OULU UNIVERSITY OF APPLIED SCIENCES

Zhejia Tapani

Study and Integrate Bootstrap 3 for OpixManager

Study and Integrate Bootstrap 3 for OpixManager

Zhejia Tapani
Bachelor’s thesis
Autumn 2014
Degree program in Business Information Technology
Oulu University of Applied Sciences

Oulu University of Applied Sciences
Degree Programme: Business Information Technology

Author(s): Zhejia Tapani
Title of thesis: Study and Integrate Bootstrap 3 for OpixManager
Supervisor: Liisa Auer
Term and year of completion: 2014 Number of pages: 45+ 8 appendices

---

ABSTRACT

This bachelor thesis is about how to study and integrate Bootstrap 3 into OpixManager. The purpose is to improve user interface of OpixManager application. OpixManager is constructed by using CodeIgniter and Model-View-Controller (MVC) framework. OpixManager application is for project management. It includes staff augmentation, customer management, report management and so on. It is to support both scrum and traditional software development process.

There are two major parts included in this thesis work. The theoretical part is about studying what is responsive web design, and what components are included in Bootstrap 3. The author also study how OpixManager's CodeIgniter interrelates among the model, the view and the controller in the theoretical part. The theoretical framework gives the author knowledge how to integrate Bootstrap template to OpixManager's user interface.

The empirical part is actual implementation and testing of template page, login page and home page as well as its link pages in OpixManager. Final results has been tested in cross-browser through local host server such as Mozilla Firefox, in Internet Explorer and in Chrome.

I used Freelancer template's design in this thesis. Typography and color palette are balanced and modern. Navigation bar and grid system are responsive, they are squeezed onto a tiny mobile device or stretched onto a wide screen.

Keywords: Bootstrap, Responsive Web Design, CodeIginiter, MVC, CSS, JavaScript

TABLE OF CONTENTS

1 INTRODUCTION

Modern devices have rapidly developed such as computer tablets, smart phones, notebooks. Devices vary among screen resolutions, screen sizes, functionality, and orientations. Some experts predict that devices' browsing is expected to overtake desktop-based access within the next few years. Designers are facing the challenge of variety of devices, input modes, and browsers than ever era. Users are expecting their web experience to translate on to each of the devices they use. Hence, from long-term trend traditional website design is unable to keep up the speed of changing. It is almost impossible and impractical to create a web site for each resolution and new device. (Knight 2011, cited 26.5.2014.)

Therefore, the concept of responsive web design (RWD) has been firstly introduced by Ethan Marcotte in 2010 in his article on A list Apart. He referred the example of responsive architecture, which explained how architecture responded to surrounding people and environment in the design of physical spaces. Architects considered structures according to responsive environment and various personal requirements such as automatic climate, lighting control and usage of spaces. Based on the idea of a responsive environment, he suggested that rather than creating different designs for different web devices, responsive design will provide more flexible and adaptive solutions to various web devices in order to maximize user web experience.

The choice of this thesis topic was originated from development plan of OpixManager. OpixManager is constructed by using CodeIgniter and Model-View-Controller (MVC) framework. OpixManager application is for project management. It includes staff augmentation, customer management, report management and so on. It is to support both scrum and traditional software development process. (Opixproject Homepage 2014, cited 27.5.2014.) The plan is to improve responsiveness and modernization of OpixManager's user interface. OpixManager is a web-based open source application that is developed by students and lecturers in Oulu University of Applied Sciences, in both Finnish and English Information Technology Degree Programme.

In this thesis Bootstrap 3 is chosen as primary responsive frond-end development framework so as to improve OpixManager's user interface. There are two reasons to ultimately select Bootstrap 3. First of all, Bootstrap 3 is chosen as top one most popular open source responsive HTML5 frameworks, boilerplates and tools for front-end web development (HTML, CSS, and JavaScript). It is well-documented and has plenty of tutorials, as well as refers example sites. Secondly, Bootstrap 3's functionality and usability are stood out among other open source front-end frameworks, which has a 12-grid responsive layout, 13 custom jQuery plugins for common user interfaces like carousels and modal windows. (Bootstrap Homepage 2014, cited 30.5.2014.)

In this thesis, the author studies and integrates Bootstrap 3 framework for OpixManager in both theoretical and empirical aspects. This thesis research is divided into following chapters. First of all, chapter 2 and 3 comprise with theoretical part through four parts. The first part aims at constituting a theoretical framework of responsive web design. The second section gives general introduction to Bootstrap 3. After that, the third part describes terms of Model–View–Controller (MVC) and CodeIgniter. Ultimately the final part analyzes current coding system of OpixManager and figure out how to immigrate current code into Bootstrap 3 framework that based on HTML5, CSS3, and JavaScript, jQuery in order to design and implement partial new user interfaces of OpixManager

Chapter 4 is the empirical part of this thesis. The partial pages of OpixManager are integrated using Bootstrap 3 framework. There are three prime sections in order to gain objectives. The first section aims at choosing proper user interface for OpixManager. In the second section, the primary purpose is to integrate Bootstrap 3 template into OpixManager's CodeIgniter framework by editing and modifying code in both Bootstrap 3 and OpixManager. In the ultimate section, final results are capable of testing and implementing. The last two chapters 5 and 6 conclude and discuss some implications of the research.

1.1 Research Questions and Theoretical Framework

The main research problem of this thesis is to study and integrate Bootstrap 3 into OpixManager. The question is answered through integrating theories with empirical results.

Design and implementation of OpixManager's user interfaces are based on requirement specification, architecture&design, implementation, testing and installation. Flow chat (Figure 1) demonstrates preliminary work flow in this thesis project. HTML5, CSS5, jQuery, JavaScript, PHP, MySQL are required technologies skills in this thesis.

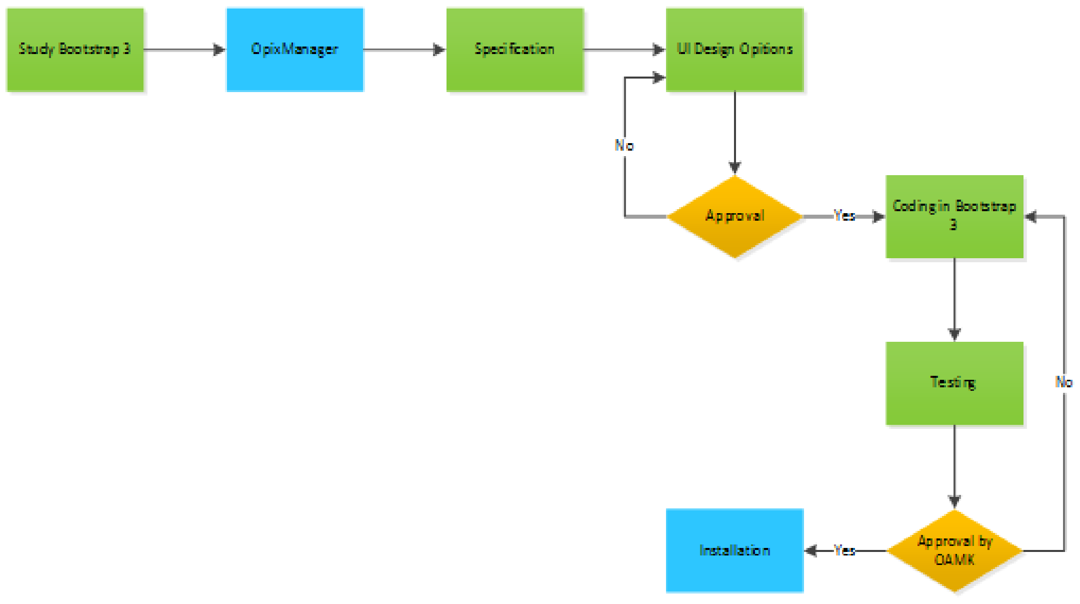


FIGURE 1. Work flow in project

Study Bootstrap 3: Understand Bootstrap 3 including elements and documentations, how to start and how to code base on it.
Specification: Specify what kind of functions and specific requirements of user interface design commissioner wants
UI Design Options: Define visual UI design according to specification, which need to be approved by commissioner
Coding in Boostrap: On this stage the actual programming begins. The output will be ready for testing and evaluation.
Testing: Start the testing and evaluating the final OpixManager application. If there are some faults on coding or design, it will go back to coding stage and revise.
Installation: Depend on commissioner, they can decide if they want to officially install the results or not.

1.2 OpixManager Introduction

OpixManager is a web-based open source application that is developed by students and lecturers in Oulu University of Applied Sciences, in both Finnish and English Information Technology Degree Programme. OpixManager has been created in 2011 and was developed in educational purposes and needs. OpixManager is available in opixproject that provides an open source environment for designing and implementing OpixManager. (Opixproject Homepage 2014, cited 27.5.2014.)

The user interface of OpixManager is implemented using HTML5, CSS3 and jQuery language, the database is MySQL and the programming language is PHP. MySQL database stores all the data used in the application. CodeIgniter is using as OpixManager framework, which almost all development work is based on. CodeIgniter includes PHP ready functions that are used in the application, which reduces the programming considerably. (Opixproject Homepage 2014, cited 27.5.2014.) The author would choose NetBeans software as main editor in empirical part. The reason to choose NetBeans is because the author is familiar with, and it has been widely used in school.

OpixManager application is for project management. It includes staff augmentation, customer management, report management and so on. Project management in OpixManager is applicable to two different project types that are Scrum and traditional. It is to support both Scrum and traditional software development process. The application has been implemented in all the translations in English and Finnish.

Traditional project management is identified as step-to-step approach. It assess the project through five stages: initiation, planning and design, execution and construction, monitoring and controlling system, completion. Each step begins when preceding stage has been completed. It is not easy to anticipate significant changes and respond in time. (Wikipedia 2014 b, cited 24.9.2014.) See details in figure 2:

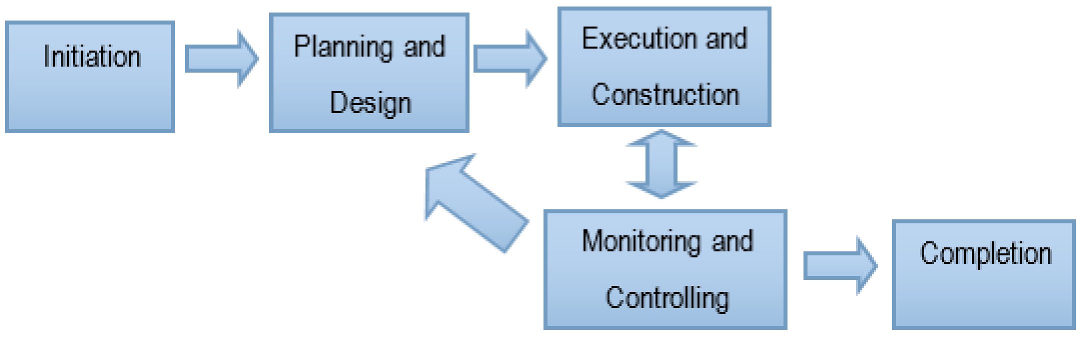


FIGURE 2. Traditional project management

Scrum is a commonly used agile process for product development, especially for software development. Scrum is an effective project management framework that is applicable to any project with complex processes, aggressive deadlines, and demanding requirements. Unlike traditional project management, Scrum projects move forward via a series of iterations called sprints. Each sprint lasts typically two to four weeks long. Within each sprint period, daily Scrum meeting is required for monitoring progress. (Scrum Overview for Agile Software Development, cited 28.5.2014.) See details in figure 3:

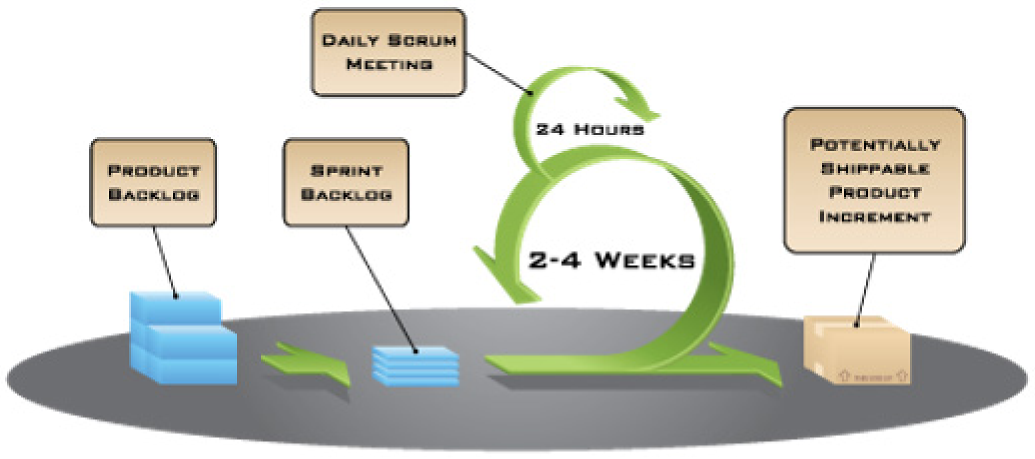


FIGURE 3. Scrum visual introduction (Scrum Overview for Agile Software Development, cited 28.5.2014)

2 RESPONSIVE WEB DESIGN AND BOOTSTRAP 3

2.1 Responsive Web Design

Responsive Web design (RWD) is a web design approach that aims at providing users optimal viewing experience. As the user switches cross a wide range of devices, the website should automatically switch to accommodate for variety of resolution, image size and scripting abilities. In other words, the website should automatically respond to different web devices in order to meet the user's preferences. Responsive Web Design would eliminate the need for a different design and development phase for each new gadget on the market. (Knight 2011, cited 26.5.2014; Adrian, cited 26.5.2014.) Below is an RWD example (see figure 4).

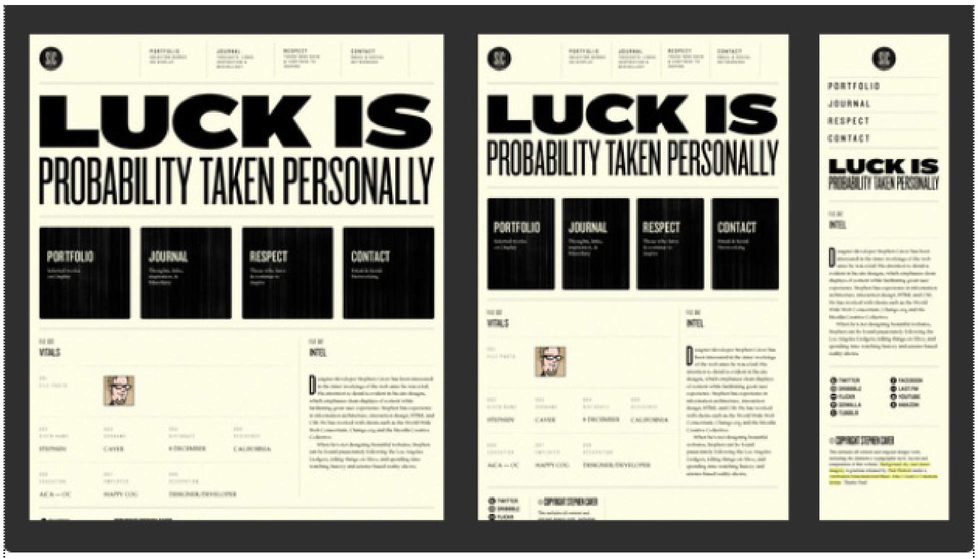
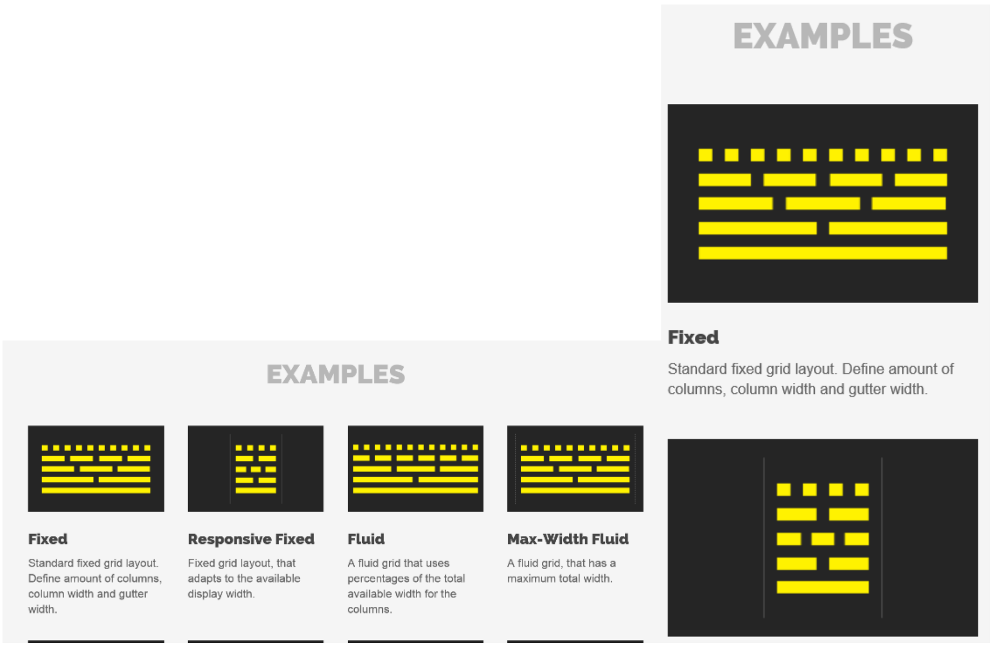


FIGURE 4. Examples of RWD (Adrian, cited 26.5.2014)

A site designed with RWD adapts the layout to the viewing environment by using fluid, proportion-based grids, flexible images, and CSS3 media queries.

Fluid Grids

Responsive design is the usage of fluid grid. Compare to traditional liquid layout that is a fixed number of pixels across and centred on the page, the fluid grid is more carefully designed in terms of proportions. In other words, when a layout is squeezed onto a tiny mobile device or stretched across a wide screen, all elements in the layout will resize their widths in relation to one another as figure 5 shows.



Standard fixed grid layout. Define amount of columns, column width and gutter width.

Fixed grid layout, that adapts to the available display width.

A fluid grid that uses percentages of the total available width for the columns.

A fluid grid, that has a maximum total width.

Standard fixed grid layout. Define amount of columns, column width and gutter width.

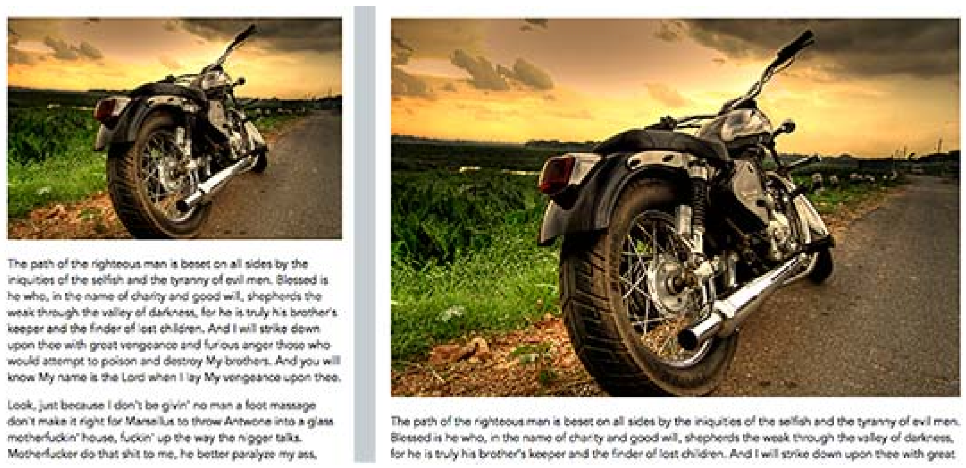
FIGURE 5: Fluid Grid (Profoundgrid, cited 06.11.2014)

Flexible images

Flexible images (or Fluid images) are sized in relative units rather than absolute pixel dimensions, so as to prevent images display outside of contained elements. Although there are number of techniques to resize, the most popular relative solution is to set max-width as 100% that noted in Ethan Marcotte's article on fluid images but first experimented with by Richard Rutter.

```
img { max-width: 100%; }
```

Images with maximum width 100% will display their original size unless browser window width becomes narrower than images' original width. Images will automatically scale to fit browser container, as you can see in the illustration following figure 6. However, this solution also comes with few limitations, for instance scaled image would be relative too large to text if original image size is over 420 pixels, the layout could "pop" when the user visits the page for the first time because not setting the image's height and width explicitly in the css. (Knight 2011, cited 26.5.2014.)


The path of the righteous man is beset on all sides by the iniquities of the selfish and the tyranny of evil men. Blessed is he who, in the name of charity and good will, shepherds the weak through the valley of darkness, for he is truly his brother's keeper and the finder of lost children. And I will strike down upon thee with great vengeance and furious anger those who would attempt to poison and destroy My brothers. And you will know My name is the Lord when I lay My vengeance upon thee.

Look, just because I don't be givin' no man a foot massage don't make it right for Marsellus to throw Antwone into a glass motherfuckin' house, fuckin' up the way the nigger talks. Motherfucker do that shit to me, he better paralyze my ass.

The path of the righteous man is beset on all sides by the iniquities of the selfish and the tyranny of evil men. Blessed is he who, in the name of charity and good will, shepherds the weak through the valley of darkness, for he is truly his brother's keeper and the finder of lost children. And I will strike down upon thee with great

*FIGURE 6. Comparison of max-width scaled image with narrowed browser window (*Storey 2014*, cited 30.7.2014)*

Media queries

Media queries are to use different css style rules based on various characteristics of web devices. The min-width sets a minimum browser or screen width that a certain set of styles can apply for. And the max-width set a maximum browser or screen width. Anything above maximum or below minimum would not apply for the responsive media queries, below is an example. (Knight 2011, cited 26.5.2014.)

```
@media screen and (min-width: 800px) and (max-width: 1200px) {
     .classForaMediumScreen {
          background: #cc0000;
          width: 30%;
          float: right;
     }
}
```

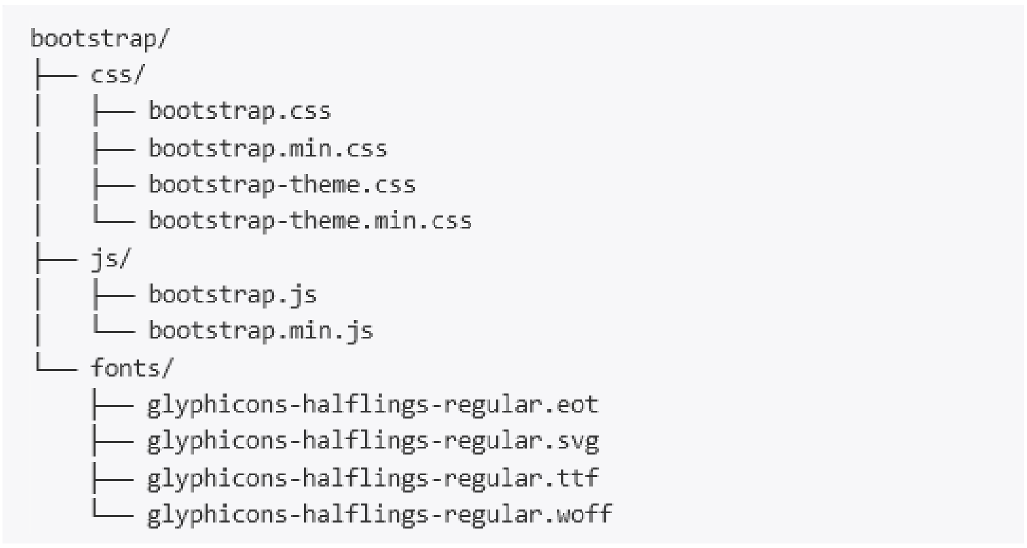
2.2 Bootstrap 3

Bootstrap 3 is an open-source front-end framework for creating websites and web applications. It has a 12-grid responsive layout, and 13 custom jQuery plugins. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Recently, community members have translated Bootstrap

3's documentation into various languages, including Chinese, Spanish and Russian. (Bootstrap 3 Homepage 2014, cited 30.5.2014; David 2013, cited 30.5.2014.)

Bootstrap 3 is compatible with the latest versions of all major browsers Internet Explorer, Firefox, Chrome, and so on. Version 2.0 it also supports responsive web design, the layout of web pages adjusts dynamically by switching cross variety of resolution, image size and scripting abilities in devices (desktop, tablet, mobile phone). Starting with version 3.0, it also adopts a mobile first design with emphasizing responsive design by default.

When Bootstrap 3 package is downloaded, the compiled structure of Bootstrap 3 folder (see in figure 7) contains compiled CSS and JavaScript (Bootstrap 3.*), as well as compiled and minified CSS and JavaScript (Bootstrap 3.min.*), fonts from glyphicons as the optional Bootstrap 3 theme.



```
bootstrap/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.min.css
│   ├── bootstrap-theme.css
│   └── bootstrap-theme.min.css
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
    ├── glyphicons-halflings-regular.eot
    ├── glyphicons-halflings-regular.svg
    ├── glyphicons-halflings-regular.ttf
    └── glyphicons-halflings-regular.woff
```

FIGURE 7. Complied Bootstrap 3 files (Bootstrap Homepage 2014, cited 30.5.2014.)

2.3 CSS in Bootstrap 3

CSS settings in Bootstrap 3 theme are used to set up HTML elements styles such as layout, colour and user interface. CSS settings are applied in custom styles by adding and overwriting extensible classes. (Bootstrap Homepage 2014, cited 30.5.2014.)

HTML5 doctype: Due to usage of HTML elements and CSS, Bootstrap 3 requires the use of the HTML5 doctype.

```
<!DOCTYPE html>
<html lang="en">
  ...
</html>
```

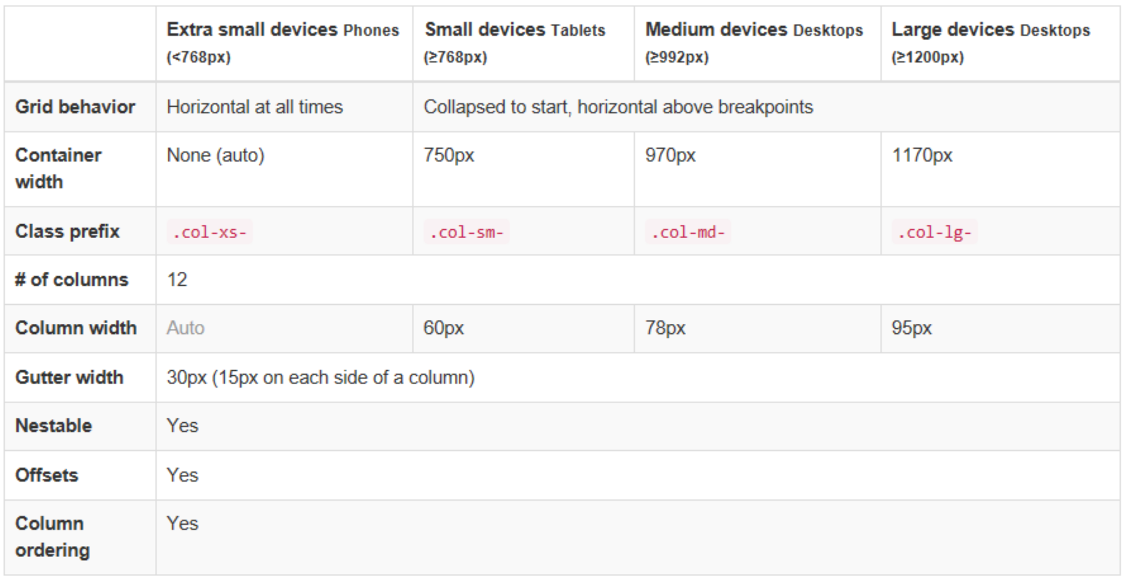
Typography and links: Bootstrap 3 sets definition of background colour, typography, and link styles.

Containers: .container class sets page contents in centre and adjusts width at various media query breakpoints to match grid system.

```
<div class="container">
  ...
</div>

@media (min-width: 568px) {
  .container {
    width: 550px;
  }
}
```

Grid system

Bootstrap 3 includes a responsive, mobile first 12 columns grid layout while the device or viewport size changes. It includes predefined classes for simplifying layout options, as well as contains powerful mixins that are used in conjunction with the grid variables to generate semantic css. Grid systems are used for creating page layouts through a series of rows and columns that constructs page contents. Figure 8 shows how the Bootstrap 3 grid system works across multiple devices. (Bootstrap Homepage 2014, cited 30.5.2014.)



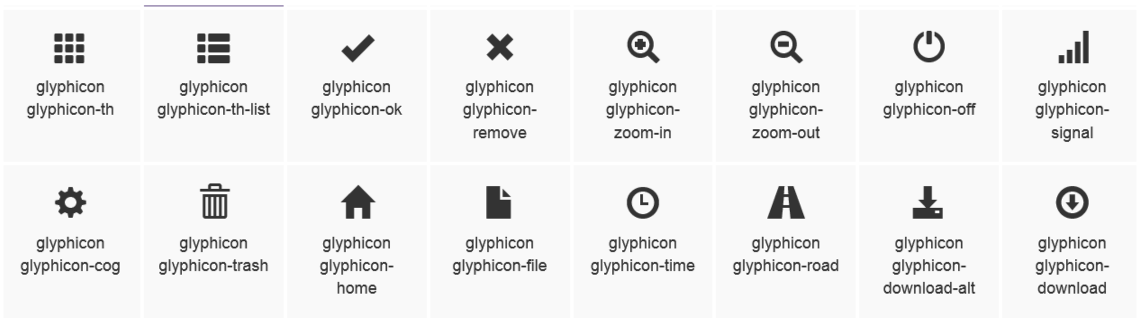
	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints		
Container width	None (auto)	750px	970px	1170px
Class prefix	`.col-xs-`	`.col-sm-`	`.col-md-`	`.col-lg-`
# of columns	12			
Column width	Auto	60px	78px	95px
Gutter width	30px (15px on each side of a column)			
Nestable	Yes			
Offsets	Yes			
Column ordering	Yes			

FIGURE 8. Grid options (Bootstrap Homepage 2014, cited 30.5.2014)

2.4 Examples of Components in Bootstrap 3

Bootstrap 3 contains commonly used interface components. Components include buttons with advanced features (e.g. grouping of buttons or buttons with drop-down option, horizontal and vertical tabs, etc.), labels, and glyphicons, advanced typographic, thumbnails, capabilities, warning messages and a progress bar. The author will take glyphicons and navigation bar as demonstrated ones in below paragraphs. (Bootstrap Homepage 2014, cited 30.5.2014.)

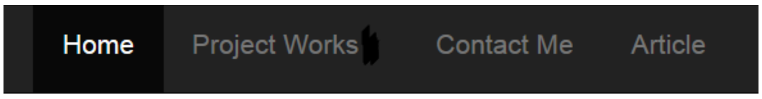
Glyphicons: Bootstrap template includes 200 glyphs in font format from the glyphicon halflings set. Glyphicons available for Bootstrap 3 are free of charge. Below figure 9 presents some examples of Glyphicons.



glyphicon glyphicon-th
glyphicon glyphicon-th-list
glyphicon glyphicon-ok
glyphicon glyphicon-remove
glyphicon glyphicon-zoom-in
glyphicon glyphicon-zoom-out
glyphicon glyphicon-off
glyphicon glyphicon-signal
glyphicon glyphicon-cog
glyphicon glyphicon-trash
glyphicon glyphicon-home
glyphicon glyphicon-file
glyphicon glyphicon-time
glyphicon glyphicon-road
glyphicon glyphicon-download-alt
glyphicon glyphicon-download

FIGURE 9. Examples of glyphicons in Bootstrap 3 (Bootstrap Homepage 2014, cited 30.5.2014)

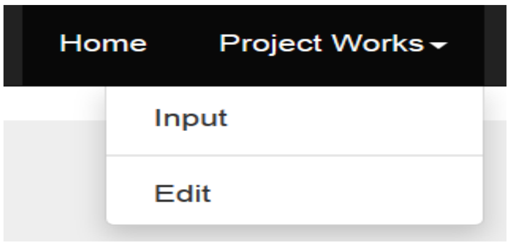
Navigation bar in Bootstrap is a responsive component that serves as navigation headers for application or site. It begins collapsed in mobile view and automatically adjusts to scale while available viewports increase. Below is an example of inverted navigation bar by simply adding `.navbar-inverse`.



```
<div class="navbar navbar-inverse navbar-static-top">
<div class="collapse navbar-collapse navHeaderCollapse">

<ul class="nav navbar-nav navbar-right">

<li class="active"><a href="index.html">Home</a></li>
<li><a href="#">Project Work</a></li>
<li><a href="#">Contact Me</a></li>
   <li><a href="#">Article</a></li>
   </ul>
   </div>
   </div>
```

FIGURE 10. Inverted navigation bar

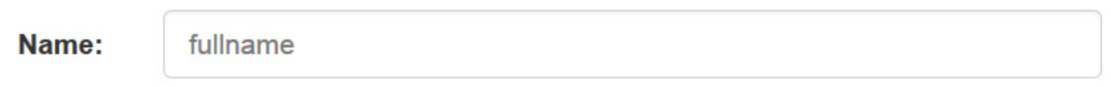
Dropdown menu within navigation bar is to add `.dropdown` class under `.navbar` class and use `<span class="caret"></span>` to act as an indicator.



```
<ul class="nav navbar-nav navbar-right">
<li class="dropdown">
<a href="#" class="dropdown-toggle" data-
toggle="dropdown">Project Works<span
class="caret"></span></a>
<ul class="dropdown-menu" aria-labelledby="dropdownmenu1">
<li><a href="#">Input</a></li>
<li class="divider"></li>
<li><a href="#">Edit</a></li>
</ul>
```

FIGURE 11. Dropdown within navigation bar

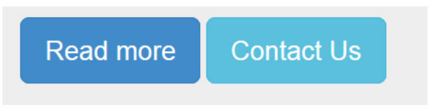
Forms are placed content within `.form-group` and collapsed behavior in narrow viewports.



```
 <div class="form-group">
 <label    for="contact-name"    class="col-lg-2    control-
label">Name: </label>
 <div class="col-lg-10">
<input  type="text"  class="form-control"  id="contact-name"
placeholder="fullname">
</div>
</div>
```

FIGURE 12. Form

Buttons are defined to add `.btn` class in `<button>` tag. Button types can be defined by adding `.btn-primary` or `.btn-info`.

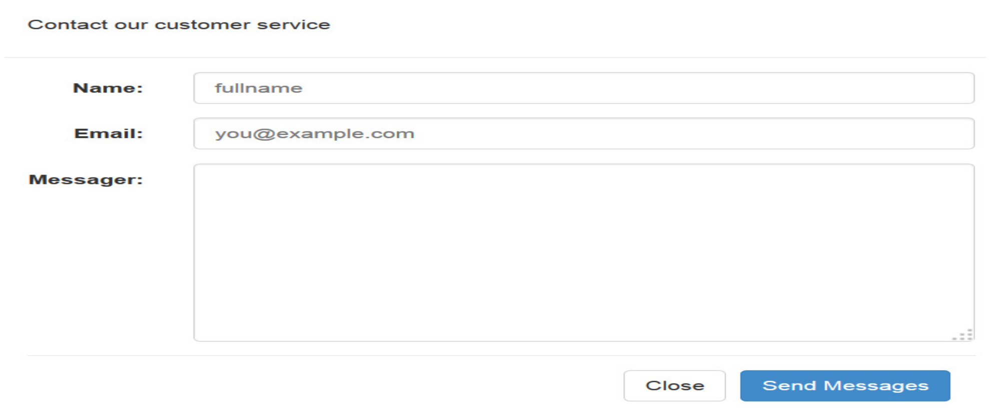


```
<button type="button" class="btn btn-primary btn-lg">Read
more</button>
<button type="button" class="btn btn-info btn-lg">Contact
Us</button>
```

FIGURE 13. Buttons

2.5 JavaScript in Bootstrap 3

There are over a dozen of custom jQuery plugins available in Bootstrap 3 JavaScript. They are either included in Bootstrap *.js or Bootstrap.min.js file. Main JavaScript components are modal, dropdown, scrollspy, tab, tooltip, and popover and so on. Hence, when coding in index.html, the user do not need to write the whole JavaScript but linking JavaScript files in `<script>` tag. Below figure 14 shows an example of modal form. First of all, Bootstrap.js and jquery-1.9.0.js are required to link in `<script>` tag. After that, modal is defined in `.modal`, `.fade` class in `<div>` tag. (Bootstrap Homepage 2014, cited 30.5.2014.)


Contact our customer service

Name: fullname

Email: you@example.com

Messager:

```
<div class="modal fade" id="contact" role="dialog">
<script src="http://code.jquery.com/jquery-
1.9.0.js"></script>
<script src="js/Bootstrap.js"></script>
</div>
```

FIGURE 14. Modal

3 OPIX MANAGER CODING SYSTEM

3.1 CodeIgniter

"CodeIgniter (CI) is a powerful PHP-based MVC framework that has a very small footprint, especially for PHP coders who need a simple and easy-to-use toolkit to create full-featured web applications or sites" (Ellislab 2014 a, cited 15.8.2014). CodeIgniter is a PHP framework based on MVC (Model–View–Controller). It is widely used in many sites or applications because of user interface, logical structure and well-documented tutorials available in website. Using CodeIgniter, programmers and designers can simultaneously work on the same project, without having to wait for each other to complete their parts. It is the main difference between CodeIgniter and other PHP framework. (Wikipedia 2014 a, cited 24.9.2014.)

There are several advantages of using CodeIgniter. Migration from one server to another is just to change the URL address. Actual installation that just need to download files from www.codeIgniter .com, extract it and put it into the server. Based on MVC framework, it has flexibility. The collection of libraries that users are capable of processing. Clear and concise manual guide convenient any coder to follow and reach in the whole framework. (Macronimous, cited 24.8.2014.)

3.2 Model-View-Controller (MVC)

Model–View–Controller (MVC) is a software architectural pattern for implementing user interfaces. Its architectural pattern divides a given software application into three components: the model, the view, and the controller. The MVC separation also increases team work efficiency in web application development. Developers can reuse the application's logic when implementing a different view. Different developers in the team can work on the view, the controller logic, and the model logic simultaneously without affecting each other. (Wikibook 2014, cited 24.9.2014.)

The core component, the model is to store and update data in database. The second component, the view is a visual representation of the model. The third part, the controller handles input and converts commands for the model or view. Figure 15 illustrates interconnection of three main components. (Wikibook, cited 24.9.2014.)

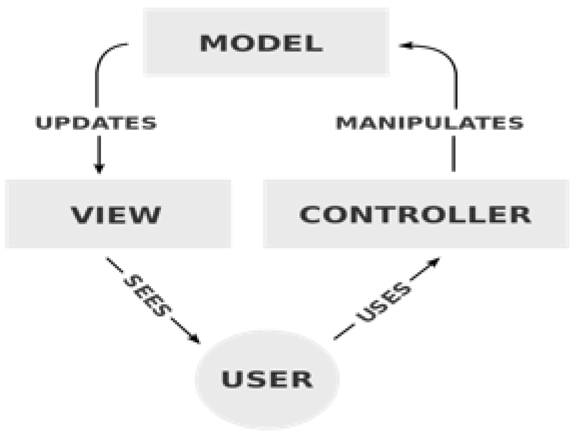


FIGURE 15. MVC process (Wikipedia 2014 c, cited 24.09.2014)

Model: The model is the architecture of data structures in the application. Briefly the model stores data that is to be accessed by the view and written to by the controller. For instance, if there has been a change in state, the model will notify associated views and controllers. So the views will produce updated output, and the controllers will change set of commands. (Ellislab 2014 b, cited 15.8.2014)

View: The view is to display the application's user interface (UI). Data in user interface is typically generated to the user by querying the model to get its data created from the model data. (Ellislab 2014 b, cited 15.8.2014)

Controller: The controller takes the user input and send commands to the model for updating the model's state. The controller is an interconnected bridge between the user and the model, and any other resources needed to process the HTTP request and generate a web page. (Ellislab 2014 b, cited 15.8.2014)

3.3 Analysis Current Coding System of OpixManager

Current OpixManager is designed and architected using CodeIgniter. In this thesis, OpixManager will run in localhost server, and the author will mainly apply NetBeans and Xampp to edit and test new user interface in localhost server.

OpixManager application flow chat as below:

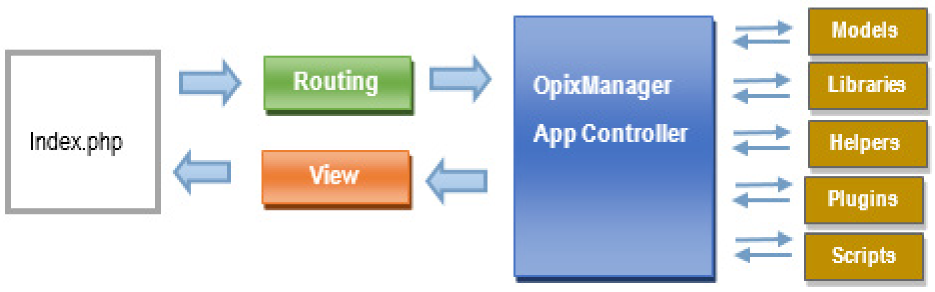
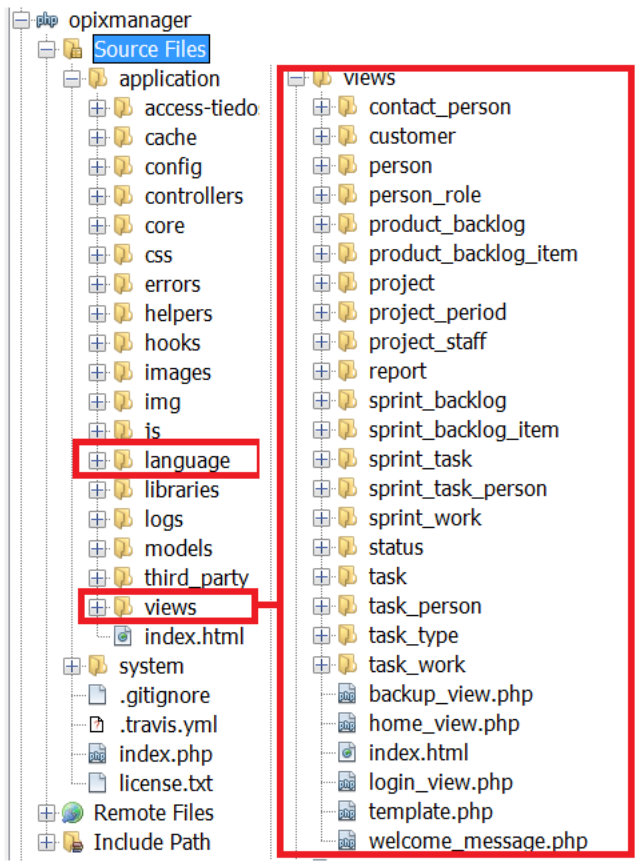


FIGURE 16. OpixManager application work chat

- The index.php serves as the front controller, initializing the base resources needed to run CodeIgniter.
- The Router examines the HTTP request to determine what should be done with it.
- The OpixManager controllers load the model, core libraries, plugins, helpers, and any other resources needed to process the specific request.
- The finalized view is rendered then sent to the web browser to be seen.

There are mainly four parts in directory structure of OpixManager's CodeIgniter framework (see details in figure 17):

- index.php receives all requests and routes to the right controllers classes and actions, parameters are included in the URL
- /system contains all CodeIgniter classes and libraries provided by the framework
- /application this is where application code is located, including the model, view and controller classes
- JavaScript files are in js folder in OpixManager application



```
opixmanager
  Source Files
    application
      access-tiedo
      cache
      config
      controllers
      core
      css
      errors
      helpers
      hooks
      images
      img
      js
      language
      libraries
      logs
      models
      third_party
      views
      index.html
    system
    .gitignore
    .travis.yml
    index.php
    license.txt
  Remote Files
  Include Path

views
  contact_person
  customer
  person
  person_role
  product_backlog
  product_backlog_item
  project
  project_period
  project_staff
  report
  sprint_backlog
  sprint_backlog_item
  sprint_task
  sprint_task_person
  sprint_work
  status
  task
  task_person
  task_type
  task_work
  backup_view.php
  home_view.php
  index.html
  login_view.php
  template.php
  welcome_message.php
```

FIGURE 17. Screenshot of OpixManager's directory structure

In regard to this thesis, primary modification and integration will be done in views and language files in order to apply new user interface for OpixManager. Views contains files that construct user interface structure of OpixManager as figure 17 highlights. Language files contain all the string in the user interface and are loaded to views files. The author will integrate Bootstrap 3 into template page, login page, and home page as well as their relevant link pages. General analysis of current code of template.php, login_view.php and home_view.php will be given in below paragraphs as well as connections with relevant models, and controllers. It will provide readers an overall view of how OpixManager constructs user interfaces, functions and where the author should edit code so as to execute new user interfaces properly.

The template.php constructs user interface template that is applied for all OpixManager's pages. It is also where Bootstrap template is mainly integrated into. There are `<head>`, `<nav>`, `<section>` and `<footer>` elements in template.php. Elements are also where the author will mainly modify codes in implementation. Figure 18 demonstrates the user interface of template page's navigation, log in and footer in OpixManager.

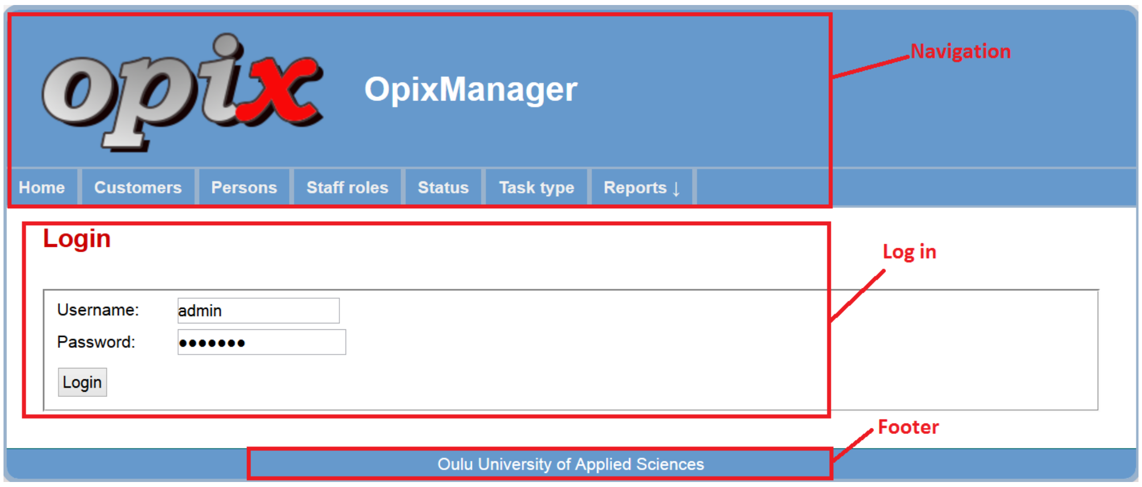
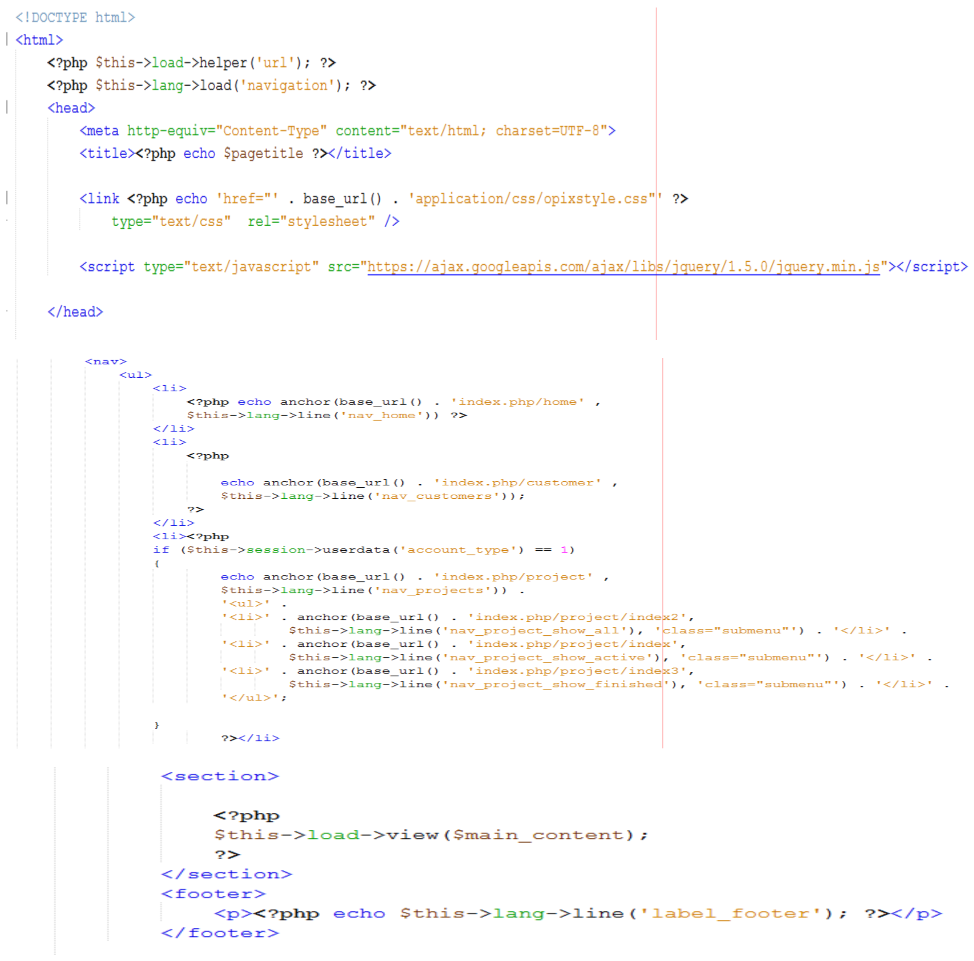


FIGURE 18. User interface of template page's navigation, log in and footer in OpixManager

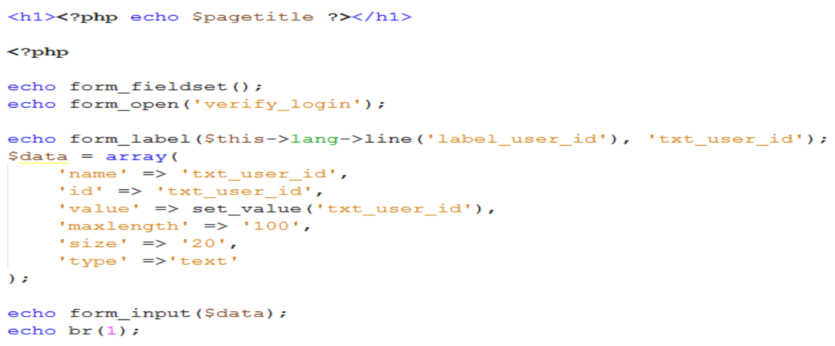
Figure 19 illustrates codes in `<head>`, `<nav>`, `<section>` and `<footer>` elements in template.php. CSS (opixstyle.css) and JavaScript are linked in `<head>` element. The `<nav>` element defines a set of navigation links (see figure 18), navigation texts are loaded from application/language/navigation_lang.php file: `$this->lang>line ('filename'),` navigation links are through `anchor (base_url (). 'index.php/home')` command. Projects, backup, sprint work and task work are displayed when the user successfully logs in by `if ($this->session->userdata ('logged_in'))` command. In `<section>` element, `$this->load->view ('$main_content')` loads current view and user interface from views/login_view.php. In `<footer>`, footer content is loaded from application/language/navigation_lang.php file.



```
<!DOCTYPE html>
<html>
    <?php $this->load->helper('url'); ?>
    <?php $this->lang->load('navigation'); ?>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title><?php echo $pagetitle ?></title>

        <link <?php echo 'href="' . base_url() . 'application/css/opixstyle.css"' ?>
            type="text/css"  rel="stylesheet" />

        <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.5.0/jquery.min.js"></script>

    </head>

        <nav>
            <ul>
                <li>
                    <?php echo anchor(base_url() . 'index.php/home' ,
                    $this->lang->line('nav_home')) ?>
                </li>
                <li>
                    <?php

                        echo anchor(base_url() . 'index.php/customer' ,
                        $this->lang->line('nav_customers'));
                    ?>
                </li>
                <li><?php
                if ($this->session->userdata('account_type') == 1)
                {
                        echo anchor(base_url() . 'index.php/project' ,
                        $this->lang->line('nav_projects')) .
                        '<ul>' .
                        '<li>' . anchor(base_url() . 'index.php/project/index2',
                                $this->lang->line('nav_project_show_all'), 'class="submenu"') . '</li>' .
                        '<li>' . anchor(base_url() . 'index.php/project/index',
                                $this->lang->line('nav_project_show_active'), 'class="submenu"') . '</li>' .
                        '<li>' . anchor(base_url() . 'index.php/project/index3',
                                $this->lang->line('nav_project_show_finished'), 'class="submenu"') . '</li>' .
                        '</ul>';

                }
                        ?></li>

                <section>

                    <?php
                    $this->load->view($main_content);
                    ?>
                </section>
                <footer>
                    <p><?php echo $this->lang->line('label_footer'); ?></p>
                </footer>
```

FIGURE 19. Codes of template page's navigation, log in and footer in OpixManager

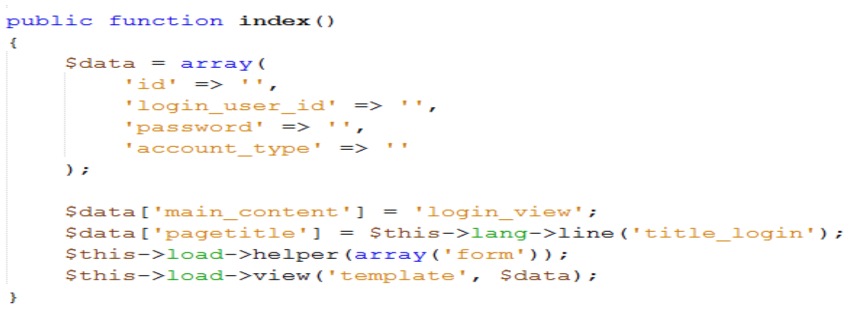
In figure 20, the login_view.php displays visual interface of login page. The login view is loaded by both login.php and verify_login.php controllers. Login labels are loaded from application/language/login_lang.php file: `$this->lang->line ('label_user_id')`. Input data is defined in $data array ().



```
<h1><?php echo $pagetitle ?></h1>

<?php

echo form_fieldset();
echo form_open('verify_login');

echo form_label($this->lang->line('label_user_id'), 'txt_user_id');
$data = array(
    'name' => 'txt_user_id',
    'id' => 'txt_user_id',
    'value' => set_value('txt_user_id'),
    'maxlength' => '100',
    'size' => '20',
    'type' =>'text'
);

echo form_input($data);
echo br(1);
```

FIGURE 20. Screenshot of login_view

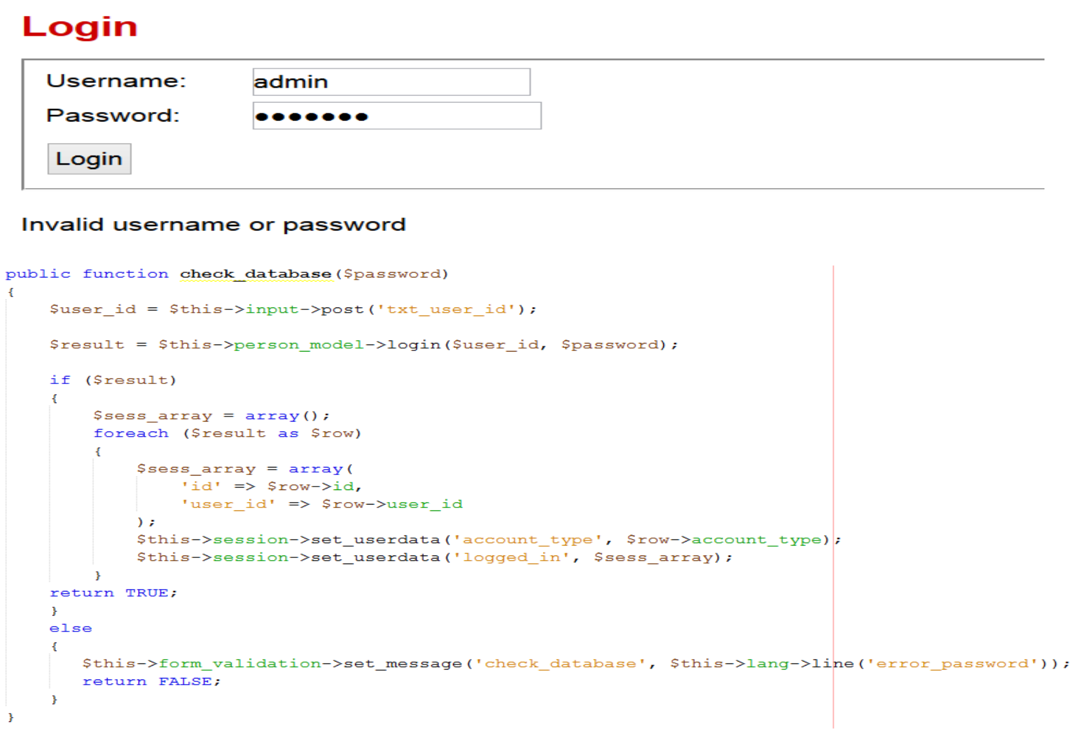
Below figure 21 shows the part of login controller's code. It includes function index () method to load page title from application/language/login_lang.php file: `$this->lang->line ('title_login')` and to call template in `$this->load->view ('template', $data)`.



```php
public function index()
{
    $data = array(
        'id' => '',
        'login_user_id' => '',
        'password' => '',
        'account_type' => ''
    );

    $data['main_content'] = 'login_view';
    $data['pagetitle'] = $this->lang->line('title_login');
    $this->load->helper(array('form'));
    $this->load->view('template', $data);
}
```

FIGURE 21. Screenshot of login controller method to show login_view

Verify_login controller loads person_model from application/models/person_model.php file. It contains function check database () method to send commands to database for checking whether username and password are correct ones to log in or not. If not, invalid username or password is displayed in login page (as figure 22).



Login

Username:	admin
Password:	●●●●●●●

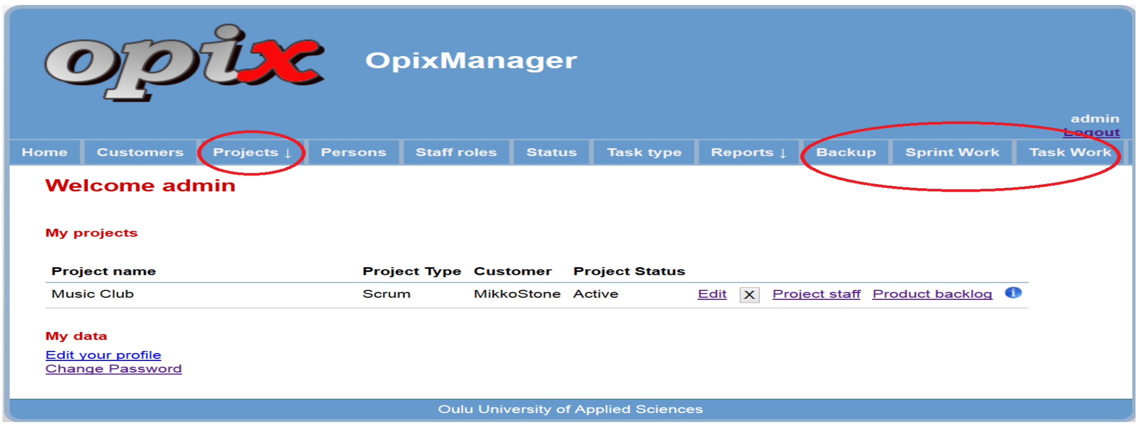
Login

Invalid username or password

```php
public function check_database($password)
{
    $user_id = $this->input->post('txt_user_id');

    $result = $this->person_model->login($user_id, $password);

    if ($result)
    {
        $sess_array = array();
        foreach ($result as $row)
        {
            $sess_array = array(
                'id' => $row->id,
                'user_id' => $row->user_id
            );
            $this->session->set_userdata('account_type', $row->account_type);
            $this->session->set_userdata('logged_in', $sess_array);
        }
    return TRUE;
    }
    else
    {
        $this->form_validation->set_message('check_database', $this->lang->line('error_password'));
        return FALSE;
    }
}
```

FIGURE 22. Screenshot of validation in verify_login controller

Projects, backup, sprint work, and task work are displayed in navigation bar when the user successfully logs in as figure 23 shows. Home_view.php constructs home page's user interface in `<table>` element and is loaded from home.php controller. Home page includes edit project, project staff, product backlog, edit your profile and change password linked pages (see more in appendix 1).



Welcome admin

My projects

Project name	Project Type	Customer	Project Status
Music Club	Scrum	MikkoStone	Active

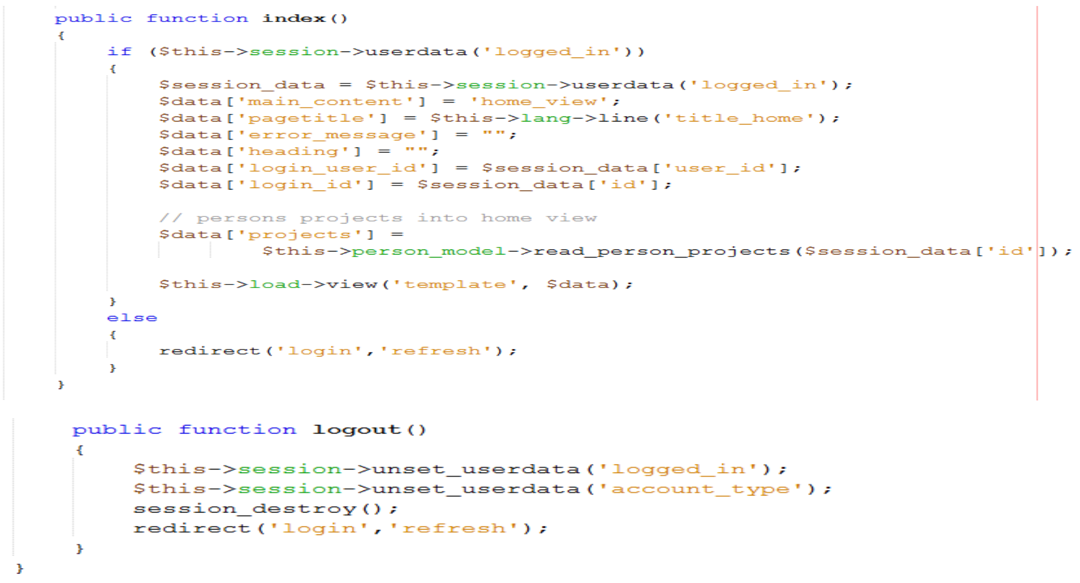
My data

Edit your profile

Change Password

FIGURE 23. Screenshot of home page and its linked pages

Home controller loads language package from application/language/home_lang.php and session library. It includes method to open home page and logout user (see figure 24).



```
    public function index()
    {
        if ($this->session->userdata('logged_in'))
        {
            $session_data = $this->session->userdata('logged_in');
            $data['main_content'] = 'home_view';
            $data['pagetitle'] = $this->lang->line('title_home');
            $data['error_message'] = "";
            $data['heading'] = "";
            $data['login_user_id'] = $session_data['user_id'];
            $data['login_id'] = $session_data['id'];

            // persons projects into home view
            $data['projects'] =
                    $this->person_model->read_person_projects($session_data['id']);

            $this->load->view('template', $data);
        }
        else
        {
            redirect('login','refresh');
        }
    }

    public function logout()
    {
        $this->session->unset_userdata('logged_in');
        $this->session->unset_userdata('account_type');
        session_destroy();
        redirect('login','refresh');
    }
}
```

FIGURE 24. Screenshot of method to open home page and logout users

4 DESIGN AND IMPLEMENT OPIXMANAGER

4.1 User Interface Design

Through analysis of current OpixManager's web design, there are few improvements needed to be considered in the design of new user interface. Colour and typography are out of modern. Layout is not so appearing to the user. And the website can not automatically respond to different web devices e.g. in mobile phone, in tablet. Base on above improvements, the author decides to choose Freelancer template that is partially integrated into OpixManager framework (see figure 25). Below there are reasons to select this template rather than others.

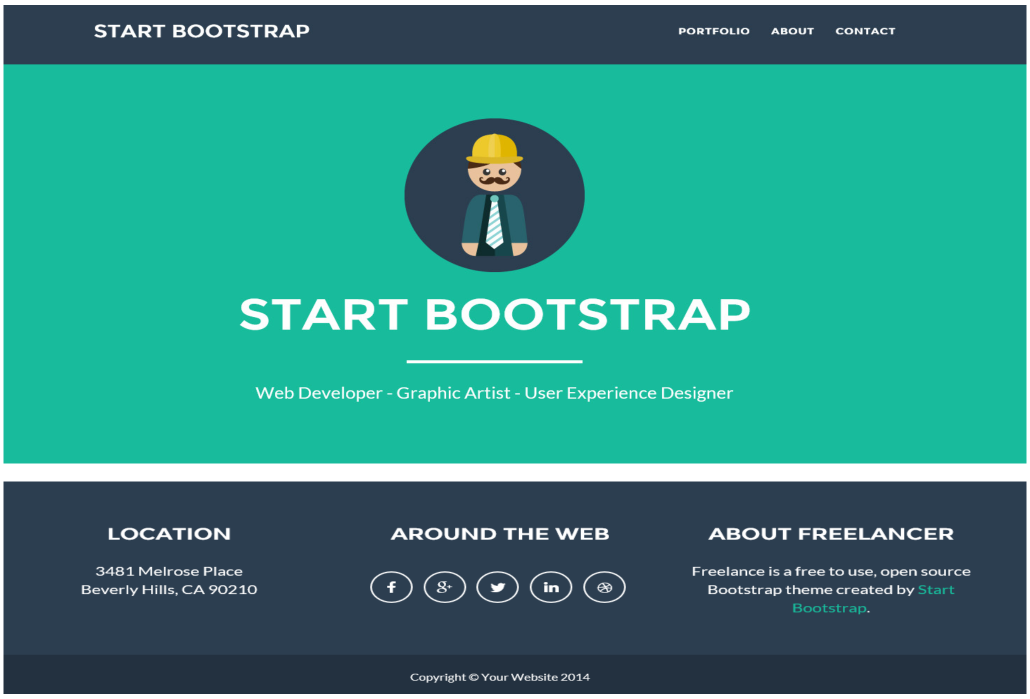

START BOOTSTRAP
PORTFOLIO
ABOUT
CONTACT
START BOOTSTRAP
Web Developer - Graphic Artist - User Experience Designer
LOCATION
3481 Melrose Place
Beverly Hills, CA 90210
AROUND THE WEB
ABOUT FREELANCER
Freelance is a free to use, open source Bootstrap theme created by Start Bootstrap.
Copyright © Your Website 2014

FIGURE 25. Screenshot of Freelancer template (StartBootstrap 2014, cited 28.9.2014)

Usability: Good web design should satisfy specific needs for targeted website users. In regard to OpixManager's website, common users are students and teachers in OAMK, both Finnish and English Information Technology Degree Programme. The purpose of OpixManager application is project management that includes both traditional and scrum projects. Hence, the choice of Freelancer template is to keep features of simplicity and directness in old user interface but present in more modern design. Freelancer template exactly fulfil those factors with simple design in navigation, footer and page content.

Typography: Font-family choice should fit the style of design. Font sizes should be consistent, large enough to be read, and proportioned so that headings and sub-headings stand out appropriately. In Freelancer template, Arial sans Serif fonts is easier to read online. Font size in body is 15px that suitable size for content, h1 size is 2em. Different point size keeps design streamlined. However, font-size 19px is rather too large for navigation bar that will be adjusted in actual implementation stage.

Colour: Balanced colour palette enhances visual experience. Normally a well-designed colour palette includes contrasted colour for the text and background, harmony background colours among navigation bar, page content and footer. In Freelancer template, navigation bar and footer background colour is hex ■#2c3e50 (very dark desaturated blue). Active/hover colour is hex 🟩#18bc9c (strong cyan.). Page content background colour is hex #ffffff (White). Body font-colour is ■#2c3e50 (very dark desaturated blue).

Navigation: Navigation function is to guide users finding information they are looking for in an easy, effective way. A logical page hierarchy, bread crumbs, designing clickable buttons are commonly used. In Freelancer's navigation bar, navigation titles appear as uppercase in horizontal hierarchy, and active show different colour from inactive ones. It is simple and users can easily navigate themselves to right information within few clicks. Furthermore, navigation bar collapses into its vertical mobile view when the viewport is narrower than defined min-width. Figure 26 shows an example of switching to mobile view.

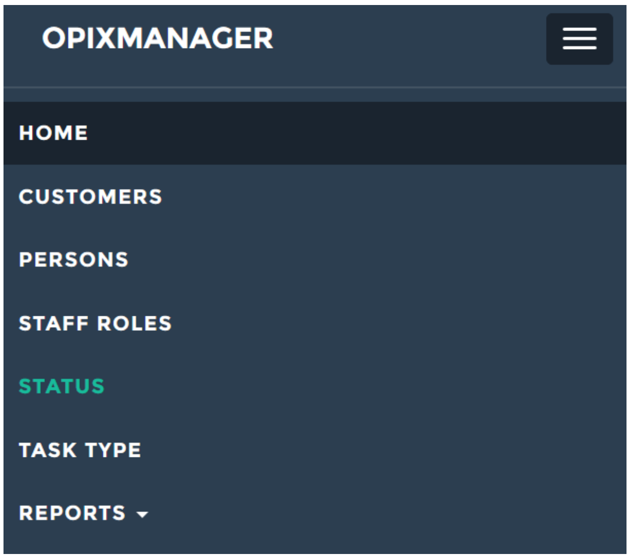


FIGURE 26. Screenshot of Freelancer's collapsed navigation bar

Responsive Grid layout: Grid system is responsively squeezed onto a tiny mobile device or stretched across a wide screen, all elements in the layout should resize their widths in relation to one another. Figure 27 shows an example of switching to mobile view.

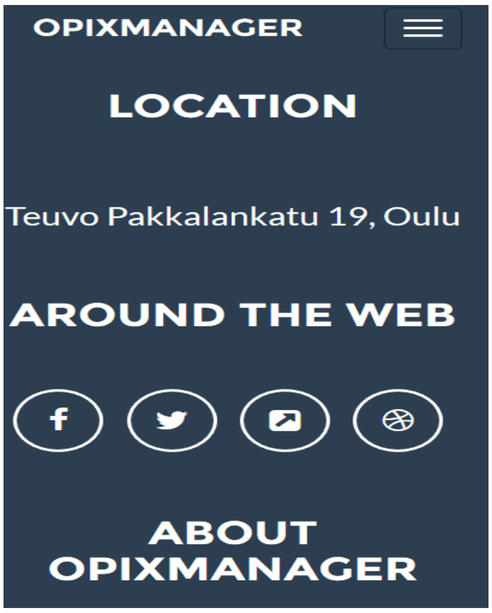
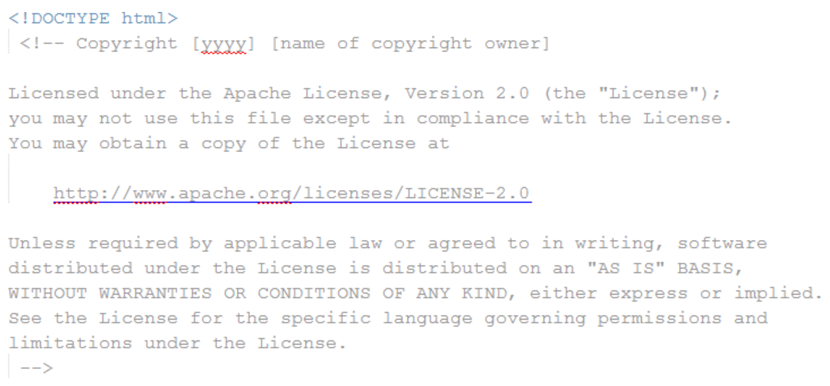


FIGURE 27. Screenshot of responsive grid layout

The next important question is whether Freelancer can be legally used in OpixManager or not, and how the author can legally apply in this thesis work. Freelancer template copyrights is owned by

Start Bootstrap and licence version is Apache 2.0. Apache licence states each Contributor is granted to "a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form."(Apache, cited 02.10.2014). In order to legally apply the Apache License to this thesis work, the author needs to attach a notice in OpixManager's integrated pages (show in figure 28).



```
<!DOCTYPE html>
<!-- Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
```

FIGURE 28. Copyrights of Freelancer template

4.2 Integrate Freelancer Template into OpixManager

In order to successfully integrate Freelancer template into OpixManager, the first thing is to ensure NetBeans IDE and Xampp are appropriately installed in own computer. OpxiManager is unzipped and copied into the folder C:\Xampp\htdocs\ and rename it as OpixManager-master for this project work implementation. The result is C:\Xampp\htdocs\OpixManager-master. A new database called opixmanager is required to create and import into Xampp MySQL server.

The next step is to download Freelancer's archive file and unzip. There are three main directories (CSS, Font, and JavaScript) that the author will integrate into OpixManager. The css folder contains three css files: Bootstrap.css, Bootstrap.min.css, and freelancer.css. Bootstrap.min.css is the minified version of Bootstrap.css, it is the core css file and must include in HTML pages. Freelancer.css is the additional css file that allowed editor to customize css by adding and overwriting extensible classes. It is also where the author will mainly edit custom css for OpixManager.

The font folder includes font icons called glyphicons, they are used in buttons, button groups for a toolbar, navigation, or form inputs within `<span>` tag. In js folder, the author will only use Bootstrap.js, Bootstrap.min.js, cbpAnimatedHeader.js, cbpAnimatedHeader.min.js, classie.js, Freelancer.js, jquery-1.11.0.js among nine .js files (see in figure 29). The Bootstrap.js and Bootstrap.min.js are core JavaScript. CbpAnimatedHeader.js, cbpAnimatedHeader.min.js, and classie.js are plugin JavaScript. Freelancer.js is allowed the author customizing JavaScript for OpixManager.

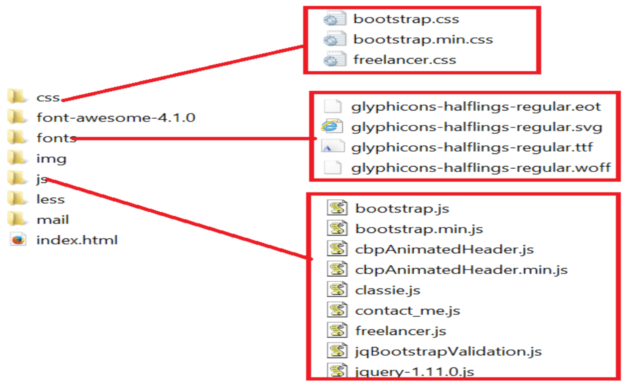


FIGURE 29. Screenshot of Freelancer folder

Furthermore, css files and font files(see in figure 30) in font-awesome-4.1.0 folder are also required to copy into both css and font folder in order to use font awesome's scalable icons in footer section.

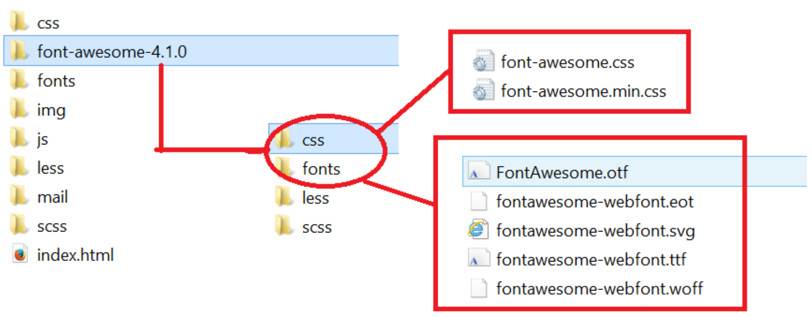
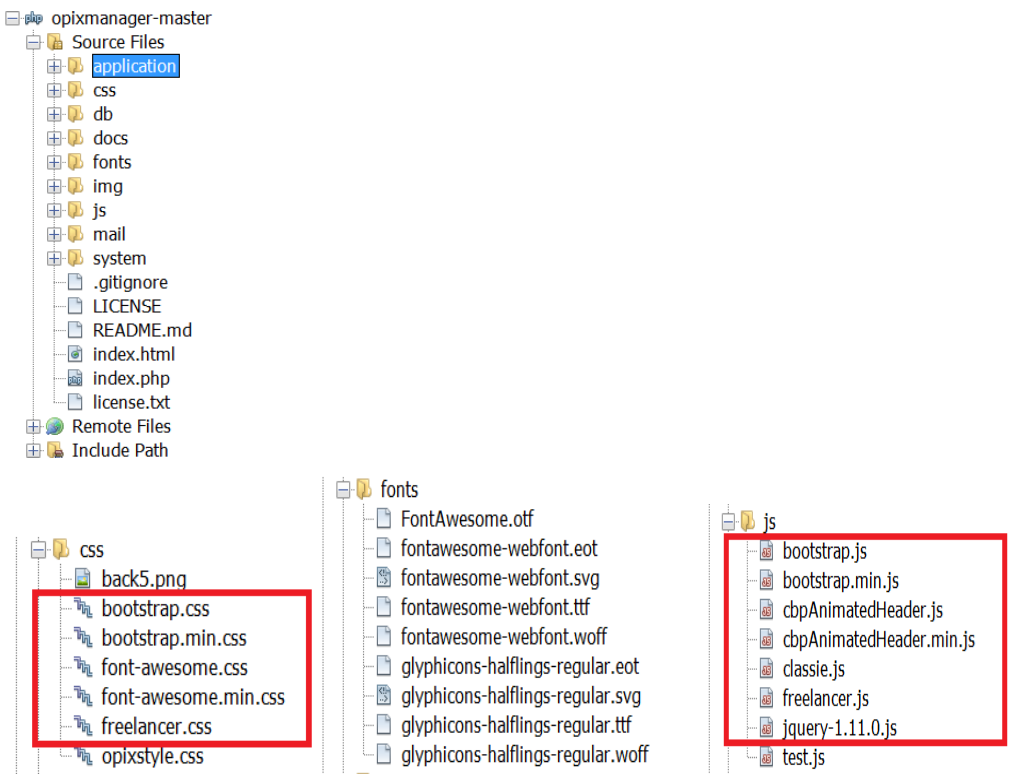


FIGURE 30. Screenshot of font-awesome-4.1.0 folder

After all necessary files in right folder, css, fonts and js folders should be copied into a new destination where OpixManager project locates: c:\Xampp\htdocs\OpixManager-master (as figure

31 shows). Until this stage, Freelancer template is successfully integrated into OpixManager-master and ready for modification.



```
opixmanager-master
  Source Files
    application
    css
    db
    docs
    fonts
    img
    js
    mail
    system
    .gitignore
    LICENSE
    README.md
    index.html
    index.php
    license.txt
  Remote Files
  Include Path
```

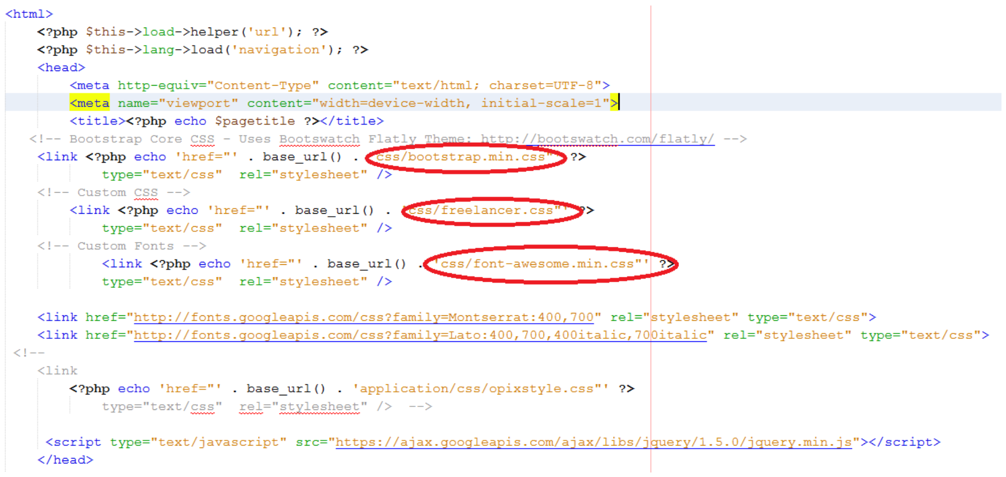
```
css
  back5.png
  bootstrap.css
  bootstrap.min.css
  font-awesome.css
  font-awesome.min.css
  freelancer.css
  opixstyle.css
```

```
fonts
  FontAwesome.otf
  fontawesome-webfont.eot
  fontawesome-webfont.svg
  fontawesome-webfont.ttf
  fontawesome-webfont.woff
  glyphicons-halflings-regular.eot
  glyphicons-halflings-regular.svg
  glyphicons-halflings-regular.ttf
  glyphicons-halflings-regular.woff
```

```
js
  bootstrap.js
  bootstrap.min.js
  cbpAnimatedHeader.js
  cbpAnimatedHeader.min.js
  classie.js
  freelancer.js
  jquery-1.11.0.js
  test.js
```

FIGURE 31. Screenshot of css, fonts and js folder in OpixManager-master

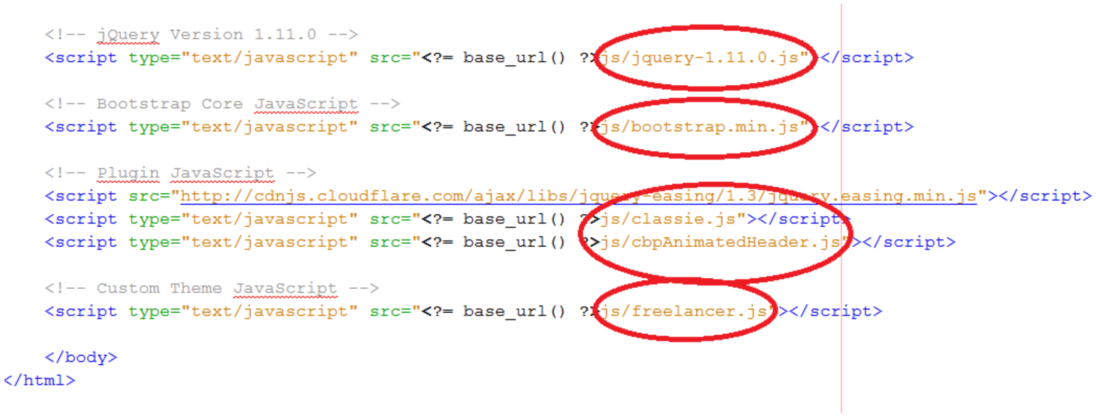
When all folders are available in OpixManager-master project, the author must start to load css and fonts into the `<head>` section of template page. As earlier paragraph describes, Bootstrap.min.css as core css, freelancer.css as custom css and font-awesome.min.css as custom fonts are linked to `<head>` section by `<link <?php echo 'href="'.base_url().'css/**.css"' ?> type="text/css" rel="stylesheet" />` command. The reason to load min.css instead of .css is because it is for actual application deployment online. Moreover, the author has commented original opixstyle.css just in case of overlapping commands (details in figure 31). Fonts are loaded from href links in `<link>` tag, for instance: `<link href="http://fonts.googleapis.com/css?family=Montserrat:400,700" rel="stylesheet" type="text/css">`. Details is in figure 32.



```
<html>
    <?php $this->load->helper('url'); ?>
    <?php $this->lang->load('navigation'); ?>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title><?php echo $pagetitle ?></title>
   <!-- Bootstrap Core CSS - Uses Bootswatch Flatly Theme: http://bootswatch.com/flatly/ -->
    <link <?php echo 'href="' . base_url() . 'css/bootstrap.min.css"' ?>
            type="text/css"  rel="stylesheet" />
    <!-- Custom CSS -->
        <link <?php echo 'href="' . base_url() . 'css/freelancer.css"' ?>
            type="text/css"  rel="stylesheet" />
    <!-- Custom Fonts -->
            <link <?php echo 'href="' . base_url() . 'css/font-awesome.min.css"' ?>
            type="text/css"  rel="stylesheet" />

    <link href="http://fonts.googleapis.com/css?family=Montserrat:400,700" rel="stylesheet" type="text/css">
    <link href="http://fonts.googleapis.com/css?family=Lato:400,700,400italic,700italic" rel="stylesheet" type="text/css">
 <!--
    <link
        <?php echo 'href="' . base_url() . 'application/css/opixstyle.css"' ?>
            type="text/css"  rel="stylesheet" />  -->

     <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.5.0/jquery.min.js"></script>
    </head>
```

FIGURE 32. Screenshot of linked CSS

The template.php should include .js files by adding the below code (see in figure 33) just before the `</body>` tag inside `<script>` tag. And it is very necessary to ensure that the jQuery library is added before "Bootstrap.min.js".



```
    <!-- jQuery Version 1.11.0 -->
    <script type="text/javascript" src="<?= base_url() ?>js/jquery-1.11.0.js"></script>

    <!-- Bootstrap Core JavaScript -->
    <script type="text/javascript" src="<?= base_url() ?>js/bootstrap.min.js"></script>

    <!-- Plugin JavaScript -->
    <script src="http://cdnjs.cloudflare.com/ajax/libs/jquery-easing/1.3/jquery.easing.min.js"></script>
    <script type="text/javascript" src="<?= base_url() ?>js/classie.js"></script>
    <script type="text/javascript" src="<?= base_url() ?>js/cbpAnimatedHeader.js"></script>

    <!-- Custom Theme JavaScript -->
    <script type="text/javascript" src="<?= base_url() ?>js/freelancer.js"></script>

    </body>
</html>
```

FIGURE 33. Screenshot of linked JavaScript

In addition, navigation bar is a prominent component of OpixManager application. Hence, the first modification in template.php is to write code so as to adopt old navigation items into new Bootstrap style. First of all, it is to add classes `.navbar, .navbar-default, navbar-fixed-top` and `role="navigation"` to the `<nav>` tag. The next step is to add a header class `.navbar-header, .page-scroll` to the `<div>` element. OpixManager text is included in an `<div>` element with class. `navbar-brand`, which give the text larger, outstanding appearance.

```
<nav class="navbar navbar-default navbar-fixed-top" role="navigation">
  <div class="container-fluid">
    <div class="navbar-brand">OpixManager</div>
      <div class="navbar-header page-scroll">
```

The following step is to add responsive features to this navigation bar, navigation titles and subtitles will be wrapped in a `<div>` with classes: `.collapse, .navbar-collapse, .navbar-right` defining align position, `.id` being consistent with `.data-target.` In `<button>` tag, classes`.navbar-toggle`, `.data-toggle, .data-target` are required to add into. Class`.data-toggle` is used to send commands to js files for button function, and `.data-target` indicates which element to toggle by #.id. Three classes of `.icon-bar` will toggle the elements that are in the `.nav-collapse <div>`. Because OpixManager contains 11 navigation items, font-size should be modified to 0.87em in Freelancer.css for better fitting. Below demonstrates partial code in OpixManager navigation as an example as well as changes in Freelancer.css.

```
<button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#bs-example-navbar-collapse-1">
 <span class="sr-only">Toggle navigation</span>
<span class="icon-bar"></span>
<div class="navbar-collapse collapse navbar-right″ id="bs-example-
navbar-collapse-1">
<ul class="nav navbar-nav navbar-right">
  <li class="active"><?php echo anchor(base_url() . 'index.php/home′,
$this->lang->line('nav_home'));
?>
   </li>

.navbar-collapse
{
    font-size: 0.87em;
}
```

When navigation coding is completed, below figure 34 illustrates vertical mobile view and horizontal non-mobile view of new OpixManager's navigation bar.

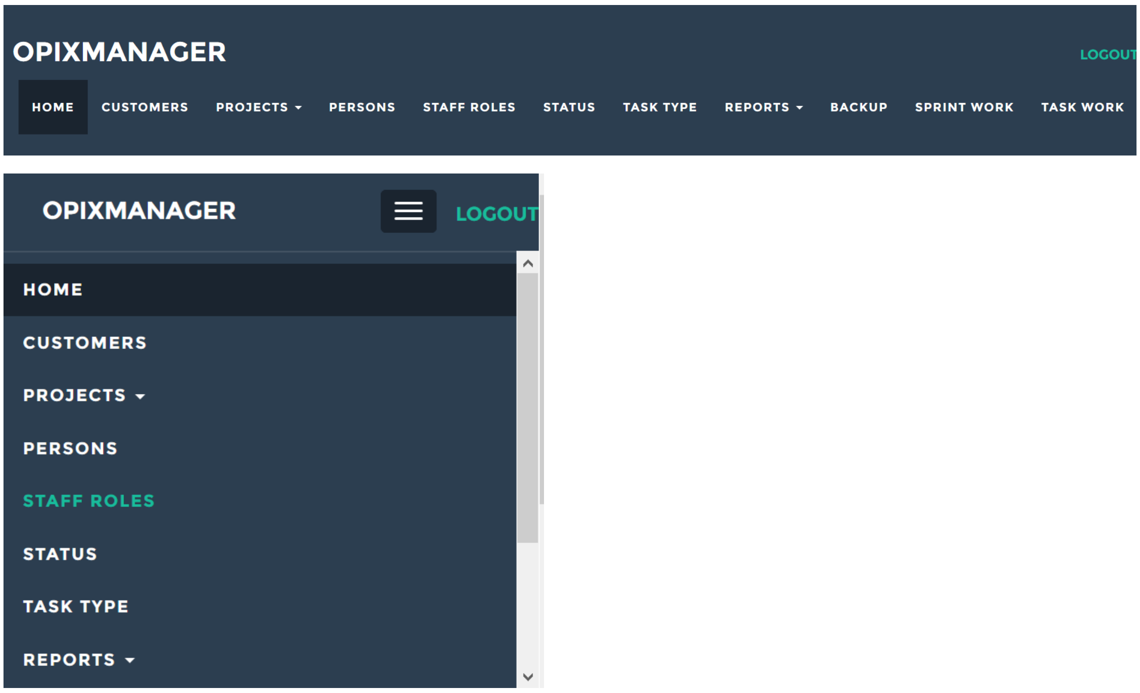


FIGURE 34. Screenshot of navigation bar in both mobile and non-mobile view

During implementation, the author finds out that Freelancer template lacks of `.dropdown-submenu` component, which requires in report menu item. Figure 35 shows the screenshot of new dropdown and dropdown submenu.

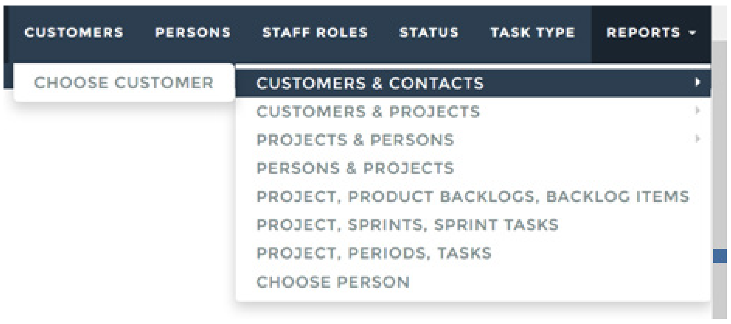
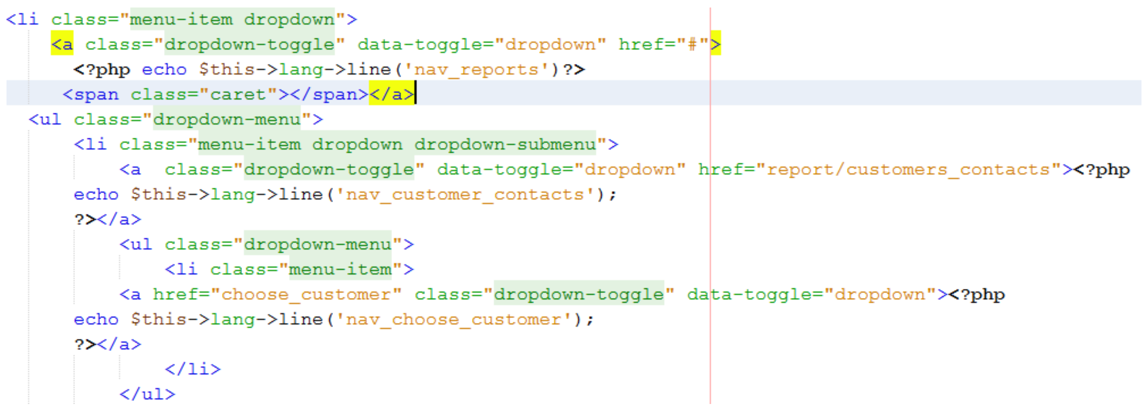


FIGURE 35. Screenshot of dropdown and dropdown-submenu

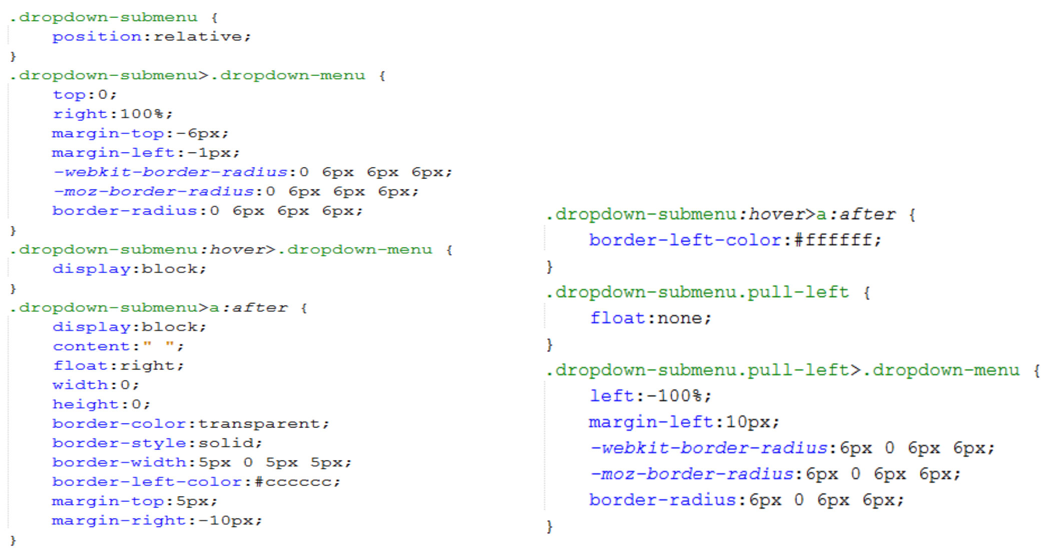
Dropdown items are wrapped in a `<li>` with classes: `.menu-item, .dropdown.` In `<a>` tag, classes `.dropdown-toggle, .data-toggle="dropdown" are` required to add for toggling a dropdown. The next step is to add a submenu class `.menu item,.dropdown,.dropdown-submenu` to the `<li>` element.



```
<li class="menu-item dropdown">
   <a class="dropdown-toggle" data-toggle="dropdown" href="#">
     <?php echo $this->lang->line('nav_reports')?>
    <span class="caret"></span></a>
  <ul class="dropdown-menu">
      <li class="menu-item dropdown dropdown-submenu">
          <a  class="dropdown-toggle" data-toggle="dropdown" href="report/customers_contacts"><?php
      echo $this->lang->line('nav_customer_contacts');
      ?></a>
          <ul class="dropdown-menu">
              <li class="menu-item">
          <a href="choose_customer" class="dropdown-toggle" data-toggle="dropdown"><?php
      echo $this->lang->line('nav_choose_customer');
      ?></a>
              </li>
          </ul>
```

FIGURE 36. Screenshot of dropdown and dropdown-submenu

After that, the author also has to insert relevant code in Freelancer.css to construct `.dropdown-submenu` user interface. Details are in figure 37.

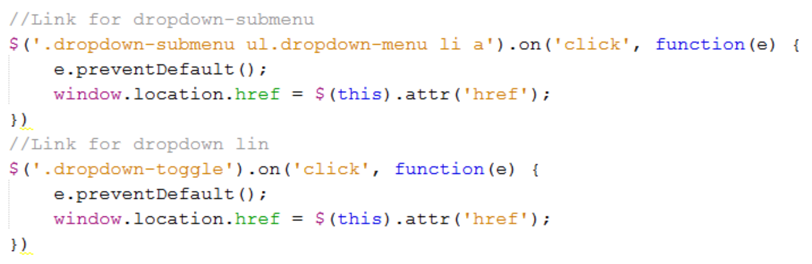


```
.dropdown-submenu {
    position:relative;
}
.dropdown-submenu>.dropdown-menu {
    top:0;
    right:100%;
    margin-top:-6px;
    margin-left:-1px;
    -webkit-border-radius:0 6px 6px 6px;
    -moz-border-radius:0 6px 6px 6px;
    border-radius:0 6px 6px 6px;
}
.dropdown-submenu:hover>.dropdown-menu {
    display:block;
}
.dropdown-submenu>a:after {
    display:block;
    content:" ";
    float:right;
    width:0;
    height:0;
    border-color:transparent;
    border-style:solid;
    border-width:5px 0 5px 5px;
    border-left-color:#cccccc;
    margin-top:5px;
    margin-right:-10px;
}
```

```
.dropdown-submenu:hover>a:after {
    border-left-color:#ffffff;
}
.dropdown-submenu.pull-left {
    float:none;
}
.dropdown-submenu.pull-left>.dropdown-menu {
    left:-100%;
    margin-left:10px;
    -webkit-border-radius:6px 0 6px 6px;
    -moz-border-radius:6px 0 6px 6px;
    border-radius:6px 0 6px 6px;
}
```

FIGURE 37. Inserted css code of .dropdown-submenu

Furthermore, the author has added function method in Freelancer.js to enable submenu links by clicking (details in figure 38).

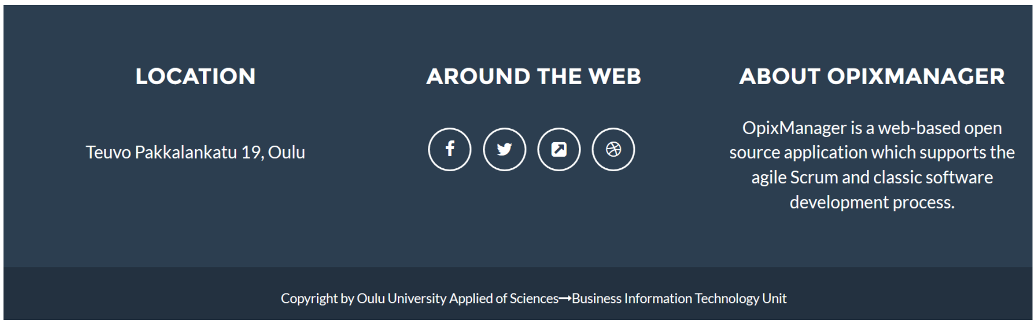


```
//Link for dropdown-submenu
$('.dropdown-submenu ul.dropdown-menu li a').on('click', function(e) {
    e.preventDefault();
    window.location.href = $(this).attr('href');
})
//Link for dropdown lin
$('.dropdown-toggle').on('click', function(e) {
    e.preventDefault();
    window.location.href = $(this).attr('href');
})
```

FIGURE 38. Inserted js code of .dropdown-submenu

Footer contents are contained in `<footer>` element. There are three parts in footer (see in figure 39), each part is located by `.col-md-4` in `<div>` element. In second part, font-awesome icons are used for link buttons by `class="fa fa-fw"` in `<i>` element. For example:

```
<div class="footer-col col-md-4">
<li><a href="https://www.facebook.com/oamk.liike" class="btn-social
btn-outline" target="_blank"><i class="fa fa-fw fa-facebook"></i></a>
 </li>
```



LOCATION

Teuvo Pakkalankatu 19, Oulu

AROUND THE WEB

ABOUT OPIXMANAGER

OpixManager is a web-based open source application which supports the agile Scrum and classic software development process.

Copyright by Oulu University Applied of Sciences→Business Information Technology Unit

FIGURE 39. Screenshot of footer section

4.3 Implementation in Login Page, Home Page

In login page implementation, the first step is define `.form-control` max-width to 25em in Freelancer.css. It is calculated in 16px=1em, 400/16=25em formula. 25em is to define maximum width of form control, and page content is to appear in vertical proper alignment while screen switches to mobile view size. Figure 40 demonstrates screenshot of new login page.

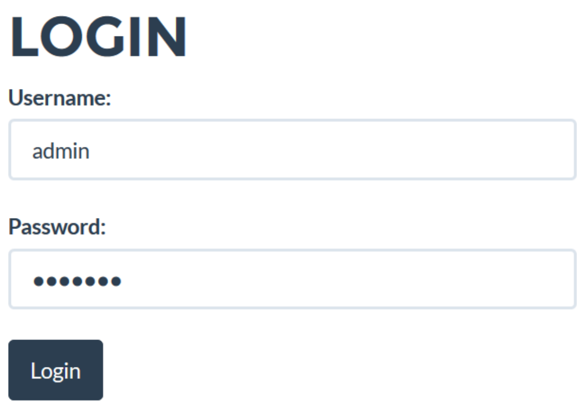
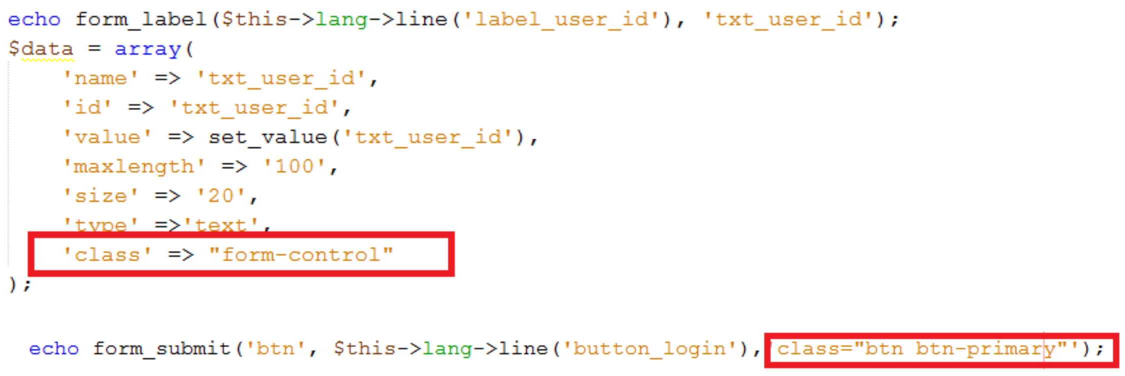


FIGURE 40. Screenshot of login page

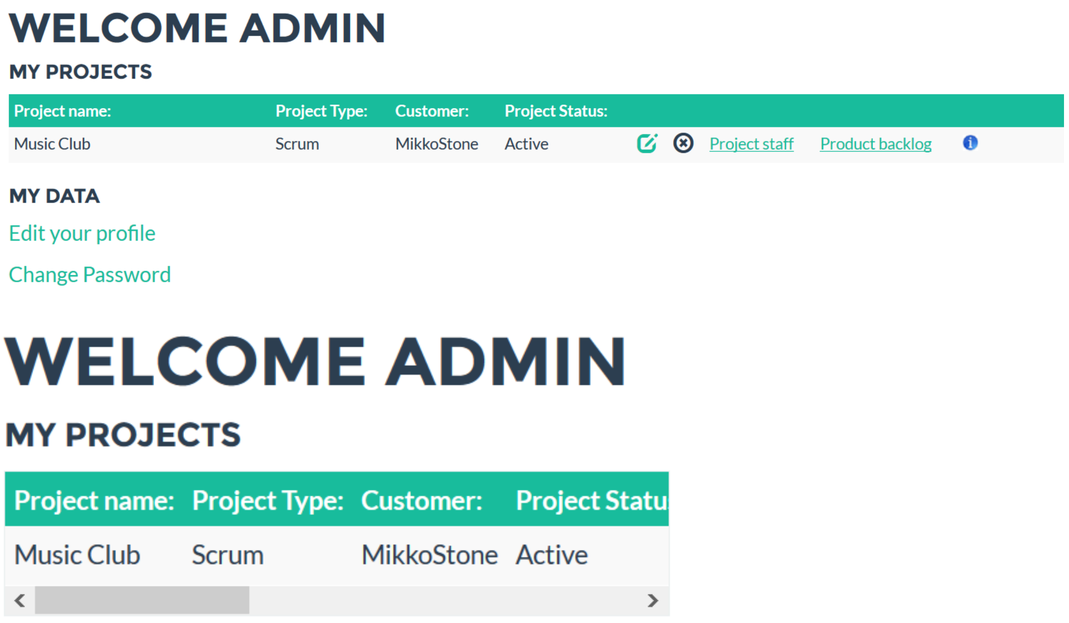
After that in login_view.php, class `'class' => "form-control"` is added in $data array () for applying new defined input-form. `'class="btn btn-primary"'` is inserted in `echo form_submit` in order to create a primary style button replacing the old one(see in figure 41).



```
echo form_label($this->lang->line('label_user_id'), 'txt_user_id');
$data = array(
    'name' => 'txt_user_id',
    'id' => 'txt_user_id',
    'value' => set_value('txt_user_id'),
    'maxlength' => '100',
    'size' => '20',
    'type' =>'text',
    'class' => "form-control"
);

  echo form_submit('btn', $this->lang->line('button_login'),'class="btn btn-primary"');
```

FIGURE 41. Screenshot of login page's modified code

In home page implementation, glyphicon size is modified by add `span.glyphicon {font-size: 1.3em ;}` to Freelancer.css. Figure 42 demonstrates screenshot of new home page. Other relevant linked pages in home page are applied in similar implementation, appendix 2 shows screenshots of linked pages in the new user interface.


WELCOME ADMIN

MY PROJECTS

Project name:	Project Type:	Customer:	Project Status:					
Music Club	Scrum	MikkoStone	Active			Project staff	Product backlog	

MY DATA

Edit your profile

Change Password

WELCOME ADMIN

MY PROJECTS

Project name:	Project Type:	Customer:	Project Statu
Music Club	Scrum	MikkoStone	Active

FIGURE 42. Screenshot of home page

Responsive Bootstrap table form is integrated by adding `.table-responsive` in `<div>` element, and classes: `.table-stripped`, `.table-condensed` are inserted into `<table>` tag. The first row is emphasized in success coloring by adding `class="success"`. It is also to apply edit glyphicon in `<span class="glyphicon glyphicon-edit"></span>` and delete glyphicon in `<span class="glyphicon glyphicon-remove-circle"></span>`. Details are in figure 43.

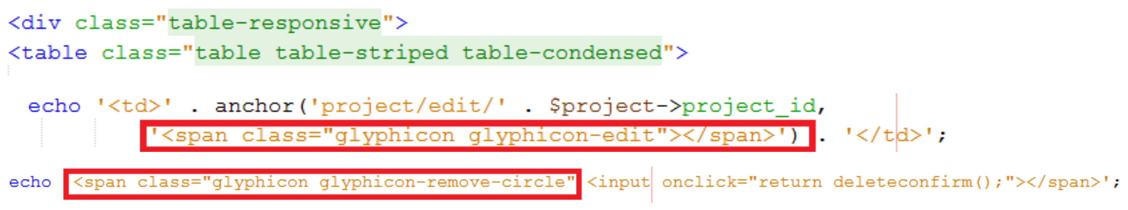
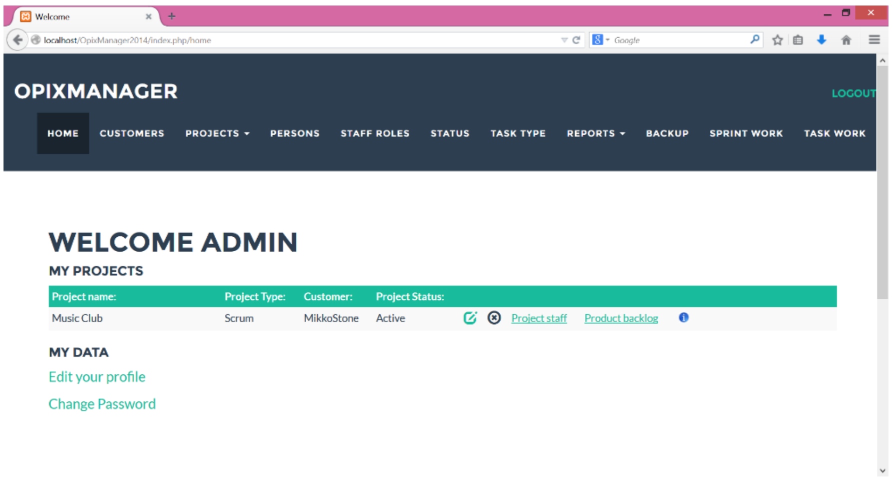

```
<div class="table-responsive">
<table class="table table-striped table-condensed">

  echo '<td>' . anchor('project/edit/' . $project->project_id,
            '<span class="glyphicon glyphicon-edit"></span>') . '</td>';
echo '<span class="glyphicon glyphicon-remove-circle" <input onclick="return deleteconfirm();"></span>';
```

FIGURE 43. Screenshot of home page's modified code

4.4 Cross-Browser Testing

When new user interface has completely applied for template page, login page and home page, the author will run cross-browser testing in order to make sure new applied user interface running in different browser. Google Chrome, Mozilla Firefox and Internet Explorer are the main web browser used in 2014. The author can choose above three browsers in the NetBeans to run in localhost. Below are screenshot of in Chrome (figure 44), Internet Explorer (figure 45), and in Mozilla Firefox (figure 46).



WELCOME ADMIN

MY PROJECTS

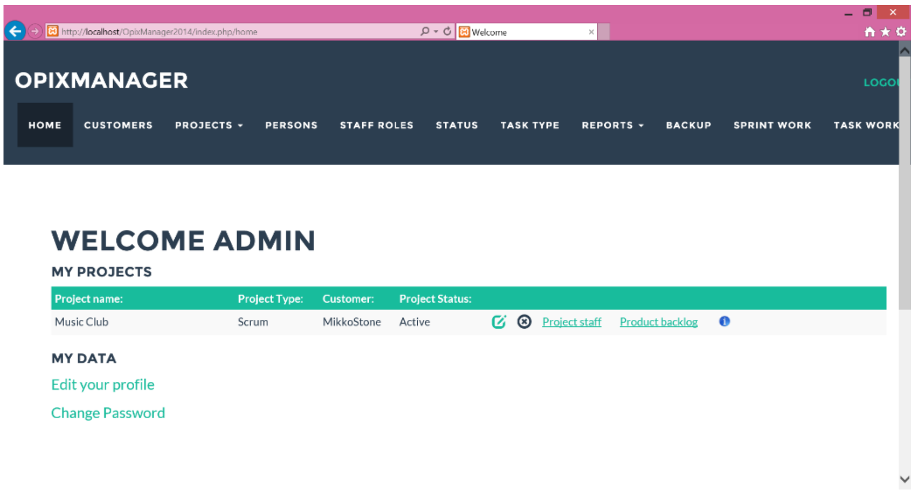
Project name:	Project Type:	Customer:	Project Status:					
Music Club	Scrum	MikkoStone	Active			Project staff	Product backlog	

MY DATA

Edit your profile

Change Password

FIGURE 44. Screenshot of Chrome



WELCOME ADMIN

MY PROJECTS

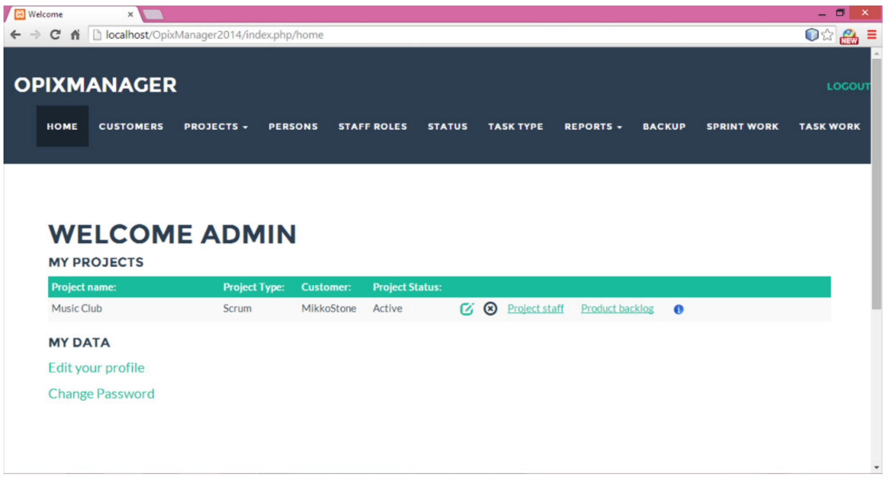
Project name:	Project Type:	Customer:	Project Status:					
Music Club	Scrum	MikkoStone	Active			Project staff	Product backlog	

MY DATA

Edit your profile

Change Password

FIGURE 45. Screenshot of Internet Explorer



WELCOME ADMIN

MY PROJECTS

Project name:	Project Type:	Customer:	Project Status:				
Music Club	Scrum	MikkoStone	Active		Project staff	Product backlog	

MY DATA

Edit your profile

Change Password

FIGURE 46. Screenshot of Mozilla Firefox

As above figures show that new user interfaces of template page, login page and home page are running properly in Google Chrome, Mozilla Firefox and Internet Explorer browsers.

5 Conclusion

As mentioned in earlier introduction chapter, the main research problem of this thesis is to study and integrate Bootstrap 3 into OpixManager. The question has been answered through applying theoretical framework for actual implementation in empirical part.

In chapter 2 and 3, the theoretical framework of this thesis has built up by studying what is responsive web design, what components are included using Bootstrap 3 and how OpixManager CodeIgniter that based on MVC framework interrelates among the model, the view and the controller. Through theoretical framework, the author has knowledge how to integrate Bootstrap template to OpixManager's user interface and where to edit code.

In chapter 4, Freelancer template has been chosen as integrated template in OpixManager because of balanced colour palette, simple layout and satisfied usability. Due to limitation of work time, template has been integrated only in template page, login page and home page as well its linked pages. CSS, Font and JS files should be properly imported in template.php. Editing and modifying code has been done in template.php, login_view.php, and home_view.php. Final results has been tested in cross browser through localhost server. For instance: Mozilla Firefox, Internet Explorer and Chrome.

However, two major obstacles which have affected the result of implementation. First of all, compatibility problem has occurred during integration Freelancer template base on HTML, CSS into OpixManager's CodeIgniter based on PHP. More explanation will be in discussion chapter. Secondly there are really limited open resources available about integration Bootstrap into CodeIgniter, hence the author has spent quite lots of time to solve problems without useful tutorials.

All in all, the final implementation is quite satisfactory. Actual execution proves Freelancer is suitable Bootstrap template for integrating in OpixManager. User interface is simple and modern, it is compatible with all major browsers, for example Internet Explorer, Firefox, and Chrome, and adopted a mobile first design with emphasizing responsive design by default.

6 Discussion

The thesis work has started at the middle of May. Due to summer break, the progress of thesis has slowed down for two months. The whole project took around 4 months for both theoretical and empirical parts.

This work gave me the opportunity to learn Bootstrap and CodeIgniter, and how to integrate Bootstrap template into OpixManager application. The author does not have any experience on open-source front-end framework before that. Especially Bootstrap 3 is very popular responsive HTML5 frameworks nowadays, the author is interested in front-end coding and very glad to have this opportunity to learn and familiarize Bootstrap framework.

The biggest challenge that the author faced in this thesis work is compatibility problem occurring during integration Freelancer template base HTML, CSS into OpixManager's CodeIgniter based on PHP. Due to limited available resources about integration Bootstrap into CodeIgniter in the Internet, the author has spent quite lots of time on solving unexpected problem. For instance, login in method could not be appropriately called because of linking wrong external JavaScript file in template.php. And PHP method could not run properly inside `<div>` tag or glyphicons could not be loaded in `<span>` tag. Therefore, hopefully there are more references and tutorials available for future editor and potential sections in those implemented pages can be improved.

I have learnt a lot about how to solve frequent problems of Bootstrap integration, and how to fully use limited resources for instance, stack overflow is very useful forum where users can freely upload code problems and people are willing to provide suggested answers. Stack overflow helps the author a lot during actual implementation. This project has given me an opportunity to test my skills. The final execution is rather satisfactory because of thesis tutor and teachers help.

REFERENCES

Adrian. Responsive Web Design: 50 Examples and Best Practices.Cited 25.5.2014, http://designmodo.com/responsive-design-examples/

Apache 2004. Apache License, Version 2.0. Cited 02.10.2014, http://www.apache.org/licenses/

Beresford, Toby. 2008. Benefits of the CodeIgniter Framework. Cited 30.5.2014, http://www.slideshare.net/tobyberesford/benefits-of-the-code-igniterframework

Bootstrap 3 Homepage 2014. Cited 30.5.2014, http://getBootstrap 3.com/getting-started/

Butler, Tom. 2010. Model-View-Controller. Cited 24.9.2014, https://r.je/mvc-in-php.html

David, Natalia. 2013. 11 Reasons to Use Twitter Bootstrap 3. Cited 30.5.2014, http://www.sitepoint.com/11-reasons-to-use-twitter-bootstrap/

Ellislab 2014 a. Cited 15.8.2014, https://ellislab.com/CodeIgniter

Ellislab 2014 b. Model-View-Controller. Cited 15.8.2014, https://ellislab.com/CodeIgniter/userguide/overview/mvc.html

Gube, Jacob. 2013. 10 Best Responsive HTML5 Frameworks and Tools. Cited 26.5.2014, http://designinstruct.com/roundups/html5-frameworks/

Knight, Kayla. 2011. Responsive Web Design: What It Is and How To Use It. Cited 26.5.2014, http://www.smashingmagazine.com/2011/01/12/guidelines-for-responsive-web-design/

Macronimous 2014. Cited 30.5.2014, http://www.macronimous.com/resources/using_CodeIgniter_for_PHP_application_development.asp

Marcotte, Ethan. 2009 a. Fluid Grids. Cited 29.5.2014, http://alistapart.com/article/fluidgrids

Marcotte, Ethan. 2009 b. Fluid Images. Cited 29.7.2014, http://alistapart.com/article/fluid-images

OpixProject Homepage 2014. Cited 27.5.2014, http://opixproject.opiskelijaprojektit.net/index.php/opixproject/general

Pettit,Nick. 2012. Beginner's Guide to Responsive Web Design. Cited 28.5.2014, http://blog.teamtreehouse.com/beginners-guide-to-responsive-web-design

Profound grid 2014. A responsive grid system. Cited 06.11.2014, http://www.profoundgrid.com/

Scrum Overview for Agile Software Development. Cited 28.5.2014, http://www.mountaingoatsoftware.com/agile/scrum/overview/

StartBootstrap 2014. Freelancer Template. Cited 28.9.2014, http://startBootstrap.com/templates/freelancer/

Storey, Dudley. 2014. CSS Fluid Image Techniques for Responsive Site Design. Cited 30.07.2014, http://demosthenes.info/blog/586/CSS-Fluid-Image-Techniques-for-Responsive-Site Design

Wikepedia 2014 a. CodeIgniter. Cited 30.5.2014, http://en.wikipedia.org/wiki/CodeIgniter

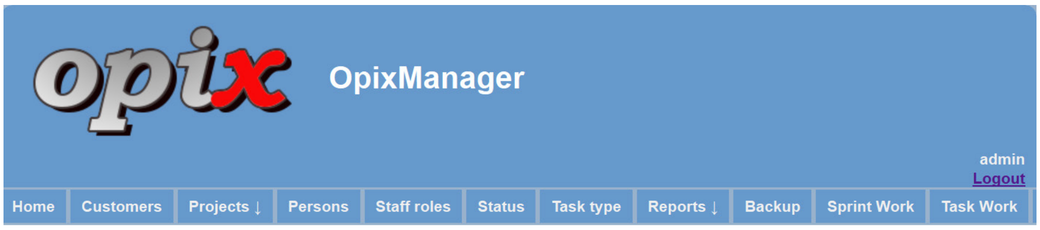
Wikepedia 2014 b. Project managementr. Cited 24.9.2014, http://en.wikipedia.org/wiki/Project_management

Wikipedia 2014 c. Model–view–controller. Cited 24.9.2014, http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

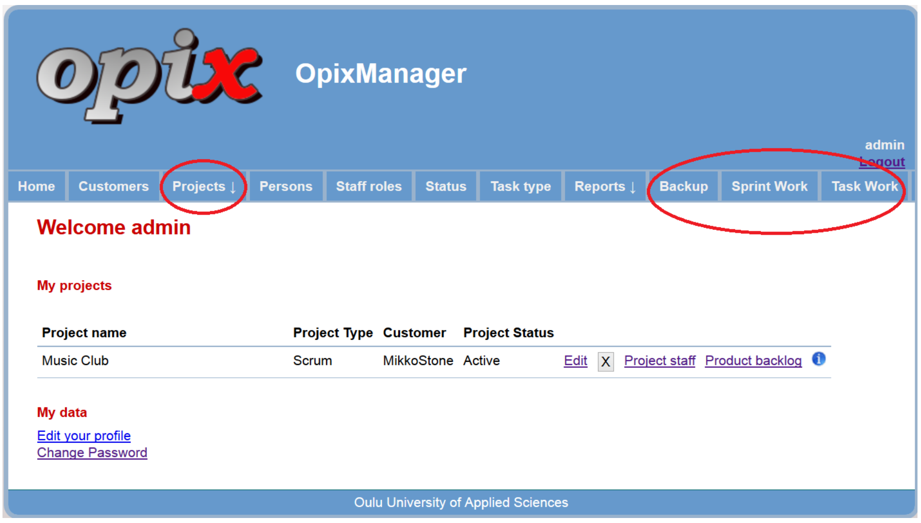
Wikibook 2014. Model–view-controller. Cited 24.9.2014, http://en.wikibooks.org/wiki/Computer_Science_Design_Patterns/Model%E2%80%93 ew%E280%93controller

APPENDICES

Screenshots of linked pages in old user interface



Navigation bar



Welcome admin

My projects

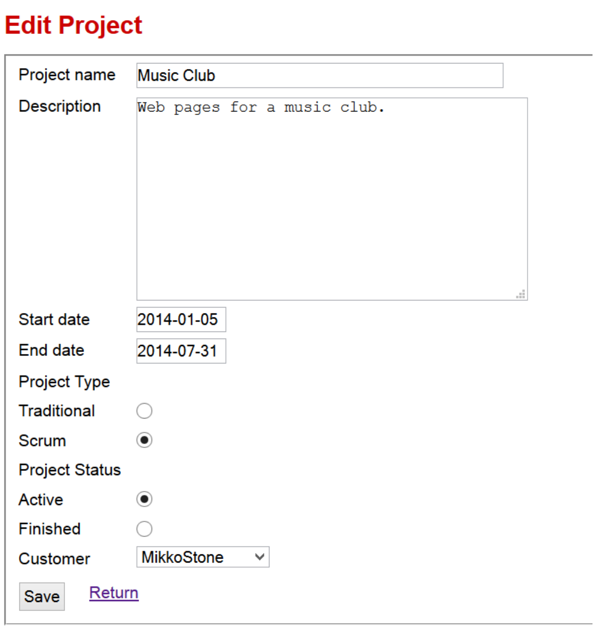
Project name	Project Type	Customer	Project Status					
Music Club	Scrum	MikkoStone	Active	Edit	X	Project staff	Product backlog	ℹ

My data

Edit your profile
Change Password

Oulu University of Applied Sciences

Home page

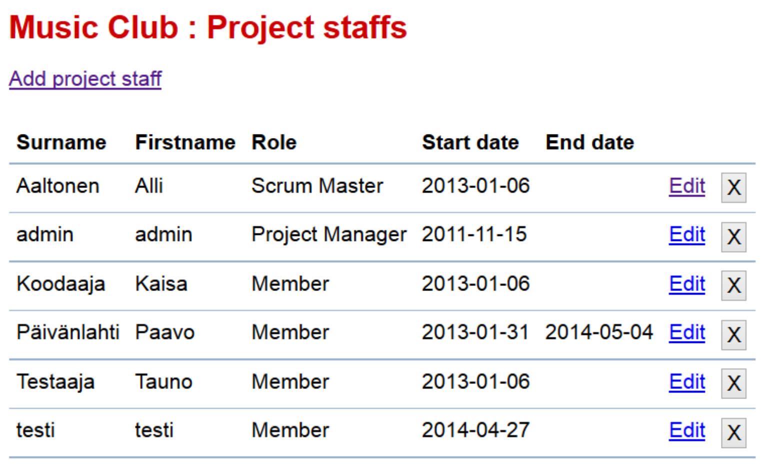


Edit Project

Project name	Music Club
Description	Web pages for a music club.
Start date	2014-01-05
End date	2014-07-31
Project Type	
Traditional	○
Scrum	●
Project Status	
Active	●
Finished	○
Customer	MikkoStone

Save Return

Edit project page

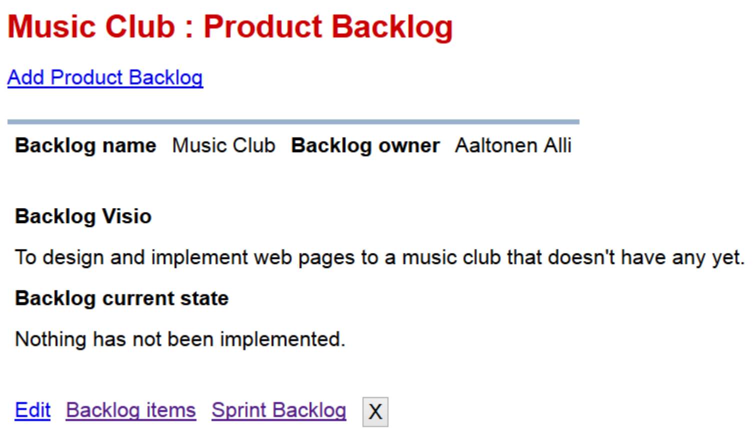


Music Club : Project staffs

Add project staff

Surname	Firstname	Role	Start date	End date		
Aaltonen	Alli	Scrum Master	2013-01-06		Edit	X
admin	admin	Project Manager	2011-11-15		Edit	X
Koodaaja	Kaisa	Member	2013-01-06		Edit	X
Päivänlahti	Paavo	Member	2013-01-31	2014-05-04	Edit	X
Testaaja	Tauno	Member	2013-01-06		Edit	X
testi	testi	Member	2014-04-27		Edit	X

Project staff page



Music Club : Product Backlog

Add Product Backlog

Backlog name Music Club **Backlog owner** Aaltonen Alli

Backlog Visio

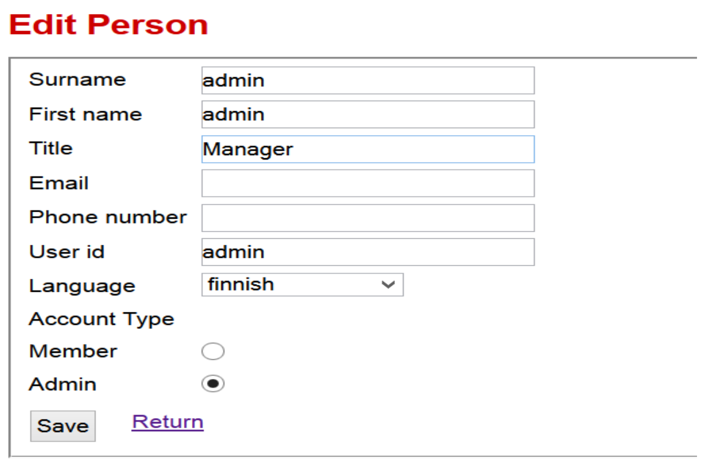
To design and implement web pages to a music club that doesn't have any yet.

Backlog current state

Nothing has not been implemented.

Edit Backlog items Sprint Backlog X

Product backlog page

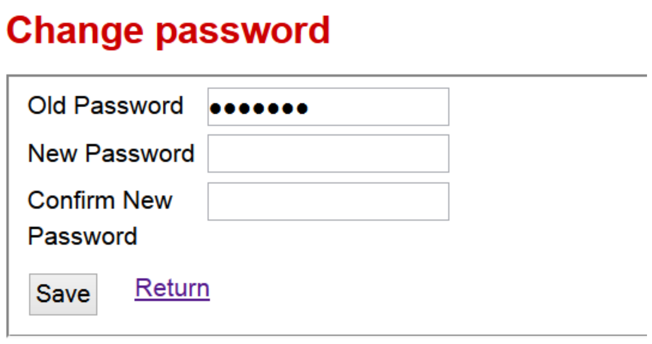


Edit Person

Surname	admin
First name	admin
Title	Manager
Email	
Phone number	
User id	admin
Language	finnish
Account Type	
Member	○
Admin	●

Save Return

Edit your profile page

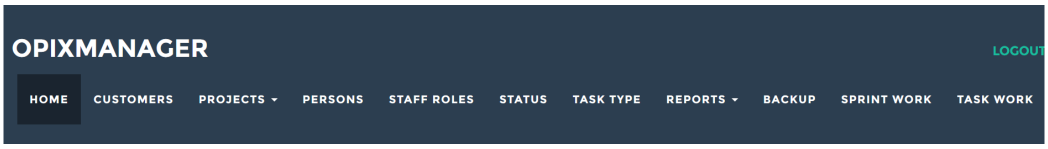


Change password

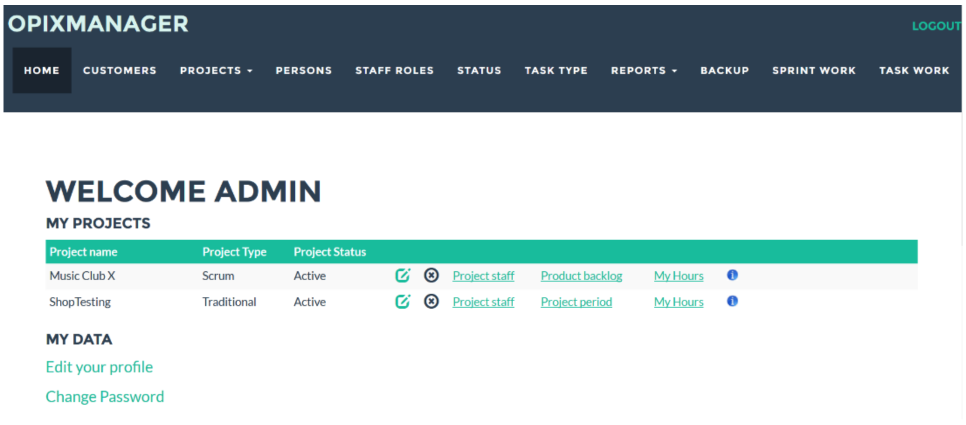
Old Password	•••••••
New Password	
Confirm New Password	

Change password page

Screenshots of linked pages in new user interface



Navigation bar



WELCOME ADMIN

MY PROJECTS

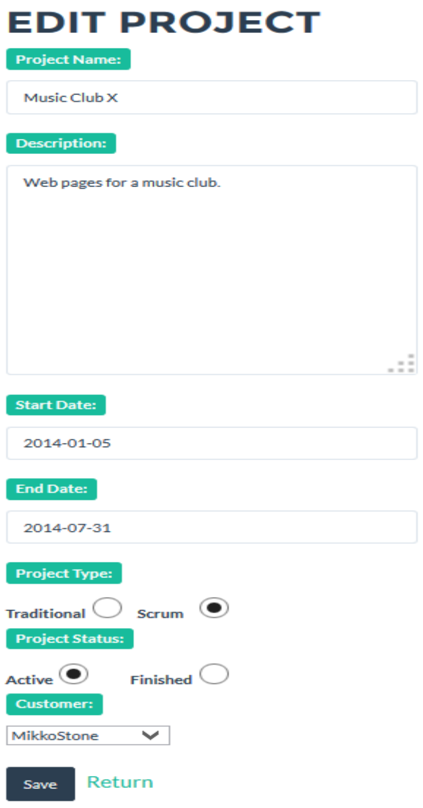
Project name	Project Type	Project Status					
Music Club X	Scrum	Active	Project staff	Product backlog	My Hours		
ShopTesting	Traditional	Active	Project staff	Project period	My Hours		

MY DATA

Edit your profile

Change Password

Home page



EDIT PROJECT

Project Name: Music Club X

Description: Web pages for a music club.

Start Date: 2014-01-05

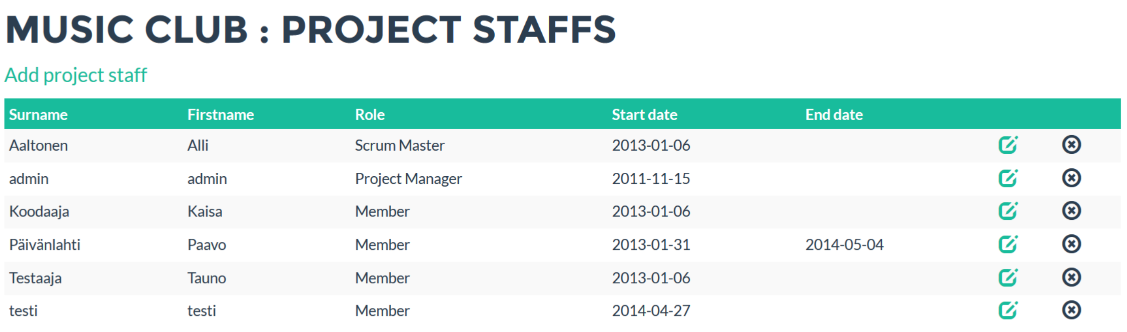
End Date: 2014-07-31

Project Type: Traditional [✓ Scrum]

Project Status: [✓ Active] Finished

Customer: MikkoStone

Edit project page

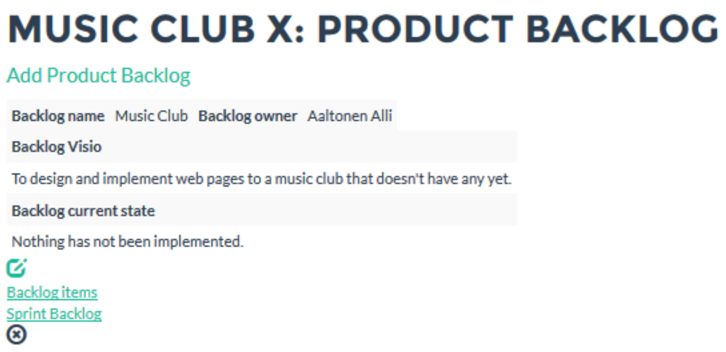


MUSIC CLUB : PROJECT STAFFS

Add project staff

Surname	Firstname	Role	Start date	End date		
Aaltonen	Alli	Scrum Master	2013-01-06			
admin	admin	Project Manager	2011-11-15			
Koodaaja	Kaisa	Member	2013-01-06			
Päivänlahti	Paavo	Member	2013-01-31	2014-05-04		
Testaaja	Tauno	Member	2013-01-06			
testi	testi	Member	2014-04-27			

Project staffs page



MUSIC CLUB X: PRODUCT BACKLOG

Add Product Backlog

Backlog name	Music Club	Backlog owner	Aaltonen Alli

Backlog Visio

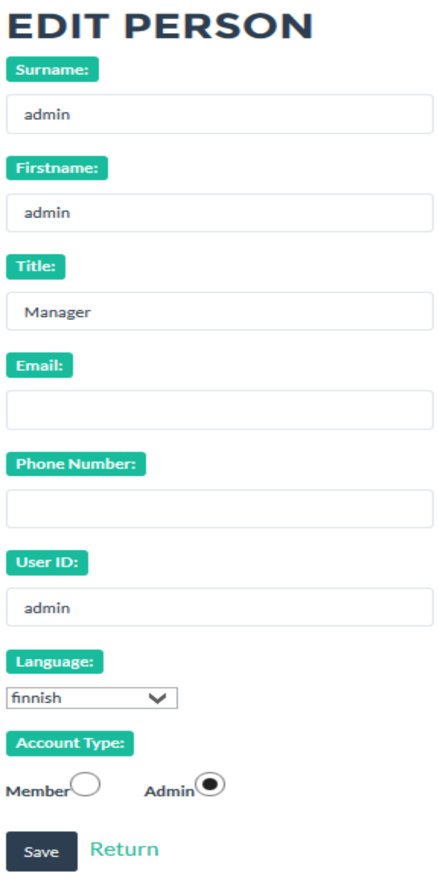
To design and implement web pages to a music club that doesn't have any yet.

Backlog current state

Nothing has not been implemented.

Backlog items
Sprint Backlog

Project backlog page

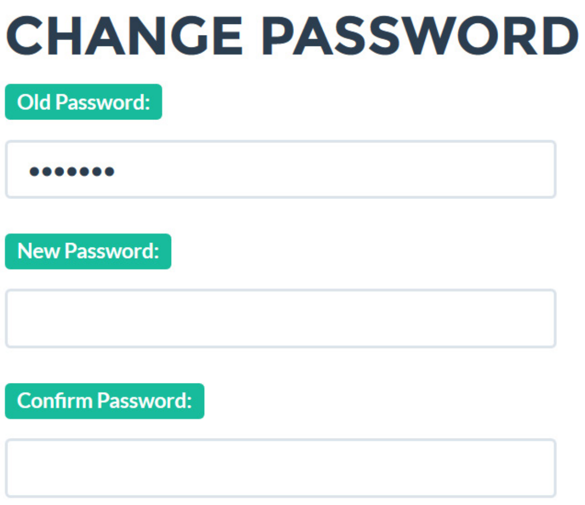


EDIT PERSON

Surname:	admin
Firstname:	admin
Title:	Manager
Email:	
Phone Number:	
User ID:	admin
Language:	finnish
Account Type:	Member [✓ Admin]

Save Return

Edit your profile page


CHANGE PASSWORD

Old Password:

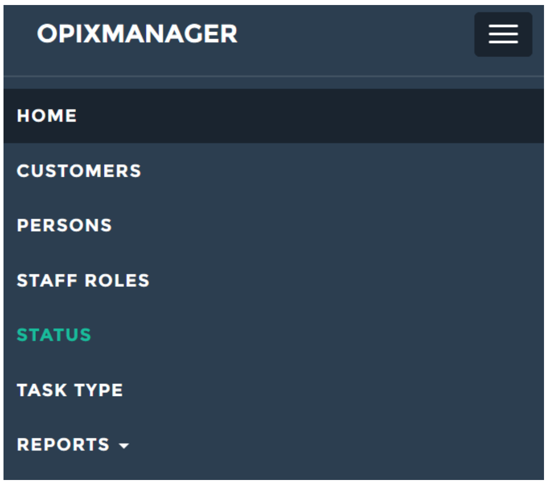
•••••••

New Password:

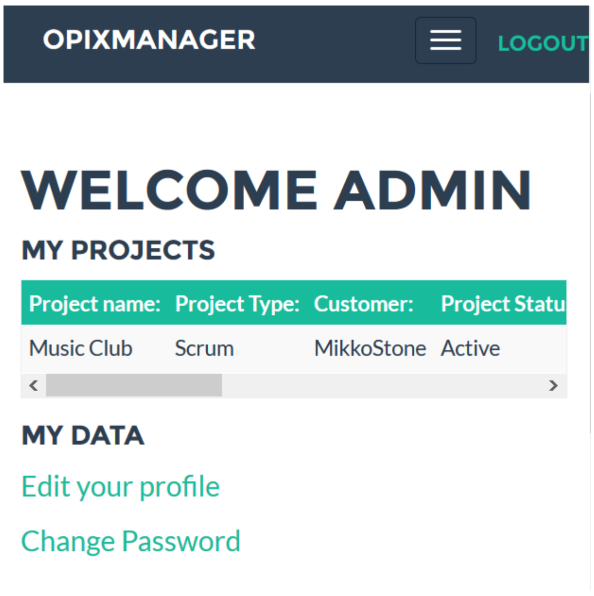
Confirm Password:

Change password page

Mobile Views in new user interface



Navigation bar



WELCOME ADMIN

MY PROJECTS

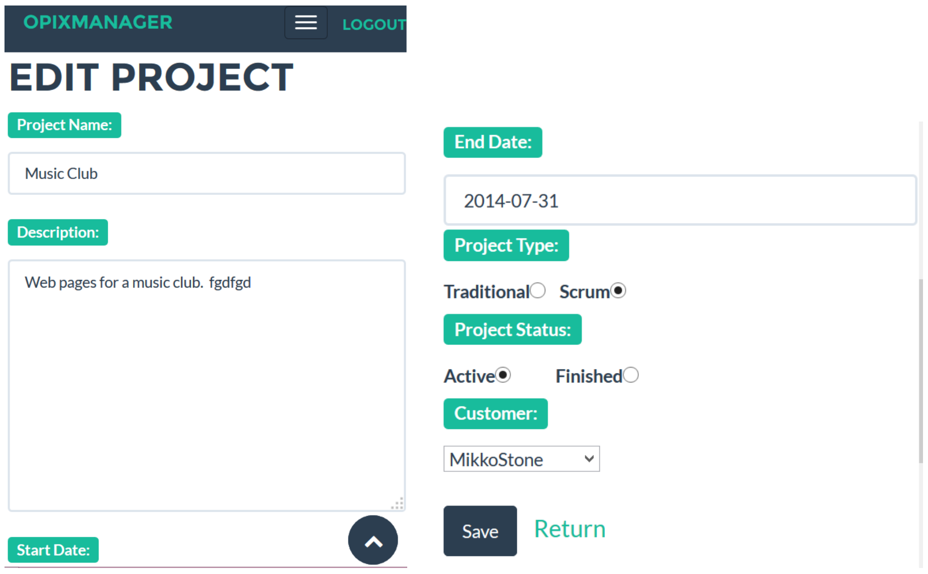
Project name:	Project Type:	Customer:	Project Statu
Music Club	Scrum	MikkoStone	Active

MY DATA

Edit your profile

Change Password

Home page



OPIXMANAGER LOGOUT

EDIT PROJECT

Project Name: Music Club

Description: Web pages for a music club. fgdfgd

Start Date:

End Date: 2014-07-31

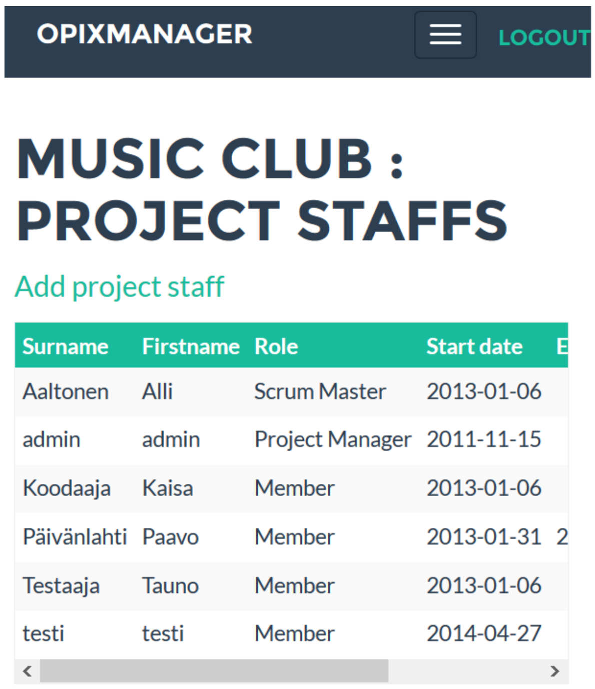
Project Type: Traditional Scrum [✓ Scrum]

Project Status: Active [✓ Active] Finished

Customer: MikkoStone

Save Return

Edit project page



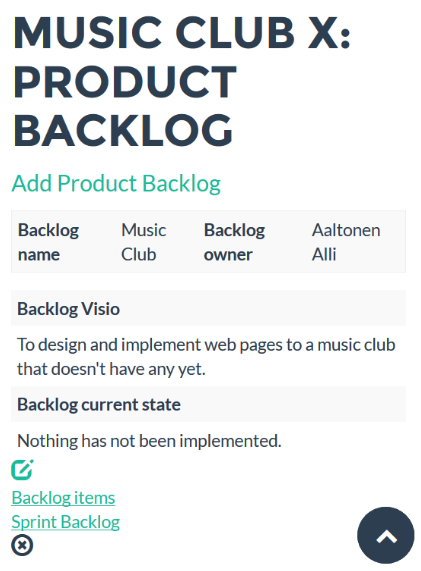
OPIXMANAGER LOGOUT

MUSIC CLUB : PROJECT STAFFS

Add project staff

Surname	Firstname	Role	Start date	E
Aaltonen	Alli	Scrum Master	2013-01-06	
admin	admin	Project Manager	2011-11-15	
Koodaaja	Kaisa	Member	2013-01-06	
Päivänlahti	Paavo	Member	2013-01-31	2
Testaaja	Tauno	Member	2013-01-06	
testi	testi	Member	2014-04-27	

Project staffs page


MUSIC CLUB X: PRODUCT BACKLOG

Add Product Backlog

Backlog name	Music Club	**Backlog owner**	Aaltonen Alli

Backlog Visio

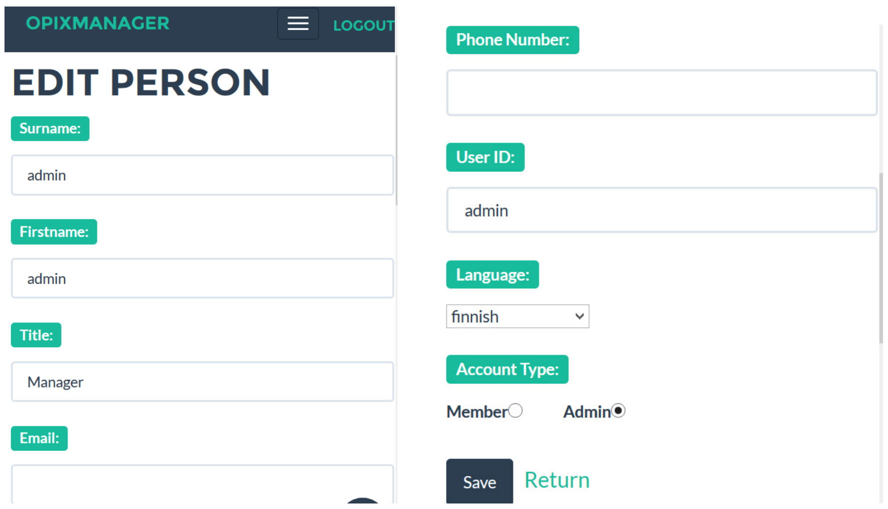
To design and implement web pages to a music club that doesn't have any yet.

Backlog current state

Nothing has not been implemented.

Backlog items
Sprint Backlog

Product backlog page



EDIT PERSON

Surname: admin

Firstname: admin

Title: Manager

Email:

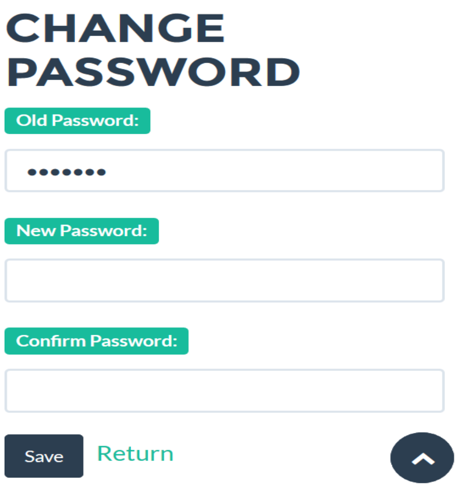
Phone Number:

User ID: admin

Language: finnish

Account Type: Member ○ Admin ●

Edit your profile page


CHANGE PASSWORD

Old Password: •••••••

New Password:

Confirm Password:

Change password page