**Red Team**

**Stargazer**

**Automatic Telescope Control System**

**Developer’s Guide**

**Team Members:**

Jason Dunscombe

Rob Grmek

Robert Smith

**Instructor:**

**Youry Khmelevsky**

**Course:**

**COSC 471**

**Date:**

**January 18th, 2010**

# ****Revision History****

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 10/21/2009 | 1.0 | Document created; have draft of:   * Architecture * Use cases * Configuration Management * Installation Guide * Appendix: Glossary * Appendix: References | Robert, Rob |
| 10/28/2009 | 1.1 | Added more installation guide information. | Robert, Rob, Jason |
| 11/08/2009 | 1.2 | Added more installation guide information, Git/SCM information, and architectural diagram. | Rob |
| 11/13/2009 | 1.3 | Added class diagram, updated resources and use case diagram. | Rob |
| 11/20/2009 | 1.4 | Added Apache information. | Rob |
| 11/29/2009 | 1.5 | Jason and Robert created new diagrams for domain model, use case diagram and use case information.  Rob added new relational diagram for models in web application. | Jason, Robert, Rob |
| 1/18/2009 | 1.6 | Architecture updates to correspond with changes made to the system. | Rob |

# ****Table of Contents****

**Page**

|  |  |
| --- | --- |
| Introduction | 1 |
| Installation Guide | 2 |
| Configuration Management | 7 |
| Architecture | 9 |
| Use Case: Manages User Accounts | 14 |
| Use Case: Manage Scheduling of Telescope Position | 17 |
| Use Case: Compile Composite Image | 22 |
| Use Case: Notify Operator of New Objects | 24 |
| Appendix A: Glossary | 25 |
| Appendix B: References | 26 |
| Appendix C: Using Git | 28 |
| Appendix D: Using Ruby on Rails | 32 |
| Appendix E: Full Listing of Installed Ruby Gems | 34 |

# ****Developer’s Guide****

In this document contains the information for:

* System installation
* System backup and recovery
* System architecture
* Coding standards
* Versioning control system access

Virtual Server Information:

Windows Server 2003 R2:

|  |  |
| --- | --- |
| IP Address: | 10.1.144.60 |
| Name: | CIS470STAR09 |

Fedora 11: (For developing the Ruby on Rails web system which handles user input)

|  |  |
| --- | --- |
| Local IP Address: | 10.1.144.61 |
| Name: | CIS470STAR09F |
| Domain Name: | cis470Star09F.okanagan.bc.ca |
| Outside IP Address: | 142.23.93.226 |

Windows XP: (For developing the application for controlling the telescope)

|  |  |
| --- | --- |
| IP Address: | 10.1.144.62 |
| Username: | ocstudent |
| Password: | stargazer09 |

# Installation Guide

The following guides and references can be utilized to:

* Install the developmental tools and software needed to make modifications.
* Install the web system.
* Backing up the web system.
* Recovering the web system.

#### Installation Tasks Overview: (Web server)

* Install operating system: Fedora 11.
* Update packages.
* Set up environment:
  + Install Git.
  + Install Ruby and gems.
  + Install Ruby on Rails and any missing plug-ins.
* Install web server: Apache.
  + Setup configuration to handle SSL.
* Install Oracle client.
* Install the web application via Git.

#### Installation Tasks Overview: (Telescope server)

* Install operating system: Windows XP.
* Install database: Oracle 10g (Express Edition).
* Create database schemas:
  + Each schema corresponds to an environment mode in the Rails web application.
  + E.g. schema names – STARGAZER\_DEV, STARGAZER\_TEST, and STARGAZER\_PROD.
  + Make sure they have proper permissions, such as CREATE ANY JOB.
* Run database migrations from the web server.

#### Setting-up Environment: (Web server)

Reference used: <http://www.technetra.com/2009/04/22/howto-setting-up-ruby-on-rails-for-fedora-10-and-11/>

See if tools are installed:

which gcc make git

Installing SQLite: (the development database)

sudo yum install sqlite sqlite-devel

Installing Ruby: (the language)

sudo yum install ruby ruby-devel ruby-libs ruby-mode ruby-rdoc ruby-irb ruby-ri ruby-docs ruby-mysql ruby-sqlite3

Installing Ruby Gems: (packaging system for Ruby libraries)

wget -q http://rubyforge.org/frs/download.php/55066/rubygems-1.3.2.tgz

tar xzf rubygems-1.3.2.tgz

cd rubygems-1.3.2

sudo ruby setup.rb

Installing Ruby on Rails: (the framework)

sudo gem install rails

Installing Mongrel: (the development web server)

sudo gem install mongrel mongrel\_cluster

Installing JSON: (XML alternative)

sudo gem install json

#### Installing SQLite-Ruby gem:

sudo gem install sqlite3-ruby

#### Installing Rake gem:

sudo gem install rake

If you’re missing any dependencies run the following command or see Appendix E:

sudo rake gem:install

Also, it is important to remember to install the Oracle client on this Linux machine as the web application will be remotely connecting to the database residing on the Windows XP machine.

#### Installing the Web System:

This can simply be done using the Git clone command to copy the GitHub repository. Using this command will also create the directory that will hold the files.

In Apache’s configuration we used the directory: /var/www/html/rails/Stargazer as the application root. Therefore we go to the directory above it:

cd /var/www/html/rails/

Issue the Git clone command:

git clone [git@github.com:RedTeamCOSC470/Stargazer.git](mailto:git@github.com:RedTeamCOSC470/Stargazer.git)

Now all the project files will be retrieved and put into the newly created directory:

cd Stargazer

The database also needs to be created on the telescope server. Once the Oracle 10g XE (Express Edition) database is installed and the schemas for the different rails environments are created then run the database migration and start the web server.

#### System Backup:

System backup is achieved through regular commits both locally and to the remote GitHub repository.

See Appendix C for instructions for using Git.

#### System Recovery:

System recovery is achieved from “pulling” from the remote GitHub repository.

See Appendix C for instructions for using Git.

#### Using Apache:

For installing mod\_rails, the Apache module:

sudo gem install passenger

sudo yum install httpd-devel

sudo yum install apr-devel

sudo passenger-install-apache2-module

# then add the following lines to /etc/httpd/conf/httpd.conf:

LoadModule passenger\_module /usr/lib/ruby/gems/1.8/gems/passenger-2.2.7/ext/apache2/mod\_passenger.so

PassengerRoot /usr/lib/ruby/gems/1.8/gems/passenger-2.2.7

PassengerRuby /usr/bin/ruby

# also add the following lines to /etc/httpd/conf/httpd.conf:

NameVirtualHost:80

<virtualhost \*:80>  
 ServerName cis470Star09F.cis.okanagan.bc.ca  
 DocumentRoot /home/rgrmek/Stargazer/public  
 RailsEnv production  
 RailsBaseURI /home/rgrmek/Stargazer  
</virtualhost>

To start the web server, run as root:

sudo /etc/init.d/httpd start

To restart web server, run as root:

sudo /etc/init.d/httpd restart

To run on bootup:

chkconfig --levels 235 httpd on

When changes are made to the application in production environment:

To migrate development database to production:

rake db:migrate RAILS\_ENV=”production”

To restart a rails application:

touch tmp/restart.txt

#### Setting-up the Database: (Telescope server)

Download and install Oracle 10g XE (Express Edition) from the Oracle website onto the Windows XP machine.

Oracle XE has all the benefits of an advanced, feature-heavy database and is also free, with some limitations such as a maximum data storage capacity of 4 GB. Since no large objects (images, text files, etc) are being stored into the database, Oracle XE is a viable, free solution.

Schemas need to be created which represent the environment modes of the Rails application. There are three environment modes: development, testing, and production. Therefore, there needs to be one schema created to correspond to one mode.

Login as system and create the users: STARGAZER\_DEV, STARGAZER\_TEST, and STARGAZER\_PROD. For the purposes of this document and development of the application, we used simply the following password: stargazer09.

The following role privileges need to be given:

* connect
* resource

The following system privileges need to be given:

* alter session
* create any job
* create external job
* create job
* create sequence
* create session
* create synonym
* create view
* execute any class
* execute any program
* manage scheduler
* unlimited tablespace

Example SQL code:

*-- Create the user*

**create** **user** STARGAZER\_DEV

**default** **tablespace** **USERS**

**temporary** **tablespace** TEMP

**profile** **DEFAULT**

**quota** **unlimited** **on** **users**;

*-- Grant/Revoke role privileges*

**grant** **connect** **to** STARGAZER\_DEV;

**grant** **resource** **to** STARGAZER\_DEV;

*-- Grant/Revoke system privileges*

**grant** **alter** **session** **to** STARGAZER\_DEV;

**grant** **create** **any** job **to** STARGAZER\_DEV;

**grant** **create** **external** job **to** STARGAZER\_DEV;

**grant** **create** job **to** STARGAZER\_DEV;

**grant** **create** **sequence** **to** STARGAZER\_DEV;

**grant** **create** **session** **to** STARGAZER\_DEV;

**grant** **create** **synonym** **to** STARGAZER\_DEV;

**grant** **create** **view** **to** STARGAZER\_DEV;

**grant** **execute** **any** **class** **to** STARGAZER\_DEV;

**grant** **execute** **any** program **to** STARGAZER\_DEV;

**grant** **manage** scheduler **to** STARGAZER\_DEV;

**grant** **unlimited** **tablespace** **to** STARGAZER\_DEV;

After the schemas have been created, the database.yml file (path /Stargazer/config/database.yml) on the web server now may need to be updated. The database “stargazer” is the name given in the tns\_names.ora file in the Oracle client directory on the web server. If the schema names are different, then they may need to be changed in this database.yml configuration file.

An example from the web application’s database configuration file (database.yml):

development:

adapter: oracle\_enhanced

database: stargazer

username: STARGAZER\_DEV

password: stargazer09

encoding: utf8

cursor\_sharing: similar

Now, the database migrations can be run from the web server using the rake command in the Stargazer directory:

rake db:migrate

Likewise, run the db/seeds.rb file to create seed data (such as initial user of admin/stargazer09):

rake db:seed

The last step is to create the triggers which will create the scheduling jobs.

# Configuration Management

The purpose of this configuration management document is to mention versioning control methods, naming conventions, coding conventions, and methods for system backup and recovery.

#### Versioning Control:

All project files will be protected through the versioning control system ‘Git’.

All web application project files will be uploaded to the ‘Stargazer’ directory.

All documentation files will be uploaded to the ‘Documentation’ directory.

By using Git we avoid several problems such as team members simultaneously updating files as well as errors resulting in loss of data or system failure. In this case an older version would need to be retrieved and used.

Tutorials for using Git can be found in Appendix C.

#### Github Account:

Login at: <http://github.com/>

Username: RedTeamCOSC470

Password: stargazer09

#### Web Application Repository Information:

Name: Stargazer

Public Clone URL: <git://github.com/RedTeamCOSC470/Stargazer.git>

My Clone URL: [git@github.com:RedTeamCOSC470/Stargazer.git](http://github.com/RedTeamCOSC470/git@github.com:RedTeamCOSC470/Stargazer.git)

#### Documentation Repository Information:

Name: Documentation

Public Clone URL: [git://github.com/RedTeamCOSC470/Documentation.git](git://github.com/RedTeamCOSC470/Stargazer.git)

My Clone URL: [git@github.com:RedTeamCOSC470/Documentation.git](mailto:git@github.com:RedTeamCOSC470/Documentation.git)

#### Setting up Git and using Git:

See Appendix C.

#### Naming Conventions:

Each document that is created is named:

1. Prefix starting with ‘Stargazer’.
2. No spaces are used, underscores are used instead.
3. Only alphabetic characters and underscores are used.
4. Book title capitalization is used.

An example of a document name: Stargazer\_Vision.doc

#### Coding Standards:

The following code conventions are used:

Don’t use tab indenting, instead use 2 spaces.

Line length: maximum 80 characters.

Wrapping lines: break after comma and break after.

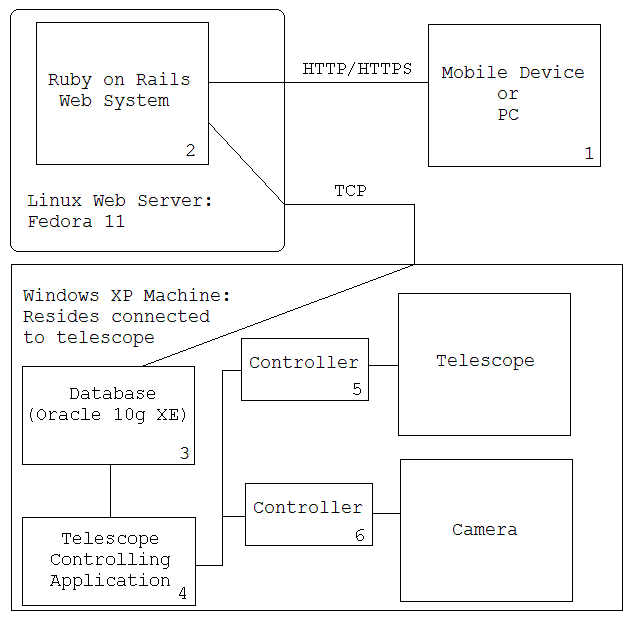
|  |  |  |
| --- | --- | --- |
| Syntax Type | Correct | Incorrect |
| Method Definitions | my\_method(my\_arg)  or  my\_method my\_arg | my\_method( my\_arg ) |
| Boolean Testing | && or || | AND or OR |

# Architecture

The system has three main functional components:

* scheduling the telescope to orient itself given the coordinates, location and date entered by the operator
* processing the images taken by the camera to compose a larger, high resolution image so it can compare the previous images with the new image
* if new objects are found, to notify the operator

### Architecture Diagram:



Legend:

1. Mobile Device and PC access:

* Allows users to connect to the Linux web server.
* Using a web browser, users will login to the web application and then be able to schedule telescope positions.
* Also, they can see low resolution composite images (approx 120x320 pixels). If clicked, the image will be enlarged to see the full, high-resolution image.

1. Ruby on Rails web application:

* Operating system: Fedora 11
* The web server handles client requests from user devices (#1).
* Has authentication; users will need to login in order to access the web application’s functionality.
* Will allow users to schedule telescope position; will display a form with the necessary inputs
* Will allow users to review scheduling logs to see previous history or make changes to future schedules.
* Will display latest composite images that have been captured.
* Handles email notifications.
* Receives composite images from the telescope server.

1. Database:

* Uses Oracle 10g XE (Express Edition).
* Also handles scheduling and jobs; a trigger exists on the scheduling table to create a job every time a new schedule is created. Once the schedule’s start time occurs, it will a run a batch file and passing in arguments to it. This batch file will call the telescope controlling application (#4) and run the program as a command line argument to move the telescope. The batch file will then copy the images to the web server and delete any local copies, then connect to the web server via SSH and run the rake task to “build” the Schedule’s images.

1. Telescope controlling program:

* Is hosted on the Windows XP machine which is connected to the telescope.
* Handles actual communication with the telescope and camera through their respective drivers. This allows for example: moving the telescope, taking pictures, etc.
* Handles image processing; will compile the many hundred smaller images into a large composite image.
* Handles image recognition; checks composite images with libraries to see if new objects have been identified.
* Written in C#.

1. Telescope controller:

* Has drivers which allows for the telescope-controlling program (#3) to communicate with the telescope.

1. Camera controller:

* Has drivers which allows for the telescope-controlling program program (#3) to communicate with the camera.

#### Hardware Components:

|  |  |
| --- | --- |
| Camera | Sony a900 DSLR |
| Telescope | Meade ETX-60AT-TC |

#### Development Environment: Web Server

|  |  |
| --- | --- |
| Operating System | Fedora 11 |
| Development Framework | Ruby on Rails |
| Unit Testing | Ruby’s built-in Unit Tests |
| Text Editor | GEdit, E, Emacs, VIM |
| Versioning Control System | Git |
| Web Server | Production: Apache  Development: Mongrel |
| Database | Production: Oracle 10g XE |

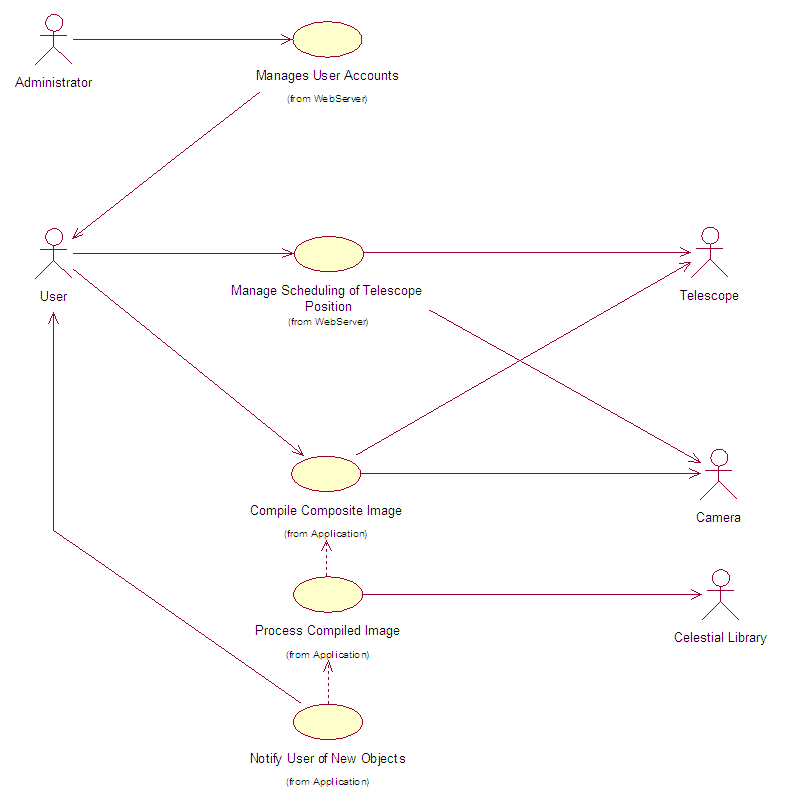
### Relational Diagram for Models in Web Application:

### C:\Documents and Settings\Robert Grmek\rails\documentation\Stargazer_Web_Models.png

### Domain Model:

### C:\Documents and Settings\Robert Grmek\rails\documentation\Stargazer_DomainModel.gif

### Use-Case Diagram:



# Use Case: ****Manages User Accounts (Creating Users)****

#### Brief Description:

This use case allows the administrator users to create new user accounts.

The main actor in this use case is the administrator.

#### Flow of Events:

The use case begins after the administrator has logged into the web site and clicked on Manage Users link at the top of the home page.

Basic Flow:

1. The administrator clicks on the New User link at the top of the user list.
2. The administrator is presented with an input form for entering user information.
3. The administrator fills in all information for this user by selecting the applicable fields and typing in the user’s information.
4. The administrator clicks the save button to create the user.

#### Special Requirements:

No special requirements have been specified for this use case at this time.

#### Pre-Conditions:

1. The administrator is logged in:

Before this use case begins the administrator has logged onto the web site.

#### Post-Conditions:

None

#### Extension Points:

There are no extension points associated with this use case.

# Use Case: ****Manages User Accounts (Editing Users)****

#### Brief Description:

This use case allows the administrator users to manage other user accounts and update their information. This allows an administrator to change the users user name, email address, password or their level or access to the site.

The main actor in this use case is the administrator.

#### Flow of Events:

The use case begins after the administrator has logged into the web site and clicked on Manage Users link at the top of the home page.

Basic Flow:

1. The administrator clicks on the Edit link of the user they wish to modify.
2. The administrator is presented with an input form for entering user information; with their current information already filled in.
3. The administrator changes any information they wish to change by selecting the applicable field and typing in new information.
4. The administrator clicks the save button to confirm the changes.

#### Special Requirements:

No special requirements have been specified for this use case at this time.

#### Pre-Conditions:

1. The administrator is logged in:

Before this use case begins the administrator has logged onto the web site.

1. There is a user to modify:

Before the use case can continue there must be another user in the system for the administrator to edit.

#### Post-Conditions:

None

#### Extension Points:

There are no extension points associated with this use case.

# Use Case: ****Manages User Accounts (Deleting Users)****

#### Brief Description:

This use case allows the administrator users to manage other user accounts and remove users.

The main actor in this use case is the administrator.

#### Flow of Events:

The use case begins after the administrator has logged into the web site and clicked on Manage Users link at the top of the home page.

Basic Flow:

1. The administrator clicks on the Delete link of the user they wish to remove.
2. The administrator is prompted to confirm the deletion of this user.
3. The user is deleted and the list of users is refreshed.

#### Special Requirements:

No special requirements have been specified for this use case at this time.

#### Pre-Conditions:

1. The administrator is logged in:

Before this use case begins the administrator has logged onto the web site.

1. There is a user to delete:

Before the use case can continue there must be another user in the system for the administrator to remove.

#### Post-Conditions:

None

#### Extension Points:

There are no extension points associated with this use case.

# Use Case: Manage ****Scheduling of Telescope Position****

# ****(Create New Schedule)****

#### Brief Description:

This use case allows the user to program the telescope with one or more specific times and dates and corresponding locations for observation. The system will instruct the telescope to point to the coordinates of that location at the scheduled time. The camera will then be instructed to take pictures of that area of the sky.

The main actor in this use case is the user.

#### Flow of Events:

The use case begins after the user has logged into the web site.

Basic Flow:

1. The user selects the New Schedule link from the top of the schedule.
2. The user is presented with an input form for entering information.
3. The user fills in all the fields.
4. The user clicks the save button to submit the observation information to the system.
5. The web site shows the newly created schedule with the information that was entered.

#### Special Requirements:

No special requirements have been specified for this use case at this time.

#### Pre-Conditions:

1. The user is logged in:

Before this use case begins the user has logged onto the web site.

#### Post-Conditions:

1. The telescope is told to move to the specified coordinates on the date provided by the user.
2. The camera begins taking pictures through the telescope once the telescope is in position.

#### Extension Points:

There are no extension points associated with this use case.

# Use Case: Manage ****Scheduling of Telescope Position****

# ****(Editing a Schedule)****

#### Brief Description:

This use case allows the user to modify a schedule to change the time and date or corresponding location for observation.

The main actor in this use case is the user.

#### Flow of Events:

The use case begins after the user has logged into the web site.

Basic Flow:

1. The user clicks on the Edit link of the schedule entry they wish to modify.
2. The user is presented with an input form for entering observation information; with the current information already filled in.
3. The user changes any information they wish to change by selecting the applicable field and typing in new information.
4. The user clicks the save button to confirm the changes.
5. The web site shows the modified schedule with the new information that was entered.

#### Special Requirements:

Schedules are only editable by the administrator or the user who created it.

#### Pre-Conditions:

1. The user is logged in:

Before this use case begins the user has logged onto the web site.

#### Post-Conditions:

1. The telescope is told to move to the specified coordinates on the date provided by the user.
2. The camera begins taking pictures through the telescope once the telescope is in position.

#### Extension Points:

There are no extension points associated with this use case.

# Use Case: Manage ****Scheduling of Telescope Position****

# ****(Deleting a Schedule)****

#### Brief Description:

This use case allows the user to delete a schedule.

The main actor in this use case is the user.

#### Flow of Events:

The use case begins after the user has logged into the web site.

Basic Flow:

1. The user clicks on the Delete link of the schedule entry they wish to remove.
2. The user is prompted to confirm the deletion of this entry.
3. The entry is deleted and the schedule list is refreshed.

#### Special Requirements:

Schedules are only removable by the administrator or the user who created it.

#### Pre-Conditions:

1. The user is logged in:

Before this use case begins the user has logged onto the web site.

#### Post-Conditions:

None

#### Extension Points:

There are no extension points associated with this use case.

# Use Case: Manage ****Scheduling of Telescope Position****

# ****(Show a Schedule)****

#### Brief Description:

This use case allows the user to view a schedule containing the time, date and corresponding location for observation.

The main actor in this use case is the user.

#### Flow of Events:

The use case begins after the user has logged into the web site.

Basic Flow:

1. The user clicks on the Show link of the schedule entry or entry’s date for the schedule they wish to view.
2. The web site shows the schedule with all the information.

#### Special Requirements:

No special requirements have been specified for this use case at this time.

#### Pre-Conditions:

1. The user is logged in:

Before this use case begins the user has logged onto the web site.

#### Post-Conditions:

None.

#### Extension Points:

There are no extension points associated with this use case.

# Use Case: Manage ****Scheduling of Telescope Position****

# ****(Viewing a Schedule’s images)****

#### Brief Description:

This use case allows the user to modify a schedule to change the time and date or corresponding location for observation.

The main actor in this use case is the user.

#### Flow of Events:

The use case begins after the user has logged into the web site.

Basic Flow:

1. The user clicks on the Show link of the schedule entry or entry’s date for the schedule they wish to view.
2. The user clicks on the List Images link on the Show Schedule page.
3. A list of all the images associated with that schedule will be shown.

#### Special Requirements:

Images will only be shown if the scheduled observation has been completed.

#### Pre-Conditions:

1. The user is logged in:

Before this use case begins the user has logged onto the web site.

#### Post-Conditions:

None.

#### Extension Points:

There are no extension points associated with this use case.

# ****Use Case: Compile Composite Image****

#### Brief Description:

This use case allows the images captured from the camera to be compiled into a composite image with a larger resolution.

The main actor in this use case is the user.

#### Flow of Events:

The use case begins after the camera has finished taking a set of pictures.

Basic Flow:

1. The pictures are stored on the hard drive of the server.
2. All pictures are combined together using an algorithm to produce a higher resolution picture.
3. The high resolution composite image is stored with the original images for later review.

#### Special Requirements:

The user must start the image compiler.

#### Pre-Conditions:

1. The camera has finished taking pictures:

Before this use case begins the camera must have completed all pictures for the specified observation period.

#### Post-Conditions:

There are no post-conditions associated with this use case.

#### Extension Points:

After the operation is completed the Process Compiled Images use case is started. (See ‘Use Case: Process Compiled Images’).

# ****Use Case: Process Compiled Images****

#### Brief Description:

This use case allows the images to be processed in an attempt to find new objects. If any new objects are found, the operator is notified.

This use case is an extension of the Compile Composite Image use case.

#### Flow of Events:

The use case begins after the camera has finished taking a set of pictures.

Basic Flow:

1. After the composite image is compiled, the composite, high resolution picture is then compared to the Celestial Library and older pictures from similar locations to identify any previously unrecorded objects.

#### Special Requirements:

No special requirements have been specified for this use case at this time.

#### Pre-Conditions:

1. The composite image has been compiled:

Before this use case begins the image has been compiled.

#### Post-Conditions:

There are no post-conditions associated with this use case.

#### Extension Points:

1. If new object(s) are found:

If a new object is found after image processing, then the operator is sent a notification. (See ‘Use Case: Notifies User of New Objects’).

# ****Use Case: Notify User of New Objects****

#### Brief Description:

This use case informs the user of any new objects detected while processing the images in the Compiles Composite Image use case.

The main actor in this use case is the operator.

#### Flow of Events:

The use case begins after a new object has been detected while compiling the composite image.

Basic Flow:

1. The high resolution picture containing the previously unidentified object is stored on the hard drive of the server.
2. A notice is placed on the web site alerting the operator of the new object.
3. The picture of the object is shown to the operator as well as some indicator of where the new object is located.
4. The user clicks the save button to store the high resolution image to a separate location.

#### Special Requirements:

No special requirements have been specified for this use case at this time.

#### Pre-Conditions:

1. The system has found an unidentified object in a picture:

Before this use case begins an unidentified object must have been identified by the system.

#### Post-Conditions:

There are no post-conditions associated with this use case.

#### Extension Points:

There are no extension points associated with this use case.

# Appendix A: Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Git | The versioning control system the project team uses. |
| Ruby on Rails | Framework used for rapid web development of a dynamic web system. |
| Stargazer | Internal project name for the automated telescope control system. |

# Appendix B: References

For setting up a Ruby on Rails development environment on Fedora 11, see here:

Technetra:

<http://www.technetra.com/2009/04/22/howto-setting-up-ruby-on-rails-for-fedora-10-and-11/>

For setting up a Ruby on Rails development environment on Windows, see here:

Akitaonrails – Setting up the Best Rails Environment on Windows:

<http://www.akitaonrails.com/2009/1/13/the-best-environment-for-rails-on-windows>

O’Reilly - Rolling with Ruby on Rails: (another installation guide)

<http://oreilly.com/ruby/archive/rails.html>

Oracle guide for installing Rails with Oracle with a practice tutorial for creating a simple application:

<http://www.oracle.com/technology/pub/articles/haefel-oracle-ruby.html>

Using Git on Windows:

<http://beans.seartipy.com/2008/12/09/setting-up-ruby-on-rails-projects-with-git-and-github/>

Other resources:

Official Ruby on Rails guides:

<http://guides.rubyonrails.org>

Tutorials Point guides:

<http://www.tutorialspoint.com/ruby-on-rails-2.1/index.htm>

Rails Video Tutorials:

<http://railscasts.com>

<http://asciicasts.com/>

Distributing Ruby Applications:

<http://www.erikveen.dds.nl/distributingrubyapplications/rails.html>

Git and Rails:

<http://harryseldon.thinkosphere.com/2009/01/14/git-and-rails-a-detailed-tutorial-including-plugins-submodules-development-and-production>

Validates\_Timeliness Plugin Information:

<http://www.railslodge.com/plugins/1160-validates-timeliness>

Authlogic Plugin Tutorial:

<http://github.com/binarylogic/authlogic_example>

<http://asciicasts.com/episodes/160-authlogic>

Railroad Plugin Information:

<http://www.railroad.rubyforge.org/>

Understanding the Ruby language:

<http://www.robertsosinski.com/2008/12/21/understanding-ruby-blocks-procs-and-lambdas/>

<http://rubylearning.com/satishtalim/tutorial.html>

<http://www.meshplex.org/wiki/Ruby/Ruby_on_Rails_programming_tutorials>

<http://www-users.math.umd.edu/~dcarrera/ruby/0.3/chp_01/first_steps.html>

<http://www.tekniqal.com/>

# Appendix C: Using Git

This set up takes place after a Github account has been created and a repository has been created. Currently the information for the account and repository are:

#### Github Account:

Login at: <http://github.com/>

Username: RedTeamCOSC470

Password: stargazer09

#### Repository Information:

Name: Stargazer

Public Clone URL: <git://github.com/RedTeamCOSC470/Stargazer.git>

My Clone URL: [git@github.com:RedTeamCOSC470/Stargazer.git](http://github.com/RedTeamCOSC470/git@github.com:RedTeamCOSC470/Stargazer.git)

Information for setting up Git: (taken from GitHub)

#### Global setup:

Initial setup of Git on a computer.

|  |
| --- |
| # First, download and install Git (see installation guide)  # Add configuration information:  git config --global user.name "Your Name"  git config --global user.email "Your Email"  # Then, add your public key (see below) |

#### Adding a public key:

A public key needs to be added to validate the computer as the owner of the repository so commits can be made.

|  |
| --- |
| # Generate a public key:  ssh-keygen  # Use the ‘cat’ command on the file that was created in  # directory such as:  cat ~/.ssh/id\_rsa.pub  # Then, add the public key to the github account under:  # Account Settings -> SSH Public Keys |

#### Cloning the Repository:

Do this to recreate the directory structure with all project files.

Do this if the local Git repository has not already been made.

Can be used after the development environment is setup and freshly installed.

|  |
| --- |
| git clone [git@github.com:RedTeamCOSC470/Stargazer.git](mailto:git@github.com:RedTeamCOSC470/Stargazer.git) |

#### Other commands:

|  |
| --- |
| # create a new local repository:  git init    # add a file to the staging area:  # in other words, to setup file(s) before a local commit  git add [filename]  # to add all files use this:  git add .  # check status of the staging area files:  git status  # commit the staging area files to the local repository  # using the –m switch includes a message:  git commit –m “This is a commit message”  # show commits:  git log  # change username for only the local git repository:  git config user.name “[User Name]”  # adding a new remote destination called “origin”:  # a remote destination in this case is our  # GitHub repository  git remote add origin [git@github.com:RedTeamCOSC470/Stargazer.git](mailto:git@github.com:RedTeamCOSC470/Stargazer.git)  # list remote destinations:  git remote  # pushing the local committed files to the remote destination  # from the master branch “origin”:  # in other words, this is to commit the files to the  # GitHub repository.  git push origin master  # check if the local commit is not already pushed to  # the remote destination:  git log --pretty=oneline master...origin/master  # update the local GitHub origin master branch:  git fetch origin  # merge GitHub’s remote changes into the local master branch:  git pull origin master  # in the project root, create a file for Git to use  # to ignore certain files:  vim .gitinore  # then write the filename into the file  [filename]\*  :wq!  #then add the file  git add .gitignore  # see changes since files have last been stages:  # (file is added, but not committed)  git diff  # remove any changes in the working directory:  git checkout -- .  # unstage the file, but still have changes to the file:  git reset HEAD [filename]  # start from the last commit; remove everything:  git reset -hard    # create a branch:  git branch [branchname]  # list branches:  git branch  # checkout 1 commit back  git checkout HEAD^  git checkout HEAD~1  # 2 commit backs, etc  git checkout HEAD^^  git checkout HEAD~2  # go back to master  git checkout master  # use gitignore command to ignore some files  touch .gitignore  # add files you wish git to ignore (i.e. doc/api)  doc/api  # create a branch and checkout that branch  git checkout –b manage\_users  # or  git branch manage\_users  git checkout manage\_users    # merge the changes to master branch and delete old branch  git checkout master  git merge manage\_users  git branch –D manage\_users |

# Appendix D: Using Ruby on Rails

Run the rails command to create the initial files:

rails stargazer

Create the database:

rake db:create

Start the Mongrel web server:

script/server

To see if all gems have been installed:

rake gem:install

To list scripts:

script/generate

To drop tables:

rake db:version

rake db:migrate:down:version

To create the tables:

rake db:migrate

To generate a migration file:

script/generate migration

e.g.

script/generate migration addRightAscensionToSchedule right\_ascension:time

To generate documentation:

rake doc:app

To run tests:

rake test

rake test:units

rake test:functional

To migrate development database to production:

rake db:migrate RAILS\_ENV=”production”

To restart a rails application:

touch tmp/restart.txt

To go to console in production environment:

script/console production

#### Using Plugins: Annotate

This plugin will annotate models and unit tests with the table data for that object.

To run simply issue the command:

annotate

#### Using Plugins: yUMLmeRails

This plugin will create a class diagram from the models in the project directory. For help, see: <http://github.com/nelsonsilva/yUMLmeRails/>

To run simply issue the command:

Rake yUMLmeRails:download

#### Using Plugins: RailRoad

To create modeling diagrams using RailRoad: (see: <http://railroad.rubyforge.org/>)

Note: Run RailRoad on the Rails application's root directory.

Produces a models diagram to the file '**models.dot**':

rake doc:diagrams

Controller diagram in **PNG** format:

railroad -C | neato -Tpng > controllers.png

Model diagram in **PNG** format:

railroad -M | dot -Tpng > models.png

# Appendix E: Full Listing of Installed Ruby Gems

Full listing of installed gems:

actionmailer (2.3.4)

actionpack (2.3.4)

activerecord (2.3.4)

activerecord-oracle\_enhanced-adapter (1.2.3)

activeresource (2.3.4)

activesupport (2.3.5, 2.3.4)

annotate (2.0.2)

archive-tar-minitar (0.5.2)

arrayfields (4.7.4)

authlogic (2.1.2)

brynary-webrat (0.4.0)

builder (2.1.2)

calendar\_date\_select (1.15)

capistrano (2.5.9)

cgi\_multipart\_eof\_fix (2.5.0)

cheat (1.2.1)

chronic (0.2.3)

chronic\_duration (0.7.5)

color (1.4.0)

columnize (0.3.1)

cucumber (0.4.2)

diff-lcs (1.1.2)

dougsko-apod (0.1.3)

fastercsv (1.5.0, 1.2.3)

fattr (2.1.0)

fxri (0.3.6)

fxruby (1.6.16)

gem\_plugin (0.2.3)

highline (1.5.1)

hirb (0.2.8)

hoe (2.3.3)

hpricot (0.6.164)

httpclient (2.1.5.2)

image\_science (1.2.1)

json (1.1.9)

json\_pure (1.1.9)

libxml-ruby (1.1.3)

linecache (0.43)

log4r (1.0.5)

main (4.0.0)

mechanize (0.9.3)

mini\_magick (1.2.5)

mongrel (1.1.5)

mongrel\_service (0.3.4)

mysql (2.8.1)

net-scp (1.0.2)

net-sftp (2.0.2)

net-ssh (2.0.15)

net-ssh-gateway (1.0.1)

newgem (1.5.2)

nifty-generators (0.3.0)

nokogiri (1.3.3)

oniguruma (1.1.0)

open4 (1.0.1)

paperclip (2.3.1.1)

pdf-writer (1.1.8)

piston (2.0.6)

plist (3.0.0)

polyglot (0.2.9)

ptools (1.1.6)

rack (1.0.1)

railroad (0.5.0)

rails (2.3.4)

rake (0.8.7, 0.8.1)

RedCloth (4.2.2)

rmagick (2.7.1)

rspec (1.2.9)

rspec-rails (1.2.9)

rubigen (1.5.2)

ruby-debug (0.10.3)

ruby-debug-base (0.10.3)

ruby-oci8 (2.0.3)

ruby-opengl (0.60.0)

ruby-plsql (0.4.1)

ruby-postgres (0.7.1.2006.04.06)

rubyforge (2.0.3)

rubygems-update (1.3.5)

ruport (1.6.1)

sashimi (0.2.2)

sqlite3-ruby (1.2.3)

syntax (1.0.0)

term-ansicolor (1.0.4)

test-unit (2.0.1)

textpow (0.10.1)

transaction-simple (1.4.0)

treetop (1.4.2)

ultraviolet (0.10.2)

validates\_timeliness (2.2.2)

web-app-theme (0.4.0)

webrat (0.5.3)

whenever (0.4.0)

win32-api (1.2.1, 1.2.0)

win32-clipboard (0.4.4)

win32-dir (0.3.2)

win32-eventlog (0.5.0)

win32-file (0.5.5)

win32-file-stat (1.3.1)

win32-process (0.5.9)

win32-sapi (0.1.4)

win32-service (0.5.2)

win32-sound (0.4.1)

win32console (1.2.0)

windows-api (0.2.4)

windows-pr (0.9.3)

ZenTest (4.1.4)