Group 22: COM1001 Team Report

Testing:
As in the README file, it is important to change the directory to the project via the command: cd ~/workspace/project
You must bundle install all gems, again as shown in the README
From here, you can run the application. However, for testing you must type “rspec spec”
You can run specific tests by changing the directory to the test directory “cd ~/workspace/project/spec”, and running command “rspec test.rb”

Stories:
As a user

- I can create a mentor/mentee account with a username, password, and an email
  - A user will set up an account with personal details and be able to enter it again after leaving the site
- I can log in with my account
  - A user will be able to input their username and password and be taken to their personal profile
- My log in details and personal information will be stored securely
  - ~~Users will not have their account accessed by anyone else unless through human means (i.e., giving out their password)~~
  - ~~-~~ Details need not be stored with encryption but are kept on a database only accessible by admins. **This change was made due to a lack of understanding of the language**

As a mentee

- I can input contacts/personal details such as where they have studied/what they have studied so that mentors can assess my ability.

    - A mentee inputs their personal information, it can be viewed by mentors, and it persists once they log off and back in again
- Be able to view and edit their profile so that my information can stay up to date.
    - A mentee will be able to see their own profile, make changes, and have it all persist when re-entering my profile, as well as mentors being able to see changes
- I can search for mentors through a search bar/filtration system, so I am able to find mentor profiles to view.
    - A mentee can input names/ companies or areas of interest and find what they were looking for. This deletion came as it was unlikely that mentees would in fact search for names or companies, as mentees will be mostly concerned with the mentor having the same interests
- I am able to filter searches for mentors so that they meet my preferences
    - There is a way to search by common tags and different tags can be applied to different mentors. **This was deleted due to a lack of time and being very similar to the above story.**
- I can request to contact mentors on the mentors' page so that I can alert them to my interest
    - A mentee will be able to send mentors a request which shows up on the mentors' account
- I can contact admins for account issues so if I have a problem with my account (e.g.: loss of password) they can rectify this.
    - Admins get contacted with a personalised message.
    - **This was fairly unnecessary and could be completed with an email instead, which we added to the footer. This change was mostly made for time's sake**
- I can accept or reject a mentor after an initial meeting
    - As above, other party will be notified
- I am notified if I am rejected or accepted by the other party

    - As above
- My account will only be created if my university is whitelisted
    - **A change made after additional requests from the clients**

As a mentor

- I can view and update my profile in case details change
    - A mentor will be able to see their own profile, make changes, and have it all persist when re-entering my profile, as well as mentors being able to see changes
- I can organise an initial meeting, after a mentee has reached out, externally
    - A mentor needs to be able to find mentee personal contact information
- I can accept or reject a mentee after initial meeting
    - As above, other party will be notified
- I can view a mentee's profile that has reached out to me
    - A mentor can see all the details of a mentee's profile page, so long as they have reached out to them.
- I can contact admins for account issues so if I have a problem with my account (e.g.: loss of password) they can rectify this.
    - Admins get contacted with a personalised message.
    - **This was fairly unnecessary and could be completed with an email instead, which we added to the footer. This change was mostly made for time's sake**
- I am notified if I am rejected or accepted by the other party
    - As above
- As a mentor I can have a list view of all of my mentees
    - The list is updated consistently and will update when necessary

As an admin (generally a low priority since not needed for the system as a whole)

- I can view list of mentor/mentees so as to see all accounts

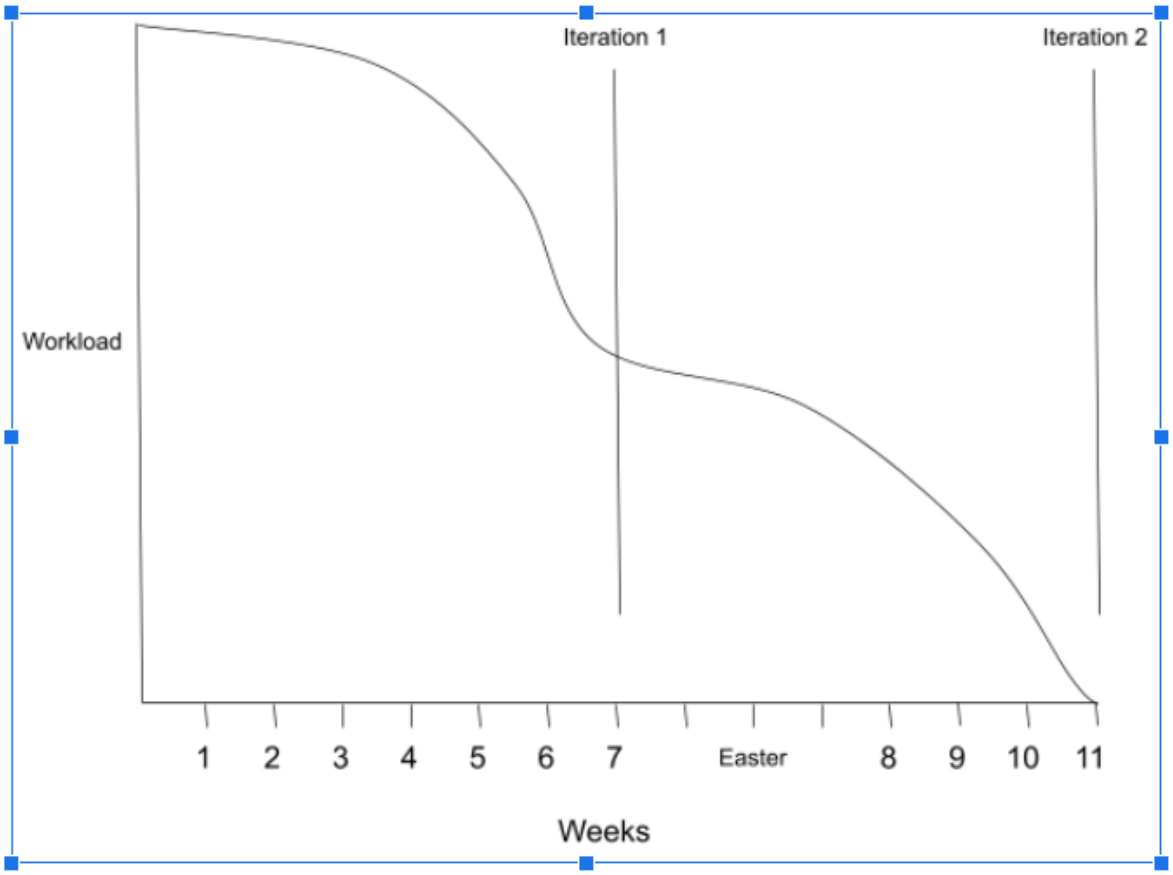
    - As above
- I can search for mentors/mentees through a search bar to find specific accounts
    - As above
- I can view account profiles to see mentee and mentor profiles in detail
    - As above
- I can edit account profiles to change things if necessary
    - As above, details will be changed
- I can delete account profiles if necessary
    - As above, accounts will no longer be able to be accessed
- I can handle password reset requests
    - As above, passwords can be reset

In the first iteration, we had planned to get very barebones work done, mostly databases sorted out. Jordan mostly focused on setting up the database and ruby GEMS along with Cameron, who also created the log in and out features and dashboard. George created the CSS and styling for the web page, as well as sorting out a few pieces here and there with the initial set up of the codio project. Daniel focused on the profile page display, as well as editing personal details. Baraka was set for starting the search function in Easter, after iteration 1. Ruixuan and Zixiang both worked on exemplar code.

In the second iteration, Jordan worked on email validation for different universities as well as the README. Cameron worked on the notification system for mentors, as well as the acceptance and rejection system. Daniel worked on the search system, viewing mentor profiles (using code Cameron had created for viewing mentees), and minor tweaks to how the website worked for each different account type (mentees, mentors, and admins). George again helped with editing the overall look and feel of the website with CSS styling and help with bug fixes. Ruixuan and Zixiang both worked on creating

the tests for the system overall, making sure everything worked. Baraka made minor adjustments to the CSS styling sheet as well minor bug fixes.

Burndown chart:



Test Coverage:

This portion of the report and code was covered by Ruixuan and Zixiang, who dealt with the testing for the

add_spec

This is a file stored in the coverage. The function of this file is to create a new user into the project and detect that a new user without detail cannot be added. When the program detects that the user does not exist in the project, you can add a new user with complete information to the project at this time.

delete_spec

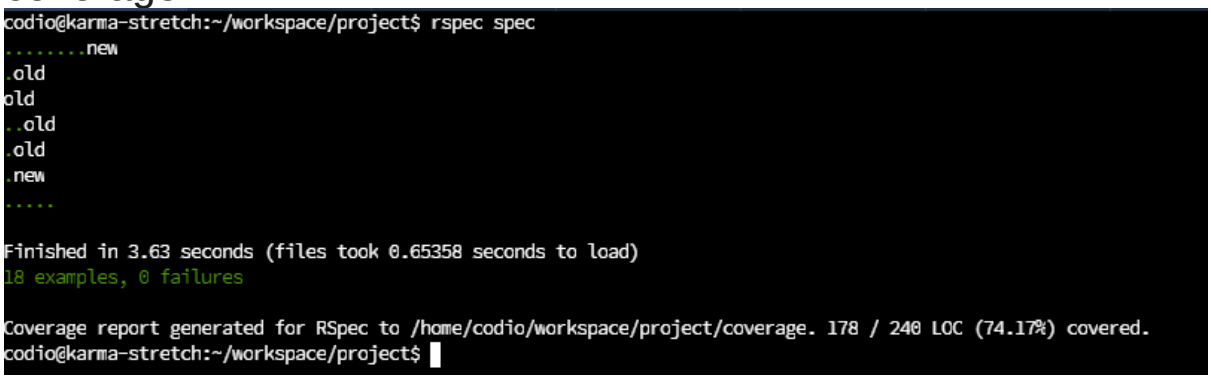
The purpose of this file is to detect whether to delete a user.

Login_spec

This file is used to check whether the user's username and password information is correct. Prevent the user's information from matching errors.

During the course of creating our app, we often created cookies that would initialise on log in that could track variables to different parts of the pages for debugging, as well as having different things displayed on temporary web pages.

Further testing added by Cameron with screenshots to show coverage.



Exemplar Code:

```ruby
get "/matches" do

    #Grabs user type to differentiate between different pages
    @account_type = request.cookies.fetch("account_type","N/A")

    if @account_type == "Mentee"
        @target = "Mentor"
        @title = "Mentor List"

    elsif @account_type == "Mentor"
        @target = "Mentee"
        @title = "Interested Mentees"

    else
        # Admin account goes to usersData
        redirect "/usersData"
    end


    #Search function for mentees
    @AoF_search = params.fetch("AoF_search", " ").strip

    @user = if @AoF_search.empty?
              User.where(Sequel.like(:account_type,"%#{@target}"))
            else
              User.where(Sequel.like(:area_of_focus, "%#{@AoF_search}%"))
            end

    erb :accountData
end
```

Listing A: ~/workspace/project/controllers/accountData.rb

```ruby
get "/request" do
    @mentor = params["mentor"]
    @mentee = request.cookies["username"]
    matches = DB[:matches]
    currentMatches = matches.where(mentee: @mentee, mentor: @mentor)

    if currentMatches.count == 0
     matches.insert(mentee: @mentee, mentor: @mentor, status: "Pending")
     @status = "new"
    else
     @status = "old"
    end

    puts @status

    dataset = DB[:Users].where(username: @mentor)
    @mentorName = dataset.first[:first_name]+" "+dataset.first[:surname]

    erb :request
end
```

Listing B: ~/workspace/project/controllers/request.rb

```ruby
post "/edit" do
  #Searches for user
  id = params["id"]
  @user = User[id] if User.id_exists?(id)

  @account_type = request.cookies.fetch("account_type","N/A")

  id = params["id"]

  #Allows display of all attributes from the user
  if User.id_exists?(id)
    @user = User[id]
    @user.loadParams(params)

    if @user.valid?
      @user.save_changes

      if @account_type == "Mentee" or @account_type == "Mentor"
          redirect "/profilePage"

      else
          redirect "/users"
      end

    end

  end
    erb :edit
end
```

Listing C: ~/workspace/project/controllers/edit.rb

```ruby
def lookup
    userLookup = User.first(username: username)
    if !userLookup.nil? && userLookup.password == password
        return true
    end
    return false
end

def loginValidate
   if username.empty? || password.empty?
       return false
   end
   return true
end

def self.id_exists?(id)
      return false if id.nil? # check the id is not nil
      return false unless Validation.str_is_integer?(id) # check the id is an integer
      return false if User[id].nil? # check the database has a record with this id
      true # all checks are ok - the id exists
end

```

Listing D: ~/workspace/project/models/user.rb

```
<% if @matchesDataset.count == 0 %>
  <p>No current matches</p>
<% else %>
<table>
  <tr>
    <th>Mentee</th>
    <th>Status</th>
  </tr>
  <% @matchesDataset.each do |record| %>
    <tr>
      <td><a href="/view?user=<%= record[:mentee] %>"><%= record[:mentee] %></a></td>
      <td><%= record[:status] %></td>
      <% if record[:status] == "Pending" %>
        <td><a href="/confirmation?mentee=<%= record[:mentee] %>&status=Accepted">Accept</a></td>
        <td><a href="/confirmation?mentee=<%= record[:mentee] %>&status=Rejected">Reject</a></td>
      <% end %>
    </tr>
  <% end %>
</table>
<% end %>

```

Listing E: ~/workspace/project/views/dashboard.erb

```
  <div class="navigation">
    <form>
      <p>Enter an account name:
      <input type="text" name="username_search" value="<%= h @username_search %>"/>
      <input id="submitButton" type="submit" value="Submit" /></p>
    </form>
  </div>
  <% if @user.count > 0 %>
```

Listing F: ~/workspace/project/views/usersData.erb