National College of Ireland
BSc in Computing
2017/2018

Deniss Strods
X14100398@student.ncirl.ie

CMS Lite Web Hosting Platform
Technical Report

Project Complete in Partial Fulfilment of the
BSc (Hons) in Computing.
Supervised by DR. Keith Maycook

National College of Ireland

Contents

Executive Summary

We have developed software solution CMS – Lite Web Hosting Platform that provides facility to build and host private web sites on the internet.

Platform allows to simplify website development and hosting for business owners that want to build fast, dynamic and beautiful websites themselves.

System consists of three main software components, Platform that facilitates all user applications, user container API Server and CMS Web Application (this part will be multiplied by number of users and the fact that one user can have even more than one app).

We have constructed web applications by utilizing power of Angular 2 for the front end, Node.js for the back end and Docker technology to separate instances of the user applications from one another. Nginx reversed proxy is used to redirect all the traffic to the appropriate user container instances.

Platform component of the system can facilitate any web application that can be docker containerized. To demonstrate this, 3rd party CMS was added to our project in order for the users to have more flexibility when choosing web builder solution.

1 Introduction

1.1 Solution

We have developed PaaS - hosting platform where end user can host and build web sites. Hosting Platform is place holder of user’s applications that are based on our own CMS web application, but not limited to it as 3rd party CMS is available as well.

Platform allows to simplify website development and hosting for professionals that want to build fast, dynamic and beautiful website themselves. As target user base we can consider GP’s, Mechanics, Barbers and theoretically any small business owners that have limited knowledge about web page development and deployment.

Users can register on Platform Portal and acquire their own subdomain name. Application instance will be launched under that subdomain. User will customize their own web page application instance non-complicated and unpainful way by using intuitive in page interface, that will be complemented by simple guidelines trough interactive demo. Users are able to develop a *fast, dynamic one-page application* that allow high customization level with various skins and style themes and highly customizable title page. We are using WYSWYG editing approach and combining it with the smart templating system ensures that even novice users are able to use it for their needs most efficiently. CMS component of the system has *dynamic code base* that would change based on the user content addition and configuration. Dynamic code helps to achieve faster performance of the running web sites and allows to assemble front end part of the application from prewritten modules that simplifies development and are tweaked by user configuration input.

In order to extend user options, it is possible to select what CMS to use when acquiring subdomain name by selecting desired application type. User can choose between CMS Lite (developed by us) or Grav CMS that is 3rd party Application.

1.2 Background

After performing preliminary research of the market, it is clear that Internet is already saturated with different kind of hosting platforms and niche in the market seems to be already taken by some well-known Platforms like Bluehost.com or Wix.com or squarespace.com. But after closer examination it was discovered that these solutions are far from perfect and there is an opportunity to develop software solution better and more efficient than currently the ones available on the market.

Bluehost.com allows its users to host WordPress applications on their platform with some minimal control of the running application entity. WordPress is well known CMS with huge community, and they are developing extra functionality for the platform daily, but this CMS is not really designed for average users, we even can consider that WordPress is more developer oriented CMS. As well as WordPress is written in PHP, it is resource demanding and slow, user heavy websites are quite demanding on resources. (wordpress.com, 2017)

Wix.com allows its user to host their own CMS app that has nice drag and drop capabilities. Problem with this solution is that it is not very flexible and is quite expensive if we are talking about nice customized web pages, if website becomes user heavy this solution does not scale well. (wix.com, 2017)

Squarespace.com similarly to wix.com are providing their own CMS that has drag and drop interface, but the problem with this one is that all web pages developed in squarespace.com look exactly the same way, the only distinction I could find was the content of the pages. Styling was the same, title pages were look alike e.c.t. (squarespace.com, 2017)

Our solution is average user oriented and allows even novice internet users to build and deploy desired web sites with high degree of the customization. As Platform and web application is written using node.js and Angular.js, applications are very efficient and fast and are not requiring as much resources as WordPress application typically would from the server. This allows reduction of expensiveness of each instance and allows to host more instances on one VM than we could with WordPress.

1.3 Technologies

Docker Daemon is installed on top of VM, this piece of software allows us to create standard images for the CMS web application and then run these applications in separate isolated containers on top of the same VM. Container is created and assigned to subdomain name, this functionality is achieved by environment variables configuration for the docker container. One of the nicest benefits of using Docker is that if needed it will be possible to scale this System horizontally by just adding extra VM’s, and transferring containers to other VM.

Google could is used as IaaS provider, they provide all necessary facilities for the project and are providing 250Eur credit for their year trail. 250Euro budget for cloud infrastructure was sufficient for the development of the System and it fits all the requirements of the project.

Ubuntu 16.0 Linux is used as the operating system for the VM’s, this seems to be a good pick from Linux flavors as is not hardest to use and is easy to maintain.

Server-side programming is conducted by means of Node.js (powered by JavaScript ES5 and chrome JS engine V8) and some additional bash scripting. Node.js is used as it provides good, maintainable way to write a code and has a lot of modules that make development less time consuming. For instance, these modules are used:

- For rapid server-side coding express Node.js module.
- Passport Node.js module will be utilized for authentication and authorization.
- Mongoose module will be used to define Mongo Database schemas

As database management system, MongoDB is used. Mongoose Node.js module is providing abstraction when programming for database layer, and reduces complexity of the code.

Client-side programming is done by utilizing power of Angular 2 framework that uses TypeScript as the main programming language (subset of JavaScript and has similar features as ES6). This way our web applications is component based follows MVC architecture. For smoother frontend JS programming jQuery library is used as well, as it is mandatory component when using bootstrap. For the look and feel of the application we are using bootstrap 4 Alpha CSS framework and mertro.css. Additional styling and animation are coded by utilizing animate.css library. For the editing page part of the application medium.js is utilized, this is editor module for the front-end text editing and text editable content change.

Comments sections for blogs are utilizing SaaS DISQUS that provides facility to add dynamic and manageable commenting sections into our page.

For unit testing purposes Karma and Jasmine unit testing framework is utilize and all the tests written for application are following Jasmine and Karma dictated coding rules.

As additional feature 3rd party CMS Grav is offered as alternative to our own built CMS. Grav is modern static file open source CMS that allows easy and interesting way to develop websites as it is static file based CMS that does not require database.

2 System

2.1 Project Overall Scope

The scope of the project was to develop a PaaS system that consists of multiple components. The system consists of a Docker control API component, Platform portal component and CMS Web Application Component.

Docker control API component has direct access to the host VM and Docker Daemon. This component does not have any security verification, just simple google infrastructure firewall security that will ensure that all outer traffic is blocked.

Platform portal component consists of:

- Login / Signup System
- Web GUI interface
- Docker container control mechanisms by utilization of Docker control API component
- User domain / subdomain name control

Platform portal component does not have control over the user web applications content.

CMS Web Application Component has:

- Login / Signup system connected to same account as platform
- Initial application build facility
- Dynamic code base that is generated based on component addition
- WYSWYG editing approach
- Title page templating engine with pre-set multiple templates
- Facility to insert new pages
- Facility to add items to the navbar
- Navbar customization functionality from pre-set styles
- Footer addition and customization functionality
- New page templating engine with pre-set multiple templates
- Facility to upload and insert Images and files into the pages
- Commenting system
- Blogging section with blog addition capabilities

CMS Web Application Component does not have facility to create new templates for the title pages and new user pages.

System is limit to 500 instances of the user applications in order to not exceed resources of VM that it is running on.

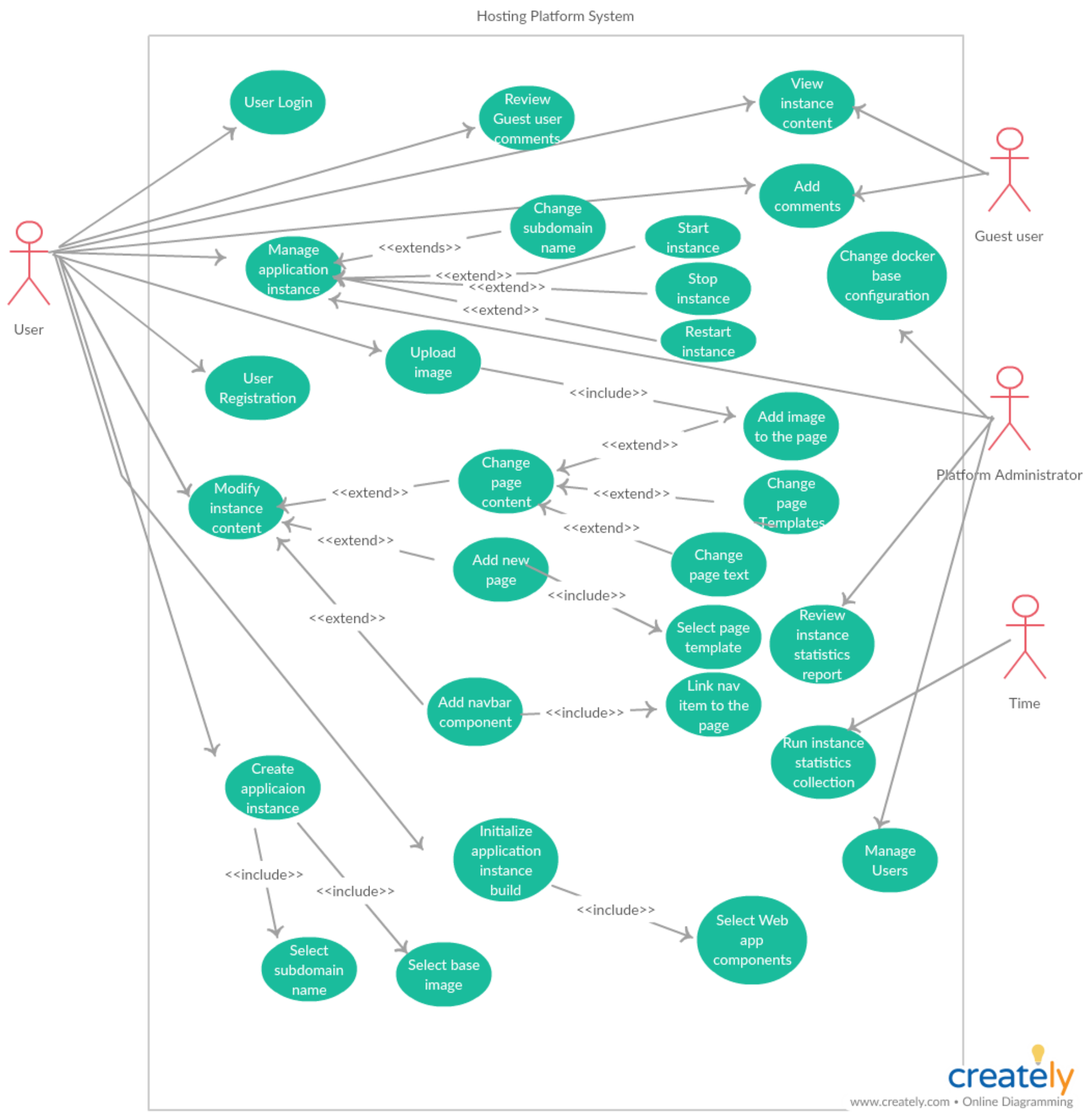
2.2 Requirements

2.2.1 Requirements gathering stage

As the system that we are developing is purely my own initiative, standard elicitation techniques cannot be applied to requirement gathering process as there are only couple or stake holders, and project owner is actually a developer himself(me). At initial stage we have generated some core requirements that we specified below and after we have iterating through them on the development stage in order to implement features in real life.

2.2.2 Functional requirements

2.2.2.1 Use Case Diagram

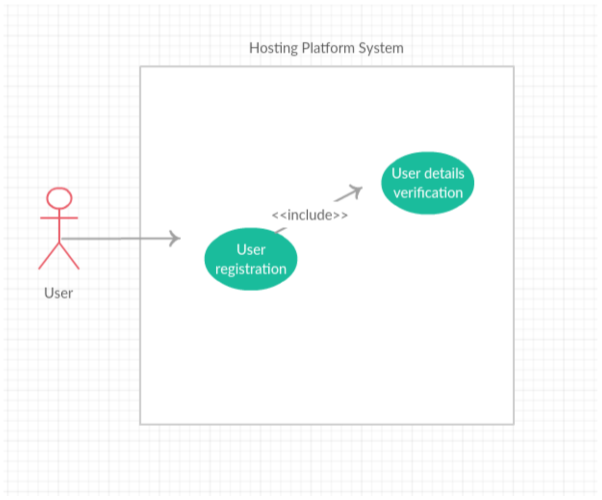


2.2.2.2 Requirement 1 <User Registration>

2.2.2.2.1 Description & Priority	User registers for the service in the main platform portal. High Priority
2.2.2.2.2 Use Case	

Scope	User shall acquire the account on platform
Description	User is asked to enter his registration details
Precondition	• User has valid registration details • User has valid e-mail address • There is no other accounts with the same e-mail
Activation	User selects registration button
Main flow	1. User enters his registration details and hits register button. 2. System creates account for the user
Alternate flow	
Exceptional flow	E1 (Server goes offline): 1. User enters his registration details and hits register button. 2. System goes offline before user finishes registration, error message is returned E2 (User tries to use already used user details): 1. User enters username that already exists, or User enters email that already was used 2. System returns error message with error explained
Post Condition	• The User has subscribed for service and has account created • The user is logged in to the system • The user has access to the functions of the system

Use Case Diagram

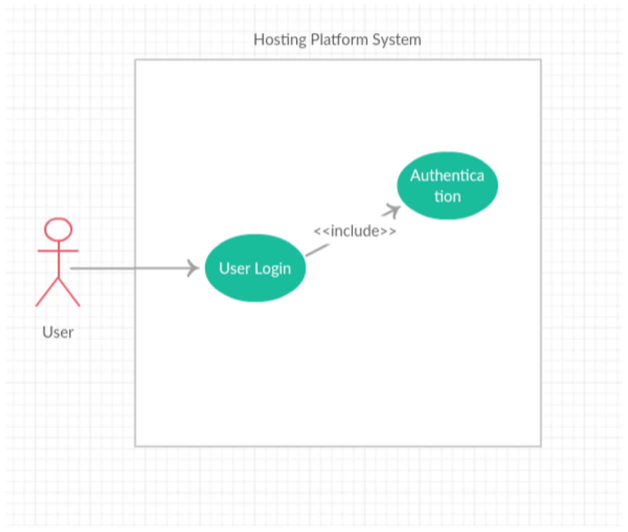


2.2.2.3 Requirement 2 <User Login>

2.2.2.3.1 Description & Priority	User is able to login with his credentials. High Priority
2.2.2.3.2 Use Case	
Scope	Scope of the requirement is for the user to login.
Description	User is asked to enter his login details
Precondition	• User has valid login details • User have account on system
Activation	User selects login button
Main flow	1. User enters his login details and hits login button. 2. System authenticates User
Alternate flow	
Exceptional flow	E1 (Wrong login details entered): 1. User enters wrong username or password

	2. System returns error message with error explained E2 (Server is offline): 1. User enters his login details and hits login button. 2. System goes offline before user finishes registration; error message is returned
Post Condition	• The user is logged in to the system • The user has access to the functions of the system

Use Case Diagram



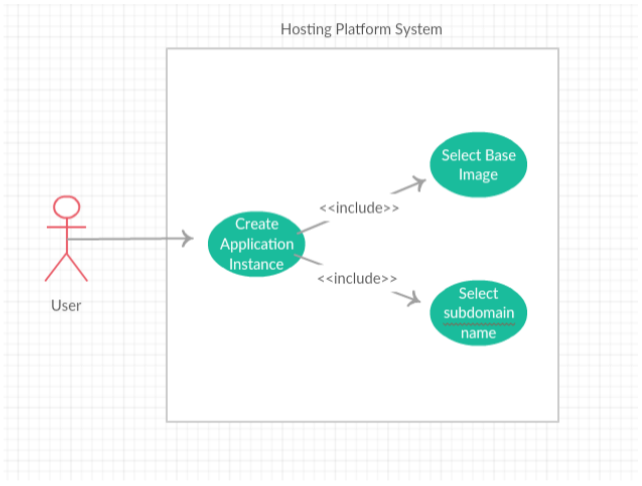
2.2.2.4 Requirement 3 <Acquire application instance>

2.2.2.4.1 Description & Priority	High Priority
2.2.2.4.2 Use Case	
Scope	The scope of this requirement is, for user to acquire CMS application instance by selecting it from existing pool of start images and specifying subdomain.

Description	User acquires application instance on the platform.
Precondition	• User is logged in. • User has not exceeded limit of instances.
Activation	User presses Add Virtual Machine button.
Main flow	1. User clicks Add Virtual Machine button. 2. User selects start image to use. 3. User specifies subdomain name. 4. User presses create button. 5. System creates new application instance 6. System launches application instance under selected subdomain
Alternate flow	A1(Subdomain is taken): 1. User clicks Add Virtual Machine button. 2. User selects start image to use. 3. User specifies already existing subdomain name. 4. User presses create button. 5. System returns error with message. 6. User specifies different subdomain name 7. User presses create button 8. System creates new application instance 9. System launches application instance under selected subdomain

Exceptional flow	E1(Application server goes offline): 1. User clicks Add Virtual Machine button. 2. User selects start image to use. 3. User specifies subdomain name. 4. User presses create button. 5. System returns error with message. E2(VM limit is exceeded): 1. User clicks Add Virtual Machine button with exceeded VM limit. 2. System returns meaningful error.
Post condition	• System creates new application instance • System launches application instance under selected subdomain • Running Web application under subdomain

Use Case Diagram



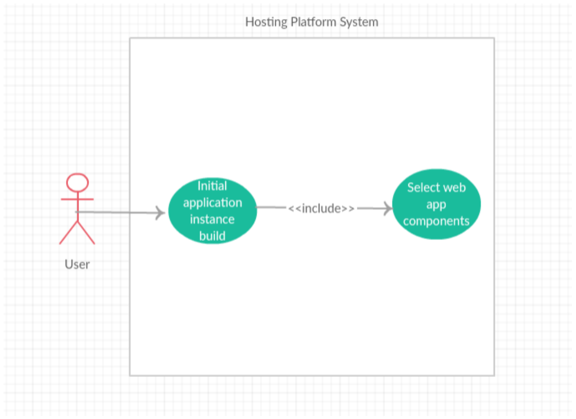
2.2.2.5 Requirement 4 <Initial application instance build>

2.2.2.5.1 Description & Priority	User selects what features to be added to the application. High Priority

2.2.2.5.2 Use Case	
Scope	When logging in first time on to web application instance, user shall select what elements to include in the application before building frontend.
Description	User shall select what elements are required for his web application before initializing and building frontend
Precondition	• Application instance is running • User is logged in to running instance under subdomain. • Application is at initiation stage
Activation	User loges in to the running instance under subdomain.
Main flow	1. User loges in to the running instance under subdomain. 2. User selects what features to include in application. 3. User presses initialize button 4. System builds frontend based on User selection
Alternate flow	A1(User does not select any features): 1. User loges in to the running instance under subdomain. 2. User skips feature selection stage 3. User presses initialize button 4. System builds frontend based on Default feature set
Exceptional flow	E1(Application server goes offline): 1. User loges in to the running instance under subdomain.

	2. User selects what features to include in application. 3. User presses initialize button 4. System returns error with message.
Post condition	• Frontend is configured and ready to be edited • System builds frontend based on feature set

Use Case Diagram

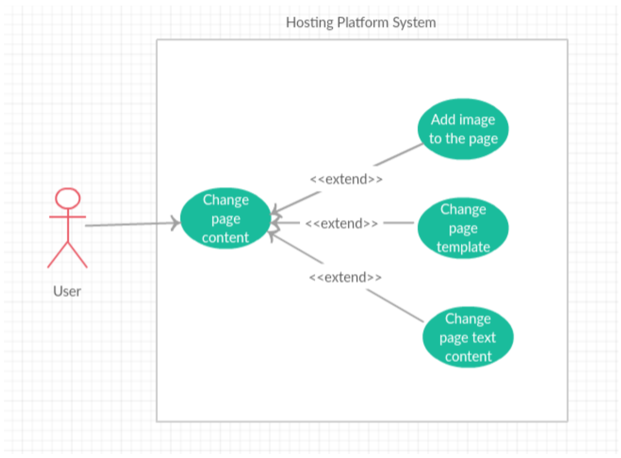


2.2.2.6 Requirement 5 <Modification of the page content WYSWYG>

2.2.2.6.1 Description & Priority	Simple in page text modification. High Priority
2.2.2.6.2 Use Case	
Scope	User can modify in page content, everything is modifiable except navbar itself
Description	User is modifying content
Precondition	• Application instance is running • User is logged in to running instance under subdomain. • Application is post initiation stage

Activation	User presses global edit button under the menu bar
Main flow	1. User loges in to the running instance under subdomain. 2. User clicks on the text he wants to modify 3. User types in the new text content 4. User presses save button under the menu bar
Alternate flow	A1(User tries to switch the page before he saves content): 1. User loges in to the running instance under subdomain. 2. User clicks on the text he wants to modify 3. User types in the new text content 4. User tries to switch the page without saving 5. System prompts if changes should be saved before switching page 6. User answers affirmative to the prompt
Exceptional flow	E1(Application server goes offline): 1. User loges in to the running instance under subdomain. 2. User modifies content 3. User save button 4. System returns error with message.
Post condition	• Web application has different text content

Use Case Diagram

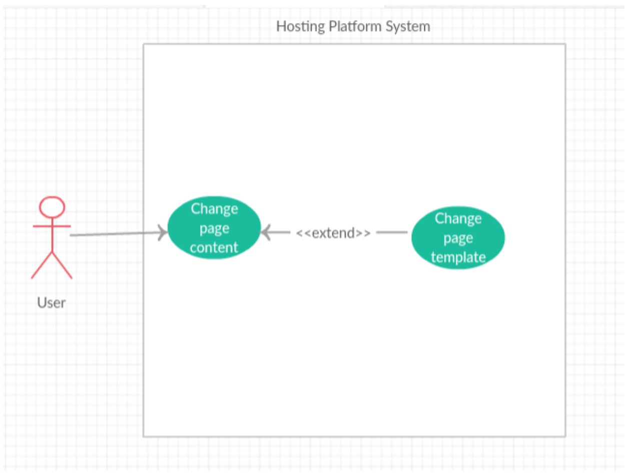


2.2.2.7 Requirement 6 <Selection of title page template>

2.2.2.7.1 Description & Priority	This feature will allow to add uniqueness to the user website. High Priority
2.2.2.7.2 Use Case	
Scope	User can select style of the title page based on the multiple templates provided
Description	Selecting what template to use for title page
Precondition	• Application instance is running • User is logged in to running instance under subdomain. • Application is post initiation stage
Activation	User clicks change template button under title page
Main flow	1. User loges in to the running instance under subdomain. 2. User clicks change template button under title page 3. User selects template

	4. System switches template based on User selection
Alternate flow	
Exceptional flow	E1(Application server goes offline): 1. User loges in to the running instance under subdomain. 2. User clicks change template button under title page 3. User selects template 4. System switches template based on User selection System returns error with message.
Post condition	• Web application has different template selected

Use Case Diagram

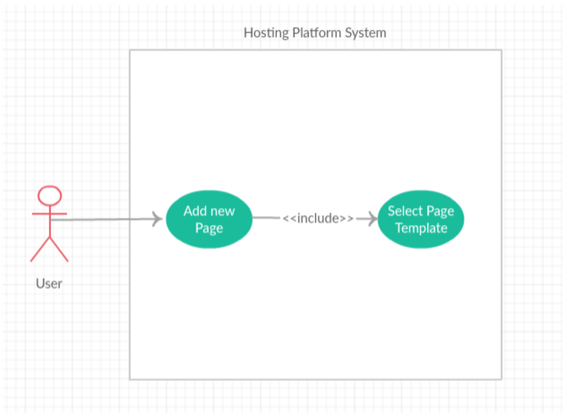


2.2.2.8 Requirement 7 <Adding new page based on template>

2.2.2.8.1 Description & Priority	Core requirement for any CMS System. High Priority
2.2.2.8.2 Use Case	
Scope	User creates new pages based on the templates provided by system

Description	Creating new page based on template
Precondition	• Application instance is running • User is logged in to running instance under subdomain. • Application is post initiation stage
Activation	User presses add new page button
Main flow	1. User loges on the running instance under subdomain. 2. User presses add new page button 3. User selects template 4. System creates new page based on User selection
Alternate flow	
Exceptional flow	E1(Application server goes offline): 1. User loges on the running instance under subdomain. 2. User presses add new page button 3. User selects template 4. System returns error with message.
Post condition	Web application has new page added based on selected template

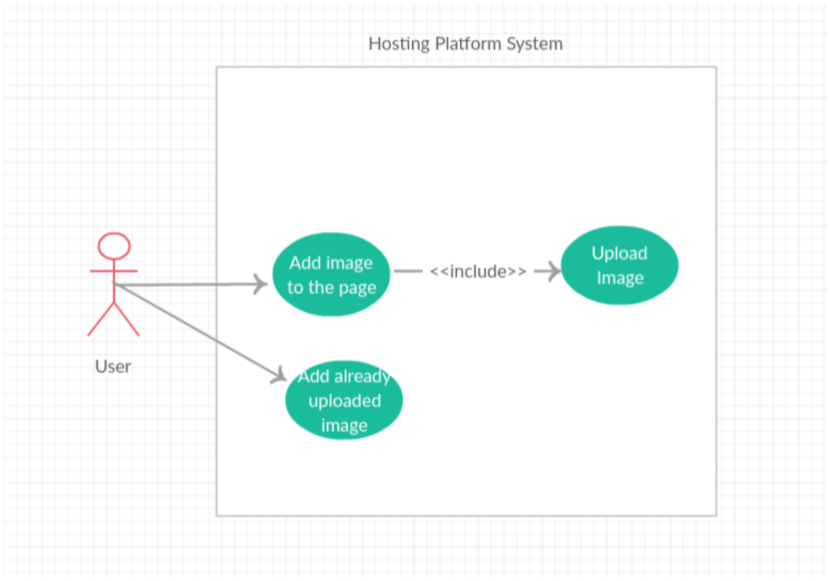
Use Case Diagram



2.2.2.9 Requirement 8 <Upload and add images to the page>

2.2.2.9.1 Description & Priority	Supporting image insertion. High Priority
2.2.2.9.2 Use Case	
Scope	User adds images to the page
Description	Uploading and adding images
Precondition	• Application instance is running • User is logged in to running instance under subdomain. • Application is post initiation stage
Activation	User presses add new image button
Main flow	1. User presses add image button 2. User selects image from the list 3. System adds image to the page
Alternate flow	A1(Adding image with upload): 1. User presses add image button 2. User uploads image from file system 3. System adds image to the list 4. User selects image from the list 5. System adds image to the page
Exceptional flow	E1(Application server goes offline): 1. User presses add image button 2. User selects image from the list 3. System adds image to the page 4. System returns error with message.
Post condition	Image is inserted into the page

Use Case Diagram

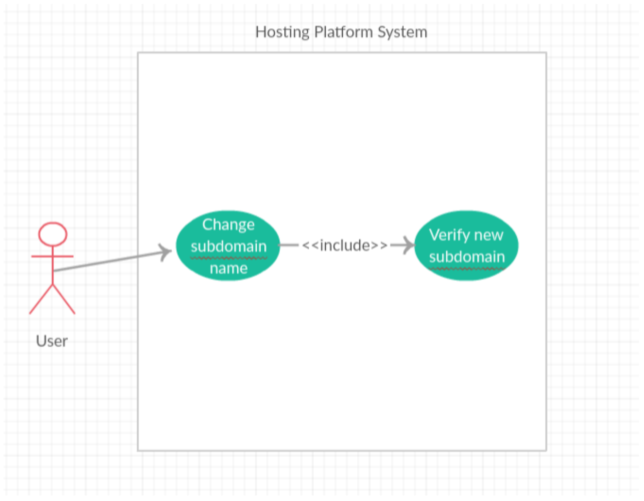


2.2.2.10 Requirement 9 <Change subdomain name>

2.2.2.10.1 Description & Priority	In case if user decides to change his domain, medium priority
2.2.2.10.2 Use Case	
Scope	User changes subdomain name
Description	Changing subdomain name
Precondition	• User is logged in to the platform portal • User has application instances • Selected Application instance is stopped
Activation	User presses change subdomain button
Main flow	1. User goes to the instance properties section 2. User selects required instance 3. User presses change subdomain button 4. User types new subdomain name

	5. User presses save button 6. System changes instance subdomain name
Alternate flow	A1 (User selects running instance): 1. User goes to the instance properties section 2. User selects required instance 3. User changes subdomain name 4. User presses save button 5. System prompts if it is ok to stop the instance 6. User responds ok 7. System changes instance subdomain name
Exceptional flow	E1(Application server goes offline): 1. User goes to the instance properties section 2. User selects required instance 3. User presses change subdomain button 4. User types new subdomain name 5. User presses save button 6. System returns error with message E2 (Subdomain is already in use): 1. User goes to the instance properties section 2. User selects required instance 3. User presses change subdomain button 4. User types new subdomain name 5. User presses save button 6. System returns error with message
Post condition	Domain name of the instance is changed.

Use Case Diagram

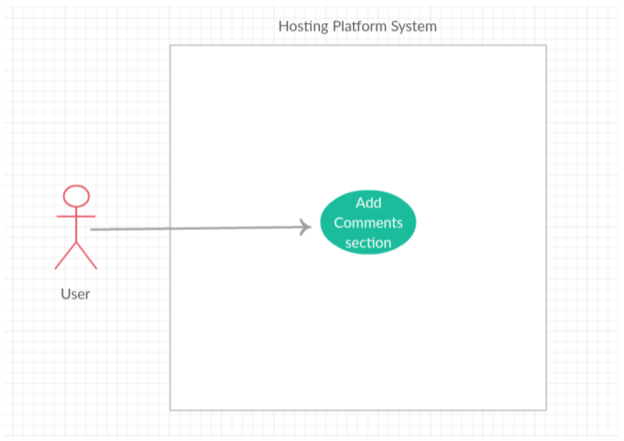


2.2.2.11 Requirement 10 <Add comment section>

2.2.2.11.1 Description & Priority	Enable Guest user feedback, Medium priority
2.2.2.11.2 Use Case	
Scope	User adds comment section to one of the pages
Description	Adding comment section
Precondition	• Application instance is running • User is logged in to running instance under subdomain. • Application is post initiation stage • User is editing content not on title page
Activation	User presses add comment section from menu
Main flow	1. User presses add comment section button 2. System adds comment section
Alternate flow	

Exceptional flow	E1(Application server goes offline): 1. User presses add comment section button 2. System returns error message
Post condition	Comment section is added to the page

Use Case Diagram

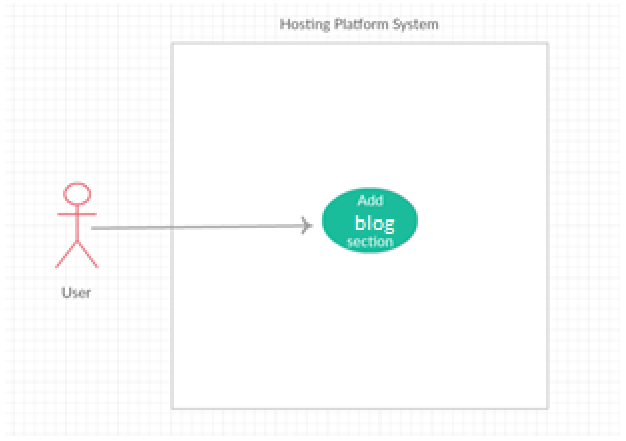


2.2.2.12 Requirement 11 <Add Blog section>

2.2.2.12.1 Description & Priority	Enable user to add blogs to this section, High priority
2.2.2.12.2 Use Case	
Scope	User adds blogging section to the web page
Description	Adding blogging section
Precondition	• Application instance is running • User is logged in to running instance under subdomain. • Application is post initiation stage • User is editing content
Activation	User presses add blog section from menu

Main flow	3. User presses add blog section button 4. System adds blog section
Alternate flow	
Exceptional flow	E1(Application server goes offline): 3. User presses add blog section button 4. System returns error message
Post condition	Blog section is added to the page

Use Case Diagram

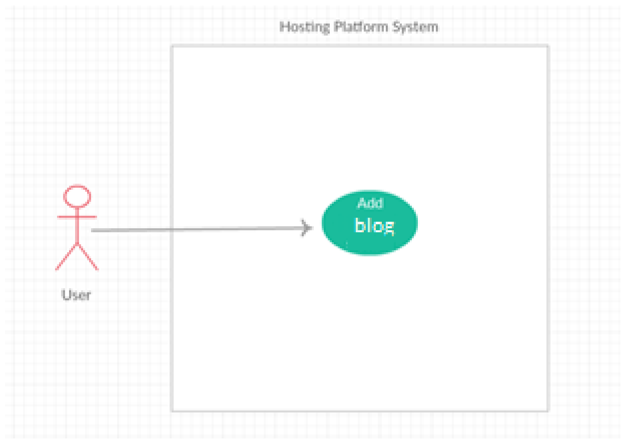


2.2.2.13 Requirement 12 <Add Blog>

2.2.2.13.1 Description & Priority	Add blogs to his blog section, High priority
2.2.2.13.2 Use Case	
Scope	User creates new blog
Description	Creating Blog
Precondition	• Application instance is running • User is logged in to running instance under subdomain. • Application is post initiation stage

	• User is editing content • User has added blogging section
Activation	User presses add blog in blogging section
Main flow	1. User presses add blog button 2. System adds blog
Alternate flow	
Exceptional flow	E1(Application server goes offline): 1. User presses add blog button 2. System returns error message
Post condition	Blog is added to the blog section page

Use Case Diagram

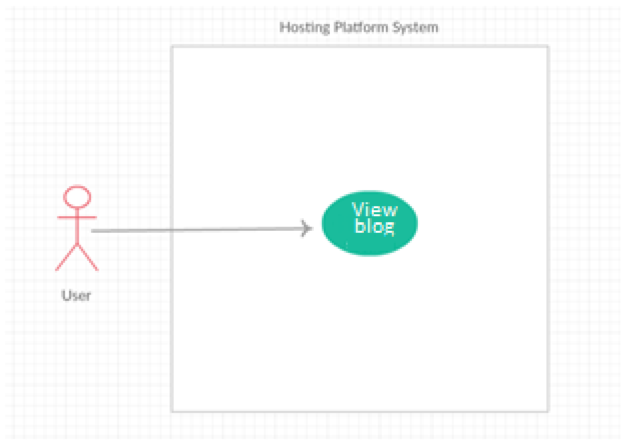


2.2.2.14 Requirement 13 <View Blog>

2.2.2.14.1 Description & Priority	Guest user has ability to view and read blog
2.2.2.14.2 Use Case	
Scope	User / Guest user views blog
Description	View Blog

Precondition	• Application instance is running • User is logged in to running instance under subdomain. • Application is post initiation stage • User has added blog • User has added blogging section
Activation	User presses on desired blog from the list
Main flow	1. User presses view blog button 2. System displays blog
Alternate flow	
Exceptional flow	E1(Application server goes offline): 1. User presses view blog button 2. System returns error message
Post condition	Blog is displayed to the user

Use Case Diagram

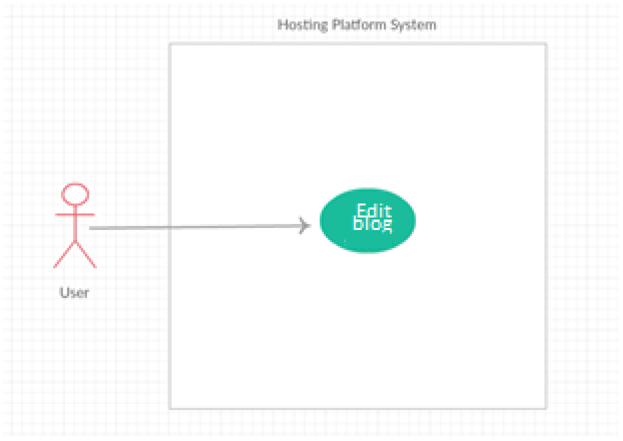


2.2.2.15 Requirement 14 <Update Blog>

2.2.2.15.1 Description & Priority	Update blogs in his blog section, High priority
2.2.2.15.2 Use Case	

Scope	User updates new blog
Description	Updating Blog
Precondition	• Application instance is running • User is logged in to running instance under subdomain. • Application is post initiation stage • User is editing content • User has added blogging section • User has added Blog
Activation	User presses edit blog in blogging section
Main flow	1. User presses edit blog button 2. System enables blog editing
Alternate flow	
Exceptional flow	E1(Application server goes offline): 1. User presses edit blog button 2. System returns error message
Post condition	Blog is edited in the blog section page

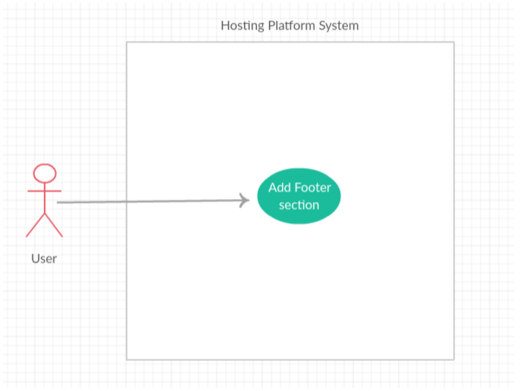
Use Case Diagram



2.2.2.16 Requirement 15 <Add footer section>

2.2.2.16.1 Description & Priority	Adding footer to the selected page. Medium priority
2.2.2.16.2 Use Case	
Scope	User adds footer section to all
Description	Adding footer section
Precondition	• Application instance is running • User is logged in to running instance under subdomain. • Application is post initiation stage • User is editing content not on title page
Activation	User presses add footer section from menu
Main flow	1. User presses add footer section button 2. System adds footer section
Alternate flow	
Exceptional flow	E1(Application server goes offline): 1. User presses add footer section button 2. System returns error
Post condition	Footer section is added to the page

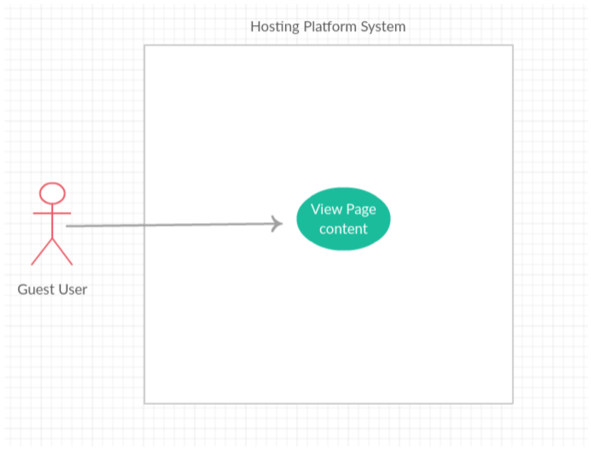
Use Case Diagram



2.2.2.17 Requirement 16 <View Platform User Web page content>

2.2.2.17.1 Description & Priority	Guest users can view the Web sites hosted on the platform. Medium priority
2.2.2.17.2 Use Case	
Scope	Guest User can view web pages hosted on the platform
Description	Viewing page content
Precondition	• Application instance is running
Activation	Guest user broses to subdomain
Main flow	1. Guest user broses to subdomain 2. System returns website under that subdomain
Alternate flow	
Exceptional flow	E1(Application server goes offline): 1. Guest user broses to subdomain 2. System returns error
Post condition	Website is returned upon request

Use Case Diagram

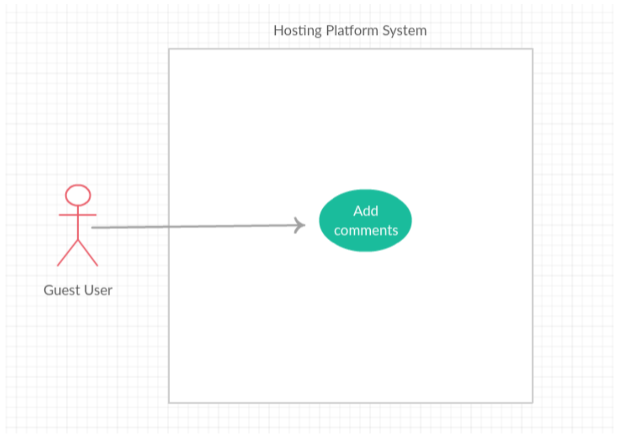


2.2.2.18 Requirement 17 <Add comments>

2.2.2.18.1 Description & Priority	Comments can be left by guest users and owner
2.2.2.18.2 Use Case	
Scope	Guest User adds comment to one of the pages, owner of the website can leave comments
Description	Adding comment
Precondition	• Application instance is running • Comment section exists on the page
Activation	Guest user adds comment
Main flow	1. Guest user adds comment 2. System adds comment to the database 3. System sends review request to the page owner
Alternate flow	1. Owner adds comment 2. System adds comment to the database

	3. System adds comment to comment section
Exceptional flow	E1(Application server goes offline): 1. Guest user adds comment 2. System returns error
Post condition	• Comment is added to the database • Review request is sent

Use Case Diagram

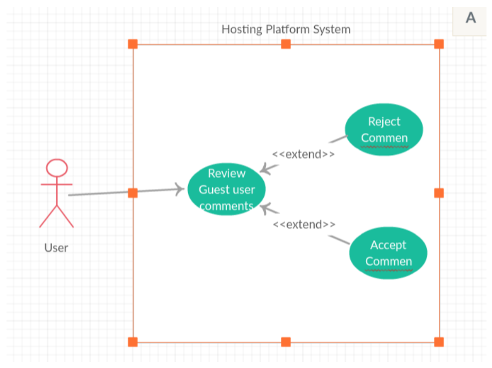


2.2.2.19 Requirement 18 <Review comments>

2.2.2.19.1 Description & Priority	Website owner reviews guest user comments
2.2.2.19.2 Use Case	
Scope	Website owner reviews Guest user comments
Description	Reviewing comment
Precondition	• Application instance is running • Comment section exists on the page • Guest users have published the comments

Activation	User Goes to review comment section
Main flow	1. User goes to review comments section 2. User reviews and selects comments 3. User accepts comments 4. User presses confirm button
Alternate flow	A1 (User rejects the comment): 1. User goes to review comments section 2. User reviews and selects comments 3. User rejects comments 4. User presses confirm button
Exceptional flow	E1(Application server goes offline): 1. User goes to review comments section 2. User reviews and selects comments 3. System returns error
Post condition	• Comments cleared from the review comments section. • Comments added or removed from the page

Use Case Diagram

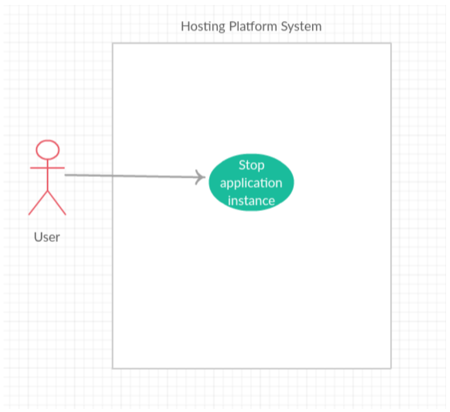


2.2.2.20 Requirement 19 <Stop application instance>

2.2.2.20.1 Description & Priority	User stops running docker instance, Medium Priority
2.2.2.20.2 Use Case	
Scope	The scope of this requirement is, for user to stop running instances
Description	User stops application instance on the platform.
Precondition	• User is logged in. • User has running application instances.
Activation	User selects instance from the list and presses stop button
Main flow	1. User selects instance from the list 2. User presses stop button 3. System stops selected instance
Alternate flow	
Exceptional flow	E1(Application server goes offline): 1. User selects instance from the list 2. User presses stop button 3. System stops selected instance E2(Instance already stopped):

	1. User selects instance from the list 2. User presses stop button 3. System returns error with message.
Post condition	• Selected application instance is stopped

Use Case Diagram

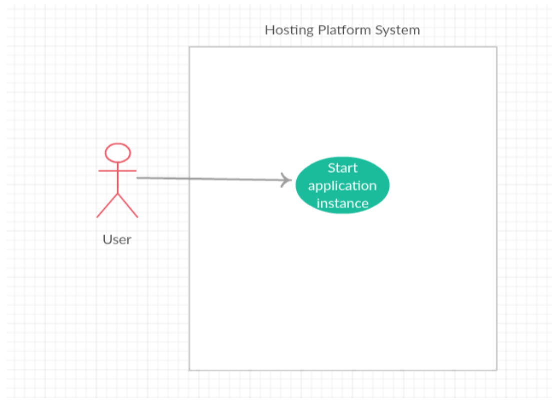


2.2.2.21 Requirement 20 <Start application instance>

2.2.2.21.1 Description & Priority	User starts running docker instance, Medium Priority
2.2.2.21.2 Use Case	
Scope	The scope of this requirement is, for user to start application instances
Description	User starts application instance on the platform.
Precondition	• User is logged in. • User has stopped application instances.
Activation	User selects instance from the list and presses start button
Main flow	1. User selects instance from the list

	2. User presses start button 3. System starts selected instance
Alternate flow	
Exceptional flow	E1(Application server goes offline): 1. User selects instance from the list 2. User presses start button 3. System returns error with message E2(Instance already stopped): 1. User selects instance from the list 2. User presses stop button 3. System returns error with message.
Post condition	• Selected application instance is started

Use Case Diagram

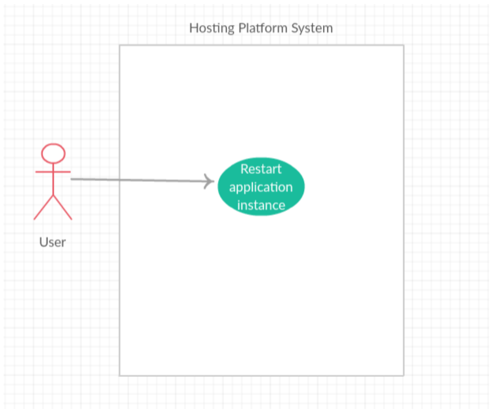


2.2.2.22 Requirement 21 <Restart application instance>

2.2.2.22.1 Description & Priority	User restarts running docker instance, Medium Priority
2.2.2.22.2 Use Case	
Scope	The scope of this requirement is, for user to restart application instances

Description	User restart application instance on the platform.
Precondition	• User is logged in. • User has running application instances.
Activation	User selects instance from the list and presses restart button
Main flow	1. User selects instance from the list 2. User presses restart button 3. System restart selected instance
Alternate flow	
Exceptional flow	E1(Application server goes offline): 1. User selects instance from the list 2. User presses restart button 3. System returns error with message E2(Instance already stopped): 1. User selects instance from the list 2. User presses restart button 3. System returns error with message.
Post condition	• Selected application instance is restarted

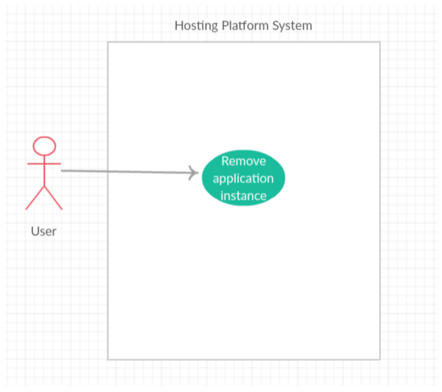
Use Case Diagram



2.2.2.23 Requirement 22 <Remove application instance>

2.2.2.23.1 Description & Priority	User removes docker instance, Medium Priority
2.2.2.23.2 Use Case	
Scope	The scope of this requirement is, for user to remove application instances
Description	User removes application instance on the platform.
Precondition	• User is logged in. • User has application instances.
Activation	User selects instance from the list and presses remove button
Main flow	1. User goes to manage instance section 2. User selects instance from the list 3. User presses remove button 4. System removes selected instance
Alternate flow	
Exceptional flow	E1(Application server goes offline): 1. User goes to manage instance section 2. User selects instance from the list 3. User presses remove button 4. System removes selected instance System returns error with message
Post condition	• Selected application instance is removed

Use Case Diagram



2.2.2.24 Requirement 23 <Dynamic navbar section>

2.2.2.24.1 Description & Priority	Navbar changes based on the user interaction with CMS
2.2.2.24.2 Use Case	
Scope	Navbar changes based on user interaction with system automatically
Description	Navbar changes
Precondition	• Application instance is running • User is logged in to running instance under subdomain. • Application is post initiation stage • User is editing content not on title page • User is adding or removing items from pages
Activation	User is adding or removing items from pages
Main flow	• User is adding or removing items from pages • Navbar model changes
Alternate flow	
Exceptional flow	E1(Application server goes offline):

	1. User presses add footer section button 2. System returns error
Post condition	Navbar model have changed

Please note that low priority requirements that were related to commenting section were dropped in favour of core requirements.

2.2.3 Non-Functional Requirements

2.2.3.1 Performance/Response time requirement

Website builder solution will have low response rate, all time-consuming operations will be transparent to the user and will be performed in background in asynchronous fashion.

Start and Stop of the instances will be completely transparent to the user and from user's perspective will seem instant.

2.2.3.2 Availability requirement

Up time of the System shall be 99% of the time to ensure that user accessibility would not be interrupted.

2.2.3.3 Recover requirement

User docker containers will be backed up and stored in the container repository, in case of system failure containers can be carried over to the backup instance of the system and restarted. There shall be disaster recovery database that will be copied over from the production environment every week (this will only be available if system will be launched for users in real world).

2.2.3.4 Security requirement

System shall only allow the owner of the application to modify the content of instance. Database will not be reachable outside of google infrastructure network, docker container API shall not be reachable outside of the google infrastructure network.

2.2.3.5 Reliability requirement

Application shall be stable and crush resistant, or in case of the crush shall have mechanism that will restart it automatically.

2.2.3.6 Portability requirement

All web application GUI interfaces shall support mobile device, tablet and pc views and will be optimized to use with various devices.

2.2.3.7 Extendibility requirement

Solution shall be scaled up horizontally in case if need arises by adding VM instances that will be spinned up based on VM image, this will not be automatic process.

2.2.3.8 Reusability requirement

CMS component of the system shall use templates for the user website page build and title page.

2.2.4 User requirements

Application will allow user to create web pages that fits requirements of modern internet users. It will be achieved by utilizing modern frontend frameworks and fast, dynamic backend languages plus unique coding techniques.

System shall be designed for novice users that have limited knowledge regarding development and hosting of the websites. To support these novice users, we will develop in page edit system with utilization of content editable HTML5 elements (what you see, you can edit right away - WYSWYG).

In order to train user on how to use the system, system shall have automated functionality walkthrough that user will be able to familiarize themselves with the system specifics.

After simplistic manipulations, user will have modern looking web site that would fit his requirements.

2.2.5 Environmental requirements

We will deploy system to the Google Infrastructure, that will ensure wide network access for our platform and scaling capabilities if necessary. Initially VM with 4 GB of memory and 2 core CPU will be created. This VM will facilitate our System. Mongo DB will be facilitated on separate VM that will allow to achive higher efficiency.

2.2.6 Application Programming Interfaces (API)

System component that will be responsible for controll of the docker containers will be running directly on the host VM and would provide API interface to control these containers. API will include interface to start, stop, restart and create containers, this shall not be RESTfull API.

Platform portal and CMS components of the system shall be using Angular 2 as a frontend, so backend will have to be developped in form of API to support all frontend functionality, this shall not be RESTfull API.

2.3 System Architecture

2.3.1 High Level System diagram

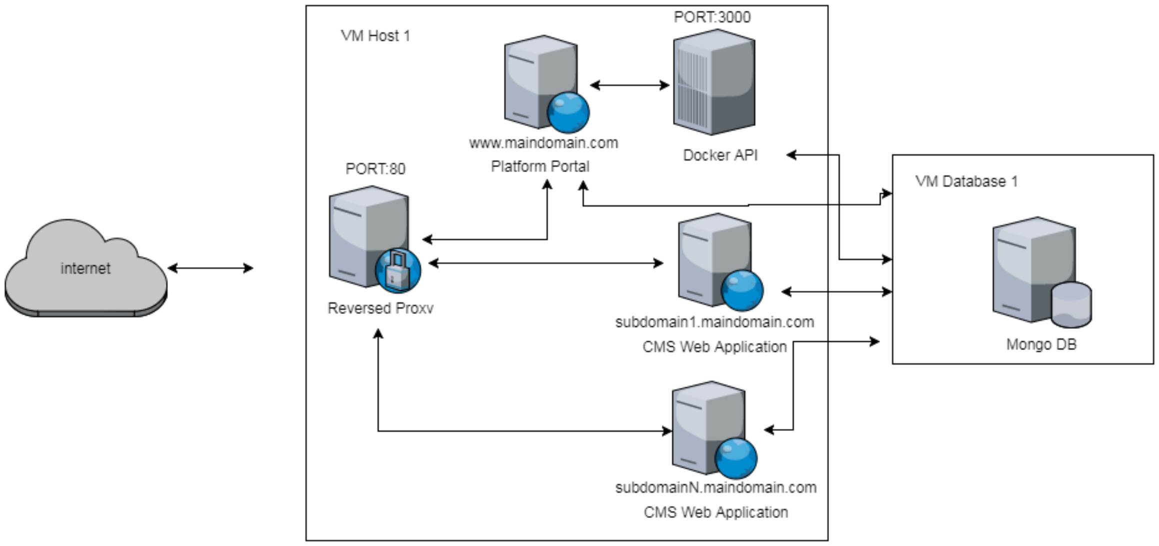


Figure 1, High level system architecture diagram

2.3.2 Class Diagrams

Detailed class diagrams for backend components were generated automatically by utilizing Wavi utility.

2.3.2.1 Docker API Component

High level diagram:

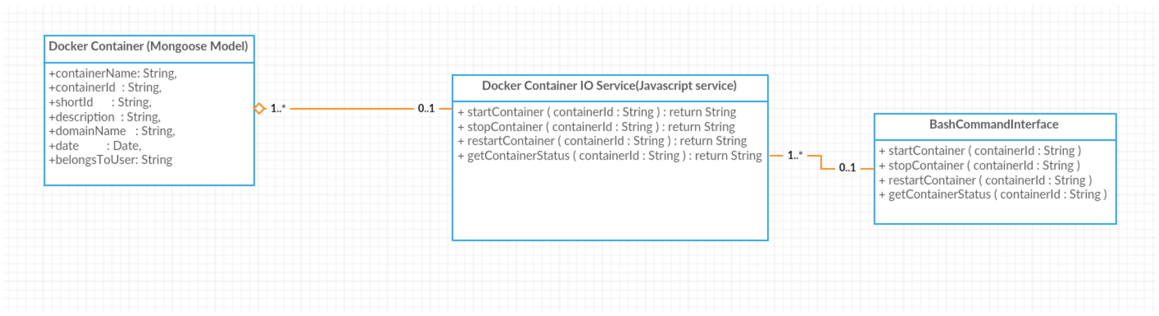

Docker Container (Mongoose Model)
+containerName: String,
+containerId : String,
+shortId : String,
+description : String,
+domainName : String,
+date : Date,
+belongsToUser: String
1..*
0..1
Docker Container IO Service(Javascript service)
+ startContainer (containerId : String) : return String
+ stopContainer (containerId : String) : return String
+ restartContainer (containerId : String) : return String
+ getContainerStatus (containerId : String) : return String
1..*
0..1
BashCommandInterface
+ startContainer (containerId : String)
+ stopContainer (containerId : String)
+ restartContainer (containerId : String)
+ getContainerStatus (containerId : String)

Figure2, High level class diagram Docker API

Full Diagram Source file:

container-server-api.
svg

Full Diagram: (please open the above SVG, as emending diagram had impact on MSWord performance)

2.3.2.2 Platform Portal Component

High Level diagram:

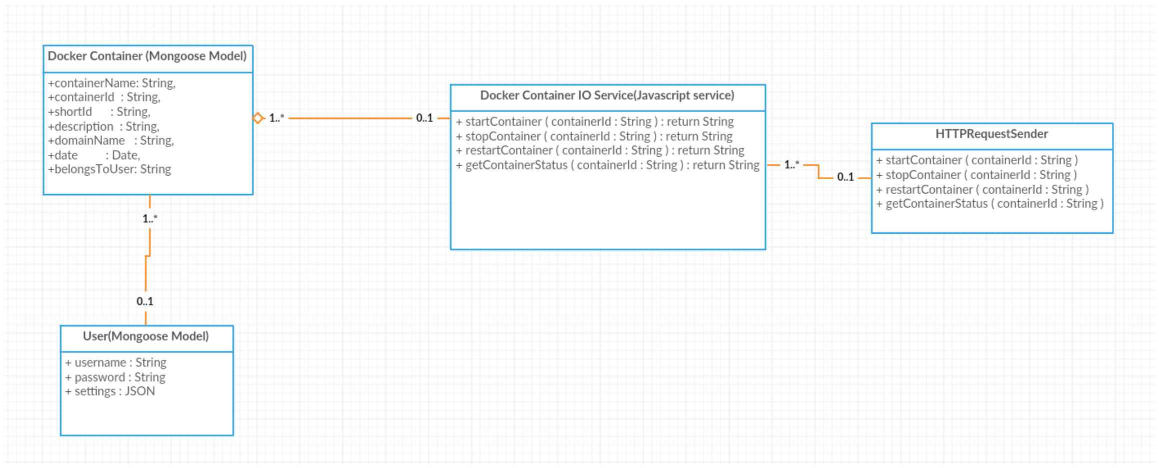

Docker Container (Mongoose Model)
+containerName: String,
+containerId : String,
+shortId : String,
+description : String,
+domainName : String,
+date : Date,
+belongsToUser: String
1..*
0..1
Docker Container IO Service(Javascript service)
+ startContainer (containerId : String) : return String
+ stopContainer (containerId : String) : return String
+ restartContainer (containerId : String) : return String
+ getContainerStatus (containerId : String) : return String
1..*
0..1
HTTPRequestSender
+ startContainer (containerId : String)
+ stopContainer (containerId : String)
+ restartContainer (containerId : String)
+ getContainerStatus (containerId : String)
1..*
0..1
User(Mongoose Model)
+ username : String
+ password : String
+ settings : JSON

Figure 3, High level Platform portal diagram

Full Diagram Source file:

platform-portal.svg

Full Diagram: (please open the above SVG, as emending diagram had impact on MSWord performance)

2.3.2.3 Web CMS Component

High Level Diagram of database models:

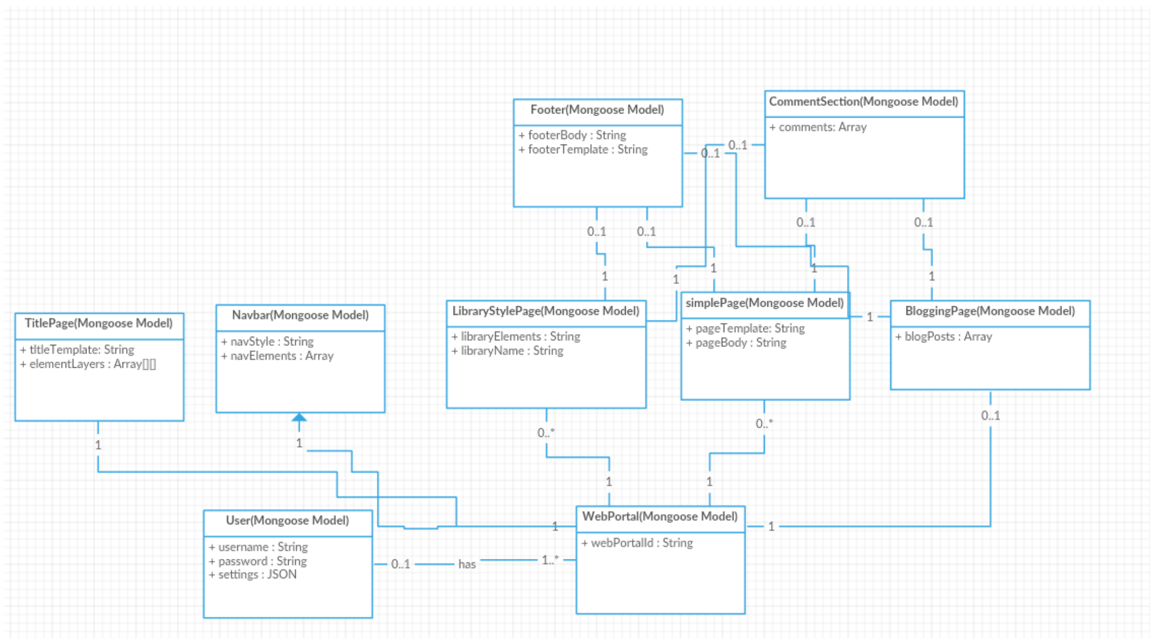


Figure 4, High level CMS Component diagram database side

Full Diagram Source file:

cms-system.svg

Full Diagram: (please open the above SVG, as emending diagram had impact on MSWord performance)

2.4 Implementation

There is currently working application instance running on google cloud under domain http://deniss-strods.com

Please feel free to check it out. I will leave all system components running until 2018.06.30

2.4.1 Cloud Infrastructure description

Implementation was conducted on Google Infrastructure. SSH Connection to c9 IDE was used to utilize all benefits of cloud-based IDE's on development phase.

Infrastructure consists of Google Virtual Private Network that encapsulates VM with all the applications. VPN is configured in a way that only port 80 is exposed to facilitate HTTP/HTTPS protocol communication. Database component of the application is hosted on separate VM instance and is hosting MongoDB, this provides flexibility of scaling database if such need occurs.

2.4.2 Application Component functionality description

Please refer to figure 1 for graphical representation.

This section can be mapped directly to initial requirements, we can see how high lever requirements have become features in the application trough the iterative development and analysis process.

2.4.2.1 Nginex Reverse proxy load balancer

Nginex is crucial component of our system that reroutes traffic based on the domain name in the request headder to appropriate docker containers with running applications. This element was created from jwilder/nginx-proxy Docker container that stores routing tables and reroutes traffic to the container with appropriate environment variable as host name. Container is launched on host VM.

2.4.2.2 Node Docker API Server

Node server was developed to enable docker container control. Server has multiple API Entry points that allow to control docker containers. Server utilizes linux bash scripting language and runs native linux commands on host machine. Appropriate environment variables are initialized in the docker containers and track of all created and running containers is stored in the DB.

GitHub deployment: https://github.com/DenMantm/container-server-api.git

Application accumulative code base combined is 23,489 lines of code.

2.4.2.3 Platform Portal Application

Platform Portal has Node.js backend and angular.js frontend. Node Server has multiple API entry points that allow Angular.js frontend to exchange necessary data with server side. Node Server is communicating with Docker API Server to allow control and management of the docker daemon on the host machine. System component itself is encapsulated into docker container, and traffic to it is rerouted trough Nginx load balancer.

2.4.2.3.1 Acquiring account

To start using our service, user needs to create user account on the platform portal. In order to do that he has to fill the form with required details.

2.4.2.3.2 Managing websites

After acquiring account user can start creation of the website. In order to do this, he would be required to select website subdomain name and Website builder solution type. Then after creating is clicked, Platform portal website makes API call to the docker control API that takes care of the application creation and launch.

Website creation, starting and stopping is facilitated trough menu bar. User can delete their websites if they want to do so.

Subdomain name can be changed trough the separate section in the application. IF user wishes to acquire his own domain name, he would have to contact support team.

GitHub deployment: https://github.com/DenMantm/saas-registration-portal.git

Application accumulative code base combined is 74,421 lines of code.

2.4.2.4 CMS Lite web builder solution

This system component is running node backend and angular.js frontend and was most challenging to implement as it has dynamic JavaScript and HTML codebase.

2.4.2.4.1 Page Text editing functionality and element interaction

Title page and every subsequently added page in the Web CMS System is editable in WYSIWYG (What You See Is What You Get) fashion. After user logs in to the portal and presses edit button, all text in page becomes editable. Text properties can be changed by double clicking the desired text by utilizing medium.js functionality. IF component that is edited consists of multiple blocks, these blocks can be dragged and dropped to change places between the block elements. There is button to add additional block element into the section if user will wish to do so.

After save button is pressed page model is saved to the database.

2.4.2.4.2 Managing elements on page

Layout editor facility allows to manage existing elements and to add different variety of new elements into the page based on pre-set element templates. (Element templates were created from templates provided by https://github.com/BlackrockDigital).

User can select from:

- creative-service
- agency-service
- creative-headder
- agency-headder
- blog-info
- blog-headder
- blog-body
- agency-portfolio
- creative-portfolio
- agency-about
- agency-amazing-team

It is possible to switch element places in layout editor by drag and drop functionality. Elements can be removed from the page by clicking remove item, or theme of the element can be changed to the otherer element theme of the same class.

It is possible to select if user wants element to be visible in navbar or not for simplified navigation to it on the page.

After user clicks save button, node server in background rewrites HTML template code dynamically and changes database model in order to facilitate the change.

2.4.2.4.3 Managing pages and sections

In the page manager, user can add custom pages to the web application. User can specify page name and template from which page should be generated. There is facility to Enable or disable page sections like blogging section if user would wish to do so. It is possible to delete created pages.

When generate page button is clicked, System dynamically generates HTML Template and Dynamically generates TypeScript code for the additional page element. TypeScript is transpiled to JavaScript after by transpiler and new page becomes available for view and editing.

2.4.2.4.4 Changing backgrounds, images and icons

In enable edit mode each page block has button that allows to change block background. Background can be selected from existing images, of new images can be uploaded. As well if user do not wish to use background image it is possible to select just background colour.

Each in page image has button to switch to the different image by invoking image manager. Page icons can be changed to the desired icon from the list by invoking icon manager.

After change and save buttons are pressed, page model information is changed in database.

2.4.2.4.5 Selecting button actions

IF there is a button in the page section, it Is possible to change action of the button trough button manager. User has option to point the button to existing website pages or sections or add custom link.

2.4.2.4.6 Managing and creating blogs

User has ability to create new blog posts with options to add different kind of elements into his blog page, starting with images and finishing with code blocks. Each blog element after can be edited and elements in that blog can be moved around.

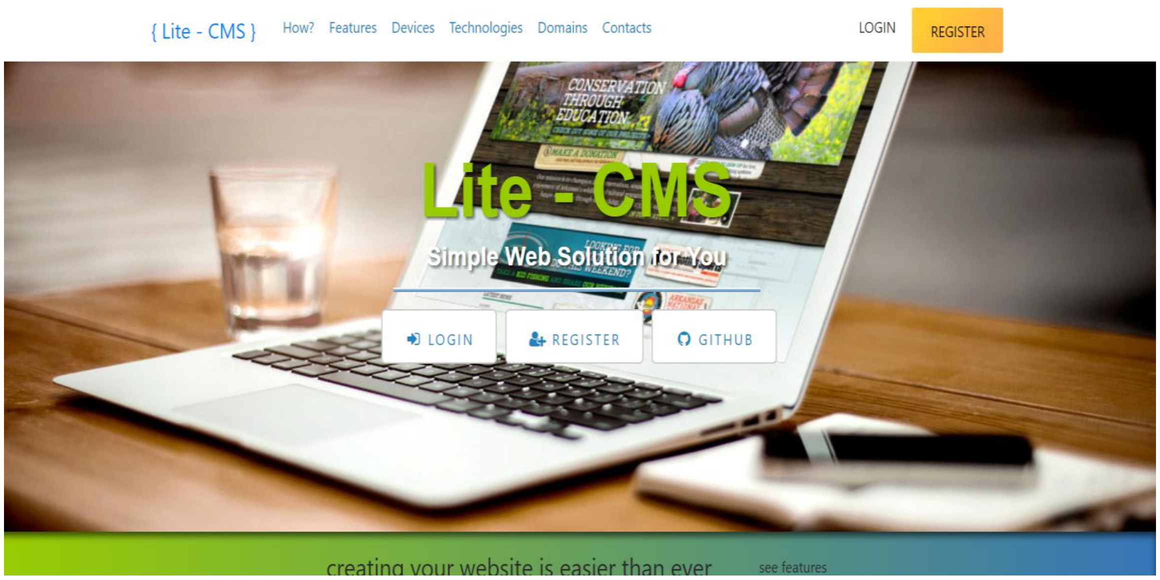
GitHub deployment: https://github.com/DenMantm/nci-project-cms.git

Application accumulative code base combined is 154,401 lines of code

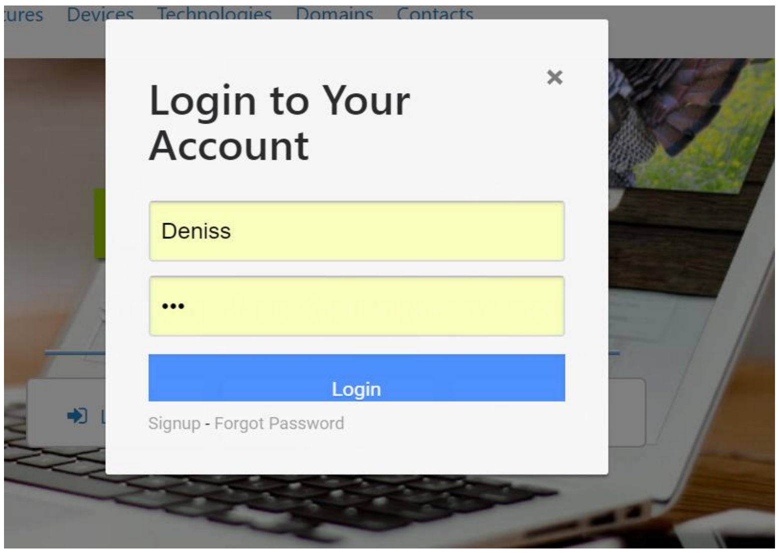
2.4.3 GUI

Please see the implementation of GUI below

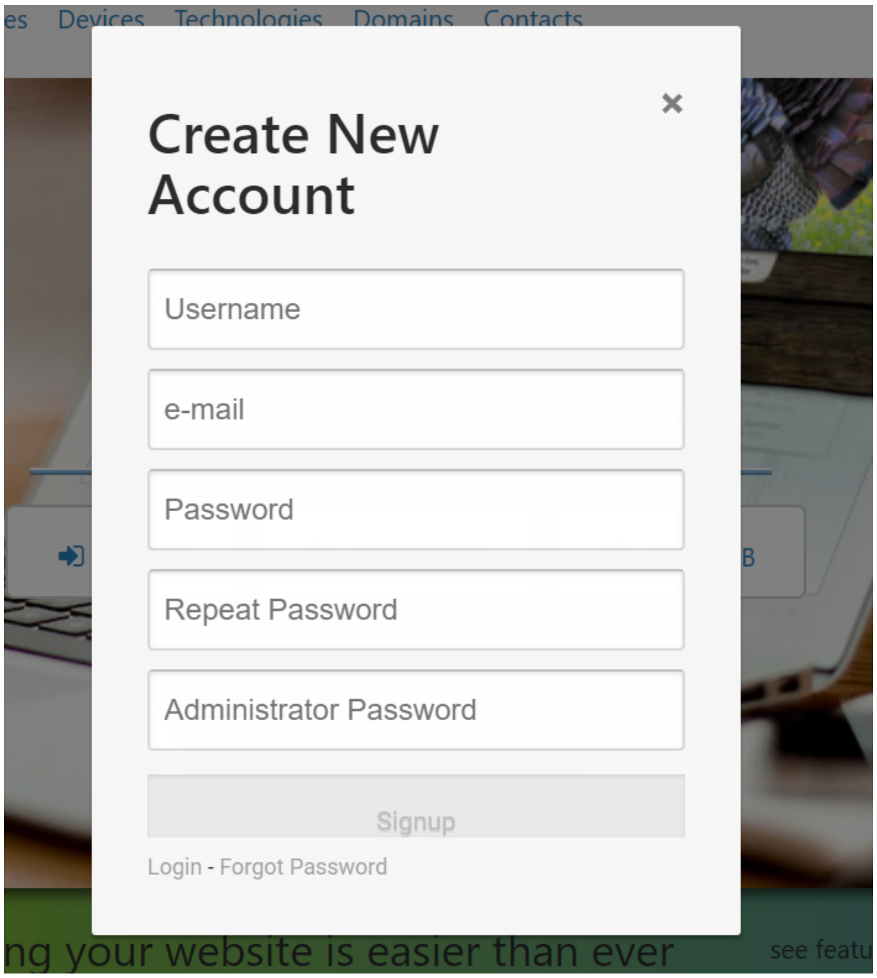
2.4.3.1.1 Platform Portal



2.4.3.1.2 Login



2.4.3.1.3 Signup



Create New Account

Username

e-mail

Password

Repeat Password

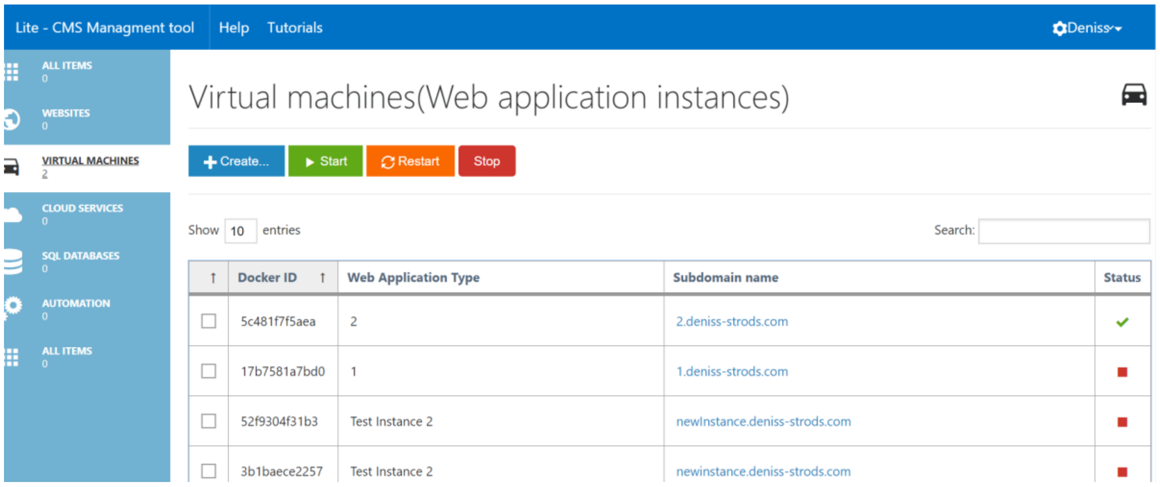
Administrator Password

Signup

Login - Forgot Password

es Devices Technologies Domains Contacts
ng your website is easier than ever
see featu

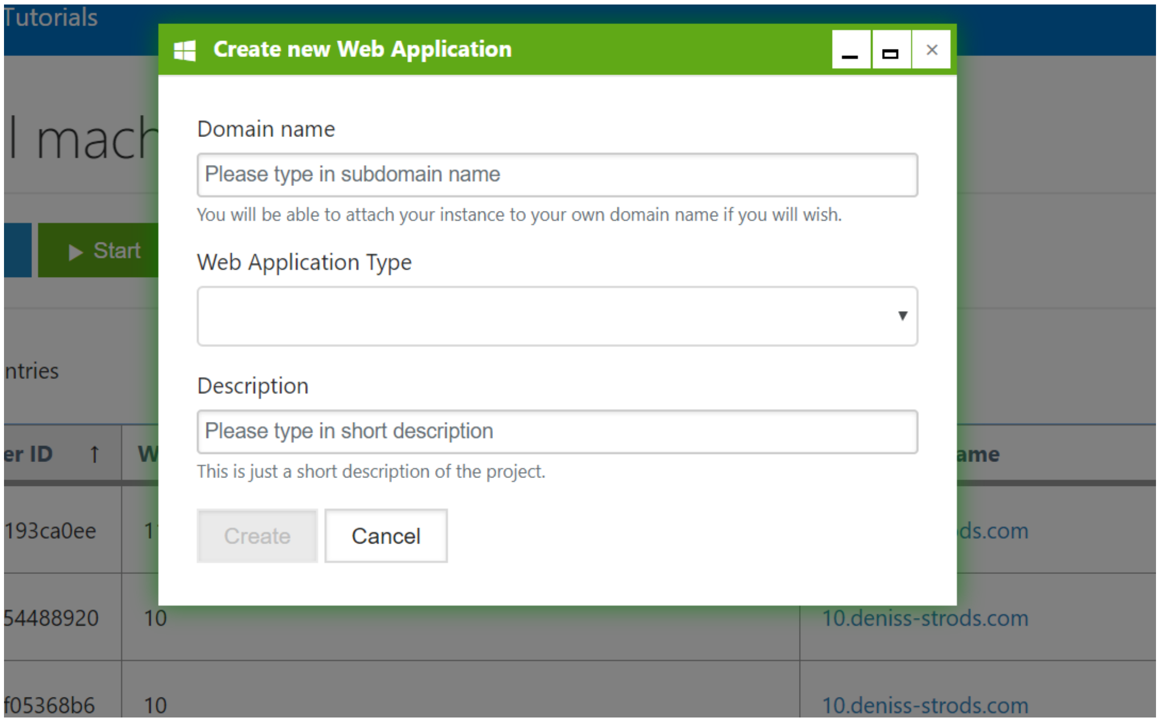
2.4.3.1.4 Control VM instances



Virtual machines(Web application instances)

	Docker ID	Web Application Type	Subdomain name	Status
☐	5c481f7f5aea	2	2.deniss-strods.com	✓
☐	17b7581a7bd0	1	1.deniss-strods.com	🟥
☐	52f9304f31b3	Test Instance 2	newInstance.deniss-strods.com	🟥
☐	3b1baece2257	Test Instance 2	newinstance.deniss-strods.com	🟥

2.4.3.1.5 Create new instance



Create new Web Application

Domain name

Please type in subdomain name

You will be able to attach your instance to your own domain name if you will wish.

Web Application Type

Description

Please type in short description

This is just a short description of the project.

Create | Cancel

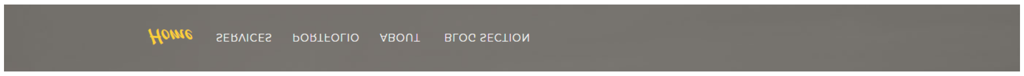
2.4.3.2 CMS Lite Web application

2.4.3.2.1 Initial title page

This is what user will see when he is logged in for the first time

2.4.3.2.1.1 Navbar

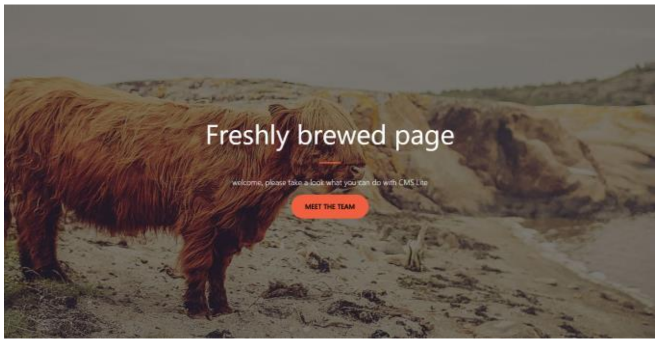
Dynamic navbar that changes based on extra section or page addition



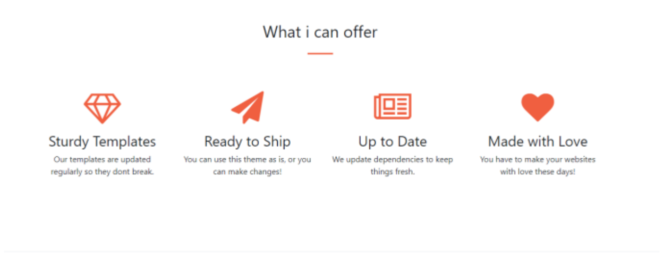
2.4.3.2.1.2 Main page elements

After application initiation, user sees default Web application

Headder



Services



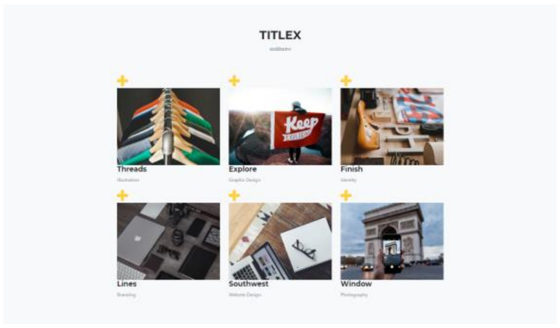
Sturdy Templates
Our templates are updated regularly so they dont break.

Ready to Ship
You can use this theme as is, or you can make changes!

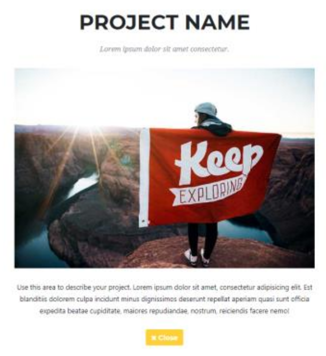
Up to Date
We update dependencies to keep things fresh.

Made with Love
You have to make your websites with love these days!

Portfolio



Portfolio item

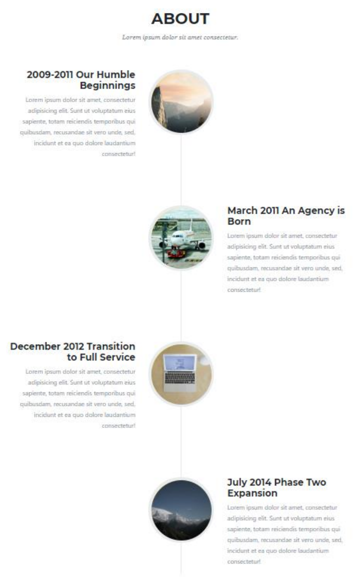


PROJECT NAME

Lorem ipsum dolor sit amet consectetur.

Use this area to describe your project. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Est blanditiis dolorem culpa incidunt minus dignissimos deserunt repellat aperiam quasi sunt officia expedita beatae cupiditate, maiores repudiandae, nostrum, reiciendis facere nemo!

About



ABOUT

Lorem ipsum dolor sit amet consectetur.

2009-2011 Our Humble Beginnings

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sunt ut voluptatum eius sapiente, totam reiciendis temporibus qui quibusdam, recusandae sit vero unde, sed, incidunt et ea quo dolore laudantium consectetur!

March 2011 An Agency is Born

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sunt ut voluptatum eius sapiente, totam reiciendis temporibus qui quibusdam, recusandae sit vero unde, sed, incidunt et ea quo dolore laudantium consectetur!

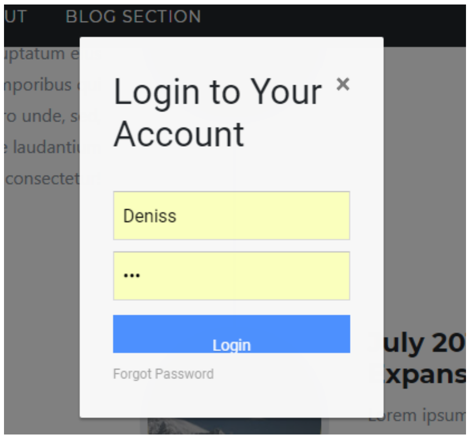
December 2012 Transition to Full Service

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sunt ut voluptatum eius sapiente, totam reiciendis temporibus qui quibusdam, recusandae sit vero unde, sed, incidunt et ea quo dolore laudantium consectetur!

July 2014 Phase Two Expansion

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sunt ut voluptatum eius sapiente, totam reiciendis temporibus qui quibusdam, recusandae sit vero unde, sed, incidunt et ea quo dolore laudantium consectetur!

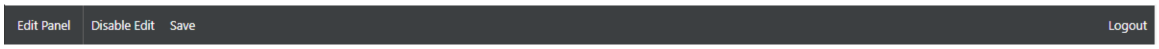
Login



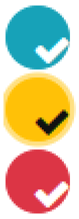
2.4.3.2.1 Page building and management elements

Page builder management elements are visible when user is logged in and has enabled editing option

2.4.3.2.1.1 Management navbar

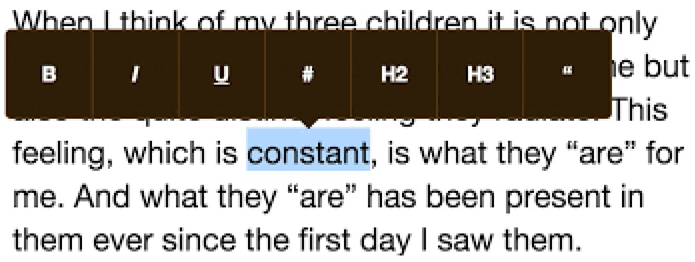


2.4.3.2.1.2 Management side navbar



2.4.3.2.1.3 Text editing control

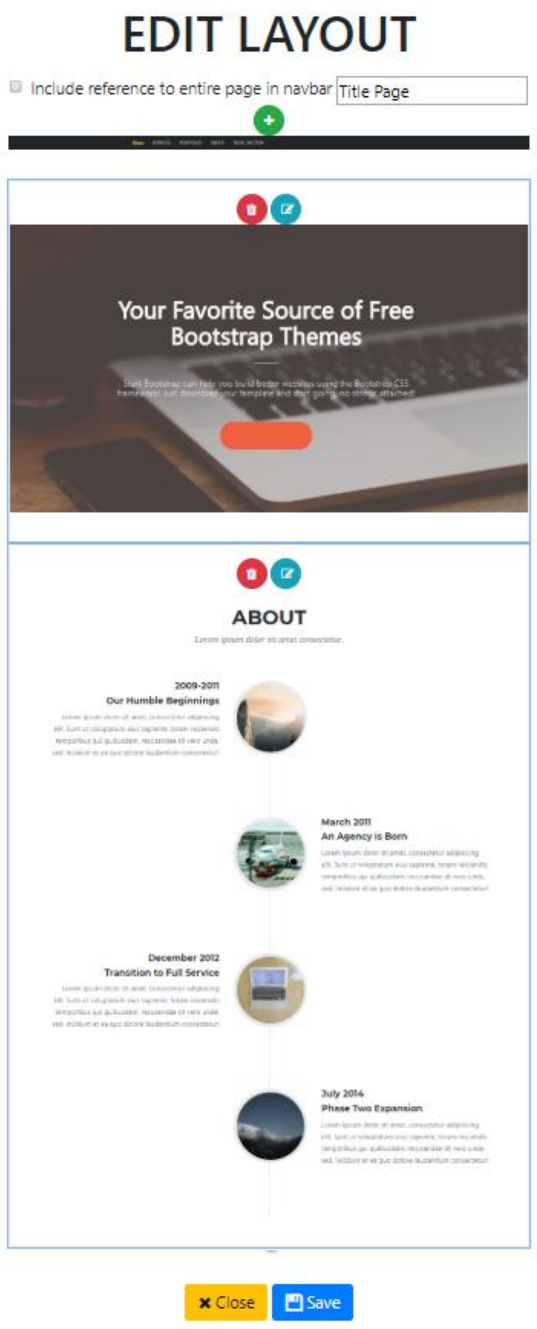
This toolbar becomes visible if editing mode is enabled and text is selected



When I think of my three children it is not only … e but … This feeling, which is constant, is what they “are” for me. And what they “are” has been present in them ever since the first day I saw them.

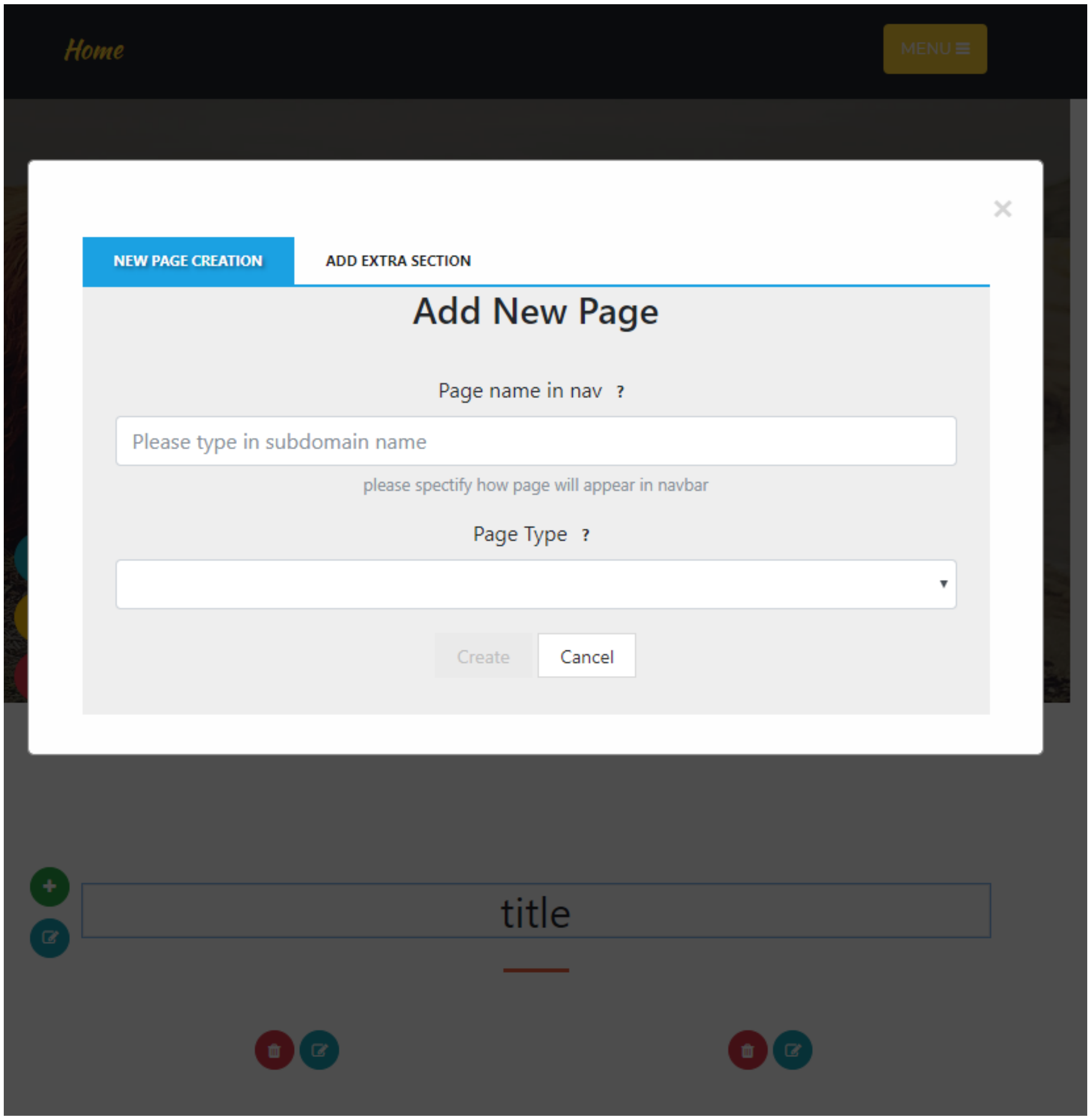
2.4.3.2.1.4 Layout editor

Layout editor can be opened by pressing button on management side navbar on a side



2.4.3.2.1.5 Page manager

Page manager can be opened by pressing button on management side navbar on a side



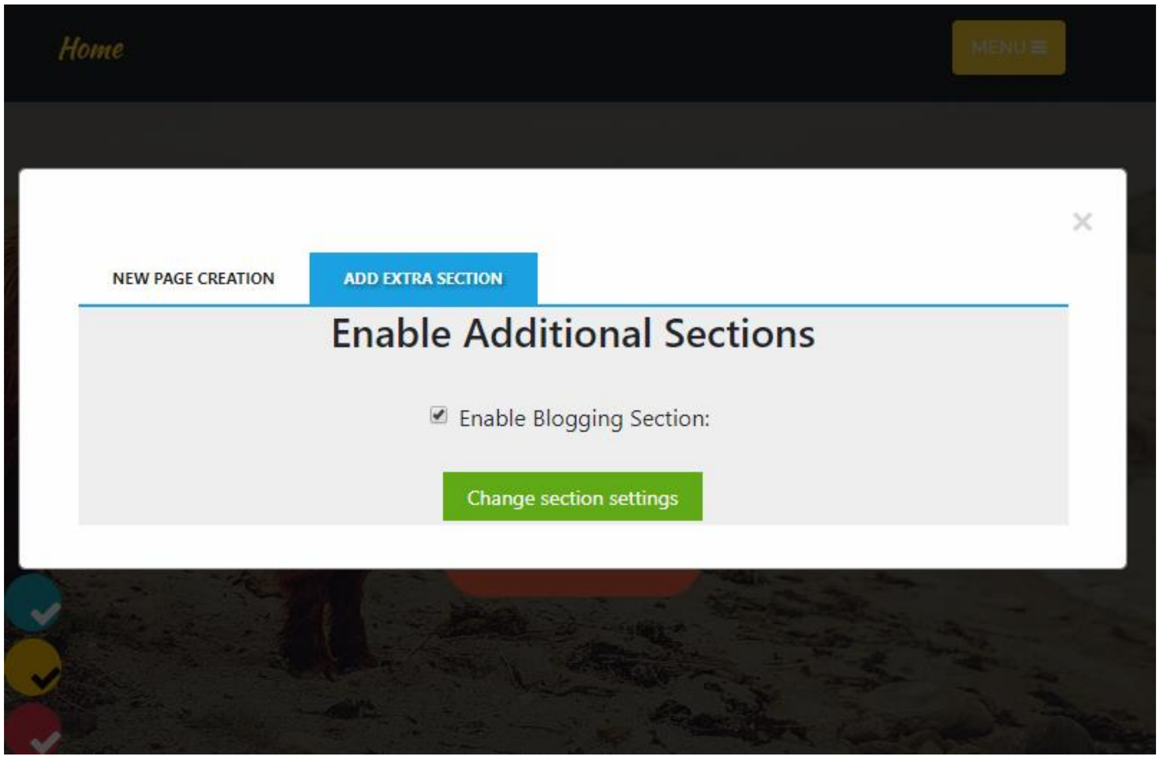
Add New Page

Page name in nav ?

Please type in subdomain name

please spectify how page will appear in navbar

Page Type ?

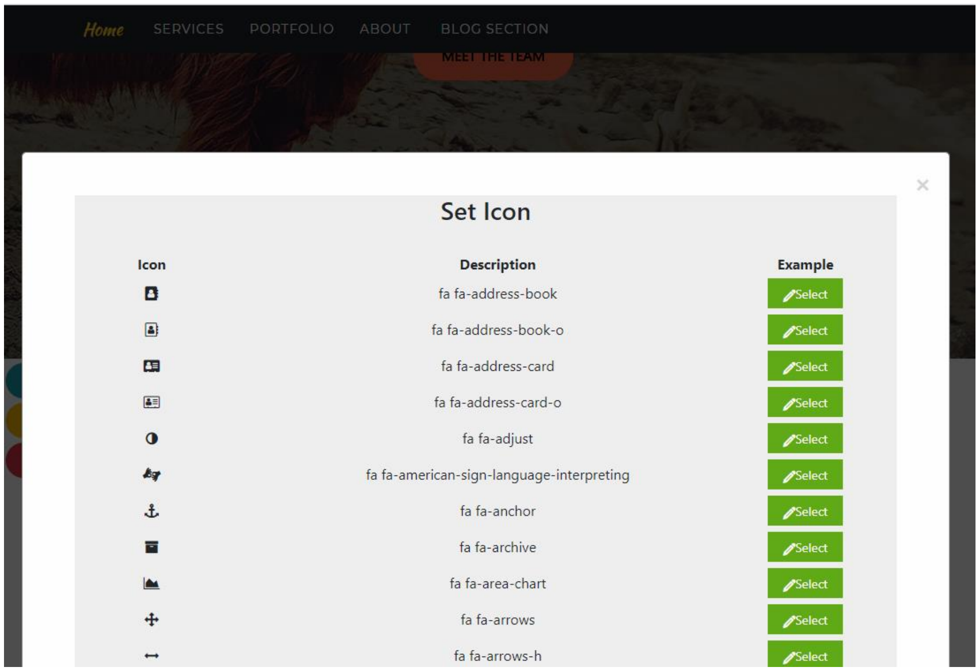


Enable Additional Sections

☑ Enable Blogging Section:

2.4.3.2.1.6 Icon manager

Icon manager can be opened by pressing on control button beside icons in application

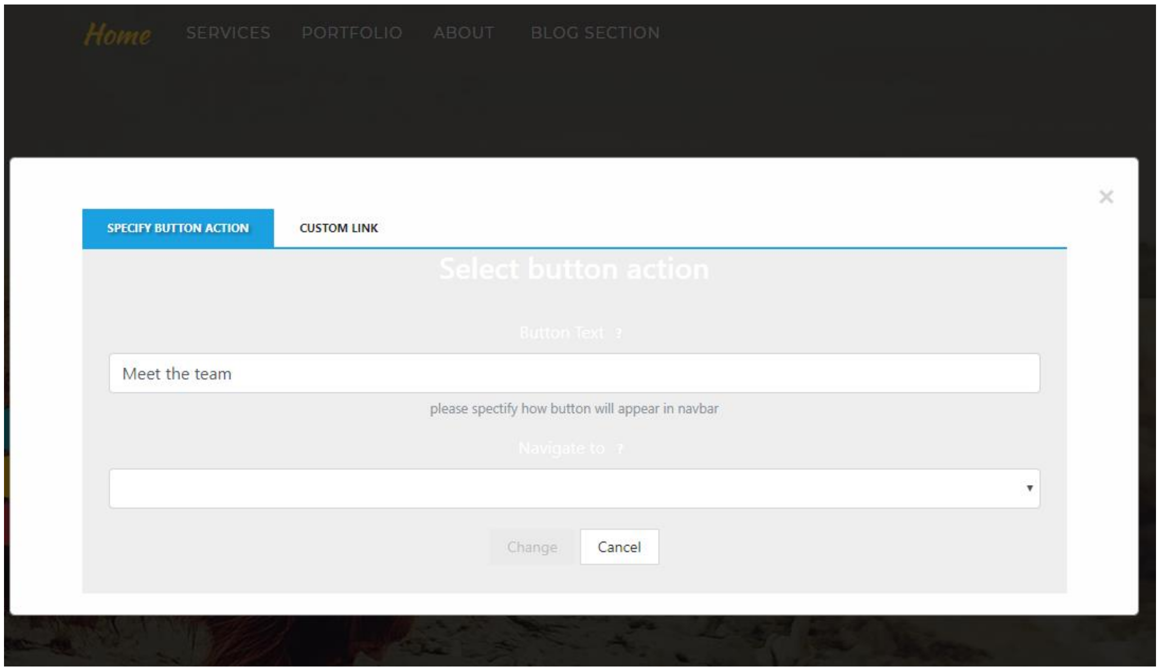


Set Icon

Icon	Description	Example
	fa fa-address-book	Select
	fa fa-address-book-o	Select
	fa fa-address-card	Select
	fa fa-address-card-o	Select
	fa fa-adjust	Select
	fa fa-american-sign-language-interpreting	Select
	fa fa-anchor	Select
	fa fa-archive	Select
	fa fa-area-chart	Select
	fa fa-arrows	Select
	fa fa-arrows-h	Select

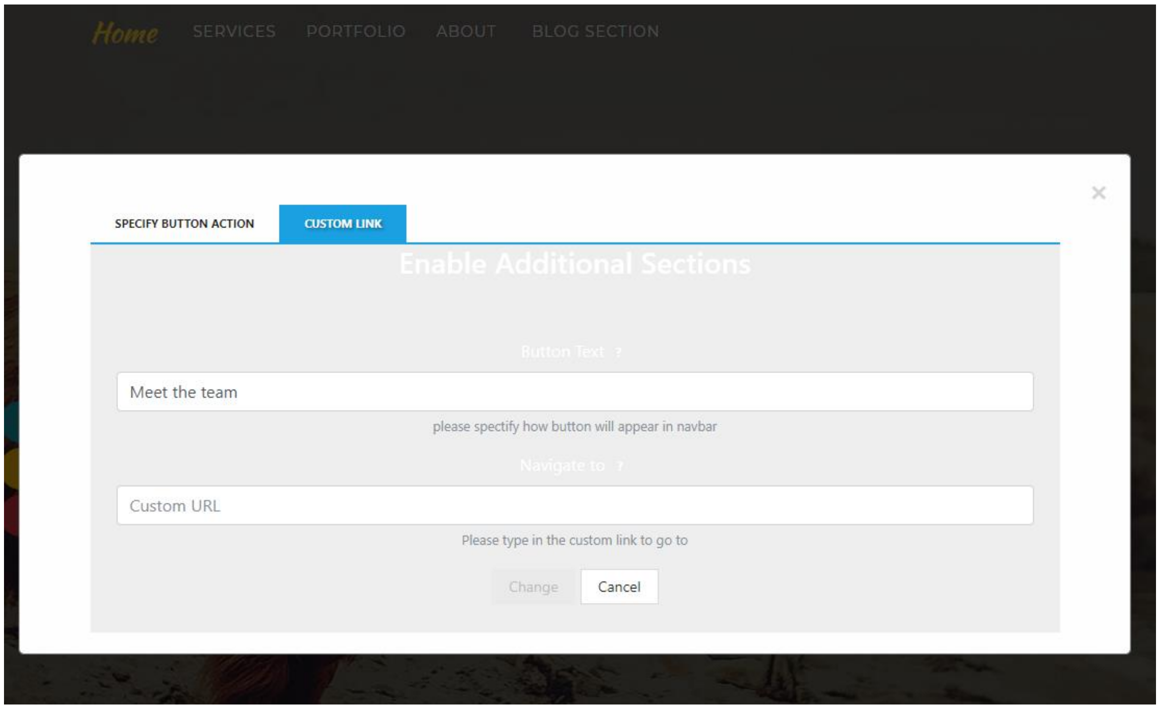
2.4.3.2.1.7 Button Manager

Button manager can be opened by pressing on control button beside buttons in application



Meet the team

please spectify how button will appear in navbar



Meet the team

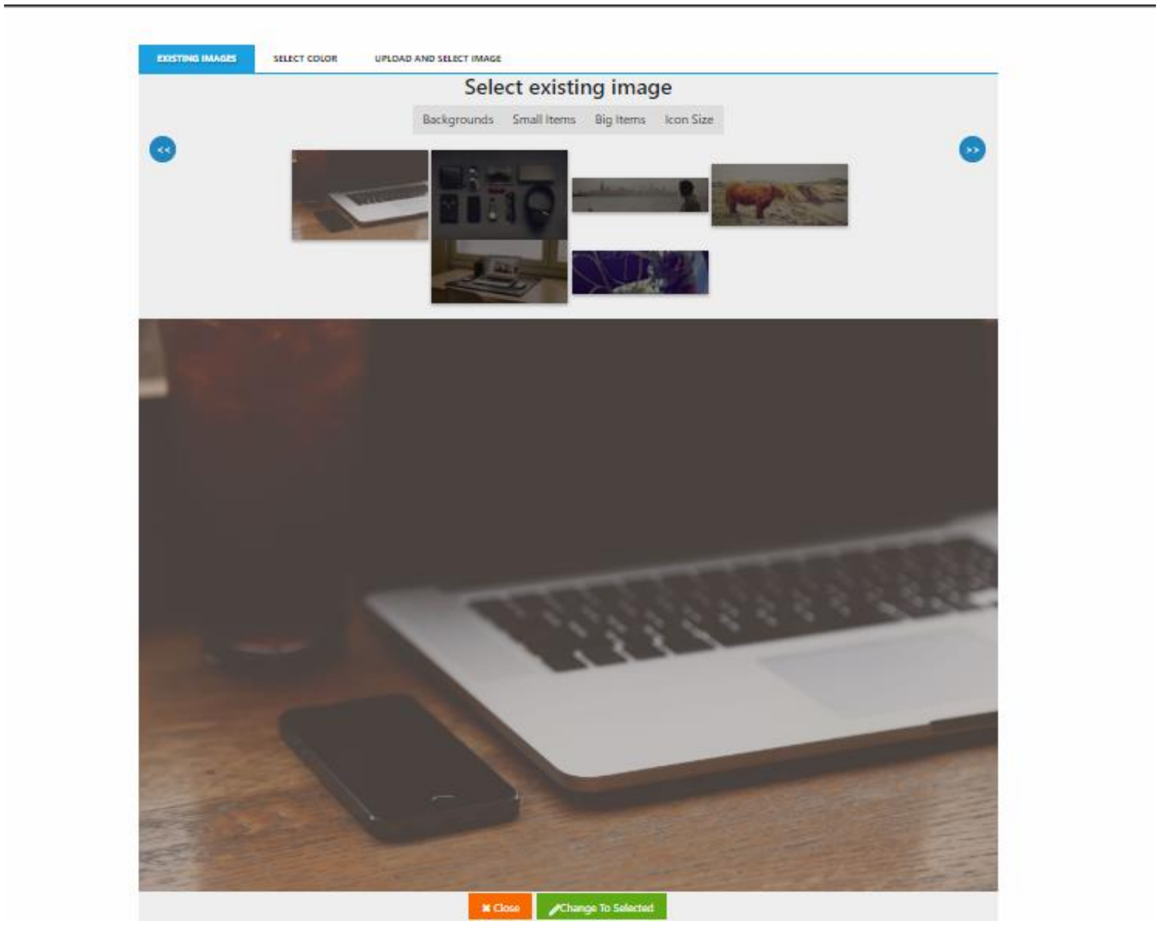
please spectify how button will appear in navbar

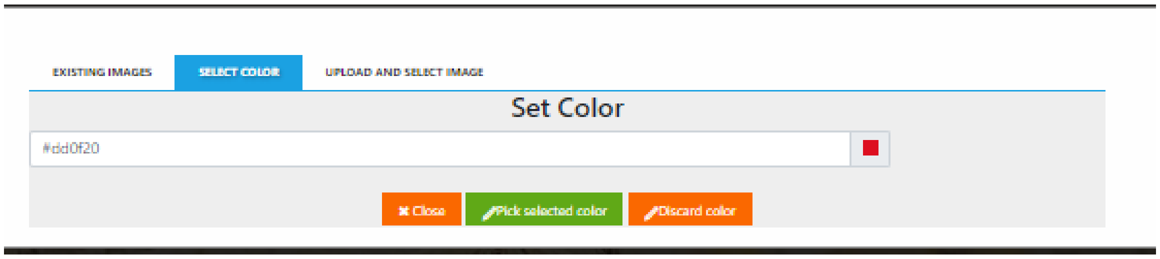
Custom URL

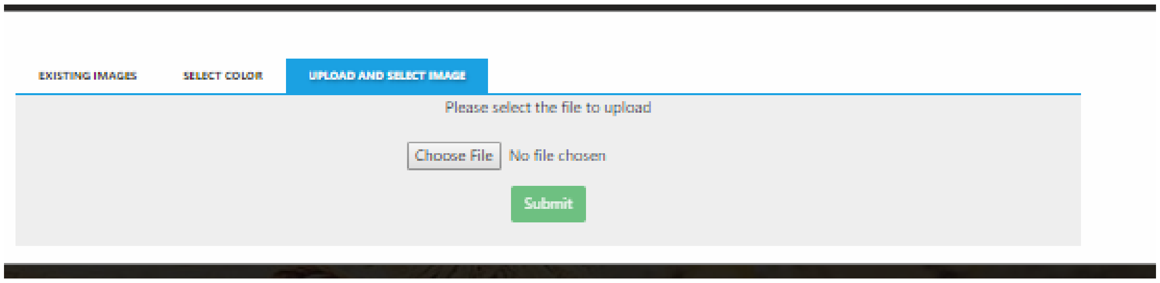
Please type in the custom link to go to

2.4.3.2.1.8 Background Manager

Background manager can be opened by pressing on control button beside elements

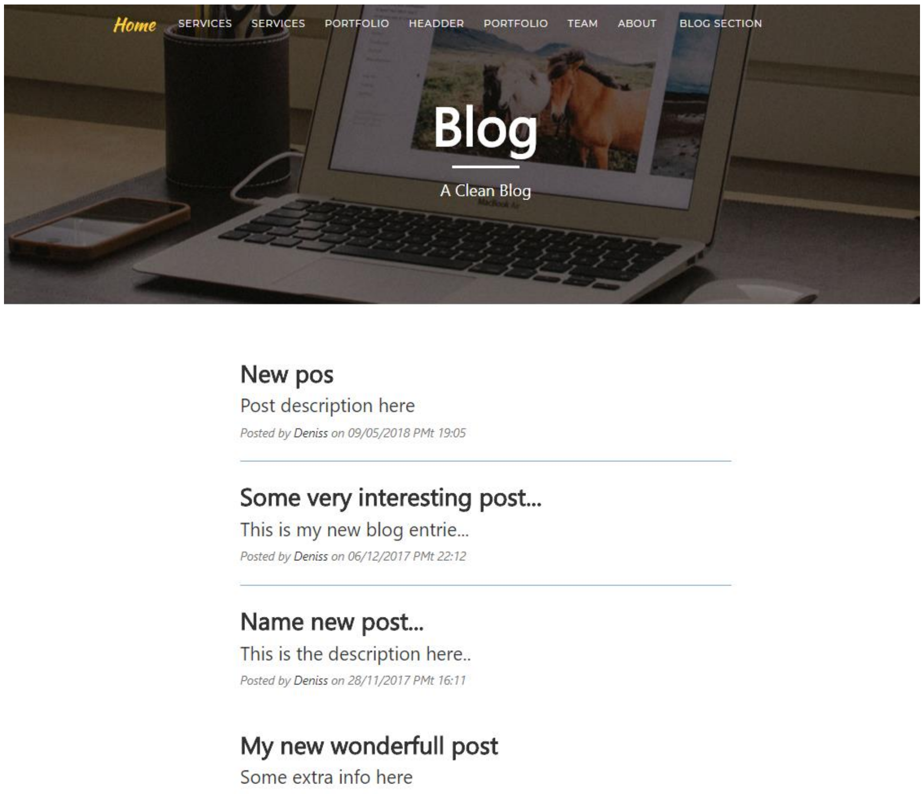






2.4.3.2.1 Blog post section

Blogs can be viewed in the section, and if user is logged in, new blogs can be created.



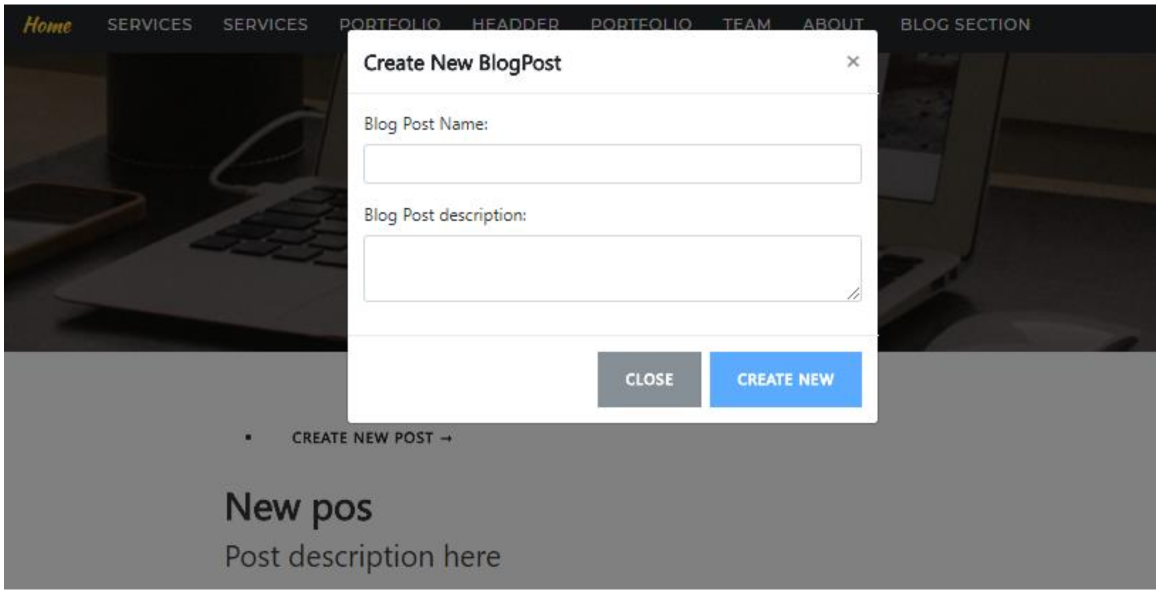
New pos
Post description here
Posted by Deniss on 09/05/2018 PMt 19:05

Some very interesting post...
This is my new blog entrie...
Posted by Deniss on 06/12/2017 PMt 22:12

Name new post...
This is the description here..
Posted by Deniss on 28/11/2017 PMt 16:11

My new wonderfull post
Some extra info here

Create new blog form



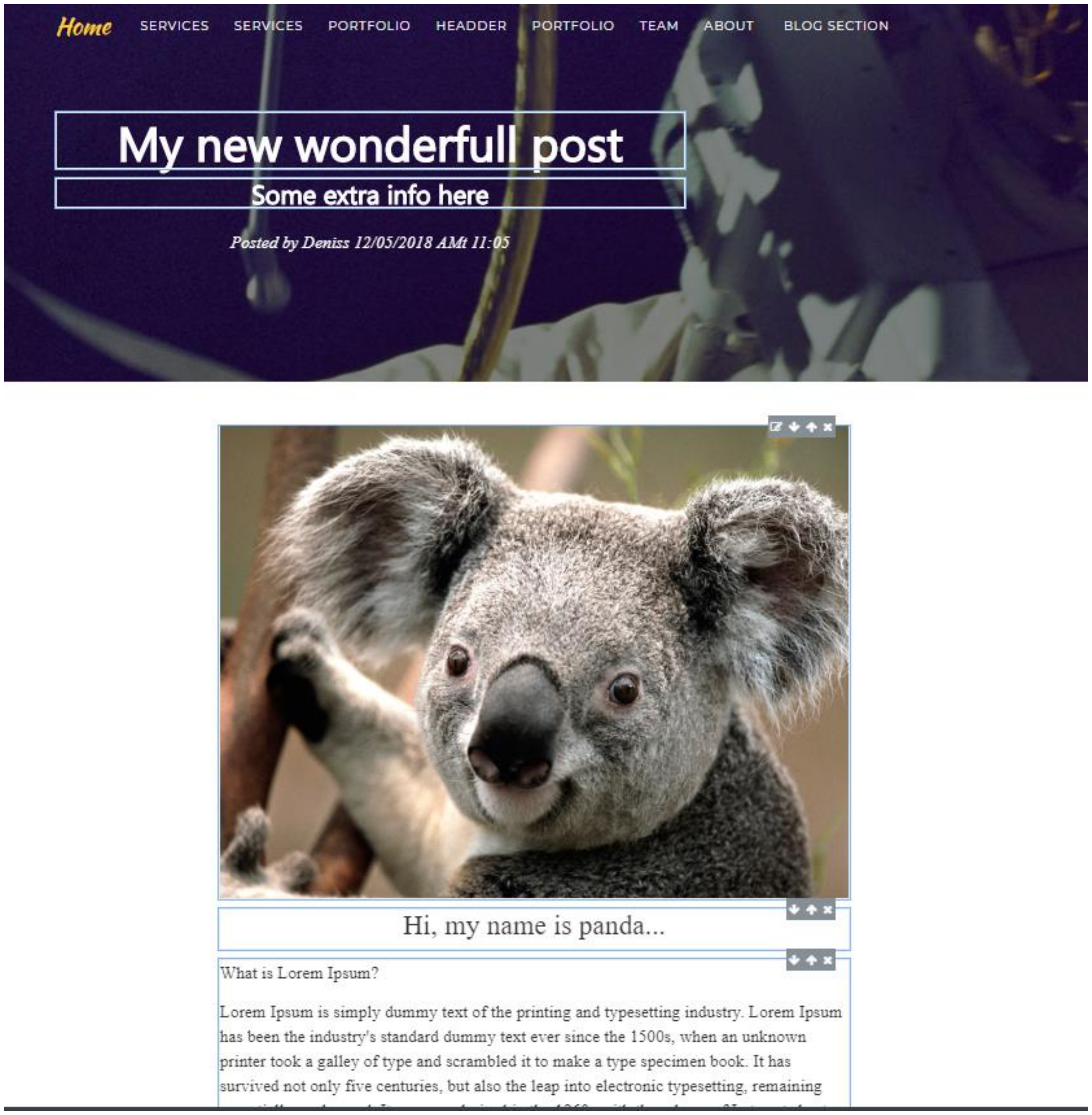
Create New BlogPost

Blog Post Name:

Blog Post description:

New pos
Post description here

Blog post itself



My new wonderfull post

Some extra info here

Posted by Deniss 12/05/2018 AMt 11:05

Hi, my name is panda...

What is Lorem Ipsum?

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining

2.4.4 Design Patterns used

For rapidity of development of our software platform, following design patterns were used

2.4.4.1 Adaptor Pattern

Docker API Server is transforming incoming http requests into native linux commands that are then executed based on action that is required from the requestor.

2.4.4.2 Singleton Pattern

Singleton pattern was used to initially configure docker containers with the environment variables. In the angular system components app.module are initialized just once during the runtime and follow singleton pattern as well.

2.4.4.3 Observer Pattern

Rx.js library is used in order to facilitate node.js and angular.js communication. After http call is conducted, side that conducted the call is waiting for the response, and this way is observing and following observer pattern.

2.4.4.4 Builder Pattern

Builder pattern is used to construct JSON object from multiple small elements. In this way when layout editor in the application is used, and content is modified, it extracts information of what was changed and rebuilds database JSON object from smaller pieces that were selected by user.

2.4.4.5 Prototype Pattern

When the user acquires new instance of the application, it is created from the prototype docker image that is kept in the repository. All the templates database models are built from the prototypes as well.

2.4.4.6 Filter Pattern, Iterator Pattern

These patterns are used trough out the application, for example each loop can be considered part of iterator pattern, each database call with where conditions can be looked at ad filter pattern ascendant.

2.4.4.7 MVC Pattern

As our system is using Angular JS for the front end, it is following the framework standard and is using MVC Pattern trough out the system components.

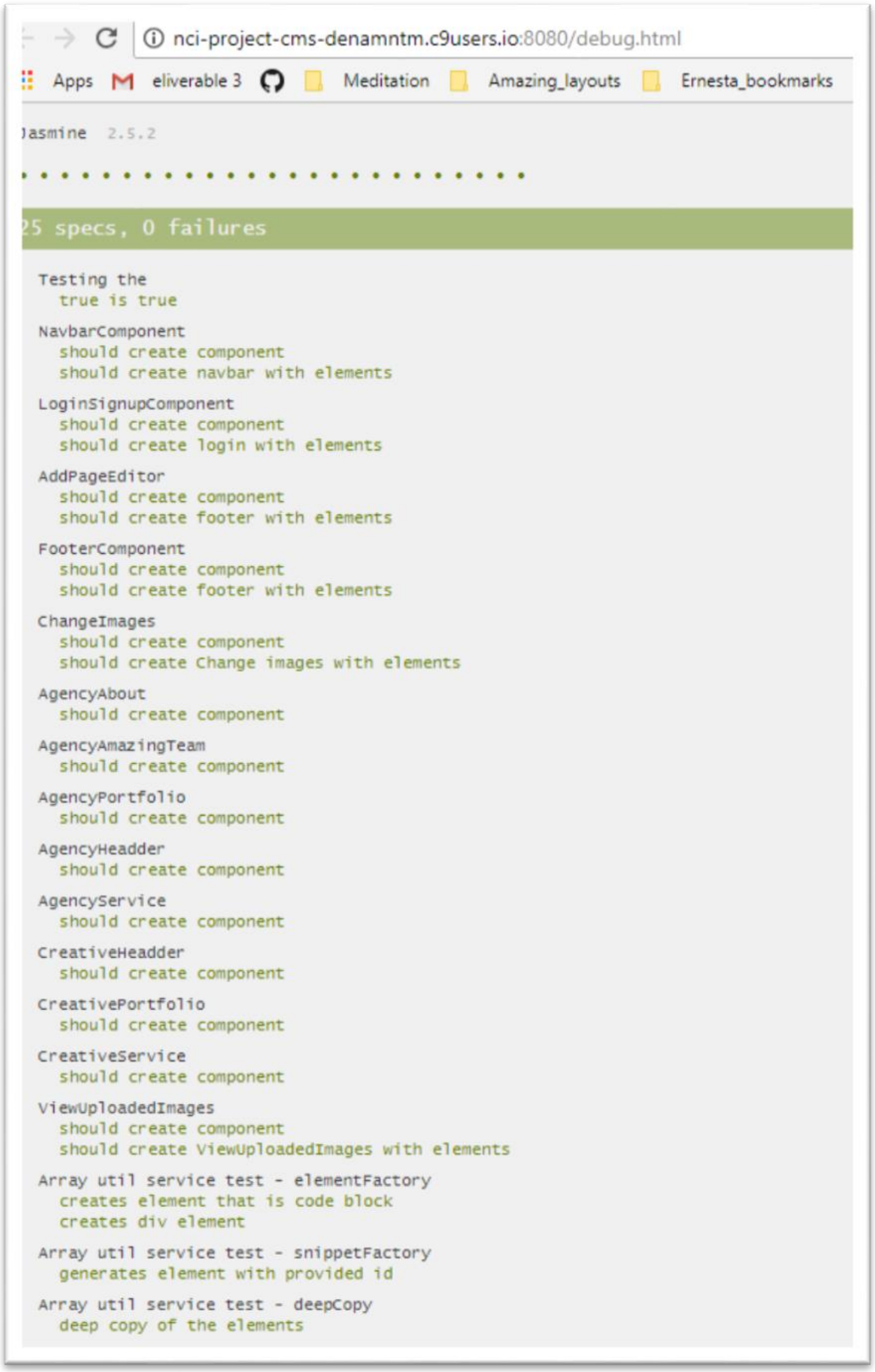
2.5 Evaluation

2.6 Testing

2.6.1 Unit Testing

System was developed using test driven coding practices combined with some basic unit tests that were developed at the end of each component development. Karma and Jasmine unit testing framework combination for JavaScript was utilized in order to enable benefits of unit testing in our application. Now, if need arises for additional development, it can be conducted without fear that extra functionality and logic will cause uncaught problems in the application modules.

25 tests in total were developed in order to cover most of component initiation validation and some custom tests for the array manipulation services were added as well, please see the screenshot below.


nci-project-cms-denamntm.c9users.io:8080/debug.html

Apps eliverable 3 Meditation Amazing_layouts Ernesta_bookmarks

```
Jasmine 2.5.2

• • • • • • • • • • • • • • • • • • • • • • • • •

25 specs, 0 failures

Testing the
  true is true
NavbarComponent
  should create component
  should create navbar with elements
LoginSignupComponent
  should create component
  should create login with elements
AddPageEditor
  should create component
  should create footer with elements
FooterComponent
  should create component
  should create footer with elements
ChangeImages
  should create component
  should create Change images with elements
AgencyAbout
  should create component
AgencyAmazingTeam
  should create component
AgencyPortfolio
  should create component
AgencyHeadder
  should create component
AgencyService
  should create component
CreativeHeadder
  should create component
CreativePortfolio
  should create component
CreativeService
  should create component
ViewUploadedImages
  should create component
  should create ViewUploadedImages with elements
Array util service test - elementFactory
  creates element that is code block
  creates div element
Array util service test - snippetFactory
  generates element with provided id
Array util service test - deepCopy
  deep copy of the elements
```

2.6.2 User acceptance testing

Before system was finalized, user acceptance testing was performed. Test users (In this case test user base were 5 people that I know) were given task to create their websites by means of the system, afterwards they were required to give their feedback by filling questionnaire.

As the result of the user feedback we can say that in general application is easy to grasp and use. Please see summary of the questionnaires below.

Questionnaire summary

1. Do you consider UI of the application to be easy to use? (Yes, No)

   5 out of 5 answered yes

2. Is it easy to subscribe to the hosting service of the platform? (Yes, No)

   5 out of 5 answered yes

3. IS it easy to create your own website on the platform? (Yes, No)

   5 out of 5 answered yes

4. Would this web platform be useful to you? (Yes, No)

   2 out of 5 answered yes

5. Would you recommend this service to a friend? (Yes, No)

   4 out of 5 answered yes

6. What else can be improved in the platform? (Open Question)

   In the open question general feedback was that additional templates for insertion need to be added in order to increase variety of use cases and looks of the web pages created through the CMS Lite.

2.6.3 System Testing

System test was conducted at final stage of the application development. It was verified that each functionality of the application is working as expected. Following actions were tested in order to verify that key features of the system are working correctly

2.6.3.1 Platform Portal System Test

Feature	Description	Status
Create new user	New user creation	Pass
User Login	User logged in	Pass
Create new web app	Creating and Launching new application	Pass
Managing App instances	Start/Stop/Restart/Remove actions	Pass

2.6.3.2 CMS Web Builder System Test

Feature	Description	Status
User Login	User logged in	Pass
Edit text on pages	Checked if content is editable	Pass
Saving content	Check to see if changed content on pages have been saved	Pass
Add new sections to page	Check if new sections can be added to pages	Pass
Change section Theme	Check if theme can be switched	Pass
Change section background	Check if background can be changed	Pass
Add/ Remove blogging section	Check if blogging section can be added / removed	Pass
Add / Remove new page	Check if new page can be added / removed	Pass
Upload images	Check if images can be uploaded	Pass

Add blogs	Check if new blogs can be added	Pass

2.7 System Evolution

It is possible to extend the system beyond the one VM System model and complement it with additional Nginx reverse proxy server with rooting table. It would redirect traffic to additional VM's based of the domain names in XML HTTP request headers.

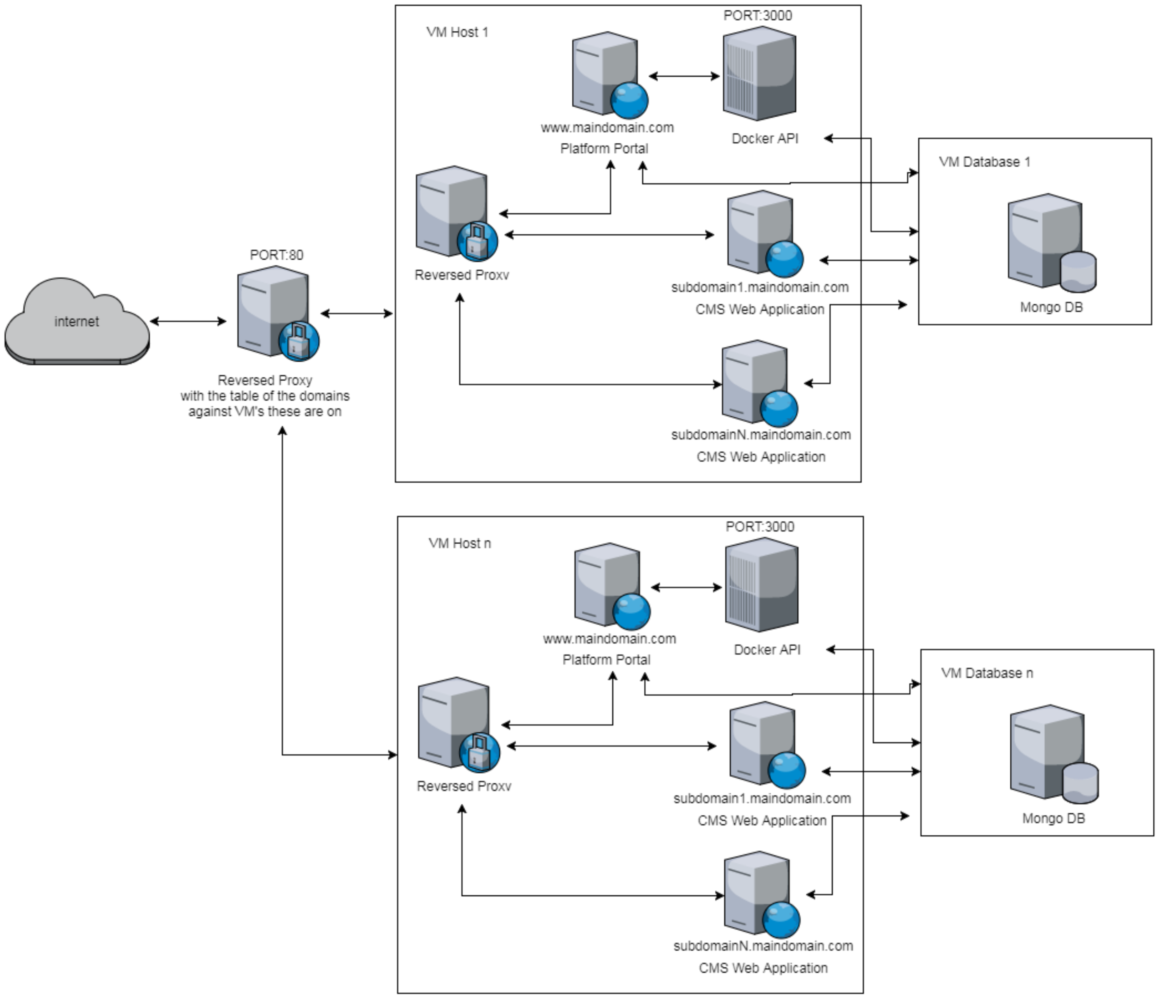


Figure 5, extended version of the system

Figure 5 draws out potential solution with scaling capabilities, main distinction between this model and currently implemented model is that there is the load balancer with the table of domains mapping them to specific VM IP addresses, this way this solution can be scaled horizontally as long as additional VM resources are available. And for example, in the use case when one of the user applications becomes more user demanding thanks to Docker technology we can easily migrate this app to the separate VM with additional resources.

Additionally, if our own built CMS solution will not succeed, platform can be easily reconfigured and adopted to support other CMS Web Systems like WordPress or Joomla to provide additional options for the users.

Additional element blocks can be added in order increase use cases of the built websites on our platform.

System management controls for admins is something that was not developed for this web hosting platform and admins at the moment need to be very skilled in order to support functioning of the system. This means that for system to reach good production level these administrative controls have to be developed in the feature.

3 Bibliography

squarespace.com, 2017. *squarespace.com.* [Online].

wix.com, 2017. *wix.com.* [Online].

wordpress.com, 2017. *wordpress.com.* [Online].

4 Appendix

4.1 Project Proposal

Project-Proposal-x14
100398.docx

4.2 Other Documents

Project Gant
Chart.mpp

ProjectRequirements
Specification.docx

EthicsApprovalForm
_x14100398.docx