Summary of new user stories, new code/functionality and tests for milestone 3

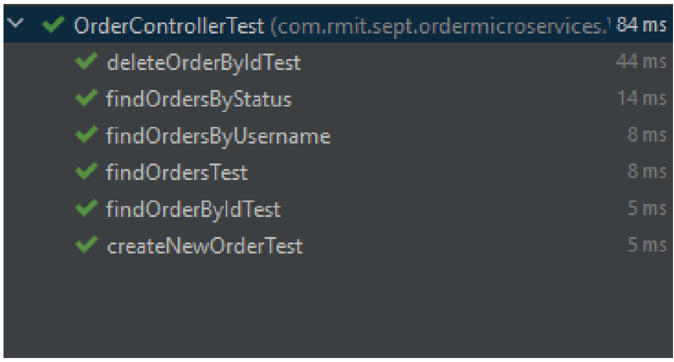
Note: screenshots of product backlog, sprint backlog and testing our found in other documents in the Milestone 3 folder 😊

Now for user stories, we didn't find it necessary to create any new ones. However, we did rearrange order of our stories and priorities in terms of completing the project. This is evident through Jira where you can see scope changes of the sprint backlog and the product backlog shown. As we finished previous sprints, we realized that some of our user stories needed to be swapped on the list so we can ensure a sprint with the most effective work completed.

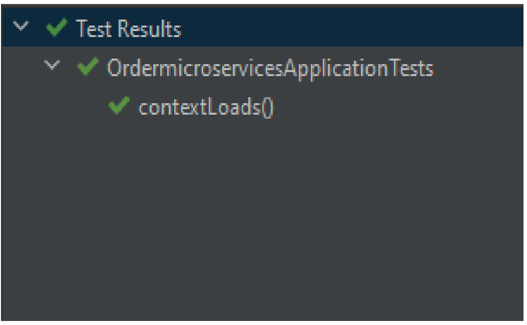
New code implementation/Functionality:

For milestone 3, we focused mainly on customer being able to buy books. This meant implementing a new microservice called order. We also integrated the PayPal API, allowing the customer to pay for said books using PayPal. The other major feature we implemented was the review feature. Customers can now review books that they have purchased. We made another microservice called review to implement this feature. We also did a lot of touching up on the other microservices, to ensure that they function according to our user stories and that they interact correctly with each other. We also setup docker for deployment and circle ci to help with continuous testing, as well as introducing a logger to help with future error traceability.

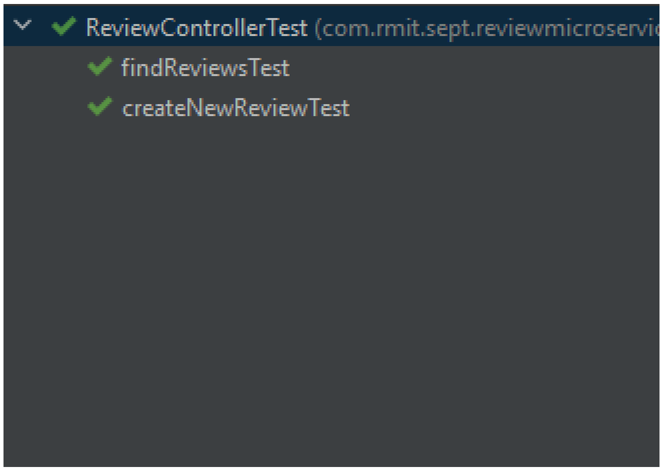
New Tests implemented:



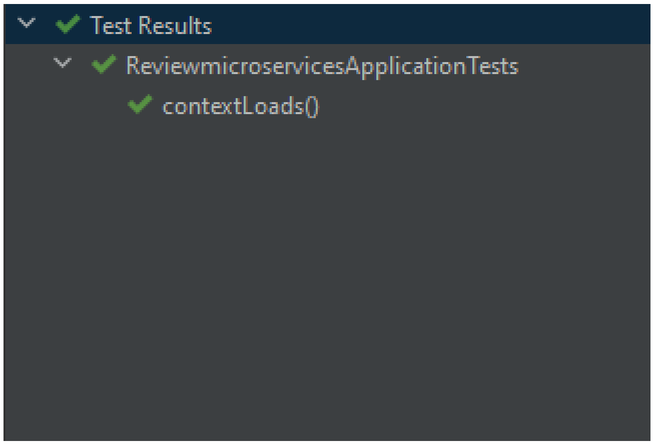
```
✓ OrderControllerTest (com.rmit.sept.ordermicroservices. 84 ms
    ✓ deleteOrderByIdTest                                44 ms
    ✓ findOrdersByStatus                                 14 ms
    ✓ findOrdersByUsername                                8 ms
    ✓ findOrdersTest                                      8 ms
    ✓ findOrderByIdTest                                   5 ms
    ✓ createNewOrderTest                                  5 ms
```



```
✓ Test Results
  ✓ OrdermicroservicesApplicationTests
      ✓ contextLoads()
```



```
✓ ReviewControllerTest (com.rmit.sept.reviewmicroservic
    ✓ findReviewsTest
    ✓ createNewReviewTest
```



```
✓ Test Results
    ✓ ReviewmicroservicesApplicationTests
        ✓ contextLoads()
```

We were able to fully test all parts of the backend and frontend for all implemented features.

We added unit tests for all the microservices we have implemented to test the backend functionality.

Further testing was done using POSTMAN to make sure all post mapping, get mapping, delete mapping and update mapping functions work correctly

We also wrote unit tests for approving/rejecting user registration being an admin

We also did extensive testing on the front end, constantly cross referencing our database to ensure all of the features we implemented interact correctly with our backend

Note: screenshots of tests are provided in the PROJECT REPORT file