UNIVERSITEIT VAN PRETORIA
Department of Information Science

IMY 220
Advanced Markup Languages II
Examination

Examiner: 19 – 20 October 2022
Internal: Mr ID Bosman Marks: 36
Time*:* 24 hours
External: Mr YL Wong

This test will run from 08:00 on 19 October to 08:00 on 20 October. Download the test files from ClickUP, which contain the folder structure you must adhere to when submitting your files. You must submit a zip file with the contents of the S1 and S2 folders on ClickUP (**excluding the node_modules folder**). You also need to add your position and surname as indicated in the various files. Do not submit this specification.

Section 1 [20]
JavaScript, ES6, jQuery, AJAX

Use only the files provided in the "S1" folder to complete this section.

This section requires you to write functionality in JavaScript. You must, wherever appropriate, make use of the following ES6 syntax:

- Arrow functions
- Appropriate variable declarations
- Template strings
- Object destructuring

All functions must be written as function expressions. Furthermore, you are not allowed to use any loops, *forEach* function, or *$.each* function to implement array-based functionality. You must make use of the other JS array functions as discussed in class. Array-based functionality must work for any number of array items.

Complete the following questions in <u>script.js</u>.

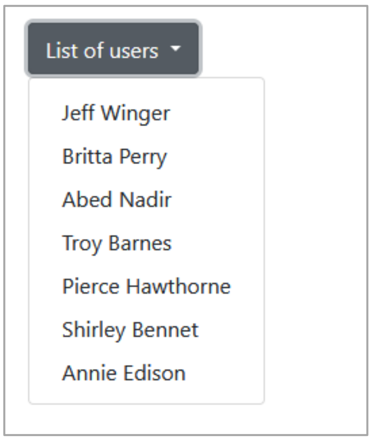
1.1 Write a function called *getUserDetails* which receives two parameters: the name of a file and an array of user IDs. The function will be used to read a JSON file and return an array of user details from this file (you may assume that the format will always be similar to that found in *users.json*). If an array of user IDs is supplied in the function (4) parameters, it must only return the details for the users whose IDs match those given. If no array of user IDs are supplied to the function, it must simply return the details for all users listed in the file name given. You must use a default parameter to accomplish this.

1.2 Write a function called *getFriendList* which receives two parameters: the name of a file and a single user ID. The function will be used to read a JSON file and return an array containing the list of user IDs that correspond with that of the given user (you may assume that the format will always be similar to that found in *friends.json*). (4)

1.3 Write a function called *createUserRow* which receives a JS object containing the name and surname of a user and returns a table row with those details. You must use the syntax recommended in the notes for creating jQuery elements. (3)

1.4 Using these three functions, create following functionality: (3)

a) When the page loads, populate the dropdown list with the names of all the users in *users.json*. You do not need to call an additional function for this. This must look as follows (when clicked on):



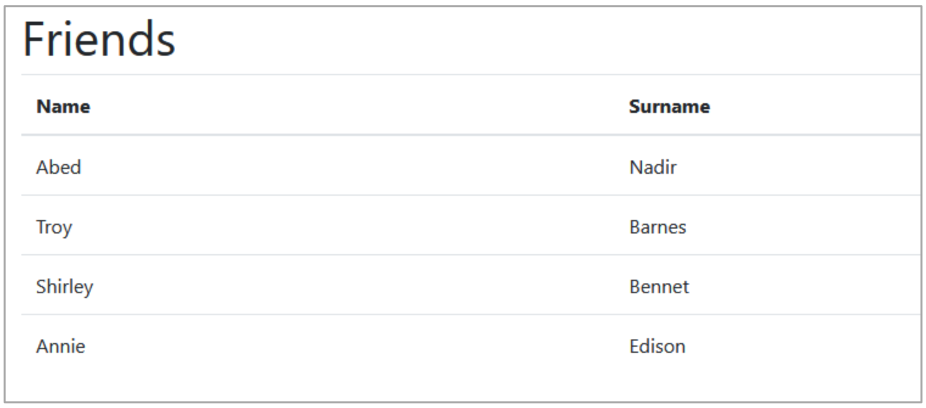
b) You must also use a data-attribute to save the userid of each user listed in the dropdown list.

1.5 When clicking on any of the dropdown list items, their list of friends (as found in *friends.json*) must be displayed in a table. This must work as follows: (4)

a) The userid of the user must be saved to a variable.

b) sYou must use the *getFriendList* and *getUserDetails* functions to get these details. Note that this will require you to use sequential/chained asynchronous calls, since you must use the results of one function when calling the other. You may **not** use async/await functionality to implement this and you may not manage the array data in this function call (the correct array must be returned from *getUserDetails*).

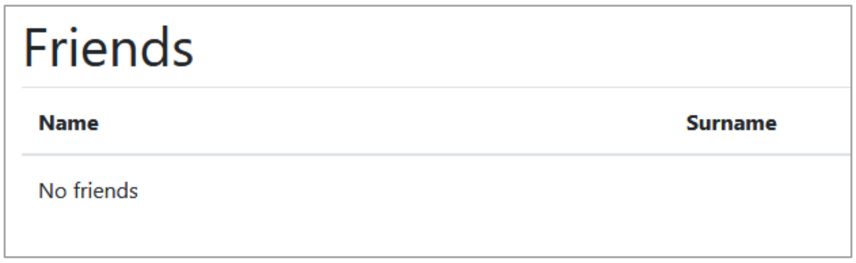
c) Use the *createUserRow* function to create and return the table rows and then append them to the table. For example, when clicking on the second name in the list, the results must be as follows:



Friends

Name	Surname
Abed	Nadir
Troy	Barnes
Shirley	Bennet
Annie	Edison

d) If a user has no friends in their friends array in *friends.json,* only a row must be added with an appropriate message, for example: (1)



Friends

Name	Surname
No friends	

e) If a user has themselves in their friend list, their name should not be displayed. (1)

f) Only the details for the last user clicked on must be shown at any given time.

Note: wherever you are unable to provide the required functionality in the specified manner, such as not using loops, it is recommended that you focus on providing the functionality. You will receive more marks for working code that does not meet the specifications than for code that does not work at all.

Section 2 [16]
NodeJS + ReactJS

Use only the files provided in the "S2" folder to complete this section.

This section requires you to write functionality in JavaScript. You must, wherever appropriate, make use of the following ES6 syntax:

- Arrow functions
- Appropriate variable declarations
- Template strings
- Object destructuring

All functions must be written as function expressions. Furthermore, you are not allowed to use any loops, *forEach* function, or *$.each* function to implement array-based functionality. You must make use of the other JS array functions as discussed in class. Array-based functionality must work for any number of array items.

Install all the required modules for being able to transpile and bundle JSX into valid JS and serve a webpage that renders JSX components. You must use the recommended directory structure and core file names (server.js, index.js, bundle.js, etc.) as discussed in the notes and serve the webpage to port 3000. All React components must be declared inside their own files; naming rules apply. You must also create the necessary scripts inside *package.json* to be able to create all the necessary files and serve the page by running *npm start*.

You will use the data found inside *questions.json* for the core functionality of this section. This file contains a list of web-related questions in a structured format; you may not alter this file in any way.

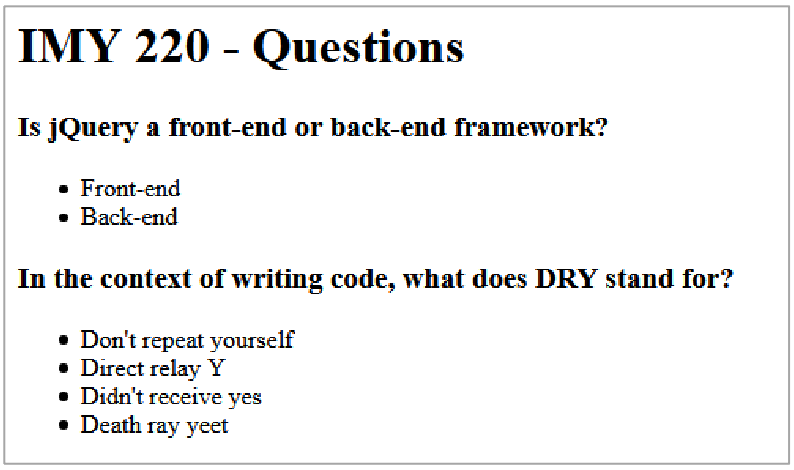
Do not include your **node_modules** folder in your final submission.

2.1 Use NodeJS to read the contents of questions.json and use socket.io to send the file (which can only be read server-side) to the client. (4)
HINT: you can use socket.io's client-side functionality in the same file you use to render your React components.

2.2 Define a component called QuestionList, which takes an array of questions (in the format given in the file) and renders an appropriate heading along with a Question component for each question object in the given file (each question has a unique name, e.g. "q1"). (4)

2.3 A Question component takes a single question object, which contains the question itself and list of possible answers. The Question component must render a div with an id containing the name of the question (e.g. *div#q1*) which contains a level-3 heading containing the question itself and an unordered list containing the list of possible answers. The list items must contain data attributes which indicate which answer is correct. (4)

2.4 The resulting HTML must be served at localhost:3000 and the page must not show errors or warnings in the console. This must look as follows: (4)



IMY 220 - Questions

Is jQuery a front-end or back-end framework?

- Front-end
- Back-end

In the context of writing code, what does DRY stand for?

- Don't repeat yourself
- Direct relay Y
- Didn't receive yes
- Death ray yeet

Note: where you are unable to provide the required functionality in the specified manner, such as not sending the contents of the file to the client using socket.io, it is recommended that you focus on providing the functionality, such as creating a variable with the information contained by the file to render this information. You will receive more marks for working code that does not meet the specifications than for code that does not work at all.