SWENG 500: Software Engineering Studio
Test Plan and Report

Group 5

Jeffrey Chiampi

Brandon Hessler

Kyle Hughes

Ryan McDonald

Overview

This document contains the results of white box testing the Science Olympiad application. The application was designed in two modules. The first module, the server API, is a fully functional API, which interacts with the system's database. The second module is responsible for the user interface (UI) and utilizes the API. It is primarily responsible for the presentation and input of data. Since these modules were developed with separate programming languages, their testing strategies and methods varied.

In total, 252 tests were developed. The continuous integration and continuous delivery development strategy we employed was facilitated by a Jenkins server. Jenkins is an open source automation server written in Java. It ingested our source code from GitHub, built our applications, ran the tests, and provided reports. If the tests passed, Jenkins deployed our app to production. The nature of this process meant that all tests had to pass in order for the application to be deployed.

Testing the Server API

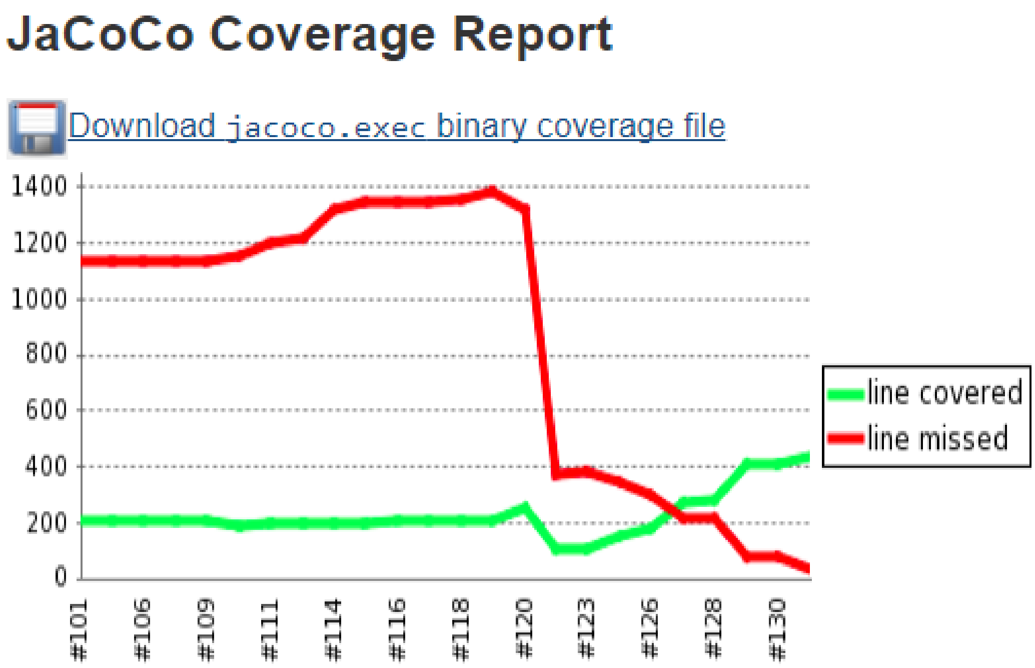
In order to reach 93% instruction coverage, 95% method coverage, and 100% class coverage 54 API tests were developed. Instruction coverage refers to the smallest unit JaCoCo counts. This is equivalent to a single Java byte code instruction. A complete coverage summary can be view in the following section. While this coverage percentage is not 100% it does represent testing every critical component.

Coverage Summary

The following table shows coverage percentages from the Server API's controllers.

Instruction	Branch	Complexity	Line	Method	Class
93%	89%	87%	92%	95%	100%

The following chart shows the coverage growth. Note that the drop off was created when we figure out how to successfully exclude library files from the testing report. These files were skewing our coverage report and making it seem like our tests were not effective.

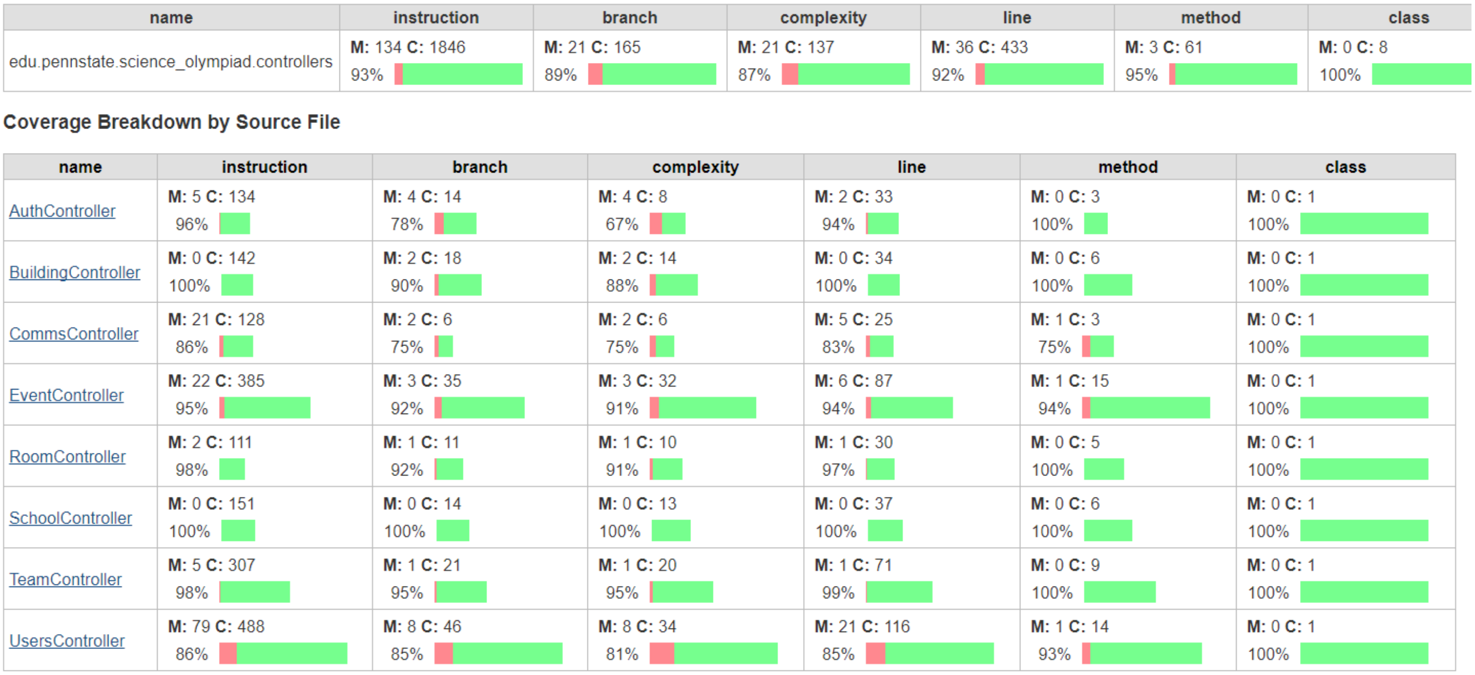


JaCoCo Coverage Report

Download jacoco.exec binary coverage file

Coverage breakdown by source code file

The following screen show shows a more detailed breakdown by source file.



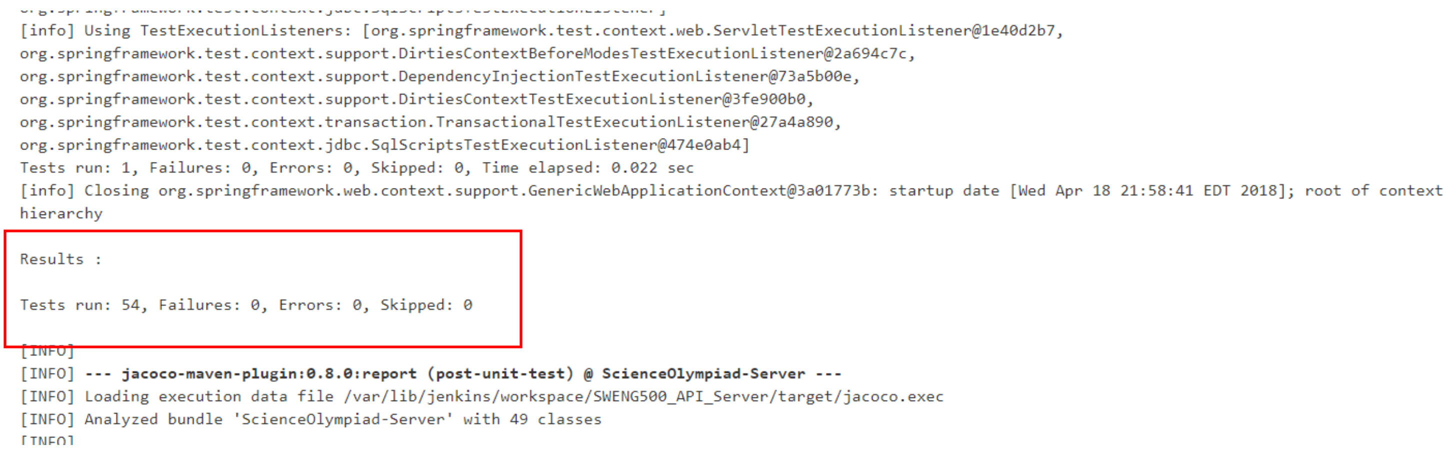
name	instruction	branch	complexity	line	method	class
edu.pennstate.science_olympiad.controllers	**M:** 134 **C:** 1846 93%	**M:** 21 **C:** 165 89%	**M:** 21 **C:** 137 87%	**M:** 36 **C:** 433 92%	**M:** 3 **C:** 61 95%	**M:** 0 **C:** 8 100%

Coverage Breakdown by Source File

name	instruction	branch	complexity	line	method	class
AuthController	**M:** 5 **C:** 134 96%	**M:** 4 **C:** 14 78%	**M:** 4 **C:** 8 67%	**M:** 2 **C:** 33 94%	**M:** 0 **C:** 3 100%	**M:** 0 **C:** 1 100%
BuildingController	**M:** 0 **C:** 142 100%	**M:** 2 **C:** 18 90%	**M:** 2 **C:** 14 88%	**M:** 0 **C:** 34 100%	**M:** 0 **C:** 6 100%	**M:** 0 **C:** 1 100%
CommsController	**M:** 21 **C:** 128 86%	**M:** 2 **C:** 6 75%	**M:** 2 **C:** 6 75%	**M:** 5 **C:** 25 83%	**M:** 1 **C:** 3 75%	**M:** 0 **C:** 1 100%
EventController	**M:** 22 **C:** 385 95%	**M:** 3 **C:** 35 92%	**M:** 3 **C:** 32 91%	**M:** 6 **C:** 87 94%	**M:** 1 **C:** 15 94%	**M:** 0 **C:** 1 100%
RoomController	**M:** 2 **C:** 111 98%	**M:** 1 **C:** 11 92%	**M:** 1 **C:** 10 91%	**M:** 1 **C:** 30 97%	**M:** 0 **C:** 5 100%	**M:** 0 **C:** 1 100%
SchoolController	**M:** 0 **C:** 151 100%	**M:** 0 **C:** 14 100%	**M:** 0 **C:** 13 100%	**M:** 0 **C:** 37 100%	**M:** 0 **C:** 6 100%	**M:** 0 **C:** 1 100%
TeamController	**M:** 5 **C:** 307 98%	**M:** 1 **C:** 21 95%	**M:** 1 **C:** 20 95%	**M:** 1 **C:** 71 99%	**M:** 0 **C:** 9 100%	**M:** 0 **C:** 1 100%
UsersController	**M:** 79 **C:** 488 86%	**M:** 8 **C:** 46 85%	**M:** 8 **C:** 34 81%	**M:** 21 **C:** 116 85%	**M:** 1 **C:** 14 93%	**M:** 0 **C:** 1 100%

Test Results

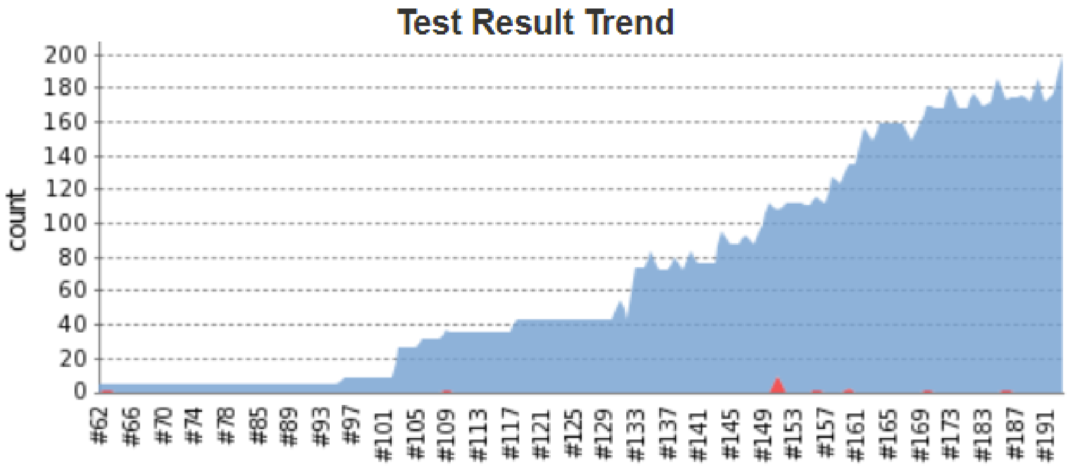
The following outline is an expert from the Jenkins server's console. It shows that all 54 server API tests successfully passed.



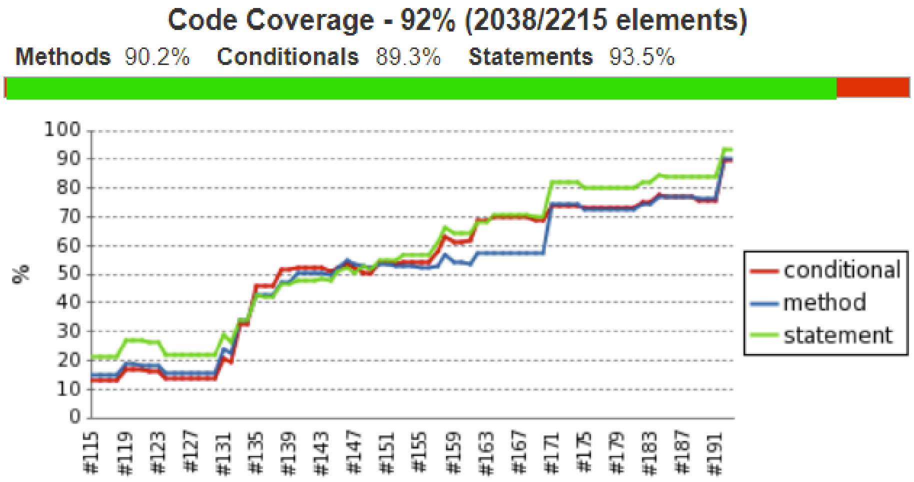
```
org.springframework.test.context.jdbc.SqlScriptsTestExecutionListener]
[info] Using TestExecutionListeners: [org.springframework.test.context.web.ServletTestExecutionListener@1e40d2b7,
org.springframework.test.context.support.DirtiesContextBeforeModesTestExecutionListener@2a694c7c,
org.springframework.test.context.support.DependencyInjectionTestExecutionListener@73a5b00e,
org.springframework.test.context.support.DirtiesContextTestExecutionListener@3fe900b0,
org.springframework.test.context.transaction.TransactionalTestExecutionListener@27a4a890,
org.springframework.test.context.jdbc.SqlScriptsTestExecutionListener@474e0ab4]
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.022 sec
[info] Closing org.springframework.web.context.support.GenericWebApplicationContext@3a01773b: startup date [Wed Apr 18 21:58:41 EDT 2018]; root of context
hierarchy

Results :

Tests run: 54, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- jacoco-maven-plugin:0.8.0:report (post-unit-test) @ ScienceOlympiad-Server ---
[INFO] Loading execution data file /var/lib/jenkins/workspace/SWENG500_API_Server/target/jacoco.exec
[INFO] Analyzed bundle 'ScienceOlympiad-Server' with 49 classes
[INFO]
```

Testing the UI

In order to reach 93.56% coverage 198 UI tests were developed. The following graphs show how our test coverage evolved as the project was developed. As you can see we began testing early in development and continued a steady growth as features were added.

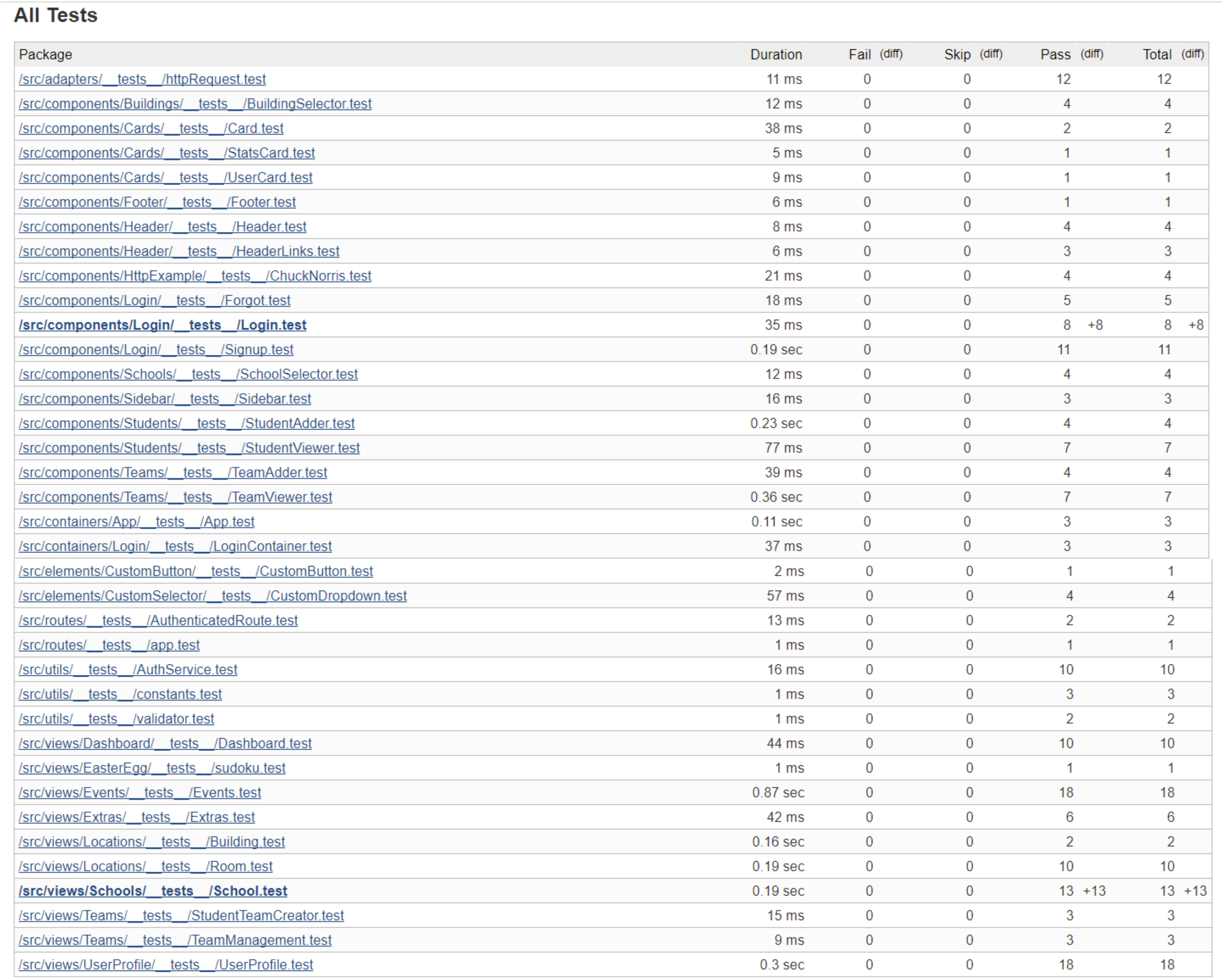


The next graph shows the type of coverage we were able to achieve. While the 93.56% coverage is less than we had hoped to achieve we feel that it represents a substantial accomplishment, which covers the primary system functionality. The asynchronous nature of JavaScript meant that some components were difficult to test due to their state changes.



Test Results

The following report shows that app 198 tests successfully passed when the project was built.



All Tests

Package	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
/src/adapters/__tests__/httpRequest.test	11 ms	0		0		12		12	
/src/components/Buildings/__tests__/BuildingSelector.test	12 ms	0		0		4		4	
/src/components/Cards/__tests__/Card.test	38 ms	0		0		2		2	
/src/components/Cards/__tests__/StatsCard.test	5 ms	0		0		1		1	
/src/components/Cards/__tests__/UserCard.test	9 ms	0		0		1		1	
/src/components/Footer/__tests__/Footer.test	6 ms	0		0		1		1	
/src/components/Header/__tests__/Header.test	8 ms	0		0		4		4	
/src/components/Header/__tests__/HeaderLinks.test	6 ms	0		0		3		3	
/src/components/HttpExample/__tests__/ChuckNorris.test	21 ms	0		0		4		4	
/src/components/Login/__tests__/Forgot.test	18 ms	0		0		5		5	
/src/components/Login/__tests__/Login.test	35 ms	0		0		8	+8	8	+8
/src/components/Login/__tests__/Signup.test	0.19 sec	0		0		11		11	
/src/components/Schools/__tests__/SchoolSelector.test	12 ms	0		0		4		4	
/src/components/Sidebar/__tests__/Sidebar.test	16 ms	0		0		3		3	
/src/components/Students/__tests__/StudentAdder.test	0.23 sec	0		0		4		4	
/src/components/Students/__tests__/StudentViewer.test	77 ms	0		0		7		7	
/src/components/Teams/__tests__/TeamAdder.test	39 ms	0		0		4		4	
/src/components/Teams/__tests__/TeamViewer.test	0.36 sec	0		0		7		7	
/src/containers/App/__tests__/App.test	0.11 sec	0		0		3		3	
/src/containers/Login/__tests__/LoginContainer.test	37 ms	0		0		3		3	
/src/elements/CustomButton/__tests__/CustomButton.test	2 ms	0		0		1		1	
/src/elements/CustomSelector/__tests__/CustomDropdown.test	57 ms	0		0		4		4	
/src/routes/__tests__/AuthenticatedRoute.test	13 ms	0		0		2		2	
/src/routes/__tests__/app.test	1 ms	0		0		1		1	
/src/utils/__tests__/AuthService.test	16 ms	0		0		10		10	
/src/utils/__tests__/constants.test	1 ms	0		0		3		3	
/src/utils/__tests__/validator.test	1 ms	0		0		2		2	
/src/views/Dashboard/__tests__/Dashboard.test	44 ms	0		0		10		10	
/src/views/EasterEgg/__tests__/sudoku.test	1 ms	0		0		1		1	
/src/views/Events/__tests__/Events.test	0.87 sec	0		0		18		18	
/src/views/Extras/__tests__/Extras.test	42 ms	0		0		6		6	
/src/views/Locations/__tests__/Building.test	0.16 sec	0		0		2		2	
/src/views/Locations/__tests__/Room.test	0.19 sec	0		0		10		10	
/src/views/Schools/__tests__/School.test	0.19 sec	0		0		13	+13	13	+13
/src/views/Teams/__tests__/StudentTeamCreator.test	15 ms	0		0		3		3	
/src/views/Teams/__tests__/TeamManagement.test	9 ms	0		0		3		3	
/src/views/UserProfile/__tests__/UserProfile.test	0.3 sec	0		0		18		18	

UI Test

The next section lists each UI test. You can click on any test for more information.

All files

83.29% Statements `1266/1520` **75.4%** Branches `426/565` **76.32%** Functions `390/511` **83.58%** Lines `1257/1504`

File ▲		Statements		Branches		Functions		Lines	
src		100%	1/1	100%	0/0	100%	0/0	100%	1/1
src/adapters		100%	45/45	100%	27/27	100%	13/13	100%	44/44
src/components/Buildings		100%	21/21	66.67%	8/12	84.62%	11/13	100%	20/20
src/components/Cards		100%	3/3	100%	14/14	100%	3/3	100%	3/3
src/components/Footer		100%	1/1	100%	0/0	100%	1/1	100%	1/1
src/components/Header		93.55%	29/31	87.5%	7/8	100%	12/12	93.55%	29/31
src/components/Login		95.59%	130/136	87.5%	49/56	89.74%	35/39	96.24%	128/133
src/components/Schools		100%	16/16	50%	4/8	90%	9/10	100%	15/15
src/components/Sidebar		100%	11/11	83.33%	10/12	100%	6/6	100%	11/11
src/components/Students		100%	76/76	100%	19/19	100%	37/37	100%	76/76
src/components/Teams		100%	105/105	100%	15/15	100%	46/46	100%	105/105
src/containers/App		96.77%	30/31	95%	19/20	100%	7/7	96.77%	30/31
src/containers/Login		100%	29/29	100%	13/13	100%	10/10	100%	29/29
src/elements/CustomButton		100%	4/4	100%	0/0	100%	1/1	100%	4/4
src/elements/CustomSelector		100%	26/26	78.57%	11/14	100%	12/12	100%	24/24
src/routes		100%	4/4	100%	4/4	100%	2/2	100%	4/4
src/utils		100%	104/104	100%	50/50	100%	28/28	100%	104/104
src/views/Dashboard		73.17%	60/82	37.5%	6/16	71.05%	27/38	78.95%	60/76
src/views/EasterEgg		100%	1/1	100%	0/0	100%	1/1	100%	1/1
src/views/Events		65.56%	238/363	63.79%	74/116	36.26%	33/91	65.56%	238/363
src/views/Extras		86.05%	37/43	57.14%	8/14	93.75%	15/16	86.05%	37/43
src/views/Locations		56.11%	101/180	37.5%	27/72	42.11%	24/57	56.11%	101/180
src/views/Schools		85.87%	79/92	61.76%	21/34	60.71%	17/28	85.87%	79/92
src/views/Teams		100%	19/19	100%	8/8	100%	12/12	100%	19/19
src/views/UserProfile		100%	96/96	96.97%	32/33	100%	28/28	100%	94/94

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src

100% Statements `1/1` **100%** Branches `0/0` **100%** Functions `0/0` **100%** Lines `1/1`

File ▲		Statements		Branches		Functions		Lines	
setupTests.js	🟩	100%	1/1	100%	0/0	100%	0/0	100%	1/1

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/adapters

100% Statements 45/45 **100%** Branches 27/27 **100%** Functions 13/13 **100%** Lines 44/44

File ▲		Statements		Branches		Functions		Lines	
httpRequest.js	🟩	100%	45/45	100%	27/27	100%	13/13	100%	44/44

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/components/Buildings

100% Statements 21/21 **66.67%** Branches 8/12 **84.62%** Functions 11/13 **100%** Lines 20/20

File		Statements		Branches		Functions		Lines	
BuildingSelector.js	🟩	100%	21/21	66.67%	8/12	84.62%	11/13	100%	20/20

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/components/Cards

100% Statements 3/3 **100%** Branches 14/14 **100%** Functions 3/3 **100%** Lines 3/3

File		Statements		Branches		Functions		Lines	
Card.js	🟩	100%	1/1	100%	14/14	100%	1/1	100%	1/1
StatsCard.js	🟩	100%	1/1	100%	0/0	100%	1/1	100%	1/1
UserCard.js	🟩	100%	1/1	100%	0/0	100%	1/1	100%	1/1

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/components/Footer

100% Statements `1/1` **100%** Branches `0/0` **100%** Functions `1/1` **100%** Lines `1/1`

File ▲		Statements		Branches		Functions		Lines	
Footer.js	🟩	100%	1/1	100%	0/0	100%	1/1	100%	1/1

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/components/Header

93.55% Statements `29/31` **87.5%** Branches `7/8` **100%** Functions `12/12` **93.55%** Lines `29/31`

File ▲		Statements		Branches		Functions		Lines	
Header.js		91.3%	21/23	87.5%	7/8	100%	6/6	91.3%	21/23
HeaderLinks.js		100%	8/8	100%	0/0	100%	6/6	100%	8/8

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/components/Login

95.59% Statements `130/136` **87.5%** Branches `49/56` **89.74%** Functions `35/39` **96.24%** Lines `128/133`

File ▲		Statements		Branches		Functions		Lines	
Forgot.js		100%	17/17	100%	2/2	100%	9/9	100%	17/17
Login.js		100%	24/24	100%	12/12	100%	10/10	100%	23/23
Signup.js		93.68%	89/95	83.33%	35/42	80%	16/20	94.62%	88/93

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/components/Schools

100% Statements `16/16` **50%** Branches `4/8` **90%** Functions `9/10` **100%** Lines `15/15`

File ▲		Statements		Branches		Functions		Lines	
SchoolSelector.js	🟩	100%	16/16	50%	4/8	90%	9/10	100%	15/15

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/components/Sidebar

100% Statements `11/11` **83.33%** Branches `10/12` **100%** Functions `6/6` **100%** Lines `11/11`

File		Statements		Branches		Functions		Lines	
Sidebar.js	🟩	100%	11/11	83.33%	10/12	100%	6/6	100%	11/11

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/components/Students

100% Statements 76/76 **100%** Branches 19/19 **100%** Functions 37/37 **100%** Lines 76/76

File		Statements		Branches		Functions		Lines	
StudentAdder.js	🟩	100%	33/33	100%	10/10	100%	13/13	100%	33/33
StudentViewer.js	🟩	100%	43/43	100%	9/9	100%	24/24	100%	43/43

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/components/Teams

100% Statements `105/105` **100%** Branches `15/15` **100%** Functions `46/46` **100%** Lines `105/105`

File ▲		Statements		Branches		Functions		Lines	
TeamAdder.js	🟩	100%	31/31	100%	8/8	100%	12/12	100%	31/31
TeamViewer.js	🟩	100%	74/74	100%	7/7	100%	34/34	100%	74/74

All files src/containers/App

96.77% Statements `30/31` **95%** Branches `19/20` **100%** Functions `7/7` **96.77%** Lines `30/31`

File ▲		Statements		Branches		Functions		Lines	
App.js		96.77%	30/31	95%	19/20	100%	7/7	96.77%	30/31

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/containers/Login

100% Statements 29/29 **100%** Branches 13/13 **100%** Functions 10/10 **100%** Lines 29/29

File		Statements		Branches		Functions		Lines	
LoginContainer.js	🟩	100%	29/29	100%	13/13	100%	10/10	100%	29/29

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/elements/CustomButton

100% Statements `4/4` **100%** Branches `0/0` **100%** Functions `1/1` **100%** Lines `4/4`

File ▲		Statements		Branches		Functions		Lines	
CustomButton.js	🟩	100%	4/4	100%	0/0	100%	1/1	100%	4/4

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/elements/CustomSelector

100% Statements 26/26 **78.57%** Branches 11/14 **100%** Functions 12/12 **100%** Lines 24/24

File		Statements		Branches		Functions		Lines	
CustomDropdown.js	🟩	100%	26/26	78.57%	11/14	100%	12/12	100%	24/24

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/routes

100% Statements 4/4 **100%** Branches 4/4 **100%** Functions 2/2 **100%** Lines 4/4

File ▲		Statements		Branches		Functions		Lines	
AuthenticatedRoute.js	🟩	100%	3/3	100%	4/4	100%	2/2	100%	3/3
app.js	🟩	100%	1/1	100%	0/0	100%	0/0	100%	1/1

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/utils

100% Statements `104/104` **100%** Branches `50/50` **100%** Functions `28/28` **100%** Lines `104/104`

File ▲		Statements		Branches		Functions		Lines	
AuthService.js	🟩	100%	69/69	100%	32/32	100%	22/22	100%	69/69
constants.js	🟩	100%	10/10	100%	6/6	100%	4/4	100%	10/10
validator.js	🟩	100%	25/25	100%	12/12	100%	2/2	100%	25/25

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/views/Dashboard

73.17% Statements 60/82 **37.5%** Branches 6/16 **71.05%** Functions 27/38 **78.95%** Lines 60/76

File		Statements		Branches		Functions		Lines	
CoachCount.js		78.57%	11/14	50%	1/2	71.43%	5/7	84.62%	11/13
Dashboard.js		37.5%	3/8	16.67%	1/6	100%	1/1	37.5%	3/8
EventCount.js		75%	12/16	50%	1/2	62.5%	5/8	85.71%	12/14
JudgeCount.js		78.57%	11/14	50%	1/2	71.43%	5/7	84.62%	11/13
TeamCount.js		78.57%	11/14	50%	1/2	71.43%	5/7	84.62%	11/13
TodaysEvents.js		75%	12/16	50%	1/2	75%	6/8	80%	12/15

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/views/EasterEgg

100% Statements `1/1` **100%** Branches `0/0` **100%** Functions `1/1` **100%** Lines `1/1`

File ▲		Statements		Branches		Functions		Lines	
sudoku.js	🟩	100%	1/1	100%	0/0	100%	1/1	100%	1/1

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/views/Events

65.56% Statements 238/363 **63.79%** Branches 74/116 **36.26%** Functions 33/91 **65.56%** Lines 238/363

File ▲		Statements		Branches		Functions		Lines	
EventDetail.js		41.24%	40/97	26.67%	8/30	25.64%	10/39	41.24%	40/97
Events.js		74.44%	198/266	76.74%	66/86	44.23%	23/52	74.44%	198/266

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/views/Extras

86.05% Statements 37/43 **57.14%** Branches 8/14 **93.75%** Functions 15/16 **86.05%** Lines 37/43

File ▲		Statements		Branches		Functions		Lines	
Extras.js		86.05%	37/43	57.14%	8/14	93.75%	15/16	86.05%	37/43

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/views/Locations

56.11% Statements `101/180` **37.5%** Branches `27/72` **42.11%** Functions `24/57` **56.11%** Lines `101/180`

File		Statements		Branches		Functions		Lines	
Buildings.js	🟥	27.59%	24/87	13.89%	5/36	25.93%	7/27	27.59%	24/87
Rooms.js	🟩	82.8%	77/93	61.11%	22/36	56.67%	17/30	82.8%	77/93

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/views/Schools

85.87% Statements 79/92 **61.76%** Branches 21/34 **60.71%** Functions 17/28 **85.87%** Lines 79/92

File ▲		Statements		Branches		Functions		Lines	
Schools.js		85.87%	79/92	61.76%	21/34	60.71%	17/28	85.87%	79/92

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/views/Teams

100% Statements 19/19 **100%** Branches 8/8 **100%** Functions 12/12 **100%** Lines 19/19

File ▲		Statements		Branches		Functions		Lines	
StudentTeamCreator.js	🟩	100%	10/10	100%	0/0	100%	7/7	100%	10/10
TeamManagement.js	🟩	100%	9/9	100%	8/8	100%	5/5	100%	9/9

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files src/views/UserProfile

100% Statements 96/96 **96.97%** Branches 32/33 **100%** Functions 28/28 **100%** Lines 94/94

File ▲		Statements		Branches		Functions		Lines	
UserProfile.js	🟩	100%	96/96	96.97%	32/33	100%	28/28	100%	94/94

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src setupTests.js

100% Statements `1/1` **100%** Branches `0/0` **100%** Functions `0/0` **100%** Lines `1/1`

```
     import { configure } from 'enzyme';
     import Adapter from 'enzyme-adapter-react-16';

37x  configure( { adapter: new Adapter() } );
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/adapters httpRequest.js

100% Statements 45/45 100% Branches 27/27 100% Functions 13/13 100% Lines 44/44

```
     /** http_request.js
      *
      * Wrapper for Axios library for making HTTP Requests and handling sessions
      */
     import AuthService from '../utils/AuthService';
     import axios from 'axios';
28x  var emptyHeader = {};

28x  axios.interceptors.response.use(response => {
27x      return response;
     }, error => {
29x      var filteredErr = HttpRequest.errorHandler(error);
29x      return Promise.reject(filteredErr);
     })


     export default class HttpRequest {

         static httpRequest(url, httpMethod, httpHeader, requestData, needsSession = false) {
58x          if (httpHeader == null) httpHeader = emptyHeader;

58x          var promise = new Promise(function (resolve, reject) {

                 //IF request requires session, check if User is still Authorized
                 //  and update timestamp if they are
58x              if (needsSession && !AuthService.isAuthorized(true)) {
 1x                  window.location = '/#/login';
 1x                  reject('ERROR: Must be an authorized user.  Please login.');
                 }

58x              if (httpMethod == null) {
 1x                  reject('ERROR: Must supply an HTTP Request.  Please try : GET, DELETE, PUT, POST.');
                 }

58x              if (httpMethod.toLowerCase() === 'get') {
27x                  axios.get(url, httpHeader).then(function (result) {
11x                      resolve({status: result.status, body: result.data});
                     }).catch(function (error) {
16x                      reject(error);
                     })
30x              } else if (httpMethod.toLowerCase() === 'put') {
 2x                  axios.put(url, requestData, httpHeader).then(function (result) {
 1x                      resolve({status: result.status, body: result.data});
                     }).catch(function (error) {
 1x                      reject(error);
                     })
28x              } else if (httpMethod.toLowerCase() === 'post') {
21x                  axios.post(url, requestData, httpHeader).then(function (result) {
12x                      resolve({status: result.status, body: result.data});
                     }).catch(function (error) {
 9x                      reject(error);
                     })
 7x              } else if (httpMethod.toLowerCase() === 'delete') {
 6x                  axios.delete(url, httpHeader).then(function (result) {
```

```
 3x                      resolve({status: result.status, body: result.data});
                     }).catch(function (error) {
 3x                      reject(error);
                     })
                 } else {
 1x                  reject('ERROR: The requested HTTP method ' + httpMethod + ' is not supported');
                 }
             });

58x          return promise;
         }

         static errorHandler(error) {
34x          var errResponse = {};
34x          errResponse.status = 500;

34x          if (error.response) {
33x              if (error.response.status === 401) {
 3x                  if(!error.response.data.includes("incorrect")) {
 2x                      AuthService.revokeAuth(true);
 2x                      window.location = '/#/login';
                     }
                 }

33x              if (error.response.status) {
32x                  errResponse.status = error.response.status;
                 }

33x              if (error.response.data) {
19x                  errResponse.message = error.response.data;
                 } else {
14x                  errResponse.message = "SOMETHING BAD HAPPENED WITH RESPONSE FROM BACKEND.";
                 }
             } else {
 1x              errResponse.message = "SOMETHING BAD HAPPENED WITH RESPONSE FROM BACKEND: " + error;
             }

34x          return errResponse;
         }
     }

```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/components/Buildings BuildingSelector.js

100% Statements 21/21 **66.67%** Branches 8/12 **84.62%** Functions 11/13 **100%** Lines 20/20

```
     import React from 'react';
     import SelectField from 'material-ui/SelectField';
     import MenuItem from 'material-ui/MenuItem';
     import HttpRequest from '../../adapters/httpRequest';
     import constants from '../../utils/constants';

     class BuildingSelector extends React.Component {

         constructor(props) {
 4x          super(props);

 4x          this.state ={
                 errorMsg: this.props.errorMsg,
                 buildingList: [],
                 roomList:[],
             }

         }

         sortByKey(array, key) {
 6x          return array.sort(function(a, b) {
 6x              var x = a[key]; var y = b[key];
 6x              return ((x < y) ? -1 : ((x > y) ? 1 : 0));
             });
         }

         componentWillMount() {
 3x          var _this = this;
 3x          _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/getAllRooms',
 3x  'GET',constants.useCredentials(), null).then(function (result) {
                 _this.setState({roomList: _this.sortByKey(result.body, "roomName")})
             }).catch(function (error) {

 3x          })
 3x          _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/getBuildings',
     'GET',constants.useCredentials(), null).then(function (result) {
                 _this.setState({buildingList: _this.sortByKey(result.body, "building")})
             }).catch(function (error) {

             })
         }

 1x      componentWillReceiveProps(nextProps) {
 1x
             E if(JSON.stringify(this.props.errorMsg) !== JSON.stringify(nextProps.errorMsg))
                 this.setState({errorMsg: nextProps.errorMsg})
         }
 1x
         updateValues = (event, index, value) =>  {
             this.props.callBack(event, index, value);
         }
12x
 5x      render() {
             if (Object.keys(this.state.buildingList).length !== 0 && Object.keys(this.state.roomList).length
     !== 0 ) {
                 return (
                     <SelectField
                         name={"buildings"}
                         hintText={this.props.hintText}
                         errorText={this.state.errorMsg}
                         floatingLabelText={this.props.labelText}
                         onChange={this.updateValues}
                         fullWidth={true}
10x                      value={this.props.selected}>
20x                      {
10x                          Object.keys(this.state.buildingList).map(function (key) {
                                 var items = Object.keys(this.state.roomList).map(function (roomKey){

     if(this.state.buildingList[key].id.indexOf(this.state.roomList[roomKey].buildingID) > -1) {
                                         return (
                                             <MenuItem key={this.state.roomList[roomKey].id}
                                                       primaryText={this.state.buildingList[key].building +' --
     ' +  this.state.roomList[roomKey].roomName}
10x                                          //  secondaryText={this.state.roomList[roomKey].roomName}
                                                       value={this.state.roomList[roomKey].id}/>
                                         )
                                     }
                                 }, this)
                                 return (
                                     items
                                 )
                             }, this)

 7x                      }
                     </SelectField>
                 )
             }
             else
                 return ("ERROR LOADING BUILDINGS")
         }
     }

     export default BuildingSelector;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/components/Cards Card.js

100% Statements 1/1 **100%** Branches 14/14 **100%** Functions 1/1 **100%** Lines 1/1

```
    import React, { Component } from 'react';


    export class Card extends Component{
        render(){
13x         return (
                <div className={"card"+(this.props.plain ? " card-plain":"")} style={this.props.style}>
                    <div className={"header"
                        + (this.props.hCenter ? " text-center":"")}>
                        <h4 className="title">{this.props.title}</h4>
                        <p className="category">{this.props.category}</p>
                    </div>
                    <div className={"content"
                        + (this.props.ctAllIcons ? " all-icons":"")
                        + (this.props.ctTableFullWidth ? " table-full-width":"")
                        + (this.props.ctTableResponsive ? " table-responsive":"")
                        + (this.props.ctTableUpgrade ? " table-upgrade":"")}>

                        {this.props.content}

                        <div className="footer">
                            {this.props.legend}
                            {this.props.stats != null ? <hr />:""}
                            <div className="stats">
                                <i className={this.props.statsIcon}></i> {this.props.stats}
                            </div>
                        </div>
                    </div>
                </div>
            );
        }
    }

    export default Card;

```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/components/Cards StatsCard.js

100% Statements `1/1` **100%** Branches `0/0` **100%** Functions `1/1` **100%** Lines `1/1`

```
    import React, { Component } from 'react';
    import { Row, Col } from 'react-bootstrap';

    export class StatsCard extends Component{
        render(){
11x         return (
                <div className="card card-stats" style={{'text-align':'left'}}>
                    <div className="content">
                        <div className="numbers">
                            <p style={{'text-align':'left','font-weight':'bold', 'font-size':'14pt'}}>
    {this.props.bigIcon} {this.props.statsText}</p>
                        </div>
                        <div className="numbers">
                            <p style={{'text-align':'left'}}>{this.props.statsValue}</p>
                        </div>
                    </div>
                </div>
            );
        }
    }

    export default StatsCard;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/components/Cards UserCard.js

100% Statements 1/1 **100%** Branches 0/0 **100%** Functions 1/1 **100%** Lines 1/1

```
    import React, { Component } from 'react';


    export class UserCard extends Component{
        render(){
1x          return (
                <div className="card card-user">
                    <div className="image">
                        <img src={this.props.bgImage} alt="..."/>
                    </div>
                    <div className="content">
                        <div className="author">
                             <a href="#/app/user">
                                <img className="avatar border-gray" src={this.props.avatar} alt="..."/>
                                <h4 className="title">
                                    {this.props.name}
                                    <br />
                                    <small>{this.props.userName}</small>
                                </h4>
                            </a>
                        </div>
                        <p className="description text-center">
                            {this.props.description}
                        </p>
                    </div>
                    <hr />
                    <div className="text-center">
                        {this.props.socials}
                    </div>
                </div>
            );
        }
    }

    export default UserCard;

```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/components/Footer Footer.js

100% Statements `1/1` **100%** Branches `0/0` **100%** Functions `1/1` **100%** Lines `1/1`

```
   import React, {Component} from 'react';
   import { Grid } from 'react-bootstrap';

   class Footer extends Component {
           render() {
3x                 return (
               <footer className="footer">
                   <Grid>
                       <nav className="pull-left">
                           <ul>
                               <li>
                                   <a href="/#/app/dashboard">
                                       Home
                                   </a>
                               </li>
                               <li>
                                   <a href="http://www.psu.edu">
                                       PSU
                                   </a>
                               </li>
                               <li>
                                   <a href="https://www.soinc.org/">
                                       Science Olympiad
                                   </a>
                               </li>
                           </ul>
                       </nav>
                       <p className="copyright pull-right">
                           &copy; {(new Date()).getFullYear()} - SWENG500 - Group 5
                       </p>
                   </Grid>
               </footer>
                   );
           }
   }

   export default Footer;

```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/components/Header Header.js

91.3% Statements 21/23 **87.5%** Branches 7/8 **100%** Functions 6/6 **91.3%** Lines 21/23

```
     import React, { Component } from 'react';
     import { Navbar } from 'react-bootstrap';

     import HeaderLinks from './HeaderLinks.js';

     import appRoutes from '../../routes/app.js';

     class Header extends Component{
         constructor(props){
 5x          super(props);
 5x          this.mobileSidebarToggle = this.mobileSidebarToggle.bind(this);
 5x          this.state = {
                 sidebarExists: false
             };
         }
         mobileSidebarToggle(e){
 2x          if(this.state.sidebarExists === false){
 1x              this.setState({
                     sidebarExists : true
                 });

             }
 2x          e.preventDefault();
 2x          document.documentElement.classList.toggle('nav-open');
 2x          var node = document.createElement('div');
 2x          node.id = 'bodyClick';
 2x          node.onclick = function(){
 1x              this.parentElement.removeChild(this);
                 document.documentElement.classList.toggle('nav-open');
             };
 2x          document.body.appendChild(node);
         }
         getBrand(){
 8x          var name = null;
 8x          appRoutes.map((prop,key) => {
80x              if(prop.redirect){
 8x                  I if(prop.path === this.props.location.pathname){
                         name = prop.name;
                     }
                 }else{
72x                  if(prop.path === this.props.location.pathname){
 5x                      name = prop.name;
                     }
                 }
80x              return null;
             })
 8x          return name;
         }
         render(){
 8x          return (
                 <Navbar fluid>
                     <Navbar.Header>
                         <Navbar.Brand>
                             <div id="brand-display">{this.getBrand()}</div>
```

```
                    </Navbar.Brand>
                    <Navbar.Toggle onClick={this.mobileSidebarToggle}/>
                </Navbar.Header>
                <Navbar.Collapse>
                    <HeaderLinks />
                </Navbar.Collapse>
            </Navbar>
        );
    }
}

export default Header;

```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/components/Header HeaderLinks.js

100% Statements `8/8` **100%** Branches `0/0` **100%** Functions `6/6` **100%** Lines `8/8`

```
    import React, {Component} from 'react';
    import { NavItem, Nav } from 'react-bootstrap';

    import AuthService from '../../utils/AuthService';


    class HeaderLinks extends Component{

        constructor(props) {
4x          super(props);

4x          this.onLogoutClick = this.onLogoutClick.bind(this);
        }

        onLogoutClick(event) {
2x          event.preventDefault();

2x          AuthService.logout().then(function (result) {
1x              console.log(result);
            }).catch(function (error) {
1x              console.log(error);
            }).then( () => {
2x              window.location = '/';
            });

        }

        render(){
5x          return (
                <div>
                    <Nav pullRight>
                        <NavItem eventKey={1} href="/#/app/user">Account</NavItem>
                        <NavItem eventKey={3} onClick={this.onLogoutClick}>Log out</NavItem>
                    </Nav>
                </div>
            );
        }
    }

    export default HeaderLinks;

```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/components/Login Forgot.js

100% Statements 17/17 **100%** Branches 2/2 **100%** Functions 9/9 **100%** Lines 17/17

```
    import React, { Component } from 'react';
    import MuiThemeProvider from 'material-ui/styles/MuiThemeProvider';
    import RaisedButton from 'material-ui/RaisedButton';
    import TextField from 'material-ui/TextField';
    import AppBar from 'material-ui/AppBar';
    import NotificationSystem from 'react-notification-system';

    import {Redirect} from 'react-router-dom';

    import AuthService from '../../utils/AuthService';
    import HttpRequest from "../../adapters/httpRequest";
    import constants from "../../utils/constants";
    import {style} from "../../variables/Variables";
    import $ from "jquery";
    import Login from "./Login";
    import PropTypes from "prop-types";

    class Forgot extends Component {
        constructor(props) {
5x          super(props);

5x          this.state = {
                email: ''
            };

5x          this.handleClick = this.handleClick.bind(this);
        }

        handleClick(event) {

4x          if(this.state.email.trim() !== '') {
3x              var body = {};
3x              var _this = this;

3x              body.emailAddress = this.state.email.trim();;

3x              _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
1x  '/sweng500/resetPassword', 'POST', constants.useCredentials(), body).then(function (result) {
1x                  _this.setState({email : "", emailRequired: null}, () => {
                        _this.props.notify(
                            "Success: Password reset email sent",
                            "success",
                            "tc",
                            5
1x                      );
                        $('#reset-container').addClass('collapse');
                    });
2x              }).catch(function (error) {
                    _this.setState({
                        email: _this.state.email.trim(),
                        emailRequired: "Provide valid email address."
2x                  }, () => {
                        _this.props.notify(
                            "Error: Could not send password reset email.  Provide a valid email address.",
                            "error",
                            "tc",
                            5
                        );
                    });
                })

1x            } else {
                this.setState({
                    email: this.state.email.trim(),
                    emailRequired: "An email address is required."
                })
            }
        }

10x        render() {
            return (
                <div id='login-div'>
                    <MuiThemeProvider>
                       <div>
                            <AppBar showMenuIconButton={false} title="Password Recovery"/>
                           <div id='reset-container'>
                                <TextField
                                    hintText="Enter your email address"
                                    errorText={this.state.emailRequired}
1x                                    floatingLabelText="Email Address"
                                    onChange={(event, newValue) => this.setState({email: newValue})}
                                    required={true}
                                />
                                <br/>
4x                                <RaisedButton label="Submit" primary={true} style={{margin:15}}
                                              onClick={(event) => this.handleClick(event)}/>
                           </div>
                        </div>
                    </MuiThemeProvider>
                </div>
            )
        }
    }

    export default Forgot;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/components/Login Login.js

100% Statements 24/24 **100%** Branches 12/12 **100%** Functions 10/10 **100%** Lines 23/23

```
    import React, { Component } from 'react';
    import PropTypes from 'prop-types';
    import MuiThemeProvider from 'material-ui/styles/MuiThemeProvider';
    import RaisedButton from 'material-ui/RaisedButton';
    import TextField from 'material-ui/TextField';
    import AppBar from 'material-ui/AppBar';
    import PasswordField from 'material-ui-password-field'

    import {Redirect} from 'react-router-dom';

    import AuthService from '../../utils/AuthService';

    class Login extends Component {
        constructor(props) {
8x          super(props);

8x          this.state = {
                emailAddress: '',
                password:'',
                emailRequired: undefined,
                passRequired: undefined,
                redirect:false
            }

8x          this.handleClick = this.handleClick.bind(this);
        }

        handleClick(event) {
5x          if(event) { event.preventDefault(); }

5x          if(this.state.emailAddress.trim()) {
4x              if(this.state.password.trim()) {

3x                  AuthService.login(this.state.emailAddress.toLowerCase(),
    this.state.password).then(function (result) {

2x                  }).catch(function (error) {
                        console.log(error);
3x                  }).then(() => {
1x                          if (AuthService.isLoggedIn()) {
                                this.setState({
                                    redirect: true
                                })
2x                          } else {
                                this.props.notify(
                                    "ERROR: Please provide valid login credentials.",
                                    "error",
                                    "tc",
                                    6
                                )
                            }
                        }
                    );
1x              } else {
```

```
                        this.setState({
                            emailRequired: undefined,
                            passRequired: "Password is required."
                        })
                    }
1x              } else {
                    this.setState({
                        emailAddress: this.state.emailAddress.trim(),
                        emailRequired: "An email address is required.",
                        passRequired: undefined
                    })
                }
            }

8x          componentDidMount() {
1x              if(AuthService.isLoggedIn()) {
                    this.setState({
                        redirect: true
                    })
                }
            }

19x         render() {
16x             if(!this.state.redirect) {
                    return (
                        <div id='login-div'>
                            <MuiThemeProvider>
                                <div style={{width:275}}>
                                    <AppBar showMenuIconButton={false} title="Login"/>
                                    <TextField
                                        name={"email"}
                                        type="Email Address"
                                        hintText="Enter your email address"
                                        errorText={this.state.emailRequired}
2x                                      floatingLabelText="Email Address"
                                        onChange={(event, newValue) => this.setState({emailAddress: newValue})}
                                        required={true}
                                        fullWidth={true}
                                    />
                                    <br/>
                                    <PasswordField
                                        name={"password"}
                                        type="password"
                                        errorText={this.state.passRequired}
2x                                      floatingLabelText="Password"
                                        onChange={(event, newValue) => this.setState({password: newValue})}
                                        fullWidth={true}
                                        required={true}
                                    />
                                    <br/>
7x                                  <RaisedButton label="Submit" primary={true} style={{margin:15}}
                                                  onClick={(event) => this.handleClick(event)}/>
                                </div>
                            </MuiThemeProvider>
                        </div>
                    )
3x              } else {
                    return (
                        <Redirect from="/" to="/app/dashboard" />
                    )
                }
            }
        }
    3x
        Login.propTypes = {
            notify: PropTypes.any,
    3x  }
        Login.defaultProps = {
            notify : null
        }

        export default Login;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/components/Login Signup.js

93.68% Statements 89/95 83.33% Branches 35/42 80% Functions 16/20 94.62% Lines 88/93

```
1         import React from 'react';
2         import {
3             Step,
4             Stepper,
5             StepLabel,
6             StepContent,
7         } from 'material-ui/Stepper';
8         import RaisedButton from 'material-ui/RaisedButton';
9         import FlatButton from 'material-ui/FlatButton';
10        import MuiThemeProvider from 'material-ui/styles/MuiThemeProvider';
11        import TextField from 'material-ui/TextField';
12        import AppBar from 'material-ui/AppBar';
13        import InputMask from 'react-input-mask'
14        import {Row, Col, Grid} from 'react-bootstrap';
15        import HttpRequest from '../../adapters/httpRequest';
16        import constants from '../../utils/constants';
17        import PasswordField from 'material-ui-password-field';
18        import SchoolSelector from '../Schools/SchoolSelector';
19
20        class Signup extends React.Component {
21
22            constructor(props) {
23  11x           super(props);
24
25  11x           this.state = {
26                    finished: false,
27                    stepIndex: 0,
28                    firstName: '',
29                    lastName: '',
30                    phoneNumber: '',
31                    emailAddress: '',
32                    password: '',
33                    confirm: '',
34                    school: '',
35                    httpResponse: '',
36                    accountMessage: '',
37                    schoolList: {}
38                };
39
40  11x           this.callBack = this.callBack.bind(this);
41            }
42
43            componentDidMount() {
44  11x           var _this = this;
45
46  11x           _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/getSchools',
47        'GET', null, null).then(function (result) {
48  11x
49                    _this.state.schoolList = result.body;
50
51                }).catch(function (error) {
52
53                })
54            }
```

```
    
        // Checks to see if an email has a host, @ symbols, and domain.
2x      validEmail(text) {
2x          let reg = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
1x          if (reg.test(text) === false)
                return true;
1x          else
                return false;
        }
    
        // Check if a passowrd is valid. 8 characters, uppercase, lowercase, and numbers
5x      validPassword(text) {
2x          if (text.length < 8) return false;
2x          var hasUpperCase = /[A-Z]/.test(text);
2x          var hasLowerCase = /[a-z]/.test(text);
2x          var hasNumbers = /\d/.test(text);
2x          var hasNonalphas = /\W/.test(text);
            I if (hasUpperCase + hasLowerCase + hasNumbers + hasNonalphas < 3) return false;
2x  
            return true;
        }
    
        callBack(event, index, value) {
    
            this.setState({school: value});
        }
    
        // Handles the next button
9x      handleNext = () => {
9x          const {stepIndex} = this.state;
9x          var _this = this;
            var body = {};
    
9x          // Flag to indicate if we can proceed
            var missingInfo = false;
    
    
9x          // Form page 1
            if (stepIndex === 0) {
    
3x              // Checks first name
1x              if (this.state.firstName.trim()) {
                    this.setState({
                        firstName: this.state.firstName.trim(),
                        firstNameRequired: undefined
                    })
                }
2x              else {
                    this.setState({
                        firstName: this.state.firstName.trim(),
                        firstNameRequired: "First name is required."
2x                  })
                    missingInfo = true;
                }
    
3x              // Checks last name
1x              if (this.state.lastName.trim()) {
                    this.setState({
                        lastName: this.state.lastName.trim(),
                        lastNameRequired: undefined
                    })
                }
2x              else {
                    this.setState({
                        lastName: this.state.lastName.trim(),
                        lastNameRequired: "Last name is required."
2x                  })
                    missingInfo = true;
                }

3x              // Checks phone number
1x              if (this.state.phoneNumber.trim()) {
                    this.setState({
                        phoneNumber: this.state.phoneNumber.trim(),
                        phoneNumberRequired: undefined
                    })
                }
2x              else {
                    this.setState({
                        phoneNumber: this.state.phoneNumber.trim(),
                        phoneNumberRequired: "A phone number is required."
2x                  })
                    missingInfo = true;
                }

3x              // Checks emails address is not blank
                if (this.state.emailAddress.trim()) {

2x                  // If the address is not valid
1x                  if (this.validEmail(this.state.emailAddress.trim())) {
                        this.setState({
                            emailAddress: this.state.emailAddress.trim(),
                            emailAddressRequired: "A valid email address is required."
                        })
1x
                        missingInfo = true;
                    }
                    else {

                        // Check to see if the email address already exists
1x
                        body.emailAddress = this.state.emailAddress.toLowerCase();
1x
                        _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
    '/sweng500/emailAvailable', 'POST', null, body ).then(function (result) {
1x

                            _this.setState({
                                emailAddress: _this.state.emailAddress.trim(),
                                emailAddressRequired: "",
                                httpResponse: result,
1x                          })

1x                          E if (missingInfo === false)
                            {
                                _this.setState({
                                    stepIndex: stepIndex + 1
                                })
                            }

                        }).catch(function (error) {


                            if (_this.state.httpResponse.status !== 200)
                                missingInfo = true;

                            _this.setState({
                                emailAddress: _this.state.emailAddress.trim(),
                                emailAddressRequired: "This address has already been registered.",
                                httpResponse: error
                            })
                        })
                    }

1x              }
                else {
                    this.setState({
                        emailAddress: this.state.emailAddress.trim(),
1x                      emailAddressRequired: "An email address is required."
                    })
                    missingInfo = true;
                }

6x
            } // Form page 2
            else E if (stepIndex === 1) {
6x
1x              // Checks to see if the passwords are equal to each other
                if (this.state.password.trim() !== this.state.confirm.trim()) {
                    this.setState({
                        confirmRequired: "Passwords must match.",
1x                      passwordRequired: "Passwords must match."
                    })
                    this.props.notify(
                        "ERROR: Your passwords do not match.",
                        "error",
                        "tc",
1x                      6
                    );
                    missingInfo = true;
                }
5x              else {
2x                  // Tests the password complexity
                    if (this.validPassword(this.state.password.trim())) {
                        this.setState({
                            password: this.state.password.trim(),
                            passwordRequired: undefined
                        })
3x                  }
                    else {
                        this.props.notify(
                            "ERROR: Your password must be 8 or more characters, contain capital letters, lower case letters, and at least one number.",
                            "error",
3x                          "tc",
                            10
                        );
                        this.setState({
                            confirmRequired: "A complex password is required.",
3x                          passwordRequired: "A complex password is required."
```

```
                        })

                        missingInfo = true;
6x                  }
                }
5x
                if (this.state.school.trim() === '')
                {
                    this.setState({
                        schoolRequired: "Please select your school district."
1x                  })
                }
                else {
6x                  this.setState({schoolRequired: undefined})
                }
2x
                if (!missingInfo) {

                    this.setState({
                        stepIndex: stepIndex + 1,
                        finished: stepIndex >= 2,
2x                  });
2x
2x                  // Create the user account
2x                  var cleanPhoneNumber = this.state.phoneNumber;
2x                  cleanPhoneNumber = cleanPhoneNumber.replace(/\s/g, '');         // Remove spaces
                    cleanPhoneNumber = cleanPhoneNumber.replace(/\(|\)/g,'');       // Remove ( and )
2x                  cleanPhoneNumber = cleanPhoneNumber.replace(/-/g,"");           // Remove -
                    cleanPhoneNumber = '+' + cleanPhoneNumber;                      // Add +
2x
2x                  body = {};
2x
2x                  body.firstName = this.state.firstName;
2x                  body.lastName = this.state.lastName;
2x                  body.emailAddress = this.state.emailAddress.toLowerCase();
2x                  body.phoneNumber = cleanPhoneNumber;
                    body.password = this.state.password;
                    body.minutesBeforeEvent = 10;
2x                  var schoolID = this.state.school;

2x
2x                 _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
    '/sweng500/addUser/?userType=COACH&schoolID=' + _this.state.school, 'POST', null, body ).then(function
    (result) {

                        E if (result.status === 200)
                            _this.setState({accountMessage: "Congratulations! Your account has been created.
    Please return to the login screen."})

                    }).catch(function (error) {

                        _this.setState({accountMessage:"There was an error creating your account. Please try
    again later."})

                    })

1x              }

1x          }
1x
```

```
        };

        handlePrev = () => {
            const {stepIndex} = this.state;
98x
98x         E if (stepIndex > 0) {
98x             this.setState({stepIndex: stepIndex - 1});
            }
        };

        render() {
            const {finished, stepIndex} = this.state;
            const contentStyle = {margin: '0 16px'};
            return (
                <MuiThemeProvider>
                    <div>
                        <AppBar showMenuIconButton={false} title="Account Registration"/>
                        <Stepper activeStep={stepIndex} orientation={'vertical'} style={{maxWidth: 700,
    margin: '0',border: '1 solid black'}}>
                            <Step>
                                <StepLabel>Personal Information</StepLabel>
                                <StepContent>
 1x                                 <Grid>
                                        <Row className="show-grid">
                                            <Col xs={7} md={3}>
                                                <TextField
                                                    name="fname"
                                                    hintText="Enter your first name"
                                                    errorText={this.state.firstNameRequired}
                                                    floatingLabelText="First name"
                                                    onChange={(event, newValue) => this.setState({firstName:
    newValue})}

                                                    value={this.state.firstName}
 1x                                                 fullWidth={true}
                                                    required={true}/>
                                            </Col>
                                            <Col xs={7} md={3}>
                                                <TextField
                                                    name="lname"
                                                    hintText="Enter your last name"
                                                    errorText={this.state.lastNameRequired}
                                                    floatingLabelText="Last name"
                                                    onChange={(event, newValue) => this.setState({lastName:
    newValue})}

                                                    value={this.state.lastName}
 1x                                                 fullWidth={true}
                                                    required={true}/>
                                            </Col>
                                        </Row>
                                        <Row className="show-grid">
                                            <Col xs={7} md={3}>
                                                <TextField
                                                    name="phone"
                                                    errorText={this.state.phoneNumberRequired}
                                                    floatingLabelText="Phone number"
                                                    onChange={(event, newValue) => this.setState({phoneNumber:
    newValue})}

                                                    value={this.state.phoneNumber}
                                                    fullWidth={true}
                                                    required={true}>
```

```
1x
                                                    <InputMask mask="1 (999) 999-9999" maskChar="#"
                                                               value={this.state.phoneNumber}/>
                                                </TextField>
                                            </Col>
                                            <Col xs={7} md={3}>
                                                <TextField
                                                    name="email"
                                                    hintText="Enter your email address"
                                                    errorText={this.state.emailAddressRequired}
                                                    floatingLabelText="Email address"
                                                    onChange={(event, newValue) =>
    this.setState({emailAddress: newValue})}
                                                    value={this.state.emailAddress}
                                                    fullWidth={true}
                                                    required={true}/>
                                            </Col>
                                        </Row>
                                    </Grid>
                                </StepContent>
                            </Step>
1x                          <Step>
                                <StepLabel>Account Information</StepLabel>
                                <StepContent>
                                    <Grid>
                                        <Row className="show-grid">
                                            <Col xs={7} md={3}>
                                                <PasswordField
                                                    name={"password"}
                                                    type="password"
                                                    hintText="Enter your password"
                                                    errorText={this.state.passwordRequired}
                                                    floatingLabelText="Password"
1x                                                  onChange={(event, newValue) => this.setState({password:
    newValue})}
                                                    value={this.state.password}
                                                    fullWidth={true}
                                                    required={true}/>
                                            </Col>
                                            <Col xs={7} md={3}>
                                                <PasswordField
                                                    name={"confirm"}
                                                    type="password"
                                                    hintText="Confirm your password"
                                                    errorText={this.state.confirmRequired}
                                                    floatingLabelText="Confirm your password"
                                                    onChange={(event, newValue) => this.setState({confirm:
    newValue})}
                                                    value={this.state.confirm}
                                                    fullWidth={true}
                                                    required={true}/>
                                            </Col>
                                        </Row>
                                        <Row className="show-grid">
                                            <Col xs={7} md={3}>
                                                <SchoolSelector selected={this.state.school} errorMsg=
    {this.state.schoolRequired}
                                                                callBack={this.callBack} labelText={"School"}
    hintText={"Select your school"}/>
                                            </Col>
                                        </Row>
                                    </Grid>
                                </StepContent>
                            </Step>
                            <Step>
                                <StepLabel>Account Creation</StepLabel>
                                <StepContent>
                                    <Grid>
                                        <Row className="show-grid">
                                            <Col>
                                                <div>{this.state.accountMessage}</div>
                                           </Col>
                                        </Row>
                                    </Grid>
                                </StepContent>
                            </Step>
                        </Stepper>
                        <div style={contentStyle} className={stepIndex === 2 ? 'collapse' : ''}>
                            <div>
                                <div style={{marginTop: 12}}>
                                    <FlatButton
                                        label="Back"
                                        disabled={stepIndex === 0 || stepIndex === 2}
                                        onClick={this.handlePrev}
                                        style={{marginRight: 12}}/>
                                    <RaisedButton
                                        label={stepIndex === 2 ? 'Return to login' : 'Next'}
                                        primary={true}
                                        onClick={this.handleNext}/>
                                </div>
                            </div>
                        </div>
                    </div>
                </MuiThemeProvider>
            );
        }
    }

    export default Signup;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/components/Schools SchoolSelector.js

100% Statements 16/16 **50%** Branches 4/8 **90%** Functions 9/10 **100%** Lines 15/15

```
     import React from 'react';
     import SelectField from 'material-ui/SelectField';
     import MenuItem from 'material-ui/MenuItem';
     import HttpRequest from '../../adapters/httpRequest';
     import constants from '../../utils/constants';

     class SchoolSelector extends React.Component {

         constructor(props) {
4x           super(props);

4x           this.state = {
                 errorMsg: this.props.errorMsg,
                 schoolList: [],
             }

         }

         sortByKey(array, key) {
3x           return array.sort(function(a, b) {
3x               var x = a[key]; var y = b[key];
3x               return ((x < y) ? -1 : ((x > y) ? 1 : 0));
             });
         }

         componentWillMount() {
3x           var _this = this;

3x           _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/getSchools',
3x   'GET', null, null).then(function (result) {
                 _this.setState({schoolList: _this.sortByKey(result.body, "schoolName")})
             }).catch(function (error) {

             })
         }

         componentWillReceiveProps(nextProps) {
1x
1x           E if(JSON.stringify(this.props.errorMsg) !== JSON.stringify(nextProps.errorMsg))
                 this.setState({errorMsg: nextProps.errorMsg})
         }

1x       updateValues = (event, index, value) =>  {
             this.props.callBack(event, index, value);
         }

9x       render() {
5x           if (Object.keys(this.state.schoolList).length !== 0) {
                 return (
                     <SelectField
                         name={"schools"}
                         hintText={this.props.hintText}
                         errorText={this.state.errorMsg}
                         floatingLabelText={this.props.labelText}
```

```
                        onChange={this.updateValues}
                        maxHeight={200}
                        value={this.props.selected}>
                        {
10x                         Object.keys(this.state.schoolList).map(function (key) {
                                return (
                                    <MenuItem key={this.state.schoolList[key].id}
                                              primaryText={this.state.schoolList[key].schoolName}
                                              value={this.state.schoolList[key].id}/>
                                )
                            }, this)

                        }
                    </SelectField>
                )
            }
4x          else
                return ("ERROR LOADING SCHOOLS")
        }
    }

    export default SchoolSelector;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/components/Sidebar Sidebar.js

100% Statements 11/11   83.33% Branches 10/12   100% Functions 6/6   100% Lines 11/11

```
     import React, {Component} from 'react';
     import { NavLink } from 'react-router-dom';

     import HeaderLinks from '../Header/HeaderLinks.js';

     import imagine from '../../assets/img/Mountain.jpg';
     import logo from '../../assets/img/reactlogo.png';

     import appRoutes from '../../routes/app.js';
     import AuthService from "../../utils/AuthService";

     class Sidebar extends Component{
         constructor(props){
 4x          super(props);
 4x          this.state = {
                 width: window.innerWidth
             }
         }
         activeRoute(routeName) {
47x          return this.props.location.pathname.indexOf(routeName) > -1 ? 'active' : '';
         }
         updateDimensions(){
 4x          this.setState({width:window.innerWidth});
         }
         componentDidMount() {
 4x          this.updateDimensions();
 4x          window.addEventListener("resize", this.updateDimensions.bind(this));
         }
         render(){
 9x          const sidebarBackground = {
                 backgroundImage: 'url(' + imagine + ')'
             };
 9x          return (
                 <div id="sidebar" className="sidebar" data-color="black" data-image={imagine}>
                     <div className="sidebar-background" style={sidebarBackground}></div>
                         <div className="logo">
                             <a href="#dashboard" className="simple-text logo-mini App-logo">
                                 <div className="logo-img">
                                     <img src={logo} alt="logo_image"/>
                                 </div>

                             </a>
                             <a href="#dashboard" className="simple-text logo-normal">
                                 Science Olympiad
                             </a>
                         </div>
                     <div className="sidebar-wrapper">
                         <ul className="nav">
                             { this.state.width <= 991 ? (<HeaderLinks />):null }
                             {
                                 appRoutes.map((prop,key) => {
                                     //DO NOT RENDER SIDEBAR LINK IF USER IS NOT ALLOWED TO VIEW
90x                                  if(prop.redirect || prop.linkOnly ||(prop.users &&
43x  !AuthService.isUserRoleAllowed(prop.users))) {
```

```
                                    return null;
                                }
47x
                                return (
                                    <li className={prop.upgrade ? "active active-pro" :
this.activeRoute(prop.path)}
                                        key={key}>
                                        <NavLink to={prop.path} className="nav-link"
activeClassName="active">
                                            <i className={prop.icon}></i>
                                            <p>{prop.name}</p>
                                        </NavLink>
                                    </li>
                                );
                            })
                        }
                    </ul>
                </div>
            </div>
        );
    }
}

export default Sidebar;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/components/Students StudentAdder.js

100% Statements 33/33 100% Branches 10/10 100% Functions 13/13 100% Lines 33/33

```
    import React, {Component} from 'react';
    import {MuiThemeProvider, AppBar, TextField, RaisedButton, FontIcon} from 'material-ui';
    import {Grid, Row, Col} from 'react-bootstrap';

    import CustomDropdown from "../../elements/CustomSelector/CustomDropdown";
    import constants from "../../utils/constants";
    import Validator from "../../utils/validator";
    import HttpRequest from "../../adapters/httpRequest";

    class StudentAdder extends Component {
        constructor(props) {
4x          super(props);

4x          this.state = {
                firstName: "",
                lastName: "",
                emailAddress: "",
                selectedSchool: undefined,
                errors: {}
            }

4x          this.selectedSchool = this.selectedSchool.bind(this);
4x          this.validateStudentForm = this.validateStudentForm.bind(this);
        }

        selectedSchool(value) {
1x          this.setState({
                selectedSchool: value
            })
        }

        addStudent() {
2x          var body = {};
2x          body.firstName = this.state.firstName;
2x          body.lastName = this.state.lastName;
2x          body.emailAddress = this.state.emailAddress;

2x          var _this = this;

2x          _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/addUser/?
1x  userType=STUDENT&schoolID=' + _this.state.selectedSchool.id, 'POST', constants.useCredentials(), body,
    true).then(function (result) {
                _this.setState({
                    firstName: "",
                    lastName: "",
                    emailAddress: "",
1x                  selectedSchool: undefined
1x              }, () => {
                    _this.props.addNotification(<div><b>{body.firstName + ' ' + body.lastName}</b> has been
    created.</div>, 'success');
                    _this.props.updateTable();
                });
1x
                // alert(result.body);
```

```
             }).catch(function (error) {
                 _this.props.addNotification(<div><b>{body.firstName}</b> could not be created because: <em>
     {error.message}</em></div>, 'error');
6x           });
         }
6x   
1x       validateStudentForm() {
             var errors = {};
6x   
2x           if(this.state.firstName.trim() === "") {
                 errors.firstNameError = "First name is required";
             }
6x           if(this.state.lastName.trim() === "") {
                 errors.lastNameError = "Last name is required";
6x           }
3x   
             let result = Validator.validateEmail(this.state.emailAddress);
     
6x           if(!result.isValid) {
4x               errors.emailAddressError = result.message;
             }
     
6x           if( !this.state.selectedSchool ){
6x               errors.schoolError = "School is required"
2x           }
     
             this.setState({errors: errors}, () => {
                 if(constants.isEmpty(this.state.errors)) {
                     this.addStudent();
                 }
19x          });
         }
     
         render() {
             return (
                 <MuiThemeProvider>
                     <div style={{textAlign: 'center'}}>
                         <AppBar showMenuIconButton={false} title="Register Student" style={{zIndex: 10}}/>
                         <Grid>
                             <Row className="text-center" >
                                 <Col xs={7} md={3}>
                                     <TextField
2x                                       name="fname"
                                         hintText="Student's First Name"
                                         errorText={this.state.errors.firstNameError}
                                         floatingLabelText="First name"
                                         onChange={(event, newValue) => this.setState({firstName:
     newValue.trim()})}
                                         value={this.state.firstName}
                                         fullWidth={true}
                                         required={true}
                                         style={{textAlign: 'left'}}
                                     />
                                 </Col>
                                 <Col xs={7} md={3}>
2x                                   <TextField
                                         name="lname"
                                         hintText="Student's Last Name"
                                         errorText={this.state.errors.lastNameError}
                                         floatingLabelText="Last name"
```

```
                                    onChange={(event, newValue) => this.setState({lastName:
newValue.trim()})}
                                    value={this.state.lastName}
                                    fullWidth={true}
                                    required={true}
                                    style={{textAlign: 'left'}}
                                />
                            </Col>
                        </Row>
                        <Row className="show-grid">
2x                          <Col xs={7} md={3}>
                                <TextField
                                    name="email"
                                    hintText="Student's Email Address"
                                    errorText={this.state.errors.emailAddressError}
                                    floatingLabelText="Email address"
                                    onChange={(event, newValue) => this.setState({emailAddress:
newValue.trim()})}
                                    value={this.state.emailAddress}
                                    fullWidth={true}
                                    required={true}
                                    style={{textAlign: 'left'}}
                                />
                            </Col>
                            <Col xs={7} md={3}>
                                <CustomDropdown
                                    textAlign={"left"}
                                    name={"school"}
                                    labelText={"School"}
                                    hintText={"Select School"}
                                    errorText={this.state.errors.schoolError}
                                    selected={this.state.selectedSchool}
                                    endpoint={"/sweng500/getSchools"}
                                    sortKey={"schoolName"}
1x                                  textKeys={["schoolName"]}
                                    selectedValue={this.selectedSchool}
                                />
                            </Col>
                        </Row>
                        <Row className="center-block">
                            <Col sm={6} smOffset={1} style={{maxWidth: 200}}>
                                <RaisedButton icon={<FontIcon className="pe-7s-close-circle" />}
label="Cancel" onClick={event => {this.props.togglePanel("")} } />
                            </Col>
                            <Col sm={6}  style={{maxWidth: 200}}>
                                <RaisedButton icon={<FontIcon className="pe-7s-like2" />} primary={true}
onClick={this.validateStudentForm} label="Confirm"/>
                            </Col>
                        </Row>
                    </Grid>
                </div>
            </MuiThemeProvider>
        )
    }

}

export default StudentAdder;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/components/Students StudentViewer.js

100% Statements 43/43 **100%** Branches 9/9 **100%** Functions 24/24 **100%** Lines 43/43

```
    import React, {Component} from 'react';
    import {MenuItem, RaisedButton, SelectField, MuiThemeProvider} from 'material-ui';
    import Button from '../../elements/CustomButton/CustomButton';
    import {Panel} from 'react-bootstrap';
    import FontIcon from 'material-ui/FontIcon';

    import ReactTable from 'react-table';
    import constants from "../../utils/constants";
    import HttpRequest from "../../adapters/httpRequest";
    import matchSorter from "match-sorter";

    class StudentViewer extends Component {
        constructor(props) {
8x          super(props);

8x          this.state = {
                studentsSelectable: [],
                selectedStudents: [],
                openStudentSelector: false
            };

8x          this.studentMenuItems = this.studentMenuItems.bind(this);
8x          this.selectionRenderer = this.selectionRenderer.bind(this);
8x          this.toggleStudentSelector = this.toggleStudentSelector.bind(this);
8x          this.handleSubmit = this.handleSubmit.bind(this);
        }

        toggleStudentSelector() {
1x          this.setState({openStudentSelector: false, selectedStudents: []});
        }

        handleSubmit() {
2x          this.setState({
                openStudentSelector: false
            });

2x          var body = JSON.parse(JSON.stringify(this.props.teamProp));

2x          this.state.selectedStudents.forEach(function (student) {
2x              body.students.push(student);
            });

2x          var _this = this;
2x          _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
1x  '/sweng500/updateStudentsInTeam', 'POST', constants.useCredentials(), body, true).then(function (result) {
                var updatedTeam = result.body;
1x
1x              _this.props.updateTeam(updatedTeam, _this.props.viewIndex);
                _this.props.addNotification(<div><b>{body.name}</b> has been updated.</div>, 'success');
1x          }).catch(function (error) {
1x              _this.props.addNotification(<div><b>{body.name}</b> could not be updated at this time.</div>,
    'error');
                console.log(error.message);
            });
        }
2x  
        getStudentsInSchoolDistrict(schoolId) {
2x          var promise = new Promise(function (resolve, reject) {
1x  
                HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/getStudentsFromSchool/?
1x  schoolId=' + schoolId, 'GET', constants.useCredentials(), null, true).then(function (result) {
                    resolve(result.body);
                }).catch(function (error) {
                    reject(error.message);
2x              })
            });

            return promise;
2x      }

2x      clickAddStudentToTeam(team) {
1x          var _this = this;

            _this.getStudentsInSchoolDistrict(team.school.id).then(function (students) {
                _this.setState({
                    studentsSelectable: students,
1x                  openStudentSelector: true
                })
            }).catch(function (error) {
                _this.setState({
                    studentsSelectable: [],
1x                  openStudentSelector: true
                });

                console.log(error);
            });
19x     }
7x  
        renderSelectFieldIfContent() {
            if(this.state.studentsSelectable && this.state.studentsSelectable.length > 0) {
                return (
                    <SelectField
                        multiple={true}
                        autoWidth={true}
1x                      hintText="Select Students"
                        selectionRenderer={this.selectionRenderer}
                        value={this.state.selectedStudents}
                        onChange={ (event, index, values) => this.setState({selectedStudents: values})}
                    >
                        {this.studentMenuItems()}
12x                 </SelectField>
                )
            } else {
                return (
                    <div><b>No students are currently available to be assigned to a new team within this
    district</b></div>
                )
3x          }
        }
1x  
        selectionRenderer() {
1x          switch (this.state.selectedStudents.length) {
                case 0:
1x                  return '';
```

```
                 case 1:
                     return this.state.selectedStudents[0].firstName + " " +
     this.state.selectedStudents[0].lastName + " selected";
                 default:
7x                   return `${this.state.selectedStudents.length} names selected`;
7x           }
         }

         studentMenuItems() {
             return this.state.studentsSelectable.map((student) => (
                 <MenuItem
                     key={student.id}
                     insetChildren={true}
                     checked={this.state.selectedStudents.indexOf(student) > -1}
                     value={student}
                     primaryText={student.firstName + " " + student.lastName}
19x              />
             ));
         }

         render() {
             return (
                 <MuiThemeProvider>
                     <div style={{padding:'20px'}}>
                         <div style={{textAlign: 'center', fontSize: 'large'}}>
                             <em>Students in team: <b>{this.props.teamProp.name}</b></em>
                         </div>
                         <ReactTable
                             data={this.props.teamProp.students}
                             columns={this.columns2}
                             filterable
                             defaultPageSize={this.props.teamProp.students.length}
                             showPagination={false}
2x                           className="-striped -highlight"
                             defaultSorted={[{id: "firstName"}]}
                         />
                         <br />
                         <div>
                             <Button fill bsStyle="info" onClick={event =>
     {this.clickAddStudentToTeam(this.props.teamProp)}} disabled={this.state.openStudentSelector}>
                                 Add Students to <b>{this.props.teamProp.name}</b>
                             </Button>
                             <br />
                             <Panel id="student-to-team-collapsible-panel" expanded=
     {this.state.openStudentSelector} onToggle={ () => {}}
                                    style={{border: 0, width: '50%', marginLeft: '25%', marginRight: '25%'}}>
                                 <Panel.Body collapsible>
                                     {
                                         this.renderSelectFieldIfContent()
                                     }
                                     <br />
                                     <RaisedButton icon={<FontIcon className="pe-7s-close-circle" />}
     label="Cancel"
                                                   onClick={this.toggleStudentSelector}/>&nbsp;&nbsp;
                                     <RaisedButton icon={<FontIcon className="pe-7s-like2" />} primary={true}
     label="Add Students"
                                                   onClick={this.handleSubmit} disabled=
     {this.state.selectedStudents.length === 0 ? true : false}/>
                                 </Panel.Body>
                             </Panel>
```

```
1x                      </div>
                    </div>
                </MuiThemeProvider>
            )
        }

        columns2 = [
1x              {
                    Header: 'First Name',
                    filterMethod: (filter, rows) =>
                        matchSorter(rows, filter.value, {keys: ["firstName"]}),
                    filterAll: true,
                    accessor: 'firstName' // String-based value accessors!
                },
1x              {
                    Header: 'Last Name',
                    filterMethod: (filter, rows) =>
                        matchSorter(rows, filter.value, {keys: ["lastName"]}),
                    filterAll: true,
                    accessor: 'lastName' // String-based value accessors!
                },
                {
                    Header: 'Email',
                    filterMethod: (filter, rows) =>
                        matchSorter(rows, filter.value, {keys: ["emailAddress"]}),
                    filterAll: true,
                    accessor: 'emailAddress' // String-based value accessors!
                },
                {
                    Header: 'Actions',
                    accessor: 'menuActions', // String-based value accessors!
                    style:{textAlign:'center'},
                    sortable: false,
                    filterable: false
                }
        ];
    }

    export default StudentViewer;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/components/Teams TeamAdder.js

100% Statements 31/31 100% Branches 8/8 100% Functions 12/12 100% Lines 31/31

```
    import React, {Component} from 'react';
    import {MuiThemeProvider, AppBar, TextField, RaisedButton, FontIcon} from 'material-ui';
    import {Grid, Row, Col} from 'react-bootstrap';

    import CustomDropdown from "../../elements/CustomSelector/CustomDropdown";
    import Validator from "../../utils/validator";
    import constants from "../../utils/constants";
    import HttpRequest from "../../adapters/httpRequest";

    class TeamAdder extends Component {
        constructor(props) {
4x          super(props);

4x          this.state = {
                selectedSchool: undefined,
                selectedCoach: undefined,
                teamName: "",
                unallocatedStudents: [],
                errors: {}
            }

4x          this.selectedCoach = this.selectedCoach.bind(this);
4x          this.selectedSchool = this.selectedSchool.bind(this);
4x          this.validateTeamForm = this.validateTeamForm.bind(this);
        }

        selectedCoach(value) {
1x          this.setState({
                selectedCoach: value
            })
        }

        selectedSchool(value) {
1x          this.setState({
                selectedSchool: value
            })
        }

        addTeam() {
2x          var body = {};
2x          body.name = this.state.teamName;
2x          body.coach = this.state.selectedCoach;
2x          body.school = this.state.selectedSchool;

2x          var _this = this;

2x          _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/addTeam/',
1x  'POST', constants.useCredentials(), body, true).then(function (result) {
                _this.setState({
                    selectedSchool: undefined,
                    selectedCoach: undefined,
                    teamName: ""
1x              }, () => {
1x                  _this.props.addNotification(<div><b>{body.name}</b> has been created.</div>, 'success');
```

```
                     _this.props.updateTable()
                 });

1x           }).catch(function (error) {
                 _this.props.addNotification(<div><b>{body.name}</b> could not be created because: <em>
     {error.message}</em></div>, 'error');
             });
         }
4x
         validateTeamForm() {
4x           var errors = {};

4x           let result = Validator.validateTeamName(this.state.teamName);
2x
             if(!result.isValid) {
                 errors.teamNameError = result.message;
4x           }
2x
             if( !this.state.selectedSchool ){
                 errors.schoolError = "School is required"
4x           }
2x
             if( !this.state.selectedCoach ){
                 errors.coachError = "Coach is required"
4x           }
4x
2x           this.setState({errors: errors}, () => {
                 if(constants.isEmpty(this.state.errors)) {
                     this.addTeam();
                 }
             });
         }
14x
         render() {
             return (
                 <MuiThemeProvider>
                     <div style={{textAlign: 'center'}}>
                         <AppBar showMenuIconButton={false} title="Register Team" style={{zIndex: 10}}/>
                         <Grid>
                             <Row className="text-center">
                                 <Col xs={14} md={6}>
                                     <TextField
                                         name="tname"
                                         hintText="Team's Name"
2x                                       errorText={this.state.errors.teamNameError}
                                         floatingLabelText="Team Name"
                                         onChange={(event, newValue) => this.setState({teamName: newValue})}
                                         value={this.state.teamName}
                                         fullWidth={true}
                                         required={true}
                                         style={{margin: '1px', textAlign: 'left'}}
                                     />
                                 </Col>
                                 <Col xs={7} md={3}></Col>
                             </Row>
                             <Row className="show-grid">
                                 <Col xs={7} md={3}>
                                     <CustomDropdown
                                         name={"school"}
                                         labelText={"School"}
```

```
                                    hintText={"Select School"}
                                    errorText={this.state.errors.schoolError}
                                    selected={this.state.selectedSchool}
                                    endpoint={"/sweng500/getSchools"}
                                    sortKey={"schoolName"}
                                    textKeys={["schoolName"]}
                                    selectedValue={this.selectedSchool}
                                />
                            </Col>
                            <Col xs={7} md={3}>
                                <CustomDropdown
                                    name={"coach"}
                                    labelText={"Coach"}
                                    hintText={"Select Coach"}
                                    errorText={this.state.errors.coachError}
                                    selected={this.state.selectedCoach}
                                    endpoint={"/sweng500/getCoaches"}
                                    sortKey={"lastName"}
                                    textKeys={["firstName","lastName"]}
                                    selectedValue={this.selectedCoach}
                                />
                            </Col>
                        </Row>
1x                      <Row className="center-block">
                            <Col sm={6} smOffset={1} style={{maxWidth: 200}}>
                                <RaisedButton icon={<FontIcon className="pe-7s-close-circle" />}
label="Cancel" onClick={event => {this.props.togglePanel("")} } />
                            </Col>
                            <Col sm={6} style={{maxWidth: 200}}>
                                <RaisedButton icon={<FontIcon className="pe-7s-like2" />} primary={true}
onClick={this.validateTeamForm} label="Confirm"/>
                            </Col>
                        </Row>
                    </Grid>
                </div>
            </MuiThemeProvider>
        )
    }

}

export default TeamAdder;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/components/Teams TeamViewer.js

100% Statements 74/74 **100%** Branches 7/7 **100%** Functions 34/34 **100%** Lines 74/74

```
     import React, {Component} from 'react';
     import {RaisedButton, AppBar, FontIcon, FlatButton, MuiThemeProvider} from 'material-ui';
     import ReactTable from 'react-table';
     import constants from "../../utils/constants";
     import HttpRequest from "../../adapters/httpRequest";
     import matchSorter from "match-sorter";
     import StudentViewer from "../Students/StudentViewer";

     import Loader from 'react-loader'

     import {Modal} from 'react-bootstrap';

     class TeamViewer extends Component {
         constructor(props) {
 8x          super(props);

 8x          this.state = {
                 teams: [],
                 selectedTeam: null,
                 selectedStudent: null,
                 studentToRemoveFromTeam: null,
                 modal: false,
                 modalInfo: constants.getEmptyModalInfo(),
                 expanded: {},
                 isLoaded: false
             };

 8x          this.updateTeam = this.updateTeam.bind(this);
 8x          this.closeModal = this.closeModal.bind(this);
 8x          this.deleteTeamButtonClicked = this.deleteTeamButtonClicked.bind(this);
 8x          this.handleRowExpanded = this.handleRowExpanded.bind(this);
         }

         componentWillReceiveProps(nextProps) {
 2x          if(this.props.tableUpdateToggler !== nextProps.tableUpdateToggler) {
 1x              this.getTeams();
             }
         }

         componentWillMount() {
 8x          this.getTeams();
         }

         getTeams() {
10x          var _this = this;

10x          _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/getTeams',
 9x  'GET', constants.useCredentials(), null, true).then(function (result) {
                 let resultTeams = result.body;
 9x
54x              resultTeams.forEach(function (team) {
                     _this.addButtonsToTeam(team);
                 });
 9x
```

```
            _this.setState({
                isLoaded:true,
                teams: resultTeams,
                expanded: {},
                selectedTeam: null,
                selectedStudent: null,
                modal:false,
                modalInfo: constants.getEmptyModalInfo(),
            })
        }).catch(function (error) {
            _this.props.addNotification(<div>Could not retrieve teams at this time. Try again later.</div>, 'error');
            console.log(error);
        })
    }

    deleteTeam(team) {
        var _this = this;
        var removeId = team.id;

        _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + "/sweng500/removeTeam/" + removeId, "DELETE", constants.useCredentials(), null, true).then(function (result) {
            var tempTeams = _this.state.teams.filter(t => {
                return t !== team;
            });

            _this.setState({
                teams: tempTeams,
                selectedTeam: null,
                selectedStudent: null,
                modal:false,
                modalInfo: constants.getEmptyModalInfo(),
                expanded: {}
            });

            _this.props.addNotification(<div>Team <b>{team.name}</b> has been deleted.</div>, 'success');
        }).catch(function (error) {
            _this.props.addNotification(<div>Team <b>{team.name}</b> could not be deleted because: <em>{error.message}</em></div>, 'error');
        })
    }


    deleteStudent(student) {
        var _this = this;
        var studentId = student.id;

        _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + "/sweng500/deleteStudent/" + studentId, "DELETE", constants.useCredentials(), null, true).then(function (result) {
            _this.props.addNotification(<div>Student <b>{student.firstName + ' ' + student.lastName}</b> has been deleted.</div>, 'success');

            _this.getTeams();

        }).catch(function (error) {
            console.log(error);
            _this.props.addNotification(<div>Student <b>{student.firstName + ' ' + student.lastName}</b> could not be deleted because: <em>{error.message}</em></div>, 'error');
        })
```

```
2x      }

2x      removeStudentFromTeam(student, team) {
2x
1x          var tempStudents = team.students.filter(s => {
                return s !== student;
1x          });
6x
            team.students = tempStudents;

1x          var _this = this;
1x          _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
    '/sweng500/updateStudentsInTeam', 'POST', constants.useCredentials(), team, true).then(function (result) {
1x              var updatedTeam = result.body;

                var tempTeams = _this.state.teams.filter(t => {
                    return t.id !== team.id;
                });

                _this.addButtonsToTeam(updatedTeam);
                tempTeams.push(updatedTeam);

1x              _this.setState({
                    teams: tempTeams,
1x                  selectedTeam: null,
1x                  selectedStudent: null,
                    modal:false,
                    modalInfo: constants.getEmptyModalInfo(),
                    expanded: {}
                });
1x
                _this.props.addNotification(<div><b>{student.firstName + ' ' + student.lastName}</b> has been
1x  removed from <b>{team.name}</b>.</div>, 'success');
6x          }).catch(function (error) {
1x              _this.props.addNotification(<div><b>{team.name}</b> could not be updated at this time.</div>,
    'error');
1x              console.log(error.message);
            });
        }

1x      updateTeam(updatedTeam, viewIndex) {
            var tempTeams = this.state.teams;

            tempTeams.forEach(function (team) {
1x              if(team.id === updatedTeam.id) {
                    this.addButtonsToTeam(updatedTeam);

                    team.students = updatedTeam.students;
                }
1x          }, this);

            this.setState({
                teams: tempTeams
            });

            this.handleRowExpanded({}, viewIndex);
        }

2x      // Close the modal
        closeModal() {
            this.setState({
                modal: false,
                selectedTeam: null,
                selectedStudent: null,
                modalInfo: constants.getEmptyModalInfo()
            })
        }

        deleteTeamButtonClicked(team) {
            this.setState({
                selectedTeam: team,
                modal: true,
                modalInfo: {
                    title: 'Delete Team',
                    body:
2x                      <div style={{textAlign: 'center'}}>
                            <b>Are you sure you wish to delete team <u><em>{team.name}</em></u>?</b>
                            <br />
                            Note: This action cannot be undone.
                        </div>,
                    modalAction: 'DELETE'
                }
            });
        }

        deleteStudentClicked(student) {
            this.setState({
                selectedStudent: student,
                modal: true,
                modalInfo: {
                    title: 'Delete Student',
                    body:
2x                      <div style={{textAlign: 'center'}}>
                            <b>Are you sure you wish to delete student <u><em>{student.firstName + ' ' +
    student.lastName}</em></u>?</b>
                            <br />
                            Note: This action cannot be undone.
                        </div>,
                    modalAction: 'DELETESTUDENT'
                }
            })
        }

        removeStudentFromTeamButtonClicked(student, team) {
            this.setState({
                selectedStudent: student,
                selectedTeam: team,
                modal: true,
6x              modalInfo: {
                    title: 'Update Team',
2x                  body:
2x                      <div style={{textAlign: 'center'}}>
                            <b>Are you sure you wish to remove student <u><em>{student.firstName + ' ' +
2x  student.lastName}</em></u> from team <u><em>{team.name}</em></u>?</b>
2x                      </div>,
                    modalAction: 'REMOVESTUDENTFROMTEAM'
2x              }
2x          })
        }

        handleModalActionClicked(modalAction) {
            switch(modalAction) {
56x             case 'DELETE':
                    this.deleteTeam(this.state.selectedTeam);
                    break;
                case 'REMOVESTUDENTFROMTEAM':
                    this.removeStudentFromTeam(this.state.selectedStudent, this.state.selectedTeam);
56x                 break;
81x             case 'DELETESTUDENT':
                    this.deleteStudent(this.state.selectedStudent);
                    break;
            }
        }

        addButtonsToTeam(team) {
            team.menuActions =
                <div>
                    <RaisedButton style={{minWidth: '30%'}} icon={<FontIcon className="pe-7s-trash" />}
 3x secondary={true} onClick={this.deleteTeamButtonClicked.bind(this, team) } />
                </div>;

            for(let studIndex in team.students) {
                team.students[studIndex].menuActions =
                    <div >
31x                     <RaisedButton style={{minWidth: '40%'}} icon={<FontIcon className="pe-7s-less" />}
    backgroundColor="#FFC300" onClick={this.removeStudentFromTeamButtonClicked.bind(this,
    team.students[studIndex], team)} />
                        <RaisedButton style={{minWidth: '40%'}} icon={<FontIcon className="pe-7s-trash" />}
    secondary={true} onClick={this.deleteStudentClicked.bind(this, team.students[studIndex])} />
                    </div>;
            }
        }

        //Handling Row Expansion
        handleRowExpanded(newExpanded, index) {
            this.setState({
                expanded: {[index]: !this.state.expanded[index]}
            })
 1x     }

        render() {
            return (
                <MuiThemeProvider>
                    <div>
                        <Loader color="#3498db" loaded={this.state.isLoaded}>
                            <ReactTable
                                data={this.state.teams}
                                columns={this.columns}
                                expanded={this.state.expanded}
                                onExpandedChange={this.handleRowExpanded}
                                filterable
                                defaultPageSize={10}
                                className="-striped -highlight"
                                defaultSorted={[{id: "name"}]}
                                SubComponent={row => (
                                    <StudentViewer teamProp={row.original} viewIndex={row.viewIndex}
    updateTeam={this.updateTeam} updateTable={this.props.updateTable} addNotification=
    {this.props.addNotification}/>
 6x                             )}
                            />
                        </Loader>
                        <Modal show={this.state.modal} onHide={this.closeModal}>
                            <Modal.Header>
                                <Modal.Title> <AppBar
                                    iconElementRight={<FlatButton label="Close"/>}
                                    showMenuIconButton={false}
                                    onRightIconButtonClick={this.closeModal}
                                    title={this.state.modalInfo.title}
                                /></Modal.Title>
                            </Modal.Header>
1x                          <Modal.Body>
                                {this.state.modalInfo.body}
                            </Modal.Body>
                            <Modal.Footer>
                                <RaisedButton icon={<FontIcon className="pe-7s-close-circle" />}
    label="Cancel"
                                          onClick={this.closeModal}/>&nbsp;&nbsp;
1x                              <RaisedButton icon={<FontIcon className="pe-7s-like2" />} primary={true}
    label="Confirm"
                                          onClick={ () =>
    this.handleModalActionClicked(this.state.modalInfo.modalAction)}/>
                            </Modal.Footer>
                        </Modal>
                    </div>
1x              </MuiThemeProvider>
            )
        }

        columns = [
            {
                Header: 'Team Name',
                filterMethod: (filter, rows) =>
                    matchSorter(rows, filter.value, {keys: ["name"]}),
                filterAll: true,
                accessor: 'name' // String-based value accessors!
            }, {
                Header: 'Coach',
                id: 'coachName',
                filterMethod: (filter, rows) =>
                    matchSorter(rows, filter.value, {keys: ["coachName"]}),
                filterAll: true,
                accessor: 'coach.name' // String-based value accessors!
            }, {
                Header: 'School',
                id: 'schoolName',
                filterMethod: (filter, rows) =>
                    matchSorter(rows, filter.value, {keys: ["schoolName"]}),
                filterAll: true,
                accessor: 'school.schoolName' // String-based value accessors!
            },
            {
                Header: 'Actions',
                accessor: 'menuActions', // String-based value accessors!
                style:{textAlign:'center'},
                sortable: false,
                filterable: false
            }
        ];

    }
```

```
export default TeamViewer;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/containers/App App.js

96.77% Statements 30/31 **95%** Branches 19/20 **100%** Functions 7/7 **96.77%** Lines 30/31

```
    import React, { Component } from 'react';
    import NotificationSystem from 'react-notification-system';
    import {
        Switch,
        Redirect
    } from 'react-router-dom';

    import {style} from "../../variables/Variables.js";
    import AuthenticatedRoute from  '../../routes/AuthenticatedRoute';
    import Header from '../../components/Header/Header';
    import Footer from '../../components/Footer/Footer';
    import Sidebar from '../../components/Sidebar/Sidebar';
    import appRoutes from '../../routes/app.js';


    class App extends Component {
        constructor(props){
2x          super(props);
2x          this.componentDidMount = this.componentDidMount.bind(this);
2x          this.notify = this.notify.bind(this);
2x          this.state = {
                _notificationSystem: null
            };
        }

        notify(message, level, position, autoDismiss, optionalTitle){
2x          this.state._notificationSystem.addNotification({
                title: optionalTitle ? optionalTitle : (<span data-notify="icon" className="pe-7s-gift">
    </span>),
                message: (
                    <div>
                        {message}
                    </div>
                ),
                level: level ? level : 'info',
                position: position ? position : 'tc',
                autoDismiss: autoDismiss ? autoDismiss : 10,
            });
        }

6x      static getRandoNotification() {
6x          var notifyJson = {};
6x          notifyJson.position = 'tr';
            notifyJson.autoDismiss = 10;
6x
6x          let color = Math.floor((Math.random() * 4) + 1);
            switch (color) {
1x              case 1:
1x                  notifyJson.level = 'success';
                    break;
1x              case 2:
1x                  notifyJson.level = 'warning';
                    break;
1x              case 3:
 1x                  notifyJson.level = 'error';
                     break;
 2x              case 4:
 2x                  notifyJson.level = 'info';
                     break;
 1x              default:
 1x                  notifyJson.level = 'info';
                     break;
             }
 6x
             return notifyJson;
         }

 1x      componentDidMount(){
 1x          this.setState({_notificationSystem: this.refs.notificationSystem});
             var _notificationSystem = this.refs.notificationSystem;
 1x
             let notifyJson = App.getRandoNotification();
 1x
             _notificationSystem.addNotification({
                 title: (<span data-notify="icon" className="pe-7s-gift"></span>),
                 message: (
                     <div>
                         Welcome to <b>Science Olympiad Dashboard</b> - a central hub for Science Olympiad Competitions.
                     </div>
                 ),
                 level: notifyJson.level,
                 position: notifyJson.position,
                 autoDismiss: notifyJson.autoDismiss,
             });
 2x      }
         componentDidUpdate(e){
             I if(window.innerWidth < 993 && e.history.location.pathname !== e.location.pathname && document.documentElement.className.indexOf('nav-open') !== -1){
                 document.documentElement.classList.toggle('nav-open');
 3x          }
         }
         render() {
             return (

                 <div className="wrapper">
                     <NotificationSystem ref="notificationSystem" style={style}/>
                     <Sidebar {...this.props} />
                     <div id="main-panel" className="main-panel">
                         <Header {...this.props}/>
30x                          <Switch>
 3x                              {
                                     appRoutes.map((prop,key) => {
                                         if(prop.redirect)
                                             return (
27x                                              <Redirect from={prop.path} to={prop.to} key={key}/>
                                             );

                                         return (
                                             <AuthenticatedRoute path={prop.path} component={prop.component} users={prop.users} key={key} notify={this.notify}/>
                                         );
                                     })
                                 }
```

```
                        </Switch>
                    <Footer />
                </div>
            </div>
        );
    }
}

export default App;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/containers/Login LoginContainer.js

100% Statements 29/29 100% Branches 13/13 100% Functions 10/10 100% Lines 29/29

```
    import React, {Component} from 'react';
    import Login from "../../components/Login/Login";
    import Signup from "../../components/Login/Signup";
    import Forgot from "../../components/Login/Forgot";
    import $ from 'jquery';
    import NotificationSystem from 'react-notification-system';

    import mountainBackground from '../../assets/img/SmallerMountain.jpg';

    import {style} from "../../variables/Variables";
1x  const backgroundStyle = {
        background: 'url(' + mountainBackground + ') no-repeat center center fixed',
        backgroundSize: 'cover',
        backgroundPosition: 'center center no-repeat'
    };

    class LoginContainer extends Component {

        constructor(props) {
3x          super(props);

3x          this.state = {
                view: "login",
                _notificationSystem: null
            };

3x          this.addNotification = this.addNotification.bind(this);
        }

        toggleLoginView(event) {
5x          event.preventDefault();

5x          var value = event.target.attributes.getNamedItem('data-type').value;

5x          switch (value) {
                case 'register':
1x                  $('#login-container-slider').addClass('collapse');
1x                  $('#signup-container-slider').removeClass('collapse');
1x                  break;
                case 'forgot':
1x                  $('#login-container-slider').addClass('collapse');
1x                  $('#forgot-container-slider').removeClass('collapse');
1x                  break;
                case 'slogin':
1x                  $('#login-container-slider').removeClass('collapse');
1x                  $('#signup-container-slider').addClass('collapse');
1x                  break;
                case 'flogin':
1x                  $('#login-container-slider').removeClass('collapse');
1x                  $('#forgot-container-slider').addClass('collapse');
1x                  break;
                default :
1x                  break;
            }
        }

        addNotification(message, level, position, autoDismiss) {
6x          if(this.state._notificationSystem) {
2x              this.state._notificationSystem.addNotification({
                    title: (<span data-notify="icon" className="pe-7s-door-lock"></span>),
                    message: (
                        <div>
                            {message}
                        </div>
                    ),
                    level: level ? level : 'error',
                    position: position ? position : 'tc',
                    autoDismiss: autoDismiss ? autoDismiss : 10,
                });
            }
        }

        componentDidMount() {
4x          this.setState({_notificationSystem: this.refs.notificationSystem}, () => {
4x              this.addNotification(
                    <span>Please <b>Login</b> to the Science Olympiad System.</span>,
                    'info',
                    'tc',
                    10
                )
            })
        }

        render() {

9x          return (
                <div id='login-container' className="animated fadeIn login-center"  style={backgroundStyle}>
                    <NotificationSystem ref="notificationSystem" style={style}/>
                    <div id='login-container-card' className="login-card row">
                        <div id='login-container-slider' className="animated fadeIn">
                            <Login notify={this.addNotification}/>
1x                          <a data-type={'forgot'} onClick={event => this.toggleLoginView(event)}
    className="col-sm-10">Forgot your password?</a>
1x                          <br/>
                            <a data-type={'register'} onClick={event => this.toggleLoginView(event)}
    className="col-sm-10">Register for a new account</a>
                            <br/>

                        </div>
                        <div id='signup-container-slider' className="animated collapse fadeIn">
2x                          <Signup notify={this.addNotification}/>
                            <br/>
                            <a data-type={'slogin'} onClick={event => this.toggleLoginView(event)}
    className="col-sm-9">Return to the login screen</a>
                        </div>
1x                      <div id='forgot-container-slider' className="animated collapse fadeIn">
                            <Forgot notify={this.addNotification}/>
                            <br/>
                            <a data-type={'flogin'} onClick={event => this.toggleLoginView(event)}
    className="col-sm-9">Return to the login screen</a>
                        </div>
                    </div>
                </div>
            )
        }

    }

    export default LoginContainer;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/elements/CustomButton CustomButton.js

100% Statements 4/4 **100%** Branches 0/0 **100%** Functions 1/1 **100%** Lines 4/4

```
    import React, { Component } from 'react';
    import { Button } from 'react-bootstrap';
    import cx from 'classnames';
    import PropTypes from 'prop-types';
 
    class CustomButton extends Component {
        render() {
2x          const { fill, simple, pullRight, round, block, ...rest } = this.props;
 
2x          const btnClasses = cx({
                'btn-fill': fill,
                'btn-simple': simple,
                'pull-right': pullRight,
                'btn-block': block,
                'btn-round': round
            });
 
2x          return (
                <Button
                    className={btnClasses}
                    {...rest}
                />
            );
      }
    }
 
10x CustomButton.propTypes = {
        fill: PropTypes.bool,
        simple: PropTypes.bool,
        pullRight: PropTypes.bool,
        block: PropTypes.bool,
        round: PropTypes.bool
    }
 
    export default CustomButton;
 
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/elements/CustomSelector CustomDropdown.js

100% Statements 26/26 **78.57%** Branches 11/14 **100%** Functions 12/12 **100%** Lines 24/24

```
    import React, {Component} from 'react';
    import SelectField from 'material-ui/SelectField';
    import MenuItem from 'material-ui/MenuItem';
    import HttpRequest from '../../adapters/httpRequest';
    import constants from '../../utils/constants';

    class CustomDropdown extends Component {

        constructor(props) {
4x          super(props);

4x          this.state = {
                errorText: this.props.errorText,
                list: [],
            }

4x          this.updateValues = this.updateValues.bind(this);

            //INCOMING PROPS:
            //selected - "the item selected"
            //endpoint - "/sweng500/getCoaches"
            //sortKey - "lastName"
            //hintText - "the label before clicking"
            //labelText - "the label"
            //textKeys - ["firstName", "lastName"]
        }

        sortByKey(array, key) {
3x          if(key === undefined) return array;

2x          return array.sort(function(a, b) {
2x              var x = a[key]; var y = b[key];
2x              return ((x < y) ? -1 : ((x > y) ? 1 : 0));
            });
        }

        buildLabel(listItem) {
8x          var label = "";
8x          if(this.props.textKeys && this.props.textKeys.length > 0) {
6x              this.props.textKeys.forEach(function (key) {
6x                  label += (listItem[key] + " ");
                });

6x              return label.trim();
            }

2x          return " - ";
        }

        componentDidMount() {
4x          var _this = this;

4x          _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + this.props.endpoint,
3x  'GET', constants.useCredentials(), null, true).then(function (result) {
```

```
                _this.setState({list: _this.sortByKey(result.body, _this.props.sortKey)})
1x          }).catch(function (error) {
                _this.setState({
                    errorText: "Failed to get data"
                });
            })
        }

2x      componentWillReceiveProps(nextProps) {
1x          if(this.state.errorText !== nextProps.errorText)
                this.setState({errorText: nextProps.errorText})
        }

1x      updateValues(event, index, value) {
            this.props.selectedValue(value);
        }

9x      render() {
4x          if (this.state.list.length !== 0) {
                return (
                    <SelectField
                        name={this.props.name}
                        hintText={this.props.hintText}
                        errorText={this.state.errorText}
                        floatingLabelText={this.props.labelText}
                        onChange={this.updateValues}
                        value={this.props.selected}
                        maxHeight={200}
                        fullWidth={true}
                        style={{textAlign: 'left'}}>
                        {
8x                          Object.keys(this.state.list).map(function (key) {
                                return (
                                    <MenuItem key={this.state.list[key].id}
                                              primaryText={this.buildLabel(this.state.list[key])}
                                              value={this.state.list[key]}/>
                                )
                            }, this)

                        }
                    </SelectField>
                )
            }
5x          else
                return <span></span>
        }
    }

    export default CustomDropdown;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/routes AuthenticatedRoute.js

100% Statements 3/3 **100%** Branches 4/4 **100%** Functions 2/2 **100%** Lines 3/3

```
    import React from 'react';
    import {
        Route,
        Redirect
    } from 'react-router-dom';

    import AuthService from "../utils/AuthService";

    /**
     * We have to make our own Authenticated Route.... So here it is -_-
     * @param Component
     * @param rest
     * @returns {*}
     * @constructor
     */
2x  const AuthenticatedRoute = ({ component: Component, ...rest }) => (
3x      <Route {...rest} render={props => (
3x          AuthService.isLoggedIn() && AuthService.isUserRoleAllowed(rest.users)? (
                <Component {...props}/>
            ) : (
                <Redirect to={{
                    pathname: '/login',
                    state: { from: props.location }
                }}/>
            )
        )}/>
    )

    export default AuthenticatedRoute;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/routes app.js

100% Statements `1/1` **100%** Branches `0/0` **100%** Functions `0/0` **100%** Lines `1/1`

```
   import Dashboard from '../views/Dashboard/Dashboard';
   import UserProfile from '../views/UserProfile/UserProfile';
   import Events from '../views/Events/Events'
   import Schools from '../views/Schools/Schools'
   import Buildings from '../views/Locations/Buildings'
   import Rooms from '../views/Locations/Rooms'
   import TeamManagement from "../views/Teams/TeamManagement";
   import Scoring from "../views/Scoring/Scoring";
   import Extras from "../views/Extras/Extras";

4x const appRoutes = [
       { path: "/app/dashboard", name: "Dashboard", icon: "pe-7s-graph", component: Dashboard },
       { path: "/app/events", name:"Events", icon:"pe-7s-global", component :Events},
       { path: "/app/schools", name:"Schools", icon:"pe-7s-home", component :Schools, users:['ADMIN']},
       { path: "/app/buildings", name:"Buildings", icon:"pe-7s-map", component: Buildings, users:['ADMIN']},
       { path: "/app/rooms", name:"Rooms", icon:"pe-7s-ribbon", component: Rooms, users:['ADMIN']},
       { path: "/app/teams", name: "Team Management", icon: "pe-7s-users", component : TeamManagement, users:
   ['ADMIN', 'COACH']},
       { path: "/app/scoring", name: "Event Scores", icon: "pe-7s-medal", component : Scoring },
       { path: "/app/extras", name:"Extras", icon:"pe-7s-gleam", component :Extras, users:['ADMIN']},
       { linkOnly: true, path: "/app/user", name:"User Profile", icon:"pe-7s-global", component :UserProfile},
       { redirect: true, path:"/", to:"/app/dashboard", name: "Dashboard" }
   ];

   export default appRoutes;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/utils AuthService.js

100% Statements 69/69 **100%** Branches 32/32 **100%** Functions 22/22 **100%** Lines 69/69

```
    import jwt from 'jwt-simple';
    import moment from 'moment';

    import constants from './constants';
    import HttpRequest from '../adapters/httpRequest';

    export default class AuthService {

        static login(username, password) {
2x          var promise = new Promise(function (resolve, reject) {
2x              var body = {};
2x              body.emailAddress = username;
2x              body.password = password;

                //var _this = this;
2x              HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/auth/login', 'POST',
1x  constants.useCredentials() , body ).then(function (result) {
                    let toEncode = {
                        emailAddress: result.body.emailAddress,
                        role: result.body.role,
                        expiration: moment().add(20, 'minutes').format('YYYY-MM-DDTHH:mm:ss.SSS')
                    }
1x
                    var encoded = jwt.encode(toEncode, result.body.session);
1x
1x                  AuthService.setSession(result.body.session);
                    AuthService.setToken(encoded);
1x
                    resolve({status: result.status, message: "Authorized", emailAddress:
1x  result.body.emailAddress});
1x              }).catch(function (error) {
                    console.log(error);
                    reject(error);
                })
2x          });

            return promise;
        }
3x
3x      static logout() {
            var promise = new Promise(function (resolve, reject) {
3x              var body = {};
3x
3x              const token = AuthService.getToken();
                const session = AuthService.getSession();
2x              if(token && session) {

2x                  body.emailAddress = jwt.decode(token, session).emailAddress;
1x
1x                  HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/auth/logout', 'POST',
    constants.useCredentials(), body ).then(function (result) {
1x                      console.log(result);
                        AuthService.revokeAuth();
1x
1x                      resolve({status: result.status, message: result.body});
1x                  }).catch(function (error) {
                        AuthService.revokeAuth();
                        console.log(error);
1x                      resolve({status: error.status, message: error.message});
1x                  })
                } else {
                    AuthService.revokeAuth();
                    resolve({status: 410, message: "Token is expired."});
3x              }
            });

            return promise;
1x      }

        static isLoggedIn() {
            return AuthService.isAuthorized();
4x      }
4x
4x      static isAuthorized(isServerCall) {
3x          const token = AuthService.getToken();
3x          const session = AuthService.getSession();
1x          if (token && session) {
1x              var decoded = AuthService.decodeSessionVars();
                if (!decoded || moment(decoded.expiration).isBefore(moment())) { // Checking if token is expired.
2x                  AuthService.revokeAuth(true);
1x                  return false;
                }
                else {
2x                  if(isServerCall) {
                        AuthService.updateTimeStamp(decoded);
                    }
1x
1x                  return true;
                }
            } else {
                AuthService.revokeAuth();
                return false;
3x          }
2x      }

        static isUserRoleAllowed(listOfAllowedRoles) {
1x          if(listOfAllowedRoles && listOfAllowedRoles.length > 0){
                return listOfAllowedRoles.indexOf(AuthService.getUserRole()) > -1;
            }

7x          return true;
7x      }

7x      static revokeAuth(withAlert) {
            AuthService.revokeToken();
2x          AuthService.endSession();

            if(withAlert) {
                //DO SOMETHING BETTER THAN THIS
                alert("Your session has ended.  Please log back in to continue");
4x          }
4x      }
```

```
         static getUserRole() {
             let decoded = AuthService.decodeSessionVars();
 2x          return (decoded && decoded.role) ? decoded.role : undefined;
 2x      }

         static getUserEmail() {
             let decoded = AuthService.decodeSessionVars();
11x          return (decoded && decoded.emailAddress) ? decoded.emailAddress : undefined;
11x      }

11x      static decodeSessionVars() {
 8x          const session = AuthService.getSession();
 8x          const token = AuthService.getToken();

 3x          if(token && session) {
                 var sessionInfo = jwt.decode(token, session);
                 return sessionInfo;
             } else {
                 return undefined;
 2x          }
         }
 2x
 2x      static updateTimeStamp(decoded) {
             decoded.expiration = moment().add(20, 'minutes').format('YYYY-MM-DDTHH:mm:ss.SSS');
 2x
             let encoded = jwt.encode(decoded, AuthService.getSession());
             AuthService.setToken(encoded);

11x          return true;
         }

         static setSession(session) {
23x          localStorage.setItem('blob', session);
         }

         static getSession() {
 8x          return localStorage.getItem('blob');
         }

         static endSession() {
22x          localStorage.removeItem('blob');
         }

         static getToken() {
13x          return localStorage.getItem('id_token');
         }

         static setToken(idToken) {
 8x          localStorage.setItem('id_token', idToken);
         }

         static revokeToken() {
             localStorage.removeItem('id_token');
         }

     }
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/utils constants.js

100% Statements 10/10 **100%** Branches 6/6 **100%** Functions 4/4 **100%** Lines 10/10

```
      
 29x  module.exports.getServerUrl = function () {
125x      if(process.env.NODE_ENV && process.env.NODE_ENV.toLowerCase() === 'development') {
  1x          return 'http://localhost:8080';
          } else {
124x          return window.location.protocol + '//server.sweng500.com';
          }
      };
      
 29x  module.exports.useCredentials = function () {
176x      return {withCredentials: true};
      };
      
 29x  module.exports.getEmptyModalInfo = function () {
 20x      return {
              title: '',
              body: null,
              modalAction: ''
          }
      };
      
 29x  module.exports.isEmpty = function (obj) {
 12x      return Object.keys(obj).length === 0 && obj.constructor === Object;
      };
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/utils validator.js

100% Statements 25/25 **100%** Branches 12/12 **100%** Functions 2/2 **100%** Lines 25/25

```
     
9x   module.exports.validateTeamName = function (teamName) {
8x       var result = {isValid: false};
     
8x       if(!teamName) {
3x           result.message = "Team Name is required.";
3x           return result;
         }
     
5x       if(teamName.trim().length < 5) {
1x           result.message = "Team Name is too short.";
1x           return result;
         }
     
4x       let reg = /^[a-z0-9|\s]+$/i;
4x       if(!reg.test(teamName.trim())){
1x           result.message = "May only include Alphanumeric";
1x           return result;
         }
     
3x       result.isValid = true;
3x       return result;
     };
     
     // Checks to see if an email has a host, @ symbols, and domain.
9x   module.exports.validateEmail = function(text) {
11x      var result = {isValid: false};
     
11x      if(!text || text.trim() === "") {
5x           result.message = "Email address is required";
5x           return result;
         }
     
6x       let reg = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
6x       if(!reg.test(text)) {
2x           result.message = "Must be proper Email address";
2x           return result;
         }
     
4x       result.isValid = true;
4x       return result;
     };
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/views/Dashboard CoachCount.js

78.57% Statements 11/14 **50%** Branches 1/2 **71.43%** Functions 5/7 **84.62%** Lines 11/13

```
     import React, { Component } from 'react';
     import PropTypes from 'prop-types';
     import HttpRequest from "../../adapters/httpRequest";
     import constants from "../../utils/constants";
     import {StatsCard} from '../../components/Cards/StatsCard.js';
     
     class CoachCount extends Component {
     
         constructor(props) {
1x           super(props);
     
1x           this.state = {
                 result: []
             };
     
1x           this.getCoaches();
     
1x           this.getCoaches = this.getCoaches.bind(this);
         }
     
         getCoaches(event) {
1x           I if(event) {event.preventDefault() };
     
1x           var _this = this;
1x           _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + "/sweng500/getCoaches",
1x   "get", constants.useCredentials(), null, true).then(function (result) {
                 _this.setState({
                     result: result.body
1x               })
                 console.log(result.body);
             }).catch(function (error) {
                 console.log(error);
             })
         }
     
         formatBody() {
2x   
             return (
                 <div>
                     {this.state.result.length}
                     <div className="footer">
                         <hr />
                         <div className="stats">
                             <a style={{cursor:'pointer'}} onClick={event => this.getCoaches(event)}><i
     className="fa fa-refresh"></i> Update Now</a>
                         </div>
                     </div>
                 </div>
             );
         }
     
2x       render() {
     
             return (
                 <StatsCard
                     bigIcon={<i className="pe-7s-graph1 text-warning"></i>}
                     statsText="Number of Coaches"
                     statsValue={this.formatBody()}
                 />

             );
         }

     }

     export default CoachCount;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/views/Dashboard Dashboard.js

37.5% Statements 3/8 **16.67%** Branches 1/6 **100%** Functions 1/1 **37.5%** Lines 3/8

```
    import React, { Component } from 'react';
    import { Grid, Row, Col } from 'react-bootstrap';
    import ChuckParent from '../../components/HttpExample/ChuckParent.js'
    import TodaysEvents from '../../views/Dashboard/TodaysEvents.js'
    import EventCount from "./EventCount";
    import TeamCount from "./TeamCount";
    import CoachCount from "./CoachCount";
    import JudgeCount from "./JudgeCount";
    import AuthService from "../../utils/AuthService";

    class Dashboard extends Component {

        render() {

2x          const userType = AuthService.getUserRole();

2x          E if (userType === "ADMIN") {

2x              return (
                    <div className="content">
                        <Grid fluid>
                            <Row>
                                <Col lg={3} sm={6}>
                                    <EventCount/>
                                </Col>
                                <Col lg={3} sm={6}>
                                    <TeamCount/>
                                </Col>
                                <Col lg={3} sm={6}>
                                    <CoachCount/>
                                </Col>
                                <Col lg={3} sm={6}>
                                    <JudgeCount/>
                                </Col>
                            </Row>
                            <Row>
                                <Col lg={3} sm={6}>
                                    <TodaysEvents/>
                                </Col>
                                <Col lg={3} sm={6}>
                                    <ChuckParent/>
                                </Col>
                            </Row>
                        </Grid>
                    </div>
                );
            }
            else if (userType == "COACH") {

                return (
                <div className="content">
                    <Grid fluid>
                        <Row>
                            <Col lg={3} sm={6}>
                                <EventCount/>
                            </Col>
                            <Col lg={3} sm={6}>
                                <TeamCount/>
                            </Col>
                        </Row>
                        <Row>
                            <Col lg={3} sm={6}>
                                <TodaysEvents/>
                            </Col>
                        </Row>
                    </Grid>
                </div>
                );
            }
            else if (userType == "JUDGE") {
                return (
                    <div className="content">
                        <Grid fluid>
                            <Row>
                                <Col lg={3} sm={6}>
                                    <EventCount/>
                                </Col>
                                <Col lg={3} sm={6}>
                                    <TeamCount/>
                                </Col>
                                <Col lg={3} sm={6}>
                                    <CoachCount/>
                                </Col>
                            </Row>
                            <Row>
                                <Col lg={3} sm={6}>
                                    <TodaysEvents/>
                                </Col>
                            </Row>
                        </Grid>
                    </div>
                );
            }
            else
            {
                return (
                    <div className="content">
                        <Grid fluid>
                            <Row>
                                <Col lg={3} sm={6}>
                                    <TodaysEvents/>
                                </Col>
                            </Row>
                        </Grid>
                    </div>
                );
            }

        }
    }

    export default Dashboard;

```

All files / src/views/Dashboard EventCount.js

75% Statements 12/16 **50%** Branches 1/2 **62.5%** Functions 5/8 **85.71%** Lines 12/14

```
    import React, { Component } from 'react';
    import PropTypes from 'prop-types';
    import HttpRequest from "../../adapters/httpRequest";
    import constants from "../../utils/constants";
    import {StatsCard} from '../../components/Cards/StatsCard.js';

    class EventCount extends Component {

        constructor(props) {
1x          super(props);

1x          this.state = {
                result: []
            };

1x          this.getEvents();

1x          this.getEvents = this.getEvents.bind(this);
        }

        getEvents(event) {
1x          I if(event) {event.preventDefault() };

1x          var _this = this;
1x          _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + "/sweng500/events", "get",
1x  constants.useCredentials(), null, true).then(function (result) {
                _this.setState({
                    result: result.body
1x              })
                console.log(result.body);
            }).catch(function (error) {
                console.log(error);
            })
        }

        formatBody() {
2x

            return (
                <div>
                    {this.state.result.length}
                    <div className="footer">
                        <hr />
                        <div className="stats">
                            <a style={{cursor:'pointer'}} onClick={event => this.getEvents(event)}><i
    className="fa fa-refresh"></i> Update Now</a>
                        </div>
                    </div>
                </div>
            );
        }

2x      render() {

2x          var refresh = <a style={{cursor:'pointer'}} onClick={event => this.props.getEvents(event)}><i
    className="fa fa-refresh"></i> Update Now</a>

            return (
                <StatsCard
                    bigIcon={<i className="pe-7s-graph1 text-warning"></i>}
                    statsText="Number of Events"
                    statsValue={this.formatBody()}
                />

            );
        }

    }

    export default EventCount;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/views/Dashboard JudgeCount.js

78.57% Statements 11/14 50% Branches 1/2 71.43% Functions 5/7 84.62% Lines 11/13

```
    import React, { Component } from 'react';
    import PropTypes from 'prop-types';
    import HttpRequest from "../../adapters/httpRequest";
    import constants from "../../utils/constants";
    import {StatsCard} from '../../components/Cards/StatsCard.js';

    class CoachCount extends Component {

        constructor(props) {
1x          super(props);

1x          this.state = {
                result: []
            };

1x          this.judgeCount();

1x          this.judgeCount = this.judgeCount.bind(this);
        }

        judgeCount(event) {
1x          I if(event) {event.preventDefault() };

1x          var _this = this;
1x          _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + "/sweng500/getJudges",
1x  "get", constants.useCredentials(), null, true).then(function (result) {
                _this.setState({
                    result: result.body
1x              })
                console.log(result.body);
            }).catch(function (error) {
                console.log(error);
            })
        }

        formatBody() {
2x
            return (
                <div>
                    {this.state.result.length}
                    <div className="footer">
                        <hr />
                        <div className="stats">
                            <a style={{cursor:'pointer'}} onClick={event => this.judgeCount(event)}><i
    className="fa fa-refresh"></i> Update Now</a>
                        </div>
                    </div>
                </div>
            );
        }

2x      render() {

            return (
                <StatsCard
                    bigIcon={<i className="pe-7s-graph1 text-warning"></i>}
                    statsText="Number of Judges"
                    statsValue={this.formatBody()}
                />

            );
        }

    }

    export default CoachCount;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/views/Dashboard TeamCount.js

78.57% Statements 11/14 **50%** Branches 1/2 **71.43%** Functions 5/7 **84.62%** Lines 11/13

```
    import React, { Component } from 'react';
    import PropTypes from 'prop-types';
    import HttpRequest from "../../adapters/httpRequest";
    import constants from "../../utils/constants";
    import {StatsCard} from '../../components/Cards/StatsCard.js';

    class TeamCount extends Component {

        constructor(props) {
1x          super(props);

1x          this.state = {
                result: []
            };

1x          this.getTeams();

1x          this.getTeams = this.getTeams.bind(this);
        }

        getTeams(event) {
1x          I if(event) {event.preventDefault() };

1x          var _this = this;
1x          _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + "/sweng500/getTeams",
1x  "get", constants.useCredentials(), null, true).then(function (result) {
                _this.setState({
                    result: result.body
1x              })
                console.log(result.body);
            }).catch(function (error) {
                console.log(error);
            })
        }

        formatBody() {
2x
            return (
                <div>
                    {this.state.result.length}
                    <div className="footer">
                        <hr />
                        <div className="stats">
                            <a style={{cursor:'pointer'}} onClick={event => this.getTeams(event)}><i
    className="fa fa-refresh"></i> Update Now</a>
                        </div>
                    </div>
                </div>
            );
        }

2x      render() {

            return (
                <StatsCard
                    bigIcon={<i className="pe-7s-graph1 text-warning"></i>}
                    statsText="Number of Teams"
                    statsValue={this.formatBody()}
                />
            );
        }

    }

    export default TeamCount;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/views/Dashboard TodaysEvents.js

75% Statements 12/16 **50%** Branches 1/2 **75%** Functions 6/8 **80%** Lines 12/15

```
    import React, { Component } from 'react';
    import PropTypes from 'prop-types';
    import HttpRequest from "../../adapters/httpRequest";
    import constants from "../../utils/constants";
    import {StatsCard} from '../../components/Cards/StatsCard.js';

    class TodaysEvents extends Component {

        constructor(props) {
1x          super(props);

1x          this.state = {
                result: []
            };

1x          this.getEvents();

1x          this.getEvents = this.getEvents.bind(this);
        }

        getEvents(event) {
1x          I if(event) {event.preventDefault() };

1x          var _this = this;
1x          _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + "/sweng500/events", "get",
1x  constants.useCredentials(), null, true).then(function (result) {
                _this.setState({
                    result: result.body
                })
                console.log(result.body);
1x          }).catch(function (error) {
                console.log(error);
            })
        }

        formatBody(){
2x
            var namesList = this.state.result.map(function(user, i) {
                    return <li key={i}>{user.name}</li>
                }
            )
1x
            return (
            <div id="contact-chuck-container-div" key="contact-chuck-container-div"
    className="chuckNorrisClass">
                <ol> {namesList} </ol>
                <div className="footer">
                    <hr />
                    <div className="stats">
                        <a style={{cursor:'pointer'}} onClick={event => this.getEvents(event)}><i className="fa
    fa-refresh"></i> Update Now</a>
                    </div>
                </div>
            </div>
            );
        }

        render() {
    2x



            return (
                <StatsCard
                    bigIcon={<i className="pe-7s-global text-warning"></i>}
                    statsText="Upcoming Events"
                    statsValue={this.formatBody()}
                />

            );
        }

    }

    export default TodaysEvents;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/views/EasterEgg sudoku.js

100% Statements `1/1` **100%** Branches `0/0` **100%** Functions `1/1` **100%** Lines `1/1`

```
    import React, {Component} from 'react';
    
    import Sudoku from 'sudoku-react-component';
    
    class EasterEgg extends Component {
    
        render() {
1x          return (
                <div id='sudoku-container'>
                    <Sudoku />
                </div>
            )
        }
    
    }
    
    export default EasterEgg;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/views/Events EventDetail.js

41.24% Statements 40/97 **26.67%** Branches 8/30 **25.64%** Functions 10/39 **41.24%** Lines 40/97

```
    import React, {Component} from 'react';
    import { Modal, PageHeader, Panel, Grid, Col, Row,} from 'react-bootstrap';
    import HttpRequest from "../../adapters/httpRequest";
    import constants from "../../utils/constants";
    import MuiThemeProvider from 'material-ui/styles/MuiThemeProvider';
    import RaisedButton from 'material-ui/RaisedButton';
    import "react-table/react-table.css";
    import {Map, Marker, GoogleApiWrapper} from 'google-maps-react';
    import Loader from 'react-loader'
    import Card from '../../components/Cards/Card.js';
    import Divider from 'material-ui/Divider';
    import AppBar from 'material-ui/AppBar'
    import FlatButton from 'material-ui/FlatButton';
    import FontIcon from 'material-ui/FontIcon';
    import Dialog from 'material-ui/Dialog';



    import {
        Table,
        TableBody,
        TableHeader,
        TableHeaderColumn,
        TableRow,
        TableRowColumn,
    } from 'material-ui/Table';
    import CustomDropdown from "../../elements/CustomSelector/CustomDropdown";
    import AuthService from "../../utils/AuthService";


    class EventDetail extends Component {
        constructor(props) {
1x          super(props);
1x          this.formatTimeString = this.formatTimeString.bind(this);
1x          this.selectedTeam = this.selectedTeam.bind(this);
1x          this.closeTeamModal = this.closeTeamModal.bind(this);
1x          this.removeJudge = this.removeJudge.bind(this);
1x          this.removeTeam = this.removeTeam.bind(this);
            //for the map
1x          this.divStyle = {
                height: '300px',
                width: '450px',
                position: 'relative',
                overflow: 'scroll'
            };

1x          this.state = {
                loading: false,

                //needed so initial backend call will not have blank eventid
                eventId: props.eventId,
                latitude: '',
                longitude: '',
                eventDetail: {},
                eventDate: '',
                startTime: '',
                endTime: '',
                buildingName: '',
                judgesDetail: {},

                //teams
                teamModal: false,
                //custom dropdown returns an object
                selectedTeamValue: null,
                teamSelectorError: '',
                teamsDetail: {},
                loadAddTeam: true,
                //delete
                showDeleteBtn:true,
                deleteTeam:false,
                deleteJudge:false,
                deleteId:'',
                confirmDialog:false,
                confirmMessage:''
            };

        }

        // If there is a latitude and longitude then display it
        addMarker = () => {
            if (this.state.latitude !== undefined)
                return (
                    <Marker name={'Current location'} position={{lat: this.state.latitude, lng:
    this.state.longitude}}/>);
        }

        //This component gets called initially with main events page, need to update when this is called with a
2x  no prop
2x      componentWillReceiveProps(nextProps) {
            var _this = this;
            _this.setState({
                eventId: nextProps.eventId
            });
        }
1x
        componentDidMount() {
1x          var _this = this;
            //hide things from non admins
1x          let modifyRoles = ['ADMIN','COACH'];
1x          //eventually only allow an admin to delete judges and teams
1x          let deleteRole = ['ADMIN'];
1x          let allowModify = AuthService.isUserRoleAllowed(modifyRoles);
            let allowDelete = AuthService.isUserRoleAllowed(deleteRole);
                _this.setState({
                    showRegisterBtn: allowModify,
                    showDeleteBtn:allowDelete
                });
1x
1x
1x          _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + "/sweng500/event/" +
1x  this.state.eventId, "get", constants.useCredentials(), null, true).then(function (result) {
                const start = _this.formatTimeString(new Date(result.body.startTime));
                const endT = _this.formatTimeString(new Date(result.body.endTime));
                const building = result.body.room.buildingID;
                _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + 
    "/sweng500/getBuilding/" + building, "get", constants.useCredentials(), null, true).then(function 
    (buildResult) {

                    _this.setState({
                        eventDetail: result.body,
                        latitude: buildResult.body.lat,
                        longitude: buildResult.body.lng,
                        buildingName: buildResult.body.building,
                        eventDate: new Date(result.body.eventDate).toDateString(),
                        startTime: start,
                        endTime: endT,
                        loading: true
                    })
                }).catch(function (error) {
1x              })
1x

            }).catch(function (error) {
            })

            _this.serverRequestJudge = HttpRequest.httpRequest(constants.getServerUrl() +
    "/sweng500/event/judges/" + _this.state.eventId, "get", constants.useCredentials(), null,
    true).then(function (judgeResult) {
1x              _this.setState({
1x                  judgesDetail: judgeResult.body,
                })

            }).catch(function (error) {
            })

            _this.serverRequestJudge = HttpRequest.httpRequest(constants.getServerUrl() +
    "/sweng500/event/teams/" + _this.state.eventId, "get", constants.useCredentials(), null,
    true).then(function (teamResult) {
2x              _this.setState({
2x                  teamsDetail: teamResult.body,
2x              })
2x
2x          }).catch(function (error) {
2x          })
        }

        formatTimeString(date) {
            var hours = date.getHours();
            var minutes = date.getMinutes();
            var ampm = hours >= 12 ? 'pm' : 'am';
            hours = hours % 12;
            hours = hours ? hours : 12;
            return hours + ":" + minutes + " " + ampm;
        }

        closeTeamModal() {
            this.setState({loadAddTeam: true, teamModal: false, teamSelectorError: ''})
        }

        selectedTeam(value) {
            this.setState({selectedTeamValue: value})
        }

        registerTeam() {
```

```
        var _this = this;
        //custom dropdown stores the entire object so just get the idea
        if (_this.state.selectedTeamValue === null) {
            _this.setState({teamSelectorError: 'Please select a team'});
        } else {
            var body = {teamName: _this.state.selectedTeamValue.name, eventName: 
_this.state.eventDetail.name};

            _this.serverRequestJudge = HttpRequest.httpRequest(constants.getServerUrl() + 
"/sweng500/event/" + _this.state.eventId + "/" + _this.state.selectedTeamValue.id, "POST", 
constants.useCredentials(), body, true).then(function (judgeResult) {
6x              _this.props.addNotification("Success, the team has been registered ", "success", "tc", 6);
                _this.setState({loadAddTeam: true, teamModal: false, teamSelectorError: ''})
                _this.componentDidMount();

            }).catch(function (error) {
                _this.setState({teamSelectorError: 'Team already registered'});
            })

        }
    }

    renderIfEventFound() {
        I if (this.eventDetail !== null && Object.keys(this.state.eventDetail).length !== 0) {
            //only display delete events if an admin
            let showActionBar=null;
            if(this.state.showDeleteBtn) {
                showActionBar = <TableHeaderColumn>Action</TableHeaderColumn>
            }
            return (
                <MuiThemeProvider>
                    <Row>
                        <Col md={8} mdOffset={2} xs={7}>
                            <Panel bsStyle="info">
                                <Panel.Heading>Event </Panel.Heading>
                                <Panel.Body>
                                    <PageHeader>
                                        {this.state.eventDetail.name} <br/>
                                        <Divider/>
                                        <small
                                            style={{'overflow-wrap': 'break-word'}}>
{this.state.eventDetail.description}</small>
                                    </PageHeader>
                                </Panel.Body>
                            </Panel>
                        </Col>
                        <Divider/>
                    </Row>
                    <Row>
                        <Col md={6} xs={8}>
                            <Panel bsStyle="info">
                                <Panel.Heading>Event Details</Panel.Heading>
                                <Panel.Body><h4>Event Date : {this.state.eventDate} </h4><br/><br/>
                                    <Divider/>
                                    <h4>Start Time : {this.state.startTime} </h4> <br/> <br/>
                                    <Divider/><br/>
                                    <h4> End Time : {this.state.endTime} </h4></Panel.Body>
                            </Panel>
                        </Col>
                        <Col md={6} xs={8}>
```

```
                            <Panel bsStyle="info">
                                <Panel.Heading>Building: {this.state.buildingName} ----
                                    Room: {this.state.eventDetail.room.roomName}</Panel.Heading>
                                <Panel.Body>
                                    <div id="map"
                                         style={{height: 300, width: 400, marginLeft: 'auto', marginRight:
'auto'}}>
                                        <Map
                                            style={this.divStyle}
                                            google={this.props.google}
                                            zoomControl={true}
                                            initialCenter={{
                                                lat: 41.306610,
                                                lng: -76.015437
                                            }}
                                            zoom={16}
                                            clickableIcons={false}>
                                            {this.addMarker()}
                                        </Map>
                                    </div>

                                </Panel.Body>

                            </Panel>
                        </Col>


                    </Row>
                    <Row>
                        <Col md={6} xs={8}>
                            <Panel bsStyle="info">
                                <Panel.Heading>Judges</Panel.Heading>
                                <Panel.Body><Table selectable={false}>
                                    <TableHeader displaySelectAll={false}
                                                 adjustForCheckbox={false}>
                                        <TableRow>
                                            <TableHeaderColumn>First Name</TableHeaderColumn>
                                            <TableHeaderColumn>Last Name</TableHeaderColumn>
                                            {showActionBar}
                                        </TableRow>
                                    </TableHeader>
                                    {this.renderIfJudgesFound()}

                                </Table></Panel.Body>
                            </Panel>
                        </Col>
                        <Col md={6} xs={8}>
                            <Panel bsStyle="info">
                                <Panel.Heading>Teams</Panel.Heading>
                                <Panel.Body><Table selectable={false}>
                                    <TableHeader displaySelectAll={false}
                                                 adjustForCheckbox={false}>
                                        <TableRow>
                                            <TableHeaderColumn>Team Name</TableHeaderColumn>
                                            <TableHeaderColumn>School Name</TableHeaderColumn>
                                            {showActionBar}
                                        </TableRow>
                                    </TableHeader>
                                    {this.renderIfTeamsFound()}
                                </Table></Panel.Body>
                            </Panel>
                        </Col>

                    </Row>
                </MuiThemeProvider>
            )
        }
    }

    //Return the judges for this event, only show first and last name
    renderIfJudgesFound() {
        if (this.state.judgesDetail !== null && Object.keys(this.state.judgesDetail).length !== 0) {
            let deleteBtnJudge = null;
            return (
                <TableBody displayRowCheckbox={false}
                           showRowHover={true}>
                    {
                        Object.keys(this.state.judgesDetail).map(function (key) {
                            if(this.state.showDeleteBtn) {
                                deleteBtnJudge=  <TableRowColumn><RaisedButton icon={<FontIcon 
className="pe-7s-trash" />}
                                                              secondary={true} label="Delete"
                                                              onClick={this.confirmJudgeDelete.bind(this,
this.state.judgesDetail[key])}/>
                                                </TableRowColumn>
                            }
                            return (
                                <TableRow key={key}>
                                    <TableRowColumn>{this.state.judgesDetail[key].firstName}
</TableRowColumn>
                                    <TableRowColumn>{this.state.judgesDetail[key].lastName}
</TableRowColumn>
                                    {deleteBtnJudge}

                                </TableRow>
                            )

                        }, this)
                    }
                </TableBody>
            )
        }
    }

    confirmJudgeDelete = (s) => {
        this.setState({
            confirmDialog:true,
            confirmMessage:'Are you sure you want to delete the following judge:  ' + s.firstName + " " +
s.lastName,
            deleteTeam:false,
            deleteJudge:true,
            deleteId:s.id
        })
    }

    removeJudge() {

        var _this = this;
        var removeId = _this.state.deleteId;
        //just making remove a post for now
        _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + 
"/sweng500/event/"+this.state.eventId+"/removeJudge/" + removeId, "DELETE", constants.useCredentials(), 
null, true).then(function (result) {
            _this.setState({confirmDialog: false});
            _this.props.addNotification(
                "Success: The judge has been removed",
                "info",
                "tc",
                6
            );
            _this.componentDidMount();
        }).catch(function (error) {
            _this.setState({confirmDialog: false});
            _this.props.addNotification(
                "Error: The judge could not be removed",
                "error",
                "tc",
                6
            );
        })
    }

    //Return the judges for this event, only show first and last name
    renderIfTeamsFound() {
        if (this.state.teamsDetail !== null && Object.keys(this.state.teamsDetail).length !== 0) {
            let deleteBtnTeam =null;

            return (
                <TableBody displayRowCheckbox={false}
                           showRowHover={true}>
                    {
                        Object.keys(this.state.teamsDetail).map(function (key) {
                            if(this.state.showDeleteBtn) {
                                deleteBtnTeam= <TableRowColumn> <RaisedButton icon={<FontIcon 
className="pe-7s-trash" />}
                                                              secondary={true} label="Delete"
                                                              onClick={this.confirmTeamDelete.bind(this, 
this.state.teamsDetail[key])}/>
                                </TableRowColumn>
                            }
                            return (
                                <TableRow key={key}>
                                    <TableRowColumn>{this.state.teamsDetail[key].name}</TableRowColumn>
                                    <TableRowColumn>{this.state.teamsDetail[key].school.schoolName}
</TableRowColumn>
                                    {deleteBtnTeam}

                                </TableRow>
                            )

                        }, this)
                    }
                </TableBody>
            )
        }
    }
    confirmTeamDelete = (s) => {
        this.setState({
            confirmDialog:true,
            confirmMessage:'Are you sure you want to delete the following team: ' + s.name,
            deleteTeam:true,
            deleteJudge:false,
            deleteId:s.id
        })
    }

    removeTeam() {
        var _this = this;
        var removeId = _this.state.deleteId;
        //just making remove a post for now
        _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
6x  "/sweng500/event/"+this.state.eventId+"/removeTeam/" + removeId, "DELETE", constants.useCredentials(),
6x  null, true).then(function (result) {
            _this.setState({confirmDialog: false});
            _this.props.addNotification(
                "Success: The team has been removed",
                "info",
                "tc",
                6
            );
            _this.componentDidMount();
        }).catch(function (error) {
            _this.setState({confirmDialog: false});
            _this.props.addNotification(
                "Error: The team could not be removed",
                "error",
6x                  "tc",
                6
            );
        })
    }

    closeConfirmDialog = () => {
        this.setState({confirmDialog: false});
    }

    render() {
        let deleteActions=null;
        I if(this.state.deleteTeam) {
6x               deleteActions = [
                <FlatButton
                    label="Cancel"
                    primary={true}
                    onClick={this.closeConfirmDialog}
                />,
                <FlatButton
6x                      label="Delete"
                    primary={true}
6x                      onClick={this.removeTeam}
5x                  />,
            ];
        } else {
             deleteActions = [
                <FlatButton
                    label="Cancel"
                    primary={true}
1x                      onClick={this.closeConfirmDialog}
                />,
                <FlatButton
6x                      label="Delete"
```

```
                    primary={true}
                    onClick={this.removeJudge}
                />,
            ];
        }
        let modalTitleBar = <AppBar
            iconElementRight={<FlatButton label="Close"/>}
            showMenuIconButton={false}
            onRightIconButtonClick={(event) => this.closeTeamModal()}
            title="Register a team to an event"
        />

        let cardButtons=null;
        //only show register buttons to admins and coaches
        if (this.state.showRegisterBtn) {
            cardButtons = <div><RaisedButton icon={<FontIcon className="pe-7s-angle-left-circle"/>}
primary={true}
                                        label="Go back to events"
                                        onClick={event => this.props.showEvents(event)}/>&nbsp;&nbsp;
                <RaisedButton icon={<FontIcon className="pe-7s-study"/>} primary={true} label="Register a
Team"
                          onClick={event => this.showTeamModal(event)}/></div>

        } else {
            cardButtons =   <RaisedButton icon={<FontIcon className="pe-7s-angle-left-circle" />} primary=
{true} label="Go back to events"
                                          onClick={event => this.props.showEvents(event)}/>
        }
        return (
            <div className="content">

                <MuiThemeProvider>
                    <Grid fluid>
                        <Row>
                            <Col md={12}>
                                <Card
                                    hCenter={this.state.eventDetail.eventName}
                                    // title={this.state.eventDetail.eventName}
                                    category={
                                        <div>
                                            {cardButtons}
                                        </div>}
                                    content={
                                        <Loader color="#3498db" loaded={this.state.loading}>
                                            {this.renderIfEventFound()}
                                        </Loader>
                                    }/>
                            </Col>
                        </Row>
                    </Grid>
                    <Modal bsSize="medium" show={this.state.teamModal} onHide={this.closeTeamModal}>
                        <Modal.Header>
                            <Modal.Title> {modalTitleBar}</Modal.Title>
                        </Modal.Header>

                        <Modal.Body>
                            <Grid>
                                <Row>
                                    <Col xs={7} md={3}>
                                        <CustomDropdown
```

```
                                            name={"team"}
                                            labelText={"Team"}
                                            hintText={"Select team"}
                                            selected={this.state.selectedTeamValue}
                                            endpoint={"/sweng500/getTeamsByUser"}
                                            sortKey={"name"}
                                            textKeys={["name"]}
                                            selectedValue={this.selectedTeam}
                                            errorText={this.state.teamSelectorError}
                                        />
                                    </Col>
                                </Row>
                            </Grid>
                        </Modal.Body>

                        <Modal.Footer>
                            <Loader loaded={this.state.loadAddTeam}></Loader>
                            <RaisedButton icon={<FontIcon className="pe-7s-like2"/>} primary={true}
                                          label="Register Team"
                                          onClick={event => this.registerTeam()}/>
                        </Modal.Footer>
                    </Modal>
                    <Dialog
                        actions={deleteActions}
                        modal={false}
                        open={this.state.confirmDialog}
                        onRequestClose={this.closeConfirmDialog}
                    >
                        {this.state.confirmMessage}
                    </Dialog>
                </MuiThemeProvider>

            </div>

        )
    }

    showTeamModal(event) {
        this.setState({teamModal: true})
    }
}

export default GoogleApiWrapper({
    apiKey: "AIzaSyC7xiiV97LyRQd-GB9aBmiJaYFGW5DVIbM"
})(EventDetail);
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/views/Events Events.js

74.44% Statements 198/266 **76.74%** Branches 66/86 **44.23%** Functions 23/52 **74.44%** Lines 198/266

```
      import React, {Component} from 'react';
      import {PageHeader, Well, Alert, Grid, Col, Row, Modal} from 'react-bootstrap';
      import Loader from 'react-loader'
      import {
          Step,
          Stepper,
          StepLabel,
          StepContent,
      } from 'material-ui/Stepper';
      import $ from 'jquery';
      import HttpRequest from "../../adapters/httpRequest";
      import constants from "../../utils/constants";
      import MuiThemeProvider from 'material-ui/styles/MuiThemeProvider';
      import RaisedButton from 'material-ui/RaisedButton';
      import TextField from 'material-ui/TextField';
      import DatePicker from 'material-ui/DatePicker'
      import TimePicker from 'material-ui/TimePicker'
      import AppBar from 'material-ui/AppBar'
      import {blue500} from 'material-ui/styles/colors'
      import FlatButton from 'material-ui/FlatButton';
      import FontIcon from 'material-ui/FontIcon';
      import ReactTable from 'react-table'
      import "react-table/react-table.css";
      import matchSorter from 'match-sorter'
      import SelectField from 'material-ui/SelectField';
      import MenuItem from 'material-ui/MenuItem';
      import Dialog from 'material-ui/Dialog';
      import EventDetail from './EventDetail'
      import NotificationSystem from 'react-notification-system';
      import {style} from "../../variables/Variables";
      import BuildingSelector from "../../components/Buildings/BuildingSelector";
      import * as Promises from "axios";
      import AuthService from "../../utils/AuthService";

      class Events extends Component {
          constructor(props) {
18x           super(props);
18x           this.closeModal = this.closeModal.bind(this);
18x           this.createNewEvent = this.createNewEvent.bind(this);
18x           this.createEventPost = this.createEventPost.bind(this);
18x           this.removeEvent = this.removeEvent.bind(this);
18x           this.nextStep = this.nextStep.bind(this);
18x           this.previousStep = this.previousStep.bind(this);
18x           this.eventDetails = this.eventDetails.bind(this);
18x           this.showEvents = this.showEvents.bind(this);
18x           this.addJudgeInputs = this.addJudgeInputs.bind(this);
18x           this.removeNewJudge = this.removeNewJudge.bind(this);
18x           this.addNotification = this.addNotification.bind(this);
18x           this.buildingCallback = this.buildingCallback.bind(this);

18x           this.state = {
                  events: {},
                  editMode:false,
                  editEventId:'',
            loading: false,
            loadCreateEvent: true,
            modal: false,
            confirmDialog: false,
            confirmMessage: '',
            deleteID: '',
            stepIndex: 0,
            //used to send to event detail as a prop
            eventId: '',

            eventName: '',
            eventDate: '',
            startTime: '',
            endTime: '',
            eventLocation: '',
            eventDescription: '',
            //for adding judges
            judgeInputs: [],
            //keep track of the actual new Judges
            judgeCount: 0,
            //hold the existing judges
            existingJudgeValues: [],
            existingJudgeEmails: [],
            _notificationSystem: null


        };
    }


    buildingCallback(event, index, value) {

        this.setState({eventLocation: value});
    }

    addNotification(message, level, position, autoDismiss) {
        if(this.state._notificationSystem) {
            this.state._notificationSystem.addNotification({
                title: (<span data-notify="icon" className="pe-7s-note2"></span>),
                message: (
                    <div>
                        {message}
                    </div>
                ),
                level: level ? level : 'info',
                position: position ? position : 'tc',
                autoDismiss: autoDismiss ? autoDismiss : 10,
            });
        }
    }

    //launch the modal to enter an event or edit one
    createNewEvent = (mode) => {
        const _this = this;
        if(mode.status === "edit") {
            _this.serverRequestJudge = HttpRequest.httpRequest(constants.getServerUrl() + 
"/sweng500/event/judges/" + mode.id, "get", constants.useCredentials(), null, true).then(function 
(judgeResult) {
                let judgeVals = judgeResult.body;
                //have the judges for this event already selected
                var selectedJudgeIds = [];
                for (let value in judgeVals) {
                    selectedJudgeIds.push(judgeVals[value].id);
                }
                _this.setState({
                    modal: true,
                    stepIndex: 0,
                    editEventId: mode.id,
                    editMode:true,
                    eventName: mode.name,
                    eventDate: new Date(mode.eventDate),
                    startTime: new Date(mode.startTime),
                    endTime: new Date(mode.endTime),
                    eventDescription: mode.description,
                    eventLocation: mode.room.id,
                    existingJudgeValues: selectedJudgeIds,
                    judgeInputs: [],
                    judgeCount: 0,

                })

            }).catch(function (error) {
                //no judges assigned to the event yet
                _this.setState({
                    editEventId: mode.id,
                    modal: true,
                    stepIndex: 0,
                    editMode:true,
                    eventName: mode.name,
                    eventDate: new Date(mode.eventDate),
                    startTime: new Date(mode.startTime),
                    endTime: new Date(mode.endTime),
                    eventDescription: mode.description,
                    eventLocation: mode.room.id,
                    existingJudgeValues: [],
                    judgeInputs: [],
                    judgeCount: 0,

                })
            })
        } else {
            //creating new event
            _this.setState({
                modal: true,
                editMode:false,
                editEventId: '',
                stepIndex: 0,
                eventName: '',
                eventDate: '',
                startTime: '',
                endTime: '',
                eventDescription:'',
                eventLocation: '',
                existingJudgeValues: [],
                judgeInputs: [],
                judgeCount: 0,
            })
        }
    }
  2x
```

```
2x      createEventPost() {
2x          var _this = this;
2x          var body = {};
            var gatherjudges = {};
2x          var newJudgeList = [];
1x          var i;
1x          for (i = 1; i <= this.state.judgeCount; i++) {
1x              var tempFname = "#judgefname" + i;
                var tempLname = "#judgelname" + i;
1x              var tempEmail = "#judgemail" + i;
1x              // alert($(tempFname).val() + "     " + $(tempLname).val() +  "     " + $(tempEmail).val());
1x              gatherjudges.fname = $(tempFname).val();
                gatherjudges.lname = $(tempLname).val();
1x              gatherjudges.email = $(tempEmail).val();
1x
                newJudgeList.push(gatherjudges);
                gatherjudges = {};


2x          }

2x              var event = {};
2x
2x              event.name = this.state.eventName;
2x              event.description = this.state.eventDescription;
2x              event.eventDate = this.state.eventDate;
2x              event.room = this.state.eventLocation;
                event.startTime = this.state.startTime;
2x              event.endTime = this.state.endTime;
                //the actual string
2x              body.eventJson = event;
2x              //exiting judge values has an extra pair of quotes around each item in the list...
2x              body.existingJudgeValues = this.state.existingJudgeValues;
2x              body.newJudgeValues = newJudgeList;
                console.log(JSON.stringify(body));
                _this.setState({
                    //show the spinner
2x                  loadCreateEvent: false
2x              });
                E if(this.state.editMode) {
                    _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
2x  "/sweng500/updateEvent/"+this.state.editEventId, "POST", constants.useCredentials(), body,
    true).then(function (result) {
                        //show some success and then clear the fields
                        //refresh after an add to show new event in table
                        _this.setState({
                            modal: false,
                            stepIndex: 0,
                            loadCreateEvent: true,
                            eventName: '',
                            eventDescription: '',
                            //this is a date object
                            eventDate: '',
                            startTime: '',
                            endTime: '',
                            eventLocation: '',
                            judgeInputs: [],
                            judgeCount: 0,
                            existingJudgeValues: [],
2x                          existingJudgeEmails: [],
                            renderDetails: false,

                        });
                        _this.addNotification(
                            "Success: The event has been saved.",
    2x                      "success",
                            "tc",
                            6
                        );
                        _this.componentDidMount();

                    }).catch(function (error) {
                        _this.setState({
                            loadCreateEvent: true,
                        });
                        _this.addNotification(
                            "Error: There was a problem editing the event.",
                            "error",
                            "tc",
                            6
                        );
                        _this.componentDidMount();
                    })
                } else {
                    _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
    "/sweng500/addEvent/", "POST", constants.useCredentials(), body, true).then(function (result) {
                        //show some success and then clear the fields
                        //refresh after an add to show new event in table
                        _this.setState({
                            modal: false,
                            stepIndex: 0,
                            loadCreateEvent: true,
                            eventName: '',
                            eventDescription: '',
                            //this is a date object
                            eventDate: '',
                            startTime: '',
                            endTime: '',
                            eventLocation: '',
                            judgeInputs: [],
                            judgeCount: 0,
                            existingJudgeValues: [],
                            existingJudgeEmails: [],
                            renderDetails: false,

                        });
                        _this.addNotification(
                            "Success: The event has been added.",
                            "success",
                            "tc",
                            6
                        );
                        _this.componentDidMount();

                    }).catch(function (error) {
                        _this.setState({
                            loadCreateEvent: true
                        });
                        _this.addNotification(
                            "Error: There was a problem creating the event.",
                            "error",
                            "tc",
                            6
                        );
                        _this.componentDidMount();
                    })
1x              }
    }

    closeModal() {
        //reset when closing modal
        this.setState({
            modal: false,
            editMode:false,
            editEventId:''
4x      })
4x  }
1x
    // Checks to see if an email has a host, @ symbols, and domain.
3x  validEmail(text) {
        let reg = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
        if (reg.test(text) === false)
            return true;
        else
            return false;
6x  }
6x
6x  nextStep() {
        //validate event entries here
6x
2x      var _this = this;
2x      var missingInfo = false;
1x      const {eventName, stepIndex} = this.state;
1x      // Checks first name
        if (stepIndex == 0) {
            var promises =[];
            if (this.state.eventName.length < 1) {
                missingInfo = true;
1x              this.setState({
1x                  eventName: this.state.eventName.trim(),
                    eventNameError: "Event name is required"
                })
                //only check if duplicate event if we are creating a new event
            } else E if(!this.state.editMode){
                promises.push(_this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
2x "/sweng500/verifyEvent/" + eventName, "get", constants.useCredentials(), null, true));
1x          } else {
1x              this.setState({
                    eventNameError: undefined
                })
            }
1x          if (this.state.eventDate.length < 1) {
                missingInfo = true;
                this.setState({
                    eventDateError: "Event date is required"
2x              })
1x          } else {
1x              this.setState({
                    eventDateError: undefined
                })
            }
1x              if (this.state.startTime.length < 1) {
                missingInfo = true;
                this.setState({
                    startTimeError: "Event start time is required"
2x                  })
1x              } else {
1x                  this.setState({
                    startTimeError: undefined
                })
            }
1x              if (this.state.endTime.length < 1) {
                missingInfo = true;
                this.setState({
                    endTimeError: "Event end time is required"
2x                  })
1x              } else {
1x                  this.setState({
                    endTimeError: undefined
                })
            }
            if (this.state.eventLocation.length < 1) {
1x                  missingInfo = true;
                this.setState({
                    eventLocationError: "Please select a building for the event"
                })
            }
2x              else {
1x                  this.setState({
1x                      eventLocationError: undefined
                })
            }

1x              if (this.state.eventDescription.length < 1) {
                missingInfo = true;
                this.setState({
                    eventDescriptionError: "Event description is required"
                })
2x              } else {
1x                  this.setState({
1x                      eventDescriptionError: undefined
                })
1x              }

            if (!missingInfo) {
                E if(promises.length > 0) {
                    Promises.all(promises).then(function () {

                            _this.setState({
                                eventNameError: undefined,
                                stepIndex: stepIndex+1
                            })

                    }).catch(function (error) {
                            _this.setState({
                                eventNameError: "Event Name already exists"
                            });
                            console.log(error);
                    })
4x                  } else {
```

```
                            _this.setState({
                                stepIndex: stepIndex + 1
1x                          })
                        }
                    }
            } else if (stepIndex == 1) {
                //no validation since we are just pulling judges from the db
                //clear the new judge inputs since we don't keep state on them
                _this.setState({
                    stepIndex: stepIndex + 1,
3x                  judgeInputs: [],
3x                  judgeCount: 0
3x              })
3x          } else {
                //only verify the last email and send to create event post
3x
1x              var validateFields = false;
1x              let judgeCnt = _this.state.judgeCount;
1x              let idname = "#judgemail" + judgeCnt;
1x              var promises = [];
1x              //do not run when displaying the first set of inputs
1x              if (judgeCnt > 0) {
                    var email = $(idname).val();
                    var body = {};
                    E if (email != null && email.length > 0) {
                        E if (!this.validEmail(email.trim())) {
                            body.emailAddress = email;
                            promises.push(_this.serverRequest =
HttpRequest.httpRequest(constants.getServerUrl() +
                                '/sweng500/emailAvailable', 'POST', null, body));
                        } else {
                            validateFields = true;
                            $(idname).parent().parent().parent().parent().find(".errorText").text("Email is
not in the correct format");
                            $(idname).parent().parent().parent().parent().find(".errorText").css("display",
3x  "block");
1x                      }
1x                  } else {
1x                      validateFields = true;
1x                      $(idname).parent().parent().parent().parent().find(".errorText").text("Email is
required");
1x                      $(idname).parent().parent().parent().parent().find(".errorText").css("display",
"block");
                    }
                }
                //only continue to create event if this email is good
                if (promises.length > 0) {
                    Promises.all(promises).then(function (results) {
                        $(idname).attr("disabled", true);
2x                      $(idname).parent().parent().parent().parent().find(".errorText").text("");
2x                      $(idname).parent().parent().parent().parent().find(".errorText").css("display",
1x  "none");
                        //finally send the event request
1x                      _this.createEventPost();

                    }).catch(function (error) {
                        $(idname).parent().parent().parent().parent().find(".errorText").text("Email is
already in use");
                        $(idname).parent().parent().parent().parent().find(".errorText").css("display",
1x  "block");
```

```
                    });
                } else {
                    //just create event if no new judges and no syntax error
                    E if (!validateFields) {
                        if (_this.state.existingJudgeValues.length > 0) {
                            _this.createEventPost()
                        } else {
                            _this.addNotification(
1x                              "Error: Please assign at least one judge to the event or create a new one.",
1x                              "error",
                                "tc",
                                6
                            );
                            _this.componentDidMount();
                        }
                    }
                }
            }
1x      }

        //go back a step while creating event
        previousStep() {
1x          const {stepIndex} = this.state;
1x          this.setState({
                stepIndex: stepIndex - 1
            })
        }

        //go to the event page to show all of the information available to edit
        //will need a similar method to go back to event main page..
        eventDetails(id) {
            //alert(id);
            this.setState({
                renderDetails: true,
                eventId: id
            })
            $('#eventDetails').removeClass('collapse');
            $('#eventPage').addClass('collapse');
1x
1x      }

1x      //called from the subclass to go back to events
1x      showEvents(event) {
1x          event.preventDefault();
            this.setState({
                renderDetails: false
            })
            $('#eventDetails').addClass('collapse');
            $('#eventPage').removeClass('collapse')
1x      }

        removeEvent() {
            var _this = this;
            var removeId = _this.state.deleteID;
            //just making remove a post for now
            _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
    "/sweng500/removeEvent/" + removeId, "DELETE", constants.useCredentials(), null, true).then(function
    (result) {
                _this.setState({confirmDialog: false});
                _this.addNotification(
            "Success: The event has been deleted.",
1x          "info",
            "tc",
            6
          );
          _this.componentDidMount();
        }).catch(function (error) {
          _this.setState({confirmDialog: false});
          _this.addNotification(
            "Error: The event has not been deleted.",
2x          "error",
            "tc",
2x          6
          );
2x        _this.componentDidMount();
2x        console.log(error);
2x      })
        _this.setState({confirmDialog: false});
2x
1x
1x    }
1x
1x    //dynamically add in new input fields when clicked, validate the previous entry first
1x    addJudgeInputs() {
1x      //get a local copy so we dont set state here and re-render, update state after

        var _this = this;
        //control whether we execute the promise based on validation
        var validateFields = true;

        let judgeCnt = _this.state.judgeCount;
        let idname = "#judgemail" + judgeCnt;
        var promises = [];
        //do not run when displaying the first set of inputs
        if (judgeCnt > 0) {
          var email = $(idname).val();
          var body = {};
2x        E if (email != null && email.length > 0) {
2x          E if (!this.validEmail(email.trim())) {
              body.emailAddress = email;
2x            promises.push(_this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl()
+ '/sweng500/emailAvailable', 'POST', null, body));
            } else {
              //not the prettiest code....
              $(idname).parent().parent().parent().parent().find(".errorText").text("Email is not
in the correct format");
              $(idname).parent().parent().parent().parent().find(".errorText").css("display",
"block");
2x            validateFields = false;
            }
          } else {
            $(idname).parent().parent().parent().parent().find(".errorText").text("Email is
required");
            $(idname).parent().parent().parent().parent().find(".errorText").css("display",
"block");
2x          validateFields = false;
          }
        }
        E if (validateFields) {
          judgeCnt = judgeCnt + 1;
```

```
    
                const fnameJudge = [
                    <TextField
                        id={"judgefname" + judgeCnt}
2x                      floatingLabelText="First Name"
1x                      required={true}
1x                      autoFocus={true}
1x                      fullWidth={true}
1x                  />];
                const lnameJudge = [
1x                  <TextField
                        id={"judgelname" + judgeCnt}
                        floatingLabelText="Last Name"
                        required={true}
                        fullWidth={true}
                    />];
                const emailJudge = [
                    <TextField
                        id={"judgemail" + judgeCnt}
                        floatingLabelText="Email"
                        required={true}
                        fullWidth={true}
                    />];

                //only add thew new row if the previous email is valid or
                if (promises.length > 0) {
                    Promises.all(promises).then(function (results) {
                        $(idname).attr("disabled", true);
                        $(idname).parent().parent().parent().parent().find(".errorText").text("");
                        $(idname).parent().parent().parent().parent().find(".errorText").css("display",
    "none");

                        _this.setState({
                            judgeCount: _this.state.judgeCount + 1,
1x
                            //add judges class to use jquery to loop over reach one
                            judgeInputs: _this.state.judgeInputs.concat(<Row>
                                <div className={"col-md-9 col-xs-7" + " row" + judgeCnt}><Well>
                                    <h3>New Judge {judgeCnt}</h3>
                                    <Row><Alert style={{display: "none"}} bsStyle="danger"
                                                className="errorText"></Alert></Row>
                                    <Row><Col md={4} xs={7}>{fnameJudge}</Col></Row>
                                    <Row><Col md={4} xs={7}>{lnameJudge}</Col></Row>
                                    <Row><Col md={4} xs={7}>{emailJudge}</Col></Row></Well>
                                </div>
                            </Row>)
                        });
                    }).catch(function (error) {
                        //ugly way to add error text

                        $(idname).attr("disabled", false);
                        $(idname).parent().parent().parent().parent().find(".errorText").text("Email is
    already in use");
                        $(idname).parent().parent().parent().parent().find(".errorText").css("display",
    "block");
1x
1x                  });
1x              } else {
1x                  _this.setState({
1x                      judgeCount: _this.state.judgeCount + 1,
1x                      //add judges class to use jquery to loop over reach one
                        judgeInputs: _this.state.judgeInputs.concat(<Row>
1x                          <div className={"col-md-9 col-xs-7" + " row" + judgeCnt}><Well>
                                <h3>New Judge {judgeCnt}</h3>
                                <Row> <Alert style={{display: "none"}} className="errorText"
1x  bsStyle="danger"></Alert></Row><Row>
1x                              <Col md={4} xs={7}>{fnameJudge}</Col></Row>
                                <Row><Col md={4} xs={7}>{lnameJudge}</Col></Row>
                                <Row><Col md={4} xs={7}>{emailJudge}</Col></Row></Well>
                            </div>
                        </Row>)
                    });
                }
            }

        }

1x      //Remove the text fields for the most recent judge
1x      removeNewJudge = (judgeCount) => {
            //alert("judge count" + judgeCount);
            var _this = this;
            var tempCount = _this.state.judgeCount;
            var domId = ".row" + tempCount;
            $(domId).remove();
            tempCount -= 1;
            domId = ".row" + tempCount;
            //reenable the previous row
22x          $(domId).find("#judgemail" + tempCount).attr("disabled", false);
22x
22x          //remove from the array and decrement
            _this.state.judgeInputs.pop();
            _this.setState({
                judgeCount: _this.state.judgeCount - 1,

22x          })
22x

22x      }
22x
22x      // Delete the event
22x      confirmEventDelete = (s) => {
            // alert("Got here");
            this.setState({deleteID: s.id});
            this.setState({confirmMessage: "Are you sure you want to delete : " + s.name + " ?",
    confirmDialog: true});
        }

        closeConfirmDialog = () => {
            this.setState({confirmDialog: false});
        }

        componentDidMount() {
            //hide things from non admins
            let modifyRoles = ['ADMIN'];
            let allowModify = AuthService.isUserRoleAllowed(modifyRoles);
                this.setState({
                    showDeletebtn: allowModify
                });

148x          //Make call out to backend
```

```
  58x          var _this = this;
               this.setState({_notificationSystem: this.refs.notificationSystem});

               _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + "/sweng500/events",
       "get", constants.useCredentials(), null, true).then(function (result) {
                   _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
       "/sweng500/getJudges", "get", constants.useCredentials(), null, true).then(function (judgeResult) {
                       console.log(result.body);
                       _this.setState({
                           events: result.body,

                           existingJudgeEmails: judgeResult.body,
                           loading: true
                       })

                   }).catch(function (error) {
                       console.log(error);
                   })

  58x          }).catch(function (error) {
 116x              console.log(error);
 114x          })
 114x
           }

           renderIfEventsFound() {

               if (this.state.events !== null && Object.keys(this.state.events).length !== 0) {
                   const columns = [{
                       Header: 'Event Name',
                       filterMethod: (filter, rows) =>
                           matchSorter(rows, filter.value, {keys: ["name"]}),
                       filterAll: true,
                       accessor: 'name' // String-based value accessors!
   2x              }, {
                       Header: 'Event Description',
                       filterMethod: (filter, rows) =>
                           matchSorter(rows, filter.value, {keys: ["description"]}),
                       filterAll: true,
                       accessor: 'description' // String-based value accessors!
                   }, {
                       Header: 'Actions',
  58x                  accessor: 'menuActions', // String-based value accessors!
                       style: {textAlign: 'center'},
                       sortable: false,
                       filterable: false
                   }];
                   for (let value in this.state.events) {
                       if (this.state.showDeletebtn) {
                           this.state.events[value].status = "edit";
                           this.state.events[value].menuActions = <div>
                               <RaisedButton style={{minWidth: '30%'}} icon={<FontIcon className="pe-7s-note2"
       />}
                                   primary={true} label="View Details"
                                   onClick={(event) =>
       this.eventDetails(this.state.events[value].id)}/>&nbsp;&nbsp;
  90x                          <RaisedButton style={{minWidth: '30%'}} icon={<FontIcon className="pe-7s-edit"
       />}
                                   primary={true} label="Edit"
                                   onClick={this.createNewEvent.bind(this,
      this.state.events[value])}/>&nbsp;&nbsp;
150x                          <RaisedButton style={{minWidth: '30%'}} icon={<FontIcon className="pe-7s-trash"
      />}
                                  secondary={true} label="Delete"
                                  onClick={this.confirmEventDelete.bind(this, this.state.events[value])}/>
                              </div>
                          } else {
                              this.state.events[value].menuActions = <div>
150x                              <RaisedButton style={{minWidth: '30%'}} icon={<FontIcon className="pe-7s-note2"
150x  />}
                                  primary={true} label="View Details"
150x                              onClick={(event) => this.eventDetails(this.state.events[value].id)}/></div>
118x                      }
                      }


 32x                  return (
  5x                      <ReactTable
                              data={this.state.events}
                              filterable
  5x                          defaultFilterMethod={(filter, row) =>
                                  String(row[filter.id]) === filter.value}
                              columns={columns}
                              defaultPageSize={10}
                              className="-striped -highlight"
 27x                          defaultSorted={[{id: "name"}]}
                          />
                      )
                  } else {
 27x
 25x                  <h1>Not Found</h1>
                  }
              }
  2x
              render() {
                  const styles = {
                      formFields: {
                          color: blue500,
150x                      textAlign: 'center',
  2x                      width: '100%'
                      }
                  };
                  let actionButton = null;
                  let backButton = null;
                  //control which buttons show up in the modal
                  if (this.state.stepIndex == 0) {
                      actionButton =
148x                      <RaisedButton icon={<FontIcon className="pe-7s-angle-down-circle"/>} primary={true}
      label="Judges"
                                    onClick={this.nextStep}
                          />;
                  } else if (this.state.stepIndex == 1) {
                      backButton =
                          <RaisedButton icon={<FontIcon className="pe-7s-angle-up-circle"/>} primary={true}
      label="Back to Info"
                                    onClick={this.previousStep}/>;
                      actionButton = <RaisedButton icon={<FontIcon className="pe-7s-angle-down-circle"/>} primary=
      {true}
                                                 label="Add new judges"
                                                 onClick={this.nextStep}/>;
```

```
148x  
  9x          } else {
                  backButton =
                      <RaisedButton icon={<FontIcon className="pe-7s-angle-up-circle"/>} primary={true}
                                    label="Back to Existing"
                                    onClick={this.previousStep}/>;
148x                                if(this.state.editMode) {
128x                                    actionButton = <RaisedButton icon={<FontIcon className="pe-7s-
      like2"/>} primary={true} label="Save Event"
                                                                   onClick={this.nextStep}/>;
                                    } else {
                                        actionButton = <RaisedButton icon={<FontIcon className="pe-7s-
      like2"/>} primary={true} label="Create Event"
                                                                   onClick={this.nextStep}/>;
                                    }

148x          }
 39x          if (this.state.renderDetails) {
                  return (
                      <div class="content">
                          <NotificationSystem ref="notificationSystem" style={style}/>
                          <EventDetail addNotification={this.addNotification} showEvents={this.showEvents}
      eventId={this.state.eventId}/>
                      </div>
109x              )
              }

              const deleteActions = [
                  <FlatButton
                      label="Cancel"
                      primary={true}
148x                  onClick={this.closeConfirmDialog}
                  />,
                  <FlatButton
                      label="Delete"
                      primary={true}
                      onClick={this.removeEvent}
                  />,
              ];
              let removeJudgeBtn;
              if (this.state.judgeCount > 0) {
                  removeJudgeBtn = <RaisedButton icon={<FontIcon className="pe-7s-close"/>}
                                                 secondary={true} label="Remove judge"
                                                 onClick={this.removeNewJudge.bind(this)}/>;
              }
              let createEventBtn;
              if (this.state.showDeletebtn) {
                  createEventBtn = <Row className="show-grid">
                      <Col md={4} mdOffset={4}>
                          <RaisedButton icon={<FontIcon className="pe-7s-display1" />} primary={true}
      label="Create New Event"
                                        onClick={this.createNewEvent}/>
                          <br/>
                      </Col>
                  </Row>
              }
              let modalTitleBar;
              if(this.state.editMode) {
                 modalTitleBar= <AppBar
                      iconElementRight={<FlatButton label="Close"/>}
```

```
                showMenuIconButton={false}
                onRightIconButtonClick={(event) => this.closeModal()}
                title="Modify Event"
            />
        } else {
           modalTitleBar= <AppBar
                iconElementRight={<FlatButton label="Close"/>}
                showMenuIconButton={false}
                onRightIconButtonClick={(event) => this.closeModal()}
                title="Create New Event"
            />
        }
        return (

            <div className="content">
                <NotificationSystem ref="notificationSystem" style={style}/>
                <div id="eventPage" key="notFound-key" className="notFoundClass">
                    <MuiThemeProvider>
                        <Grid fluid>
                            {createEventBtn}
                            <br/>
                            <br/>
                            <Row className="show-grid">

                                <Loader color="#3498db" loaded={this.state.loading}>

                                    {this.renderIfEventsFound()}
                                </Loader>

                            </Row>
                        </Grid>
                    </MuiThemeProvider>

                </div>
                <MuiThemeProvider>
                    <Modal bsSize="large" show={this.state.modal} onHide={this.closeModal}>
                        <Modal.Header>
                            <Modal.Title> {modalTitleBar}</Modal.Title>
                        </Modal.Header>

                        <Modal.Body>
                            <Grid>
                                <Stepper activeStep={this.state.stepIndex} orientation={'vertical'}>
                                    <Step>
                                        <StepLabel>Event Information</StepLabel>
                                        <StepContent>

                                            <Row className="show-grid">
                                                <Col md={3} mdOffset={1} xs={7}>
                                                    <TextField
                                                        id={"eventName"}
                                                        floatingLabelText="Event Name"
                                                        errorText={this.state.eventNameError}
                                                        onChange={(event, newValue) =>
this.setState({eventName: newValue})}
                                                        value={this.state.eventName}
                                                        required={true}
                                                        autoFocus={true}
                                                        fullWidth={true}
                                                    />
                                                </Col>
                                                <Col md={3} mdOffset={1} xs={7}>
                                                    <DatePicker
                                                        errorText={this.state.eventDateError}
                                                        onChange={(event, newValue) =>
this.setState({eventDate: newValue})}
                                                        value={this.state.eventDate}
                                                        hintText="Event Date"
                                                        floatingLabelText="Event Date"
                                                        textFieldStyle={styles.formFields}

                                                        mode="landscape"/>

                                                </Col>
                                            </Row>
                                            <br/>
                                            <Row className="show-grid">

                                            </Row>
                                            <br/>
                                            <Row className="show-grid">
                                                <Col md={3} mdOffset={1} xs={7}>
                                                    <TimePicker
                                                        errorText={this.state.startTimeError}
                                                        floatingLabelText="Start Time"
                                                        onChange={(event, newValue) =>
this.setState({startTime: newValue})}
                                                        value={this.state.startTime}
                                                        textFieldStyle={styles.formFields}
                                                        hintText="Start time"
                                                        autoOk={true}
                                                    />
                                                </Col>
                                                <Col md={3} mdOffset={1} xs={7}>
                                                    <TimePicker
                                                        errorText={this.state.endTimeError}
                                                        floatingLabelText="End Time"
                                                        onChange={(event, newValue) =>
this.setState({endTime: newValue})}
                                                        value={this.state.endTime}
                                                        hintText="End time"
                                                        textFieldStyle={styles.formFields}
                                                        autoOk={true}
                                                    />
                                                </Col>
                                            </Row>
                                            <Row className="show-grid">
                                                <Col md={5} mdOffset={1} xs={7}>
                                                    <BuildingSelector selected=
{this.state.eventLocation}
                                                                      errorMsg=
{this.state.eventLocationError}
                                                                      callBack={this.buildingCallback}
                                                                      labelText={"Building - Room"}
                                                                      hintText={"Select a
building/room"}/>
                                                </Col>
                                            </Row>
                                            <Row className={"show-grid"}>
                                                <Col md={5} mdOffset={1} xs={7}>
```

```
                                                          <TextField
                                                              id={"eventDescription"}
                                                              floatingLabelText="Event Description"
                                                              errorText={this.state.eventDescriptionError}
                                                              onChange={(event, newValue) =>
      this.setState({eventDescription: newValue})}
                                                              value={this.state.eventDescription}
                                                              required={true}
                                                              fullWidth={true}
                                                              multiLine={true}
                                                              rows={3}
                                                          />

                                                      </Col>

                                                  </Row>


                                              </StepContent>
                                          </Step>
                                          <Step>
                                              <StepLabel>Assign Existing Judges</StepLabel>
                                              <StepContent>
                                                  <Row>
                                                      <Col md={7} xs={5}>
                                                          <Alert bsStyle="info">
                                                              Select an existing judge or move to next step to
                                                              create new judges
                                                          </Alert>
                                                      </Col>
                                                  </Row>
                                                  <Row>
                                                      <Col md={7} xs={5}>
                                                          <SelectField
                                                              multiple={true}
                                                              fullWidth={true}
                                                              autoWidth={true}
                                                              hintText="Existing judges"
                                                              value={this.state.existingJudgeValues}
                                                              onChange={this.judgeMenuClick}
148x                                                      >
148x
104x  {this.judgeMenuItems(this.state.existingJudgeValues)}
                                                          </SelectField>
                                                      </Col>
                                                  </Row>
                                              </StepContent>
                                          </Step>
                                          <Step>
                                              <StepLabel>Create New Judges</StepLabel>
                                              <StepContent>

                                                  {this.state.judgeInputs}
                                                  <Row>
                                                      <Col md={3}>
                                                          <RaisedButton icon={<FontIcon className="pe-7s-
      angle-down-circle"/>}
                                                                    primary={true} label="Add new judge"
                                                                    onClick={this.addJudgeInputs}/>
                                                      </Col>
```

```
                                                <Col md={3}>
                                                    {removeJudgeBtn}
                                                </Col>

                                            </Row>
                                        </StepContent>
                                    </Step>
                                </Stepper>
                            </Grid>
                        </Modal.Body>

                        <Modal.Footer>
                            <Loader loaded={this.state.loadCreateEvent}></Loader> {backButton}
{actionButton}
                        </Modal.Footer>
                    </Modal>
                    <Dialog
                        actions={deleteActions}
                        modal={false}
                        open={this.state.confirmDialog}
                        onRequestClose={this.closeConfirmDialog}
                    >
                        {this.state.confirmMessage}
                    </Dialog>

                </MuiThemeProvider>
            </div>
        );
    }

    //adds in the menu items for existing judges
    judgeMenuItems = (existingJudgeValues) => {
        E if (this.state.existingJudgeEmails != null) {
            return this.state.existingJudgeEmails.map((obj) => (
                <MenuItem
                    key={obj.id}
                    insetChildren={true}
                    targetOrigin={{horizontal: "right", vertical: "bottom"}}
                    checked={existingJudgeValues && existingJudgeValues.indexOf(obj.id) > -1}
                    value={obj.id}
                    primaryText={obj.firstName + "     " + obj.lastName + "  --   " + obj.emailAddress}
                />
            ));
        }
    }

    //allows the check mark to be applied next to the selections
    judgeMenuClick = (event, index, values) => {
        this.setState({existingJudgeValues: values});
    }
}

export default Events;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/views/Extras Extras.js

86.05% Statements 37/43   57.14% Branches 8/14   93.75% Functions 15/16   86.05% Lines 37/43

```
    import React, { Component } from 'react';
    import {
        Grid, Row, Col,
    } from 'react-bootstrap';

    import InputMask from 'react-input-mask';
    import Button from '../../elements/CustomButton/CustomButton.js';

    import {TextField} from "material-ui";
    import HttpRequest from "../../adapters/httpRequest";
    import constants from "../../utils/constants";
    import NotificationSystem from 'react-notification-system';
    import {style} from "../../variables/Variables";
    import AppBar from 'material-ui/AppBar';
    import MuiThemeProvider from 'material-ui/styles/MuiThemeProvider';
    import Sudoku from 'sudoku-react-component';
    import {Panel, PanelGroup} from 'react-bootstrap';

    class Extras extends React.Component {

        constructor(props) {
6x          super(props);

6x          this.sendEmail = this.sendEmail.bind(this);
6x          this.sendText = this.sendText.bind(this);
6x          this.handleSelect = this.handleSelect.bind(this);

6x          this.state = {
                user : {},
                _notificationSystem : null,
                emailAddress : "",
                phoneNumber : "",
                activeKey: '1'
            };
        }

        componentWillMount() {
6x          var _this = this;
            //This is good shit to remember for later
            // AuthService.getUserEmail();
            // AuthService.getUserRole();
            //This sends our credentials in the header

6x          HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/getUserProfile', 'GET',
                constants.useCredentials(), null).then(function (result) {
                console.log(result);
                _this.setState({
                    user: result.body
                });

            }).catch(function (error) {
6x              console.log(error);
            })
            //here we will get the user type
        
        }
        
        componentDidMount() {
6x          this.setState({_notificationSystem: this.refs.notificationSystem});
        }
        
        notify(message, level, position, autoDismiss) {
6x          this.state._notificationSystem.addNotification({
                title: (<span data-notify="icon" className="pe-7s-door-lock"></span>),
                message: (
                    <div>
                        {message}
                    </div>
                ),
                level: level ? level : 'error',
                position: position ? position : 'tc',
                autoDismiss: autoDismiss ? autoDismiss : 10,
            });
        }
        
        handleSelect(activeKey) {
1x          this.setState({ activeKey });
        }
        
        
        sendEmail(e) {
2x          var body = {};
2x          var _this = this;
        
2x          body.emailAddress = this.state.emailAddress;
2x          _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
    '/sweng500/sendTestEmail', 'POST',
                constants.useCredentials(), body).then(function (result) {
1x
1x              E if (result.status === 200) {
                    _this.notify(
                        "Test email sent",
                        "success",
                        "tc",
                        5
1x                  );
                    _this.setState( { emailAddress : "" } );
                } else if(result.status === 409) {
                    _this.notify(
                        "Could not send email",
                        "error",
                        "tc",
                        10
                    );
                }
        
1x          }).catch(function (error) {
1x              console.log(error);
                _this.notify(
                    "Could not send email",
                    "error",
                    "tc",
                    10
                );
            })
        }

 2x     sendText(e) {
 2x         var body = {};
            var _this = this;

 2x         // Create the user account
 2x         var cleanPhoneNumber = this.state.phoneNumber;
 2x         cleanPhoneNumber = cleanPhoneNumber.replace(/\s/g, '');         // Remove spaces
 2x         cleanPhoneNumber = cleanPhoneNumber.replace(/\(|\)/g,'');       // Remove ( and )
 2x         cleanPhoneNumber = cleanPhoneNumber.replace(/-/g,"");           // Remove -
            cleanPhoneNumber = '+' + cleanPhoneNumber;                      // Add +
 2x
 2x         body.phoneNumber = cleanPhoneNumber;
            _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/sendTestText',
    'POST',
 1x                 constants.useCredentials(), body).then(function (result) {
 1x
                E if (result.status === 200) {
                    _this.notify(
                        "Test text sent",
                        "success",
                        "tc",
 1x                     5
                    );
                    _this.setState( { phoneNumber : "" } );
                } else if(result.status === 409) {
                    _this.notify(
                        "Could not send text",
                        "error",
                        "tc",
                        10
                    );
                }
 1x
 1x         }).catch(function (error) {
                console.log(error);
                _this.notify(
                    "Could not send text",
                    "error",
                    "tc",
                    10
                );
            })
        }
24x
        render() {
            return (
                <div className="content">
                    <NotificationSystem ref="notificationSystem" style={style}/>
                    <MuiThemeProvider>
                        <Grid fluid>
                            <PanelGroup id="group"
                                accordion
                                activeKey={this.state.activeKey}
                                onSelect={this.handleSelect}>
                                <Panel id="email-panel" eventKey="1">
                                    <Panel.Heading  style={{backgroundColor: '#194287'}}>
                                        <Panel.Title toggle style={{color: 'white'}}>
                                            Test Email Functionality
                                        </Panel.Title>
                                    </Panel.Heading>
                                    <Panel.Collapse>
                                        <Panel.Body collapsible>
                                            <TextField
                                                name="emailAddress"
    2x                                          hintText="Email Address"
                                                floatingLabelText="Email Address"
                                                onChange={(event, newValue) => this.setState({emailAddress:
    newValue})}
                                                value={this.state.emailAddress}
                                            />
                                            {    }
                                            <Button
                                                bsStyle="info"
                                                fill
                                                type="submit"
                                                onClick={this.sendEmail}
                                                style={{backgroundColor: '#194287', borderColor : '#194287'}}
                                            >
                                                Send Email
                                            </Button>
                                        </Panel.Body>
                                    </Panel.Collapse>
                                </Panel>
                                <Panel id="text-panel" eventKey="2">
                                    <Panel.Heading  style={{backgroundColor: '#194287'}}>
                                        <Panel.Title toggle style={{color: 'white'}}>
                                            Test SMS Texting Functionality
                                        </Panel.Title>
                                    </Panel.Heading>
                                    <Panel.Collapse>
                                        <Panel.Body collapsible>
    2x                                      <TextField
                                                name="phone"
                                                floatingLabelText="Phone number"
                                                onChange={(event, newValue) => this.setState({phoneNumber:
    newValue})}
                                                value={this.state.phoneNumber}
                                            >
                                                <InputMask mask="9 (999) 999-9999" maskChar="#"
                                                           value={this.state.phoneNumber}/>
                                            </TextField>
                                            {    }
                                            <Button
                                                bsStyle="info"
                                                fill
                                                type="submit"
                                                onClick={this.sendText}
                                                style={{backgroundColor: '#194287', borderColor : '#194287'}}
                                            >
                                                Send Text
                                            </Button>
                                        </Panel.Body>
                                    </Panel.Collapse>
                                </Panel>
                                <Panel id="egg-panel" eventKey="3">
                                    <Panel.Heading  style={{backgroundColor: '#194287'}}>
                                        <Panel.Title toggle style={{color: 'white'}}>
```

```
                                        Easter Egg
                                    </Panel.Title>
                                </Panel.Heading>
                                <Panel.Collapse>
                                    <Panel.Body collapsible>
                                        <div id='sudoku-container' style={{width : '70%',
paddingLeft:'10%', paddingRight:'0%'}}>
                                            <Sudoku />
                                        </div>
                                    </Panel.Body>
                                </Panel.Collapse>
                            </Panel>
                        </PanelGroup>
                    </Grid>
                </MuiThemeProvider>
            </div>
        );
    }
}


export default Extras;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/views/Locations Buildings.js

27.59% Statements 24/87 **13.89%** Branches 5/36 **25.93%** Functions 7/27 **27.59%** Lines 24/87

```
    import React, {Component} from 'react';
    import {Grid, Col, Row, Modal} from 'react-bootstrap';
    import Loader from 'react-loader'
    import HttpRequest from "../../adapters/httpRequest";
    import MuiThemeProvider from 'material-ui/styles/MuiThemeProvider';
    import RaisedButton from 'material-ui/RaisedButton';
    import TextField from 'material-ui/TextField';
    import AppBar from 'material-ui/AppBar';
    import FlatButton from 'material-ui/FlatButton';
    import FontIcon from 'material-ui/FontIcon';
    import Card from '../../components/Cards/Card.js';
    import ReactTable from 'react-table';
    import matchSorter from 'match-sorter';
    import Dialog from 'material-ui/Dialog';
    import constants from "../../utils/constants";
    import NotificationSystem from 'react-notification-system';
    import {style} from "../../variables/Variables";
    import {Map, Marker, GoogleApiWrapper} from 'google-maps-react';


    class Buildings extends Component {
        constructor(props) {
2x          super(props);

2x          this.closeModal = this.closeModal.bind(this);
2x          this.openModal = this.openModal.bind(this);
2x          this.addNotification = this.addNotification.bind(this);

2x          this.divStyle = {
                height: '300px',
                width: '450px',
                position: 'relative',
                overflow: 'scroll'
            };

2x          this.state = {
                loading: false,
                modal: false,
                modalTitle: '',
                modalAction: '',
                modalButton: '',
                editModal: false,
                confirmDialog: false,
                confirmMessage: '',
                deleteID: '',
                buildingID: '',
                building: '',
                longitude: null,
                latitude: null,
                buildingRequired: '',
                buildingList:[],
                _notificationSystem: null
            };
        }

        // When the map is clicked record the latitude and longitude
        onMapClicked = (mapProps, map, event) => {

            this.setState({
                latitude: event.latLng.lat(),
                longitude: event.latLng.lng()
            });
        }

        // If there is a latiude and longitude then display it
        addMarker = () => {
            E if (this.state.latitude !== undefined)
                return(<Marker name={'Current location'} position={{lat: this.state.latitude, lng:
    this.state.longitude}}/>);
        }

        // Display notification
        addNotification(message, level, position, autoDismiss, optionalTitle){
            this.state._notificationSystem.addNotification({
                title: optionalTitle ? optionalTitle : (<span data-notify="icon" className="pe-7s-map">
    </span>),
                message: (
                    <div>
                        {message}
                    </div>
                ),
                level: level ? level : 'info',
                position: position ? position : 'tc',
                autoDismiss: autoDismiss ? autoDismiss : 10,
            });
        }

        // Open the modal
        openModal = (mode) => {

            // Change from add to edit mode
            if (mode.status === "add")
            {
                this.setState({
                    modal: true,
                    modalAction: 'add',
                    modalTitle: "Add Building",
                    modalButton: "Add Building",
                    latitude: null,
                    longitude: null,
                    building: '',
                    buildingRequired: '',
                    markerRequired: '',
                })
            }
            else if (mode.status === "edit")
            {
                this.setState({
                    modal: true,
                    modalAction: 'edit',
                    modalTitle: "Edit Building",
                    modalButton: "Update Building",
                    latitude: mode.lat,
                    longitude: mode.lng,
                    building: mode.building,
                    buildingID: mode.id,
                    buildingRequired: '',
                    markerRequired: '',
                })

                this.addMarker();
            }
        }

        // Close the modal
        closeModal() {
            this.setState({
                modal: false
            })
        }

        // Confrim Delete building
        confirmBuildingDelete = (s) => {
            this.setState({deleteID: s.id});
            this.setState({confirmMessage: "Are you sure you want to delete " + s.building + "?", 
    confirmDialog: true});
        }

        // Close the confirm dialog
        closeConfirmDialog  = () => {
            this.setState({confirmDialog: false});
        }

        // Remove the building
        deleteBuilding =()=> {

            var id = this.state.deleteID;
            var _this = this;

            _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + 
    '/sweng500/removeBuilding/' + id, 'DELETE', constants.useCredentials(), null, true).then(function (result) 
    {

                _this.setState({confirmDialog: false});

                if (result.status === 200) {

                    _this.addNotification(
                        "Success: The building has been deleted.",
                        "info",
                        "tc",
                        6
                    );

                    _this.componentDidMount();
                }

            }).catch(function (error) {
                console.log(error);

                _this.addNotification(
                    "Error: The building has not been deleted.",
                    "error",
                    "tc",
                    6
                );


                _this.setState({confirmDialog: false});
            })
        }

        // Check inputs and adds a new building
        handleSubmit = () => {

            var blankInputs = false;
            var building = this.state.building.trim();
            var lat = this.state.latitude;
            var lng = this.state.longitude;

            // Checks first name
            if (building) {
                this.setState({buildingRequired: undefined})
            }
            else {
                this.setState({buildingRequired: "A building name is required"})
                blankInputs = true;
            }

            // Checks first name
            if (lat !== null && lng !== null) {
                this.setState({markerRequired: undefined})
            }
            else {
                this.setState({markerRequired: "You must identify the building by placing a marker on the map."})
                blankInputs = true;
            }

            // If no information is missing we can add the building
            if (!blankInputs)
            {

                var _this = this;

                var body = {};

                body.building = building;
                body.lat = lat;
                body.lng = lng;

                if (this.state.modalAction === "add") {

                    _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
    '/sweng500/addBuilding', 'POST', constants.useCredentials(), body, true).then(function (result) {

                        if (result.status === 200) {

                            // Post a notification
                            _this.addNotification(
                                "Success: The building has been added.",
                                "info",
                                "tc",
                                6
                            );

                            // Close the modal and show a success message
                            _this.setState({modal: false})

                            // Update the component
                            _this.componentDidMount();
                        }

                    }).catch(function (error) {
                        console.log(error);

                        _this.addNotification(
                            "Error: The building has not been added.",
                            "error",
                            "tc",
                            6
                        );

                        // Close the modal and show a failed message
                        _this.setState({modal: false})

                    })
                }
                else if (this.state.modalAction === "edit")
                {

                    _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
    '/sweng500/updateBuilding/'+ this.state.buildingID, 'POST', constants.useCredentials(), body,
    true).then(function (result) {

                        if (result.status === 200) {

                            // Close the modal and show a success message
                            _this.setState({
                                modal: false,
                            })

                            _this.addNotification(
                                "Success: The building has been updated.",
                                "info",
                                "tc",
                                6
                            );

                            _this.componentDidMount();

                        }

                    }).catch(function (error) {
                        console.log(error);

                        _this.addNotification(
                            "Error: The building has not been updated.",
                            "error",
                            "tc",
                            6
                        );

                        // Close the modal and show a failed message
                        _this.setState({modal: false})

                    })
                }
            }
        }

        // Fetch a list of buildings
        componentDidMount() {
            this.setState({_notificationSystem: this.refs.notificationSystem});

            //Make call out to backend
            var _this = this;

            _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/getBuildings',
    'GET', constants.useCredentials(), null, true).then(function (result) {

                _this.setState({
                    buildingList: result.body,
                    loading: true
                })

            }).catch(function (error) {
                console.log(error);
            })
        }


        // Generate the table if the fetch was successful
        renderIfBuildingsFound() {
            if (this.state.buildingList !== null && Object.keys(this.state.buildingList).length !== 0) {

                const columns = [{
                    Header: 'Building Name',
                    filterMethod: (filter, rows) =>
                        matchSorter(rows, filter.value, { keys: ["building"] }),
                    filterAll: true,
                    accessor: 'building' // String-based value accessors!
                }, {
                    Header: 'Latitude',
                    filterMethod: (filter, rows) =>
                        matchSorter(rows, filter.value, { keys: ["latitude"] }),
                    filterAll: true,
                    accessor: 'lat' // String-based value accessors!
                }, {
                    Header: 'Longitude',
                    filterMethod: (filter, rows) =>
                        matchSorter(rows, filter.value, { keys: ["longitude"] }),
                    filterAll: true,
                    accessor: 'lng' // String-based value accessors!
                }, {
                    Header: 'Actions',
                    accessor: 'menuActions', // String-based value accessors!
                    style:{textAlign:'center'},
                    sortable: false,
                    filterable: false
                }];

                for(let value in this.state.buildingList) {
                    this.state.buildingList[value].status = "edit";
```

```
                    this.state.buildingList[value].position = value;
                    this.state.buildingList[value].menuActions = <div><RaisedButton
                        primary={true} onClick={this.openModal.bind(this,this.state.buildingList[value])}
    label="Edit"/>&nbsp;&nbsp;&nbsp;<RaisedButton
                        secondary={true} onClick=
    {this.confirmBuildingDelete.bind(this,this.state.buildingList[value])} label="Delete"/></div>;
                }

                return(
                    <ReactTable
                        data={this.state.buildingList}
4x                      filterable
                        defaultFilterMethod={(filter, row) =>
                            String(row[filter.id]) === filter.value}
                        columns={columns}
                        defaultPageSize={10}
5x                      className="-striped -highlight"
                        defaultSorted={[{id: "building"}]}
                    />
                )

            }
            else
                return("Error loading building list.")
        }

5x      render() {
5x
            const actions = [
                <FlatButton
                    label="Cancel"
                    primary={true}
                    onClick={this.closeConfirmDialog}/>,
                <FlatButton
                    label="Delete"
                    primary={true}
                    onClick={this.deleteBuilding}/>,
            ];
            var status = {"status" : "add"};
            return (
                <MuiThemeProvider>
                    <div className="content">
                        <NotificationSystem ref="notificationSystem" style={style}/>
                            <Grid fluid>
                                <Row>
                                    <Col md={12}>
                                        <Card
                                            style={{margin:10}}
                                            title="Registered Buildings"
                                            category={
                                                <div>These locations are registered with the Science Olympiad
    system.<br/>They will appear in the registration systems and reports.
                                                    <br/><br/>
                                                    <RaisedButton primary={true} label="Add a new building"
    onClick={this.openModal.bind(this,status)}/>
                                                </div>}
                                            ctTableFullWidth ctTableResponsive
                                            content={
                                        <Loader color="#3498db" loaded={this.state.loading}>
                                            {this.renderIfBuildingsFound()}
                                        </Loader>
                                        }/>
                                    </Col>
                                </Row>
                            </Grid>
                        <Modal show={this.state.modal} onHide={this.closeModal}>
                            <Modal.Header>
                                <Modal.Title>
                                    <AppBar
                                    iconElementRight={<FlatButton label="Close"/>}
                                    showMenuIconButton={false}
                                    onRightIconButtonClick={(event) => this.closeModal()}
                                    title={this.state.modalTitle}/>
                                </Modal.Title>
                            </Modal.Header>
                            <Modal.Body>
                                <div style={{overflow: 'auto'}}>
                                    <TextField
                                        id={"buildingName"}
                                        floatingLabelText="Building Name"
                                        onChange={(event, newValue) => this.setState({building: newValue})}
                                        value={this.state.building}
                                        required={true}
                                        autoFocus={true}
                                        fullWidth={true}
                                        hintText="Enter a name for the building"
                                        errorText={this.state.buildingRequired}/>

                                    <div>Click on the map to place a marker on the build's location.</div>
                                    <br/>
                                    {this.state.markerRequired}

                                    <div id="map" style={{height:300, width:450, marginLeft: 'auto',
    marginRight: 'auto'}}>
                                        <Map
                                        style={this.divStyle}
                                        google={this.props.google}
                                        zoomControl={true}
                                        initialCenter={{
                                            lat: 41.306610,
                                            lng: -76.015437
                                        }}
                                        zoom={16}
                                        onClick={this.onMapClicked}
                                        clickableIcons={false}>
                                            {this.addMarker()}
                                        </Map>
                                    </div>
                                </div>
                            </Modal.Body>
                            <Modal.Footer>
                                <RaisedButton icon={<FontIcon className="pe-7s-close-circle" />} primary={true}
    label="Cancel"
                                          onClick={this.closeModal}/>&nbsp;&nbsp;
                                <RaisedButton icon={<FontIcon className="pe-7s-like2" />} primary={true} label=
    {this.state.modalButton}
                                          onClick={this.handleSubmit}/>
                            </Modal.Footer>
                         </Modal>
                        <Dialog
```

```
                        actions={actions}
                        modal={false}
                        open={this.state.confirmDialog}
                        onRequestClose={this.closeConfirmDialog}>
                        {this.state.confirmMessage}
                    </Dialog>
                </div>
            </MuiThemeProvider>
        );
    }
}

export default GoogleApiWrapper({
    apiKey: "AIzaSyC7xiiV97LyRQd-GB9aBmiJaYFGW5DVIbM"
})(Buildings)
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/views/Locations Rooms.js

82.8% Statements 77/93 **61.11%** Branches 22/36 **56.67%** Functions 17/30 **82.8%** Lines 77/93

```
    import React, {Component} from 'react';
    import {Grid, Col, Row, Modal} from 'react-bootstrap';
    import Loader from 'react-loader'
    import HttpRequest from "../../adapters/httpRequest";
    import MuiThemeProvider from 'material-ui/styles/MuiThemeProvider';
    import RaisedButton from 'material-ui/RaisedButton';
    import TextField from 'material-ui/TextField';
    import AppBar from 'material-ui/AppBar';
    import FlatButton from 'material-ui/FlatButton';
    import FontIcon from 'material-ui/FontIcon';
    import Card from '../../components/Cards/Card.js';
    import ReactTable from 'react-table';
    import matchSorter from 'match-sorter';
    import Dialog from 'material-ui/Dialog';
    import constants from "../../utils/constants";
    import NotificationSystem from 'react-notification-system';
    import {style} from "../../variables/Variables";
    import BuildingSelector from '../../components/Buildings/BuildingSelector';


    class Rooms extends Component {
        constructor(props) {
8x          super(props);
8x          this.closeModal = this.closeModal.bind(this);
8x          this.openModal = this.openModal.bind(this);
8x          this.addNotification = this.addNotification.bind(this);
8x          this.buildingCallback = this.buildingCallback.bind(this);

8x          this.state = {
                loading: false,
                modal: false,
                modalTitle: '',
                modalAction: '',
                modalButton: '',
                editModal: false,
                confirmDialog: false,
                confirmMessage: '',
                roomName: '',
                building: null,
                buildingRequired: null,
                roomCapacity: null,
                buildingError: null,
                deleteID: '',
                roomID: '',
                roomCapacityRequired:'',
                roomList:[],
                buildingList:[],
                _notificationSystem: null
            };
        }

        addNotification(message, level, position, autoDismiss, optionalTitle){
            this.state._notificationSystem.addNotification({
                title: optionalTitle ? optionalTitle : (<span data-notify="icon" className="pe-7s-ribbon">
    </span>),
                message: (
                    <div>
                        {message}
                    </div>
                ),
                level: level ? level : 'info',
                position: position ? position : 'tc',
                autoDismiss: autoDismiss ? autoDismiss : 10,
            });
        }

        buildingCallback(event, index, value) {

            this.setState({
                building: value});
        }

        // Open the modal
        openModal = (mode) => {
2x
            if (mode.status === "add")
1x          {
                this.setState({
                    modal: true,
                    modalAction: 'add',
                    modalTitle: "Add New Room",
                    modalButton: "Add Room",
                    roomCapacity: '',
                    roomName: '',
                    roomNameRequired: '',
                    roomCapacityRequired:'',
                    building: '',
                    buildingRequired: ''
                })
1x          }
            else E if (mode.status === "edit")
1x          {
                this.setState({
                    modal: true,
                    modalAction: 'edit',
                    modalTitle: "Edit Room",
                    modalButton: "Update Room",
                    roomCapacity: mode.capacity,
                    roomName: mode.roomName,
                    building: mode.buildingID,
                    roomID: mode.id,
                    roomNameRequired: '',
                    roomCapacityRequired:''
                })
            }
        }

        // Close the modal
1x      closeModal() {
            this.setState({
                modal: false
            })
        }

```

```
         // Delete room
1x       confirmRoomDelete = (s) => {
1x           this.setState({deleteID: s.id});
             this.setState({confirmMessage: "Are you sure you want to delete " + s.roomName + "?", confirmDialog: true});
         }

         getBuildingName(id)
84x      {

136x         for(let value in this.state.buildingList) {

             I if (this.state.buildingList[value].id === id)
                     return this.state.buildingList[value].building;
84x          }

             return "Invalid Building ID";
         }

         closeConfirmDialog  = () => {
             this.setState({confirmDialog: false});
         }

1x       deleteRoom =()=> {
1x
             var id = this.state.deleteID;
1x           var _this = this;

1x           _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/removeRoom/' + id, 'DELETE', constants.useCredentials(), null).then(function (result) {
1x
                 _this.setState({confirmDialog: false});
1x
               E if (result.status === 200) {

                     _this.addNotification(
                         "Success: The room has been deleted.",
                         "info",
                         "tc",
1x                       6
                     );

                     _this.componentDidMount();
                 }

             }).catch(function (error) {

                 _this.addNotification(
                     "Error: The room has not been deleted.",
                     "error",
                     "tc",
                     6
                 );


                 _this.setState({confirmDialog: false});
             })
         }

3x       // Check inputs and add a new room
```

```
3x      handleSubmit = () => {
3x  
3x          var blankInputs = false;
            var roomName = this.state.roomName.trim();
            var roomCapacity = this.state.roomCapacity;
3x          var buildingID = this.state.building;
2x  
            // Checks first name
            if (roomName) {
1x              this.setState({roomNameRequired: undefined})
1x          }
            else {
                this.setState({roomNameRequired: "A room name or number is required"})
                blankInputs = true;
3x          }
2x  
            // Checks contact name
            if (roomCapacity) {
1x              this.setState({roomCapacityRequired: undefined})
1x          }
            else {
                this.setState({roomCapacityRequired: "A room capacity is required"})
                blankInputs = true;
3x          }
2x  
            // Checks for a building
            if (buildingID) {
1x              this.setState({buildingRequired: undefined})
1x          }
            else {
                this.setState({buildingRequired: "A building is required"})
                blankInputs = true;
3x          }
    
            // If no information is missing we can add the room
2x          if (!blankInputs)
            {
2x  
                var _this = this;
2x  
2x              var body = {};
2x  
                body.roomName = this.state.roomName.trim();
2x              body.capacity = this.state.roomCapacity;
                body.buildingID = this.state.building;
1x  
                if (this.state.modalAction === "add") {
1x  
                    _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
    '/sweng500/addRoom', 'POST', constants.useCredentials(), body).then(function (result) {
1x  
                    E if (result.status === 200) {
    
                        // Post a notification
                        _this.addNotification(
                            "Success: The room has been added.",
                            "info",
                            "tc",
1x                          6
                        );

                        // Close the modal and show a success message
                        _this.setState({modal: false})

                        // Update the component
                        _this.componentDidMount();

                    }

                }).catch(function (error) {

                    _this.addNotification(
                        "Error: The room has not been added.",
                        "error",
                        "tc",
                        6
                    );

                    // Close the modal and show a failed message
                    _this.setState({modal: false})

                })
            }
            else E if (this.state.modalAction === "edit")
            {
                _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
'/sweng500/updateRoom/'+ this.state.roomID, 'POST', constants.useCredentials(), body).then(function
(result) {

                    E if (result.status === 200) {

                        // Close the modal and show a success message
                        _this.setState({
                            modal: false,
                        })

                        _this.addNotification(
                            "Success: The room has been updated.",
                            "info",
                            "tc",
                            6
                        );

                        _this.componentDidMount();

                    }

                }).catch(function (error) {

                    _this.addNotification(
                        "Error: The room has not been updated.",
                        "error",
                        "tc",
                        6
                    );

                    // Close the modal and show a failed message
                    _this.setState({modal: false})

                })
```

```
11x          }
           }
         }
11x
         // Fetch a list of rooms
11x      componentDidMount() {
11x        this.setState({_notificationSystem: this.refs.notificationSystem});

           //Make call out to backend to get a list of rooms
           var _this = this;

           _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/getAllRooms',
       'GET', constants.useCredentials(), null).then(function (result) {
             _this.setState({
               roomList: result.body,
11x            loading: true
             })
11x
           }).catch(function (error) {
           })

           //Make call out to backend to get a list of buildings
           _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
       '/sweng500/getBuildings', 'GET', constants.useCredentials(), null, true).then(function (result) {

             _this.setState({
               buildingList: result.body,
               loading: true
             })

           }).catch(function (error) {
80x          console.log(error);
           })
42x


         }

         // Generate the table if the fetch was successful
         renderIfRoomsFound() {
           if (this.state.roomList !== null && Object.keys(this.state.roomList).length !== 0) {

             const columns = [{
               Header: 'Room Name or Number',
               filterMethod: (filter, rows) =>
                 matchSorter(rows, filter.value, { keys: ["roomName"] }),
               filterAll: true,
               accessor: 'roomName' // String-based value accessors!
             }, {
               Header: 'Building Name',
               filterMethod: (filter, rows) =>
                 matchSorter(rows, filter.value, { keys: ["buildingName"] }),
               filterAll: true,
               accessor: 'buildingName' // String-based value accessors!
             }, {
               Header: 'Room Capacity',
               filterMethod: (filter, rows) =>
                 matchSorter(rows, filter.value, { keys: ["capacity"] }),
               filterAll: true,
               accessor: 'capacity' // String-based value accessors!
42x          }, {
```

```
                    Header: 'Actions',
42x                 accessor: 'menuActions', // String-based value accessors!
                    style:{textAlign:'center'},
84x                 sortable: false,
84x                 filterable: false
84x             }];

                var tempList = this.state.roomList;

                for(let value in tempList) {
42x
                    tempList[value].status = "edit";
                    tempList[value].buildingName =
    this.getBuildingName(this.state.roomList[value].buildingID);
                    tempList[value].menuActions = <div><RaisedButton
                        primary={true} onClick={this.openModal.bind(this,this.state.roomList[value])}
    label="Edit"/>&nbsp;&nbsp;&nbsp;<RaisedButton
                        secondary={true} onClick=
    {this.confirmRoomDelete.bind(this,this.state.roomList[value])} label="Delete"/></div>;
                }

                return(
                    <ReactTable
                        data={tempList}
                        filterable
38x                     defaultFilterMethod={(filter, row) =>
                            String(row[filter.id]) === filter.value}
                        columns={columns}
                        defaultPageSize={10}
                        className="-striped -highlight"
80x                     defaultSorted={[{id: "roomName"}]}
                    />
                )

            }
            else
                return("ERROR LOADING ROOMS")
        }

        render() {

            const actions = [
80x             <FlatButton
80x                 label="Cancel"
                    primary={true}
                    onClick={this.closeConfirmDialog}
                />,
                <FlatButton
                    label="Delete"
                    primary={true}
                    onClick={this.deleteRoom}
                />,
            ];
            var status = {"status" : "add"};
            return (
                <MuiThemeProvider>
                    <div className="content">
                        <NotificationSystem ref="notificationSystem" style={style}/>
                            <Grid fluid>
                                <Row>
                                <Col md={12}>
                                    <Card
                                        style={{margin:10}}
                                        title="Registered Rooms"
                                        category={
                                            <div>These rooms are registered with the Science Olympiad
system.<br/>They are assigned to an existing building.
                                                <br/><br/>
                                                <RaisedButton primary={true} label="Create a new room"
onClick={this.openModal.bind(this,status)}/>
                                            </div>}
                                        ctTableFullWidth ctTableResponsive
                                        content={
                                    <Loader color="#3498db" loaded={this.state.loading}>
                                        {this.renderIfRoomsFound()}
                                    </Loader>
                                    }/>
                                </Col>
                            </Row>
                         </Grid>
                    <Modal show={this.state.modal} onHide={this.closeModal}>
                        <Modal.Header>
                            <Modal.Title> <AppBar
                                iconElementRight={<FlatButton label="Close"/>}
                                showMenuIconButton={false}
                                onRightIconButtonClick={(event) => this.closeModal()}
                                title={this.state.modalTitle}
                            /></Modal.Title>
                        </Modal.Header>
                        <Modal.Body>
                            <TextField
                                id={"roomName"}
                                floatingLabelText="Room Name or Number"
                                onChange={(event, newValue) => this.setState({roomName: newValue})}
                                value={this.state.roomName}
                                required={true}
                                autoFocus={true}
                                fullWidth={true}
                                hintText="Enter a room name or number"
                                errorText={this.state.roomNameRequired}
                            />
                            <br/>
                            <TextField
                                id={"roomCapacity"}
                                floatingLabelText="Room Capacity"
                                onChange={(event, newValue) => this.setState({roomCapacity: newValue})}
                                value={this.state.roomCapacity}
                                required={true}
                                fullWidth={true}
                                hintText="Enter the room's capacity"
                                errorText={this.state.roomCapacityRequired}
                            />
                            <br/>
                            <BuildingSelector selected={this.state.building}
                                              errorMsg={this.state.buildingRequired}
                                              callBack={this.buildingCallback}
                                              labelText={"Building"}
                                              hintText={"Select a building"}/>
                        </Modal.Body>
                        <Modal.Footer>
```

```
                            <RaisedButton icon={<FontIcon className="pe-7s-close-circle" />} primary=
{true} label="Cancel"
                                          onClick={this.closeModal}/>&nbsp;&nbsp;
                            <RaisedButton icon={<FontIcon className="pe-7s-like2" />} primary={true}
label={this.state.modalButton}
                                          onClick={this.handleSubmit}/>
                        </Modal.Footer>
                    </Modal>
                    <Dialog
                        actions={actions}
                        modal={false}
                        open={this.state.confirmDialog}
                        onRequestClose={this.closeConfirmDialog}
                    >
                        {this.state.confirmMessage}
                    </Dialog>
                </div>
            </MuiThemeProvider>
        );
    }
}

export default Rooms;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/views/Schools Schools.js

85.87% Statements 79/92 **61.76% Branches** 21/34 **60.71% Functions** 17/28 **85.87% Lines** 79/92

```
    import React, {Component} from 'react';
    import {Grid, Col, Row, Modal} from 'react-bootstrap';
    import Loader from 'react-loader'
    import HttpRequest from "../../adapters/httpRequest";
    import MuiThemeProvider from 'material-ui/styles/MuiThemeProvider';
    import RaisedButton from 'material-ui/RaisedButton';
    import TextField from 'material-ui/TextField';
    import AppBar from 'material-ui/AppBar';
    import FlatButton from 'material-ui/FlatButton';
    import FontIcon from 'material-ui/FontIcon';
    import Card from '../../components/Cards/Card.js';
    import InputMask from 'react-input-mask';
    import ReactTable from 'react-table';
    import matchSorter from 'match-sorter';
    import Dialog from 'material-ui/Dialog';
    import constants from "../../utils/constants";
    import NotificationSystem from 'react-notification-system';
    import {style} from "../../variables/Variables";


    class Schools extends Component {
        constructor(props) {
12x         super(props);
12x         this.closeModal = this.closeModal.bind(this);
12x         this.openModal = this.openModal.bind(this);
12x         this.addNotification = this.addNotification.bind(this);

12x         this.state = {
                loading: false,
                modal: false,
                modalTitle: '',
                modalAction: '',
                modalButton: '',
                editModal: false,
                confirmDialog: false,
                confirmMessage: '',
                formattedPhone: '',
                deleteID: '',
                schoolID: '',
                schoolName: '',
                schoolContactPhone: '',
                schoolContactName:'',
                schoolNameRequired: '',
                schoolContactPhoneRequired: '',
                schoolContactNameRequired:'',
                schoolList:[],
                _notificationSystem: null

            };
        }

        addNotification(message, level, position, autoDismiss, optionalTitle){
            this.state._notificationSystem.addNotification({
                title: optionalTitle ? optionalTitle : (<span data-notify="icon" className="pe-7s-home">
    </span>),
                message: (
                    <div>
                        {message}
                    </div>
                ),
                level: level ? level : 'info',
                position: position ? position : 'tc',
                autoDismiss: autoDismiss ? autoDismiss : 10,
            });
        }

        // Open the modal
        openModal = (mode) => {
2x
            if (mode.status === "add")
1x          {
                this.setState({
                    modal: true,
                    modalAction: 'add',
                    modalTitle: "Add New School",
                    modalButton: "Add School",
                    schoolContactName: '',
                    schoolContactPhone: '',
                    schoolName: '',
                    schoolNameRequired: '',
                    schoolContactPhoneRequired: '',
                    schoolContactNameRequired:''
                })
1x          }
            else E if (mode.status === "edit")
1x          {
                this.setState({
                    modal: true,
                    modalAction: 'edit',
                    modalTitle: "Edit School",
                    modalButton: "Update School",
                    schoolContactName: mode.schoolContactName,
                    schoolContactPhone: mode.schoolContactPhone,
                    schoolName: mode.schoolName,
                    schoolID: mode.id,
                    schoolNameRequired: '',
                    schoolContactPhoneRequired: '',
                    schoolContactNameRequired:''
                })
            }
        }

        // Close the modal
1x      closeModal() {
            this.setState({
                modal: false
            })
        }

        // Delete school
1x      confirmSchoolDelete = (s) => {
1x          this.setState({deleteID: s.id});
            this.setState({confirmMessage: "Are you sure you want to delete" + s.schoolName + "?",
    confirmDialog: true});
```

```
        }

        closeConfirmDialog  = () => {
            this.setState({confirmDialog: false});
        }

1x      deleteSchool =()=> {
1x
            var id = this.state.deleteID;
1x          var _this = this;

1x          _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/removeSchool/'
    + id, 'DELETE', constants.useCredentials(), null).then(function (result) {
1x
                _this.setState({confirmDialog: false});
1x
                E if (result.status === 200) {

                    _this.addNotification(
                        "Success: The school has been deleted.",
                        "info",
                        "tc",
1x                      6
                    );

                    _this.componentDidMount();
                }

            }).catch(function (error) {

                _this.addNotification(
                    "Error: The school has not been deleted.",
                    "error",
                    "tc",
                    6
                );


                _this.setState({confirmDialog: false});
            })
        }

3x      // Check inputs and add a new school
3x      handleSubmit = () => {
3x
3x          var blankInputs = false;
            var schoolName = this.state.schoolName.trim();
            var schoolContactName = this.state.schoolContactName.trim();
3x          var schoolContactPhone = this.state.schoolContactPhone.trim();
2x
            // Checks first name
            if (schoolName) {
1x              this.setState({schoolNameRequired: undefined})
1x          }
            else {
                this.setState({schoolNameRequired: "School name is required"})
                blankInputs = true;
3x          }
2x
            // Checks contact name
```

```
             if (schoolContactName) {
1x               this.setState({schoolContactNameRequired: undefined})
1x           }
             else {
                 this.setState({schoolContactNameRequired: "School contact name is required"})
                 blankInputs = true;
3x           }
2x
             // Checks first name
             if (schoolContactPhone) {
1x               this.setState({schoolContactPhoneRequired: undefined})
1x           }
             else {
                 this.setState({schoolContactPhoneRequired: "School phone number is required"})
                 blankInputs = true;
3x           }

             // If no information is missing we can add the school
2x           if (!blankInputs)
             {
2x
                 var _this = this;
2x
2x               var body = {};
2x
2x               var cleanPhoneNumber = this.state.schoolContactPhone.trim();
                 cleanPhoneNumber = cleanPhoneNumber.replace(/\s/g, '');         // Remove spaces
2x               cleanPhoneNumber = cleanPhoneNumber.replace(/\(|\)/g, '');       // Remove ( and )
2x               cleanPhoneNumber = cleanPhoneNumber.replace(/-/g, "");           // Remove -
2x
                 body.schoolName = this.state.schoolName.trim();
2x               body.schoolContactName = this.state.schoolContactName.trim();
                 body.schoolContactPhone = cleanPhoneNumber;
1x
                 if (this.state.modalAction === "add") {
1x
                     _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
     '/sweng500/addSchool', 'POST', constants.useCredentials(), body).then(function (result) {
1x
                       E if (result.status === 200) {

                             // Post a notification
                             _this.addNotification(
                                 "Success: The school has been added.",
                                 "info",
                                 "tc",
1x                               6
                             );

1x                           // Close the modal and show a success message
                             _this.setState({modal: false})

                             // Update the component
                             _this.componentDidMount();

                         }

                     }).catch(function (error) {

                         _this.addNotification(
                             "Error: The school has not been added.",
                             "error",
                             "tc",
                             6
                         );

                         // Close the modal and show a failed message
 1x                      _this.setState({modal: false})

 1x                  })
                 }
 1x              else E if (this.state.modalAction === "edit")
                 {
                     _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
 1x  '/sweng500/updateSchool/'+ this.state.schoolID, 'POST', constants.useCredentials(), body).then(function
     (result) {

                         E if (result.status === 200) {
 1x
                             // Close the modal and show a success message
                             _this.setState({
                                 modal: false,
                             })

                             _this.addNotification(
 1x                              "Success: The school has been updated.",
                                 "info",
                                 "tc",
                                 6
                             );

                             _this.componentDidMount();

                         }

                     }).catch(function (error) {

                         _this.addNotification(
                             "Error: The school has not been updated.",
                             "error",
                             "tc",
                             6
                         );

                         // Close the modal and show a failed message
                         _this.setState({modal: false})

                     })
 14x             }
                 }
             }
 14x
         // Fetch a list of schools
 14x     componentDidMount() {
 14x         this.setState({_notificationSystem: this.refs.notificationSystem});

             //Make call out to backend
             var _this = this;

             _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/getSchools',
     'GET', constants.useCredentials(), null).then(function (result) {
                 _this.setState({
                     schoolList: result.body,
                     loading: true
                 })

 80x         }).catch(function (error) {
 80x         })
 80x     }

 80x     // Reformat the phone number
         formatPhoneNumber(str)
         {
             var part1 = str.substr(1, 3)
             var part2 = str.substr(4, 3)
 85x         var part3 = str.substr(7, 4)

 40x         return("(" + part1 + ") " + part2 + "-" + part3)
         }

         // Generate the table if the fetch was successful
         renderIfSchoolsFound() {
             if (this.state.schoolList !== null && Object.keys(this.state.schoolList).length !== 0) {

                 const columns = [{
                     Header: 'School Name',
                     filterMethod: (filter, rows) =>
                         matchSorter(rows, filter.value, { keys: ["schoolName"] }),
                     filterAll: true,
                     accessor: 'schoolName' // String-based value accessors!
                 }, {
                     Header: 'School Contact',
                     filterMethod: (filter, rows) =>
                         matchSorter(rows, filter.value, { keys: ["schoolContactName"] }),
                     filterAll: true,
                     accessor: 'schoolContactName' // String-based value accessors!
                 }, {
                     Header: 'School Phone Number',
                     filterMethod: (filter, rows) =>
                         matchSorter(rows, filter.value, { keys: ["schoolContactPhone"] }),
                     filterAll: true,
                     accessor: 'formattedPhone' // String-based value accessors!
                 }, {
 40x                 Header: 'Actions',
                     accessor: 'menuActions', // String-based value accessors!
 80x                 style:{textAlign:'center'},
 80x                 sortable: false,
 80x                 filterable: false
                 }];

                 for(let value in this.state.schoolList) {

 40x                 this.state.schoolList[value].formattedPhone =
     this.formatPhoneNumber(this.state.schoolList[value].schoolContactPhone);
                     this.state.schoolList[value].status = "edit";
                     this.state.schoolList[value].menuActions = <div><RaisedButton
                         primary={true} onClick={this.openModal.bind(this,this.state.schoolList[value])}
     label="Edit"/>&nbsp;&nbsp;&nbsp;<RaisedButton
                         secondary={true} onClick=
     {this.confirmSchoolDelete.bind(this,this.state.schoolList[value])} label="Delete"/></div>;
```

```
                }

                return(
                    <ReactTable
                        data={this.state.schoolList}
                        filterable
                        defaultFilterMethod={(filter, row) =>
45x                         String(row[filter.id]) === filter.value}
                        columns={columns}
                        defaultPageSize={10}
                        className="-striped -highlight"
                        defaultSorted={[{id: "schoolName"}]}
85x                 />
                )

            }
            else
                return("ERROR LOADING SCHOOLS")
        }

        render() {

            const actions = [
                <FlatButton
85x                 label="Cancel"
85x                 primary={true}
                    onClick={this.closeConfirmDialog}
                />,
                <FlatButton
                    label="Delete"
                    primary={true}
                    onClick={this.deleteSchool}
                />,
            ];
            var status = {"status" : "add"};
            return (
                <MuiThemeProvider>
                    <div className="content">
                        <NotificationSystem ref="notificationSystem" style={style}/>
                            <Grid fluid>
                                <Row>
                                    <Col md={12}>
                                        <Card
                                            style={{margin:10}}
                                            title="Registered Schools"
                                            category={
                                                <div>These schools are registered with the Science Olympiad system.<br/>They will appear in the registration systems and reports.
                                                    <br/><br/>
                                                    <RaisedButton primary={true} label="Create a new school" onClick={this.openModal.bind(this,status)}/>
                                                </div>}
                                            ctTableFullWidth ctTableResponsive
                                            content={
                                        <Loader color="#3498db" loaded={this.state.loading}>
                                            {this.renderIfSchoolsFound()}
                                        </Loader>
                                        }/>
                                    </Col>
                                </Row>
```

```
                             </Grid>
                        <Modal show={this.state.modal} onHide={this.closeModal}>
1x                          <Modal.Header>
                                <Modal.Title> <AppBar
                                    iconElementRight={<FlatButton label="Close"/>}
                                    showMenuIconButton={false}
                                    onRightIconButtonClick={(event) => this.closeModal()}
                                    title={this.state.modalTitle}
                                /></Modal.Title>
                            </Modal.Header>
                            <Modal.Body>
                                <TextField
                                    id={"schoolName"}
                                    floatingLabelText="School Name"
1x                                  onChange={(event, newValue) => this.setState({schoolName: newValue})}
                                    value={this.state.schoolName}
                                    required={true}
                                    autoFocus={true}
                                    fullWidth={true}
                                    hintText="Enter a name for the school"
                                    errorText={this.state.schoolNameRequired}
                                />
                                <br/>
                                <TextField
                                    id={"schoolContactName"}
1x                                  floatingLabelText="Contact Name"
                                    onChange={(event, newValue) => this.setState({schoolContactName:
   newValue})}

                                    value={this.state.schoolContactName}
                                    required={true}
                                    fullWidth={true}
                                    hintText="Enter a first and last name for the school contact"
                                    errorText={this.state.schoolContactNameRequired}
                                />
                                <br/>
                                <TextField
                                    id={"schoolContactPhone"}
                                    floatingLabelText="Phone Number"
                                    onChange={(event, newValue) => this.setState({schoolContactPhone:
   newValue})}

                                    value={this.state.schoolContactPhone}
                                    required={true}
                                    fullWidth={true}
                                    hintText="Enter the school's phone numer"
                                    errorText={this.state.schoolContactPhoneRequired}
                                >
                                    <InputMask mask="1 (999) 999-9999" maskChar="#" value=
   {this.state.schoolContactPhone}/>
                                </TextField>
                            </Modal.Body>
                            <Modal.Footer>
                                <RaisedButton icon={<FontIcon className="pe-7s-close-circle" />} primary=
   {true} label="Cancel"
                                              onClick={this.closeModal}/>&nbsp;&nbsp;
                                <RaisedButton icon={<FontIcon className="pe-7s-like2" />} primary={true}
   label={this.state.modalButton}
                                              onClick={this.handleSubmit}/>
                            </Modal.Footer>
                        </Modal>
                        <Dialog
                            actions={actions}
                            modal={false}
                            open={this.state.confirmDialog}
                            onRequestClose={this.closeConfirmDialog}
                        >
                            {this.state.confirmMessage}
                        </Dialog>
                    </div>
                </MuiThemeProvider>
            );
        }
   }

   export default Schools;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/views/Teams StudentTeamCreator.js

100% Statements `10/10` **100%** Branches `0/0` **100%** Functions `7/7` **100%** Lines `10/10`

```
    import React, {Component} from 'react';
    import {MuiThemeProvider, Popover, Menu, MenuItem, RaisedButton} from 'material-ui';
    import {Panel, PanelGroup} from 'react-bootstrap';
    import {Grid, Row, Col} from 'react-bootstrap';
    import StudentAdder from "../../components/Students/StudentAdder";
    import TeamAdder from "../../components/Teams/TeamAdder";

    class StudentTeamCreator extends Component {
        constructor(props) {
3x          super(props);

3x          this.state = {
                popOverOpen: false,
                addPanel: "",
            };

3x          this.handleAddClick = this.handleAddClick.bind(this);
3x          this.toggleAdd = this.toggleAdd.bind(this);
        }

        handleAddClick(event) {
1x          event.preventDefault();

1x          this.setState({
                popOverOpen: true,
                anchorEl: event.currentTarget
            });
        }


        toggleAdd(type) {
2x          this.setState({popOverOpen: false, addPanel: type});
        }

        render() {
6x          return (
                <MuiThemeProvider>
                    <div style={{marginLeft: '10%', marginRight: '10%', width: '80%', maxWidth: 800, textAlign:
    'center', display: 'inline-block'}}>
                        <RaisedButton primary={true} onClick={this.handleAddClick} label="Create Student/Team"/>
                        <Popover
                            style={{maxWidth: '200px', width: '200px'}}
                            open={this.state.popOverOpen}
                            anchorEl={this.state.anchorEl}
                            anchorOrigin={{horizontal: 'left', vertical: 'bottom'}}
                            targetOrigin={{horizontal: 'left', vertical: 'top'}}
                        >
1x                          <Menu>
1x                              <MenuItem style={{textAlign: 'center', width: '200px'}} primaryText="Create
    Student" onClick={ () => this.toggleAdd("student")}/>
                                <MenuItem style={{textAlign: 'center', width: '200px'}} primaryText="Create
    Team" onClick={ () => this.toggleAdd("team")}/>
                            </Menu>
                        </Popover>
```

```

                                <PanelGroup style={{ width: '80%', marginLeft: '10%', marginRight: '10%',
textAlign: 'center'}}
                                    id="add-collapsible-panel-group"
                                    accordion
                                    activeKey={this.state.addPanel}
                                    onSelect={() => {}}
                                    key="add-collapsible-panel-group"
                                >
                                    <Panel id="add-user-collapsible-panel" eventKey={"student"} style=
{{border: 0}}>
                                        <Panel.Body collapsible>
                                            <StudentAdder togglePanel={this.toggleAdd} updateTable=
{this.props.updateTable} addNotification={this.props.addNotification}/>
                                        </Panel.Body>
                                    </Panel>
                                    <Panel id="add-team-collapsible-panel" eventKey={"team"} style=
{{border:0}}>
                                        <Panel.Body collapsible>
                                            <TeamAdder togglePanel={this.toggleAdd} updateTable=
{this.props.updateTable} addNotification={this.props.addNotification}/>
                                        </Panel.Body>
                                    </Panel>
                                </PanelGroup>

                </div>
            </MuiThemeProvider>
        )
    }

}

export default StudentTeamCreator;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/views/Teams TeamManagement.js

100% Statements 9/9 **100%** Branches 8/8 **100%** Functions 5/5 **100%** Lines 9/9

```
    import React, {Component} from 'react';
    import {MuiThemeProvider} from 'material-ui';
 
    import TeamViewer from "../../components/Teams/TeamViewer";
    import StudentTeamCreator from "./StudentTeamCreator";
    import {style} from "../../variables/Variables";
 
    import NotificationSystem from 'react-notification-system';
 
    class TeamManagement extends Component {
        constructor(props) {
3x          super(props);
 
3x          this.state = {
                tableUpdateToggler: false,
                _notificationSystem: null
            }
 
3x          this.updateTable = this.updateTable.bind(this);
3x          this.addNotification = this.addNotification.bind(this);
        }
 
        updateTable() {
1x          this.setState({
                tableUpdateToggler: !this.state.tableUpdateToggler
            })
        }
 
        addNotification(message, level, position, autoDismiss) {
4x          if(this.state._notificationSystem) {
2x              this.state._notificationSystem.addNotification({
                    title: (<span data-notify="icon" className="pe-7s-info"></span>),
                    message: (
                        <div>
                            {message}
                        </div>
                    ),
                    level: level ? level : 'info',
                    position: position ? position : 'tr',
                    autoDismiss: autoDismiss ? autoDismiss : 5,
                });
            }
        }
 
        componentDidMount() {
4x          this.setState({_notificationSystem: this.refs.notificationSystem});
        }
        render() {
10x         return (
                <MuiThemeProvider>
                    <div className="content" style={{textAlign: 'center'}}>
                        <NotificationSystem ref="notificationSystem" style={style}/>
                        <StudentTeamCreator updateTable={this.updateTable} addNotification=
    {this.addNotification}/>
```

```
                    <TeamViewer updateTable={this.updateTable} tableUpdateToggler=
{this.state.tableUpdateToggler} addNotification={this.addNotification} />
                </div>
            </MuiThemeProvider>
        )
    }

}

export default TeamManagement;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)

All files / src/views/UserProfile UserProfile.js

100% Statements 96/96 **96.97% Branches** 32/33 **100% Functions** 28/28 **100% Lines** 94/94

```
     import React, { Component } from 'react';
     import {
         Grid, Row, Col,
     } from 'react-bootstrap';

     import {UserCard} from '../../components/Cards/UserCard.js';
     import InputMask from 'react-input-mask';
     import Button from '../../elements/CustomButton/CustomButton.js';

     import {TextField} from "material-ui";
     import HttpRequest from "../../adapters/httpRequest";
     import constants from "../../utils/constants";
     import AuthService from "../../utils/AuthService";
     import NotificationSystem from 'react-notification-system';
     import {style} from "../../variables/Variables";
     import AppBar from 'material-ui/AppBar';
     import MuiThemeProvider from 'material-ui/styles/MuiThemeProvider';
     import PasswordField from 'material-ui-password-field';

     class UserProfile extends Component {

         constructor(props) {
18x          super(props);

18x          this.updateProfile = this.updateProfile.bind(this);
18x          this.changePassword = this.changePassword.bind(this);
18x          this.validPassword = this.validPassword.bind(this);
18x          this.cleanPhone = this.cleanPhone.bind(this);
18x          this.notify = this.notify.bind(this);

18x          this.state = {
                 user : {firstName: "", lastName: "", phoneNumber: "", receiveText: false, minutesBeforeEvent:
     "0"},
                 _notificationSystem : null,
                 password: "",
                 currentPassword : "",
                 newPassword : "",
                 confirmPassword : "",
                 imageUrl : "",
                 desc : "",
                 code: 0
             };

         }

18x      componentWillMount() {
             var _this = this;
             //This is good shit to remember for later
             // AuthService.getUserEmail();
             // AuthService.getUserRole();
             //This sends our credentials in the header
18x
             HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/getUserProfile', 'GET',
10x              constants.useCredentials(), null).then(function (result) {
```

```
                _this.setState({
                    user: result.body,
                    desc: result.body.school ? "School: " + result.body.school.schoolName : ""
                })

 8x         }).catch(function (error) {
                console.log(error);
            })
18x         //here we will get the user type
            const userType = AuthService.getUserRole();
18x 
14x         if (userType === "ADMIN") {
 4x             this.setState({imageUrl :
 1x "https://telegram.org/file/811140509/b45/dQTLEwKZ9gs.22232.gif/4580677d940852f30e"});
            } else if (userType === "COACH") {
                this.setState({
 3x                 imageUrl : "https://baseballmomstuff.com/wp-content/uploads/2016/02/coach-cartoon.jpg"
 1x             });
 2x         } else if (userType === "JUDGE") {
 1x             this.setState({imageUrl : "https://www.how-to-draw-funny-cartoons.com/image-files/cartoon-
    judge-010.jpg"});
            } else if (userType === "STUDENT") {
                this.setState({
                    imageUrl : "https://classroomclipart.com/images/gallery/Clipart/Science/TN_female-student-
    holding-flask-and-test-tube-in-science-lab-science-clipart.jpg"
                });
19x         }
        }

        componentDidMount() {
 8x         this.setState({_notificationSystem: this.refs.notificationSystem});
 2x     }

        notify(message, level, position, autoDismiss) {
            if(this.state._notificationSystem) {
                this.state._notificationSystem.addNotification({
                    title: (<span data-notify="icon" className="pe-7s-door-lock"></span>),
                    message: (
                        <div>
                            {message}
                        </div>
                    ),
                    level: level ? level : 'error',
                    position: position ? position : 'tc',
                    autoDismiss: autoDismiss ? autoDismiss : 10,
                });
 3x         }
 3x     }
 3x 
 3x     cleanPhone(cleanPhoneNumber) {
            cleanPhoneNumber = cleanPhoneNumber.replace(/\s/g, '');         // Remove spaces
 3x         cleanPhoneNumber = cleanPhoneNumber.replace(/\(|\)/g,'');       // Remove ( and )
            cleanPhoneNumber = cleanPhoneNumber.replace(/-/g,"");           // Remove -
            cleanPhoneNumber = '+' + cleanPhoneNumber;                      // Add +

 4x         return cleanPhoneNumber;
 4x     }

 4x     updateProfile(e) {
            var body = {};
```

```
3x          var _this = this;

            if (this.state.password !== "") {
3x
                body.password = this.state.password;

                //check to ensure the password matches
2x              _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() + '/sweng500/validate',
    'POST',
                    constants.useCredentials(), body, true).then(function (result) {
2x
                    //Create deep copy of user object
2x                  body.user = JSON.parse(JSON.stringify(_this.state.user));

1x                  //update the user, first clean the phone number
                    body.user.phoneNumber = _this.cleanPhone(body.user.phoneNumber);

                      _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
    '/sweng500/updateUser', 'POST', constants.useCredentials(), body.user, true).then(function (result) {

                          _this.notify(
1x                            "Profile Successfully Updated",
                              "success",
                              "tc",
                              5
                          );
1x
1x                        _this.setState({
                              password: ""
                          })

                      }).catch(function (error) {
                          console.log(error);
1x                        _this.notify(
                              "Could not update profile at this time. Please try again later.",
                              "error",
1x                            "tc",
1x                            5
                          );
                        _this.setState({code : 11});
                      })
                }).catch(function (error) {
                    console.log(error);
1x                  _this.notify(
                        "Please provide the correct password in order to update your profile.",
                        "error",
1x                      "tc",
                        5
                    )
                    _this.setState({code : 14})
                })
            } else {
1x              _this.notify(
                    "Please enter your current password",
                    "error",
                    "tc",
                    5
                );
7x              _this.setState({code : 1})
5x          }
```

```
5x      }
5x  
5x      // Check if a passowrd is valid. 8 characters, uppercase, lowercase, and numbers
5x      validPassword(text) {
            if (text.length < 8) return false;
4x          var hasUpperCase = /[A-Z]/.test(text);
            var hasLowerCase = /[a-z]/.test(text);
            var hasNumbers = /\d/.test(text);
            var hasNonalphas = /\W/.test(text);
6x          if (hasUpperCase + hasLowerCase + hasNumbers + hasNonalphas < 3) return false;
6x  
            return true;
        }
6x  
        changePassword(e) {
            var body = {};
5x          var _this = this;
    
4x          //check to ensure that current password is filled in
            if (this.state.currentPassword !== "" && this.state.newPassword !== "" &&
3x  this.state.confirmPassword !== "") {
    
                //check to make sure current password is correct
3x              if (this.state.newPassword === this.state.confirmPassword) {
    
                    if (this.validPassword(this.state.newPassword)) {
    
2x                      body.password = this.state.currentPassword;
    
2x                      //ensure their current password is correct
                        _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
    '/sweng500/validate', 'POST',
1x                          constants.useCredentials(), body).then(function (result) {
    
                            //then change the password
                            body.password = _this.state.newPassword;
    
                            _this.serverRequest = HttpRequest.httpRequest(constants.getServerUrl() +
    '/sweng500/changePassword', 'POST',
1x                              constants.useCredentials(), body).then(function (result) {
    
                                _this.notify(
1x                                  "Password changed successfully.",
                                    "success",
1x                                  "tc",
                                    5
                                );
    
                                _this.setState({currentPassword: "", newPassword: "", confirmPassword: "",
    code: 8})
1x  
                            }).catch(function (error) {
                                console.log(error);
1x  
                                _this.notify(
                                    "Could not update at this time.  Please try again later.",
                                    "error",
                                    "tc",
                                    7
                                );
```

```
 1x                             _this.setState({code : 10})
                            })
                        }).catch(function (error) {
                            _this.notify(
 1x                             "Incorrect current password.  Please provide your current password",
                                "error",
                                "tc",
                                5
                            );

 1x                         _this.setState({code : 7})
                        })
                    } else {
                        //put up a notify
 1x                     this.notify(
                            "ERROR: Your password must be 8 or more characters, contain capital letters, lower
case letters, and at least one number.",
                            "error",
                            "tc",
                            10
 1x                     );
                        this.setState({code : 4})
                    }
                } else {
 1x                 //put up a notify
                    this.notify(
                        "New passwords do not match",
                        "error",
                        "tc",
                        5
                    );
                    this.setState({code : 3})
                }
            }
85x         else {
                this.notify(
                    "Please enter your current password, a new password, and confirm the new password.",
                    "error",
                    "tc",
                    7
                );
            }
        }

        render() {
            return (
                <div className="content">
                    <NotificationSystem ref="notificationSystem" style={style}/>
                    <MuiThemeProvider>
                        <Grid fluid>
                            <Row className="show-grid">
                                <Col md={4}>
                                    <UserCard
                                        bgImage="https://ununsplash.imgix.net/photo-1431578500526-
4d9613015464?fit=crop&fm=jpg&h=300&q=75&w=400"
                                        //avatar={avatar}
                                        avatar= {this.state.imageUrl}
                                        name= {this.state.user.firstName + " " + this.state.user.lastName}
                                        userName={this.state.user.emailAddress}
                                        //maybe in description put the events they are doing? or school
```

```
   information?
                                       description={<span> {this.state.desc}  </span> }
                                   />
1x                             </Col>
1x                         </Row>
1x                         <Row className="show-grid">
                               <Col md={12}>
                                   <AppBar showMenuIconButton={false} title="User Profile"/>
                               </Col>
                           </Row>
                           <Row className="show-grid">
                               <Col md={4}>
                                   <TextField
                                       name="fname"
1x                                     hintText="First name"
1x                                     floatingLabelText="First name"
1x                                     onChange = {(event, newValue) => { var ryanRocks = this.state.user;
                                           ryanRocks.firstName = event.target.value;
                                           this.setState({user : ryanRocks})}}
                                       value={this.state.user.firstName}
                                   />
                               </Col>
                               <Col md={6}>
                                   <TextField
                                       name="lname"
                                       hintText="Last name"
1x                                     floatingLabelText="Last name"
1x                                     onChange ={ (event, newValue) => {var ryanRocks = this.state.user;
1x                                         ryanRocks.lastName = event.target.value;
                                           this.setState({user : ryanRocks})}}
                                       value={this.state.user.lastName}
                                   />
                               </Col>
                           </Row>
                           <Row className="show-grid">
                               <Col md = {4}>
                                   <TextField
                                       name="phone"
                                       floatingLabelText="Phone number"
                                       onChange = {(event, newValue) => {var ryanRocks = this.state.user;
1x                                         ryanRocks.phoneNumber = event.target.value;
1x                                         this.setState({user : ryanRocks})}}
1x                                     value={this.state.user.phoneNumber}
                                   >
                                       <InputMask mask="9 (999) 999-9999" maskChar="#"
                                                  value={this.state.user.phoneNumber}/>
                                   </TextField>
                               </Col>
                               <Col md={4}>
                                   <TextField
                                       name="timeBeforeEvent"
                                       hintText="Time Before Event"
                                       floatingLabelText="Time Before Event"
                                       onChange = {(event, newValue) => {var ryanRocks = this.state.user;
                                           ryanRocks.minutesBeforeEvent = event.target.value;
                                           this.setState({user : ryanRocks})}}
                                       value={this.state.user.minutesBeforeEvent}
                                   >
                                       <InputMask mask="99" maskChar=""
                                                  value={this.state.user.minutesBeforeEvent}/>
```

```
1x                                      </TextField>
                                    </Col>
                                </Row>
                                <Row className="show-grid">
                                    <Col md={4}>

                                    </Col>
                                    <Col md={6}>
                                        <PasswordField
                                            name="currentPassword"
                                            id="password"
1x                                          // hintText="Current Password"
                                            floatingLabelText="Current Password"
1x                                          onChange={(event, newValue) => this.setState({password: newValue})}
1x                                          value={this.state.password}
                                        />
                                    </Col>
                                </Row>
                                <Row className="show-grid">
                                    <Col md={4}>
                                        <input
                                            name="receiveText"
                                            type="checkbox"
                                            checked={this.state.user.receiveText}
                                            onChange={(event, newValue) => {var ryanRocks = this.state.user;
                                                // console.log(event.target.value);
                                                ryanRocks.receiveText = !ryanRocks.receiveText; //; ?  true :
    false);
                                                this.setState({user : ryanRocks})}} />
                                        <label>
                                            Receive Text Messages
                                        </label>
                                    </Col>
                                    <Col md={2}>
                                        <Button
                                            bsStyle="info"
                                            pullRight
                                            fill
                                            type="submit"
                                            onClick={this.updateProfile}
                                        >
                                            Update Profile
                                        </Button>
                                    </Col>
                                </Row>
                                <Row className={"show-grid"}>
1x                                  <label>

                                    </label>
                                </Row>
                                <Row className={"show-grid"}>
                                    <AppBar showMenuIconButton={false} title="Change Password"/>
                                </Row>
                                <Row className={"show-grid"}>
                                    <Col md={12}>
                                        <PasswordField
                                            name="currentPassword"
                                            id="currentPassword"
1x                                          // hintText="Current Password"
                                            floatingLabelText="Current Password"
```

```
                                                 onChange={(event, newValue) => this.setState({currentPassword:
     newValue})}
                                                 value={this.state.currentPassword}
                                             />
                                         </Col>
                                     </Row>
                                     <Row className={"show-grid"}>
                                         <Col md={12}>
                                             <PasswordField
                                                 name="newPassword"
1x                                               id="newPassword"
                                                 // hintText="New Password"
                                                 floatingLabelText="New Password"
                                                 onChange={(event, newValue) => this.setState({newPassword: newValue})}
                                                 value={this.state.newPassword}
                                             />
                                         </Col>
                                     </Row>
                                     <Row className={"show-grid"}>
                                         <Col md={12}>
                                             <PasswordField
                                                 name="confirmPassword"
                                                 id="confirmPassword"
                                                 // hintText="Confirm Password"
                                                 floatingLabelText="Confirm Password"
                                                 onChange={(event, newValue) => this.setState({confirmPassword:
     newValue})}
                                                 value={this.state.confirmPassword}
                                             />
                                         </Col>
                                     </Row>
                                     <Row className={"show-grid"}>
                                         <Col md={6}>
                                             <Button
                                                 bsStyle="info"
                                                 pullRight
                                                 fill
                                                 type="submit"
                                                 onClick={this.changePassword}
                                             >
                                                 Change Password
                                             </Button>
                                         </Col>
                                     </Row>
                                 </Grid>
                             </MuiThemeProvider>
                         </div>
                     );
         }
     }



     export default UserProfile;
```

Code coverage generated by istanbul at Mon Apr 16 2018 22:47:35 GMT-0400 (EDT)