SOFTWARE ENGINEERING
Learning Journel

for

Disappster

Version 1.0

Adam Ryan (14395076)

COMP30830

April 23, 2021

Contents

1 Introduction

1.1 Purpose

As part of the COMP30830 module, Group 16 developed a DublinBikes application named 'Disappster' to track DublinBikes bike availability and predict bike availability over the next five days via the creation of a web application. As part of the deliverables for this project, an application was deployed on AWS, a report was submitted, and a Confluence was created.

This document is designed to chart a personal learning journal of this application's development, focusing on the following areas:

- My personal experience of the assignment.
- What I learned under the following categories:
  - Technical aspects.
  - Teamwork aspects.
  - Professional aspects.
- My own contribution and areas to improve upon.
- Opinions which are not featured in the group submission.

2 Personal Experience

This chapter is designed to chart personal experiences of the project.

2.1 My Experiences

The completion of the Dublin Bikes application was overall a highly enjoyable process and has been the standout aspect of the course to date.

Coming into the project, I have had previous experience leading the backend development of a Django-based web application hosted on Azure for a financial services company. From this experience, coming into the project I had some familiarity with a few of the key processes which would be involved in the development of the DublinBikes application, yet as I'm currently employed full-time, I was slightly concerned at the beginning regarding both the workload of the project (particularly when taking into account other modules) and also how external commitments would impact upon the members within my team, but I was also excited as a way to bring together many aspects of the degree. An area of particular concern I had was trying not to 'steer' the project, and was particularly conscious of trying to avoid the 'one man band' or 'every man for himself' team structure touched upon in Lecture 6 Slide 42.

Having seen projects managed via Excel Trackers with documentation in Word/PDF documents, Smartsheet tracked projects, and Atlassian-managed projects, and the vastly different experiences and impression that is given as a team member (with the Excel/Word managed projects feeling particularly chaotic, while well-managed projects in Atlassian made larger-scale projects far more manageable and much easier on the development team), one thing I wanted to do very quickly was to establish a strong Confluence structure for documentation of every aspect of the project and also to establish a Jira board with automations for Epic-monitoring. This was largely done as an initial step following consultation with my team members both to try and reduce the project management overhead on a three-person project, and also to provide a production-like structure to the project documentation and issue tracking. In hindsight, while I believe that many aspects of Confluence went largely untouched by the team, I think the meeting documentation in particular turned out to be a very valuable resource over the course of the project and Confluence was a useful resource in the ultimate delivery of the project report (albeit I think Confluence itself was a bit more useful in charting this area). I believe part of the foundation of a strong and scalable project is in the creation of a significant body of documentation detailing why the design decisions were made, what the system architecture and design consists of, and how a project was managed in order to trace the project path and think Confluence offers the best solution for this purpose, however I think there was potentially a bit of skepticism during the project within the team as to how beneficial documenting elements on Confluence (outside of meetings) would be given the time commitment. Initially, while I think for the first two sprints many aspects of the setup was of limited value given our decision to develop Sprint 1 and 2 features individually (where tickets are necessarily duplicated as a result) and then complete a Code Merge in Sprint 3, for the later sprints in particular it gave a very strong structure and was noted by the team a number of times as to how 'nice' the ticketing system was in aiding development and in avoiding feature duplication.

One of the aspects of the project which I wish had been completed was a greater emphasis on a Discovery Phase in Sprint 1. I think completing an SRS or at least capturing User Stories for the entire-course of the project, and creating Epics for Sprint 1-4 from the outset would have helped both in directing development and also possibly helped direct the sprint planning meetings to cover what features would be archived and really help fine-tune the features which would make it into the production release. Along similar lines, the lack of meetings with the Product Owner during Sprint 2 over the course of the mid-term resulted in the Jira board being largely neglected when combined with our DevOps structure, and I think this could have potentially been avoided by directing Daily Standup discussions around the Jira board tickets.

Over the course of the application's development, I believe the first two sprints went quite well as we had established our scrapers, constructed skeleton FlaskApps, and had identified and developed solutions early for some key areas. I would have preferred if, in the early sprint stages, we had established a full schema before beginning any development or implementation. In particular, I would have preferred if the backend structure and tables were setup via flask-sqlalchemy using Classes and Models, primarily to enable more Pythonic interaction with the database and to more neatly structure the application (while also allowing and defined natural linkages), which would have been more easily implemented if the database schema was initially established (as otherwise deployment packages are required). While I investigated this during Sprint 2, at that stage the learning curve for the team as a whole would have likely been too high to justify a shift. Similarly, I think fit the overall user journeys had been established from the beginning, we could have added some of the user stories from later sprints into Sprint 1 and 2 (frontloading the project development) which I think would have been very valuable in fine-tuning some of the features and allowing for a UAT stage at the end.

In Sprint 3 and 4, I felt the application development was quite pressurised, particularly as other assignments and external factors became much more significant during this time period. Because of the pressurised nature of Sprints 3 and 4, I largely focused on backend features in these sprints (in particular the model development, queries to enable data retrieval, and methods to enable graphing of features) and in the project management side as I believed I could push features most quickly here rather than engaging heavily in the front-end design (beyond some the initial changes), but now I believe it likely would have been better to develop many of these aspects in Sprint 2 to allow Sprint 3 and 4 to focus more heavily on refinement which I believe would have produced a stronger application.

The time-management aspect which was an initial worry became a very significant factor in Sprint 4 in particular, where the combination of closing out open features, writing the report, completing other projects, and final bug-squashing resulted in quite a stressful final few days of the project and I believe some of these areas could have been delivered to a higher quality if we as a team had potentially identified some of these earlier. A last minute bug was identified in the final few hours of the project whereby one of the Javascript functions was written such that when different stations were selected consecutively the model would get continuously called for each station and while the correct result would ultimately display, the UX was very poor as each result displayed in the UI before it arrived statically undermining user confidence in the model results. As a result of this last minute bug, a series of rapid hotfixes had to be pushed in the hours immediately prior to the project delivery which I think would have been very avoidable. If we had offloaded some of this work to Sprint 2, I think we could have implemented the model quicker, identified some of these bugs sooner, and potentially increased the selection of unit tests more substantially. Similarly, although I thought it was feasible to create the CSS for the About page within the final day (and I think it would have been as it's a small task) the presence of final bugs served as a barrier to its completion. Finally, the time management aspect resulted in a dissolution of the version control which had previously been fairly well stuck to as the need for hotfixes and final tuning resulted in the last week almost solely working on the main branch which was very risky development practice and far from best practice.

I thought communication was overall pretty good and I think Jane, Finnian, and I worked pretty well together. We were all pretty aligned on our approach to the project, I think everybody had a strong contribution to the project in developing features for the application, and there weren't any clashes during the project. I think it was made slightly more challenging by nature of being remote, but the combination of Discord, Google Mettings, and Jira/Confluence ultimately worked well. While I am not a massive fan of the Agile process (and prefer Kanban in practice) I think for creating a structured approach to the project we clear development milestones it worked quite well for the condensed project nature, and I think it helped structure a lot of the team meetings, discussions, and proposed Epics. The one area which I do think could have led to a stronger outcome was if a democratic team structure could have been replaced with a dedicated Jira-lead/developer covering the four sprints. In my experience, having a singular project lead helps to create a more unified direction on development projects and creates consistency in the managerial aspects.

Overall, I enjoyed the module. I would have much preferred to have been able to dedicate a lot more time into the project if external commitments were not a factor, however I think the ultimate outcome was satisfactory given the scope of the module (even if significant room for improvement in the fine-tuning of features, removal of bugs, and tidying of the front-end). While deciding whether to enroll in the course, this module was a key factor in deciding to choose the course as a strong motivator for completing the masters was to gain insight into development best practices and design principles, and having completed the module I'm very happy with how the project and module went overall even if there are some areas which I would change.

3 Learnings

This chapter is designed to chart my learnings during the project.

3.1 Technical Learnings

The key technical learnings from the project consists of developing an understanding of how Flask works, and how it can be used for the rapid development of web applications, and an appreciation for the differences (and appropriate contexts of usage) between the Flask and Django Python-web frameworks.

During the early stage project development, I learned how to access API information, access items in the JSON format, store information from the API into an internal database, and create scheduled processes that perform reliably. In identifying the presence of apparent duplicates within the data, I gained an appreciation for exploooratory analysis in a practical context and the importance of early-stage schema-design to allow for application flexibility a sensible, scalable web application; this helped ground the material covered in the Relational Databases and Data Analytics modules into a project/practical context. One area which was of significant interest was in learning many of the common Git-workflows which are used and how these are completed; although we ultimately went with a centralised branch structure, it was interesting to gain a context in how larger development teams can use Git to align their work, facilitate code reviews, and chart the project development. I believe the usage of a centralised repository given the team size was also useful in helping to enhance the need for continued communication between the team as we needed to communicate when we'd done git pushes/git pulls in order to coordinate and synchronise our changes which resulted in an ongoing conversation. Learning how Flask applications connect front-end systems to backend systems and the interplay between Python, Jinja/HTML/CSS, and Javascript was helpful in reinforcing my understanding of each of the programming languages and putting them into a unified context, and it was interesting to learn how to use Google APIs beyond just Google Analytics. One takeaway from the application development which has been significantly useful is in learning how modules can be employed and what structure is required. Previously, I didn't know as to how modules or individual packages could be installed from Git, and it has turned out to be very useful for easily deploying, sharing, and retrieving ML modules in my work's team.

During the mid-stages of the project, it was useful to be conscious of the risk of circular importing structures within the FlaskApp as initially integrating the tests.py file into the top of the app.py file resulted in the applicaiton not launching and I was not certain as to why this was. After investigating the issue, I learnt tha tthe issue was due to a circular importing between the unit tests and methods, so it is helpful to be mindful of the import structure in future projects. The insight into testing modules in Python and test-driven development was useful. Although I don't believe we had many bugs at the backend, I do think it made me conscious that potentially serving a default results JSON in the event of database failure could have been interesting as a method to protect against seemingly catastrophic failures and is likely something I'll implement in future projects.

During the late stage of development, I learned how to deploy, assess, and access models in a production environment. Although I've previously used XGBoost in the context of tax classification and customer clustering, it was useful to see that it is similarly performant and Kaggle-winning in the context of bike sharing predictions. The runtime costs associated with using GridSearchCV was also very good to be aware of as the generation of modules with even a small set of hyperparameters to test greatly increased the overall runtime cost given that one (XGBoost) model was being generated per station. I've not previously deployed pre-trained models that don't relearn based on whether their predictions were correct or incorrect, so it was really useful to learn how advantageous deploying pre-trained models can be for performance regions when result accuracy isn't incredibly critical and is something I've already been able to integrate into a churn prediction project.

The final, and perhaps most important technical learning, was in helping to understand the AWS system. Having not previously used AWS, it was very useful to see the wealth of platform features. Particularly when AWS is so commonly featured within job specifications, it was very valuable to gain an initial familiarity with the platform and I think it's likely something I'll investigate further after the module. One area which I do regret is not having fully explored the features early in the project development, but I was slightly worried about what the credit and cost implications would be however now it looks like I likely would have been fine to at least sample some of the offering.

3.2 Teamwork Learnings

In working as a group using a traditional Agile methodology to complete the development of the project, I was able to appreciate how for rapid development a structured Agile approach can be very useful in generating iterative releases. Although I do believe a Kanban structure with a dedicated product owner managing the Jira board and Confluence works better for large-scale releases, I came to appreciate the direction that the Sprint process provided in each stage of the application development. During the project, within Sprint 2 when the Jira board was poorly maintained, there were some instances of redundant development where Jane and Finnian had both worked on a feature which the other was developing and the implementations were ultimately mutually exclusive. While at the time this was certainly less-than-desirable, I think over the course of the project and long-term in terms of personal development this ended up actually being a useful experience to the team as a whole as it resulted in a real-life and practical example of why Jira/Confluence-style management software ends up incredibly important within application development an in keeping development on track while also emphasising the importance of teamwork and communications. The use of daily standups, retrospectives and planning sessions, and product owners helped us all stay up to date on what the current progress was of the project and what areas were to be developed or improved and it really emphasised the importance of communication in a development project.

One of the aspects of the project which I think I've learned from personally is that while initially I was very hesitant of an experience-gap between myself and other members of the team in relation to application development and was consciously making an effort not to steer the project, as the project progressed I think this became slightly de-emphasised as we each naturally gravitated towards different areas of the project which we could complete efficiently (e.g. Finnian worked heavily on some of the Javascript functions and some of the routes, Jane worked heavily on the CSS, page layouts, and some of the Javascript functions, and I worked predominantly on backend functionality and some small adjustements to the front-end and CSS). I think working with everybody as a team helped us not only leverage areas which we were strong in, but also helped us gain an insight into how other members of the team approached various challenges (e.g. frontend design, Javascript development, backend structures) which allowed each of us to improve our own programming as we learned different ways of addressing problems. A primary example where I believe this became present was at the end of the project when the bug was identified regarding the repeated calls to the ML model for iterative stations. I'd identified the bug but was unable to develop a solution, while Finnian was able to very rapidly deploy a solution which resolved the problem and in reviewing the code I was able to improve my ability to debug front-end problems. In hindsight, I would have liked more pair programming sessions beyond the Code Merge and some initial exploration of solutions as I think it would have helped give us more of an insight into how we were each viewing problems and why we were selecting certain avenues.

One of the most significant learnings was to develop a mindfulness around how team members may have different expectations towards the standard of the project outcome and what user stories are essential, and that it's important to balance the views of group members in order to achieve cohesion and focus on developing the features which truly are most important. Although there were a few examples where this arose, the most immediate example is in the development of the About Page. While this occurred in the very final push of the project to populate the page, my view was that the development of the CSS was only about two or three hours worth of effort and completing this was very easily achievable in the final hours of the project. I thought it was essential that CSS to be added in order to provide more of a professional appearance of the website, and that although this page contained little more than basic HTML, it was really important to adding to the appearance of a productionised application. In contrast, Finnian and Jane viewed the page as being relatively unessential, and that as the key features of the project were in the Main section, and few marks would likely be distributed to the About Page, it should not be a priority to finish it off and that they were okay with leaving it as-is. Although I still don't necessarily agree with this decision and think tidying it would have been a small job, I had to be conscious that as we were all peers and the majority of the group saw this feature differently, it was important to get behind the group decision in leaving this in a more 'raw' format and particularly with the identification of the aforementioned bug shortly after in the final hours of the project, it was possibly accurate. Similarly, a feature which I thought was really important was being able to dynamically add options to the graphs (e.g. choosing to view weather, choosing time periods, less of a 'raw' appearance of the graphs). While raw Javascript and the Google Charting library doesn't posess the power of d3, I was conscious that it was a significant contribution from Finnian and that it would not have been fair to the effort he had put into the feature to require other elements when both the product owner and other team member were satisfied with how the results displayed and having the graphs be rather minimal. Ultimately, the importance of balancing input and trying to restrain an impulse I have to try and steer development is something which was really emphasised over the course of the project. Potentially my team members view how successful I was in doing this differently, but it was definitely an aspect I was really conscious of, and while in hindsight I think we might have been able to develop more features add more of a professional sheen to the website I don't believe it would have been constructive to a positive team atmosphere which is incredibly important for not just delivery but also on-going maintainence and phase two development.

3.3 Professional Learnings

The single most substantial component of learning in the project was in improving my time-management and prioritisation. While completing this project, in addition to the five other modules which are underway in the MSc., there are currently three large-scale development projects which I'm leading in work. As a result, the semester was, in a single word, hectic. Balancing the many commitments both professionally and in my academic pursuits, and trying not to weigh down my team members was incredibly challenging, particularly in the latter stages of the project when many other assignments were also due. One area which I thought was very important was to be able to accurately assess how long many aspects of projects are expected to take in order to schedule appropriate time for each task, but also to be mindful of how expected timelines for yourself might not necessarily translate to expected timelines for others and a margin needs to be taken into consideration. Although it is arguably not reflected in this report, I think the key for me during the project was to identify that concise but impactful contributions which hit the key purpose are far more valuable than longwinded meetings/statements/explanations/etc which ultimately amount to very little. It's very important to be conscious of everybody's time and commitments, and to respect those. An area where this was particularly useful was in driving the usage of agendas within meetings that we had so there was always a clear set of points to discuss and actions to complete rather than having conversations which were somewhat aimless. I think one aspect that I probably would have done differently in this regard, as discussed in previous sections, is to have identified early on that the latter portion of the project was going to get quite stressful with other commitments and to have made more of an effort to frontload development tasks; while some aspects of the project management and documentation had been front-loaded, I think there was a lot of room for additional aspects to be created early that could have resulted in an improved project outcome.

4 Personal Contributions

This chapter is designed to capture my own contributions and areas to improve.

4.1 My Contributions

My key contributions into the project can be summarised in approximately chronological order:

- Establishing the Atlassian tools and building the Confluence structure.
- Creating the Epics (user stories) and tasks on the Jira board.
- Building the scraper which was implemented in production.
- Overseeing the Sprint 3 code merge and aligning developed code.
- Developing the backend application structure (Methods, Routes, SQL, Data Dictionary).
- Documenting some development steps on Confluence outside of Sprint Records and Workshop Records (e.g. System Design and Backend Architecture, Integrations).
- Developing dynamic SQL queries in the absence of models for data retrieval and methods to call these queries (e.g. data retrieval for charting and analytics).
- Segmenting multi-faceted code blocks into methods.
- The single unit test.
- The XGBoost model development and linear model development (and methods to pass these in the route).
- The forecast scraper.
- The Sprint 1, System Architecture, and Data Analytics sections in the report.

4.2 Areas to Improve

The key areas on which I need to improve are:

- Communication - I think I could have been more conscious in some of the communication with team members that they did not necessarily have a background in development and as such could have potentially been a bit more clear in providing context, task, and purpose relating to tasks. Similarly, I think I could have perhaps better emphasised why Confluence/Jira ends up hugely valuable during development tasks and how chaotic and stressful poorly managed projects can be (albeit I think this ended up being a naturally learned experience with the lack of oversight of the Jira board in Week 2).

- Effort Assessment - I think assessing task effort for other team members is something which I need to improve. I believe there were some occassions where I had very different expectations around how quickly certain tasks could be achieved compared to the rest of the team (e.g. I thought building scrapers should only really take two or three hours at most after data exploration is complete, but this was not accounting for the fact that the team had no experience with data scraping). Similarly, due to being overburdened/overcommited, there were some tasks which I thought I could complete very quickly which ended up taking slightly longer than expected due to other items having to be prioritised (e.g. connecting models to frontend).

- Frontend Development - I'm not a fan of frontend development and do not like programming using Javascript, but it is clear from this project how useful and functional it can be. Although I would rather focus efforts on learning React rather than pure Javascript, I think it would be incrediby valuable for application development to gain a deeper understanding of frontend development so I can support developers across all facets of applications (even if backend development would still be my preference). Similarly, while I'm familiar with d3, the learning curve is quite significant and I think there's a lot of room to learn how to develop more impactful graphics. One of the key reasons I believe frontend development would be highly valuable would be in the capability of self-development and deploying an internal data science platform rather than relying on BI tools, third-party software and integrations with licences, or the outputs of scripts, so this is really a key area to potentially improve. While this project didn't really touch those areas (e.g. frontend frameworks or more business-appropriate visualisations) I think it's an area where I'd like to learn more.

- Design Patterns - While I have a basic understanding of common design patterns, this is the area which I believe I really need to focus on improving the most. While I think I have a solid understanding of fundamentals, one area where I think I could grow significantly is in learning what are standard best practices in leading tech companies. A number of the projects I have completed (including this one) have focused significantly just on delivering a functional application. While I think that is okay for small to medium scale applications, I think without solid design patterns underpinning the application you massively cripple the application's capacity for scaling and also severly hamper your data science team. I think I need to more heavily investigate design patterns and best practices for large-scale scalability and high data volumes to be more adaptable and deliver higher quality applications rather than merely 'functional' applicatios.

- Alternative Backends Systems - I'm not so familiar with non-RDS databases or database structures. I need to learn and gain experience in these systems in order to deepen my understanding of backend engineering. These were touched on in the lecture notes and I've seen some advertised very heavily, so I'd really like the opportunity to learn some of these.

5 Miscellaneous Opinions

This chapter is designed to capture any other opinions not captured previously.

5.1 Other Opinions

Overall, I think Finnian and Jane really made a very strong contribution to the project, and the team as a whole worked pretty well. There are really three items which I really would have liked in the module:

- If the project began sooner than Week 5, I think we would have had more of an opportunity to explore more industrial frameworks (e.g. Django, or implement an MVC framework in Flask) that might have better captured what real-world application development. Although understandably the learning curve on these aspects is significantly steeper, I think it would have helped gain more of a perspective on the differences in approaches which are possible and allowed for easier adoption of using models to interact with the database and a more Pythonic development approach. I think if the module has started with the development of the scraper, and setting up a skeleton flask app in the initial weeks, the module could have then naturally moved on to focus on development principles and the business processes to facilitate a leeway into developing an SRS for the application which we are working on (while also ensuring that everybody has an application up and running to experiment with). After the business processes section is complete, the structure could then return to application development, and the assignment of creating an SRS for the application could then serve as the discovery phase for developing the Dublin Bike application. This has the downside of potentially making the module seem slightly disconnected (e.g. with a lull in the middle so it's understandable why this might be decided against) but I think it might have helped intgrate a discovery phase sprint into the application development.
- As mentioned before, I really would have liked if there was a lecture on design patterns and best practices. While it's a very deep area that could easily cover a module, aspects like how the backend should be structured, how the frontend should be structured, best practices for scalability and high data/traffic volumes would have been really interesting to touch on. While the testing-focused design element did touch on this, I would have really liked more insight into this area.
- I think the remote nature of the project slowed down development slightly as most of us had never met previously. While this is slightly reflective of how an actual development team may function, I think the sprint schedule and subsequent

readjustment of the sprint schedule probably added slightly to what felt like a rather frantic final weeks. Similar to the first point, I think if the module had dived into the application development and scraper development at the beginning, and then pivotted towards SRS/business process and then back to application development, it might have allowed for some earlier development on some more advanced features.

- Thank you! I really enjoyed the module, and thought it was delivered fantastically. The product owner we had assigned to our group (Karl) was not only very supportive in guiding the team but was very helpful in keeping us on track and checking that we hit our targets, so thanks very much for what was ultimatley a very rewarding module.