FINAL PROJECT DESCRIPTION

For the final project you will use your knowledge of front and back-end web development to produce an awesome web application that can be used by friends, family or any of the other billions of people who use the Internet. The type of web applications you create is your choice, but don't worry; you have some time to think about what would like to do.

The objective of this project is to:

- To demonstrate understanding of all topics covered during this course.
- To apply knowledge gained during this course by building a web application from the ground up.
- To use back-end and front-end web development skills to create an efficient website that is compatible with many browsers.

The final encapsulates everything learned in this course. We want to make sure you are successful in your venture, so to stay on track we suggest you take our milestone suggestions seriously. In addition, we will help define scope and guide you through the process.

CORE REQUIREMENTS

These features must be included in your project.

- **Styled with CSS.** Include CSS to style your website.
- **Hosted.** No need to pay for hosting services, you can use GitHub and Heroku.
- **JavaScript.**
    - Make sure you use event handling as necessary to monitor and respond to user behavior.
    - Use JavaScript function to logically group your code.
- **User Generated Content.** Your web application must have user-generated content that can be discovered by other users. It will maintain user accounts and profiles.
- **Security Features.** Login credentials must be validated before users can gain access to your application. Your site should have public and private content.
- **Ruby Gems.** Use a rubygem (not the Rails defaults, obviously they don't count). Some ideas to look into: will_paginate, client_side_validations, simple-navigation, cells, formtastic, show_for, devise and Koala.

- **Handles invalid data.** Forms in your application should validate data and handle incorrect inputs. Validate sign up information, verify valid email addresses and secure passwords.
- **Partials.** Use partials to DRY (Don't Repeat Yourself) up your views.

NICE TO HAVES

Include at least 2 of the following in your project

- **Sprite Images**. Make fetching images faster. Use sprites instead of multiple separate image files.
- **Interactive custom animations.** Make your site visually interesting. Include animations and interactivity. Practice good design and don't over do it.
- **CSS3 techniques.** Include at least two of the CSS3 properties introduced during the course along with the proper vendor prefixes: border-radius, text-shadow, box-shadow, and opacity, RGBA.
- **Namespace.** Implement your own application namespace. .
- **Responsive:** Create and use media queries to make your site respond to various screen sizes.

MILESTONES

Week 4	Wireframes
Week 5	Draft html/css code
Week 11	Project Proposal and List of Classes
Week 12	Model Associations Diagram
Week 13	List Views, Models and Associations
Week 16	Project due

DELIVERABLES

Link to Heroku hosted project and/or link to source code on GitHub.

CHALLENGE

Up for a challenge? Test your understanding of JS and use some external libraries.

- **underscore.js.:** It will help you with some of the boilerplate js functions you may have to write from scratch. In general, try not to reinvent the wheel, but also understand exactly why you are using something.
- **Authentication API:** Using an external api, allow signup to your application
    - Facebook Authentication
    - OmniAuth
    - Devise and OmniAuth (cost $$)
- **Gems:** Implement one of the following gems
    - bdd test suite with cucumber.
    - Sass
    - Compass
- Integrate Ajax into web site to load pages faster.

BEST PRACTICES.

Your instructor and T.As will provide feedback on how well you execute best practices. Even though it is not part of the requirements, you should keep these in mind.

- **Clean And Readable Code**. Instructors and T.As should be able to read and follow code easily. Maintain clean and readable code including: consistent indentation, code commenting (e.g. when closing tags, demarcating sections of code, describing possibly ambiguous code choices) and use proper and consistent naming conventions.
- **Correct use of classes and ID.** Use classes for identifying a type of element, and IDs for identifying a specific unique element on the page.
- **Search Engine Optimization (SEO).** Implement SEO best practices related to HTML markup, and content optimization.
- **Avoid deprecated tags.** Uses best practices and build using supported HTML and CSS tags.
- **Test It.** Go through test cases to make sure models work correctly.
- **RESTful**. Implement some non-default (defaults are: add, create, update, destroy) RESTful routes to demonstrate your understanding of RESTful routing in Rails.

GRADING

A project is considered satisfactory if it meets all core requirements and includes at least 2 nice to haves. Completing the challenge in addition to the requirements would be above satisfactory. Sample projects can be found here.