```
angular.module('ClassMaticApp', ['ClassMaticApp.controllers', 'ClassMaticApp.directives', 'ClassMaticApp.services', 'ngAnimate']);

//main Angular module
angular.module('ClassMaticApp.controllers', []).controller('mainController', function ($scope, $rootScope, $interval) {

    //accessible from any scope as long as the $rootScope is injected
    $rootScope.fields = {
        upload: false,
        loading: false,
        droppedFiles: [],
        folderName: "/",
    }

    //object representing the login credential information and term selection
    $scope.login = {
        sid: "",
        pin: "",
        year: "",
        term: ""
    }

    //variable to keep track of whether the user is authenticated or not
    $scope.authed = false;
    //variable to keep track of whether the user was unsuccessfully authenticated or has no classes
    $scope.error = false;

    //variable to keep track of whether the future reveal field is enabled
    $scope.futureReveal = false;
    //variable to associate with the upload for future reveal
    $scope.revealTime;

    //variable to keep track of whether student mode is active (controls like moving and deleting and uploading are disabled)
    $scope.student = true;
    //variable to keep track of whether admin mode is active (if it is, then the toggle for student mode is active; otherwise, student mode is enforced with no toggle)
    $scope.admin = false;

    //the master data structure of files/folders for all classes that is kept updated via websockets
    $scope.files = {};

    //variable to keep track of the current date; for comparison to future reveal dates
    $scope.currentDate = Date.now();
    //keep the date updated to the second
    $interval(function() {
        $scope.currentDate = Date.now();
    }, 1000);

    //keeps track of the classes the user can enumerate, populated from BannerWeb
    $scope.classes;
    //builds a list of years for the user to choose from, based on the current year
```

```
        first
50      //can probably limit to the current year
51      $scope.years = [];
52      //builds a list of terms the user can choose from, based on the current term first
53      $scope.terms = [];
54
55      //keeps track of which class the user is currently viewing/interacting with
56      $scope.activeClass;
57      //keeps track of the loading indicator between submitting login details and
        receiving the list of classes
58      $scope.loading = false;
59
60      $scope.buildYears = function() {
61          $scope.years = [];
62          var currentYear = new Date().getFullYear();
63          for (var i = currentYear; i > currentYear - 4; i--) {
64              $scope.years.push(i);
65          }
66          $scope.login.year = $scope.years[0];
67      }
68
69      $scope.buildTerms = function() {
70          $scope.terms = [];
71          var currentMonth = new Date().getMonth() + 1;
72          if (currentMonth <= 5) {
73              $scope.terms.push("Spring");
74              $scope.terms.push("Fall");
75              $scope.terms.push("Summer");
76          } else if (currentMonth <= 8) {
77              $scope.terms.push("Summer");
78              $scope.terms.push("Spring");
79              $scope.terms.push("Fall");
80          } else {
81              $scope.terms.push("Fall");
82              $scope.terms.push("Summer");
83              $scope.terms.push("Spring");
84          }
85          $scope.login.term = $scope.terms[0];
86      }
87
88      //initialize the Socket.IO environment
89      $scope.socket = io();
90      //whenever the socket disconnects and reconnects, request the files from the
        server
91      $scope.socket.on('reconnect', function(num) {
92          $scope.requestFiles();
93      });
94      //sends a socket message asking for the active class' files
95      $scope.requestFiles = function() {
96          $scope.socket.emit('message', $scope.activeClass);
97      }
98      //sends a socket message with the login details and enables the loading indicator
99      $scope.sendLogin = function() {
100         $scope.error = false;
101         $scope.socket.emit('classes', $scope.login);
102         $scope.loading = true;
103     }
```

```
104       //when a message is received from the socket with the relevant class' files...
          load them
105       $scope.socket.on('message', function (msg){
106           $scope.$apply(function () {
107               for (var className in msg) {
108                   //only care about the classes we have asked about before or the
                      current class (even if never asked for before), since all classes
                      that get updated are sent out to all sockets
109                   if ($scope.files[className] || $scope.activeClass == className) {
110                       //overwrite the current data structure with the new information
111                       $scope.files[className] = msg[className];
112                   }
113               }
114           });
115       });
116       //when a message is received from the socket with the list of classes... load them
117       $scope.socket.on('classes', function (msg){
118           $scope.$apply(function () {
119               //hide the loading indicator since this is what we were waiting for
120               $scope.loading = false;
121               //if there is nothing in the response, blank the password field and show
                  the error
122               if (!msg || !msg.classes || !msg.classes.length > 0) {
123                   $scope.login.pin = "";
124                   $scope.authed = false;
125                   $scope.error = true;
126               } else {
127                   //if there are classes in the response, add them to the controller
128                   $scope.classes = msg.classes;
129                   //select the first class in the interface dropdown by default
130                   $scope.activeClass = $scope.classes[0];
131                   //fetch the files for the selected class from the server
132                   $scope.requestFiles();
133                   //set the appropriate combinations based on whether admin status is
                      indicated by the socket
134                   if (msg.admin) {
135                       $scope.admin = true;
136                       $scope.student = false;
137                   } else {
138                       $scope.admin = false;
139                       $scope.student = true;
140                   }
141                   //since we received something back, there was no error
142                   $scope.error = false;
143                   //we are now logged in, so update that flag
144                   $scope.authed = true;
145               }
146           });
147       });
148   });
```