Assessment:	**Web Programming 252**
Duration:	180
Total	70

Assessor:	T. Mkwaira
Moderator:	M. Magorimbo

WEB PROGRAMMING 252 SUMMATIVE TEST

INSTRUCTIONS: GENERAL

- Five minutes are included in the allocated time so that you may read and understand the instructions. During this interval at the start of the assessment, you must seek clarity if you feel you do not comprehend any part of the instructions.
- Keep refreshing assessmentQ to see the accurate remaining time, or time yourself using a watch.
- Invigilation is only conducted on HybeFlex.
- This test will be ended by the invigilator precisely at the end of the stipulated period. Any updates to the assessment will be posted on HybeFlex and the shared message board..
- No instructions or directives of the invigilator shall be disregarded, if any of the instructions are disobeyed, candidates shall expose themselves to disqualification from the current and future assessments.

INSTRUCTIONS: SUBMITTING YOUR TEST

- Uploading: This assessment must be submitted on AssessmentQ only.
- Naming conventions: When uploading work, ensure naming conventions are followed as instructed. See the saving instructions at the end of this assessment if you are not sure.
- Zipping your work: At the end of this assessment, you should only upload a single zip file containing all your work. See the saving instructions at the end of this assessment if you are not sure.
- File formats: Any work that is uploaded must be in zip format ONLY. Other file types may give problems!
- Optimising your solution & preventing archive errors: Delete *node_modules* before zipping and uploading your final solution.
- Emailing (In case of difficulties during submission): Do not use MS Teams for submissions for whatever reason. In case of technical difficulties, you should submit your assessment to your lecturer with a proper subject line that follows the naming convention WPR252 ST: StudentSurname StudentName. Your zip file should also follow the same naming convention.
- Reporting technical issues: Any technical issue related to AssessmentQ and HybeFlex must be reported to the invigilator immediately. You may be asked to provide proof of the error, and to follow reporting procedures that may include using the inbuilt reporting tool in HybeFlex.

Assessment:	**Web Programming 252**
Duration:	180
Total	70

INSTRUCTIONS: INTEGRITY AND LIABILITY

- Integrity: Observe the Honor Code1. Each candidate is expected to maintain academic integrity. Any form of plagiarism (including self-plagiarism) or cheating that is detected can lead to disciplinary measures being taken.
- Liability: It is your responsibility to ensure that you have saved your assessment correctly! We will not accept excuses and explanations after the assessment.
- Late submissions: Late assessments will not be admitted for marking.

INSTRUCTIONS: FORMAT OF THE ASSESSMENT

- This assessment does not include any theory.
- Marks may be reserved for best practices including commenting the important parts of your code etc. and error handling.
- Organise your code well as per the instructions.
- The allocated time includes time for uploading/ saving your solution.
- The mark allocation is subject to change.
- Penalties may be applied for not following the instructions.

Assessment:	**Web Programming 252**
Duration:	180
Total	70

OVERVIEW OF THIS ASSESSMENT.

This practical assessment contains two tasks. The first task is about creating a RESTful API, and the second one is on building a GUI-based application that uses a webpage to create interactions with a user. This assessment considers your skills in HTML (EJS), CSS, Node, Express, MongoDB and other utilities such as REST client. Read the instructions carefully; and **good luck ☺ !**

Remember to read the entire question before you begin, including the notes at the end of each task.

THE REQUIREMENT FOR THIS ASSESSMENT

> *Towards the end of their studies, Belgium Campus students are provided with an opportunity for internship. In order to ensure that there is a need to assist students with this regard, there is need to register the students in a system. The registration will require the full name of the student, a status indicating whether they have already been placed, and the number of subjects that are outstanding.*

You have been asked to build an application that requires the relevant details to be recorded. A detailed description is given in Table 1 below. Please note that the table also includes the validation rules for the required data, and the table applies to both Task 1 and Task 2.

To store values, please note that Task 1 will use MongoDB and Task 2 will only use an array. Remember that if you want to save a value in an array you can use:

Array.push()

Here is the table of constraints for Task 1 and Task 2:

The required data and the validation is described as: -

Property	Data Type	Constraints
studentName	String	Required, Any Valid String e.g. *John Peters*
internshipStatus	String	Required; Can only be either placed, unplaced or unknown
outstandingSubjects	Number	Required, Defaults to 0.

Table 1: Description of values

Assessment:	**Web Programming 252**
Duration:	180
Total	70

TASK 1: REST API, EXPRESS, MONGO DB [35 MARKS]

Write a Node application that allows the user to:

1. Add a new record to your database collection by sending a request to the endpoint */add.* The request must be sent using REST Client, and in JSON format.
2. View all records in the MongoDB collection by sending a request to the endpoint /view.
3. Update a record by sending a request to the endpoint /delete/<id> where the <id> is a value generated by MongoDB when the record was created.

Note

- Use the routes stated in the question. The API will be tested only using the stated routes!
- You must only use mongoose as your ORM, and a local instance of MongoDB as demonstrated during class.
- The name of the database should be *students*.
- You will use JSON for data interchange.
- Use appropriate HTTP codes for response.
- Your schema must be well defined.
- All validation must be handled by the schema.
- Separate all your concerns to improve code maintainability.
- Use comments to explain non-obvious parts of your code.
- Your application must use the npm command to run.

TASK 2: EJS, ARRAY METHODS, EXPRESS [30 MARKS]

Task 2 will no longer use a database, but will use an array of objects to store values instead. The application must receive the values from a form. Validation must be applied using JavaScript (not via a schema, this question does not use a mongoose schema or a database).

By creating a separate project, develop a form-driven application that allows a user to:

1. Add a new record to an array by sending a request to the endpoint '/add'. This route must load a page that contains a form. You may call this page index.ejs.
2. View all the records in the array by navigating to the home route ('/view'). If there are no items in the array, the application must inform the user by writing some text on the web page. Otherwise, the application prints out the requested values. You can use an HTML table or an ordered list.
3. View records for students who have either been placed or not. You will use a URL parameter for this part.

Note

- Create an HTML form for the data entry. If you struggle with the form formatting, do not bother with CSS! However, you should know that marks are reserved for the choice of form elements, variety of the form elements, and CSS application. This may count for a maximum of 5 marks.
- Use a partial to display the form on the index.ejs.

Assessment:	**Web Programming 252**
Duration:	180
Total	70

HOW TO SUBMIT: ZIP ONLY! ZIP ONLY!

NAMING CONVENTION

You must submit a **ZIP** file that uses the Campus standard naming convention. Any assessment that a student creates for electronic submission must conform to the following naming convention:

WPR252 Assessment_StudentSurname_StudentName.zip

The reason for this is to ensure that your assessment is received correctly by your lecturer; this prevents any inconvenience for you as the student and the lecturer.

PACKAGING YOUR SOLUTION

1. Create a folder that follows the naming convention explained above.
2. Move the Tasks folder into the named folder you created in Step 1 (remember to delete node_modules).
3. Finally, create a **zip** archive of the folder you created in Step 1. This folder should now contain the screenshots and the project files.
4. Upload your zip archive.