LMS Presentation

Ethan Nguyen
Evan Park
Rhitu Thapa
Jonathan Medina
Victor Tovar

Agenda

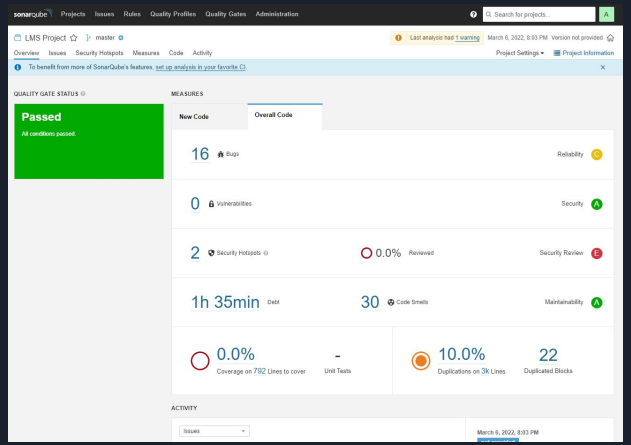
1. Milestone 1 (setting up project)
2. Milestone 2 (static analysis)
3. Milestone 3 (selenium testing)
4. Milestone 4 (CRUD)
5. Milestone 5 (Load Testing)
6. What to refactor
7. Functional Testing
8. Regression Testing
9. Maintenance

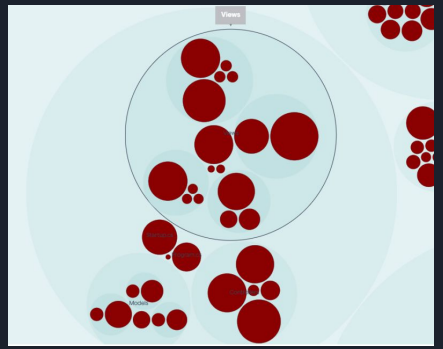
Milestone 1 - Setup

- Installed Visual Studio, Docker, Resharper, Datagrip, and LMS GitHub repository
- Set up WSL environment and Docker account
- Set up Visual Studio; Installed .NET 5.0 and other dependencies
- Ran ‘LightLib.sln’
- Started Docker through cmd
- Ran build with ‘LightLib.Web’ as starter program
- Accessed webpage through ‘localhost:8000’

Milestone 2 - Static Analysis

- Sonarqube
  - 0 vulnerabilities
  - 2 security hotspots
  - 16 bugs (8 major, 8 minor)
  - 30 code smells
- CodeScene
  - Outlined Hotspots
  - 9.75/10 health rating
  - 2 commits, 0% abandonment





Milestone 3 - Selenium Testing

Selenium testing saves a lot of time by not having to manually run through our program after every iteration of change.

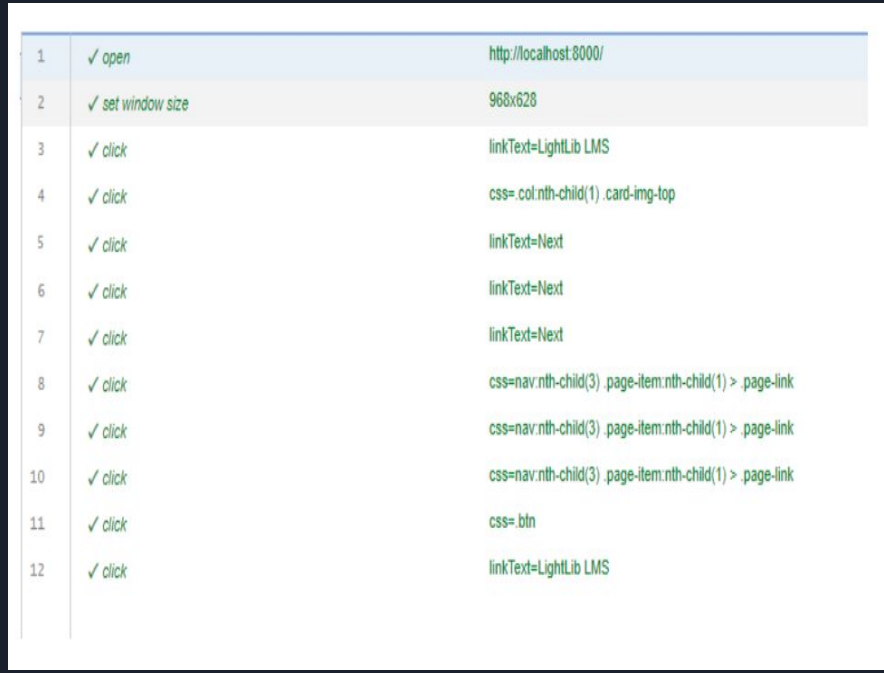
For this milestone, we downloaded selenium ide as an extension for google chrome, we ran lightlib solution and we deployed to test that solution in Selenium.

Finally, we tested the project using Selenium and recorded the test.

Lessons learned:

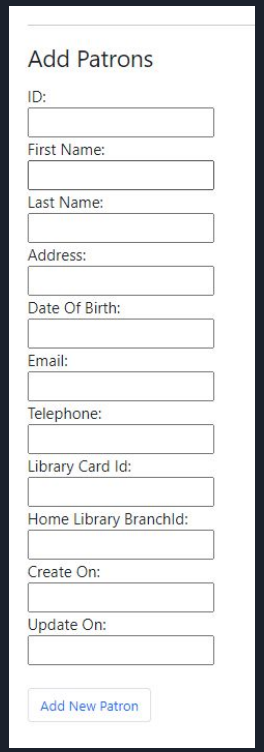
We learnt how to figure out whether an application is responsive, progressive or regular using Selenium testing.

It makes functional testing very easy.

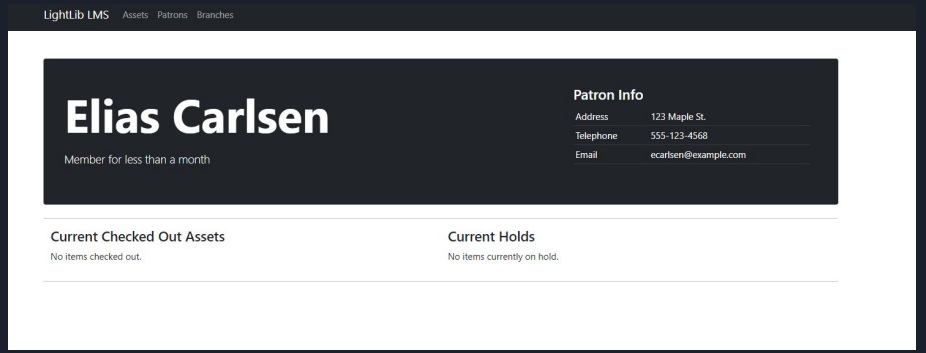


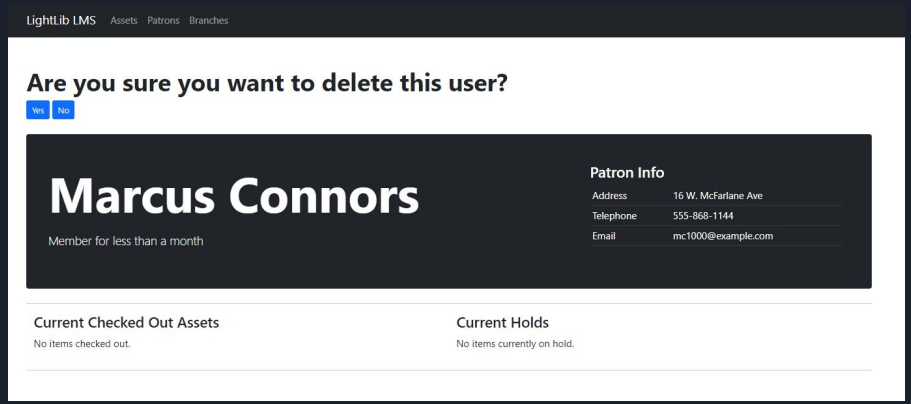
1	✓ open	http://localhost:8000/
2	✓ set window size	968x628
3	✓ click	linkText=LightLib LMS
4	✓ click	css=.col:nth-child(1) .card-img-top
5	✓ click	linkText=Next
6	✓ click	linkText=Next
7	✓ click	linkText=Next
8	✓ click	css=nav:nth-child(3) .page-item:nth-child(1) > .page-link
9	✓ click	css=nav:nth-child(3) .page-item:nth-child(1) > .page-link
10	✓ click	css=nav:nth-child(3) .page-item:nth-child(1) > .page-link
11	✓ click	css=.btn
12	✓ click	linkText=LightLib LMS

Milestone 4 - CRUD



- CRUD: basic functions for any database system
- Create/Update - similar in function
- Retrieve - prints data to new page
- Delete - requires verification to delete an entry





Milestone 5 - Load Testing

For the milestone 5- we used load Ninja for load and stress testing on LMS.

Setting the user limit to 15, the testing showed how well the program ran under the stress.

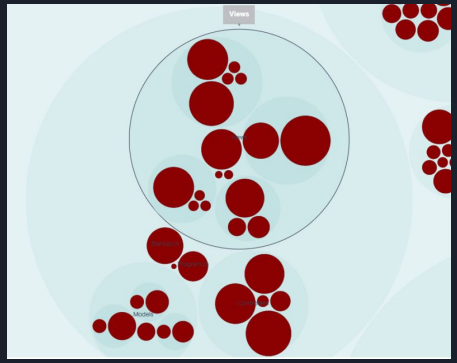
The diagram on the side shows our overall performance summary for the testing.

Lessons learned:

From this milestone, we learnt how the system behaves under an unexpected load and stress test helped us understand the upper limit of system's capacity using load beyond expected maximum.

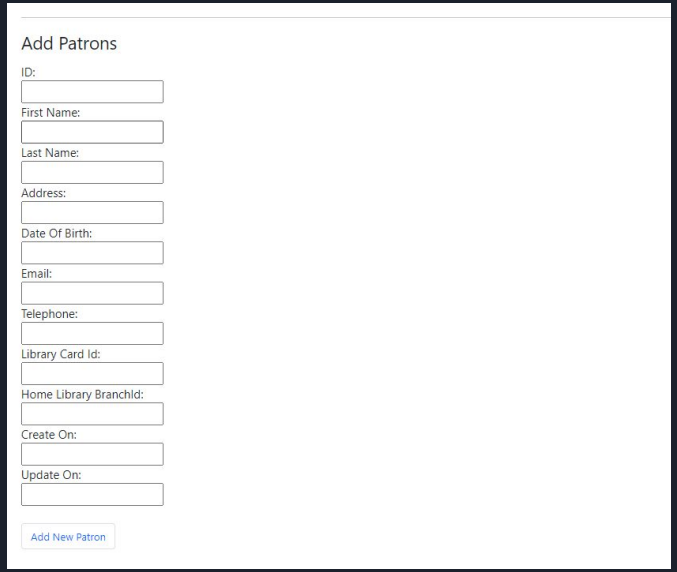
What should we Refactor?

- Based on our results from previous tests, we believe that we should look to refactor most elements in “LightLib.web”
  - When we used some static code analysis tools, one of the results was a “hotspot” graph/image
  - This showed us which parts of our code was being used the most and Inside LightLib.web was the majority of the hot zones
  - This is also where we implemented our CRUD feature which is crucial for this program to work properly
- Some elements we could refactor would be:
  - Startup.cs
  - Program.cs
  - Index.cshtml (home)
- The reason we choose these:
  - We have to go through these elements no matter what part of the code we are testing.
  - It is the main part of the project that most other elements in LightLib.web build off of



Functional Testing

- We decided to implement the CRUD features for the patron class.
- We had to test added features + other features already implemented
- Our validation rules included:
  - Check for correct inputs on name (not null)
  - Check for correct inputs on ID (not null)
  - Check for correct inputs on libraryCardID (not null)
- Everything else has less stricter validation rules.



- Code/Process
  - We connected out postgreSQL database to our LMS System
  - Our file displayed the elements in the database in a Table format
  - Inside the table were buttons that would allow us to View, delete or edit elements in the DB.

Last	First	Email	Fees	Delete	Edit
Carlsen	Elias	ecarlsen@example.com	$0	Edit	Delete
Smith	Pam	psmith@example.com	$0	Edit	Delete
Connors	Marcus	mc1000@example.com	$0	Edit	Delete
Wilson	Dalante	dwilson@example.com	$0	Edit	Delete
Andersen	Ben	bander@example.com	$0	Edit	Delete
Porter	Krista	krista_porter@example.com	$0	Edit	Delete
Smith	Elma	elma_jean_smith@example.com	$0	Edit	Delete
Black	Sandy	sandy_201@example.com	$0	Edit	Delete
Li	Jin	jinli000@example.com	$0	Edit	Delete
Lopez	Maria	lopez_maria@example.com	$0	Edit	Delete

Previous Next

Regression Testing

- During our functional testing we noticed that some pages would error depending on inputs/Missing data
  - Because of this we recommend to always test to go through every page to make sure there are no errors
  - Also test pages with different data/no data/Full of data (excess)
- Our test selection:
  - Every visible page/accessible page to the user
  - We will not look the same page twice so we can minimize the test cases
  - We will also prioritize pages with data elements because these are pages that will change a lot, depending on the amount of data held

Maintenance

- We recommend to review the code for each part of the system monthly.
  - Every new patch should include code review before merging to the main branch
  - Also while working on the project, we found it helpful to make different modifications when needed but never merged it to the main branch since features were not needed in the main program
- Test automation
  - Automatically generating data and adding it to the LMS system could be useful
  - This would help continuously test that the system can work under heavy loads
  - We also want to make sure we cover most of the pages that need to access our database so we know that our program is still functioning

Q&A