Testing report

Group 5 Track & Trace

Unit tests

- For the unit test we use JUnit 5 in the unit package
- All unit tests that access the database make use of a testing schema which is a copy of the normal schema. So the tests won't clutter the normal schema.
- We have written all tests that cover 75% of all our codes
- Parts are not covered (25%):
    + "servlets" package - classes that handle HTTP Requests because it'll be tested in integration test with SELENIUM (20% line of code)
    + Some throwing exceptions (5%) because some classes are quite hard to set up to get the expected exception(Ex: SQL exception: because Java has formatted param so we cannot input invalid param or we cannot suddenly interrupt the connection)
- Most of our test cases try to cover all methods in the DAO package (100% methods with 85% lines coverage). For each DAO class, we carefully test how it connect with our database:
    + For getX() type method: We check if the SQL SELECT code returns expected results or try to get an invalid param and if it throws expected errors(Ex: test with not exist id, invalid param)
    + For updateX() type method: We check them by first updating (SQL UPDATE) the data in the database and then call getX() method to check whether the results are modified as expected. We also try to update values that will violate constraints and expect exceptions.
    + For createX() type method: We check them by first creating the data (SQL INSERT) in the database and then call getX() method to check whether the data exists with expected values.
- We also cover the connection test in our test case (transaction, database, env)
    + The env test cover all the get, set method to set up our database environment with our schema (78% coverage)
    + The database test covers if the connection is get and closed correctly with JDBC (79.3% coverage)

    + The transaction test will check if a user is still online and which actions belong to which transaction by check matching token assigning to user or is offline to close the connection (100% coverage)
- Our unit test also cover 7/8 possible exceptions that users will have (only DaoAvailabilityException cannot be tested due to complicated setup)

Integration tests

- For the integration test we use SELENIUM in Java with JUnit 5 in integration package
- All integration tests are run on a separate server which uses the testing schema mentioned in unit testing to not affect the normal user data.
- The integration test covers the last ***20%*** of the servlet packet(classes to handle HTTP Requests) and the correctness of ***85%*** of all functions that we have
- Parts are not covered:
    + The upload and download functions (because SELENIUM doesn't support function to test them => will be covered in user testing)
    + The chat application and prediction (they are complicated to set up tests => will be covered in user testing)
- For convenience and clarity we categorize the test by core functionalities:
    + **TestLogin:** check if the user is linked to the correct page if logged in as service provider(SP) or customer. Check if it doesn't allow the user to move to the next page if they enter the wrong username or password.
    + **Test HomeScreenLoggedIn page**, check if the first page after logging in as customer is shown correctly
    + **Test ApplyApplicationForm**, check if the application form is in correct format and page functions work correctly
    + **Test ApplicationStep** checks if after submitting the application form (sending POST) it does create a new mortgage in the correct state and status in the table of ApplicationOverview and MyMortgage pages.
    + **TestMyMortgagePage/ApplicationOverview** check if the table showing all mortgages in correct format (after GET applicationForm) and corresponding to the user.
    + **TestDeleteMortgage** checks if the Delete button is working correctly for customer and SP: The alert box, PUT update status and DELETE (If customer clicks => send PUT to update status to Delete Request and SP clicks => send DELETE request to delete mortgage)
    + **TestDocumentCheck** will check the DocumentCheck step. It'll check if the check box filled by SP is shown correctly on the customer side (including the explanation). If 1 checkbox is false then it checks if the status is updated to Failed (PUT request).It also if the new upload document button exists.
    + **TestInterstOffer** check if the interest rate is shown correctly corresponding to user's duration (GET interestOffer) and if it is updated to SP if customer choose option to modify his duration and interest rate(PUT and then GET interestOffer)

+ **TestBindingOffer** check the last step before done. It'll check if the view binding offer function works as expected (link to correct next page), accept, decline, download, upload function exist.

User tests

Features to be tested by the user:

1. The homepage
2. Sign up
3. Sign in as a customer
4. Create a new mortgage application (filling in the application form)
5. View the list of mortgages for the customer
6. View current state for the customer and the progress bar (that is filled earlier)
7. Delete request of the mortgage
8. Chat with the service provider
9. Signing in as a service provider
10. View the list of mortgages as a Service provider and Search for a specific mortgage
11. Try out all the states of the mortgages and their actions.
12. Uploading and viewing document (SP and customer side)
13. Accepting or changing Binding offer and interest offer

Test Subjects:

1. Female 24 Years old, Industrial Design student
2. Female 42 years old, Laboratory technicians

Test:

The test is conducted by Hayel Akel. After each change the users are asked if everything makes sense and if there is any feedback to give

Feedback obtained per feature:

Challenge	Feedback 1	Feedback 2
The homepage	The user thinks that this is informative and very nice that we added a short summary of the functionality of the website	The user thinks that this Is quite nice and clear. The colors look very good and no need to change anything
Sign up	Pretty straight forward and clear for the user	The user stated that there is nothing special about this. It looks the usual, so clear
Sign in as a customer	The user stated that It’s good that they can see how long it’s going to take to validate the credentials. Nothing to be changed here	It is nice that it shows what goes wrong if the email or password are wrong.
Create a new mortgage application (filling in the application form)	The fields of the application form are clear to the user and make sense. However, it took the user sometime to read through everything and think of a reasonable input. It is also clear that the user does this once they request the mortgage	The styling is really nice and all in all it looks clear and no issues when filling the form according to the user

View the list of mortgages for the customer	The colors of the status and the Delete/View button are similar, therefore a bit confusing for the user at first. The user suggested altering the color a bit	The user finds this a very nice interface for the customer and from the looks of it, it looks clear and informative to what the customer can see. Also, the status of the mortgage request is visible and the user saw some useful message that the request is under processing or failed(denied)
View current state for the customer and the progress bar (that is filled earlier)	Once the user clicked on View, they saw the progress bar that indicated the state and then they sew the application they just filled in which was a nice experience. Also, when hovering over a certain state the step shows a timestamp of when the state has finished the user found this useful since she can pair it with the prediction. However, they stated that It would be nice if we can add that as a customer you shouldn't take any action here, the application is already filled in.	This Is clear and for the user and they can see that this page is divided into 3 parts, the general information, the progress bar and the actions that should be taken. The confusing part for the customer here is that the third part is called Action State (in order to perform an action) and in the application state is also added. But the user obviously doesn't have to do anything here

Delete request of the mortgage	The user has the option to delete the mortgage request and it is quite clear how that would work namely when viewing My Mortgage there is the button Delete that the customer can press in order to delete his mortgage request	The user was actually able to make a second mortgage just to test the functionality of the Delete button of the mortgage request and the user was satisfied when it worked.
Chat with the service provider	This is not done yet so it’s not functional yet	It looks like a template and it should be fixed
Signing in as a service provider	Since the user got the staff credentials from the developer then they can just normally log in as SP, so no issues were stated here	Again this was an easy task for the user since they already have the username and password. So no changes needed
View the list of mortgages as a Service provider and Search for a specific mortgage	The user finds this a nice interface, but with the crowded mortgage requests the colors are a bit distracting since again the status colors are the same as the View/Delete colors. The user also liked the sorting and searching fields that are created to filter the mortgages.	This is clear and the styling looks neat. But once there are a lot mortgages then it gets clustered, but it is still pretty clear on what has to be done, etc.

Try out all the states of the mortgages and their actions	The user was instructed to open 2 tabs to simulate the service provider and the customer. The user will go as an SP and move the step to document check and the user checks out the upload and preview document. For the user this is straight forward only it took a while before it’s entirely clear on what are the series of actions that should be taken. The same thing holds for the interest offer and the binding offer but those two still a bit easier	The user was instructed to open 2 tabs to simulate the service provider and the customer. The user again moves the step and uploads bunch of documents. But it also took a while for the user to determine exactly what should be done in order to get all the actions sorted out. Also, the document view after uploading was a bit vague since we have to switch between multiple pages in order to show the document.
Uploading and viewing document (SP and customer side)	The user found this quite straight forward since on the service provider side is actually very clear and once the state is changed the service provider needs to ask the customer for documents and that can be done easily. On the customer side the user found the experience of uploading document also straight forward, it was clear what the intentions are.	The user found the uploading experience alright and indicated that it is very good that the documents can immediately support the claims on the application form. This makes it clear and transparent. The user found the application view button to be quite helpful. Also, the user has the ability to add explanation with the uploaded documents

Accepting or changing Binding offer and interest offer	The interface of the interest offer is nice, but it lacks the confirmation of the chosen interest offer. If we click on Confirm this interest, a pop up will show and then the user has to choose, but after pressing OK, the user doesn't receive any confirmation on it. For the binding offer it looks clear and the user likes the idea of having the ability to download the binding offer when needed	It is quite confusing since I want to change the interest offer, but then it doesn't give me any confirmation that the interest offer is submitted. And on the service provider side it only shows the requested interest offer and the service provider can't perform another action here except for moving step.

Suggestions from the users:

- Binding and interest offer should have a confirmation sign that tells the user if they confirmed or changed the offers
- The user suggests that the developer should choose better colors for the mortgage list on the service provider side.
- Also add at the application state that the customer has no actions to take since he/she already filled in the application form once they requested a mortgage
- An idea was suggested that when a customer already filled in the application form and he is on Document check, there should be an indication on some of the fields of the form that needs evidence (in this case document upload) rather than a separate field where the SP should put requested files manually.
- For the interest offer, make sure you include some way of confirmation after choosing or accepting the interest offer.

Conclusion:

At the end of our last sprint review we managed to finish most of the must-have requirements and those are really reflected on the user since they were able to acknowledge the use and advantage of such an application where Mortgages can be made easier. Also, we got some feedback about some minor issues that we are going to address in the future project sessions to be fixed.