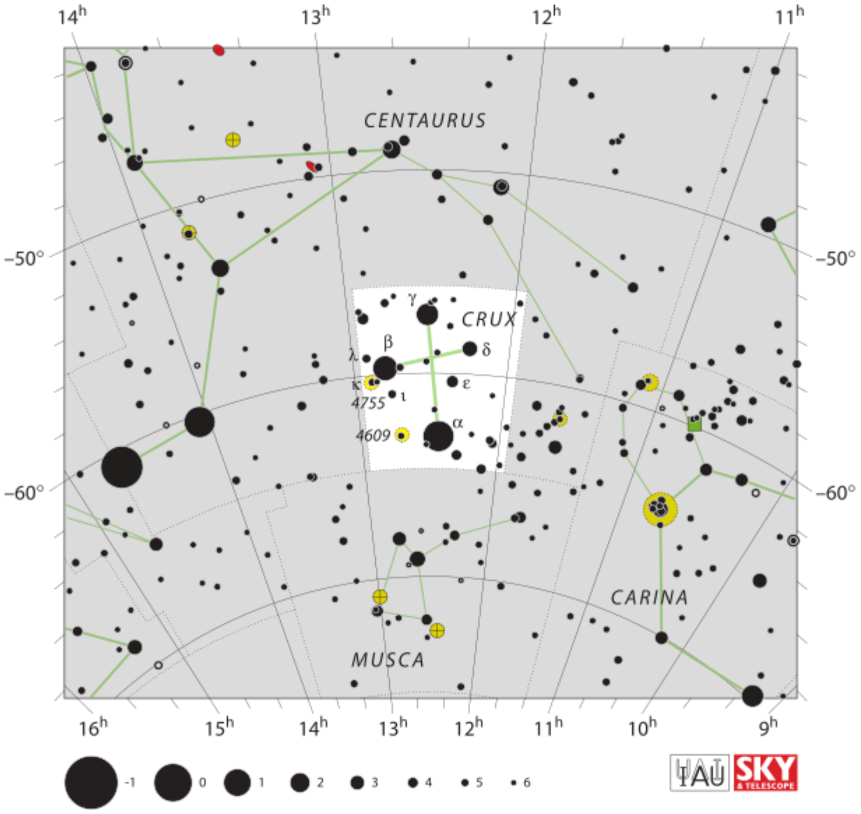


PROJECT PROPOSAL & MANAGEMENT REPORT FRAMEWORK

5WCM0018 – Software Development Exercise

CRUX

Rick Fawley (17027782)
Matt Holburn (14138290)
Peter Hanna (14032305)

Version

1st April 2018 v1.0 (Final)

1.0 Managing Software Development Projects

This section is to analyse and decide on the best development method for the task we have been assigned. We will be comparing popular and applicable strategies, and then deciding on one to use throughout the development process.

1.1 A Comparison of Common Development Strategies

There are many ways software can be developed, especially within a group, these are known as Development Strategies. There are some common strategies out there for various applications. These include Waterfall, Prototype, Dynamic, Agile, Spiral and Extreme Programming. Each has various advantages and disadvantages, but for the case of our project, and especially being a small team, we have decided to focus on Waterfall, Agile, and Spiral modelling.

1.1.1 Waterfall Development Model

The waterfall model is based on the idea of having set phases of development. These are usually: Requirements, System Design, Implementation, Integration and Testing, Deployment of System, and Maintenance. This model can be very useful with small scale projects with a small team as each phase can ensure that no overlapping of processes takes place, which helps to ensure each phase is on budget and on time. This gives better control over deadlines and helps to predict if more assistance is required in a specific area. It is very easy to understand, and its simplicity enables clients and developers to understand the specific goals required for each phase.

Despite this simplicity, the waterfall model does have some significant disadvantages, including having limited ability to go back and change something after going through the testing stage, as no working software is produced until late in the development cycle. This also means if something does not work, or the requirements change during development, it could result in starting practically from scratch again, so the requirements need to be clear and simple. This makes it best for small, short projects, with clearly defined requirements.

1.1.2 Agile Development Model

The agile model is the most popular model since the waterfall model as it allows for constant analysis, improvement and user input, whilst also allowing to be structured and easy to manage.

Agile development consists of smaller incremental rapid cycles of development which gives incremental releases for testing to ensure quality, and requirements are being kept to.

For many team sizes, agile development allows for constant interaction between developers, customers, and testers to constantly improve the quality of the software, and frequently providing progress that is visible to clients. Late changes can even be incorporated fairly easily, which is not easily done in many other models.

Sadly, as with all models, some problems can arise from this flexibility, for instance, if many changes are made during each cycle, the entire project can be thrown off track with newer additions and deeper changes to the software. This could cause extreme delays with the initial final product, as a lot more has been added. Also, newly trained programmers may find the development process difficult, as some difficult decisions may be required fairly quickly during the process.

1.1.3 Spiral Development Model

Very similar to the agile method, this spiral method continually moves through four stages, ensuring the project is successful, and practically risk free.

As with agile development, prototypes and iterations of the software are created during each spiral, but this time involves analysis of risks during each cycle. This means higher risk projects have more time dedicated to ensure problems are smoothly dealt with before they can cause larger issues. This development process incorporates four main sections, Evaluation, Planning, Risk Analysis, and Engineering. Each using the previous sections results to improve the final product.

Although the reduced chance of risk sounds appealing, this method can be costly, in both money and time, and requirements for projects such as this can be complex.

We have decided to go with an Agile Development Process as this give us more flexibility and manageability throughout the project.

1.2 An Overview of Scrum's project management framework supported with an Agile development strategy

Scrum is designed to streamline an Agile development method. At its core, whilst keeping the incremental and iterative practices in agile development but breaks this workload down into manageable chunks and tasks. Generally, scrum project management consists of short "sprints" of workload to complete these small iterations of the final product. Within a business, meetings would be held either daily, or weekly, to review and analyse the previous sprint, and plan this sprints tasks.

The tasks are then completed by each team member, within their allotted timescale, and if more resources are required elsewhere within a different sprint, then they can be allocated accordingly.
The Scrum framework allocates the following to help the progression of the product:

Product Backlog – The complete requirements of the product usually given by the end user, or the customer. The most important of these will be taken to form a release backlog for the next release of the product for testing.

Release Backlog – The prioritised list of requirements for this release. These will also be allocated time to complete, so larger sections may be split up further to complete quicker. For each full release, multiple sprints would be carried out to keep the progression flowing smoothly and easily manageable.

Sprint Backlog – The list of items to work in this short development cycle. If items on this list do not get completed in time, then individual items can be managed to prevent delay of the full and final product before the delay becomes a larger issue.

Burndown Charts – This is the schedule and overview of the whole project and used to see what has been done, and what is left to do. This will also show if the project is delayed or ahead of schedule by showing each sprint's progress in relation to its initial position at the time.

As a team of three, we have some positives and negatives by using a scrum method, the most obvious being that we may struggle with a workload that is initially designed for 4 people. On the other hand, this will allow us to delve into more sections together and mean we will have to work well as a team to accomplish the goals of the project. Despite this, even if one member of the team does not pull their weight, then it could mean the final product could suffer either by being incomplete, or just not being as polished as we would like.

Scheduling could also be an issue, as all parties involved would be working at different times, whether it be in different time zones, or purely evenings or weekends. This does not become so much of an issue if we all have individual tasks within the workload, but as we will be required to expand upon other users work, this could be somewhat troublesome. We have created a Teamweek.com project to manage our work, which is documented in section 5.4.

2.0 Project Outline

This section is designed to explain the basis of the project given to us by the client, FDM. Various items will be discussed, including a history of existing technology, requirements, and the aims of this project.

2.1 Problem Statement & Project Background

We have been requested to produce a project proposal for a piece of software for a company called FDM. They require a solution to allow their users to easily, quickly and efficiently report their working times, and have them authorised. They have requested some main features, and when speaking to the end users, they have also requested their own requirements.

FDM are an IT Consultancy Provider. They employ a diverse group of people, from many walks of life. This means any system they request will need to be easy to use, clear to understand, and be able to be flexible enough to adapt to each user's information.

Currently each user fills in a time sheet and manually emails it through to a line manager. One common complaint about this system is that no pre-set times can be saved for each user, forcing them to input their times every single week one by one. After sending this form off, it is then authorised by a line manager. This can take time as the manager manually has to go through and check each employee individually. This authorisation is required by FDM in order to verify employees' hours to ensure no hours are being claimed maliciously. Once confirmed, it is then forwarded to payroll for processing. This whole process can be very arduous, repetitive and time consuming for everybody involved. Many end users requested a multi-platform easy solution.

2.2 Client Requirements

The client has requested many features, all of which are important to the overall quality of the end product. We have listed each of the specified requirements in order of importance, of course security and privacy are high on the list, followed by features the end product "should have". There is nothing that the end product currently won't have, whether that be via time constraints or system restrictions. This could change throughout the project's development, but that will be where analysis mid-way though the project will assist with.

Must Have Secure access only, integration with payroll, automatic submissions to next stage.

Should Have Visibility only to authorised users, notifications to all users, ability to set and change password, notification to line manager for approval, indicators for full and half days off, bank holidays set to 0 hours by default, Multiple time zone support, bank holiday support for different regions.

Could Have Pre-population of regular hours, access via multiple devices.

Won't Have New requirements, without removing existing requirements

2.3 Business Case - Project Goals & Benefits

This project will benefit the client in many areas of business. Most importantly with savings in time and money. Without realising it, many of the employees have discussed basic improvements to the current system to benefit the client as well as themselves. As everyone has to perform these tasks on a weekly basis, the important factor to remember is that the process should be streamlined and simple.

Time is a major benefit to the client. Nearly every user interviewed mentioned something about the time they had to allocate to the filling out of timesheets. Some people took even longer than expected as the current system did not perform well when actually trying to complete a timesheet.

One person interviewed said that he had to take time out of his working day on a Friday to fill in the form. That could have been 2-3 minutes more work instead of timesheet filling out. Of course, this time cannot be taken away completely, but ensuring access to the timesheet system from a mobile device or from home means that the users aren't finishing working early to fill it out.

With regards to a financial point of view, being able to allow users to fill out their timesheets on time, will mean the chances of the user forgetting the exact hours they have worked, and just estimating are greatly reduced. Although some of the human error factors still stand, they would not be able to blame lateness of form filling on the line manager or payroll department.

This may also help out payroll, as all timesheets should be in on time, and payroll are notified once a sheet has been authorised. Financially, all overtime can be worked out by the system, instead of being totalled up by the user or the line manager, this saves them not only time, but also reduces the user error that can happen, especially if the line manager or user is feeling like the task is repetitive or mundane.

Without this new system, the client may find that with more time, more human errors would occur, and if not so already, one day somebody will not get paid on time, and cause larger issues. This could be the fault of many people in the chain, but without the ability to notify all involved parties that the timesheet is on the next stage, a lot of time, energy, and money is wasted double checking every step.

2.4 General Aims & Objectives

As we have been asked to produce a prototype for this project, we have set ourselves the following aims and objectives:

- To complete the project by the agreed completion date
- All deadlines must be met, and any delays, agreed upon up to 48 hours of deadline. This is especially relevant given Crux is a reduced size team.
- To meet all functional and non-functional requirements, though some may be simulated for prototype.
- For the Client to review and decide that the project is worth continuing. They may do this via financial gain, or time gained back via ease of use.

3.0 Project Management and Development Strategy

Project Management and Development Strategies are powerful tools to be used by a team, as it is an effective way to increase your chance of a satisfied client/user. By using guidelines, techniques, and strategy to achieve completion of the project.

3.1 Preferred Approach to Project Management and Development

We have chosen to use the Agile method of Development. Agile method of development is an approach that allows the client/user to examine and direct the project until the client/user is satisfied. There are 12 principles that give a better understanding of the method itself.

Agile methodology is based on 12 principles:

1. Customer satisfaction is priority through all Iterations
2. Welcoming change in requirements at anytime of the process
3. Sprints are kept to a maximum or 4 weeks
4. Daily collaboration between all parties.
5. Supportive, impowering environment.
6. Face to face meetings to transfer information.
7. Progress is measured with functional software.
8. Maintaining constant pace.
9. Continuous attention to excellence, quality and design.
10. Simplicity to fit the fast-paced development.
11. Self-organizing teams.
12. Teams open to adaptation and inspection, to aid in becoming more effective.

In maintaining the project time iterations are used, these iterations are called sprint. After each sprint the client/user is shown the product and may change some requirements in the project. Each sprint can have a duration from 1 week up to 4 weeks. Difference between agile and other methods is that in agile the client/user is very hands on, with every sprint the client/user previews the project giving new requirements each time this process is repeated multiple times until the user is satisfied with the outcome of the project. There are 6 phases in the agile method.

Agile Method Phases:

- Concept
- Inception
- Iteration
- Release
- Production
- Retirement

Benefits of Agile Method:

Fast Paced Development
Due to the project being divided into sprints with simple achievable goals, the project gets accelerated as everyone is collaborating and working to aid one another.

Reduced Risks/Transparency
Having a client/user actively involved throughout the entire process, decreased the change of errors as well, therefore the risks that are taken are calculated due to the change/edits/addition of certain requirements.

Ability to Customize
Due to the number of iterations the change would appear very fast after client/user request, as the maximum number of weeks for an iteration is 4 weeks.

3.2 Software Engineering and Professional Ethics

As we are living in a world that relies on technology, from the infrastructure of an entire city to a person's wrist. This is all due to Software Engineering, but what is software engineering.

Software Engineering is defined as the "discipline that is concerned with all aspects of software production form initial conception to operation and maintenance". As these techniques and tools are use by professionals, there is also an ethical part to them. Ethics of software engineering is related to the Confidentiality, intellectual property, competence, and misuse. As the market is filled with software and applications, the 4 mentioned ethical responsibilities of an engineering professional.

3.3 Provisional Project Gantt Chart and Task Schedule

As indicated in Section 1.2, we have created a Teamweek.com project to manage our work, which is documented in section 5.4.

3.4 Project Risks

Risk will be managed through a Risk response matrix detailing the likelihood and impact of each eventuality, alongside a plan to be invoked if it occurs. Response and mitigations include

Avoid – Change plans to circumvent the problem;
Control / mitigate / modify / reduce – Reduce threat impact or likelihood (or both) through intermediate steps;
Accept / retain – Assume the chance of the negative impact (or *auto-insurance*), eventually *budget* the cost (e.g. via a contingency budget line); or
Transfer / share – Outsource risk (or a portion of the risk) to a third party or parties that can manage the outcome. This is done financially through insurance contracts or hedging transactions, or operationally through outsourcing an activity.

The greatest risk is associated with and ineffectual Team Lead, and the next, non-or asymmetrically contributing participants.

4.0 Team Structure & Setup

Documentation on team structure and available skillsets follows. Some skillsets will be enhanced and developed during the software development lifecycle. Where this is required, this will be managed in line with the project scope, taking into account design requirements, and the practical experience of delivery under agile and SCRUM.

4.1 Team Members & Team Roles

Crux is composed of three, and not four, team members. In line with agile principles, we aspire to deliver our design as collaborating Full Stack Engineers, who work as Technical Writers when required to do so.

While there may be a need to develop specialisation, our starting point is based on the assumption that the generalised complexity of any particular task is likely to be similar, but the completion time and effort will vary. We believe that we can maximise our likelihood of success by making sure we are able to apportion workload based on priority and load, not based on ring fencing resources in line with an artificial skills demarcation.

This approach largely discounts Belbin Team Role Inventories, and does so deliberately as we are a small team working for a short time on a tightly defined project. We are more likely to succeed through collaboration facilitated by communicating actively and frequently, than by attempting to artificially assign three contributors with data random Belbin profiles, to nine Belbin team Roles.

Since neither the task can be adapted to the team, not can the team be adapted for the task, that approach would add little practical value.

Name	Bio	Role(s)
Rick Fawley	Rick Fawley is an Enterprise Architect for a large International Bank. He's a qualified project manager and an ITIL Expert, but has focussed on Infrastructure and specialist product support, rather than coding, design and development. Professionally he's seeking to qualify as a Google and Azure cloud specialist over the summer.	Team Lead, Full Stack Engineer, Technical & Project Writer
Matt Holburn	Matt is a Tool Hire manager for a national building supplier, that has a large scale IT department. Alongside working full time in this career, he is looking to move into IT based roles, hopefully within networking, or second/third line support. With some knowledge within web development, design and video editing, he likes to configure and build pc's in his spare time. He has been studying Computer Science at the University of Hertfordshire since 2015 and hopes to graduate in 2021.	Full Stack Engineer, Technical & Project Writer
Peter Hanna	Peter recently graduated from the German University in Cairo with a Bachelors Degree in Pharmacy, currently managing a chain of pharmacies. Shifting careers into the IT world, as it is a passion of his hoping to graduate from the University of Herefordshire in 2020.	Full Stack Engineer, Technical & Project Writer

Team Roles are based on the following definitions:

Team Lead

This is a co-ordination and escalation role. It has both responsibility and accountability for establishing the channels for team communication, for work allocation (taking the input and preferences of team members into account), and handing escalations for any problems that are not owned elsewhere.

In a SCRUM environment this is a servant / leader role, characterised as follows:

> The Scrum Master is responsible for promoting and supporting Scrum as defined in the Scrum Guide. Scrum Masters do this by helping everyone understand Scrum theory, practices, rules, and values.
>
> The Scrum Master is a servant-leader for the Scrum Team. The Scrum Master helps those outside the Scrum Team understand which of their interactions with the Scrum Team are helpful and which aren't. The Scrum Master helps everyone change these interactions to maximize the value created by the Scrum Team.

Full Stack Engineer

This definition is based upon the top voted Quora answer to the question What does a "Full Stack" developer do, in simple terms?

Specifically, this summary:

> **What does a full-stack developer mean?**
>
> The term full-stack means developers who are comfortable working with both back-end and front-end technologies.
>
> To be more specific, it means that the developer can work with databases, PHP, HTML, CSS, JavaScript and everything in between, also, venturing as far as converting Photoshop designs to front-end code.
>
> A full-stack developer doesn't need to master all of the areas and technologies he needs to work it, because that just makes it nearly impossible, he just needs to be comfortable working with those technologies, and that's a lot too.

Tech / Project Writer

This definition is derived from the top voted Quora answer to the question How do I become a technical writer?

Specifically, this extract:

> 1. End user documentation (Help Files, Reference Manuals, user guides etc.). These are documents that make complex tasks simple, so even a novice can use a complicated product. These writers are great at breaking down complex tasks and making them seem easy to do.

2. Technical documentations (Datasheets, Installation Guides, repair manuals, inspection guide, programmer guides, engineering specifications etc.). These are written for a technical audience, such as maintenance personnel. These guys are also great writers, but they are also subject matter experts. Having the right credentials and qualifications is often a legal requirement if you want to publish documents.

3. Marcom, also known as technical marketing documentation (technical parts of business proposals, datasheets/spec sheets, etc.). These are documents written for experienced customers who want to know the technical details of a product. Geeks if you will. Because the information has to be EXACT, these documents are written by people from the dev team and not from marketing. Because the dev team isn't usually very good at selling with their words the actual writing is often done by technical writers.

4.2 Software and Project Management Team Skills Matrix

Full Stack Engineer definitions and categories are adapted from What is a Full Stack Developer by Laurent Gellert. The Technical and project Writer definition is derived from the top voted Quora answer to the question How do I become a technical writer?

The sections are deliberately functional, rather than technology or framework dependent. This is because we are designing a new system, not adapting an existing one. Hence the choice of technologies and frameworks is our own.

Capability	**Description**	**Rick Fawley**	**Matt Holburn**	**Peter Hanna**
Server, Network, and Hosting Environment	This involves understanding what can break and why, taking no resource for granted. Appropriate use of the file system, cloud storage, network resources, and an understanding of data redundancy and availability is necessary. How does the application scale given the hardware constraints? What about multi-threading and race conditions? … Full stack developers can work side by side with DevOps. The system should provide useful error messages and logging capabilities.	Advanced	Beginner	Beginner

Capability	**Description**	**Rick Fawley**	**Matt Holburn**	**Peter Hanna**
Data Modelling	If the data model is flawed, the business logic and higher layers start to need [bad] code to compensate for corner cases the data model doesn't cover. Full stack developers know how to create a reasonably normalized relational model, complete with foreign keys, indexes, views, lookup tables, etc.	Intermediate	Beginner	Beginner
Business Logic	The heart of the value the application provides. Solid object oriented skills are needed … Frameworks might be needed here as well.	Intermediate	Basic	Intermediate
API layer / MVC	How the outside world operates against the business logic and data model. Frameworks at this level should be used heavily. Full stack developers have the ability to write clear, consistent, simple to use interfaces.	Basic	Beginner	Beginner
User Interface	Full stack developers: a) understand how to create a readable layout, or b) acknowledge they need help from artists and graphic designers. Either way, implementing a good visual design is key. Can include mastery of HTML5 / CSS. JavaScript is the up and coming language of the future and lots of exciting work is being done in the JavaScript world (node, backbone, knockout…)	Basic	Intermediate	Basic
User Experience	Full stack developers appreciate that users just want things to work … A full stack developer can step back and look at a process that needs 8 clicks and 3 steps, and get it down to one click. Full stack developers write useful error messages [that support and not annoy the user]	Advanced	Basic	Beginner
Understanding what the customer and the business need	[While] blurring into the line of architect … Full stack developers have a grasp of what is going on in the field when the customer uses the software. They also have a grasp of the business.	Advanced	Intermediate	Intermediate

Capability	Description	Rick Fawley	Matt Holburn	Peter Hanna
Coding and build	Practical code creation, testing, integration and release to production	Intermediate	Basic	Basic
Technical & Project Writing	Creation of End User documentation, technical documentation and technical marketing documentation, each written for a different purpose and for a different audience. Documentation can be single authored or collaborative, depending upon the task.	Advanced	Basic	Beginner
Team Lead / SCRUM	This involves acting as a focal point for communication, driving consensus and for escalation. This must be a value added role that facilitates the overall success of both the project and each team member within it, by achieving consensus on tasks, their durations and general expectations of each team member.	Advanced	Basic	Beginner

Additional areas of interest, and specific technical skills

Team Member	Additional areas of interest	Specific technical skills
Rick Fawley	Cybersecurity, Cryptography, AIOps	HTML, Java, Linux, SQL
Matt Holburn	Java, C, C++	HTML, CSS, UI
Peter Hanna	AI, Security	HTML, Java, CSS

5.0 Development, Testing & Deployment

Documentation on our development, testing and deployment strategy follows. Detailed within this section are our target architecture for system development and hosting, as well as our proposed and actual collaboration technologies and mechanisms.

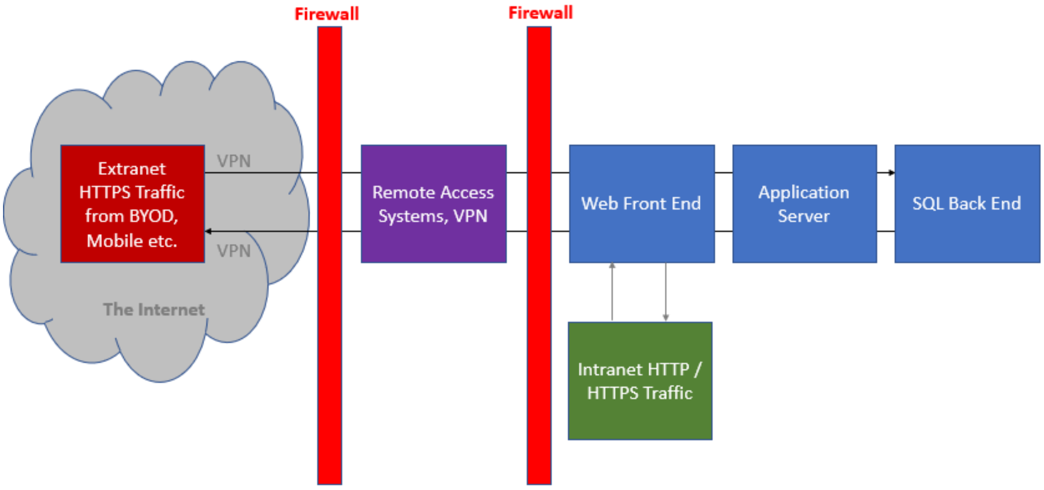
Some of this will be *subject to ongoing review* as we progress through agile and SCRUM stand-ups and cycles, and some will be fixed from the point of decision.

5.1 Target Platform - Web

Discussions with users and other stakeholders suggests that the right solution will be accessible online. Taking Microsoft Doc's word List term collections for Internet, Intranet and Extranet:

> … *internet* refer[s] to the worldwide collection of networks that use open protocols such as TCP/IP to communicate with one another.
>
> … *intranet* refer[s] to a communications network based on web technology but that's available only to certain people, such as the employees of a company.
>
> … *extranet* refer[s] to an extension of an intranet that uses internet protocols to give a uthorized outside users limited access to the intranet.

The right solution is extranet, as our users have a need to access an internal system, which potentially hosts corporate and personally sensitive data, using amongst others, BYOD capabilities. Extranet meets this need.


Firewall
Firewall
Extranet HTTPS Traffic from BYOD, Mobile etc.
VPN
VPN
The Internet
Remote Access Systems, VPN
Web Front End
Application Server
SQL Back End
Intranet HTTP / HTTPS Traffic

Example extranet architecture showing internal access via intranet ('trusted'), and external access via extranet ('trusting').

Development and deployment will use standard tools and techniques with open-source, well documented and easily available systems, where community support is available.

5.2 Development and Testing Platform

In line with recommendations, our development and testing environment will be based on XAMPP, which is a free and open-source cross-platform web server solution stack that contains, amongst others:

- **Apache -** A collaborative software development effort aimed at creating a robust, commercial-grade, feature-rich and freely available source code implementation of an HTTP server.

- **MariaDB -** A community developed, commercially supported fork of the MySQL RDBM, that maintains high compatibility with it.

- **PHP -** Hypertext pre-processor is a general programming language used for web development, that can be executed via CLI (Command Line Interface) embedded into HTML (Hypertext Markup Language).

- **Perl** - A generalised programming language often referred to as 'the glue that holds the internet together', due to it's ubiquity.

In addition, the following will be required:

- **HTML5 / CSS -** Used for formatting the structure and style of content within web-pages, accessible via browser, from a variety of devices and platforms.

- **Python / JavaScript / Java -** Additional programming languages that may be used for back end application coding and development, if required.

These languages and tools are widely available and well proven with strong online community support, and readily available and accessible free training, templates and examples to aid development, hence their selection.

5.3 Project Collaboration and Sharing

Collaboration and sharing will be managed through a variety of channels, depending on the requirements for speed, assured delivery, confirmed delivery, and traceability.

Communication / Collaboration Type	**Channel / Repository**
Short and time critical communication	WhatsApp Group Chat
Formal communication, for example relating to task allocation, with audit trail and history	Team members Preferred Email address, copying all group members
General Shared Document and Object Repository	Office 365 OneDrive Shared Area
Source Code	github
Screen sharing and active collaboration	webex
Project Management Scheduling	Teamweek.com
Other / New	As agreed, if none of the channels above are suitable.

Cadence is expected to be high throughout the most critical development period (April 2019).

Team members will operate with collective delegated authority, actively communicating decisions and assumptions made, rather than federating them and awaiting confirmation - unless federation is explicitly required for the design.

5.4 Links to Online Repositories

The following collaboration communities have been established for use during the development cycle:

Communication / Collaboration Type	**Channel / Repository**	**Link**
General Shared Document Repository	Office 365 OneDrive Shared Area	*https://herts365-my.sharepoint.com/personal/rf17aaw_herts_ac_uk/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Frf17aaw%5Fherts%5Fac%5Fuk%2FDocuments%2FCanvas%2FCrux*
Source Code	github	https://github.com/rickfawley/UH-SDE-Crux.git
Screen sharing and active collaboration	webex (30 day free trial)	https://rickfawley.my.webex.com/meet/rickfawley
Project Scheduling	teamweek	https://app.teamweek.com/#timeline/419034/projects/1842040?zoom=week

6.0 System Requirements

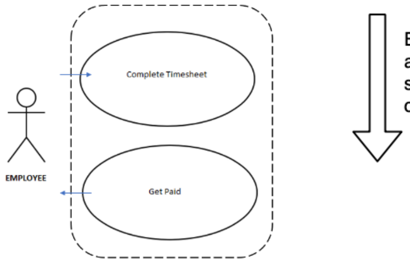
An analysis of system requirements follows, taking into account the expectations of users in terms of actors and their user stories, functional requirements of the system at a high and low level, and the non-functional requirements necessary to maintain the integrity and usability of the system.

6.1 Existing System Use-Cases

The viewpoints of Employee, Manager and Project Officer have been selected for modelling using UML Behavioural Diagrams; each are documented as follows:

6.1.1 **User Story** "As an Employee I need to complete my timesheet so that I get paid"

Timesheet Completion

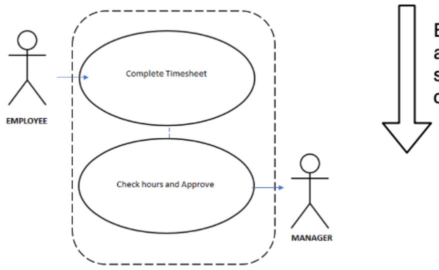


Each Use Case action/behaviour is a step in the system's overall objective

The employee's motivation for completing their timesheet is so they can get paid for the work they have completed. While the processes that sit behind the approval are relevant, they essentially collapse to input and output.

6.1.2 **User Story** "As a Manager I need to know when my people are working excessive hours so I can take appropriate action"

Working Hours Check

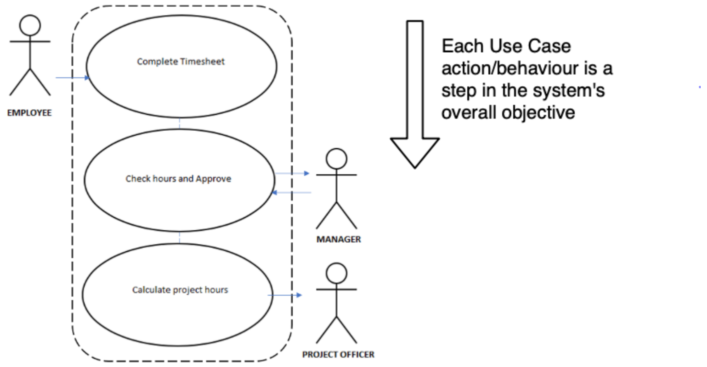


Each Use Case action/behaviour is a step in the system's overall objective

The manager has a duty of care of the employee to ensure they are not working excessively, and to the company to ensure employees are at least completing their obligations. If the manager is not practically managing the project or assignment themselves, they care that the employee inputs their data, not how it is processed, or where it goes next.

6.1.3 **User Story** “As a Project Officer I need accurate data on how much effort is being expended on particular tasks so I can make sure I manage my completion schedule”

Review Project Effort



The Project Officer is concerned with the overall budget and accumulated hours associated with each task, so they can manage the schedule and any additional costs, such as for overtime. Their primary concern is that the employees and their managers have completed the activities allocated to them in a timely manner, so the Project Officer can process and react to the aggregated data.

6.2 System Requirements

This section includes a brief explanation of Functional and Non-Functional requirements.

> **Functional requirements** describe facilities and features. They are concerned with what the software system should do, or provide for users. In ITIL terms these items describe the *utility* of a service – whether it is *Fit for Purpose*.
>
> **Non-functional requirements** are concerned with system quality attributes such as security, usability, performance and maintainability. In ITIL terms these items describe the *warranty* of a service – whether it is *Fit for Use*.

6.2.1 Project’s Functional Requirements

The following functional requirements are taken explicitly from the specification, provided:

1. Users can log in to the app
    a. Username can be an email
    b. Ability to set and change password.
    c. Time sheets are only visible to admins, logged in user & logged in user’s line manager
2. Users shall be able to log time worked into the app.
    a. User should receive a notification to complete a weekly time sheet
    b. There will be a differential between contracted hours & overtime
    c. App will pre-populate with default hours and templates
3. The system will integrate with the payroll / approval process
    a. Upon submission the app shall send notification to user’s line manager for approval.
    b. Line manager will receive notification when a time sheet requiring their approval has been submitted

    c. Line manager can approve / reject time sheet
    d. If approved, the timesheet shall be submitted to payroll system
4. The Timesheet app should indicate bank holidays/full and half days off
5. Bank holidays shall default to 0 hours worked, but can be overwritten.
6. The App shall work across multiple time zones and countries.
    a. Taking into account regional bank holidays

Additionally, from user and stakeholder interviews:

7. The App should provide confirmation to users once timesheets have been approved
8. The App should be accessible remotely, via BYOD

6.2.2 Project's Non-Functional Requirements

The following non-functional requirements are taken explicitly from the specification, provided:

1. Acceptability
    a. The software must acceptable to the type of user for which it is designed. This means it must be understandable, usable and compatible with other systems they use.
    b. System must integrate with existing systems, and be available from the same user journeys (e.g. App, menus' etc).
    c. To ensure the system can be taught to any user within 20 minutes, and all basic functions can be performed
    d. For accessibility purposes, ensure the final UI complies to contrast checks from https://contrastchecker.com/
2. Dependability and Security
    a. Software dependability includes a range of characteristics including reliability, security and safety.
    b. Security – software should be secure so that malicious users cannot access or damage the system
        i. Enforced password policy
        ii. Attempt to force entry locks account
3. Efficiency
    a. Software should not make wasteful use of system resources such as memory or processor cycles. Efficiency includes responsiveness, processing timer, resource utilisation
        i. Software should not use more than 10% of available system resources (taking onto account a minimum supported specification)
4. Maintainability
    a. Software should be written in a way that it can evolve to meet the changing needs of customers.
        i. Software will be maintainable within a predefined outage window (full major upgrade within 30 minutes)
        ii. 90% of releases must require no outage to service

--- END OF REPORT ---