Critique of TaxiCloud TNG

Requirements and Specification

The TaxiCloud TNG group had a clear idea of what they wanted from the project, as can be seen from the abstract they wrote. They wanted to achieve a cloud-based system that was fully integrated with third party web services that modernized and optimized the way the taxi company operated. The group set a clear list of objectives based on what they thought the software should do paired with what the client wanted out of the system. The original code had a lot of bugs and errors causing problems including: users not able to change email address, not all bookings were shown, errors were visible to a customer visiting the page, the email system didn't work and there were security flaws. The group listed all the problems with the system within their documentation and put an asterisk next to the problems they prioritized to fix as soon as possible.

After the first iteration of their code (named Iteration 4), they were able to fix all the bugs within the current system, fixed the customer and company tables, bookings were fixed and validation was able to run on all elements of the forms. They were, however, unable to fix a few problems such as "Predictive Journey" and "Predictive Fuel".

The next iteration showed more progress for the group. Fully implemented API was achieved. Permissions of the users was fixed and administrators were given full control to edit permissions of other users.

Iteration 6 achieved drivers being able to input unavailable dates that they cannot work, such as holidays and such. Along side this, an invoice function was added, which could be done for a single booking or multiple.

Iteration 7 came after the group had a meeting with the client so they could outline what functionality they wanted from the system. The group noted it all down and those goals became the objectives for iteration 7. These included: the price being in pounds and not in pence, adding a return journey for a booking and a day report. All objectives were met with the latest iteration of code, leading to a successful project.

Design and Implementation

The documentation for this group is exceptional. The layout is clear, professional and easy to read. There is a table of contents with clear headings allowing ease of access. Each iteration has a base set of documents, such as Introduction, Planning – Research, Requirements and Evaluation. Every step of the process is documented, along with changes, problems had with the code or research, how they solved the problem and what they achieved per iteration. At the bottom of each document is a Revision History, showing who wrote it and when, followed by if it was edited and who is was edited by, finishing with who proof read it and when.

Technologies Used

The team had a wide variety of different software they could have used to complete this project, but there were documents that presented the advantages and disadvantaged of each alongside any other relevant information that needed to be taken into account. When deciding which software to use, they stated clearly (normally in bullet points) why they went for the software that they did. They decided to use OpenID, which differed from the TaxiCloud app.

Testing

The group performed thorough testing on their code, with testing with both a local copy of the software and a live test on the website. The final local testing document consisted of over 40 pages worth of code with only a few failing but for minor reasons (one failed due to a spelling error in the error message). But as the API agreements stated, no code can be completely without errors. There will always be errors in code but this group did everything possible to ensure that the software was appropriately tested. They had local and live site testing with each iteration, testing the same things each time (Regression Testing) and all the new features.

With the live site testing, there were multiple errors, most of which concerned actually making a booking, all failing when the site created an incorrect booking with the pickup and drop-off location being incorrect, but these bookings only concerned bookings made by a customer that was associated with a company on the website. Similar errors were experienced when attempting to edit a booking for a customer that was associated with a company on the database. Another major error was that when creating the return journey, the pickup and drop-off point were the wrong way around.

API

The API is very well designed. They follow good programming etiquette when it comes to naming the functions, making it very clear what each function does by just reading the name of the command, for example: getUserWithID().

The reference document that the group created lists each function and how it connect to use the said function, the parameters specification, the expected response, which is supported with JSON examples. They then dedicate the entire fourth section to how any possible errors are handled.

Giving how to connect to the function, how to request information and what expected response should be received will make it very easy when working with it so if any unexpected problems do occur, it should be easily diagnosed.