

# ECM 1417: Web Development

## Continuous Assessment

Diego Marmsoler

Submission date: Friday 27th March

For your continuous assessment you will develop a multi-user task management application. You can get up to 100 marks for the assessment. The assessment is worth 30% of the module mark. You need to submit your source code and any supporting files via the electronic submission system at <http://empslocal.ex.ac.uk/submit/><sup>1</sup>.

If you are using a local development environment keep in mind that you need to deploy your final application to one of the blue servers which have PHP version 5.4.16 installed. Thus, either install the same version for your local development environment or deploy your application from time to time to check if everything works as expected.

Moreover, note that on the server there is no `mysqlnd` installed. Thus, you cannot use `get_result()` for prepared statements. Instead you need to use `bind_result()`.

In the following, I list the requirements of the application. For every requirement you can find the number of marks which you can get for implementing this requirement.

---

<sup>1</sup>I will provide a document with detailed instructions on how to submit your work on the ELE page.

# 1 Basic Requirements

20 marks

The application should provide the following functionality:

- Login for existing users.
- Registering a new user.
- Display all tasks.
- Add a new task.
- Edit a task.
- Delete a task.

For the purpose of this application, a task consists of:

- a name,
- a description,
- a due date, and
- a state which is either *done* or *not done*.

In the following, we describe the basic functions in more detail. Note that the web application is required to run on different web browsers. Thus, to avoid any problems you should stick to well-formed HTML.

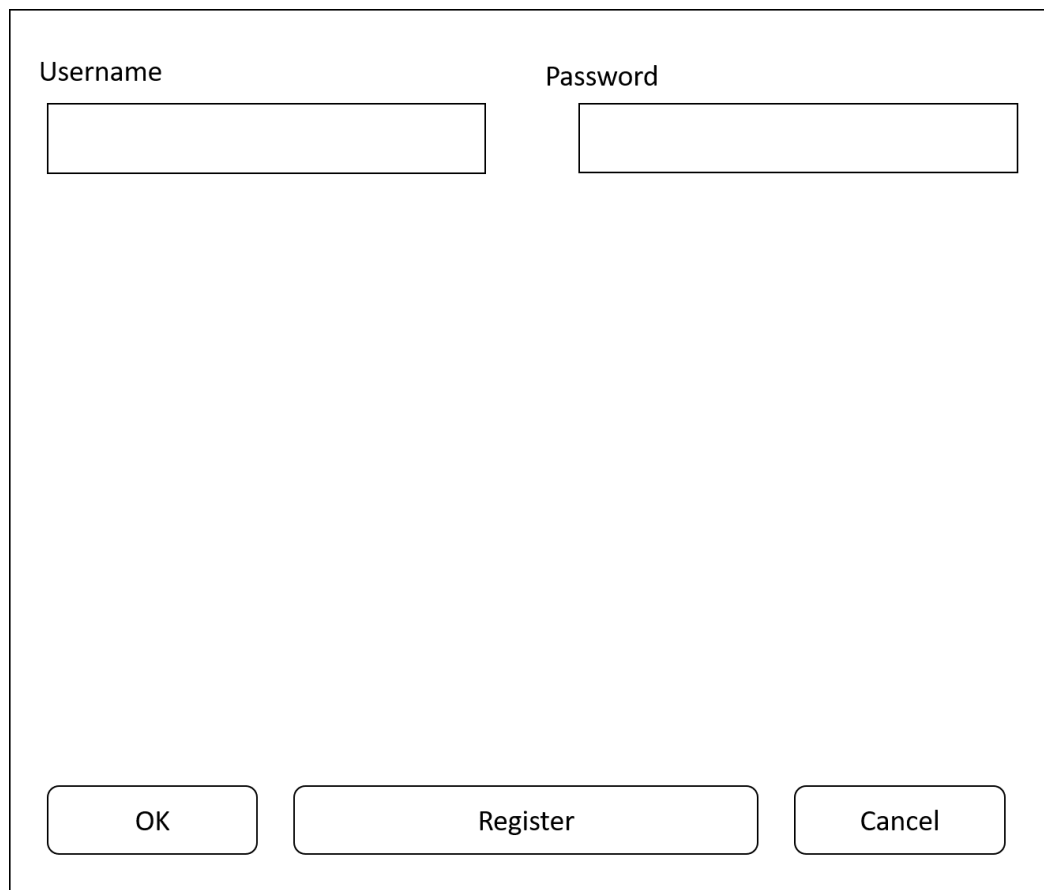
Throughout this text I will provide some mock-up forms to give you an idea of how a form should look like. However, this is only to provide you with a rough direction and you are welcome to adapt it to your own preferences.

## 1.1 Login

The application should start with a login page where an existing user can login by means of a username and password. Figure 1Login formfigure.1 shows a sketch of such a page. To register new users, the page should also contain a link to a corresponding registration page.

## 1.2 Registration

The registration page should allow a user to register a new username and password. Figure 2Registration formfigure.2 depicts a possible layout of such a page.



A login form diagram enclosed in a rectangular border. At the top left, the label "Username" is positioned above a rectangular input field. To its right, the label "Password" is positioned above another rectangular input field. At the bottom of the form, there are three rounded rectangular buttons arranged horizontally: "OK" on the left, "Register" in the center, and "Cancel" on the right.

Figure 1: Login form.

A registration form with a white background and a black border. At the top left, the label "Username" is positioned above a rectangular text input field. To its right, the label "Password" is positioned above another rectangular text input field. At the bottom left, there is a rounded rectangular button labeled "OK". At the bottom right, there is a rounded rectangular button labeled "Cancel".

Figure 2: Registration form.

Add	<input type="checkbox"/>	Name / Due Date	Edit	Delete
	<input type="checkbox"/>	Name / Due Date	Edit	Delete
	<input type="checkbox"/>	Name / Due Date	Edit	Delete
	<input type="checkbox"/>	Name / Due Date	Edit	Delete

Figure 3: Display form.

### 1.3 Display List of Tasks

The application should allow to display a user's task list. For each task, the state of the task, its name and its due date must be displayed. In addition it should be possible to edit or delete tasks or to add new tasks. Figure 3Display formfigure.3 displays a possible layout for the display page.

### 1.4 Edit

It should be possible to display the details of a task (including its description) and, if necessary, to edit them. Figure 4Edit formfigure.4 shows a possible layout for an edit form.

**Taskname**

Description

Due Date

State

OK Cancel

The form is a rectangular box with a light gray background. At the top left, the label 'Taskname' is in bold. Below it is the label 'Description' followed by a large rectangular text input field. Below the description field are two labels: 'Due Date' and 'State'. Under 'Due Date' is a rectangular text input field. Under 'State' is a rectangular container with two radio button options: 'Done' (with a filled black circle) and 'To be Done' (with an empty circle). At the bottom left is a rounded rectangular button labeled 'OK'. At the bottom right is a rounded rectangular button labeled 'Cancel'.

Figure 4: Edit form.

## 2 Styling Requirements

15 marks

Now that you got your basic application working, we are going to style it.

### 2.1 Basic Styling

Use CSS to add some basic styling to your application.

Keep in mind that I only require you to do *basic* styling. Thus, be careful not to put too much effort into the styling of your pages.

### 2.2 Responsive Design

Use CSS's responsive design support to provide an alternative layout for your application if used on smaller displays, such as a mobile phone. Figure 5 Login form figure.5 - Fig. 8 Edit form figure.8 display possible layouts for small screens (Compare them with the corresponding layout for larger screens provided above).

Username

Password

OK

Cancel

Register

Figure 5: Login form.

Username

Password

OK

Cancel

Figure 6: Registration form.

<input type="checkbox"/>	Name Due Date	
	Edit	Delete
<input type="checkbox"/>	Name Due Date	
	Edit	Delete
<input type="checkbox"/>	Name Due Date	
	Edit	Delete
Add		

Figure 7: Display form.

Taskname

Description

Due Date

State

☒ Done
☐ To be Done

OK

Cancel

Figure 8: Edit form.



### 3 User Requirements

15 marks

Use secure sessions to keep a user logged in when working with your application. Add a button to the display form to let a user log off.

In addition, create two different themes for your web application (for example a light and dark theme)<sup>2</sup>. Moreover, add a page where a user can set her/his preference theme. Use cookies to store a user's preferences and display the web application using the correct theme.

---

<sup>2</sup>Again, keep the themes simple. It could just be different background colors, for example.

## 4 Security Requirements

15 marks

User passwords must be encrypted using additional salt and pepper. In addition, the application must be resistant against Cross Site Scripting attacks and SQL injections.

## 5 Requirements for Client Side Scripting

15 marks

Use client side scripting to increase the user experience. To this end, use JavaScript to change the state of a task on the display page. Moreover, use AJAX to notify the server about such changes and store them to the database.

## 6 Architecture Requirements

10 marks

Use the Model View Controller pattern to structure your application:

- Strictly separate programming code which accesses the database from code which merely displays code.
- Use a front controller to rout requests to the correct page.
- Reuse views for different pages: for example use only one view for editing/adding a task.

Name	method	endpoint	input	output	Exceptions
<i>getTasks</i>	GET	tasks.php	n/a	tasklist	405
<i>getTask</i>	GET	task.php/{id}	n/a	task	405,400,404
<i>check</i>	PUT	check.php/{id}	user	n/a,	405, 400, 404, 409
<i>uncheck</i>	PUT	uncheck.php/{id}	user	n/a	405, 400, 404, 401
<i>addTask</i>	POST	add.php	taskinfo	taskid	405, 400, 404, 409

Table 1: REST API of external project management system.

## 7 Web Service Requirements

10 marks

For the purpose of this assessment I prepared an external web service which provides operations to import and manipulate tasks from an external application. In the following you can find a description of the web service's API as well as a description of the requirement you should address.

### 7.1 Web Service API

The web service provides four operations to get a list of tasks, get details of a certain task, and mark a task as *done* or *not done*. An external task also contains information about which user has finished a certain task. The web service is accessible at the following location: <http://students.emps.ex.ac.uk/dm656/>

The service's REST API is described in Tab. 1 REST API of external project management system table.1. It lists the name of an operation, the method which can be used to access it, the endpoint to address it, as well as input and output parameters.

In the following we describe the service's data format and error handling mechanism.

#### 7.1.1 Data Formats

The service uses XML to encode the different types of data. Note that due dates are always encoded in the form: YYYY-MM-DD HH:MM:SS

**Tasklist** A tasklist contains multiple tasks containing an identifier, name, and due date, each. An example encoding of a tasklist is shown below:

```
<tasks>
  <task>
    <id>1</id>
    <name>Do Web Development Coursework</name>
    <due>2020-03-27 23:59:59</due>
  </task>
```

```
<task>
  <id>2</id>
  <name>Learn for exam</name>
  <due>2020-01-24 19:54:09</due>
</task>
</tasks>
```

**Taskinfo** Taskinformation contains a name, due date, and description. An example encoding of a taskinformation is shown below:

```
<taskinfo>
  <name>Do Web Development Coursework</name>
  <due>2020-03-27 23:59:59</due>
  <description>Some initial description</description>
</taskinfo>
```

**Task** A task contains an identifier, name, due date, and description. An example encoding is given below:

```
<task>
  <id>1</id>
  <name>Do Web Development Coursework</name>
  <due>2020-03-27 23:59:59</due>
  <description>Some initial description</description>
</task>
```

**Taskid** A task identifier contains only an identifier for a task. An example encoding of a task identifier is shown below:

```
<task>
  <id>1</id>
</task>
```

**User** A user contains only an identifier. An example encoding of a user is shown below:

```
<user>
  <id>1</id>
</user>
```

Add	<input type="checkbox"/>	Name / Due Date	Edit	Delete
Import	<input type="checkbox"/>	Name / Due Date	Edit	Delete
	<input type="checkbox"/>	Name / Due Date	Edit	Delete
	<input type="checkbox"/>	Name / Due Date	Edit	Delete

Figure 9: Extended display form (large).

<input type="checkbox"/>	Name Due Date	
	Edit	Delete
<input type="checkbox"/>	Name Due Date	
	Edit	Delete
<input type="checkbox"/>	Name Due Date	
	Edit	Delete
Add		
Import		

Figure 10: Small.

### 7.1.2 Failure Handling

The service uses the following error messages:

**400 Bad Request** if an identifier (user or task) was passed to an operation which is not a number.

**401 Unauthorized** if a user tries to uncheck a task which she/he did not originally check.

**404 Not Found** if a task with a certain identifier does not exist.

**405 Method Not Allowed** if an operation is accessed using the wrong HTTP method.

**409 Conflict** if a user tries to mark a task as done which is already marked as done.

## 7.2 Import Requirement

Add a feature to your application which allows you to import tasks from the project management platform using the web service described above. To this end, first add a new button to the display form to open the import form (Fig. 9Extended display form (large)figure.9 and Fig. 10Smallfigure.10).

This should bring up an import form such as the one depicted in Fig. 11Import form (large)figure.11 and Fig. 12Smallfigure.12.

After selecting the tasks to be imported, a confirmation window, such as the one displayed in Fig. 13Import form (large)figure.13 and Fig. 14Smallfigure.14, should pop up to ask the user for confirmation. If the user agrees, the different

Select the tasks to import:

Name -- Description -- Due Date
Name -- Description -- Due Date
Name -- Description -- Due Date
Name -- Description -- Due Date
Name -- Description -- Due Date
Name -- Description -- Due Date
Name -- Description -- Due Date

OK Cancel

Figure 11: Import form (large).

Select the tasks to import:

Name Due Date
Name Due Date
Name Due Date
Name Due Date
Name Due Date
Name Due Date
Name Due Date

OK Cancel

Figure 12: Small.

tasks should be imported using the web service’s API. Note that also the initial description should be obtained when importing the tasks into the application<sup>3</sup>. If a remote task is marked as *done* or *not done*, this information should be stored in the local database and reflected in the database of the remote application, i.e., you need to change the external tasks state using the REST API<sup>4</sup>. If the task was already done by someone else in the meantime, then you should notify the user about this and delete the task from the local database.

### 7.3 Export Requirement

Modify the edit form to add a new export button such as depicted in Fig. 15Extended edit form (large)figure.15 and Fig. 16Smallfigure.16.

After clicking the export button, a confirmation window, such as the one displayed in Fig. 17Edit form (large)figure.17 and Fig. 18Smallfigure.18, should pop up to ask the user for confirmation.

After confirming the export, the task should be added to the external system using the REST API. Note that the task should still be available in your local system.

<sup>3</sup>You may need to add a new column to the task table which stores the remote taskid of a (remote) task

<sup>4</sup>Note that this requires that you store the identifier of the external task with your local task.



Select the tasks to import:

Name	Description	Due Date
Name	Description	Due Date
Name	Description	Due Date
Name	Description	Due Date
Name	Description	Due Date
Name	Description	Due Date
Name	Description	Due Date
Name	Description	Due Date
Name	Description	Due Date

Are you sure you want to import xxx tasks?

OK Cancel

OK Cancel

Figure 13: Import form (large).

Select the tasks to import:

Name	Description	Due Date
Name	Description	Due Date
Name	Description	Due Date
Name	Description	Due Date
Name	Description	Due Date
Name	Description	Due Date
Name	Description	Due Date
Name	Description	Due Date
Name	Description	Due Date

Are you sure you want to import xxx tasks?

OK Cancel

OK Cancel

Figure 14: Small.

**Taskname**

Description

Due Date

State

☒ Done
 ☐ To be Done

OK

Export

Cancel

Figure 15: Extended edit form (large).

**Taskname**

Description

Due Date

State

☒ Done
 ☐ To be Done

OK

Cancel

Export

Figure 16: Small.

The large edit form contains the following elements:

- Taskname**: A text input field.
- Description**: A large text area.
- Due Date**: A date input field.
- Status**: A section containing two radio buttons: ☐ Done and ☐ To be Done.
- Buttons**: Three buttons at the bottom: **OK**, **Export**, and **Cancel**.

A confirmation dialog is displayed in the center, asking: "Are you sure you want to export the task?". It has two buttons: **OK** and **Cancel**.

Figure 17: Edit form (large).

The small edit form contains the following elements:

- Taskname**: A text input field.
- Description**: A text area.
- Due Date**: A date input field.
- Status**: A section containing two radio buttons: ☐ Done and ☐ To be Done.
- Buttons**: Three buttons at the bottom: **OK**, **Cancel**, and **Export**.

A confirmation dialog is displayed in the center, asking: "Are you sure you want to export the tasks?". It has two buttons: **OK** and **Cancel**.

Figure 18: Small.