Jebutu, Taiye [19067126]

Team Name | VErsion – 1.0

Project Proposal

Automated Payroll System

Table of Contents

[1.0 Managing Software Development Projects 2](#_Toc138149671)

[1.1 A Comparison of Common Development Strategies 2](#_Toc138149672)

[1.2 A choice of Development strategy 2](#_Toc138149673)

[2.0 Project Outline 2](#_Toc138149674)

[2.1 Problem Statement & Project Background 2](#_Toc138149675)

[2.2 Client Requirements 2](#_Toc138149676)

[2.3 Business Case - Project Goals & Benefits 3](#_Toc138149677)

[2.4 General Aims & Objectives 3](#_Toc138149678)

[3.0 Project Management and Development Strategy 3](#_Toc138149679)

[3.1 Preferred Approach to Project Management and Development 3](#_Toc138149680)

[3.2 Provisional Project Gantt Chart and Task Schedule 4](#_Toc138149681)

[3.3 Project Risks 4](#_Toc138149682)

[4.0 Team Structure & Setup 4](#_Toc138149683)

[4.1 Team Members & Team Roles 5](#_Toc138149684)

[4.2 Software and Project Management Team Skills Matrix 5](#_Toc138149685)

[5.0 Development, Testing & Deployment 6](#_Toc138149686)

[5.1 Target Platform 6](#_Toc138149687)

[5.2 Development and Testing Platform 7](#_Toc138149688)

[5.3 Project Collaboration and Sharing 7](#_Toc138149689)

[5.4 Link to Online Repository 8](#_Toc138149690)

[6.0 System Requirements 8](#_Toc138149691)

[6.1 Existing System Use-Cases 8](#_Toc138149692)

[6.1.1 System Requirements 8](#_Toc138149693)

[6.2 Project’s Functional Requirements 8](#_Toc138149694)

[6.3 Project’s Non-Functional Requirements 9](#_Toc138149695)

# Managing Software Development Projects

*Provide a short introduction about the purpose of this section of the report (up to 50 words)*

## 1.1 A Comparison of Common Development Strategies

*After a brief (initial and individual) exploration of the topics below, as a group – decide, discuss and document (here) three common strategies; common strategies include: Waterfall, Prototype, Dynamic, Agile, Spiral. Extreme Programming, RAD. Try to include details such as the general origin, similarities, each method’s overall approach, typical implementation as well as the typical and commonly understood advantages and disadvantages. (up to 400 words)*

## 1.2 A choice of Development strategy

*Discussion of your chosen strategy with suitable justification (up to 200 words)*

# Project Outline

*Provide a short introduction about the purpose of this section of the report (up to 50 words)*

## 2.1 Problem Statement & Project Background

*o help you plan and complete this section – try to define who the client is (and what they do) then explore, establish and document the project’s background and possible origin; consider issues such as (for example): the business need that relates to the problem and (now) this project, the current setup (what the existing method or system is – if there is any?), existing issues/constraints/limitations, user experience with the existing system as well as its purpose and requirement, etc.). (up to 250 words)*

## 2.2 Client Requirements

*As a team, identify, agree and document in a prioritised order (based on any direct, indicated or inferred information) your client’s requirements for this project. Try to limit the number of requirements to a maximum of 10 and separate the “must haves” from the “should haves”, etc. For more information on structuring requirements explore the prioritisation method called “MoSCoW”. Client requirement are generally focused on the business need for information or process and are used to help create (provide a basis for) the overall project’s goals, aims and objectives. (up to 200 words)*

## 2.3 Business Case - Project Goals & Benefits

*As a team, discuss and agree 2 primary, business goals for the project and support them with a narrative that explains in more detail the purpose and nature of the project’s goals; include a considered analysis of the possible investment needed (e.g. what might the “client” need to provide or problems or issues might they need to appreciate; financial, time, skill, engagement with the project/development team) and what the anticipated return or the possible benefits of that investment will be (e.g. financial reductions, more efficient or more accurate collection of data, happier staff, etc.); this section will help illustrate yours’ and the client’s responsibilities. You should also include a reflection on what the possible impact may be if the client does not do the project. (up to 350 words)*

## 2.4 General Aims & Objectives

*Review the module’s related unit on project aims and objectives for information and consideration about how best to address this section; Ideally try to ensure that you use SMART when considering your project’s objectives (try to keep your objectives to a maximum of 12)*

# Project Management and Development Strategy

*Provide a short introduction about the purpose of this section of the report (up to 50 words)*

## 3.1 Preferred Approach to Project Management and Development

*As a team and after reviewing the common approaches (discussed in section 1.1) explain and justify your preferred/chosen approach to managing and developing this (your) solution; you are free to take different aspects of different strategies and blend this together to better suit your teams’ style or agreed/preferred method of working.*

*Don’t forget, in addition to explanation of the approach you should also include a critical review of the potential benefits as well as any possible, associated issues; explain how these possible issues may be mitigated in section*

*3.3 Risks. (up to 500 words)*

## 3.2 Provisional Project Gantt Chart and Task Schedule

*Regardless of the variety of different software development methodologies and lifecycles – there are (in general) several “key” activities that must be completed in order to successfully design and deliver a working software solution. In some methodologies these activities are repeated and further developed over time. However, for the purpose of this module (and regardless of your team’s selected approach) your client would like to see a provisional (i.e. subject to change and modification) schedule for the following common tasks: 1) gathering and analysing requirements, 2) system/interface/workflow design, 3) software development, 4) system testing and finally – 5) delivery. As a team, research, explore and produce a simple Gantt chart that illustrates when you think these tasks will start and the time period during which they will be able to be completed – try to ensure that your delivery date coincides with your clients expected demo. When preparing a Gantt chart, you specify “considered” timeframes (similar to Agile “timeboxes”) during which the tasks will be completed; please remember that these times are not the required time to complete it is the time “during which” the task can be completed. For example: if a task is expected to take 8 hours of work – you could schedule it over 3 days (meaning that the 8 hours will be invested somewhere during that period). By doing this you can build in a level of tolerance to delays and problems – however if the completion of a task is linked to the start of another, you might find that you quickly run out of time; the key is to try to carefully balance time needed with time wanted. Also, don’t forget to consider possible contingencies when planning and scheduling time. As part of your Gantt chart you will be expected to include short explanations justifying your provisional dates and times.*

*Although many people may offer alternative views - using Gantt charts for monitoring “actual” progress (in and of themselves) is generally not possible. Gantt charts do not normally or accurately show (or chart) a project’s actual progress – there most obvious value is typically early project planning and the ability to clearly visualise activities and illustrate linked and concurrent tasks with the project’s key milestones. (up to 200 words)*

## 3.3 Project Risks

*Review the module’s related unit on risk management for information and ideas about how best to address and apply possible considerations to this section. (up to 200 words)*

# Team Structure & Setup

*Review the module’s related unit on risk management for information and ideas about how best to address and apply possible considerations to this section. (up to 50 words)*

## 4.1 Team Members & Team Roles

*Provide a brief introduction to your team, include a short (up to 80 word) bio on each member as well as a general description of their team “title”. In addition, also include details on agreed general roles and responsibilities.*

*Although you are free to agree your own team’s roles - as a possible starting point (and to help you identify common roles and responsibilities) it is suggested that you further explore and consider Belbin’s team roles.*

## 4.2 Software and Project Management Team Skills Matrix

*In industry, teams are assembled or assigned for many different reasons and in many different ways – however, regardless of whether (as a manager) you can support, motivate or positively influence your members it is important to know what your team can handle.*

*Although this is not an industry environment it is designed to be a professional, work-related exercise – as such (and as a group), discuss, develop and create a “team” based “Skills Matrix” that confirms the skills, knowledge and interest of your team members.*

*In general, your skills matrix is simply a table that displays people’s proficiencies (i.e. their skills and knowledge) in specified areas of experience or interest. As part of your project’s management you will collectively need to know what each member of your team is able to do (as well as any areas of their personal interest) or aspects that may need to be further developed or explored.*

*The overall purpose of this module is to create opportunities for you to gain knowledge, new insights, consideration and experience while working to develop a software-based solution to your client’s brief. Use this matrix to help establish a general base-line of your team’s collective skills and capabilities and to better review, manage, design, and develop your team’s custom software project.*

*As a starting point, consider skills such as: Technical Writing, Research, Graphic Design, Programming, Database Design, etc. Give each skill a metric value that indicates the current level of skill or knowledge, for example:*

* *0 may be no experience/skill*
* *1 may be a basic level of knowledge or understanding*
* *2 may be an intermediate level of skill with regards to that aspect,*

*Try using or adapting the following points to help you think about how to complete your team’s matrix:*

*Each member should discuss his or her skills, knowledge and interests related to the thought-about activities associated with this*

*Each member should have the opportunity to explain or defend their level of interest in working on the tasks that she or he has asked for or been initially given responsibility*

*Incorporate all the gathered information into your skills matrix before finally reviewing and analysing each member’s portion; in addition to helping identify areas of strength and possible weakness with regards to potential areas of project risk - a matrix can also be used to more fairly distribute*

*Try to ensure you have back-ups for more complex challenges*

# Development, Testing & Deployment

*Provide a short introduction about the purpose of this section of the report (up to 50 words)*

## 5.1 Target Platform

*Explain (in some detail) the target platform that will be used to host and support your solution. This should be based on your client’s brief together with your team’s assessment and understanding of what the needed, expected, wanted or ‘nice to have’ features associated with your clients requirements are – for example, if the requirements mean that your solution must support multiple users and be access via mobile devices, or web browsers then you should consider using an online platform i.e. Internet hosted or via a locally managed extranet based service. If for example, you are designing a web based system you should use this section to briefly explain the difference between an Internet, intranet and extranet web site as well as provide an overview on how your solution might work and integrate (use) browsers, login security, hosting, client & server-side programming, HTML & CSS together with a database system. If you are designing an installed application, you should frame the above points, so the same level of detail is provided but related to your specific platform.*

*For example, if you are intending to develop your project as a web-based solution (and to help you decide how best to setup and configure your team’s development and testing environment) we would suggest that you consider using standard tools and techniques together with open-source, well documented and easily available systems – a common, example might be:*

*Apache: Web-servers are (typically) software that is designed to respond to individual user requests for information by creating, formatting, adapting and providing (i.e. serving) copies of specific (locally stored) web pages (e.g. HTML documents) back to the requesting*

*HTML5 / CSS: these (non-programming) languages are used to format the structure and style of content within a typical web-page; web-pages are (in essence) HTML “documents” that are viewable and usable by a web browser (e.g. Chrome, Edge, Firefox) exactly the same as DOCX documents are viewable and usable by Word, or XLSX documents are usable by Excel).*

*MySQL or MariaDB: Database servers are typically setup (within a multi-user or shared user environment) to allow systems (i.e. other services, or users, etc) to hold, modify, organise, control and access (or limit access) to requested information; a common language to talk to databases is*

*PHP/Python/Perl/Ruby: these programming languages are used (in this type of environment) to extend the capability of your web-server so that it can access, store, use or modify information within a database. In addition, these languages (and many others) can be designed to make decisions about what content should (or should not) be shown to a specific user. In essence, these languages (and their results) are made available to the web-server (typically) as “scripts” embedded within a web document; the web-server executes (i.e. runs) the “script” and can insert the result anywhere within a page in order to create “custom” or “unique” content. Clear examples of this can be seen when using such sites as Amazon or Google – although everyone accesses the “same” set of pages the results are “user”*

*There are a number of open-source and free, full-stack development and testing environments that would provide your team with a reasonable platform for local development and testing, these include:*

*XAMPP (Windows/Linux/Mac OS)*

*WampServer (Windows)*

*MAMP (Mac OS/Windows)*

*Please note: some operating systems (Windows included) can be installed with its own local web server (e.g. Windows can be configured to use Microsoft’s IIS, while Mac OS can be setup with its own version of Apache) – although these can be fully configured and setup as local development and testing environments the three packaged applications (listed above) will likely be easier, faster and better documented. You, as a team will be expected to make a well-considered decision about what (and which) platform you feel best suits your needs and requirements. As a module – we will be providing examples using XAMPP and MAMP – however, all our examples should be comparable and even if not directly compatible with your version will provide reasonable insights about possible issues and potential solutions to specific problems. (up to 150 words)*

## 5.2 Development and Testing Platform

*Explain and justify your selected/preferred platform for development and testing. As mentioned above, part of your team’s initial discussions should be on which platform to agree and use – there is likely not a (direct) right or wrong answer to this, it may be personal choice or be based collectively on your group’s cumulative or existing experience. However, it is suggested that you seek to agree, setup and test your local development environment as quickly as possible. It is also suggested that each member of your team installs their own version of the chosen platform (local to themselves) and that you “share” and collaborate on the software development. (up to 200 words)*

## 5.3 Project Collaboration and Sharing

*Provide a short introduction about your team’s preferred method of collaboration; include details on the function and value of using online collaborative tools for working in teams where members are geographically separated.*

*There are many tools that support geographically separated teams and projects – however, we would ask that (as a module) your team considers using GitHub. Although you are free to choose whichever tool best suits your team - GitHub is a widely known, industry focused service that supports a useful range of features. Regardless of whether you decide to use GitHub, you will need to setup an online repository for your team’s project – at the end of your development, your repository should include:*

*A full project description*

*Copies of all your developed code (including internal comments) (up to 80 words)*

## 5.4 Link to Online Repository

*Provide a valid link to your team’s GitHub (or equitant) project repository.*

# System Requirements

*Provide a short introduction about the purpose of this section of the report (up to 50 words)*

## 6.1 Existing System Use-Cases

*Provide 3 general (but well considered and simply annotated) Use-Cases that reflect your team’s carefully considered knowledge of the client’s existing (or ideal) system; take time to review each diagram so that you are satisfied it is a reasonable illustration of the current process. As a team, decide which 3 viewpoints (perspectives) you should document so as to demonstrate a reasonable understanding of the system as it currently is. (up to 80 words)*

### 6.1.1 System Requirements

*Use this section to provide a (team agreed) brief explanation of what a function and non-functional requirement is and how it relates to a project.*

## 6.2 Project’s Functional Requirements

*As a team, agree and document your project’s primary functional requirements; as part of this task, consider how best to identify, prioritise and format each requirement so that you include a brief explanation of each point and how it relates to the client’s requirements and/or project (see section 2). Try to limit the number of requirements to a maximum of 10.*

## 6.3 Project’s Non-Functional Requirements

*As a team, agree and document your project’s primary non-functional requirements; as part of this task, consider how best to identify, prioritise and format each requirement so that you include a brief explanation of each point and how it relates to the client’s requirements and/or project (see section 2). Try to limit the number of requirements to a maximum of 10.*