**-- URL query parameters and JSON responses --**

**Conventions**

In the following examples, user-supplied input is denoted by <angle brackets>.

Optional input is denoted by [square brackets], and optionally repeated input

is denoted by [ellipses...].

As is standard in URL queries, the beginning of a parameter list is denoted

by a question mark (?), and individual query parameters are separated by

ampersands (&). Parameters may be in the form of simple <attributes>, or

may be in the form of <attribute>=<value> pairs. Multi-valued attributes are

specified as repeated attributes followed by square brackets ([]), ie.

<attribute>[]=<value1>&<attribute>[]=<value2>

For example, in the 'search cookbooks by attribute' URL

http://www.recipeserver.com/cookbooks/list?<attribute>=<value>[&<attribute>=<value>...]

the URL of the script itself is given by <http://www.recipeserver.com/cookbooks/list>, the beginning of the parameter list is denoted by a question mark (?), the first <attribute> and <value> are user-supplied and necessary, and as many repeated <attribute>=<value> pairs as desired may be supplied afterwards, separated by ampersands (&).

The temporary server URL http://www.recipeserver.com/ is used until we can come up with something nicer ;)

**HTTP requests**

Responses shall be obtained from the server via HTTP requests to the specified URLs with the specified parameters.

Most requests shall be made via an HTTP GET request with a header in the following format:

GET /<path>/<script> HTTP/1.1

host: http://www.recipeserver.com/

Creation requests for cookbooks and recipes shall be made via a multipart HTTP POST request with a header in the following format (see <http://www.w3.org/Protocols/rfc1341/7_2_Multipart.html> for details on the multipart/form-data content-type)::

POST /<path>/create HTTP/1.1

content-type: multipart/form-data; boundary=<boundary>

host: <http://www.recipeserver.com/>

content-length: <parameter length>

--<boundary>

content-disposition: form-data; name=”<parameter>”

[content-type: application/octet-stream]

<value>

[--<boundary>

…

…

…]

--<boundary>--

Note particularly that each parameter is delimited by two dashes and the boundary included in the header (--<boundary>) and the final parameter is ended with two dashes, the boundary and two more dashes (--<boundary>--). Parameters default to ASCII text if no content-type is specified, which shall be the format for all data except for the image data, which shall include the header line content-type: application/octet-stream.

Since the creation requests require potentially long data such as descriptions and images to be sent, it's not possible to use a GET request with query parameters (many browsers are limited to slightly over 2000 characters in a URL).

Additional lines may be specified in the header if it simplifies the implementation. However, the lines mentioned above must be present in all requests.

**Scripts**

*cookbooks/*

Cookbook-related scripts will be accessed via the cookbooks/ subdirectory on the server. The scripts contained in cookbooks/ will be:

*cookbooks/search*  Search the database for cookbooks, with optional criteria to specify certain attributes. cookbooks/search will also allow the optional parameter show\_only=<number> and sort\_by=<attribute\_name> to allow sorting of the output and restriction of requests on the database. sort\_by will by default return attributes sorted in descending order - to specify ascending order instead, specify the attribute a -<attribute\_name> (ie. with a leading negative sign). Finally, the rating parameter shall specify a minimum rating for returned recipes, and the rating\_max parameter shall specify a maximum.

REQUIRED PARAMETERS: none

OPTIONAL PARAMETERS:

name

author\_name

rating

rating\_max

show\_only

sort\_by

dietary\_restriction

*cookbooks/create*  Create a new cookbook with specified criteria in POST request.

REQUIRED PARAMETERS (in POST request):

name

author\_name

description

rating

OPTIONAL PARAMETERS:

dietary\_restriction

image

*cookbooks/add* Add a recipe to a cookbook.

REQUIRED PARAMETERS:

name

author\_name

description

OPTIONAL PARAMETERS:

dietary restriction

*cookbooks/remove*  Remove a recipe from a cookbook.

REQUIRED PARAMETERS:

name

author\_name

OPTIONAL PARAMETERS: none

*cookbooks/show-recipes* List all of the recipes in a specified cookbook.

REQUIRED PARAMETERS:

name

author\_name

OPTIONAL PARAMETERS: none

*cookbooks/rate* Rate a specified cookbook.

REQUIRED PARAMETERS:

username

author\_name

cookbook\_name

rating

OPTIONAL PARAMETERS: none

*recipes/*

Recipe-related scripts will be accessed via the recipes/ subdirectory on the server. The scripts contained in recipes/ will be:

*recipes/search* Search the database for recipes, with optional criteria to specify certain attributes. recipes/search will also support the show\_only=<number> and sort\_by=<attribute> options (see cookbooks/search above). In addition, it will support the options using=<ingredient> and using\_only=<ingredient>. Both may be specified multiple times, and the final list will be used to filter recipes down to those which use (or only use) the specified ingredients. The rating and rating\_max options shall be identical to cookbooks/search, and the prep\_time and prep\_time\_max options shall support the same minimum/maximum behaviour.

REQUIRED PARAMETERS: none

OPTIONAL PARAMETERS:

name

author\_name

instructions

prep\_time

prep\_time\_max

rating

rating\_max

description

show\_only

sort\_by

using

using\_only

dietary\_restriction

*recipes/create*  Create a recipe with the specified parameters.

REQUIRED PARAMETERS (in POST request):

name

author\_name

instructions

prep\_time

portions

description

OPTIONAL PARAMETERS:

dietary\_restriction

image

*recipes/delete*  Remove a recipe from the database and all associated cookbooks.

REQUIRED PARAMETERS:

recipe\_name

author\_name

OPTIONAL PARAMETERS: none

*recipes/rate*  Rate a specified recipe.

REQUIRED PARAMETERS:

username

recipe\_name

author\_name

rating

OPTIONAL PARAMETERS: none

**URL encoding**

All parameters in GET requests must be encoded to escape all illegal URL entities therein, to avoid confusing URL and request parsing. This encoding shall be performed by the standard libraries for the language, such as HttpServerUtility.UrlEncode() in C#, or urlencode() in PHP.

**JSON encoding**

All results returned by the server will be in JSON format. For requests which return rows from a table, each row will be returned as a separate JSON object (collection of key-value pairs). JSON objects are represented by braces ({/}) surrounding lists of comma-separated "name":"value" pairs. Elements shall be returned in the order they appear in the database.

ex:

{"user":"lmitchell","password\_hash":"12345678"}

For requests which only require a success/failure response (eg. creating a recipe, rating a recipe), the server shall respond with the JSON object

{"result":"<success/failure>","error":"<error message>"}

In the successful case, the "error" field's value will contain an empy string.

JSON objects shall be escaped as per the escaping rules followed by PHP's json\_encode() function, and shall be decoded into PHP associative arrays via the json\_decode() function, or C# objects via Json.Decode().

Examples (note: key/value pairs are separated by newlines for clarity - however, the JSON returned by the server will NOT contain any additional characters beyond the JSON objects themselves):

Requests to recipes/search and cookbooks/show-recipes shall return JSON similar to the following:

{

"name":"mac and cheese",

"author\_name":"nloison",

"instructions":"melt cheese and cook noodles",

"picture":"pics\/mac.png",

"prep\_time":"45.0",

"portions":"4",

"rating":null,

"description":"melted cheese and cooked macaronis"

}

{

"name":"shepherd's pie",

"author\_name":"lmitchell",

"instructions":"cook it for like an hour or whatever",

"picture":"pics\/pie.png",

"prep\_time":"55.0",

"portions":"8",

"rating":null,

"description":"potatoes and meat and some corn and stuff!"

}

{

"name":"steak dinner",

"author\_name":"adjuric",

"instructions":"grill em!",

"picture":"pics\/steak.png",

"prep\_time":"25.0",

"portions":"4","rating":null,

"description":"grilled meat!"

}

Requests to cookbooks/create, cookbooks/delete, cookbooks/add, cookbooks/remove, cookbooks/rate, recipes/create, recipes/delete and recipes/rate shall return a success/failure JSON object:

{“result”:”success”,”error”:””}

{“result”:”failure”,”error”:”recipe does not exist”}

**Use cases and example HTTP requests / URL parameters**

**USE CASE:** User login

**URL:**

**USE CASE:** Show all cookbooks

**URL:** <http://www.recipeserver.com/cookbooks/search>

**USE CASE:** Search cookbooks by attribute

**URL:** http://www.recipeserver.com/cookbooks/search?<attribute>=<value>[&<attribute>=<value>...]

**USE CASE:** Show all of a user's cookbooks

**URL:** http://www.recipeserver.com/cookbooks/search?author\_name=<username>

**USE CASE:** Show top 10 cookbooks by rating

**URL**: http://www.recipeserver.com/cookbooks/seach?show\_only=10&sort\_by

**USE CASE:** Create cookbook

**URL:** http://www.recipeserver.com/cookbooks/create

**POST REQUEST:**

POST /cookbooks/create HTTP/1.1

content-type: application/x-www-form-urlencoded

host: http://www.recipeserver.com/

content-length: <parameter length>

author\_name=<username>

&cookbook\_name=<cookbook\_name>

&cookbook\_description=<description>

**USE CASE**: Add recipe to cookbook

**URL:** http://www.recipeserver.com/cookbooks/add?cookbook\_name=<cookbook\_name>&cookbook\_author=<cookbook\_author\_name>&recipe\_name=<recipe\_name>&recipe\_author=<recipe\_author\_name>

**USE CASE:** Remove recipe from cookbook

**URL:** http://www.recipeserver.com/cookbooks/remove?cookbook\_name=<cookbook\_name>&cookbook\_author=<cookbook\_author\_name>&recipe\_name=<recipe\_name>&recipe\_author=<recipe\_author\_name>

**USE CASE:** Show all recipes in a cookbook

**URL:** http://www.recipeserver.com/cookbooks/show-recipes?name=<name>&author=<username>

**USE CASE:** Rate a cookbook

**URL:** http://www.recipeserver.com/cookbooks/rate?username=<username>&author\_name=<author\_name>&cookbook\_name=<cookbook\_name>&rating=<0..5>

**USE CASE:** Show all recipes

**URL:** http://www.recipeserver.com/recipes/search

**USE CASE:** Show all of a user's recipes

**URL:** http://www.recipeserver.com/recipes/search?author\_name=<username>

**USE CASE:** Search recipes by attribute

**URL:** http://www.recipeserver.com/recipes/search?<attribute>=<value>[&<attribute>=<value>...]

**USE CASE:** Show top rated recipes

**URL:** http://www.recipeserver.com/recipes/search?sort\_by=rating

**USE CASE:** Show recipes using certain ingredients

**URL:** http://www.recipeserver.com/recipes/search?using=<ingredient>[&using=<ingredient>...]

**USE CASE:** Show recipes using ONLY certain ingredients

**URL:** http://www.recipeserver.com/recipes/search?using\_only=<ingredient>[&using\_only=<ingredient>...]

**USE CASE:** Create recipe

**URL:** http://www.recipeserver.com/recipes/create

**POST REQUEST:**

POST /recipes/create HTTP/1.1

content-type: application/x-www-form-urlencoded

host: http://www.recipeserver.com/

content-length: <parameter length>

recipe\_name=<recipe\_name>

&author\_name=<author\_name>

&instructions=<instructions>

&picture=</path/to/picture>

&prep\_time=<prep\_time>

&portions=<portions>

&description=<description>

**USE CASE:** Rate recipe

**URL:** http://www.recipeserver.com/recipes/rate?username=<username>&recipe\_name=<recipe\_name>&author\_name=<author\_name>&rating=<0..5>