Team 33 Report

To start the application, the git repository needs to be cloned and codio needs to be loaded;this should be done by typing git clone and "git@git.shefcompsci.org.uk:com1001-2020-21/team33/project". The directory should be changed to the project by typing "cd project" in the terminal window. Then a bundle install would be handy to run to ensure the gems are installed. To run the application, type "ruby app.rb" in the terminal. In order to use the application in the browser you should look for your codio box domain, which is displayed at the top of the terminal. It should look something like "XX-XX.codio.io" Copy-paste it in your browser, add -4567 (url should now look like "XX-XX-4567.codio.io" and the app will load.

To access different pages on the application, the test accounts provided below can be used:

For mentee:

Username: mentee1 Username: mentee2
Password: mentee1 Password: mentee2

For mentor:

Username: mentor1 Username: mentor2
Password: mentor1 Password: mentor2

For admin:

Username: admin Password: admin

To sign up another admin account, you will need an admin code. We've set up the following codes for the purposes of the demonstration:

adminCode1, adminCode2, adminCode3, adminCode4, adminCode5

As a mentee, you can look through mentors, filter them and send requests via the find a mentor page.
As a mentor, you can accept and decline requests via the requests page.
As an admin, you can view all the mentees and mentors and spot any problematic ones; as well as this you can change the passwords and update the q&a table.

Well done for adding the basic functionalities.
You could have made it more explicit though.

Stories:

Story	Acceptance Criteria	Priority	Est Effort (hrs)
1-As a user, I want to be able to register for an account. Iteration 1 - did it w2-w3	• After choosing whatever user they want to be, the user is asked to provide some information in order to set up their account.	High	13
2-As a mentee I want to be able to edit my profile information after setting up my account. Iteration 1 - did it w4-w5	• If a mentee's account is setup, they should be able to edit the account • No one should be able to edit an account that isn't theirs • A mentee shouldn't be able to edit their account if it hasn't finished being set-up	High	8
3-As a mentor I want to be able to edit my biography at any time. Iteration 1 - did it w4-w5	• If a mentors account is setup, they should be able to edit their biography • Mentors shouldn't be able to edit other mentor's biographies, only their own.	High	8
4-As a mentee I want to be able to use filters when searching for a mentor to save time finding the most suitable mentor for me. Iteration 1 - did it w6-w7	• When I use a filter for a specific subject field, only mentors from that field should be displayed. • If I use no filter, all mentors should be available to see. • If I try to use a filter that no mentors fit into, it should let me know.	High	8
5-As an administrator, I want to be able to change a user's password when they request it. iteration 2 - did it w10	• If a mentee requests a password change, the admin should be able to access the mentee's profile and change their password.	High	4
6-As an admin, I should receive a code from the university in order to register. Iteration 1 - did it w7	• Prior to registration, the university should send codes to anyone they want to register as an admin. • When registering for an admin account, there should be a field to enter this code. • Admin registration should only happen if this code matches with one on a database.	High	3
7-As a user, I want to be able to choose between 3 registration pages depending on which type of user I am	• When a user is creating an account, they should have the option to choose between 3 different account types, and be sent to a matching registration page based on their choice	High	2

Iteration 1 - did it w2-w3			
8-As a mentee I want to be notified by email when a mentor accepts me iteration 2 - did it w10	• If a mentor accepts me, an email should be sent to me.	Medium / High	5
9-As a mentee I want to be able to decline my current mentor so I can contact another one. iteration 2 - did w10	• If a mentee has accepted a mentor already, there should be an option to cancel it. • Selecting the option should remove that mentor from being assigned to them and free up that mentor for other people. • If a mentee has cancelled a match-up with a mentor, other mentors should be available to request.	Medium /High	5
10-As a mentor, I want to be able to accept and decline mentees based on my availability and criteria. iteration 2- did it w9	• If a mentee requests a mentor, there should be an option for the mentor to decline the request.	Medium / High	5
11-As a mentee I should be able to contact my mentor by email. iteration 2- did it w10	• A request-contact button should be available for mentee after mentor accepts him/her. • A request-contact button should send an email to the mentor, sent using the system's email with an auto generated message.	Medium /High	3
12-As an administrator, I want to be able to be able to see a list of all users and basic personal information. iteration 2-did it w9	• If logged in as an administrator, a list should be available of all users and their profiles.	Medium	8
13-As an admin, I should be able to see how many requests a mentee has made, and be able to stop them if they're abusing the system. iteration 2-did it w10	• When logged in as an admin, alongside users should be a history of their requests, the problematic students being coloured in red and a way of blocking them from making any more requests.	Medium	5
14-As a user, I want to be able to change my email address Iteration 1 - did it w4-w5	• If a user needs to change their email address, there should be the option to do this in their profile. • They see the edit profile button that takes them to a form and their email address is updated in the database as well once it was changed.	Medium	5

15-As a mentor I need my email to be hidden until I accept the mentee , after that he or she can email me. iteration 2-did it w9	• If a mentee looks at my profile, no email should be displayed. • If I accept a mentee, when they look at my profile, an email should be displayed. • If I decline a mentee, my email should be hidden to them	Medium	2
16-As a user, I want to be able to reset my password. This can be via contacting the admin rather than automatic process. iteration 2-did it w10	• An admin email should be available on the site to contact an admin for a password change.	Medium	1
17-As a mentee or mentor I want to be able to keep my email private until the match has been made iteration 2-did it w9	• If a mentor's profile is looked at by a mentee who is not their mentee, their email address should be hidden. • If a mentor's profile is looked at by their mentee, all relevant information should be available.	Medium / Low	3
18-As an administrator I want to be notified when a mentee gets declined more than three times so that I can check the mentee's profile iteration 2-did itw10	• If a mentee has more than 3 declines, admins should be notified. • If a mentee has 3 or less declines, there should be no emails regarding that mentee	Low	8
19-As a mentee, I want to be able to access a help page with frequently asked questions. iteration 2-did it w9	• If I am logged in as a mentee, there should be a link to a help page	Low	5
20-As a mentor I want to be able to respond to requests or through the website. iteration 2-did it w9	• If a mentor gets a request, a new option should show up to respond through the website one time. • If a mentor hasn't been requested by a mentee, there should be no way of contacting the mentee through the website.	Low	5
21-As an administrator, I want to be able to update the help page easily. iteration 2-did it w10	• As an administrator, I should be able to update the help page so any answers that can be useful to other mentees can be published. This includes any questions that the mentees email the administrators directly especially ones which are frequently asked.	Low	4
22-As a mentee, I want to be able to contact the administrator in case I	• If logged in as a mentee, there should be access to an administrator email at all times.	Low	3

need support with any issues I encounter on the website or with any mentor iteration 2-did it w10			

Alteration reasonings:

As per previous feedback, we were advised to use mentee consistently rather than student and mentee.

Story 6- We added a code verification to make sure only valid users can register as an admin.

Story 11- We wanted to make a way for the mentee to be able to connect with the mentor via notifying them to go check the website.

Story 13- We dedicated just to notify the admin when there's a problematic student and they would contact them by email, so no way of banning the student through the app.

Story 14- As per feedback, it wasn't clear how we are going to be able to make the user change email addresses so we added another acceptance criteria to clarify.

Burndown chart:

The above is very difficult to read. You could have used other ways of highlighting in green

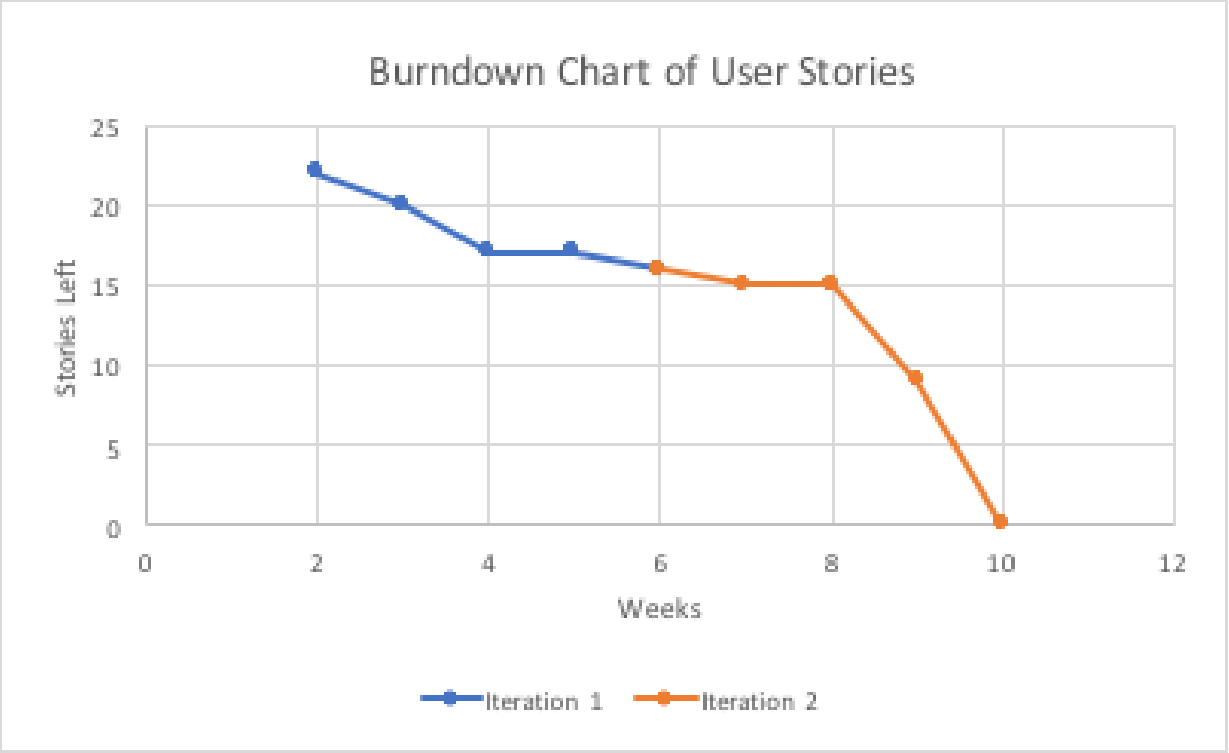

Figure 1: Burndown Chart showing the user story completion over Iteration 1 and 2

Testing and Test Coverage:

The vast majority of testing done was unit and acceptance testing, with a few manual test exceptions. For acceptance testing, the way we went about it was to go through the acceptance criteria of each story and to write one or possibly a number of tests to encompass what the project needed to be showing to the user in order to be meeting these

criteria. All acceptance tests are named for their purpose in the general form: ‘story_[number]_spec.rb’, where the number relates to the story specified in the table above, making it easy to cross-reference what each test is for without needing a large description attached to each of them. Due to this, it is easy to run specific acceptance tests with the command: ‘rspec spec/features/story_[number]_spec.rb’.
The manual tests needed were for the stories: 8 and 18. Both of these tests revolve around a user being sent emails containing information pertinent to the system. As these cannot be checked for in a ‘unit’ manner, these have to occasionally be checked for success manually by team members.
For story 8, the manual test involves creating a mentee and mentor account on the system, logging in as a mentee and requesting said mentor, then logging in as the mentee and accepting the mentor’s request. Doing this should cause the mentee to receive an email that their request has been accepted. This can fail if the email server is down or taking a while to send the email.
The story 18 test is quite similar; It involves creating a mentee account, three mentor accounts and an admin account in the system. The mentee needs to log in and request all three mentor accounts, after which the mentors all need to reject the requests. Doing this should send an email to the admin account though this could fail for the same reasons as story 8.
As of 10/05/2021, the output from the acceptance tests is:

```
codio@victor-armada:~/workspace/project$ rspec spec/features
.........................................

Finished in 30.41 seconds (files took 0.67504 seconds to load)
41 examples, 0 failures

Coverage report generated for RSpec to /home/codio/workspace/project/coverage. 499 / 540 LOC (92.41%) covered.
```

There are no acceptance tests that fail, as every story is implemented.

The unit tests are named in correspondence to the file or the function that they are testing, currently the only failing tests correspond to the requests system as we couldn’t get unit tests working for it, however it is tested with capybara in the above acceptance tests.

As of 10/05/2021, the output from the unit tests is:

```
Finished in 14.66 seconds (files took 0.70481 seconds to load)
72 examples, 4 failures

Failed examples:

rspec ./spec/unit/requests_mentor_spec.rb:41 # Requests Page POST /acceptMethod accepts a request
rspec ./spec/unit/requests_mentor_spec.rb:48 # Requests Page POST /acceptMethod can't accept a request with wrong mentor_id
rspec ./spec/unit/requests_mentor_spec.rb:71 # Requests Page POST /rejectMethod rejects a request
rspec ./spec/unit/requests_mentor_spec.rb:78 # Requests Page POST /rejectMethod can't accept a request with wrong mentor_id

Coverage report generated for RSpec to /home/codio/workspace/project/coverage. 408 / 540 LOC (75.56%) covered.
```

Exemplar Code:

```
68                              <option value="Retail">Retail</option>
69                              <option value="Sport">Sport</option>
70                              <option value="Transport">Transport</option>
71                              <option value="Tourism">Tourism</option>
72                          </select>
73                          <label for="company">Edit company</label>
74                          <input type="company" name="company" id="company" value="<%= h @mentor.company%>" required>
75                          <label for="biography">Edit your biography:</label>
76                          <textarea name="biography" id="biography" rows="10" cols="40" required><%= h @mentor.biography%></textarea>
77                          <button type="submit" action="submit" id="button" name="save"> save changes</button>
78                      </form>
79                  <script>
80                      var submitButton = document.getElementsByClassName('button')[0]
81                      var accdetails = document.getElementsByClassName('accDetails')[0]
82                      submitButton.addEventListener('click',function(a){accdetails.classList.toggle('active')})
83                  </script>
84              </section>
85          </main>
86  </div>
87  <%= erb :"commons/footer" %>
```

Figure A - Sinatra views (views/my_account_mentor.erb line 68-87) – Editing account details

```
100                 </div>
101             <% if @mentors.count > 0 %>
102             <div class = "mentorsList">
103                <% (@mentors).each do |mentor| %>
104                   <% if !@already_req.include? mentor[:id] %>
105                       <div class="mentor">
106                             <section id="mentor-top">
107                                 <form method="post" action="/requestMethod" >
108                                     <button id="request" type="submit">Request</button>
109                                     <input type="hidden" id="mentorId" name="mentorId" value=<%= mentor[:id] %> >
110                                     <input type="hidden" id="mentorEmail" name="mentorEmail" value=<%= mentor[:email] %> >
111                                 </form>
112                             </section>
113                             <section id="mentor-bottom">
114                                 <aside>
115                                     <div id="profile">
116                                         <figure>
117                                         <img src="/mentor.jpg" alt="personal Id picture">
118                                         </figure>
119                                     </div>
120                                 </aside>
121                                 <main>
122                                  <h1> <%= mentor[:first_name] %> <%= mentor[:surname] %> </h1>
123                                  <p>Industry: <%= mentor[:industry] %></p>
124                                  <p>Degree field: <%= mentor[:degree_field] %></p>
125                                  <p>Bio: <%= mentor[:biography] %></p>
126                                 </main>
127                             </section>
128                         </div>
129                <% end %>
130             <% end %>
131             </div>
132             <% else %>
133                 <p>Your search revealed no Mentors.</p>
134             <% end %>
135         </div>
136     </main>
137 </div>
138
139 <script>
140  var submitButton = document.getElementsByClassName('button')[0]
141  var accdetails = document.getElementsByClassName('accDetails')[0]
142  submitButton.addEventListener('click',function(a){accdetails.classList.toggle('active-form')})
143 </script>
```

in my opinion, both listing A and B shouldn't be implemented as views. They are part of the backend and should be dealt by controllers.

Figure B - Sinatra views (views/find_mentor.erb line 100-143) - Filtering through mentors and displaying them

```ruby
    describe "Index Form Submission" do

        before(:all) do
            create_admin_model
        end

        after(:all) do
            clear_db
        end

        it "Redirects on a success" do
            post "/", "username" => "user", "password" => "test1234"
            expect(last_response).to be_redirect
            expect(last_response.location).to include('/my-account-admin')
        end

        it "Fails when there's no password" do
            post "/", "username" => "user"
            expect(last_response.body).to include('Username/Password combination incorrect')
        end

        it "Fails when there's no username" do
            post "/", "password" => "test1234"
            expect(last_response.body).to include('Username/Password combination incorrect')
        end

        it "Fails when there's no parameters" do
            post "/"
            expect(last_response.body).to include('Username/Password combination incorrect')
        end

        it "Fails when a false username is inputted" do
            post "/", "username" => "us3r", "password" => "test1234"
            expect(last_response.body).to include('Username/Password combination incorrect')
        end
    end
```

Figure C - Unit test for logging into the index page (spec/unit/index_spec.rb line 16-51)

```ruby
post "/change-password" do

    #Gets the ID/new password of the user
    user_data = Hash.new
    user_data["id"] = params.fetch("id", "").strip
    id = user_data["id"]
    user_data["password"] = params.fetch("password", "").strip
    password = user_data["password"]


    user = User[id] if User.id_exists?(id)

    #Updates the correct database for that user
    if user[:user_type] == "mentee"
        mentee = Mentee[id]
        mentee.load_passw_change(params)
        user.load_passw_change(params)

        mentee.save_changes(:validate => false)
        user.save_changes
        redirect "/all-mentees"
        erb :all_mentees
    end

    if user[:user_type] == "mentor"
        mentor = Mentor[id]
        mentor.load_passw_change(params)
        user.load_passw_change(params)

        mentor.save_changes(:validate => false)
        user.save_changes
        redirect "/all-mentors"
        erb :all_mentors
    end
```

Figure D - Ruby Controller (controllers/change-password.rb) Changing a user's password by the admin

```ruby
require_relative "../../helpers/spec_helper.rb"

describe "The Filter System" do

    it "shows all available mentors when no filter is selected" do
        # Creates two new mentors in the system, that are both distinct and can be tested for
        register_mentor "Mister","Planes","planes123","Male","Aerospace","Aerospace","planes123@test.com"
        register_mentor "Miss","Plants","plants123","Female","Agriculture","Agriculture","plants123@test.com"

        register_mentee "MenteeFirst","MenteeSur","TestMentee","Male","1st","mentee123@test.com"
        log_in "TestMentee"
        click_link '> Find A Mentor'
        expect(page).to have_content "Mister"
        expect(page).to have_content "Miss"
    end

    # Check each filter in turn
    it "filters by gender" do
        log_in "TestMentee"
        click_link '> Find A Mentor'
        click_button "Filters"
        choose "Male"
        click_button "Apply filters"
        expect(page).to have_content "Mister"
        expect(page).not_to have_content "Miss"
    end

    it "filters by industry" do
        log_in "TestMentee"
        click_link '> Find A Mentor'
        click_button "Filters"
        select "Agriculture", from: "industry"
        click_button "Apply filters"
        expect(page).not_to have_content "Mister"
        expect(page).to have_content "Miss"
    end

    it "filters by degree field" do
        log_in "TestMentee"
        click_link '> Find A Mentor'
        click_button "Filters"
        select "Aerospace", from: "degree_field"
        click_button "Apply filters"
        expect(page).to have_content "Mister"
        expect(page).not_to have_content "Miss"
    end

    it "displays a message if no mentors are available with the specified filters" do
        log_in "TestMentee"
        click_link '> Find A Mentor'
        click_button "Filters"
        select "Tourism", from: "industry"
        click_button "Apply filters"
        expect(page).not_to have_content "Mister"
        expect(page).not_to have_content "Miss"
        expect(page).to have_content "no Mentors"
        clear_db
    end
end
```

Figure E - Acceptance test for the filtering system (spec/features/story_four_spec.rb)

```ruby
post "/acceptMethod" do
    #Updates mentee's DB details
    id = session[:id]
    mentee = Hash.new
    mentee["menteeId"] = params.fetch("menteeId", "").strip
    mentee_id = mentee["menteeId"].to_i
    mentee = Mentee[mentee_id]
    mentee[:has_mentor] = 1
    mentee[:mentor_id] = id
    mentee.save_changes(:validate => false)

    #Updates mentor's DB details
    mentor = Mentor[id]
    mentor[:number_of_mentees] += 1
    mentor.save_changes(:validate => false)

    #Updates request DB
    @requests = DB[:requests]
    @requests.each do |request|
        if request[:mentee_id] == mentee_id && request[:mentor_id] == id
            request = Request[request[:id]]
            request[:accepted] = 1
            request.save_changes
        elsif request[:mentee_id] == mentee_id
            request = Request[request[:id]]
            request.delete
        end
    end

    #Sends notification email to mentee
    require "net/http"

     def send_mail(email, subject, body)
       response = Net::HTTP.post_form(URI("http://www.dcs.shef.ac.uk/cgi-intranet/public/FormMail.php"),
                                      "recipients" => email,
                                      "subject" => subject,
                                      "body" => body)
       response.is_a? Net::HTTPSuccess
     end

     mentee["menteeEmail"] = params.fetch("menteeId", "").strip
     email = mentee["menteeEmail"]

     subject = "Accepted request"

     body = "One of your requests was accepted. Go check it on the website."


     send_mail(email, subject, body)

    redirect "/requests-mentor"
end

```

Figure F - Ruby Controller (controllers/requests_mentor.rb line 23-75) - Accepting method from the controller for mentor's request page