Exemplar Code

```ruby
#Mentee uses this route to send mentor applications
get "/applications" do
  @menteeIDList = [] #List to store mentee
  IDList = Request.where(mentorID: $mentors.id) #Finds the request where it finds the specified mentor ID
  IDList.each do |id|
    mentee = Mentee.first(id: id.menteeID)
    @menteeIDList.push(mentee) unless mentee.nil?
  end
  erb :mentee_applications
end

post "/match" do
  #When mentor accepts mentee request, mentorMatch field updated to value of mentor ID in the mentees table
  Mentee.where(id: params[:menteeID]).update(:mentorMatch => $mentors.id)

  #When mentor accepts mentee request, menteeMatch field updated to value of mentee ID in the mentors table
  Mentor.where(id: $mentors.id).update(:menteeMatch => params[:menteeID])

  #Deletes the all requests a mentor has from the table after mentor accepts a mentee request
  Request.where(mentorID: $mentors.id).delete
  redirect "/MentorDashboard"
end

get "/myMentor" do
  #List to store the mentor matched to a mentee
  @mentorMatchedList = []
  #Finds the ID of the mentor that is equal to the value stored in the mentorMatch column in the mentees table
  MentorIDList = Mentor.where(id: $mentees.mentorMatch)
  MentorIDList.each do |id|
    @mentorMatchedList.push(Mentor.first(id: id.id)) #Stores the mentor record found in the list to be displayed
  end

  erb :myMentor
end

get "/myMentee" do
  #List to store the mentee matched to a mentor
 @menteeMatchedList = []
  #Finds the ID of the mentee that is equal to the value stored in the menteeMatch column in the mentors table
  MenteeIDList = Mentee.where(id: $mentors.menteeMatch)
  MenteeIDList.each do |id|
    @menteeMatchedList.push(Mentee.first(id: id.id)) #Stores the mentor record found in the list to be displayed
  end

  erb :myMentee
end
```

Listing A project/controllers/applications.rb

```ruby
get "/MenteeDashboard" do
  redirect "/login" unless session[:logged_in] #If mentee logged in then display mentee dashbard, otherwise redirect to login page
  username = session[:mentees_username] #Logs user in
  #Gets the mentee information that corresponds to the username
  $mentees = Mentee.first(username: username) # '$' used to make it a global variable
  erb :mentee_dashboard
end
```

Listing B project/controllers/login.rb

```ruby
get "/MenteeSignUpForm" do
  @mentees = Mentee.new
  erb :mentee_signup
end

post "/MenteeSignUpForm" do
  @mentees = Mentee.new
  @mentees.load(params)
  @error = nil

  if @mentees.valid?
      if @mentees.exist_signup?
        @error = "Username/email exists"
      else
        @mentees.save_changes
        #Sends an email to mentee that the sign up was successful
        send_mail(@mentees.email,
          "Successful Sign up!",
          "Hi "+@mentees.fname+" "+@mentees.lname+" !\n"+
          "Your username is: "+@mentees.username+"\n"+
          "Your email: "+@mentees.email+"\n"+
          "Please use these credentials to login into your mentee account \n"+
          "\n\n\nRegards\nTeam 6")
        redirect "/login"
      end
  end
```

Listing C project/controllers/add.rb

```ruby
 #Checks if username/email already in the mentors database to prevent multiple sign ups
def exist_signup?
  other_mentors = Mentor.first(username: username)
  mentors =  Mentor.first(email: email)
  !other_mentors.nil? ||  !mentors.nil?
end
```

Listing D project/models/mentor.rb

```ruby
get "/search" do
  #New variable to enable mentees to search for mentors based on course name
  @courseName_search = params.fetch("courseName_search", "").strip

  #This error corresponds to the /addApplication route, this is to prevent
  #the error from being discarded when page is redirected to search
  @error = true if params.fetch("error", "") == "1"

  #If no course name is being searched, it displays the list of all mentors in alphabetical order
  #else it searches through the course name field in the mentors table and displays mentors' whose
  #course name contains the course name being searched
  @mentors = if @courseName_search.empty?
               Mentor.order(:courseName).all
             else
               Mentor.order(:courseName).where(Sequel.ilike(:courseName, "%#{@courseName_search}%")) #ilike used to make search case insensitive
             end

  erb :mentor_search
end
```

Listing E project/controllers/search.rb

```ruby
post "/addApplication" do
  @error = nil #initializes variable
  @requests = Request.new #creates a new instance of requests
  #The params are from buttons in the mentor_search.erb, they correspond to the mentee sending the request
  #and the mentor being requested
  @requests.load(params[:menteeID].to_s, params[:mentorID].to_s)
  @mentors = Mentor.first(id: params[:mentorID]) #Creates a new instance based on mentor id of mentor being requested

  #If application exists throws error and redirects to below URL(line 33)
  #else saves changes by updating the requests table and sending the mentor an email
  if @requests.exist_application?
    @error = "Application already sent"
    redirect "/search?error=1"
  else
    @requests.save_changes
    #Sends an email to mentor that a mentee sent them an application
    send_mail(@mentors.email,
          "You have a new mentee application!",
          "Hi "+@mentors.fname+" "+@mentors.lname+" !\n"+
          "You have a new mentee application. Please login into your mentor account and view your mentee applications \n"+
          "\n\nRegards\nTeam 6")
  end
    redirect "/search"
end
```

Listing F project/controllers/search.rb

```ruby
#Function called when editing profile. Second parameter displays the stored values from the databases
def loadEdit(params)
  self.id = params.fetch("id", self.id).strip
  self.fname = params.fetch("fname", self.fname).strip
  self.lname = params.fetch("lname", self.lname).strip
  self.email = params.fetch("email", self.email).strip
  self.phoneNum = params.fetch("phoneNum", self.phoneNum).strip
  self.username = params.fetch("username", self.username).strip
  self.password = params.fetch("password", self.password).strip
  self.jobTitle = params.fetch("jobTitle").strip unless params.fetch("jobTitle").strip == ""
  self.courseName = params.fetch("courseName").strip unless params.fetch("courseName").strip == ""
  #Done the last bit since initially description is empty
  self.description = params.fetch("description").strip unless params.fetch("description").strip == "" || params.fetch("description").nil?
end
```

Listing G project/models/mentor.rb

```ruby
  #Checks if application already sent to prevent multiple submissions
  def exist_application?
    #Goes through all the records in requests table and returns true if mentor id is present
    other_request = Request.where(menteeID: menteeID)
    other_request.each do |requestVar|
     return true if requestVar.mentorID == mentorID
    end
    return false
  end
```

Listing H project/models/requests.rb

```ruby
require_relative "../spec_helper"
#tests the mentor search when it is empty
describe "the search page" do
  context "with an empty database" do
    it "says the database is empty" do
      visit "/search"
      expect(page).to have_content "Your search revealed no mentors"
    end
  end
#tests the mentor search when there is only
  context "with one record in the database" do
    it "lists the mentors" do
      add_test_mentor
      add_test_user
      visit "/login"
      fill_in "username", with: "123"
      fill_in "password", with: "123"
      click_button "Submit"
      visit "/search"
      expect(page).to have_content "Sam Mentor"
      clear_database
    end
#checks that mentor doesnt appear when wrong course is entered
    it "does not list the mentor when a different courseName is searched for" do
      add_test_mentor
      add_test_user
      fill_in "username", with: "123"
      fill_in "password", with: "123"
      click_button "Submit"
      visit "/search"
      fill_in "searchForMentor", with: "Architecture (BA)"
      click_button "Submit"
      expect(page).to have_content "Your search revealed no mentors"
      clear_database
    end
#checks that mentor appears when wrong course is entered
    it "lists the mentor when their courseName is searched for" do
      add_test_mentor
      add_test_user
      fill_in "username", with: "123"
      fill_in "password", with: "123"
      click_button "Submit"
      visit "/search"
      fill_in "searchForMentor", with: "Accounting and Financial Management (BA)"
      click_button "Submit"
      expect(page).to have_content "Sam Mentor"
      clear_database
    end
  end
end
```

Listing I project/spec/features/search_spec.rb

```ruby
require_relative "../spec_helper"

RSpec.describe Mentor do
    #checks that name returns correctly from database
  describe "#name" do
    it "returns the mentor's full name" do
      mentor = described_class.new(fname: "test", lname: "test")
      expect(mentor.name).to eq("test test")
    end
  end
#checks that email returns correctly from database
 describe "#email" do
    it "returns the mentor's email" do
      mentor = described_class.new(email: "test@gmail.com")
      expect(mentor.email).to eq("test@gmail.com")
    end
  end
#checks that phoneNum returns correctly from database
      describe "#phoneNum" do
    it "returns the mentor's full phoneNum" do
      mentor = described_class.new(phoneNum: "123456789")
      expect(mentor.phoneNum).to eq("123456789")
    end
  end
#checks that username returns correctly from database
      describe "#username" do
    it "returns the mentor's username" do
      mentor = described_class.new(username: "1234")
      expect(mentor.username).to eq("1234")
    end
  end
#checks that password returns correctly from database
      describe "#password" do
    it "returns the mentor's password" do
      mentor = described_class.new(password: "1234")
      expect(mentor.password).to eq("1234")
    end
  end
end
```

Listing J project/spec/unit/mentor_spec.rb