Team 👌 - Deliverable 5

Chedy Sankar
Omar Chehab
Jacob McMorrow
Dayde Reid
Chengrong Zhang

12/11/2018

Code Reviews - v2

Table of Contents

Tests

As before, we have broken testing into two categories, unit-tests/integration-tests and acceptance-tests. The details for each have been recapped below.

Unit-tests/integration-tests:

All of the unit-tests and integration-tests of our application are automated using a combination of JUnit, and the testing functionalities provided by Spring Boot. Our strict unit-testing is performed with the standard JUnit tools, and is found in our factories, utilities, expressions, operators, and parameters.

Our integration-tests come in two forms, DataJpaTests and WebEnvironment tests. The former used on our model tests, implements the TestEntityManager supplied by Spring to mock our persistence layer. This allows us to integration-test by testing the interactions between our models, factories, and databases with accurate responses while not permanently altering our data.

The WebEnvironment tests, used on our controllers, directly interact with our persistence layer. Since the controllers make use of all other components of our application, we are in effect integrating-testing our entire system at this point.

Acceptance-tests:

The acceptance-testing of our application is performed manually. To ensure consistent testing, a script outlining our procedure has been created. The script is broken down into each of our user stories and the specific tasks to be performed on our GUI required to accomplish them. Each task is given a test id, description, expected result, and a section to note the outcome of the test.

When a tests passes, a “P” is placed in the corresponding outcome box, and an “F” when it fails. Every time a test is repeated and passes during the script, an other “P” is added, e.g. two successful tests reads “PP”. Any outcome marked with “NI” represents a feature that is not implemented at this time. Should a test fail after one or more passes an “F” is place in the corresponding position to clearly mark when a failure occurred, simplifying debugging. E.g. two passes and a fail reads “PPF”.

The following is our current round of acceptance testing:

Test	**Description**	**Expected Result**	**Outcome**
U1-00	Click "Add Partner" link	Displays form to input information for new organization	P
U1-01	Submit blank organization	Reminded to fill in required fields	P
U1-02	Create organizations	A new organization with information provided appears in organization view page	P

U2-00	View organizations	Clicking the "Partners" link displays a page with table of organizations	PP
U3-00	Click "View" link for an organization	Displays a page with organization information and link to update information	PPPP
U3-01	Click "Edit" link on organization page	Displays for to update information for this organization	PP
U3-02	Update organizations	Updated information displays on single and all organization pages	P
U4-00	Delete organizations, repeat U3-000, U3-001	The deleted organization is removed from the table of organizations	P
U5-00	Select template type from drop-down menu, repeat U3-000	The template upload options are desplayed and can be selected	P
U5-01	Browse computer for template to upload	File navigation window appears	p
U5-02	Select wrong file type	Prevented from uploading the file	P
U5-03	Select correct file type	Page allows you to upload template	p
U5-04	Upload iCare template	The supplied template's information is added to the organization's template table	p
U6-01	View data input by my organization, repeat U2-000, U3-000	Displays all of the template type selected upload by this organization	P
U6-02	Apply select parameter to templates table	Only the chosen columns are shown in the table view	P
U7-00	Click "Data" link	Displays a page with links to pages for each template type	P
U7-01	Apply join parameter to templates table (e.g. ?join=ClientProfileTemplate)	Display a cartesian product of the current table with chosen tables	P
U8-00	Apply sort parameter to templates table (e.g. ?sort=postalCd&sortDirection=desc)	Display the table sorted by the chosen column in the chosen direction	P
U8-01	Apply group parameter to templates table (e.g. ?group=preferredOfficialLanguageId)	Rows are grouped by their value for a chosen column	P
U8-02	Apply where parameter to templates table (e.g. ?where={"$eq":["$id$",4]})	Filter the table rows to only those that match the boolean expression given	P
U9-00	Click "Queries" link	Displays a page with like to create a query	PPP
U9-01	Click "Create" link	Display parameters to select from to	P

		create a query	
U9-02	Create reuseable SQL queries	SQL query is saved for future use and appears on table of queries	P
U10-00	View SQL queries	Clicking the "Queries" link displays a page with table of acounts	P
U11-00	Click the name of SQL query to delete, repeate U9-00	Page with query options are shown, inclueing delete button	P
U11-00	Delete SQL queries	The deleted query is removed from the table of queriess	P
U12-00	Execute premade queries to make reports, repeat U9-00	Displays the resulting information of the query	P
U13-00	View existing reports	Clicking the "Reports" link displays a page showing existing reports	PP
U13-01	View existing charts	Clicking the "Charts" link displays a page showing existing charts	PP
U14-00	Click "View" link for an existing report, repeat U13-00	Displays the content of selected report	P
U14-01	Click "Edit" link	Displays page with fields to update report	P
U14-02	Delete existing reports	The deleted report is removed from report page	P

Review Strategy

The same as last time, we will be looking at completed tasks from the sprint backlog and looking for the following: **Design, Readability, Unhandled boundary cases, Bugs/functionality, (in order of importance)**

Chedy Will Review: Omar
Summary of Findings:
The idea of the new-user-friendly homepage was great and was executed nicely. After reviewing the code for this portion, there was great evidence of organization leading to ease of readability, code re-use (through thymeleaf fragments) and was functional overall.
Moving on to another task that Omar completed (Comparison Operators), this portion of code was found to be done very nicely, having an appropriate interface which all Comparison Operators implemented, as well as having really organized and readable functioning code. Code re-use was excellent, as Omar even implement the not-equals operator by simply negating an instance of an is-equal operator. There was nothing to really fix in these couple of cases; Omar has done an astounding job.

Dayde Will Review: Jacob
Summary of Findings:
Once again, Jacob has done an excellent job with his work these previous sprints. His work on the chart backend follows our existing conventions perfectly, and he knows enough to split up functions once they become to large and unfocused. Additionally, the testing suite which he has created and maintained does an excellent job of ensuring all of our code is functioning properly. There is one slight issue with regards to the readability of the code, however. Some of the spacing in his work on the chart controller is a little erratic, making it harder to know what lines are apart of which method. However, other than this minor issue, Jacob's work is fantastic.

Omar Will Review: Dayde
Summary of Findings:
I reviewed Dayde's implementation of charts. With regards to single responsibility principle, Dayde broke down the task into isolated classes and templates thoughtfully. He had a model, repository, thymeleaf template, and controller each of which is responsible for a single task. Dayde imported Chart.JS and bridged the data connectivity from Java to JavaScript using a CDATA block.

With regards to duplication, I suggest merging the chart update and create templates into a single template which handles both. That way, when we decide to change the chart form, we would only have to change it in one place.

Chengrong Will Review: Chedy

Summary of Findings:

Chedy did great job for these two sprints. The model he made is perfectly worked with the front-end and each element is well designed. This makes the model very clear and easy to use when working with the integration with the front and back end. The styles of web page views are looks beautiful and every page looks consistent. The source code of the pages are easy to follow, you can quickly find the elements in the webpages. After all, great work done by Chedy.

Jacob Will Review: Chengrong

Summary of Findings:

Chengrong performed well over the last two sprints. His work on the Reports front end is consistent with the rest of our project, very readable, and accomplishes what it sets out to with out any bloat. His work on the Query Model, Repository, and Controller all fit the project conventions perfectly, using Spring Boot components optimally in the first two. In the case of the Controller, he recognized when code was to be re-used, and/or did not fit perfectly in one function, so moved the code to its own method.