http://web.cse.ohio-state.edu/~giles.25/3901/labs/project6.html

Student resource (routed)
- name: string
- email: string
- password:string (in the database: password_digest:string)
- signed: boolean
- has_and_belongs_to_many Groups
- has_many FeedbackItems

Instructor resource (routed)
- name: string
- email: string
- password:string (in the database: password_digest:string)

Group resource (routed)
- name: string
- has_and_belongs_to_many Students
- has_many FeedbackItems

```
def feedback_for(proj)
    self.feedback_items.where(project: proj)
```

Project resource (routed)
- name: string
- is_open: boolean (so an admin can close a project)

```
def feedback_for(group):
    group.feedback_for(self)
```

FeedbackItem resource
- (indexed) belongs_to Project
- belongs_to Student -> The author
- (indexed) belongs_to Student -> The target (who the feedback is about)

- participation: integer 1-5
- quality: integer 1-5
- disagreements: integer 1-5
- comments: text

StudentsController (Kaiyuan)

GET /students (index)

Only be visible to instructors. If a student tries to go here redirect to /students/{id}

Should show a list of students, w/ view, edit, and delete buttons

Sort in alphabetical order

Link to add new student

GET /students/new (new)

This is the sign-up page which instructor registers emails and names of students in this step. Password and signed up will be set to default.

In this page:

Hide the password entry

Allow any email and name

POST /students (create)

This is what the form on the signup page calls

Create should be able to handle if the student's email exists AND password is not set

GET /students/{id}/{mode} (show)

Mode denotes student or instructor

If not authenticated properly, show a "permission denied" page

When a student logs in, they get redirected here

IF logged in as student:

Show the projects this student needs to leave feedback for (like catme)

IF logged in as instructor:

Show the feedback items this student got

Sort in most-recent-first order

GET /students/{id}/edit/{mode} (edit)

Mode demonstrate that who(student or instructor is manipulating)

If not authenticated properly, show a "permission denied" page

Lets student edit their name/password

Lets instructor edit student's name/email

After a student changes the default password, signed is set as true

PATCH /students/{id} (update)

Called by the form on the edit page

DELETE /students/{id} (destroy)

Called by the delete button on the /students page

Go through and delete all the feedback items about this student

GET /students/login/{id} (login_process)

Use the email get from (email) to process login

GET /students/login (login)

Redirect students to (email).

POST /students/login/{id} (check)

Logs in the student based on the form above

GET /students/email

Ask the student for email to decide whether redirect to sign up page or log in page.

POST /student/email

Respond to the form of (email)

DELETE /students/logout

Log out the student

GET /students/signup

Redirect students to (email)

InstructorsController (Evan)

GET /instructors

Only visible to instructors

Add/delete instructors (like what rails generates by default)

GET /instructors/new

Only visible if signed in as an instructor, OR if no instructors are available

Shows form to create instructor

POST /instructors

This is what the form on the signup page calls

GET /instructors/{id}

Only visible to instructors

Basically what rails generates by default

GET /instructors/{id}/edit

Ditto

PATCH /instructors/{id}

Ditto

DELETE /instructors/{id}

Ditto

GET /instructors/login

Static page

Shows login form for student

POST /instructors/login

Logs in the student based on the form above

GroupsController (Chris)

GET /groups

Only visible to instructors

Add/delete/edit groups (like what rails generates by default)

GET /groups/new

Ditto

Give a UI to let the instructors add members to a group

POST /groups

Ditto

GET /groups/{id}

Ditto

GET /groups/{id}/edit

Ditto

Give a UI to let the instructors add members to a group

PATCH /groups/{id}

Ditto

DELETE /groups

Ditto

ProjectsController (Chris)

GET /projects

Only visible to instructors

Add/delete/edit/view projects (like what rails generates by default)

Sorted in most-recent-first order

GET /projects/new

Ditto

POST /projects

Ditto

GET /projects/{id}

Ditto

Show a group-by-group breakdown of ratings

Each group gets a table, where each row is a member of the group, and the columns are:

- Aggregate score (out of 15, just the sum of the subscores)
- Link to view more details (GET /feedback_item/{id})

GET /projects/{id}/edit

Ditto

Give a UI to close/reopen the project

PATCH /projects/{id}

Ditto

DELETE /projects

Ditto

Go through and delete all the feedback items associated w/ this project

Password1!

FeedbackItemsController (Adrian)

POST /feedback_items (create/update)

If not authenticated properly, show a “permission denied” page

Post data includes all the data necessary to create the feedback item

GET /feedback_items/{id}

Only allow instructors to view

Shows a feedback item’s content

EvaluationsController (Adrian)

GET /evaluations/{group}/{project}
- Only accessible to logged in students w/ membership in group
- Returns a page w/ instructions and a next button
- Next button takes you to the first member

GET /evaluations/{group}/{project}/{member_idx}
- member_idx is index into group.members array
- Only accessible to logged in students w/ membership in group
- Returns a page w/ format on it to leave one FeedbackItem
- Form POSTs to /feedback_items
- Prefill form with any existing FeedbackItem
- Next button
  - If member_idx == group.members.length, go to /evaluations/done
  - Else go to /evaluations/{group}/{project}/{member_idx + 1}

GET /evaluations/done
- Static page
- Says you're done and includes a back or go home button

ApplicationController (Adrian)

GET /
- IF logged out, redirect to /login
- IF logged in a student, redirect to /students/{id}
- IF logged in as instructor, redirect to /projects

GET /login
- Page w/ buttons "are you a [student] or [instructor]", which redirect to endpoint

GET /full-system-reset-for-testing-only
- Page asking to enter reset key

POST /full-system-reset-for-testing-only
- If given the right key, resets the database
- Reset key: 0c90301e-0b89-4840-b42b-0a86d3eef8d4

LoginHelper (app/helpers/login_helper.rb)

Read 7 and 8, it gives an idea how to do this
Logic for handling login belongs here

student_logged_in?(student)
current_student (returns the current logged in student obj)
log_in_student(obj) (stores this user as the one that's logged in)

instr_logged_in?(instr)
current_instr
log_in_instr(obj)

logged_in? (if anyone is logged in)

log_out! (log out anyone who’s currently logged in)

Page to collect feedback from one user about another

Rate ${name}'s participation in ${project}:
(Insufficient) [] [] [] [] [] (Sufficient)

${name} produced high quality work on ${project}:
(Disagree) [] [] [] [] [] (Agree)

You had significant disagreements with ${name}
(Disagree) [] [] [] [] [] (Agree)

Leave comments for ${name}:
[

]

[Back][Next]