**Spark Registration**

**By**

**Brogrammers**

**Views, Behavior, and URLs**

* Main page – ‘/’
  + registration\_home/index.html.erb
    - Elementary Registration button
      * Connects user to register as an Elementary Student
      * Takes them to ‘/student\_users/new?student\_level=Elementary’
        + This is the view in app/views/student\_users/new.html.erb
    - Middle School Registration button
      * Connects user to register as a Middle School Student
      * Takes them to ‘/student\_users/new?student\_level=Middle’
        + This is the view in app/views/student\_users/new.html.erb
    - High School Registration button
      * Takes the user to select to register as an individual or with a team, if the user wants to register with a team they must provide the team code on the next page.
      * Takes them to ‘/high/index’
        + Select the ‘Individual’ button to register normally

Takes them to ‘/student\_users/new?student\_level=High’

This is the view in app/views/student\_users/new.html.erb

* + - * + Select the ‘Team’ button to register with a team.

Enter a code to join the team

Takes them to ‘/join\_team/index’

* + - Login link
      * Takes the user to ‘/login’
      * Allows the user to type in their information to login an d view their profile.
    - Admin Login link
      * This link is strictly for the admins to use, it goes through Texas A&M’s CAS validation system. **Permission is needed for this link to function**.
* Tests
  + Cucumber
    - bundle install
    - rake db:setup
    - sudo service postgresql start
    - cucumber
    - If this does not work make sure you have the proper databases setup.
  + Rspec
    - Bundle install
    - Rake db:setup
    - rake tests
  + SimpleCov
    - Whenever you run your tests normally, the SimpleCov will update automatically with the latest coverage report in index.html under the coverage folder
* Login link – ‘/sessions/new’
  + Begins a new session for users
  + If incorrect user/pass is entered, will redirect you to ‘/login’
  + If the user/pass is correct, then you will be direct to the profile of the user
    - This is also ‘/login’
    - This view can be found under app/views/student\_users/index.html.erb
    - Here the student should have the open to:
      * Show information: ‘/student\_users/{#id}’
        + This view can be found under app/views/student\_users/show.html.erb
      * Edit information: ‘/student\_users/{#id}/edit’
        + This view can be found under app/views/student\_users/edit.html.erb

This view calls ‘form’ which displays their information in a comprehensive manner with text boxes.

‘form’ can be found in views under app/views/student\_users/\_form.html.erb

* Logging in as an Admin with Login link
  + If you’re not using the CAS system through A&M, then logging in as an admin is exactly the same as logging in as a student via login link, NOT the Admin login link.
    - This form of Admin control should be used for TESTING ONLY. For the real website, admin login through the regular login link should be blocked, and the admin should use the ‘Admin Login’ link through CAS.
  + Once logged in, the display should have a button for ‘Logout’ and a link that says ‘Advisor & Student Information’
    - This url should also be ‘/login’
    - This view can be found under app/views/admins/home.html.erb
    - Upon clicking ‘Advisor & Student Information’
      * As the admin, you should be able to view all existing users and advisors including all of their information except their passwords.
      * This url should be ‘/admins/see\_info’
      * This view can be found under app/view/admins/see\_info.html.erb
      * The admin also has the ability to:
        + Email specific groups of users

By pressing any of the email buttons you’ll be redirected to an email

The email page should be ‘/admins/email\_paid’ or ‘/admins/email\_unpaid’ or ‘/admins/send\_all’ depending on the button you pressed to email a specific group of people.

The view can be found under app/views/admins/email\_page

* + - * The admin can also click the ‘mark paid’ button per user to move them to the paid section of the users
      * The admin can also destroy users by clicking the destroy link for the respective user, upon doing this a prompt will pop up to double check if you’re sure you really want to destroy the user. There’s no going back.

**Controllers**

* admins\_controller.rb
  + index
    - This function helps generate the index page.
  + home
    - This function directs the admin to the admins home.
  + show
    - show displays all the admin’s information.
  + editlogin
    - Finds the current admin logged in and redirects the admin to the page that allows them to edit their information.
  + Super\_admin
    - Finds the super admin from the database of admins
  + Create
    - Helps create admins
  + Update
    - Updates the admin’s information
  + Changelogin
    - Helps the admin change their email address and checks if it’s valid
  + Mark\_paid
    - Allows the admin to move a user to the paid portion of the database.
  + Mark\_unpaid
    - Allows the admin to remove the user from the paid portion of the database.
  + Send\_email
    - Gives the admin the ability to send an email to an individual email in the database.
  + Send\_admin\_email
    - Lets the admin email another admin
  + Unpaid\_email\_group
    - Helps the admin send emails to all the user that have not paid.
  + email\_unpaid\_stud
    - This creates a premade email that is sent to all the users that have not yet paid when the admin chooses to do so.
  + Paid\_email\_group
    - Helps the admin send emails to all the users that have paid.
  + Send\_stud\_email
    - Sends the student an email
  + Admin\_email
    - Used by super admin to email admins
  + Edit\_email
    - Gives the email page the details about who the admin is emailing.
  + Email\_all
    - Allows the admin to email everyone, if the super admin is logged in all of the admins will also receive emails.
  + Email\_advisors
    - Allows the admin to email all of the advisors
  + Changepassword
    - Helps the admin change his/her password
  + Destroy
    - Allows the super admin to destroy other admins, only the super admin may do so.
  + Check\_permission
    - Checks if the admin is logged in for security.
  + Admin\_params
    - Skeleton for admin params
  + Student\_user\_params
    - Skeleton for student\_user params
* Advisor\_controller.rb
  + Create\_team
    - Allows the advisor to create a team for student users to join.
  + genTeamCode
    - Creates a code for the team the advisor creates for students to use to join a specific team.
* Advisor\_user\_controller.rb
  + Index
    - Creates a list of all the current advisors
  + New
    - Creates new advisor that’s empty
  + genPayCode
    - Creates a code for the team the advisor creates for students to use to join a specific team.
  + Create
    - Creates a new advisor with preset parameters
  + Update
    - Updates the advisor with present parameters
  + Destroy
    - Destroys an advisor in the database
  + Set\_advisor\_user
    - Sets the current user to the advisor based on their ID #
  + Check\_permission
    - Prevents users from accessing advisor information without being logged in as the advisor for the specific session.
  + Advisor\_user\_params
    - This is where the advisor’s params may be updated, Create and update get their information for what to do with this function.
* Registration\_home\_controller.rb
  + Index
    - Resets the session disabling any inaccessible urls.
* Session\_controller.rb
  + Login
    - Checks for nil session, if true, then the user is redirected to the home page.
  + Check\_permission
    - Checks for a students user session to be nil, if true, then the use is redirected to the home page
  + Log\_out
    - Logs out of CAS (not yet implemented)
  + Log\_out\_students
    - Logs out for non CAS users and redirects to the home page.
  + Create
    - Creates a new sessions for the specific student, admin, or advisor.
* Student\_user\_controller.rb
  + Index
    - Gets student user by ID
  + Generate
    - Creates a csv file with all of the students in the database for the admins to view
  + Show
    - Displays all the information of the student
  + New
    - Creates an empty new student
  + Create
    - Creates a new student with preset parameters based on student\_user\_params
  + Update
    - Updates the student with preset parameters based on student\_user\_params
  + Changepassword
    - Changes the student user’s password
  + Set\_student\_user
    - Sets the current student user
  + Check\_permission
    - Checks the session to ensure that the user isn’t accessing pages above their level of authorization.
  + Student\_user\_params
    - This is the preset params for create and update.
* Join\_team\_controller.rb
  + Parse\_comments
    - Gets the team code then checks If the team exists in the database. Redirect to student user path if successful, else throw an error message.
* The following controllers have no relevant content in them, but are necessary for the program to run due to structuring requisites.
  + High\_controller.rb
  + Middle\_controller.rb
  + Elementary\_controller.rb
  + Registration\_complete\_controller.rb
  + Students\_controller.rb
  + Consent\_forums\_controller.rb

**Mailers**

* User\_mailer.rb
  + Welcome\_email
    - Creates the email to send to a user as soon as they successfully register
  + Unpaid\_email\_groups
    - Creates the email for the admins to send to the users that have not yet paid
  + Thanks\_email
    - Creates the thank you email to send to users as they register.

**Helpers**

* It should be noted that although there are 11 helper files, only 1 was actually used, thus the others have been left out of the documentation.
* Sessions\_helper.rb
  + Log\_in
    - Logs in an advisor
  + Current\_user
    - Finds the current user through ID
  + Logged\_in?
    - Returns true if someone is logged in, false if nobody is logged in.
  + Student\_log\_in
    - Logs in a student.
  + Student\_current\_user
    - Finds the current student\_user logged by ID.
  + Student\_logged\_in?
    - Returns true if a student is logged in, false if else.
  + Admin\_log\_in
    - Logs in an Admin
  + Admin\_current\_user
    - Finds the current admin user by ID.
  + Admin\_logged\_in?
    - Returns true if an admin is logged in, false if else.

**Models**

* Admin.rb
  + This contains the class for Admin which validates all of the input that is going into admin create or update to ensure it meets the minimum character parameters for each field.
* Advisor\_user.rb
  + Validiates all of the advisor\_user inputs that are sent through create and update.
* Student\_user.rb
  + Validates all of the student user’s information before processing it into a student\_user object and inserting it into the database. Also helps generate the csv file.

**Views**

* Admins
  + \_change\_market\_url\_form.html.erb
    - Creates a forum where an admin may change the marketplace url.
    - The marketplace url should go into the textfield “:mkt\_place\_url”
  + \_change\_right\_sig\_url\_form.html.erb
    - Similar to the previous view except this url change is for the consent forum url.
  + \_changeloginform.html.erb
    - Creates a forum that allows the admin to change their email address.
  + \_changepasswordform.html.erb
    - Creates a forum that allows the admin to change their password.
  + \_contactform.html.erb
    - Creates a forum that displays the admin’s contact information such as phone number.
  + \_form.html.erb
    - Creates a forum that a user may use to sign up for the event.
  + change\_market\_url.html.erb
    - calls \_change\_market\_url\_form.html.erb to import it into a view.
  + change\_right\_sig\_url.html.erb
    - calls \_change\_right\_sig\_url\_form.html.erb to import it into a view.
  + changeloginform.html.erb
    - calls \_changeloginform.html.erb to import it into a view
  + changepasswordform.html.erb
    - calls \_changepasswordform.html.erb to allow the admin to edit his/her password in a view.
  + edit.html.erb
    - calls \_contactform.html.erb to allow the admin to edit his.her contact information in a view.
  + email\_page.html
    - Gives the admin a view where he/she can create an email to send out to specific users based on the will of the controller.
  + Home.html.erb
    - This is the first thing the admin sees upon login, it has 4 links, one to view the user database, one to edit email, one to edit password, and one to logout.
  + Index.html.erb
    - This is a view where the super admin can view the admin database, the super admin may destroy or create admins at will. The super admin may also email the other admins from here.
  + New.html.erb
    - This view allows the super admin to create a new admin.
  + See\_info.html.erb
    - This view is the page where an admin can see the entire database of advisors and student\_users.
  + Show.html.erb
    - Shows all of the admins information as well as links to change any of it.
  + Super\_admin.html.erb
    - Provides a page for the super admin upon login. From here they can access the other admins, or change their information.
* Advisor
  + index.html
    - This view is what an advisor sees immediately after login. It has a create team button, see students in your team button, and contact SEC info button.
  + Info.html.erb
    - Displays the contact information for the SEC admins.
  + See\_team.html.erb
    - Displays the teams in a view that the advisor is captain of.
  + Team.html.erb
    - Creates a view where the advisor may create a new team.
* Advisor\_users
  + \_form.html.erb
    - Creates a forum for the advisor to input their information to register.
  + Edit.html.erb
    - Creates a view for the advisor to edit his/her information by calling \_forum.html.erb
  + index.html.erb
    - Lists all of the advisors in a view
  + new.html.erb
    - calls the \_forum.html.erb to create a new advisor
  + show.html.erb
    - displays a portion of the advisors information as well as gives them to option to edit or login.
* Join\_team
  + Index.html.erb
    - This view asks the user for a team code in order to join a team.
  + joining\_team.html.erb
    - displays a message that says the user either successfully or failed to join a team.
* Layout
  + Application.html.erb
    - Generates the banner that goes on top of each page.
* Payment
  + Index.html.erb
    - After registering successfully, you will be taken to this view where payment options are displayed, as well as a link to the consent forum.
* Registration\_complete
  + Index.html.erb
    - This view is at the end of the registration process and simply displays a successful registration message.
* Registration\_home
  + Index.html.erb
    - This view is the home page for registration, you can login or register as a high, middle, or elementary school student.
* Sessions
  + New.html.erb
    - This is essentially the login page, it is titled new because it is a new sessions after the user successfully logs in. There is also a link to create a new advisor here.
  + New\_student.html.erb
    - This view is the same as the previous except for students.
* Student\_users
  + \_changeloginform.html.erb
    - Creates a forum that allows a student to change their email.
  + \_changepasswordform.html.erb
    - Creates a forum that allows a student to change their password.
  + \_form.html.erb
    - Creates a forum that allows students to register.
  + changelogin.html.erb
    - displays the view that calls \_changeloginform.html.erb to let the student change their email through the website.
  + Changepassword.html.erb
    - Displays the view that calls \_changepassword.html.erb to let the student change their password through the website.
  + Edit.html.erb
    - Displays all of the student’s information by calling the forum \_forum.html.erb and lets them edit it as they please.
  + Generate.html.erb
    - This view helps generate the csv file of students.
  + index.html.erb
    - This view displays the student’s information after successfully logging in.
  + Index.xls.erb
    - Helps generate the xls and csv files for students.
  + New.html.erb
    - Calls \_forum.html.erb to this view that allows the student to register.
  + Show.html.erb
    - Shows the student’s information in a view.

**dB**

* seeds.rb
  + Creates premage admins, student users and advisors. Edit this file if you want to change the default admins and super admin.
* Schema.rb
  + Sets up the database for the students, admins, advisors, and teams.

**Tips**

* To reset the database locally:
  + rake db:setup
* To import the seed.rb locally:
  + rake db:seed
* Testing
  + You will need a database server running before you can run tests, to start a server type in:
    - Sudo service postgresql start
  + For cucumber type in “cucumber” to test.
    - This generates tests based on files in the features folder.
  + For rspec, type in “rake test”