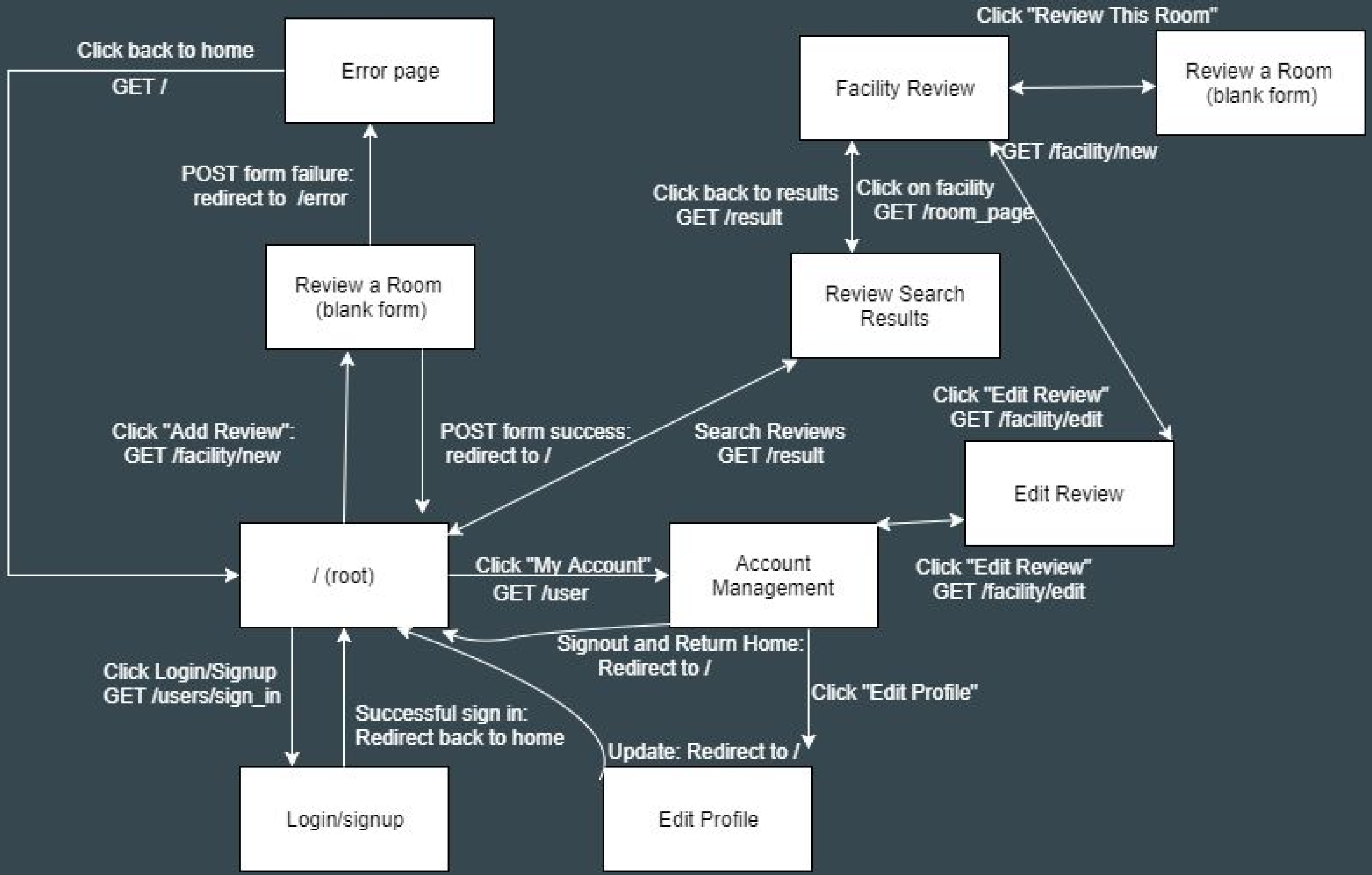
FaciliRate

Computer Science Squad (CSS)

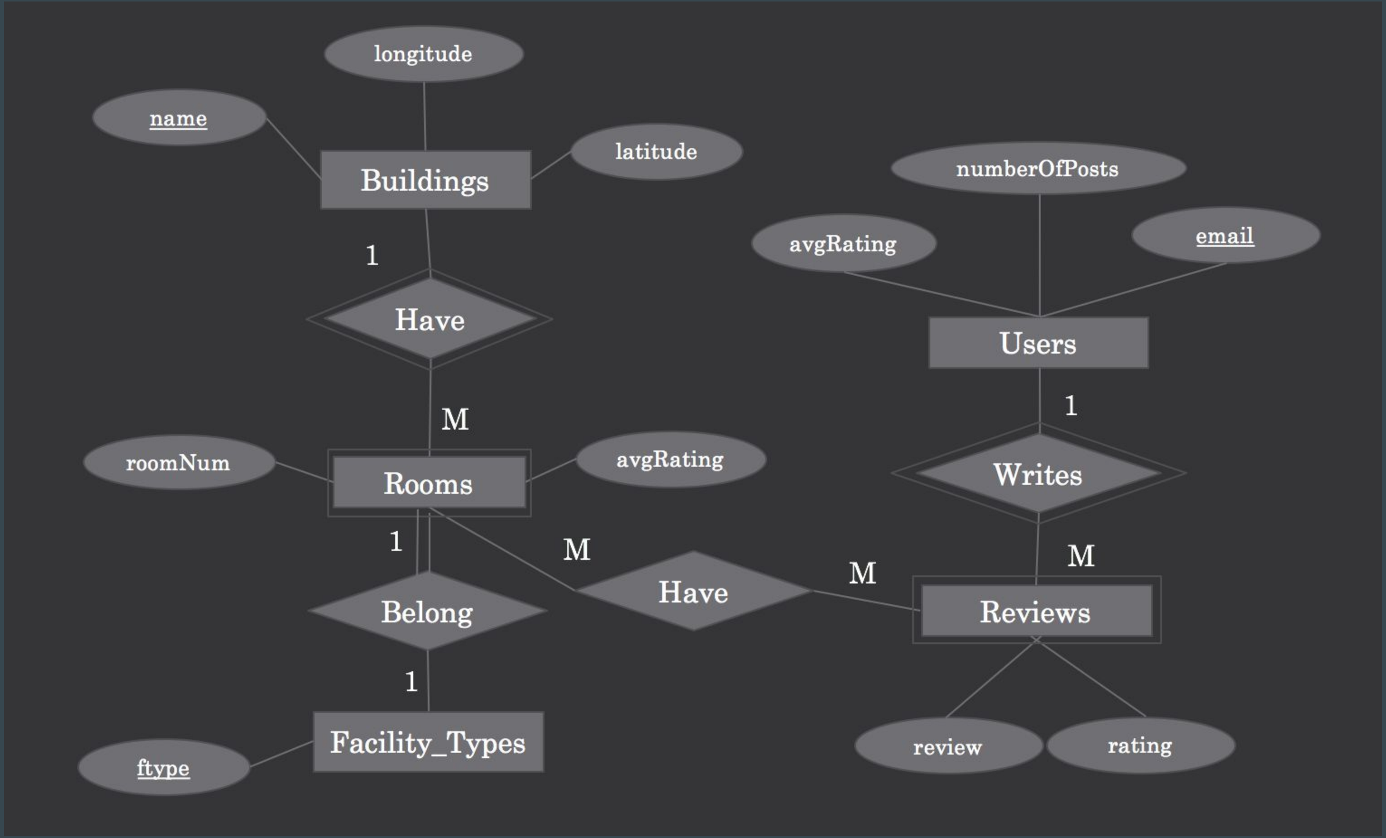
Overview

- Schema presented through E.R. diagram
- Controllers and Views (Home, User, Result, Room, and Facility)
- Frameworks
- Demo

Website Diagram

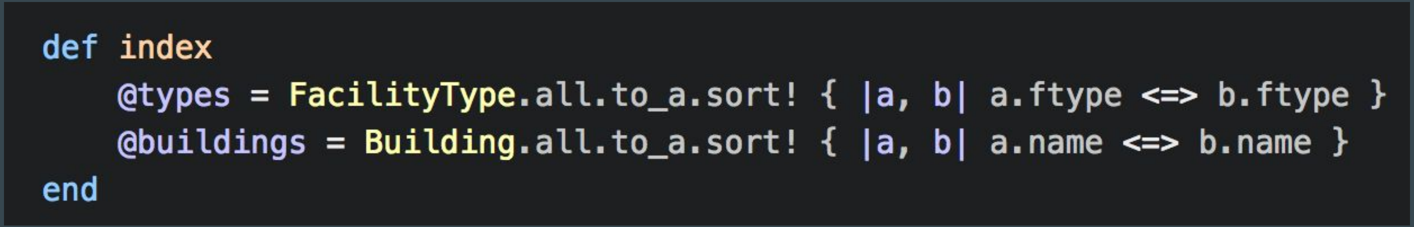

Error page
Click back to home
GET /
POST form failure: redirect to /error
Review a Room (blank form)
Click "Add Review": GET /facility/new
POST form success: redirect to /
/ (root)
Click "My Account" GET /user
Account Management
Click Login/Signup GET /users/sign_in
Successful sign in: Redirect back to home
Login/signup
Signout and Return Home: Redirect to /
Click "Edit Profile"
Update: Redirect to /
Edit Profile
Search Reviews GET /result
Review Search Results
Click back to results GET /result
Click on facility GET /room_page
Facility Review
Click "Review This Room"
Review a Room (blank form)
GET /facility/new
Click "Edit Review" GET /facility/edit
Edit Review
Click "Edit Review" GET /facility/edit

E.R Diagram


longitude
name
Buildings
latitude
1
Have
M
numberOfPosts
avgRating
email
Users
1
Writes
M
roomNum
Rooms
avgRating
1
M
Have
M
Belong
Reviews
1
Facility_Types
ftype
review
rating

Home Controller

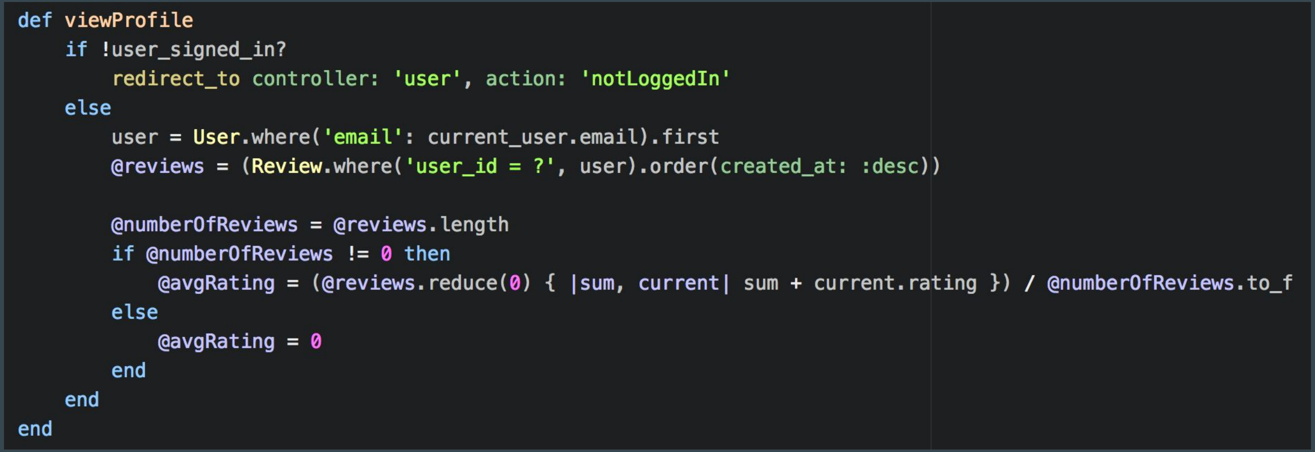
- Sets up the variables needed for the home page to give user options for autocomplete text entry (Buildings) and for drop down menus (Facility Types)
- Creates an array variables to contain all possible types of facilities and all possible buildings



```
def index
    @types = FacilityType.all.to_a.sort! { |a, b| a.ftype <=> b.ftype }
    @buildings = Building.all.to_a.sort! { |a, b| a.name <=> b.name }
end
```

User Controller

- Allows only signed in users to reach the profile page
- Assigns a variables to contain all reviews made by the user, the number of reviews made by the user and the average rating made by the user

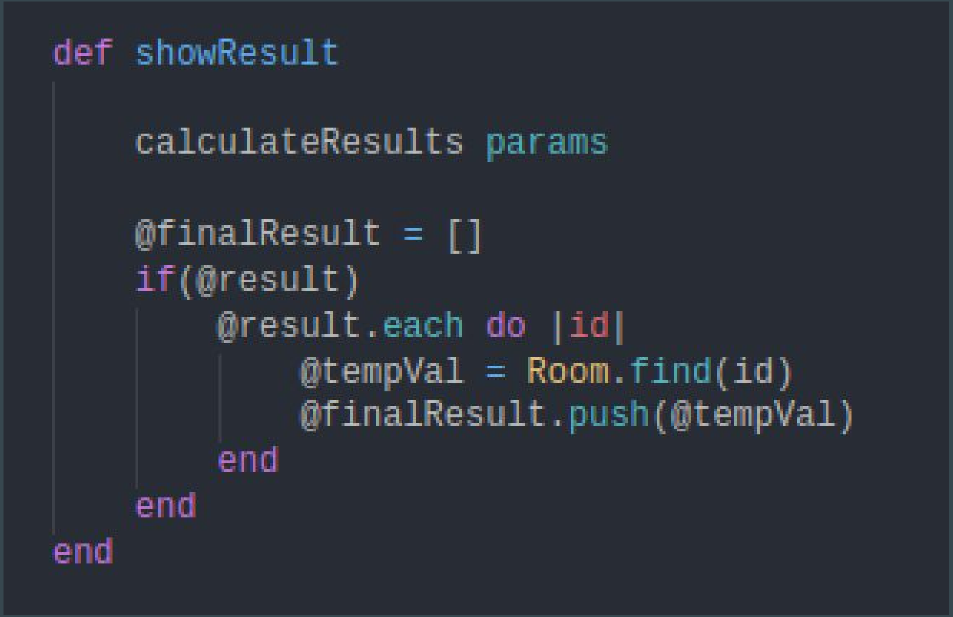


```
def viewProfile
    if !user_signed_in?
        redirect_to controller: 'user', action: 'notLoggedIn'
    else
        user = User.where('email': current_user.email).first
        @reviews = (Review.where('user_id = ?', user).order(created_at: :desc))

        @numberOfReviews = @reviews.length
        if @numberOfReviews != 0 then
            @avgRating = (@reviews.reduce(0) { |sum, current| sum + current.rating }) / @numberOfReviews.to_f
        else
            @avgRating = 0
        end
    end
end
```

Result Controller

- Search request in homepage routes to showResult method in result controller

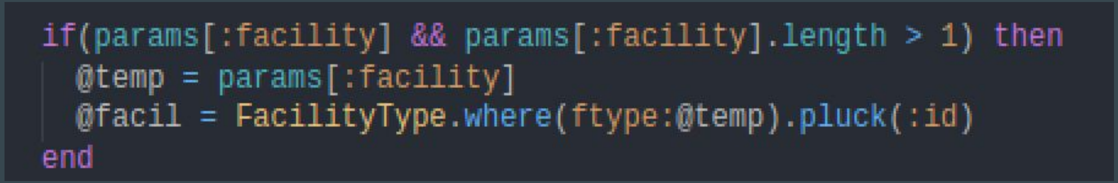
- @finalResult contains array of rooms and in the model the variable is looped to create a table



```
def showResult

    calculateResults params

    @finalResult = []
    if(@result)
        @result.each do |id|
            @tempVal = Room.find(id)
            @finalResult.push(@tempVal)
        end
    end
end
```

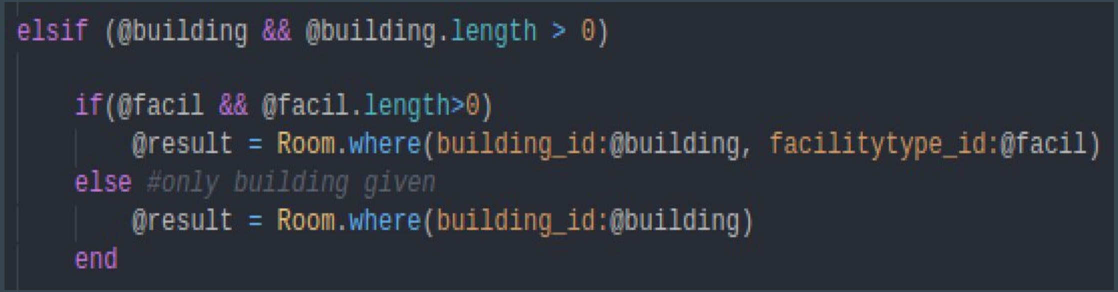
Result Controller(Helper)

- Helper method is called from result controller with arguments



```
if(params[:facility] && params[:facility].length > 1) then
  @temp = params[:facility]
  @facil = FacilityType.where(ftype:@temp).pluck(:id)
end
```

- Goes through nested if else conditions and obtain rooms matching the criteria user requested

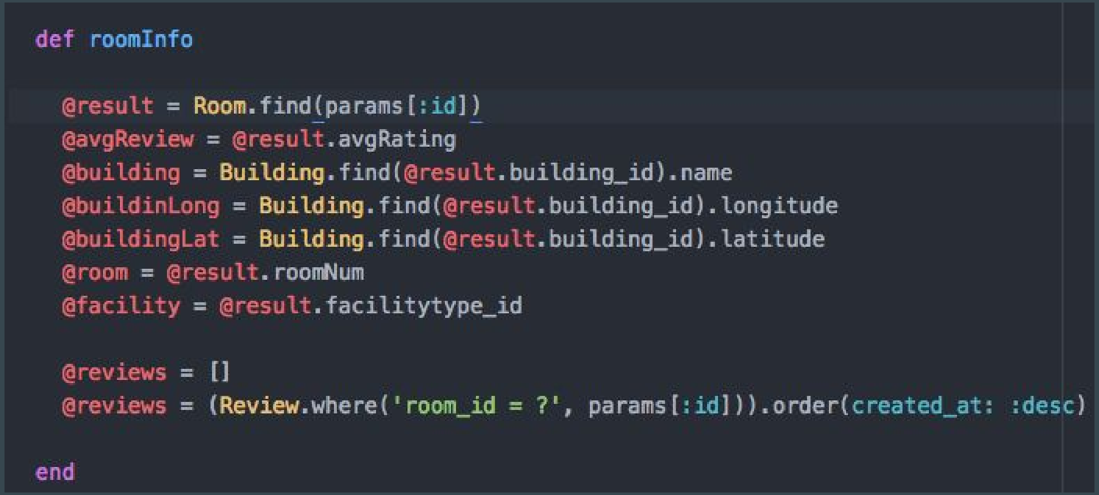


```
elsif (@building && @building.length > 0)

    if(@facil && @facil.length>0)
        @result = Room.where(building_id:@building, facilitytype_id:@facil)
    else #only building given
        @result = Room.where(building_id:@building)
    end
```

Room Page Controller

- Gathers building, facility, and location information about a specific room

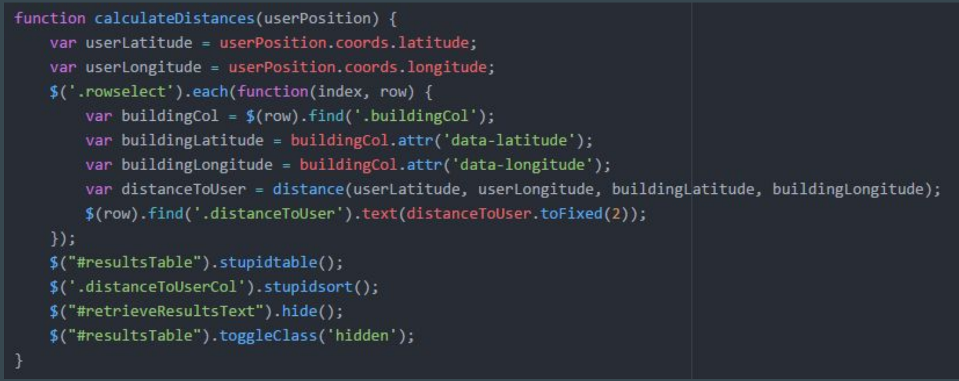
- Creates an array of reviews for the room to be displayed on the room's page



```
def roomInfo

  @result = Room.find(params[:id])
  @avgReview = @result.avgRating
  @building = Building.find(@result.building_id).name
  @buildinLong = Building.find(@result.building_id).longitude
  @buildingLat = Building.find(@result.building_id).latitude
  @room = @result.roomNum
  @facility = @result.facilitytype_id

  @reviews = []
  @reviews = (Review.where('room_id = ?', params[:id])).order(created_at: :desc)

end
```

Geolocation

- Fetched every OSU building, along with its longitude and latitude and inserted into database
- HTML5 Geolocation API
  - Handling if not available
- Haversine formula to calculate distance from user coordinates to each building
- Displaying distance in results table
  - Sortable using jQuery plugin

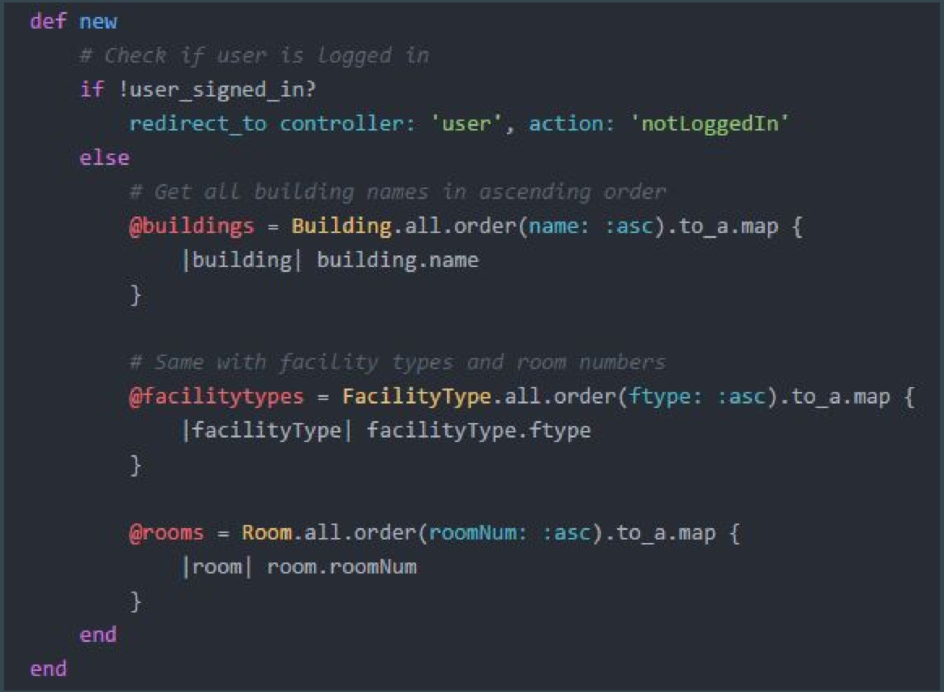


```
function calculateDistances(userPosition) {
    var userLatitude = userPosition.coords.latitude;
    var userLongitude = userPosition.coords.longitude;
    $('.rowselect').each(function(index, row) {
        var buildingCol = $(row).find('.buildingCol');
        var buildingLatitude = buildingCol.attr('data-latitude');
        var buildingLongitude = buildingCol.attr('data-longitude');
        var distanceToUser = distance(userLatitude, userLongitude, buildingLatitude, buildingLongitude);
        $(row).find('.distanceToUser').text(distanceToUser.toFixed(2));
    });
    $("#resultsTable").stupidtable();
    $('.distanceToUserCol').stupidsort();
    $("#retrieveResultsText").hide();
    $("#resultsTable").toggleClass('hidden');
}
```

Facility Controller - New

- Renders an empty form for the user to create a review

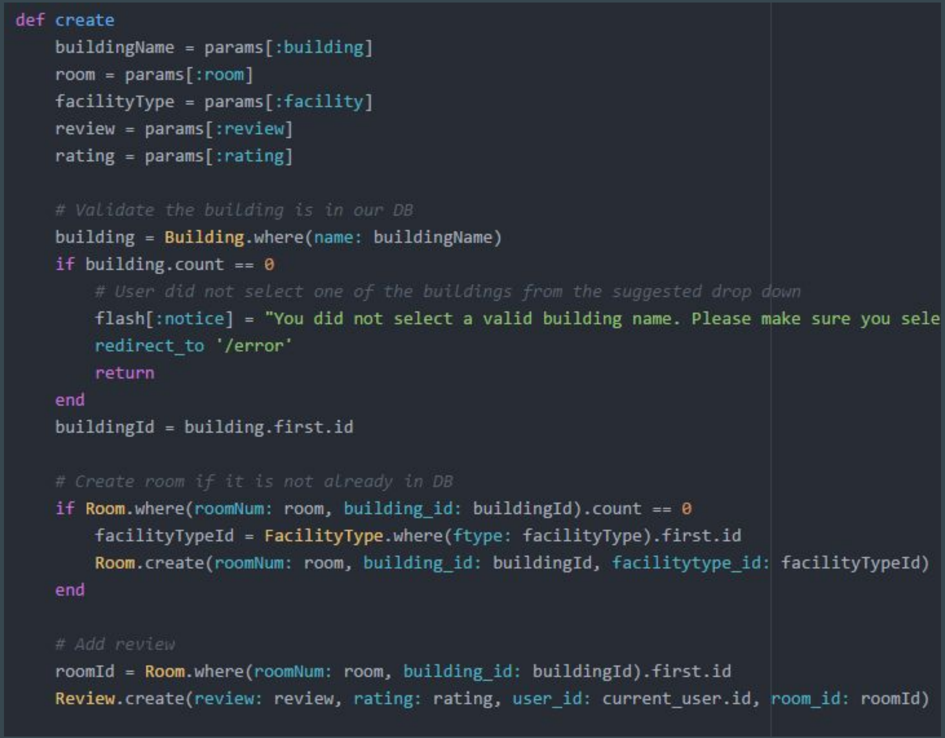
- Fetches buildings, facility types, and rooms so client input fields have autocomplete suggestions



```ruby
def new
    # Check if user is logged in
    if !user_signed_in?
        redirect_to controller: 'user', action: 'notLoggedIn'
    else
        # Get all building names in ascending order
        @buildings = Building.all.order(name: :asc).to_a.map {
            |building| building.name
        }

        # Same with facility types and room numbers
        @facilitytypes = FacilityType.all.order(ftype: :asc).to_a.map {
            |facilityType| facilityType.ftype
        }

        @rooms = Room.all.order(roomNum: :asc).to_a.map {
            |room| room.roomNum
        }
    end
end
```

Facility Controller - Create

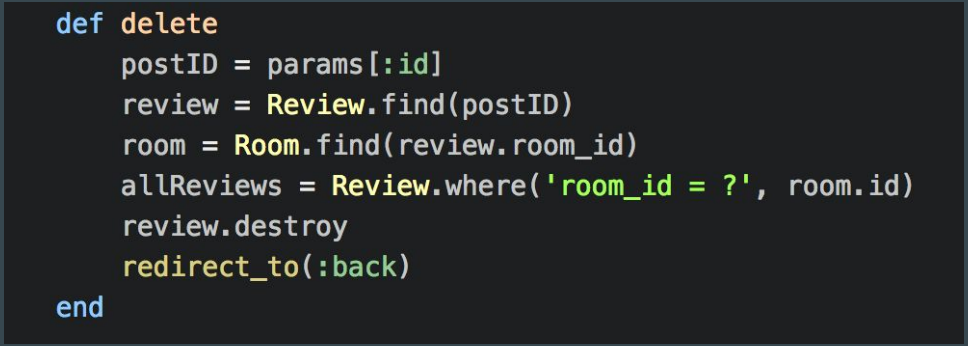
- Given a filled form of review data, creates a review in the database
- Creates facility if not already present in database
- In addition to client-side validation, performs some basic server-side validation



```ruby
def create
    buildingName = params[:building]
    room = params[:room]
    facilityType = params[:facility]
    review = params[:review]
    rating = params[:rating]

    # Validate the building is in our DB
    building = Building.where(name: buildingName)
    if building.count == 0
        # User did not select one of the buildings from the suggested drop down
        flash[:notice] = "You did not select a valid building name. Please make sure you sele
        redirect_to '/error'
        return
    end
    buildingId = building.first.id

    # Create room if it is not already in DB
    if Room.where(roomNum: room, building_id: buildingId).count == 0
        facilityTypeId = FacilityType.where(ftype: facilityType).first.id
        Room.create(roomNum: room, building_id: buildingId, facilitytype_id: facilityTypeId)
    end

    # Add review
    roomId = Room.where(roomNum: room, building_id: buildingId).first.id
    Review.create(review: review, rating: rating, user_id: current_user.id, room_id: roomId)
```

Facility Controller - Delete

- Deletes the review that is passed through the url from the Review table
- Updates the value of the average rating of the room to account for the deleted review



```ruby
def delete
    postID = params[:id]
    review = Review.find(postID)
    room = Room.find(review.room_id)
    allReviews = Review.where('room_id = ?', room.id)
    review.destroy
    redirect_to(:back)
end
```

Frameworks Used in FaciliRate

- Bootstrap
- Devise
- jQuery
- Autocomplete
- gmaps4rails

Demo