# **Software Requirements Specification**

# **for**

# **MemCache project**

# 

# **Prepared by**

# **Team Name: *Rapid Slowpokes***

# 

# **Rasul Tumyrkin (rasul.tumyrkin@gmail.com)**

# **Larisa Safina (safinalrr@gmail.com)**

# **Dinara Nikolaeva (na13dinara@mail.ru)**

# 

# **University: Innopolis University**

# **Country: Russia**

# 

# 

# 

# **28th February 2015**

# **Table of contents**

[1 Introduction](#h.vo7chqf5ttgr)

[1.1 Purpose](#h.hy930417egy3)

[1.2 Scope](#h.iyyrdladhh74)

[1.3 Definitions, Acronyms and Abbreviations](#h.l8zt6yucoa8v)

[1.4 References](#h.hfprqtabi3lv)

[1.5 Overview](#h.8r1xmbu8tec3)

[2 Overall Description](#h.twh5ctfj3oar)

[2.1 Product perspective](#h.g1d5v6ll2wh1)

[2.2 Product functions](#h.i62gje5gvs7)

[2.2.1 Product supported functions](#h.6qmrcd4pijwm)

[2.2.1 Product unsupported functions](#h.wps3rbicdm1k)

[2.3 User characteristics](#h.59d1uusu2hes)

[2.4 Constraints](#h.6u0zj74xy7b)

[2.5 Assumptions and dependencies](#h.asa8qo5zfx99)

[3 Specific Requirements](#h.6pungq6owjow)

[3.1 Functionality](#h.d989h7voj5n2)

[3.1.1 Functional requirement one (FR1)](#h.gjvyxo5tbo1t)

[3.1.4 Functional requirement one (FR4)](#h.65le6gunbnfj)

[3.1.5 Functional requirement one (FR5)](#h.suh9x4o6xgzv)

[3.1.6 Functional requirement one (FR6)](#h.snbauwbusjal)

[3.1.7 Functional requirement one (FR7)](#h.sm5mirrf8zu)

[3.1.8 Functional requirement one (FR8)](#h.w7165qeo6c4r)

[3.1.9 Functional requirement one (FR9)](#h.f1zvfz43h1mf)

[3.2 Quality attributes support](#h.5449zvfstl3o)

[3.2.1 Reliability](#h.zbs6nw4yrk9b)

[3.2.2 Performance](#h.yhi94jiiwfvu)

[3.2.3 Portability](#h.4hdhv3iu0ylx)

[3.3 Design Constraints](#h.ryp0d4t3qtls)

[3.4 On-line User Documentation and Help System Requirements](#h.zz9ikmp9pr4)

[3.5 Purchased Components](#h.k0xbvwlt8in0)

[3.6 Interfaces](#h.fx57mduih9bh)

[3.6.1 User Interfaces](#h.5wmvuqitijf3)

[3.6.2 Hardware Interfaces](#h.4ooxtriidqsm)

[3.6.3 Software Interfaces](#h.9nigfun69umh)

[3.6.4 Communications Interfaces](#h.g0k15re91bs)

[3.7 Licensing Requirements](#h.z35fhtjqb3bx)

[3.8 Legal, Copyright, and Other Notices](#h.66xdyjvw1qo7)

[4 Supporting Information](#h.o2xnstkmagkc)

# **Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author(s)** |
| 03/02/2015 | 0.1 | Project’s purpose, scope and functions defining. | Larisa Safina |
| 03/05/2015 | 0.2 | Functional requirements(version 1), reliability, design constraints | Rasul Tumyrkin, Dinara Nikolaeva |
| 03/08/2015 | 0.3 | Functional requirements(version 2) | Rasul Tumyrkin, Dinara Nikolaeva |
| 03/10/2015 | 1.0 | Functional requirements, quality attributes and constraints have been added, fluff is removed | Larisa Safina |

# 

# 

# **1 Introduction**

## 1.1 Purpose

This document represents the Software Requirement Specification (SRS) for the MemCache project. It is aimed to cover the project’s scope functional requirements, quality attributes and constraints. It is addressed to any member of a developing team and to any potential stakeholder.

## 1.2 Scope

The scope of the MemCache project is an implementation of MemCache library. The main goal of the library is to provide the “key-value” based mechanism for caching data in memory. This SRS will describe only the functionality related to the MemCache library and not for the possible external systems that may use it. Memcached is an in-memory key-value store for small chunks of arbitrary data (strings, objects) from results of database calls, API calls, or page rendering.

## 1.3 Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Explanation** |
| SRS | Software Requirements Specification |
| Caching | Data storing so future requests for that data can be served faster |
| “Key-value” | Type of attribute-value data representation, representing by the collection of tuples <key, value> |
| MemCache | The memory caching library |
|  |  |

## 1.4 References

This project referenced to the following documents:

* 830-1998 - IEEE Recommended Practice for Software Requirements Specifications
* Software Architecture course project 2015

## 1.5 Overview

Section 2 defines the general product functions, intended application constraints to be respected and the assumption made in order to define requirements.

Section 3 specifies functional (Section 3.1) and non-functional requirements (all other sections), supported quality attributes (reliability, performance and portability considerations), and constraints in a level of detail sufficient to enable designers to design a system to satisfy these requirements and testers to test that the system satisfies these requirements.

Section 4 contains supporting information.

# 

# 

# **2 Overall Description**

## 2.1 Product perspective

The proposed product represents a library with the functions listed in the section 2.2. The solution will be written in Eiffel language.

## 2.2 Product functions

## 2.2.1 Product supported functions

MemCache library shall provide capability of

* creating a “key-value” relation for the data;
* storing data in memory;
* retrieving stored data from the memory
  + by the key,
  + by the key and parameter (if set as hash);
* deleting stored data from the memory
  + by the key,
  + all stored data;
* maintaining cache capacity (erasing less-frequently used data);
* searching for stored data by the value;
* parametrizing time of data storing
  + set range in minutes,
  + use predefined default value;
* allocating size of the cache storage;
* cache size and preemption strategy initialization;
* implementing preemption strategies
  + lifo,
  + fifo
  + less-frequently used;
* key existence checking;

## 2.2.1 Product unsupported functions

MemCache library shall not provide capability of:

* concurrent library using;
* any other interfaces implementation.

## 2.3 User characteristics

The library is intended to be used as a part of any other external applications, so there is no direct interaction with the human.

## 2.4 Constraints

The document represents a study project, not a real-life SRS, and misses detailed description and requirement for many areas. It gives only directions and requirement templates for creating a MemCache library.

However, there are the following mandatory requirements exist:

* Using of Eiffel programming language;
* All requirements have to be implemented until 28 of April.

## 2.5 Assumptions and dependencies

There is no explicit dependencies elicited at the moment.

# **3 Specific Requirements**

## 3.1 Functionality

## 3.1.1 Functional requirement one (FR1)

|  |  |
| --- | --- |
| **Tittle** | Storing data in the cache |
| **Priority** | Mandatory |
| **Risk** | Critical |
| **Description** | Memcache shall allow to put data provided by the user to the cache and store it there as <key,value> tuple until the end of program’s execution. |

**3.1.2 Functional requirement two (FR2)**

|  |  |
| --- | --- |
| **Tittle** | Getting data from the cache |
| **Priority** | Mandatory |
| **Risk** | Critical |
| **Description** | Memcache shall allow to retrieve the value inquired by the user by the corresponding key or provide the message that the key wasn’t found. |

**3.1.3 Functional requirement three (FR3)**

|  |  |
| --- | --- |
| **Tittle** | Deleting elements from the cache by user request |
| **Priority** | Mandatory |
| **Risk** | Critical |
| **Description** | 1. Memcache shall be able to delete the data from the memory by the corresponding key, entered by user, or provide a message that the key wasn’t found;  2. Memcache shall be able to delete all stored data by the user’s request. |

## 3.1.4 Functional requirement four (FR4)

|  |  |
| --- | --- |
| **Tittle** | Retrieving the data by the key and parameter |
| **Priority** | Mandatory |
| **Risk** | Major |
| **Description** | If memcache user inquire the data by the key and parameter, the value retrieved by means of memcache library have to correspond this parameter.  *For example, if we have set the data “cat{4: paws, 1:tail}”, method get(cat.paws) will retrieve 4.* |

## 3.1.5 Functional requirement five (FR5)

|  |  |
| --- | --- |
| **Tittle** | Deleting elements from the cache automatically |
| **Priority** | Mandatory |
| **Risk** | Critical |
| **Description** | 1.System should automatically remove elements after ending of program runtime;  2. When list is full and it is need to add new element System should remove elements from the cache by the default strategy (fifo) or by strategy determined by user:   * lifo, * fifo * less-frequently used; |

## 3.1.6 Functional requirement six (FR6)

|  |  |
| --- | --- |
| **Tittle** | Searching for stored data by the value |
| **Priority** | Mandatory |
| **Risk** | Critical |
| **Description** | System shall provide the ability to search the stored data by the value provided by the user. |

## 3.1.7 Functional requirement seven (FR7)

|  |  |
| --- | --- |
| **Tittle** | Parametrizing time of data storing |
| **Priority** | Mandatory |
| **Risk** | Critical |
| **Description** | Memcache shall provide the ability to parametrize time of data storing by setting range in minutes or using predefined default value (30 min); |

## 3.1.8 Functional requirement eight (FR8)

|  |  |
| --- | --- |
| **Tittle** | Key existence checking; |
| **Priority** | Mandatory |
| **Risk** | Critical |
| **Description** | System shall provide the ability to check if there is some data stored by the key provided by the user. |

## 3.1.9 Functional requirement nine (FR9)

|  |  |
| --- | --- |
| **Tittle** | Cache size and preemption strategy initialization |
| **Priority** | Mandatory |
| **Risk** | Critical |
| **Description** | Memcache shall initialize the following parameters at the first using by default values:   * cache size (100 mb); * preemption strategy (fifo); * storing time (30 min). |

## 3.2 Quality attributes

## 3.2.1 Reliability

Memcache library shall support reliability in terms of fault tolerance. That means that the library shall avoid unexpected crashes.

### 3.2.2 Performance

Memcache library shall support performance in terms of time-behavior. Stored resources shall be retrieved faster (at at least approx. 1 sec) than the equivalent data not stored in RAM.

### 3.2.3 Portability

Memcache library shall support portability in the sense that it should be easily set up and used in any program written in Eiffel language.

## 3.3 Design Constraints

**3.3.1 Design Constraint 1**

Capacity of the memory on the machine where memcache library is running. Cache size can’t be set more than this capacity.

**3.3.3 Design Constraint 2**

Memcached is constrained to operations we can implement using O(1) algorithms. Everything from set to flush has approximately constant algorithmic complexity. Nothing scales up with number of items. There's no regex, tag, prefix, etc. operations because we can't do them without slowing down everything.

**3.3.3 Design Constraint 3**

Memcache should be implemented by means of Eiffel language and is able to be run only in Eiffel environment.

## 3.4 On-line User Documentation and Help System Requirements

The link can be provided to any person, who is involved in this particular project.

<https://docs.google.com/document/d/1a_ePCO7vkiSV4vSEQ1ZnmRo_7LWERt9tIV5zbjTmLns/edit#>

## 3.5 Purchased Components

Not applicable.

## 3.6 Interfaces

### 3.6.1 User Interfaces

No user interface according to the implementation just of functionality.

### 3.6.2 Hardware Interfaces

The memcached interface comprises a small number of simple key-value operations. Memcached is attractive because the server interface is general — data is simply stored and retrieved according to a single key, a model which can be applied to nearly any workload

### 3.6.3 Software Interfaces

The Memcached interface provides the basic primitives - insertion, deletion, and retrieval. Data is stored as individual items, each including a key, a value, and possibly parameters. Item size can vary from a few bytes up to maximum cache size, although most Memcached implementations are heavily skewed toward smaller items.

### 3.6.4 Communications Interfaces

Not applicable.

## 3.7 Licensing Requirements

The driver is released under the Innopolis University license.

\* Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

\* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

\* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

\* Neither the name of the Danga Interactive nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

## 3.8 Legal, Copyright, and Other Notices

Not applicable.

# **4 Supporting Information**

The deadlines for this project are

1. requirements: 17 feb - 3 mar, final result: 10 mar
2. design: 10 mar - 24 mar, final result: 31 mar
3. implementation: 31 mar - 21 apr, final result: 28 apr
4. test plan: 31 mar - 28 apr, final result: 5 may
5. testing: 5 may - 26 may, final result: 2 jun
6. exam: 9 jun