CS5003 — MASTERS PROGRAMMING PROJECTS

CS5003 PRACTICAL 1: STACSYAK

Deadline: Friday the 28th February 2020 at 21:00

Credits: 30% of the coursework (and the module)

MMS is the definitive source for deadline and credit details

You are expected to have read and understood all the information in this specification and any accompanying documents at least a week before the deadline. You must contact the lecturer regarding any queries well in advance of the deadline.

AIMS

The main aim of this project is to teach you to write clearly structured single-page web applications. You will practice using APIs to access remote services and incorporate the results into your application, and maipulating arrays.

DETAILS

In this assignment, you are going to write a client for a new messaging service. It's called StacsYak and it lets the users post messages, called Yaks, to a server and read the Yaks other people post. The StacsYak system is based around a server-side Web API that can be used to post Yaks, read all the current Yaks, and delete Yaks. It can also be used to up vote and down vote Yaks, and get information about the users of the system. Each user needs a user id allowing them to access the StacsYak API. All CS5003 students have been emailed their id already. If you have not recieved yours please contact ruth.letham@st-andrews.ac.uk

Your application should consist of client-side JavaScript, using the appropriate API calls to provide suitable questions. Your submission should be written in modern JavaScript (a.k.a. ECMAScript version 6+) with appropriate HTML, CSS and any libraries you find useful. It should not be written in any other language or JavaScript dialect (e.g. CoffeeScript, HAML, Pug, etc.).

STACSYAK API

StacsYak has a number of API endpoints, all of which are documented in the online documentation. Each endpoint returns HTTP status codes that should be dealt with appropriately, for example, 403 errors might be logged to the console or displayed to the user in some way.

The API root can be found at https://cs5003-api.host.cs.st-andrews.ac.uk/api, and the API documentation can be found at https://cs5003-api.host.cs.st-andrews.ac.uk/. You can use the API documentation to test out the API, but you must input your user id using the *authorise* button in the top right for this to work.

There are several API endpoint that can be used, such as up voting and down voting, but the following two above are essential to meet the *basic* requirements of this assignment:

- GET /yaks – this returns a list of all the Yaks posted in the last 2 days. The Yaks are returned as a JSON array containing objects. Each object has the following fields:
    - id – a unique id for the Yak.
    - content – the text that was submitted for the Yak.
    - userNick – the nickname of the user that posted the Yak (this is optional and will only be there if the nickname has been set by the user).
    - votes – the total number of votes that the Yak has.
    - userVote – tell you if your user (the one used to authorise) has voted on the Yak and how they voted.
    - timestamp – a time stamp telling you when the Yak was posted.
- POST /yaks – this allows you to post a single new Yak up to the server. If successful, along with the status code 200, the server will return a JSON object containing the key "status" with the value "ok". The request to post a yak must have:
    - Content-Type header set to "application/json" because we are posting JSON to the server.
    - Body containing a JSON object with a single property called "content" containing the text you want to post.

AUTHORISATION

StacsYak can only be used by users who have been issued a user id. Because of this, every request that you make to the API must provide a URL query parameter, called key, with the StacsYak user id that you were sent via email. For example, if we were calling the GET /yaks endpoint and our user id was "test123", the full URL would look like this:

https://cs5003-api.host.cs.st-andrews.ac.uk/api/yaks?key=test123

NOTE ON CONDUCT

As we are essentially building a small social network, be aware that Yaks that you post will be visible to the entire class (and anyone on the internet with a valid user id). As a result of this, please make sure that the contents of your Yaks abide by the principles of good student conduct outlined here: https://www.st-andrews.ac.uk/students/rules/conductdiscipline/conduct/.

If you feel uncomfortable with anything that is posted, please feel free to get in touch with a member of staff such as the course Lecturer, the Director of Post Graduate Teaching, or the Director of Teaching.

REQUIREMENTS

Your application should provide the functionality described below. You must make sure you have completed all the requirements in each section before moving onto the next.

BASIC

Your application should provide the following:

- Allow users to see all the Yaks that are currently available from the server, including content, id and votes for each.
- Allow users to post Yaks. After successfully posting a Yak, re-display the updated list of Yaks from the server *without* reloading the page.

- Allow users to delete Yaks they have created. After successfully deleting a Yak, re-display the updated list of Yaks from the server *without* reloading the page.
- Allow users vote Yaks up and down. After successfully up/down voting a Yak, re-display the updated list of Yaks from the server *without* reloading the page.
- Allow users change their nickname. After successfully changing the nickname, display it in the interface *without* reloading the page.

INTERMEDIATE

Your application should provide all the requirements described in Section Basic, plus the following:

- Allow users to filter the current Yaks based on nick name.
- Allow users to filter the current Yaks based on hashtags in the content.
- Allow users to sort the current Yaks based on the number of votes.
- Allow users to sort the current Yaks based on the timestamp.

ADVANCED

Your application should provide all the requirements described in Sections Basic and Intermediate, plus the following:

- Allow users to have multiple `chatrooms' displayed on the page. Each should display messages filtered based on the poster's nickname or on hashtags in the content. e.g. one `chatroom' filter to show only messages from `bob' another to show only messages including the hashtag `#CSRules'. The user should be able to sort each chatroom independently.
- Poll the server every 5 seconds to get and display the updated list of Yaks from the server *without* reloading the page.
- Display a barchart showing the average number of Yaks posted per hour over the last 2 days. You may achieve this using CSS and DOM manipulation or by researching and using a JavaScript library that provides this functionality. If you use a library, be sure to clearly state in your report what library you used, how you found it, why you chose it, and how you used it.

DELIVERABLES

A single .zip file must be submitted electronically via MMS by the deadline. It should contain:

- The source code for your application, including your CSS, HTML and Javascript.
- A short report (approximately 1000 words), in PDF format detailing the design of your solution, discussing any requirements and design decisions taken, and reflecting on the success of your application. Try to focus on the reasons for your decisions rather than just providing a description of what you did.

Submissions in any other formats may be rejected.

MARKING CRITERIA

Marking will follow the guidelines given in the school student handbook: https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback. html#Mark_Descriptor Your submission will not be evaluated based on aesthetic appeal (this is not a visual design course), but use of CSS and DOM scripting which enhances the experience and interactivity will be rewarded. Some specific descriptors for this assignment are given below:

- A **poor implementation in the 0–7 mark band** will be missing nearly all required functionality. It may contain code attempting a significant part of a solution, but with little success, together with a report describing the problems and the attempts made at a solution.
- A **reasonable implementation in the 8–10 mark band** should provide some of the functionality described in Section Basic, and demonstrate reasonable use of HTML and JavaScript. The code should be documented well enough to allow the marker to understand the logic. The report should describe what was done but might lack detail or clarity.
- A **competent implementation in the 11–13 mark band** should provide all the functionality described in Section Basic, demonstrate competent use of HTML and CSS (e.g. for layout and positioning), allowing a user to view, post, delete and vote on Yaks, and set their nickname. The code should be documented well enough to allow the marker to understand the logic and should be contained in JavaScript files that are separate from the HTML code. The report should describe clearly what was done, with good style.
- A **good implementation in the 14–16 mark band** should provide all the functionality described in Section Basic and some or all of the functionality from Section Intermediate. It should demonstrate good code quality, good comments, and proper error handling. Good use of CSS for layout and styling is expected for a submission in this range (not unmodified defaults). The report should describe clearly what was done with some justification for decision, with good style, showing a good level of understanding.
- An **excellent implementation in the 17 and higher mark band** should provide all the functionality described in Sections Basic and Intermediate, and some or all of the functionality from Section Advanced. It should demonstrate high-quality code and be accompanied by a clear and well written report showing real insight into the subject matter.

Note: For this practical you do not need to invent your own extensions. Concentrate on providing high quality, sophisticated implementations of the requirements in this specification and writing an insightful report demonstrating understanding.

WORD LIMIT

An advisory word limit of approximately 1000 words, excluding references and appendices, applies to this assignment. No automatic penalties will be applied based on report length but your mark may still be affected if the report is short and lacking in detail or long and lacking focus or clarity of expression. A word count must be provided at the start of the report.

LATENESS PENALTY

The standard penalty for late submission applies (Scheme B: 1 mark per 8 hour period, or part thereof): https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#lateness-penalties

GOOD ACADEMIC PRACTICE

The University policy on Good Academic Practice applies: https://www.st-andrews.ac.uk/students/rules/academicpractice/

Author: Dr Ruth Letham *Module*: CS5003 – Practical 1: StacsYak