

Assignment 2

by Nathan Morris

Submission date: 10-Dec-2022 09:50PM (UTC+0000)

Submission ID: 191830667

File name: 42098_Nathan_Morris_Assignment_2_472250_819649561.docx (519.09K)

Word count: 3225

Character count: 15171



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

University of Wales Trinity Saint David
School of Computing

ACCA7000 ACCA7001

ACCA7002

Assignment 2: Block Test

Software Dev and Testing/Agile SEP/Database Systems

By Nathan Morris (2222137)

Table of Contents

1. Introduction	3
2. Main Report	4
2.1 Information for reviewer	4
2.2 Stories & tasks completed - total story points completed	4
2.3 Evaluation of design – User story's	5
2.4 Evaluation of design – Database and Database Testing	7
2.5 Relative Contribution to project	9
2.6 Self-reflection and evaluation of Agile processes used.	9
3. Conclusion	11

1. Introduction

This report will evaluate the design of the group project and the project database. The report will provide a detailed discussion of three user stories that were written and used during the project. It also sets out to evaluate the agile processes used and the benefits of the agile sprint process. The report will provide details on tasks completed including the number of user story points completed and an evaluation of contributions made towards the project.

2. Main Report

2.1 Information for reviewer

Miro board: https://miro.com/app/board/uXjVP8q41cQ=

GitHub username: FuzzyString

GitHub repository: <https://github.com/glugg12/UniCarRentalProject.git>

Video link: <https://youtu.be/hQ2pat-6ReM>

Database access: Database has been set up in the cloud.
Reviewer will need a file named db.properties saved in
codeJava/carRentalProject/src/main/Resources/
With the following details saved in the file:
User = postgres
Password = admin

Time of writing: Saturday 10th December 2022 21:44

2.2 Stories & tasks completed - total story points completed (141)

- As a customer I want to view all vehicles at a site so that I can choose a suitable rental.
Tasks: After a customer has selected site show all available cars at the site with Registration number, vehicle type, Make and Model.
Story points: 8
- As a staff member I want to be able to mark a vehicle as out on maintenance so that I can see all vehicles that are on maintenance.
Tasks: View all cars out on maintenance or repair and include registration number, make, model, manufacturer and status.
Option to then book in vehicle from repair/maintenance
Story points: 8
- As a staff member, I want to remove a vehicle so that it cannot be accessed again.
Tasks: See a list of all available vehicles.
Enter registration so that vehicle is removed.
Ask for confirmation before vehicle is removed.
Story points: 8
- As a staff member I want to be able to mark a car as picked up so that the company knows it's with the customer.
Tasks: Modify status of car to be out for rental.
Story points: 8
- As a staff member I want to be able to mark a car as returned so that it can be rented out again.

Tasks: Find car based on status with customer.
Modify status of car to be returned/ready for rental.
Story points: 8

- Original database design allocated to create Staff table, staff role table, booking details table and to create a trigger and function to log when a vehicle status was changed. (See NMSQL file in codeSQL folder in GitHub repository).

2.3 Evaluation of design – User story's

The team started off by writing out every story it would take in order to complete the entire project brief. During the design phase, the team decided that there were two significant 'Users' that should be considered. The team categorised these users as staff member and a customer. The team decided to split the user story map into two separate maps, one for the customer and the other for the staff member. On reflection this was a decision that helped with the organisation of coding. When the code was being written, a lot of the work done on the menu's functionality was split into the two sections. This meant that when code was being written it was easy to understand how the menus and functions linked together and in turn helped with the readability of the code. The team then decided to split stories into three release stages, the team did not know how much work was achievable in the time period, by splitting the project into three release stages meant there was an achievable goal to create an MVP with the flexibility to do more if time allowed. The team then set out the sprint goal of achieving an MVP by the end of the week, with the ambition of releasing further versions if time allowed. I feel that the original design of an initial MVP released worked well, as the team is on track to achieve that goal, with sufficient time left to write a report. If the team had not split the task into an MVP and further releases, the team might not have ended up with a working product.

The first story that I worked on was as follows

As a customer I want to view all vehicles at a site so that I can choose a suitable rental

Attach Add a child issue Link issue

Description

Normal text B I ... A

Task

After a customer has selected site show all available cars at the site with

Registration number

Vehicle type

Make

Model

It was decided that allowing a customer to view all vehicles available for rent at a particular site was a key feature that was required for an MVP. After a discussion within the group, it was decided that the ticket was small enough to fit into one task. The task had several display related requirements that the team felt were critical for the customer. These were registration details, vehicle type, make and model. The team felt that this would be sufficient information for a customer to decide on selecting a rental vehicle. In a real-life situation, after the story had been completed, the functionality could be reviewed by a potential customer to

see if the functionality and information provided was at a sufficient standard. For instance, the customer could highlight that they would like to see the fuel type and colour. Based on this information a new story could then be added to include these requirements.

The second story that I worked on was as follows

As a staff member I want to be able to mark a vehicle as out on maintenance so that I can see all vehicles that are on maintenance

 Attach  Add a child issue  Link issue 

Description

Tickets to search for a vehicle to then book it in from maintenance and repair.

Tasks

View all cars out on maintenance or repair and include registration number, make, model, manufacturer and status

Option to then book in vehicle from repair/maintenance

The team decided that it would be important for a rental company to be able to view its vehicles that were currently out on maintenance so that accurate fleet information was always on hand. The story was split into two tasks, the first task was to view a list of all vehicles out on maintenance or repair, the second was to be able to book a vehicle back in for use once it was returned. The Story is a good MVP story, but on reflection, there could have been further stories included in the design to include more functionality. For example, there could have been another story to include the date that the vehicle had been booked out on maintenance. In its current form, the user is unable to tell if the vehicle has been out on maintenance for 1 week or 100 weeks.

The Third story that I worked on was as follows

As a staff member, I want to remove a vehicle so that it can not be accessed again

 Attach  Add a child issue  Link issue 

Description

Tasks

See a list of all available vehicles

Enter registration so that vehicle is removed

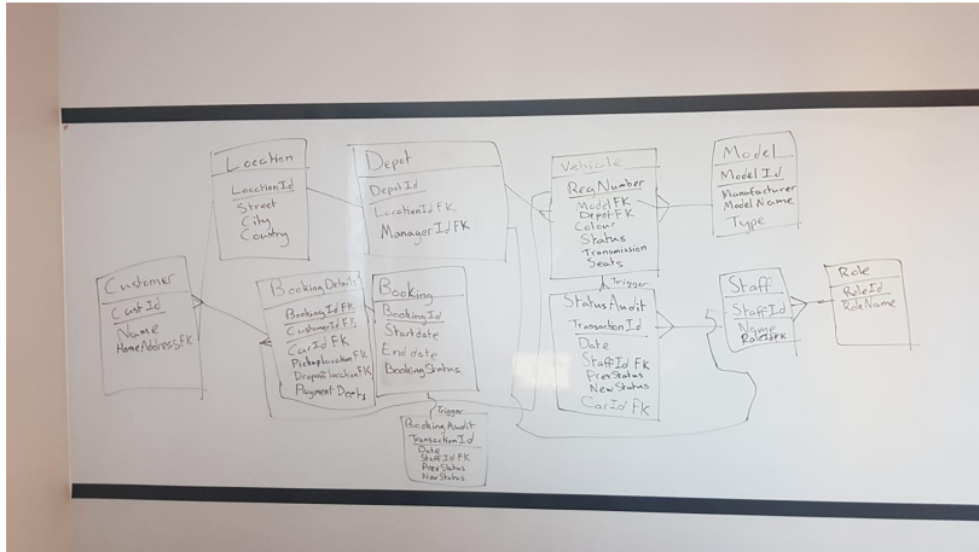
Ask for confirmation before vehicle is removed

The team decided that a core piece of the software's functionality should include functionality to ensure that a company's fleet records is accurate. It would be disappointing for a customer to choose a vehicle that is no longer part of the fleet. The story was broken down into three parts, the first was a task that would show the member staff a list of all current vehicles. This was chosen as the user might want to view a list of current vehicles to check if a vehicle has already been removed or not. The second task was to enable a staff member to remove the vehicle by typing in the vehicle registration number. The last task was added so that a vehicle cannot be removed in error and the user has a chance to check that they have chosen the right action. I think that the story and task were well thought out, a further story could have been added for later release to show all the vehicles that have been marked as removed and a reason why. In reality, this functionality would need to be included in the software. With no history behind the vehicle removals there would be a high risk of fraud, i.e. a member of staff could remove a vehicle and then take it from the company.

There could also be more stories around the functionality around security. Before the vehicle is marked out as removed there could be an additional log in so that only authorised personnel could access this option. The team did not include a story for this but on reflection I would add it in to a later release.

2.4 Evaluation of design – Database and Database Testing

The database design was conducted as a group exercise, on day two we had a physical meeting as a group and produced the following database design. Primary keys are underlined and foreign keys are marked with a FK notation.



At this stage the only concern I had with the database was that the vehicle status should have been put into a table on its own. My thoughts on this were that there should only be a standard set of vehicle statuses available to use which would be kept in its own table. The vehicle table could then have a status ID column set as a foreign key so that whenever a status was changed or updated it could then be checked against the statuses in the status table. Depending on the software design, the database design above would allow any number of different vehicle statuses to be added to each vehicle which could cause issues when querying the vehicle by status. The team decided not to make a status table, so I had to take this into consideration when developing the code for changing the vehicle status. I had to ensure that standard statuses were used in the code so that the user could not type in any status against the vehicle. I also wasn't particularly in agreement with the separate tables for booking details and bookings, I felt that this information would be best placed in one table rather than splitting it into two as there was no benefit in doing so. This was then later changed when the database was amended.

One of the team members did alter and write code around a new version of the database. The team member first informed the other team members of the changes on day four of the project, most of the team members code had already been written at this point and the original agreed database design had also been populated. The team worked through the issue and compromised on the design of the database. Most of the changes, if not all were improvements on the original design. However, as the team member had not informed anyone of these changes or intentions to change the database, some of the changes could

not be implemented as the project was too far into its development. The whole team gained valuable experience from this incident, the person who made the changes realised they needed to communicate more efficiently with the team, especially when changes to the database were being proposed. As a member of rest of the team, I realised that we could have taken steps to avoid the miscommunication, keeping in touch with the team member more frequently could have avoided this issue, as well as ensuring that the teams' goals and ambitions were clearly understood by everyone.

I conducted the following unit tests using JUNIT automated tests in Java

```
@Test
void testUpdateStatus() throws SQLException
{
    resetDatabase();

    Tests run to check different vehicle registrations and different statuses

    Assertions.assertEquals( expected: "Maintenance", DBTestCommands.updateVehicleStatusTest( reg_number: "ABC0123", status: "Maintenance", validation: "y" ));
    Assertions.assertEquals( expected: "WithCustomer", DBTestCommands.updateVehicleStatusTest( reg_number: "BALL813", status: "WithCustomer", validation: "y" ));
    Assertions.assertEquals( expected: "InDepot", DBTestCommands.updateVehicleStatusTest( reg_number: "HAL0117", status: "InDepot", validation: "y" ));
    Assertions.assertEquals( expected: "Sold", DBTestCommands.updateVehicleStatusTest( reg_number: "1234RAT", status: "Sold", validation: "y" ));
    Assertions.assertEquals( expected: "OnRepair", DBTestCommands.updateVehicleStatusTest( reg_number: "1234RAT", status: "OnRepair", validation: "y" ));

    //Testing the validation works with a capital Y
    Assertions.assertEquals( expected: "Sold", DBTestCommands.updateVehicleStatusTest( reg_number: "1234RAT", status: "Sold", validation: "Y" ));
    Assertions.assertEquals( expected: "WithCustomer", DBTestCommands.updateVehicleStatusTest( reg_number: "HAL0117", status: "WithCustomer", validation: "Y" ));

    //Testing that the status does not change if the validator does not equal Y
    Assertions.assertEquals( expected: "InDepot", DBTestCommands.updateVehicleStatusTest( reg_number: "ABC0123", status: "InDepot", validation: "n" ));
    Assertions.assertEquals( expected: "InDepot", DBTestCommands.updateVehicleStatusTest( reg_number: "ABC0123", status: "OnRepair", validation: "n" ));
    Assertions.assertEquals( expected: "OnRepair", DBTestCommands.updateVehicleStatusTest( reg_number: "1234RAT", status: "OnRepair", validation: "n" ));
    Assertions.assertEquals( expected: "OnRepair", DBTestCommands.updateVehicleStatusTest( reg_number: "1234RAT", status: "Sold", validation: "n" ));

    //Testing that nothing is returned when an invalid numberplate is used
    Assertions.assertEquals( expected: "", DBTestCommands.updateVehicleStatusTest( reg_number: "FAKE", status: "Sold", validation: "y" ));
    Assertions.assertEquals( expected: "", DBTestCommands.updateVehicleStatusTest( reg_number: "1234RA", status: "Sold", validation: "y" ));
}
```

Most of my user stories involved using a function that I created to change the status of a vehicle. All of the tests are testing that the function updates the vehicles status correctly based on different parameters. The first two blocks of tests check that a status is correctly updated when the information is passed into the function and the user validation is set to yes (The function asks a user to validate and confirm that the status they have chosen is correct).

The second block of tests checks that the status does not change when the user does not enter yes. The first test in this block sets the status to a new status and the second then checks that this does not change as the user validation is set to no.

The final block of tests check that no status is returned if an invalid numberplate is passed into the function. On reflection, had there been more time, I would have implemented error handling in the case where an incorrect numberplate is passed into the software to warn the user that the numberplate is invalid.

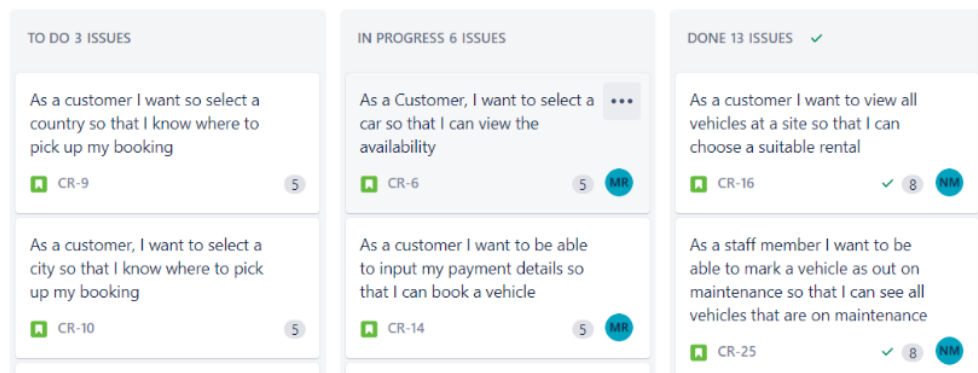
This project has also taught me the importance of conducting automated unit tests. I did the testing towards the end of the project. Early on in the project, before I had made the tests, every time I made a change to a particular area of code, I would conduct manual checks to see if everything was working correctly. Had I made the tests earlier on, the automated testing would have saved me from conducting manual tests each time the code was changed. This would have saved me a lot of time in aggregate. In future I will write tests as soon as I possibly can, potentially before the code behind functionality is even written.

2.5 Relative Contribution to Project

I completed a total of 40 story points out of the total estimated completed story points of 141 (28%). The database implementation was not included within the story points, however, all team members were present at the database design and everyone contributed equally to its design and population. There were some team members who were more technically experienced than myself and others who were less technically experienced than myself. In some instances, the more technically experienced member contributed more to the project in terms of adding more complex code to the codebase. They also spent time helping less experienced people with issues. I feel that I have contributed equally to the project, I received some help and guidance from a more experienced member of the team but also helped one of the team members who was less experienced than myself. Overall, the team worked very well together, everyone pulled their weight and even when there was a misunderstanding around the database design, everyone played a role in working out a solution and helped to correct any issues. There were no issues with people not working hard or contributing to the goals of the team. One member of the team is working behind the others in terms of deadlines, the aim was to have all code submitted by Friday, all team members except for one had achieved this goal. There appears to have been no technical or skill set issue behind the lag in work submission, the member of the team has been asked if they need any help or if they are struggling with anything, but everything seems ok.

2.6 Self-reflection and Evaluation of Agile Processes used.

Overall, I am pleased with the work that I completed and the work that the group completed at the time of writing. The group is on track to complete its MVP, 3 members of the team have completed their coding and tests, with one member's final coding and testing to be submitted. The agile processes helped the team to plan and complete the MVP, without using agile processes the team would have found it difficult to move in a clear direction and work to achieve a clear goal. The Story mapping done at the start of the project was extremely important, as explained previously in my report the mapping of stories into an MVP with further releases gave the team a clear goal to achieve. By mapping stories out into multiple releases, the team had the flexibility to extend its goal by working on an additional release if the MVP was completed. The scoring of the stories helped the team to plan its work, each team member took on a similar number of stories. Without the group exercise of scoring the stories it would have been difficult to gauge the amount of work required to complete each story. The team used a Kanban board which was extremely helpful to track progress and record how the team was getting on. See the below screenshot of the team's Kanban board.



Some of the teams at the DVLA have an additional column for 'testing' where stories are moved to after they have been completed while testing is being conducted. On reflection, I think this would have been helpful if the team had also included a column for testing, as on the final few days, most team members had completed stories but had not done the testing. If there was a testing column, the team would also have had visibility of the split between still coding and testing.

I am pleased with the code that I have written, and I am especially pleased with the unit tests that I have written for the SQL functions. I had not written tests for SQL functions before, it was a challenge to create the tests, but I managed to write the tests myself, a process that was very rewarding once completed. As previously mentioned, the project environment has shown me the added value of writing unit test for functions as soon as a function has been created (or even before). Before the test was written I lost a lot of time having to manually check my function and code each time I amended them. I will now endeavour to write unit tests for my code as early as possible.

During the previous three weeks, everyone took part in a project at the DVLA. One of the big takeaways that everyone learned was of the importance of continuous integration and frequent merging into the main code base. During the three-week project, everyone had merge issues at some point, and everyone quickly learned that smaller, shorter integrations were much easier to complete and took less time to resolve. During this project everyone produced branches and merged their code at least once a day (after day 5 there had been a total of 35 pull requests which equates to well over one merge a day per team member. This project reiterated the importance of continuous integration, the frequent merge requests meant that the team spent less time dealing with conflicts and more time writing code towards the goal of the project.

3. Conclusion

In conclusion, at the time of writing the team is on track to produce an MVP. At this time, we have a fully functional menu that navigates to a variety of different functions, which all link back to user stories included on the user story map. Agile processes such as user story mapping and tools such as Kanban boards worked well for our team, the agile processes broke down the large task of completing the project into smaller pieces of work. The team also abided by continuous integration and merged into the main branch frequently, so that time spent working out merge conflicts was minimised. The team designed a database where I had a minor concern but overall, I agreed with. The team did have a minor issue with a change to the database design, however, the team worked together to resolve the issue. The conflict highlighted to everyone the importance of getting the database correct and finalised from the start of the project. The team produced unit tests for the SQL functions, and I learned of the added value of creating unit test early in the development process.

Assignment 2

GRADEMARK REPORT

FINAL GRADE

86 / 100

GENERAL COMMENTS

Instructor

Sw Des Eval (30):28

Database Eval (30):23

Code & Test (30): 25

Reflection (10):10

Feedback:

=====

Good software design and discussion of user stories. Could have provided a little more discussion of the story point estimation and sprint planning.

We were hoping more formally presented database design, but there is evidence of an ongoing design process within the team.

Very coding and testing demonstrated.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7
