

TALLINN UNIVERSITY OF TECHNOLOGY  
Faculty of Information Technology  
Department of Informatics

IDU40LT

Vladislav Goltjajev 135217IAPB

**DESIGN AND IMPLEMENTATION OF AN  
INFORMATION SYSTEM FOR  
DISTRIBUTING SOFTWARE**

Bachelor's thesis

Supervisor: Erki Eessaar  
Doctor of Philosophy  
in Engineering  
Associate Professor

Tallinn 2016

TALLINNA TEHNIKAÜLIKOOL  
Infotehnoloogia teaduskond  
Informaatikainstituut

IDU40LT

Vladislav Goltjajev 135217IAPB

# **TARKVARA JAOTAMISE INFOSÜSTEEMI DISAINIMINE JA REALISEERIMINE**

Bakalaureusetöö

Juhendaja: Erki Eessaar  
doktor  
dotsent

Tallinn 2016

## **Author's declaration of originality**

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Vladislav Goltjajev

28.04.2016

## **Abstract**

### **DESIGN AND IMPLEMENTATION OF AN INFORMATION SYSTEM FOR DISTRIBUTING SOFTWARE**

The aim of this bachelor's degree thesis was to create a universal and flexible software distribution web-based information system which would serve as a replacement for the existing system for distributing software and giving access to resources (remote places for getting software) for Tallinn University of Technology (TUT) students (<http://zaurus.ttu.ee/>). The aim was to create the software part of the system by using familiar technologies for the future maintainers, meaning using the PHP programming language and PostgreSQL as the database management system. The system is needed because distributors of software have to make sure that only certain people (in this case TUT students or staff) can access these.

An Information system is not only software and hardware but also processes and people around these. The newly created system was designed to be flexible in the sense that it is possible to specify for different resources different processes for distributing and granting access to these.

The usability of the new web-based system surpassed the old one with the addition of new functionality as well as a user-friendly interface using the Twitter Bootstrap CSS framework.

The thesis describes the structure of the project, describes its functionality, and is finished with a comparison between the old zaurus.ttu.ee user interface and the new software distribution system web interface.

The main result of this thesis was the creation of a functioning and customizable web-based system. It is in PHP and associated with a PostgreSQL database that allows students to request access to resources and files and download the requested files. The

software is general enough to be usable in case of other organizations that require similar functionality. The currently (as of May 2016) operational system can be found at: <http://viktor.ld.ttu.ee/software>.

This thesis is written in English and is 47 pages long, including 6 chapters, 24 figures and 1 table.

## **Abstract**

### **TARKVARA JAOTAMISE INFOSÜSTEEMI DISAINIMINE JA REALISEERIMINE**

Käesoleva bakalauruse lõputöö eesmärk oli luua universaalne ja kohandatav tarkvara jaotamise veebipõhine infosüsteem, mis asendaks olemasoleva TTÜ tarkvarajaotuse ja ressurssidele juurdepääsu (kaugkohad tarkvara allalaadimiseks) andmise süsteemi (<http://zaurus.ttu.ee/>). Eesmärgiks oli luua süsteemi tarkvara osa tulevaste haldajate jaoks tuttavaid vahendeid kasutades, täpsemalt kasutades PHP programmeerimiskeelt ja PostgreSQL andmebaasisüsteemi.

Infosüsteem ei ole mitte ainult tarkvara ja riistvara, vaid ka protsessid ja inimesed selle ümber. Uus süsteem kavandati paindlikuna, mis tähendab, et erinevate ressursside jaoks saab kirjeldada erinevaid jagamise ning pääsuõiguste andmise protsesse.

Uue veebipõhise süsteemi kasutatavus ületas vana oma, kuna lisati uut funktsionaalsust koos kasutajasõbraliku kasutajaliidesega kasutades Twitter Bootstrap CSS raamistikku.

Töö kirjeldab projekti struktuuri, liidese funktsionaalsust ja lõpeb uue ja vana tarkvara jaotuse süsteemi kasutajaliidese võrdlusega.

Töö põhitulemuseks oli funktsioneeriva ja kohandatava veebipõhise süsteemi loomine. See on kirjutatud PHPs ja on seotud PostgreSQL andmebaasiga. See võimaldab üliõpilastel küsida juurdepääsu ressurssidele ja tarkvara failidele ning laadida alla soovitud faile. Tarkvara on piisavalt üldine, et see oleks kasutatav teiste sama funktsionaalsust vajavate organisatsioonide puhul. Hetkel (2016. aasta mai seisuga) on töötav veebirakendus aadressil: <http://viktor.ld.ttu.ee/software>.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 47 leheküljel, 6 peatükki, 24 joonist, 1 tabel.

## List of abbreviations and terms

CASE	<b><i>Computer Aided Software Engineering</i></b> CASE tools are set of software application programs, which are used to automate systems development life cycle activities [1] .
CSS	<b><i>Cascading style sheets</i></b> CSS is a stylesheet language that describes the presentation of an HTML document [2] .
HTML	<b><i>HyperText Markup Language</i></b> HTML is a markup language for describing web documents [3] .
JavaScript	Client-side scripting language for dynamically controlling HTML elements [4] .
MIT license	Software distribution license originating from the Massachusetts Institute of Technology (MIT) that enforces very limited restrictions on the use of copyrighted software [5] .
PEAR	<b><i>PHP Extension and Application Repository</i></b> PEAR is a framework and distribution system for reusable PHP components [6] .
PHP	<b><i>PHP: Hypertext Preprocessor</i></b> PHP is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML [7] .
SQL	<b><i>Structured Query Language</i></b> Programming language used for accessing and manipulating relational databases [8] .
TUT	<b><i>Tallinn University of Technology</i></b>

## Table of Contents

1	<a href="#">Introduction.....</a>	13
1.1	<a href="#">Problem.....</a>	13
1.2	<a href="#">Objective.....</a>	14
1.3	<a href="#">Implementation methods.....</a>	14
1.4	<a href="#">Finding existing software distribution systems.....</a>	15
2	<a href="#">System Analysis.....</a>	16
2.1	<a href="#">Actors.....</a>	16
2.2	<a href="#">Areas of competence.....</a>	16
2.3	<a href="#">Use case scenarios.....</a>	16
2.3.1	<a href="#">Cross-cutting concerns.....</a>	16
2.3.2	<a href="#">Request management.....</a>	17
2.3.3	<a href="#">Resource management.....</a>	17
2.3.4	<a href="#">User management.....</a>	17
2.4	<a href="#">Database analysis.....</a>	18
2.4.1	<a href="#">Entity-relationship diagram.....</a>	18
2.4.2	<a href="#">State diagram of main request register object.....</a>	18
2.4.3	<a href="#">Registers.....</a>	18
3	<a href="#">Back-end Design.....</a>	19
3.1	<a href="#">Back-end implementation methods.....</a>	19
3.2	<a href="#">Database.....</a>	20
3.2.1	<a href="#">Security.....</a>	20
3.2.2	<a href="#">Data integrity.....</a>	20
3.3	<a href="#">PHP.....</a>	21
3.3.1	<a href="#">Security.....</a>	21
3.3.2	<a href="#">Database connection.....</a>	23
3.3.3	<a href="#">Views.....</a>	24
3.3.4	<a href="#">Language switching.....</a>	26



3.3.5	<a href="#">Emails.....</a>	<a href="#">26</a>
3.3.6	<a href="#">CSV file generation.....</a>	<a href="#">27</a>
3.3.7	<a href="#">File downloads.....</a>	<a href="#">27</a>
3.3.8	<a href="#">Home page content editing.....</a>	<a href="#">28</a>
4	<a href="#">Front-end Design.....</a>	<a href="#">29</a>
4.1	<a href="#">Prototype.....</a>	<a href="#">29</a>
4.2	<a href="#">Front-end implementation methods.....</a>	<a href="#">29</a>
4.3	<a href="#">Functionality.....</a>	<a href="#">30</a>
4.3.1	<a href="#">Home page.....</a>	<a href="#">30</a>
4.3.2	<a href="#">Navigation.....</a>	<a href="#">30</a>
4.3.3	<a href="#">Registration.....</a>	<a href="#">31</a>
4.3.4	<a href="#">Login.....</a>	<a href="#">32</a>
4.3.5	<a href="#">Resource list.....</a>	<a href="#">32</a>
4.3.6	<a href="#">Individual resource view.....</a>	<a href="#">32</a>
4.3.7	<a href="#">Add resource page.....</a>	<a href="#">35</a>
4.3.8	<a href="#">Request list.....</a>	<a href="#">36</a>
4.3.9	<a href="#">Individual request view.....</a>	<a href="#">36</a>
4.3.10	<a href="#">Account view.....</a>	<a href="#">38</a>
4.3.11	<a href="#">User list.....</a>	<a href="#">39</a>
4.3.12	<a href="#">Individual user view.....</a>	<a href="#">40</a>
4.3.13	<a href="#">Home page content editing.....</a>	<a href="#">40</a>
4.3.14	<a href="#">Application settings editing.....</a>	<a href="#">41</a>
5	<a href="#">Comparison Between the Old and New Versions of the System.....</a>	<a href="#">42</a>
6	<a href="#">Summary.....</a>	<a href="#">44</a>
	<a href="#">References.....</a>	<a href="#">45</a>
	<a href="#">Appendix 1 – Use Case Diagrams.....</a>	<a href="#">47</a>
	<a href="#">Appendix 2 – Entity-relationship Diagram.....</a>	<a href="#">50</a>
	<a href="#">Appendix 3 – Request State Diagram.....</a>	<a href="#">51</a>
	<a href="#">Appendix 4 – Physical Database Design Diagrams.....</a>	<a href="#">52</a>
	<a href="#">Appendix 5 – Database SQL Statement Examples.....</a>	<a href="#">55</a>
	<a href="#">Appendix 5 – Initial Interface Prototype.....</a>	<a href="#">56</a>
	<a href="#">Appendix 6 – New User Interface Screenshots.....</a>	<a href="#">58</a>



## List of figures

Figure 1: Code snippet showing the main security variables.....	21
Figure 2: Code snippet showing how non-logged in users are redirected from a page..	21
Figure 3: Code snippet shown how users without administrator rights are redirected from a page.....	22
Figure 4: Code snippet showing the functions that create a password hash and validate it.....	22
Figure 5: Use of the htmlspecialchars() function to prevent-cross-site scripting.....	22
Figure 6: Code snippet showing the clean_input() function.....	23
Figure 7: Server's IPS detecting an SQL injection.....	23
Figure 8: Basic structure of PHP files used to display information to users.....	24
Figure 9: Content example of translations_en.php (left) and translations_ee.php (right).....	26
Figure 10: Code snippet describing how .csv files are created for the administrator to download.....	27
Figure 11: Home page.....	30
Figure 12: Registration page.....	31
Figure 13: Resource page for requesters.....	33
Figure 14: Resource page for administrators (basic information).....	34
Figure 15: Resource page for administrators (request attribute and status controls)....	34
Figure 16: Add resource page.....	35
Figure 17: Request page for software.....	37
Figure 18: Submitted request.....	37
Figure 19: Accepted request.....	38
Figure 20: Account view.....	39
Figure 21: User list.....	39
Figure 22: User view.....	40
Figure 23: Content editing form.....	41

Figure 24: Application settings editing form.....41

## **List of tables**

Table 1: Pages shown to the user.....	24
---------------------------------------	----

# 1 Introduction

Students often find themselves in need of certain software, for which open source alternatives either do not exist or are poorly made. TUT offers students software downloads (for instance, Rational Rose and Enterprise Architect CASE tools) and access to external resources such as Microsoft DreamSpark to complement their studies. Although a system with this function already exists, it has poor usability and poor visual design.

The author of this thesis chose this particular topic because an upgraded system for software distribution would be a useful asset for TUT and it could be implemented straight away and replace the existing one.

## 1.1 Problem

The current TUT web-based software distribution system at [zaurus.ttu.ee](http://zaurus.ttu.ee) suffers from a number of problems:

- Poor visual design – bare HTML, almost no style or color, no consistent website structure.
- Poor user interface – limited navigation.
- Poor functionality – students do not have a clear overview of their submitted requests and their statuses, administrators do not get all the needed information in a submitted request, emails with request status updates are not sent to students, students have to register twice to get access to local software (files) and an external resource (Microsoft DreamSpark), no possibility to add new external resources for which students could submit requests.
- Poor accessibility – no possibility to change the website language.

- Errors in the implementation – students can lose access to the resources that they requested and are thus forced to submit requests multiple times.

## **1.2 Objective**

The objective of this thesis was the creation of a fully-functional software distribution system to replace the existing one. The list of improvements made:

- Improved visual design – application of a modern look to all website components.
- Improved user interface – added header and footer, improved resource list and own request history for students, improved submitted request list, resource list and user list view for administrators; improved individual request, resource and user views for administrators.
- Improved functionality – simplified process of adding new resources for administrators, added request attributes for resource requests to allow administrators to better weigh their decision concerning requests, added emails to requesters when a decision regarding their request is made, added comments for request decisions, improved generation of request lists as CSV files for administrators, added a fully customizable homepage, added secure file downloads straight from the request page, implemented language switching on the go, without losing query results
- Improved accessibility – added Estonian and English translations to accommodate both local and foreign students with the ability to easily add other language support

## **1.3 Implementation methods**

The author used model driven development in the sense that a lot of information about the requirements was received in the form of UML models from the supervisor. This information was elaborated and discussed in collaboration with the supervisor. The author created the database design model and the database implementation based on the

conceptual data model received as an input. The author used Enterprise Architect CASE tool and its model transformations for that purpose. PHP was chosen as the programming language for the creation of the website. Because the PHP version on the server at the time of the project implementation was 5.3.5, no frameworks were chosen and the code was written in basic PHP. PostgreSQL was chosen as the database system for the project. These systems were selected in order to simplify maintenance of the system by its administrators.

The user interface prototype was made manually with HTML and CSS. However, the resulting interface had little in common with the initial design. Feedback was continuously received from the thesis supervisor regarding the quality and usability of the interface to ensure maximum satisfaction with the end product.

The front end was written in HTML with Twitter Bootstrap 3 components. Animated components required jQuery and jQueryUI, JavaScript libraries. Form validation was implemented on the server side rather than the client side to avoid circumvention by disabling JavaScript. Secure software and CSV file downloads were handled with PHP scripts.

#### **1.4 Finding existing software distribution systems**

Attempts were made to find a software distribution system template online, however, they did not yield any relevant results. Examples of keywords used in Google search were “software download system php”, “software distribution system php”, “software download system template”, “software distribution system template”, “software distribution system php”.

The only somewhat relevant results found with these keywords were associated with limiting PHP software functionality through license keys.



## **2 System Analysis**

The base functionality of the resulting application has to conform to the following requirements, posed by the supervisor.

### **2.1 Actors**

- Administrator – responsible for request, resource, and user management
- Requester – user authorized to view active resources, submit requests, and download files
- Web guest – user that can only view the home page, register, and log in to the system

### **2.2 Areas of competence**

The system has three areas of competence:

- Administrator area of competence
- Requester area of competence
- Web guest area of competence

### **2.3 Use case scenarios**

In order to correctly analyse and implement actor-specific functionality, use case diagrams were used (Appendix 1).

#### **2.3.1 Cross-cutting concerns**

Both the administrator and the user needed to be identified in order to use the website (Figure 24).

### **2.3.2 Request management**

The administrators need to be able to view a list of all requests, view specific requests from that list, accept, reject or close the request, and view a list of accepted requests submitted in a specific periods and the emails of requesters associated with those requests (Figure 25).

The requesters need to be able to submit a request, view one's own submitted requests, and download the requested software files.

### **2.3.3 Resource management**

Both the administrators and the users need to be able to view a list of active resources (Figure 26).

The administrators need to be able to view a list of resources, both active and inactive, add, modify, activate and inactivate individual resources.

### **2.3.4 User management**

The web guests need to be able to register as a user of the system (Figure 27).

The requesters need to be able to view one's own data and modify it and modify one's own password.

The administrators need to be able to view one's own data and modify it, modify one's password, view the data of all users and modify it as well as their passwords, activate and inactivate individual users, and add and remove administrator rights from users.

## **2.4 Database analysis**

The created database accommodated all the previously discussed system requirements.

### **2.4.1 Entity-relationship diagram**

The database had to conform to the entity-relationship diagram, posed by Erki Eessaar (Appendix 2, Figure 28).

### **2.4.2 State diagram of main request register object**

The state diagram for requests was provided by Erki Eessaar (Appendix 3, Figure 29).

### **2.4.3 Registers**

The database contains four registers (Appendix 4):

- Classifier register (Figure 30)
- User register (Figure 31)
- Request register (Figure 32)
- Resource register (Figure 33)

## **3 Back-end Design**

The following chapter describes the back-end design of the application.

### **3.1 Back-end implementation methods**

Because the application is web-based, PHP was chosen as the programming language because it is optimized for creating and rapidly deploying such applications. It is also one of the most popular programming languages in the world, currently (as of May 2016) holding sixth place in terms of popularity [9] .

PHP does not require extensive configurations and compilation like Java, the code is executed each time a page is loaded. The only action needed to deploy an application on the server is to move the files containing the scripts into the HTTP server's web page directory.

Due to the fact that the PHP version on the server is 5.3.5, all modern frameworks are incompatible with it. Attempts were made to install the Laravel framework, the most advanced and popular PHP framework [10] . Laravel 4.2, the oldest Laravel version that is still maintained, requires PHP at least version 5.4. Yii 1.1, which was supported by the server, was not considered because the framework is deprecated and it would be unproductive to spend time learning it when Yii 2, the modern version of the framework, is completely different from its predecessor [11] .

Because of the aforementioned reasons and the fact that the application functionality was not very extensive, it was decided that implementing the application with basic PHP was the best solution.

For such a project, the database management system choice was not imperative, so PostgreSQL was chosen as the database management system of choice for the project

because it was already set up and configured on the server. It provides good functionality, and is familiar to the author.

## **3.2 Database**

The following subchapter describes the database used in the system.

### **3.2.1 Security**

Depending on the type of user currently browsing the application website, the connection to the database is carried out by different database users: `sd_guest`, `sd_requester`, `sd_administrator` – according to the areas of competence described in the previous chapter. These users have the minimal set of privileges that are needed to do their job in the database.

The database employs a public interface through the use of views and functions (Appendix 5, Figures 34-35). This was done to simplify the process of restricting access to certain tables for certain users.

In functions, the SECURITY DEFINER allows the function to act with the privileges of its creator, meaning that no specific table restrictions must be imposed for database users authorized to use the function. Therefore, all PUBLIC privileges were revoked from all tables to ensure maximum security.

The search path includes the `pg_temp` schema last in order to prevent the misuse of a SECURITY DEFINER function, due to that schema being writeable for everyone. The PUBLIC privileges of the functions were also revoked [12] .

### **3.2.2 Data integrity**

In order to preserve data integrity, ON UPDATE and ON TRUNCATE triggers were added to key tables. An ON UPDATE trigger was added to the request table, preventing invalid state changes that do not correspond to the request state diagram.

### 3.3 PHP

The following subchapter describes the PHP basis of the system.

#### 3.3.1 Security

The main security type that prevents unauthorized users accessing restricted pages is creating session variables that show whether or not a user is logged in and if the logged in user is an administrator or not. The session variables are set when a user successfully logs in (Figure 1).

```
if (validate_password($password, $user['password_hash'])) {
    if ($user['is_active'] == 'f') {
        $login_error = $tr_login_error;
    } else {
        if ($user['is_admin'] == 't') {
            $_SESSION['user_is_admin'] = true;
        } else {
            $_SESSION['user_is_admin'] = false;
        }

        $id = $user['user_id'];
        $_SESSION['user_id'] = $id;
    }
}
```

Figure 1: Code snippet showing the main security variables.

Other pages include the session\_handler.php file, which redirects the user to the index.php page if the user is not logged in (Figure 2).

```
if (empty($_SESSION['user_id'])) {
    header('Location: index.php');
    exit;
} else {
```

Figure 2: Code snippet showing how non-logged in users are redirected from a page.

Pages available only to administrators include the admin\_rights\_handler.php file, which redirects logged in users with no administrator rights to the home page, exactly like in the previous example (Figure 3).

```

<?php
if (!isset($_SESSION['user_is_admin'])
    || $_SESSION['user_is_admin'] == false) {
    header('Location: index.php');
    exit;
}

```

Figure 3: Code snippet shown how users without administrator rights are redirected from a page.

Because the PHP version on the server was 5.3.5 and password functions were only introduced in 5.5.0, secure password hashing was implemented with the help of an external library called phpass by Openwall [13] to enable hashing and validating passwords with the bcrypt algorithm.

Functions to create password hashes during user registration and password modification as well as validate passwords were implemented in the file functions.php (Figure 4), which also includes other functions that supplement the functionality of the application. The 10 in the PasswordHash constructor parameter indicates the number of passes, the same number as with modern PHP versions' native password\_hash function. The FALSE indicates the use of the existing bcrypt algorithm, which is implemented in PHP 5.3 and above.

```

function get_password_hash($password)
{
    $hasher = new PasswordHash(10, FALSE);
    return $hasher->HashPassword($password);
}

function validate_password($password, $password_hash)
{
    $hasher = new PasswordHash(10, FALSE);
    return $hasher->CheckPassword((string)$password, $password_hash);
}

```

Figure 4: Code snippet showing the functions that create a password hash and validate it.

Cross-site scripting is prevented by the use of the htmlspecialchars() PHP function, which translates injected script tags into HTML entities, preventing their execution (Figure 5).

```

<form id="login-form" method="post" action="<?= htmlspecialchars($_SERVER["PHP_SELF"]) ?>" role="form">

```

Figure 5: Use of the htmlspecialchars() function to prevent-cross-site scripting.

All user input is also checked for cross-site scripting and SQL injections with the `clean_input()` function (Figure 6).

```
function clean_input($data)
{
    global $connection;
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    $data = pg_escape_string($connection, $data);
    return $data;
}
```

Figure 6: Code snippet showing the `clean_input()` function.

SQL injections were also prevented with parameterized queries using the `pg_query_params` function and passing user input as an array of values, ensuring that the user input is inserted as values and there is no interpolation of malicious SQL statements into the query strings [14].

The efficiency of the implemented SQL injection prevention measures could not be tested due to the server at `viktor.ld.ttu.ee` containing an intrusion prevention system that detects SQL injections. A message (Figure 7) appeared in the browser window when entering “`qwe@ttu.ee; drop table droppable --`” in the user list query, the injection was prevented and the client computer’s IP address was blocked from the server.

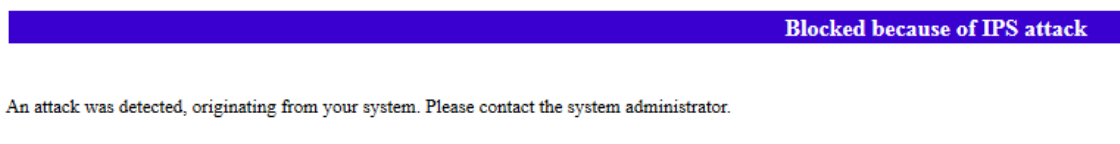


Figure 7: Server's IPS detecting an SQL injection.

### 3.3.2 Database connection

The connection to the database, which is required for the functionality of almost all PHP pages in the project, was done through the `db_connection_handler.php` file using the `pg_connect()` function.



For list pages, a separate class PostgreSQLPaginator.php [15] was created. Its functionality is getting a list of objects from the database for a specific page as well as generating HTML of the page indicator bar displayed below each list.

For individual SQL queries, a separate class PostgreSQLConnector.php was created. The class deals with SQL queries that accept parameters and the ones that do not, as well as SQL queries that are known to produce results and the ones that are not.

The PHP functions that connect to the database and retrieve data in the aforementioned classes were preceded with the @ marker to suppress error warnings as well a try-catch block in order to avoid leaking system and database information to the user in case of errors.

### 3.3.3 Views

Because no frameworks were used, most of the project structure follows the basic model of PHP code on top and HTML on the bottom for files that interact with the user (Figure 8) [16].

```
<?php
$var1 = 1;
$var2 = 2;
?>
<!DOCTYPE html>
<html>
<body>
<p><?= $var1 ?> + <?= $var2 ?> = <?= $var1 + $var2 ?></p>
</body>
</html>
```

Figure 8: Basic structure of PHP files used to display information to users.

Pages the website user sees in the browser are located in the root project directory, for each page there is a separate file (Table 1).

Table 1: Pages shown to the user.

<b>File name</b>	<b>Function</b>
add_resource.php	Allows administrators to add resources.
edit_content.php	Allows administrators to modify the HTML and PHP content of the home page.
edit_configurations.php	Allows administrators to modify email filters for user registration and change user password settings.
index.php	Displays basic website information specified by the website administrator.
login.php	Allows web guests to log into the system.
my_account.php	Displays own user data to administrators and users. Allows administrators and users to modify one's own data and change one's password.
register.php	Allows web guests to register in the system.
request_list.php	Displays own requests for users and a full list of requests for administrators. Allows administrators to find requests by requester or resource name and submission date and download a .csv file with the search results.
request.php	Displays request information for users and administrators. Allows administrators to accept or reject submitted requests and close accepted requests. Allows users to download files if the request is accepted.
resource_admin.php	Displays resource information for administrators and allows them to edit it.
resource_list.php	Displays a list of active resources for users and a full list of resources for administrators. Allows users and administrators
resource.php	Displays resource information for users and allows them to submit a request.
user_list.php	Displays a full list of users for administrators. Allows administrators to search for users by name.
user.php	Displays user information for administrators. Allows administrators to activate, deactivate the user, add and remove administrator rights from the user, modify user data and modify the user's password.

The user also sees the header and footer, which are separate PHP files included in each of the aforementioned page files.

### 3.3.4 Language switching

The footer contains links to the current page with current URI parameters, but adds another parameter, “lang”, to it. The website supports two languages: English and Estonian. When a user first views the website, Estonian is set as the default language. After a user clicks on the language change icons in the footer, the page reloads with new translations whilst retaining the previously present URI parameters, such as query parameters for list pages.

All the text labels on the project, aside from the home page content, are parameterized and loaded from a PHP file. There are two PHP files containing translations: translations\_ee.php and translations\_en.php (Figure 9). The Estonian translations were provided by Erki Eessaar.

<pre>\$tr_home_page = "Kodu"; \$tr_register = "Registreeri"; \$tr_log_in = "Logi sisse";</pre>	<pre>\$tr_home_page = "Home"; \$tr_register = "Register"; \$tr_log_in = "Log in";</pre>
--	---

Figure 9: Content example of translations\_en.php (left) and translations\_ee.php (right)

In order to add a new language, a new file named “translations\_<language\_code>.php” must be added to the content directory of the application with the same variable names as the other translation files. A country flag icon must be added to the img/flags directory of the application and referenced as a link in the footer.php file in the same manner as the other language switching links. The flag icons for the application were taken from [GoSquared](#). The index page is shown in English for all new added languages, only the interface language is affected.

### 3.3.5 Emails

To improve user experience, emails were implemented to notify requesters of the status of their requests whenever the request was accepted, rejected, or closed.

The email sending system was implemented with the PEAR Mail extension. A class called Mailer.php was created to simplify the process of sending emails. The SMTP server used for emails was TUT's own onyx.ttu.ee. When the administrator makes a decision regarding a request and adds a comment, the comment is also shown in the email. For automatically accepted requests emails are not sent out.

### 3.3.6 CSV file generation

Administrators are able to generate a semicolon-separated CSV file of requests that fit the search criteria and ordering. The script accepts the same search parameters as the request list page.

The PHP script request\_csv\_generator.php (Figure 10) then generates the CSV file and offers it for the administrator to download.

```
if (!empty($results)) {
    header('Content-Type: text/csv; charset=utf-8');
    header('Content-Disposition: attachment; filename=requests.csv');
    $output = @fopen('php://output', 'w');

    foreach ($results as $row) {
        @fputcsv($output, $row, ";");
    }
} else {
    die($str_no_requests_found_warning);
}
```

Figure 10: Code snippet describing how .csv files are created for the administrator to download.

### 3.3.7 File downloads

Authorized users with accepted requests for software can download files from the request page. Downloads are handled with the download.php script, which accepts the software ID and the file name to download, for example,

<http://viktor.ld.ttu.ee/software/download.php?id=34&file=README.txt>.

It is not possible to download another file on the system by entering, for example, “../secret\_directory/secret\_file.txt” because the script strips the base name of the file from the specified path and searches for the file only in the directory associated with the

specified software ID. The currently logged in requester's ID is also checked to see if it corresponds to an accepted request for the specified software.

If the validation is successful, the script then sends out headers, the most important of which is the content header "application/octet-stream", which forces any file type to be downloaded, rather than displayed in the browser (in case of text files) [17] .

### **3.3.8 Home page content editing**

The project features a fully-customizable home page. The content can be filled with HTML, JavaScript and PHP. The content files `content_en.php` and `content_ee.php` are located in the same folder as the translation files.

Editing the page occurs from the user interface and not necessarily through direct editing of the files on the server.

For the editing to work, the two content files must have write permissions enabled.

## **4 Front-end Design**

The following chapter describes the front end design of the application.

### **4.1 Prototype**

The initial prototype (Appendix 5, Figures 36-38) for the user interface was written in HTML. The final design differs greatly from the prototype.

It was decided that implementing the user interface with Bootstrap components would be the most time-efficient means rather than writing the HTML manually, the same was as for the prototype.

### **4.2 Front-end implementation methods**

The front-end was designed with Twitter Bootstrap 3, a free HTML/CSS framework released under the MIT license. It provides a modern look and the ability to make a fully-customizable user-friendly interface.

Some elements, such as drop-down lists and the date picker, required the inclusion of jQuery. JQuery, also released under the MIT license is one of the most popular JavaScript libraries [18] .

Mobile view for the website was not considered because it was expected that both students and administrators would use the website on their PCs rather than mobile devices, considering how the software available is intended to be installed on PCs.

It was decided to base the design on the “10 Usability Heuristics for User Interface Design” [19] .

The resulting design was tested in Firefox 45.0.2 and Internet Explorer 11.0.9600.18282.

## 4.3 Functionality

The following subchapter describes the functionality of each view.

### 4.3.1 Home page

The current address (as of May 2016) of the application is <http://viktor.ld.ttu.ee/software>. The user is first presented with a title page with content specified by the administrator (Figure 11).

The screenshot shows the home page of the TUT Software Distribution application. At the top, there is a navigation bar with a 'Home' link and 'Register' and 'Log in' buttons. The main heading is 'TUT Software Distribution'. Below this, there is a paragraph explaining the service: 'Requesting software or access to it through some other environment (Dreamspark) that is needed in the informatics or business information technology related courses in the Tallinn University of Technology.' A blue button labeled 'Currently available resources' is positioned below the text. The Microsoft DreamSpark logo is centered on the page. Underneath the logo, there is a paragraph about registration: 'In this page you can register to TUT-related Microsoft DreamSpark Premium program. In registration page you can create request for new DreamSpark account. Below you find some possible questions and answers about registration.' This is followed by a 'Frequently Asked Questions' section with two Q&A pairs. The footer contains the copyright notice '© Tallinna Tehnikaülikool' and language selection icons for Estonian and English.

Figure 11: Home page.

The user is redirected to the home page when trying to access restricted pages and when logging out.

### 4.3.2 Navigation

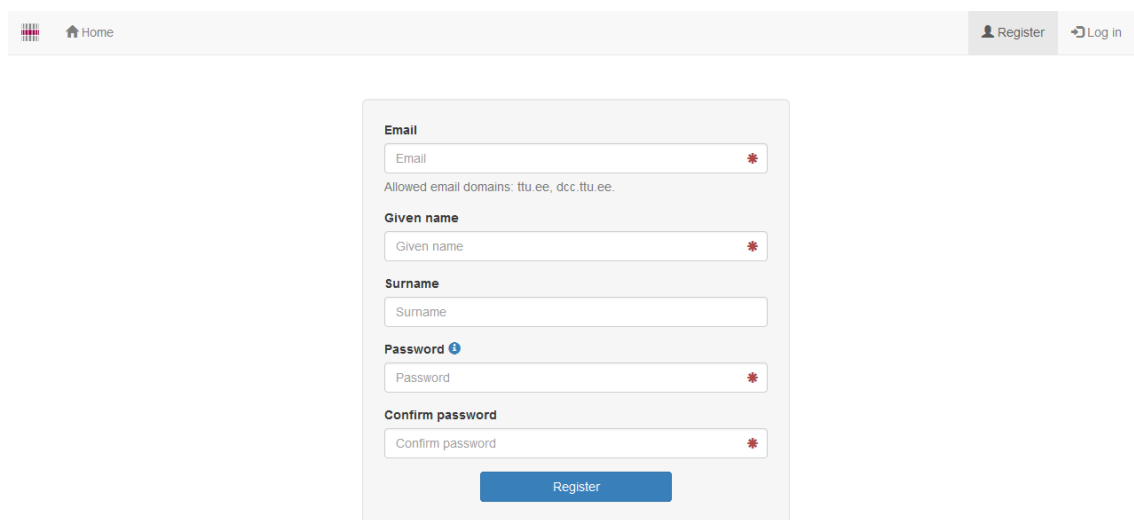
The main navigation occurs through the header, which is different depending on the user's account type and reflects the allowed user actions (Appendix 6, Figures 39-42).

Administrators can also see how many new requests there are that require manual accepting. A badge indicating the number is shown next to the “Requests” tab.

The main functionality of the footer (Appendix 6, Figure 43) lies in the ability to change the system language. In order to do that, the user can simply click on the flags in the bottom right corner.

### 4.3.3 Registration

The registration page (Figure 12) contains the registration form. The mandatory fields are indicated by red asterisks.



The screenshot shows a registration form on a web page. At the top, there is a navigation bar with a home icon and the text 'Home', and on the right, 'Register' and 'Log in' buttons. The form itself is centered and contains the following fields:

- Email**: A text input field with a red asterisk to its right. Below it, a note reads 'Allowed email domains: ttu.ee, dcc.ttu.ee'.
- Given name**: A text input field with a red asterisk to its right.
- Surname**: A text input field.
- Password**: A text input field with a red asterisk to its right and a small blue information icon to its left.
- Confirm password**: A text input field with a red asterisk to its right.

At the bottom of the form is a blue button labeled 'Register'.

Figure 12: Registration page.

For all required fields, not only on this page, but on the others as well, the “required” option in the HTML input tag is used, with server-side validation in PHP complementing it in case the user’s browser does not support this option (for example, Safari).

Email validation is primarily handled with the `<input type=”email”>` HTML tag, however, a PHP validation function using the `FILTER_VALIDATE_EMAIL` filter exists in case the user’s browser does not support this type of input. The email domain is also checked to ensure that the user has a valid University email. Other forms of email validation on the server side include the maximum length of 254 characters [20] and whether or not the email is already taken by another user.



Only the first name is made mandatory to allow for mononymous users with no last names to register [21] . The given name and surname are checked for the maximum length of 100 characters.

The password must contain at least the number of characters specified by the administrator, include a lower case letter, an upper case letter, and a number. The password must then be entered another time in order for the user to be sure that the right password is entered. Registration error messages are shown next to each field.

If entering valid data, in this case, test.user@ttu.ee with the password Password123 and given name User, the user is notified that registration is successful. The user can then immediately log into the system and start using it.

#### **4.3.4 Login**

The login page (Appendix 6, Figure 44) features the typical views one would find in most login pages: an email and password fields.

In case the user enters a wrong password or the email does not exist in the database, a uniform error is shown static only that the user name or password was invalid. The error is the same for both cases, in order to prevent potential data leakage to hackers.

The page also contains a link to an advanced password management system developed by Eerik Mägi, a TUT student [22] . It is possible to change and recover forgotten passwords using that system.

#### **4.3.5 Resource list**

The resource list page is different for the requesters (Appendix 6, Figure 45) and administrators (Appendix 6, Figure 46). The user is only displayed active resources, whereas administrators can view all resources. Both account types can search for resources based on their names.

#### **4.3.6 Individual resource view**

From the list page, requesters are redirected to the page containing basic resource data and a form for submitting a request for the selected resource (Figure 13). If the resource

has any request attributes the user must provide values for the mandatory attributes but can skip optional attributes. For instance, one might use attributes to collect information that is needed in order to give a software license to the requester. If a description of a request attribute is available, then it is displayed in both the text area placeholder and the tooltip. The input for each attribute must not exceed 1000 characters.

After a requester submits a request, the requester is redirected to the submitted request page, which will be discussed later.

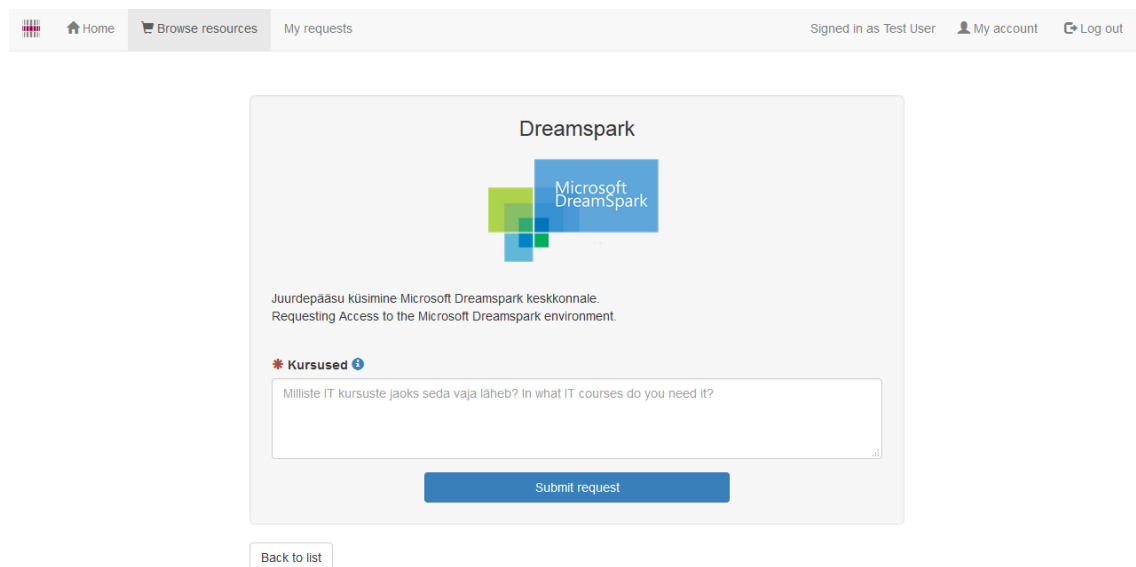


Figure 13: Resource page for requesters

If the requester views the same resource again, a message appears indicating that repeating requests are not allowed with no ability to submit another request.

From the resource list page, administrators are redirected to a different resource view (Figures 14-15). On this page, administrators can activate or deactivate the resource, edit the product name, description, whether or not manual accepting is required, software location, and the request attributes presented to requesters. Request attributes can be added and deleted, activated or deactivated, and made mandatory or optional.

Resource description can contain HTML, and a preview of the description can be seen below the field so that the administrator can see how requesters would see the description.

When changing the resource's name, a check is made to determine if that name already exists in the database. If it does, an error is displayed under the resource name field.

When specifying a software location, the administrator can immediately see whether or no the entered file path is correct by the text output below the field. The directory is scanned and the file contents are displayed. Software locations must also be unique for each added resource and if a software location currently in use is specified, an error is shown under the software location field.

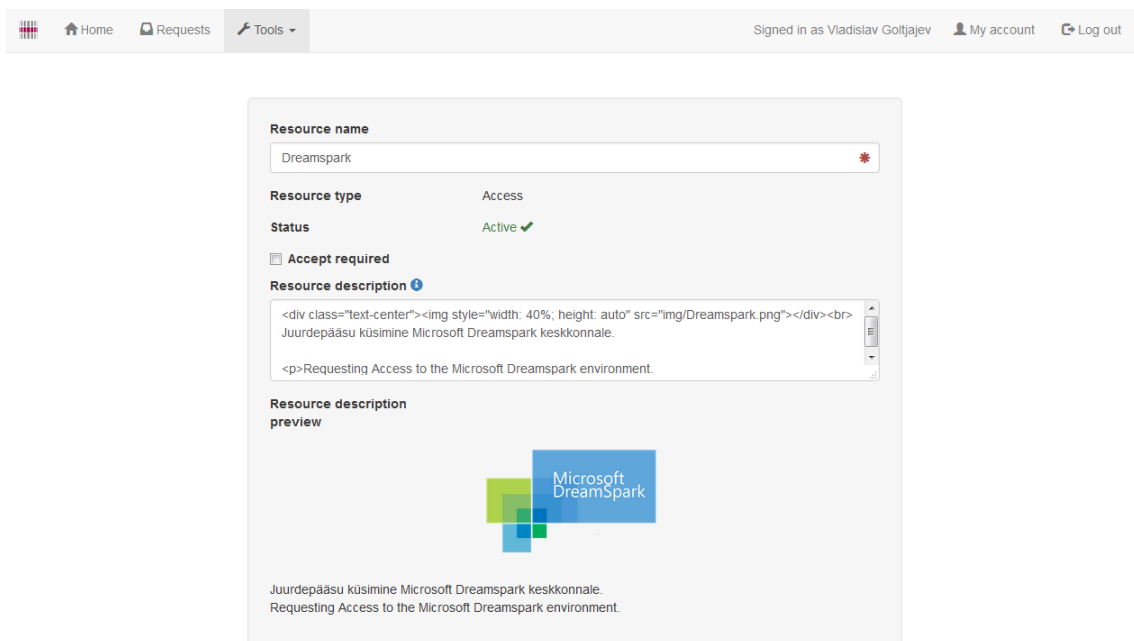


Figure 14: Resource page for administrators (basic information).

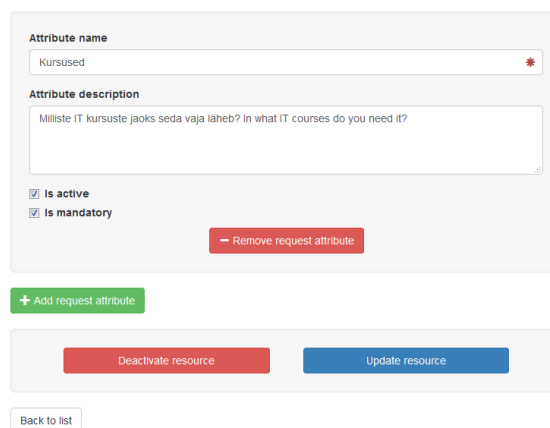


Figure 15: Resource page for administrators (request attribute and status controls).

Activating and deactivating a resource is done by pressing the “activate resource” and “deactivate resource” buttons. When the resource is inactive, it cannot be updated and new request attributes cannot be added. The only action available is activation. To update request attributes, the administrator must make all the needed changes and press “update resource”.

When removing a request attribute, a check is made to determine if there is a submitted request which contains a value associated with this attribute. If a value exists, an error message is displayed near the attribute marked for deletion. It is possible to make such attributes inactive, meaning that one does not see the attributes and thus does not have to provide values for these in case of new requests.

Request attribute names must be unique for a resource. If the user enters duplicate attribute names, errors are shown stating that the attribute name already exists for the selected resource. If all changes are valid, a success message appears when updating the resource.

#### 4.3.7 Add resource page

The add resource page contains a form administrators can fill out to add a resource (Figure 16). If the resource type is software, an additional field to enter the software location is added.

The screenshot shows the 'Add resource' page. At the top, there is a navigation bar with 'Home', 'Requests', and 'Tools' menus. On the right, it says 'Signed in as Vladislav Goltjajev' with links for 'My account' and 'Log out'. The main form is titled 'Add resource' and contains the following fields and controls:

- Resource name:** A text input field with a red asterisk indicating a required field.
- Resource type:** Radio buttons for 'Software' (selected) and 'Access'.
- Resource location:** A text input field with a red asterisk, only visible when 'Software' is selected.
- Accept required:** A checkbox.
- Resource description:** A text area.

Below the form, there is a green button labeled '+ Add request attribute' and a blue button labeled 'Add resource'.

Figure 16: Add resource page.

The resource name and location, in the case of software, are checked to determine if they already exist in the database. If they do, an error is shown under the fields. If a request attribute is added and the name is not filled, an error is shown under the attribute name field.

If the entered data is valid, the resource is registered and a success message is shown.

#### **4.3.8 Request list**

Requesters can only see requests submitted by themselves. Administrators can view the full list of requests (Appendix 6, Figures 47-48).

In the query form, administrators can specify requester first and last names as well as requested resources, request statuses, and find requests submitted after a specific date. If no statuses are selected, a search is performed for all statuses.

The date picker [23] was implemented with jQuery and jQuery UI libraries. It supports localizations (Appendix 6, Figure 49) and does not select future dates and dates older than 01.01.2000.

The “CSV” button allows administrators to download the request list fitting the search criteria to a .csv file.

If no requests are found, a message is shown and the CSV button is hidden.

#### **4.3.9 Individual request view**

Requesters, after submitting a request for a resource, are redirected to the request page (Figure 17). If the resource does not require manual accepting by administrators, a message is shown that the request is automatically accepted. The page contains the request data, submitted request attribute values, and files available for download if the resource type is software.

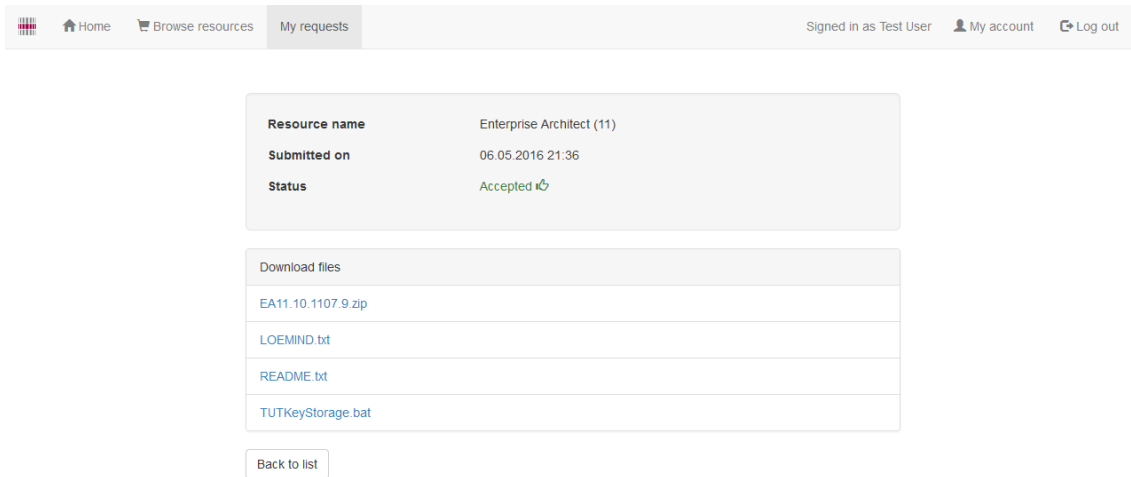


Figure 17: Request page for software.

The user can immediately start downloading the files if the request is accepted by selecting the file links. If the request is later closed, the download links are removed and the user is not able to download files any more even if manually using a previously acquired download link.

For administrators, the request page contains a form for accepting, rejecting, or closing requests (Figures 18-19). If a request is submitted, the administrator can accept or reject it. An optional comment can be included, which will be shown in the request information as well as the email sent to the requester.

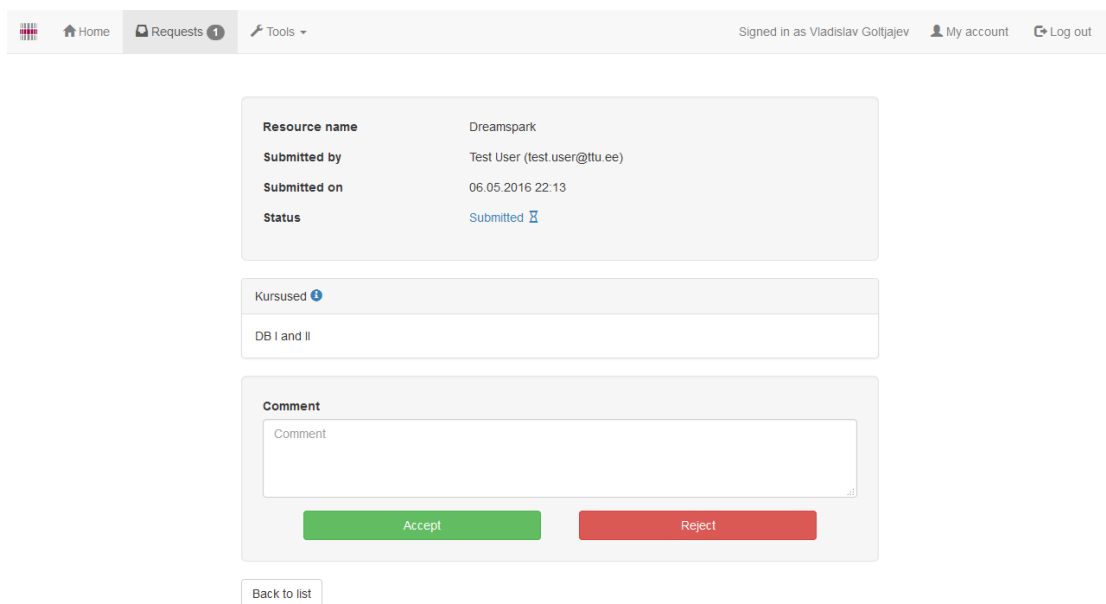


Figure 18: Submitted request.

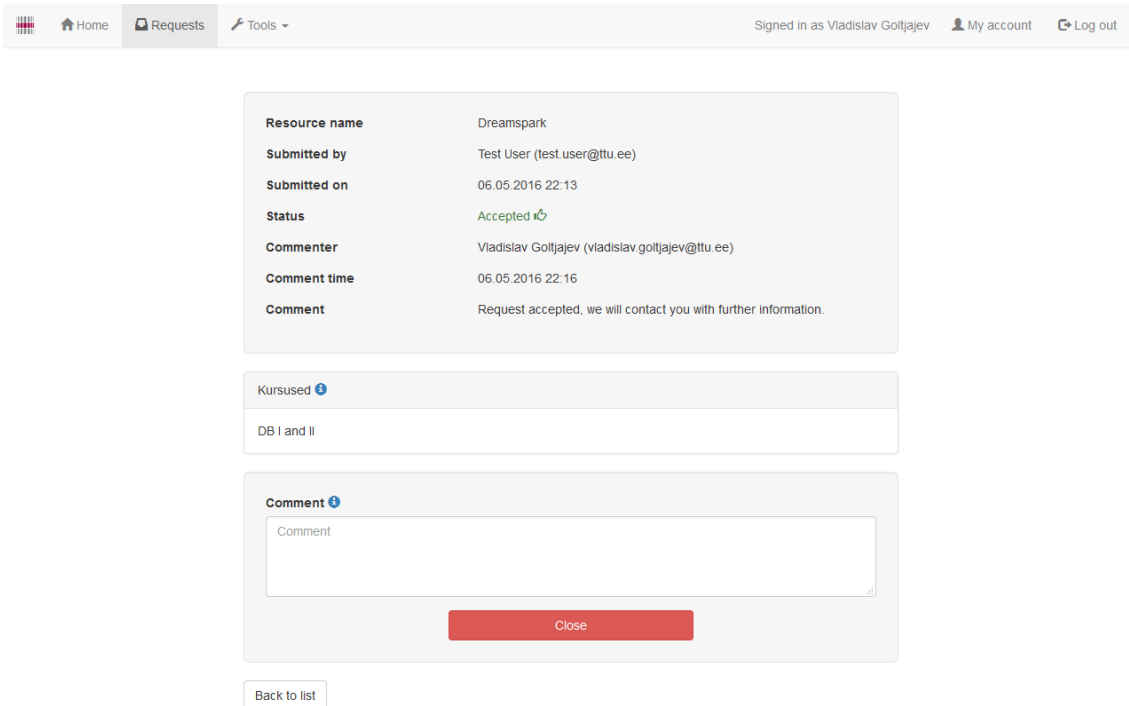


Figure 19: Accepted request.

After accepting or rejecting a request, a success message will appear. If the request status is “accepted”, the administrator can close the request and disable access to the resource. The comment submitted during request closing will overwrite the existing comment made during request accepting.

After a request is closed, a success message is shown and no further actions can be done with the request.

#### 4.3.10 Account view

For both requesters and administrators, the account view is the same (Figure 20). The page contains two forms: one for updating account details and the other for updating the password. The form validation is done in the same manner as with user registration, with the single addition of the old password field.

The account view also contains a link to the aforementioned advanced password changing system.

**Email**

**Given name**

**Surname**

**Last login** 22.05.2016 18:01

[Update account](#)

**Old password**

**New password**

**Confirm new password**

[Advanced password management](#)

[Change password](#)

Figure 20: Account view.

The user must enter the old password and specify the new one two times in order to change it. If the old password is invalid or the new password is too weak or does not match the confirmation password, errors are displayed below the fields. If valid data is entered, a success message is shown.

### 4.3.11 User list

Administrators can view the full list of users registered in the system (Figure 21). The query form allows searching for user first names, last names and emails.

Home Requests Tools Signed in as Vladislav Gottjajev My account Log out

Name	Email	Account type	Status	Last login
Eerik Mägi	eerik.magi@ttu.ee	User	Active	18.05.2016 01:56
Erki Eessaar	Erki.Eessaar@ttu.ee	User	Active	22.05.2016 17:16
Test Admin	test.admin@ttu.ee	Administrator	Active	20.05.2016 00:06
Test User	test.user@ttu.ee	User	Active	22.05.2016 18:01
Vladislav Gottjajev	vladislav.gottjajev@ttu.ee	Administrator	Active	23.05.2016 02:15

« 1 »

Figure 21: User list.



### 4.3.12 Individual user view

From the user list page, administrators are redirected to the user view page (Figure 22).

The screenshot shows a web application interface for viewing and editing a user. At the top, there is a navigation bar with 'Home', 'Requests', and 'Tools' menus, and a user profile section indicating 'Signed in as Vladislav Goltjajev' with 'My account' and 'Log out' links. The main content area is divided into three sections. The first section is a form for user details, with fields for 'Email' (test.user@ttu.ee), 'First name' (Test), and 'Last name' (User). Below these fields is a table showing 'Account type' as 'User', 'Status' as 'Active', and 'Last login' as '06.05.2016 22:12'. A blue button labeled 'Update user details' is positioned below the table. The second section contains two buttons: a red 'Inactivate' button and a green 'Add administrator rights' button. The third section is a form for changing the password, with fields for 'New password' and 'Confirm new password', and a blue 'Change password' button.

Figure 22: User view.

Administrators can update the user's details, activate and deactivate the user, add and remove administrator rights from user and update a user's password. Form validation is done the same way as in the account view page and the success and error messages are identical.

When viewing own user details, administrators cannot remove administrator rights and deactivate themselves.

### 4.3.13 Home page content editing

Administrators can edit the HTML and PHP content of the home page (Figure 23). After editing the code and pressing the "save" button, the page content is immediately reflected.

If the files that contain the content are not found or do not have write permissions, an error message is displayed for each file.

**Home EE:**

```

<div class="text-justify">
  <div class="text-center">
    <h2><?= $tr_sd_title ?></h2>
  </div>
  <div class="vspacer-20px"></div>

  <p>TTÜs informaatika ja äriinfotehnoloogia erialade jaoks vajalike programmide allalaadimine või neile juurdepääsu taotlemine mõnes teises keskkonnas (Microsoft Dreamspark) </p>

  <?php
$connector = new PostgreSQLConnector($connection);
$sql = "SELECT software_id, access_id, name FROM active_resource_list_limited";
$results = $connector->executeQueryWithResult($sql);
?>

  <?php if (!empty($results)): ?>

```

**Home EN:**

```

<div class="text-justify">
  <div class="text-center">
    <h2><?= $tr_sd_title ?></h2>
  </div>
  <div class="vspacer-20px"></div>

  <p>Requesting software or access to it through some other environment (Dreamspark) that is needed in the informatics or business information technology related courses in the Tallinn University of Technology.</p>

  <?php
$connector = new PostgreSQLConnector($connection);
$sql = "SELECT software_id, access_id, name FROM active_resource_list_limited";
$results = $connector->executeQueryWithResult($sql);
?>

  <?php if (!empty($results)): ?>

```

[Save](#)

Figure 23: Content editing form.

#### 4.3.14 Application settings editing

Administrators can specify which email domains are allowed during new user registration, the minimum password length and password recovery resource links (Figure 24).

Home Requests Tools Signed in as Vladislav Goltjajev My account Log out

**Edit email filters**

Enter permitted email domains for new users. Current users will not be affected. Each domain must be on a new line and MUST NOT contain the '@' character. If the domain list is empty, all emails are permitted.

ttu.ee  
dcc.ttu.ee

**Edit password settings**

**Minimum password length**

10

**Link to password changing resource**

http://viktor.id.ttu.ee/passwords/

**Link to password recovery resource**

http://viktor.id.ttu.ee/passwords/?page=ForgotPassword

[Save](#)

Figure 24: Application settings editing form.

## **5 Comparison Between the Old and New Versions of the System**

The current version of the system, located at <http://zaurus.ttu.ee/>, has many drawbacks as compared to the newly created system (Appendix 7).

The first thing one notices is the almost complete lack of any design and the prevalence of grey color in all elements (Appendix 7, Figure 50). The first distinguishing lack of functionality is seen in the inability to change the user interface language, which greatly limits its accessibility to foreign students. This limitation is fixed in the new system version by allowing users to switch between English and Estonian.

The login page (Appendix 7, Figure 51) contains unnecessary information about available resources.

The registration page (Appendix 7, Figure 52) does not explicitly show which fields are required and which are not. It was presumed that the bold field labels were supposed to be indicative of the fields being mandatory, however, a red asterisk, the common indicator in such cases, would be more suitable. The only user information administrators can base their decisions on are the student code and studied IT subjects, which allows for poor flexibility in terms of adding resources that require more information about the requester to base the request decision on (Appendix 7, Figure 53).

Adding and editing resources (Appendix 7, Figure 54, provided by Erki Eessaar) offers some degree of flexibility, however, the inability to specify resource-specific request attributes is a considerable disadvantage.

Users are divided into two groups, ones who apply specifically for the DreamSpark program and those who apply for software downloads. There is a separate registration form for DreamSpark accounts.

The DreamSpark user type page, contains only the option to update user details. The page also contains unnecessary available resource information, when there is no option of making requests.

Error handling on the system is also poorly implemented, seeing that PHP errors show debug data to the user.

Everything is incoherently cramped together on one page. The new design addresses this issue by splitting up the website part of the system into separate pages, each with its own function. This is an example of applying separation of concerns design principle [24] .

## 6 Summary

The goal of this thesis was to design and implement a universal system for distributing software and granting access to other resources for students. The system had to be written in PHP and connected to a PostgreSQL database, and be an upgrade to the currently existing TUT software distribution system.

The set goal was completely fulfilled. The system is functional and ready to be used. Language support for the system is not limited to English and Estonian. If necessary, it can be expanded to include other languages with minimal code changes.

The implemented system is generic enough to be usable in case of other organizations that need the same functionality. As it sometimes happens in the software field, after implementing the system one starts to see other possible use cases of the system. In this case the system can be used to distribute other content as well assuming that the content is in files. The only problem in case of the current implementation is that it refers to the concept "software" not to more generic concept "content". Regarding the original task of the system one should not consider this as a weakness.

Due to time constraints, some features were left unimplemented, such as sorting resource, request and user lists, which may be considered as a possible future improvement.

Another possible improvement would be rewriting the website part of the system using the Laravel PHP framework. Among other features, such as pagination, emails, localizations, support for multiple databases with a simple query interface, it would not be necessary to implement own measures of security, as they would already be available out of the box, such as cross-site scripting prevention, user authentication and even a system for resetting passwords.

## References

- [1] Software Case Tools Overview – TutorialsPoint (2016) – [http://www.tutorialspoint.com/software\\_engineering/case\\_tools\\_overview.htm](http://www.tutorialspoint.com/software_engineering/case_tools_overview.htm) [Online] (21.05.2016)
- [2] CSS Tutorial – W3Schools (2015) – <http://www.w3schools.com/css/> [Online] (10.03.2016)
- [3] HTML Introduction – W3Schools (2016) – [http://www.w3schools.com/html/html\\_intro.asp](http://www.w3schools.com/html/html_intro.asp) [Online] (10.03.2016)
- [4] About JavaScript – Mozilla Developer Network (2016) – [https://developer.mozilla.org/en-US/docs/Web/JavaScript/About\\_JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript) [Online] (16.03.2016)
- [5] MIT License – Open Source Initiative – <https://opensource.org/licenses/MIT> [Online] (17.04.2016)
- [6] PEAR - PHP Extension and Application Repository – The PHP Group (2016) – <https://pear.php.net/> [Online] (25.04.2016)
- [7] PHP – The PHP Group (2016) – <http://php.net/> [Online] (25.02.2016)
- [8] SQL Tutorial – W3Schools (2016) – <http://www.w3schools.com/sql/> [Online] (25.02.2016)
- [9] TIOBE Index for May 2016 – TIOBE (2016) – [http://www.tiobe.com/tiobe\\_index](http://www.tiobe.com/tiobe_index) [Online] (14.03.2016)
- [10] Why laravel is best php framework in 2016 – Amar InfoTech (2016) – <http://www.amarinfotech.com/why-laravel-is-best-php-framework-in-2016.html> [Online] (01.03.2016)
- [11] Upgrading from Version 1.1 – Yii Framework (2014) – <http://www.yiiframework.com/doc-2.0/guide-intro-upgrade-from-v1.html> [Online] (01.03.2016)
- [12] CREATE FUNCTION – PostgreSQL Documentation (2016) – <http://www.postgresql.org/docs/9.1/static/sql-createfunction.html> (22.05.2016)
- [13] phpass – Openwall – <http://www.openwall.com/phpass/> [Online] (07.03.2016)
- [14] pg\_query\_params – The PHP Group (2016) – <http://php.net/manual/en/function.pg-query-params.php> [Online] (07.04.2016)

- [15] How to Paginate Data with PHP – Envato Tuts+ (2009) – <http://code.tutsplus.com/tutorials/how-to-paginate-data-with-php--net-2928> [Online] (01.03.2016)
- [16] Web Applications I materials – Tanel Tammet (2014) – [http://lambda.ee/wiki/V%C3%B5rgurakendusel\\_I](http://lambda.ee/wiki/V%C3%B5rgurakendusel_I) (20.03.2016)
- [17] Force file download with php using header() –StackOverflow (2013) – <http://stackoverflow.com/questions/8485886/force-file-download-with-php-using-header> [Online] (26.04.2016)
- [18] Usage of JavaScript libraries broken down by ranking – Q-Success (2016) – [http://w3techs.com/technologies/cross/javascript\\_library/ranking](http://w3techs.com/technologies/cross/javascript_library/ranking) (01.04.2016)
- [19] 10 Usability Heuristics for User Interface Design – Jakob Nielsen (1995) – <https://www.nngroup.com/articles/ten-usability-heuristics/> [Online] (13.03.2016)
- [20] RFC 3696 addendum – RFC Editor (2005) – [https://www.rfc-editor.org/errata\\_search.php?rfc=3696](https://www.rfc-editor.org/errata_search.php?rfc=3696) [Online] (19.03.2016)
- [21] Falsehoods Programmers Believe About Names – Kalzumeus (2010) – <https://www.kalzumeus.com/2010/06/17/falsehoods-programmers-believe-about-names/> [Online] (15.03.2016)
- [22] Eraldiseisev iseteeninduslik paroolide vahetamise veebitarkvara olemasolevate rakenduste jaoks – Eerik Mägi (2016) Bakalaureusetöö. TTÜ Informaatikainstituut.
- [23] Datepicker – jQuery UI (2016) - <https://jqueryui.com/datepicker/> [Online] (01.04.2016)
- [24] Separation of Concerns – Wikipedia (2016) – [https://en.wikipedia.org/wiki/Separation\\_of\\_concerns](https://en.wikipedia.org/wiki/Separation_of_concerns) [Online] (22.05.2016)

## Appendix 1 – Use Case Diagrams

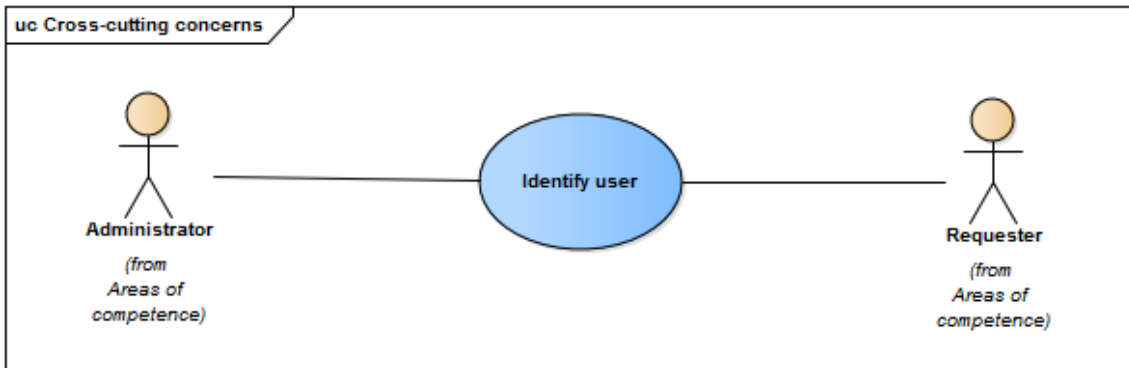


Figure 24: Cross-cutting concerns.

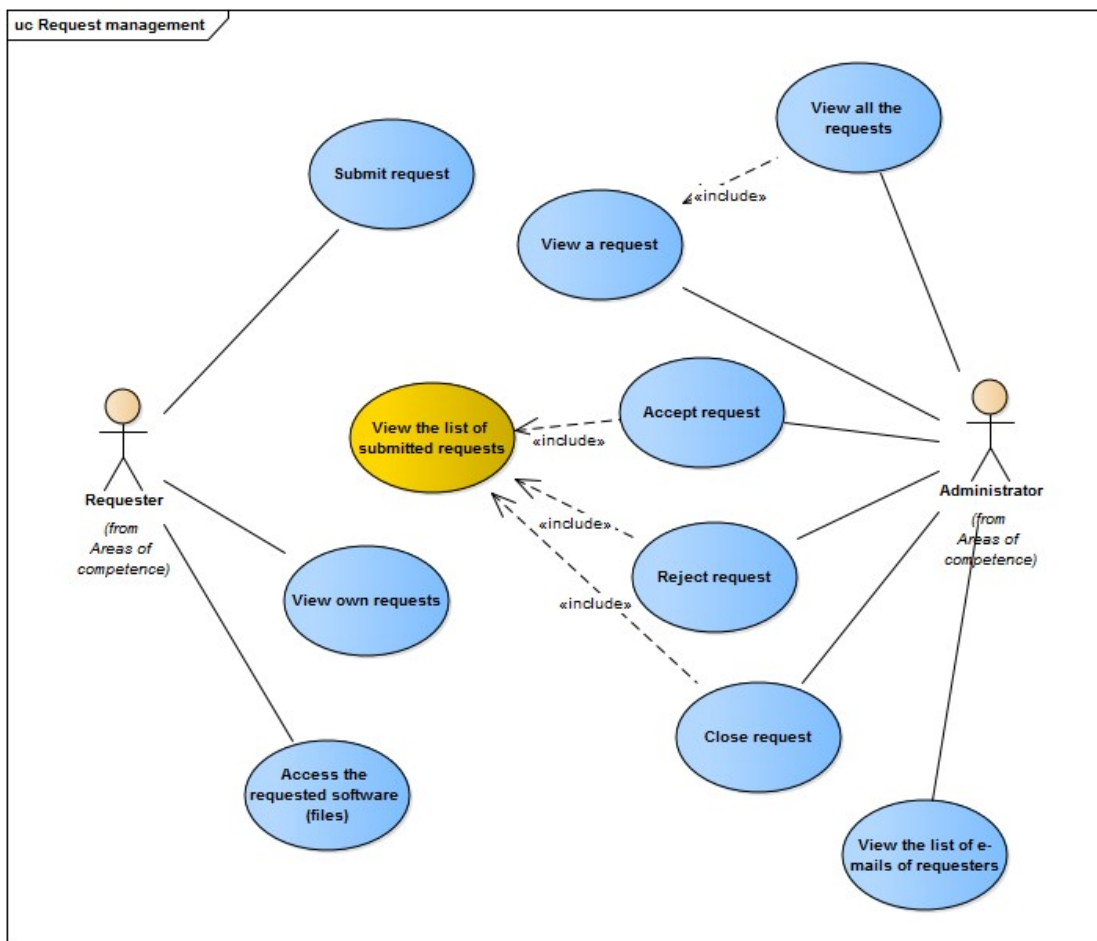


Figure 25: Request management.



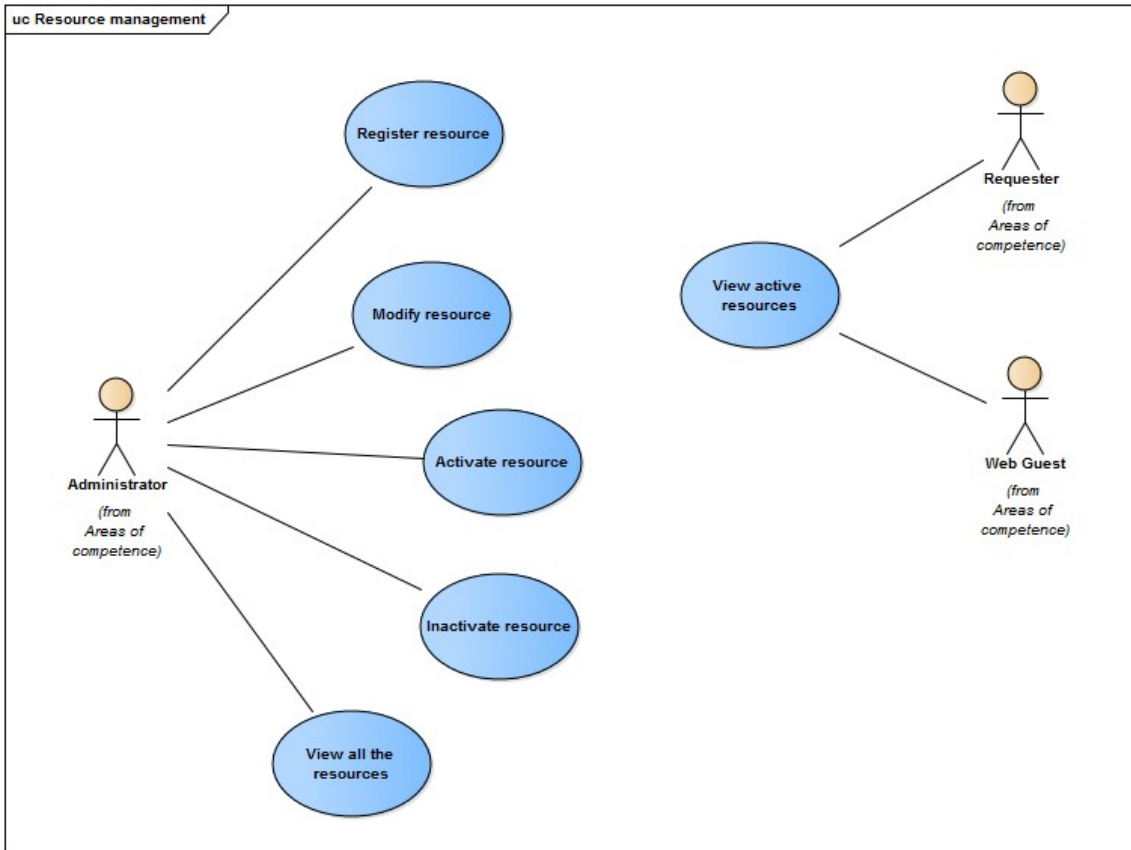


Figure 26: Resource management.

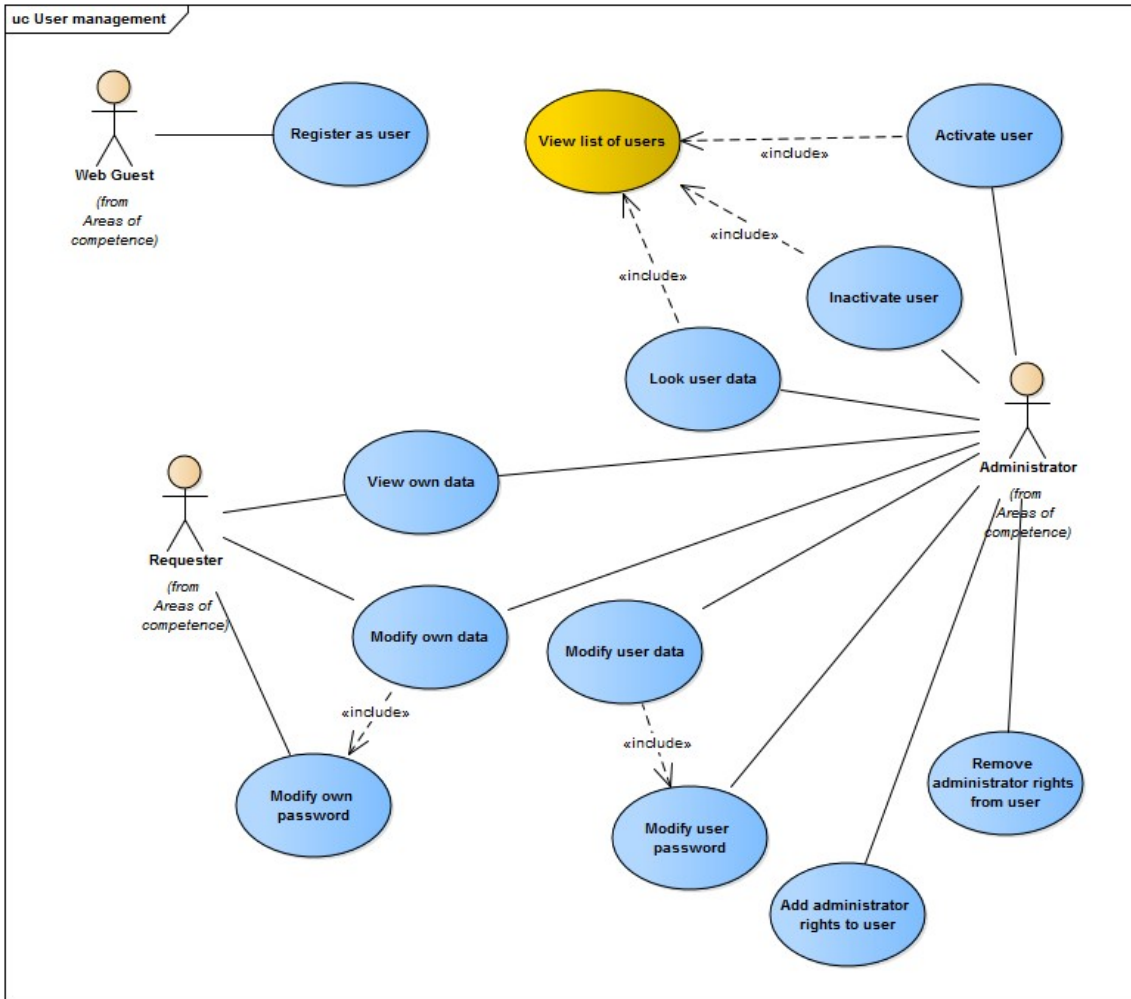


Figure 27: User management.

## Appendix 2 – Entity-relationship Diagram

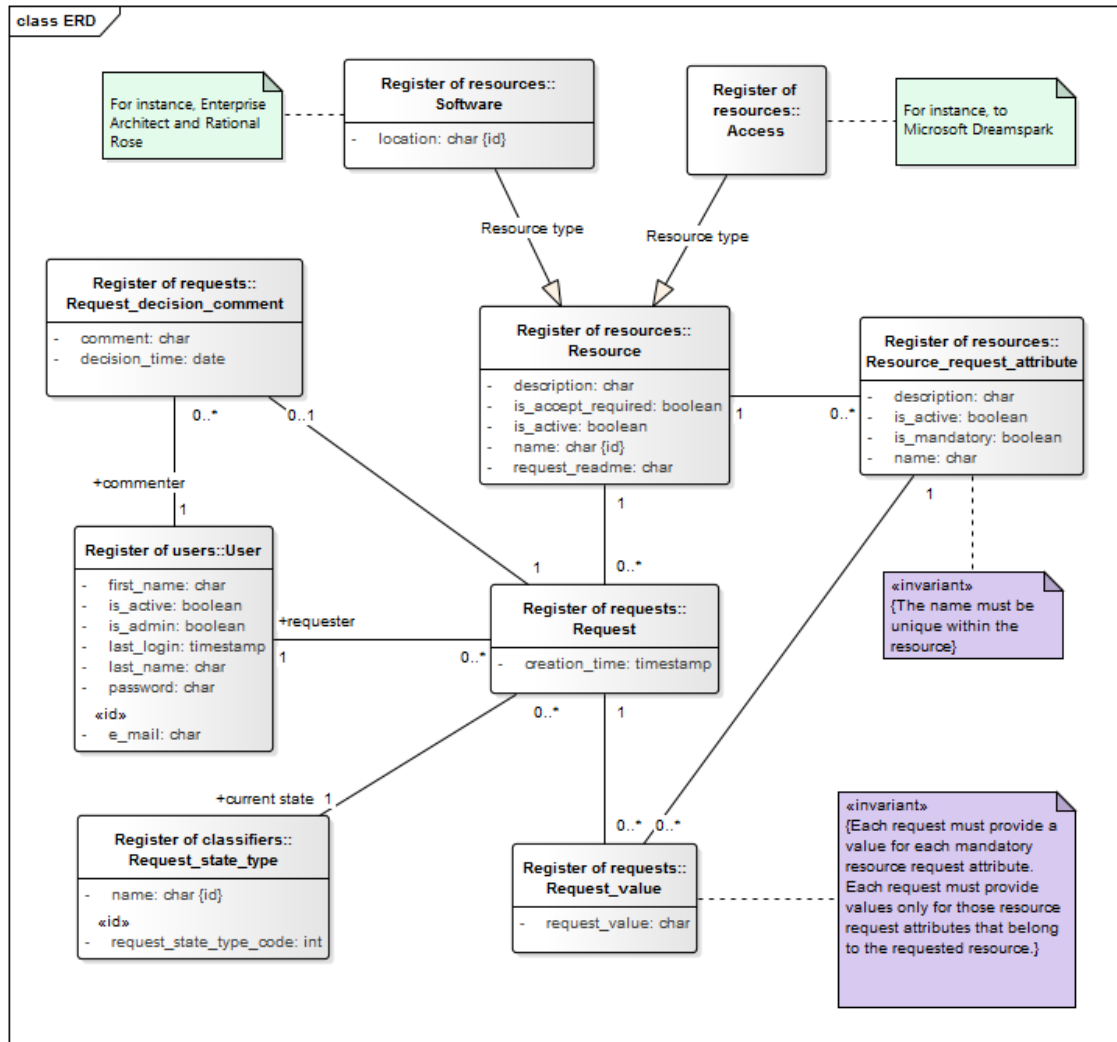


Figure 28: Entity-relationship diagram.

## Appendix 3 – Request State Diagram

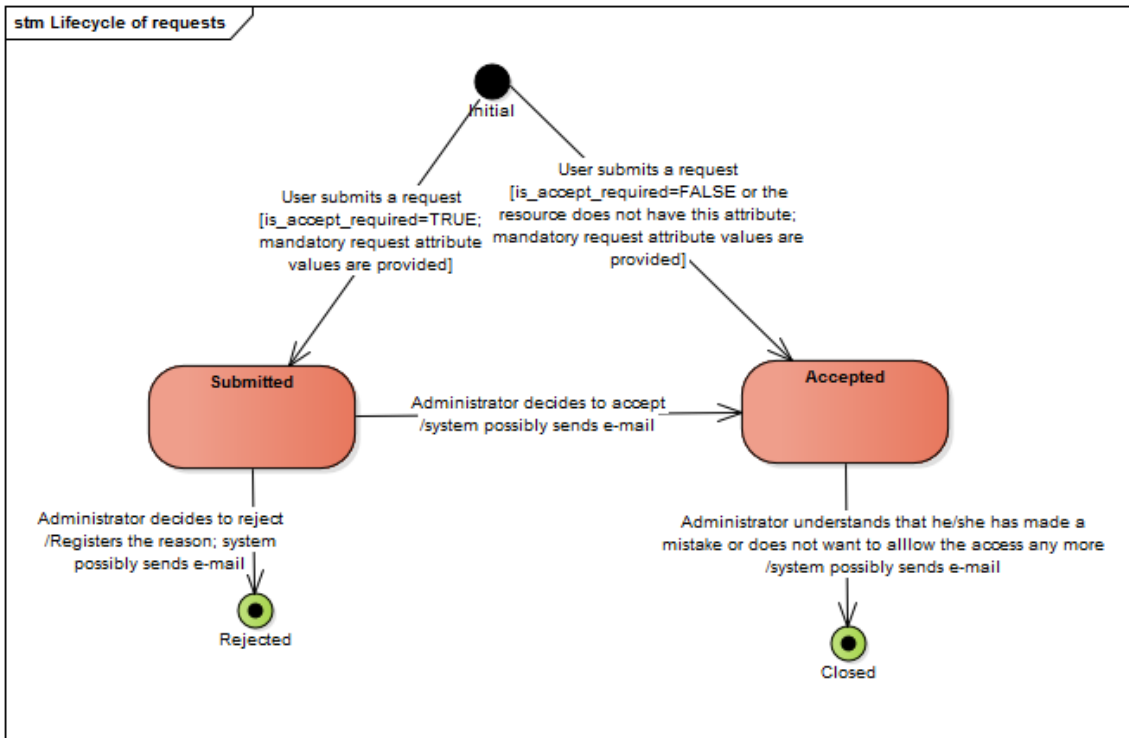


Figure 29: Request state diagram.

## Appendix 4 – Physical Database Design Diagrams

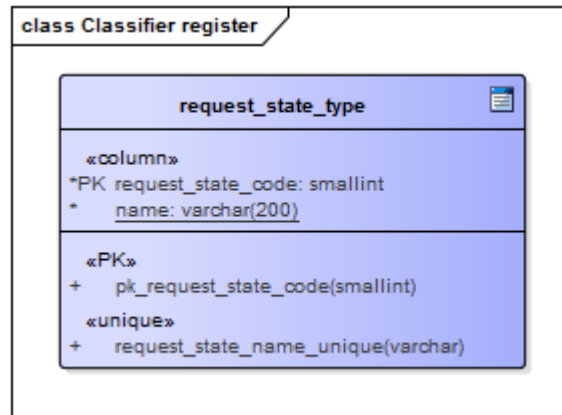


Figure 30: Classifier register.



Figure 31: User register.

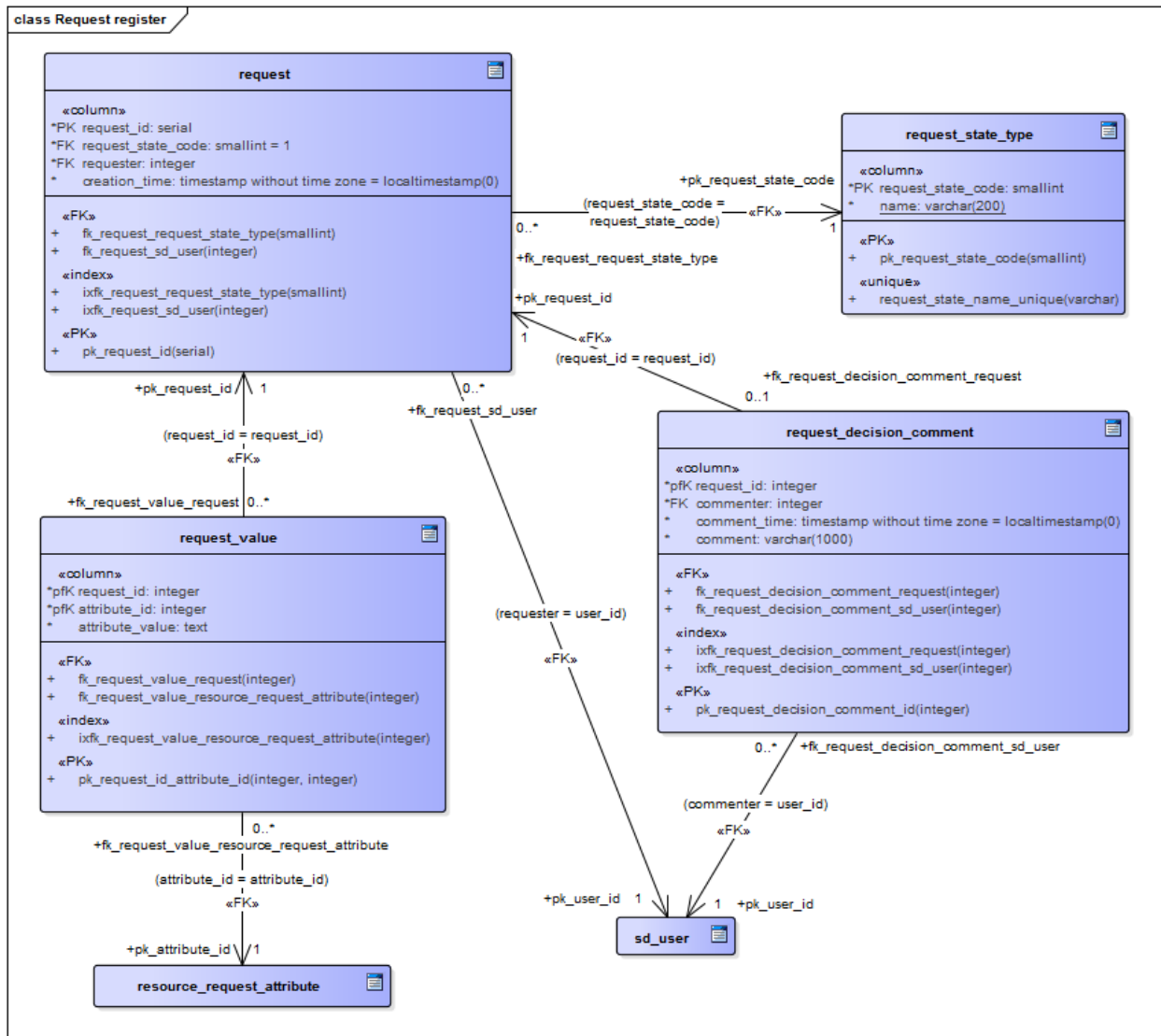


Figure 32: Request register.

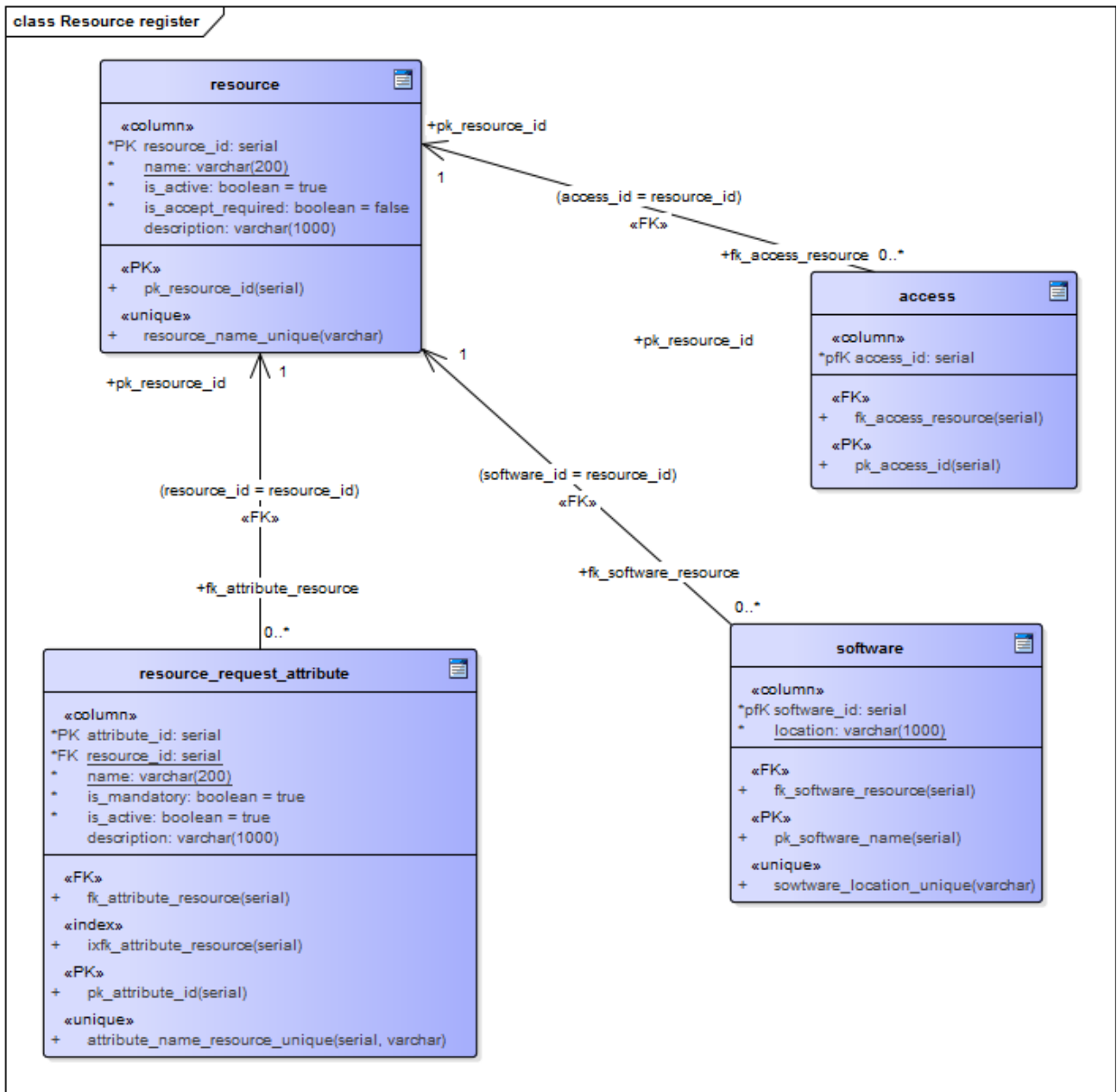


Figure 33: Resource register.

## Appendix 5 – Database SQL Statement Examples

```
CREATE OR REPLACE VIEW user_list AS
SELECT user_id,
email,
first_name,
last_name,
last_login,
is_active,
is_admin
FROM sd_user
ORDER BY first_name ASC;
COMMENT ON VIEW user_list
IS 'Shows a list of user IDs, emails, first and last names, last login dates
and whether or not a user is active and is an administrator.';
```

Figure 34: Example of a view used in the system.

```
CREATE OR REPLACE FUNCTION update_user_last_login(sd_user.user_id%TYPE)
RETURNS VOID
AS $$
    UPDATE sd_user SET last_login=DEFAULT WHERE user_id=$1;
$$ LANGUAGE SQL
RETURNS NULL ON NULL INPUT
SECURITY DEFINER
SET search_path = public, pg_temp;
COMMENT ON FUNCTION update_user_last_login(sd_user.user_id%TYPE)
IS 'Updates user last login date. Fails if no user ID is specified.';
```

Figure 35: Example of a function used in the database.



## Appendix 5 – Initial Interface Prototype

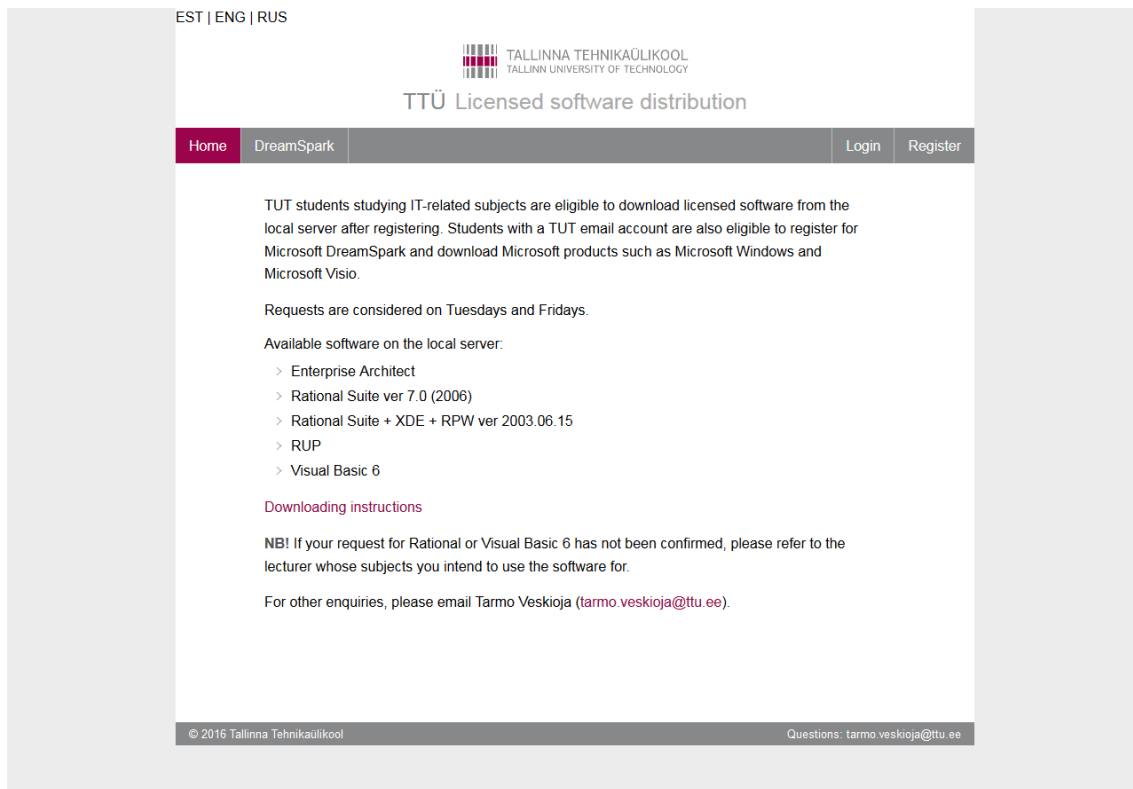


Figure 36: Home page prototype.

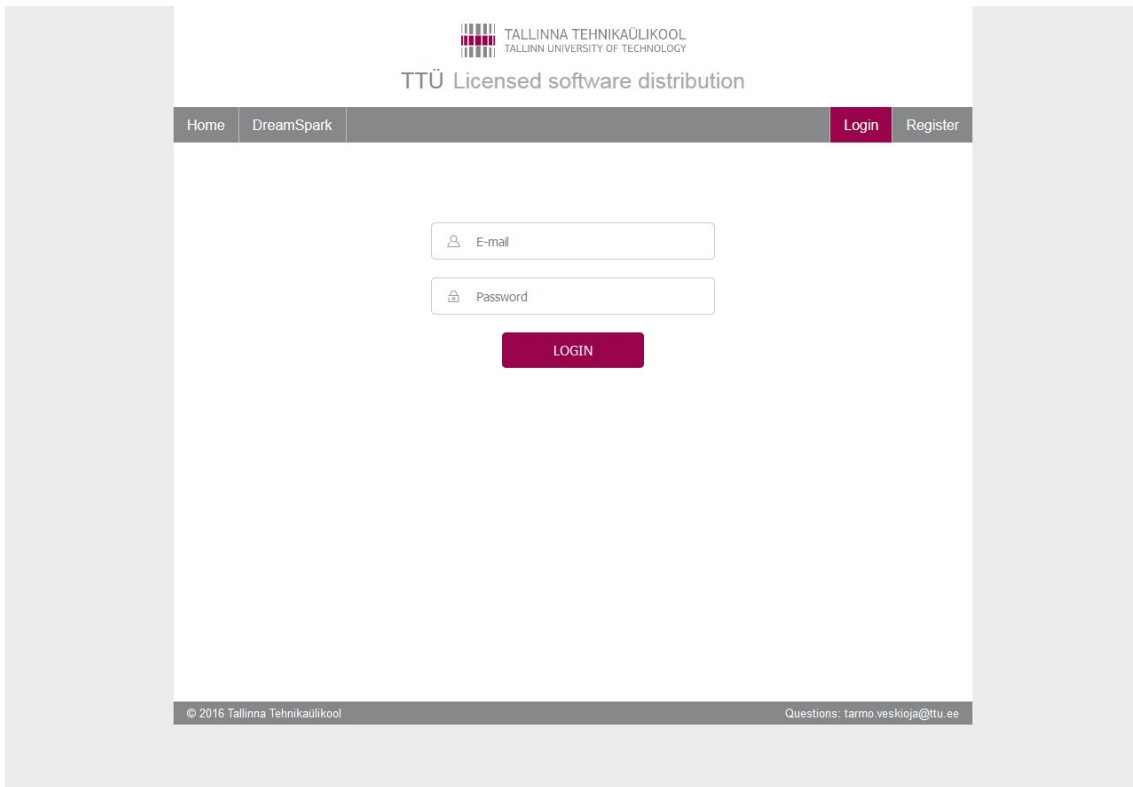


Figure 37: Login page prototype.

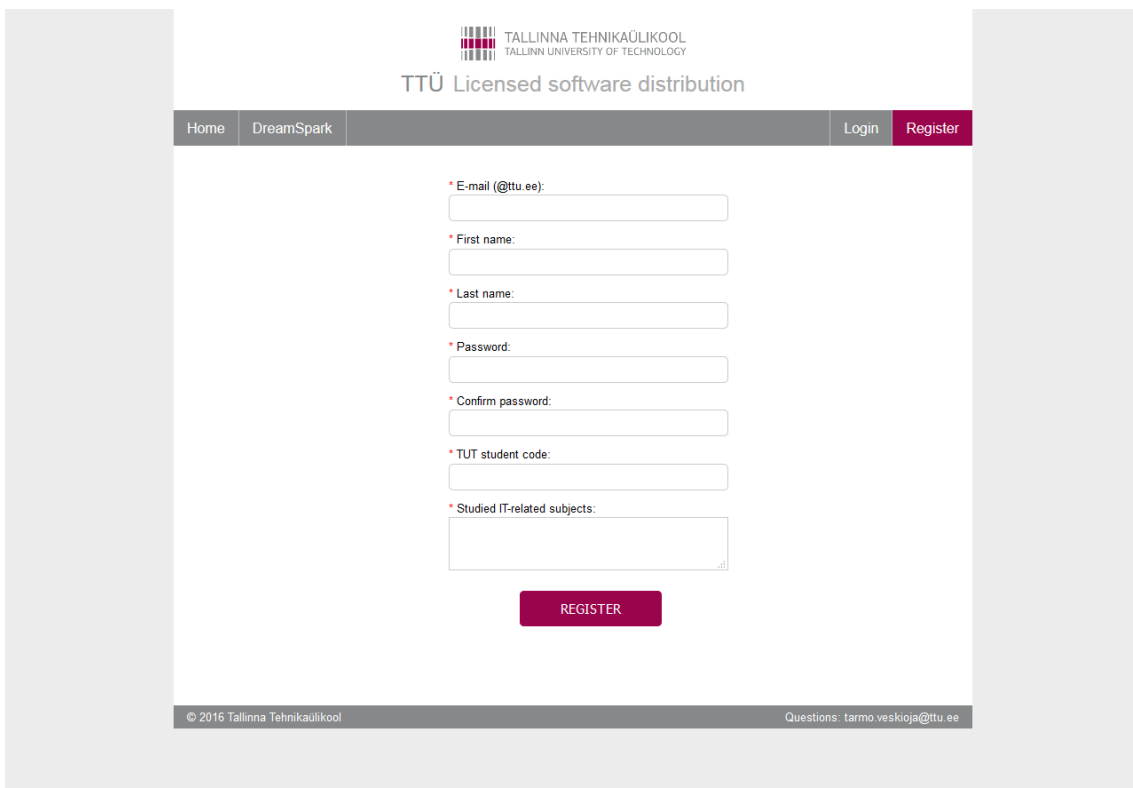


Figure 38: Registration page prototype.

## Appendix 6 – New User Interface Screenshots



Figure 39: Non-logged in user header.

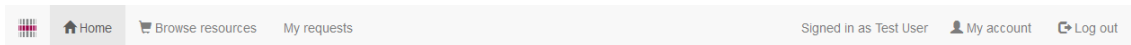


Figure 40: Requester header.

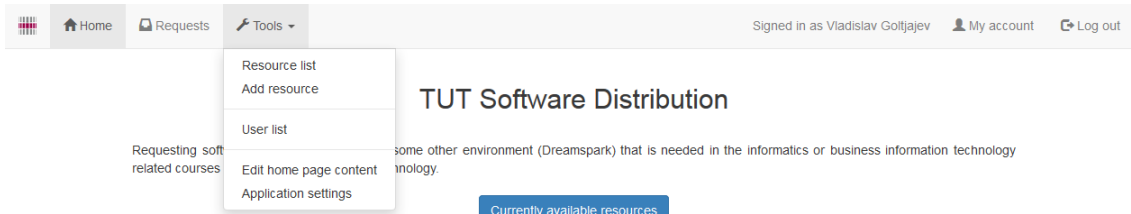


Figure 41: Administrator header.

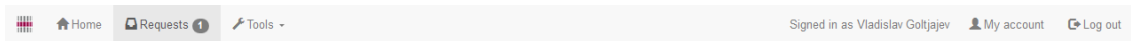


Figure 42: Header with one new request.

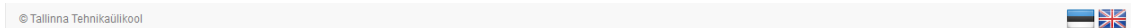


Figure 43: Footer.

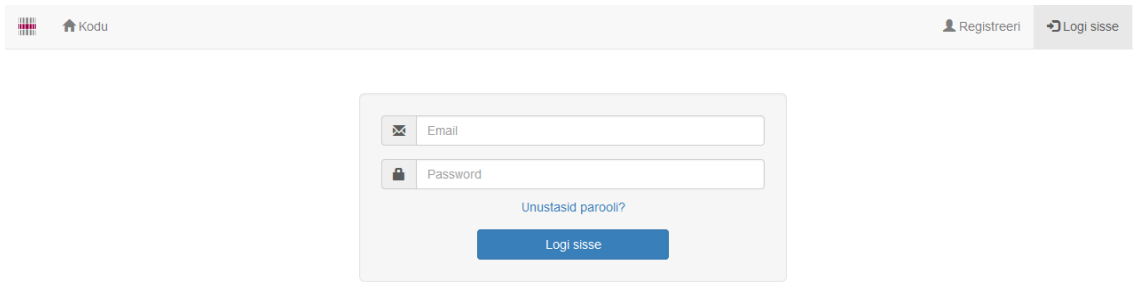


Figure 44: Login page.

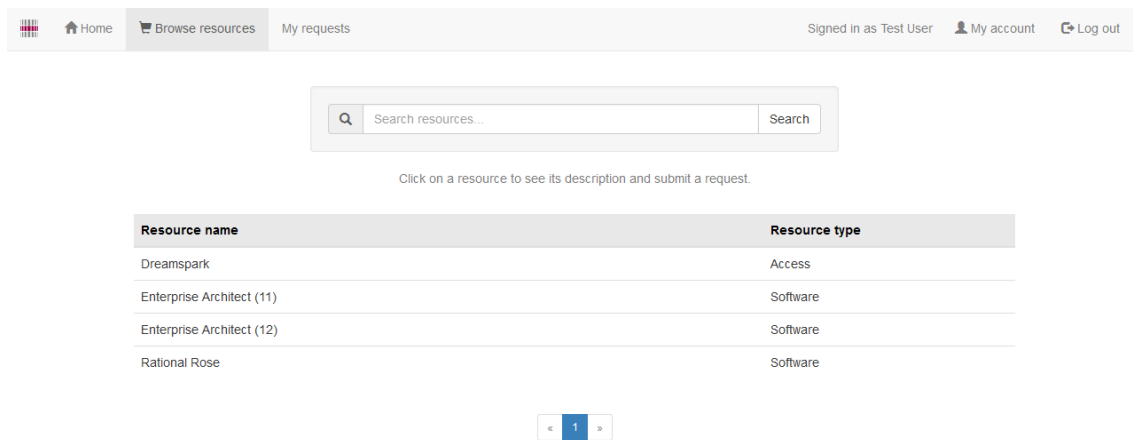


Figure 45: Resource list page for requesters.

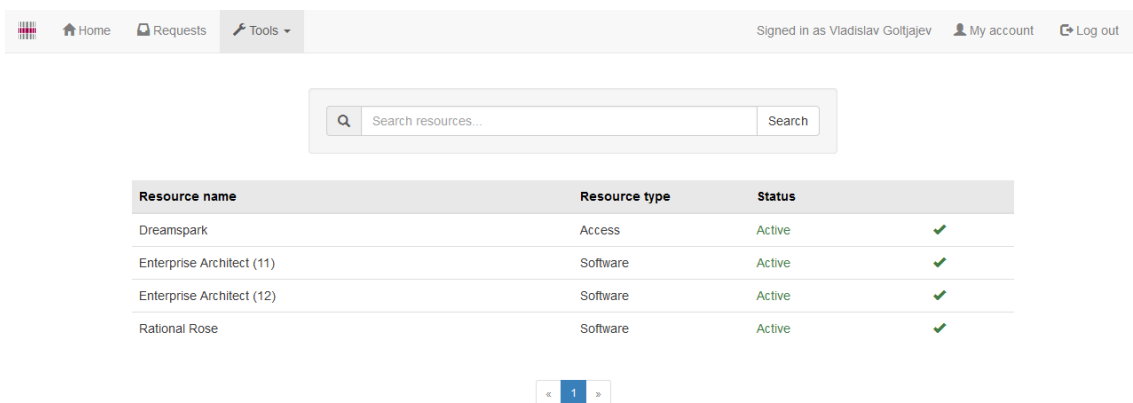


Figure 46: Resource list page for administrators.

Home Browse resources My requests Signed in as Test User My account Log out

Resource name	Submitted on	Status
Dreamspark	02.05.2016 16:14	Submitted

< 1 >

Figure 47: Request list for requesters.

Home Requests Tools Signed in as Vladislav Gottjajev My account Log out

Search requesters and resources... Search

Show requests not older than...

Submitted  Accepted  Rejected  Closed CSV

Resource name	Submitted by	Submitted on	Status
Enterprise Architect (11)	test.user@ttu.ee	22.05.2016 00:42	Accepted
Enterprise Architect (12)	test.user@ttu.ee	21.05.2016 23:28	Closed
Dreamspark	test.user@ttu.ee	06.05.2016 22:13	Accepted
Dreamspark	test.user@ttu.ee	06.05.2016 22:09	Closed
Enterprise Architect (11)	test.user@ttu.ee	06.05.2016 21:36	Closed
Dreamspark	test.user@ttu.ee	02.05.2016 17:43	Rejected
Dreamspark	test.user@ttu.ee	02.05.2016 17:22	Closed
Dreamspark	test.user@ttu.ee	02.05.2016 17:14	Closed
Rational Rose	test.user@ttu.ee	02.05.2016 17:04	Closed
Dreamspark	test.user@ttu.ee	02.05.2016 16:14	Rejected

< 1 >

Figure 48: Request list for administrators.

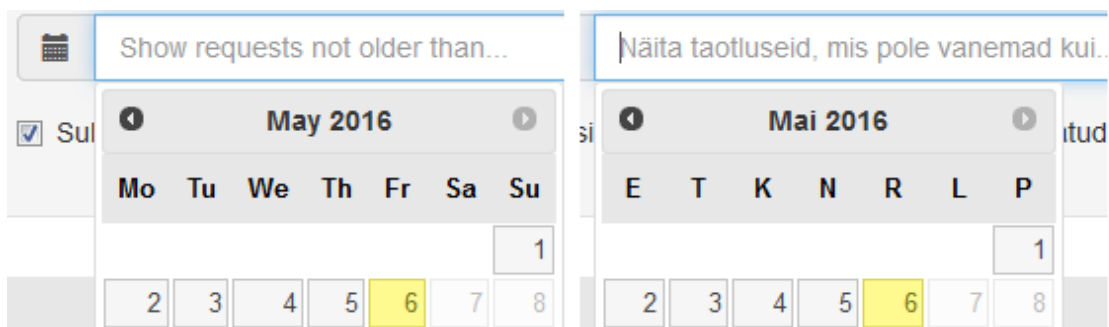


Figure 49: Date picker with different localizations.

## Appendix 7 – Current zaurus.ttu.ee design

DreamSpark programmi kasutajatale / for DreamSpark users - go to <http://zaurus.ttu.ee/dreamspark/>

Siin serveris on võimalik registreerida ennast KOHALIKUS SERVERIS oleva tarkvara allatömbajaks (Rational , Visual Basic 6)

---

Rationali allalaadimise kinnitusi ja litsentside jagamisi tehakse reeglina 2 korda nädalas: teisipäeval ja reedel.

[Tarkvara allalaadimine TTÜ IT tudengitele - alla-laadimine kohalikust serverist \(Rational , Visual Basic 6\): KOHALIKU SERVERI KASUTAJA](#)

[Allalaadimise juhend](#)

Peale konto registreerimise aktsepteerimist näidatakse teile linki, sellelt lingilt saate peale autentimist tarkvara alla laadida. Veebserver küsib teilt allalaadimislingile vajutades kasutajanime ja parooli. Sisestage sama kasutajanimi ja parooli mida kasutate veebisüsteemi sisse logimiseks.

Allalaadimis ja registreerimisprobleemide probleemide korral pöörduge aadressil [tarmo.veskioja@ttu.ee](mailto:tarmo.veskioja@ttu.ee)

**NB! NB! RATIONALI JA VISUAL BASIC6 TARKVARA JAGAMISE SISULISTE KÜSIMUSTEGA (näiteks miks on alla-laadimis soov kinnitamata) PÕÕRDUGE SELLE ÕPPEJÕU POOLE KELLE AINE RAAMES TE TARKVARA KASUTADA SOOVITE !**

Tarkvara võivad kasutada ainult TTÜ IT erialasid õppivad tudengid.

Figure 50: zaurus.ttu.ee home page.

[\[registreeri kohaliku serveri kasutajaks\]](#)

kasutajanimi:

parool:

**Tarkvara mis on saadaval sellest serverist**

Enterprise Architect

Rational Suite ver 7.0 (2006)

Rational Suite + XDE + RPW ver 2003.06.15

RUP

Visual Basic 6

Figure 51: zaurus.ttu.ee login page.

Registreerimine kohaliku serveri tarkvara kasutajaks

**kasutajanimi:**

**parool:**

**eesnimi:**

**perekonnanimi:**

**e\_mail:**

**matrikli number:**

**õpitavad IT ained:**

Figure 52: zaurus.ttu.ee registration

[\[algusse\]](#) [\[registreeritud kasutamised\]](#) [\[konto andmed\]](#) [\[logi välja\]](#)

sisse logitud: 2016-5-2 19:23:16

kasutaja: testuser007

kasutaja tüüp: kohalik server

kasutajale testuser007 registreeritud tarkvara seisund: allalaadimine [kinni](#)

nimetus:	kirjeldus:	<input type="button" value="Filtreeri"/>	<input type="button" value="Näita kõik"/>
41	<a href="#">kinni</a>		
nimi	Enterprise Architect		
kirjeldus			
asukoht	ligipääs aktsepteerimata		
installeerimisvõti	ligipääs aktsepteerimata		
	<input type="button" value="Registreeri"/>		

nimi	kirjeldus	
Enterprise Architect		<a href="#">vaata andmeid ja registreeri</a>
Rational Suite ver 7.0 (2006)		<a href="#">vaata andmeid ja registreeri</a>
Rational Suite + XDE + RPW ver 2003.06.15		<a href="#">vaata andmeid ja registreeri</a>
RUP	Rational Unified Process	<a href="#">vaata andmeid ja registreeri</a>
Visual Basic 6		<a href="#">vaata andmeid ja registreeri</a>

Figure 53: zaurus.ttu.ee request submission.

[\[algusse\]](#) [\[objektid\]](#) [\[MS kasutajad\]](#) [\[MS-i maili-fail\]](#) [\[tyhjenda ms emaili-fail\]](#) [\[uus objekt\]](#) [\[logi välja\]](#)

sisse logitud:2016-4-29 10:51:25

kasutaja: **xyri**

Aktsepteeri kõik

	<a href="#">kinni</a>
nimi	x1
failikataloog	/usr/local/apache2/htdocs/soft/isiks/
kirjeldus	<input type="text"/>
märkus	<input type="text"/>
vabalt allalaetav?	<input type="checkbox"/>
individuaalsed võtmed	<input type="checkbox"/>
avalik?	<input type="checkbox"/>
võti	<input type="text"/>
objekti tüüp	allalaetav iso ▾
sisu tüüp	raamat ▾
kasutatavus	vähemkasutatav ▾
	Salvesta

[\[lisa võti\]](#)

kasutaja tarkvara seisund |||

nimetus	
ee	<a href="#">vaata/muuda</a> genereeri paroolifailid
Enterprise Architect	<a href="#">vaata/muuda</a> genereeri paroolifailid
Rational Suite ver 7.0 (2006)	<a href="#">vaata/muuda</a> genereeri paroolifailid
Rational Suite + XDE + RPW ver 2003.06.15	<a href="#">vaata/muuda</a> genereeri paroolifailid
RUP	<a href="#">vaata/muuda</a> genereeri paroolifailid
Visual Basic 6	<a href="#">vaata/muuda</a> genereeri paroolifailid
x1	<a href="#">vaata/muuda</a> genereeri paroolifailid

Figure 54: zaurus.ttu.ee resource management.