****

**SESSION:** 2020 / 21 **DIET: 1st**

**Web Platform Development 2**

**(M3I324182)**

**Coursework Specification for Coursework 2**

**(This coursework contributes 50% to your overall module mark.)**

**LEVEL: 3**

**Module Leader: Katrin Hartmann**

**Submission:**

**On or before the 6th December 2020 (unless agreed otherwise)**

1. **Description of the Application (relevant to coursework 1 and coursework 2)**

Coping with many coursework deliverables is a stressful part of every student’s life. Design and implement a web-based coursework scheduling application to help students cope with multiple coursework assignments.  At its core, the coursework scheduling application contains a list of coursework projects that have to be submitted. The application allows the user to define his or her own milestones for each of the coursework projects. These are only visible to this user, i.e. require a login. In order to be useful, the coursework projects and their milestones exist for longer than one session.

The properties of a coursework project can be designed by you but should at least consist of:

* A title of the coursework project,
* The module for which it is to be submitted,
* A breakdown of milestones for each coursework project,
* An intended due date and
* The actual completion date.

The coursework scheduling application should provide the following core functionality for coursework projects:

* Coursework projects can be added.
* Coursework projects can be removed.
* Coursework projects can be modified.
* A listing of all incomplete courseworks and their milestones can be viewed.
* Users should be able to share a project with other users by sending a link. At a base level you can assume that anyone who has the link has permission to see that project.

The application should be developed using Node.js and Node Express.

Any application features not specified here can be designed by you. If your application extends the specification you should highlight the additional features in your report.

1. **Working Details**

You should work individually on this project.

The project should be under GIT version control and regularly pushed to a private repository on either Bitbucket or Github.

All functionality of the application must be in the master branch.

GIT functionality should be utilized, e.g. regular commits, useful comments, branches etc. Simply pushing a completed project to the remote before the submission date will not yield many marks. The remote repository is also part of the deliverables and should be well presented, e.g. have a readme.md file.

The project should run from npm (npm run install and npm run build).

The sequence of commands to install and run the application should be documented in the readme.md file.

Push your final coursework to the remote repository before the deadline. You can submit your project early; in this case simply push your last commit to the remote at an earlier date.

Reports should be uploaded to GCULearn by the deadlineusing the Report Upload link in the Assignments section.

1. **Marking Scheme**

**Distribution of Marks**

* 60% for the functionality of the developed application and
* 40% for a report of approximately 2000 words in length explaining how you implemented the points listed below:
* The link design within the application, i.e. a mapping of links to functionality and logic of the link schema. (10%)
* A description of the persistence mechanism including any database schema or document structure and the data access layer of the application. (10%)
* Documentation of the functionality of the application and test reports. (10%)
* An appraisal of the application security. NB: this should include a reflection on the actual security of the application as well as show your awareness of possible security threats. Highlight if you have implemented any security measures but also describe additional security measures that you haven’t implemented (as this isn’t a specific security module). Here marks are also awarded for awareness. (10%)
* Add the statement: “This piece of coursework is work of <insert name of the author> and has not been submitted elsewhere in fulfilment of the requirement of this or any other award.” to the title page.

In the report it is practical to cover each discussion point in one section. The marks are allocated equally between the sections, so you can distribute the volume \_roughly\_ equally between the sections. The volume is indicative but not linked to the marking scheme.

**Interpretation of the Marks**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| < 40% | 40% - 49% | 50% - 59% | 60% - 69% | 70%+ |
| No solution provided or provided solution has major omissions so that it doesn’t cover what has been asked for or isn’t useful. | A solution has been provided but is below of what can be expected with omissions or non-functioning components. | As expected; the solution covers what was asked for but with room for improvement. | A very good, thoughtful, clear and complete solution. | An excellent extensible solution that goes beyond of what has been asked for in some way. |

Further guidance of how this commentary is applied to the individual sections can be found below.

**Marking Scheme for the Assessment of Application Functionality**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| < 40% | 40% - 49% | 50% - 59% | 60% - 69% | 70%+ |
| Does not run or shows major omissions of the required functionality. | A usable solution has been provided but it doesn’t work well or has omissions in features. | Provided functionality covers what has been asked for but with room for improvement. | A very good and complete solution for the required functionality. | An excellent extensible solution that goes beyond of what has been asked for in some aspect. |

**Coursework Report Template**

1. Link design

[Explain the mapping of links to functionality as well as the rationale for the links.]

1. Persistence

[Describe the persistence mechanism including any database schema and the data access layer of the application.]

1. Test reports

[A collection of test reports and summary of results.]

1. Application security

[Provide a reflection on the actual security of the application as well as show your awareness of possible security threats. Highlight if you have implemented any security measures but also describe additional security measures that you haven’t implemented.]

**3.1 Further guidance on the link design section**

List the URLs to which your application responds and explain the rationale for your URL design. For each URL describe briefly what kind of response it produces. **Keep in mind that URLs may get bookmarked. For reference examine production level examples of URL design, e.g. Stackoverflow, Bitbucket, GitHub etc.**

The URI identifies a resource on the web. The general build is: **scheme://domain/path?query\_string#fragment\_identifier**

Explain the reasoning behind your scheme, what functionality does your application provide and how does a user get to each page. Each web page can typically be reached by users through at least a single uniform resource locator (URL).

Here you can only control the part of the URL after the domain. Assume one domain.

Could your scheme accommodate additional functionality if it was to grow?

**Good examples (there are many good schemes, don’t look for a single “correct” solution):**

* are consistent,
* human readable,
* not too long (unless the first part also works alone),
* come from one domain,
* allow for future addition of new application features (i.e. pages),
* don’t have any of the “features” listed below (without clear rationale).

**Bad examples:**

* Session ids in the URL, e.g. http://my.example.com/index.jsp;JSESSIONID=ABCD123
* Non-ascii characters in the URL
* Many useless search terms in the URL
* Show private data in the URL, e.g. e.g. http://my.example.com/account.jsp?creditcard=12345678

/feature/<username> is ok for publicly available data

private data should just come after, e.g. /account

Keep in mind that you are designing URLs for an application, not an API.

If you want to go beyond there are modules which can help you to visualise your URLs:

https://www.npmjs.com/package/express-list-routes

https://www.npmjs.com/package/express-routes-visualizer

**Marking Scheme:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| < 40% | 40% - 49% | 50% - 59% | 60% - 69% | 70%+ |
| Incomplete routes and/or showing poor rationale or design. | Provides sufficient routes for the application to work but little coherent design or rationale. | Complete set of routes provided but incomplete or inconsistent design. | Clear and complete outline of routes with good design and justification. | Excellent extensible link schema that has been thoughtfully designed and implemented. |

**3.2 Further guidance on the persistence section**

Outline your document structure, i.e. the data structure that is persisted by your application and how you map it to key value pairs as well as how you keep it in your datastore.

Explain how you handle persistence in your application, e.g. do you have a designated part of the application that is responsible for this?

What methods does your code provide to the rest of the application?

**Marking scheme:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| < 40% | 40% - 49% | 50% - 59% | 60% - 69% | 70%+ |
| Incomplete or non-functioning persistence. | Provides persistence but not always complete or minor non-functioning methods. | Provides sufficient persistence but persistence not provided in a separate application layer or documents don’t have a clear structure. | A very good persistence solution with a good document structure and a complete set of methods providing access to the data in a separate layer of the application. | Excellent extensible persistence solution. |

**3.3 Further guidance on the test reports**

Think about how you know that the application works the way you intend it to work. In the context of this module there are 2 main levels of tests that can be applied:

**Unit tests:**

In this section list the unit tests (but not the code). State the test case and class / method under test. State whether the tests run clear.

**System tests:**

In this section test the integrated system in order to evaluate the system’s compliance with the specified requirements. This section is the more important section.

**Fictional Sample Test Section for System Tests (based on a login example):**

**Test Scope**  
Functional Testing was carried out for the following modules:

1. **Landing page (index.html)**
2. **Login**
3. **Sending messages**

**Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Action** | **Expected outcome** | **Status** | **Evidence** |
| **1.1** | **localhost:9000** | **Landing page loads** | OK |  |
| **1.2** | **localhost:9000/index.html** | **Landing page loads** | OK |  |
| **1.3** | **Click “Send a message” (user not logged in)** | **Login page loads** | OK |  |
| **1.4** | **Click “Send a message” (user logged in)** | **Message page loads for this user** | OK |  |
| **2.1** | **Provide username and password for registered user and click Login button** | **Message page loads for this user** | OK |  |
| **…** |  |  |  |  |

**Summary test results**

The functionality of the landing page works as expected. All test results are as expected.

Most of the functionality of the login page (11 out of 12 tests) works as expected. The remaining issue is that when a non-registered user uses the login button (instead of the register button) the user is directed back to the login page but no error message is shown. …

**Marking scheme:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| < 40% | 40% - 49% | 50% - 59% | 60% - 69% | 70%+ |
| Insufficient test cases or TEST REPORTS ARE NOT TRUTHFUL (!)\* | Limited number of test cases considered. | Tests cover most of the functionality and the application passes a good part of those. | Tests cover all required functionality and the application passes most of those. Good honest appraisal. | Excellent and complete coverage of tests including atypical test cases which the application passes. Excellent summary of results. |

**\***Although passing all tests demonstrates a better application it is acceptable that the application doesn’t pass all tests because knowledge of failing tests allows us to fix these later. The worst-case scenario is a test report that hides non-functioning parts of the application. In addition, the choice of test cases shows awareness of the application requirements.

The best case: The application passes all tests and the tests provide a good coverage of the required functionality.

Second best: A good test coverage of functionality but the application doesn’t pass all the tests (yet).

The worst case: A test report that either doesn’t cover those parts of the functionality that aren’t working or says that functionality is working when it is not.

It is also useful to include (small!) screen shots of test results.

* 1. **Further guidance on the appraisal of the application security**

**Consider:**

The type of application that you have developed and where potential security risks might be.

**Example Issues:**

1. Does your server use sessions?

* Any session cookie can be observed in developer tools, application tab
* Observe it: what happens if you delete it, is it always the same?
* What does this mean for application security?
* Could the application be made more secure, e.g. using httpOnly or secure flags?

1. Does your application access user generated input?

* Can users be relied on to use the application as intended? What might happen if they don’t?
* Can that be mitigated? Could additional measures be taken to make the application more secure, independent of whether this has been implemented? Which measures?

1. Are you working with passwords?

* Are there any security risks, e.g. should the password be stored as clear text in the database?
* What can be done to make an application more secure?

1. What information is reviled when your application crashes, or when someone accesses a link that doesn’t work? Can that be exploited?

**Highlight if you have implemented any measures. Also highlight, if you are aware of additional measure that you haven’t implemented (because this isn’t a security module).**

**NB: Here you are also marked on awareness and not necessarily on how much you have already implemented.**

**Marking scheme:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| < 40% | 40% - 49% | 50% - 59% | 60% - 69% | 70%+ |
| Little awareness of security risks. | Some awareness of security risks but not always in the context of this application. | A basic awareness of security risks but awareness doesn’t cover many aspects of this application or vulnerabilities are not always clearly linked to this application. | Very clearly described awareness of security risks and their mitigation in the context of this application. Some (but not all) mitigation measures implemented. | Excellent awareness of many aspects of application security. Clearly linked to the application developed. Some (but not all) mitigation measures implemented. |

**[3.5 Add the true and signed statement of authorship to the title page.]**