Neil Browning

B00652112

Project Monitoring and Control Report

COM547

Contents

[Introduction 2](#_Toc505540394)

[Actual vs. Planned Values 2](#_Toc505540395)

[Commitments 3](#_Toc505540396)

[Monitoring Risks 3](#_Toc505540397)

[Management of Project Data 4](#_Toc505540398)

[Stakeholder Involvement 4](#_Toc505540399)

[Reviewing Progress, Performance, and Issues 4](#_Toc505540400)

[Reviewing Accomplishments and Results 5](#_Toc505540401)

[Analysing Issues and Determining Corrective Action 6](#_Toc505540402)

[Appendix 1 7](#_Toc505540403)

# Introduction

This report intends to evaluate the progress and performance of the project so far, by comparing the actual values for several metrics against the values detailed in the original project plan that was completed before the project started. This evaluation should grant some more clarity into how the project is going objectively and may help to prioritise tasks as the project deadline nears. The outcome should help stakeholders to realistically estimate the value they expect to see from the project at the project’s deadline.

# Actual vs. Planned Values

In the initial project plan, a Gantt chart was created to schedule the project work tasks. The work required was broken down into manageable chunks and organised to be able to complete in 2-week sprints. During the implementation of the project, such tasks were migrated to issues on the Github repository that holds the project code, making it easier to track each work item with the code that was required to complete it and to see how much work had been done up to that point. No changes have been made so far to the requirements or scope of the project as it relates to the original project plan.

At this stage in the implementation of the project, there are 3 tasks remaining: student module feedback; email heuristics (the email reporting itself is already implemented); and the security considerations for data storage and transmission. At the time of writing (beginning of February), the first 2 of these tasks were scheduled to be completed already, putting the project behind schedule by 2 sprints. This can be attributed to a few potential factors. Firstly, the original estimates planned for 6d of work capacity per sprint (as can be seen in the Gantt chart). At 3 man-days per week for one developer, this was quite over-ambitious in terms of the real capacity that a developer would have to work on the project given full-time work etc. This is the main factor that affected the schedule. There are other factors that took some time away from the tasks defined in the schedule, but these are also advantageous to the project itself and will pay dividends as the project approaches completion. One such factor is a robust set of unit tests – almost 80 tests so far – which achieves 3 goals: they reduce the manual effort of testing new functionality; they help to prove the integrity of the implementation and verify the behaviour of the application if done correctly; and they aid in any code changes in the project to show where regressions may occur elsewhere in the code. The other factor here is the integration of a CI build tool, Travis, into the project. This Travis build runs automatically for new commits and pull requests to the repository and will display on the Github repository next to these commits. This automatic building and running of the project makes the unit tests useful (if they’re not run, there’s no point in having them).

Although behind in schedule, the factors listed previously (unit tests and CI) are beneficial to the correctness of the project and will save time that would be spent manually testing as the project nears completion. Another factor to consider here is that a project buffer was included in this schedule for ~2 months – while the project is behind schedule, there were also precautions in place for this, and given the points listed, I am still confident that the project will be completed for the deadline.

# Commitments

The aim of the project and the commitments made in the project plan have remained unchanged during the span of the project so far. The features developed so far were clearly outlined in the project plan, and there has been no features developed that were not specified in the plan.

# Monitoring Risks

The project plan included a risk assessment to determine any factors that could impede the progress of the project, so that they were realised and so that controls could be identified to mitigate these risks to a reasonable degree of impact or probability of occurrence.

The first risk identified was the risk of data loss, e.g. of the project code. Time spent re-doing work would impede the project schedule or could kill the project in the worst case of a loss of all project data. The control to mitigate this risk was the use of a repository that could act as a backup. Github and git have been used as version control and a project backup so far, and have even helped for a seamless development environment transition as the developer moved from a laptop to a desktop PC as their main platform in the middle of the project. The second risk identified was schedule slippage. While the schedule is behind (as previously mentioned), this slippage is a natural result of overestimating the development capacity available for the developer. The control, to refer to the project schedule, would not have helped in this case but the inclusion of a lengthy buffer will help to mitigate this. Feature creep was identified as the third risk, which could result in the schedule being lengthened. This risk was mitigated by the control of early requirements definition to understand the main features, along with frequent communication with stakeholders. The requirements were figured out at the start, and the project’s updates have been communicated to the main point-of-contact every few weeks, mostly in person. The fourth risk, unforeseen circumstances, is the most enigmatic risk that any project faces. The interference of circumstances that can’t be reasonably identified or accounted for in advance means needing to just try to mitigate it as best as possible. At the end of the project schedule, an almost 2-month buffer was added to reasonably address this risk, and there is still room here for this buffer to help protect against such circumstances. The final risk identified was with unavailable resources. For this project, the main technical project details were worked out very early (development language, purpose of project), and because of the nature of the project – a web application – there were very little resources required outside of internet access and a development machine needed for the project.

So far in the project, only one risk’s effects have been seen so far (schedule slippage), and its potential impact has been mitigated by the controls put in place. The probability that these risks will further affect the project to impede it are low – some of the risks have already been nullified e.g. data loss or unavailable resources, and short of a catastrophic failure of Github & the development machine simultaneously it appears that the remainder of the project is relatively safe as far as potential risks go.

# Management of Project Data

For managing project data, it was identified in the project plan that the implementation of the project would require secure storage of data, both at rest and while data is being transmitted. This facet hasn’t yet been covered but was planned for in the project plan using encryption at rest and Let’s Encrypt to supply certificates for HTTPS usage. The other consideration here was with the actual data used during developing / testing the project. The main approaches outlined in the project plan were to keep any development-related keys and any database out of the repository, and to use dummy data while developing the project. These practices have been adhered to, and result in maintaining the integrity and security of the project during development.

# Stakeholder Involvement

The project plan listed Professor Colin Turner as the product owner over the span of the project, and detailed that meetings would be scheduled on a weekly / fortnightly basis via Skype or face-to-face meetings. This was intended to help keep the project on track, and to resolve any queries regarding the project’s implementation. So far in the project, meetings have been set up every 2-4 weeks when needed (every week was too short a timeframe for sufficient work to be done in-between), and the meetings have proved very beneficial. Each meeting has consisted of a walkthrough of any new code / features and to ask any in-depth questions about the design and requirements needed from the system to make sure it stayed relevant to the goal of the final system.

# Reviewing Progress, Performance, and Issues

The project is going well so far, and although behind schedule, the project’s features completed so far are fleshed out and well-tested. The test coverage in the project has helped immensely with refactoring and reducing manual effort, with Django’s test framework being so robust as to cover things such as making requests / validating responses and email functionality out of the box. The majority of the requirements for the project have been met, and the developer is confident that the rest of the project will be completed in good time for the project deadline. The progress and performance of the project can also be traced on the repository through the Github issues at <https://github.com/nbrowning1/FYP/issues>. Each issue covers a piece of functionality, bugfix, or an enhancement, and have screenshots and descriptions of what is done to go along with the code that implemented that change. When issues are resolved, they are closed, and over the course of the project the history of these issues gives great traceability for the work done so far and would be helpful to any future project maintainers as well.

The main issue encountered during the project so far has been the complexity of the university’s systems. The system has been tricky to model and design with regards to the uniqueness of modules, courses, and years. Defining uniqueness and making the application functional with such complex relationships has been a challenge and has required some refactoring at points, but the project is currently in a stable state regarding this, and may not require any further changes here.

# Reviewing Accomplishments and Results

Outside of the project’s technical implementation, the project requires several written reports to be submitted for strict deadlines. So far, each report has been finished in a timely manner and to a high degree of quality (as indicated by the feedback). Half of these reports have been submitted thus far, and this report will mark the penultimate submission before the final project report is required. For the technical side of the project, each sprint goal that has been meet up to this point has been done to a high standard, accompanied by extensive unit testing to maintain the validity of the system as the project progresses. A look back at the issues in the repository that have been closed so far gives an example of the type of information that is tied to all the functionality completed in the project.

*Closed Github issues*

A screenshot of a cell phone

Description generated with very high confidence

*Information for an added feature, with development commit, Travis build link and screenshot evidence*

A screenshot of a cell phone

Description generated with very high confidence

# Analysing Issues and Determining Corrective Action

The main issue encountered so far, as previously mentioned, is the complexity of the relationships between entities in the university ecosystem. How a module is uniquely identified, how it relates to different courses (and multiple courses) that students are enrolled in, and other such questions have been frequently posed to the project supervisor during the development of the system as the implementation has forced such quirks to light. In a recent meeting, it appears that this complexity may have finally been resolved and the data models have been updated to reflect this newly gathered information.

The system is mostly able to be demoed in its present state which will help at the end of the project, but some features may be harder to demo without having the application fully deployed first, e.g. the reporting of attendances to different users via email. This may be solved by the unit tests and with Django’s ability to easily simulate email functionality by creating them as text files in a target directory. The command line functionality used to generate these emails also have a flag which can be used to specify that the functionality should be ran in “test mode”, which will output information to the command line window instead of actually trying to send emails.

I don’t believe any corrective action needs to be taken yet, as there has been no difficulties encountered so far in the technical implementation of the project. The project supervisor has also responded positively to each status report so far and seems happy with the progress of the project to date.

# Appendix 1

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence