

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

|  |  |
| --- | --- |
| **Document title** | **SMS-Email** |
| **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

# Unisa SMS-Email component is an easy to use component using the php stencil it adds the capability to send SMS/MMS and email messages to mobile phones from the web. It currently supports a free way to send SMS messages through E-Mail gateways provided by the wireless carriers.The SMS-Email component also supports 9 paid services, Call Fire, EZTexting, Flowroute, LabsMobile, Mozeo, Nexmo, Plivo, Twilio and Zenvia amongst the sms services. The Emails utilises laravel drivers for SMTP, Mailgun, Mandrill, Amazon SES, PHP's mail function, and sendmail, allowing you to quickly get started sending mail through a local or cloud based service of your choice.

# How to Install the SMS-Email Component

# Clone the component from the github repo and put the project in your htdocs directory so that it it accessible from the host machine.

1. Open the terminal.
2. Go to the directory where the project is using cd.
3. Enter the command composer install.
4. Type copy .env.example .env if using cmd.
5. Type php artisan key:generate.
6. Open your .env file and change the database name to whatever you have, username to YOUR\_USER\_NAME and the password field to YOUR\_PASSWORD.
7. Enter the command php artisan migrate.
8. Enter command php artisan serve.
9. run the project on the browser DONE

## Configuration Files

All of the configuration files for the SMS- Email App are stored in the config directory. Each option is documented, so feel free to look through the files and get familiar with the options available to you.

## Directory Permissions

After installing the SMS-Email App you may need to configure some permissions. Directories within the storage and the bootstrap/cache directories should be writable by your web server or The component will not run. If you are using the [Homestead](https://laravel.com/docs/5.4/homestead) virtual machine, these permissions should already be set.

## Application Key

The next thing you should do after installing the Component is set your application key to a random string. You can archive this by the following command within the directory of the application php artisan key:generate command.

Typically, this string should be 32 characters long. The key can be set in the .env environment file. If you have not renamed the .env.example file to .env, you should do that now. **If the application key is not set, your user sessions and other encrypted data will not be secure!**

# How to use SMS- Email Component

## Sending Mail

The component allows you to store your e-mail messages in [views](https://laravel.com/docs/5.1/views). For example, to organize your e-mails, you could create an emails directory within your resources/views directory:

To send a message, use the send method on the Mail [facade](https://laravel.com/docs/5.1/facades). The send method accepts three arguments. First, the name of a [view](https://laravel.com/docs/5.1/views) that contains the e-mail message. Secondly, an array of data you wish to pass to the view. Lastly, a Closure callback which receives a message instance, allowing you to customize the recipients, subject, and other aspects of the mail message:

Below is the code snippet that actually sends the email.

public function sendmail(MailFormValidationRequest $request)

{

if($request->has('message') && ($request->has('email')))

{

$body=['message'=>$request->message];

Mail::send('mail.newemail', ['body'=>$body] , function($message) use ($request)

{

$message->to($request->email)->from('b.mmuffy@gmail.com','UNISA MAIL')->subject($request->subject);

});

Session::flash('flashmessage', 'Email sent successfuly.');

return redirect('mail');

}

}

## Basic SMS sending

Simple SMS operates in much of the same way as the Mail service. If you are familiar with this then SMS should feel like home. The most basic way to send a SMS is to use the following:

public function sendsms (MailFormValidationRequest $request)

{

If ($request->has ('text') && ($request->has('to')))

{

$to = $request->to;

$from = $request->from;

$text = $request->text;

$nexmo = app ('Nexmo\Client');

$nexmo->message()->send([

'to' =>$to,

'from' =>$from,

'text' =>$text

// return view ('welcome');

]);

Session::flash ('flashmessage', 'Sms sent successfuly.');

return redirect('sms');

}

}

In the config/nexmo directory, If you're using API credentials, change these settings. Get your credentials from https://dashboard.nexmo.com | 'Settings'.

API Credentials

'api\_key' => '7d57\*\*\*\*\*\*',

'api\_secret' => 'eb5\*\*\*\*\*\*\*\*\*\*\*68a',