I. Implementation Requirements

A. Critical Requirements (Needed for the system to work properly)

- Better implementation of file uploads and error-checking. Honestly, this is easier to do with Redmine's built in Attachment Model features. I have already starting setting this up in the second partial form used by the JobAttachment Model (app/views/job_attachments/_form_02.rhtml) and have set the JobAttachment association "acts_as_attachable". Please refer to Redmine's Document and File models, controllers, and views as reference.

- Implement code for Job Application model, controller, and view. Job Applications are dependent on the nested forms I was working to complete in the Job model. Once the forms for creating the new Job work properly, implementation of Job Applications should be fairly straightforward.

- Login/Registration Implementation. Users need to be able to register as a user in the Redmine system in order to secure a user_id. The Applicant model is already set in an association with the User model used by Redmine. Be sure to uncomment this code

when ready to implement login. You may want to see if other plugins either extend or make use of a belongs_to relationship with Redmine's User model.

- Modify routing so that references to bad links don't result in a Redmine system error

- Complete Implementation of the new :has_and_belongs_to_many join relationship shared between the Apptracker and Applicant models. Applicants will no longer display properly since Apptrackers and Applicants are connected via a join table. Code must be modified in the Applicants controller. Please see my Dia MVC diagrams 13 and 14 for a point of visual reference.

- Referrers: There are some questions as to whether or not 1) a referrer should need to log in to the system in order to complete a reference, 2) if a referrer should receive an e-mail with a custom link that will allow them to complete a reference for an applicant, or 3) an applicant will need to secure and attach references beforehand (a temporary workaround for this version of the application tracker plugin). Once this is decided on, implementation of the JobApplicationReferrer model can be done.

- Permissions: Permissions throughout the plugin should be checked before going live with the plugin. I have before_filter plugin permissions set for controller action access by admins, as well as roles and permissions set via the plugin's init.rb file. However, as implementation continues, permissions will need to be reviewed and added.

B. Non-critical Requirements

- Modification of Migration 011 (test data injection) for filling in the new join table 'applicants_apptrackers' with appropriate test data

- Integration with Redmine's Email System

- Logging

II. Working with Redmine Plugins

A. Migrating Plugins

1) Copy the redmine_apptracker directory to redmine's app/vendors/plugins/ directory

2) Before migrating the plugins, it's a good idea to backup the redmine database. For mysql, use the following command:

bash> **mysqldump -p -u root *database_name* > *path_and_filename*.sql**

Recovery of the database can be done via the following command:
bash> **mysql -u root -p *database_name* < *path_and_filename*.sql**

3) Migrate the plugins:

bash> **rake db:migrate:plugins RAILS_ENV=production**

B. Setting Plugin Permissions

1. init.rb

A plugin's permissions can be set in the plugin's init.rb file, found in the plugin's root folder. Permissions can then be turned off and on for user roles via Redmine's Administration -> Roles and Permissions -> Permissions Report Interface. Be sure to allow viewing of Apptrackers and Jobs for anonymous and nonmember roles.

2. before_filter

Controller-level access can be set per action via before_filters. I currently make use of the :require_admin feature for setting admin access to actions.

III. Programming Tactics Used

A. Paramaters & Hidden Form Fields

I had initially kept track of data using Rails' session hash, but this allowed for concurrency problems if a user had multiple browser tabs/windows open. As a replacement to storing needed info in session parameters, I started to pass data as parameters at the controller and view levels. These parameters could also be stored as hidden fields within forms and accessed later by the controller. Although I have done my best to eliminate old session data and add submit proper parameters to every view and form that require extra data, be on the lookout for errors such as "Can't find [model name] without ID".

B. User Access

Although Redmine's login process still needs to be integrated into the application tracker, I have been careful to understand that there are three major levels of separation between users:

- administrative level
- logged-in user (a user that exists in the Redmine system)
- anonymous user (not logged in)

Each category of user implies different access settings, all of which have enforced via 1) the plugin's init.rb file & Redmine's Role & Permissions settings (in the Administration section), 2) controller-level 'before_filters', and 3) via view-level enforcement using if/elsif/else control statements. It may be worth looking into other Redmine plugins to see what the generally accepted method for setting access permission may be; I'm fairly certain that my view-level control statements are not the most popular method for access enforcement.

C. Partials

1. Tabbed Interface

Within my plugin's 'views 'folder, I created a 'shared' folder for any views that one or more models will take advantage of. This is where I store the tabbed interface file (_tabs.html.erb))used by admins, registered users, and anonymous users for getting arround the application tracker plugin. View-level access is set here for each category of user.

2. Tabular Listing of Objects

Each model has a partial level view in it's 'views' folder that provides a listing of related instances to that particular model. I use these partials in various parts of the application for showing a particular object's connection to other objects (for example, to show the jobs related to a particular apptracker).

IV. Other Documentation

A. Workflow

2010.08.18
bugfix: only apptrackers associated with a particular project now show up at the apptracker index page; passed the project id to the Apptracker module via the Apptracker tab (set in the Apptracker's init.rb file)
bugfix: session data set properly to reflect project_id, apptracker_id, and job_id

2010.08.20

- upgrade: upgraded to Redmine 1.0
- bugfix: repaired plugin migration problems
- added JobApplicationTemplate model and controller
- added timestamps to all tables
- repaired controllers: fixed external table references to make use of more association-like code
Brainstorming:
should application materials belong to an apptracker? instead, i think they would be better associated with a job application

2010.08.25
- fixed validation for applicants
- have home_address info set for validation; dorm/apartment address not validated
- added gpa, gpa_scale, and academic_other_notes fields
- added drop-down for gender
- created a migration containing test data

2010.08.26
- Created partials that are provided on-screen dependent on whether the user is anonymous, an applicant, or an admin
- Added :before_filter that sets permissions for various user types
- Moved Apptracker status from boolean type to string type (more descriptive)

2010.08.27
- Continued to add and clean up partials
- Finished Referral Model, Controller, and Views

2010.09.03
- session data removed/replaced
- jobs model updated (fused data from job application template model into job model)

2010.09.04
- learned how to pass :locals = {} symbol to partial renders
- implementation of job_application model, controllers, and views
- removed remaining job application template remnants

2010.09.09
- updated config/routes.rb to include job applications
- more modification of any leftover session data

2010.09.12
- updated DIA diagram; added CustomFields, JobAttachments, JobApplicationReferrals, JobApplicationCustomFields, and JobApplicationMaterials tables
- updated the following models: Job,
- deleted the ApplicationMaterial and Referral Models
- created/updated migrations
- added new test data for testing new models

2010.09.14
- started documentation of required implementation

2010.09.18
- code refactorization
- addition of new FIXME, TODO, and NOTE comments
- creation of plugin's docs folder
- addition of documentation to plugin's docs folder

2010.09.20
- View bug fixes
- Apptracker controller and view bug fixes
- More work on nested forms
- updated Dia MVC diagram to version 13

2010.09.21 - 2010.09.22
- added status to apptracker listing (admin view)
- disallowed nonadmins from seeing any inactive apptrackers
- corrected validation for apptracker form
- fixed 'back' link in apptracker edit form
- disallowed nonadmins from seeing any inactive jobs (anonymous and registered users can still see 'active' and 'filled' positions)
- added 'required data' asterisks to forms (indicating fields that will be validated)
- removed leftover session data from a few remaining view headers
- started fix of applicant migrations and views due to the new :has_and_belongs_to_many relationships between the Apptracker and Applicant models
- created a new migration for the applicants_apptrackers join table
- more documentation added, Google doc reference created for Anita
- updated Dia MVC diagram to version 14
- further FIXME and TODO comment updates

B. TODO Commenting

Note: Redmine's built-in rake:notes task only works with particular directories within Redmine. Instead, I made use of the grep-fu gem to find comments posted throughout plugin code.

```
./config/routes.rb:9:
        # TODO phase out the application materials model
./config/routes.rb:13:
        # TODO phase out the referrer model
./app/controllers/job_application_referrals_controller.rb:1:
        # TODO implement this controller
./app/controllers/job_application_custom_fields_controller.rb:1:
        # TODO build this controller
```

./app/controllers/application_materials_controller.rb:6:
	# TODO Phase this controller out
./app/controllers/application_materials_controller.rb:9:
	# TODO USER VIEW: add a section that finds application materials for a particular user
./app/controllers/application_materials_controller.rb:45:
	# TODO move some of the following values as hidden fields in the form
./app/controllers/application_materials_controller.rb:76:
	# TODO Add RESTful resource route to routes.rb
./app/controllers/application_materials_controller.rb:81:
	# TODO Check if the Berkman Apache server can make use of the :x_sendfile option
./app/controllers/applicants_controller.rb:22:
	# TODO uncomment this after job applications are implemented
./app/controllers/job_application_materials_controller.rb:1:
	# TODO implemente this controller
./app/controllers/job_attachments_controller.rb:1:
	# TODO Implement this controller
./app/controllers/job_custom_fields_controller.rb:1:
	# TODO implement this controller

	Binary file ./app/controllers/.job_applications_controller.rb.swo matches
./app/controllers/job_applications_controller.rb:3:
	# TODO make sure an applicant cannot add a new job application to a job they have already applied to
./app/controllers/job_applications_controller.rb:4:
	# TODO find out how to use authentication for allowing access to :new and :edit actions
./app/controllers/job_applications_controller.rb:10:
	# TODO establish calling page in order to return proper search results (from job scope or applicant scope)
./app/controllers/referrers_controller.rb:1:
	# TODO phase out this controller
./app/controllers/referrers_controller.rb:5:
	# TODO find out how to use authentication for allowing access to :new and :edit actions
./app/views/jobs/_form.html.erb:41:
	# TODO Provide setting of application material types
./app/views/jobs/show.html.erb:26:
	# TODO Connect the anonymous user login to Redmine's login system
./app/models/job.rb:11:
	# TODO if necessary, modify :reject_if code for more advanaced error checking
./app/models/job.rb:20:
	# TODO convert these values into variables that can be set from a settings page within Redmine
./app/models/job_application.rb:12:
	# TODO incorporate reject_if code
./app/models/job_application.rb:18:
	# TODO implement validations
./app/models/job_application.rb:21:
	# TODO convert these values into variables that can be set from a settings page within Redmine
./app/models/application_material.rb:2:
	# TODO PHASE OUT THIS MODEL
./app/models/applicant.rb:21:
	# TODO fix regex validation fields
./app/models/applicant.rb:28:
	# TODO validate uniqueness of login name once plugin is integrated into Redmine's user login/authentication
./app/models/applicant.rb:32:
	# TODO convert these values into variables that can be set from a settings page within Redmine
./app/models/job_application_material.rb:3:
	# TODO implement saving of files using Redmine's Attachment model
./app/models/job_application_material.rb:9:

TODO implement validation
./app/models/job_application_material.rb:12:
TODO convert these values into variables that can be set from a settings page within Redmine
./app/models/job_application_material.rb:16:
TODO correct storage path
./app/models/job_application_material.rb:17:
TODO serialize filename
./app/models/job_application_material.rb:18:
TODO Provide warning before overwriting of a file with the same name
./app/models/job_application_material.rb:19:
TODO Sanitize filename
./app/models/apptracker.rb:12:
TODO convert these values into variables that can be set from a settings page within Redmine
./app/models/job_attachment.rb:6:
TODO Does a delete permission need to be added to the acts_as_attachable association?
./app/models/job_attachment.rb:12:
TODO Should I implement internal file renaming for identical filenames uploaded by different applicants?
./db/migrate/011_add_test_data.rb:3:
TODO remove all references to my e-mail address
./db/migrate/011_add_test_data.rb:4:
TODO remove my .gitignore reference to this file
./db/migrate/011_add_test_data.rb:5:
TODO add filenames for files
./db/migrate/003_create_applicants.rb:2:
TODO look to see if some fields are better set as integer types (phone number, area codes, etc.)
./init.rb:2:
TODO find out if all RESTful actions need a matching permission

C. FIXME Commenting

./config/routes.rb:3:
FIXME Add routes for catching bad links when 1) a link is misspelled, and 2) when a page cannot be built due to a lack of needed params[]
./app/controllers/applicants_controller.rb:9:
FIXME Update this to reflect join condition with the Apptracker Model
./app/views/job_applications/new.html.erb:6:
FIXME Implement an Ajax interface that allows the user to input an :application_material_count, generating a data input table with a row count matching the value found in :application_material_count
./app/models/job.rb:4:
FIXME Should job_applications to a job be destroyed if a job is deleted?
./app/models/applicant.rb:10:
FIXME uncomment this when starting to implement Redmine login functionality
./app/models/applicant.rb:15:
FIXME Validations that fail are coming up with a 'translation missing: en' error