

Feasibility Report  
Gigapan Education Ruby on Rails Project  
Marissa Morgan and Iain MacQuarrie

Client: CREATE Lab, Carnegie Mellon University  
4/28/15

### **Abstract**

This report will determine whether or not the creation of a new website for the GigaPan Education project at Carnegie Mellon University's CREATE Lab can be considered feasible as a senior design project. The current status of the CREATE Lab's website is presented along with the problems and proposed solutions. The feasibility of the proposed solution, with respect to time and other resources, is then considered.

## Table of Contents

1 Introduction	pg. 3
2 Background	pg. 4
2.1 Client	pg. 4
2.2 GigaPan	pg. 4
2.3 Current Website	pg. 6
2.4 Major Problems and Proposed Solutions	pg. 10
3 Design Decisions	pg. 15
3.1 Framework	pg. 15
3.2 Administration	pg. 16
3.3 Authentication	pg. 17
3.4 Authorization	pg. 18
4 Legal	pg. 20
5 Spikes	pg. 20
5.1 Displaying GigaPans	pg. 20
5.2 RailsAdmin	pg. 21
5.3 Devise	pg. 21
5.4 CanCanCan	pg. 22
6 Stories	pg. 23
6.1 Release One	pg. 23
6.2 Release Two	pg. 26
7 Resources	pg. 28
8 Feasibility	pg. 30
9 Glossary	pg. 31
10 References	pg. 32

# **1 Introduction**

This feasibility report proposes the development of a Ruby on Rails website based on the existing site for the GigaPan Education project by CREATE Lab at Carnegie Mellon University (CMU). The CREATE lab is part of the Robotics Institute at CMU and primarily develops robots for community science and learning. The current site does not fit their needs and has not been maintained in years. The report consists of the following sections: background, design decisions, legal matters, spikes, stories, resources, and feasibility. The following paragraphs will describe each section.

The background section will describe in detail the client's goals, personnel, and resources, and provide a description of the GigaPan and GigaPan Education projects. This will be followed by an explanation of the problems with the existing site and the reasoning for why a new one is needed.

The design decision section will outline some of the major options for developing a solution to the problems, and provide reasoning for the decisions made in developing the solution. These decisions will include potential web frameworks, user authentication systems, authorization systems, and administration systems.

The legal section will describe the ownership and licensing details behind the finished project, as well as any and all legal decisions pertaining to the project.

The spikes section will describe what a spike is, how they are useful, and what spikes have been done for this project.

The stories section will start off by explaining stories and their importance in development. This will be followed by a breakdown of the desired functionality of the website into releases numbers. Each functionality will have a time and risk estimate associated with it.

The resources section will list the required hardware and software needed to complete the project.

Finally, the feasibility section will argue that with the given time, client, and resources, the project is feasible.

## **2 Background**

The background section is an overview of the client (CREATE Lab), GigaPan robot, GigaPan Education project, and the current website. The client subsection will describe the mission, personnel, capabilities, and funding of the CREATE Lab. The GigaPan subsection will explain how a GigaPan is created from a robot and a point and shoot camera, as well as the commercialized and educational parts of the GigaPan project. The final subsection will describe the design of the current GigaPan Education website and the problems with its design.

### **2.1 Client**

The CREATE (Community Robotics, Education and Technology Empowerment) Lab is part of the Robotics Institute at Carnegie Mellon University. Their mission is to explore “socially meaningful innovation and deployment of robotic technologies” [1]. Their main goals are to create a technologically fluent generation and to empower citizens to scientifically study their world. The lab has been around for almost fifteen years and employs around thirty programmers, engineers, and outreach professionals. Most of the programmers and engineers are current or past PhD/Masters candidates from Carnegie Mellon University.

The employees of the lab have a broad range of technology skills centered around robotics. Their skills range from circuit board design to web development to firmware development. They currently run a couple of servers with a least a dozen websites and web applications. Being a part of a university also gives them access to a wide range of skills if they run into a problem they can not solve.

The Lab is almost entirely grant and donation funded through local foundations and tech companies like Google. This money has enabled them to spread their projects and technology to 28,721 people representing 650 different organization from across the globe.

### **2.2 GigaPan**

The GigaPan robot and GigaPan Education project combine the goals of CREATE Lab, enabling students to become technologically fluent while learning science and other subjects. The Gigapan robot is an adaptation of the imaging system used on NASA’s mars rovers. Using the bundled software, a point and shoot camera can be used to create large zoomable giga-pixel panoramas that can be easily shared over the internet.

A camera is attached to the robotic tripod head called the GigaPan robot. The user then specifies the upper and lower right and left coordinates of the panorama. The robot then calculates the number of photos that needs to be taken, moves the camera to each location, and pushes the shutter button. After the picture taking is done, the user imports the photos to their computer where the bundled application stitches the photos together. This stitched panorama can then be uploaded to gigapan.com to share with others.

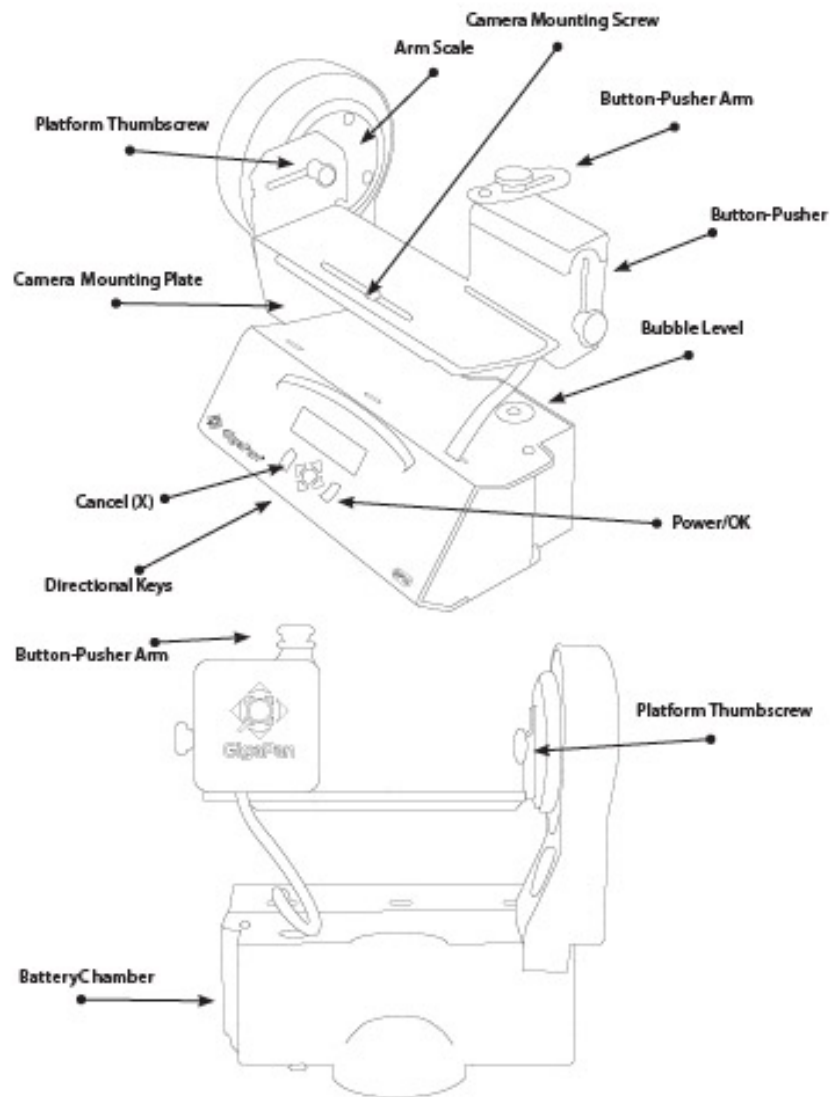


Figure 2.2.1: the GigaPan Robot



Figure 2.2.2: Images loaded into software ready to be stitched and uploaded to gigapan.com

Since being developed at the CREATE Lab, the Gigapan project has spun off into a private company called GigaPan Systems. They are based in Portland, Oregon and currently handle the manufacturing of the GigaPan robot. They also maintain the gigapan.com website and database of uploaded Gigapans. There are over 100,000 GigaPan hosted on the site [14]. The size of the images range from .7 giga-pixels to 50 giga-pixels. The robot and software for stitching the panoramas can be bought through them. However, the robot and software are not required to use the GigaPan Education site.

The GigaPan Education project was started as a way to share these images in an educational setting across the globe. Participating schools can create lessons about a certain subject using the multitude of GigaPan panoramas uploaded to gigapan.com. On the site, schools from around the world can partner with each other to learn about different cultures and languages.

### 2.3 Current Website

The current GigaPan Education website was written in Django in 2009. The site is still in use today, despite not being actively maintained for years. It has users from 13 countries representing 190 schools, 31 universities, and 80 organizations. The site has over 4965 users (Students: 4267, Teachers: 698), however, many are inactive. The

current admin section makes it difficult to determine exactly how many accounts are still active. Students on this site range from Elementary to High School age.

The site is centered around creating collections of teachers, students and GigaPans called projects. Each project has a theme or lesson that is discussed using the GigaPans as subjects for commentary. These projects can have multiple teachers and students from schools across the globe. For example, there is a project called Alimentation/Nutrition between schools in Switzerland and West Virginia. The GigaPans associated with this specific project showcase the different “stages” of food in each country. Users can, for each GigaPan in the project, zoom-in and comment on a specific section of the image. The students ask questions, in both French and English, about each other’s farms and supermarkets. The students are not only learning about another culture, but practicing their language skills. Many of the projects on the website have a cross subject and cross cultural aspect to them.



Figure 2.3.1 Showcases how the GigaPans, snapshots, and comments are used on the website. When the first comment is clicked the panorama zooms to what is shown in the bubble.



The current site has various parts on different servers that work together to deliver the web pages the users see. On the website's server is the Django framework which uses a MySQL database to store data. Various Javascript files provide code for displaying GigaPans, taking snapshots, and making comments. In order to use a GigaPan on the site it needs to be uploaded to gigapan.com. When a GigaPan is added to a project, the site queries the gigapan.com database using the id number of that GigaPan. This query returns a JSON (JavaScript Object Notation) object that provides the current site with the data required to display and use the GigaPan. The location of the data is stored in the site's database, while the actual data remains on the gigapan.com server. The figure 2.3.2 illustrates how the various parts fit together.

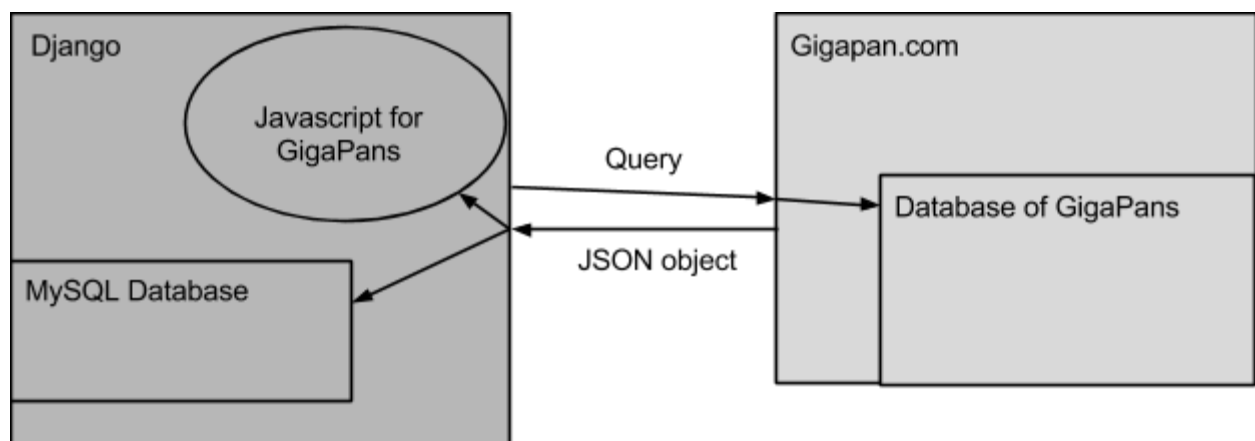


Figure 2.3.2 Passage of GigaPan data between GigaPan Education site and gigapan.com

The sitemap for the current website depends on the users access level. Students have the least number of actions they can perform. They can comment on GigaPans in project they are a part of and can edit certain aspects of their profile page. Teachers can perform most of the actions on the site. Along with the actions students can do, they can create/edit projects, add/remove GigaPans to projects, create new students, and add/remove users from projects. Admin users are very similar to teachers on the public side of the site, however, they have access to the administration panel. From this section of the site they can add, edit, delete, and archive users, projects, GigaPans, and comments. They can also view lists of all the users, schools, projects, GigaPans, and comments on the site. The full layout of the site along with user level permissions can be seen in figure 2.3.3.

## WEBSITE MAP

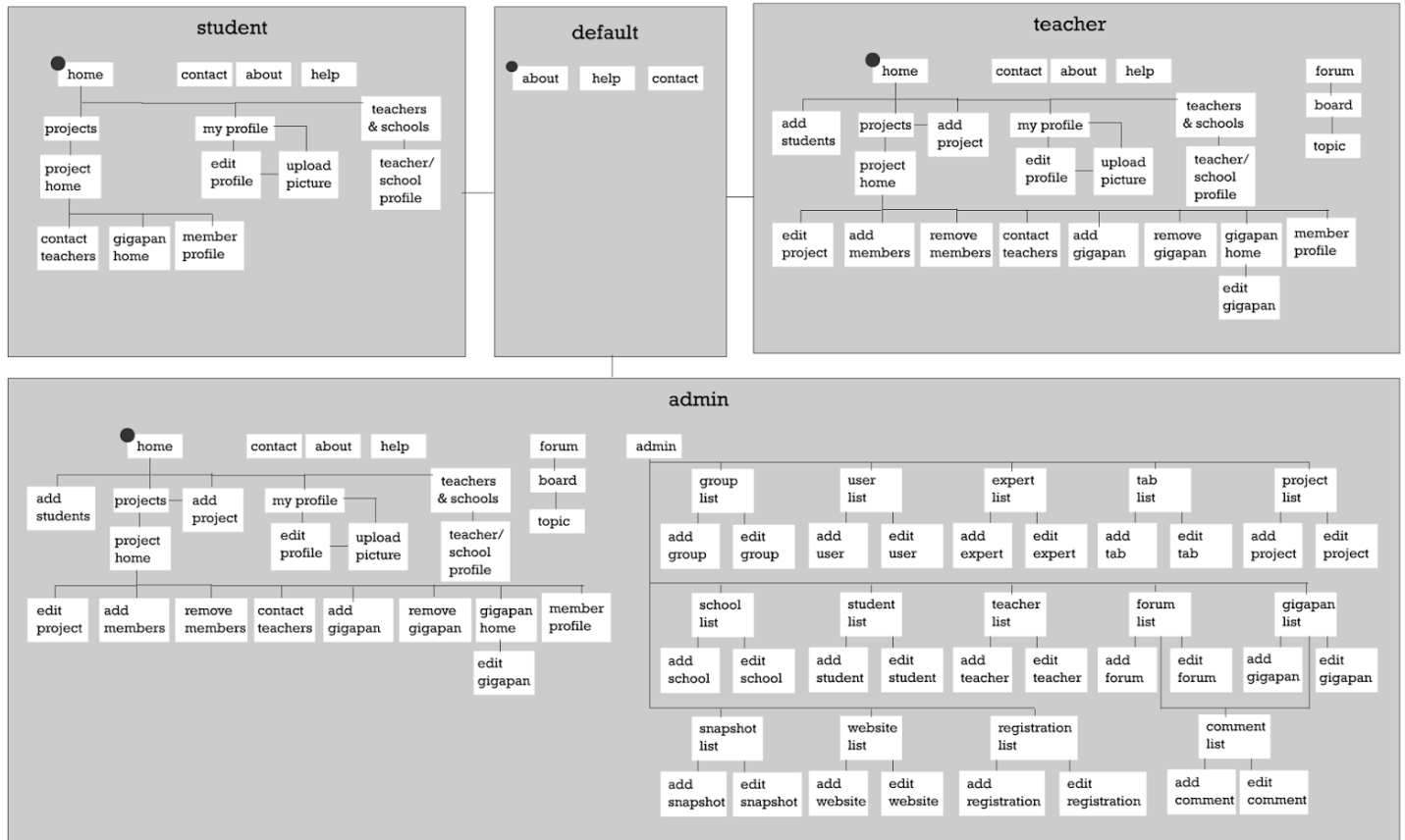




































Figure 2.3.3 GigaPan Education Sitemap

The problem CREATE Lab has with the current GigaPan education site is that it is written in Django and has been unsupported for years. The back end is not conducive to the needs of lab members and educators. One of the major issues is the way that Django has automatically created the backend, which is very confusing for general users. For example, one can create a new user under the Auth heading but this user will have no privileges and can not log in. To create a user that can actually use the site, they must be created under the Basic heading. The general database structure is incredibly inefficient as it stands, and has much room for improvement. In terms of promotion, the website is a bit of a black box to the general public. Without a login, it is hard to see how this site and the technology is used in education. The lab would like to change that by showcasing its projects to the public. As it stands now, the site makes it difficult to see recent activity in terms of projects and comments.

## Django administration

### Site administration

Auth		Recent Actions	
Groups	 Add  Change	<b>My Actions</b>	
Users	 Add  Change	None available	
Basic			
Experts	 Add  Change		
Group tabs	 Add  Change		
Projects	 Add  Change		
Schools	 Add  Change		
Students	 Add  Change		
Teachers	 Add  Change		
Comments			
Comments	 Add  Change		
Forum			
Forums	 Add  Change		
Topics	 Add  Change		
Pano			
GigaPans	 Add  Change		
Gigapan snapshots	 Add  Change		
Sites			
Sites	 Add  Change		
Threadedcomments			
Free Threaded Comments	 Add  Change		
Threaded Comments	 Add  Change		
Workshop			
Workshop2010 registrations	 Add  Change		

### 2.3.4 Django Administration for current website

#### 2.4 Major Problems and Proposed Solutions

Most of the issues with the current website can be boiled down to a couple of specific problems. The primary goal of the new website will be to solve these issues. The main problems with the website and some proposed solutions to fix them are discussed below.

- **2.4.1:** The current database structure separates each user role into a separate table, duplicating a lot of the values in user table instead of connecting the user table to a roles table which defines roles and their permissions. Not only is this approach inflexible for adding new user roles, but new users will not be associated with any role until created again under that role's table.

Home > Auth > Users

### Select user to change

Action:

<input type="checkbox"/>	Username	E-mail address	First name	Last name	Staff status
<input type="checkbox"/>	120039271db			Dylan Black	⊖
<input type="checkbox"/>	120039994pb			Paige Bowen	⊖
<input type="checkbox"/>	120040009cb			Carley Bledsoe	⊖
<input type="checkbox"/>	120042169			Soundarya Ferguson	⊖
<input type="checkbox"/>	120046339ta			Tracy Adkins	⊖
<input type="checkbox"/>	160006215			Tyler Sears	⊖
<input type="checkbox"/>	160006439			Brandon Sears	⊖

Home > Basic > Students

### Select student to change

<input type="checkbox"/>	Last name	Username	Initial password	E-mail address
<input type="checkbox"/>	Schwamborn	bschwamborn		
<input type="checkbox"/>	Alencastro	valencastro		
<input type="checkbox"/>	Mike	mblount66	12345678	
<input type="checkbox"/>	Qingyuan	qluo	12345678	

School	Date joined	Active	Staff status	Super
Escola Internacional de Aldeia - EIA	April 7, 2015, 2:01 p.m.	✓	⊖	⊖
Escola Internacional de Aldeia - EIA	April 1, 2015, 1:41 p.m.	✓	⊖	⊖
Mount Lebanon School District	March 31, 2015, 1:51 p.m.	✓	⊖	⊖
Mount Lebanon School District	March 30, 2015, 8:09 p.m.	✓	⊖	⊖

Home > Basic > Teachers

### Select teacher to change

<input type="checkbox"/>	Last name	Username	Initial password	E-mail address
<input type="checkbox"/>	Sousa	esousa		
<input type="checkbox"/>	Kevin Kieffer	kkieffer	wvu	kkieffer@k1
<input type="checkbox"/>	Iain	iain	changeme	

School	Date joined	Active	Staff status	Super
Escola Internacional de Aldeia - EIA	April 9, 2015, 7:36 a.m.	✓	⊖	⊖
North Elementary School	March 23, 2015, 8:48 a.m.	✓	⊖	⊖
Carnegie Mellon University	March 12, 2015, 4 p.m.	✓	✓	✓

Figure 2.4.1 Duplication of User Attributes Across the Database

- To solve this issue, a roles table will be created that defines roles and their permissions. Users will be linked directly to roles, and require a role before they can be created. Roles for users should be selectable from a list of available roles. The roles table will be editable by admins to include new roles and define their permissions.
- **2.4.2:** The “forum” and “workshop” tables have not been used for almost four years, but are still kept in the database.

[Home](#) › [Teachers' forum](#) › [New projects](#)

## New projects

This forum will be a place for teachers to suggest new topics, recruit and join partnerships, and develop ideas for joint projects.

[create new topic...](#)

Topic	Posts	Latest post
<b>Project Launched!</b> by Keat  , 4 years, 4 months ago	4 posts	"and another thing ... Anyone else who is interested, you ..." by Keat  , <a href="#">4 years, 4 months ago</a>
<b>Holiday Season</b> by Clara Phillips  , 4 years, 4 months ago	2 posts	"Oneof my students has added a picture of Market Square ..." by Keat  , <a href="#">4 years, 4 months ago</a>
<b>Global Health?</b> by Ms. Tomokiyo  , 4 years, 5 months ago	3 posts	"I have moved the panorama "UNESCO-IBE, HIV & AIDS Education ..." by Ms. Tomokiyo  , <a href="#">4 years, 2 months ago</a>
<b>New Project: Behind the Scenes</b> by Keat  , 4 years, 1 month ago	1 post	"I just started posting gigapans to a new project - ..." by Keat  , <a href="#">4 years, 1 month ago</a>
<b>Diálogos Escolares GigaPan</b> by Dror  , 3 years, 12 months ago	1 post	"This website is now available in Spanish (scroll to bottom ..." by Dror  , <a href="#">3 years, 12 months ago</a>
<b>4th grade class looking for a partner!</b> by Mrs. Goodburn  , 3 years, 8 months ago	4 posts	"Dear Jacque, Did you get any response on that? If ..." by Clara Phillips  , <a href="#">3 years, 6 months ago</a>
<b>Join in the Global Conversation with GigaPan!</b> by Labanya  , 3 years, 6 months ago	1 post	"Hello Teachers & Students worldwide, I am excited to invite ..." by Labanya  , <a href="#">3 years, 6 months ago</a>
<b>Looking for a dialogue</b> by Brynestad  , 1 year, 5 months ago	3 posts	"I would love to talk to both of you about ..." by Roberta Purper Brandão  , <a href="#">1 year, 5 months ago</a>

[create new topic...](#)

Preferred language: [English](#) [Español](#)

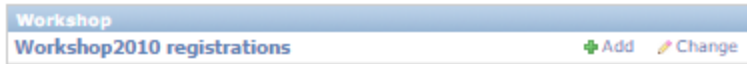


Figure 2.4.2 Unused Sections of the Website

- These tables and any other sections of the site that have not been frequently used should be depreciated.
- **2.4.3:** The current database does not clear out old or inactive user accounts, making the backend grow unwieldy if not maintained.

<input type="checkbox"/>	490011145			Bailey Jack	➖
<input type="checkbox"/>	701GBCCT2			Gollum	➖
<input type="checkbox"/>	701ghost7			ghost	➖
<input type="checkbox"/>	701jpmr04			meninos	➖
<input type="checkbox"/>	701LVSPC6			LVSPC	➖
<input type="checkbox"/>	701MCPFS5			MCPFS	➖
<input type="checkbox"/>	701nigga1			nigga	➖
<input type="checkbox"/>	701pluft3			pluft	➖
<input type="checkbox"/>	701rasta8			rasta	➖
<input type="checkbox"/>	720018175			Trey Smith	➖
<input type="checkbox"/>	720019141			Joseph Ord	➖
<input type="checkbox"/>	720019551			Jakob Stover	➖
<input type="checkbox"/>	720021156			John McVey	➖
<input type="checkbox"/>	720021580			Lachrshia Basham	➖

1 2 3 4 ... 58 59 5877 users

Figure 2.4.3 Unwieldy Database size

- A schedule for cleaning the database will be set up to automatically remove accounts that have not been active for at least one year.
- **2.4.4:** The help page uses an embedded google doc file which is difficult to navigate and does not integrate well with the look and feel of the rest of the website.

# Help

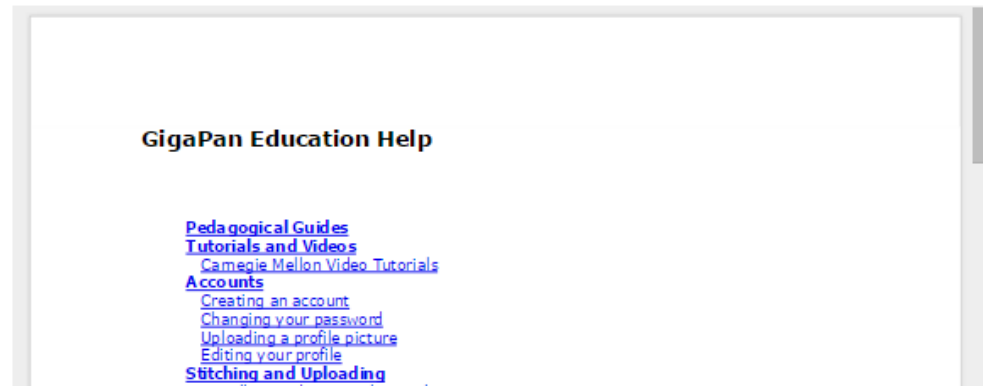


Figure 2.4.4 Embedded Google Docs Help Page

- The new website's help page will provide only the most relevant information from the google doc. More detailed information will be viewable by following a link.

### 3 Design Decisions

Design decisions are major choices to be made when designing a software application or website. These decisions vary from project to project, based on the needs of the client. Decisions are made over the course of development as major choices present themselves. For this project, the major design decisions made thus far have been what web framework, administration system, authorization system, and authentication system to use. Each subsection presents and compares various options for each decision and concludes with the team's reasoning for their chosen solution.

#### 3.1 Framework

One of the most important design decisions the team made was what framework to use. Out of all the possible frameworks the team seriously considered Ruby on Rails and Django. The factors compared for each framework were: ease of use, testing capabilities, resources, and support. The resources category factors in the availability of outside libraries and packages along with the availability of tutorials and solutions. The support factor takes into account the documentation and development of the frameworks along with the ability of the client to support the framework. The frameworks were rated on a 1-10 scale with the greater scores referring to better results.

Framework	Ease of Use	Testing capabilities	Resources	Support
Ruby on Rails	6	9	9	8
Django	4	7	8	7

Ruby on Rails is a web application framework written in Ruby. The guiding philosophies are Don't Repeat Yourself and Convention over Configuration [2]. This speeds up development because the programmer doesn't have to rewrite code for every configuration. Django is a web application framework written in Python. Django's goal is make development faster by automatically generating common web application aspects [3]. Both have a steep learning curve, but Ruby on Rails has a larger user base. Much of Ruby on Rails' functionality comes from libraries, called gems, that are not included with the base code. Django comes with many of these functionalities, like user authentication, written into the core program. Both frameworks are widely used across the internet but in recent years Ruby on Rails has seen a rise in popularity and demand.

The team decided to use Ruby on Rails because of its bigger user base, coding style and testing capabilities. One of the problems with the old website is that it wasn't maintained because it was written in Django. The original developer is the only one who



speaks Django at the lab and is too busy to support the website. However, the lab already has a few people who can develop in Ruby on Rails as well as a few supported Ruby on Rails websites. Also with its bigger user base, the lab should have no trouble finding someone else to maintain the website. After trying to learn both frameworks, the team felt more comfortable with Ruby on Rails' conventions. Because the website will most likely be updated infrequently. The development team wants to make sure as few bugs as possible make it to a release. Ruby on Rails robust testing environment will help isolate as many bugs as possible.

### 3.2 Administration

The second major design decision the team made was what Ruby on Rails administration gem to use. The client had no preference, so the team surveyed possible gems. The two most popular gems for administration are Active Admin and RailsAdmin. These two gems are very different in how they solve the administration problem. Active Admin is a scaffolding to create an admin interface. It is highly customizable and expects the developer to use it as a framework to build off of [11]. RailsAdmin is an automatically generated admin interface. It describes itself as "Rails engine that provides an easy-to-use interface for managing your data" [10]. While not as customizable as Active Admin, RailsAdmin does allow some modification. It is often compared to Django's automatic backend. The factors considered for each admin gem were ease of use, integration with other gems, the needs of the client, and documentation and support. The gems were rated on a 1-10 scale with the greater scores referring to better results.

Gem	Ease of Use	Integration with other Gems	Needs of the Client	Documentation and Support
Active Admin	5	7	6	7
RailsAdmin	6	9	7	6

The team decided to use RailsAdmin as the administration gem because of its automatic generation and integration with other gems. RailsAdmin is often suggested as a starting point for administration interfaces [9]. It is fairly easy to use and doesn't require a lot of code to work. The team and client do not need all of the customization that Active Admin gives. Because of its framework nature, Active Admin takes more setup time and more code to work. RailsAdmin, with its automatic generation, enables the development team to quickly get an admin interface to the client to test. RailsAdmin

is also better at integrating with other user management gems like CanCanCan (user authorization [12]) and Devise (user authentication [13]).

### 3.3 Authentication

Another crucial decision was deciding what authentication method to use. Of the options available, Devise, Authlogic, and a custom built authentication solution using Rails 4's built in functions were considered. Authentication systems built on OAuth or Open ID which use log-in credentials across multiple websites were discounted in the interest of keeping student login information specific to the school, and with the knowledge that many students using the website may not have their own email address.

Each solution was compared on the following criteria:

- ease of use
- speed of integration
- reliability of security
- documentation/community support
- flexibility/customization

Solutions were rated on a 1-10 scale with the greater scores referring to better results.

Solution	Ease of use	Speed of integration	Reliability of security	Documentation and Support	Flexibility and Customization
Custom built	10	6	4	4	10
Devise	6	8	9	9	5
Authlogic	6	7	9	5	6

Both Authlogic and Devise are difficult to use in their own ways. Devise hides most of its logic in files which can only be accessed with “devise” commands, so while its simple integration can be set up within a matter of seconds, customizing it to fit anything other than a general use application can be a pain [15,16]. Authlogic, on the other hand, makes its initial setup more difficult. It requires the creation of several classes before it will work on a basic level that are not generated on installation [23]. Authlogic does make customization easier however, due to its small size and simplistic structure. A custom solution, while much easier to implement and customise is much less secure than both Authlogic and Devise, which are community driven gems that have continuing support.

The primary weakness of Authlogic when compared to Devise is its documentation. Devise provides very thorough documentation, and has a much larger following than Authlogic, meaning more support [24]. Authlogic, by comparison, neglects to include even the most basic installation guide in its main readme file [25]. Despite this, it still scores higher than the custom built solution in terms of support.

Devise is the best solution for the project in terms of security and continued support. While not as easy to use as a custom solution, or as easy to customize as Authlogic, the extra security and documentation outdo the other solutions, giving it a clear advantage.

### 3.4 Authorization

After an authentication system is decided, an authorization system has to be developed to handle access to specific sections of the site. Rails supports a variety of different authorization gems, but the authentication gem chosen needed to support role-based authentication easily. In this system, users would be assigned a set of permissions called a role. Role-based authentication would also need to allow for easy integration with RailsAdmin. The possible solutions included Pundit, CanCanCan, and a custom-made solution. CanCan was disqualified as a possible solution due to it being unsupported post Rails4, and CanCanCan serving as its replacement.

Each solution was compared on the following criteria:

- ease of use
- speed of integration
- reliability of security
- documentation/community support
- flexibility/customization
- Integration with RailsAdmin

Solutions were rated on a 1-10 scale with the greater scores referring to better results.

Solution	Ease of use	Speed of integration	Reliability of security	Documentation and Support	Flexibility and Customization	Rails Admin
Custom built	10	1	4	4	10	1
Pundit	5	5	9	7	8	7
CanCanCan	10	8	9	8	10	10

In terms of ease of use, Pundit is the weakest of the three. Pundit utilizes a “Policy” system, where each new model has to have its own set of policies associated with it. These classes are not generated and have to be created each time a new model is generated [20]. Compared to CanCanCan, which consolidates all permissions into a single “ability” class, this system takes far more work [17]. The syntax is also much harder to understand than CanCanCan’s ‘can? :action’ style conditionals. The custom system also gets high marks for ease of use since it was created in-house.

The custom model gets the lowest score for speed due to the time taken to develop and integrate an entirely new solution. Pundit’s “Policy” model lowers its score for speed of integration, but rake installation helps keep it higher than the custom solution. CanCanCan has the highest score for speed because of a quick two command integration, but loses points because much of the customization must be done by hand [19].

Both Pundit and CanCanCan get high marks for security, documentation, and support due to being supported community driven projects, while the custom solution gets low marks for being only as secure and documented as one could make it. Pundit’s support community is slightly smaller than CanCanCans, due to it being much newer than the base CanCanCan was built off of, so it scores lower [17].

CanCanCan and the custom solution both rate high for customization. Since CanCanCan easily accepts custom method calls in the ability class, it works almost like an add on the a highly customizable self-built solution [19]. Pundit gets a lower score for making this process more difficult [20].

Finally, we have the integration with Rails Admin. This was the more important criteria out of all of them. The custom solution’s integration with RailsAdmin would have to be built from scratch, so it gets the lowest score. CanCanCan has a two step integration with RailsAdmin, making it the easiest [22]. Pundit’s integration is not as easy to understand and requires more time [21].

In the end CanCanCan was chosen due to the simplicity and level of customization it offers. CanCanCan solves potential issues that could have arisen from a custom built solution, and provides a simple, easy to understand syntax that can be quickly customized, unlike Pundit.

## **4 Legal**

Legal protections and ownership are important to consider when creating a project. The CREATE Lab did not require a non-disclosure agreement for the development team. The team has access to all the resources and code related to this project. All software and libraries used in this project are either open source or available to use for free.

The CREATE lab has asked the team to develop this site because they currently do not have the money to employ another developer. During development, the team will retain the ownership rights of the site. Upon completion, the team will donate the software to the lab. They are free to keep it under a personal license or release it as open source.

## **5 Spikes**

Spikes are small pieces of an application that demonstrate how a particular part is going to be implemented. They are also used to determine the best design or library to use on a certain part of the application. They can include more than just code snippets, and can consist of research into coding standards and best practices. More spikes will be created as the team implements stories. Spikes currently completed or being worked on for this project are: displaying GigaPans and configuring RailsAdmin, CanCanCan, and Devise gems.

### **5.1 Displaying GigaPans**

The purpose of this spike was to learn how to display GigaPans using Javascript, Seadragon and HTML. The original website uses Microsoft's Seadragon library and Javascript to display GigaPans. The Seadragon library is written in Javascript and is used to display deep-zoomable images on the internet [6]. The additional Javascript converts the GigaPan image tiles into an object that can be opened by the Seadragon viewer. This viewer can be placed inside any HTML div element. The GigaPan image then can be zoomed and panned. This setup was successful and can be used on the new site however support for the Seadragon library has been stopped.

There is an alternative to Seadragon called OpenSeadragon. It is an open source continuation of the Seadragon library [5]. However it calls the viewer differently, breaking the additional Javascript. Currently the plan is to use the Seadragon library because it will cost the least in terms of time and risk. It also has been working without any reported bugs on the old site for years.

## 5.2 RailsAdmin

The purpose of this spike was to learn how RailsAdmin generates the admin backend. The setup for RailsAdmin is very simple. Once installed a backend interface is automatically generated at domain.com/admin. As models and controllers are created and migrated into the database they appear in the backend. The interface enables a user to add, edit and delete records from the database as well as see the record inside the app. RailsAdmin uses associations to group data for display. For example, an app has two models, users and comments. Users have a has\_many relationship with comments. When the users section is displayed in the backend, it links the associated comment data. RailsAdmin also has a built in export of the data into CSV, JSON, and XML formats.

RailsAdmin is designed to work well with user authentication and authorization gems like CanCanCan. Integration tests can be run on the admin interface using the Capybara gem. It also can be extended with plugins and the RailsAdmin API.

RailsAdmin is a very quick way to implement most of the required admin backend functionality. It provides an easy to use powerful interface for non-technical admin users of the project site. This spike will be returned to as a testing ground for new features when more of the site is developed.

## 5.3 Devise

The purpose of this spike was to understand how Devise integrates with a typical Rails installation. Devise is one of the most popular authentication tools for Rails, but it was rumored to have a few issues with customization. This spike was developed to see if the issues would be a barrier to its implementation. Devise is fairly easy to set-up. All that is required after adding the gem is a devise installation command and the generation of a user model using the rails g devise command [16].

Devise can be customized to better fit your application, however this process can be difficult. In order to look under the hood at the default Devise files to see what is going on, one would have to run the associated rails devise command for the files. For example, to see views associated with devise, run 'rails devise:views' [15]. The Devise config file can also be adjusted to further customise authentication. For example, to create custom authentication fields, one would have to make an action for the field in the application controller and tell devise that the field will be used for authentication in the Devise config file [15].

Devise takes a lot of work to custom fit to your application, but is fairly easy to set-up and is a stable, secure system. This spike will be returned to whenever testing new customization of devise.

#### **5.4 CanCanCan**

The purpose of this spike was to test how CanCanCan handled user roles and authorization to view pages. CanCanCan was created by the CanCan community as a replacement for CanCan, a popular authorization gem that was abandoned by the release of Rails4. CanCanCan follows a similar structure to Devise installation, requiring a CanCan install command, and the creation of a cancan “ability” class [17].

The ability class primarily works as a way to check if a user has permission to view, create, or modify elements. These are defined with a simple if-else system. In a typical application permissions are associated with roles. CanCanCan doesn’t generate roles out of the box, however. In order to get CanCanCan to work with a role-based authentication system, one would have to generate a many-to-many User-Role relationship table, and then create an action within the User model to check the role of the user. This action can then be used in the ability class to define the abilities associated with each role, and used in the views and controllers to check for a specific role before performing an action. An authorization method has to be made in the application controller to control access to pages. Within the action all that needs to be done is to check for the presence of a certain role or set of abilities [19].

CanCanCan’s primary purpose is to set up typical permissions to check if a user can perform actions. This allows for easy catch-all conditions in views or controllers to determine whether the logged-in user can or cannot do something no matter what their role is. For example, if one wanted to check if a user could modify a field, the check would be ‘if can? :update, field’ This is a simple method that avoids checking against a list of user roles and allows for flexibility later on [18].

CanCanCan is easy to integrate and works as a flexible system for authentication that has a great support community. It integrates well with Devise without requiring any additional work. This spike will be returned to whenever additional authorization customization needs to be tested.

## 6 Stories

A story is short description of a specific functionality for a system. They are used to keep the development team and client on the same page in terms of how the software will function [7]. Stories are organized in priority order. The development team and client then draw a line to determine what stories are implemented for each release. The development team then estimates the time and risk involved in implementing each story.

Time is estimated for each story based on the risk of implementation and the skills of the development team. Actual implementation may take longer or shorter than estimated. The time estimates are used to show how long development of a release will take and how fast the client can expect to see new functionality.

Risk is the chance that a story will take longer to implement than estimated, or may cause difficulty or conflicts when implementing with the rest of the project. The risk is estimated from one to five, with five being extremely difficult to implement. Most of the risk in this project is associated with the development team's lack of experience with some of the software and programming languages required. Risk is also influenced by the client and resources. Stories that involve more resources and client input are riskier. By estimating time and risk for each story, the client is aware of how much it will cost, in both time and money, for each change to take place. Risks for this project will be measured on a one to five scale, with one being low risk, and five being the highest risk.

### 6.1 Release One

For release one, the client and development team have agreed upon having the same functionality as the original website. This entails having three user levels with various permissions. The highest user level is admin which have these privileges:

- Backend access and edit
  - Admin has access to view the administration back end for users, projects, gigapans, and comments. They will also have the ability to add/modify/delete certain attributes of users, projects, gigapans, and comments from this backend.
- Can create new users
  - Admin can create new student from their profile page and create a new user from the backend.
- Can create projects
  - Admin can create projects from their profile page.
- Can comment
  - Admin are able to comment only on project they are a part of.



- Can edit user info
  - Admin can edit their own user info from their profile page and edit certain attributes of other users from the back end.
- Activity section
  - A section on their profile page that shows recent project activity. This activity would include information about new projects, GigaPans, and comments.

The team estimates that implementing the admin user level will take 40 hours and is at risk level three.

The second highest user level is teacher. They are able to:

- Create projects
  - From their profile page, teachers can create new projects.
- Create student accounts
  - Teachers can create new students from their profile or project page.
- Comment
  - Teachers can comment on a project they are a part of. If they would like to comment on a project they can email the teacher using the link on the project teacher's profile page.
- Edit user info
  - Teachers can edit their own user info as well as some of their students data.
- Activity section
  - A section on their profile page that shows recent project activity. This activity would include information about new projects, GigaPans, and comments.

The teacher user level is estimated to take 25 hours to implement at risk level 2.5.

The lowest user level is student. They have very limited privileges that protect their privacy. These privileges are:

- Can comment
  - Student would be only able to comment on projects they are a part of.
- Can edit some user info
  - Students are only able to change their profile picture, description, and password. Students can only change their password while logged in. Forgotten password recovery for students would use the email of the teacher who added them to the site.
- Activity section

- A section on their profile page that shows recent project activity. This activity would include information about new projects, GigaPans, and comments.

The team estimates that implementing the student user level will take 15 hours at risk level two.

Projects are a collection of related GigaPans that are used by a teacher(s) to discuss a certain issue with students. The user levels interact with projects in different ways. The stories for projects are:

- Edit project detail
  - Teachers and Admin users can edit details of the project like name, description, and curriculum area.
- Add GigaPans to project
  - GigaPans can be added to the project using ID number or searched for on gigapan.com. All GigaPans must be hosted on gigapan.com. Data about the GigaPan is copied to the sites database from gigapan.com. A preview image is shown on project page. When clicked it opens a page that displays the full GigaPan with the discussion underneath.
  - Comments on the GigaPan can be made with and without a snapshot of the GigaPan. A snapshot is a reference to a specific sample of the image. When clicked on, the displayed GigaPan zooms and pans to that sample. Each snapshot starts a new discussion thread where other users can comment. Each of these comments are color coded so the user level of the commenter is known.
- Add people to project
  - Teachers and admin can add other users to the project. Once added these users can then comment on the project's GigaPans.
- List of all projects
  - A list of projects on the site that is visible to all users. The name, description, location, teacher, and curriculum area would be displayed.

Implementing the stories for projects and GigaPans is estimated to take 50hrs at risk level four. Project implementation is the bulk of the project and requires the most outside resources.

Most of the pages on the site require a person to be signed in to see. However there are a few public pages that explain the site to outsiders. These pages include:

- about/homepage
  - A description of what the site is about. This would also be the homepage where users could login from.

- contact page
  - A contact form that non-users can use to email the Lab's outreach team about the site.
- help/resources page
  - A page for users and non-users to learn more about the site and how it works.

Since these pages are mostly static they require very little time and risk to implement. They are estimated to take ten hours at risk level one.

The total amount of time required for release one is 140 hours. This time estimate is acceptable for the client and development team. The biggest risk is implementing the GigaPan project functionality because it requires resources the team currently does not have. However this risk will be reduced once the team has access to the required resources in May.

## **6.2 Release Two**

Release two is mostly comprised of new functionality the client would like to see in the new site. Again, the stories for this release can be categorized by their location in the site. The addition functionality requested for the backend are:

- Batch upload users by admin and teacher users
  - Authorized users would be able to upload a csv or excel file that would create new student users based on the data. The file would need to be checked format to prevent database corruption.
- Database export to csv
  - Admin users would be able to export tables in the database to use for grant proposals and reports
- Database search
  - Admin users would be able to search the database by project and student attributes.
- Duplicate checking of users and projects
  - Upon creation of a new users or project, the site would prevent duplicate project names and user's usernames.

The development team estimates that implementation of these stories will take fifteen hours and is a risk level two.

The client would also like to have an additional user level called expert. Experts would be able to:

- Comment on project they are a part of
- Can edit their user info

- Activity section
  - A section on their profile page that shows recent project activity. This activity would include information about new projects, GigaPans, and comments.

Because their privileges are very similar to the student user level the team estimates that it will take fifteen hours to implement and is at risk level two.

The additional functionality wanted for the projects are:

- Attach a curriculum PDF
  - Teachers and Admin would be able to upload a PDF of the projects curriculum so that other teachers can use it while planning a similar GigaPan project.
- Archiving inactive projects
  - Inactivity would be based on how long since the last comment on a gigapan related to the project was. An admin user would sort the projects based on this time and archive the project. Archiving the project would remove it from the list of projects but it would still show up on the users page and in the database. Teachers would be able to archive and unarchive their projects from the projects page.

The team expects this to take fifteen hours to implement with a risk level of two.

For the public side of the website the client would like to:

- showcase excellent projects
  - Excellent projects would be determined by the number and frequency of comments. GigaPans from these projects would be cycled on the homepage to showcase the site.
- a search tool for projects
  - The tool would search the projects by title, geographic location, and subject area.

The team estimates that this will take twenty-two hours at risk level three.

The total amount of time required for release two is 67 hours. This time estimate is acceptable for the client and development team. The biggest risk is implementing the GigaPan project search and showcase functionality because it requires complex database commands and significant client input and testing. Because of this, it may get pushed to release three, when part of the development team will be working from the client's office.

## 7 Resources

The hardware resources required for this project are minimal. Most of the software resources have been acquired and the rest are either owned by the client or are open source ruby on rails gems. The hardware and software currently being used for development are:

- Repository - Github public git repository
- IDE/Development server - Cloud 9 Ruby on Rails Virtual Machine
- Development/testing server - Google Compute Engine and a virtual machine on the cs servers at Western Michigan University. Currently running Apache 2.2 with Ruby 2.2 and a MySQL database.

We are using Cloud 9 for development because it enables us to do peer programming without being in the same room. Each of us has access to the others workspace to see what is being worked on even if it hasn't been pushed to version control.

For software and gems, the project currently requires:

- GigaPan Javascript API - Using the Seadragon library, this API displays and makes snapshot comments of GigaPans.
- Ruby 2.2 - Using the newest stable version to minimize future upgrade issues and take advantage of new security fixes.
- Rails 4.2 - Using the newest stable version to take advantage of new features and security. It will also minimize future upgrade issues.
- mysql2 - Using a MySQL Active Record database to maximize compatibility with previous site's database.
- bcrypt - for creating secure passwords

Gems required to implement stories will be added as needed. While developing, the team will need access to the old site, including its source code, to duplicate functionality and maximize compatibility. The team will also need access to the database of GigaPans at gigapan.com. The team is expecting to get the source code and database access in May.

For deployment, the client is responsible for supplying the server. The specifications are:

- Apache 2.2
- MySQL 2
- Ruby 2.2

The client will have no problem creating and supporting the server based upon the agreed specifications. The client will also supply a location to backup the site and database in case of hardware failure.

The resources required for this project have either been obtained or have a plan for acquiring them. The risk involved with resource acquisition is currently high but a plan is in place to reduce the risk. Once the outstanding resources have been secured the risk associated with resources will be minimal.

## **8 Feasibility**

The development team believes that release one can be accomplished by December 2015 based on development time and access to client and resources. The team has seven months until the scheduled release date with most of the development expected to be completed during the summer. The total time estimated for release one is 140 hours. Accounting for possible risk and setbacks, the time remaining for development is more than enough to complete the project. Based on the total time estimate for release one, the project should be completable by December 2015.

Since the project has an out of state client, development during the school year has been difficult. However, this summer, one team member will be working from the client's office. This will make getting feedback on new functionality and asking questions about implementation much quicker. They will also have better access to the sites original authors if any questions arise about the old website's implementation.

Currently, the resources required for this project have been obtained or are easy to obtain. The team already has a repository and a development server. As well as the code for displaying and commenting on the GigaPans. The few outstanding resources can be acquired from the client this summer. Based on the time, client, and resources the team is confident that this project can be completed by the release date.

## 9 Glossary

Authentication	A method of checking that the user really is who they say they are. This is usually handled with a log-in.
Authorization	A method of checking whether a user has permission to interact with certain sections of the website.
Gem	An add-on with additional functionality for ruby on rails. Gems can be anything from a single class to a full functional suite of tools.
GigaPan	A high-resolution, interactive gigapixel panoramic image.
JSON	JavaScript Object Notation is a format for transmitting data, using attribute-value pairs, that is easily read/written by humans and machines.
OpenSeadragon	The name of the continuing open source development of Seadragon.
Project	In terms of this website, a project is a collection of GigaPans that can be commented on by users assigned to the project.
Seadragon	A visualization software for viewing large high resolution images over the web without significant loading time. It was written by Microsoft in Javascript and released as an open source library.
Web Framework	A framework for developing web applications. Provides libraries and a structure to build websites from.



## 10 References

- [1] <http://www.cmucreatelab.org/>
- [2] <http://rubyonrails.org/>
- [3] <https://www.djangoproject.com/>
- [4] <https://bernardopires.com/2014/03/rails-vs-django-an-in-depth-technical-comparison/>
- [5] <https://openseadragon.github.io/docs/>
- [6] <https://msdn.microsoft.com/en-us/expression/gg413361>
- [7] Extreme Programming Explained by Kent Beck (2000)
- [8] <https://education.gigapan.org/>
- [9] <http://batsov.com/articles/2011/11/20/admin-interfaces-for-rails-apps-railsadmin-vs-activeadmin/>
- [10] [https://github.com/sferik/rails\\_admin](https://github.com/sferik/rails_admin)
- [11] <http://activeadmin.info/index.html>
- [12] <https://github.com/ryanb/cancan>
- [13] <https://github.com/plataformatec/devise>
- [14] <http://gigapan.com/>
- [15] <https://github.com/plataformatec/devise/wiki/How-To:-Allow-users-to-sign-in-using-their-username-or-email-address>
- [16] <https://github.com/plataformatec/devise>
- [17] <https://github.com/CanCanCommunity/cancancan>
- [18] <http://railscasts.com/episodes/192-authorization-with-cancan?view=asciicast>
- [19] <http://stackoverflow.com/questions/12051513/rails-cancan-set-assign-roles>
- [20] <https://github.com/elabs/pundit/blob/master/README.md>
- [21] [https://github.com/sudosu/rails\\_admin\\_pundit](https://github.com/sudosu/rails_admin_pundit)
- [22] [https://github.com/sferik/rails\\_admin/wiki/CanCan](https://github.com/sferik/rails_admin/wiki/CanCan)
- [23] [https://github.com/binarylogic/authlogic\\_example](https://github.com/binarylogic/authlogic_example)
- [24] [https://www.ruby-toolbox.com/categories/rails\\_authentication](https://www.ruby-toolbox.com/categories/rails_authentication)
- [25] <https://github.com/binarylogic/authlogic>