

COSC2675 Rapid Application Development

Computer Science and Information Technology
School of Science, RMIT

Assignment 2 – Semester 1 2018

Introduction

This assignment is worth **35%** towards your final grade. You are required to develop a web application as specified below. It is designed to:

- Practise your knowledge of Rails development
- Practise testing and debugging

This is a group assignment. A group should not have more than 2 members. Individual submission does not receive any extra marks. By default, both group members receive the same mark unless there is a significant discrepancy in their contributions.

Academic Integrity

The submitted assignment must be your own work. No marks will be awarded for any parts, which are not created by you. More on <http://www.rmit.edu.au/academicintegrity>.

Plagiarism is treated very seriously at RMIT. Plagiarism includes copying code directly from other students, internet or other resources without proper reference. Sometimes, students study and work on assignments together and submit similar files which may be regarded as plagiarism. Please note that you should always create your own assignment even if you have very similar ideas.

Plagiarism-detection tools may be used for all submissions. Penalties will be applied in cases of plagiarism.

Client Request

This assignment is developed based on the following fictitious message from the school admin team.

Dear Andy

Our school is thinking of building a news website to let everyone share relevant news. RAD students may be able to help as we've heard from many that they know how to quickly build a good website from scratch.

There is no formal specification yet. We just want a news-sharing site, which is easy to use and has good look-n-feel, for our staff and students. People with a proper RMIT email address should be able to register to be a user. Do you think we should only allow registered users to post, update and remove news? Viewing the news should not be a problem. Mmm... it would be nice that the site looks like Hacker News. Students and staff can comment on the news and see latest comments from fellow users.

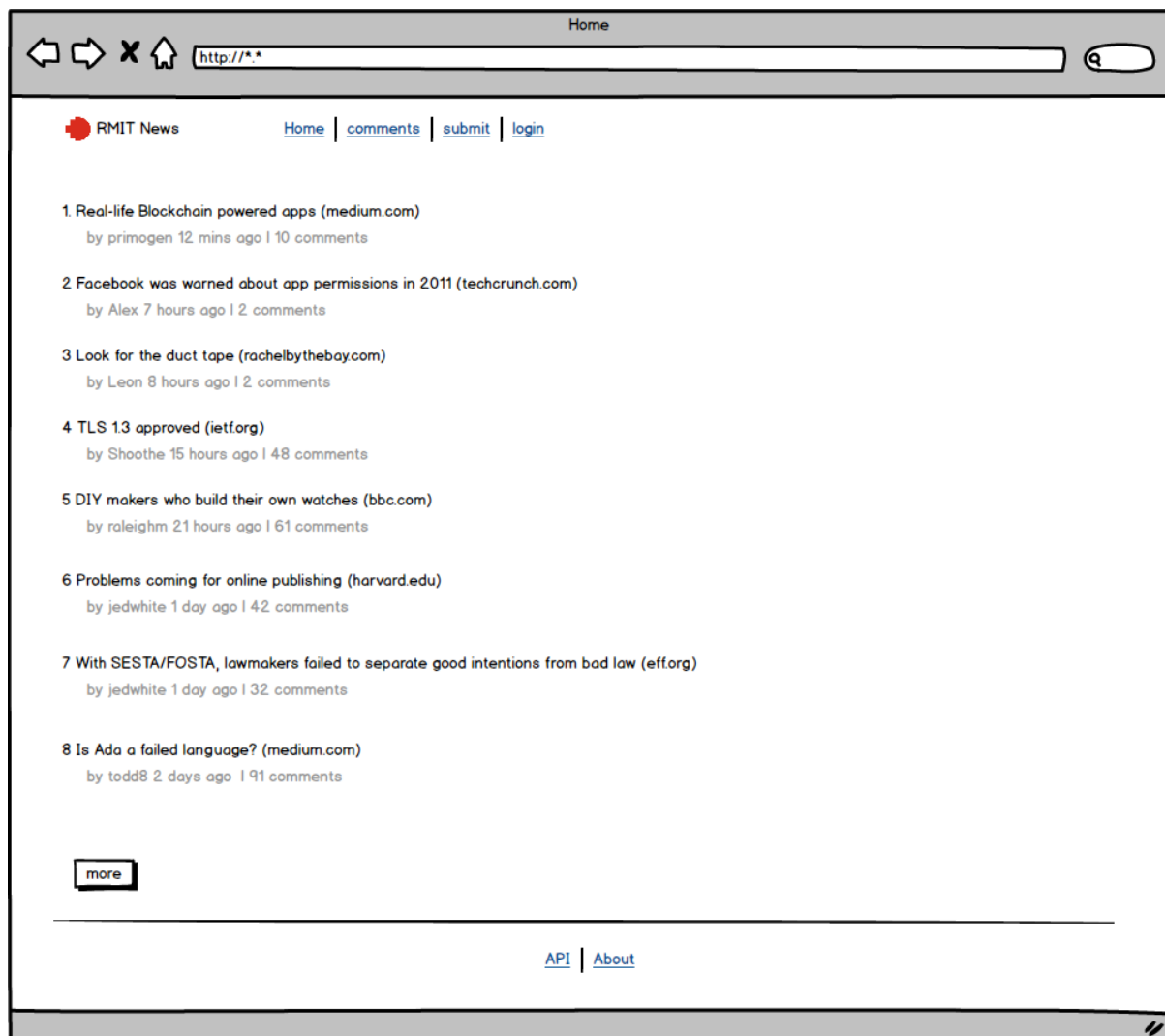
Thank you and talk soon.

RMIT News

In this assignment, you will build a news sharing web application for RMIT. That is kind of a subset of famous [hacker news](https://news.ycombinator.com/) (<https://news.ycombinator.com/>). For whom is new to Hacker News, it is a social news website focusing on computer science and entrepreneurship. It is run by Paul Graham's investment fund and startup incubator, Y Combinator. In general, news that can be submitted is defined as "anything that gratifies one's intellectual curiosity".

Part 1

Section 1 - Home Page



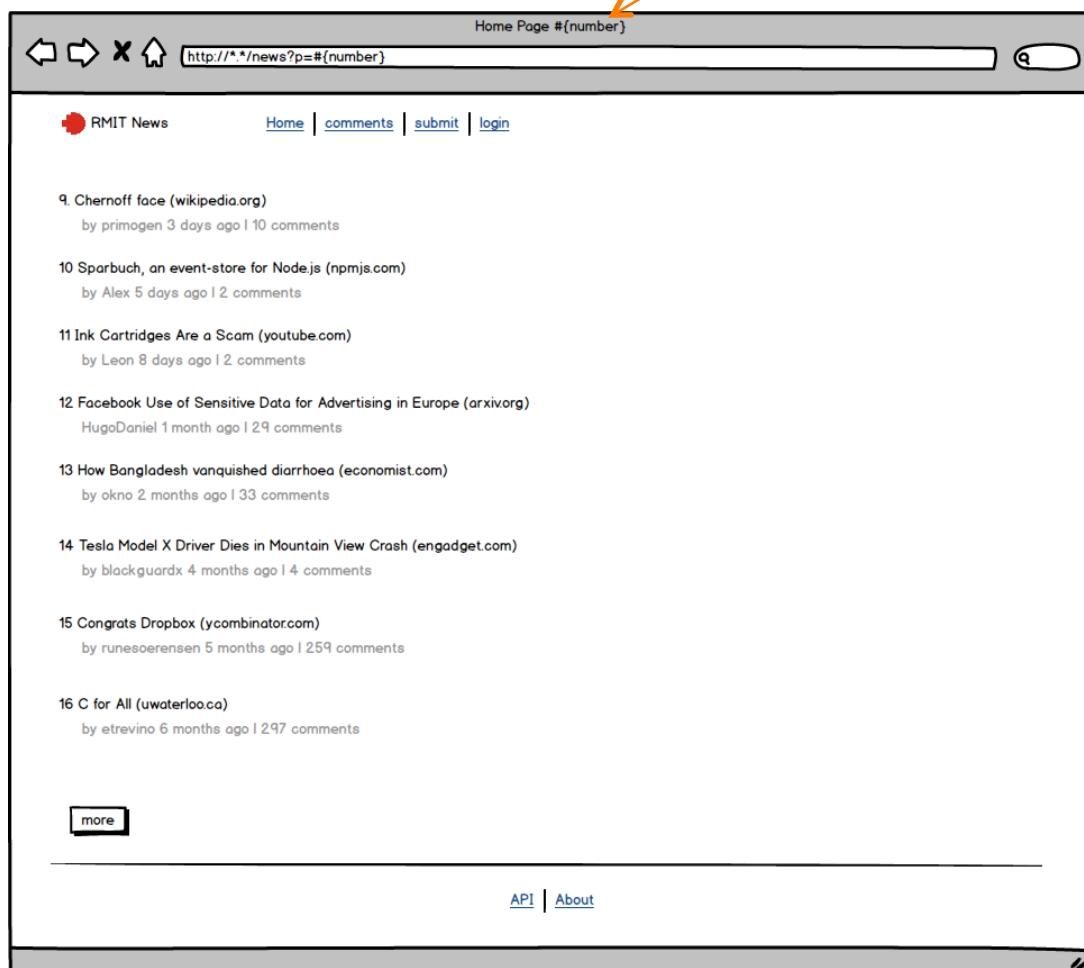
Your application should meet the following requirements

1. It has a navigation bar with RMIT logo and site name on it
2. The navigation bar must contains at least four hyperlinks
 - a. Home – to show the home page
 - b. Comments – to show the latest comments (of all news)
 - c. Submit – to submit a news (login required)
 - d. Login – login or create user account
3. The news content

- a. The news posts are sorted by the creation date using Unix timestamps.
 - b. The headline of a news contains the title and the source of the news (if exists)
 - i. By clicking on the source, the user will be redirected to the source website.
 - ii. The source news text is the domain name of that particular news website.
 - c. The meta information about each news contains:
 - i. The author who submitted the news
 - ii. The time when the news got submitted
 - iii. The number of total comments on that news
 - iv. The text e.g. "25 comments", is clickable and will take the user to the comments page of that news. (see below)
4. The footer
 - a. API: a link to the [API documentation](https://github.com/pigfly/RMIT-CSIT-RAD/tree/master/2018_S1/assignment2) (https://github.com/pigfly/RMIT-CSIT-RAD/tree/master/2018_S1/assignment2)
 - b. About: the makers of your web application
 - i. The names of your group members
 - ii. The student number of each member.
 - iii. Your heroku deployed URL
 - iv. Your assignment 2 bitbucket URL
5. Pagination
 - a. The news home page will show eight news per page
 - b. The pagination is triggered by pressing the more button

Section 2 - Home Page Pagination

2.3



Requirements:

1. When the more button is clicked, the page will show the next eight news
2. The page's relative URL is `/news?p=#{number}` e.g. `/news?p=2`
3. The page title should be changed into

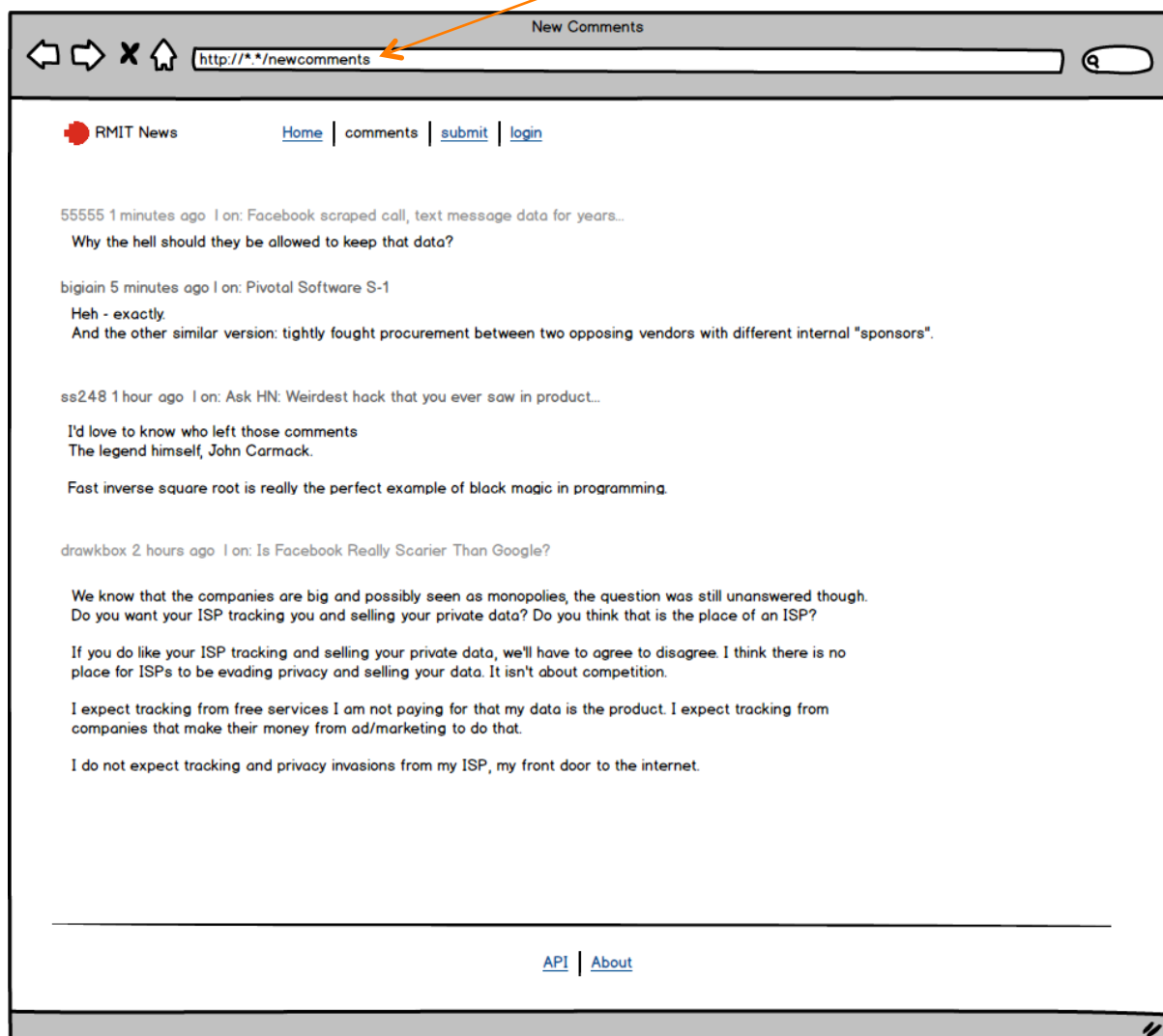
`Home page #{your_current_pagination_number}`

The page number for the above example is 1.

The page number is zero indexed, meaning that number for the Home page is 0.

Section 3 - New Comments

3.2

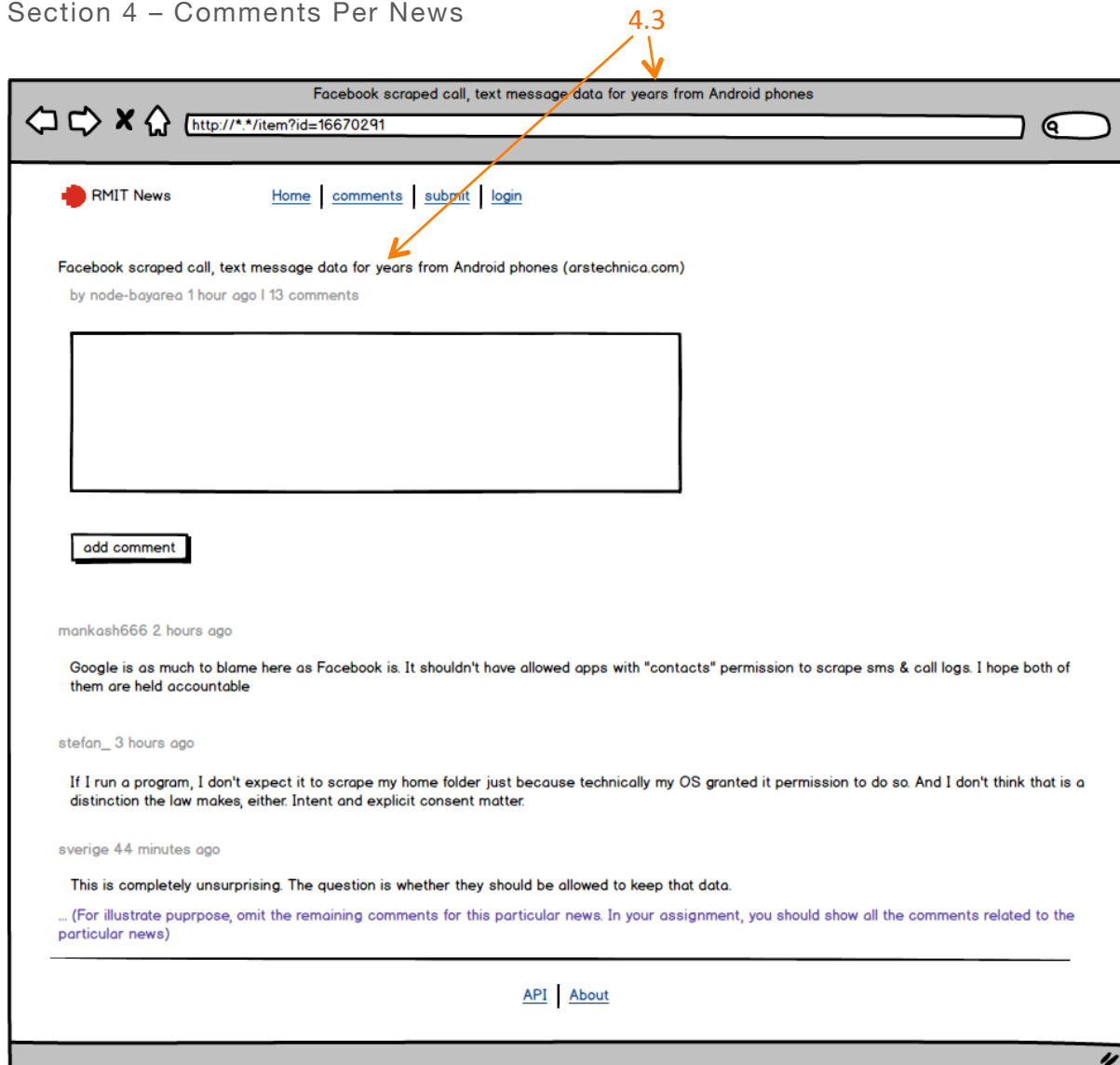


Requirements:

1. The New Comments page shows the latest comments of all news posts.
2. The New Comments page's relative URL path is `/newcomments`
3. The New Comments page is sorted by the creation date & time

4. Each comment has two parts:
 - a. **Headline:** include the user who commented on that particular news, the date that the comment was created, and the title of the news.
 - b. **Body:** the comment message body
5. The New comments page will show and only show three comments sorted by the creation date.

Section 4 – Comments Per News



Requirements:

1. The Comments page of each news shows a text area for commenting and all the comments related to that particular news. Comments are sorted by date and time.
2. The Comment Per News page's relative URL path is `/item?id=#{id_number}`
3. The page title is the same title of the news, which is to be commented on.
4. The comment section has four parts:
 - a. **Headline:** the news title

- b. Sub header: the creator of the news, the news creation date/time and the total number of comments at present
 - c. A text area for the commenting
 - d. A submit button - “add comment”
- 5. Validation rules for comment messages
 - a. Must not be blank
 - b. Must be at least 3 characters **excluding** white spaces, new lines and tabs
 - c. Must be less than 1000 characters
- 6. Banner messages:
 - a. If the comment **doesn't** pass the validation, show a warning banner message (see banner message page in Section 6).
 - b. If comment **does** pass the validation, show a success banner message (see banner message page in Section 6) and redirect the user to </newcomments>
 - c. You can decide the text for both types of banner messages yourself.
- 7. The list of all comments
 - a. It should show all the comments that belongs to this particular news
 - b. All the comments that belongs to this particular news should be sorted by creation date/time
 - c. It should show no comments if there is no comment for that news

Section 5 - Submit

The screenshot shows a web browser window titled "Submit a news". The address bar contains "http://*.*/submit". The page header includes the RMIT News logo and navigation links: Home, comments, submit, and login. The main content area features a large text input field labeled "(required)", a smaller input field labeled "news source (optional)", and a "submit" button. The footer contains links for API and About.

Submit a news

http://*.*/submit

RMIT News Home | comments | submit | login

(required)

news source (optional):

submit

API | About

Requirements:

1. The Submit Page allows the user to submit a news
2. The Submit Page's relative URL path is </submit>
3. If the user is not logged in, redirect the user to the login page (</login>)
4. The navigation title of this page should be "Submit a news"
5. The submit section has three parts
 - a. The text area for the submitted news title (required)
 - b. The source of the news (optional)
 - c. The submit button
6. Validation rules for the submit text area:
 - a. Must not be blank
 - b. Must be at least 10 characters (**including** white spaces, new lines and tabs)
 - c. Must be less than 200 characters

Note: The text area is for entering the news title only. There is no news body.

7. Validation rule for the source of the news
 - a. Must be a valid URL if the URL is not empty
8. Banner messages
 - a. If a submission **doesn't** pass the validation, show a warning banner message (see banner message page in Section 6).
 - b. If comment **does** pass the validation, show a success banner message (see banner message page in Section 6) and redirect the user to **the home page**.

You can decide the text for both types of banner message yourself.

Section 5 - Login

The screenshot shows a web browser window titled 'Login'. The address bar contains 'http://*.*/login'. The page header includes 'RMIT News' and navigation links: [Home](#), [comments](#), [submit](#), and [login](#). The main content area features two forms. The first form, titled 'Login', has a 'username:' label, a text input field, a 'password:' label, a text input field, and a 'login' button. The second form, titled 'Create Account', has a 'username:' label, a text input field, a 'password:' label, a text input field, and a 'create' button. The footer contains links for [API](#) and [About](#).

Requirements:

1. The login page shows the login and creates an account
2. The login page's relative URL is [/login](#)
3. The validation rule for user name
 - a. Must not be blank
 - b. Can only contain letters, digits, dashes and underscores.
 - c. Should be between 2 and 15 characters.
 - d. Must be a unique username in the database.
4. The validation rule for password
 - a. Must not be blank
 - b. Must contain at least one uppercase letter, one special character, one number and one lowercase letter
 - c. Must contain at least 10 characters
5. Both username and password must be valid in order to pass the validation
6. Banner messages
 - a. If a submission **doesn't** pass the validation, show a warning banner message (see banner message page in Section 6).
 - b. If comment **does** pass the validation, show a success banner message (see banner message page in Section 6) and redirect the user to **the home page**.

You can decide the text for both types of banner message yourself.

Section 6 - Banner Message

The image is a hand-drawn wireframe of a web browser window. The browser's address bar shows 'http://*/*/login'. The page title is 'Login'. A red banner message at the top states: 'Usernames can only contain letters, digits, dashes and underscores, and should be between 2 and 15 characters long. Please choose another.' Below this, the 'RMIT News' logo is on the left, and navigation links 'Home | comments | submit | login' are on the right. The main content area has two sections: 'Login' and 'Create Account'. Each section contains a 'username:' label, a text input field, a 'password:' label, another text input field, and a button ('login' for the first section, 'create' for the second). At the bottom, there are links for 'API' and 'About'.

Browser window showing a login page for RMIT News.

Page title: Login

Address bar: http://*/*/login

Message: Usernames can only contain letters, digits, dashes and underscores, and should be between 2 and 15 characters long. Please choose another.

Navigation: RMIT News | Home | comments | submit | login

Form sections:

- Login**
 - username:
 - password:
 - login
- Create Account**
 - username:
 - password:
 - create

Footer: API | About

Guidelines

- Your web **application** should be mobile friendly
- You can use your favourite CSS/HTML style as you wish

Part 2 API

In this part, you will implement your API endpoint for RMIT news.

Overview

This is sample documentation for RMIT News API (RAD course)

Please post your relevant thoughts in canvas if found anything unclear.

URI and Versioning

The first version will have URIs prefixed with http://*.*/v0/ and is structured as described below.

Design

The v0 API is essentially a dump of your database.

It's not the ideal public API, but it's the one you can get a taste how RESTful API works internally.

Items

News, comments are just items. They're identified by their ids, which are unique integers, and live under `/v0/item/<id>`.

All items have some of the following properties, with required properties in bold:

Field	Description
id	The item's unique id.
type	The type of item. One of "news", "comment"
by	The username of the item's author.
time	Creation date of the item, in Unix Time .
text	The comment text

Field	Description
url	The source URL of the news
title	The title of the news

For example, a news: <http://yourownurl/v0/item/8863.json>

```
{
  "by" : "dhouston",
  "id" : 8863,
  "time" : 1175714200,
  "title" : "My YC app: Dropbox - Throw away your USB drive",
  "type" : "news",
  "url" : "http://www.getdropbox.com/u/2/screencast.html"
}
```

comment: <http://yourownurl/v0/item/2921983.json>

```
{
  "by" : "norvig",
  "id" : 2921983,
  "text" : "Aw shucks, guys ... you make me blush with your compliments.<p>Tell you what, Ill make a deal: I'll keep writing if you keep reading. K?",
  "time" : 1314211127,
  "type" : "comment"
}
```

another news: <http://yourownurl/v0/item/121003.json>

```
{
  "by" : "tel",
  "id" : 121003,
  "text" : "<i>or</i> HN: the Next Iteration<p>I get the impression that with Arc being released a lot of people who never had time for HN before are suddenly dropping in more often. (PG: what are the numbers on this? I'm envisioning a spike.)<p>Not to say that isn't great, but I'm wary of Diggification. Between links comparing programming to sex and a flurry of gratuitous, ostentatious adjectives in the headlines it's a bit concerning.<p>80% of the stuff that makes the front page is still pretty awesome, but what's in place to keep the signal/noise ratio high? Does the HN model still work as the community scales? What's in store for (++) HN?",
  "time" : 1203647620,
  "title" : "The Arc Effect",
  "type" : "news",
  "url" : ""
}
```

Endpoint to implement:

- [/v0/item/<id>](#)
 - o Request Type: GET
 - o Authentication: None
 - o Return status code: 200 for OK, 400 for Errors

- [/v0/item/create](#)
 - Request Type: POST
 - Request Body:

```
{  
  "username" : "alex",  
  "text" : "Create a sample news",  
  "type" : "news",  
  "new source" : "https://www.rmit.edu.au"  
}
```

- Authentication: None
 - Return status code: 200 for OK, 400 for Errors
-

Additional Notes

- 1: Your application is expected to be production ready. Proper test and data validation should be embedded in your application.
- 2: There is no need for your application to look and behave exactly like the above example. You can make necessary adjustment to improve the design. However the aforementioned functionalities are the minimum requirement and should be well supported.
- 3: Only the following GEMS should be used in your application.

```
gem 'rails'  
gem 'bootstrap-sass'  
gem 'puma'  
gem 'sass-rails'  
gem 'uglifier'  
gem 'coffee-rails'  
gem 'jquery-rails'  
gem 'turbolinks'  
gem 'jbuilder'  
gem "font-awesome-rails"  
gem 'bcrypt'
```

Submission

- This assignment will also be using bitbucket to host all your Rails files

<https://bitbucket.org/>

- o Create a **private repo** named as
`RAD18_Assign_2_group_{studentNumber_small}_{studentNumber_large}`

ONLY ONE submission for each group is needed. For duplicate group submissions, we will randomly select one to mark.

- o Make a branch called “**submission**”
- o Invite the following people as default reviewers to your project
 - junliang.jiang@rmit.edu.au
 - anto.dominic@rmit.edu.au
 - andy.song@rmit.edu.au
- o Make a pull request to merge your branch **submission** to branch **master** and include the following default reviewers (to ensure our access)
 - junliang.jiang@rmit.edu.au
 - anto.dominic@rmit.edu.au
 - andy.song@rmit.edu.au
- o **The due date: 11:59pm of the 20th of May.**
- o **A final submission before 15th May 11:59pm will receive 10% extra as the bonus.**
- o **This assignment will attract additional marks for individual lab demonstration.**
- o **For one professional developer working alone, the expected completion time is less than 5 full days.**
 - It is a good habit to keep track of the hours that you spent on a project. For that purpose, we recommend toggl.com, a free online time tracker.
- **Warning**
 - o You will receive ZERO mark if you don't host your assignment in bitbucket before the due day (see details later).
 - o You will receive ZERO mark if your application fails to run (see details later).