

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| **Document title** | **Student Progression** |
| **Document reference number** | **UEL -** |
| **Project name** | **UNISA e-Content 20140363** |
| **Unisa Department** | **Department of Economics** |

Table of Contents

1 Description of the component 5

2 How to Install The component 5

3 how to use the component 5

Document sign-off

This document has been seen and accepted by the following people:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Title | Date | Signature |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Compiler | Purpose | Change Summary |
| <yyyymmdd> | Name Surname | Document Created,  Internal Review,  External Review,  etc | Initial release,  Updates based on internal review,  Updates to section xxx,  etc |
|  |  |  |  |
|  |  |  |  |

# Description of component

Student Progression is a feature that enforces a student to follow a learning path by tracking their interaction with a storyline**.**

# How to Install the Student Progression Component

### Step 1.1

Install Laravel framework. The following command will install the latest version of Laravel with the project name LaravelDefault.

composer create-project laravel/laravel LaravelDefault

### Step 1.2

Download Zip from [student-progression repo.](https://github.com/EonConsulting/storyline-table)

## Step 2 - Configuration:

### Step 2.1

Create a folder called "Packages" inside of the root of the Laravel project.

### Step 2.2

Paste the folder from the Git repo (Step 1.2) inside the Packages folder.

### Step 2.3

Inside the root Laravel project, open the composer.json file. In that file, look for:

"autoload" : {

"psr-4": {

...

Inside of the psr-4 tag, load the following:

"EONConsulting\\Storyline\\Table\\": "packages/student-progression/src/"

### Step 2.4 - Use as a global function instead of by namespace (Optional)

## Step 3 - Registering the Service Provider

Open up config/app.php. In the providers array, enter the following:

***\EONConsulting\Storyline\Table\StorylineTableServiceProvider::class****,*

## Step 4 - Tsugi

To be able to use tsugi, the APP\_URL in your env file needs to be correct. So change it to the URL you are using for your testing environment.

## Step 5 - The Finale

In the command line / terminal, enter the following command in the root of the Laravel project:

composer dump-autoload –o

## Usage

### DefaultController

namespace EONConsulting\Student\Progression\Http\Controllers;

use EONConsulting\LaravelLTI\Http\Controllers\LTIBaseController;

use Illuminate\Http\Request;

use EONConsulting\Storyline2\Models\Storyline;

use EONConsulting\Storyline2\Models\StorylineItem;

use App\Models\StudentProgress;

//use EONConsulting\Storyline\Table\Csv;

class DefaultController extends LTIBaseController {

/\*\*

\*

\* @param Request $request

\* @return type

\*/

public function storeProgress(Request $request) {

$StudentProgress = new StudentProgress();

$StorylineItem = new StorylineItem();

$progress = $StudentProgress::whereStudentId($request->get('student'))->first();

if (!empty($progress->id)) {

$ItemArray = $this->topics($StorylineItem, $request->get('storyline'));

$current = $this->save\_progress($StudentProgress, $request->get('id'),$progress->id,$ItemArray);

$StorylineItem = $StudentProgress::find($progress->id);

if($current === 'true'){

$message = 'true';

$story = $progress->furthest;

}elseif($current === 'current'){

$message = 'true';

$story = $StorylineItem->current;

}elseif($current === 'false'){

$message = 'error';

$story = $StorylineItem->furthest;

}

else{

$message = 'error';

$story = $StorylineItem->furthest;

}

} else {

$ItemArray = $this->topics($StorylineItem, $request->get('storyline'));

$ItemId = $this->save($StudentProgress,$StorylineItem, $request,$ItemArray);

$progress = $StudentProgress::find($ItemId);

$progress->furthest = 2;

$progress->save();

$message = 'false';

$story = $ItemId;

}

$response = array(

'msg' => $message,

'story' => $story,

);

return \Response::json($response);

}

/\*\*

\*

\* @param type $StudentProgress

\* @param type $current

\* @param type $progressId

\* @param type $ItemArray

\* @return string

\*/

public function save\_progress($StudentProgress, $current,$progressId,$ItemArray) {

$Progress = $StudentProgress::find($progressId);

$currentIndex = array\_search($current, $ItemArray);

$furthestIndex = array\_search($ItemArray[$Progress->furthest], $ItemArray);

//$array = array\_diff($ItemArray, [$Progress->root,$Progress->current,$Progress->furthest]);

//dd($currentIndex,$furthestIndex);

if ($currentIndex > $furthestIndex) {

return 'false';

} elseif($currentIndex == $furthestIndex){

$Progress->current = $Progress->furthest;

end($ItemArray);

$lastIndex = key($ItemArray);

if($lastIndex === $furthestIndex){

$Progress->furthest = $furthestIndex;

}else{

$Progress->furthest = $furthestIndex+1;

}

$Progress->save();

return 'current';

}elseif($currentIndex < $furthestIndex){

return 'true';

}

}

/\*\*

\*

\* @param type $StudentProgress

\* @param type $request

\* @return type

\*/

public function save($StudentProgress, $StorylineItem, $request, $ItemArray) {

$ItemId = (int) $request->get('id');

$Item = $StorylineItem::find($ItemId);

$StudentProgress->student\_id = (int) $request->get('student');

$StudentProgress->course\_id = (int) $request->get('course');

$StudentProgress->storyline\_id = (int) $request->get('storyline');

$Index = array\_search((int) $Item->id, $ItemArray);

$StudentProgress->furthest = $Index;

$StudentProgress->current = $Index;

$StudentProgress->root = $Index;

if ($StudentProgress->save()) {

return $StudentProgress->id;

}

}

/\*\*

\*

\* @param type $StorylineItem

\* @param type $storylineId

\* @return type

\*/

public function topics($StorylineItem, $storylineId) {

$result = $this->items\_to\_tree(Storyline::find($storylineId)->items);

usort($result, [$this, "self::compare"]);

foreach ($result as $descendant) {

$children[] = $descendant['id'];

}

return $children;

}

/\*\*

\*

\* @param type $items

\* @return type

\*/

public function items\_to\_tree($items) {

$map = [];

foreach ($items as $k => $node) {

$map[] = [

'id' => (string) $node['id'],

'text' => $node['name'],

'parent\_id' => ($node['parent\_id'] === null) ? "#" : $node['parent\_id'],

'rgt' => $node['\_rgt'],

'lft' => $node['\_lft']

];

}

return $map;

}

/\*\*

\*

\* @param type $a

\* @param type $b

\* @return int

\*/

public function compare($a,$b){

if($a['lft'] == $b['lft']){return 0;}

return ($a['lft'] < $b['lft']) ? -1 : 1;

}

}