SE31520 Assignment : Enhancing the CS-Alumni Application

Helen Harman
Student Number : 110007212
Computer Science Department, Aberystwyth University

December 5, 2014

Contents

1 Introduction

This document describes the workings of the CSA application that has been provided to us. A detailed test plan for doing performing cucumber testing on the application has been given.

2 Requirements Analysis

This section contains a description of the functionality of the CSA application, without any implementation details.

2.1 Use Case Diagrams

I have created several use case diagrams to show the current functionality that has been implemented within the CSA application.

2.1.1 Broadcasts

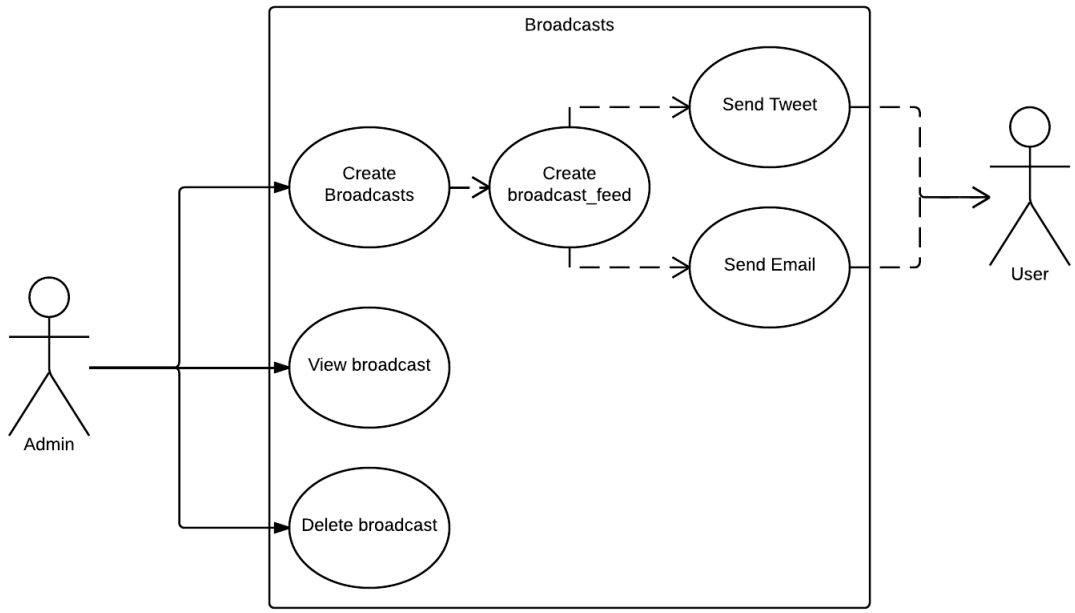


Figure 1: Architecture : Use case for broadcasts.

An admin user should be able to :

- Create any number of broadcasts. This includes broadcast to multiple feeds, including Email and Twitter. A user will then receive the broadcast in an email or/and be able to view the tweet. The admin will choose which feeds they wish to broadcast too.
- An admin should be able to view the list of broadcasts, and view a particular broadcast's content.
- An admin should be able to delete a broadcast.

The admin will interact with the view part of the application, the POST/DELETE/UPDATE action described above will update the model with the correct data.

2.1.2 User

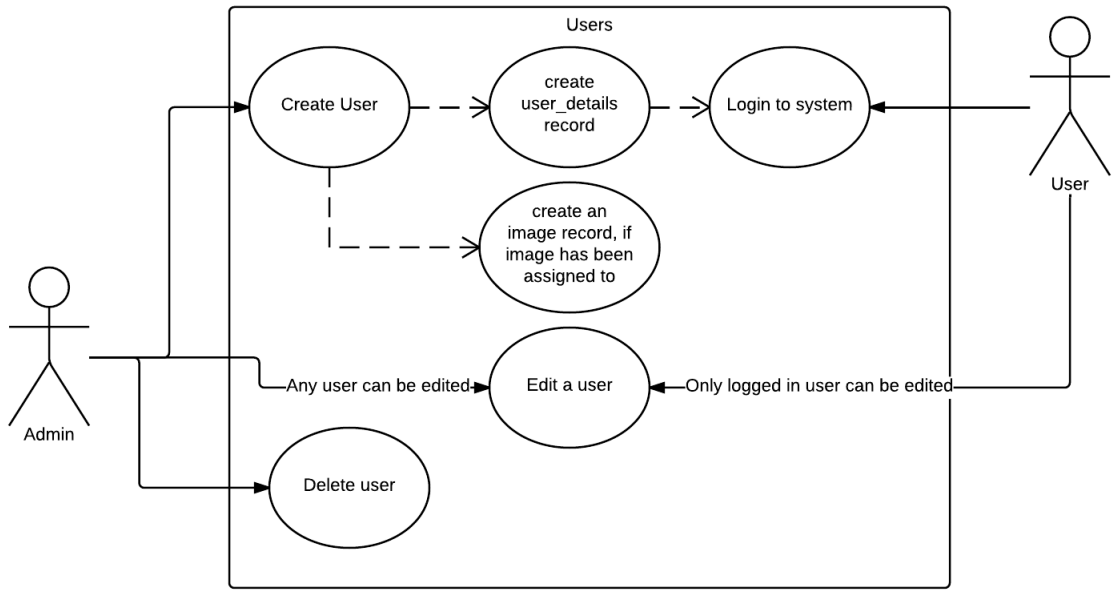


Figure 2: Architecture : Use case for users.

- The admin should be able to create a user, using a web form. Doing this will create a record in the UserDetails model and an image record if an image has been uploaded.
  - A user will then be able to login to the application using the UserDetails the admin has entered.
- An admin should be able to edit any of the users within the application using the web page.
- A user should only be able to edit their own account.
- An admin should be able to delete a user.

2.2 State Diagram

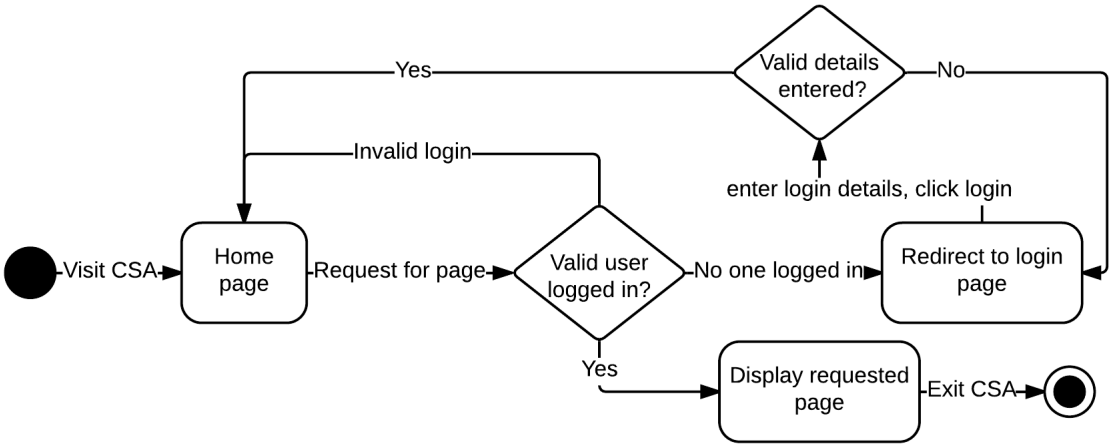


Figure 3: Architecture : State diagram.

The state diagram in figure 3 shows the state of the system when a user accesses the CSA application, requests a page and logs into the application.

3 Design

I have not changed the functionality of the CSA application as I choose to do the cucumber testing, below I have provided a description of Loftus [1, 5] class diagram.

3.1 Class Diagram

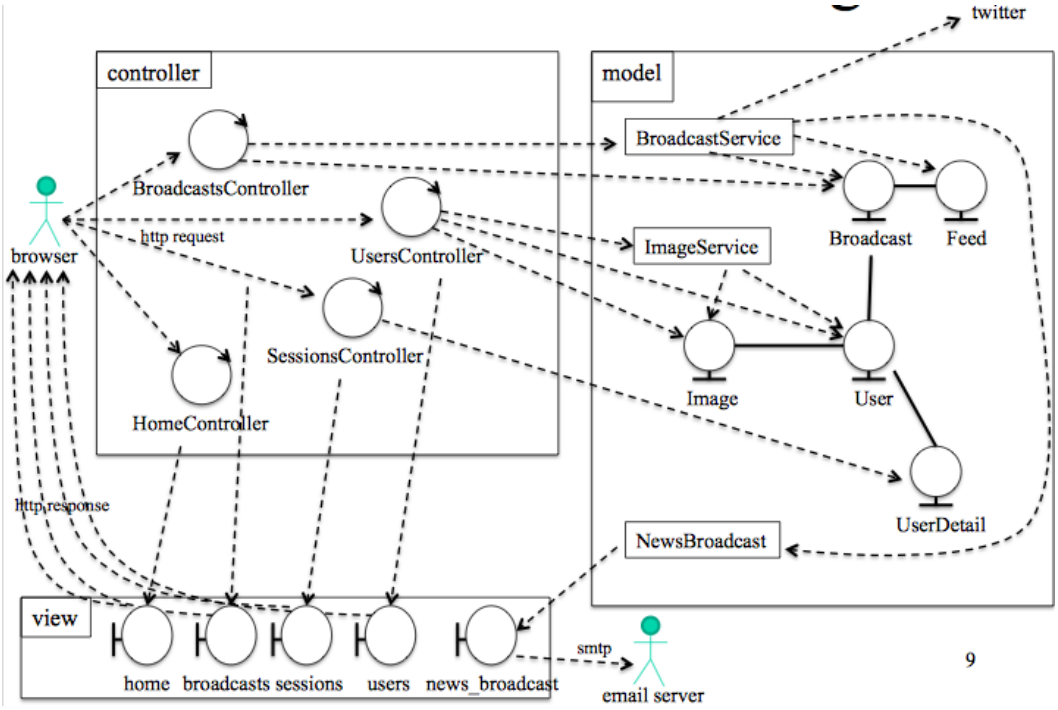


Figure 4: Design : Class diagram. This class diagram was produced by Loftus[1, 5]

.

When a client/browser sends a request to the CSA application, the request gets sent to the controller.

- In the case of the Home page, then HomeController gets run and the home view gets sent as a HTTP response to the client.
- When a client requests to be logged in (or is logged in) their request gets sent to the SessionsController. The userDetails in the session parameters are compared to those in the UserDetail model. If the login details are correct then the sessions view is given as a HTTP response to the client/browser.
- When a users page is requested the UsersController accesses the Image and User model to get the data for the user(s), the users view is then sent to the browser as a HTTP response.
- When a broadcast is posted to the BroadcastsController a record in the broadcast model is added. BroadcastServices is used to handle broadcasting the broadcast to the different feeds. A Feed record is created for each type of feed the broadcast is being sent to.
  - When an email is being created the email content is entered into the news_broadcast and SMTP is used to send the email to the users.

4 Test Strategy

For the addition I made to the application I decided to add cucumber tests using capybara and selenium. This way I could see what parts of the application are yet to be implemented, and what improves could be made. Having a full set of tests before changing an application is important, as you do not want to break any of the current system.

Section	Action	Test content	Input	Output	Criteria / Pass?
Users	Admin login	Check that the admin user can login.	Login: admin password: taliesin	“Logged in successfully” should be displayed.	Message displayed to user, and home page displayed.
		Admin can edit other users	Visit user 40’s edit page. Grad year: 2001 Click update button.	User 40’s page shows 2001 for the grad year.	User has be correctly updated.
	Edit user (S9)	Check that invalid information can not be entered into the edit user form.	firsname: empty	“Surname can't be blank” should be displayed.	Message displayed to the user and redirects to edit user form.
			surname: empty	“Surname can't be blank” should be displayed.	As above
			email: aaa@aber	"Email Bad email address format”	As above
			Grad Year: 1969	“Grad year must be greater than or equal to 1970"	As above
			Grad Year: 2015	"Grad year must be less than or equal to 2014”	As above
		Valid information entered.	Grad Year: 2013	User 41 page should be displayed.	User 41 page shown with updated information.
	User - update image	Check that the user’s image can be updated.	Upload file : /images/dragonFly.jpg	User page displayed, showing the new image.	The new user’s image is displayed.
	New user	Check that valid information can be entered into the new user form.	Firstname : Fred Surname: Blogs Phone: 07576947302 Grad Year: 2014 Email: fred@outlook.com login: fred password: password password: password	“Account was successfully created” should be displayed.	Message is displayed to user.
		(valid input for range check for the grad year)	As above but with Grad Year: 1970	As above	As above
			As above but with Grad Year: 2000	As above	As above
		Check that invalid information can not be entered into the new user form.	Empty firstname.	“Firstname can't be blank” should be displayed.	Message displayed to the user and redirects to new user form.
			Empty surname.	“Surname can't be blank” should be displayed.	As above
			Empty Grad Year	“Grad year can't be blank”	As above
			Grad Year: abcd	“Grad year is not a number"	As above
		(invalid input for range check for the grad year)	Grad Year: 1969	“Grad year must be greater than or equal to 1970"	As above
			Grad Year: 2015	"Grad year must be less than or equal to 2014”	As above
			Empty email	“Email can't be blank”	As above
			Email: invalid	"Email Bad email address format”	As above
			Email: cwl1@aber.ac.uk	"Email has already been taken“	As above
			Empty login	"User detail login can't be blank”	As above
			cwl1	"User detail login has already been taken”	As above
			Empty password	"User detail login has already been taken”	As above
			Password: password Confirm password: different	“Confirm password should match password"	This feature is NOT implemented. Test will be added once implemented.
	GET user	Check that user gets redirected when trying to visit a none existing account.	Visit user 70 page.	“Account no longer exists"	Message is displayed and user is redirected to users page.

Section	Action	Test content	Input	Output	Criteria / Pass?
	Delete User	Check that a user can be deleted.	Click delete on user 1.	User should no longer exist.	User that has been deleted does not get shown in the list of users.
		Check you can not delete the user that is logged in.	Delete the admin user, who is logged in.	Should show error message “Can not delete current user”	This is not implemented. You can delete the current user and then “RecordNotFound” error is displayed.
	Search Users	Check that the search drop down list works correctly	Search: Lo Click on Chris Loftus in the drop down list.	User details are displayed.	Can click a user on the drop down list and the user is shown.
		Check that user can search using the different search options.	Checked : surname Search: Surname	All users but Loftus are shown.	Only users with a surname containing Surname are displayed.
			Check: first name Search: Firstname	All users but Chris are shown.	Only users with a first name containing Firstname are displayed.
		Check that no users are displayed when the database does not contain any users matching the search.	Search: None	“No entries found” is displayed.	The correct message is displayed to the user.
	Normal user accessibility.	Check that the user does not have added to other users and the broadcasts page.	cwl2 is logged in	Profile tab available, Broadcast tab not available	The correct tabs are displayed to the user.
		Check that if the user tries to visit a page they are unauthorised to access they are redirected to the home page.	cwl2 is logged in. Visit users page.	Redirected to home page.	User gets redirected to the home page.
			cwl2 is logged in. Visit user 40 page.	Redirected to home page.	User gets redirected to the home page.
			cwl2 is logged in. Visit user 40 edit page.	Redirected to home page.	User gets redirected to the home page.
	Login	Check that an invalid login does not get accepted.	login: empty password: taliesin	"Couldn't log you in as” should be displayed.	The message is displayed to the user.
			login: admin password: empty	"Couldn't log you in as” should be displayed.	As above
			login: hello password: taliesin	"Couldn't log you in as” should be displayed.	As above
			login: admin password: invalid	"Couldn't log you in as” should be displayed.	As above
	Not Logged In	Check that when is logged in they can not accessed anything but the login and the home page.	Visit user 40 page.	Redirected to the login page	The login page is displayed to the user.
	Log out	Check that a user can log out.	Login as admin. Logout.	Profile tab should not be available.	The user is logged out.
Broadcasts	Create Broadcast (S10, S6)	Check that a new broadcast can be created and added to twitter	Check: Twitter Broadcast: cucumber broadcast (plus the current time)	The broadcast is successfully created. The broadcast is also posted on twitter.	Broadcast shown on to user. Tweet has been made.
		Check that a broadcast can be made to multiple feeds.	Check: Twitter and Email. Broadcast: cucumber 1st of multiple broadcasts	The broadcast is successfully created. It has been broadcast to Twitter and Email.	The Broadcast and “Email, Twitter” is displayed to the user.
		Check that multiple broadcasts can be made.	Broadcast: cucumber 2nd of multiple broadcasts	Both the broadcast should be successfully created.	Two broadcasts are shown on the broadcasts page.
		Check that a broadcast of over 150 characters can not be entered when the twitter tick box is checked.	Check: Twitter Broadcast: "this post will be longer than a twitter feed post is allowed to be. It has been written for cucumber testing of CSA app. Should not be posted"	"Broadcast was successfully saved, but problems broadcasting to one or more feeds” should be displayed.	Message given to user and the broadcast is not made. Broadcasts page should be shown.

Section	Action	Test content	Input	Output	Criteria / Pass?
	Delete Broadcast (S11)	Check that a broadcast can be deleted.	Click delete on the first broadcast.	The broadcast is removed from the table.	The broadcast is no longer displayed in the table.
	View Broadcast	Check that the user can view a broadcast, and the back button returns to the broadcasts page.	Check the view broadcast button.	Broadcast is shown.	“cucumber broadcast” is displayed to the user
			Click the back link.	List of broadcasts is shown.	Have returned to the broadcasts page.

4.1 A few of the issues found

- While logged in as an admin, you can delete the admin user.
- The "Shorten URL" on the broadcasts page does not do anything.
- The confirm password field does not have any validation. It can be different from the password.

4.2 Script To Execute Tests

I have created a python script that executes the cucumber tests; saves the results to a database (testResults.db) and plots a graph to show the test results over time. This has been used to show the progress made while creating tests and the amount of code these tests are covering. Showing progress often helps to keep developers more focused.

The sequence diagram in figure 5 shows how this script works.

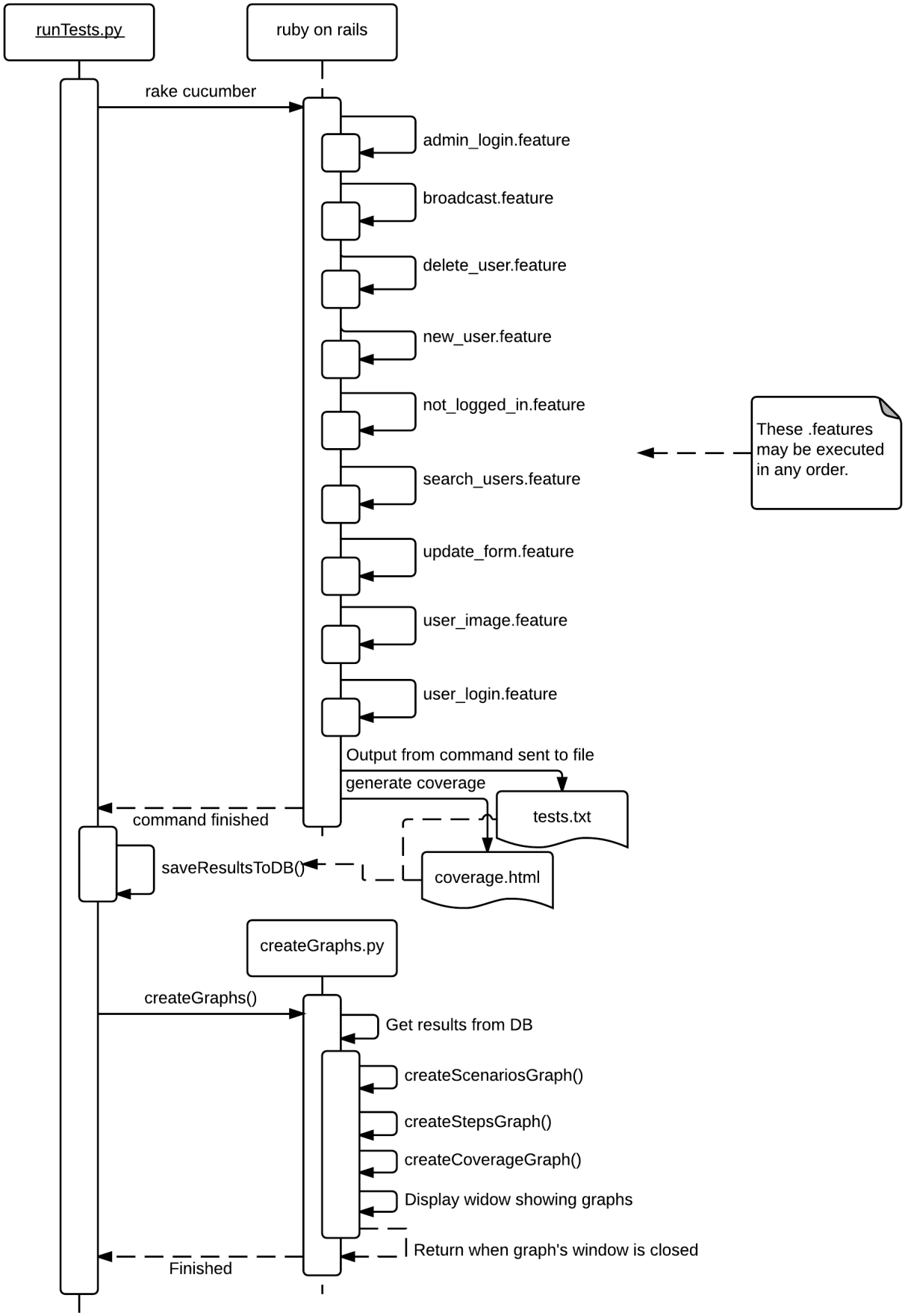


Figure 5: Test Strategy : Sequence Diagram.

4.3 Helper Files

I also created the following support files:

- env.rb
  - Runs the seed.rb file to fill the test database with the same data as the development database. The database gets cleaned between each of the scenarios.
  - Sets up the fillers and groups for the simpleCov to be run.
- paths.rb
  - Gets the path to the given page. Where page is the string from the feature. This is used to visit a page and to check that the current page is correct.
- search_helper.rb
  - Used to fill in the search filled, and select an item from the drop down menu.
- helper.rb
  - Any addition functions that are needed. Currently just contains function to get the current time.

4.4 Test Results

Figure 6, 7 and 8 show the output from running the cucumber tests. As you can see from the figures 31 scenarios with 385 steps have been run. These have managed to cover 87.5% off the CSA application. It would be hard to increase the coverage any further, with the current implementation.

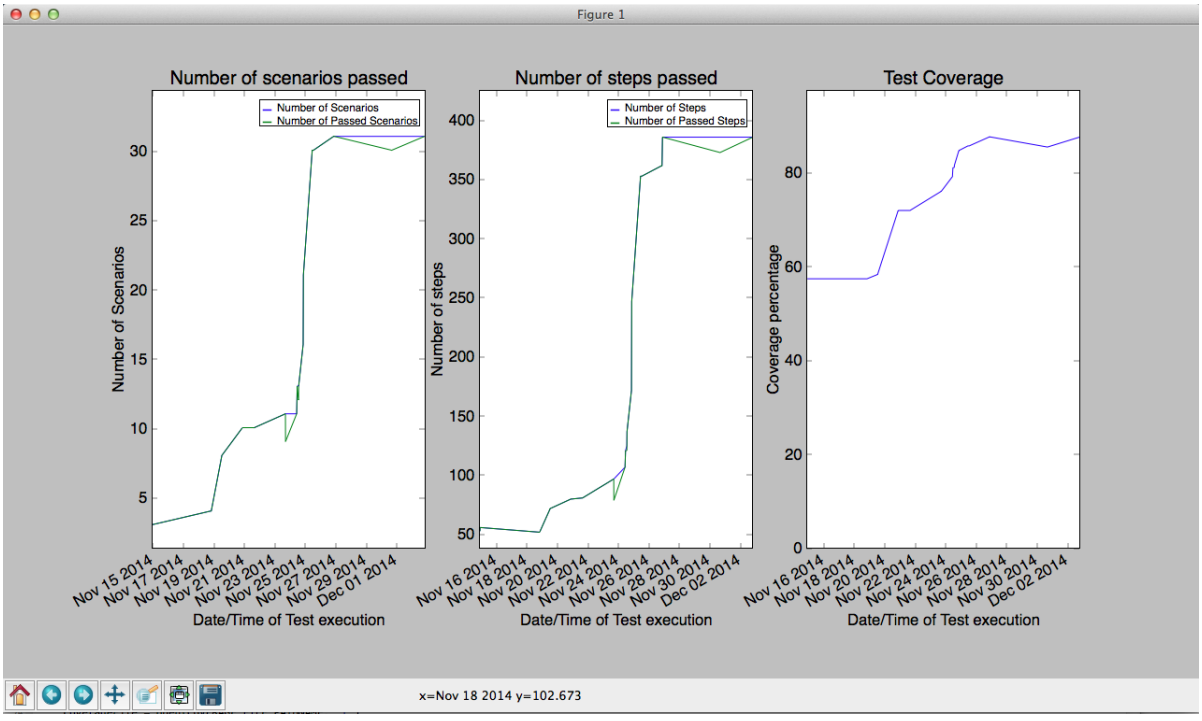


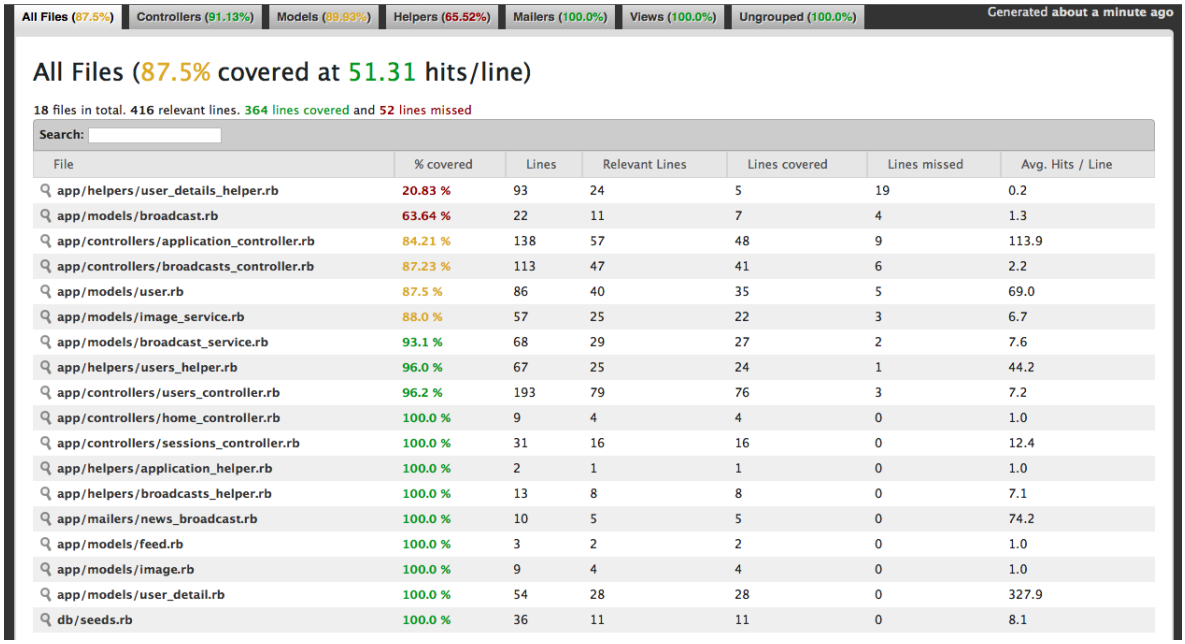
Figure 6: Test Results : Output from running the python test runner.

```
31 scenarios (31 passed)

385 steps (385 passed)
```

Figure 7: Test Results : The scenarios and steps that have been run.

The output shown in figure 8 is produced by the simpleCov gem. My tests manage to cover 87.5% of the code in the CSA application.



All Files (87.5%) | Controllers (91.13%) | Models (89.83%) | Helpers (65.52%) | Mailers (100.0%) | Views (100.0%) | Ungrouped (100.0%) | Generated about a minute ago

All Files (87.5% covered at 51.31 hits/line)

18 files in total. 416 relevant lines. 364 lines covered and 52 lines missed

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/helpers/user_details_helper.rb	20.83 %	93	24	5	19	0.2
app/models/broadcast.rb	63.64 %	22	11	7	4	1.3
app/controllers/application_controller.rb	84.21 %	138	57	48	9	113.9
app/controllers/broadcasts_controller.rb	87.23 %	113	47	41	6	2.2
app/models/user.rb	87.5 %	86	40	35	5	69.0
app/models/image_service.rb	88.0 %	57	25	22	3	6.7
app/models/broadcast_service.rb	93.1 %	68	29	27	2	7.6
app/helpers/users_helper.rb	96.0 %	67	25	24	1	44.2
app/controllers/users_controller.rb	96.2 %	193	79	76	3	7.2
app/controllers/home_controller.rb	100.0 %	9	4	4	0	1.0
app/controllers/sessions_controller.rb	100.0 %	31	16	16	0	12.4
app/helpers/application_helper.rb	100.0 %	2	1	1	0	1.0
app/helpers/broadcasts_helper.rb	100.0 %	13	8	8	0	7.1
app/mailers/news_broadcast.rb	100.0 %	10	5	5	0	74.2
app/models/feed.rb	100.0 %	3	2	2	0	1.0
app/models/image.rb	100.0 %	9	4	4	0	1.0
app/models/user_detail.rb	100.0 %	54	28	28	0	327.9
db/seeds.rb	100.0 %	36	11	11	0	8.1

Figure 8: Test Results : The output produced by the coverage gem.

5 Critical Evaluation

Before the start of this module I had no experience with using ruby, and did not have much experience with different web technologies. I took me awhile to learn these new technologies and to find my way around the CSA application. I also had to learn about cucumber testing, and found it strange to work in a behaviour driven way. Once I had gotten use to writing features, developing the tests became a lot quicker. Learning the technologies meant that I spent a lot longer on the assignment than the recommended 50 hours.

I have had some past experience with python but had not used the matpotlib before so wanted to gain experience of using this module. Creating the graphs gave me this experience. The Python scripts could then easily be started from a continuous integration sever like Jenkins/Hudson.

5.1 Technologies Used

- Ruby 2.1.3 - Programming Language
- Rails 4.1.5 - Framework
- SQLite3 - Database (including the test result database)
- Cucumber - Testing gem using description of the behaviour of the application

- Capybara-webkit - used to automate the cucumber tests.
- Selenium-webdriver
- JSON - used to check that a tweet has been created.
- Javascript and AJAX - Used when performing the search for a user.
- SimpleCov - Produces the HTML coverage, for running the cucumber tests.
- Python - Create the graphs for the test results. matpotlib has been used.
- SQLite3 - Used in setting up the test results database and when using the test database.

5.2 Mark

I would give the following marks to each section:

- Successfully running the provided CSA prototype : 4%
  - I have included a screencast showing the working CSA application.
- Design and documentation (Cucumber test project): 10%
  - Design lacks some details, but range of diagrams used to show how the applications works.
- Implementation: 25%
  - OK identifiers used, code is commented. Only one function in some helper files, so these could be combined.
- Flair: 9%
  - Included Python program that creates graphs of the results.
- Testing (Cucumber test project): 22%
  - Decent coverage percentage, test table included and test plan included. More details about the choice of tests could be included.
- Evaluation: 4%
  - Lots of details.

  With a total mark of 74%

5.3 Problems Encounter

1. I had lots of issues getting the CSA application running. Eventually I found the places that the issues with the force SSL where happening, and commenting them out got the application running.
2. When first running the cucumber tests I did not realise that it was running from the test database rather than development database. I spent a long time changing pieces of code until I realised that a admin couldn't login because they didn't exist in the database.
3. Software, Framework and GEM Versions.
   While Googleing for solutions for many of the issues I had, I received many results that did not work for the versions of the software I am using.
   - I had a lot of issues with dealing with the pop-up received when a broadcast or user is deleted. I tried a lot of methods before finding one that works.
   - I had similar issues when trying to test the search for a user as AJAX had been used to perform the search.
4. After installing ImageMagick uploading a image to the user still did not work. Later I found that this was because I did not have the path set to the correct place.

5. Within the tests after posting the Tweet I then check that twitter has been updated with the tweet. I tried many different way to get this working. This included getting the page the Tweet is on, but this would be different for every tweet I create. In the end I just use the Twitter token to get the last tweet that the account created. This means that if another Tweet happens between the Tweet it creates and when I check it, the test will fail.

6. I also had issues with created the same tweet multiple times. Tweet does not allow a user to make the same tweet multiple times within a certain time frame. To solve this I added the time to the tweet. If the time (minutes/hours) changes between the creation of the tweet and the checking of the tweet, then the test will fail. Out of all the times I have run the tests this has only happened once.

All of these problems are simple to solve once you know how, and most are fixed with just one or two lines of code.

6 Attributions

Additional Gems and rails related attributions:

1. Cucumber
   - The instructions I used for using cucumber can be found at http://cukes.info/
   - To learn how to use cucumber I used the following tutorial: http://loudcoding.com/posts/quick-tutorial-starting-with-cucumber-and-capybara-bdd-on-rails-project/. Using a paths.rb support file was suggested by this tutorial.
2. I used Capybara API (http://www.rubydoc.info/github/jnicklas/capybara/Capybara/) while writing the steps.
3. Learnt the ruby regular expressions from http://www.agileforall.com/2010/07/just-enough-regular-expressions-for-cucumber/
4. I found the simpleCov gem at https://rubygems.org/gems/simplecov
5. I learnt how to upload files with cucumber using https://cassiomarques.wordpress.com/2009/01/23/how-to-test-file-uploads-with-cucumber/

Other attributions:

1. All figures I have created have been formatted using http://lucidchart.com/
2. When creating the python code I used http://matplotlib.org to learn how to use the matplotlib library.

References

[1] Chris Loftus (2014), Requirements/Design for the CS-Alumni Application, [Online], Available: https://blackboard.aber.ac.uk/bbcswebdav/pid-487373-dt-content-rid-767088_1/xid-767088_1 [30 November 2014].