CptS 583 Software Quality

Spring 2021

Project Deliverable 2-3
LA583

Quality Report
Product Quality Metrics

Quality: Availability
Goal: The system shall be always accessible.
Metrics: System availability should be 99.9%.
Validation Results:
According to AWS Health Dashboard, there were no failed requests or issues. Also, read latency and write latency worked in milliseconds as confirmed by AWS RDS monitoring. By March 15, read latency was measured much higher than normal, but this was also 20 milliseconds.

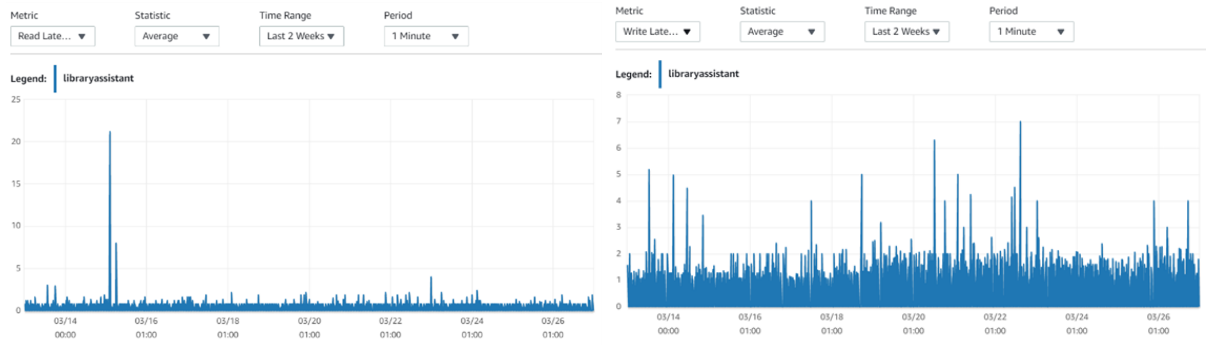


Image 1,2. Read Latency and Write Latency

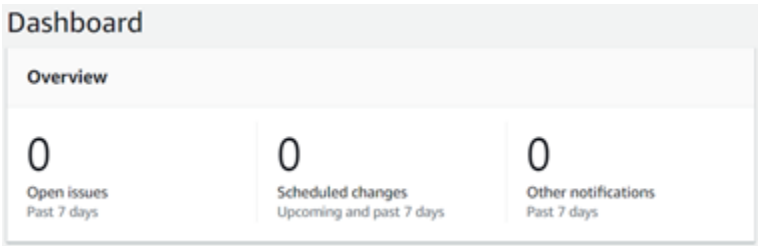


Image 3. AWS RDS Personal Dashboard for server health

Quality: Reliability
Goal: The system shall always be in working order.
Metric: The system error occurrences should be less than 2%.

Validation Results:
According to AWS RDS monitoring, all DB connection requests were 100% successful. In other words, there are no cases in which a request for connection from a web app to a DB has failed. In addition, there has been no error/issue log for AWS RDS us-west2 server in the last three months.

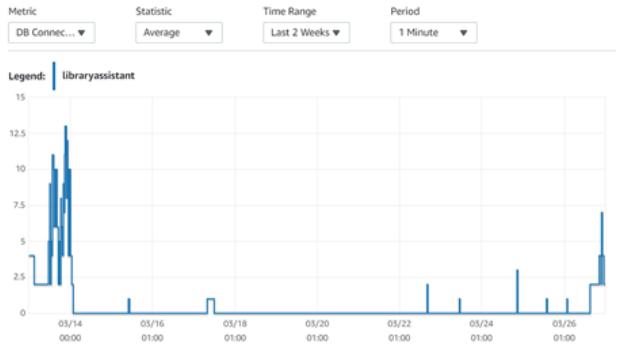


Image 4. The count of DB connection requests

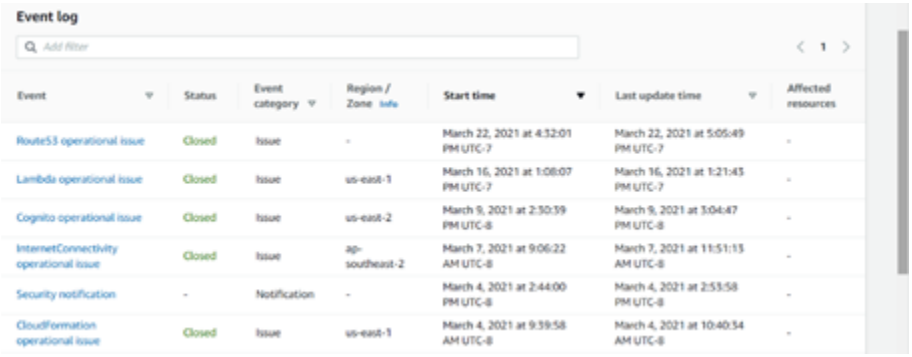


Image 5. AWS RDS server event log

Quality: Robustness
Goal: The system should be able to accept any appropriate user inputs.
Metric: The system Should not return any errors with appropriate inputs.
Validation results:
No systematic problems or database connection errors were found when unit tests confirmed the results of each function being recalled to the database. Each column in the database can store sufficiently long strings for new user registration, user information editing, book addition, or book editing, but long input strings could cause errors in the system.

Quality: Learnability
Goal: The system shouldn't be too complicated and easy to learn.
Metric: The user should be able to fully navigate the system within an hour.
Validation results:
As a result of potential users using the program, it took them 5 to 10 minutes to learn all the

functions such as register, search, add book to cart, check-out. The system is minimal which keeps the learnability time low. Only one minor issue with the design of our library assistant app. The main screen doesn't have any indication to the user on how to add a book to a cart, they would most likely discover how to by accident by double clicking a selection from the table on the main page. This can easily be fixed by adding a small label to direct the user on the main page.

Quality: Usability
Goal: The system should be simple to use.
Metric: The system should be straightforward with a simple interface.
Validation results:
Potential users of the system were satisfied with the usability of the software. The software was straightforward and provided all the information necessary for them to find a book they liked, along with the information of their account. The system only has the important functionality for the librarian and user to keep the UI to a minimum and to keep minimal confusion while using the system.

Quality: Efficiency
Goal: The system should respond to user inputs in a timely manner.
Metric: The system should return search immediately.
Validation results:
When the functions to call the database were used, the read latency and write latency were exceeded no more than 10 millisecond. Also, the program gives quick feedback on users' input. However, the login process is slow to respond to users' input due to security factors. Since our data is reliant on AWS the performance accessing the database should be consistent for every user.

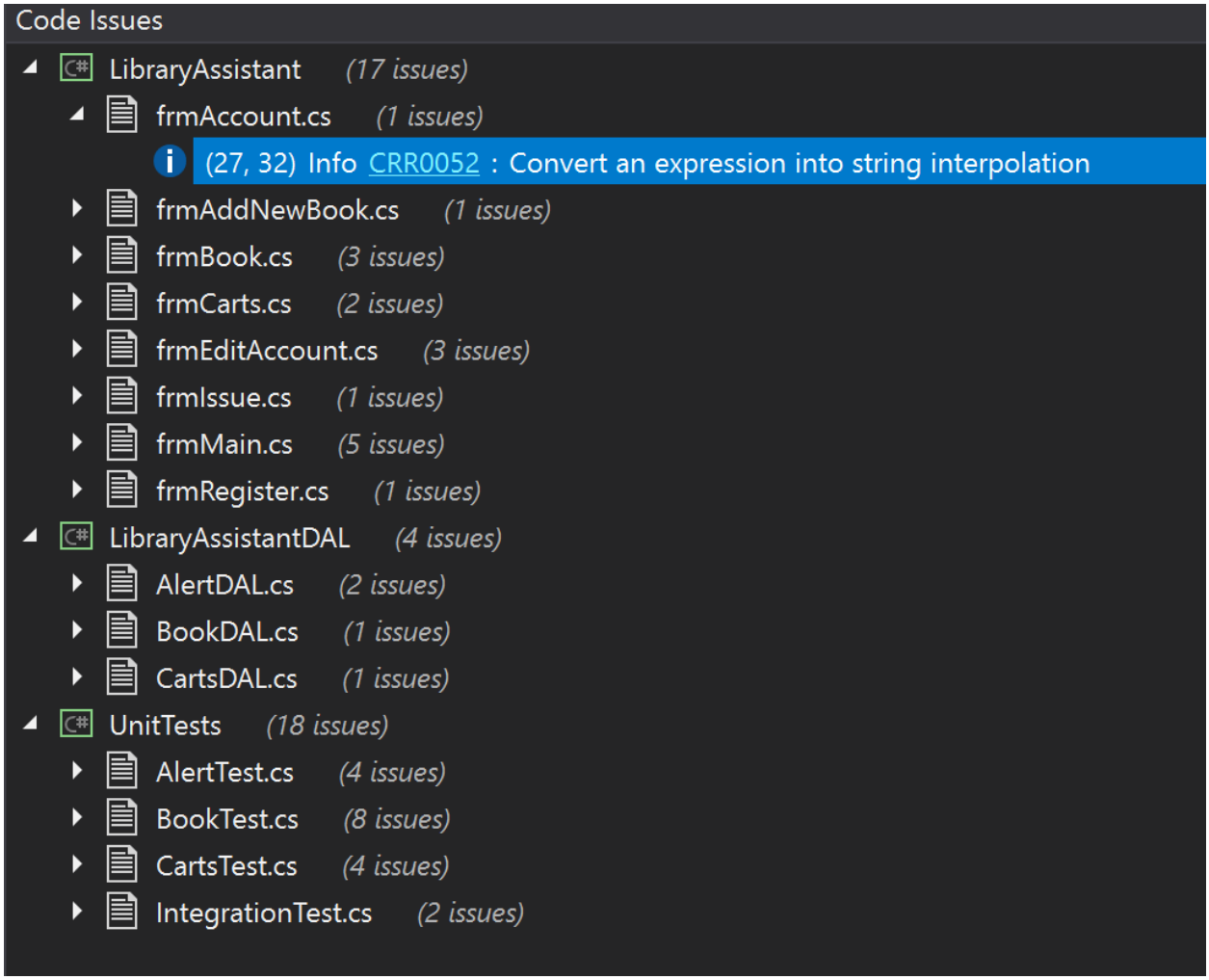
Quality: Security
Goal: The system should be safe and secure for user information.
Metric: User data cannot be accessed by someone else, even the librarian. User password will be encrypted.
Validation results:
To make sure that our systems is safer, we ran test to encrypt the password so only the user should be able to access their account. Tests were done where the user tries with an unencrypted password and encrypted password to login. The test results show that the user was not able to log in when the user's password was not encrypted but successfully logged in when the password was encrypted. This is done to prevent people who attempt to use SQL injection to get access to users information and the system.

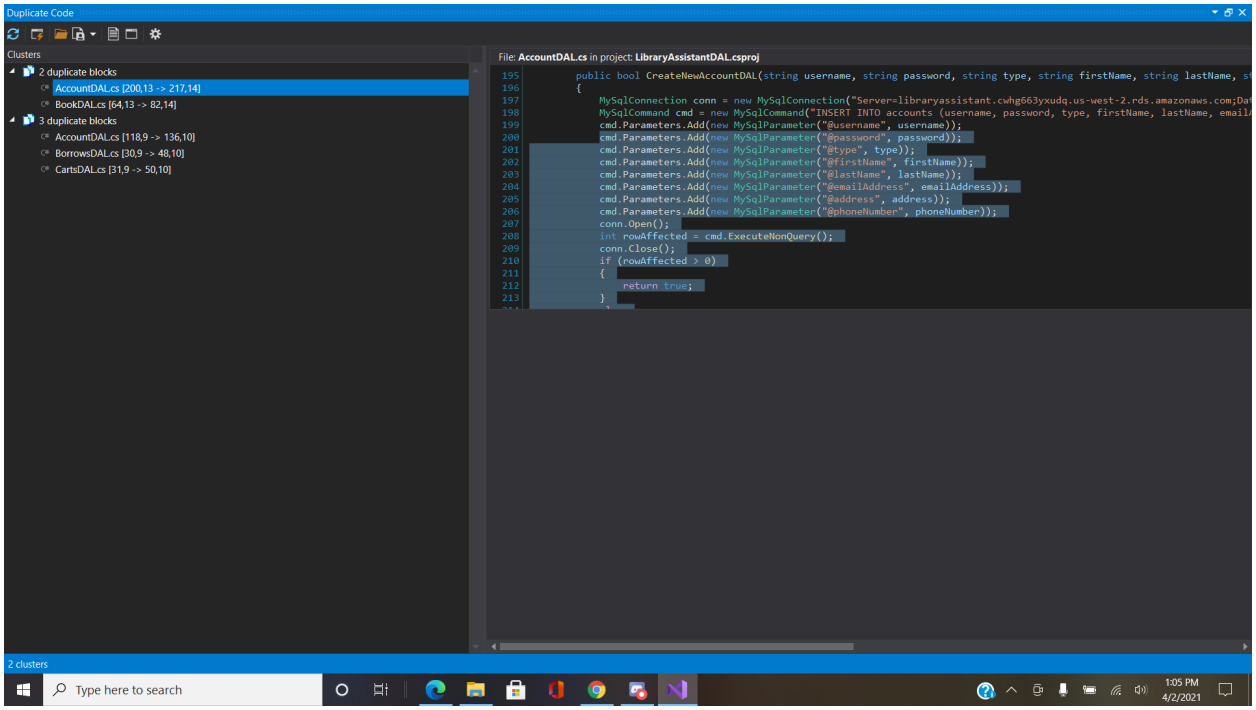
Quality: Portability

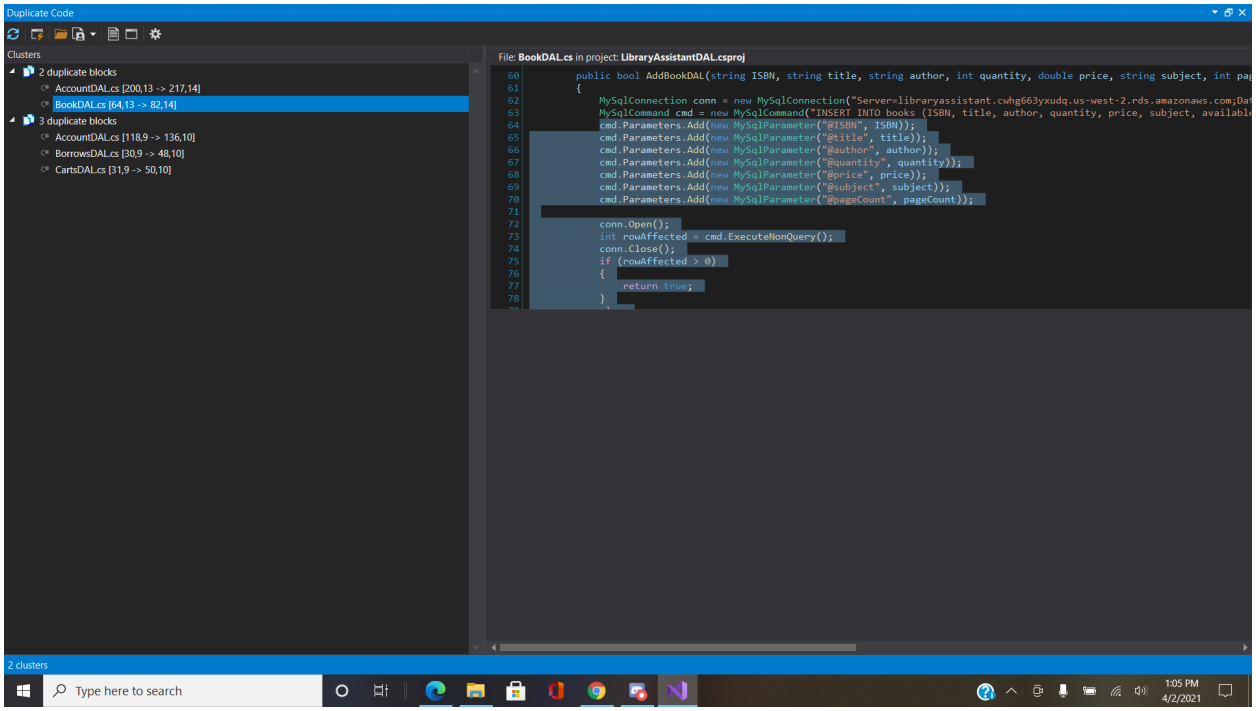
Goal: The system should work with any database.
Metric: The system should work with modern databases.
Validation results:
The database was created to run on Mysql server, and it can be migrated to another database format. Transition to other databases was possible through methods supported by AWS or through JSON files(export SQL file and convert it as JSON file). In addition, moving the database platform worked without any problems.

Process Quality Metrics

Quality: **Maintainability**
Goal: The system should not be complex to maintain.
Metric: the source code should not contain code smells
Validation results:







In terms of maintainability the system was tested with the tool coderush for finding instances that needed refactoring and finding code smells. The results as seen above show that there was many area that need to be refactored.The above figure shows that there are minimal code issues using coderush and when there are, they are small issues like using " --" + (var) instead of $"------{var}". The following screenshots show that our code suffers from duplicate code within the accountDAL, bookDAL, cartsDAL,and borrowsDAL. As seen by the result the codes

among those 5 files are rather similar but not necessarily exactly the same. As a result our project failed to avoid any code smells and would need to be heavily refactored.

Quality: Testability
Goal: The system should be in a good environment for testing.
Metric: Every class within the system should have both black and white box testing.
Validation results:
This system is easy to carry out unit tests, because it has good decomposability, stability, and understandability for testing by dividing the system into UI parts, Business layers, and Date analysis layers. Most of the classes were structured to be suitable for unit testing, but some classes need to add the codes for deleting test data after testing. For this reason, the existing code was modified for easier testing. We also implemented integration tests for this system, the results show that the current system is quite good.

Product Quality	**Quality Goals**	**Quality Metrics**	**Verifications and Validation Results**
Availability	The system shall be always accessible.	System availability should be 99.9%.	See the detailed description above
Reliability	The system shall always be in working order.	The system error occurrences should be less than 2%.	See the detailed description above
Robustness	The system should be able to accept any appropriate user inputs.	The system Should not return any errors with appropriate inputs.	See the detailed description above
Leanability	The system shouldn't be too complicated and easy to learn.	The user should be able to fully navigate the system within an hour.	See the detailed description above
Usability	The system should be simple to use.	The system should be straightforward with a simple interface.	See the detailed description above
Efficiency	The system should respond to user inputs in a timely manner.	The system should return search immediately.	See the detailed description above

Security	The system should be safe and secure for user information.	User data cannot be accessed by someone else, even the librarian. User password will be encrypted.	See the detailed description above
Portability	The system should work with any database.	The system should work with modern databases.	See the detailed description above
ProcessQuality	**Quality Goals**	**Quality Metrics**	**Verifications and Validation Results**
Maintainability	The system should not be complex to maintain.	The source code should not include code smells.	See the detailed description above
Testability	The system should be in a good environment for testing.	Every class within the system should have both black and white box testing.	See the detailed description above

References:

[1] Coderush , http://developer.rovicorp.com/docs

[2] AWS https://docs.aws.amazon.com/AmazonRDS/latest/APIReference/Welcome.html