Company: Agile Developer, Inc.
Project Coordinator: Venkat Subramaniam
Communication: We'll use a web-form/Wiki to communicate. Also, when needed, we'll communicate using Google hangout.

The site agilelearner.com is run by Agile Developer, Inc. (a training and mentoring company). The site offers videos on various programming topics. It's a commercial site, but offers free membership to user group leader and students.

Project Details:

This project will add a few new features to the site. There are some strong requirements for the project, since the code will be integrated into existing system:

1. The project has to be on Ruby-on-Rails. The site is fully implemented using Ruby-on-Rails.
2. Needs 100% code coverage. Every single line of code on the site currently is automated and tested.
3. The code needs to be of high quality.

If given a choice between doing more with less quality and doing less with more quality, the latter is preferred on this project.

The following are the candidate features for the project, in the order if high priority to low:

- Discussion
- Student membership request and authorization
- Reward points
- Sorting with infinite scrolling

All the code for this project will be brand new, however, it does need some data from the application. The project coordinator will provide the necessary API to get these data for testing purposes.

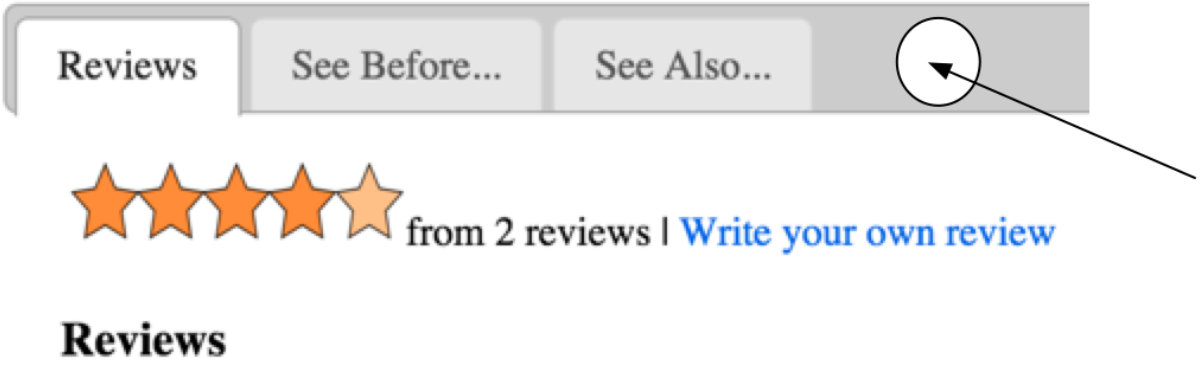
High level details of the features:

Quite a few details of the features will emerge during discussions, design, and implementation. Here are some high level details:

Discussion:

The site has many presentations. Currently viewers can write a review for any presentation. However, there's no place for discussions at the moment. The discussion feature will add a tab to the presentation page.

For example, visit <https://www.agilelearner.com/presentation/401>. You'll see the tabs like the one shown in the figure below. The Discussion tab will go where shown in the figure.



Reviews

When a user clicks on the Discussion tab, the client side will make an ajax call to get all the posts for that topic. A user can create a new topic or reply to an existing post. When a user creates a new topic, it is saved (using ajax) and the Discussion page is updated. An email is sent to the presentation authors a new post is created. When a user creates a reply for a post, an email is sent both to the presentation authors and the writer of the post. Email should not be sent at any time to the person who creates a post or reply. Replies are posts too and can receive replies.

The user interaction should be smooth and require minimum full page refreshes.

Student membership request and authorization:

The site offers free membership to students. Currently, students request for membership by email. A backend script creates the membership, but the rest of the verification steps are manual. We want to create a good workflow for students to request for free membership.

1. Active subscribers (they already have a paid or free subscription) can't request.
2. To request, they fill in a form:
   Faculty first name, last name, email
3. When the complete this step, an email is sent to the faculty asking them to verify the student (a link is sent).
3. Student may view a pending verification request.
4. Student may cancel a pending verification request.
5. When faculty visits, ask them for their own URL associated with their university or school.
6. Thank them and update the status to waiting for approval (student can't cancel verification at this time). Email admin that a new verification is waiting for approval.
7. Admin can view all pending and verified requests.
8. Admin can't change pending requests.
9. Admin can view a verified request.

10. Admin can click on a link to view faculty page.
11. Admin can approve (add optional message) or disapprove (message required).
12. If approved, system gives 12 months free subscription to the student
12. Email sent to student

Reward points:

This is a loyalty program. Users can view their current reward points and renew it for some rewards.

1. User's profile page shows the points.
2. Each time they write a review, they get x number of points. The value of x may change over time. The value should be calculated based on the points valid at the time the review is first written.
3. Each time they pay for a subscription, they get y number of points. Same rules like above.
4. User can click on the rewards point and will be taken to the rewards page.
5. The page will display details about what the points are for, how they could earn, and how they could use it.
6. The user can apply a reward. Display various rewards. For example, they could get a book, a one month subscription, a T-shirt, etc . If it's a T-shirt or something that requires size, then they should be able to select the size.
7. When they order a product, reward points should be reduced from their account. If it is non-electronic item, then they should enter their address for shipping, etc.
8. Admin is notified by email when a user claims a reward.
9. Admin can create award items. They can make an award item active (can be used) or inactive (no one can see it anymore for claim).

More details will emerge later.

Sorting with infinite scrolling:

Currently the site has infinite scrolling. It does not offer any sorting of data presented. Explore how sorting on various columns can be included with infinite scrolling.

Happy Learning.