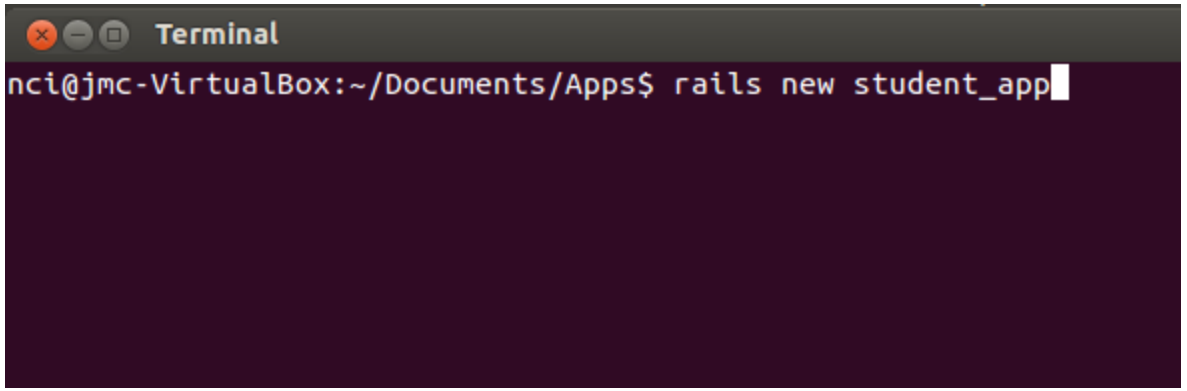
AngularJS

CRUD Application example with AngularJS and Rails 4

Slides By: Jonathan McCarthy

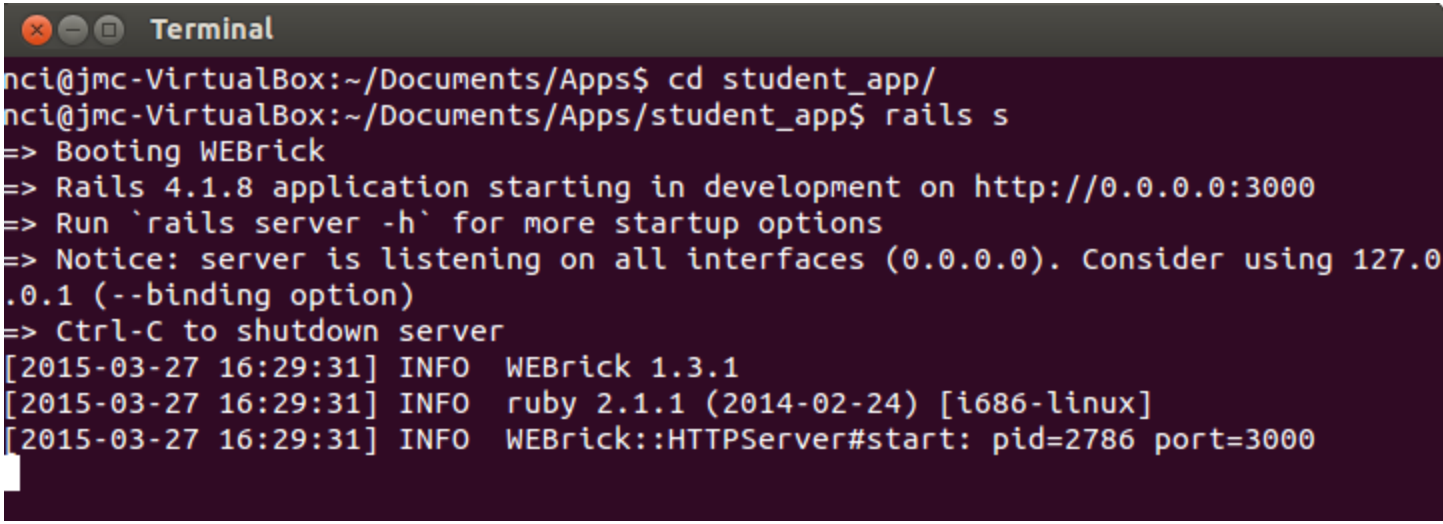
Create a new Rails App

- For this example we will create an application to store student details.
- Create a new rails 4 app
  - rails new student_app



Run the Application

- After creating the new application, cdange directory to the new project folder.
  - cd student_app
- Run the app
  - rails s



- Go to http://127.0.0.1:3000/

Create a Student Scaffold

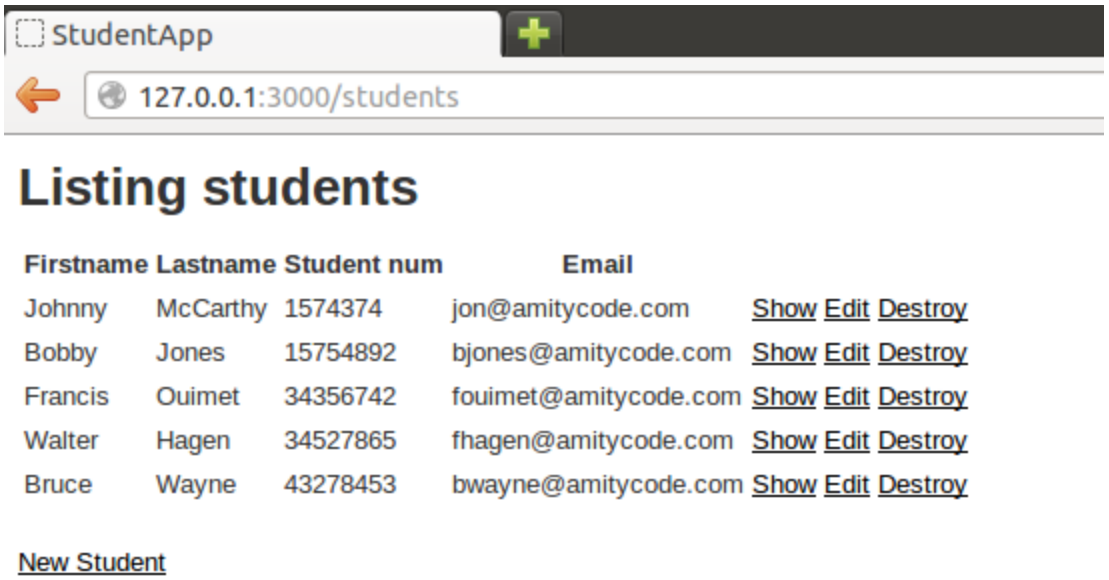
- A student should have the following values:
  - Firstname
  - Lastname
  - Student Number
  - Email Address

- Run the following command to create the scaffold
- rails generate scaffold student firstname:string lastname:string student_num:string email:string

- Create the database (run the following command)
  - rake db:migrate

Add some sample data

- Run the server (rails s), go to http://127.0.0.1:3000/students
- Add 5 students to the system



Setup Angular

Add the following gems to the app

- Open the gemfile in the project folder for editing. Add the following to the file:
  - gem 'angularjs-rails'
  - gem 'angular-ui-bootstrap-rails'
- The file should look like:
- Run the command:
  - bundle install

Gemfile

```
source 'https://rubygems.org'

gem 'angularjs-rails'
gem 'angular-ui-bootstrap-rails'

# Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
gem 'rails', '4.1.8'
# Use sqlite3 as the database for Active Record
gem 'sqlite3'
# Use SCSS for stylesheets
gem 'sass-rails', '~> 4.0.3'
# Use Uglifier as compressor for JavaScript assets
gem 'uglifier', '>= 1.3.0'
# Use CoffeeScript for .js.coffee assets and views
```

Including Angular

- By adding the two gems in the previous step, this will add Angular to the web application.
- Update the template:
- ng-app="mycrudapp"

- <div class="maincontent" ng-view>
- <%= yield %>
-

Application.html.erb

- The template file should look as follows:

application.html.erb

```
<!DOCTYPE html>
<html ng-app="mycrudapp">
<head>
  <title>StudentApp</title>
  <%= stylesheet_link_tag    'application', media: 'all', 'data-turbolinks-track' => true %>
  <%= javascript_include_tag 'application', 'data-turbolinks-track' => true %>
  <%= csrf_meta_tags %>
</head>
<body>

<div class="maincontent" ng-view>
   <%= yield %>
</div>

</body>
</html>
```

Including Angular in Rails Page

- Update /app/assets/javascripts/application.js to include the fillowing Angular references:
  - //= require angular
  - //= require angular-route
  - //= require angular-resource
  - //= require angular-ui-bootstrap
- The file should now look as follows (next slide):

Including Angular in Rails Page

- **Application.js** is a manifest file that'll be compiled into application.js, which will include all the files listed below. After modification the file should look as follows:

```
// This is a manifest file that'll be compiled into application.js, which will include all the files
// listed below.
//
// Any JavaScript/Coffee file within this directory, lib/assets/javascripts, vendor/assets/javascripts,
// or vendor/assets/javascripts of plugins, if any, can be referenced here using a relative path.
//
// It's not advisable to add code directly here, but if you do, it'll appear at the bottom of the
// compiled file.
//
// Read Sprockets README (https://github.com/sstephenson/sprockets#sprockets-directives) for details
// about supported directives.
//

//= require jquery
//= require jquery_ujs
//= require angular
//= require angular-route
//= require angular-resource
//= require angular-ui-bootstrap
//= require turbolinks
//= require_tree .
```

Setup JS Angular files

- In the assets\Javascripts directory, create a new folder named angular
  - mkdir angular
  - cd angular
  - mkdir controllers
- This will be used to contain the logic for the Angular functionality.

Create a Angular Student Controller

- Create a new file named **student_controller.js** in the /assets/javascripts/angular/controllers folder.
- Add the following to the file:
  - var mycrudapp = angular.module('mycrudapp', ['ngRoute', 'ngResource']);

student_controller.js

```
var mycrudapp = angular.module('mycrudapp', ['ngRoute', 'ngResource']);
```

Factory and Providers

- The Factory is used to send and recieve information from out server side application.
- The usage of the Factory will create an object and prepoerties will be added to the object. When the object is passed to the controller, these prepoerties that are part of the object will be available in the controller for use through the Factory.

- See: https://docs.angularjs.org/guide/providers

Setup the Factory

- **Add the following to student_controller.js**

```
mycrudapp.factory('Students',
    ['$resource',function($resource){
  return $resource('/students.json', {},{
    query: { method: 'GET', isArray: true },
    create: { method: 'POST' }
  })
}]);
```

The first Factory is named Students.
We can query students or create a new student.

Setup the Factory

- **Add the following to student_controller.js**

```
mycrudapp.factory('Student', ['$resource', function($resource){
  return $resource('/students/:id.json', {}, {
    show: { method: 'GET' },
    update: { method: 'PUT', params: {id: '@id'} },
    delete: { method: 'DELETE', params: {id: '@id'} }
  });
}]);
```

- The second Factory is named Student and allows us to show students, update an existing student and to delete a student. This is following Rails restful resources.

student_controller.js

- The file student_controller.js should now look as follows:

```
var mycrudapp = angular.module('mycrudapp', ['ngRoute', 'ngResource']);

// Factory
mycrudapp.factory('Students', ['$resource',function($resource){
  return $resource('/students.json', {},{
    query: { method: 'GET', isArray: true },
    create: { method: 'POST' }
  })
}]);

mycrudapp.factory('Student', ['$resource', function($resource){
  return $resource('/students/:id.json', {}, {
    show: { method: 'GET' },
    update: { method: 'PUT', params: {id: '@id'} },
    delete: { method: 'DELETE', params: {id: '@id'} }
  });
}]);
```

Routing in Angular

- $routeProvider provides routing functionality to the Angular application with minimal effort. $routeProvider is the provider for the $route service. The $route service allows us to easily link controllers, templates and the URL in the address bar. The $route service can be used preserve the back and forward navigation in the browser.

Setup the Routes

- **Add the following to student_controller.js**

```
// This is the Routes Section
mycrudapp.config([
 '$routeProvider', '$locationProvider', function($routeProvider, $locationProvider) {
 $routeProvider.when('/students',{
    templateUrl: '/templates/students/index.html',
    controller: 'StudentListCtr'
 });
 $routeProvider.when('/students/new', {
   templateUrl: '/templates/students/new.html',
   controller: 'StudentAddCtr'
 });
 $routeProvider.when('/students/:id/edit', {
   templateUrl: '/templates/students/edit.html',
   controller: 'StudentUpdateCtr'
 });
 $routeProvider.otherwise({
   redirectTo: '/students'
 });
 }
]);
```

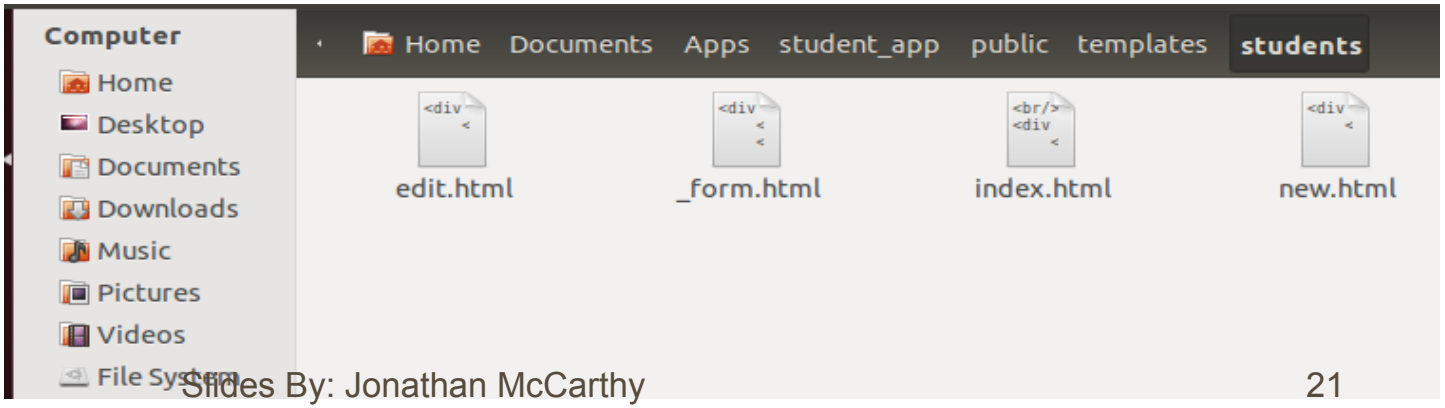
Create the Angular Templates

- For this simple CRUD example, we will create the following templates:
  - Index.html
  - New.html
  - Edit.html
  - _form.html

- The templates will be stored in the public folder in the Rails applcation.
- The next slide wil detail the setup of the templates.

Create the Angular Templates

- In the rails projects public folder, create a new folder named templates and in this create a folder named students. Create 4 new files in the student with the following names:
  - index.html
  - new.html
  - edit.html
  - _form.html



Form Explanation

-
- <label class="control-label col-md-3">First name * </label>
-
- <input type="text" name="firstname" class="form-control" ng-model="student.firstname" required placeholder="first name"/>
-
- First name is required.
-

_form.html

```
<div class="form-group" ng-class="{'has-error' : submitted && studentForm.firstname.$invalid}">
   <label class="control-label col-md-3">First name <span class="required">* </span> </label>
   <div class="col-md-4">
      <input type="text" name="firstname" class="form-control" ng-model="student.firstname" required placeholder="first name"/>
   </div>
   <p ng-show=" submitted && studentForm.firstname.$invalid" class="help-block">First name is required.</p>
</div>

<div class="form-group" ng-class="{'has-error' : submitted && studentForm.lastname.$invalid}">
   <label class="control-label col-md-3">Last name <span class="required">* </span> </label>
   <div class="col-md-4">
      <input type="text" name="lastname" class="form-control" ng-model="student.lastname" required placeholder="last name"/>
   </div>
   <p ng-show=" submitted && studentForm.lastname.$invalid" class="help-block">Last name is required.</p>
</div>

<div class="form-group" ng-class="{'has-error' : submitted && studentForm.email.$invalid}">
   <label class="control-label col-md-3">Email <span class="required">* </span> </label>
   <div class="col-md-4">
      <input type="email" name="email" class="form-control" ng-model="student.email" required placeholder="test@example.com"/>
   </div>
   <p ng-show="submitted && studentForm.email.$error.required" class="help-block">Email is required.</p>
   <p ng-show="submitted && studentForm.email.$error.email" class="help-block">Enter valid email</p>
</div>

<div class="form-group">
   <label class="control-label col-md-3">Student Number </label>
   <div class="col-md-4">
      <input type="text" name="student_num" class="form-control" ng-model="student.student_num" placeholder="9876543210"/>
   </div>
   <p ng-show="submitted && studentForm.student_num.$invalid" class="help-block">student Number is required.</p>
</div>

<div class="form-group">
   <div class="pull-left">
      <a class="btn btn-default" href="/"> Back</a>
      <button type="submit" class="btn btn-primary" ng-click="submitted=true">Submit</button>
   </div>
</div>
```

new.html

```
<div class="col-md-10">
    <form class="tab-pane active form-horizontal" id="first"
     name="studentForm" novalidate ng-submit="save()">
        <h3 class="block">Add new student</h3>
        <div ng-include src="'/templates/students/_form.html'"></div>
    </form>
</div>
```

edit.html

```
<div class="col-md-10">
    <form class="tab-pane active form-horizontal" id="first"
     name="studentForm" novalidate ng-submit="update()">
        <h3 class="block">Edit student</h3>
        <div ng-include src="'/templates/students/_form.html'"></div>
    </form>
</div>
```

index.html

```
<br/>
<div class="row">
   <div class="col-md-12">
      <a class="btn btn-primary" href="#/students/new">Create a student</a>
      <h3 class="block">Students</h3>
      <table class="table table-striped">
         <tr>
            <th>First Name</th>
            <th>Last Name</th>
            <th>Student Number</th>
            <th>Email</th>
            <th></th>
         </tr>
         <tr ng-hide="students.length" >
            <td colspan="5">No students found, Please create one.</td>
         </tr>
         <tr ng-show="students.length" ng-repeat="student in students">
            <td>{{student.firstname}}</td>
            <td>{{student.lastname}}</td>
            <td>{{student.student_num}}</td>
            <td>{{student.email}}</td>
            <td>
               <a href="#/students/{{student.id}}/edit">Edit</a> | <a href="" ng-click="deleteStudent(student.id)">Remove</a>
            </td>
         </tr>
      </table>
   </div>
</div>
```

Adding Controllers to the App

- Next we will add the Angular controllers to the application

List Controller

```
mycrudapp.controller("StudentListCtr", ['$scope', '$http', '$resource',
    'Students', 'Student', '$location', function($scope, $http, $resource,
    Students, Student, $location) {

  $scope.students = Students.query();

  $scope.deleteStudent = function (studentId) {
    if (confirm("Are you sure you want to delete this student?")){
      Student.delete({ id: studentId }, function(){
        $scope.students = Students.query();
        $location.path('/');
      });
    }
  };
}]);
```

Add Controller

```
mycrudapp.controller("StudentAddCtr", ['$scope', '$resource', 'Students',
    '$location', function($scope, $resource, Students, $location) {
  $scope.student = {}
  $scope.save = function () {
    if ($scope.studentForm.$valid){
      console.log('Here' + $scope.student);
      //Students.create({student: $scope.student}, function(){
      Students.create($scope.student, function(){
      $location.path('/');
    }, function(error){
      console.log(error)
    });
  }
 }
}]);
```

Update Controller

```
mycrudapp.controller("StudentUpdateCtr", ['$scope', '$resource', 'Student',
    '$location', '$routeParams', function($scope, $resource, Student, $location,
    $routeParams) {
  $scope.student = Student.get({id: $routeParams.id})
  $scope.update = function(){
   if ($scope.studentForm.$valid){
console.log('Here2' + $scope.student);
     Student.update($scope.student,function(){
       $location.path('/');
     }, function(error) {
       console.log(error)
    });
   }
  };
}]);
```

Where to place the controller code

- Add the code for the controllers to the student_controller.js

How the Module, Factory, Routes and Controllers work together

- **Module**
  - var mycrudapp = angular.module('mycrudapp', ['ngRoute', 'ngResource']);
  - The module has been assigned a variable name of mycrudapp, we will be adding additional functionality to this module.

How the Module, Factory, Routes and Controllers work together

- **Factory**

```
mycrudapp.factory('Students', ['$resource',function($resource){
  return $resource('/students.json', {},{
    query: { method: 'GET', isArray: true },
    create: { method: 'POST' }
  })
}]);
```

How the Module, Factory, Routes and Controllers work together

- **Factory**

```
mycrudapp.factory('Student', ['$resource', function($resource){
  return $resource('/students/:id.json', {}, {
    show: { method: 'GET' },
    update: { method: 'PUT', params: {id: '@id'} },
    delete: { method: 'DELETE', params: {id: '@id'} }
  });
}]);
```

How the Module, Factory, Routes and Controllers work together

- **Routes**

```
mycrudapp.config([
 '$routeProvider', '$locationProvider', function($routeProvider, $locationProvider) {
 $routeProvider.when('/students',{
    templateUrl: '/templates/students/index.html',
    controller: 'StudentListCtr'
 });
 $routeProvider.when('/students/new', {
   templateUrl: '/templates/students/new.html',
   controller: 'StudentAddCtr'
 });
 $routeProvider.when('/students/:id/edit', {
   templateUrl: '/templates/students/edit.html',
   controller: "StudentUpdateCtr"
 });
 $routeProvider.otherwise({
   redirectTo: '/students'
 });
 }
]);
```

How the Module, Factory, Routes and Controllers work together

- **Controllers**

```
mycrudapp.controller("StudentAddCtr", ['$scope', '$resource',
    'Students', '$location', function($scope, $resource, Students,
    $location) {
  $scope.student = {}
  $scope.save = function () {
    if ($scope.studentForm.$valid){
      console.log('Here' + $scope.student);
      //Students.create({student: $scope.student}, function(){
      Students.create($scope.student, function(){
      $location.path('/');
    }, function(error){
      console.log(error)
    });
  }
 }
}]);
```

How the Module, Factory, Routes and Controllers work together

- **Controller**

```
mycrudapp.controller("StudentListCtr", ['$scope', '$http',
    '$resource', 'Students', 'Student', '$location', function($scope,
    $http, $resource, Students, Student, $location) {

  $scope.students = Students.query();

  $scope.deleteStudent = function (studentId) {
    if (confirm("Are you sure you want to delete this student?")){
      Student.delete({ id: studentId }, function(){
        $scope.students = Students.query();
        $location.path('/');
      });
    }
  };
}]);
```

How the Module, Factory, Routes and Controllers work together

- **Controller**

```
mycrudapp.controller("StudentUpdateCtr", ['$scope',
    '$resource', 'Student', '$location', '$routeParams',
    function($scope, $resource, Student, $location,
    $routeParams) {
   $scope.student = Student.get({id: $routeParams.id})
   $scope.update = function(){
     if ($scope.studentForm.$valid){
console.log('Here2' + $scope.student);
       Student.update($scope.student,function(){
         $location.path('/');
       }, function(error) {
         console.log(error)
      });
     }
   };
}]);
```

CSRF – Edit application_controller.rb

- **Update /app/controllers/application_controller.rb to look as follows. The changes will allow Rails to work with the Angular app.**

```
class ApplicationController < ActionController::Base
  # Prevent CSRF attacks by raising an exception.
  # For APIs, you may want to use :null_session instead.
  protect_from_forgery with: :exception

  after_filter  :set_csrf_cookie_for_ng

  def set_csrf_cookie_for_ng
    cookies['XSRF-TOKEN'] = form_authenticity_token if protect_against_forgery?
  end

  protected

  def verified_request?
    super || form_authenticity_token == request.headers['X-XSRF-TOKEN']
  end

end
```

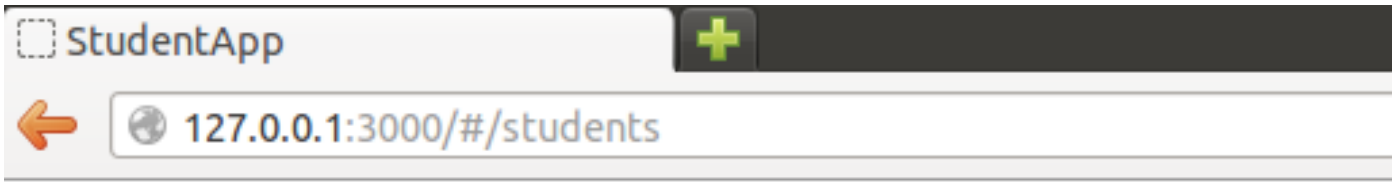
Update Routes.rb

- Update routes.rb to look as follows:

```
Rails.application.routes.draw do
  resources :students
  root 'students#index'
end
```

Run the App

- Run the application and test the functionality.
- http://127.0.0.1:3000



Create a student

Students

First Name	Last Name	Student Number	Email	
Johnny	McCarthy	111111	johnny@amitycode.com	Edit | Remove
Bobby	Jones	15754892	bjones@amitycode.com	Edit | Remove
Francis	Ouimet	34356742	fouimet@amitycode.com	Edit | Remove
Walter	Hagen	34527865	fhagen@amitycode.com	Edit | Remove
Bruce F.	Wayne	43278453	bwayne@amitycode.com	Edit | Remove
Rocky	Balboa	12345	rocky@email.com	Edit | Remove

Questions