# SJSU Course Evaluation

Team 131

# Source Code

**Model:**

**class *Course*** < ***ApplicationRecord*** *belongs\_to* **:user** *validates* **:user\_id**, **presence**: **true** *validates* **:type**, **presence**: **true** *has\_many* **:evaluations

 def self**.*search*(*search*)
 *where*(**"title LIKE ?"**, **"%**#{*search*}**%"**)

 **end
end**

**class *TextCourse*** < ***Course*** *validates* **:title**, **presence**: **true**, **uniqueness**: { **scope**: **:title**}

 *validates* **:body**, **presence**: **true
end**

**class *Evaluation*** < ***ApplicationRecord*** belongs\_to **:course** belongs\_to **:user** *validates* **:course\_id**, **presence**: **true** *validates* **:user\_id**, **presence**: **true
end**

**Controller:**

**class *CoursesController*** < ***ApplicationController****before\_action* **:authenticate\_user!
 def** *index* **if :search
 @courses**=***Course***.search(params[**:search**]).*order*(**"created\_at DeSC"**)
 **else
 @courses**=***Course***.all.*order*(**'Created\_at DESC'**)
 **end

 end
 def** *show* **@course**=***Course***.*find*(params[**:id**])
 **end
end**

**class *TextCoursesController*** < ***ApplicationController* def** *new* **@text\_course** =***TextCourse***.new
 **end

 def** *create* **@text\_course**=current\_user.text\_courses.build(text\_course\_params)
 **if @text\_course**.save
 *redirect\_to* course\_path(**@text\_course**), **notice**: **"Course Created"
 else** *render* **:new**, **alert**: **"Error Creating Course"
 end
 end
 def** *edit* **@text\_course** = current\_user.text\_courses.*find*(params[**:id**])
 **end
 def** *update* **@text\_course** = current\_user.text\_courses.*find*(params[**:id**])
 **if @text\_course**.update(text\_course\_params)
 *redirect\_to* course\_path(**@text\_course**), **notice**: **"Course Updated!"
 else** *render* **:edit**, **alert**: **"Error updating post."
 end
 end
 def** *show* **@text\_course**=***TextCourse***.*find*(params[**:id**])
 **end** *private* **def** *text\_course\_params* params.*require*(**:text\_course**).permit(**:title**, **:body**)
 **end

end**

**class *EvaluationsController*** < ***ApplicationController*** *before\_action* **:authenticate\_user!
 def** *create* **@evaluation**=current\_user.evaluations.build(evaluation\_params)

 **if @evaluation**.save
 *redirect\_to* course\_path(**@evaluation**.course\_id),**notice**: **'Evaluation was sucessfully created.'
 else** *redirect\_to* course\_path(**@evaluation**.course\_id), **alert**:**'Error creating Evaluation'
 end
 end

 def** *index* **@evaluations**= ***Evaluation***.all
 **end
 def** *show* **@course** = ***Course***.*find*(params[**:id**])

 **end
 def** *destroy* **@evaluation** = ***Evaluation***.*find* (params[**:id**])
 **if @evaluation**.destroy
 *redirect\_to* course\_path(**@evaluation**.course\_id), **notice**: **'Evaluation Deleted'
 else** *redirect\_to* course\_path(**@evaluation**.course\_id), **notice**: **'Error Deleting Evaluation'
 end

 end
 def** *edit* **@evaluation** = current\_user.evaluations.*find*(params[**:id**])
 **end
 def** *update* **@evaluation** = current\_user.evaluations.*find*(params[**:id**])
 **if @evaluation**.update(evaluation\_params)
 *redirect\_to* course\_path(**@evaluation**.course\_id), **notice**: **"Evaluation Updated!"
 else** *render* **:edit**, **alert**: **"Error updating Evaluation."
 end
 end** *private* **def** *evaluation\_params* params.*require*(**:evaluation**).permit(**:body**, **:course\_id**)
 **end
end**

**View:**

**<%=** *form\_tag***(courses\_path, :method *=>* "get", id: "search-form") do %>
 <%= text\_field\_tag :search, params[:search], placeholder: "Search Courses" %>
 <%= submit\_tag "Search" %>
<% end %>
<% if @courses.present? %>** <**div class="post-preview"**>

 **<%=** *render* **@courses%>** </**div**>
 <**hr**>
**<% else %>** <**p**>There are no Courses containing the term(s), would you like to create a new course **<%= params[:search] %>**.</**p**>
**<% end %>**

<**div class="page-header"**>
 <**h1**>Course</**h1**>
</**div**>

**<%=** *render* **@course %>

<%=** *link\_to* **"Courses", courses\_path, class: "btn btn-default" %>**<**h2**>Evaluations</**h2**>
**<%=** *render* **@course.evaluations %>**<**h3**>New Evaluation</**h3**>
**<%=** *form\_for* **@course.evaluations.build do |***f***| %>** <**div class="form-group"**>
 **<%=***f***.label :body%>**<**br**>
 **<%=** *f***.text\_area :body, class: "form-control"%>** </**div**>

 **<%=** *f***.hidden\_field :course\_id %>
 <%=***f***.submit class: "btn btn-primary"%>
<% end %>**

**<%=** *form\_for* **@text\_course do |***f***|%>** <**div class="form-group"**>
 **<%=***f***.label :title%>
 <%=** *f***.select(:title, options\_for\_select([['-Select-','NULL'],
 ['CMPE 30','CMPE 30'],
 ['CMPE 50','CMPE 50'],
 ['CMPE 102','CMPE 102'],
 ['CMPE 120','CMPE 120'],
 ['CMPE 126','CMPE 126'],
 ['CMPE 127','CMPE 127'],
 ['CMPE 130','CMPE 130'],
 ['CMPE 131','CMPE 131'],
 ['CMPE 132','CMPE 132'],
 ['CMPE 133','CMPE 133'],
 ['CMPE 137','CMPE 138'],
 ['CMPE 146','CMPE 146'],
 ['CMPE 148','CMPE 148'],
 ['CMPE 150','CMPE 150'],
 ['CMPE 152','CMPE 152'],
 ['CMPE 165','CMPE 165'],
 ['CMPE 172','CMPE 172'],
 ['CMPE 180','CMPE 180'],
 ['CMPE 187','CMPE 187'],
 ['CMPE 188','CMPE 188'],
 ['CMPE 189','CMPE 189'],
 ['CMPE 195A','CMPE 195A'],
 ['CMPE 195B','CMPE 195B'],
 ['CMPE 195D','CMPE 195D'],
 ['CMPE 197','CMPE 197'],
 ['CMPE 198','CMPE 198']])) %>** </**div**>
<**div class="form-group"**>
 **<%=***f***.label :body%>
 <%=***f***.text\_area :body, class: "form-control"%>**</**div**>

 **<%=***f***.submit class: "btn btn-primary"%>
 <%=***link\_to* **'Cancel', courses\_path, class: "btn btn-default"%>

<%end %>**

# Checklist

## Documentation

* All methods are commented in clear language. If it is unclear to the reader, it is unclear to the user.
* All source code contains an author for all code.
* A version should be included as required.
* All class, variable, and method modifiers should be examined for correctness.
* Describe behavior for known input corner-cases. Ex courses should only allow course selection and brief description.
* Complex algorithms should be explained with references.
* Code that depends on non-obvious behavior in external frameworks is documented with reference to external documentation to better understand the code.
* Incomplete code is marked with //TODO or //FIXME markers in source code.

## Testing

* Unit tests are added for each code path, and behavior.
* Unit tests **must** cover error conditions and invalid parameter cases. Ex. User id valid.
* Unit tests for standard algorithms should be examined against the standard for expected results.
* Ensure that the code fixes the issue and that the unit test confirms it.

## Error Handling

* Invalid parameter values are handled properly early in methods (Fast Fail).
* Consider using a general error handler to handle known error conditions.
* An Error handler **must** clean up state and resources no matter where an error occurs.
* Display error message when creating and editing a course

## Thread Safety

* Objects accessed by multiple threads are accessed only through a lock, or synchronized methods.

## Performance

* Objects are duplicated only when necessary.
* Avoid large objects in memory.
* Do not leave debugging code in production code.
* Web page rendering is processed in a timely matter.

# Key Use Case Description

|  |  |
| --- | --- |
| **Use Case name:** | New Class Section |
| **Product name:** | SJSU Class Evaluation |
| **Team:** | Team131 |
| **Date:** | 2/23/17 |

| 1. Goal |
| --- |
| Create a new class section  |

| 2. Summary |
| --- |
| Create a new class section to inform other students about previous course experience |

| 3. Actors |
| --- |
| Actor 1: Veteran Student |

| 4. Preconditions |
| --- |
| * Logged in to their account successfully
* In the courses listing
 |

| 5. Trigger |
| --- |
| Students click “Add Course” button |

|  |
| --- |
| 6. Primary Sequence |
| **Step** | **Action** |
| 1 | Student click on the comment box |
| 2 | Give their own opinion about the course, class, and their professor |
| 3 | The paragraph should be more than 100 words |
| *4* | Rate class on helpfulness, difficulty, time spent on classwork, and garde received |

| 7. Primary Postconditions |
| --- |
| * Class section is created
* Class section can be seen in Post Listing
 |

|  |
| --- |
| 8. Alternate Sequences |
| **Alternate Trigger** |
| Class Section already exists |
| **Step** | **Action** |
| 1 | Display error message |
| 2 | Ask user to search for class section instead  |
| Alternate Postconditions  |
| * System returns to the comment box
 |

| 9. Nonfunctional Requirements  |
| --- |
| * The system respond to student within 15 seconds
* System would log out if student are idle more than 15 minutes
 |

| 10. Glossary |
| --- |
| * Student- who need to provide the information about the teacher
 |