

Ivan Khokhlachev

Web Application for Course Management

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

11 May 2017



Author(s) Title	Ivan Khokhlachev Web Application for Course Management
Number of Pages Date	52 pages + 0 appendices 11 May 2017
Degree	Bachelor of Engineering
Degree Program	Information Technology
Specialization option	Software Engineering
Instructor(s)	Ilpo Kuivanen, Senior Lecturer
<p>The goal of this thesis was to create a web application for course management using the Yii2 PHP framework, document the main aspects of application and discuss the concepts and main features of the Yii2 framework. This paper serves as a usage guide and API reference as well. The course management application handles such entities as users, courses and tasks. The application is built to support an unlimited number of users, courses and tasks. The main purpose of the application is usability and extensibility. The code as well as functionality will be discussed and taken apart in this paper.</p>	
Keywords	Yii2, PHP, MVC

Tekijä(t) Otsikko	Ivan Khokhlachev Web kurssijärjestelmä
Sivumäärä Aika	52 sivua + 0 liitettä 11.5.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Lehtori Ilpo Kuivanen
<p>Tämän työn tarkoituksena on ollut luoda ja dokumentoida kurssijärjestelmän henkilökohtaisten valmentajien käyttöön. Muokkaamalla kyseistä järjestelmä sitä voi käyttää yleisien kursien ajastamiseen ja hallinnoimiseen. Järjestelmä on rakennettu käyttäen PHP:ta ja kehystä nimeltään Yii2. Järjestelmän avulla pystyy luomaan ja hallinnoimaan käyttäjiä, kursseja ja tehtäviä. Järjestelmään kuuluu sekä hallintaympäristö että käyttäjäympäristö. Tämä dokumentti keskittyy työkaluihin ja konsepteihin, joita on käytetty järjestelmän rakentamiseen.</p>	
Keywords	Yii2, PHP, MVC

Contents

List of Abbreviations

1	Introduction	5
2	Tools	5
2.1	PHP	5
2.2	WebMVC Structure	6
2.3	Composer	7
2.4	Yii2	7
2.4.1	Community and Documentation	8
2.4.2	Multi-purpose Applications and 3rd Party Software Integration	8
2.4.3	Frontend	9
2.4.4	Logging, Debugging and Error Reporting	9
2.4.5	Security	9
2.4.6	Installation	9
2.5	Bootstrap	10
3	Basic Concepts and Application Structure	10
3.1	Database Interface	11
3.2	Namespaces	12
3.3	Controllers and Actions	12
3.4	Models and Classes	13
3.5	Search Models	14
3.6	Routing	15
3.7	Configuration Files	15

	2
3.8 Widgets	17
3.9 Layouts	17
3.10 Forms	17
3.11 Third Party Software	17
3.11.1 Kartik GridView	18
3.11.2 Kartik DetailView	18
3.11.3 CKEditor	18
3.11.4 Yii2 full calendar	18
4 Implementation and Core Components	19
4.1 Users	19
4.2 RBAC	21
4.3 User Configuration	23
4.4 Courses and tasks	23
4.4.1 Course	23
4.4.2 Tasks	25
4.4.3 Task Submit Class	27
4.4.4 Measurements	28
4.4.5 News and Articles	29
4.5 Administration Area (Backend)	30
4.5.1 User Management	30
4.5.2 Course and Task Management	31
4.6 User Area (Frontend)	34
4.6.1 Site Controller	35
4.6.2 News Controller	36
4.6.3 Task Controller	36
4.6.4 User Controller	38
4.6.5 Styling and HTML	38
5 Usage Guide	39
5.1 Login	39

5.1	Backend	39
5.1.1	Manage Users	41
5.1.2	Editing Users	42
5.1.3	Course Management	43
5.1.4	Tasks	45
5.1.5	List of Submitted Tasks	46
5.2	Frontend	47
5.2.1	Navigation	47
5.2.2	Front Page	48
5.2.3	Task Submitting	48
5.2.4	User Management	49
6	Discussion and Conclusions	50
7	Sources	51

Acronyms

Yii2	PHP framework used to build this project
RBAC	Role Based Access control
MVC	Model View Controller
PHP	Recursive acronym for PHP: Hypertext Preprocessor. Programming language used to build this project
OOP	Object-oriented Programming
SQL	Structured Query Language, language used to generate database queries
MySQL	Open source relational database management system
URL	Uniform resource locator, can as well be referred to as web address or link
REST	Representational state transfer, way of communicating and transferring data between systems
GUI	Graphical User Interface

1 Introduction

The aim of this project was to build a simple course management web app for personal trainers. Currently the system is capable of handling users, courses and tasks. Most of the components have many-to-many relationships in one way or another, so it is possible to add users to different courses with different tasks.

The users have different groups to manage accesses and security. Currently the following groups are supported: guest, normal user and administrator.

The web app is easy to install and modify, since it's written using MVC framework with packet management system, named Composer.

2 Tools

2.1 PHP

Initially, PHP was a simple scripting tool released by Rasmus Lerdorf to create dynamic websites. In 1993 the first version named PHP/FI or Personal Home Page/Forms Interpreter was released and was used to parse simple CGI applications written in C language. Parser could call CGI scripts and return pure HTML.

CGI is an acronym for Common Gateway Interface. CGI was and still is used to execute console-like applications, usually written in C, but other languages such as Perl can be used as well. Hosting providers are still offering CGI functionality, though it is not used as often.

In 1995 Lerdorf opensourced the PHP- project so the community could provide bug fixes and new functionality. However, in 1998 the PHP core was completely rewritten and PHP/FI became the PHP known today. With the introduction of version 2.0, PHP can be defined as a programming language. With version 3.0 it finally acquired some OOP func-

tionality, and with version 5.3 garbage collector was first introduced. The Garbage Collector (GC) gave huge performance boost for long-running scripts and applications. Memory is being freed in cycles when more memory is required. Figure 1 shows an illustration of GC performance. [1]

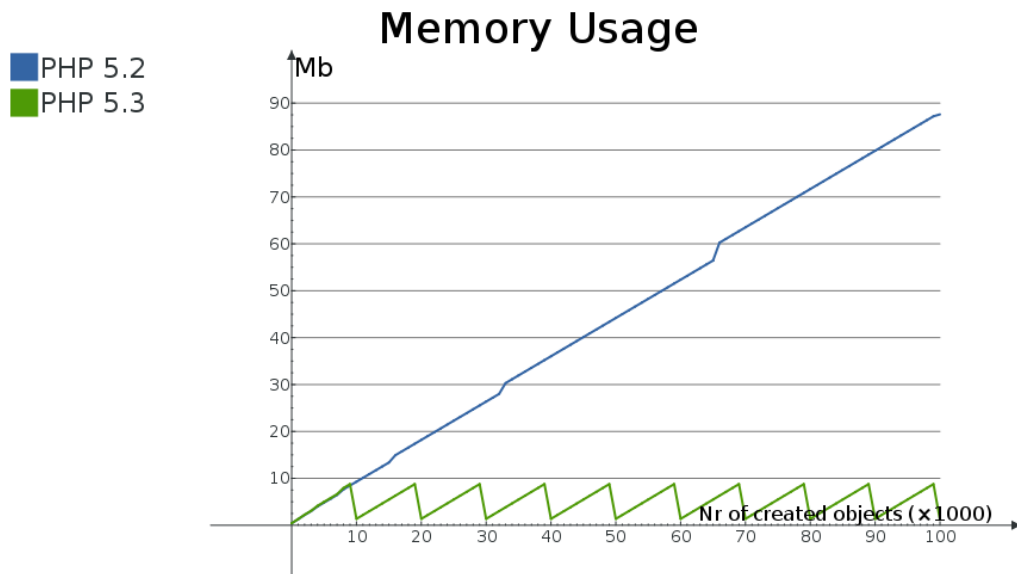


Figure 1: Illustration of GC performance.

Image source: <http://php.net/manual/en/features.gc.performance-considerations.php>

Despite some debatable features and scripting heritage, PHP is a strong OOP language which can be used as it is. However, in order to benefit from all PHP features and to program applications in fast and comfortable way, usage of frameworks is advised.

2.2 WebMVC Structure

WebMVC is a derivative of the MVC software architectural pattern, the main difference being that WebMVC has to handle http- requests and uses database as the main data storage.

MVC is an acronym for Model View Controller. MVC is one of the most popular patterns in programming because of its flexibility and approach to extensibility in OOP- languages.

MVC consists of the following components: [2]

- Model is used to manage all data-related logic, e.g. data can be retrieved or saved using methods provided by model.
- View is used for rendering data on screen, for example UI, data, etc.
- Controller is used to process business logic between models and views.

2.3 Composer

Composer is a package dependency management software for PHP. Composer uses the composer.json- file to keep track of all libraries and packages that have to be updated or installed.

When the composer is installed it can be used through command line. All software installed through composer can be found in the "vendor" folder. However it is not considered good practice to leave the composer.json- file on the production server, it is advised that composer is not used in production and the project is deployed with a ready-to-use vendor folder. [3]

2.4 Yii2

Yii2 is a PHP WebMVC framework which is being developed by a big team all around the world. Yii2 is a simple, extensible, fast framework with comprehensible documentation. The framework is still being rapidly developed – quick fixes and new features are added at least once per quarter. [4]

The main reasons for selecting Yii2 as the main framework of the application are reusable widgets, MVC- structure and Bootstrap- framework that is supported by default and provides nice and responsive design.

For two years, the author has been working with the Yii2 framework as a full stack developer, and Yii2 has proven itself as a diverse and adaptable tool. Overall, Yii2's pros

could be divided into several groups: community and documentation, multi-purpose applications, integration with 3rd party software, pre-made graphical interfaces or frontend, logging and debugging, and security.

2.4.1 Community and Documentation

When picking up a new piece of software or a new tool, documentation is one of the most important aspects of programming workflow. Yii2 has a comprehensive wiki with guides and API reference. When wiki is not enough, it's possible to turn to community for help and ask questions on forums or IRC.

Yii2's community is fast, nice and reliable source of information, since people answering questions on forums are usually framework developers themselves or developers of third party components. Personally, with Russian being the native language of the author, it was a surprise that the Russian section of the forums is on a par with the English-spoken segment.

2.4.2 Multi-purpose Applications and 3rd Party Software Integration

During the work several different applications were created – from usual websites to complex systems. In complex applications Yii2's potential can be unleashed to the fullest – systems can be built upon console applications for Linux and database integration, or application can be fully integrated into totally different systems using RESTful Web Services or interfaces provided by the Yii2 core libraries.

Elastic Search (ES) interface may work as a perfect example of such integration. Elastic Search is an open source search and analytics engine that interacts mainly through RESTful Web Services, and needs a Java Server to function properly. While ES can be used through REST without additional components, Yii2 has full section of core libraries devoted to communication between application and Elastic Search server. Because of those libraries, REST is simply not needed anymore, and Elastic Search can be used through standard Yii2 objects.

2.4.3 Frontend

Frontend or GUI is simple with Yii2 – all widgets and default blocks have their own styles and themes. When those styles are not graphical masterpieces and not as pleasant to the eyes as handcrafted designs, they are useful when building administration panels or any project with simple graphical interface.

2.4.4 Logging, Debugging and Error Reporting

Yii2 has its own set of tools for debugging and logging for code, memory and SQL. The most notable debugging tool is Debugging panel, which is, nowadays, the de facto standard in most modern PHP frameworks. In the panel a developer can view all requests to web server and database, versions of used software, history of emails sent through the application, and other information. Debugger is highly extensible and can be configured by the developer. When an error has occurred, application shows a comprehensible stack trace error page and writes all related data into log file.

2.4.5 Security

Security is a major feature in the Yii2 framework. Because of the OOP approach and various helpers that ensure data consistency, it's nearly impossible to perform most popular attacks.

2.4.6 Installation

Yii2 is easy to install given that the required environment is configured properly.

Requirements

The following requirements have to be met in order for Yii2 function properly: [5]

- PHP version 5.4
- MySQL 4.1 or later
- At least the following PHP modules:

- php_bz2, php_curl, php_mbstring, php_exif, php_fileinfo, php_gd2, php_gettext, php_intl, php_mysql, php_mysqli, php_pdo_mysql, php_pdo_sqlite
- Apache or Nginx web server

Normally, the web server does not require any configuration and can be used out of the box, however it is recommended that the `mod_rewrite` module is installed.

`Mod_rewrite` Apache module allows url rewrites through Apache web host configuration or `.htaccess` files. With `mod_rewrite` it is possible to hide script names and omit unnecessary chunks of data in url.

2.5 Bootstrap

Bootstrap is a responsive CSS/HTML framework that provides ready-to-use CSS classes. Main purpose of CSS is to provide tools required to style HTML. Bootstrap takes CSS functionality to a completely new level, introducing grid patterns for structured layouts and numerous modules like collapsible text blocks or progress bars.

3 Basic Concepts and Application Structure

The application is built using Yii2 advanced template which consists of four main folders - backend, console, common and frontend. Backend is being used for the administration area, when frontend is used for the user area. The “common” folder is used for storing components and configuration files that can be used by other folders or namespaces. The “console” folder is not required for the current project, since there are no console scripts. Figure 2 shows the skeleton of the Yii2 advanced template.

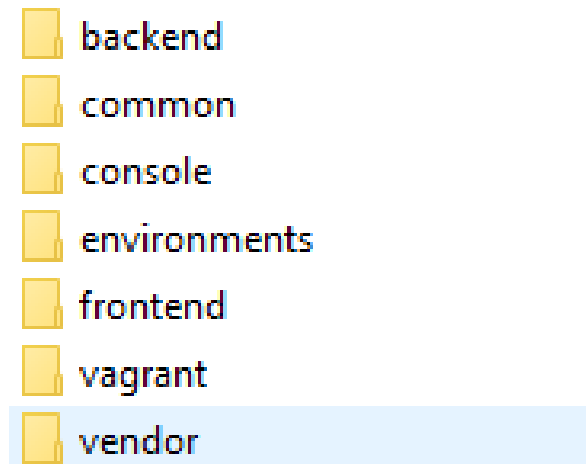


Figure 2: Skeleton of Yii2 advanced template

Apart from standard templates by Yii2 developers, community has created a variety of templates for different appliances. During planning of the project, it is preferable that developer chooses the most suitable template for the project.

3.1 Database Interface

Yii2 provides a standard object-oriented interface for database applications named Active Record. Most models used in the application discussed here are using the ActiveRecord interface or extend models with ActiveRecord interface.

The main benefit of the ActiveRecord interface is the simple and fast usage of database through object without using actual SQL queries.

Setting the ActiveRecord class is simple – it has to extend `yii\db\ActiveRecord` class and have static method `tableName()` that returns database table name associated with that class. Other methods for queries are provided by parent class and can be used out of the box. Figure 3 shows the example of the ActiveRecord method.

```
/**
 * FIND BY PARENT
 */
public static function findByParent($id)
{
    return static::findOne(['user_id' => $id]);
}
```

Figure 3: Example of query using `findOne()` method provided by `ActiveQuery` class

The framework offers other database interfaces, and depending on the goal different interface may be used for different need, e.g. for more seamless integration with 3rd Party Software. The `ActiveRecord` is the only interface used in the project, and therefore only `ActiveRecord` is discussed in this paper, for more information on `ActiveRecord` and other interfaces it is advisable to consult with official Yii2 documentation.

3.2 Namespaces

Namespace is the concept that was first introduced in PHP 5. Namespaces are designed to solve problems encountered when using reusable code components and modules. Namespaces are used to specify path for files in same folders as relative, which eases usage of components and adds ability to give aliases to class names.

In Yii2 namespaces are used in all classes, the application will not run if the namespace is not specified. If the classes `User` and `Course` exist in the same folder and have the same namespace - object `Course` can be referred in the `User` class without path declaration and vice versa.

3.3 Controllers and Actions

From the PHP perspective, controllers are basic classes which extend `yii\base\Controller` class. As stated in the MVC pattern controllers are used to control the overall applications workflow, and exchange information between views and models. In Yii2, controllers are responsible for request handling as well as for generating responses.

When all models are common for both the user and administrator area, different controllers with the same name are created for different namespaces. All controllers extend a single controller from “common”- namespace named MainController. MainController consists of a single beforeAction()- method to manage user accesses.

There are two types of methods in controllers: action methods and normal PHP functions. Action methods cannot be static, and are used for routing. All actions must start with “action”- keyword. An example of a typical controller action can be seen in Figure 4.

```
/**
 * Logout action.
 *
 * @return string
 */
public function actionLogout()
{
    Yii::$app->user->logout();

    return $this->redirect(['site/login']);
}
```

Figure 4: Example of an action

When the action is used, it is implied that the function in question has a relation with some page on the website, for example, it renders the view or redirects to other action.

3.4 Models and Classes

Basically, classes can be divided into two sections: classes that are not using database, e.g. classes that are used to create forms (extending BaseObject Yii2-class), and classes that are using database interfaces (extending ActiveQuery- class). Controllers will not be discussed in this section - while being classes by definition, they do not fall into MVC-pattern workflow as models.

Both BaseObject and ActiveQuery classes provide methods for managing and validating data. One of the most important methods is the rules() method, which is used to validate data passed to an object before writing to database. This "client-side" validation helps to keep database structure intact, enhances performance and eases the creation of custom validation rules when needed. Figure 5 show the example of rules() method.

```

/**
 * @inheritdoc
 */
public function rules()
{
    return [
        [['description'], 'string'],
        [['status'], 'integer'],
        [['start_date', 'end_date'], 'safe'],
        [['name'], 'string', 'max' => 255],
    ];
}

```

Figure 5: Example of rules() method

The "rules()" - method shown in figure 5 consists of 2D-array. The first cell contains a list of validated attributes, other cells contain validation information, e.g. data type or specific rules like length or regular expression. In a current example "string" means that an attribute has to contain a word or characters, "integer" validates only when an integer is contained within an attribute, "safe" is the keyword which allows any information to be passed to an attribute, and "max" is the maximum length of a string.

3.5 Search Models

Search models are subclasses of models with the ActiveQuery- interface. Search models are used to ease the building of more complex queries and retrieval of data from the database.

Only two conditions have to be met when creating a search model – the search model class has to extend class with the `ActiveQuery` interface and implement method `search()`, which returns the instance of `yii\data\DataProviderInterface`- class.

`DataProviderInterface` is an interface that returns an array of data divided by pages (pagination). If needed, the pagination can be disabled, however it is not advised since the amount of data can vary and potentially lead to a long loading time or even server crash.

3.6 Routing

Most of the routes or url's are built using controller-action pair. Url is composed from the following components in order: host or hostname, controller, action and attributes.

For example, if the hostname is "example.com", the controller and action are named respectively "user" and "view" and attribute passed to "view" action is an "id" attribute with value of 1 – Yii2 will generate the following url: `example.com/user/view/1`.

The values in Url may be omitted or added using Yii2's URL manager or `mod_rewrite` functionality provided by Apache web server. [6]

3.7 Configuration Files

Each of the main folders or namespaces (backend, console, common and frontend) have their own configuration files. In Yii2, advanced template main configuration files are held in the "common" folder so that other namespaces can be configured from the same place. Database, modules, aliases and other Yii2 functionality is being held in common-namespace. List and description of configuration files can be seen in the Table 1.

Table 1: List of the configuration files

Name of file	Description of content
bootstrap	Used for bootstrapping application components when needed. In current implementation only aliases are being specified during bootstrapping.
main	Main configuration file, which in theory is immutable once deployed to production server. Modules, components and paths are specified in main configuration file.
main-local	Local main- configuration file which is used only in deployed applications. It is used to specify local data like mailing and databases.
params	Hardcoded parameters like keys can be specified at any time in params- configuration files. Yii2 provides its own interface to obtain data listed in params- file. Immutable in theory once project is deployed to production server.
params-local	Local file for parameters. Used in same way as main-local configuration file – is environment dependable.

The configuration files in other namespaces are mostly used for url rewriting and specifying namespace specific variables and values.

3.8 Widgets

Widgets are reusable independent blocks of code that are used mainly in views. Every widget has to extend from `yii\base\Widget` and override the `yii\base\Widget::init()` and/or `yii\base\Widget::run()` methods.

Widgets can be used when data has to be obtained or viewed during runtime after rendering of a page, without using additional actions or redirects.

3.9 Layouts

Reusable layouts can be created and attached to any controller at any time. Layouts can be seen as wrappers that consist of more static information like metadata, scripts, navigation bars, wrapper containers and immutable HTML blocks, e.g. footers. Overall layout files can be seen as an HTML page skeleton, with dynamically inserted views.

3.10 Forms

By definition, forms used in Yii2 are widget instances that render input and use models to validate data. Additional customization like addons and styles may be applied to widget instances. All validation and logic is kept within normal classes that extend Yii2 `BaseModel` class.

3.11 Third Party Software

Several third party libraries and widgets are used in the project. Description of the used components can be seen in the next section.

3.11.1 Kartik GridView

Kartik GridView is an extended version of Yii2's GridView widget that adds some functionality and eases overall usage of Yii2's default GridView.

The main purpose of the GridView- widget is to list data provided by search model in the grid table. It is possible to query and sort data through interfaces provided by GridView via AJAX.

The main difference between standard implementation and Kartik's version is the usability and additional features. Kartik's version is more stable and provides much more options during instantiation.

3.11.2 Kartik DetailView

Same as GridView, Kartik DetailView is an extended version of DetailView widget provided by the Yii2 core library. DetailView is used to output and manage data of a single model through the built-in form functionality. In Kartik's implementation DetailView comes with AJAX support by default which eases form submission and improves overall usability.

3.11.3 CKEditor

CKEditor is a free, open source text editor. Since CKEditor is not part of Yii2's core library in any way, community wrapper was used. Community wrapper is used to provide CKEditor's functionality through widget.

3.11.4 Yii2 full calendar

The Yii2 full calendar is a Yii2 wrapper of JQuery Fullcalendar, which is a highly customizable and responsive javascript calendar. Widget supports AJAX, timed events and different views.

4 Implementation and Core Components

4.1 Users

User information is stored in two tables: `user_users` and `user_user`. The first table, `user_users` is a parent table of `user_user` and is needed for storing system and authorization information such as passwords, access groups and access and tokens. Access groups are discussed later in the RBAC (Role Based Access Filter) section. Both user tables are implemented in models `Users` (database table named: `user_users`) and `User` (database table named: `user_user`).

The database table `user_user` contains user information e.g. name, age and other information. The user model will be referred to as `Account`, so that any confusion between user tables and models could be avoided. Description of the data stored in `User_Users` database table can be seen in Table 2, and `User_user` database table in Table 3.

Table 2: List of data stored in `User_users` database table

Name	Description	Type
id	Unique id of user	int(11)
username	Username of user, has to be an email address	varchar(255)
password	Hashed password of a user	varchar(255)
signup	Date of registration	date
last_login	Latest login date	date
authKey	Unique authorisation key for user	varchar(255)
accessToken	Unique access token for user	varchar(255)
identityClass	Identity class of specific user. Users can have different identity classes to provide different functionality. IdentityClass is not used in	varchar(255)

	current implementation, since all users are instances of the same class.	
enableAutoLogin	Value that returns boolean and logs user automatically if has value of "1"	tinyint(4)
identityCookie	Unique cookie value of a user	text
group	User group for RBAC	Varchar(50)

Both user tables contain the "id"- attribute which is the unique integer for database records. In User_Users database table column "username" contains the unique email address of a user, column "password" contains hash value of a password, "signup" is the date of registration, "lastlogin" is the date of the last login into system, "authKey" , "authToken" and "identityCookie" are the unique authorization key, authorization token and identity cookie that are used for security, "identityClass" is the name of the class which the Users class has to extend, "enableAutoLogin" is the value that allows to keep user logged in the system, "group" is the name of the RBAC group that user belongs to.

Table 3: List of data stored in User_user database table

Name	Description	Type
id	Unique account id	int(11)
user_id	Foreign key to user_users-table, that provides authorisation information	int(11)
name	Users name	varchar(255)
surname	Users surname	varchar(255)
date_of_birth	Users date of birth	date
active	Boolean, user seen as suspended if value of "active" attribute is 0	tinyint(4)
phone	Phone number of user	varchar(50)

sex	Sex of a user, male where 0 and female where 1	tinyint(4)
instagram	Link to user's isntagram profile	varchar(255)
facebook	Link to user's facebook profile	varchar(255)

Apart from “id” column, table User_user consists of following columns: “user_id”, “name”, “surname”, “date_of_birth”, “active”, “phone”, “sex”, “Instagram” and “facebook”. Column “user_id” is the value of the unique “id” of table User_users which User_user record is related to; “name”, “surname” and “date_of_birth” columns contain actual name, surname and date of birth of a user, “active” column tells if user is allowed to use the system or not, “phone” is the phone number, “sex” is the biological sex of a user, “Instagram” and “facebook” columns contain links to user’s Instagram and Facebook profiles.

4.2 RBAC

RBAC is an acronym for Role Based Access Filter. As stated in the name – the basic concept of RBAC is to manage user groups and accesses or permissions for specific action. In Yii2 there are two ways of configuring RBAC, using two different managers – PhpManager and DbManager. PhpManager is used to manage accesses and user groups using php file with hardcoded data, whereas DbManager is using database to store access information. Only DbManager is discussed here, since it is used in the current implementation.

DbManager uses the following tables to store access data: [7]

- itemTable: the table for storing authorization items. Defaults to "auth_item".
- itemChildTable: the table for storing authorization item hierarchy. Defaults to "auth_item_child".
- assignmentTable: the table for storing authorization item assignments. Defaults to "auth_assignment".
- ruleTable: the table for storing rules. Defaults to "auth_rule".

Most of the data is stored in `auth_item` and `auth_item_child` tables. `Auth_assignment`-table is not used at all, and `auth_rule`- table stores single rule for user groups named "userGroup". Rule "userGroup" was generated using Yii2 built-in console interface tools. Example of user inheritance according to RBAC can be seen in Figure 6.

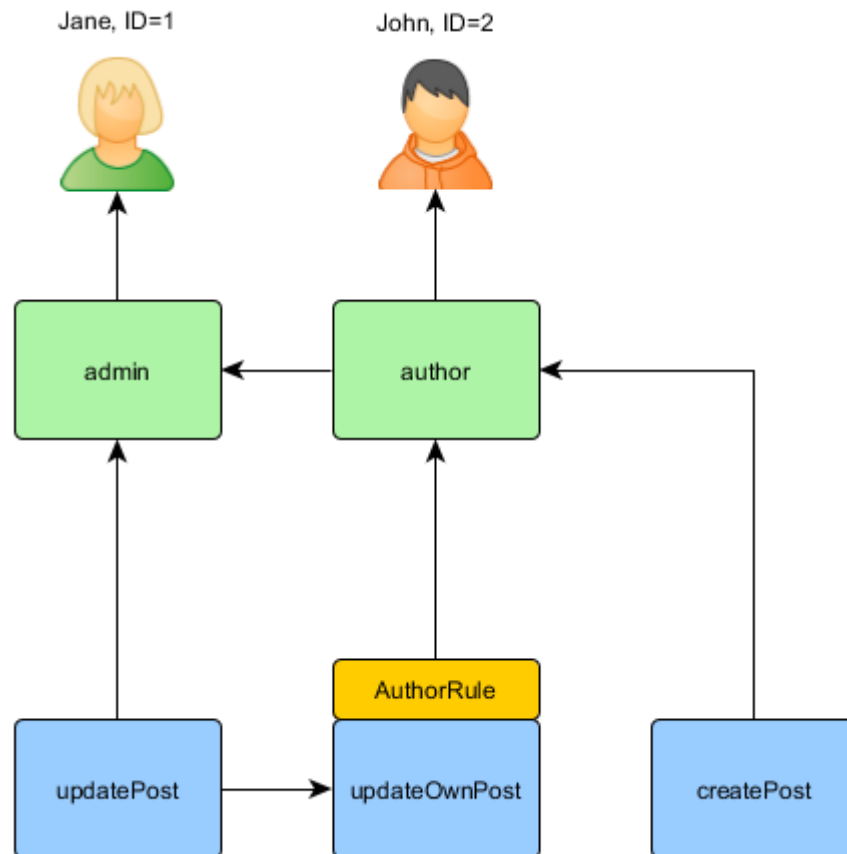


Figure 6: Example of RBAC inheritance.

Image source: <http://www.yiiframework.com/doc-2.0/guide-security-authorization.html>

Currently there are 3 groups in the system: admin, user and guest. "Admin" group is the most privileged group, meaning it can perform all actions of the "user" and "guest" groups. RBAC hierarchy is built upon inheritance, e.g. highest group in hierarchy inherits all permissions of sub- groups and adds them to own unique permission.

4.3 User Configuration

All Yii2 templates have the default user model ready for use, however, the default class does not use the database to store user information – all user-related data is hardcoded as an array in the model's variables.

In the current implementation the initial user model was replaced with a custom user model named Users. The model extends the Yii2 ActiveRecord class for storing information in the database and implements Yii2 IdentityInterface. IdentityInterface has to be implemented by any class that is used for authorization.

4.4 Courses and tasks

Main purpose of this application is to allow administrator to manage courses: its participants, tasks, dates and other related information. Courses and tasks will be discussed in one section, while course can be used without tasks, tasks cannot function without being assigned to course.

4.4.1 Course

Course has many-to-many relationship with user table and one-to-many relationship with tasks table. Once course is created, administrator can add users to course and create tasks. When course is deleted, users are removed and all tasks assigned to course are deleted. The course database table can be seen in Table 4. Most of the data listed in course database table is not obligatory and needed only for descriptive purposes, since all relationship is contained in map tables. Column "id" is the unique id of a course, "name" is the title of a course, "description" is the descriptive information about course, "status" is the status of a course, e.g. is course active or not. Columns "start_date" and "end_date" are starting and ending dates of a course.

Table 4: Course database table

Name	Description	Type
id	Unique id of course	int(11)

name	Name of course	varchar(255)
description	Course description	text
status	If course is enable, status returns 1	tinyint(1)
start_date	Starting date of course	date
end_date	Ending date of course	date

To ensure the many-to-many relationship between courses and users, a class called CourseUserMap was created. The course_id is the foreign key to the course entry (unique id of a course), and the user_id is the foreign key to the user_users entry (unique id). This construction makes system capable of handling the many to many relationships, in other words different users can be assigned to different courses without limitations or constraints. Table 5 shows data used in the course_user_map database table.

CourseUserMap structure:

Table 5: Course User Map database table

Name	Description	Type
id	Unique id of an entry	int(11)
course_id	Foreign key to course- table	int(11)
user_id	Foreign key to user_users- table	int(11)

Course Class Reference

Course extends Yii2's ActiveRecord- interface. List and explanation of important methods can be seen below in Table 6. Deletion logic and deletion process of a course is explained in description of “beforeDelete()” method. Many to many relationship between Course model and User model is established through “getCourseUser()”, this method makes User entities related to instated Course object available through variables.

Table 6: Course class reference

Name	Description	Return
beforeDelete()	BeforeDelete() is a standart method provided by Yii2's ActiveRecord interface. BeforeDelete() is called before deletion of a model. Before course is deleted all tasks and entries in CourseUser-Map related to to-be deleted course are deleted.	Boolean
getCourseUser()	Establishes relation between Courses and Users	ActiveQuery instance

When course is created, tasks and users may be assignend to it. Tasks are discussed in the next chapter.

4.4.2 Tasks

The tasks have a many to one relationship with the courses and a one-to-many relationship with the TaskSubmit- class. The tasks are used to create assignments for each user assigned to the course. The task database table is explained in Table 7. In database table only important and obligatory data are the name of the task and the parent_id of the task. Column "parent_id" is the unique id of the course related to the task in question. Columns "name" and "description" contain information with the name of a task and description of a task. If task is unpublished, in other words, not visible to client – column "status" contains 0, column "status" contains 1 if task is published. "Create_date" and "publish_date" contain date information, "create_date" contains information when task was saved for the first time, "publish_date" is the date when task becomes visible. Column "upload" contains public path to file related to task.

Table 7: Course_tasks database table

Name	Description	Type
id	Unique id of task	int(11)
parent_id	Foreign key to course table, where parent_id in task table is id in course table	int(11)
name	Name of task	varchar(255)
description	Description of task	text
status	Boolean, if value is 0 – task is considered unpublished	tinyint(4)
create_date	Creation date of task	date
publish_date	Publishing date of task	date
upload	Path to file related to specific task	text

Task class reference is explained in Table 8. Method “beforeDelete()” is called before task is delete, method “afterSave()” is called every time when task data is written into database. Method “upload()” is called every time task form is being validated, it checks and uploads the file that was put by user.

Table 8: Task class reference

Name	Description	Return
upload()	Upload()- method is called every time when form related to task is submitted. Main purpose of upload() method is to validate, upload and save information about file related to task.	Boolean
beforeDelete()	Before task is deleted, all related TaskSubmit records are deleted from database.	Boolean

afterSave()	After Task instance is saved, system creates TaskSubmit records for each user assigned to course.	Boolean
-------------	---	---------

Overall the Task and the Course classes consist of the same, reusable methods, and along with user management they form the backbone and the main feature of the system.

4.4.3 Task Submit Class

The purpose of task submit is to provide users with a mechanism to submit assignments and leave comments. Submit records are created for all users assigned to course once task is created. Description of task submit database table can be seen in Table 9. TaskSubmit needs user_id of the unique user and the parent_id (unique id of task) to be created. When user submits the task, task submit related to submitted task will be marked as “done” in the database table. Code is the same as in Task Class, the only exception being that there is no “upload” method. Task Class reference can be seen in Table 8. Column “id” is the unique id of a task submit record, column “done” is used to determine if assigned task to specific user is done, task is considered done when value of the column is 1 and incomplete when value is 0. Column “comment” user’s feedback, column “done_date” is updated when user completes the task.

Table 9: Task submit database table

Name	Description	Type
id	Unique id of task submit record	int(11)
user_id	Foreign key to user_users table. Id of a user assigned to task	int(11)
parent_id	Foreign key to course_tasks	int(11)
done	Boolean. Task is submitted when value is 1	tinyint(4)
comment	Comment that user leaves during task submission	text
done_date	Date of task submission	date

4.4.4 Measurements

The course management system made for personal trainers needs a way of tracking the client's progress. Progress is measured by changes in the client's body, e.g. weight, muscle growth.

All measurements are not obligatory and are used to describe physical size of the client, more detailed information about measurements can be viewed in the Table 10. Column "id" is the unique id of a measurement entry, column "user_id" is the id of the related user. Columns "weight", "breast", "hip", "legs" and "hand" are used to contain measurement information in centimeters. Column "comments" is used to contain comments about related data entry. The "time_created" and the "time_updated" columns are used to track creation and update dates.

Table 10: Measurements database table

Name	Description	Type
id	Unique id of "measurement" entry	int(11)

user_id	Foreign key to user_users table	int(11)
weight	Weight of user	decimal(5,2)
breast	Breast size of a user, in cm.	decimal(5,2)
hip	Hip size of a user, in cm.	decimal(5,2)
legs	Leg size of a user, in cm.	decimal(5,2)
hand	Hand size of a user, in cm.	decimal(5,2)
comments	Comments about related measurement entry	text
time_created	Creation date of an entry	date
time_updated	Update date of an entry	date

When Measurement functionality is completed, it is planned that statistics and different information about courses and people can be gathered and evaluated. For now, measurements are used simply for the tracking of the user's progress.

4.4.5 News and Articles

Currently, only the news functionality is implemented. All data is stored in a single table with a text column. News has no relationship to anything. See Table 11. Text of the article is contained in "description" column, status or whether the article is published or not is contained in "status" column, "name" column is used for the title of an article, and "create_date" and "publish_date" are creation and publish dates of an article.

Table 11: News database table

Name	Description	Type
id	Unique id of an article	int(11)
name	Title of an article	varchar(255)
description	Text of an article	text
status	Status of an article, where 0, article is unpublished and where 1 article is published	tinyint(4)

create_date	Creation date of an article	date
publish_date	Publish date of an article	date

4.5 Administration Area (Backend)

Only users assigned to the user group “admin” are able to the access admin area. In the admin area administrator is able to manage users, courses and tasks.

4.5.1 User Management

Users can be created, modified and deleted. As stated above, the class Users used for authorization is referred to as User class and sub model of Users called User will be referred to as Account. User controller is used to view and modify user related data using actions like “actionNewUser()” to create new user, “actionListUsers()” to list users in the system, “actionView(id)” which shows and describes the selected user, “actionDelete()” to delete user from the system completely, including all course and task progress. For more detailed information, see Table 12.

Table 12: User controller

Name	Description	Return
actionNewUser()	Action renders view with AddNewUser form. Once form is submitted, POST request is caught and new user entry is created. Account entry is created automatically after User entry is inserted into database	Renders view with form when model is not submitted. If form is submitted – new user is created and administrator is redirected to the main page.
actionListUsers()	Action lists all users found in the system using UserSearch() class.	Render page with GridView widget which lists user information and links to view and delete actions.

actionView(user id)	<p>Action takes user id as an attribute and renders view where user can be modified.</p> <p>Catches two different post requests – one is for form that changes password, the other is for other user and account information.</p>	<p>Renders form inside DetailView, where basic user information can be modified.</p> <p>Renders simple password form.</p> <p>Renders measurements records inside GridView widget, where all measurement records can be examined or modified.</p> <p>Renders button to add new measurement- record.</p>
actionDelete()	<p>Deletes user. Before user is deleted, the system deletes account information, all task submits assigned to user and removes user from all courses.</p>	<p>Redirects to ListUsers() action.</p>

User controller may be considered as the main and the most crucial controller of the system. Without users, it would be impossible to use and manage data in the system.

4.5.2 Course and Task Management

The courses can be listed, created and modified. Since the courses have a many-to-many relationship with the users – multiple users can be assigned to courses and vice versa.

The tasks can only be created, deleted and updated through the Course management view. Course controller is described in Table 13, and Task controller in Table 14. Course and task controllers have methods and actions similar to other controllers used in this project and can be viewed more thoroughly in corresponding tables.

Course controller consists of multiple actions to redirect and manage the workflow, as well as one method “findModel”, which adds a more approachable way to find course instance by corresponding course id. Following actions are included into course controller functionality: “actionListCourses”, “actionNewCourse”, “actionView” and “actionDelete”.

To list all courses in the system, action “actionListCourses” is used. Upon calling the “actionListCourses” system renders the list of all courses found in the system. To create a new course, action the “actionNewCourse” is used – it renders a form which is used to define a new course. The “actionView” renders the form, where all course related information, as well as tasks and users assigned to the course, can be managed. To delete a course, administrator must call the “actionDelete” action. When course is deleted, all users are removed from the to be deleted course, and all tasks and task submits relevant to the deleted course are removed from the system as well.

Table 13: Course controller

Name	Description	Return
actionListCourses()	Lists all courses using CourseSearch- class	Render page with GridView widget which lists course information and links to view and delete actions.
actionNewCourse()	Catches submitted course data and validates it. If validation is passed – creates new entry in Course table.	Render form, where administrator can input title, description, status, start date and ending date of a course.

		When form is submitted re-directs to actionList-Courses()
actionView(course id)	Action takes course id as an attribute and renders course data. When form is submitted actionView catches POST data and modifies data in course dable. If users were assigned to or deleted from course – CourseUser-Map table is being updated	Renders a form within DetailView to manage course data. Renders GridView for tasks, where tasks can be viewed and modified. Renders button to create new task.
actionDelete(course id)	Action deletes course from the system. Before course is deleted, all subtasks, task submits and data in CourseUserMap is deleted.	Redirects user to actionList-Courses()
findModel(course id)	Private method that is used for fast instantiation of Course model by given Id	Course class instance

Tasks are managed in similar way to courses, e.g. tasks can be created, viewed and deleted. The “actionDelete” removes selected task and all relevant task submit data from the system, the “actionNewTaskFromParent” is used to create a new task from the active course view (course is considered a parent in this context). To update task through form, the “actionUpdate” must be called. To view the user progress of a selected task, the “actionView” is used. Upon calling the “actionView” list of users and relevant task submits is shown.

Table 14: Task controller

Name	Description	Return
actionDelete(task id)	Action takes task id as an attribute. Before task is deleted, all subtasks of the task are deleted.	Redirects to current course view.
ActionNewTaskFromParent (course id)	Creates new task using course id.	Redirects to course view.
actionUpdate(task id)	Manages Task form submission data and uploaded files.	If form is not submitted – renders form to manage task data. If form is submitted, task is created and administrator is redirected to course view page.
actionView(task id)	Fetches all TaskSubmit entities through TaskSubmitSearch object.	Renders GridView with TaskSubmits for each user assigned to course.

Understanding the course class and the course controller is important to understand how system handles the data. Most of the functionality is defined in the course controller, e.g. data that defines the workflow of the system and fills system with information. Course controller makes it possible to define course-user relationship.

4.6 User Area (Frontend)

The user area has its own set of controllers and views like the administration area. Most of the controllers have same names and methods as the ones used in backend. Controllers were moved in own namespaces to ensure that normal users will not gain access to administration functions.

4.6.1 Site Controller

Site controller is the default controller and is used when action is hard to classify or doesn't require its own controller, e.g. login page, front page or contact form page. See Table 15. The default action is the "actionIndex()", which can be referenced as the "home page" of user area. Home page lists the most important information and contains links to other actions. To log users in and out actions "actionLogin()" and "actionLogout" are used.

Table 15: Site controller reference

Name	Description	Return
actionIndex()	Front page of the system, user is redirected to front page when logged in.	Renders three blocks: latest task, latest news and calendar with tasks listed by days.
actionLogin()	Check password hash and logs user into system if password is right.	Renders form if user is not logged in. Redirects to actionIndex if user is logged in.
actionLogout()	Logs user out of the system. If user is not logged in nothing happens	Logs user out and redirects to the login page. If user is not logged into the system, redirects to the login page.

Site controller is not only the default controller of the current project, it is the default controller of Yii2 templates overall. Normally it is not only used to render index, login and

logout pages, but other static pages as well. Simple websites may be built using single site controller.

4.6.2 News Controller

Methods used in News controller are used to view list of the articles or to view specific article. Following actions are implemented in news controller: “actionListNews” and “actionView”. First action lists all news articles in a grid, second action is used to view full article by unique id. Actions used in the News-related logic are described in Table 16.

Table 16: Actions used in News controller

Name	Description	Return
actionListNews()	Lists all news found in the system.	Renders GridView with all news found in the system.
actionView(id)	Views article by id if found.	Renders view with article. Throws error if no id found.

4.6.3 Task Controller

Tasks visible for user are taken from the latest selected course. Currently, there is no possibility to change selected course for user in the User Area, the latest course is considered the selected course by default. Task controller allows user to view and interact with tasks through actions “actionListAllTasks()”, “actionCompleteTask()” and “actionSaveTask()”.

The grid of all tasks is rendered upon calling action “actionListAllTasks”. To submit a task, the “actionCompleteTask” is used. The “actionCompleteTask” renders a submission form, which triggers the action “actionSaveTask” when submitted. When the form is sent, the task submit record is updated and an administrator can confirm user progress through the administration area. See Table 17 for more detailed information.

Table 17: Task controller reference

Name	Description	Return
actionListAllTasks()	Lists all tasks of selected course.	Renders GridView with all tasks of selected course. Single task in list consists of a name, deadline, task file and "Turn in" link (which redirects to actionCompleteTask()). If user has no selected courses, message "no course" is rendered.
actionCompleteTask()	Searches for TaskSubmit record related to selected task and current user. If TaskRecords is found – renders "turn in" form where user can submit data	If TaskSubmit is found, renders form where user can leave a comment for task. When "Submit" button is pressed, user is redirected to actionSaveTask() where TaskSubmit is saved.
actionSaveTask()	Saves TaskSubmit model and updates TaskSubmit database records.	TaskSubmit is saved and redirects to actionListAllTasks() and renders status message: "Task saved" or "Task not saved".

Main idea of the task controller in the user area is to provide a user with necessary functions to view and submit tasks of the current active course. Course controller in the user area is not implemented and a user is not able to switch between active courses and view tasks related to other courses.

4.6.4 User Controller

In Frontend, user controller is used only for account information management. Only one method is used in User controller: “actionViewUser()”. Method “actionViewUser()” renders form filled with the user data. For reference see Table 18.

Table 18: User controller reference

Name	Description	Return
actionViewUser()	Finds account of the logged user. If user identity ID and viewed user's ID do not match – action is not allowed. When form is submitted, user account is updated.	Renders account management form, where user can change the name, phone number and other personal information. When form is submitted user is redirected to ActionViewUser() again.

While implementing only one function, user controller in the user area is a necessity. A user must be able to change personal information at any time without problems.

4.6.5 Styling and HTML

Application follows overall Bootstrap framework guidelines to ensure responsiveness and good user experience. Backend consists mostly of pre-made styles, when frontend has got more attention and has more complicated HTML structure.

5 Usage Guide

5.1 Login

To log into the system user must input right credentials. If the data is not right or user does not have access to the area, user will be notified. Example of login screen can be seen in Figure 7.

My Company

Login

Please fill out the following fields to login:

Username

Password

Remember Me

Login

Figure 7: Default login form

Login page has common code in frontend and backend, but graphical design may differ.

5.1 Backend

Through the navigation bar the administrator can access specific areas of the system, like user or course screens. For detailed information of links listed in navigation bar see Figure 8 and Table 19.



Figure 8: Navigation in backend

Table 19: Explanation of navigation bar

Number	Description
1	Link to the homepage, there is no information on the homepage currently.
2	Calendar that shows all tasks by date. Currently not in use, since it's not ready for use.
3	Link to New User form, where user can be created.
4	List of all users in the system.
5	Dropdown with links to "new course" and "course list" pages.
6	Dropdown with links to "new article" and "news list" pages.
7	Logout link. Current user identity can be seen in brackets.

Navigation bar is divided into logical sections to provide a fast accessibility to different areas of the administration area. Links to subsections are hidden behind main links and can be identified by triangular symbols next to the links.

New user

Please fill out the following fields to add new user:

Username Username cannot be blank.

Password

Figure 9: New user form

When creating a new user, it is important to remember that username has to be in form of an email and be unique. Password has no limitations. If the username is already found in the system, user will be notified. (See Figure 9)

5.1.1 Manage Users

Users are managed through "List users" view. Example of "List user" view can be seen in Figures 10 and 11, for explanation see Table 20. "List users"- view lists all the users found in the system. Through this list administrator can delete or update users and user related data.

List users

Showing 1-2 of 2 items. ¹

Username ²	Name ³
	<input type="text"/>
admin@admin.fi	admin
ivan@t.fi	Ivan

Figure 10: First half of the user list

Surname ⁴	Date Of Birth ⁵	Actions
<input type="text"/>		
admin	1991-08-14	⁶ ⁷
Khokhlachev	(not set) ⁸	

Figure 11: Second half of the user list

Table 20: Explanation of user list

Number	Description
1	Total number of records on page and overall.
2	All data shown in the table except for the username is part of the account data, which is mutable by client. Username is immutable.
3	Name found in the account database table, can be sorted by alphabet.

4	Surname found in the account database table, can be sorted by alphabet.
5	Client's date of birth, can be sorted by date.
6	Link to view action, where account data can be changed. User editing will be discussed in the next section.
7	Button that is used to delete a user from the system completely.
8	When data is not set in the database, table will read "not set". If the data is not set, system will not crash and will continue to function properly.

The view of the user list allows easier access to the data through sorting and queries. By queries the most relevant data will be retrieved from the database.

5.1.2 Editing Users

When unique user from the list is chosen, administrator will be redirected to the user view. Personal data, password and measurement list may be altered through the user management screen.

User: admin@admin.fi 1

Name: admin

Surname: admin

Date Of Birth: 1991-08-14

Password: 2

[Change password](#)

Measurements 3

[Add row](#)

Showing 1-1 of 1 item.

Time Created ↓	Weight	Breast	Hip	Legs	Hand	Actions
2017-04-20	61.00	30.00	30.00	30.00	30.00	✎ ✕

Figure 12: User view

User editing is done through forms. It is good to remember that user data and password are different forms – when pressing "change password" button, it will not submit the form

with user data and vice versa. Visual example is seen in Figure 12, for explanation see Table 21.

Table 21: Explanation of user view

Number	Description
1	User data form
2	Password change form
3	Forms table, works like every list. Opens "add measurment" form when "Add row"- button is pressed

Measurements works in similar way to user screen. List of measurements can be queried and sorted, and records can be deleted and added. New records are added through form, where administrator enters information like dates and actual measurement data.

5.1.3 Course Management

On top of the course screen, active dates of course are viewed and they can be changed by altering "start date" and "end date" attributes in course form. When "participants" area is clicked list of available users will be shown. Administrator may pick users one by one or select all users at once. Administrator may change name, description and status of the course. For reference see Figure 13 and Table 22.

Tasks 4

Showing 1-1 of 1 item.

Name	Status	Start Date	Deadline	Actions
My first task	1	2017-04-20	2017-04-23	

Figure 13: Course editing view

Table 22: Explanation of course editing view

Number	Description
1	Course name and dates. First date is the starting date of course. Last date is the ending date of course.
2	Basic form for course data.
3	Users assigned to course. When "participants" field is clicked, dropdown with all unassigned users is shown.
4	Task management. Task can be viewed, edited or deleted. When a task is viewed, system shows user's progress on current task (done/not done)

Tasks may be queried and sorted by all attributes. Functionality is the same as in measurements, with one additional link which views list of submitted tasks in order to track user progress.

5.1.4 Tasks

Tasks are created through simple form, in which administrator may specify name and description of the task. Description may contain unlimited amount of text and may be styled with html tags. Additionally, attachment may be uploaded, currently only pdf and doc filetypes are supported. In frontend, uploaded files will be available through links. For additional information see Figure 14.

New task

Name

Description

Source | ↶ | ↷ | ✂ | 📄 | 📁 | 📁 | 📁 | 📁 | ?

☰ | ☰ | ☰ | ☰ | 🖼️ | 📄 | ☰ | Ω

B *I* U ~~S~~ x_a x^2 | I_x | 🗣️ | 🗣️ | 🚩

Do everything according to instructions in attached file

body p

Status


Not active

Active

Start Date: 2017-04-20

Deadline: 2017-04-23

Files



pdf.pdf
(423.82 KB)

pdf.pdf Remove Upload Browse ...

Create task

Figure 14: Task creation screen

When new task is created, list of submissions for every user assigned to course will be created. NB, if user is added to the course after task creation – task submit- record for new user will not be created.

5.1.5 List of Submitted Tasks

When a task is created, task submit records are created automatically for all users. Administrator can track current progress of task completion by clicking "view task" in the task list. See Figure 15 for example. In Figure 15, user "admin admin" has completed the task, when "Ivan Khokhlachev" did not.

Showing 1-2 of 2 items.

User	Completed
admin admin	<input checked="" type="checkbox"/>
Ivan Khokhlachev	<input type="checkbox"/>

Figure 15: List of users and task progress

Submission cannot be deleted manually, to delete submissions from the system corresponding task must be deleted. If corresponding task is deleted, all submissions will be lost.

5.2 Frontend

The user area does not have as many pages as the administration area, since users do not have to manage anything except for their personal information. The main purpose of the user area is to allow users to view and complete tasks.

5.2.1 Navigation

Navigation bar in the user area is similar to the one in the administration area. For explanation, see Figure 16 and Table 23.



Figure 16: Navigation bar

Table 23: Explanation of navigation bar in user area

Number	Description
1	Link to the first page.
2	List of all news.
3	List of all tasks.
4	Username of a currently logged in user. If clicked, will redirect to the account management form.
5	Logout button. When clicked, user will be logged out of the system and redirected to the login screen.

Navigation in user area is much lighter than one in administration area, since most of the information viewed by user is managed through two main screens: front page and “all tasks- page.

5.2.2 Front Page

When logged into the system, the user will see a page similar to the one in Figure 17. The most important information will appear on the main screen: latest assignment, list of latest news and a calendar with tasks.

The screenshot shows a user interface for a system. At the top is a navigation bar with a logo placeholder, and links for HOME, NEWS, ALL ASSIGNMENTS, ADMIN@ADMIN.FI, and a Logout button. Below the navigation bar is a 'WELCOME PAGE' section with a placeholder text. To the left, there is a 'LATEST ASSIGNMENT: MY FIRST TASK' card showing '10 / 13' and an image of two people exercising, with a 'Read more' link. Below that is a 'LATEST NEWS' section with two article links. To the right is a calendar for 'APRIL 2017' with a 'today' button and 'month'/'week' view toggles. The calendar grid shows dates from 1 to 30. A task 'My first task' is highlighted in a blue box on the 23rd of April.

Figure 17: Front page of the user area

Currently tasks shown in the calendar are the tasks, that belong to current or latest course. This functionality is implemented because it is not yet possible to change current active course for user.

5.2.3 Task Submitting

All tasks, completed and new, can be viewed through “all tasks”- link. On the “all tasks” page user will be presented with the name of a current course and list of all tasks relevant to current active course. See figure 18.

Home /

COURSE: MY FIRST COURSE

ASSIGNMENTS

Showing 1-1 of 1 item.

Name ↓	Deadline	Assignment file	Assignment
My first task	2017-04-23	PDF	Turn in! 📌

Figure 18: Task list

To submit a task, user has to navigate to the "All assignments" page. Tasks can be sorted by name and date. To complete a task, user has to click on the "Turn in!"- link.

5.2.4 User Management

By clicking on own name in navigation bar user will be redirected to the form, where personal data may be changed at any time. See Figure 19 for the example of the user form.

ADMIN ADMIN

Name

Surname

Sex

Date Of Birth

Phone

Instagram

Facebook

Figure 19: Data management form

All data that is changed through user form is not crucial to the system and may be changed at any time.

6 Discussion and Conclusions

Currently the backbone of the system is completed, and the system may be used for simple course and user management. However, a lot of features are still missing from the current implementation. Those include such features like ability to add and manage diverse files in a more approachable way, more complicated and versatile course management, and a more functional calendar. The basis for different modifications and additions is present in the system, and the current code may be considered as a template for a bigger and more extensive application.

The biggest problem in the planning of the system is, without doubt, the naming conventions which are not present in the most part of the code. Problems such as the names of user tables are confusing and make the system hard to read and understand. In the future, the naming problem must be the number one priority.

Overall, the system is built with the advantages of the Yii2 framework in mind, which conditions easy modification and extensibility of the code. The Yii2 framework made building of the project as fast and easy as it can get, since most of the difficult logic is already handled by the framework itself.

7 Sources

1 Official PHP documentation,

<http://www.php.net>

2 MVC Framework – Introduction,

https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm

3 Composer official documentation,

<https://getcomposer.org/doc/>

4 Yii2 official documentation,

<http://www.yiiframework.com/doc-2.0/guide-index.html>

5 Yii2 official documentation,

<http://www.yiiframework.com/doc-2.0/guide-index.html>

6 Apache official documentation,

http://httpd.apache.org/docs/current/mod/mod_rewrite.html

