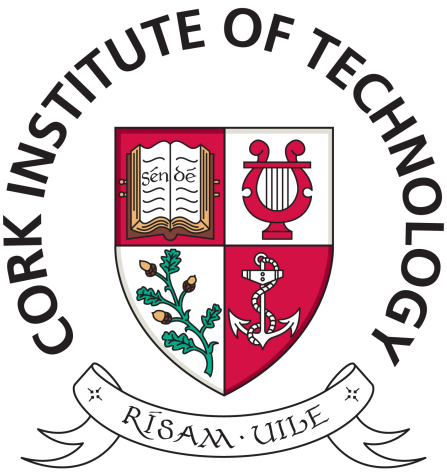


Nimbus Projects Portfolio Online Management, Analysis and Monitoring Tool

by

Aidan Dennehy

This thesis has been submitted in partial fulfillment for the degree of Bachelor of Science in WEB DEVELOPMENT

in the
Faculty of Engineering and Science
Department of Computer Science

January 2020

Declaration of Authorship

I, Aidan Dennehy , declare that this thesis titled, 'Nimbus Projects Portfolio Online Management, Analysis and Monitoring Tool' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for an undergraduate degree at Cork Institute of Technology.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Cork Institute of Technology or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

CORK INSTITUTE OF TECHNOLOGY

Abstract

Faculty of Engineering and Science
Department of Computer Science

Bachelor of Science

by Aidan Dennehy

The purpose of this project us to provide the Nimbus Research Centre an online management tool that maintains a list of their Projects and Prospects tracking costs against budgets. The system will include a Project archiving facility, will maintain a list of Clients and provide an interface that displays news articles associated with these Clients. A Dashboard feature will be incorporated that will display summary management information along with alerts when a project nears or breaches budget and time stipulations. A Reports function will be developed that will include Top 20, Points per project Report, Summary of materials budget left and Summary of travel budget left. End user access management is also required.

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project supervisor (term one and two)...

Contents

List of Figures

List of Tables

Abbreviations

LAH **L**ist **A**bbreviations **H**ere

For/Dedicated to/To my...

Chapter 1

Introduction

This chapter should comprise around 1000 words and introduces your project. Here you are setting the scene, remember the reader may know nothing about your project at this stage (other than the abstract). N.B. The sections outlined in this document are suggested, some projects will have a greater or lesser emphasis on different sections or may change titles and some will have to add other sections to provide context or detail.

1.1 Motivation

Why is it important to do a project on this topic? This should cover your key motivation for this. For example an excellent student from 2016 noticed a large number of homeless sleeping rough in Cork and was motivated to develop a system that load balanced the homeless shelters to try to accommodate the maximum number of homeless. This section can include the personal pronoun but the rest of the report should be third person passive, this is the case with most technical reports! For example here it is fine to say "... I decided to develop and app to help ...".

1.2 Contribution

This document contributes to the Projects Cost Control of Nimbus in CIT, however the principles behind this can easily be implemented in any Research Centre.

There are several modules of the Web Development Computer Science degree in CIT and during my internship at the Nimbus Research Centre that helped me with the planning of this project.

From the outset, the project has involved a lot of planning. The author sat down with Nimbus and developed a list of requirements that the project entailed and developed use cases and user stories from this – these were key items within the Requirements Engineering SOFT7007 module from the first semester of Year Two.

There were learned elements of the Object Orientated Analysis and Design (SOFT7005) course that have formed part of this project. This approach was used to design the Project Budget Data Import Module and the Web Scraping modules.

The Web Scraping module is written in Python and controlled via chron, these are elements of the Systems Scripting (SOFT7044) and the Linux Administration (COMP7036) modules.

PHP was studied and practiced in the Server-side Web Frameworks module (COMP8001) and the author expanded on this while interning at Nimbus during the Year Three Work Placement Program working on various projects under their supervisionf.

There are several new techniques that will be used in this project that go beyond the foundations laid down in CIT. The author plans on using the Web Scraping Python tool BeautifulSoup to scrape and parse news data off the popular technology news sites that come recommended by Nimbus to keep abreast of news articles relating to companies in their Contacts database. For the conversion of excel spreadsheet files, the author's research shows that using the xlsxworker library to parse and extract the Budget data from the CIT excel spreadsheets into the MySQL Database is a viable strategy.

Data Visualisation will be studied in the second semester of Year Four within the Interactive Data Visualisation (COMP8054) module and it is hoped that this module will provide a grounding for the Dashboard module within the project.

Agile and in particular Scrum will be used to manage this project, particularly within the implementation stage. This strategy emanates from the Agile Processes (COMP7039) module. Also, from this module, GitHub will be used for source control/version management and Jenkins for CI/CD.

1.3 Structure of This Document

This section is quite formulaic. Briefly describe the structure of this document, enumerating what does each chapter and section stands for. For instance in this work in Chapter 2 the guidance in structuring the literature review is given. Chapter 3 describes the main requirements for the problem definition and so on ...

Chapter 2

Background

The key question to answer in this chapter is: "What has been done/is being done".

This chapter comprises around 4000 words and should put your project into context within Computer Science. Your focus here should be on the final section "Current State of the Art". This should be at least 2500 of the 4000 words of this section.

System Architecture

System architects are the designated experts responsible for a system's architecture as well as the technical standards of a product. This includes technologies, platforms and infrastructure. They set the vision and their analysis is key to the product's successful definition, design, delivery and life-time support. They therefore need to understand not only what the business need, but also what is logical, scalable, cost effective and in-line with the overall technology goals of the organisation. [1]

One of the vital skills of an architect is to be able to view the architecture from many different standpoints: each one of them individually might not be fully relevant but combining them together gives a helicopter view of the product. These standpoints comprise of principles, standards, patterns and anti-patterns, rules of thumb and empirical practices which are essential for decision making towards a particular direction and also evaluating the project's success. [2]

SOLID Principles The SOLID principles do not only apply on software development but also when architecting a system.

Single Responsibility Principle

Each system capability (e.g. service/module/api) should have only one responsibility and as such one reason to change. Keeping the responsibilities as narrow as possible means that the users know of the intended purpose, which leads to less errors.

Open-Closed Principle

This principle proposes that it is preferable to extend a system behaviour, without modifying it. Although it is often not a good idea to try to anticipate changes in requirements ahead of time (as it can lead to over-complex designs), being able to adapt new functionality with minimum changes to existing components is key to the application's longevity. [3]

Liskov Substitution Principle

In Software Development, this means that derived classes must be substitutable for their base classes, but this principle's resemblance with Bertrand Meyer's Design by Contract is how it can be applied to Distributed Architecture: two services communicate effectively and repeatedly when there is a common 'contract' between them, which defines the inputs/outputs, their structure and their constraints. Therefore: given two distributed components with the same contract, one should be replaceable with other component with the same contract without altering the correctness of the system. [3]

Interface Segregation Principle

Interfaces/contracts must be as fine grained as possible and client specific, so calling clients do not depend on functionality they don't use. This goes hand in hand with the Single Responsibility principle: by breaking down interfaces, we favour Composition by separating by roles/responsibilities, and Decoupling by not coupling derivative modules with unneeded responsibilities. [4]

Dependency Inversion Principle High level modules should not depend on low level ones; they should both depend on abstractions. Likewise, abstractions should not depend on details, but details should depend on abstractions. As such this principle introduces an interface abstraction between higher-level and lower-level software components or layers to remove the dependencies between them.

The 'Least' Principles

The principle of Least Astonishment

The principle of least astonishment (or Least Surprise) [2] suggests that a solution or approach would not surprise a reasonably knowledgeable person in the subject area when encountered for the first time (the audience may vary e.g. end-user, programmer, tester etc). In more practical terms, the principle aims to leverage the pre-existing knowledge of users to minimise their learning curve when using a module, so anything with high unpredictability factor is a good candidate for re-design. It applies to every single aspect

of the architecture: from naming services, to the visualisation of user interfaces, to the design of the domain model.

The principle of Least Effort

This principle (also called Zipf's Law) [5] stems from a basic human behaviour: Everyone tends to follow the path that is as close to effortless as possible. So for example if our design follows a particular pattern, the next developer will follow the same pattern again and again unless there is a significantly easier way to perform the task, in which case they will change! Or, taking this further, once they find acceptable results for a task, there is no immediate need to improve the current solution.

Least effort is a variant of least work

As such it is imperative to aim for a strong start by putting the right architecture in place: it sets high expectations and ensures that the quality is not compromised in the project's lifecycle and it will be adhered to in case of future changes. Once the right design is in place, and architectural framework can be established which will be the bases for future projects of a similar architectural style.

The principles of Economics

The cost of making the most of an opportunity and the cost of delaying making decisions.

The principle of Opportunity Cost Every time we make a choice, there is a certain value we place on that choice. Value has two parts: benefits and costs. The opportunity cost of a choice is what we give up to get it. To make a good economic decision, we want to choose the option with the greatest benefit to us but the lowest cost. [6]

Architecture is weighing choices against each other and trying to make an informed decision on which one will add the most value for the project. For instance, a very common dichotomy is whether to create a tactical solution with quick time to market or a more strategic one which will be more expensive now with the view to leverage it in future projects and hence minimise the cost later down the line.

The principle of Last Responsible Moment This principle (aka Cost of Delay) originates from Lean Software Development [7] and emphasises holding on taking important actions and crucial decisions for as long as possible. This is done so as to not eliminate important alternatives until the last possible moment i.e. wait to narrow the options down until you are better informed. A strategy of not making a premature decision but instead delaying commitment and keeping important and irreversible decisions open until the cost of not making a decision becomes greater than the cost of making a decision.

One way to mitigate the risk of deciding too late is to build Proof of Concepts (POCs) to prototype the alternative options and demonstrate to the stakeholders what they are asking for.

Data Visualisation

Data visualization usually represents graphs and charts that are visual representations of data. These graphical displays provide a powerful way of summarizing and presenting data in a way that most people find easier to comprehend. Charts and graphs enable us to see the main features or characteristics of the data. The graphs not only enable us to present the numerical findings of a study, but also provide the shape and pattern of the data which are critical in data analysis and decision making. It is said that a picture is worth a thousand words; this is particularly true when a large set of data is effectively presented using charts and graphs that quickly reveal important features. Visual displays of data are easily recognizable and are found ubiquitously in business periodicals, financial magazines, on the Internet, and televisions. [8]

Suraj Thatte writes in his article that data visualization has a strong design element to it. Given the differences in domains, applications and audience it's hard to put a structure around the best way to visualize your data. [9]

Suraj shares his tips on representing data visually as follows:

Choose the right visual - Always remember that "form follows function" – purpose of a visual should be the starting point of its design

The question that should be asked is what are you visualising - The visualisation for this project is Project Budget breakdown displaying the Total Budget, Total Remaining and totals for Income, Staff and Other. A good fit for this project is a Pie Chart which will clearly highlight the Total Remaining against the all other fields.

Trivial are many but vital are few (data points) - Only show the data that the audience is interested in.

Visuals should reflect reality and not distort them. Formatting of the chart plays an important role as it sets up a frame of reference for the audience.

Use of colour should be made to add more information or to highlight key data points in a visual. In all other cases it is redundant and distracting. Lisa Charlotte Rost in her article on considering colours states that using as few contrasting colours is good, avoid gradients and consider visually impaired members of the audience. Colours should be made to add more information to a chart, in all other cases it is redundant and distracting. [10]

A visual chart should also avoid overdoing the aesthetic elements which the audience may find distracting. One of the seven wastes in the Lean Philosophy is 'Over-processing'. [7]

Web Scraping

As the name implies, it's a method of 'scraping', harvesting or extracting data from webpages. It is one of the oldest techniques for extracting web content. [11] Web scraping software may access the world wide web using HTTP or through a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes using a bot or a web crawler. It is a form of copying in which specific data is gathered and copied from the web, typically into a local database or spreadsheet for later retrieval or analysis.

Web scraping a web page involves fetching it and extracting from it. Fetching is the downloading of a page which a browser does when a user views the page, therefore web crawling is a main component of web scraping to fetch pages for later processing. Once fetched, then extraction can take place. The content of a page may be parsed, searched, reformatted or have its data copied into a database or spreadsheet. Web scrapers typically take something out of a page to make use of it for another purpose somewhere else. Within this project the author will design a program to search the relevant news sites looking for articles of companies within the Nimbus contacts database so that they may be kept up to date of news articles relating to their contacts.

Web scraping is used for content scraping and as a component for applications used for web indexing, data mining, online price change monitoring and comparison, product review, to watch the competition, gather real estate listings, weather data monitoring, web site change detection, research and many other uses.

Web pages are built using text based markup languages HTML and XHTML and frequently contain a wealth of useful data in text form. However most web pages are designed for human end users and not for ease of automated use, because of this, toolkits that scrape content were created. A web scraper is an API to extract data from a web site. Newer forms of web scraping involve listening to data feeds from web servers e.g. it is commonly used as a transport storage mechanism between the client and a web server. There are methods that some web sites use to prevent web scraping such as detecting and preventing bots from crawling and viewing their pages. In response there are web scraping systems that rely on using techniques and DOM parsing, computer vision and natural language processing to simulate human browsing to enable gathering web page content for offline parsing.

The author has focused in on the Python library called BeautifulSoup. Beautiful Soup is a Python library for parsing structured data. It allows a developer to interact with HTML in a similar way to how someone would interact with a web page using developer tools within a browser. Beautiful Soup exposes a couple of intuitive functions you can use to explore the HTML you received. It is possible to parse through web page content by any element with a 'class' or 'id' tag attached to it and extract text content from within.

2.1 Thematic Area within Computer Science

Position your topic within Computer Science. This activity will aid you in your literature review also. We zoom out to see three levels: [12]

1. What is the core topic your project is about? e.g., Mobile app for online voting.
2. What core area(s) does the project fall under? e.g., Mobile applications, Social Networking, Service Providers.
3. What main area(s) of Computer Science does the project fall under? e.g. Software Development, Cloud Computing.

The ACM Computing Classification System (http://www.acm.org/about/class) will aid you in this, use the 2012 categories. Make sure to use figures and illustrations were appropriate. LaTeX will take care of the formatting of these. Do not try to get fancy here, you should concentrate on the content and not the formatting, this is why we are specifying LaTeX.

You can specify the width and label for a figure which allows you to reference the figure and you can attribute a source in the figure caption as is done for figure 2.1. Make sure you reference all external figures (i.e. figures you did not create yourself). Also use references for all figures e.g. use "... in figure 2.1 ..." NOT "... in the figure above ...".

2.2 A Review of -INSERT THEMATIC AREA-

The focus of this section is at the heart of the project research phase. You must identify the main sources of information you should be aware of within your chosen area and pay regular attention to so as to strengthen your knowledge in the core topic you are working at. So here you should develop an knowledge of not only your core topic but also about

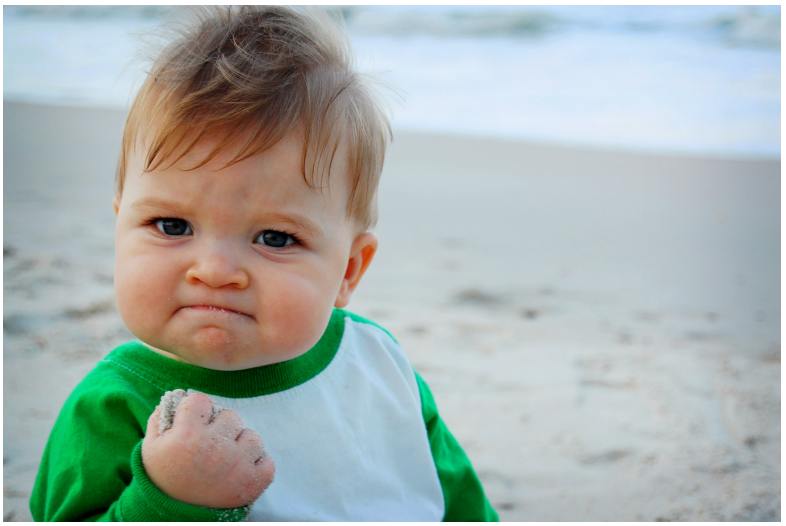


FIGURE 2.1: A picture of the success kid![**?**]

the area of computer science the topic falls under. More specifically you should research the following:

- The top 5 International Conferences and Journals most related to your topic. This is crucial, as it represents the main source for keeping you aware of what the state-of-the-art in your topic is.
  - In particular it will make you aware of what other projects related to yours have been already done (so that you can compare/position your project w.r.t. these).
  - What new techniques are being developed, so that you can apply them in your work. e.g. new frameworks for data visualization
- The top 3 most recent books/texts related to your topic. There are many free resources from which you may download a relevant text on the topic of your project. Try to either download or borrow 3 recent (no older than 10 years) texts relating to the topic your project is on which you will use throughout the project as reference material and to aid in tackling a number of the technical problems you may encounter. Any PhD/MSc thesis that have published in the last 5 years relating to the topic are also invaluable resources as they will contain a state of the art and references in your project topic. Approach these only after reading/viewing the wikis/Youtube videos you find as a certain level of knowledge will be assumed about the topic.
- The top 5 companies/organizations potentially interested in the product you are developing. Finally, this is also crucial, as it forces you extend to purely programmer view of the project to a wider view considering the market, potential

stakeholders and niches where your product can become useful. Moreover, Computer Science is a huge topic with loads of different works and roles. If you pick a project in the area you feel passionate about, and you identify what the market in this area is about, then you can drive your future professional career (from the very beginning) towards the path that makes you happier. I know that this does sound as a very technical reason, but I suppose we all agree is probably the most important of all reasons for choosing a particular project focus.

- The top 5 wiki/forums/blogs/Youtube channels most related to your topic. This is crucial to you as well, as it represents a more accessible, personal and less informal way of communication with people working/interested on the same topic as you are. This communication is extremely helpful for improving your skills, solving potential doubts and increase the interest/relevance of the topic/area itself.

You should begin your journey of discovery in reverse order to the listing above (which is given in order of academic importance/significance). So when you are researching your topic first look up some TedX talks or youtube tutorials, then research what companies are doing in the area, then get a handful of very good texts on the core topics of your area (anything older than 5 years usually is not helpful here) and finally start reading conference or journal papers (again newer is better here). In particular during this section you may need to use tables to list resources. These are also automatically formatted in latex thus allowing you to concentrate on content. for example table 2.1.

Parameter	PET
Youngs Modulus	2800-3100MPa
Tensile Strength	55-75MPa
Glass Temperature	75°C
Density	1400kg/m^3
Thermal Conductivity	0.15-0.24Wm^{-1}K^{-1}
Linear Expansion Coefficient	$7 \times 10^{-}5$
Relative Dielectric Constant @ 1MHz	3
Dielectric Breakdown Strength	17kVmm^{-1}

TABLE 2.1: PET Physical Properties

What has been done before in your community w.r.t. your topic? Once you have gotten an understanding of the topic and technologies and have identified the top 5 formal conferences/journals, wiki/forums/blogs/Youtube channels and companies/organizations the next step is to research in depth on them! And here in depth means in depth. Make sure you cite[**?**] a number of papers [**?**], luckily Latex will take care of the ordering of the citations [**?**] for you.

The aim here is that you find the trends in your topic (3), and more in general in the area in which your topic resides (2) your project falls under and from these trends

you develop your initial project question further and begin to get insights into how others have solved/approached similar problems. Think of this section as colouring in your initial idea. Before you approach this section you should read at least 4/5 good literature reviews (a selection of last years projects will be posted on blackboard to aid you but you should find other sources also).

In particular in this section, you must find and analyze at least 5 (ideally around 10) works belonging to, or at least related to, your work. You must describe these works and position your project w.r.t. them (i.e., clearly identify the similarities and differences between your project and each of these works). Also remember if you find that you are detailing topics that you have not introduced already here you need to add something to the earlier Scope section.

Chapter 3

Problem - rename to project title

The key question to be addressed in this chapter is: "What do I want to achieve".

This chapter should comprise around 1500 words and describe the problem you are trying to solve. Try to be as specific here as you can, this will help you to anticipate possible risks such as lack of support from APIs.

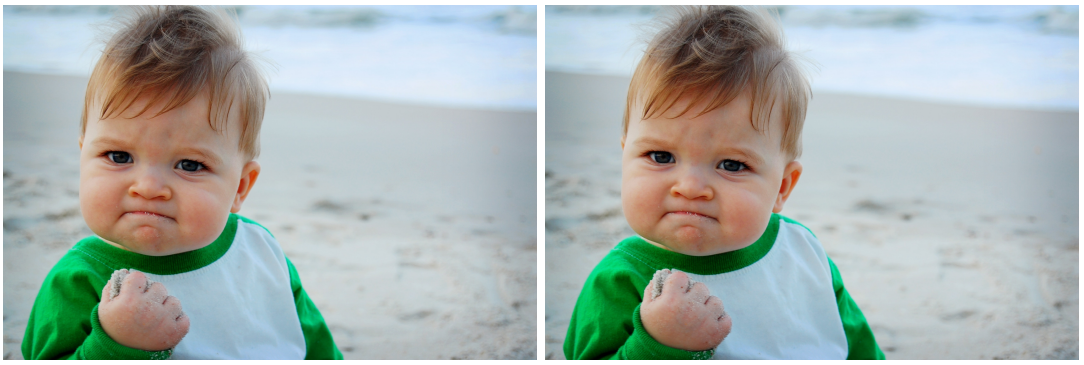
3.1 Problem Definition

Describe the problem you are trying to solve in this project. There will sometimes be a need at some point during the report to display an equation that may be core to your project. For example if the project is on gait detection what equation are you using to determine gait? If the project is on localization what is the method/formula? The formatting of these is reliably done in Latex also as we can see in equation 3.1.

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{c_i}}) - \frac{\partial L}{\partial c_i} + \frac{\partial P}{\partial \dot{c_i}} = F_i, \tag{3.1}$$

3.2 Objectives

Enumerate the objectives you want to achieve in your project. Again as this is an early stage these will tend to change but there should be a rational explanation for this change. Always document your work, keep a lab book during the term that you only use for FYP!



(a) Figure A (b) Figure B

FIGURE 3.1: Two Success kids

3.3 Functional Requirements

Enumerate the functional requirements you want your project to have.

Please, do not include the use cases here. If you want to create a one-to-one mapping between functional requirements and use cases (which does not necessarily need to be the case, indeed most likely this will not be the case) do it elsewhere. Here should purely describe what do you want to do. In no case should you use this section to provide a description of how to implement them, that is for later. For people doing projects that are not heavy implementation projects (e.g. deploying an architecture or testing a novel tool in specific conditions) this structure can still be used as it will force you to think about what you plan to achieve and what possible metrics you may need to measure success.

Let me explain this with more detail. A common mistake is that people confuse the problem description with the solution approach. This is a common mistake by confusing the *what* with the *how*. Here we are purely focused on the what: What is this project about? What are the objectives? What are the functional and non-functional requirements?

How are we going to do all these things? Well, this is a question for next chapter. Provided a problem, an objective or a functional requirement, obviously there will usually be many ways of doing it, thus there will be many *hows*, but the definition, the *what* we want to achieve will be unique.

One other display structure you may wish to use at some stage during the report is a figure array. This can also be easily done with Latex and is shown in figure 3.1

3.4 Non-Functional Requirements

Enumerate the non-functional requirements you want to achieve in your project (i.e. broadly speaking how your system will operate).

Chapter 4

Implementation Approach

The key question to be addressed in this chapter is: "How do I plan to achieve what I have outlined in the previous chapter".

This chapter should comprise around 5000 words and specify your planned implementation approach. Again all sections below are suggestions and will vary significantly from project to project, the key element to be addressed is the core question of the chapter.

4.1 Architecture

Describe the architecture of the solution that you have in mind, including:

- Technologies involved (e.g., frameworks, programming language).
- The hardware needed to develop the project (and to support at deployment stage)

Provide a high level view of the system you have in mind, including any package of classes, what is it responsible for and what other packages it communicates to. Provide a high level view of the database (or structure) needed to support the project, including what each table/document is responsible for and the hierarchy among them. You need to be as specific here as you can, why? Because this will aid you in identifying parts of the project you are vague on, this may be fine for some components but cause problems in term 2 for others. If you have hardware element in your project this is also where you provide a high level view of how these elements integrate into the project. So for a project that is cyber-physical you will have both a hardware and software architectural diagram. N.B. This is NOT a full system design but a high level overview of what you can credibly develop. This architecture should be informed by prototyping activity.

Some of the implementation focused projects may describe how do you envision tackling the functional requirements of your project via a set of use-cases. DFDs are also helpful here to understand elements of your project that may cause problems. You should describe the role of the different parts of the architecture of the solution, and the interaction among them.

4.2 Risk Assessment

Identify any potential risk precluding you from successfully complete your project. This section is really important and often neglected by students resulting in fatal risks occurring in some projects. Make sure to give this section the time it requires. Classify the risk according to their importance, possibility of arising and enumerate the decisions you can make to anticipate them or mitigate them (in case they finally arise). Table 4.1 may help with this classification. This section should include your mitigation approach for any critical risks.

TABLE 4.1: Initial risk matrix

Frequency/ Consequence	**1-Rare**	**2-Remote**	**3-Occasional**	**4-Probable**	**5-Frequent**
4-Fatal					
3-Critical					
2-Major					
1-Minor					

4.3 Methodology

Describe your personal approach on how to tackle the different parts of this project, including:

- How to tackle the needed research to fulfill the background chapter.
- How to set up your Computer Science skills to the project needs (e.g., describe your plan to learn any new technology involved on the project that you are not familiar with).
- What core project managing approach will you follow (e.g., Waterfall, Scrum, etc).

4.4 Implementation Plan Schedule

Come up with a schedule for the remaining time (including second semester), so as to describe how do you envision to achieve the implementation of your project by the end of semester 2. This plan SHOULD be ambitious but MUST be realistic and SHOULD be informed by early prototyping and MUST be discussed with your term 1 supervisor.

4.5 Evaluation

Come up with an evaluation plan that allows you to measure how much have you actually achieved the goals of your project. This again is a section that is often neglected where students loose marks. How do you plan to measure the output of your project? A binary it works/does not work is insufficient. You need to be able to quantify the success against both the functional requirements and the initial idea. These are not the same as you may meet all function requirements outlined but not solve the overall problem because you have failed to revisit these and update them with new information which you learn as you are developing the project.

4.6 Prototype

Although you do not have a fully functional project yet, you should show wireframes, snapshots or representation on how do you envision your project to look once the implementation phase has been completed. The nature of this section will vary significantly from project to project and can include anything from code snippets to snapshots of service deployments. Any prototyping you have done during the term should be summarized here that has not been captured in earlier sections. For example if you are planning to host your project using AWS in an EC2 instance you should have at least created a "hello world" setup to determine the basics, this probably should have been discussed in section 4.1.

Chapter 5

Conclusions and Future Work

This chapter should comprise 2-3 pages and enumerate conclusions of this phase of work. In your final report Discussions and Conclusions will form separate chapters and be significantly longer and more detailed.

5.1 Discussion

A reflective discussion of some of the problems you encountered during this phase of the project and how that may influence how you proceed with the next phase.

5.2 Conclusion

Enumerate the main conclusions you have got in terms of background, problem description and the solution approach you have come up with.

5.3 Future Work

Enumerate all the things you would have wanted to do should you have more time to work on this report.

Additional resources on the use of latex is below.

Tutorials:

- `https://www.latex-tutorial.com/tutorials/beginners/how-to-use-latex`

- https://en.wikibooks.org/wiki/LaTeX
- https://www.sharelatex.com/learn/Main_Page
- http://www.math.harvard.edu/texman
- https://web.stevens.edu/hfslwiki/images/a/a0/ShareLatex_Tutorial.pdf

Presentations:

- http://www.iu.hio.no/~frodes/rm/ppt/latex.ppt
- https://classes.soe.ucsc.edu/ams200/Fall09/Latex_intro.ppt
- http://www.menet.umn.edu/~blake/latexcourse/courseslides.ppt

Bibliography

[1] P. Clements, R. Kazman, M. Klein, and B. Boehm, "Evaluating a Software Architecture," pp. 1–17, 2013.

[2] S. Koen, "5 Key Principles of Software Architecture - Towards Data Science," 2019. [Online]. Available: https://towardsdatascience.com/5-key-principles-of-software-architecture-e5379cb10fd5

[3] E. Baniassad, "Making the Liskov substitution principle happy and sad," *Proceedings - International Conference on Software Engineering*, pp. 17–20, 2018.

[4] M. Noback and M. Noback, "The Interface Segregation Principle," *Principles of Package Design*, no. c, pp. 55–66, 2018.

[5] J. Kanwal, K. Smith, J. Culbertson, and S. Kirby, "Zipf's Law of Abbreviation and the Principle of Least Effort: Language users optimise a miniature lexicon for efficient communication," *Cognition*, vol. 165, pp. 45–52, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.cognition.2017.05.001

[6] E. S. Raymond, "Basics of the Unix Philosophy," 2003. [Online]. Available: http://www.catb.org/{~}esr/writings/taoup/html/ch01s06.html{#}id2878339

[7] U. Dombrowski and T. Mielke, "Lean leadership -15 rules for a sustainable lean implementation," in *Procedia CIRP*, vol. 17. Elsevier B.V., 2014, pp. 565–570.

[8] A. Sahay, "Data Visualization Volume II; Overview and Data Visualization," vol. II, pp. 1–10. [Online]. Available: https://ebookcentral.proquest.com/lib/cit-ebooks/detail.action?docID=4819435

[9] S. Thatte, "Tips for Effective Data Visualization," *towardsdatascience.com*, 2019. [Online]. Available: https://towardsdatascience.com/tips-for-effective-data-visualization-d4b2af91db37

[10] L. C. Rost, "What to consider when choosing colors for data visualization," pp. 1–8, 2018. [Online]. Available: https://blog.datawrapper.de/colors/

[11] D. Glez-Peña, A. Lourenço, H. López-Fernández, M. Reboiro-Jato, and F. Fdez-Riverola, "Web scraping technologies in an API world," *Briefings in Bioinformatics*, vol. 15, no. 5, pp. 788–797, 2013.

[12] B. R. Barricelli, F. Cassano, D. Fogli, and A. Piccinno, "End-user development, end-user programming and end-user software engineering: A systematic mapping study," *Journal of Systems and Software*, vol. 149, pp. 101–137, 2019. [Online]. Available: https://doi.org/10.1016/j.jss.2018.11.041

Appendix A

Code Snippets

Put appendix material in this section e.g. code snippets

USE THE APPENDICES

Appendix B

Wireframe Models