CA400 Testing Document

Project Title: NFC Powered DCU
Students: Eamon Crawford & Sam O'Leary
Student Numbers: 16437394, 16358763
Supervisor: Renaat Verbruggen
Completion Date: 07/05/2021

1.Frontend testing

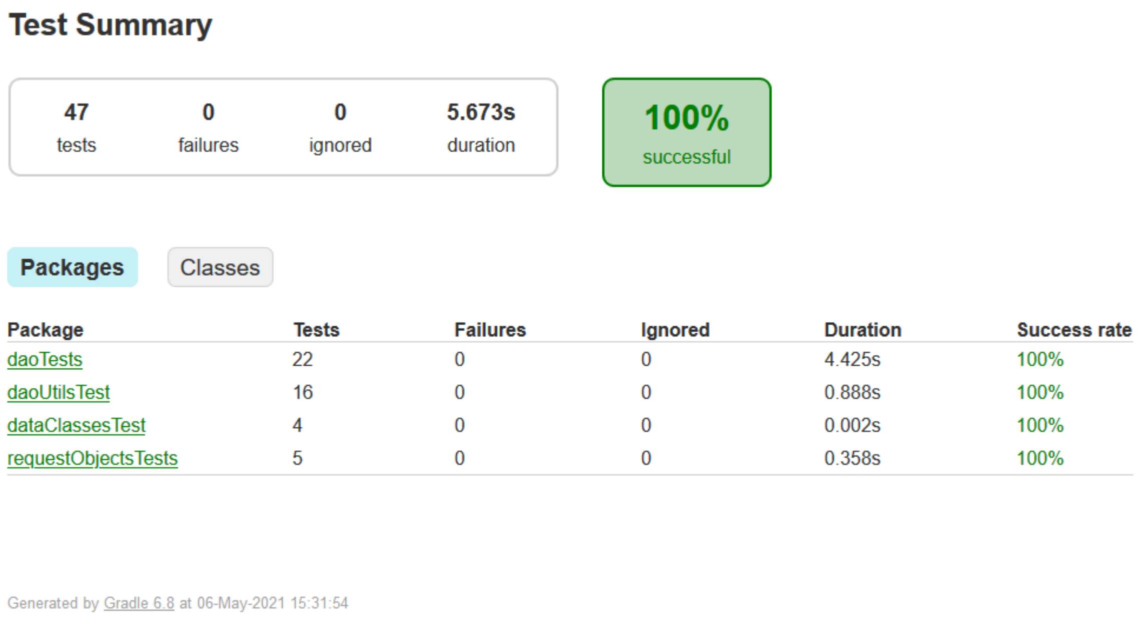
When designing the frontend of the project we tried to keep the design simplistic and consistent for users, to try minimise any potential confusion due to complexity.
We then evaluated the design with Nielsen's usability heuristics. These are ;

1. Visibility of system status: This is seen in our graphing and timetable components as the entry fields have default placeholder values, allowing the user to see the system is working correctly before receiving an input.
2. Match between system and real world: Our project does not have much comparison to the real world but we tried to achieve this with a simple timetable design that would be familiar to users as it mimics traditional paper timetables.
3. User control and freedom: User control was achieved by allowing them to modify their input when fetching the data and also through the delete button for lectures.
4. Consistency and standards: We kept things as consistent as possible for users so that required information should be apparent and properly labeled.
5. Error prevention: This objective was more abstract as our objective when building the system was for it to be stable. This was mainly achieved by ad hoc testing during the build to ensure new components were functional as individuals, and then further testing to ensure they worked correctly when part of the full build.
6. Recognition rather than recall: We tried to ensure that all areas of interaction were clearly and concisely labeled so that users did not have to remember operations
7. Flexibility and efficiency of use: This objective was the hardest for us to complete as we tried to keep the amount of actions low, which limits our ability to accelerate things for frequent actions. This was somewhat achieved by using url templates with our axios requests. However, given more time we would like to have a 'saved favourites' list for frequent users to easily select modules or courses they regularly check.
8. Aesthetics and minimalist design: In the site design we tried to keep it as simplistic and light on information as possible. This was due to any of the information not being relevant to our project itself and potentially detracted from the features we had available. If the site was further developed inorder to be adopted by an organisation, due to the design we would try to limit any text to the home page, as to not congest the graphing or timetable pages.
9. Help users recognize, diagnose and recover from errors: We attempted to make connection errors clear using the placeholder values so that users could instantly know when an error had occurred.

10. Help and documentation: As documentation is one of the deliverables for this project, we have a technical document, user manual, and api documentation, all readily available.

Most of the testing conducted on the react components were unit tests. Unit testing was chosen as our functions should return the same response data for a given value. Unit testing is also good at reducing the bug density in code.
For our unit tests we used the inbuilt JEST package that ships with react. This allowed for easy and consistent testing during the build phase. We tried to keep our components and functions as pure as possible, this allowed us to perform straightforward testing.

2.Backend testing



Test Summary

47	0	0	5.673s
tests	failures	ignored	duration

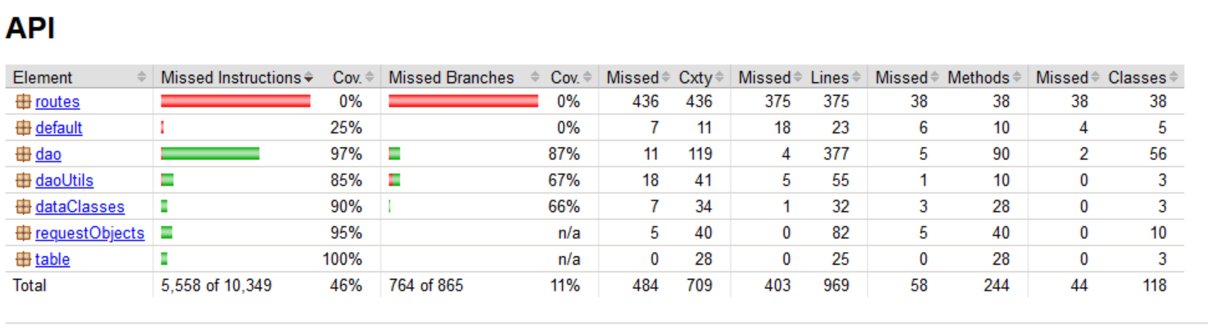
100%
successful

Package	Tests	Failures	Ignored	Duration	Success rate
daoTests	22	0	0	4.425s	100%
daoUtilsTest	16	0	0	0.888s	100%
dataClassesTest	4	0	0	0.002s	100%
requestObjectsTests	5	0	0	0.358s	100%

Generated by Gradle 6.8 at 06-May-2021 15:31:54

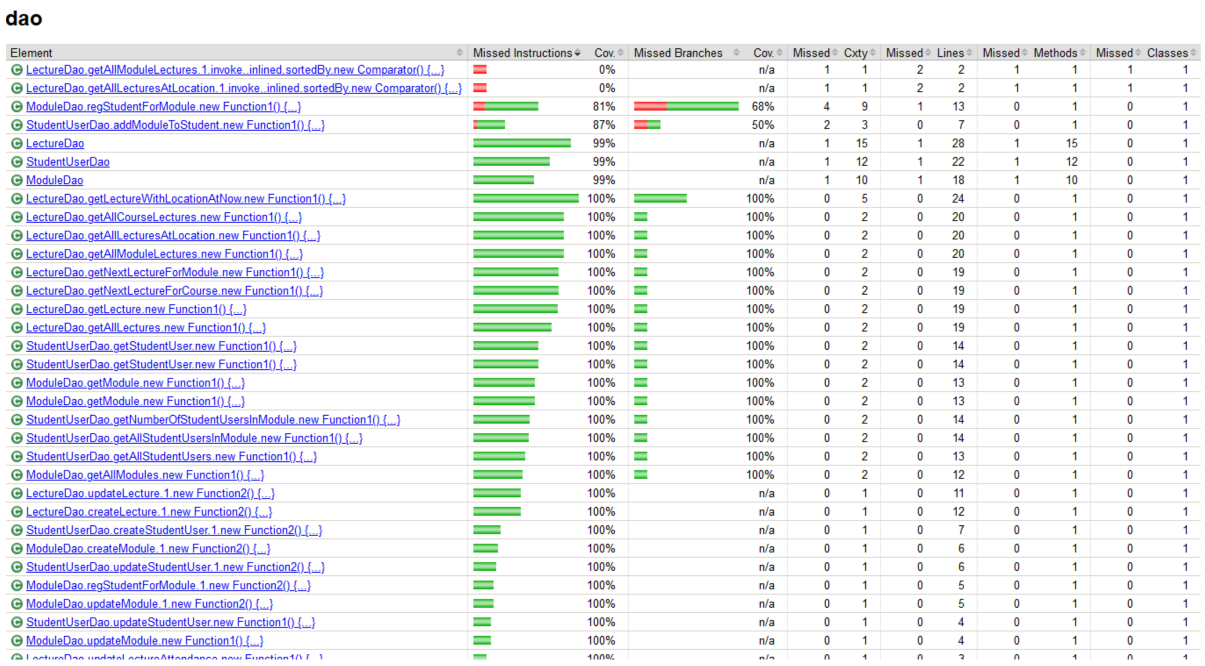
For testing the backend I used Junit5, MockK,a postgreSQL test database and a postman collection. I used Jacoco to pull code coverage metrics for what my unit and integration tests cover. This does not include code that is covered by the postman collection. The postman collection can be found in the testing files directory

I uploaded all jacoco html result output in a separate testing doc directory but here are some screen grabs.



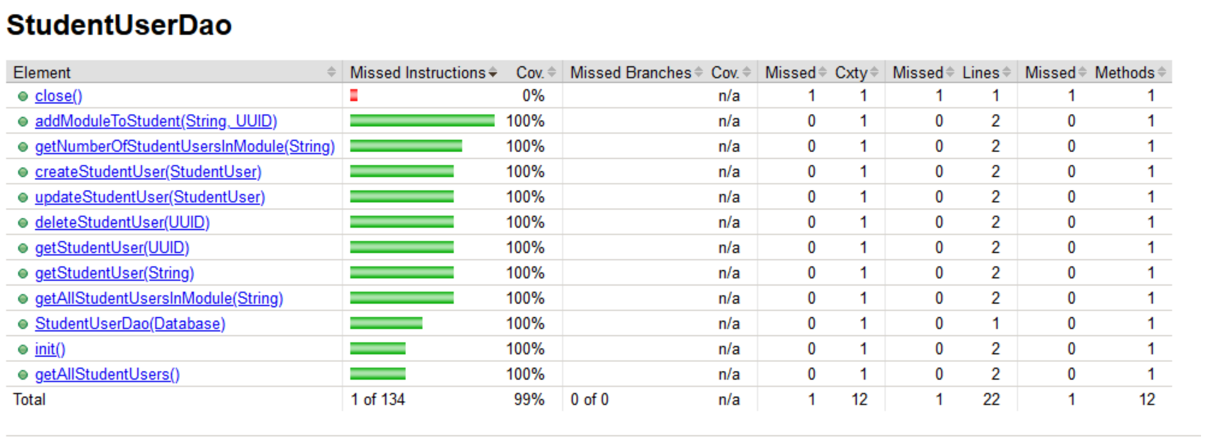
API

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
routes	🟥	0%	🟥	0%	436	436	375	375	38	38	38	38
default	🟥	25%		0%	7	11	18	23	6	10	4	5
dao	🟩	97%	🟥🟩	87%	11	119	4	377	5	90	2	56
daoUtils	🟩	85%	🟥🟩	67%	18	41	5	55	1	10	0	3
dataClasses	🟩	90%	🟩	66%	7	34	1	32	3	28	0	3
requestObjects	🟩	95%		n/a	5	40	0	82	5	40	0	10
table	🟩	100%		n/a	0	28	0	25	0	28	0	3
Total	5,558 of 10,349	46%	764 of 865	11%	484	709	403	969	58	244	44	118



dao

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
LectureDao.getAllModuleLectures.1.invoke..inlined.sortedBy.new Comparator() {...}	🟥	0%		n/a	1	1	2	2	1	1	1	1
LectureDao.getAllLecturesAtLocation.1.invoke..inlined.sortedBy.new Comparator() {...}	🟥	0%		n/a	1	1	2	2	1	1	1	1
ModuleDao.regStudentForModule.new Function1() {...}	🟥🟩	81%	🟥🟩	68%	4	9	1	13	0	1	0	1
StudentUserDao.addModuleToStudent.new Function1() {...}	🟥🟩	87%	🟥🟩	50%	2	3	0	7	0	1	0	1
LectureDao	🟩	99%		n/a	1	15	1	28	1	15	0	1
StudentUserDao	🟩	99%		n/a	1	12	1	22	1	12	0	1
ModuleDao	🟩	99%		n/a	1	10	1	18	1	10	0	1
LectureDao.getLectureWithLocationAtNow.new Function1() {...}	🟩	100%	🟩	100%	0	5	0	24	0	1	0	1
LectureDao.getAllCourseLectures.new Function1() {...}	🟩	100%	🟩	100%	0	2	0	20	0	1	0	1
LectureDao.getAllLecturesAtLocation.new Function1() {...}	🟩	100%	🟩	100%	0	2	0	20	0	1	0	1
LectureDao.getAllModuleLectures.new Function1() {...}	🟩	100%	🟩	100%	0	2	0	20	0	1	0	1
LectureDao.getNextLectureForModule.new Function1() {...}	🟩	100%	🟩	100%	0	2	0	19	0	1	0	1
LectureDao.getNextLectureForCourse.new Function1() {...}	🟩	100%	🟩	100%	0	2	0	19	0	1	0	1
LectureDao.getLecture.new Function1() {...}	🟩	100%	🟩	100%	0	2	0	19	0	1	0	1
LectureDao.getAllLectures.new Function1() {...}	🟩	100%	🟩	100%	0	2	0	19	0	1	0	1
StudentUserDao.getStudentUser.new Function1() {...}	🟩	100%	🟩	100%	0	2	0	14	0	1	0	1
StudentUserDao.getStudentUser.new Function1() {...}	🟩	100%	🟩	100%	0	2	0	14	0	1	0	1
ModuleDao.getModule.new Function1() {...}	🟩	100%	🟩	100%	0	2	0	13	0	1	0	1
ModuleDao.getModule.new Function1() {...}	🟩	100%	🟩	100%	0	2	0	13	0	1	0	1
StudentUserDao.getNumberOfStudentUsersInModule.new Function1() {...}	🟩	100%	🟩	100%	0	2	0	14	0	1	0	1
StudentUserDao.getAllStudentUsersInModule.new Function1() {...}	🟩	100%	🟩	100%	0	2	0	14	0	1	0	1
StudentUserDao.getAllStudentUsers.new Function1() {...}	🟩	100%	🟩	100%	0	2	0	13	0	1	0	1
ModuleDao.getAllModules.new Function1() {...}	🟩	100%	🟩	100%	0	2	0	12	0	1	0	1
LectureDao.updateLecture.1.new Function2() {...}	🟩	100%		n/a	0	1	0	11	0	1	0	1
LectureDao.createLecture.1.new Function2() {...}	🟩	100%		n/a	0	1	0	12	0	1	0	1
StudentUserDao.createStudentUser.1.new Function2() {...}	🟩	100%		n/a	0	1	0	7	0	1	0	1
ModuleDao.createModule.1.new Function2() {...}	🟩	100%		n/a	0	1	0	6	0	1	0	1
StudentUserDao.updateStudentUser.1.new Function2() {...}	🟩	100%		n/a	0	1	0	6	0	1	0	1
ModuleDao.regStudentForModule.1.new Function2() {...}	🟩	100%		n/a	0	1	0	5	0	1	0	1
ModuleDao.updateModule.1.new Function2() {...}	🟩	100%		n/a	0	1	0	5	0	1	0	1
StudentUserDao.updateStudentUser.new Function1() {...}	🟩	100%		n/a	0	1	0	4	0	1	0	1
ModuleDao.updateModule.new Function1() {...}	🟩	100%		n/a	0	1	0	4	0	1	0	1
LectureDao.updateLectureAttendance.new Function1() {...}	🟩	100%		n/a	0	1	0	3	0	1	0	1



StudentUserDao

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
close()	🟥	0%		n/a	1	1	1	1	1	1
addModuleToStudent(String, UUID)	🟩	100%		n/a	0	1	0	2	0	1
getNumberOfStudentUsersInModule(String)	🟩	100%		n/a	0	1	0	2	0	1
createStudentUser(StudentUser)	🟩	100%		n/a	0	1	0	2	0	1
updateStudentUser(StudentUser)	🟩	100%		n/a	0	1	0	2	0	1
deleteStudentUser(UUID)	🟩	100%		n/a	0	1	0	2	0	1
getStudentUser(UUID)	🟩	100%		n/a	0	1	0	2	0	1
getStudentUser(String)	🟩	100%		n/a	0	1	0	2	0	1
getAllStudentUsersInModule(String)	🟩	100%		n/a	0	1	0	2	0	1
StudentUserDao(Database)	🟩	100%		n/a	0	1	0	1	0	1
init()	🟩	100%		n/a	0	1	0	2	0	1
getAllStudentUsers()	🟩	100%		n/a	0	1	0	2	0	1
Total	1 of 134	99%	0 of 0	n/a	1	12	1	22	1	12