

Coursework 2 - Trello Dash Board

Adrian Martinez Moreno
40174677@live.napier.ac.uk

Edinburgh Napier University - Advanced Web Technologies (SET09103)

1 Introduction

Trello is one of the fastest growing agile development team project and management tool used by over 4.5 million users around the world [1]. As the number of project and team increases, it can become increasingly difficult from a team members point of view, to navigate a series of list and cards that are assigned to the member in a shared trello board. It is also difficult to track all actions that the user have executed on various projects and task.

This project designs a dashboard for existing Trello user to check task that are assigned to that user as well as a log of all the actions that the user have executed till date.

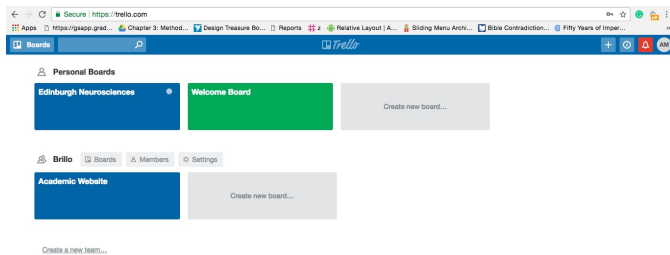


Figure 1: **Trello Website** - Dashboard after login

2 Design

The web application directory structure can be seen below:

```
app.py app_test.py etc static templates var
40174677@soc-web-liv-12:~/test/Martinez_Adrian_set09103_coursework2/sourcecodes$
```

Figure 2: **Application Directory** - File Structural Arrangement

- **app.py**: This contains server side code that logs application errors as well as serve html pages amongst other things.
- **app_test.py**: This contains test cases written to ascertain that the web page works as expected.

- An **etc folder**: This folder is used for website parameter configuration, the configuration is specified in a file called **defaults.cfg**.
- A **var folder**: This folder contains the **trello.log** files which contains application error logs mainly for 404 errors.
- A **templates folder**: This folder contains the html documents that are used in the web application
- A **static folder**: This folder contains static website artefacts like cascading style sheet, javascript files, fonts etc.

2.1 Web Page Functionality

A Trello API based single page dashboard was created with the following functionality.

- Login through Trello account based authentication and authorization
- Login out through Trello account authorization removal
- Choosing of Trello Boards
- Checking of actions carried out by an authenticated user on a Trello Board
- Checking of cards assigned to an authenticated Trello user on a particular Trello board
- Downloading of assigned cards and actions for an authenticated Trello user in csv format

2.2 Dash Board Page Design

The Trello based API dashboard is a single page website designed with a single button that at the top left to connect to the Trello authentication and authorization pop up or page. Once a user is authenticated two selects buttons appear. The first select is for choosing which board you will like to use, the second select specifies if you want to view your actions on the selected board or your assigned cards. Most of the functionality of this page is driven by client side jQuery code contained in the "trello.js" file in the static folder as well as the "download.js" folder also in the static folder.

The web page layout was done via the Bootstrap CSS framework and JQuery, while the back-end code was written in Python using the Flask framework. The Python code predominantly loads the "home.html" page using the code snippet below:

Listing 1: Home Page Python Code

```
1 @app.route('/')
2 def home():
```

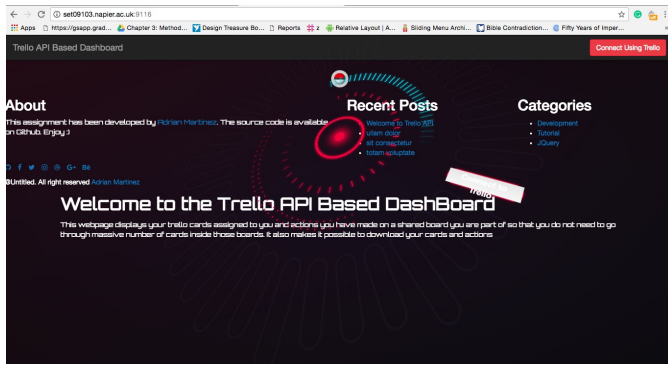


Figure 3: Home Page without authentication - Before authenticated

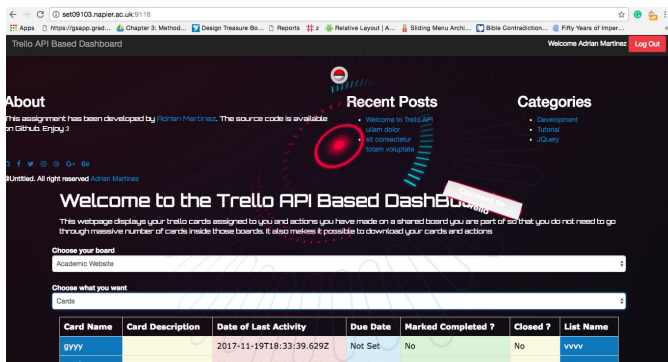


Figure 4: Home Page with authentication - After authenticated

```
3         return render_template('home.html')
4
```

The animation is based on the future and robotics. The website theme is Retro-Futuristic and the genre is Sci-Fi. The font, Orbitron, helps to create an impression of an inevitable future. The inspiration for this striking theme has come from the module "Digital Storytelling" directed by Dr. Flint. One of the web based interactive games exhibited in one of the lectures possess the same laser effect and bionic eye. The range of colours used is quite suitable for users with visual impairments and has been tested with the Color Contrast Checker tool developed by WebAIM[2]. The result is as follows:

As you may see in the figure, the contrast ratio is nearly optimal and significantly close to the maximum, which is 21:1.

2.3 Login and Logout Feature

Login in and out of the website was done via Trello authentication and authorization. When a user wants to login, he needs to first authenticate himself and then authorize that the application can use the person's details as seen below

When the user clicks the "Connect using Trello" button, a click handler is called inside "trello.js", a call is made to the Trello authorize method to bring up the Trello page for authentication and authorization. However, before this could work, there is a need to import the trello "client.js"

Color Contrast Checker

[Home](#) > [Resources](#) > Color Contrast Checker

Foreground Color

 Lightness

Background Color

 Lightness

Contrast Ratio
19.65:1
[permalink](#)

Normal Text

WCAG AA: Pass

WCAG AAA: Pass

The five boxing wizards jump quickly.

Large Text

WCAG AA: Pass

WCAG AAA: Pass

The five boxing wizards jump quickly.

Explanation

Enter a foreground and background color in RGB hexadecimal format (e.g., #FD3 or #F7DA39) or choose a color using the color picker. The Lightness slider can be used to adjust the selected color.

Figure 5: Colour Contrast - Result of the contrast between the background and the font colours

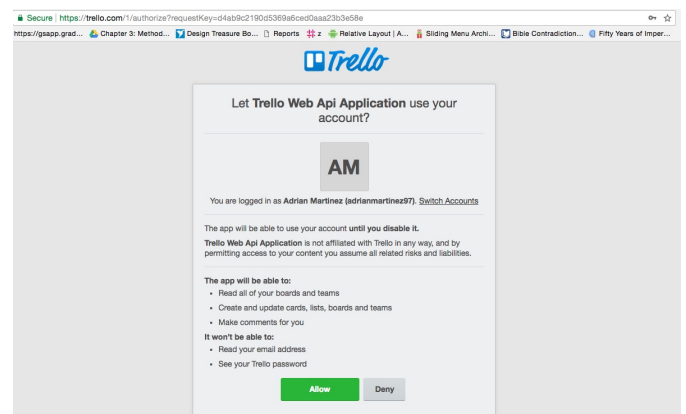


Figure 6: Trello Authorization - After authenticated, you need to authorize the application

file with an application developer key, this was placed in the **header.html** file that is inherited by the home page.

Listing 2: Import Trello's client.js

```
1 <script src="http://code.jquery.com/jquery-3.2.1.min.js"></script>
2 <script src="https://api.trello.com/1/client.js?key=eaf8c57e2e802c6e08c08132d4433d4e"></script>
3
```

If the authentication is successful, the function **authenticationSuccess** is called which in turn makes a another call to the Trello API to get the member's fullname and append a welcome to it and display it on the home page.

Listing 3: Authentication and Authorization using Trello

```
1 #//login in
2 $( "#loggedin" ).click(function(){
3     Trello.authorize({
4         type: 'popup',
5         name: 'Trello Web Api Application',
6         scope: {
7             read: 'true',
8             write: 'true' },
9         expiration: 'never',
10        success: authenticationSuccess,
11        error: authenticationFailure
12    });
13
```

```

12     });
13     });
14
15     var authenticationFailure = function() {
16         console.log("Failed authentication");
17     };
18
19
20     var authenticationSuccess = function() {
21         updateLoggedIn();
22     };
23
24     var token = Trello.token();
25     Trello.members.get("me", function(member){
26         var welcometxt="Welcome "+member.fullName+" ";
27         $("#welcome").text(welcometxt);
28     });
29     //load boards
30     loadBoards();
31 };
32
33

```

It also calls the **updateLoggedIn** which switch between the button that displays a "Connect using Trello" and the other which displays a welcome message with a "Logout" button as seen below.

Listing 4: updateLoggedIn Function

```

1     var updateLoggedIn = function() {
2         var isLoggedIn = Trello.authorized();
3
4         $("#loggedin").toggle(!isLoggedIn);
5         $("#loggedout").toggle(isLoggedIn);
6     };
7

```

Finally a call is made to load all the boards in the system. On the other hand, logging out of the system involves just removing authorization from the application. When the user clicks the "Logout" button, a click handler is called which makes a call to the Trello API to deauthorize the user as seen below.

Listing 5: Removing Authorization

```

1     //login out
2     $("#loggedout").click(function(){
3         Trello.deauthorize();
4         updateLoggedIn();
5         window.location.replace("/");
6     });
7
8

```

2.4 Loading of Trello Board

When a user is authenticated, the next thing that is done is to make a call to retrieve all the boards that the authenticated user belongs to, this is done via a call to the function **loadBoards** as seen below, a call is made to the Trello API to get the boards an authenticated member belongs to as seen below. If a failure occurs, a call is made to **boardLoadFailure** which prints to the console that there is a failure in loading the board.

Listing 6: Loading Trello Board

```

1     var loadBoards = function() {
2         //Get the users boards
3         Trello.get(
4             '/members/me/boards/',
5             loadedSelect, boardLoadFailure
6         );
7

```

```

7     };
8
9
10    var boardLoadFailure = function() {
11        console.log("Failed to load boards");
12    };
13
14    var loadedSelect= function(boards) {
15        .....
16        .....
17    };
18

```

If the board was loaded successfully, a call is made to the method **loadedSelect** which will be discussed in the next section. It is also possible to navigate between boards to see the content of the other boards. This will in turn lead to the loading of that member's action for that board or the cards the member is assigned. The code snippet is a change listener that does this.

Listing 7: Change Board Function

```

1     $('#boards').change(function() {
2         //call to load card
3         if($("#select#actiontype option:checked").val()=="cards")<->
4             {
5                 loadCards();
6             }
7         //load actions
8         loadActions();
9     });
10
11

```

2.5 Loading of Assigned Cards on a Trello Board

Once a user is authenticated, the program by default loads the card of the first board the authenticated user belongs to.

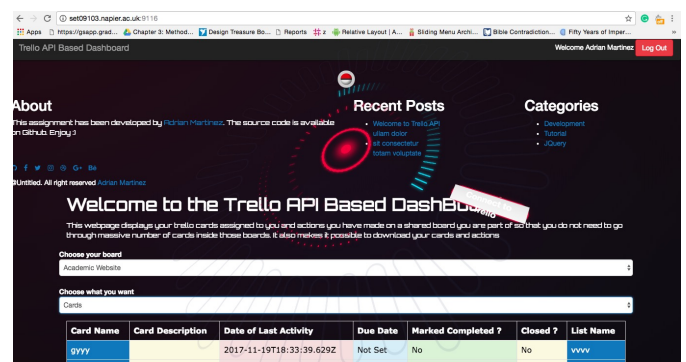


Figure 7: **Cards Loaded** - Loading of card after authenticated

This is because upon login in, the **loadedSelect** function is called which takes the result from the API and append it to an HTML select tag, it also appends the action type (Actions, Cards) to another HTML select tag before toggling that they should be visible. It then finally calls the **loadCards** or **loadActions** depending on what the actiontype is chosen (this is by default "card")

Listing 8: loadedSelect Function

```

1  var loadedSelect= function(boards) {
2
3  $.each(boards, function(index, value) {
4      $('#boards')
5          .append($"<option></option>")
6          .attr("value",value.id)
7          .text(value.name));
8
9
10 });
11
12 $('#boardsgroup').toggle(true);
13 //load actions type to the board
14
15 $('#actiontype')
16 .append($"<option></option>")
17 .attr("value", "cards")
18 .text("Cards")
19 .append($"<option></option>")
20 .attr("value", "actions")
21 .text("Actions");
22
23 $('#actiontypegroup').toggle(true);
24
25 //call to load card
26 if($("#select#actiontype option:checked").val()=="cards")↔
27 {
28     loadCards();
29 }else{
30     //load actions
31     loadActions();
32 }
33 };
34
35

```

Card information loaded include:

- Card name
- Card description
- Date of last activity
- Card due date
- Check if card is marked complete
- Check if card is closed
- The list the card belongs to

Lastly, it is important to note that the tables are generated dynamically in **loadCards** function.

2.6 Loading of User Actions on a Trello Board

Loading of user actions is also easy, the user needs to select that it wants to view "actions" for a board and not "card". Once the selection is done, a change listener is called on this select tag as seen below:

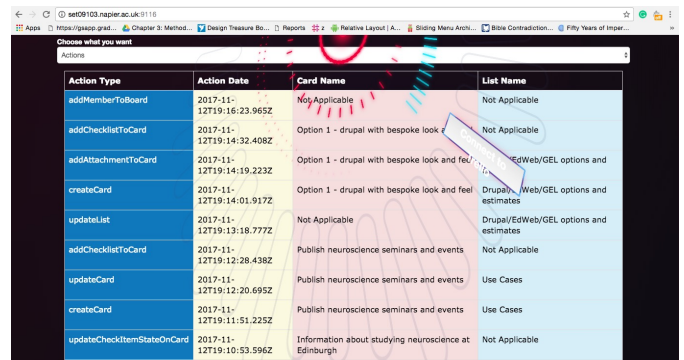
Listing 9: Change Action Function

```

1  $('#actiontype').change(function() {
2      //call to load card
3      if($("#select#actiontype option:checked").val()=="↔
4      cards"){
5          loadCards();
6      }else{
7          //load actions
8          loadActions();
9      }
10 });

```

The load action function loads every action on the board that are performed by the user using the Trello API. The page displays information as seen in the figure below.



Action Type	Action Date	Card Name	List Name
addMemberToBoard	2017-11-12T19:16:23.965Z	Not Applicable	Not Applicable
addChecklistToCard	2017-11-12T19:14:32.408Z	Option 1 - drupal with bespoke look and feel	Not Applicable
addAttachmentToCard	2017-11-12T19:14:19.223Z	Option 1 - drupal with bespoke look and feel	Drupal/EdWeb/GEL options and estimates
createCard	2017-11-12T19:14:01.917Z	Option 1 - drupal with bespoke look and feel	Drupal/EdWeb/GEL options and estimates
updateList	2017-11-12T19:13:18.777Z	Not Applicable	Drupal/EdWeb/GEL options and estimates
addChecklistToCard	2017-11-12T19:12:28.438Z	Publish neuroscience seminars and events	Not Applicable
updateCard	2017-11-12T19:12:20.695Z	Publish neuroscience seminars and events	Use Cases
createCard	2017-11-12T19:11:51.225Z	Publish neuroscience seminars and events	Use Cases
updateChecklistStateOnCard	2017-11-12T19:10:53.596Z	Information about studying neuroscience at Edinburgh	Not Applicable

Figure 8: **Actions Loaded** - Loading of board actions by a user

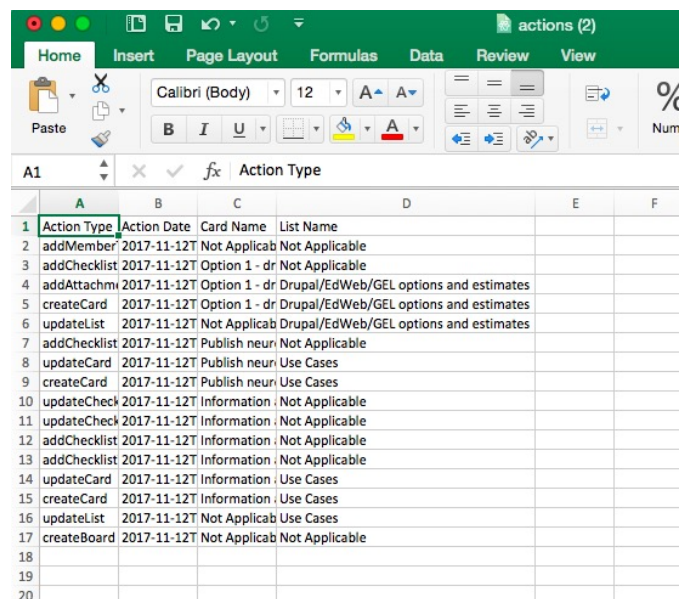
Actions information loaded includes:

- Action type
- Action Date
- Card name if applicable
- List name if applicable

Lastly, it is important to note that the tables are generated dynamically in **loadActions** function.

2.7 Downloading of User Actions and Cards on a Trello Board

At the bottom of both tables that display either a list of card details or a list of actions, an "Export To CSV File" button which downloads the content to a CSV file. A sample CSV file can be seen below:



Action Type	Action Date	Card Name	List Name
addMember	2017-11-12T	Not Applicable	Not Applicable
addChecklist	2017-11-12T	Option 1 - dr	Not Applicable
addAttachment	2017-11-12T	Option 1 - dr	Drupal/EdWeb/GEL options and estimates
createCard	2017-11-12T	Option 1 - dr	Drupal/EdWeb/GEL options and estimates
updateList	2017-11-12T	Not Applicable	Drupal/EdWeb/GEL options and estimates
addChecklist	2017-11-12T	Publish neuro	Not Applicable
updateCard	2017-11-12T	Publish neuro	Use Cases
createCard	2017-11-12T	Publish neuro	Use Cases
updateCheck	2017-11-12T	Information	Not Applicable
updateCheck	2017-11-12T	Information	Not Applicable
addChecklist	2017-11-12T	Information	Not Applicable
addChecklist	2017-11-12T	Information	Not Applicable
updateCard	2017-11-12T	Information	Use Cases
createCard	2017-11-12T	Information	Use Cases
updateList	2017-11-12T	Not Applicable	Use Cases
createBoard	2017-11-12T	Not Applicable	Not Applicable

Figure 9: **Downloaded CSV File** - CSV Actions by a user

The download file code is stored in the **download.js** of the static folder and called by the "Export To CSV File" button

are displayed in the order that they occurred, it will still be nice to filter by any column of the table.

Listing 10: Change Action Function

```
1 $("<button class='btn btn-warning' onclick=\"↵
    exportTableToCSV('actions.csv')\">Export To CSV File</↵
    button>").appendTo($('#cardsoractions'));
2
```

2.8 Configuration Files and Logging

The configuration files specifies application parameters in the file **defaults.cfg** inside "etc" folder while the trello.log file located "var" folder keeps a record of page not found errors. The content of the configuration file can be seen below:

Listing 11: Configuration File

```
1 [config]
2 debug = True
3 ip_address = 0.0.0.0
4 port = 9116
5 url = http://set09103.napier.ac.uk:9116
6 [logging]
7 name=trello.log
8 location=var/
9 level=DEBUG
10
```

2.9 Application Unit Testing

The application also contains a simple unit test to ensure that the home page is loaded with a response of 200 while also ensuring that the correct HTML content is present. The content of the file can be seen below:

Listing 12: Unit Testing File

```

1 import unittest
2 import app
3 class AppTest(unittest.TestCase):
4     def test_root(self):
5         self.app = app.app.test_client()
6         out = self.app.get('/')
7         assert '200 OK' in out.status # check
8         assert 'charset=utf-8' in out.content_type
9         assert 'text/html' in out.content_type
10        assert b'Trello API Based Dashboard' in out.data #↔
        check that page loads correctly
11
12    if __name__ == "__main__":
13        unittest.main()
14
```

3 Enhancement

The following enhancement will be useful in making the application better:

- **Further Filtering and Searching:** The application would become more flexible if it was possible to filter and search for information via various filter boxes as well as search boxes.
- **Sorting of Table:** This feature is missing from the current implementation, even though the information

- **Pagination of Table:** When a large number of information is to be displayed, a paginated table will be useful for navigating as against displaying all the information from top to bottom.
- **Text File Download Name Change:** When files are downloaded, they carry the name "action.csv" for user Actions on a board or "cards.csv" for user Cards for a board. It will be nice to append the board name to the card name to differentiate between various downloads from different boards.
- **User Assistance:** This enhancement includes the addition of instructions, a FAQ section or even an optional tutorial that would start automatically to guide the user through the web app. This is due, primarily, to the bifurcation of the user's path by having two buttons with the same functionality, one in the navigation menu and another as part of the futuristic-theme animation.

4 Critical Evaluation

The web-app works properly and provides a In terms of functionality, I have achieved what I set out to do. However, the main animation may be disliked by some users The reason why most of the code written in CSS was not deleted is because of my mindset based on that each website whose purpose is neither commercial nor real could be considered as an expression space. Each should have a very defined theme to bring to light the creativity and expertise of graphic elements, transitions in CSS and control of their timing and speed.

5 Personal Evaluation

I got to know the various concepts and models in Trello. I've also carried out the process of obtaining an Application key, which is the key that identifies my application to Trello. There was a copious amount of difficulties with the correct information being displayed in the tables, but with a deeper research, I discovered eventually the solution. The methods utilised to overcome challenges have been browser search, investigation in particular websites, for instance, the community of Trello developers (and their Get Started Under 5 Minutes guide for Trello API) and jsfiddle and, as a last resort, the creation of inquiries on Stack Overflow. In conclusion, I consider this project possesses an adequate level given the limited period of time available.

References

- [1] Trello, "Trello website," <https://trello.com/about>, 2017.
- [2] W. team, "Webaim color contrast checker," <https://webaim.org/resources/contrastchecker/>, 2017.