Assignment 5 Final Report

Topic 1. Agile software development practices

Several agile software practices were used in the app development. These include version control, delivering working software frequently, implementing user stories and creating a project backlog.
Version control using Github was crucial as it allowed other members of the team to share work quickly in a centralised environment that can be accessed anywhere. It was a record of the progress and showed the changes other members made, allowing you to isolate bugs in your code that may arise from the code of another team member. Delivering working software frequently also worked well as some user stories/tasks needed previous work (i.e. sending information to the webpage required the FaceDetection function before it could be implemented). By constantly pushing working code to the Github, other members can start to work on tasks that required the current module, as they would have a basic idea of the structure, inputs and outputs of the module from the work-in-progress and can code to these attributes. Finally, creating user stories and a project backlog helped split the requirements into smaller, obtainable goals which increased the efficiency of workflow and showed progress. Through the backlog, we could prioritise tasks, get an estimate of the amount of work each task needed and can thus timeline the development of the app so there are no chokepoints.

Working in an academic environment and working in industry was different regarding software development practices. I had multiple important assessment pieces due before each iteration submission meaning I had to plan my contribution to the development differently. In a normal development environment, progress is constant throughout the sprint since there are no other deadlines. Having important assignments due meant that I had to work in short intervals between each deadline. Thus, I started early and pushed code in short bursts. Since I had to start early, I was not able to wait for other members of the team to complete their sections of code, so I had to make a general start on their sections to complete my code so I had an equal contribution. Another option was to offload the work on my team members, which would not be fair and make it harder for them as they also had deadlines from other units to meet.

There were some agile practices we didn't adopt during development. Since the requirements didn't change, our plan from the start of development didn't need to be changed either. We also didn't have any daily meetings where we talked over our progress and what needed to be done. Since each member had different schedules, assignments and commitments, we couldn't find a suitable time to meet up every day. Instead, we communicated over instant messaging, however this was not as effective as meeting in person.

There were some aspects of the development that I would've done differently. I would've communicated more with the team, regarding my schedule, planning with them to construct a better timeline earlier in development to avoid these short bursts of work and

make it more consistent. I would've schedules more face-to-face meetings so we can refine the work that needed to be done and finish the project quicker.

Topic 2. Working in teams

It took us some time after the assignment was posted before we started to work on it. This is because many of us had other assessments and commitments that they had to complete and could thus not focus on the project. We also couldn't meet in person as we had different schedules and lived in different areas. Instead, we had to communicate and coordinate over Facebook. The first task was to generate a storyboard using Trello and split the given user stories into manageable tasks. This was done by the first person who had time free to do it. We sorted the tasks into different sizes and priorities and each member could pick whatever tasks they wanted to work on according to their strengths and when they were available. As we completed each task and put it in the 'Done' list in Trello, we sometimes had to break a further task into smaller tasks if it was more complex than what was first realised. As the deadline for each iteration approached, some members were having trouble finishing their work (running into a bug, or unable to get a certain module working), which meant I had to help them complete it as it was a core feature and needed to be finished before I could do anything further. We used Github as a platform to share code and as a means of version code. This allowed someone to push their problem code into the repository and for another member to pull and look at it instantly which improved workflow immensely. Since team members could pick and choose tasks, and we had a three-person team instead of a two-person, the tasks were not distributed equally. Since some tasks were very similar (i.e. face detection and text to speech), or depended on each other (i.e. setting up the server and client), there wasn't many tasks left for the third person to contribute. There were also a lot of problems both team members encountered during development that everyone else had to work on as the deadline was approaching. Task deadlines were not strictly enforced so sometimes we had to wait for another team member and workflow as interrupted. I think the biggest problem we faced was the lack of communication between all team members. Since we couldn't meet up, there was no motivation to communicate what tasks we were completing, what needed to be done, the problems we were facing etc.

These practises would not hold up in a small company. There would be more communication between team members as you are working in a shared space, and have no academic commitments which meant that you could focus on the project. The project outline and timeline would be planned more thoroughly and more fairly to everyone on the team to ensure a smooth workflow. Team members would be forced to give progress reports so everyone else on the team would know what needed to be done, prevent overlapping of work and know how close the project is to being done. The code would be of higher quality and be finished quicker since there is more motivation to do well in a company than university.

Topic 3. Design

The purpose of the project was to design and implement an application using IBM's Watson cloud platform. The application used a client-server methodology, where a webpage will be used to display information and interface with the user, and the server will handle all calculations and backend operations.

General operation of app:
Image files are selected in the client and uploaded to the server using socketio-file-uploader. The image is saved in the server root directory and the FaceDetection function is run on the saved image. The function uses IBM Watson's Visual Recognition API to request information on the sent image and returns information such as the age, gender and location of each face. This information is formatted into several arrays which can be accessed using the FaceDetection callback. The arrays are formatted again by face in the callback, joined together to form a sentence and sent to client to display on the webpage. The server also sends the sentence to IBM Watson's Text to Speech to generate a voice file. The voice file is streamed to the client using socket.io-stream where it is reconstructed and played. The location of each face (generated by Watson) is sent to the client where rectangles can be drawn around each detected face in the original image using HTML5 canvas.

The application was designed in a modular approach to maximise coherency with other code. Thus, different functions and modules can be added to the app without affecting other pieces of tested code. Socket.io was used for client-server communication. The benefit of using Socketio is that it allows the developer to add and remove functions relatively easily once the server is operation. By simply importing the new feature module, adding a socket.on('*feature*') and a socket.emit('*feature*'), in the client and server code, additional functionality is added, therefore the app can support many variations.

Since the code is modular, bugs are relatively easy to isolate and fix. Winston is used to log information at every stage of operation so errors can be detected quickly. Informative variable and function names were used and a clean, clutter-free coding style assists in bug fixing as well. Increasing the amount of logging in the future can make it easier to solve problems faced while constructing the application.

The visual design of the app is relatively simple, with a single button to browse for the file. Once the file is selected, the webpage automatically displays information and plays the audio file. Some steps could be taken to make the interface more visually appealing such as adding additional colour, centring the HTML elements and displaying the input and output side-by-side. The webpage also displays the raw image, meaning a large image will possibly fill the webpage making you scroll down, while with a smaller image, everything can fit on the same page. This might make it frustrating to use if the user has a small screen but is uploading a high-resolution file. To make the interface more user friendly, we can scale both images to a fixed size so that the placement of the other elements stays the same between images. The visual design was created with the intention of appealing to an adult userbase-children may find it confusing to use since there isn't any directions on how to use the website, and the font is relatively small. Hence adding more labels, increasing the font size and colour which attracts the eye will make the app more appealing to a wider audience.

Implementing these changes would be relatively easy since every element and variable is labelled and modular.