CS470 Final Writeup
Denis Nikolskiy
4/29/2010

Abstract

The Internet is a powerful resource for expressing not only opinions but also numerous forms of art. One of the most popular forms is photography. It is so popular that one can easily get lost in the sea of pictures. Numerous online photo stock websites offer their services to anyone but they have the same general Internet problem: if one posts his or her picture there, it is just one more image in the pile. This project, Photo Portfolio, is designed for a local photographer. He needs a website that can offer a professional set of features and preserve the author's personality at the same time.

1. Introduction

This website is intended for photographers who want to post their photo portfolios online. There are currently a lot of websites that let photographers post their portfolios and sell pictures online and there are a lot of personal photographers websites where they present their portfolios. In the first case, all of the photo stock websites lack personalization. In the second case, personal websites lack a lot of useful features. Therefore, photoportfolio.com is intended to bring both worlds together by letting multiple authors post pictures in one place and yet remain unique at the same time.

2. Project Overview

The client for this project is a local professional photographer who wants to post his collection in an organized manner. He also wants to collaborate with other photographers and let them post their portfolios on the website. The goal of the project is to design and implement such a website. The scope of the project is to provide a minimum bare bones structure for the website. At the end of the project the result should be deployed on a web host, so the website will be accessible from the Internet. It is not in the scope of the project to implement all the features of a professional photo stock website. The scope of the current project will not cover the commercial aspect. Visitors will not be able to buy pictures from the website. If more photographers will be using the website in the future, more features will be added but in a scope of a separate project which will be derived from this one.

2.1. Data Files

Most data files will consist of pictures.

2.2. Existing Similar Websites

There are a lot of somewhat similar websites; the most notable of them are www.istockphoto.com and www.photoportfolios.net . While istockphoto is the most famous and easy-to-navigate photo stock website, photo portfolios let visitors easily view pictures by author.

3. Project Requirements

The client did not provide very specific requirements. Therefore, prototyping methodology was used. The requirements were expressed in form of user stories. Functional specifications were derived from those user stories.

3.1. Functional Specifications

1) Homepage Top menu. Top navigation menu will provide the same choices throughout the whole website.
   a) A picture in the main area and description of the goal of the website.
   b) Photographer's account home page. Hard coded left menu.

2) Upload the website to a web host. It is important to upload the website in the very early stages of development to make sure that the web host supports the chosen framework.

3) Login page with places to enter user name and password.

4) Set up MySQL database. The following is the list of tables
   a) User (default django user table was used)
   b) Picture
   c) Picture_category (captures many-to-many between Picture and Category)
   d) Category
   e) Print (stores available prints for a specific picture)
   f) Comment
   g) Recommended picture
   h) Blog

5) Administrator account. Administrator should be able to add new users, set initial user name and password.

6) Photographers' accounts. Photographers should be able to log in to accounts created by the administrator. Accounts' urls should be like the following example, www.photoportfolio.com/photographers/photographer's_username.

7) Add a left menu to photographer's account that will provide specific choices for each account.

8) Picture upload. Logged-in photographers should be able to upload their pictures.

9) Automatically resize uploaded pictures.

10) Organize uploaded pictures on the server. Store uploaded thumbnails in thumbnails directory and main picture in main_pic directory. Each set should be in the authors directory. Create those directories as needed.

11) Category assignment. This step will provide a way to assign categories to uploaded pictures.

12) View pictures by categories. This part will enable visitors to view pictures by categories.

13) Individual picture view. This part will enable visitors to view a specific picture and information about it.

14) Key words. Photographers should be able to add keywords to their pictures.

15) Search function. Visitors should be able to search for pictures by keywords.

16) Comments. Registered photographers should be able to add comments to the pictures. Comments should be seen only by registered photographers.

17) Recommended pictures. Registered photographers should be able to recommend pictures to other registered photographers. Visitors should see which pictures a particular photographer recommends while viewing his or her profile.

18) Blog. Add blogs to each account. Each registered photographer should be able to edit his blog.

During the course of the project there were several changes to the functional requirements but the core features stayed the same. At the end of the semester all functional requirements up to 3.10 were completed.

3.2. System specifications

The web site should be viewable from all major browsers: Firefox, Chrome, Internet Explorer (newer versions), and Safari.

4. System Design

The client did not specify which approach to use in the development of the website. Therefore, I chose Django framework.

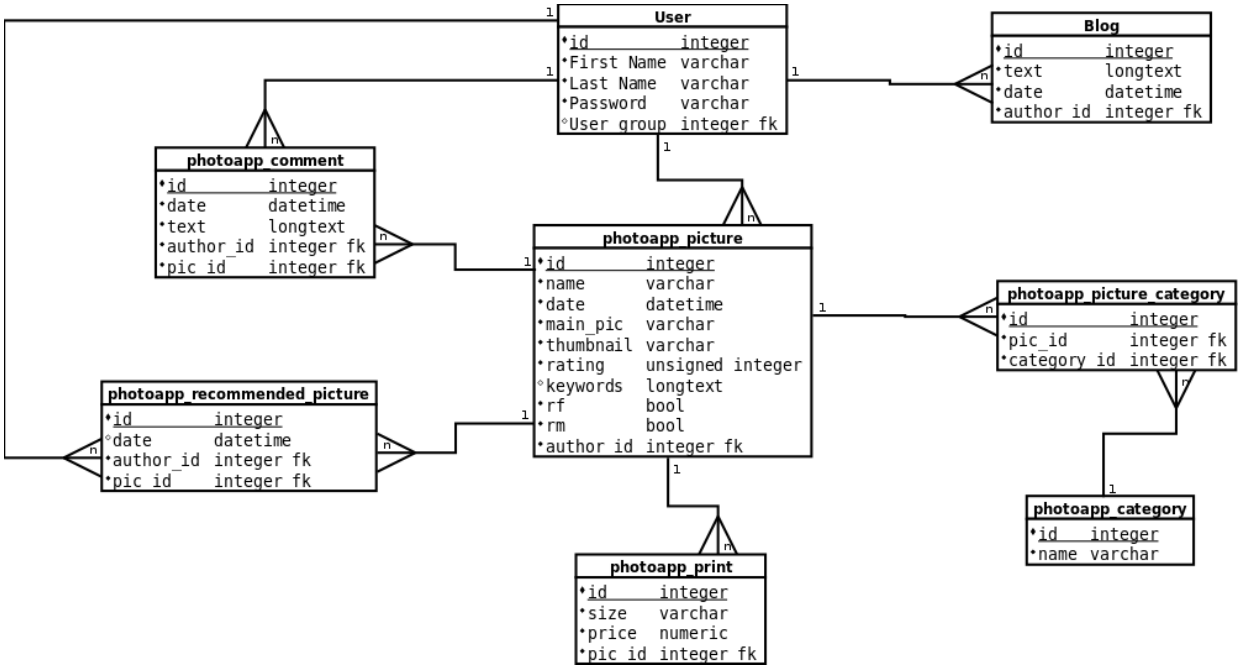
4.1. System Architecture

Django is MVC (Model View Controller) framework based on Python. However, since the Controller, the portion that delegates to a view depending on user input, is handled by the framework itself by calling the appropriate Python function for the given URL , Django is also referred as MTV (Model Template View) framework. Model is the data access layer. It contains anything and everything about the data: how to access it, how to validate it, which behaviors it has, and the relationships between the data. The template is the presentation layer. This layer contains presentation-related decisions: how something should be displayed on a web page.

View is the business logic layer. This layer contains the logic that access the model and defers to the appropriate templates.

4.2. Data structures

Most web hosts provide MySQL; therefore, MySQL will be used as the database for the website. The following are tables that will be created in the database. During the course of the project the database layout changed several times. The following layout is the latest version.

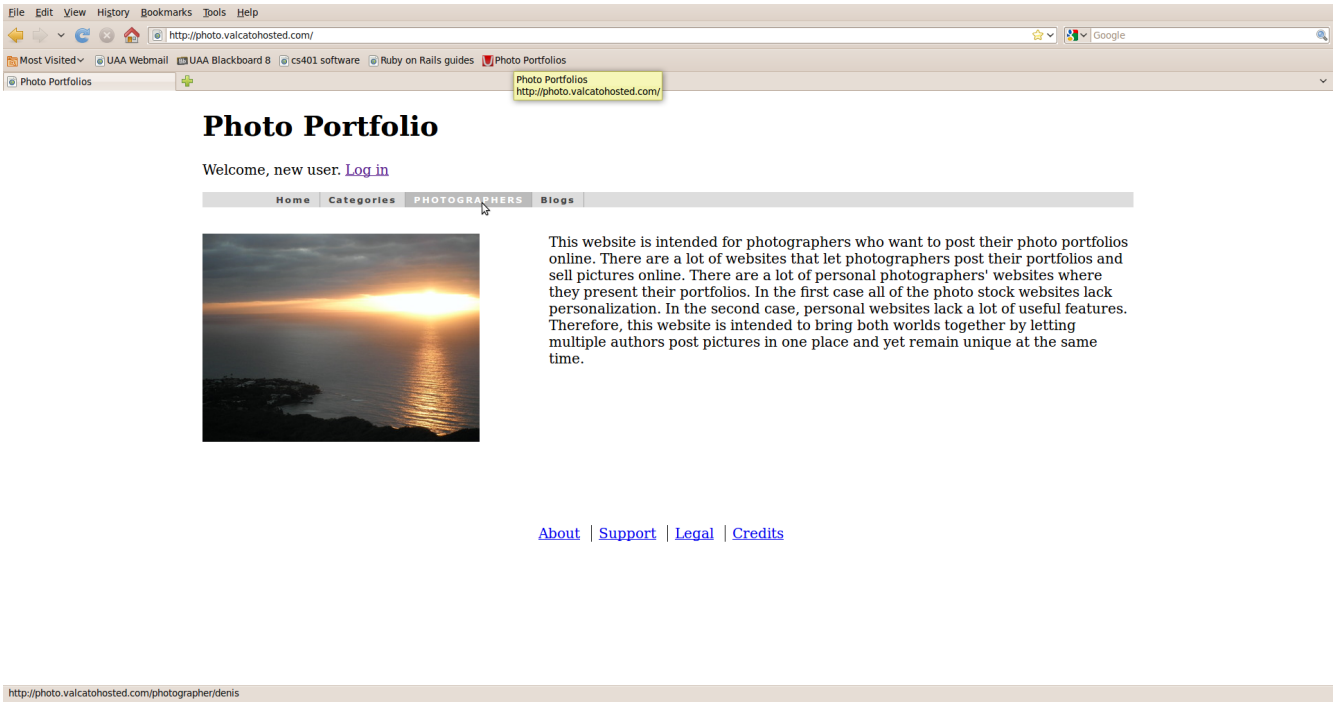


4.3. Algorithms

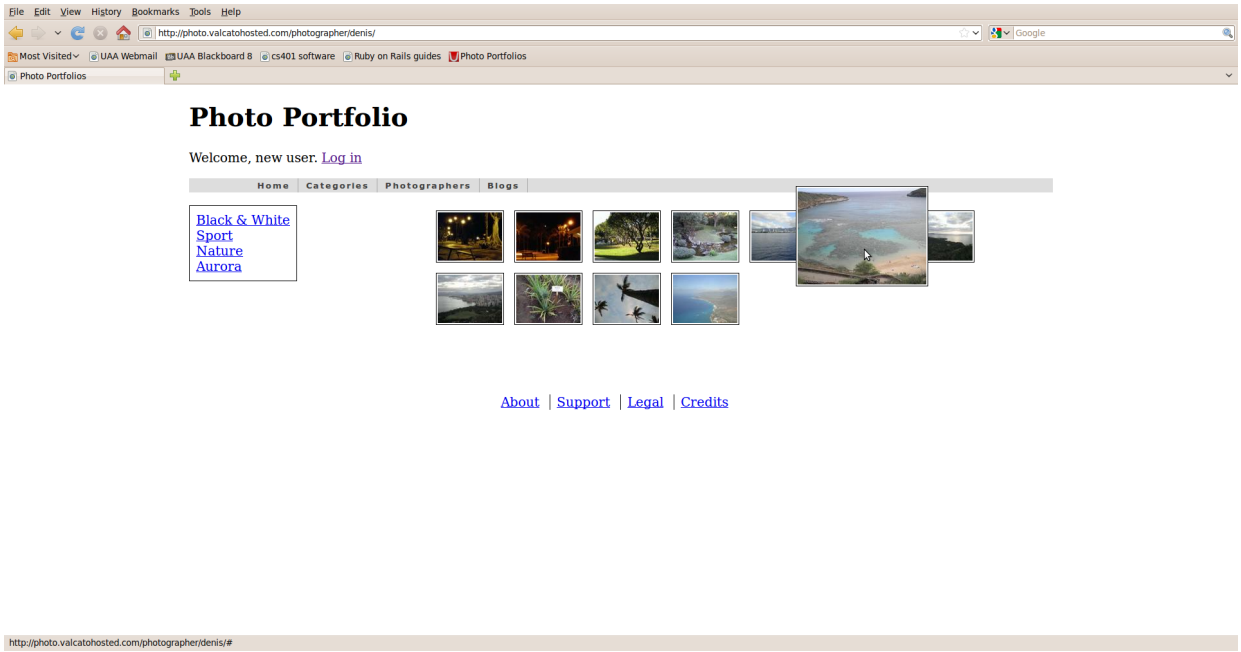
Although a big portion of the project consists of presentation logic and interaction with the database, there are several key algorithms. One is the actual download. The website is not uploading the whole image into memory. This way it avoids overloading memory with large files. It processes chunks of the file at a time. It also checks if the file is a proper image. It does not store or process any other files like text, scripts, zip files, etc. Currently the only format that it accepts is JPEG. Even though it took a little creativity and functional programming, the following functionality was achieved. The website checks the size of the upload image. If it is too big it resizes the image. It also creates a thumbnail copy of the image to speed up preview of multiple images. The website also manages the images on the server. It saves thumbnails and main images in the separate directories under the directory which belongs to the author who uploads the images. The original idea for resizing images was to use Python's ImageMagic library. However, PIL (Python Image Library) was a much more robust way to manipulate images from Python scripts.

4.4. User interface

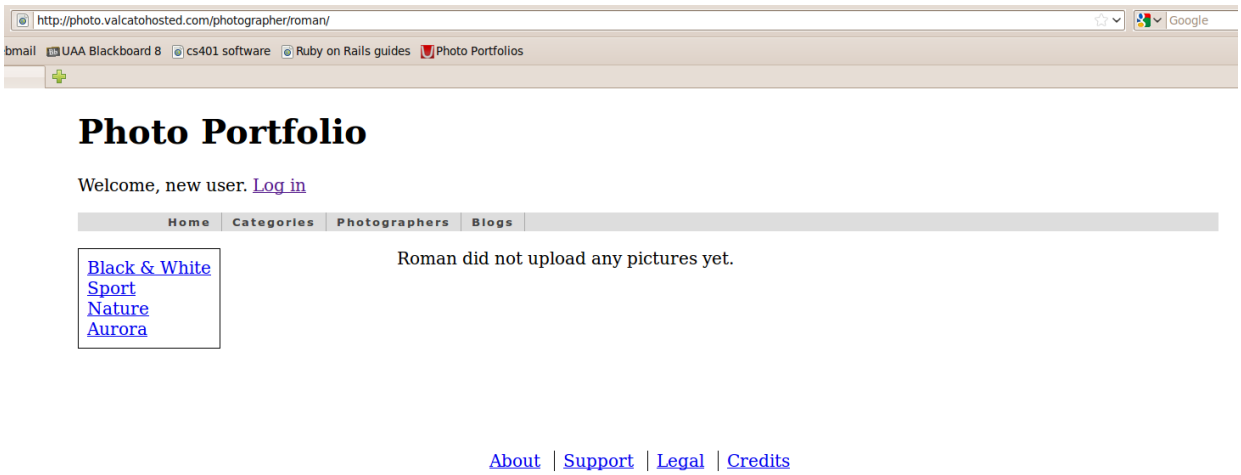
User interface is designed using HTML, CSS, Django template system, and Django admin pages. There are two menus. The top menu stays the same throughout the whole website. The left menu offers different options depending on user and current page. More options will appear in the left menu as the website grows. Currently it offers a link to an upload page to logged in users and a link to admin pages if the user is part of administration team. The following scree shots show the home page and the authors page.



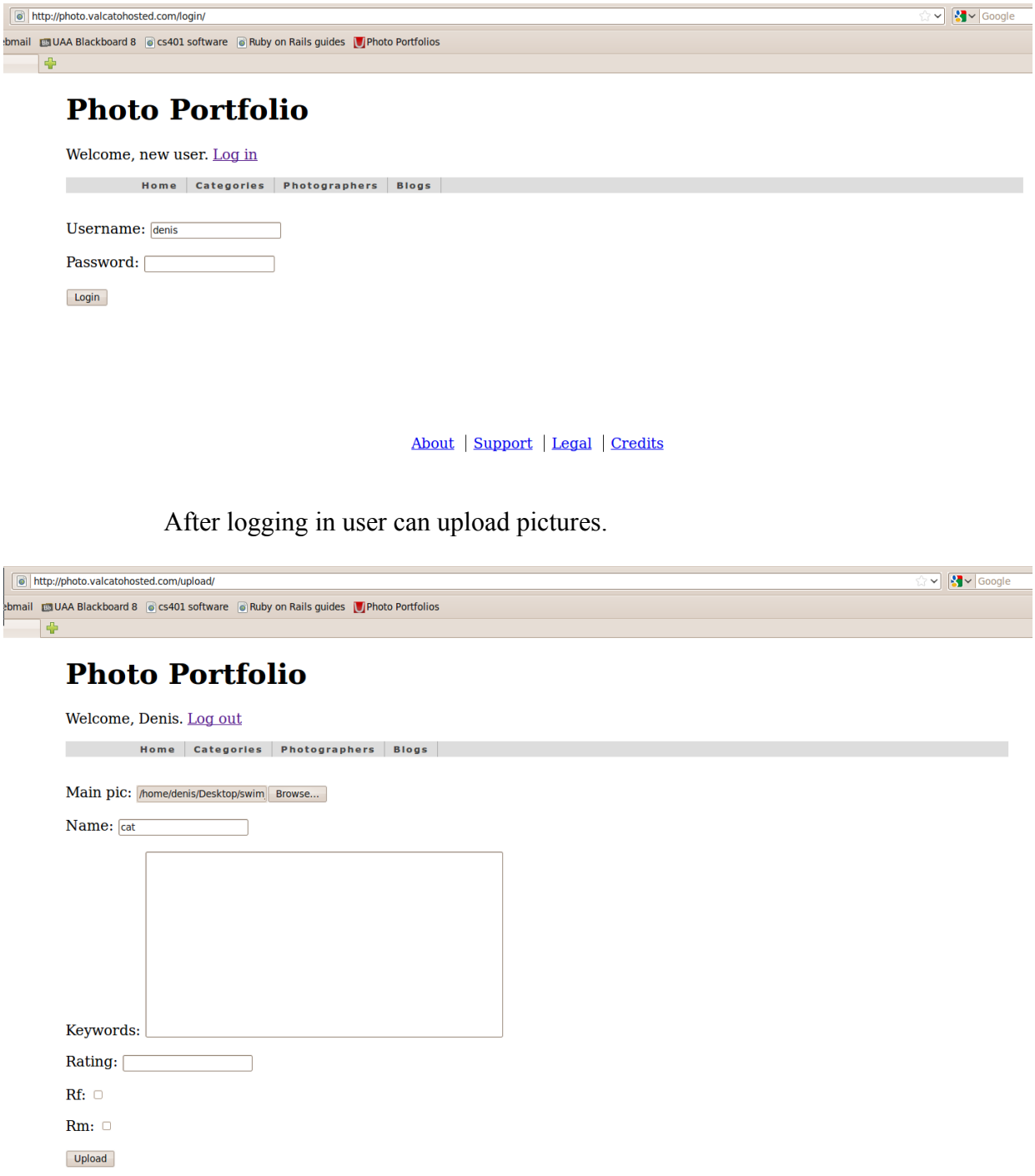
Unregistered visitors will see the following photographer's page (set of images will depend on chosen photographer).



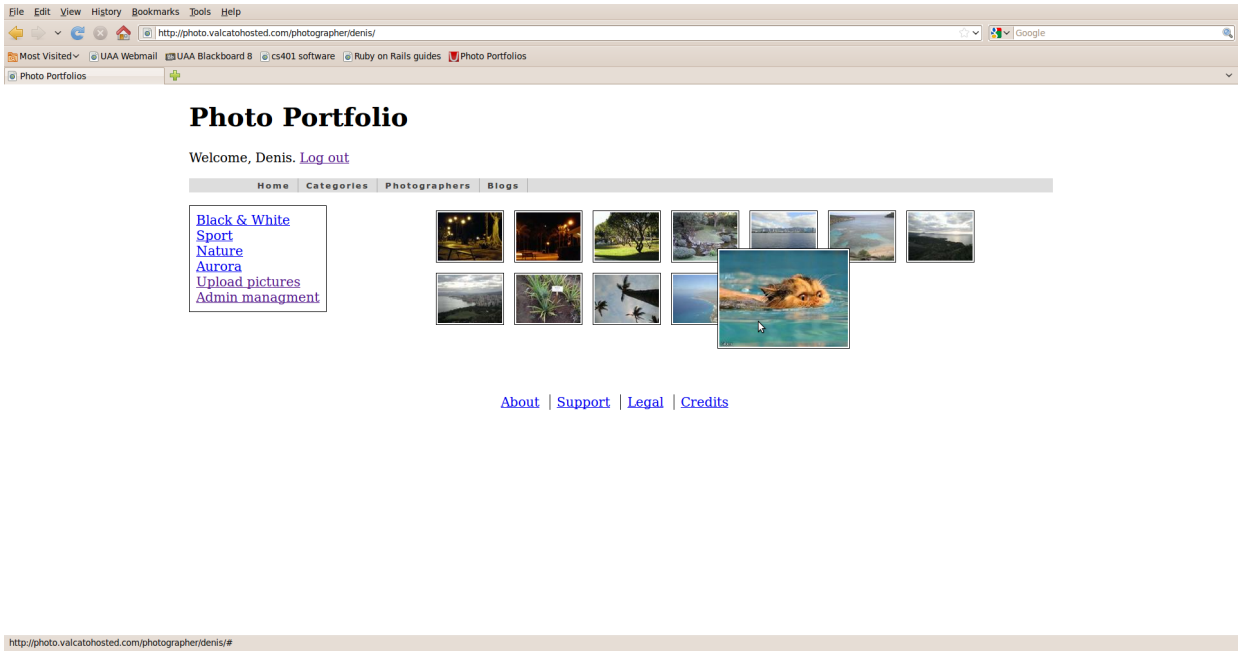
If the photographer did not upload any pictures, the page will display that there are no uploaded pictures.



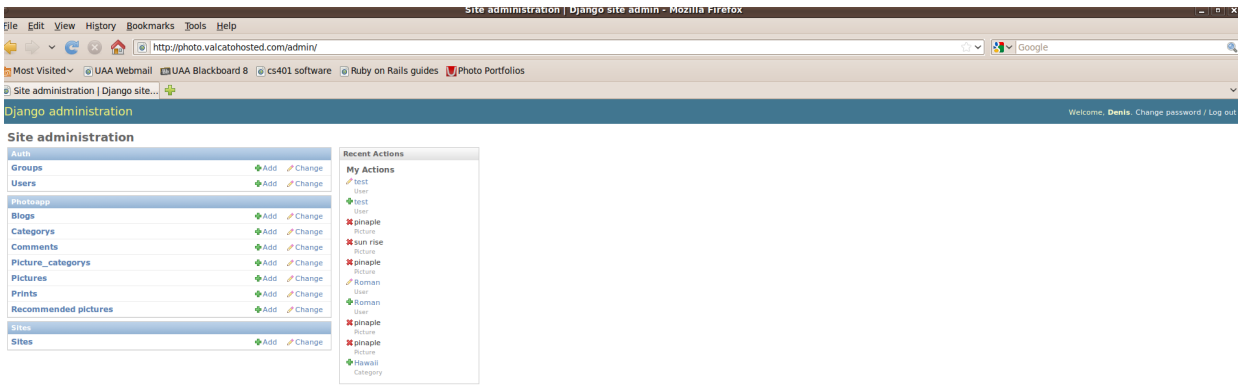
In order to login the user can choose “Log in” option above the top menu. It will bring the login page.



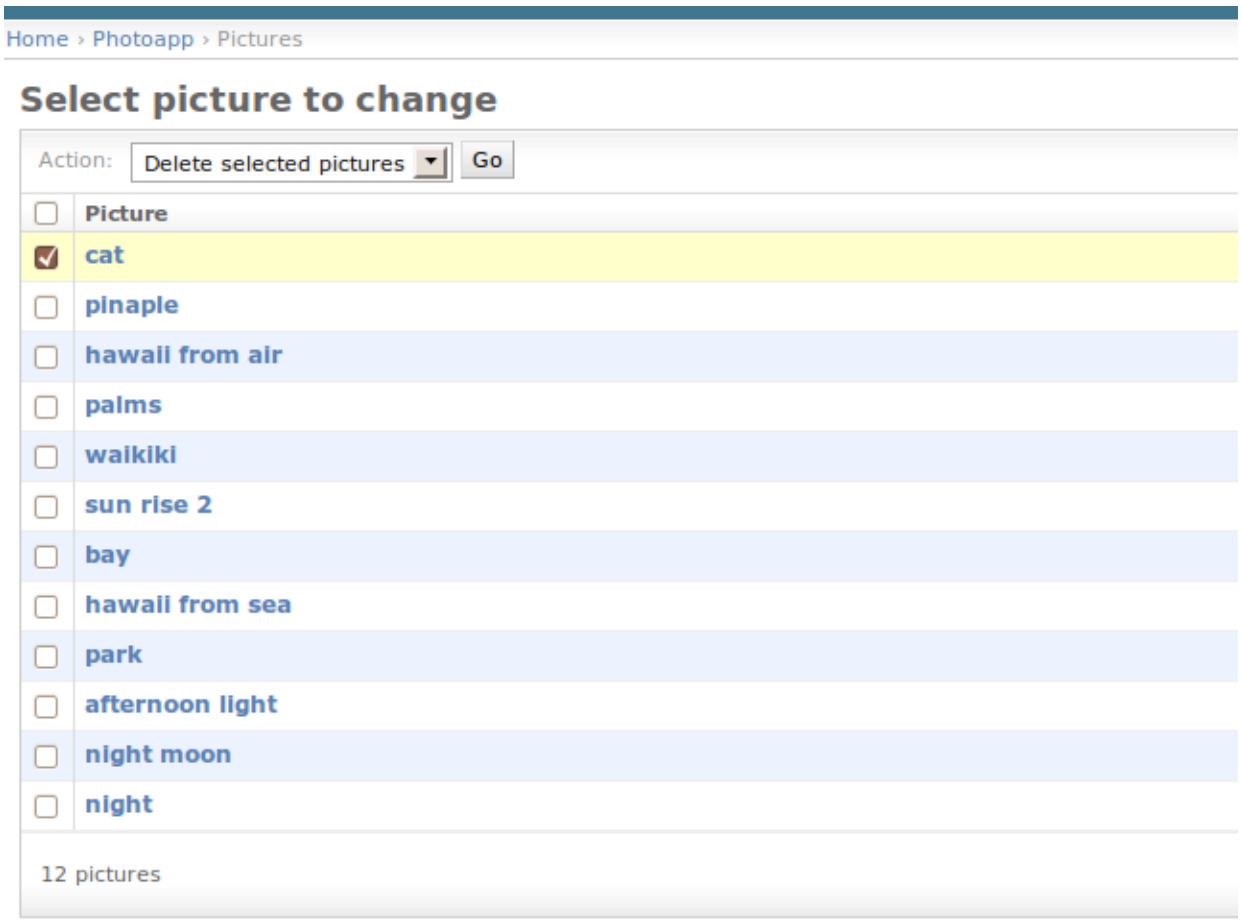
After logging in user can upload pictures.



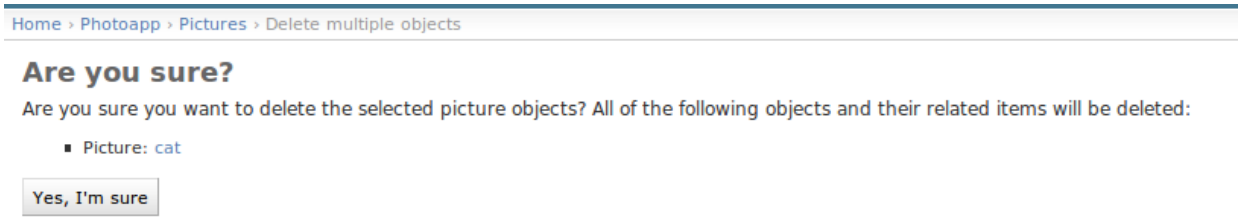
If the user belongs to the administration group he or she has the access to the admin pages and can add or remove information in the website's database.



Currently the only way to delete files is from administration pages.



After conformation the image will be deleted



Admin pages give access to all possible data manipulations in the database. More features like user ability to delete his or her images will be added later.

5. Software development process

Since the requirements were not very specific, I used a prototyping approach to the development of the project. I implemented an initial set of features and showed it to the client, gathered more information, fixed the initial set, added more functionality, showed it to the client again, and so on.

5.1. Testing and debugging

This project had an interesting complication. Since the website is published online, a lot of the debugging capability had to be hidden due to security reasons. It soon become apparent that tracking bugs on a production website is futile. Therefore, all development and most of the debugging was done locally on a testing machine. Only stable versions were pushed up to the production website. I chose to run most of the tests in pairs, one on the development website, and then the same one on the production website because of environmental differences on the production server and development server. Some additional tests had to be performed on the production website. Among those tests were compatibility with different browsers and website responsiveness (upload/download). I realized that some additional features might be needed as a result of those tests. One of those features is the upload bar. If the image is big the upload page just hangs there while the upload is in progress. It does not notify the user what is going on. It almost looks like the page is stuck. Also I discovered that the photographer's web page looks nicer in Firefox then in IE. While testing on the development version of the website I discovered and fixed the bug when the website crashes if the user uploaded not an image file.

5.2. Prototyping

Prototyping brought additional time overhead. I had to communicate with the client and discuss the features at every meeting. Also, the client requested a lot of additional features at every meeting. At the beginning I tried to implement all of them as they were requested but I soon realized that I would not have time for core functionality. Therefore, we agreed that those features will be added to “desired features” list and we focused on main functionality.

5.3. Word Breakdown

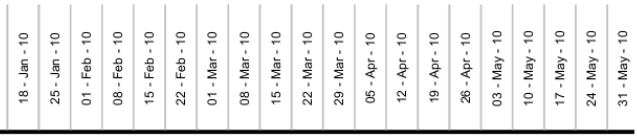
The original plan was to spend about 7 hours per week on the project and complete 2 user stories. I end up spending a lot more time then just 7 hours. Most of it was in reading documentation learning HTML, CSS, and Django. Following the Gantt chart shows finished and unfinished tasks.

Gantt Chart

Photo-stock website
CS470 project

Project Lead: Denis Nikolskiy
Today's Date: 4/29/2010 (Thu) (vertical red line)
Start Date: 1/18/2010 (Mon)
First Day of Week (Sun=1): 2

WBS	Tasks	Task Lead	Start	End	Duration (Days)	% Complete	Working Days	Days Complete	Days Remaining
1	**Initial tasks 1**	Denis	1/18/10	2/04/10	17	100%	14	17	0
1.1	Planning		1/18/10	1/20/10	2	100%	3	2	0
1.2	Requirements		1/20/10	1/25/10	5	100%	4	5	0
1.3	Design		1/25/10	1/30/10	5	100%	5	5	0
1.4	Proposal		1/30/10	2/04/10	5	100%	4	5	0
2	**Homepage (us 1)**	Denis	2/07/10	2/12/10	5	100%	5	5	0
2.1	Research CSS and HTML		2/07/10	2/09/10	2	100%	2	2	0
2.2	Top menu		2/09/10	2/10/10	1	100%	2	1	0
2.3	Main area		2/10/10	2/11/10	1	100%	2	1	0
2.4	Photographer's page.		2/11/10	2/12/10	1	100%	2	1	0
3	**Show Prototype (us 2)**	Denis	2/12/10	2/14/10	2	100%	1	2	0
3.1	Setup django on the web host and upload web pages		2/12/10	2/14/10	2	100%	1	2	0
4	**Login page (us 3)**	Denis	2/14/10	2/16/10	2	100%	2	2	0
4.1	Create and Upload login page		2/14/10	2/16/10	2	100%	2	2	0
5	**Set up MySQL (us 4)**	Denis	2/16/10	2/21/10	5	100%	4	5	0
5.1	Tables (a) - (c.)		2/16/10	2/18/10	2	100%	3	2	0
5.2	Tables (d) - (f)		2/18/10	2/21/10	3	100%	2	3	0
6	**Account (us 6 -7)**	Denis	2/21/10	2/25/10	4	98%	4	3	1
6.1	Administrator		2/21/10	2/23/10	2	100%	2	2	0
6.2	Photographer		2/23/10	2/24/10	1	100%	2	1	0
6.3	Left menu		2/24/10	2/25/10	1	90%	2	0	1
7	**us 8-11**	Denis	2/25/10	3/11/10	14	71%	11	10	4
7.1	Picture upload		2/25/10	2/28/10	3	100%	2	3	0
7.2	Category assignment		2/28/10	3/02/10	2	0%	2	0	2
7.3	Picture resize		3/02/10	3/09/10	7	100%	6	7	0
7.4	View by categories		3/09/10	3/11/10	2	0%	3	0	2
8	**us 12-15**	Denis	3/11/10	3/25/10	14	0%	11	0	14
8.1	Individual picture view		3/11/10	3/14/10	3	0%	2	0	3
8.2	Key words		3/14/10	3/17/10	3	0%	3	0	3
8.3	Search function		3/17/10	3/22/10	5	0%	4	0	5
8.4	Comments		3/22/10	3/25/10	3	0%	4	0	3
9	**Us 15-17**	Denis	3/25/10	4/08/10	14	0%	11	0	14
9.1	Recommended pictures		3/25/10	3/29/10	4	0%	3	0	4
9.2	Blog		3/29/10	4/08/10	10	0%	9	0	10
10	**Testing**	Denis	4/08/10	4/15/10	7	95%	6	6	1
10.1	Testing and fixing		4/08/10	4/15/10	7	95%	6	6	1
11	**Final presentation**	Denis	4/05/10	4/10/10	5	100%	5	5	0
11.1	Preparation		4/05/10	4/10/10	5	100%	5	5	0
12	**Preliminary presentation**	Denis	2/11/10	2/14/10	3	100%	2	3	0
12.1	Preparation		2/11/10	2/14/10	3	100%	2	3	0



6. Results and future steps

The resulting website is published online. Users can login/logout and upload images which will be displayed on their page. The administrator can create users and manage other data related to the website. Even though all planned features are not finished yet, a good base was established. The next step is to review all the features that were not finished and the once they have been added during the course of the project, assign priorities, and establish a new time line. Scheduling will be more precise since I'm more familiar with the web framework.

7. Summary and conclusion

This photo portfolio website was developed using Python based framework. The idea behind the project is to develop a website that offers functionality and preserves individuality of photographers.

Overall the project was challenging but very interesting. I learned about website publishing, Python, Django, CSS, HTML, MySQL, and even some bash scripting (automatically upload stable version on the web server). I'm definitely planning to continue working on the project until full customer satisfaction.

8. User manual

8.1. Requirements

Django 1.0 or 1.1
Python > 2.4 but not 3.x
postgresql_psycopg2, postgresql, mysql, sqlite3, or oracle (depending on chosen database. For this project I used mysql)
Web server (for development Django provides its own web server)
Operating System must support Python (the whole project was developed on Linux)

8.2. Installation

First the database must be configured. Depending on the environment (development or production), configure settings files. In the project directory settings.py must be changed. Line 5 should specify the name of development machine. The file dev_settings.py should provide database information (engine, user, password, etc). If the website is deployed on production server the same must be done in production_settings.py. In order to synchronize the database with the project in the project directory one should run the following command: ./manage.py syncdb (on Linux). The first time this command is executed (if database was set correctly) the system will prompt for admin user. After providing the requested information the development web server can be started using ./manage runserver (from the project directory). Some production servers might require additional steps. The extensive guide about how to deploy a Django website can be found here: http://www.djangobook.com/en/2.0/chapter12/