

CSPP 52553 Web Development

Winter Quarter, 2013

Department of Computer Science
University of Chicago

Final Project Requirements

Every student is required to turn in a Ruby on Rails web application, deployed and accessible from a standard web browser over the public internet.

Deployments are due by the end of the last day of class.

Pair-programming is recommended, and each student will share the same grade. However, you may work on your own if you prefer.

Project Templates

You can choose from one of these idea templates. You may suggest your own project, but your project idea must be approved no later than Week 7.

The three templates are:

1. **Social Sharing Site.** Like Pinterest. Users can sign up for accounts, submit content, content is categorized or tagged, they "follow" other users, and can comment on other user's submissions.
2. **Content Management System.** Like Tumblr, or a newspaper publication service. Users sign up for an account as a publisher or reader. A publisher can post articles of various media types: text, images, videos, and maps, and are organized into sections based on a category, tag, publish date, or some other criteria. Readers can post comments anonymously or under their account.
3. **Event Organizer.** Like Evite or Meetup.com. A way to invite groups of people to events at various locations.

Grading Criteria

Each Core and Elective Component is assessed based on the following objective criteria, in decreasing order of importance:

1. Working code (zero defects)
2. Optimal use of built-in Ruby classes and algorithms
3. Proper application of Rails classes and methods
4. Adherence to agile principles such as DRY, YAGNI, and SRP
5. Adoption of Ruby idiomatic style

(70%) Core Components

All projects must demonstrate mastery over these core elements:

1. Demonstrate best-practice use of Rails MVC architecture
2. Use all four HTTP methods at least once
3. Allow users to sign up, sign in, and sign out using cookie-based sessions.
4. User accounts must use industry-strength password encryption or OAuth-compliant authentication data exchange.
3. Contain at least six models
4. Contain at least one many-to-many association
5. Use at least three model callbacks
6. Use at least three validation rules
7. Maintain HTTP session state (e.g. user authentication)
8. Use at least one controller filter
9. Must be deployed to a live server that we can get to over the public Internet
10. Use Javascript to modify the DOM of at least one page in response to a user action
11. User Javascript to make an asynchronous http request (can be combined with #10)
12. Populate the database via seed data

(30%) Elective Components

1. Use and demonstration of understanding of **two or more** of the following:
 - 1.1. Using Rails' built-in Test-Driven Development (TDD) tools. (Scaffold-generated tests are ok but don't count for much).
 - 1.2. Advanced ActiveRecord (e.g. Scopes, Polymorphic Associations, STI)
 - 1.3. Intelligent use Javascript or CoffeeScript to create a responsive user interface
 - 1.4. Using ActionMailer to send email as a response to user action
 - 1.5. Mashup of data from an external API (Facebook, Instagram, Google Maps, etc.)
2. Creativity in execution of application idea
3. Code readability and organization
4. Application performance through use of efficient ActiveRecord techniques and caching