Team 14
Iteration 2 Report

A. How to run application

1. In the workspace run the command :

```
git clone https://git.shefcompsci.org.uk/com1001-2021-22/team14/project.git
```

2. In the project root repository run the command :

```
bundle install
```

3. In the project root repository run the command :

```
sinatra -b app.rb
```

The application can only be run in a box where the com1001 repository has been cloned and the install script has been run.
To sign in, use email and password from valid credentials table in section B.
To sign up as a reporter, username and email must not have been used by another account before. The password must have at least 8 characters, at least one uppercase letter and one number.

B. Accounts registered in the system

ID	Username	email	password
1	guest	guest@gmail.com	guest
2	Noah_2022	Noah@gmail.co.uk	Noah_2022
3	admin	admin@sheffield.ac.uk	admin
4	reporter1	reporter1@sheffield.ac.uk	reporter1
5	reporter2	reporter2@sheffield.ac.uk	reporter2
6	moderator	mod@sheffield.ac.uk	moderator
7	viewer1	viewer1@sheffield.ac.uk	viewer1
8	viewer2	viewer2@sheffield.ac.uk	viewer2
9	multiuser	multi@sheffield.ac.uk	multi1234
10	suspendee	suspenededGuy@manchester.ac.uk	susguy123

The accounts with IDs 3 to 8 are the ones specified by the assignment brief. One thing to note is that reporter1 is not an ACME staff (posts need to be approved) while reporter2 is an ACME staff (posts are auto approved and appear to viewers immediately).

The account with ID 1 and username guest is what all guest posts are tied to. This has allowed us to make users post as guests while still logged in. It has also allowed us to make a guest enter their university as opposed to logged in users having their account's university automatically added to the post.

The account with ID 2 has been used for testing different functionalities like making editing user details and suspending users.

The account with ID 9 is to showcase what a user with multiple roles looks like, while the account with ID 10 is a suspended user to showcase what happens when they log in.

C. Story modifications

Below are the stories with deletions in ~~red and crossed out,~~ and insertions or modifications and their reasoning in **bold green**.

Admin Stories
1 . "As an admin, I can login to perform admin tasks." **Acceptance criteria**: - Login details are validated using the account database. - Once logged in, I can see my homepage of issue reports. **Priority**: Must **Estimated Effort**: 3
2. "As an admin, I can view a list of password reset requests." **Acceptance criteria**: - My homepage is a list of reset password requests, ~~which I can click on and expand.~~ **(Expanding Unnecessary, requests are small and short)** - I can see the email and username of who submitted the request. - I will manually handle sending users their new password. - I have a resolved button on the expanded request page, which I can click on after I have resolved the request. - Request disappears after I mark it as resolved. **Priority**: Must **Estimated Effort**: 5

3. “As an admin, I can view a list of bug reports submitted by user”

Acceptance criteria:
- A user can submit a report regarding anything other than password resets from the login page, which I can view.
- Once as report is resolved, I mark it as resolved and it disappears from list of bug reports

Priority: Won't

Estimated Effort: 5

4. “As an admin I can search for a user using email or University.”

Acceptance criteria:
- I have a search for users panel.
- I can click on the user and open their account page which I can edit.

Priority: Could

Estimated Effort: 3

5. “As an admin I can view & edit account details.”

Acceptance criteria:
- Admins can click the username in password reset requests.
- Clicking on a user displays their account details, (except for password.)
- Account details of a user can be changed, including password.

Priority: Must
Estimated Effort: 2

6.“As an admin, I can view a reporter's account page to see all their previous posts.”

Acceptance criteria:
- After opening a reporter’s account, ~~I can click this ‘see posts’ button to open up a new page containing all their previous posts.~~ **(No button now. Admin can click on reporters’ usernames to see their posts)**
- A reporter’s list of posts appear to me as they appear to them.

Priority: Must
Estimated Effort: 3

7. “As an admin, I can delete posts.”

Acceptance criteria:
- Under a reporter, I can see all their previous posts
- Selecting a post displays a button that deletes the post.

Priority: Must

Estimated Effort: 1

8. "As an admin, I can suspend accounts."

Acceptance criteria:
- When viewing account details, there is an option to suspend the account
- Suspended accounts cannot login, nor approve/submit new posts

Priority: Must

Estimated Effort: 1

Moderator Stories

9-"As a moderator I can login so that I can see post approval requests."

Acceptance criteria:
- Given that I enter the correct (username or email) and password, I am given access to my homepage
- If I don't remember my password, I can press a button on the login page which sends me to a "report issue" page.

Priority: Must
Estimated Effort: 3

10- "As a moderator I get a list of post approval requests as my homepage so that I can (accept and publish) or reject them."

Acceptance criteria:
- When I click on a post, ~~the post expands and takes up the whole~~ page. **(posts are small and do not need expansion)**
- I can click on post link which opens in a new browser tab.
- I can click on publish or reject at the bottom of the page.

Priority: Must
Estimated Effort: 5

11- "As a moderator I can edit a post so that it has a more appropriate title or tags."

Acceptance criteria:
- Given that I have clicked on a post, I can click on its title and change it.
- I can remove tags or add tags.
- I can then immediately publish the post after editing.

Priority: Must
Estimated Effort: 5

Viewer Stories

12- "As a viewer, I want to login to see my homepage of the most recent posts."

Acceptance criteria:
- Given that I have entered the correct details, I gain access to my homepage.
- Posts are the most recent with no specific filter applied.
- Total number posts on entire platform is displayed

Priority: Must

Estimated Effort: 3

13- "As a viewer, I can search for keywords that are attached to posts."

Acceptance criteria:
- I will be able to filter for a keyword or multiple from a predefined list.
- The feed will change to display all posts with attached keywords
- The feed total number of posts will change to the number displayed.

Priority : Must

Estimated Effort : 3

~~**14- As a viewer, I can select a post to read the content of the report"**~~

~~**Acceptance criteria:**~~
- ~~Clicking on a report opens the contained link in a new tab~~
- ~~Each report displays the Header/Headline, attached keywords, the University, the name of the reporter, and the date the report was submit~~ted.
- ~~There's a back button that takes me back to my filtered feed .~~

~~**Priority**: Must~~ **(won't)**
~~**Estimated Effort**: 2~~ **(This is again based on expansion of a post after clicking on it. We decided expansion was unnecessary as posts are small and short)**

15- "As a viewer, I can remove filters from my search criteria."

Acceptance criteria:
- I will be able to deselect a keyword or multiple from the applied search criteria.
- The feed will change to display posts that match the search criteria. - The total number of posts will change to the current amount displayed.

Priority: Must
Estimated Effort: 1

~~16- “As a viewer, I will be able to view a history of the posts I have viewed.”~~

~~Acceptance criteria~~~~:~~
~~- The feed will only show posts the user has interacted with.~~
~~- The total number of posts will change to display the number the user has interacted~~ with.

~~Priority~~~~: Could (~~**Won’t)**
~~Estimated Effort~~~~: 2 (~~**Story did not fit within time constraints + it was not required)**

~~17- “As a viewer, I will be able to create a list of pinned posts that interest me.”~~

~~Acceptance criteria~~~~:~~
- ~~Able to pin a post from the feed that gets added to the list of pinned posts~~
- ~~The list will be able to be viewed, changing the feed to these posts - The total number of posts will be changed to the number in the pinned l~~ist
- ~~I can unpin po~~sts

~~Priority~~ ~~: Could (~~**Won’t)**
~~Estimated Effort~~~~: 3 (~~**Story did not fit within time constraints + it was not required)**

18- “As a viewer, I will be able to report a post by clicking on the report button that is under each post.”

Acceptance criteria:
- The report button takes me to a form which allows me to select from a drop down menu why I reported that form.
- Reasons like, hate speech, inappropriate content, etc.
- Report is sent to admins

Priority: Won't
Estimated Effort: 2

Reporter Stories

19- "As a reporter, I want to be able to post as a guest so that I can submit a report without needing to login."

Acceptance criteria:

- Login screen has a button allowing the user to sign-in as a guest.
- I create posts in the same format as logged in reporters.
- My post has "guest" instead of the name of the reporter .
- ~~I must select which university from a preset keyword list.~~ **(University is typed in instead. Moderator can reject or edit post if university value is invalid)**
- Once I have posted, I can no longer see my post.

Priority: Must
Estimated Effort: 3

20- "As a reporter, I want to login so that my reports are linked to my account."

Acceptance criteria:

- I need to type in my password and email.
- Inputs are authenticated using the account details database.
- Relevant error messages are displayed when invalid/incorrect inputs are given.
- Once logged in, I am taken to my homepage which allows me to write a new post.

Priority: Must
Estimated Effort: 3

21-"As a reporter, I want to register so that I can create a new account."

Acceptance criteria:

- Four input boxes are displayed: Name, Password, University, email.
- Inputs are validated (e.g. using specified Password Policy)
- Relevant error messages are displayed when inputs are invalid
- User is taken to the login screen once the account is created successfully.

Priority: Must
Estimated Effort: 3

22- "As a reporter, I want to create and submit new reports."

Acceptance criteria:

- New reports should have the following features:
  1. Header / Headline
  2. Keywords / Tags (selected from a preset list)
  3. Content (a link to an existing social media post)
  4. **University (automatic if logged in. Guest types it in)**
  5. **Optional company (Typed in for each post) (new requirement)**
- The report is validated (ensuring each feature has been inputted/selected) before submission.

Priority: Must

Estimated Effort: 5

23- "As a reporter, I want to view a list of preset keywords that I can tag my reports with."

Acceptance criteria:

- When uploading a post, there should be a box where reporters can add in any keywords (from a preset list) that will help viewers filter posts to provide more relevant ones when certain keywords are searched.
- ~~One of the required 'keywords' for each post should be the University name~~ as posts should be filtered by University. This is auto set according to reporter's account. **(University is a field shown with the post but not in the same way a tag is.)**
- There should be two keyword categories; Uni name and topic.
- I can select and deselect keywords.

Priority: Should
Estimated Effort: 2

24- "As a reporter, I want to view a list of only my own previous posts so that I can edit or delete them later on if I choose to."

Acceptance criteria:

- Should be an area where all previous posts made by reporter should be listed, the newer posts being higher up
- Should have an icon that provides the ability to edit posts (which will then have to then be reapproved) or delete posts .
-

Priority: Should
Estimated Effort: 3

25- "As a reporter who is an acme staff, I want my posts to be posted immediately without being sent to the moderator for approval."

Acceptance criteria:

- Only logged in acme staff reporters have this functionality.
- If my account says that I work at acme, my posts are auto approved.

Priority: Should

Estimated Effort: 1

General constraints and Non-Functional Requirements

26-“As a logged in user who works at acme, I can have multiple roles.”

Acceptance criteria:
- I can switch between those roles by switching panels.
- Each panel has the functionality of only its own role .

Priority: Must
Estimated Effort: 2

27- “As a user of any type, I have a panel which allows me the role of a guest reporter so that I don’t have to logout to report as a guest.”

Priority: Should
Estimated Effort: 3

28- “As a moderator, admin, or viewer, I will be handed my credentials and won’t need to sign up.”

Priority: Must
Estimated Effort: 0

29- “As a user of any type, if I forget my login details I want to submit an account reset request from the login page.”

Acceptance criteria:
- Entering incorrect details does not allow me to access to my account.
- Clicking a forgotten password button will send a request to the admin panel with the username/email typed in the username box.
- Once admin resets, I will be able to login in with the new correct details.

Priority: Must
Estimated Effort: 3

30-“As a user of any type, I have an account page. I can change my email, password or username from this page.”

Acceptance criteria:
- My account page lists all my details except my password.
- Each detail has an edit button next to it. There’s a change password button as well.
~~- Each detail must be typed in correctly twice to confirm change.~~ **(Too inconvenient and user can just edit again. Only password is typed in twice)**

Priority: Must
Estimated Effort : 2

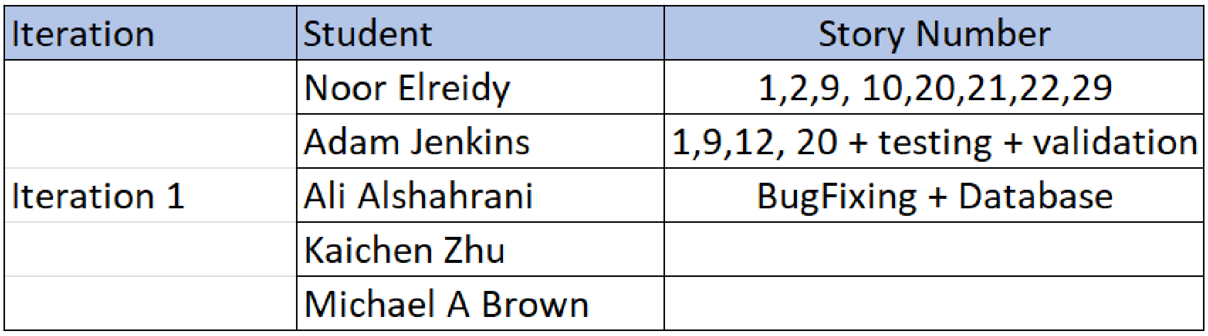
31- “Keywords that are used to tag posts are a hardcoded, pre-set list.”

Acceptance criteria:
- A list of pre-determined keywords can be selected from by the reporter to categorise each of their reports.

Priority: Should
Estimated Effort: 2

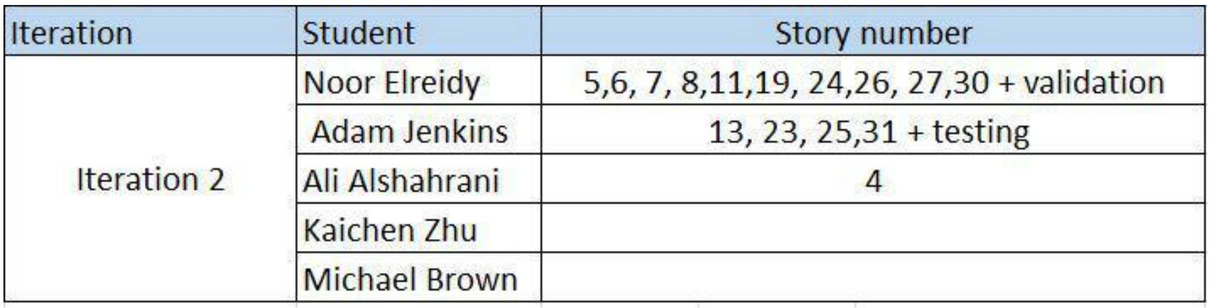
D. Story implementation progression

Stories have been numbered from 1 to 31 in the section above. This section shows which iteration stories have been implemented in and who implemented them. Stories not implemented by anyone are general constraints and requirements that do not need implementation. Figure 1 below shows iteration 1, while figure 2 shows iteration .



Iteration	Student	Story Number
Iteration 1	Noor Elreidy	1,2,9, 10,20,21,22,29
	Adam Jenkins	1,9,12, 20 + testing + validation
	Ali Alshahrani	BugFixing + Database
	Kaichen Zhu	
	Michael A Brown	

Figure 1 : Iteration 1 story implementation



Iteration	Student	Story number
Iteration 2	Noor Elreidy	5,6, 7, 8,11,19, 24,26, 27,30 + validation
	Adam Jenkins	13, 23, 25,31 + testing
	Ali Alshahrani	4
	Kaichen Zhu	
	Michael Brown	

Figure 2 : Iteration 2 story implementation

E. Burn Down Chart

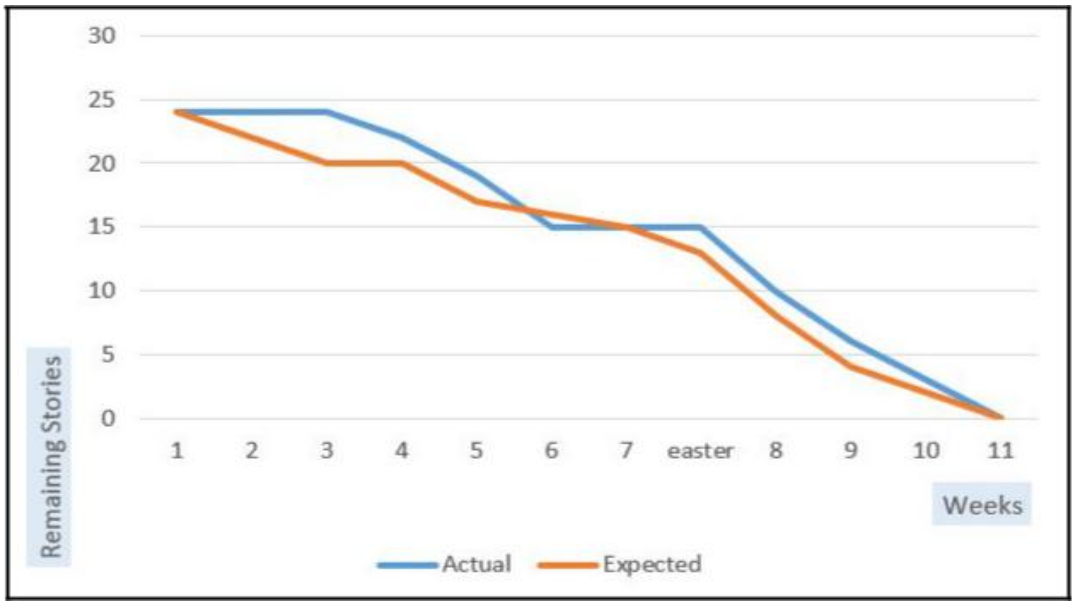


Figure 3 : Burn down chart

F. Testing and Test Coverage

Throughout our development, we used sinatra to run our system. Doing this consistently throughout development gave us good insight into what a user of our system will experience, and made it much easier to spot errors in what is displayed to the user.

To aid our testing, we utilised rspec to perform unit tests on specific methods within our controller code. Specifically, the validation of user input can be a lengthy process to manually execute repeatedly. Hence, we design unit tests to cover a wide range of potential user inputs to ensure that our validation and handling of user inputs operates as expected. The use of unit testing allowed us to repeat all of these tests rapidly whenever new changes were introduced - to avoid breaking previously working systems with new development.

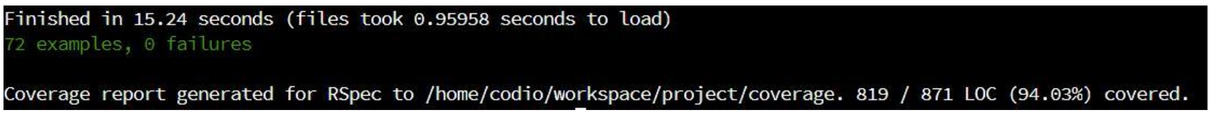
Furthermore, once a feature is completely developed, we utilise rspec and capybara to automatically simulate a user interacting with the system. By creating a large number of feature tests, we can quickly repeat testing on all features whenever we introduce new code to ensure every part of the system still works as expected. By running feature tests throughout development, we were able to avoid large errors that would otherwise have been difficult to notice without performing a large number of manual tests repeatedly - which saved lots of time and increased programming productivity significantly.

How to run the tests:

1. Ensure the test database is set-up.
   - Navigate to the spec folder (file-path: project/spec) and run the following commands:
     - rm test.sqlite
     - sqlite3 test.sqlite
     - .read production.sql
     - .exit
2. Navigate back to the project folder using the following command:
   - - cd ..
3. Finally, execute the tests using the following command:
   - rspec spec/*/*

In total, we have 70 automated tests that execute successfully with no failures.
All tests pass with 94% overall code coverage.

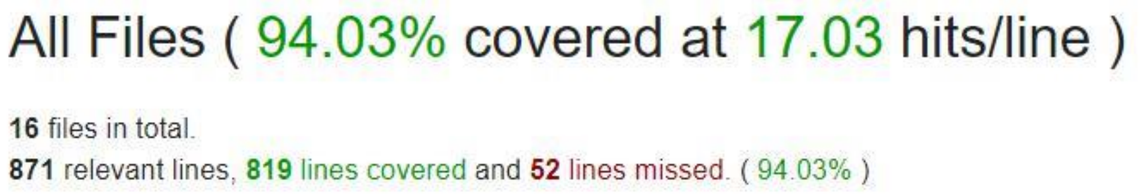
Example Run:



```
Finished in 15.24 seconds (files took 0.95958 seconds to load)
72 examples, 0 failures

Coverage report generated for RSpec to /home/codio/workspace/project/coverage. 819 / 871 LOC (94.03%) covered.
```

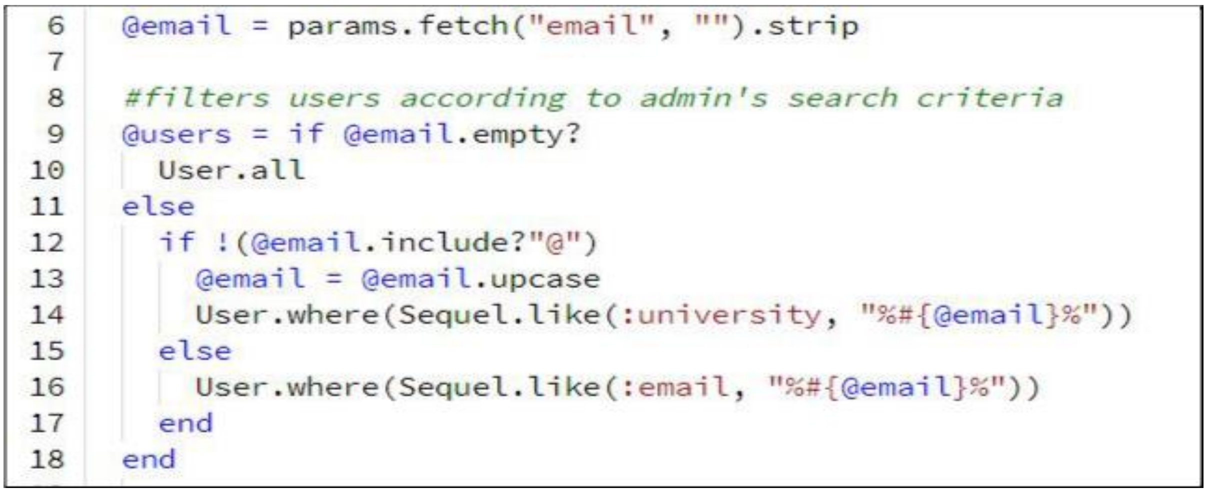
The missing 6% code coverage is mainly error handling code that deals with unexpected/corrupt data. All important features for the system are tested with an average hits/line of 17. It is important to note that this figure is slightly inflated due to the large number of times that core lines of code are repeated (e.g. repeated login of different users to test the full end-to-end use of the system). Despite this, all features are tested thoroughly with the aim of ensuring that no edge cases, unexpected or invalid inputs may cause a feature to crash the system.
In conclusion, we are confident that our tests perform a satisfactory trial of the system to ensure that each feature works as expected.



All Files (94.03% covered at 17.03 hits/line)

16 files in total.
871 relevant lines, 819 lines covered and 52 lines missed. (94.03%)

G. Exemplar Code

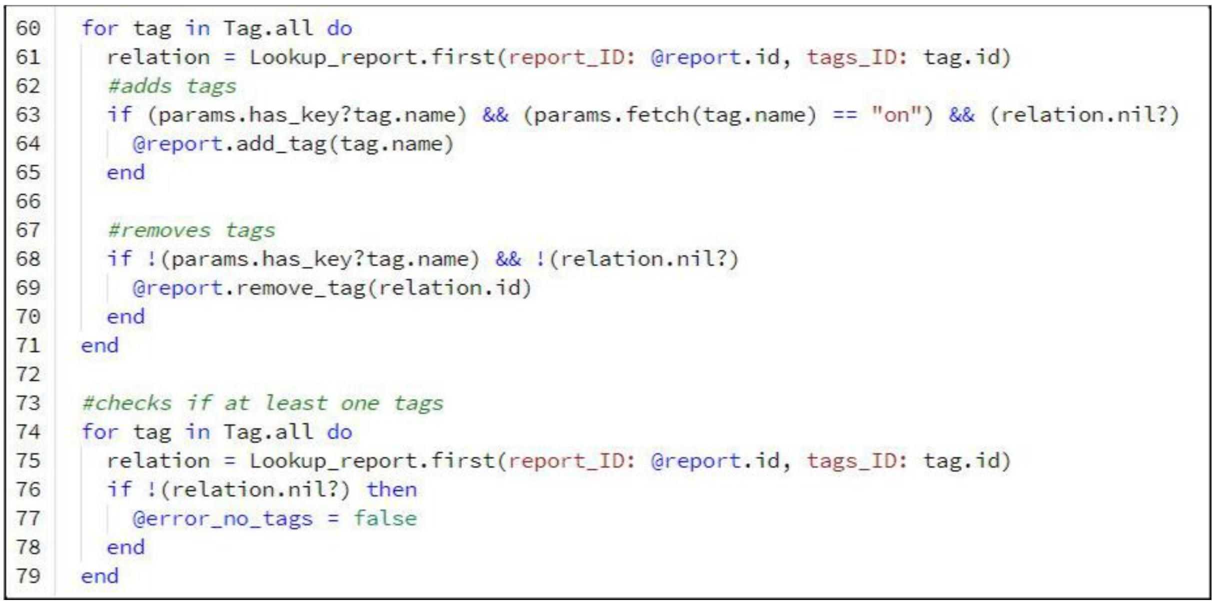


```ruby
@email = params.fetch("email", "").strip

#filters users according to admin's search criteria
@users = if @email.empty?
  User.all
else
  if !(@email.include?"@")
    @email = @email.upcase
    User.where(Sequel.like(:university, "%#{@email}%"))
  else
    User.where(Sequel.like(:email, "%#{@email}%"))
  end
end
```

Listing A : controllers/admin.rb

Listing A shows the code used to apply the admin’s filter to the user list page. If the search value does not contain an “@” , admin is filtering by university. Throughout the application, all university entries are converted to uppercase, so the filtering is not case sensitive.



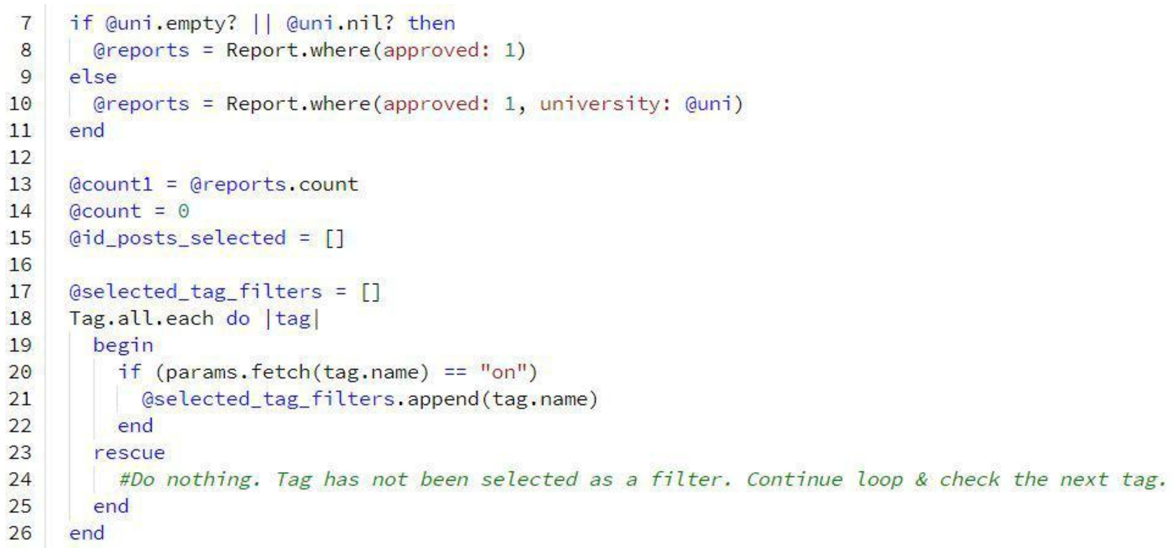
```ruby
for tag in Tag.all do
  relation = Lookup_report.first(report_ID: @report.id, tags_ID: tag.id)
  #adds tags
  if (params.has_key?tag.name) && (params.fetch(tag.name) == "on") && (relation.nil?)
    @report.add_tag(tag.name)
  end

  #removes tags
  if !(params.has_key?tag.name) && !(relation.nil?)
    @report.remove_tag(relation.id)
  end
end

#checks if at least one tags
for tag in Tag.all do
  relation = Lookup_report.first(report_ID: @report.id, tags_ID: tag.id)
  if !(relation.nil?) then
    @error_no_tags = false
  end
end
```

Listing B : controllers /moderator.rb

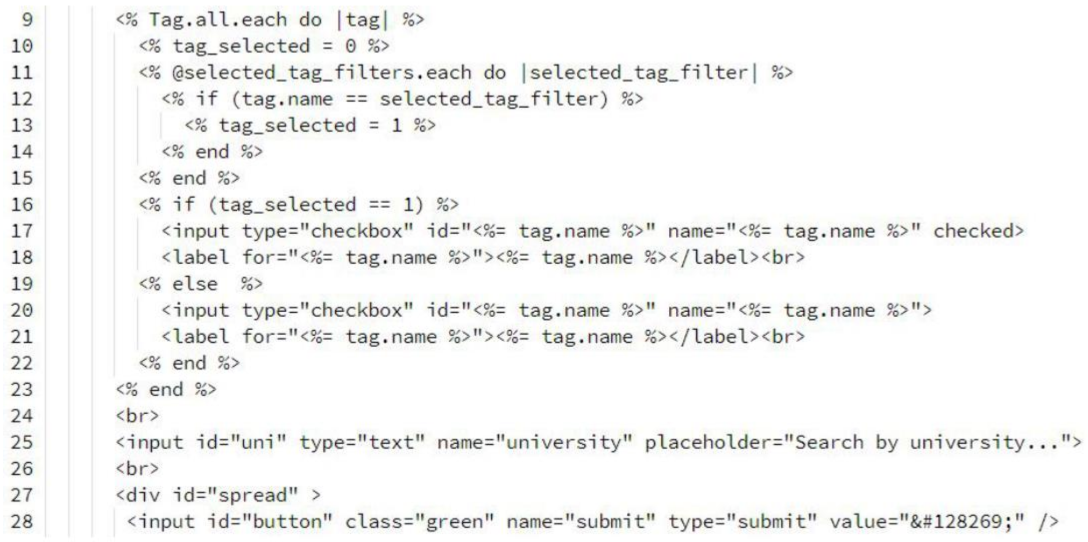
Listing B shows how tags are edited by moderators and their validation. The first if condition in the first loop checks if tag is checked and adds it to post only if there is not already a relation between tag and post (i.e. tag originally added by reporter). The second if condition checks if the tag is not checked and a relation between the tag and post exists (i.e. originally added by reporter but removed by moderator) and removes tag in that case . The second loop loops through tags and sets @error_no_tag to false only if it meets a tag that is related to the post by the table Lookup_report. If the error remains true, a moderator cannot proceed with approving the post.



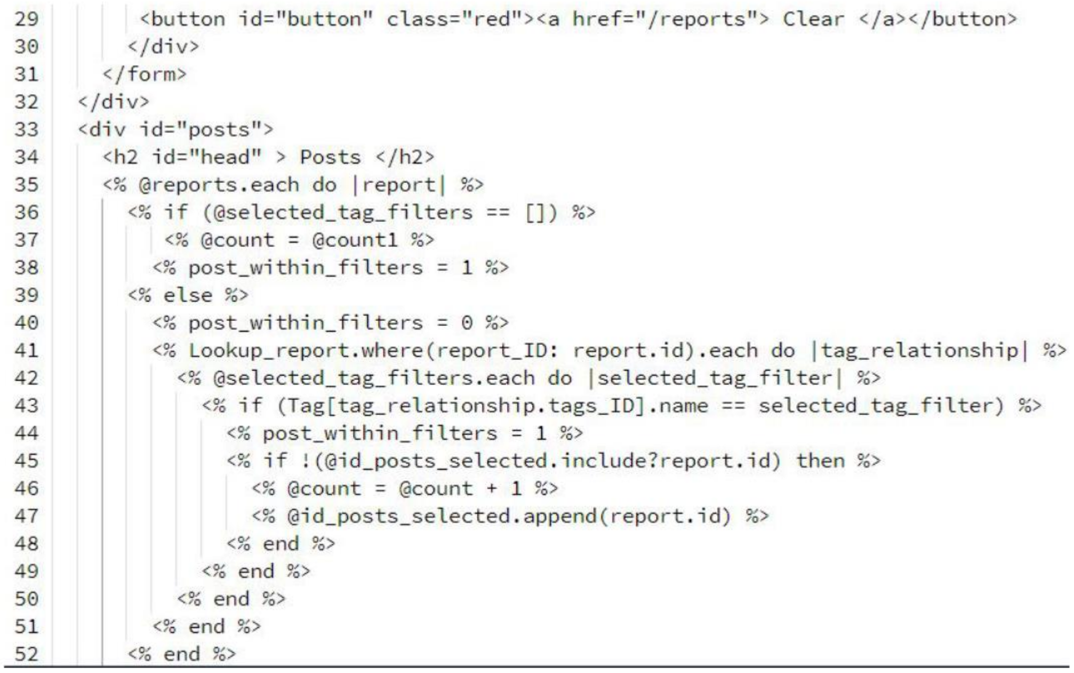
```
 7  if @uni.empty? || @uni.nil? then
 8    @reports = Report.where(approved: 1)
 9  else
10    @reports = Report.where(approved: 1, university: @uni)
11  end
12
13  @count1 = @reports.count
14  @count = 0
15  @id_posts_selected = []
16
17  @selected_tag_filters = []
18  Tag.all.each do |tag|
19    begin
20      if (params.fetch(tag.name) == "on")
21        @selected_tag_filters.append(tag.name)
22      end
23    rescue
24      #Do nothing. Tag has not been selected as a filter. Continue loop & check the next tag.
25    end
26  end
```

List C : controllers/reportsController.rb

Listing C shows how posts are filtered by both university and tags. First, the dataset of reports is set according to if the viewer has typed in a university as a filter. Lines 13 to 15 are used to display the correct post count in the view. Then, the loop checks if a tag has been selected and adds it to array @selected_tag_filters which is then used in the view to display the posts that have those tags. Listing D shows and explains how they are used in the view.



```
9      <% Tag.all.each do |tag| %>
10       <% tag_selected = 0 %>
11       <% @selected_tag_filters.each do |selected_tag_filter| %>
12         <% if (tag.name == selected_tag_filter) %>
13           <% tag_selected = 1 %>
14         <% end %>
15       <% end %>
16       <% if (tag_selected == 1) %>
17         <input type="checkbox" id="<%= tag.name %>" name="<%= tag.name %>" checked>
18         <label for="<%= tag.name %>"><%= tag.name %></label><br>
19       <% else  %>
20         <input type="checkbox" id="<%= tag.name %>" name="<%= tag.name %>">
21         <label for="<%= tag.name %>"><%= tag.name %></label><br>
22       <% end %>
23     <% end %>
24     <br>
25     <input id="uni" type="text" name="university" placeholder="Search by university...">
26     <br>
27     <div id="spread" >
28      <input id="button" class="green" name="submit" type="submit" value="&#128269;" />
```



```
29        <button id="button" class="red"><a href="/reports"> Clear </a></button>
30      </div>
31    </form>
32  </div>
33  <div id="posts">
34    <h2 id="head" > Posts </h2>
35    <% @reports.each do |report| %>
36      <% if (@selected_tag_filters == []) %>
37         <% @count = @count1 %>
38        <% post_within_filters = 1 %>
39      <% else %>
40        <% post_within_filters = 0 %>
41        <% Lookup_report.where(report_ID: report.id).each do |tag_relationship| %>
42          <% @selected_tag_filters.each do |selected_tag_filter| %>
43            <% if (Tag[tag_relationship.tags_ID].name == selected_tag_filter) %>
44              <% post_within_filters = 1 %>
45              <% if !(@id_posts_selected.include?report.id) then %>
46                <% @count = @count + 1 %>
47                <% @id_posts_selected.append(report.id) %>
48              <% end %>
49            <% end %>
50          <% end %>
51        <% end %>
52      <% end %>
```

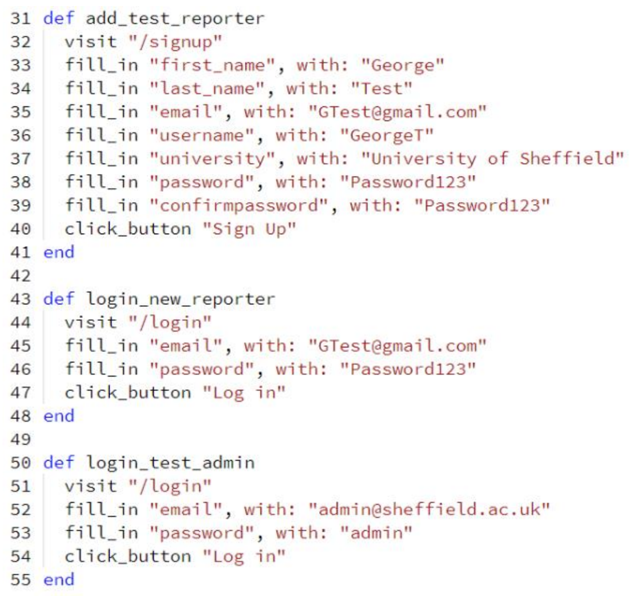
Listing D : views/posts.erb

Listing D shows the embedded ruby code used to display all of the posts to a viewer. The embedded code has two jobs: to display the available filters, and to display all of the posts that come under the selected filters.

Lines 9-29 are where the filters are displayed. Looping through each of the tags in the database, the code begins by displaying each of the available tags as filter options. The code stores the selected filters in the @selected_tag_filters array so that the filters are saved when the page is refreshed.
For the posts themselves, lines 33 onwards in Listing D loop through each of the reports in the database and check if each report has any of the selected tag filters. For posts that have a selected tag filter, that post is then displayed to the user - creating a list of posts that contain only the selected tag filters.

For each post displayed, its index is added to the array @id_posts_selected and the counter is incremented only if its index is not already in the array. This is to prevent multiple increments for posts that meet multiple tag requirements.

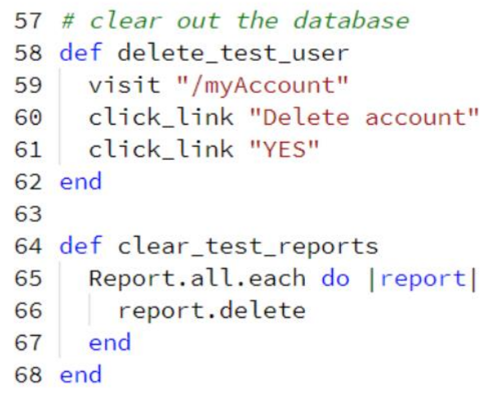
To achieve all of this, nested loops are used to iterate through the database and access related database entries (the reports and their tags).



```ruby
def add_test_reporter
  visit "/signup"
  fill_in "first_name", with: "George"
  fill_in "last_name", with: "Test"
  fill_in "email", with: "GTest@gmail.com"
  fill_in "username", with: "GeorgeT"
  fill_in "university", with: "University of Sheffield"
  fill_in "password", with: "Password123"
  fill_in "confirmpassword", with: "Password123"
  click_button "Sign Up"
end

def login_new_reporter
  visit "/login"
  fill_in "email", with: "GTest@gmail.com"
  fill_in "password", with: "Password123"
  click_button "Log in"
end

def login_test_admin
  visit "/login"
  fill_in "email", with: "admin@sheffield.ac.uk"
  fill_in "password", with: "admin"
  click_button "Log in"
end
```



```ruby
# clear out the database
def delete_test_user
  visit "/myAccount"
  click_link "Delete account"
  click_link "YES"
end

def clear_test_reports
  Report.all.each do |report|
    report.delete
  end
end
```

Listing E : unit/capybara_testing_setup.rb

Using a combination of ruby code and sql triggers, we maintain the test database starting state by removing data created during tests once the tests are complete. This means that the test database will begin in the same state every time the tests are run.
The code shown here is the ruby methods called throughout the feature tests to create and remove the test data. However, the tests require pre existing data. Hence, the database cannot be completely cleared during the test. So, sql triggers are used to delete only the targeted users and the reports that are linked to that user.
This means that pre-existing accounts (e.g. admins & moderators) can be used consistently during the tests, whilst reporter accounts can be created during the tests & deleted once a test is complete to restore the database's initial state.