



POLSKO-JAPOŃSKA WYŻSZA SZKOŁA
TECHNIK KOMPUTEROWYCH

Wydział informatyki

Katedra Baz Danych

Specjalizacja Bazy danych

Maciej Szczęsny

Nr albumu 5014

**Culinary web application
with elements of social portal**

**Serwis kulinarny
z elementami portalu społecznościowego**

Praca inżynierska

Promotor

Dr inż. Paweł Lenkiewicz

Warszawa, lipiec 2012

STRESZCZENIE W JĘZYKU POLSKIM

Celem niniejszej pracy inżynierskiej jest stworzenie serwisu internetowego, który pozwoli odwiedzającym na przeglądanie przepisów kulinarnych. Portal ma ponadto implementować funkcje społecznościowe gromadząc internautów - miłośników jedzenia - wokół wspólnego zainteresowania poprzez zakładanie profili, dodawanie potraw (i ich zdjęć), komentowanie przepisów innych użytkowników, otrzymywanie punktowej oceny swoich potraw, zdobywanie punktów doświadczenia, które to punkty będą miały odzwierciedlenie w hierarchii użytkowników.

W kolejnych rozdziałach przedstawiono kontekst problemu oraz szczegółowe wymagania funkcjonalne i нефункционалне (rozdział drugi), jak również opisano zastosowane technologie, architekturę systemu (ze szczególnym uwzględnieniem konstrukcji bazy danych) i szczegóły implementacyjne (rozdział trzeci). Rozdział czwarty opisuje przeprowadzone testy aplikacji. Instrukcję instalacji oraz opis obsługi aplikacji przedstawiono w rozdziale piątym.

Table of contents:

STRESZCZENIE W JĘZYKU POLSKIM	2
1 INTRODUCTION.....	5
2 ANALYSIS	5
2.1 GOAL	5
2.2 PROBLEM DOMAIN	5
2.3 SCOPE.....	6
2.4 SYSTEM USERS	6
2.5 FUNCTIONAL REQUIREMENTS	7
2.6 NON-FUNCTIONAL REQUIREMENTS.....	10
2.7 USE CASE DIAGRAM	12
2.8 ENTITY RELATIONSHIP DIAGRAM (ANALYSIS)	13
3 DESIGN.....	14
3.1 TECHNOLOGIES CHOSEN	14
3.2 TOOLS USED	18
3.3 SYSTEM ARCHITECTURE	20
3.4 SYSTEM MODULES	29
3.5 DATA VALIDATION	31
3.6 INTERFACE DESIGN	34
3.7 ENTITY RELATIONSHIP DIAGRAM (DESIGN).....	36
3.8 DESCRIPTION OF ENTITIES AND ATTRIBUTES	37
3.9 DATABASE TRIGGERS	39
3.10 SOFTWARE/HARDWARE REQUIREMENTS.....	40
3.11 SCREEN TRANSITION DIAGRAM	41
4 TESTING	42
4.1 WHITE BOX TESTING	42
4.2 BLACK BOX TESTING.....	42
5 MANUALS	48
5.1 INSTALLATION MANUAL	48
5.2 INSTRUCTION MANUAL	51

6	REFERENCES	71
6.1	WEB RESOURCES	71
6.2	LITERATURE.....	72

1 INTRODUCTION

Culinary web application with elements of social portal is a project that has been developed and implemented for the purposes of Bachelor thesis. The document describes in details various aspects of the project and its lifecycle.

2 ANALYSIS

2.1 GOAL

The primary aim of the project is to create a website that will allow visitors to view cooking recipes and search among them using the “let’s make something from the things I can find in my fridge” method.

The secondary aim of the project is to gather food-lovers around common idea and share cooking ideas among them via creating users’ profiles, adding recipes (along with the photos of the dishes), commenting on other users’ dishes, obtaining positive (or negative) feedback from other users on your own dishes, gaining experience points that will establish the hierarchy of users.

2.2 PROBLEM DOMAIN

Internet is a place where one can find literally everything, starting from various types of news, facts, gossips, going through e-banking, e-auctions, e-shopping, reaching internet television, radio and online gaming.

Among numerous things one can find in the Internet there are recipes, cooking advices and manuals. For sure everybody at least once in his/her lifetime had a chance to visit a page specialized in this topic, intentionally or by accident. Actually, it is rather hard to avoid such websites, as there are a lot of them: standalone (allrecipes.com, przepisy.net) or connected to some other well-known portals (shine.yahoo.com/channel/food, gotowanie.onet.pl). Some of them are driven by a community of users, such as: www.grouprecipes.com or www.epicurious.com. We do not need to highlight how successful these are, having in mind the “madness” caused by Facebook.

The website that will emerge as a result of this project is planned to be a cooking portal filled in with recipes by logged-in users. It will be using innovative method to find a desired recipe by inputting ingredients already owned by the user. Users will be encouraged

to share their cooking ideas by obtaining experience points. Experience points will be exchanged later to rewards.

2.3 SCOPE

The website will be available for everybody. Non-logged users (i.e. guests) will be allowed to search for the cooking recipes, either by inputting already owned ingredients or simply by going through the catalogue of dishes or searching for key word in names of the dishes.

Logged-in users will gain more privileges. Not only will they be able comment and rate other users' dishes, but also they will be encouraged to add their own dishes by obtaining experience points. Experience points will be given for certain achievements (e.g. adding the dish) that will be in correspondence with activity of the given user.

The site will be controlled by super-user(s) that will approve "the content" created by regular users, edit other user's profile or ban them for improper behavior on the webpage.

2.4 SYSTEM USERS

2.4.1 GUEST

Guest is a regular visitor of the page that can only view food recipes.

2.4.2 USER

User is a guest that has created a profile in the system and has gained certain privileges, mainly possibility to lead active life in the portal by adding his/her own dishes, viewing other users' dishes and giving positive or negative feedback on them.

2.4.3 ADMIN

This is a super-user that has the power to edit other users' profile, block dishes and comments, approve photos.

2.5 FUNCTIONAL REQUIREMENTS

2.5.1 STRUCTURES DESCRIPTION

2.5.1.1 USERS

System should store information about the users. Mandatory user's data consists of: email and password (these are used for logging in), nick, date of birth (these four filled in by the user when creating his/her profile), privilege level, registration date and time, amount of experience points that are translated to rank level, amount of experience points used for rewards redemption. Other data is: avatar, country, www, sex. After profile creation user status is "inactive" – email is sent with activation key. Only after clicking the link and confirming the user status will be set to "active".

2.5.1.2 RANKS

Users will be given certain ranks that will be dependent on total amount of points gained by the user. Rank data should consist of: name, description, avatar, amount of points required to obtain given rank. One will also know when the rank was assigned to the user.

2.5.1.3 DISHES

System should store information about the dishes. Dish's data consists of: name, description (preparation plan), amount of calories, preparation time, difficulty level, size (for how many people), average dish rate. List of ingredients should be available for every dish. Category (e.g. "soup", "dessert") should be assigned to every dish. It should be possible to add photo of the dish (addition time, user, status – approved or not by admin – should be known). One should also know which user has added given dish and when. Dishes can be blocked by admin.

2.5.1.4 INGREDIENTS

System should store information about ingredients. Ingredient's data consists of: name and unit.

2.5.1.5 COMMENTS

Comments can be added to dishes. One should know the user that added the comment and the date of addition. Comments can be blocked by admin.

2.5.1.6 RATES

Rates (1 to 5) can be given to dishes. One should know the date of rating and the user that rated the dish (given dish can be rated by the given user only once).

2.5.1.7 ACHIEVEMENTS

Achievements are for example “adding dish”, “adding photo of the dish”, “adding profile picture”. Achievement’s data consists of: name, description, amount of experience points. The date of assignment should be known. Multiple achievements can be assigned to one user.

2.5.1.8 REWARDS

Rewards can be redeemed by the users with experience points. Name, description and the cost (in experience points) will be stored in the system. The date of redemption, as well as address (given by the user) where the reward is to be sent, will also be known.

2.5.1.9 IMAGES

Images are photos of the dish or the avatars of users or ranks. System should store the status of the image (approved or not by admin – always approved for rank avatar) and path, as well as addition time.

2.5.2 SYSTEM FUNCTIONALITY

2.5.2.1 GUEST

- Every guest is able to search the dishes by inputting the ingredients he already owns.
- Every guest is able to search the dishes by using the search engine (search performed in names of the dishes).
- Every guest is able to view the dishes by browsing the catalogue of dishes using categories.
- Every guest is able to view dish details, as well as comments, photos and average rating, but only the nick of the user that added the dish.

2.5.2.2 USER

- Every user is able to perform actions of the guest.
- Every user is able to add his own dishes, photos of his own dishes.
- Every user is able to rate other users' dishes.
- Every user is able to comment on other users' dishes.
- Every user is able to add given dish to "to be prepared by me" list, preparing the dish is recorded in the system when the user uploads the photo of prepared dish.
- Every user is able to requests for new ingredient (notification with requested ingredient will be sent to admin's e-mail, admin will then decide whether to add the ingredient manually or not)
- Every user is able to edit his own profile.
- Every user is able to view other user' profile.
- Every user is able to reset forgotten password.

2.5.2.3 ADMIN

- Every admin is able to perform actions of the user.
- Every admin is able to edit/ban users' profiles.
- Every admin is able to add new/edit ingredients.
- Every admin is able to add new/edit categories.
- Every admin is able to approve uploaded images of dishes and users' avatars.
- Every admin is able to block comments.
- Every admin is able to block dishes.

2.5.2.4 APPROVAL RULES

- Addition of the profile is approved by the user by clicking the link (with activation code) sent to his email.
- Addition of the dish does not require admin's approval, but the dish can be blocked by the admin.
- Addition of the photo of the dish is to be approved by admin.
- Addition of the photo of user avatar is to be approved by admin.
- Addition of the comment does not require admin's approval, but the comment can be blocked by the admin.

2.5.2.5 ACHIEVEMENTS, RANKS AND REWARDS RULES

- Achievements will be assigned to users for certain activities (e.g. adding dish). Amount of experience points assigned to given achievement will be added to total amount of experience points of the user.
- Gaining certain amount of experience points will result in assigning given rank to the user, what will put him higher in the hierarchy of users.
- Experience points gather by the user can be exchanged to rewards.

2.6 NON-FUNCTIONAL REQUIREMENTS

2.6.1 PERFORMANCE REQUIREMENTS

- Website should load in not more than 10 seconds
- System should handle up to 50 concurrent users

2.6.2 TESTING REQUIREMENTS

- Testing should cover all functions of the system
- User acceptance testing for FF(Linux) and IE(Windows), sanity testing for Chrome and Opera

2.6.3 DOCUMENTATION REQUIREMENTS

- Every phase of the project should be documented in details
- User manuals should be created
- System installation manual should be created

2.6.4 SECURITY REQUIREMENTS

- Profiles and passwords for the users of the system
- Passwords stored in database in encrypted form
- HTTPS connection applied for logged in users

2.6.5 QUALITY REQUIREMENTS

- Clean, intuitive interface
- Website should be displayed correctly in modern browsers

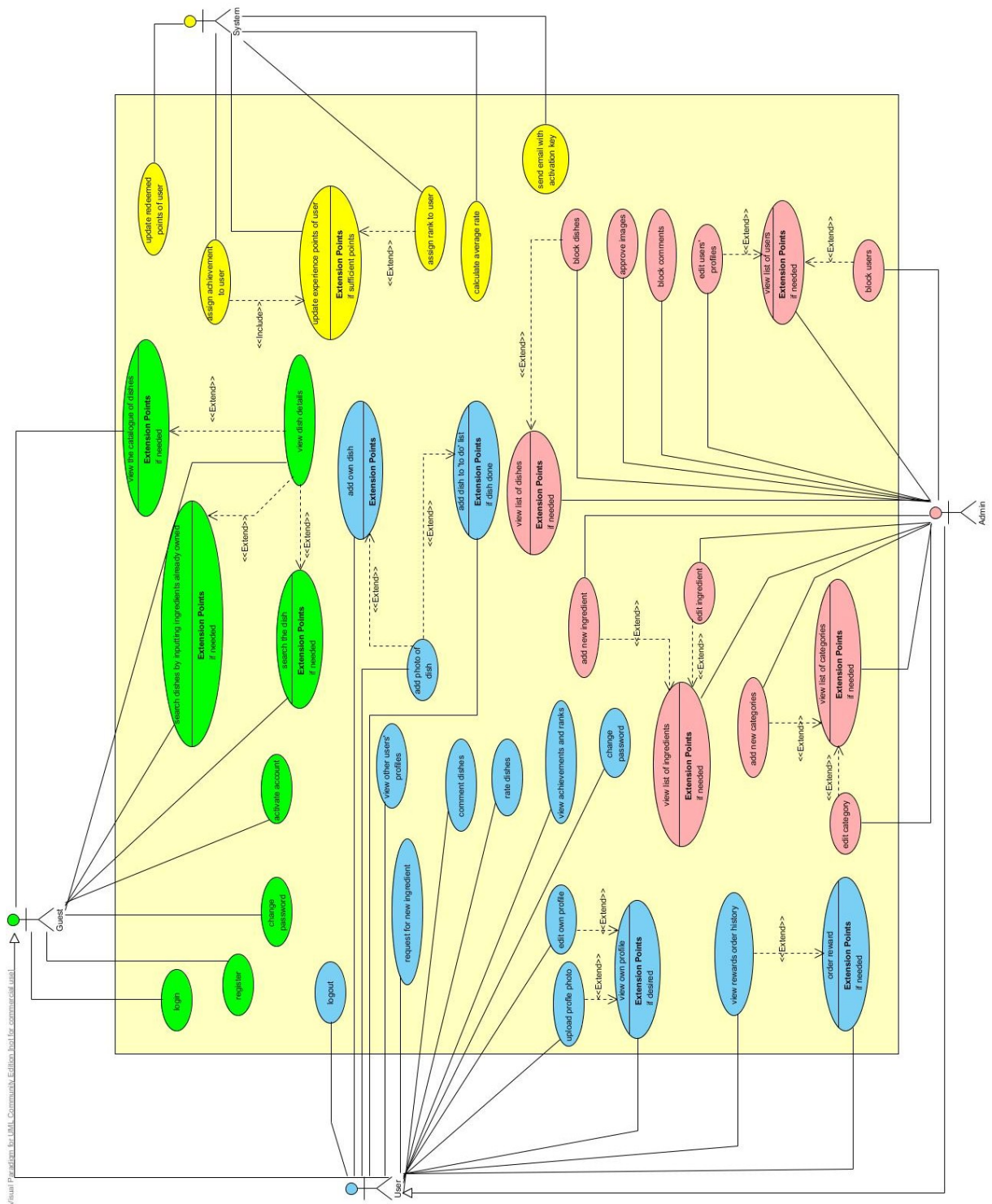
2.6.6 RELIABILITY REQUIREMENTS

- System should be available 24/7
- Maximal period of inaccessibility of the system should not be longer than 8h since raising the issue

2.6.7 MAINTENANCE REQUIREMENTS

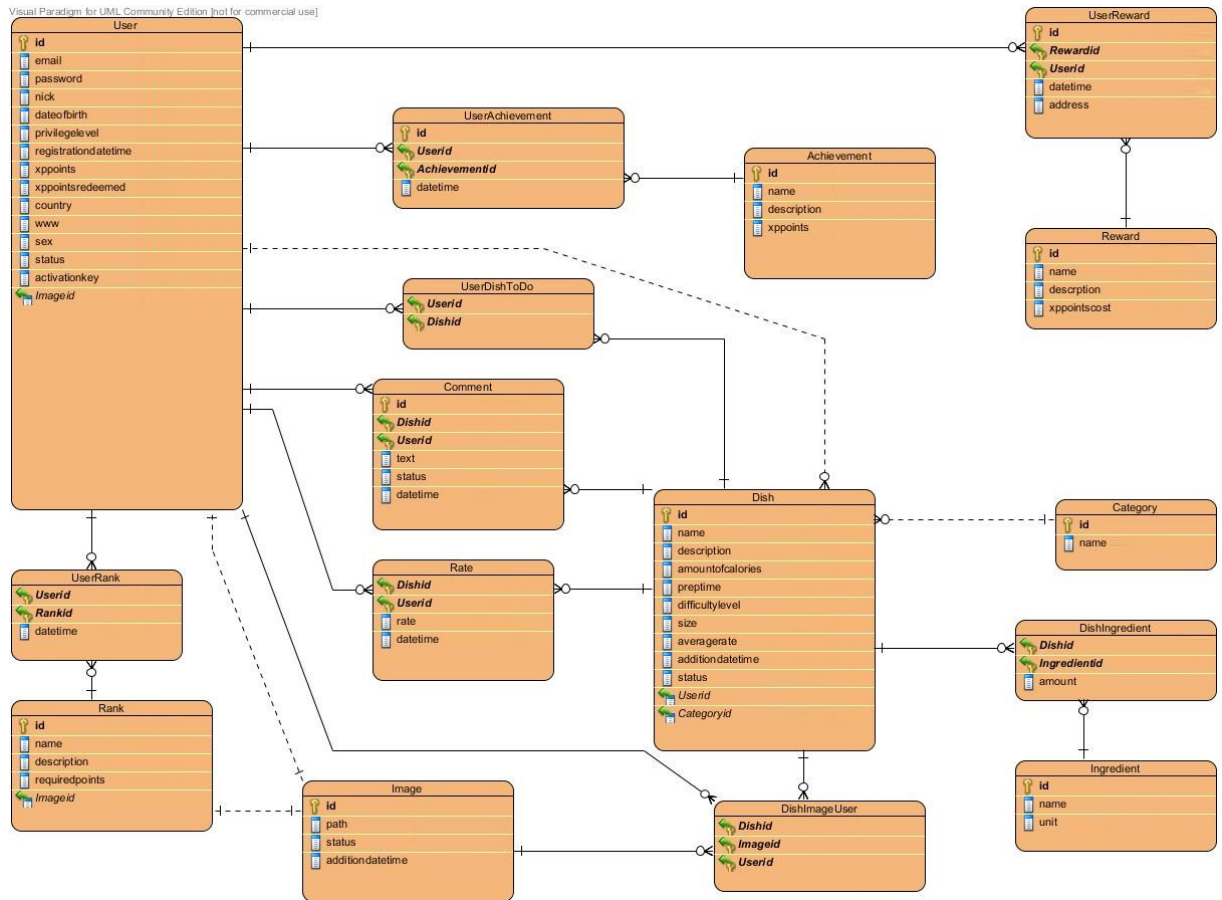
- Database backups should be performed on daily basis

2.7 USE CASE DIAGRAM



2.8 ENTITY RELATIONSHIP DIAGRAM (ANALYSIS)

Visual Paradigm for UML Community Edition [not for commercial use]



3 DESIGN

3.1 TECHNOLOGIES CHOSEN

3.1.1 JAVA PLATFORM, ENTERPRISE EDITION

Java Platform, Enterprise Edition (Java EE) is broadly used platform for server-side programming in Java. It differs from Java Standard Edition Platform as it provides libraries for development of distributed, multi-tier Java applications that are mostly built from modular components existing on application server. At the same time it inherits all advantages of Java language itself - these are [1]:

- Simplicity – Java is easy to write, compile, debug and learn comparing to other programming languages; it uses automatic memory allocation and garbage collection.
- Object-oriented programming – Java concentrates on creating objects and making them work together, what allows for modular approach and reusable coding.
- Platform-independence – easy movement from one computer system to another, what is especially crucial for WWW software development, as the platform-independence applies to source and binary levels.
- Interpreted language – programs written in Java are compiled to byte code and requires Java interpreter to be run, but the compilation occurs only once and the byte code can be run on any platform.
- Distributed programming – possessing network capabilities integrated.
- Robusticity – checking for errors occurs early, at compiler level, in other languages similar errors would come up only at runtime.
- Security of applications – Java was born to be secured: compiler, interpreter, runtime environment and language itself designed and developed with secure approach.
- Multithreading – Java can perform multiple tasks at once within the same application.

Those features added to the fact that Java EE is widely used for enterprise applications (including commercial ones) development (thus has great worldwide support) encouraged project stakeholders to use this platform for project implementation (as a matter of fact Java SE (JDK 7) with only servlet-api.jar and jsp-api.jar included in Tomcat 7 download was used).

3.1.2 TOMCAT SERVLET CONTAINER

Apache Tomcat is a servlet container developed by Apache Software Foundation. It implements Java Servlet and JavaServer Pages specifications from Oracle Corporation and provides HTTP web server environment (written in Java itself) for Java applications.

Servlet container is responsible for running and managing servlets. It provides [15]:

- Communication support – e.g. building server sockets, listening on ports, creating streams.
- Lifecycle management – e.g. loading servlet classes, initializing servlets, invoking servlet methods, making servlet instances eligible for garbage collection.
- Multithreading support – managing of threads for multiple requests.
- Declarative security – with a container you are provided with XML deploy descriptor where you can configure security without the need to hardcode it in servlet class code.
- JSP support – translating JSP into Java code.

Obviously, first choice for the servlet container implementation was Tomcat, widely used alone or bundled with application servers (e.g. Apache Geronimo or JBoss). Other pros are: free, open-source, additionally with respect to version 7 used in the project: actual in terms of servlet API compliance, optimized, with improved memory leak prevention/detection, impressive performance and efficiency, improved security [5].

3.1.3 MYSQL DATABASE

MySQL is a cross-platform relational database management system (RDBMS), previously sponsored by Swedish company MySQL AB, now owned by Oracle Corporation. Older version of MySQL followed SQL standard only to some extent, newer versions brought implementations of triggers, perspectives, cursors, stored procedures. Up till version 5.5 default storage engine was MyISAM that did not support transactions and referential integrity constraints, but InnoDB which became default engine in 5.5 version supports those features - following ACID rules and most of SQL standards [10].

MySQL 5.5 was chosen to store data for the purposes of the project due to its popularity (used by such websites like Flickr, Nokia.com, YouTube, Wikipedia, Facebook, Twitter [10]) - thus support, easy management through GUI applications (phpMyAdmin,

MySQL Workbench), good efficiency for big read/write ratio (lot more users browse the dishes in comparison to the number of users that actually add dishes), easy access from Java application (through JDBC driver).

3.1.4 STRUTS 2 FRAMEWORK

Struts is an open-source cross-platform framework used for creating enterprise-ready Java web applications. It extends Java Servlet API in order to allow developers adopt model-view-controller (MVC) architecture more easily.

Although Struts has strong competition amongst MVC frameworks (Spring MVC, Stripes, Wicket, Play!, Tapestry) some of its features make it still a very popular choice for developing and maintaining web applications [19]:

- Java – Java is already mature language and reached the point where most of the nonbusiness-related features exist as libraries;
- Plugins – core or third-party plugins can be added as needed, rather than requiring the core framework to include everything;
- Convention over configuration – configuration eliminated wherever possible, e.g. class names can provide action mappings or returned result values can provide names of to-be-rendered JSPs, also using annotations rather than XML configuration;
- Data conversion – conversion of string-based form field values to primitive types and objects (and vice versa) handled by framework itself;
- Dependency injection – all objects that action needs to interact with to perform its logic are provided through setters;
- Testability – testing Struts2 actions and interceptors are extremely easy;
- MVC pattern – Struts2 follows well-established MVC design pattern (Chapter 3.3.1).

Due to above features Struts2 framework has been chosen to implement project's requirements.

3.1.5 FRONT-END TECHNOLOGIES

3.1.5.1 XHTML

eXtensible HyperText Markup Language (XHTML) is a XML-based extension of widely used Hypertext Markup Language (HTML) used for writing web pages. Project uses XHTML 1.0 Transitional which behaves as if it was HTML 4 – this is mainly due to the need of interpreting it well by Internet Explorer browser, as only starting from version 9 of IE it is fully supported. *Transitional* means that it includes some of the presentational elements excluded from strict version [13]. This version of XHTML is imposed by the Struts2 framework itself.

3.1.5.2 CSS

Cascading Style Sheets (CSS) is a style sheet language used to describe presentation semantics (look and feel) of documents written in (X)HTML (for web pages purposes) or other markup languages. Project is based on CSS 2.1, which is supported by all nowadays web browsers [2]. CSS3 was used to apply round corners for DIV elements (not supported for IE9).

3.1.5.3 JAVASCRIPT / JQUERY / JQUERYUI

JavaScript is prototype-based, dynamic, weakly coupled scripting language with first-class functions developed by Netscape. This scripting language is broadly used for client-side interaction in web browsers. This interactivity is obtained e.g. via form validation (before sending to server), reaction on events, on the fly building and appending DOM elements [6].

JQuery is cross-browser open source JavaScript framework designed to simplify usage of pure JavaScript . Its syntax makes it easier to select DOM elements, create animations, handle events and apply Ajax calls [7]. JQuery 1.7 was used across the application.

JQueryUI is a JavaScript library built on top of JQuery used for applying more advanced elements of user interface. It comprises of themeable widgets, interactions and effects [9]. jQuery UI 1.8.16 was used to implement calendar for date fields in the project.

3.2 TOOLS USED

3.2.1 ECLIPSE WITH TOMCAT PLUGIN

Eclipse is most commonly used (together with Netbeans) integrated development environment (IDE) for Java that can be supported by extensible plug-in system. It is mostly written in Java to program in Java, although selection of programming languages can be extended by plug-ins. Tomcat plug-in for Eclipse allows for starting/stopping/restarting of Tomcat container, redirecting Tomcat context logger to Eclipse console, exporting Tomcat project as WAR file. Eclipse 3.7.1 (Indigo) was used to develop project's application.

3.2.2 XAMPP FOR WINDOWS

XAMPP is one of the most popular, free and open-source webserver solution package. Latest version of XAMPP for Windows (1.7.7) includes: Apache 2.2.31, MySQL 5.5.16, PHP 5.3.8, phpMyAdmin 3.4.5 and Tomcat 7.0.21. XAMPP was used to provide development environment for the project due to its simplicity of setup and usage: Tomcat was used as servlet container, MySQL server as database server, phpMyAdmin as a GUI browser-based tool to manage database.

3.2.3 MYSQL QUERY BROWSER

MySQL Query Browser is a tool designed to administer MySQL server by creating, analyzing and optimizing queries in a graphical environment. It was used in Linux production environment to query and edit database data in more intuitive non-command-line manner.

3.2.4 FIREFOX WITH ADD-ONS / CHROME

Firefox and Chrome are considered to be the most developer-friendly browsers. Both these browsers were extensively used during the development phase.

Chrome has built in Developer Tools that can be used to edit DOM and CSS of the webpage on the fly, debug JavaScript with graphical debugger, analyze times of execution of JavaScript functions, as well as the traffic going in and out of the browser [4].

Firefox can be equipped with many developer add-ons. For the purposes of the project Web Developer, Firebug (both similar in functions to Chrome Developer Tools), Tamper Data (used for viewing and modifying HTTP(S) headers and post parameters, as well as trace and

time responses and requests) were used. FireShot add-on was used to take webpage screenshots, edit them, using additional annotation tools (highlights) to document the captures and provide screens to the instruction manual presented in Chapter 5.2 [3].

Chrome 16, Firefox 8, Firefox add-ons: Firebug 1.8.4, Tamper Data 11.0.1, Web Developer 1.1.9, FireShot 0.95 were used during development and testing phase of the project.

3.2.5 PAINT.NET

Paint.NET is a free raster graphic editor for MS Windows developed in C#. Despite it is free it has powerful features including support for layers, blending, transparency and plug-ins. It was used to prepare graphics for the website as well as tweak diagrams in this documentation. Version of the program used: Paint.NET 3.5.10.

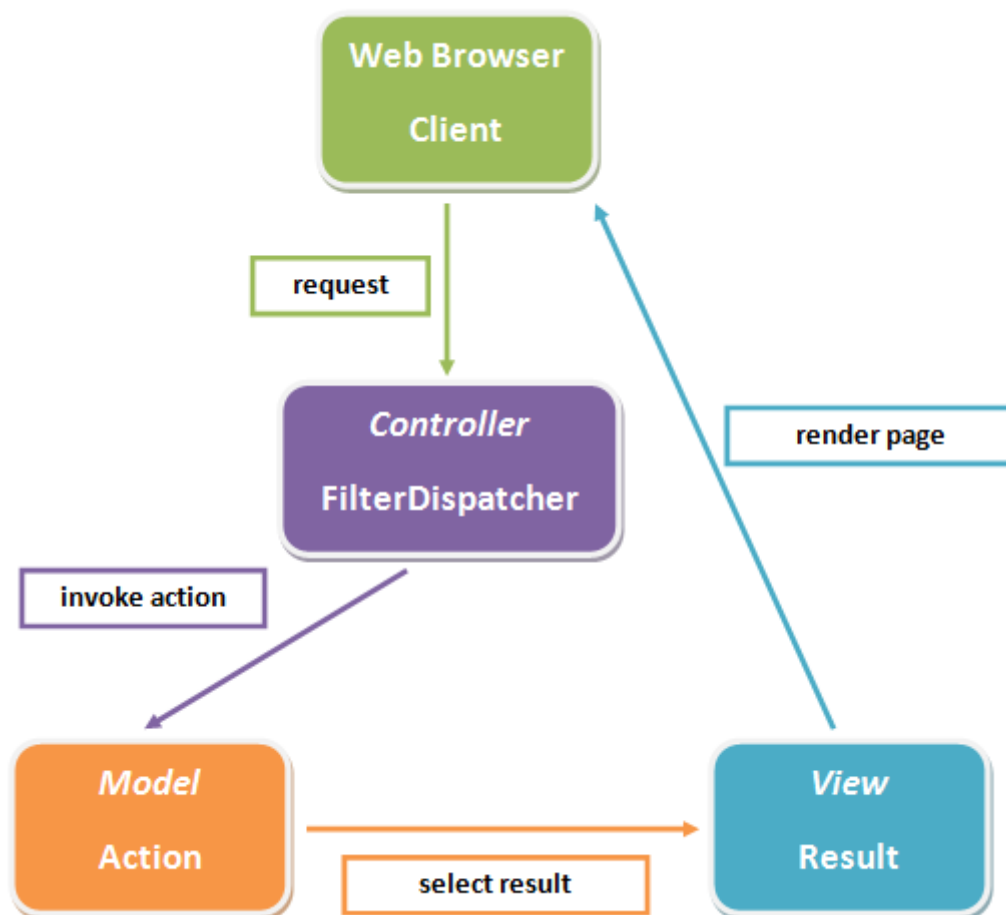
3.2.6 VISUAL PARADIGM FOR UML CE

Visual Paradigm for UML is a UML CASE tool that supports modeling, report generation, code generation. Community Edition (CE) is free of charge for non-commercial use, some of the options are not available, but it was sufficient to prepare Use Case diagram (Chapter 2.7) and Entity Relational diagram (Chapter 2.8) during analysis phase. Version of program used: VP-UML CE 8.2.

3.3 SYSTEM ARCHITECTURE

3.3.1 MODEL-VIEW-CONTROLLER

High-level Struts2 design follows MVC pattern. Separation of concerns in MVC pattern allows for managing large complex software system by dividing them into components. In MVC pattern we encounter three distinct concerns: model, view and controller, which are implemented by the framework as: action, result and `FilterDispatcher` respectively [16].



Struts is often described as a front-controller MVC. This is due to the fact that the controller is out in the front acting as first component in processing. Its task is to map requests to proper actions. This role in Struts2 framework is played by `FilterDispatcher`. This object is actually a servlet filter that investigates all incoming requests to decide which Struts2 actions should handle them. Struts2 servlet filter is registered in deploy descriptor (`web.xml`):

```

1:  <filter>
2:    <filter-name>struts2</filter-name>
3:    <filter-class>
4:      org.apache.struts2.dispatcher.FilterDispatcher
5:    </filter-class>
6:  </filter>
7:  <filter-mapping>
8:    <filter-name>struts2</filter-name>
9:    <url-pattern>/*</url-pattern>
10:  </filter-mapping>

```

Filter class is defined in line 4, and through the appropriate name mapping (lines 2 and 8) it is applied to requests that match all URLs (line 9).

In order to inform the framework which request URLs map to which actions two approaches can be introduced: XML-based configuration files or Java annotations. The project relies on XML file (`struts.xml`):

```

1:  <action
2:    name="Registration"
3:    class="culinary.secure.prelogin.Registration"
4:  >
5:    <result name="input" type="tiles">
6:      /secure/prelogin/Registration.tiles
7:    </result>
8:    <result name="success" type="tiles">
9:      /secure/prelogin/Registered.tiles
10:   </result>
11: </action>
12: <action name="GetDishes"
13:   class="culinary.secure.user.GetDishes">
14:   <result type="json">dishes</result>
15: </action>

```

Model in Struts2 is implemented by action component. Using technical terms, we can describe model as the internal state of application. It is composed both by data model and business logic. These two merge together to form monolithic state of the application. If we consider login process, both data from database and business logic will be involved in authentication. Authentication method will be provided by business logic that will compare username and password against data stored in database, consequently producing one of two states: authenticated or not authenticated. Once the controller finds mapped action, it invokes it giving the control over the request processing. Invocation will both prepare necessary data and execute action's logic. When the work of the action is finished, the view is rendered back to the submitter of the initial request.

View in Struts2 is represented by result whose role is to translate the inner state of the application into visual presentation that the user interacts with. Commonly these are JSP pages, but can be also some other presentation-layer technologies like Tiles or XML- and JSON-results (when Ajax comes into play). This is the task of action to choose proper result, both the type (e.g. JSP, tiles, JSON) and the name (e.g. SUCCESS, ERROR, INPUT) – lines 5, 8, 14 in previous listing.

Struts2 provides clean implementation of MVC idea, but to accomplish this it uses a few other key components that take part in processing the request. These are mainly: interceptors, OGNL and `ValueStack`.

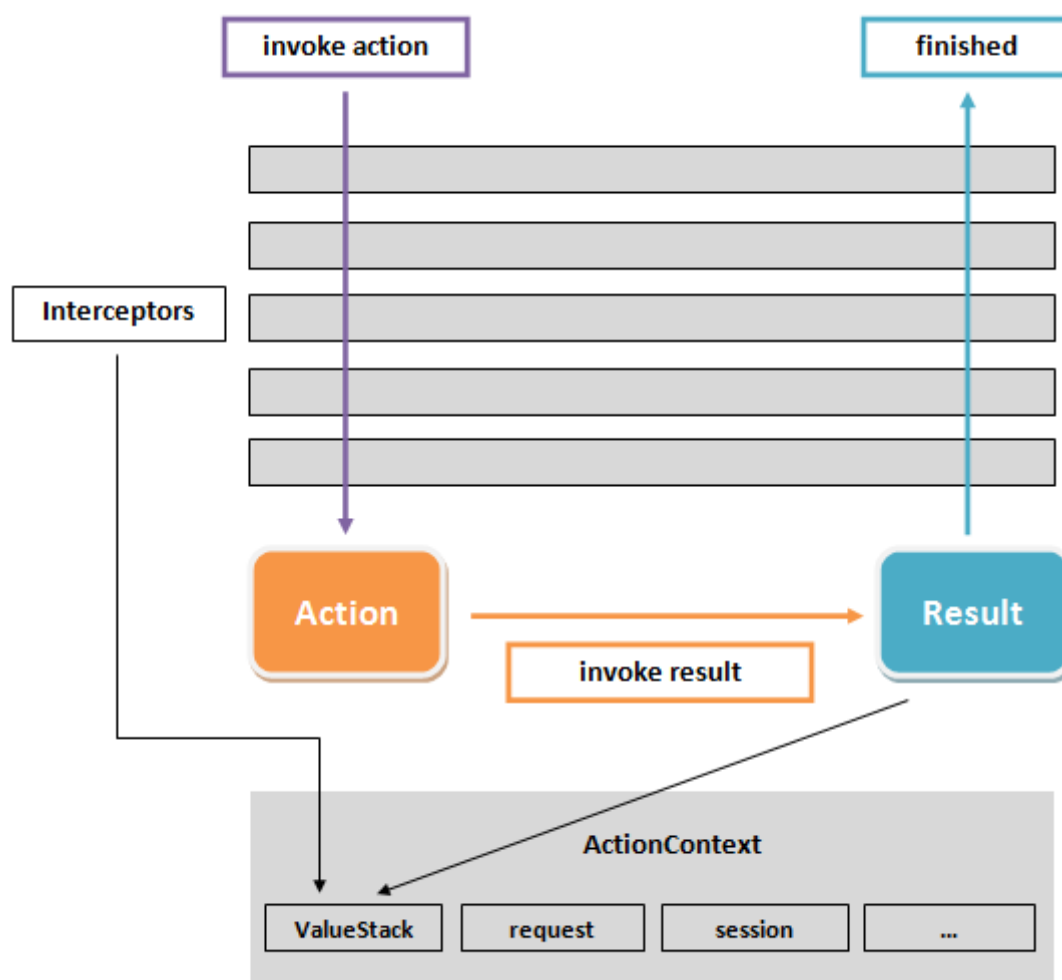


Figure above shows that `FilterDispatcher` has already chosen appropriate action to handle the request, but as you can see there is a stack of interceptors to go through before the request reaches the action. Worth noticing that these interceptors are also triggered after the result has executed. It doesn't mean that they have to do anything both times they are fired, but they can. Some of them do the their job before the action execution

and some afterwards. Interceptors, by the way they are implemented, can perform some reusable clean logic that is kept away from the action logic itself. Struts2 is equipped with some default interceptors (e.g. for data validation and conversion or file uploads), but you can write your own as well. Quite logically appears to have an interceptor that would check if the user can actually invoke the action before the request reaches it, assuming that some actions are available only for logged in users. Such authentication interceptor has been introduced in the project. It had to be configured in deploy descriptor (`web.xml`):

```
1: <interceptors>
2:   <interceptor name="authenticationInterceptor"
3:     class="culinary.utils.AuthenticationInterceptor" />
4:   <interceptor-stack name="secureStack">
5:     <interceptor-ref name="authenticationInterceptor" />
6:     <interceptor-ref name="defaultStack" />
7:   </interceptor-stack>
8: </interceptors>
9: <default-interceptor-ref name="secureStack" />
```

Lines 2 and 3 define authentication interceptor class, lines 4-7 defines new interceptor stack that consists of authentication interceptor (line 5) and Struts2 default stack of interceptors (line 6). This new interceptor stack becomes default one in line 9.

Going to the interceptor class itself, we have to implement `intercept()` method:

```
1: public String intercept(ActionInvocation actionInvocation)
2:   throws Exception {
3:   Map session =
4:     actionInvocation.getInvocationContext().getSession();
5:   User user = (User) session.get("user");
6:   if (user == null) return Action.LOGIN;
7:   else {
8:     Action action = (Action) actionInvocation.getAction();
9:     if (action instanceof UserAware)
10:       ((UserAware) action).setUser(user);
11:     return actionInvocation.invoke();
12:   }
13: }
```

First we use `ActionInvocation` object to obtain session map (lines 3-4), from where we retrieve the user object stored under known key (line 5). If the object is null, it means that the user has not been authenticated via login action (that puts user object into session) and the request is redirected to login action – line 6. Existence of the user object indicates that the user has already logged in. In line 8 we obtain reference to current action, check if it implements `UserAware` interface (line 9) and if so we inject user object into a setter method (line 10) in order to be obtainable from the inside of action class. In this way, every

action that implements mentioned interface will have user object available. Later on (line 11) we pass control to the rest of interceptors and the action.

`ValueStack` is used by the framework as a storage repository for all application data that is needed during request processing. Data is moved to `ValueStack` for the request processing preparation, manipulated there during execution of the action and retrieved from there when result's pages are rendered. Object-Graph Navigation Language (OGNL) is an expression language used for `ValueStack` data retrieval and manipulation. `ValueStack` data is available throughout all phases of request processing as it is stored in `ThreadLocal` context (`ActionContext`). `ActionContext` contains not only `ValueStack` but also request, session and application maps from Servlet API .

To understand the above idea more let us look into the following example:

```
1: <s:form action="RequestIngredient" namespace="/secure">
2:     <s:textfield key="requestingr.name" name="name" />
3:     <s:textfield key="requestingr.unit" name="unit" />
4:     <s:submit key="requestingr.submit" />
5: </s:form>
```

Names of the input fields (lines 2 and 3) are actually OGNL expressions that are resolved against `ValueStack`. Action object is placed onto `ValueStack` when processing request starts and because of the fact that the action class implements appropriate fields, getters and setters (following JavaBeans properties convention), values from input fields are passed to and can be used inside the action object:

```
1: private String name;
2: private String unit;
3: public String getUnit() {
4:     return unit;
5: }
6: public void setUnit(String unit) {
7:     this.unit = unit;
8: }
9: public String getName() {
10:    return name;
11: }
12: public void setName(String name) {
13:    this.name = name;
14: }
```

Moreover these values are accessible (through `ValueStack`) in JSP result pages, again with help of OGNL:


```
1: <p>
2:   <s:property value="name" /> (<s:property value="unit" />)
3: </p>
```

3.3.2 RESULTS

3.3.2.1 TILES RESULT

Apache Tiles is a framework that allow for developing templates in order to simplify user interfaces implementation. Struts2 Tiles plug-in introduces new result type that uses tile definitions for page rendering.

Below listing shows how to configure deploy descriptor to use Tiles plug-in:

```
1: <context-param>
2:   <param-name>
3:   org.apache.tiles.impl.BasicTilesContainer.DEFINITIONS_CONFIG
4:   </param-name>
5:   <param-value>/WEB-INF/tiles.xml</param-value>
6: </context-param>
7: <listener>
8:   <listener-class>
9:   org.apache.struts2.tiles.StrutsTilesListener
10:   </listener-class>
11: </listener>
```

We use context parameter (lines 1-6) to define the location of Tiles definitions file (line 5) and servlet context listener (lines 7-11) to define the integration with the application.

In `struts.xml` (core configuration file of the framework) below, new result type is defined (lines 3-4) that can be used in order to tell Tiles to render the result (lines 7-9):

```
1: <package name="..." namespace="..." extends="struts-default">
2:   <result-types>
3:     <result-type name="tiles"
4:       class="org.apache.struts2.views.tiles.TilesResult" />
5:   </result-types>
6:   <action name="Activate">
7:     <result type="tiles">
8:       /secure/prelogin/Activation.tiles
9:     </result>
10:   </action>
11: </package>
```

tiles.xml contains tile definitions. Listing below show the details of configuration:

```
1: <definition name="baseLayout"
2:   template="/layout/BaseLayout.jsp">
3:   <put-attribute name="header" value="/secure/Header.jsp" />
4:   <put-attribute name="menu_top"
5:     value="/secure/MenuTop.jsp" />
6:   <put-attribute name="sidebar_left" value="" />
7:   <put-attribute name="content"
8:     value="/unsecure/ShowSearch.jsp" />
9:   <put-attribute name="footer"
10:    value="/unsecure/Footer.jsp" />
11: </definition>
12: <definition name="/secure/admin/ShowBlockDish.tiles"
13:   extends="baseLayout">
14:   <put-attribute name="sidebar_left"
15:     value="/secure/SidebarLeftUser.jsp" />
16:   <put-attribute name="content"
17:     value="/secure/admin/ShowBlockDish.jsp" />
18: </definition>
```

First base layout tile was defined (lines 1-11) on top of BaseLayout.jsp file. Other tiles extends base layout tile (line 13) and inserts specific attributes (lines 14-17).

Snippet of BaseLayout.jsp below shows how the attribute is inserted (line 2):

```
1: <div id="header">
2:   <tiles:insertAttribute name="header" />
3: </div>
```

And corresponding Header.jsp (mapped in tiles.xml):

```
1: <%@ page language="java" contentType="text/html; charset=UTF-8"
2:   pageEncoding="UTF-8"%>
3: <%@ taglib prefix="s" uri="/struts-tags"%>
4: <div id="subHeader">
5:   <h1>
6:     <a href="<s:url action='ShowSearch'
7:       namespace='/unsecure' />">
8:       <s:text name="header.title" /></a>
9:   </h1>
10:   <h3>
11:     <a href="<s:url action='ShowSearch'
12:       namespace='/unsecure' />">
13:       <s:text name="header.subtitle" />
14:     </a>
15:   </h3>
16: </div>
```

As we can see, this is a regular JSP page with page and taglib directives (lines 1-3). Taglib directive allows for struts-specific tags usage in JSP page (lines 6-8, 11-13).

3.3.2.2 JSON RESULT

JavaScript Object Notation (JSON) is used for compact text-based serialization of data objects. It is more and more often used as a mean of communication between server and Ajax client. Thus Java data objects can be serialized, sent in a response and changed to JavaScript objects (arrays) on client side.

To obtain above, appropriate custom result had to be introduced: `culinary.utils.JSONResult` – a class that implements interface `com.opensymphony.xwork2.Result`. Serialization takes place in `execute()` method of this class and is performed with some open source packages: XStream [14] (used to serialize Java objects to XML) and Jettison [7] (transforms XML into JSON).

As far as `struts.xml` configuration is concerned it is analogous to Tiles one:

```
1: <result-types>
2:   <result-type name="json"
3:     class="culinary.utils.JSONResult" />
4: </result-types>
5: <action name="GetDishesTitle"
6:   class="culinary.secure.user.GetDishesTitle">
7:   <result type="json">dishes</result>
8: </action>
```

Exemplar JSON response (to Ajax request):

```
{"blockedDish": [{"string": [1, "Dish blocked."]}]}
```

3.3.3 DATABASE ACCESS

Database access is realized with the use of `ServletContextListener` registered in deploy descriptor. This listener is able to listen to two key events in servlet context's life – initialization and destruction [15].

To serve this, `culinary.utils.SQLServletContextListener` class was introduced. Object of this class gets notified when the context is initialized – then it retrieves context init parameters (database credentials and database driver name), use them to make database connection and store this connection as context attribute (`"dbConn"`) to be available in action objects; and when the context is destroyed – closes database connection.

Serving of SQL queries is handled by objects of `SQLQuery` class, created particularly for this purpose, equipped with `select()`, `insert()`, `update()` and `delete()` methods depending on the query itself. Usage of this class inside action class:

```
1:   ResultSet rs = new
2:   SQLQuery(context.getAttribute("dbConn"))
3:   .select("SELECT * FROM INGREDIENTS ORDER BY NAME ASC;");
```

Implementation of `select()` method inside `SQLQuery` class:

```
1:   public ResultSet select(String query) throws SQLException {
2:       return stmt.executeQuery(query);
3:   }
```

Solution applied for database connection seems to be very reasonable considering the fact that there is only one `ServletContext` for entire web application and all the components can share it.

3.3.4 ELEMENTS OF UNOBTUSIVE JAVASCRIPT

Main principle of unobtrusive JavaScript programming is separation of functionality (i.e. behavior layer) from structure and presentation (markup). Following this approach there is no *onclick* event added to any of the tag in whole application. Instead classes and id have been used for DOM elements identification and addition of certain events [12]. An example below (utilizing JQuery):

```
1:   <input type="submit" id="searchTxt_submit" value="Submit" />
2:   <script type="text/javascript" language="JavaScript">
3:       $('#searchTxt_submit').click(function(e) {
4:           portal.dish.model.sendSearch(
5:               '<s:url action="GetDishes" namespace="/unsecure"
6:               includeParams="none" />'
7:           );
8:           e.preventDefault();
9:           e.stopPropagation();
10:      });
11:  </script>
```

Furthermore, clearer separation of functions inside JavaScript itself was introduced. *Onclick* function (line 4) can be treated as a pseudo-controller that triggers `portal.dish.model.sendSearch()` function, that is responsible for obtaining the data (JSON via Ajax call) acting as pseudo-model and which in turns calls pseudo-view function `portal.dish.view.getSearchTitle()` that changes the page without refreshing. Listing below presents the implementation of these functions:

```

1:   var portal = {
2:       dish : {
3:           model : {
4:               sendSearch : function(url) {
5:                   ...
6:                   $.ajax({
7:                       ...
8:                       success : function(data) {
9:                           portal.dish.view.getSearchTitle(data);
10:                        }
11:                    });
12:                }
13:            },
14:            view : {
15:                gotSearchTitle : function(data) {
16:                    var json = data;
17:                    ...
18:                    var s = [];
19:                    ...
20:                    s.push(json parsing);
21:                    ...
22:                    $('#searchResults').empty()
23:                    .html(s.join('\n'));
24:                }
25:            }
26:        }
27:    };

```

Such implementation allows for dot-accessing of functions, which reflects to some extent namespacing that is also mentioned while describing unobtrusive way of programming in JavaScript.

3.4 SYSTEM MODULES

Application has been split into modules based on the criterion of access. That is why we distinguish:

1. Guest applications:

- Dishes applications (Ingredient search, Name search, Dishes Catalogue, Show Dish)
- Prelogin applications (Login, Register, Activate account, Reset password)

2. User applications:

- General applications (Ingredient search, Name search, Dishes Catalogue, Show Dish, Show user)
- Left sidebar applications (Add dish, My dishes/To-do dishes, Request for ingredient, Rewards Catalogue, View/edit profile, My achievements/ranks, Change password)

3. Admin applications:

- Inherited user applications
- Additional left sidebar applications (Add/edit category/ingredient, Browse users, Block dishes, Approve photos)

Such division has been reflected in the `struts.xml` configuration. All mentioned functionalities (actions) have been put into 4 packages:

1. Unsecure – with namespace `/unsecure` (included in actions' URL – over HTTP or HTTPS): Guest dishes applications
2. Prelogin – with namespace `/secure` (over HTTPS): Guest prelogin applications
3. User – with namespace `/secure` (over HTTPS): User applications
4. Admin - with namespace `/secure` (over HTTPS): Admin applications (without those inherited from user)

Different stacks of interceptors have been applied for different packages. For instance, User and Admin package utilizes `AuthenticationInterceptor` that checks whether actions from within these two packages are accessed by logged in user or not, for Admin package `AuthenticationInterceptorAdmin` checks if the logged in user trying to access admin functions has Admin privileges. Interceptors defined for Admin package listed below:

```
1: <package name="admin" namespace="/secure"
2:   extends="struts-default">
3:   <interceptors>
4:     <interceptor name="sessionInterceptor"
5:       class="culinary.utils.SessionInterceptor" />
6:     <interceptor name="authenticationInterceptor"
7:       class="culinary.utils.AuthenticationInterceptor" />
8:     <interceptor name="authenticationInterceptorAdmin"
9:       class="culinary.utils.AuthenticationInterceptorAdmin"
10:    />
11:   <interceptor-stack name="secureStackAdmin">
12:     <interceptor-ref name="sessionInterceptor" />
13:     <interceptor-ref name="authenticationInterceptor"
14:    />
15:     <interceptor-ref
16:       name="authenticationInterceptorAdmin" />
17:     <interceptor-ref name="defaultStack" />
18:   </interceptor-stack>
19: </interceptors>
20: <default-interceptor-ref name="secureStackAdmin" />
21: </package>
```

Even internal construction of Java packages follow the same division as the one in `struts.xml`. We distinguish packages:

1. `culinary.unsecure`
2. `culinary.secure.prelogin`
3. `culinary.secure.user`
4. `culinary.secure.admin`
5. `culinary.utils`

All these packages contain action classes except for `culinary.utils` which encompasses helper classes, such as:

- custom interceptors
- `JSONResult` (Chapter 3.3.2.2)
- `PasswordIntegrityValidator` (Chapter 3.5.2.3)
- `SendMailSSL` – used for sending emails
- `SHA256` – used for encrypting password for database storage/check
- `User` – stored in session after login
- `SQLServletContextListener` (Chapter 3.3.3)

3.5 DATA VALIDATION

3.5.1 FRONT-END VALIDATION

Due to the fact that Struts2 framework is equipped with strong back-end validation features, front-end validation was limited, although sometimes used. Most common approach was to overwrite submit function of forms using JQuery:

```
1:  $('form#AddDish').submit(function() {
2:    ...
3:    if ($('#AddDish_category option:selected').val() == "-1"){
4:      $('#AddDish_category')
5:        .css('border-color','red').css('border-width','3px');
6:      $('#AddDish_category').focus();
7:      return false;
8:    }
9:    ...
10:   return true;
11: });
```

Client-side validation was also used before triggering Ajax calls:

```
1:  sendSearchTitle : function(url, txt) {
2:      ...
3:      var intRegex = /^[aćęłńóśźżĄĆĘŁŃÓŚŻŻa-z A-Z]*$/;
4:      if (!intRegex.test(txt)) {
5:          $('#searchTxt').css('border', '3px solid red');
6:          $('#searchTxt').focus();
7:          $('#searchTxt_error').css('display', 'inline')
8:          return false;
9:      }
10:     $.ajax({...});
11: },
```

3.5.2 BACK-END VALIDATION

3.5.2.1 VALIDATE METHOD OF ACTION

One of the default interceptors in Struts2 is `workflow` interceptor that checks the destination action if it implements `validate()` method that contains the logic for checking the validity of the data passed as request POST parameters.

This kind of validation was used only once for `AddDish` action to check negative values of parameters. Implementation of the `validate()` method:

```
1:  public void validate() {
2:      if (getAmountofcalories()<0) {
3:          setAmountofcalories(0);
4:          addFieldError("amountofcalories",
5:              getText("addDish.negCal"));
6:      }
7:      ...
8:  }
```

3.5.2.2 EXECUTE METHOD OF ACTION

Validation in `execute()` method of the action was only used in cases when there was a specific need to validate some value against the database content – for `Registration` action checking if user with given e-mail already exists:

```
1:  public String execute() throws Exception {
2:      ResultSet rs = new SQLQuery(context.getAttribute("dbConn"))
3:          .select("SELECT * FROM USERS WHERE EMAIL='"+email+"'");
4:      if(SQLQuery.getResultSetSize(rs)>0) {
5:          addFieldError( "email",
6:              getText( "registration.err.email.exists" ));
7:          return INPUT;
8:      }
9:      ...
10: }
```


3.5.2.3 VALIDATION FRAMEWORK

Struts2 provides an easy way of validating data through `validation` interceptor (one of the default interceptors) that cooperates with `workflow` interceptor triggered right after. The difference between these two is that `workflow` interceptor calls `validate()` method of the action and `validation` interceptor performs the validation logic itself. This logic can be placed in XML files (as it was done in the application) or using Java annotations inside action class. XML files are named after the action they are working for: *ActionName-validation.xml*.

Let us have a look at `ChangePassword-validation.xml`:

```
1:  <validators>
2:    <field name="password1">
3:      <field-validator type="requiredstring">
4:        <message key="changepass.err.password.required" />
5:      </field-validator>
6:      ...
7:      <field-validator type="passwordintegrity">
8:        <param name="specialCharacters">$!@#?</param>
9:        <message key="changepass.err.password.integrity" />
10:      </field-validator>
11:      ...
12:    </validators>
```

Project most often uses field validators as shown above. Struts2 has many built-in types of validators, like `requiredstring` validator (lines 3-5), that checks if the field was not sent empty.

Validators can be developed as well. Custom validators have to be registered in `validators.xml`:

```
1:  <validators>
2:    <validator name="passwordintegrity"
3:      class="culinary.utils.PasswordIntegrityValidator"/>
4:  </validators>
```

Class `culinary.utils.PasswordIntegrityValidator` is used for better password security in order to enforce users to choose strong passwords that includes small and capital letter, number and a special character.

3.6 INTERFACE DESIGN

User interface assumes presence of 5 areas: header (1), top menu(2), left sidebar(3), workarea(4) and footer(5).

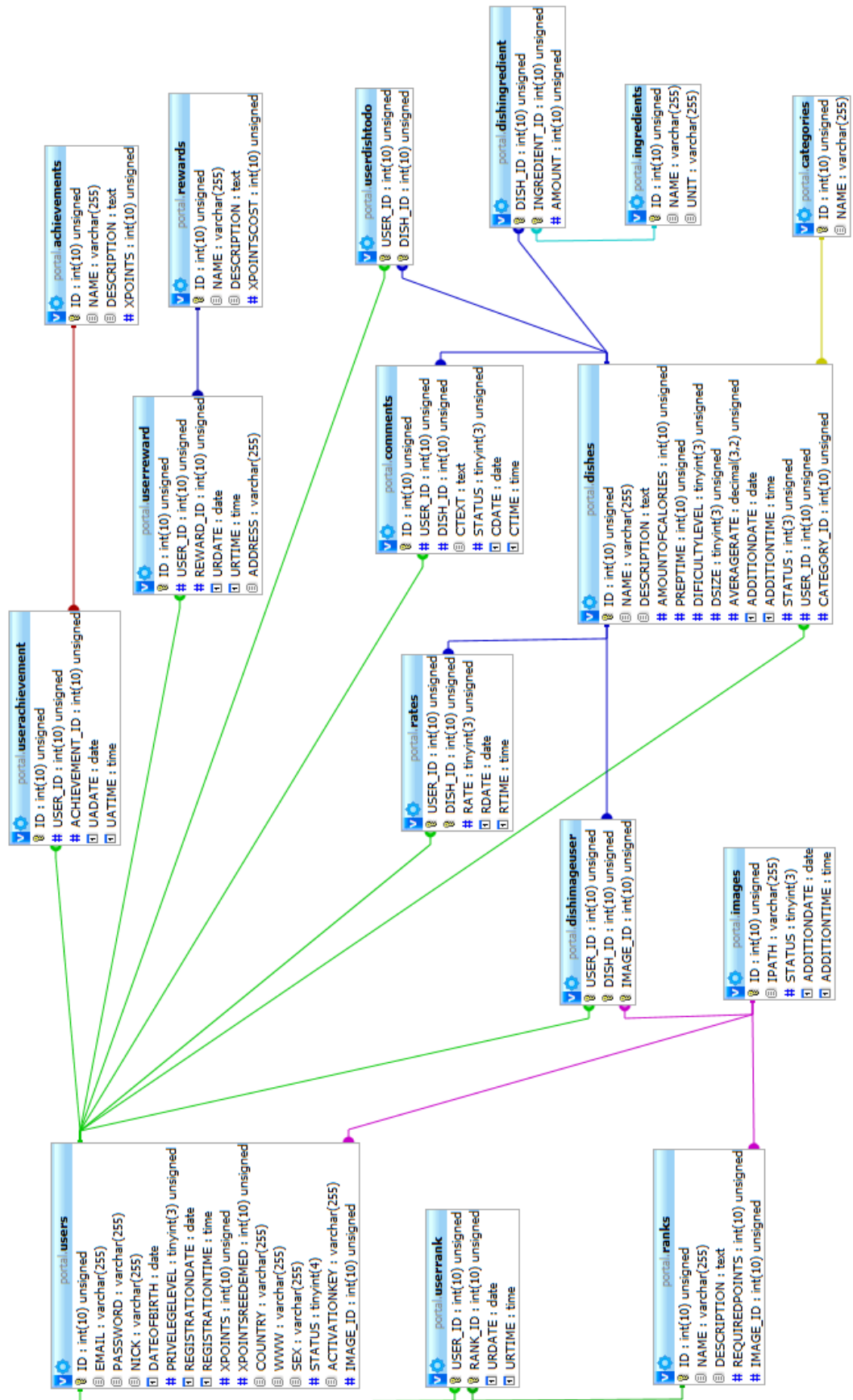


Header has graphical background, it presents name of the website which is a link to homepage (Ingredient search). Top menu is a horizontal menu that encompasses links to three main ways of browsing the dishes. Left sidebar is a vertical menu that has different set of links for guest, user and admin. For admin and user, on top of this sidebar some of the user details are shown along with the photo of the rank or own photo uploaded by the user. Workarea is changing in relation to application triggered at the moment. Footer includes copyright information.

Assumptions that were undertaken during the design of the user interface are as follows:

- Form fields should be large enough and accompanied by valid labels
- Form input errors ought to be appropriately visible and highlighted with red color
- Text ought to be readable - can be obtained by proper matching of font color against background color, appropriate font size and line height
- Links ought to be different from the rest of the text (highlighted)
- Logout button ought to be especially emphasized

3.7 ENTITY RELATIONSHIP DIAGRAM (DESIGN)



3.8 DESCRIPTION OF ENTITIES AND ATTRIBUTES

3.8.1 TABLE STRUCTURE FOR TABLE ACHIEVEMENTS

Column	Type	Null	Default	PK	FK
ID	int(10)	No		X	
NAME	varchar(255)	No			
DESCRIPTION	text	No			
XPOINTS	int(10)	No	0		

3.8.2 TABLE STRUCTURE FOR TABLE CATEGORIES

Column	Type	Null	Default
ID	int(10)	No	
NAME	varchar(255)	No	

3.8.3 TABLE STRUCTURE FOR TABLE COMMENTS

Column	Type	Null	Default	PK	FK
ID	int(10)	No		X	
<u>USER_ID</u>	int(10)	No			X
<u>DISH_ID</u>	int(10)	No			X
CTEXT	text	No			
STATUS	tinyint(3)	No	1		
CDATE	date	No			
CTIME	time	No			

3.8.4 TABLE STRUCTURE FOR TABLE DISHES

Column	Type	Null	Default	PK	FK
ID	int(10)	No		X	
NAME	varchar(255)	No			
DESCRIPTION	text	No			
AMOUNTOFCALORIES	int(10)	No	0		
PREPTIME	int(10)	No	0		
DIFICULTYLEVEL	tinyint(3)	No	0		
DSIZE	tinyint(3)	No	0		
AVERAGERATE	decimal(3,2)	No	0.00		
ADDITIONDATE	date	No			
ADDITIONTIME	time	No			
STATUS	int(3)	No	1		
<u>USER_ID</u>	int(10)	No			X
<u>CATEGORY_ID</u>	int(10)	No			X

3.8.5 TABLE STRUCTURE FOR TABLE DISHIMAGEUSER

Column	Type	Null	Default	PK	FK
<u>USER_ID</u>	int(10)	No		X	X
<u>DISH_ID</u>	int(10)	No		X	X
<u>IMAGE_ID</u>	int(10)	No		X	X

3.8.6 TABLE STRUCTURE FOR TABLE DISHINGREDIENT

Column	Type	Null	Default	PK	FK
<u>DISH_ID</u>	int(10)	No		X	X
<u>INGREDIENT_ID</u>	int(10)	No		X	X
AMOUNT	int(10)	No	1		

3.8.7 TABLE STRUCTURE FOR TABLE IMAGES

Column	Type	Null	Default	PK	FK
ID	int(10)	No		X	
IPATH	varchar(255)	No			
STATUS	tinyint(3)	No	0		
ADDITIONDATE	date	No			
ADDITIONTIME	time	No			

3.8.8 TABLE STRUCTURE FOR TABLE INGREDIENTS

Column	Type	Null	Default	PK	FK
ID	int(10)	No		X	
NAME	varchar(255)	No			
UNIT	varchar(255)	No			

3.8.9 TABLE STRUCTURE FOR TABLE RANKS

Column	Type	Null	Default	PK	FK
ID	int(10)	No		X	
NAME	varchar(255)	No			
DESCRIPTION	text	No			
REQUIREDPOINTS	int(10)	No	0		
<u>IMAGE_ID</u>	int(10)	Yes	NULL		X

3.8.10 TABLE STRUCTURE FOR TABLE RATES

Column	Type	Null	Default	PK	FK
<u>USER_ID</u>	int(10)	No		X	X
<u>DISH_ID</u>	int(10)	No		X	X
RATE	tinyint(3)	No	0		
RDATE	date	No			
RTIME	time	No			

3.8.11 TABLE STRUCTURE FOR TABLE REWARDS

Column	Type	Null	Default	PK	FK
ID	int(10)	No		X	
NAME	varchar(255)	No			
DESCRIPTION	text	No			
XPOINTSCOST	int(10)	No	0		

3.8.12 TABLE STRUCTURE FOR TABLE USERACHIEVEMENT

Column	Type	Null	Default	PK	FK
ID	int(10)	No		X	
<u>USER_ID</u>	int(10)	No			X
<u>ACHIEVEMENT_ID</u>	int(10)	No			X
UUPDATE	date	No			
UATIME	time	No			

3.8.13 TABLE STRUCTURE FOR TABLE USERDISHTODO

Column	Type	Null	Default	PK	FK
<u>USER_ID</u>	int(10)	No		X	X
<u>DISH_ID</u>	int(10)	No		X	X

3.8.14 TABLE STRUCTURE FOR TABLE USERRANK

Column	Type	Null	Default	PK	FK
<u>USER_ID</u>	int(10)	No		X	X
<u>RANK_ID</u>	int(10)	No		X	X
URDATE	date	No			
URTIME	time	No			

3.8.15 TABLE STRUCTURE FOR TABLE USERREWARD

Column	Type	Null	Default	PK	FK
<u>ID</u>	int(10)	No		X	
USER_ID	int(10)	No			
REWARD_ID	int(10)	No			
URDATE	date	No			
URTIME	time	No			
ADDRESS	varchar(255)	No			

3.8.16 TABLE STRUCTURE FOR TABLE USERS

Column	Type	Null	Default	PK	FK
<u>ID</u>	int(10)	No		X	
EMAIL	varchar(255)	No			
PASSWORD	varchar(255)	No			
NICK	varchar(255)	No			
DATEOFBIRTH	date	No			
PRIVELEGELEVEL	tinyint(3)	No	0		
REGISTRATIONDATE	date	No			
REGISTRATIONTIME	time	No			
XPOINTS	int(10)	No	0		
XPOINTSREDEEMED	int(10)	No	0		
COUNTRY	varchar(255)	Yes	NULL		
WWW	varchar(255)	Yes	NULL		
SEX	varchar(255)	Yes	NULL		
STATUS	tinyint(4)	No	0		
ACTIVATIONKEY	varchar(255)	No			
<u>IMAGE_ID</u>	int(10)	Yes	NULL		X

3.9 DATABASE TRIGGERS

3.9.1 AFTER INSERT TRIGGER ON RATES TABLE

Calculation of average rate for given dish is triggered after insertion of new row to RATES table (when user rate the dish). Average rate of dish is updated with calculated value.

3.9.2 AFTER INSERT TRIGGER ON DISHES TABLE

After the insertion of new row to DISHES table (when user adds a dish) appropriate type of achievement is assigned to the user – new row is inserted into USERACHIEVEMENT table.

3.9.3 AFTER UPDATE TRIGGER ON IMAGES TABLE

When given image (of avatar or dish) is being approved by admin (STATUS field updated in IMAGES table), appropriate achievement is assigned to the user that added the photo – new row is inserted into USERACHIEVEMENT table.

3.9.4 AFTER INSERT TRIGGER ON USERACHIEVEMENT TABLE

When new achievement is assigned to user (performed by triggers in 3.9.2 and 3.9.3), appropriate amount of experience points (specific for given achievement) is added to the experience points currently owned by the user – experience points field in USERS table is updated.

3.9.5 AFTER UPDATE TRIGGER ON USERS TABLE

This trigger performs two actions:

- when new user activates his account (status field is updated in USERS table from 0 to 1) lowest rank is assigned to him – new row is inserted into USERRANK table
- when experience points are updated for given user (performed by trigger in 3.9.4) it is checked if new value of experience points is sufficient to assign higher rank to the user – if so, new row is inserted into USERRANK table

3.9.6 AFTER INSERT TRIGGER ON USERREWARD TABLE

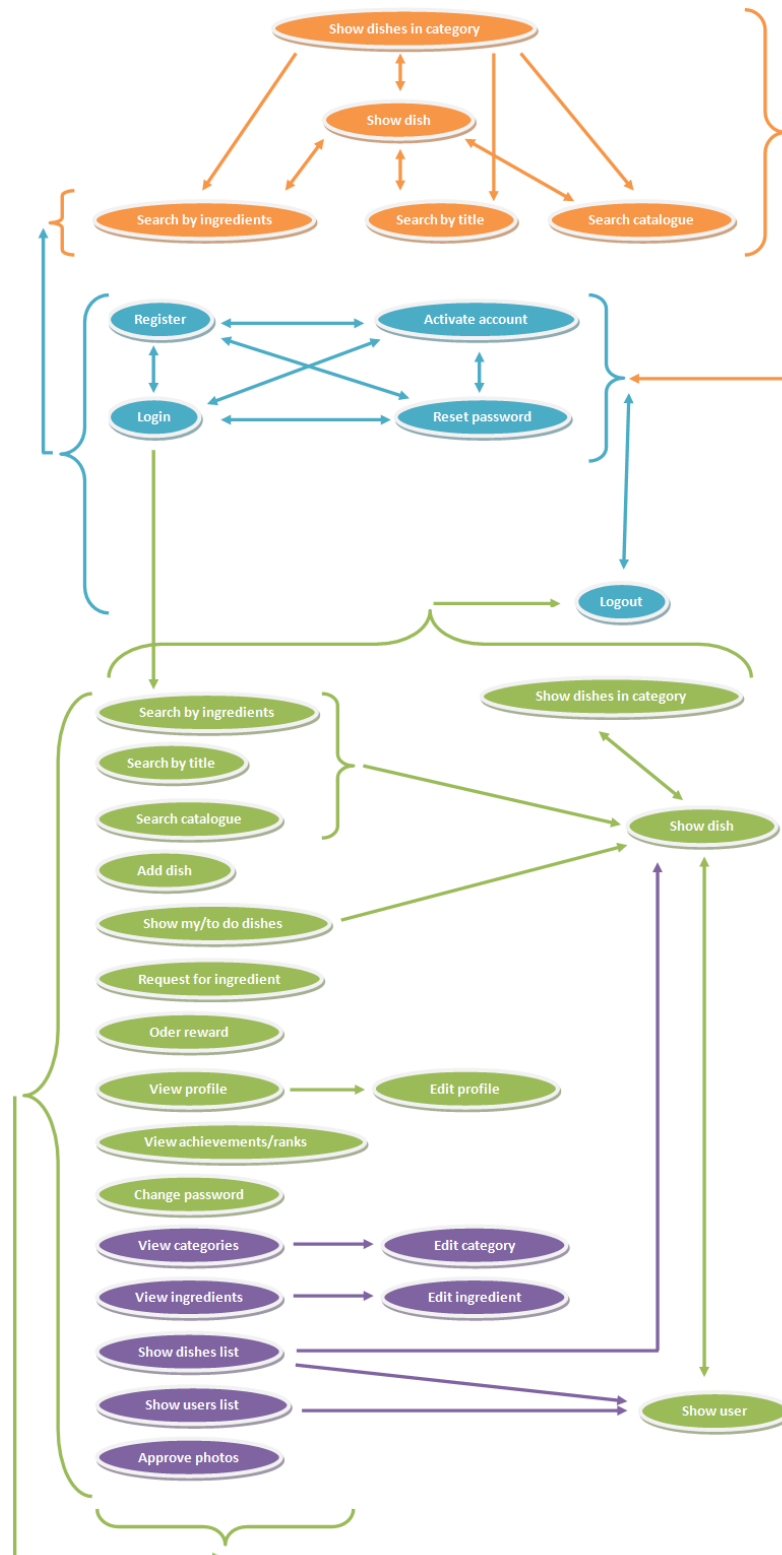
After new row is inserted into USERREWARDS (when user order a reward), the points cost of the reward is added to experience points of the user that has been used for redemption – appropriate field in USERS table is updated.

3.10 SOFTWARE/HARDWARE REQUIREMENTS

- Client: at least one browser installed – FF 8.0+, IE 8+, Opera 11+, Chrome 12+ with JavaScript
- Server minimal configuration: Dual Core 2x1.5 MHz, 2048 MB RAM, 20 GB disc space, Linux 3.0, JDK 1.7, Apache Tomcat 7, MySQL Server 5.5

3.11 SCREEN TRANSITION DIAGRAM

Orange screens belong to the unsecured part of the application. Blue screens are secure pre-login screens. Green and violet screens are post-login secure screens. Violet ones are accessible only by administrators, whereas green ones are accessible both for ordinary users and administrators.



4 TESTING

4.1 WHITE BOX TESTING

White box testing, also called glass box testing, is a method of testing software basing on the knowledge about the internal design and structure of the application. Best practice is to run it during the coding stage by the programmers themselves. It can be also run later, in the testing phase of the project, but it requires from the testers to have programming knowledge to prepare test cases – obviously this results in increase of the project cost [17].

White box testing was conducted during the development phase of the project. Each module underwent exhaustive path testing – all possible paths of control flow through the program were investigated [18]. Revealed bugs were fixed immediately.

4.2 BLACK BOX TESTING

4.2.1 BLACK BOX TESTING CHARACTERISTICS

Black box testing treats program as a black box. Functions are tested by feeding them with input and examining the output (black box testing is also called input/output-driven testing). No focus is given to internal behavior or structure of the program. The goal is to find the circumstances in which the program behaves differently to the assumptions defined in specification. [17] [18]

As far as black box testing is concerned, project underwent user acceptance testing, sanity testing, monkey testing and stress testing.

4.2.2 USER ACCEPTANCE TESTING

User acceptance testing is one of the types of black box testing, usually run as a final testing before software delivery [17]. Its aim is to compare the program with initial requirements and the current needs of the end users [18].

To conduct user acceptance testing number of test cases were prepared, basing on the criterion of exhaustive input testing to make use of every possible input condition as a test case. Tests were run in Firefox 8 (Ubuntu 11.10) and Internet Explorer 8 (Windows 7). The results are shown below:

ID	Application	Credentials	Test Case	Expected Results	Actual Results	OK / Not OK
1	Ingredient search (unsecure)	-	Input different combination of ingredients, click search, check dishes for ingredients	Ingredient search working as per requirements	Ingredient search working as per requirements	OK
2	Name search (unsecure)	-	Input different names, click search, if results found check if searched phrase is in the name of the dish	Name search working as per requirements	Name search working as per requirements	OK
3	Dishes Catalogue (unsecure)	-	Browse dishes in the catalogue, go into dish details and check if the dish is in given category	Dishes Catalogue working as per requirements	Dishes Catalogue working as per requirements	OK
4	Dish details (unsecure)	-	Browse dishes in the catalogue, go into dish details and check if the dish is in given category	Sections of the dish shown as per requirements	Sections of the dish shown as per requirements	OK
5	Registration	-	Register new user - whole flow	Registration working as per requirements (email sent to new user)	Registration working as per requirements (email sent to new user)	OK
6	Account activation	-	Activate account - use key sent in 5.	Account activation working as per requirements	Account activation working as per requirements	OK
7	Login	-	Login with the credentials from 5.	Login working as per requirements	Login working as per requirements	OK
8	Password reset	-	Reset password and then login with the new one sent to email	Reset password working as per requirements	Reset password working as per requirements	OK
9	Ingredient search (secure)	user	Input different combinations of ingredients, click search, check dishes for ingredients	Ingredient search working as per requirements	Ingredient search working as per requirements	OK
10	Name search (secure)	user	Input different names, click search, if results found check if searched phrase is in the name of the dish	Name search working as per requirements	Name search working as per requirements	OK
11	Dishes Catalogue (secure)	user	Browse dishes in the catalogue, go into dish details and check if the dish is in given category	Dishes Catalogue working as per requirements	Dishes Catalogue working as per requirements	OK
12	Dish details (secure)	user	Check dish screen if it contains sections for secure access	Sections of the dish shown as per requirements	Sections of the dish shown as per requirements	OK
13	Add dish/My dishes	user	Input add dish form couple of times with different inputs, check behaviour of the application (validation), go to My dishes section and verify if the list of the dishes is complete	Adding dish working as per requirements	Adding dish working as per requirements	OK
14	To do dishes	user	Browse the dishes, add some dishes to to-do list, check To-do dishes if all added dishes are on the list	Adding dishes to to-do list working as per requirements	Adding dishes to to-do list working as per requirements	OK
15	My dishes/ To do dishes/ Approve photos	user/ admin	Add photos to dishes (both my dishes and to-do dishes), login with admin credentials, go to Approve photos application, check list of photos to be approved, approve photos, go back to previous account and check dish details screen if the photos are present	Adding photos and approving them working as per requirements	Adding photos and approving them working as per requirements	OK

16	Request for ingredient	user	Request for new ingredient, check admin's mail if notification came	Request for ingredient application working as per requirements	Request for ingredient application working as per requirements	OK
17	Reward catalogue	user	Order reward, check order history, check available points if updated	Reward catalogue working as per requirements	Reward catalogue working as per requirements	OK
18	View/edit profile	user	View own profile, then edit and save, check changes	View and edit profile working as per requirements	View and edit profile working as per requirements	OK
19	View/edit profile	user / admin	View own profile, upload photo, login with admin credentials, go to Approve photos application, check list of photos to be approved, approve photos, go back to previous account and check if new photo is shown (in left sidebar as well, instead of rank photo)	Uploading own photo working as per requirements	Uploading own photo working as per requirements	OK
20	My achievements/ ranks	user	Go to my achievements and ranks screen, check if the points are assigned accordingly to the previous actions (dish addition, photos upload and approval), check if the ranks are assigned accordingly to the points assigned	My achievements/ranks logic working as per requirements	My achievements/ranks logic working as per requirements	OK
21	Change password	user	Change password, logout and try to login with old and new password	Changing password working as per requirements	Changing password working as per requirements	OK
22	Add/edit category	admin	View current categories, add new one and check if list of categories updated	Adding and editing category working as per requirements	Adding and editing category working as per requirements	OK
23	Add/edit ingredient	admin	View current ingredients, add new one and check if list of ingredients updated	Adding and editing ingredient working as per requirements	Adding and editing ingredient working as per requirements	OK
24	Browse users	admin	Check list of users, edit some of them and check if the changes are reflected	Browsing and editing users working as per requirements	Browsing and editing users working as per requirements	OK
25	Browse users	admin/ user	Check list of users, block some of them, logout and try to login with blocked user's credentials	Blocking users working as per requirements (blocked user cannot login)	Blocking users working as per requirements (blocked user cannot login)	OK
26	Block dishes	admin	Go to the application, block some dishes and then try to access them	Blocking dishes working as per requirements (blocked dish cannot be accessed)	Blocking dishes working as per requirements (blocked dish cannot be accessed)	OK
27	Show user	user	Go to dish details screen, click nickname of the user that added the dish, check details of the user, check this for different users	Show user screen working as per requirements	Show user screen working as per requirements	OK

4.2.3 SANITY TESTING

Sanity test is a basic, brief evaluation of program's functionality, run in order to verify if the system works roughly as expected [11].

Sanity testing was conducted for the project in Chrome 16 and Opera 11.6 (both Windows 7) and Opera Mobile 11.5 (Android 2.3). No functional issues were encountered, layout CSS-driven issues were fixed.

4.2.4 MONKEY TESTING

Monkey testing is a type of black box testing that runs with no specific test in mind. It was conducted by various users in random browsers, performing random action. All revealed bug were fixed.

4.2.5 STRESS TESTING

Stress tests investigates program's response to increased load activity. If the amount of activity that the program is to handle is defined, the stress test attempts to check whether the program fails below or above specified level [17]. Web-based applications are one of the most common recipients of stress testing, which aim is to ensure that the application, and hardware, can handle some volume of concurrent users [18].

For the purposes of the project a special page was prepared (`/portal/stressTest.html`) that uses JavaScript to send Ajax calls to post-login applications. Its aim is to simulate a regular user's behavior that goes from one sub-application to another by clicking the left menu links. After the page is opened JavaScript initiates the call to login page (with admin credentials), users data is pre-pended to the body of the testing page, then consecutive Ajax calls to the list of applications that are normally accessible for the logged in admin are triggered, ending with Logout call. These calls are sent with interval of 3 seconds plus the time of getting the result of Ajax call (which corresponds to time of loading a regular page in a browser). After the cycle is completed, new one begins until the page is closed. The log output shows the time of initiating the call, name of the resource called and time of getting the response.



user: **mszonline**
from: **2011-12-11 20:00**
rank: **Kucharz Laik**

Log:

13:6:3	PerformLogin	247
13:6:7	ShowAddDish	1502
13:6:11	ShowMyDishes	186
13:6:14	ShowRequestIngredient	184
13:6:18	ShowRewards	1279
13:6:21	ShowChangePassword	180
13:6:26	ShowAddCategory	1559
13:6:29	ShowAddIngredient	221
13:6:34	ShowUsers	1492
13:6:37	ShowBlockDish	177
13:6:41	ShowApprovePhoto	1314
13:6:44	Logout.action	81
13:6:49	PerformLogin	1574
13:6:52	ShowAddDish	435
13:7:2	ShowMyDishes	6516
13:7:7	ShowRequestIngredient	1767
13:7:10	ShowRewards	212
13:7:13	ShowChangePassword	156
13:7:16	ShowAddCategory	332

To simulate some number of concurrent users Internet Explorer 9 was used, as this browsers can open its windows as new frame processes (when `iexplore.exe -`

noframemerging is used), allowing for multiple logins to web application. For this purpose a special batch file was prepared:

```
1: @echo off
2: :start
3: set /p cnt=how many windows ?:
4: set /p coi=what interval in seconds?:
5: if %cnt% GEQ 51 goto cmon
6: if %cnt% LEQ 0 goto cmon
7: if %coi% LEQ 0 goto cmon
8: set re=0
9: :1
10: set /a re+=1
11: TIMEOUT %coi%
12: echo window %re% is opening
13: @start "" /b "C:\Program Files\Internet Explorer\iexplore.exe"
    "-noframemerging
    https://89.77.32.55:8443/portal/stressTest.html?interval=3"
14: if %re%==%cnt% goto start
15: goto 1
16: :cmon
17: echo incorrect value
18: goto start
```

Fifty windows of Internet Explorer were opened with address of /portal/stressTest.html. The test lasted for 30 minutes. Average times of getting response for the last three full cycles (login - *show* actions - logout) were calculated:

Action	Average time
PerformLogin	2886 (ms)
ShowAddDish	2793 (ms)
ShowMyDishes	2257 (ms)
ShowRequestIngredient	2051 (ms)
ShowRewards	2385 (ms)
ShowChangePassword	2194 (ms)
ShowAddCategory	3595 (ms)
ShowAddIngredient	3599 (ms)
ShowUsers	4670 (ms)
ShowBlockDish	2818 (ms)
ShowApprovePhoto	1982 (ms)
Logout	1439 (ms)

It was also checked that none of the response times during the test conduction exceeded 10000ms. The assumed performance criterion of 50 concurrent users that open the pages in less than 10 seconds occurred to be valid (Chapter 2.6.1).

Hardware configuration for the server used during the test: Dual Core 2x1.5 MHz, 2048 MB RAM, 20 GB disc space. This configuration meets the requirement of minimal hardware server configuration (Chapter 3.10).

5 MANUALS

5.1 INSTALLATION MANUAL

5.1.1 INTRODUCTION

As a production server environment for the application Linux Mint 11 LXDE was used. Thus all manuals will be applicable to this distribution, as well as it should be applicable to all Ubuntu 11.04-based distributions.

5.1.2 TOMCAT INSTALLATION

Before you install Tomcat you need to take care of Java installation first. In order to do that you need to download Java Development Kit from Oracle website. To unpack the archive run in the directory to which you have downloaded the JDK:

```
tar xvzf jdk-7u1-linux-i586.tar.gz
```

Then move the created folder to proper destination:

```
sudo mv jdk1.7.0_01/ /usr/local
```

To setup *new* Java to be used by the system by default run:

```
sudo update-alternatives --install "/usr/bin/java" "java"  
"/usr/local/jdk1.7.0_01/bin/java" 1  
sudo update-alternatives --set java /usr/local/jdk1.7.0_01/bin/java  
sudo update-alternatives --install "/usr/bin/javac" "javac"  
"/usr/local/jdk1.7.0_01/bin/javac" 1  
sudo update-alternatives --set javac  
/usr/local/jdk1.7.0_01/bin/javac
```

To setup `JAVA_HOME` and `JRE_HOME` variables run:

```
sudo gedit /etc/environment
```

and add/edit records:

```
JAVA_HOME = "/usr/local/jdk1.7.0_01/bin/" (add)  
JRE_HOME = "/usr/local/jdk1.7.0_01/jre/" (add)  
PATH = "... (other routes) : $JAVA_HOME : $JRE_HOME (edit)
```

Tomcat can be downloaded from Apache Tomcat website. To unpack the archive run in the directory to which you have downloaded Tomcat:

```
tar xvzf apache-tomcat-7.0.23.tar.gz
```

Then move the created folder to proper destination:

```
sudo mv apache-tomcat-7.0.23/ /usr/share/tomcat7
```

To be sure that the Java paths are defined you can edit `catalina.sh` file:

```
sudo gedit /usr/share/tomcat7/bin/catalina.sh
```


and add records on top of this file:

```
JAVA_HOME = "/usr/local/jdk1.7.0_01/bin/"
JRE_HOME = "/usr/local/jdk1.7.0_01/jre/"
```

5.1.3 MYSQL INSTALLATION

To install MySQL server it is enough to run in terminal:

```
sudo apt-get install mysql-server
```

During the installation you will be asked to input root password that will be used to login to MySQL server.

For database and user setup you need to login to MySQL server and upload sql file:

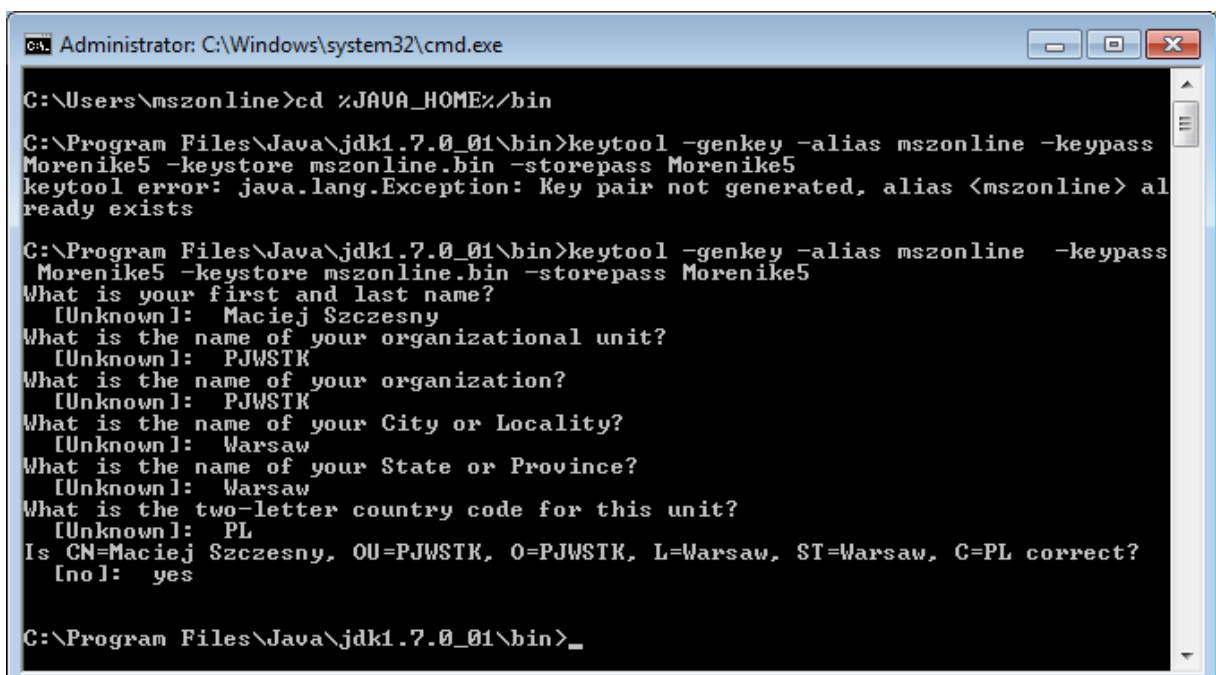
```
mysql -h localhost -u root -p
(root password input)
mysql> source /pathToFile/initialQuery.sql
```

5.1.4 CERTIFICATE GENERATION

Due to the fact that the application uses HTTPS connection it is required to use appropriate certification. For testing purposes you can use self-signed certificate. First you need to generate the *KeyStore* file using *keytool* executable shipped with JDK (can be done on Windows or Linux):

```
cd %JAVA_HOME%/bin (on Windows)
cd $JAVA_HOME/bin (on Linux)
keytool -genkey -alias mszonline -keypass Morenike5 -keystore
mszonline.bin -storepass Morenike5
```

Questionnaire will be shown for filling in:



The command will make a .bin file with the name you provided inside the bin directory itself.

This file needs to be moved to webapps directory of Tomcat:

```
sudo mv mszonline.bin /usr/share/tomcat7/webapps
```

Next you need to alter `server.xml` file inside Tomcat directory in order to tell Tomcat to use the *Keystore* which was created in the earlier step:

```
sudo gedit /usr/share/tomcat7/conf/server
```

The entry for connector with port 8443 has to be changed to:

```
<Connector port="8443" protocol="HTTP/1.1"
    SSLEnabled="true"
    maxThreads="150"
    scheme="https"
    secure="true"
    clientAuth="false"
    sslProtocol="TLS"
    keystoreFile="${catalina.home}/webapps/mszonline.bin"
    keystorePass="Morenike5"
/>
```

`web.xml` file of the application has already been modified appropriately:

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>securedapp</web-resource-name>
    <url-pattern>/secure/*</url-pattern>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

5.1.5 FINAL STEPS

Application is archived into `portal.war` file. This file needs to be moved to webapps directory of Tomcat:

```
sudo mv portal.war /usr/share/tomcat7/webapps
```

MySQL server should already be running, the only thing left is to start Tomcat:

```
sudo /usr/share/tomcat7/bin/startup.sh
```

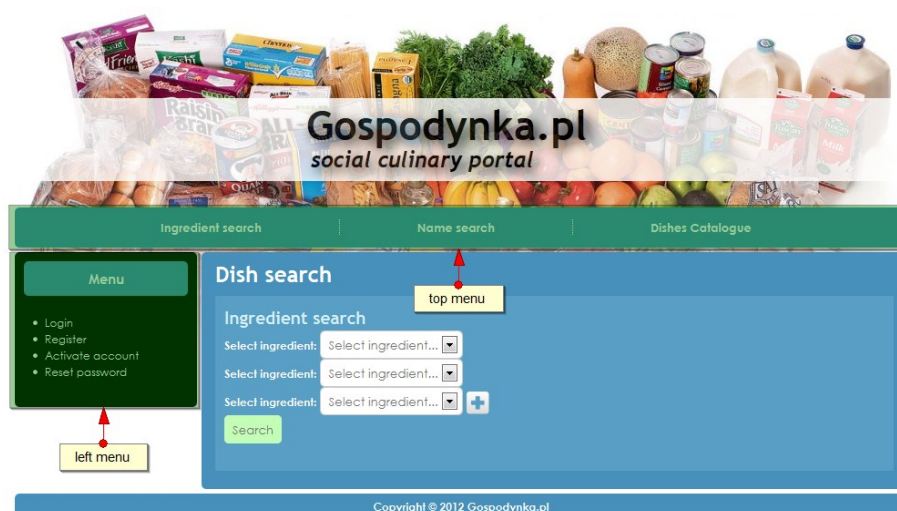
Application can be access in any browser with `http://localhost:8080/portal/`

5.2 INSTRUCTION MANUAL

5.2.1 GUEST MANUAL

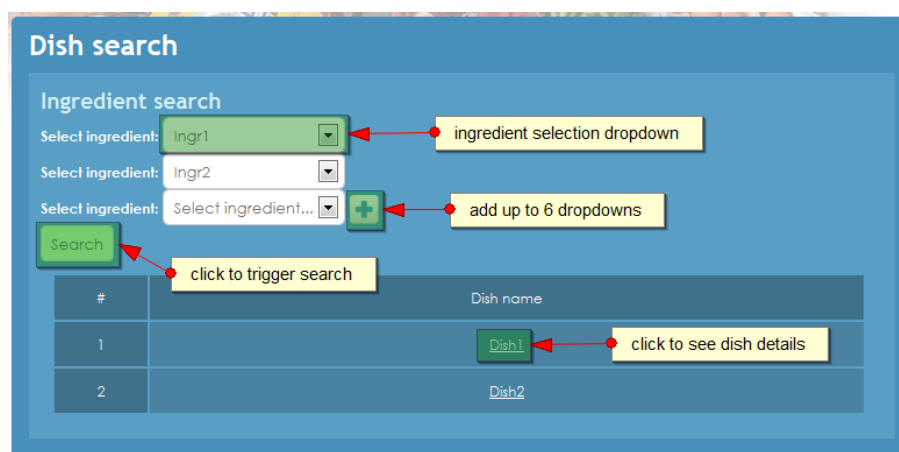
5.2.1.1 ACCESSIBLE APPLICATIONS

Guest can access *Ingredient search*, *Name search*, *Dishes catalogue* (from top navigation) and *Login*, *Register*, *Activate account*, *Reset password* (from left navigation), as well as *Show dish* and *Show dishes in category* applications.



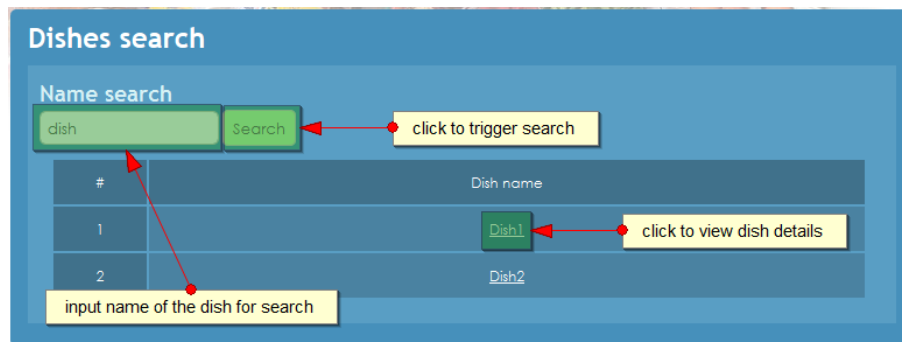
5.2.1.2 INGREDIENT SEARCH

Guest is able to search dishes by choosing the ingredients he already owns – dropdowns with ingredients are used for this. By clicking plus sign number of ingredients selections can be increased up to 6. When the search return the results, clicking the name of the dish redirects to the dish screen itself. Search by ingredients returns only those dishes that can be prepared by the guest from the ingredients he input.



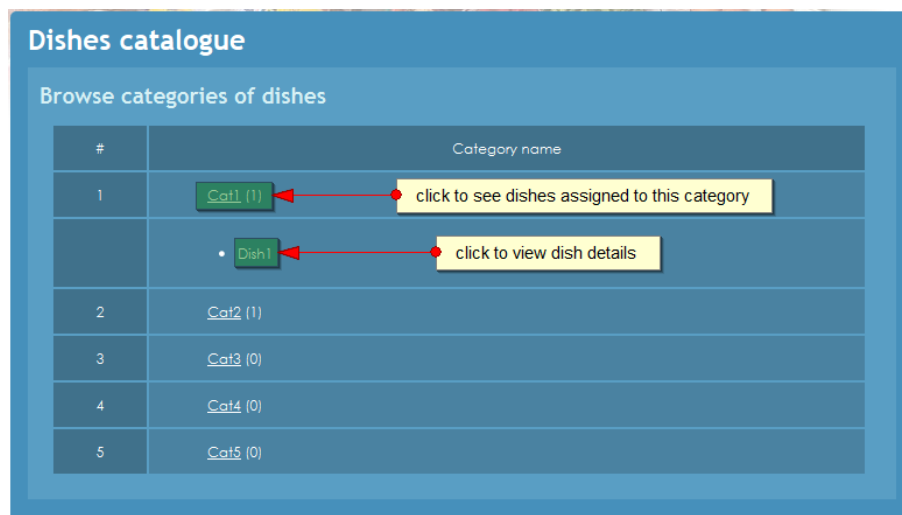
5.2.1.3 NAME SEARCH

Guest is able to search dishes by inputting the name of the dish he is looking for. This functionality checks if the string given by the guest can be matched by any of the names of dishes. When the search return the results, clicking the name of the dish redirects to the dish screen itself.



5.2.1.4 DISHES CATALOGUE

Guest is able to browse the catalogue of dishes per categories the dishes are assigned to. When the category is expanded, clicking the name of the dish redirects to the dish screen itself.



5.2.1.5 SHOW DISH

Guest can access details of the dish via *Show dish* screen. Dish screen consists of Dish details (clicking category link will show all dishes in this category, guest can only rate the dish after login, so Your rate cell shows the link to *Login* or *Register* screen), Photos (this section is shown only if photos for given dish are uploaded and approved by admin), Ingredients, Preparation plan and Comments (this section is shown only if comments for given dish are available).

Dish1

click to view all dishes in this category


login or register to rate this dish

Dish details

Dish category	Cat1		
Addition date	2011-12-18 13:16	Author	mszonline
Amount of calories	1000	Preparation time (min)	45
Difficulty level	Medium	How many people?	3
Average rate	0.0	Your rate	Login / Register

shown only if there are photos of the dish

Photos



Added by:

mszonline

Addition date:

2011-12-18 16:03

Ingredients

Ingr1	1	szt
Ingr2	1	dkg

shown only if there are comments to the dish

Preparation plan

Lorem ipsum

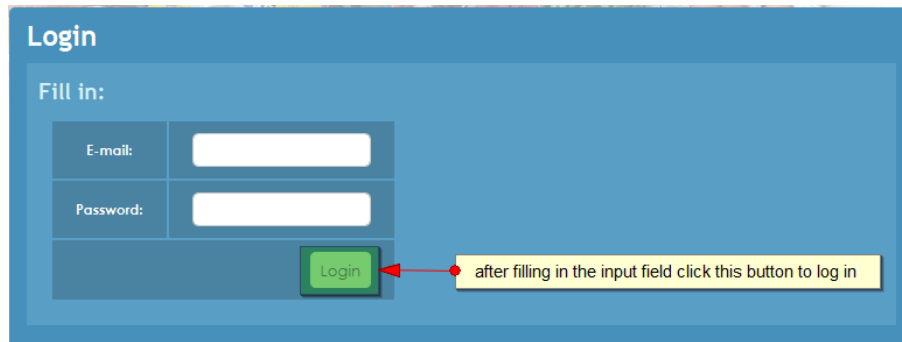
Comments

mszonline @ 2011-12-18 16:03

This is 1st comment.

5.2.1.6 LOGIN

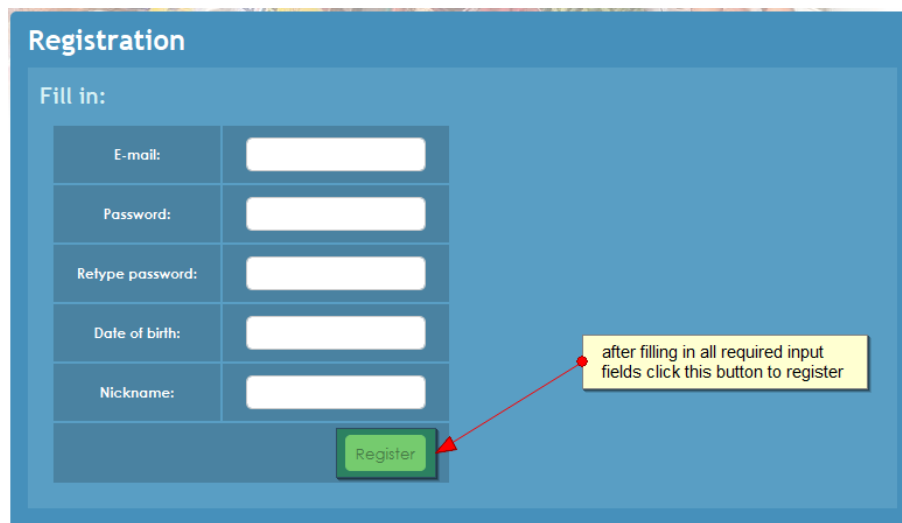
Guest can login to become the user. For this he should click *Login* link from the left menu, fill in required field and click *Login* button.



The image shows a 'Login' form with a blue header. Below the header, it says 'Fill in:'. There are two input fields: 'E-mail:' and 'Password:'. Below these fields is a green 'Login' button. A red arrow points from a yellow callout box to the 'Login' button. The callout box contains the text: 'after filling in the input field click this button to log in'.

5.2.1.7 REGISTER

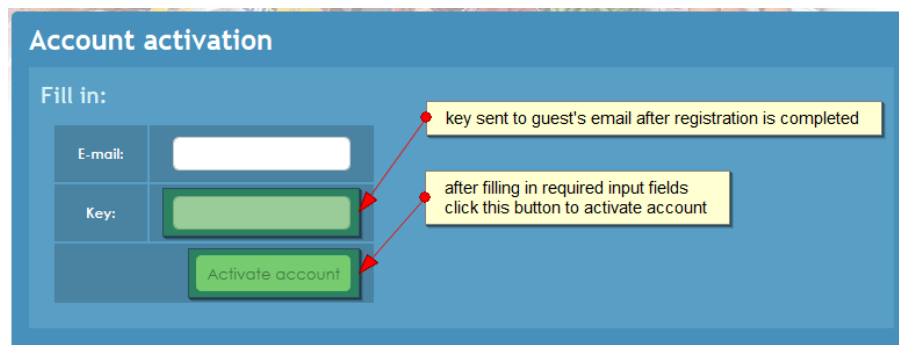
Before logging in guest needs to create his own profile by registering. This requires filling in fields shown below (password has to contain capital letter, small letter, number and one of the special characters \$!@#?):



The image shows a 'Registration' form with a blue header. Below the header, it says 'Fill in:'. There are five input fields: 'E-mail:', 'Password:', 'Retype password:', 'Date of birth:', and 'Nickname:'. Below these fields is a green 'Register' button. A red arrow points from a yellow callout box to the 'Register' button. The callout box contains the text: 'after filling in all required input fields click this button to register'.

5.2.1.8 ACTIVATE ACCOUNT

After registration, guest needs to activate his account to be able to login. Key field has to be filled in with the key sent to guest's e-mail address:

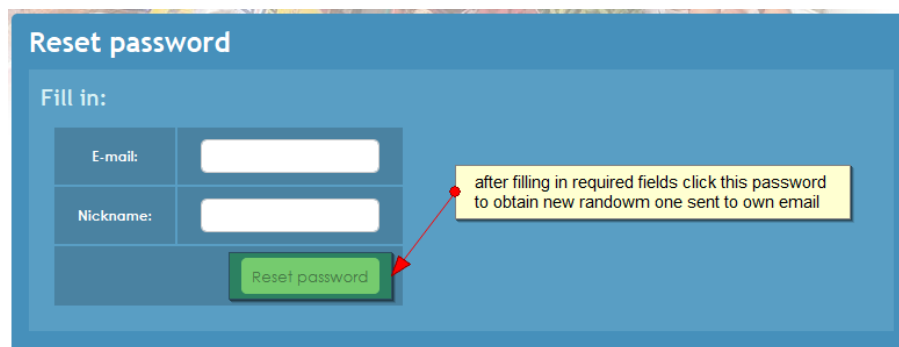


The form is titled "Account activation" and is set against a blue background. It contains a "Fill in:" section with two input fields: "E-mail:" and "Key:". Below these fields is a green "Activate account" button. Two yellow callout boxes with red arrows provide instructions: the first points to the "Key:" field and states "key sent to guest's email after registration is completed"; the second points to the "Activate account" button and states "after filling in required input fields click this button to activate account".

Account activation	
Fill in:	
E-mail:	<input type="text"/>
Key:	<input type="text"/>
<input type="button" value="Activate account"/>	

5.2.1.9 RESET PASSWORD

Guest is able to reset his password – new random one will be sent to his email.



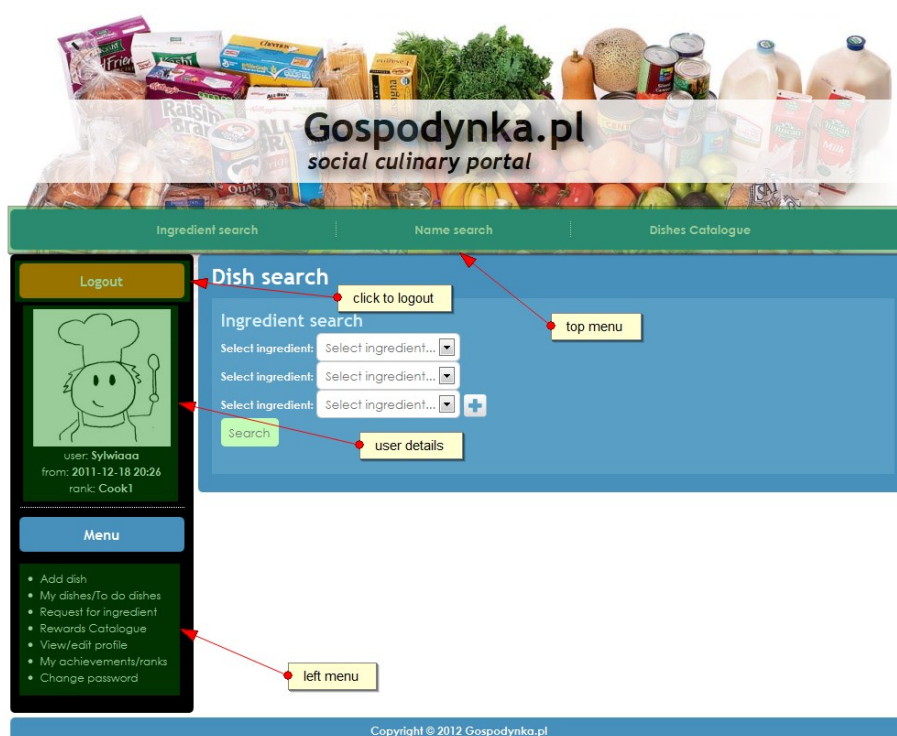
The form is titled "Reset password" and is set against a blue background. It contains a "Fill in:" section with two input fields: "E-mail:" and "Nickname:". Below these fields is a green "Reset password" button. A yellow callout box with a red arrow points to the button and states "after filling in required fields click this password to obtain new random one sent to own email".

Reset password	
Fill in:	
E-mail:	<input type="text"/>
Nickname:	<input type="text"/>
<input type="button" value="Reset password"/>	

5.2.2 USER MANUAL

5.2.2.1 ACCESSIBLE APPLICATIONS

User can access *Ingredient search*, *Name search*, *Dishes catalogue* (from top navigation – same as Guest) and *Add dish*, *My dishes/To do dishes*, *Request for ingredients*, *Reward catalogue*, *View/edit profile*, *My achievements/ranks* and *Change password* (from left navigation), as well as *Show dish*, *Show dishes in category* and *Show user applications*. Left sidebar consists of *Logout* link, logged in user details (photo of the user if uploaded and approved by admin, if not – photo of the current rank assigned, user name, registration date and rank) and left menu.



5.2.2.2 ADD DISH

User is able to add dish by inputting its category, name and preparation plan. At least one ingredient has to be assigned to the dish along with its amount. By default 5 dropdowns with ingredients selection are shown – this can be extended to 10 by clicking *More ingredients* link.

The screenshot shows a web form titled "Add dish" with a blue header. Below the header, a yellow box with the text "these filed are required" has a red arrow pointing to the "Select category..." dropdown. The form is divided into two main sections: "Fill in:" and a table for ingredients. The "Fill in:" section includes fields for "Name:", "Preparation plan:" (a large text area), "Amount of calories:", "Preparation time:", "Difficulty level:" (with radio buttons for Easy, Medium, and Difficult), and "How many people:". The table for ingredients has 5 rows, each with a "Select ingredient and input amount:" label and a dropdown menu followed by a text input field. A yellow box with the text "dish has to have at least one ingredient assigned along its amount" has a red arrow pointing to the first ingredient dropdown. At the bottom of the table, there is a "More ingredients" link. Below the table, there are two yellow boxes: one with the text "click to increase the number of optional ingredients from 5 to 10" pointing to the "More ingredients" link, and another with the text "click to add dish" pointing to the "Add dish" button.

these filed are required

Fill in:

Select category: Select category...

Name:

Preparation plan:

Amount of calories:

Preparation time:

Difficulty level: Easy Medium Difficult

How many people:

Select ingredient and input amount: Select ingredient...

Select ingredient and input amount: Select ingredient...

Select ingredient and input amount: Select ingredient...

Select ingredient and input amount: Select ingredient...

Select ingredient and input amount: Select ingredient...

dish has to have at least one ingredient assigned along its amount

More ingredients

click to increase the number of optional ingredients from 5 to 10

click to add dish

Add dish

5.2.2.3 MY DISHES/TO DO DISHES

User can view all dishes that he added by himself or dishes that he put on to-do list. Furthermore for these dishes he can upload photos (after sliding down photo upload section) that have to be approved by admin. Clicking dish name will result in viewing dish details, whereas clicking category name will show all dishes in this category.

The screenshot shows a web interface titled "Show my dishes". It contains two main sections: "My dishes" and "To-do dishes".

My dishes section:

- Annotations: "click to view dish details" points to the "Dish3" link in the "Dish name" column; "click to view dishes in this category" points to the "Cat1" link in the "Category" column.
- Table:

#	Dish name	Category	Action
1	Dish3	Cat1	Upload photo

Below the table is a file upload section:

Choose file from disc: [Browse...](#)

[Upload file](#)

Annotations: "click to upload file chosen from disc" points to the "Upload file" button; "browse local system for photo" points to the "Browse..." button.

To-do dishes section:

- Annotation: "click to slide down upload section for given dish" points to the "Upload photo" link in the "Action" column.
- Table:

#	Dish name	Category	Action
1	Dish1	Cat1	Upload photo

5.2.2.4 REQUEST FOR INGREDIENT

In case when user wants to add some dish but there is no ingredient that he requires, he can suggest new ingredient to the admin by filling in the form below and clicking the *Send request* link.

The screenshot shows a web interface titled "Request for ingredient". It contains a form with two input fields and a submit button.

Request for ingredient form:

Name:

Unit:

[Send request](#)

Annotation: "click to send request for the ingredient" points to the "Send request" button.

5.2.2.5 REWARD CATALOGUE

User can spend points that he earned by adding dishes or uploading photos on rewards. In order to do so, he has to fill in the form below, selecting the reward and sharing the delivery address. He is also able to view order history by clicking the plus sign in lower section (plus sign will change to minus sign – clicking it will hide order history).

The screenshot shows a web interface for a 'Rewards catalogue'. It is divided into two main sections: 'Order reward' and 'Orders history'.

Order reward section:

- Available points:** 10
- Select reward:** A list of two rewards: 'Reward1 description. - 40 punktów' and 'Reward2 description. - 200 punktów'. A red arrow points from the 'select reward' annotation to this list.
- mailing address::** A large green rectangular input field. A red arrow points from the 'share delivery address' annotation to this field.
- Order button:** A green button labeled 'Order'. A red arrow points from the 'click to order the reward' annotation to this button.

Orders history section:

- hide/unhide order history:** A green button with a minus sign icon. A red arrow points from the 'hide/unhide order history' annotation to this button.
- Table:** A table with 5 columns: '#', 'Name', 'Delivery address', 'Order date', and 'Cost'. It contains one row of data.

#	Name	Delivery address	Order date	Cost
1	Reward1	Lorem ipsum...	2011-12-18 23:46	40

5.2.2.6 VIEW/EDIT PROFILE

User is able to view his profile details, as well as upload photo of himself (needs to be approved by admin).

Profile

Upload profile photo

click to slide down upload profile photo section



My profile

Profile photo:	
Nickname	mszonline
E-mail	mszonline@gmail.com
Registration date:	2011-12-11
Date of birth:	1984-04-27
Experience points:	50
Experience points used for redemption:	40
Rank:	Cook1
Country:	
Webpage:	
Gender:	
<div><div>Edit profile</div><div>click to edit own profile</div></div>	

He can also edit some of the information about him (by clicking *Edit profile* link in the bottom of the screen).

Edit profile

Fill in:

E-mail:	<input type="text" value="mszonline@gmail.com"/>
Nickname:	<input type="text" value="mszonline"/>
Date of birth:	<input type="text" value="1984-04-27"/>
Country:	<input type="text"/>
Webpage:	<input type="text"/>
Gender:	<input checked="" type="radio"/> Female <input type="radio"/> Male

click to save chagnes to own profile

5.2.2.7 MY ACHIEVEMENTS/RANKS

User is able to view his achievements and ranks. Achievements are: adding dish, obtaining approval for profile photo (once only), obtaining approval for dish photo. Each achievement has number of points assigned, what means that obtaining a given achievement is equal to obtaining defined number of experience points. Obtaining a certain amount of points results in obtaining a rank (in the hierarchy of cooks). These ranks can be seen in lower section.

My achievements and ranks				
My achievements				
#	Name	Description	Assignment date	Points received
1	Dodanie potrawy/Adding dish	Dodanie potrawy jest warte 10 punktów doświadczenia.	2011-12-18 23:46	10
2	Dodanie potrawy/Adding dish	Dodanie potrawy jest warte 10 punktów doświadczenia.	2011-12-18 23:45	10
3	Dodanie zdjęcia potrawy/Adding dish photo	Dodanie zdjęcia potrawy jest warte 5 punktów doświadczenia. Naliczenie punktów odbywa się po zaakceptowaniu zdjęcia przez administratora.	2011-12-18 16:12	5
4	Dodanie zdjęcia potrawy/Adding dish photo	Dodanie zdjęcia potrawy jest warte 5 punktów doświadczenia. Naliczenie punktów odbywa się po zaakceptowaniu zdjęcia przez administratora.	2011-12-18 16:03	5
5	Dodanie potrawy/Adding dish	Dodanie potrawy jest warte 10 punktów doświadczenia.	2011-12-18 13:29	10
6	Dodanie potrawy/Adding dish	Dodanie potrawy jest warte 10 punktów doświadczenia.	2011-12-18 13:29	10
My ranks				
#	Name	Description	Assignment date	Points needed
1	Cook2	Cook2 description.	2011-12-18 23:46	50
2	Cook1	Cook1 description.	2011-12-18 13:29	0

5.2.2.8 CHANGE PASSWORD

User can change his password by typing new password twice and submitting the form.

The image shows a 'Password change' form with a blue header. Below the header, the text 'Fill in:' is followed by two input fields. The first field is labeled 'New password:' and the second is labeled 'Retype new password:'. Below these fields is a green 'Submit' button. A yellow callout box with the text 'click to set new password' has a red arrow pointing to the 'Submit' button.

Password change	
Fill in:	
New password:	<input type="text"/>
Retype new password:	<input type="text"/>
<input type="submit" value="Submit"/>	

5.2.2.9 SHOW DISH

User can view dish details same as the guest, but he can also add comments (lowest section), rate the dish, add the dish to to-do list, see dish author details as well as details of the user that added the photo.

Dish5

click to view all dishes in this category → Cat3

click to see user details → mszonline

Dish details

Dish category	Cat3		
Addition date	2011-12-18 23:46	Author	mszonline
Amount of calories	77	Preparation time (min)	77
Difficulty level	Difficult	How many people?	77
Average rate	0.0	Your rate	1 2 3 4 5
To do dish?	Add dish to to-do list		

click to add dish to to-do list → Add dish to to-do list

click to rate the dish → 1 2 3 4 5

Zdjęcia

click to see user details → mszonline

Added by: mszonline

Addition date: 2011-12-19 00:14

Ingredients

section shown only if photos are approved by admin for the dish

Ingr2	55	dkg
Ingr3	44	dkg
Ingr4	55	szt

Preparation plan

section shown only if there are comments added to the dish

Lorem ipsum ipsum

Comments

Sylwiaaa @ 2011-12-19 00:13 Lorem ipsum

Add comment

click to send comment → Send comment


5.2.2.10 SHOW USER

User can view other users' profiles, as well as their dishes and the list of dishes to be done.

Sylwiaaa


User details

Photo




Registration date	2011-12-18 20:26	Privilege level	User
Experience points	10	Rank	Cook1
Experience points used for redemption	0	Gender	
Country		Webpage	

User dishes

click to hide user's dishes 

#	Dish name	Category
1	Dish3	Cat1

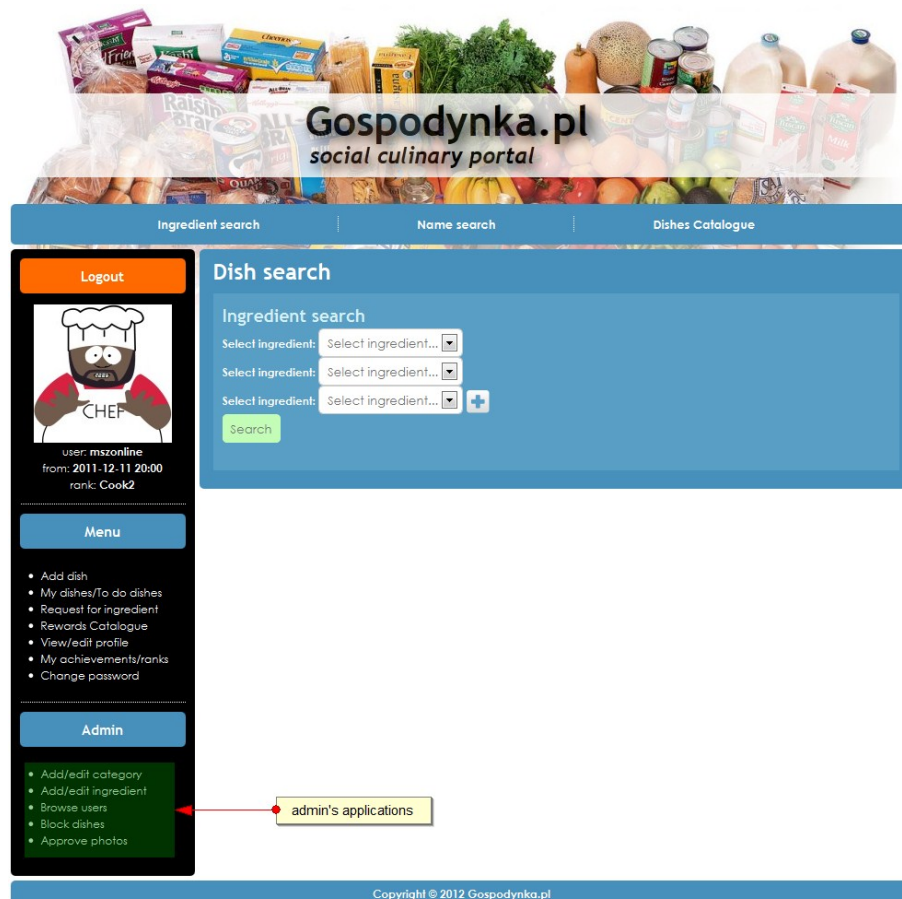
User to-do dishes

click to show user's to-do list 

5.2.3 ADMIN MANUAL

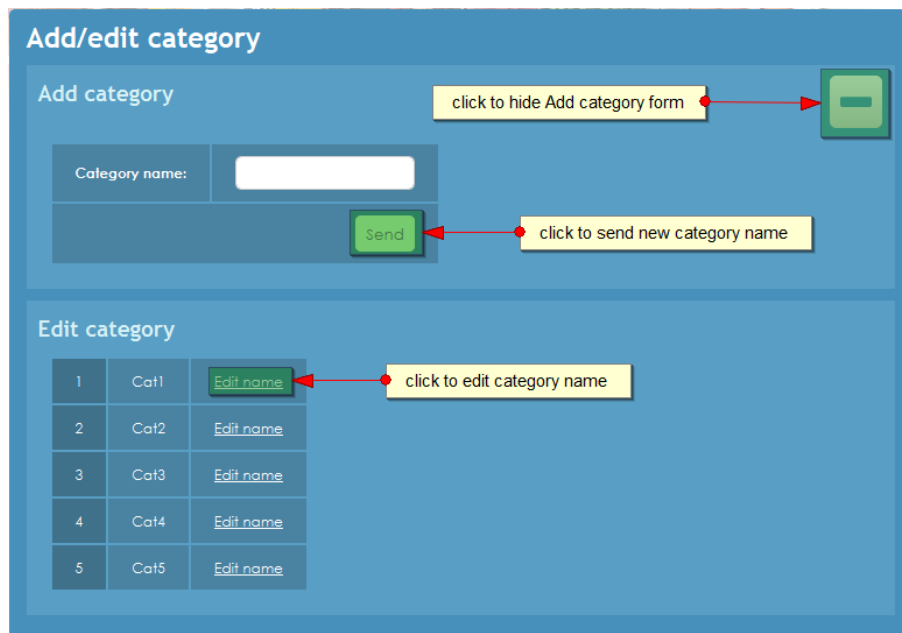
5.2.3.1 ACCESSIBLE APPLICATIONS

Admin can access same applications as user. Additional ones are accessible through left menu subtitled *Admin*. These applications are: *Add/edit category*, *Add/edit ingredient*, *Browse users*, *Block dishes*, *Approve photos*.



5.2.3.2 ADD/EDIT CATEGORY

Admin can add new category or edit existing one. Clicking plus sign will result is sliding down *Add category* form.



Add/edit category

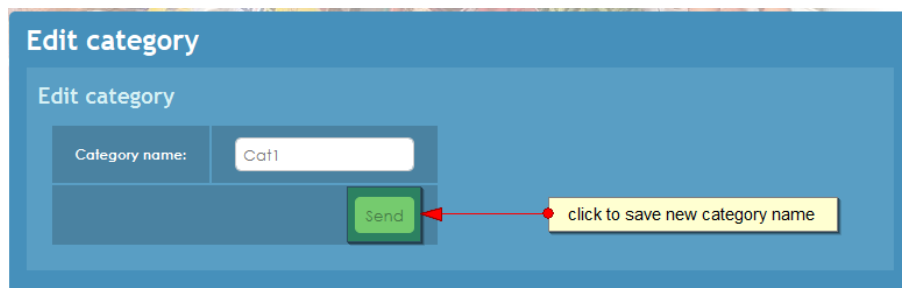
Add category

Category name:

Edit category

1	Cat1	Edit name
2	Cat2	Edit name
3	Cat3	Edit name
4	Cat4	Edit name
5	Cat5	Edit name

Clicking *Edit name* link will result is showing *Edit category* form.



Edit category

Edit category

Category name:

5.2.3.3 ADD/EDIT INGREDIENT

Admin can add new ingredient or edit existing one. Clicking plus sign will result is sliding down *Add ingredient* form.

Add/edit ingredient

Add ingredient

click to hide Add ingredient form

Ingredient name:

Ingredient unit:

Send

click to add new ingredient

Edit ingredient

1	Ingr1	szť	Edit name/unit
2	Ingr2	dkg	Edit name/unit
3	Ingr3	dkg	Edit name/unit
4	Ingr4	szť	Edit name/unit
5	Ingr5	szczypta	Edit name/unit
6	Ingr6	dkg	Edit name/unit

click to edit ingredient's name and unit

Clicking *Edit name/unit* link will result is showing *Edit ingredient* form.

Edit ingredient

Edit ingredient

Ingredient name:

Ingredient unit:

Send

click to save changes to the ingredient name and unit

5.2.3.4 BROWSE USERS

Admin can browse users and edit their profiles. He is also able to block/unblock users.

Users

view more details of the user

List of users

#	Nickname	E-mail	Registration name	Priv.	Actions
1	mszonline	mszonline@gmail.com	2011-12-11	admin	Block Edit
<div>E-mail: <input type="text" value="mszonline@gmail.com"/></div> <div>Nickname: <input type="text" value="mszonline"/></div> <div>Date of birth: <input type="text" value="1984-04-27"/></div> <div>Country: <input type="text"/></div> <div>Webpage: <input type="text"/></div> <div>Gender: <input type="radio"/> Female <input type="radio"/> Male</div> <div>Privilege level: <input checked="" type="radio"/> User <input type="radio"/> Admin</div> <div>Save changes</div>					
2	Sylwiaaa	jurkiewicz.sylwia@gmail.com	2011-12-18	user	Block Edit

click to block user

click to save changes performed on user's profile

click to slide down user edit pane

5.2.3.5 BLOCK DISHES

Admin can also block/unblock dishes.

Block dishes

click to view user's profile

click to block the dish

List of dishes

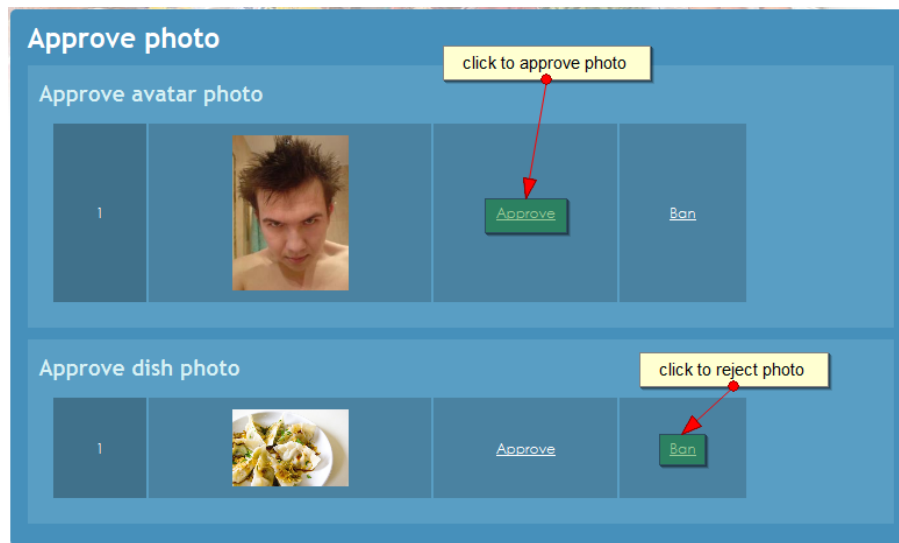
#	Dish name	Author	Action
1	Dish1	mszonline	Block
2	Dish2	mszonline	Block
3	Dish4	mszonline	Unblock
4	Dish5	mszonline	Block
5	Dish3	Sylwiaaa	Block

click to view dish details

click to unblock the dish

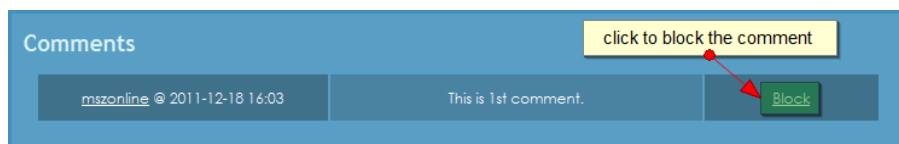
5.2.3.6 APPROVE PHOTOS

Admin can approve or reject photos of avatars and dishes.



5.2.3.7 SHOW DISH

This application is the same as for regular user, but admin is able to block the comments.



6 REFERENCES

6.1 WEB RESOURCES

- [1] Advantages of Java
<http://www.webdotdev.com/nvd/content/view/1042/204/>
- [2] CSS
<http://en.wikipedia.org/wiki/Css>
- [3] FireShot Firefox Add-on
<http://screenshot-program.com/fireshot/>
- [4] Google Chrome Developer Tools
<http://code.google.com/intl/pl-PL/chrome/devtools/>
- [5] Is Apache Tomcat 7 in your future?
<http://www.tomcatexpert.com/blog/2010/07/29/tomcat-7-your-future>
- [6] JavaScript
<http://en.wikipedia.org/wiki/JavaScript>
- [7] Jettison
<http://jettison.codehaus.org>
- [8] JQuery
<http://jquery.com>
- [9] JQueryUI
<http://jqueryui.com/>
- [10] MySQL
<http://dev.mysql.com/doc/refman/5.5/en/index.html>
<http://en.wikipedia.org/wiki/MySQL>
- [11] Sanity testing
http://en.wikipedia.org/wiki/Sanity_testing
- [12] Unobtrusive JavaScript
http://en.wikipedia.org/wiki/Unobtrusive_JavaScript
- [13] XHTML
<http://en.wikipedia.org/wiki/Xhtml>
- [14] XStream
<http://xstream.codehaus.org>

6.2 LITERATURE

- [15] Basham Bryan, Sierra Kathy, Bates Bert, Head First Servlets & JSP, Second Edition. O'Reilly Media, 2008.
- [16] Brown Don, Davis Chad Michael, Stanlick Scott, Struts 2 in Action. Manning Publications, 2008.
- [17] Kaner Cem, Falk Jack, Nguyen Hung Quoc, Testing Computer Software, Second Edition. Wiley, 1999.
- [18] Myers Glenford J., Badgett Tom, Thomas Todd M., Sandler Corey, The Art of Software Testing, Second Edition. Wiley, 2004.
- [19] Roughley Ian, Practical Apache Struts2 Web 2.0 Projects. Apress, 2007