Team 👌 - Deliverable 4

Chedy Sankar
Omar Chehab
Jacob McMorrow
Dayde Reid
Chengrong Zhang

12/11/2018

Code Reviews

Table of Contents

Tests

We have broken testing into two categories, unit-tests/integration-tests and acceptance-tests, detailed as follows.

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. Our strict unit-testing is performed with the standard JUnit tools, and is found in our factories, utilities, and operators.

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 layers. This allows use to test the integration of our models without altering our database. The WebEnvironment tests, used on our controllers, directly interact with our persistence layer. Since the controllers make use of all other components, we are in effect integrating-testing our entire system at this point.

Acceptance-tests:

The acceptance-tests of our application are performed manually. To ensure testing is performed consistently, a script for testing has been created. The script is broken down into each of our user stories and 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. Any outcome marked with “NI” represents a feature that is not implemented at this time.

The following is our current round of acceptance testing:

Test	Description	Expected Result	Outcome
U1-00	Click "Create" 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 "Organizations" 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 displayed 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	F
U5-03	Select correct file type	Page allows you to upload template	F
U5-04	Upload iCare template	The supplied template's information is added to the organization's template table	F
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 "Templates" link	Displays a page with links to pages for each template type	PP
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
U9-01	Click any of the template type links, repeat U7-000	Displays a page with table containing information corresponding to template type	P
U9-02	Create reusable SQL queries	SQL query is saved for future use and appears on table of queries	NI
U10-00	View SQL queries	Clicking the "Queries" link displays a	NI

		page with table of accounts	
U11-00	Delete SQL queries	The deleted query is removed from the table of queries	NI
U12-00	Execute premade queries to make reports	Selected query returns its associated report	NI
U13-00	View existing reports	Displays a page showing existing reports	NI
U14-00	Delete existing reports	The deleted report is removed from report page	NI
U15-00	Update immigrant data	Updated information displays for all instances of the immigrant	NI
U16-00	Delete immigrant data	Data for this immigrant is removed from all corresponding tables	NI
U17-00	Create accounts	A new account with information provided appears in account view page	NI
U18-00	View accounts	Clicking the "Accounts" link displays a page with table of accounts	NI
U19-00	Update accounts	Updated information displays on single and all account pages	NI
U20-00	Delete accounts	The deleted account is removed from the table of accounts	NI
U21-00	Add an account to an organization	This account is added to the list of accounts associate with the organization	NI

Review Strategy

We will be looking at completed tasks from the sprint backlog and looking for the following: **Bugs/functionality, unhandled boundary cases, design, readability (in order of importance)**

Chedy Will Review: Chengrong
Summary of Findings:

The code written by Chengrong compiles and works as intended. There is no room for bugs for any sort in the tasks he has written (lots of spring boot library use). The lack of prevalent boundary cases makes the code clean and very concise (The Organization update POST mapping). The design of the code is as planned in our team meetings, and paired with the conciseness, makes it very readable. There was not a whole lot to look out for/test for, but the code is now up to par.

Dayde Will Review: Jacob
Summary of Findings:

Jacob has completed his tasks in a very effective manner. Through the other parts of the project that use it, we know that all of his back end work (the organization model, and the various template factories) works as intended, and his front end contribution (the single organization view page) is effective at displaying the information it needs too. The majority of his code is designed very well, with comments not being needed due to its simplicity. However, the two template factories are currently separate classes, when they could be made to implement a single interface class due to their similarity instead. As we expand the project with new template types, this will make things easier. Other than that, Jacob has done an exceptional job.

Omar Will Review: Chedy
Summary of Findings:

Chedy was responsible for connecting several classes the team developed together in order to parse a CSV, instantiate the appropriate entities from rows in the sheet and save the entities into the database. Overall, the code works well and has some error handling built in for dealing with human error. Chedy also documented his code well making it easy to read and understand without having him explain it. He had difficulties with the RepositoryWrapper so he compromised by writing an equivalent method called getRepo inside the controller. In terms of functionality, it's okay because it gets the job done, however, when considering the single responsibility principle, the controller shouldn't have such method written inside of it. As a side effect, the getRepo code cannot be re-used inside other controllers without duplicating it.

Chengrong Will Review: Dayde
Summary of Findings:

The code written by Dayde works well. The templates he creates clearly shows the information of the forms. The attributes are mapped correctly, the content of the attribute are

parsed correctly. Also if there is an illegal input that enter by user, a warning page will appear accurately and the exception is raised appropriately. The design of the code is very good, the code has very detailed comments and this helps a lot for understanding the code. The code are working as we estimated, if the work isn't work, the webpage won't show the tables of the templates.

Jacob Will Review: Omar

Summary of Findings:

Omar's code compiles, and has been written very well. Overall the code works well, and is bug free. Although there is error checking, in CSVSheetAdapter, passing a headerRow and dataStartRow outside the bounds of a CSV is not checked for. However, in the context of how we use the adapter, this does not present an issue. The design of Omar's code is very well chosen, and I have found no issues. Omar's code is easy to read an understand for the most part, but I would like to see a little more documentation in some of the more complicated sections of code and consistency with JavaDoc. Overall, Omar has done a great job, and finding issues was difficult.

Code Review Debriefing Meeting:

https://youtu.be/koHRKjvP8wc