Photress

Software Requirements Specification

Version 1.0

Revision History

Date	Version	Description	Author
10.10.2013	0.9	Initial version of this document	Team Photress
14.10.2013	1.0	Submitted version of this document	Team Photress

Table of Contents

Software Requirements Specification

1. Introduction

1.1 Purpose

This document shall specify all requirements on our software system "Photress". This specification shall also be used for programmers and future testers as a kind of checklist to enable them to go for all possible purposes of this program.

1.2 Scope

"Photress is a system which supplies a pure, headless gallery system which can be extended with our own UI. This will be done by a powerful backend that enables you to manage your photos in an intuitive way and makes them accessable by a well-document API.
Besides, Photress will deliver an open platform for many users which can be used as a service or hosted by yourself.

And for using Photress you will not even have to know how to program or how to configure a server - Photress will be available as a service hosted by us with a default frontend. Nevertheless you will also be able to host your own instance and develop your own fascinating frontends due to our open API."

1.3 Definitions, Acronyms, and Abbreviations

All definitions and abbreviations existing in this documents are being explained in the glossary.

1.4 References

Documents, which are needed for the understanding of this specification will be referenced here.

1.5 Overview

In the overall description, the product and its environment will be explained. This will foster further comprehension. The section of specific requirements will include a deep description of the requirements and it should enable future development and testing of the program.

2. Overall Description

2.1 Product Environment

The product service runs on a server – hosted by us or by an experienced user itself. The system will provide a (default) frontend, which will be accessible by normal desktop systems.

2.2 Product Functionality

The product is meant to be used by photo-enthusiasts and people how want to share the pictures in a more individual way than using Google Picasa or Yahoo! Flickr but still without the need of writing an own gallery system or configuring a complex one. It shall enable the user to manage his uploaded pictures and share them with them world!

### 2.3 User characteristics

The usage of the service is suitable for normal users without further IT-knowledge. To install the service itself, the user should have further knowledge in fields of web services and hosting.

2.4 Prerequisites

At the moment we are planning to run our service based on "NginX" and "Ruby on Rails". The operating system will by either a Debian- or a Ubuntu Server-system.

2.5 Use-Case Overview
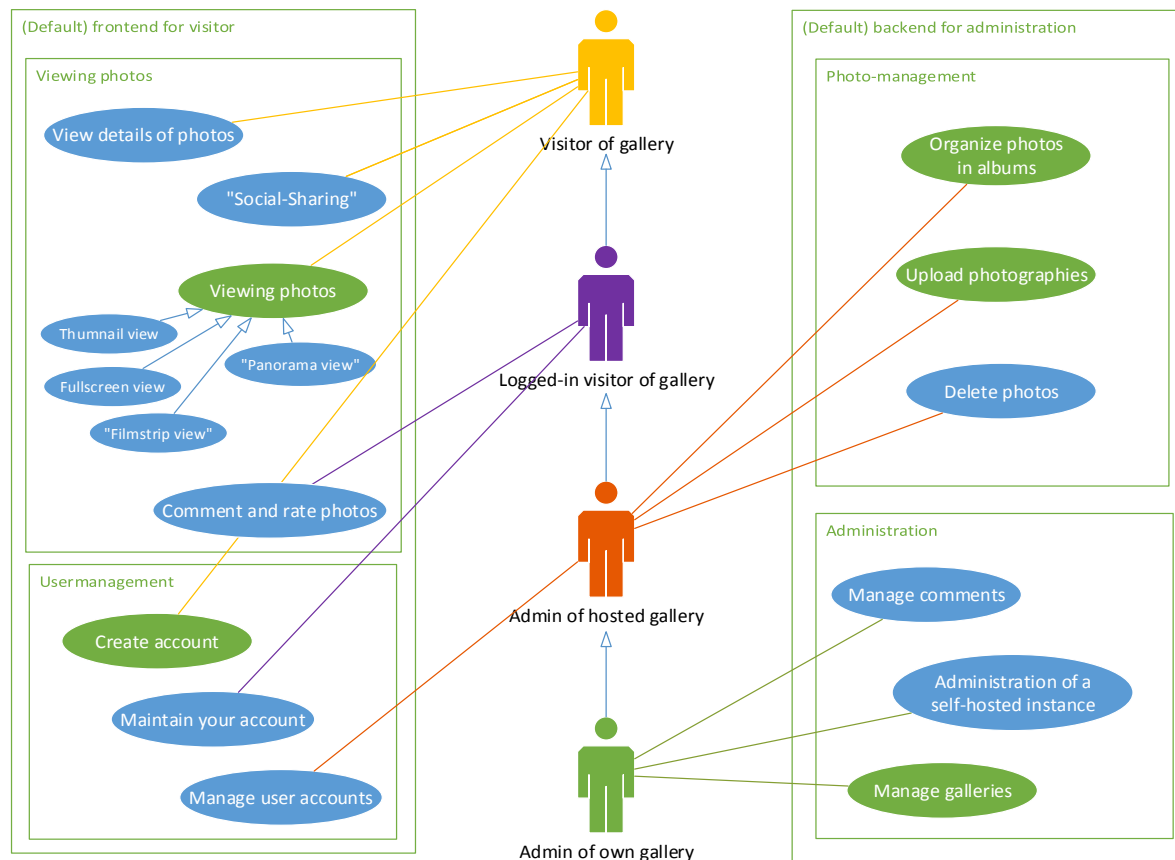
2.6 Dependencies and assumptions

There are no known dependencies (excluding the prerequisites) yet. The product is in full functionality as long as there is a hosted instance of it.

2.6.1 Market situation

There are other gallery software on the market. They are split into "full-service-providers" like Google Picasa and Yahoo! Flickr, which provide the user web apps and storage, and (open source) systems like Coppermine and Piwigo, which just provide a software system to install on the users server. But there is no service which offers central backend hosting and individual frontend hosting. This will give the users much more possibilities when it comes to individuality and freedom in the way of presenting their masterpieces. Our service can also be hosted by the user himself to give him even more freedom..

2.6.2 Delays

Sick workers can possibly delay the deployment of the software. Also defect technology can influence the time needed in a negative way.

3. Specific Requirements

3.1 Functionality

3.1.1 Viewing Photos:

The user will be able to view pictures in multiple ways, like for example in a thumbnail view, movie stripe view and full screen view using the default frontend.

3.1.2 Manage Galleries:

A user will be able to create his own gallery with setting styles to it. He or she should be also able to manage several

other settings regarding the gallery.

3.1.3 Upload Photography:

Photress will enable users to upload their own pictures to the service. The service itself takes over control of correct resizing of the uploaded pictures.

3.1.4 Organize Photos in albums:

It will be possible to organize ones pictures in multiple folders. It will be possible to add additional description to your albums (like a map or a further description).

3.1.5 Create account:

People will be able to register themselves for the service. With the registration completed, a user will have the possibility of creating his own galleries and he will be able to comment photos of other users.

3.2 Usability

3.2.1 Accessibility

The system will be accessible for most people. Therefore contrasty colors (but not garish ones) will be used. Fonts and icons will be big enough for comfortable reading.

3.2.2 Ease of handling

The system will be intuitive to work with. Furthermore there will be documentation for new users.

3.3 Reliability

3.3.1 Availability:

The system should be available over 99% of the time, excluding panned shut-downs for maintenance reasons. It should be usable until the user times out. Maintenance access will be given for administrators.

3.3.2 Mean time between failures:

Critical errors should not appear anymore when the development has been finished. Further development will be done in test environments so failures will not affect the system.

3.3.3 Mean time to Repair:

The system is allowed to be out of operation for two days.

3.3.4 Accuracy:

The system delivers pictures in the given format.

3.3.5 Maximum Bugs:

The number of maximum critical bugs will be 5.

3.3.6 Defect Rate:

Critical failures are those that force us to shut down the system and/or may cause a loss of data. Serious bugs are mistakes which lead to performance and usability issues. Non-critical bugs are bugs which only produce usability issues.

3.4 Performance

3.4.1 Response Time:

The average response time for viewing a normal picture (500x500px) is about 1s. The maximum time will be 6s for high resolution images. Actually this depends on the usage of the service and the given hardware.

3.4.2 Throughput:

This will be according to the performance of the server which is underlying our service.

3.4.3 Capacity:

This will be dependent on the performance of the server and on the storage capacity of the server.

3.5 Supportability

3.5.1 Naming Conventions:

The common conventions of the used programming languages will be adopted..

3.5.2 Maintenance Access:

Maintenance Access will be via SSH. Information will be accessible by log files and dump files.

3.6 Design Constraints

3.6.1 Simplicity:

The design of the default frontend will be as simple as possible.

3.6.2 Ergonomics:

The tools to manage images should work with few clicks.

3.7 On-line User Documentation and Help System Requirements

There will be a guide for the use of the API and for the general use of Photress.

3.8 Purchased Components

n/A

3.9 Interfaces

3.9.1 User Interfaces

There will be a default web frontend for all users. If the user itself is more advanced, a "kind-of" user interface in form of an API will exist for him or her to create an own user interface.

3.9.2 Hardware Interfaces

n/A

3.9.3 Software Interfaces

n/A

3.9.4 Communications Interfaces

There will be a RESTful-API that acts as an interface between Photress and the consuming frontend.

3.10 Licensing Requirements

We haven't decided on a licensing model right yet.

3.11 Legal, Copyright, and Other Notices

The program will be distributed according to the MIT-License.

3.12 Applicable Standards

The MVC-Standard will be supported.

4. Supporting Information

None yet.