'. The main form has a blue header with a checked checkbox and the text 'Account Profile'. Below it, a purple link says '<< Profiles'. The profile fields include 'First Name \*' (Gladys), 'Last Name \*' (Adj), 'Street Address \*' (123 lane), 'City \*' (Dallas), 'State' (Texas), 'Postal Code \*' (12345), and a dropdown for 'Country' (United States of America (the)). A red 'UPDATE PROFILE' button is at the bottom right.

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

The screenshot shows the 'Account Profile' section of the T4S application, similar to Figure 3 but without the edit functionality. It features a large circular placeholder for a profile picture with the word 'STUDENT' below it. To its right, the name 'Gladys Adj' is displayed. Below the name, the text 'gradeLevel:1' is shown. At the bottom, there are three pieces of information: 'City: Dallas', 'State: Texas', and 'Country: United States of America (the)'. The overall layout is clean and modern.

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

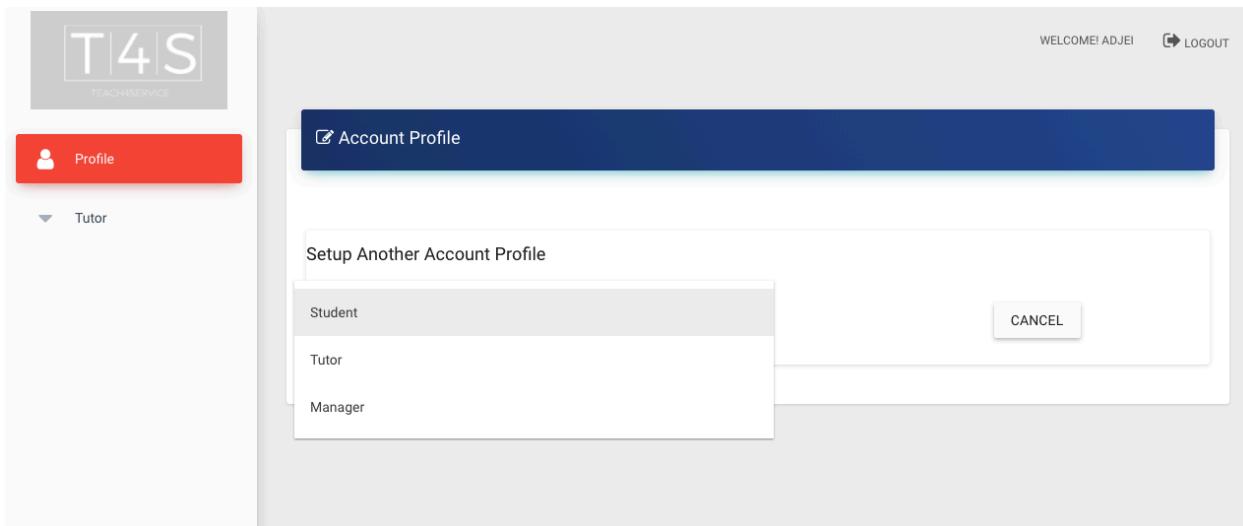


Figure 5.1 Options to add additional profiles

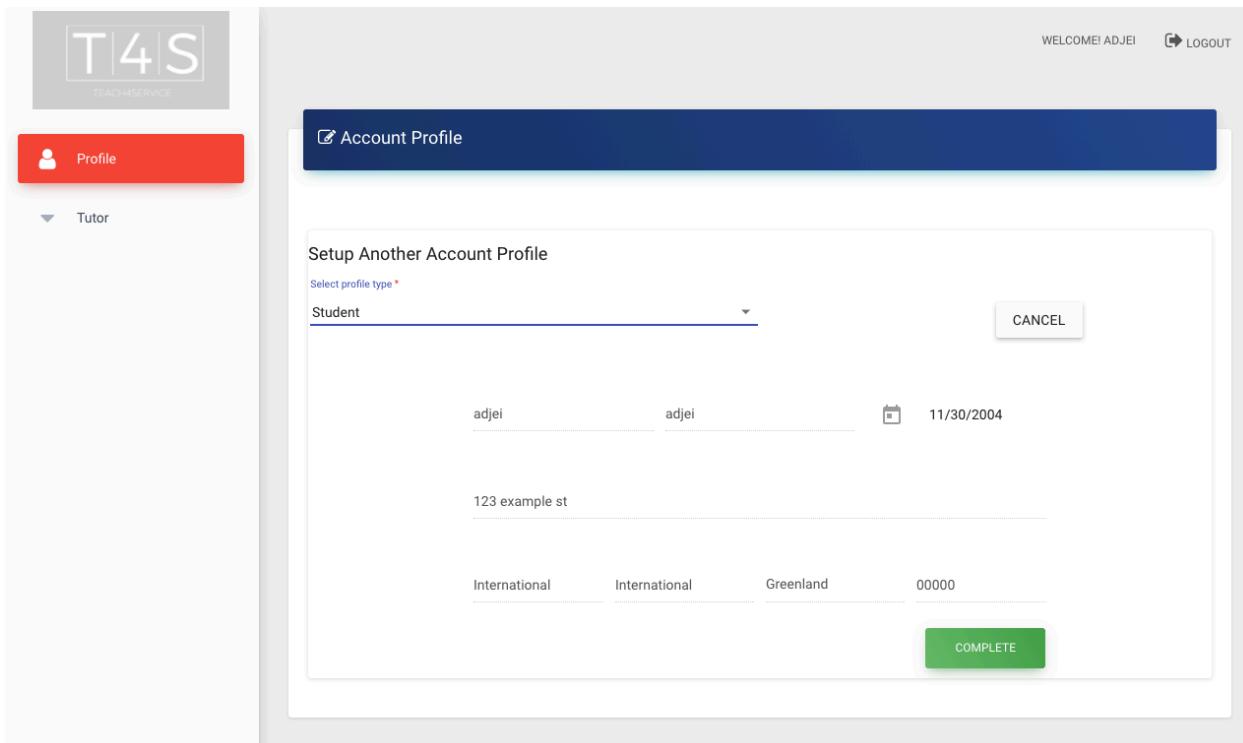


Figure 5.2 Adding a student profile

The screenshot shows the T4S Teach4Service interface. At the top, there's a header with the logo 'T4S TEACH4SERVICE'. Below it, a red navigation bar contains a profile icon and the word 'Profile'. A dropdown menu labeled 'Tutor' is open. The main content area has several input fields: 'adjei' (twice), a date field '11/30/2004', and an address '123 example st'. There are also dropdown menus for 'International' (selected twice) and 'Greenland', and a text input '00000'. A section titled 'Years of Experience' is present. Below this is a table titled 'My Schedule' with rows for each day of the week, all set to 'All day'. A section for 'Setup Tutor Expertise' follows, with fields for 'select subject field' and 'select subject name', and a 'ADD EXPERTISE' button. The overall layout is clean with a light gray background.

Figure 5.2 Adding a tutor profile

This screenshot shows a modal dialog box titled 'Account Profile' with a blue header bar containing a checkmark icon and the text 'Account Profile'. The main body of the dialog is titled 'Setup Another Account Profile' and includes a dropdown for 'Select profile type \*' with 'Manager' selected. It contains fields for 'adjei' (twice), a date field '11/30/2004', an address '123 example st', and dropdowns for 'International' (selected twice) and 'Greenland', with a text input '00000'. At the bottom right of the dialog is a green 'COMPLETE' button. The background of the main page shows the same profile information as Figure 5.2, including the 'T4S TEACH4SERVICE' header, the 'Profile' navigation bar, and the 'Tutor' dropdown.

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

The screenshot shows the T4S Teach4Service student dashboard. At the top right, it says "WELCOME! GLADYS" and has a "LOGOUT" button. On the left, there's a sidebar with a "Profile" icon, a red "Student" button, a "Dashboard" icon, a "Find Tutor" icon, a "Requests" icon, and "Session History" icon. Below these are dropdown menus for "Tutor" and "Parent". The main area has two sections: "Upcoming Sessions" (with columns for Subject, Tutor, Date, Time, and Status) and "Recent Reviews" (with a star icon).

Figure 5.1 View of student dashboard

The screenshot shows the T4S Teach4Service parent dashboard. At the top right, it says "WELCOME! GLADYS" and has a "LOGOUT" button. On the left, there's a sidebar with a "Profile" icon, a red "Parent" button, a "Dashboard" icon, a "Find Tutor" icon, a "Requests" icon, and a "Session History" icon. The main area has two sections: "Today Sessions" (with columns for Student, Subject, Tutor, Date, Time, and Status) and "Manages" (with columns for Student First Name, Student Last Name, and Student Grade Level). A table row shows "Jane" and "Lu" with a grade level of "0". At the bottom right of this section is a blue "ADD STUDENT" button.

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

Username	YearsOfExperience	State	Country	Rating
yyy	2	Alaska	Afghanistan	☆
darrensquires	60	Texas	United States of America (the)	☆

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

The screenshot shows the T4S platform interface from a parent's perspective. The left sidebar has a red 'Parent' button highlighted. The main area shows a search result for 'denney' with details: Name: Testing Denney, City: Plano, State: Texas, Country: United States of America (the), and Schedule: [empty]. Below this is a registration form with fields for Student (Jane Lu), Subject (Geometry), Date (12/1/2017), Time (12:30am), and a 'REGISTER' button.

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

The screenshot shows the T4S platform interface from a student's perspective. The left sidebar has a blue 'Requests' button highlighted. The main area shows a header 'My Requests' with columns for Request Submission Date, Tutor Username, Subject, Session Time, and Status. Below this is a table showing one request: Student: Jane Lu, Subject: Geometry, Session Time: 12/1/2017 12:30am, Status: Pending.

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

The screenshot shows the T4S Tutor dashboard. On the left sidebar, under the 'Tutor' section, the 'Requests' option is selected. At the top right, it says 'WELCOME! GLADYS' and 'LOGOUT'. The main content area has two sections: 'My Pending Requests' and 'Responded Requests'. The 'Pending Requests' section is currently empty. The 'Responded Requests' section shows one entry:

Request Submission Date	Student Username	Subject	Session Time
11/27/2017 09:36 PM	kfloter	Calculus AB	11/30/2017 12:00 AM

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

## Past Session History

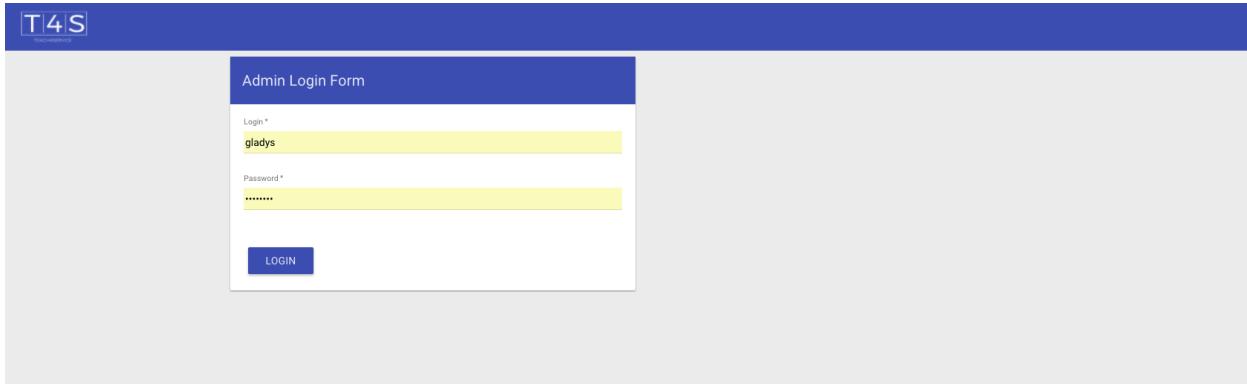
A logged in student will view sessions he/she has had in the past

The screenshot shows the T4S Student dashboard. On the left sidebar, under the 'Student' section, the 'Session History' option is selected. At the top right, it says 'WELCOME! GLADYS' and 'LOGOUT'. The main content area shows a table titled 'My Session History' with the following columns: Tutor, Subject, StartTime, EndTime, and Description. The table is currently empty.

Figure 9 Student past session history

## Admin Login

An added admin will login with their username and admin password



The screenshot shows the 'Admin Login Form' interface. At the top left is the T4S logo. The main area has a blue header bar with the text 'Admin Login Form'. Below this is a white form with two input fields: 'Login \*' containing the value 'gladys' and 'Password \*' containing several dots. A blue 'LOGIN' button is at the bottom right of the form.

Figure 10 Admin login

## Admin Functionalities

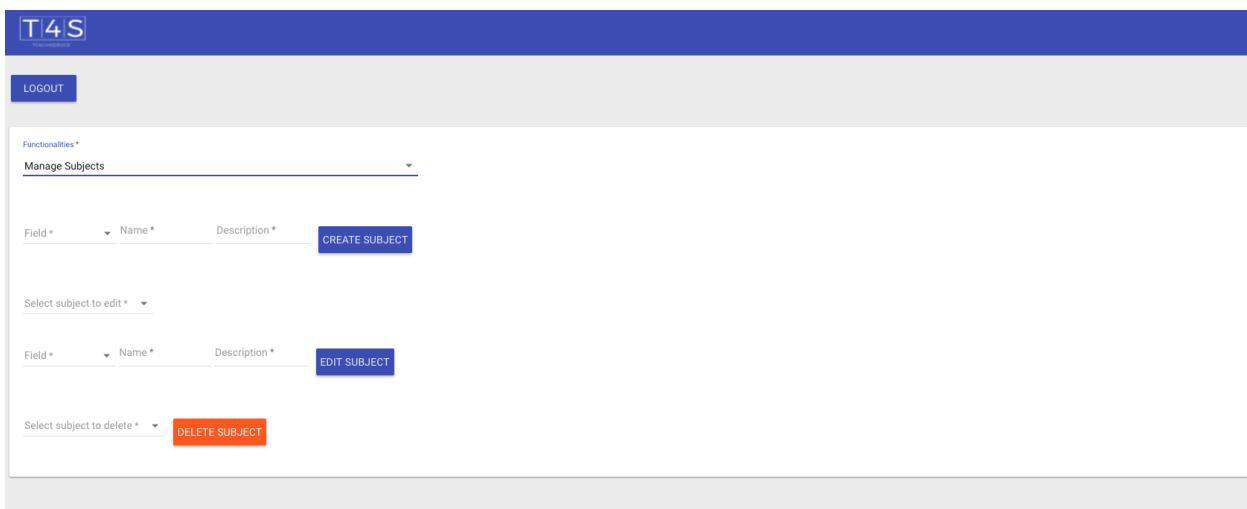


The screenshot shows the 'Manage Subjects' section of the admin interface. At the top left is the T4S logo. A blue 'LOGOUT' button is in the top left corner. On the left, there is a sidebar with three options: 'Manage Subjects' (which is selected and highlighted in grey), 'Manage Tutor', and 'Manage Admin Accounts'. The main content area is currently empty.

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

## Admin Manage Subjects

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



The screenshot shows the 'Manage Subjects' page. At the top left is the T4S logo. A blue 'LOGOUT' button is in the top left corner. The main content area has a sub-header 'Functionality \*' with 'Manage Subjects' selected. It contains three sections: 1) A 'CREATE SUBJECT' form with fields for 'Field \*', 'Name \*', and 'Description \*', followed by a blue 'CREATE SUBJECT' button. 2) An 'EDIT SUBJECT' form with similar fields and a blue 'EDIT SUBJECT' button. 3) A 'DELETE SUBJECT' section with a dropdown menu for 'Select subject to delete \*' and a red 'DELETE SUBJECT' button.

Figure 12 Admin manages subjects

## Admin Manage Tutor

A logged in admin can approve or decline a tutor

The screenshot shows a web application interface for managing a tutor. At the top, there is a blue header bar with the T4S logo and a LOGOUT button. Below the header, a dropdown menu titled "Functionality's \*" is open, showing "Manage Tutor". The main content area displays a tutor's profile information: Name: adjei adjei, Date of Birth: 2004-12-01, Address: 123 example st International International Greenland 00000, Years of Experience: 2, Expertise: SCIENCE Biology. Below the profile are two buttons: "APPROVE" (blue) and "DECLINE" (orange).

Figure 13 Admin manages tutor

## Admin Manages Other Admin Accounts

A logged in admin can create or delete an admin

The screenshot shows a web application interface for managing other admin accounts. At the top, there is a blue header bar with the T4S logo and a LOGOUT button. Below the header, a dropdown menu titled "Functionality's \*" is open, showing "Manage Admin Accounts". The main content area has input fields for "username", "email", and "password", followed by a "CREATE ADMIN" button. Below these fields is a link "Select an admin to edit \*". At the bottom of the page is a "DELETE ADMIN" button.

Figure 14 Admin manages admins

## Implementation

## HTML Code

This code is the layout for profile management



A screenshot of a code editor showing the content of a file named "profileLayout.html". The code is written in HTML and uses some CSS classes and AngularJS directives. It defines a "card" structure with a header containing a title and a content area.

```
<div class="card">
  <div class="card-header" data-background-color="blue">
    <h4><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>
```



A screenshot of a code editor showing the content of a file named "initialProfileSetup.html". This file contains more complex HTML, including a navigation bar with tabs for Student, Tutor, Parent, and Logout, and a main section for setting up an account profile.

```
<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

```

1 <div class="col-md-2"></div>
2 <div class="col-md-8">
3   <form ng-submit="completeProfile()">
4     <div layout="row">
5       <md-input-container flex="25">
6         <label>First Name</label>
7         <input type="text" ng-model="student.firstName" required>
8       </md-input-container>
9       <md-input-container flex="25">
10        <label>Last Name</label>
11        <input type="text" ng-model="student.lastName" required>
12      </md-input-container>
13      <md-input-container flex="25">
14        <label>Date of Birth</label>
15        <md-datepicker md-placeholder="Date of Birth"
16                      ng-model="student.dateOfBirth" md-min-date="minDate" md-max-date="maxDate"
17                      required></md-datepicker>
18      </md-input-container>
19      <md-input-container flex="25">
20        <label>Grade Level</label>
21        <input type="number" ng-model="student.gradeLevel" required min="1">
22      </md-input-container>
23    </div>
24    <div layout="row">
25      <md-input-container flex="100">
26        <label>Street Address</label>
27        <input type="text" ng-model="student.streetAddress" required>
28      </md-input-container>
29    </div>
30  </div>
31  <div layout="row">
32    <md-input-container flex="25">
33      <select class="form-control" ng-model="student.country" pvp-country-picker="name" required>
34        <option value="">Select country</option>
35      </select>
36    </md-input-container>
37    <md-input-container flex="25">
38      <select class="form-control" ng-model="student.state" required>
39        <option value="">Select state</option>
40        <option value="{{s}}" ng-repeat="s in states">{{s}}</option>
41      </select>
42    </md-input-container>
43    <md-input-container flex="25">
44      <label>City</label>
45      <input type="text" ng-model="student.city" required>
46    </md-input-container>
47    <md-input-container flex="25">
48      <label>Postal Code</label>
49      <input type="text" ng-model="student.postalCode" minlength="5" maxlength="10" required>
50    </md-input-container>
51  </div>
52  <md-button type="submit" class="md-raised md-primary" style="...">complete student</md-button>
53 </form>
54 </div>
55 </div class="col-md-2"></div>

```

This code is for parent profile setup

```

1 <div class="col-md-2"></div>
2 <div class="col-md-8">
3   <form ng-submit="completeProfile()">
4     <div layout="row">
5       <md-input-container flex="33">
6         <label>First Name</label>
7         <input type="text" ng-model="manager.firstName" required>
8       </md-input-container>
9       <md-input-container flex="33">
10        <label>Last Name</label>
11        <input type="text" ng-model="manager.lastName" required>
12      </md-input-container>
13      <md-input-container flex="33">
14        <label>Date of Birth</label>
15        <md-datepicker md-placeholder="Date of Birth"
16                      ng-model="manager.dateOfBirth" md-min-date="minDate" md-max-date="maxDate"
17                      required></md-datepicker>
18      </md-input-container>
19    </div>
20    <div layout="row">
21      <md-input-container flex="100">
22        <label>Street Address</label>
23        <input type="text" ng-model="manager.streetAddress" required>
24      </md-input-container>
25    </div>
26  </div>
27  <div layout="row">
28    <md-input-container flex="25">
29      <select class="form-control" ng-model="manager.country" pvp-country-picker="name" required>
30        <option value="">Select country</option>
31      </select>
32    </md-input-container>
33    <md-input-container flex="25">
34      <select class="form-control" ng-model="manager.state" required>
35        <option value="">Select state</option>
36        <option value="{{s}}" ng-repeat="s in states">{{s}}</option>
37      </select>
38    </md-input-container>
39    <md-input-container flex="25">
40      <label>City</label>
41      <input type="text" ng-model="manager.city" required>
42    </md-input-container>
43    <md-input-container flex="25">
44      <label>Postal Code</label>
45      <input type="text" ng-model="manager.postalCode" minlength="5" maxlength="10" required>
46    </md-input-container>
47  </div>
48  <md-button type="submit" class="md-raised md-primary" style="...">complete parent</md-button>
49 </form>
50 </div>
51 </div class="col-md-2"></div>

```

This code is for tutor profile setup

```

1 <div class="col-md-8"></div>
2
3 <div class="col-md-8">
4   <form ng-submit="complete()">
5     <div layout="row">
6       <md-input-container flex="33">
7         <label>First Name</label>
8         <input type="text" ng-model="tutor.firstName" required>
9       </md-input-container>
10      <md-input-container flex="33">
11        <label>Last Name</label>
12        <input type="text" ng-model="tutor.lastName" required>
13      </md-input-container>
14      <md-input-container flex="33">
15        <label>Date of Birth</label>
16        <md-datepicker md-placeholder="Date of Birth"
17                      ng-model="tutor.dateOfBirth" md-min-date="minDate" md-max-date="maxDate"
18                      required></md-datepicker>
19      </md-input-container>
20    </div>
21    <div layout="row">
22      <md-input-container flex="100">
23        <label>Street Address</label>
24        <input type="text" ng-model="tutor.streetAddress" required>
25      </md-input-container>
26
27    </div>
28    <div layout="row">
29
30      <md-input-container flex="25">
31        <select class="form-control" ng-model="tutor.country" pvp-country-picker="name" required>
32          <option value="">Select country</option>
33        </select>
34      </md-input-container>
35      <md-input-container flex="25">
36        <select class="form-control" ng-model="tutor.state" required>
37          <option value="">Select state</option>
38          <option value="{{s}}" ng-repeat="s in states">{{s}}</option>
39        </select>
40      </md-input-container>
41      <md-input-container flex="25">
42        <label>City</label>
43        <input type="text" ng-model="tutor.city" required>
44      </md-input-container>
45      <md-input-container flex="25">
46        <label>Postal Code</label>
47        <input type="text" ng-model="tutor.postalCode" minlength="5" maxlength="10" required>
48      </md-input-container>
49
50    </div>
51
52    <div class="row">
53      <div>
54        <div class="form-group label-floating">
55          <input type="number" class="form-control" placeholder="Years of Experience" min="0" max="70"
56                      ng-model="tutor.expYears" required>
57        </div>
58      </div>
59    </div>
60

```

```

60
61
62
63
64
65 <h4>My Schedule</h4>
66 <table class="table table-bordered">
67   <thead>
68     <tr>
69       <th>select</th>
70       <th>Days</th>
71       <th>Time </th>
72     </tr>
73   </thead>
74   <tbody>
75     <tr ng-repeat="x in schedules">
76       <td ><input type="checkbox" ng-model="x.selected"></td>
77       <td ng-model="x.day">{{x.day}}</td>
78       <td class="form-inline"><input class="form-control" type="time" ng-model="x.start" >
79         </input>
80         <input class="form-control" type="time" ng-model="x.end">
81       </td>
82     </tr>
83   </tbody>
84 </table>
85
86 <br>
87 <h5>Setup Tutor Expertise
88
89 </h5>
90 <div class="row">
91   <table class="table">
92     <tbody>
93       <tr ng-repeat="e in tutor.expertise">
94         <td><p style="...><i class="fa fa-trash-o" aria-hidden="true" ng-click="remove(e)"></i></p></td>
95         <td>{{e.field}}</td>
96         <td>{{e.subject}}</td>
97         <td>{{e.description}}</td>
98       </tr>
99     </tbody>
100   </table>
101 </div>
102
103 <div class="row">
104   <div class="col-md-3 form-group">
105     <select class="form-control" ng-model="selectedField" ng-change="getFieldSubjects(selectedField.field)">
106       <option value="">select subject field</option>
107     </select>
108   </div>
109   <div class="col-md-3 form-group">
110     <select class="form-control" ng-model="selectedSubject" ng-options="x.name_for_x in expertise.subjects">
111       <option value="">select subject name</option>
112     </select>
113   </div>
114   <div class="col-md-6 form-group">
115     <input type="text" class="form-control" placeholder="{{selectedSubject.description}}" disabled>
116   </div>
117 </div>
118 <div class="form-group">
119

```

This code is for editing student and parent profile

```

2   <a ui-sref="app.account.profile.list">
3     &#8000; Profiles</a>
4   </div>
5
6 <form>
7   <div class="row">
8     <div class="col-md-6">
9       <div class="form-group label-floating">
10         <input type="text" class="form-control" placeholder="First Name">
11       </div>
12     </div>
13     <div class="col-md-6">
14       <div class="form-group label-floating">
15         <input type="text" class="form-control" placeholder="Last Name">
16       </div>
17     </div>
18   </div>
19   <div class="row">
20     <div class="col-md-12">
21       <div class="form-group label-floating">
22         <input type="text" class="form-control" placeholder="Email">
23       </div>
24     </div>
25   </div>
26   <div class="row">
27     <div class="col-md-12">
28       <div class="form-group label-floating">
29         <input type="text" class="form-control" placeholder="Address">
30       </div>
31     </div>
32   </div>
33   <div class="row">
34     <div class="col-md-3">
35       <div class="form-group label-floating">
36         <input type="text" class="form-control" placeholder="City">
37       </div>
38     </div>
39     <div class="col-md-3">
40       <div class="form-group label-floating">
41         <input type="text" class="form-control" placeholder="State/Province">
42       </div>
43     </div>
44   </div>
45 <div class="col-md-3">

```

```

1 <div>
2   <a ui-sref="app.account.profile.list">
3     &#xA0;&#xA0; Profiles</h5>
4   </a>
5 
6   <form ng-submit="update()">
7     <div layout="row">
8       <md-input-container flex="50">
9         <label>First Name</label>
10        <input type="text" ng-model="manager.firstName" required>
11      </md-input-container>
12      <md-input-container flex="50">
13        <label>Last Name</label>
14        <input type="text" ng-model="manager.lastName" required>
15      </md-input-container>
16    </div>
17    <div layout="row">
18      <md-input-container flex="25">
19        <label>Street Address</label>
20        <input type="text" ng-model="manager.streetAddress" required>
21      </md-input-container>
22      <md-input-container flex="25">
23        <label>City</label>
24        <input type="text" ng-model="manager.city" required>
25      </md-input-container>
26      <md-input-container flex="25">
27        <label>State</label>
28        <select class="form-control" ng-model="manager.state" required>
29          <option value="">Select state</option>
30          <option value="{{s}}>{{s}}</option>
31        </select>
32      </md-input-container>
33      <md-input-container flex="25">
34        <label>Postal Code</label>
35        <input type="text" ng-model="manager.postalCode" minlength="5" maxlength="5" required>
36      </md-input-container>
37    </div>
38    <div layout="row">
39      <md-input-container>
40        <select class="form-control" ng-model="manager.country" pvp-country-picker="name" required>
41          <option value="">Select country</option>
42        </select>
43      </md-input-container>
44    </div>
45    <button type="submit" class="btn btn-secondary pull-right" data-background-color="red">Update Profile</button>
46  </form>
47
48
49
50

```

This code is for student public profile

```

1 <div>
2   <a ui-sref="app.account.profile.list">
3     &#xA0;&#xA0; Profiles</h5>
4   </a>
5 
6   <div class="card card-profile">
7     <div class="card-avatar">
8       <a href="#pablo">
9         
10      </a>
11    </div>
12    <div class="content">
13      <h6 class="category text-gray">Student</h6>
14      <h4 class="card-title">{{student.firstName}} {{student.lastName}}</h4>
15      <div class="card-content">
16        <form>
17          <div class="row">
18            <div class="col-md-12">
19              <div class="form-group label-floating">
20                <label>GradeLevel: {{student.gradeLevel}}</label>
21              </div>
22            </div>
23            <div class="row">
24              <div class="col-md-3">
25                <div class="form-group label-floating">
26                  <label>City: {{student.city}}</label>
27                </div>
28              </div>
29              <div class="col-md-3">
30                <div class="form-group label-floating">
31                  <label>State: {{student.state}}</label>
32                </div>
33              </div>
34              <div class="col-md-3">
35                <div class="form-group label-floating">
36                  <label>Country: {{student.country}}</label>
37                </div>
38              </div>
39            </div>
40          </form>
41        </div>
42      </div>
43    </div>
44
45
46
47
48
49
49
50

```

This code is for student dashboard

```

1 <div class="card">
2   <div class="card-header" data-background-color="blue">
3     <h4 class="title"><i class="fa fa-calendar" aria-hidden="true"></i> Upcoming Sessions</h4>
4   </div>
5   <div class="card-content table-responsive">
6     <table class="table table-responsive">
7       <thead class="text-primary">
8         <th>Subject</th>
9         <th>Tutor</th>
10        <th>Date</th>
11        <th>Time</th>
12        <th>Status</th>
13      </thead>
14      <tbody ng-repeat="s in student.sessions">
15        <tr>
16          <td>{{s.subject.name}}</td>
17          <td>{{s.tutor.person.firstName}} {{s.tutor.person.lastName}}</td>
18          <td>{{(s.status === 'ENDED') ? s.endTime : s.startTime | date:'shortDate' }}</td>
19          <td>{{(s.status === 'ENDED') ? s.endTime : s.startTime | date:'shortTime' }}</td>
20          <td>
21            <td><!-- for the enter session button --></td>
22            <td>
23              <tbody ng-repeat="s in student.sessions">
24                <tr>
25                  <td>{{s.subject.name}}</td>
26                  <td>{{s.tutor.person.firstName}} {{s.tutor.person.lastName}}</td>
27                  <td>{{(s.status === 'ENDED') ? s.endTime : s.startTime | date:'shortDate' }}</td>
28                  <td>{{(s.status === 'ENDED') ? s.endTime : s.startTime | date:'shortTime' }}</td>
29                </tr>
30              </tbody>
31            </td>
32            <td class="actions text-right" ng-if="s.status != 'ENDED'>
33              <button ng-disabled="canJoin(s)" rel="tooltip" title="Enter Session" class="btn btn-success" style="... " ng-click="enter(s)">
34                <i class="fa fa-sign-in" aria-hidden="true"></i> Enter Session
35              </button>
36              <button rel="tooltip" title="Cancel Session" class="btn btn-danger" style="... " ng-click="endSession(s)">
37                <i class="fa fa-close" aria-hidden="true"></i> Cancel Session
38              </button>
39            </td>
40          </tr>
41        </tbody>
42      </table>
43    </div>
44    <div class="card">
45      <div class="card-header" data-background-color="blue">
46        <h4 class="title"><i class="fa fa-star-half-o" aria-hidden="true"></i> Recent Reviews</h4>
47      </div>
48      <div class="card-content table-responsive">
49        <table class="table table-responsive">
50          <tbody ng-repeat="r in student.tutorReviews">
51            <tr>
52              <td>
53                <h6><i class="fa fa-user-circle-o" aria-hidden="true"></i> {{r.tutor.person.firstName}} {{r.tutor.person.lastName}}</h6>
54                <p>Participation:{{r.participation}} out of 5</p>
55                <p>Effort:{{r.effort}} out of 5</p>
56                <p>Comprehension:{{r.comprehension}} out of 5</p>
57                <p>Respectfulness:{{r.respectfulness}} out of 5</p>
58                <p>Preparedness:{{r.preparation}} out of 5</p>
59              </td>
60            </tr>
61          </tbody>
62        </table>
63      </div>
64    </div>
65  ...
66

```

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.endTime : s.startTime | date:'shortDate' }}</td>
    <td>{{(s.status === 'ENDED') ? s.endTime : s.startTime | date:'shortTime' }}</td>
    <td>{{s.status}}</td>
  </tr>

```

```

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

```

This code is for displaying student session requests

```

1  <div class="card">
2    <div class="card-header" data-background-color="blue">
3      <h4 class="title"><i class="fa fa-calendar" aria-hidden="true"></i>
4        My Requests</h4>
5    </div>
6    <div class="card-content table-responsive">
7      <h4>{{errorMsg}}</h4>
8      <table class="table table-responsive">
9        <thead class="text-primary">
10       <th>Request Submission Date</th>
11       <th>Tutor Username</th>
12       <th>Subject</th>
13       <th>Session Time</th>
14       <th>Status</th>
15     </thead>
16     <tbody ng-repeat="sr in studentSessionRequests">
17       <tr>
18         <td>{{sr.requestDate}}</td>
19         <td>{{sr.tutorUsername}}</td>
20         <td>{{sr.subject}}</td>
21         <td>{{sr.time}}</td>
22         <td>{{sr.status}}</td>
23       </tr>
24     </tbody>
25   </table>
26 </div>
27 </div>

```

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

```
1  <div ng-show="search.showSearch">
2
3    <div class="card">
4      <div class="card-content">
5        <div class="row">
6          <div class="col-md-3">
7            <div class="form-group label-floating">
8              <select class="form-control" ng-model="searchBy" ng-options="x.for_x in searchCriteria"
9                  required>
10             <option value="">search by</option>
11           </select>
12         </div>
13       <form ng-if="searchBy == 'subject'" ng-submit="searchBySubject()">
14         <div>
15           <div class="col-md-3 form-group label-floating">
16             <select class="form-control" ng-model="search.subject" ng-options="x.name_for_x in expertise.subjects" required>
17               <option></option>
18             </select>
19           </div>
20           <div class="col-md-3">
21             <div layout-gt-xs="row">
22               <div class="form-group">
23                 <button type="submit" class="btn btn-success" data-background-color="green">Search
24                 </button>
25               </div>
26             </div>
27           </div>
28         </div>
29       </form>
30       <form ng-if="searchBy == 'tutor name'" ng-submit="searchByName()">
31         <div class="col-md-3">
32           <div class="form-group label-floating">
33             <input type="text" class="form-control" placeholder="tutor first name"
34                 ng-model="search.tutorFirstName">
35           </div>
36         </div>
37       </form>
38       <form ng-if="searchBy == 'tutor last name'" ng-submit="searchByLastName()">
39         <div class="col-md-3">
40           <div class="form-group label-floating">
41             <input type="text" class="form-control" placeholder="tutor last name"
42                 ng-model="search.tutorLastName">
43           </div>
44         </div>
45       </form>
46       <div class="col-md-3">
47         <div layout-gt-xs="row">
48           <div class="form-group">
49             <button type="submit" class="btn btn-success" data-background-color="green">Search
50             </button>
51           </div>
52         </div>
53       </div>
54     </div>
55   </form ng-if="searchBy == 'tutor username'" ng-submit="searchByTutor()">
56     <div class="col-md-3">
57       <div class="form-group label-floating">
58         <input type="text" class="form-control" placeholder="tutor username"
59             ng-model="search.tutorUsername" required>
60       </div>
61     ...</div>
```

```

62 <div class="col-md-3">
63   <div layout-gt-xs="row">
64     <div class="form-group">
65       <button type="submit" class="btn btn-success" data-background-color="green">Search
66     </div>
67   </div>
68 </div>
69 </div>
70 </div ng-if="!hide">
71 <div class="row">
72   <div class="col-md-9">
73     <h6>Available Now!</h6>
74     <table class="table table-striped table-responsive">
75       <thead>
76         <tr>
77           <th><i class="fa fa-sort" aria-hidden="true"></i> Username</th>
78           <th><i class="fa fa-sort" aria-hidden="true"></i> YearsOfExperience</th>
79           <th><i class="fa fa-sort" aria-hidden="true"></i> State</th>
80           <th><i class="fa fa-sort" aria-hidden="true"></i> Country</th>
81           <th><i class="fa fa-sort" aria-hidden="true"></i> Rating</th>
82         </tr>
83       </thead>
84       <tbody ng-repeat="r in filters.availables | filter:textsearch" ng-click="goToTutorPage(r)" style="...">
85         <tr>
86           <td>{{r.username}}</td>
87           <td>{{r.expYears}}</td>
88           <td>{{r.state}}</td>
89           <td>{{r.country}}</td>
90           <td><i class="fa fa-star-o" aria-hidden="true"></i></td>
91         </tr>
92       </tbody>
93     <br>
94     <h6>Tutor Results</h6>
95     <table class="table table-striped table-responsive">
96       <thead>
97         <tr>
98           <th><i class="fa fa-sort" aria-hidden="true"></i> Username</th>
99           <th><i class="fa fa-sort" aria-hidden="true"></i> YearsOfExperience</th>
100          <th><i class="fa fa-sort" aria-hidden="true"></i> State</th>
101          <th><i class="fa fa-sort" aria-hidden="true"></i> Country</th>
102          <th><i class="fa fa-sort" aria-hidden="true"></i> Rating</th>
103        </tr>
104      </thead>
105      <tbody ng-repeat="r in filters.results | filter:textsearch" ng-click="goToTutorPage(r)" style="...">
106        <tr>
107          <td>{{r.username}}</td>
108          <td>{{r.expYears}}</td>
109          <td>{{r.state}}</td>
110          <td>{{r.country}}</td>
111          <td><i class="fa fa-star-o" aria-hidden="true"></i></td>
112        </tr>
113      </tbody>
114    </table>
115  </div>
116 </div>
117 </div>
118 </div>
119 </div>
120 </div>
121 </div>

```

```

121   </tbody>
122 </table>
123 </div>
124 <div class="col-md-3" style="...">
125   <div class="row">
126     <div ng-if="search.showsearch">
127       <md-input-container>
128         <label>Search results</label>
129         <input ng-model="textsearch">
130       </md-input-container>
131       <form>
132         <p>Filter Search Results</p>
133         <div flex-gt-sm="90" ng-repeat="subject in expertise.subjects">
134           <md-checkbox aria-label="{{subject.name}}" ng-model="subject.selected">
135             {{subject.name}}
136           </md-checkbox>
137         </div>
138         <md-button class="md-raised md-primary" ng-click="filter()">apply filter</md-button>
139         <a style="..." ng-click="reset()">Reset filter</a>
140       </form>
141     </div>
142   </div>
143 </div>
144 </div>
145 </div>
146 </div>
147 </div>
148 <div ng-if="!search.showsearch">
149   <md-button class="md-raised" ng-click="search.showsearch = true">Go back to search</md-button>
150   <div class="card">
151     <div class="card-header" data-background-color="blue">
152       <i class="title"><i class="fa fa-calendar" aria-hidden="true"></i>
153       {{tutorUsername}}
154     </div>
155     <div class="card-content">
156       <div class="col-md-6">
157         <h4>{{tutor.profile.firstName}} {{tutor.profile.lastName}}</h4>
158         <h4>{{tutor.profile.city}}</h4>
159         <h4>{{tutor.profile.state}}</h4>
160         <h4>{{tutor.profile.country}}</h4>
161         <h4>{{tutor.schedule}}</h4>
162         <div>
163           <h4 ng-repeat="t in tutor.schedule">{{t.dayAvailable}} {{t.timeAvailableDescription}}</h4>
164         </div>
165       </div>
166       <div class="col-md-6">
167         <div class="list-group" ng-repeat="r in tutor.reviews">
168           <div>
169             <i class="fa fa-user-circle-o" aria-hidden="true"></i> <b>Reviewer:</b> {{r.student.person.firstName}} {{r.student.person.lastName}}
170             <div>
171               <p>Knowledge: {{r.knowledge}} out of 5</p>
172               <p>Helpfulness: {{r.helpful}} out of 5</p>
173               <p>Preparedness: {{r.prepared}} out of 5</p>
174             </div>
175           </div>
176         </div>
177       </div>
178     </div>
179   </div>

```

```

177      </div>
178    </div>
179  </div>
180
181  <div class="card">
182    <div class="card-content">
183      <form ng-submit="register()">
184        <div class="form-group label-floating">
185          <div class="col-md-3">
186            <div class="form-group label-floating">
187              <div>
188                <select class="form-control" ng-model="reg.subject"
189                  ng-options="x.name for x in tutor.subjects" required>
190                  <option value="">select subject</option>
191                </select>
192              </div>
193            </div>
194          </div>
195        <div class="col-md-3">
196          <div layout-gt-xs="row">
197            <div class="form-group">
198              <div flex-gt-xs>
199                <md-datepicker md-placeholder="date"
200                  ng-model="reg.date" required></md-datepicker>
201              </div>
202            </div>
203          </div>
204        </div>
205      </div>
206      <div class="col-md-3">
207        <div class="form-group label-floating">
208          <div>
209            <select class="form-control" ng-model="reg.time"
210              ng-options="x.name for x in time" required>
211              <option value="">select time</option>
212            </select>
213          </div>
214        </div>
215      </div>
216    </div>
217    <div layout-gt-xs="row">
218      <div class="form-group">
219        <button type="submit" class="btn btn-success" data-background-color="green">
220          Register
221        </button>
222      </div>
223    </div>
224  </div>
225</div>
226</div>
227</div>
228</div>
229</div>
230</div>
231</div>

```

```

1 <div ng-show="search.showSearch">
2   <div class="card">
3     <div class="card-content">
4       <div class="row">
5         <div class="col-md-3">
6           <div class="form-group label-floating">
7             <select class="form-control" ng-model="searchBy" ng-options="x.name for x in searchCriteria" required>
8               <option value="">search by</option>
9             </select>
10            </div>
11          </div>
12        <form ng-if="searchBy == 'subject'" ng-submit="searchBySubject()">
13          <div>
14            <div class="col-md-3">
15              <div class="form-group label-floating">
16                <select class="form-control" ng-model="search.subject"
17                  ng-options="x.name for x in expertise.subjects" required>
18                  <option></option>
19                </select>
20              </div>
21            </div>
22            <div class="col-md-3">
23              <div layout-gt-xs="row">
24                <div class="form-group">
25                  <button type="submit" class="btn btn-success" data-background-color="green">
26                    Search
27                  </button>
28                </div>
29              </div>
30            </div>
31          </div>
32        </form>
33        <form ng-if="searchBy == 'tutor name'" ng-submit="searchByName()">
34          <div class="col-md-3">
35            <div class="form-group label-floating">
36              <input type="text" class="form-control" placeholder="tutor first name"
37                  ng-model="search.tutorFirstName">
38            </div>
39          </div>
40        </form>
41        <div class="col-md-3">
42          <div class="form-group label-floating">
43            <input type="text" class="form-control" placeholder="tutor last name"
44                  ng-model="search.tutorLastName">
45          </div>
46        </div>
47        <div class="col-md-3">
48          <div layout-gt-xs="row">
49            <div class="form-group">
50              <button type="submit" class="btn btn-success" data-background-color="green">Search
51            </button>
52          </div>
53        </div>
54      </form>
55      <form ng-if="searchBy == 'tutor username'" ng-submit="searchByTutor()">
56        <div class="col-md-3">
57          <div class="form-group label-floating">
58            <input type="text" class="form-control" placeholder="tutor username"
59                  ng-model="search.tutorUsername" required>
60          </div>
61        </div>
62      </form>

```

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

```
140         </div>
141     </div>
142   </div>
143 </div>
144 </div>
145 </div>
146 </div>
147 <div ng-if="!search.showSearch">
148
149   <md-button class="md-raised" ng-click="search.showSearch = true">Go back to search</md-button>
150
151   <div class="card">
152     <div class="card-header" data-background-color="blue">
153       <h4 class="title"><i class="fa fa-calendar" aria-hidden="true"></i> {{tutor.username}}</h4>
154     </div>
155     <div class="card-content">
156       <div class="card-content">
157         <div class="col-md-5">
158           <h4>{{tutor.profile.firstName}} {{tutor.profile.lastName}}</h4>
159           <h4>{{tutor.profile.city}}</h4>
160           <h4>{{tutor.profile.state}}</h4>
161           <h4>{{tutor.profile.country}}</h4>
162           <h4>{{tutor.schedule}}</h4>
163         </div>
164         <div ng-repeat="t in tutor.schedule">{{t.dayAvailable}} {{t.timeAvailableDescription}}</h4>
165       </div>
166       <div class="col-md-7">
167         <div class="list-group">
168           <div>
169             <div class="list-group" ng-repeat="r in tutor.reviews">
170               <h6><i class="fa fa-user-circle-o" aria-hidden="true"></i> <b>Reviewer:</b>{{r.student.person.firstName}} {{r.student.person.lastName}}</h6>
171               <p>Knowledge:{{r.knowledge}} out of 5</p>
172               <p>Helpfulness:{{r.helpful}} out of 5</p>
173               <p>Preparedness:{{r.prepared}} out of 5</p>
174             </div>
175           </div>
176         </div>
177       </div>
178     </div>
179   </div>
180 </div>
181 </div>
182 </div>
183 </div>
184 </div>
185 </div>
186 </div>
187 <div class="card">
188   <div class="card-content">
189     <form no-submit="register()">
190       <div layout="row">
191         <div input-container flex="20">
192           <label>Student</label>
193           <md-select ng-model="reg.student" required>
194             <md-option ng-repeat="s in students" ng-value="s.id">
195               {{s.person.firstName}} {{s.person.lastName}}
196             </md-option>
197           </md-select>
198         </div input-container>
199         <div input-container flex="20">
200           <label>Subject</label>
```

```

171 <div class="list-group" ng-repeat="r in tutor.reviews">
172   <div>
173     <div class="fa-user-circle-o" aria-hidden="true"></i> <b>Reviewer:</b>{{r.student.person.firstName}} {{r.student.person.lastName}}
174     <p>Knowledge:{{r.knowledge}} out of 5</p>
175     <p>Helpfulness:{{r.helpful}} out of 5</p>
176     <p>Preparedness:{{r.prepared}} out of 5</p>
177   </div>
178 </div>
179 </div>
180 </div>
181 </div>
182 </div>
183 </div>
184 </div>
185 </div>
186 </div>
187 <div class="card">
188   <div class="card-content">
189     <form ng-submit="register()">
190       <div layout="row">
191         <md-input-container flex="20">
192           <label>Student</label>
193           <md-select ng-model="reg.student" required>
194             <md-option ng-repeat="s in students" ng-value="s.id">{{s.person.firstName}} {{s.person.lastName}}</md-option>
195           </md-select>
196         </md-input-container>
197         <md-input-container flex="20">
198           <label>Subject</label>
199           <md-select ng-model="reg.subject" required>
200             <md-option ng-repeat="x in tutor.subjects" ng-value="x.id">{{x.name}}</md-option>
201           </md-select>
202         </md-input-container>
203         <md-input-container flex="25">
204           <md-datepicker md-placeholder="date" ng-model="reg.date" required></md-datepicker>
205         </md-input-container>
206         <md-input-container flex="20">
207           <label>Time</label>
208           <md-select ng-model="reg.time" required>
209             <md-option ng-value="x" ng-repeat="x in time">{{x.name}}</md-option>
210           </md-select>
211         </md-input-container>
212         <md-input-container flex="15">
213           <button type="submit" class="md-raised md-primary">
214             Register
215           </button>
216         </md-input-container>
217       </div>
218     </form>
219   </div>
220 </div>
221 </div>
222 </div>
223 </div>
224 </div>
225 </div>

```

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.username.$error" role="alert" multiple="">
            <div ng-message="required" class="my-message">Please enter your username.</div>
          </div>
        </md-input-container>
        <md-input-container class="md-block">
          <label>Password</label>
          <input type="password" name="password" placeholder="password" ng-model="admin.password" required="" />
          <div ng-messages="Form.password.$error" role="alert" multiple="">
            <div ng-message="required" class="my-message">Please enter your password.</div>
          </div>
        </md-input-container>
      </form>
    </md-card-content>
  </md-card>
</div>
<div class="col-md-2"></div>

```

This code is for the functionalities of an Admin

```

<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>
    <div layout="row">
      <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>
    <br>
    <div layout="row" ng-if="function==='tutor'">
      <div ng-show="tutors.length <= 0">No Tutors exist...</div>
      <div ng-repeat="tutor in tutors">
        <md-input-container>
          <md-card>
            <md-card-content>
              <h3>{{tutor.person.firstName}} {{tutor.person.lastName}}</h3>
              <h4>Date of Birth:</h4> {{tutor.person.dateOfBirth}}<br>
              <h4>Address:</h4> {{tutor.person.streetAddress}} {{tutor.person.city}} {{tutor.person.state}} {{tutor.person.country}} {{tutor.person.zipCode}}<br>
              <h4>Years of Experience:</h4> {{tutor.expYears}}<br>
              <div ng-repeat="e in tutor.experience">
                <div>{{e.field}} {{e.name}}</div>
              </div>
            </md-card-content>
          </md-card>
        </md-input-container>
      </div>
    </div>
    <div ng-if="function ==='admin'">
      <div layout="row">
        <form ng-submit="create()">
          <div layout="row">
            <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>
          </div>
        </form>
      </div>
    </div>
  </md-card-content>
</md-card>
</div>

```

Gradle projects need to be imported  
Import Changes Enable Auto-Import

```

<div layout="row">
  <md-input-container>
    <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>
</div>
</div>
<div layout="row">
  <form ng-submit="delete(selectedAdmin)">
    <div layout="row">
      <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>
    </div>
  </form>
</div>
</div>
<div ng-if="function =='subject'">
  <div layout="row">
    <form ng-submit="createSubject()">
      <div layout="row">
        <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 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>
      </div>
    </form>
  </div>

```

Gradle projects need to be imported  
Import Changes Enable Auto-Import

```

<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-button class="md-raised md-primary" type="submit">edit subject</md-button>
      </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

```

1 import './auth.scss';
2 import template from './auth.view.pug';
3
4
5 export default {
6   controller: function ($scope, authenticationService, $state) {
7     "ngInject";
8
9     $scope.login = {
10       user: '',
11       password: ''
12     };
13
14     $scope.signup = {
15       user: '',
16       email: '',
17       confirmEmail: '',
18       pass: '',
19       confirmPass: ''
20     };
21
22     $scope.error = {
23       email: '',
24       password: ''
25     };
26
27     $scope.loginUser = function () {
28       let userData = {
29         username: $scope.login.user,
30         password: $scope.login.password
31       };
32       login(userData);
33     };
34
35     function login(userData) {
36       authenticationService.authenticateUser(userData).then(function (response) {
37         localLogin(response);
38       }, function (err) {
39         $scope.login = {};
40         alert(err.data.message);
41       });
42     }
43
44     function localLogin(response) {
45       authenticationService.setUserToken(response, false);
46       authenticationService.setUserInfo(response.data);
47       if (authenticationService.getUserInfo().isSetup) {
48         $state.go('app.profileSetup.student');
49       } else {
50         $state.go('app.account.profile.list');
51         console.log("go to profileList");
52       }
53     }
54   }
55 }

```

```

47     if ($authenticationService.getUserInfo().isSetup) {
48       $state.go('app.profileSetup.student');
49     } else {
50       $state.go('app.account.profile.list');
51       console.log("go to profileList");
52     }
53   }
54
55   $scope.emailMsg = '';
56   $scope.passMsg = '';
57   $scope.signup = function () {
58     if ($scope.signup.email === $scope.signup.confirmEmail && $scope.signup.pass === $scope.signup.confirmPass) {
59       let userData = {
60         username: $scope.signup.user,
61         email: $scope.signup.email,
62         password: $scope.signup.pass
63       };
64
65       authenticationService.signup(userData).then(function (response) {
66         alert("You signed up successfully");
67         localLogin(response);
68         $scope.signup = {};
69       }, function (response) {
70         if (response.status === 400) {
71           // bad signup
72           alert(response.data.message)
73         } else if (response.status >= 500) {
74           alert("Server error! Try again later.");
75         } else {
76           alert("Signup failed for an unknown reason. Contact an admin.");
77           console.log("error " + response);
78         }
79       })
80     } else {
81       if ($scope.signup.email !== $scope.signup.confirmEmail) {
82         $scope.emailMsg = "email";
83         $scope.signup.confirmEmail = '';
84       }
85       if ($scope.signup.pass !== $scope.signup.confirmPass) {
86         $scope.passMsg = "password";
87         $scope.signup.confirmPass = '';
88       }
89       alert("The confirm fields for the following do not match: " + $scope.emailMsg + " " + $scope.passMsg);
90       $scope.emailMsg = '';
91       $scope.passMsg = '';
92     }
93   };
94   template: template
95 };
96 });
97
98 app;
99
100
101

```

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

```

app.js x
1  // the global CSS
2  import ...
3  // angular
4  import ...
5  // controllers
6  import ...
7  //services
8  import ...
9  //import layout htmls
10 import profileLayout from "./content/shared/profileLayout.html";
11
12 const app = angular.module("Teach4Service", ["ui.router", "ngMaterial", "ngMessages", "puigcerber.countryPicker"]);
13 authService.init(app);
14 app.config([$stateProvider, $urlRouterProvider] => {
15   $urlRouterProvider.otherwise("/login");
16
17   $stateProvider
18     .state("app", {
19       template: "<ui-view></ui-view>",
20       reload: true,
21       abstract: true
22     })
23     .state("app.auth", {
24       url: "/login",
25       reload: true,
26       controller: authComponent.controller,
27       template: authComponent.template,
28       requireLogin: false
29     })
30
31     .state('app.account', {
32       abstract: true,
33       reload: true,
34       template: mainLayoutComp.template,
35       controller: mainLayoutComp.controller,
36       requireLogin: true
37     })
38
39     .state('app.account.tutor.notapprove', {
40       url: "/profile/notapproved",
41       reload: true,
42       template: notApproveComp.template,
43       requireLogin: true
44     })
45
46     .state('app.profileSetup', {
47       url: '/profilesetup',
48       reload: true,
49       controller: profileSetup.controller,
50       template: profileSetup.template,
51       abstract: true,
52       requireLogin: true
53     })
54
55     .state('app.profileSetup.student', {
56       url: '/student',
57       reload: true,
58       template: studentProfileSetup.template,
59       controller: studentProfileSetup.controller,
60     })
61   )
62 }
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94

```

```

97     .state('app.profileSetup.tutor', {
98       url: '/tutor',
99       reload: true,
100      template: tutorProfileSetup.template,
101      controller: tutorProfileSetup.controller,
102      requireLogin: true
103    })
104
105   .state('app.profileSetup.manager', {
106     url: '/manager',
107     reload: true,
108     template: managerProfileSetup.template,
109     controller: managerProfileSetup.controller,
110     requireLogin: true
111   })
112
113   .state('app.account.profile', {
114     abstract: true,
115     template: profileLayout,
116     reload: true,
117     requireLogin: true
118   })
119
120   .state('app.account.profile.list', {
121     url: '/profile',
122     controller: profileComp.controller,
123     template: profileComp.template,
124     reload: true,
125     requireLogin: true,
126   })
127
128
129 //STUDENT ROUTES
130   .state('app.account.student', {
131     url: '/student',
132     template: '<ui-view></ui-view>',
133     reload: true,
134     abstract: true,
135     requireLogin: true
136   })
137   .state('app.account.student.dashboard', {
138     url: '/dashboard',
139     controller: studentDashboardComp.controller,
140     template: studentDashboardComp.template,
141     reload: true,
142     requireLogin: true
143   })
144   .state('app.account.profile.studentEdit', {
145     url: '/profile/student/edit',
146     template: studentEditProfileComp.template,
147     controller: studentEditProfileComp.controller,
148     reload: true,
149     requireLogin: true
150   })
151
152   .state('app.account.profile.studentView', {
153     url: '/profile/student/publicview',
154     template: studentPublicProfile.template,
155     controller: studentPublicProfile.controller,
156     reload: true,
157   })

```

```

158   })
159   .state('app.account.student.findTutor', {
160     url: '/findtutor',
161     template: searchTutorComp.template,
162     controller: searchTutorComp.controller,
163     reload: true,
164     requireLogin: true
165   })
166
167
168   .state('app.account.student.requests', {
169     url: '/requests',
170     template: studentRequests.template,
171     controller: studentRequests.controller,
172     reload: true,
173     requireLogin: true
174   })
175
176
177   .state('app.account.student.sessionHistory', {
178     url: '/sessionhistory',
179     controller: studentSessionCtrl.controller,
180     template: studentSessionCtrl.template,
181     reload: true,
182     requireLogin: true
183   })
184
185
186 //TUTOR ROUTES
187   .state('app.account.tutor', {
188     url: '/tutor',
189     template: '<ui-view></ui-view>',
190     reload: true,
191     abstract: true,
192     requireLogin: true
193   })
194   .state('app.account.tutor.dashboard', {
195     url: '/dashboard',
196     controller: tutorDashboardCtrl.controller,
197     template: tutorDashboardCtrl.template,
198     reload: true,
199     requireLogin: true
200   })
201   .state('app.account.tutor.requests', {
202     url: '/requests',
203     template: tutorRequests.template,
204     controller: tutorRequests.controller,
205     reload: true,
206     requireLogin: true
207   })
208
209
210   .state('app.account.profile.tutorEdit', {
211     url: '/profile/tutor/edit',
212     reload: true,
213     template: tutorEditProfileComp.template,
214     controller: tutorEditProfileComp.controller,
215     requireLogin: true
216   })
217   .state('app.account.profile.tutorView', {
218     url: '/profile/tutor/publicview'
219   })

```

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

```
 222  })
 223  .state('app.account.tutor.sessionhistory', {
 224    url: '/session',
 225    controller: tutorSessionCtrl.controller,
 226    template: tutorSessionCtrl.template,
 227    reload: true,
 228    requireLogin: true
 229  })
 230
 231
 232  //MANAGER ROUTES
 233  .state('app.account.manager', {
 234    url: '/manager',
 235    template: '<ui-view></ui-view>',
 236    reload: true,
 237    abstract: true,
 238    requireLogin: true
 239  })
 240  .state('app.account.manager.dashboard', {
 241    url: '/dashboard',
 242    controller: managerDashboardComp.controller,
 243    template: managerDashboardComp.template,
 244    requireLogin: true
 245  })
 246  .state('app.account.profile.managerEdit', {
 247    url: '/profile/manager/edit',
 248    template: managerEditProfileComp.template,
 249    controller: managerEditProfileComp.controller,
 250    requireLogin: true
 251  })
 252  .state('app.account.profile.managerView', {
 253    url: '/profile/manager/publicview',
 254    template: managerPublicProfile.template,
 255    controller: managerPublicProfile.controller,
 256    requireLogin: true
 257  })
 258  .state('app.account.manager.requests', {
 259    url: '/studentrequests',
 260    template: managerStudentRequestComp.template,
 261    controller: managerStudentRequestComp.controller,
 262    reload: true,
 263    requireLogin: true
 264  })
 265
 266  .state('app.account.manager.session', {
 267    url: '/session',
 268    controller: managerSessionCtrl.controller,
 269    template: managerSessionCtrl.template,
 270    reload: true,
 271    requireLogin: true
 272  })
 273  .state('app.account.manager.findTutor', {
 274    url: '/findtutor',
 275    template: managerFindTutorComp.template,
 276    controller: managerFindTutorComp.controller,
 277    requireLogin: true,
 278    reload: true
 279  })
 280
```

```
281 //Session Routes
282 .state('app.session', {
283   url: "/session/{sessionId:int}",
284   template: videoChatPageComp.template,
285   controller: videoChatPageComp.controller,
286   requireLogin: true,
287   reload: true
288 })
289 .state('app.session.tutor', {
290   url: "/tutor",
291   data: {isTutor: true},
292   requireLogin: true,
293   reload: true
294 })
295 .state('app.session.student', {
296   url: "/student",
297   data: {isTutor: false},
298   requireLogin: true,
299   reload: true
300 })
301
302 //ADMIN ROUTES
303 .state('app.admin', {
304   url: '/admin',
305   template: "ui-view><ui-view>",
306   abstract: true,
307   reload: true,
308   requireLogin: false
309 })
310
311 .state('app.admin.login', {
312   url: '/login',
313   template: adminLogin.template,
314   controller: adminLogin.controller,
315   reload: true,
316   requireLogin: false
317 })
318
319 .state('app.admin.home', {
320   url: '/home',
321   template: adminPage.template,
322   controller: adminPage.controller,
323   reload: true,
324   requireLogin: true
325 })
326
327
328 /*.state('app.account.admin.admin', {
329   url: '/session',
330   controller: adminCtrl.controller,
331   template: adminCtrl.template,
332   reload: true,
333   requireLogin: true
334 })*/
335
336 });
337
```

```

329     url: '/session',
330     controller: adminCtrl.controller,
331     template: adminCtrl.template,
332     reload: true,
333     requireLogin: true
334   })/
335 }
336 });
337
338 app.factory('authHttpInterceptor', ['$q', '$location', function ($q, $location) {
339   return {
340     response: function (response) {
341       if (response.status === 401) {
342         //console.log("Response 401");
343       }
344       return response || $q.when(response);
345     },
346     responseError: function (rejection) {
347       if (rejection.status === 401) {
348         localStorage.removeItem("_userToken");
349         $location.path('/login').search('returnTo', $location.path());
350       }
351       return $q.reject(rejection);
352     }
353   }];
354 }])

```

```

360 app.run(function ($http, $rootScope, $state, authenticationService) {
361   "ngInject"
362   $rootScope.$on('$stateChangeStart', function (event, toState, fromState) {
363     var requireLogin = toState.requireLogin;
364     if (requireLogin) {
365       if (authenticationService.getUserToken() == null) {
366         event.preventDefault();
367         $state.go('app.auth');
368       }
369     } else {
370       if (!authenticationService.getUserToken().systemUser) {
371
372         authenticationService.isApproved().then(function (response) {
373           authenticationService.setApproved(response.data);
374           authenticationService.getSelf().then(function (response) {
375             authenticationService.setUserInfo(response.data);
376             if (!authenticationService.getUserInfo().isApproved && toState.name.includes("tutor")) {
377               event.preventDefault();
378               $state.go("app.account.tutor.notapprove");
379             } else {
380               if (authenticationService.getUserInfo().isApproved && toState.name.includes("notapprove")) {
381                 event.preventDefault();
382                 $state.go("app.account.profile.list");
383               }
384             }
385             if (authenticationService.getUserInfo().isSetup && !toState.name.includes('profileSetup')) {
386               event.preventDefault();
387             }
388           } else {
389             var userType = toState.name;
390             if (!authenticationService.isAuthorized(userType)) {
391               event.preventDefault();
392             }
393           }
394         }, function () {
395           event.preventDefault();
396           authenticationService.logout();
397         })
398       }
399     }
400   }
401 }
402 } else {
403   if (authenticationService.getUserToken() != null) {
404     event.preventDefault();
405     if (authenticationService.getUserToken().systemUser) {
406       $state.go("app.admin.home");
407     } else {
408       $state.go("app.account.profile.list");
409     }
410   }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }

```

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

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

```
authentication.service.js
1  "use strict";
2
3  export default {
4    init: (app) => {
5      app.factory('authenticationService', function authenticationService($http, $rootScope, $window, $state) {
6        'ninject';
7        const userTokenKey = '_userToken';
8        let vm = {
9          isSetup: null,
10         isStudent: null,
11         isTutor: null,
12         isManager: null,
13         isApproved: false
14       };
15
16       return {
17         isAuthorized: isAuthorized,
18         authenticateUser: authenticate,
19         setUserToken: setUserToken,
20         getUserToken: getUserToken,
21         clearAuthInfo: clearAuthInfo,
22         logOut: logOut,
23         getuserInfo: getUserInfo,
24         signUp: signUp,
25         checkUsername: checkUsername,
26         studentProfileSetup: studentProfileSetup,
27         tutorProfileSetup:tutorProfileSetup,
28         managerProfileSetup:managerProfileSetup,
29         getSelf: getSelf,
30         setUserInfo: setUserInfo,
31         getSubjects: getSubjects,
32         getSubscriptions: getSubscriptions,
33         getTutorRequests:getTutorRequests,
34         getStudentRequests:getStudentRequests,
35         respondToRequest:respondToRequest,
36         findByTutor:findByTutor,
37         findBySubject:findBySubject,
38         findByTutorName:findByName,
39         getTutorSubjects:getTutorSubjects,
40         registerSession:register,
41         isApproved: isApproved,
42         setApproved: setApproved,
43         getStudentHistory:getStudentHistory,
44         getTutorHistory: getTutorHistory,
45         getStudentProfile: getStudentProfile,
46         getManagerProfile: getManagerProfile,
47         getTutorProfile: getTutorProfile,
48         getTutorProfile:getAtTutor,
49         getTutorSchedule: getTutorSchedule,
50         updateStudent:updateStudent,
51         updateManager:updateManager,
52         updateTutor:updateTutor,
53         getManagerDashboard:getManagerDashboard,
54         getTutorDashboard:getTutorDashboard,
55         getStudentDashboard:getStudentDashboard,
56         getManagerSessionHistory:getManagerSessionHistory,
57         getManagerRequest: getManagerRequest,
58         setUpStudent: setUpStudent,
59         getProfile: getProfile,
60         setAvailability: setAvailability,
61       };
62     };
63   });
64 }
```

```

125     console.log(err.data);
126   });
127 }
128
129 function loginAdmin(data) {
130   return $http.post("./api/admin/auth/login", data);
131 }
132
133 function setAvailability(data) {
134   return http('POST', './api/dashboard/isavailable', data);
135 }
136
137 function getProfile() {
138   return http("GET", "./api/profiles/user");
139 }
140
141 function setUpStudent(data) {
142   return http("POST", "./api/dashboard/addstudent", data);
143 }
144
145 function getManagerRequest(id) {
146   return http("GET", "./api/sessions/students/requests/" + id);
147 }
148
149 function getManagerSessionHistory(id) {
150   return http("GET", "./api/sessions/students/" + id);
151 }
152
153 function joinStudentSession(sessionId) {
154   return http("GET", "./api/opentok/$" + sessionId + "/student");
155 }
156
157 function joinTutorSession(sessionId) {
158   return http("GET", "./api/opentok/$" + sessionId + "/tutor");
159 }
160
161 function endSession(sessionId) {
162   return http("DELETE", "./api/opentok/" + sessionId);
163 }
164
165 function getManagerDashboard() {
166   return http("GET", "./api/dashboard/Parent");
167 }
168
169 function getStudentDashboard() {
170   return http("GET", "./api/dashboard/student");
171 }
172
173 function getTutorDashboard() {
174   return http("GET", "./api/dashboard/Tutor");
175 }
176
177 function updateManager(data) {
178   return http("PUT", "./api/profiles/parent", data);
179 }
180
181 function updateTutor(data) {
182   return http("PUT", "./api/profiles/tutor", data);
183 }
184
185 function updateStudent(data) {
186 }

```

```

282   }
283
284   function getTutorSchedule(username) {
285     return http("GET", "./api/profiles/schedule/" + username);
286   }
287
288   function getTutorHistory() {
289     return http("GET", "./api/sessions/self/tutorhistory");
290   }
291
292   function getStudentHistory() {
293     return http("GET", "./api/sessions/self/studenthistory");
294   }
295
296   function setApproved(data) {
297     vm.isApproved = data.approved;
298   }
299   function isApproved() {
300     return http("GET", "./api/profiles/isapproved");
301   }
302
303   function findByName(data) {
304     return http("POST", "./api/sessions/findtutor/bytutornname", data);
305   }
306
307   function register(data) {
308     return http("POST", "./api/sessions/register", data);
309   }
310
311   function getTutorSubjects(data) {
312     return http("GET", "./api/sessions/tutor/subjects/" + data);
313   }
314
315   function findBySubject(data) {
316     return http("POST", "./api/sessions/findtutor/bysubject", data);
317   }
318
319   function findbyTutor(data) {
320     return http("POST", "./api/sessions/findtutor/bytutor", data);
321   }
322
323   function respondToRequest(data) {
324     return http("POST", "./api/sessions/requests/respond", data);
325   }
326
327   function getTutorRequests() {
328     return http("GET", "./api/sessions/tutors/requests/self");
329   }
330
331   function getStudentRequests() {
332     return http("GET", "./api/sessions/students/requests/self");
333   }
334
335   function getSubjects(field) {
336     return http("GET", "./api/profiles/subjects/" + field);
337   }
338
339   function getAllSubjects() {
340     return http("GET", "./api/profiles/subjects");
341   }

```

```

302     function logout() {
303       http('GET','./api/users/logout').then(function () {
304         clearAuthInfo();
305         $state.go('app.auth');
306       }, function (err) {
307         console.log(err.data);
308       });
309     }
310
311     function authenticate(userData) {
312       return $http.post('./api/users/login', userData);
313     }
314
315     function getSelf() {
316       return http('GET','./api/users/self');
317     }
318
319     function getUserInfo() {
320       return vm;
321     }
322
323     function isAuthorized(userType) {
324       if (userType.includes('profileSetup')) {
325         return vm.isSetup;
326       }
327       else if (userType.includes('student')) {
328         return vm.isStudent;
329       }
330       else if (userType.includes('tutor')) {
331         return vm.isTutor;
332       }
333       else if (userType.includes('manager')) {
334         return vm.isManager;
335       }
336       else if (userType.includes('profile')) {
337         return !vm.isSetup;
338       }
339       else {
340         return false;
341       }
342     }
343
344     function setUserToken(token,bool) {
345       clearAuthInfo();
346       var userData = {
347         systemUser: bool,
348         CSRF_TOKEN: token.headers('x-csrf')
349       };
350
351       localStorage.setItem(userTokenKey, angular.toJson(userData));
352       //sessionStorage.setItem(userTokenKey, angular.toJson(token));
353     }
354
355     function clearAuthInfo() {
356       clearTokenInfo();
357     }
358
359     function clearTokenInfo() {
360       clearAuthInfo();
361     }
362
363     function clearAuthTokenInfo() {
364       clearAuthInfo();
365     }
366
367     function clearTokenInfo() {
368       localStorage.removeItem(userTokenKey);
369       //sessionStorage.removeItem(userTokenKey);
370     }
371
372     function getUserToken() {
373       let userToken = localStorage.getItem(userTokenKey);
374       if (userToken) {
375         return angular.fromJson(localStorage.getItem(userTokenKey));
376       } else {
377         return null;
378       }
379     }
380
381     function signUp(userData) {
382       return $http.post('./api/users/signup', userData);
383     }
384
385     function http(method, url, data){
386       let req = {
387         method: method,
388         url: url,
389         headers: {
390           'x-csrf': getUserToken().CSRF_TOKEN
391         }
392       };
393
394       if (method !== 'GET' && data) {
395         req.data = data;
396       }
397
398       return $http(req);
399     }
400   }
401 }

```

```

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 }

```

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

```

1 import template from "./auth.html";
2
3 export default {
4   controller: function ($scope, authenticationService, $state) {
5     "ngInject";
6     console.log("hey");
7     $scope.loginAdmin = function () {
8       console.log("here");
9       var data = {
10         username: $scope.admin.username,
11         password: $scope.admin.password
12       };
13       authenticationService.loginAdmin(data).then(function (response) {
14         authenticationService.setUserToken(response, true);
15         alert("logged in successfully");
16         $state.go("app.admin.home");
17         $scope.admin = {};
18       },function () {
19         alert("Failed to login");
20       })
21     },
22     template: template
23   }
24 }
```

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

```

1 import template from "./adminpage.html"
2
3 export default {
4   controller: function ($scope, authenticationService, $window) {
5     "ngInject";
6
7     // initialize subs
8     $scope.sub = {};
9     $scope.editSub = {};
10
11     $scope.logout = function () {
12       authenticationService.logoutAdmin();
13     };
14
15     authenticationService.getAdminTutors().then(function (response) {
16       $scope.tutors = response.data;
17     });
18
19     $scope.approve = function (id) {
20       authenticationService.adminApprove(id).then(function () {
21         $window.location.reload();
22         $scope.function = "tutor";
23       }, function () {
24         alert("Failed to approve");
25       })
26     };
27
28     $scope.deny = function (id) {
29       authenticationService.adminDeny(id).then(function () {
30         $window.location.reload();
31         $scope.function = "tutor";
32       }, function () {
33         alert("Failed to deny");
34       })
35     };
36     authenticationService.getAdmins().then(function (response) {
37       $scope.admins = response.data;
38     });
39
40     $scope.delete = function (id) {
41       authenticationService.deleteAdmin(id).then(function (response) {
42         alert("successfully deleted admin");
43         $window.location.reload();
44         $scope.function = "admin";
45       }, function (err) {
46         alert("Failed to delete admin");
47       })
48     };
49
50     $scope.create = function () {
51       authenticationService.createAdmin($scope.admin).then(function (response) {
52         alert("successfully created admin");
53         $window.location.reload();
54         $scope.function = "admin";
55       }, function () {
56         alert("Failed to create admin");
57       })
58     };
59   }
60 }
```

```
58
59
60
61     $scope.createSubject = function () {
62         authenticationService.createSubject($scope.sub).then(function (response) {
63             alert("successfully created subject");
64             $window.location.reload();
65             $scope.sub = {};
66             $scope.function = "subject";
67         }, function () {
68             alert("failed to create subject");
69         })
70     }
71
72     authenticationService.getAdminSubjects().then(function (response) {
73         $scope.subjects = response.data;
74     })
75
76     $scope.toEditSubjectSelected = function(selected)
77     {
78         $scope.editSub.field = selected.field;
79         $scope.editSub.name = selected.name;
80         $scope.editSub.description = selected.description;
81     };
82
83     $scope.editSubject = function (id) {
84         authenticationService.editSubject($scope.editSub, id).then(function () {
85             alert("successfully edited subject");
86             $window.location.reload();
87             $scope.editSub = {};
88             $scope.selectedSub = "";
89             $scope.function = "subject";
90         }, function () {
91             alert("failed to create subject");
92         })
93     }
94     $scope.deleteSub = function ($s) {
95         var data = {
96             "field": $s.field,
97             "name": $s.name,
98             "description": $s.description
99         }
100        authenticationService.deleteSubject($s.id, data).then(function () {
101            alert("successfully deleted subject");
102            $window.location.reload();
103            $scope.selectedSub = "";
104            $scope.function = "subject";
105        }, function () {
106            alert("failed to delete subject");
107        })
108    }
109
110    template: template
111
112 }
```

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

```
1 import template from './manager.dashboard.html';
2
3 export default {
4   controller: function ($scope, authenticationService, $window, $state) {
5     "ngInject";
6
7     $scope.student = {
8       firstName: '',
9       lastName: '',
10      dateOfBirth: new Date(),
11      streetAddress: '',
12      city: '',
13      state: '',
14      postalCode: ''
15    };
16
17    $scope.states = ['Alabama', 'Alaska', 'American Samoa', 'Arizona', 'Arkansas', 'California', 'Colorado', 'Connecticut', 'Delaware', 'District
18
19    $scope.minDate = new Date(
20      $scope.student.dateOfBirth.getFullYear() - 100,
21      $scope.student.dateOfBirth.getMonth(),
22      $scope.student.dateOfBirth.getDate()
23    );
24
25    $scope.maxDate = new Date(
26      $scope.student.dateOfBirth.getFullYear(),
27      $scope.student.dateOfBirth.getMonth(),
28      $scope.student.dateOfBirth.getDate()
29    );
30
31    $scope.show = true;
32    $scope.addForm = function () {
33      $scope.show = false;
34    }
35    $scope.cancel = function () {
36      $scope.show = true;
37      $scope.student = {};
38    }
39
40    authenticationService.getManagerDashboard().then(function (response) {
41      $scope.manager = response.data;
42    })
43
44    $scope.completeProfile = function () {
45      authenticationService.setUpStudent($scope.student).then(function (response) {
46        $scope.show = !$scope.show;
47        alert("Successfully added student");
48        $window.location.reload();
49        $scope.student = {};
50      }, function (err) {
51        alert("Failed to add student");
52      })
53    }
54  }
55}
```

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

```
1 import template from './student.dashboard.html';
2
3 export default {
4     controller: function ($scope, authenticationService, $state) {
5         "ngInject";
6
7         $scope.student = {};
8
9         reloadData();
10
11        function reloadData() {
12            authenticationService.getStudentDashboard().then(function (response) {
13                $scope.student = response.data;
14            });
15        }
16    }
}
```

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

```
1 import template from './tutor.dashboard.html';
2
3 export default {
4     controller: function ($scope, authenticationService, $state) {
5         "ngInject";
6
7         $scope.tutor = {};
8
9         reloadData();
10
11        function reloadData() {
12            authenticationService.getTutorDashboard().then(function (response) {
13                $scope.tutor = response.data;
14            });
15        }
16    }
}
```

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

```
1 import template from './manager.requests.html';
2
3 export default {
4     controller: function ($scope, authenticationService) {
5         "ngInject";
6
7         $scope.parent={
8
9             }
10            authenticationService.getManagerDashboard().then(function (response) {
11                $scope.students = response.data.managedStudents;
12            })
13            $scope.getRequest = function (id) {
14                authenticationService.getManagerRequest(id).then(function (response) {
15                    $scope.errorMsg = '';
16                    angular.forEach(response.data, function (obj) {
17                        obj.requestDate.toLocaleString();
18                    })
19                    $scope.studentRequests = response.data;
20                }, function () {
21                    $scope.errorMsg = "Unable to retrieve your requests at this moment";
22                })
23            }
24
25            },
26            template: template
27        }
}
```

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

```

1 import template from "./student.requests.html";
2
3 export default {
4     controller: function ($scope, authenticationService) {
5         "ngInject";
6         authenticationService.getStudentRequests().then(function (response) {
7             $scope.errorMsg = '';
8             console.log(response.data);
9             angular.forEach(response.data, function (obj) {
10                 obj.requestDate.toLocaleString();
11             });
12             console.log(response.data);
13             $scope.studentSessionRequests = response.data;
14         }, function () {
15             $scope.errorMsg = "Unable to retrieve your requests at this moment";
16         });
17     },
18     template: template
19 }
20

```

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

```

1 import template from "./tutor.requests.html";
2
3 export default {
4     controller: function ($scope, authenticationService) {
5         "ngInject";
6
7         $scope.tutorSessionRequests = {
8             pending: [],
9             req: []
10 };
11
12         function loadData() {
13             authenticationService.getTutorRequests().then(function (response) {
14                 $scope.errorMsg = '';
15
16                 // clear them first.
17                 $scope.tutorSessionRequests.pending = [];
18                 $scope.tutorSessionRequests.req = [];
19
20                 for (let sr of response.data) {
21                     if (sr.status === "PENDING") {
22                         $scope.tutorSessionRequests.pending.push(sr);
23                     } else {
24                         $scope.tutorSessionRequests.req.push(sr);
25                     }
26                 }
27                 console.log($scope.tutorSessionRequests)
28             }, function () {
29                 $scope.errorMsg = "Unable to retrieve your requests at this moment";
30             });
31         }
32
33         loadData();
34
35         $scope.decline = function (id) {
36             let data = {
37                 requestId: id,
38                 status: "DECLINED"
39             };
40             authenticationService.respondToRequest(data).then(function (response) {
41                 $scope.tutorSessionRequests = response.data;
42                 alert("successfully declined request!");
43                 loadData(); // again.
44             }, function () {
45                 $scope.errorMsg = "Failed to decline request";
46             })
47         };
48
49         $scope.approve = function (id) {
50             let data = {
51                 requestId: id,
52                 status: "APPROVED"
53             };
54             authenticationService.respondToRequest(data).then(function (response) {
55                 $scope.tutorSessionRequests = response.data;
56                 alert("successfully approved request!");
57                 loadData(); // again.
58             }, function () {
59                 $scope.errorMsg = "Failed to approve request";
60             })
61         };
62     }
63 }
64

```

## 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.getCounty());
        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.getCounty());
        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");
}
```

```

@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.getCounty());
        ArrayList<Long> userExpertise = setup.getUserExpertise();
        Tutor tutor = tdao.create(person, setup.getExpYears());
        for(Long anUserExpertise : userExpertise)
        {
            Subject subject = subdao.get(anUserExpertise);
            if(!tedao.exists(tutor, subject))
            {
                tdao.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.getCounty());
        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(); }

```

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.getStartTime().toString());
            sessions.put("endDate", s.getEndTime().toString());
            sessions.put("description", s.getDescription());
            studentSessions.add(sessions);
        }
        return Response.ok(studentSessions).build();
    }
    throw new ForbiddenException("You are not authorized 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.getStartTime().toString());
            sessions.put("endDate", s.getEndTime().toString());
            sessions.put("description", s.getDescription());
            tutorSessions.add(sessions);
        }
        return Response.ok(tutorSessions).build();
    }
    throw new ForbiddenException("You are not authorized to access this data");
}

```

```

@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 = stdao.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.getStartTime().toString());
            sessions.put("endDate", s.getEndTime().toString());
            sessions.put("description", s.getDescription());
            studentSessions.add(sessions);
        }
        return Response.ok(studentSessions).build();
    }
    throw new ForbiddenException("You are not authorized 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 authorized to access this data");
}

```

```

@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");
}

```

```

@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

```

@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");
}

```

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.getRequestId().getRequestedTo() == auth.getTutorId())
    {
        SessionRequest sr = srdao.get(response.getRequestId());
        sr.setStatus(response.getStatus());
        if(response.getStatus().equals(SessionRequestStatus.APPROVED)){
            Sessions session = srdao.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/bytutornname")
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");
}

```

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 authorized 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

```
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

```
package edu.utdallas.utdesign.teach4service.resources.requests;  
  
import ...  
  
@Data  
public class SubjectRequest  
{  
    @NotBlank  
    private String name;  
  
    @NotBlank  
    private String field;  
  
    @NotBlank  
    private String description;  
}
```

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;  
}
```

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

```
package edu.utdallas.utdesign.teach4service.resources.requests;  
  
import ...  
  
@Data  
public class FindByTutorRequest  
{  
    @NotNull  
    private String username;  
}
```

```
package edu.utdallas.utdesign.teach4service.resources.requests;  
  
import lombok.Data;  
  
@Data  
public class FindByTutorNameRequest  
{  
    private String firstName;  
    private String lastName;  
}
```

```
package edu.utdallas.utdesign.teach4service.resources.requests;  
  
import lombok.Data;  
  
@Data  
public class FindBySubject  
{  
    private long subjectId;  
}
```

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 postalCode)
    {
        return currentSession().createNativeQuery( queryString: "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.setStartTime(startTime);
    return persist(session);
}
```

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();
}
```

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

```
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

```
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

```
package edu.utdallas.utdesign.teach4service.db.entities;
public enum SubjectField
{
    MATH,
    SCIENCE,
    LANGUAGE,
    ART,
    SOCIAL,
    OTHER
}
```

Entity Class for Schedules table in the database

```
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;
import org.hibernate.annotations.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

```
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

```
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

```
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();
}
```