# *Web Programming III (420-C31-HR)*

# *Assignment 4 – Simple Kanban System*

Date assigned: Friday, November 26, 2018

Date due: **Tuesday, December 11, 2018 (last day of classes)**

**Learning Objectives**

Upon successful completion of this assignment, the student will be able to:

* Create a web application using an All-in-One form for PHP
* Use a node.js server to return information to a client
* Use AJAX to update information automatically

Overview

You are working in support for an Agile Kanban System. You are going to create a system to add tasks for a software project and track the work done on those tasks to completion. This system will have a server component running PHP which is used by day-to-day workers to create new and update existing tasks. This will be done using an all-in-one page on CSDEV (the O drive). And YES, it MUST run on the O drive.

Managers want to be able to view the Kanban board information by selecting the information from a list. They use a windows desktop browser or mobile phone (responsive design required) to access the lists. Their “dashboard” displays a list of the current “In Development” tasks and allows them to select one to display details. Managers can NOT make changes to the list.

Managers are accessing a node.js web server for all their web needs. There “dashboard” must update automatically every 30 seconds to display the current selected list each time.

The Application

1. There are obviously many multiple parts to this application involving PHP, node and AJAX (and XML or JSON too). Break things down to make it easier…

**PHP**

1. Task creating/editing. Create an all-in-one PHP form to create, update and save tasks (tasks described later). This file must be called manageTasks.php and must run on the O drive in the folder O:\C31A04PHP\manageTasks.php (URL: cadev.cegep-heritage.qc.ca/students/studnum/C31A04PHP/manageTasks.php). All tasks are stored in a SINGLE XML or JSON file (your choice) called Tasks.xml (or .json) in the folder O:\C31A04PHP\Tasks. The user must be able to add a new task or select an existing task and update its information.
2. Return list. Create a PHP file called getTaskInfo.php in the same folder as above which will return a list of all tasks list based on the requested status (All, To Do, In Development, In Test or Complete). This means that getTaskInfo.php has one parameter passed either through a get or post action. The parameter is called “status” and has the value for the status (all, todo, indev, intest, complete). This list consists of an XML or JSON response object. The fields returned are: task id, title, date updated, and status. This program has no screen interface and simply returns an HTTP response object (see getAlbums.php from assignment 1).
3. Return details. Create a PHP file called getTaskDetail.php which returns all the information about a single (requested) task from the list. This php program has one parameter passed through either a get or post which is the id of the task to return. It returns all the information about the task.

**Node**

1. Node web server. Create a node web server using port 7546 which serves files from the folder public. This must be called taskServer.js and be located in the folder C31A04Node. The node web server will render pages from the public folder located below the current folder (C31A04Node\public). The default file will be index.html but any html, htm, jpg, png, gif, xml, pdf or ico file in the folder or on a path from that folder can be returned. You must handle favicon.ico as well.
2. Node request. When node receives a request to get a list of the tasks it will send a request (use the request module) to csdev.cegep-heritage.qc.ca/student/username/C31A04PHP/ getListInfo.php with the single parameter of all, todo, indev, intest, complete. This request will return either an XML or JSON file which the node server will then return to the calling agent (manager).
3. Node request. When node receives a request to get a specific task it will send a request (use the request module) to csdev.cegep-heritage.qc.ca/student/username/C31A04PHP/ getTaskDetail.php. This will return the details about a single task either as XML or JSON. The node server **MUST** then format the information as an HTML snippet (using a template as desired) and return to the calling agent.

NOTES FOR NODE:

* The tasks need to be sorted in descending order by lastUpdate date. This can be done on the client, but it is better done in the PHP program (the “Model”) or at least here (the “Controller”) and not the client side (the “View”).
* You must provide a package.json with all its dependencies so that all the appropriate modules will be downloaded using an npm –install.

**AJAX**

1. Management will be using an html page through the node server to access the information. There will be a file called kanban.html in the public folder of on the node web server. When this file is displayed, it will run a JavaScript program which will call the node web server with an HTTP get (I would use the fetch command; you can use jQuery if you wish). The get to the node web server will make the web server call getTaskInfo.php on CSDEV to return the list. By default the “In Development” list of tasks is returned, but the manager must be able to choose (from the returned page) which list to view (All, To Do, In Development, In Test or Complete). The list updates automatically every 30 seconds with the current information.
2. The manager will also have the option from the list page to select one task to view the details of. How the task is selected is up to you (a hyperlink on the task title would be the most effective, but it could be a checkbox, a select link, or even typing in the task’s id (although that would be poor interface design!)). It is up to you to decide how it is done. When a task is chosen, a request is made to the node web server that will make the node web server call the php program on csdev with getTaskDetail.php. The detailed information returned to the node server from the php server (which is returned as HTML) is then be displayed on the device.

NOTES FOR HTML/AJAX:

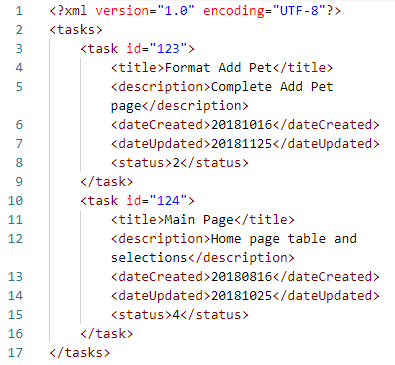
* Managers can use their phones or desktop. This means that this page must be responsive at the 768px breakpoint. Feel free to use bootstrap or another library to do this.
* Colours would be nice…maybe different statuses in different colours, etc.
* Management information can all be done in one page; that is, the details of the chosen task and the list can be displayed at the same time. This should save you some logic/work.
* Remember that CSS and JavaScript must be kept in external files in appropriate folders.

**Task Record**

There needs to be a tasks file. It can be either XML or JSON; as in Tasks.xml or Tasks.json. The following information must be collected for a task:

|  |  |  |
| --- | --- | --- |
| Item | Name | Description |
| Task Identifier | id | A unique identifier for a task. This is a sequential integer and **generated by the system**. The user never enters the ID and, indeed, should never see it |
| Title | dTitle | A short description of the task. |
| Description | description | A description of the task |
| Date Created | dateCreated | The date the task was created |
| Date Updated | dateUpdated | The last date the task was updated |
| State | state | The current state of the task from 1-4 (To Do, In Development, In Test or Complete) |

Here is a sample file in XML:



And the same data in JSON (notice there is not task root entry):



**To submit**

When you have completed the assignment, zip the files as YourUserName\_C31A04 and save the zip file to the page for the course.