**FACULTY OF SCIENCE, ENGINEERING AND COMPUTING**

**School of *Computer Science & Mathematics***

**BSc DEGREE**

**IN**

**COMPUTER SCIENCE (SOFTWARE ENGINEERING) TOP UP**

**PROJECT FINAL REPORT**

Name: Tuan Saad Ousman

ID Number: K2168332

Project Title: Construction Project Management System

Project type: Build

Date: 01/05/2022

Supervisor: Mr. Hazly Mohammed

KU London Logo

Did you discuss and agree the viability of your project idea with your supervisor? Yes

Did you submit a draft of your proposal to your supervisor? Yes

Did you receive feedback from your supervisor on any submitted draft? Yes

**Abstract**

This Final report provides information on the completed construction project management system build project. The intended end-users of the system are stakeholders involved in a construction project. The overarching goal of the project is to digitize and improve collaboration, communication, and progress tracking by providing a simple interface and workflow for process approvals, progress updates, report generation, and other activities that form the construction project management workflow.

The Contractor, Client, and Consultants will be the end-users of the software system. The system will be implemented in the form of a Web Application, allowing users to access the system via their browsers from anywhere.

With a clear central source of truth and communication regarding the Construction project, the chances of vital notifications and information being overlooked will be reduced.

The contractors will be able to add documentation, update project progress and submit various requests that need to be approved by the client and consultants. The consultants and clients will be able to approve these requests made by the contractor and update any required changes that will be agreed upon by both parties.

PDF report generation will help inform the stakeholders on the status of completed project tasks, pending approvals, unexpected and expected delays, etc. Digitization of reports will reduce the need for paperwork that can be missed and cumbersome to organize and maintain.

The client will be involved throughout the entire software development lifecycle, as the project will be implemented using the Agile-Scrum development methodology.

Emphasis will be made on making the front-end interface intuitive and user-friendly for clients with limited computer literacy.

The system will be implemented using HTML, CSS, JavaScript, and Python using Python’s Flask Framework for the backend and the Bootstrap Framework for the Front-end implementation.

The goal of the project is to provide a professional, user-friendly construction project management system that is accurate and efficient.

Contents

[1. Introduction & Literature Review 7](#_Toc102314969)

[1.1 Introduction 7](#_Toc102314970)

[1.2 Background and Motivation 8](#_Toc102314971)

[1.3 Problem in brief 10](#_Toc102314972)

[1.4 Aim & Objectives 11](#_Toc102314973)

[1.4.1 Aim 11](#_Toc102314974)

[1.4.2 Objectives 11](#_Toc102314975)

[1.5 Scope 12](#_Toc102314976)

[1.6 Deliverables 14](#_Toc102314977)

[1.6.1 Software Requirements Specification 15](#_Toc102314978)

[Purpose 16](#_Toc102314979)

[Scope 17](#_Toc102314980)

[System Risks 18](#_Toc102314981)

[Functional Requirements 19](#_Toc102314982)

[Non- Functional Requirements 20](#_Toc102314983)

[1.6.2 Project Plan 21](#_Toc102314984)

[1.7 Literature Review 22](#_Toc102314985)

[2. Analysis 23](#_Toc102314986)

[3. Design 28](#_Toc102314987)

[3.1 Design Techniques 29](#_Toc102314988)

[3.2 System Overview 34](#_Toc102314989)

[4. Product Implementation 38](#_Toc102314990)

[5. Validation 46](#_Toc102314991)

[6. Critical Review & Conclusion 56](#_Toc102314992)

[6.1 Closing executive summary 56](#_Toc102314993)

[6.2 Conclusion 57](#_Toc102314994)

[References / Bibliography 58](#_Toc102314995)

[Appendices 59](#_Toc102314996)

**List of Figures/Tables**

**Figures**

[Figure 1.6.2.1: Project Gannt chart 21](#_Toc102314997)

[Figure 2.1:User Stories: Contractor 23](#_Toc102314998)

[Figure 2.2:User Stories: Consultant 23](#_Toc102314999)

[Figure 2.3:User Stories: Client 23](#_Toc102315000)

[Figure 2.4:Use case diagram 24](#_Toc102315001)

[Figure 2.5:Reverse brainstorming board 26](#_Toc102315002)

[Figure 3. 1: MVC Architecture 28](#_Toc102308232)

[Figure 3.1. 1: Agile-Scrum Methodology 29](#_Toc102315003)

[Figure 3.1. 2: Entity-relationship diagram 30](#_Toc102315004)

[Figure 3.1. 3: Activity Diagram 31](#_Toc102315005)

[Figure 3.1. 4: Sequence Diagram 32](#_Toc102315006)

[Figure 3.1. 5: Flowchart 33](#_Toc102315007)

[Figure 3.2. 1: System Architecture Diagram 34](#_Toc102315008)

[Figure 3.2. 2: Model implemented as a python class 35](#_Toc102315009)

[Figure 3.2. 3: Object relational mapper flow 35](#_Toc102315010)

[Figure 3.2. 4: Wireframe for the application dashboard 36](#_Toc102315011)

[Figure 3.2. 5: Wireframe for the Delay Management page 36](#_Toc102315012)

[Figure 3.2. 6: Initial Version: Dashboard 37](#_Toc102315013)

[Figure 3.2. 7: Final version: Dashboard 37](#_Toc102315014)

[Figure 4. 1: SQLAlchemy functions: Create record 42](#_Toc102315015)

[Figure 4. 2: SQLAlchemy functions: Deleting records 42](#_Toc102315016)

[Figure 4. 3: Flask-login functions: log in user 43](#_Toc102315017)

[Figure 4. 4: Flask-login functions: log out user 43](#_Toc102315018)

[Figure 4. 5: PDFKit functions: generation a PDF file 43](#_Toc102315019)

[Figure 4. 6:Flask functions: downloading files from local directory 44](#_Toc102315020)

[Figure 4. 7:BCrypt:Generating and validating password hashes 44](#_Toc102315021)

[Figure 4. 8: Source code structure 45](#_Toc102315022)

[Figure 5. 1: Test Strategy 46](#_Toc102315045)

[Figure 5. 2: Functional test case 1 49](#_Toc102315046)

[Figure 5. 3: Functional test case 1 (passed) 49](#_Toc102315047)

[Figure 5. 4: Functional test case 2 50](#_Toc102315048)

[Figure 5. 5: Functional test case 2 (passed) 50](#_Toc102315049)

[Figure 5. 6: Functional test case 3 51](#_Toc102315050)

[Figure 5. 7: Functional test case 3 (passed) 51](#_Toc102315051)

[Figure 5. 8: Functional test case 4 52](#_Toc102315052)

[Figure 5. 9: Functional test case 4 (passed) 52](#_Toc102315053)

[Figure 5. 10: Functional test case 5 53](#_Toc102315054)

[Figure 5. 11: Functional test case 5 (passed) 53](#_Toc102315055)

[Figure 5. 12: Functional test case 6 54](#_Toc102315056)

[Figure 5. 13: Functional test case 6 (passed) 54](#_Toc102315057)

**Tables**

[Table 1 : Excerpt from the questionnaire survey (Kesavan et al. 2015) 8](#_Toc102315058)

[Table 2: SWOT Analysis 13](#_Toc102315059)

[Table 4: Usability Test Report 47](#_Toc102315060)

[Table 5: Functional Test Report (Task Management Module) 48](#_Toc102315061)

[Table 6: Client Feedback Report 55](#_Toc102315062)

**Glossary of Terms**

**CRUD:** Create, Read, Update, Delete

**CSS:** (Cascading Style Sheets) a style sheet language used for describing the presentation of a document written in a markup language such as HTML

**DB:** (Database) an organized collection of structured information, or data, typically stored electronically in a computer system

**Git:** Software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code

**HTML:** (Hypertext Markup Language) The standard markup language for documents designed to be displayed in a web browser.

**MVC:** (Model-View-Controller) A pattern in software design commonly used to implement user interfaces, data, and controlling logic

**RAM:** (Random-access memory) A form of computer memory that can be read and changed in any order, typically used to store working data and machine code

**RBAC:** (Role-based access control) A method of restricting access based on the roles of individual users

**SRS:** (Software requirements specification) A description of a software system to be developed.

**SWOT:** (Strengths, Weaknesses, Opportunities, and Threats) A strategic planning and strategic management technique for identifying strengths, weaknesses, opportunities, and threats in business competition or project planning.

**URL:** (Uniform Resource Locator) A reference to a web resource that specifies its location on a computer network.

**VM:** (Virtual machine) The virtualization/emulation of a computer system

**WSGI:** (Web Server Gateway Interface) A simple calling convention for web servers to forward requests to web applications or frameworks written in the Python

# Introduction & Literature Review

## 1.1 Introduction

A Construction project at any scale demands a great deal of effort put into its management. Due to many concurrent activities like material purchasing, variation requests , change requests, etc. that take place during a project, the increasing complexity associated with the management and tracking of these tasks increases the demand for systems and processes to for their simplification.

The keys to increasing the odds of a project meeting its deadlines on budget successfully are effective communication and collaboration between the Stakeholders including the client, contractor, and consultant.

This is where this build project comes in with its main objective being to implement an efficient, user-friendly software solution that caters to the needs of collaboration and project management in construction projects in key areas that are well-known to make the most impact on the overall success of the project.

Through this solution, stakeholders can be relieved of the burden of manually keeping track of approvals, requests, project progress, payment requests, etc., which results in a reduction of associated delays.

With a clear source of truth and real-time notifications, the accountability of stakeholders is made clear, and tasks and approvals required to be carried out from their ends are well defined and notified to all users. With the dates and times of each task being captured, the system provides data that can be reviewed and audited to validate activities by the users.

## 1.2 Background and Motivation

**Background**

The construction industry is one of the most dynamic and volatile sectors that contributes to the development of a country’s economy. It’s an effective method through which the society reaches their targets of developing the rural and urban areas.

KPMG’s 2015 Global Construction Survey of project owners found that most projects failed to come within 10 percent of budget or deadline, with over half of respondents suffering one or more underperforming projects in the previous year (Armstrong and Gilge, 2016).

When delays take place during a construction project is in process, this can lead to issues such as the increase of expenses, cost incensement, disputes, working rapidly to complete the project, efficiency loss, and not being able to complete the project on time, may end up in termination of the Contract (Trauner, 2009).

The reasons for the delays can be either client or contractor related. Delays in approvals and payments along with poor decision making are major reasons for delays by the client. Incorrect assessment of activities or lack of task clarity and progress updating may lead to delays by the contractor. Most of the above issues stem from ineffective communication and collaboration between clients and contractors.

Proper communication among clients, consultants, contractors, subcontractors, and suppliers is very much important to meet each of their requirements with fewer conflicts. The client should provide all necessary details and information (shop drawings, modifications, etc.) to the contractor as requested. Responding of each party at their earliest is highly required to avoid delays. (Kesavan, Gobidan and Dissanayake, 2015).

The data analysis carried out with a questionnaire survey that included 107 Sri Lankan construction projects, based on the Liker’s scale of five ordinal measures from 1 to 5 (very low effect to very high effect), revealed the causes for Construction Project Delays. (Kesavan et al. 2015). Five of which are stated in the below table:

Table 1 : Excerpt from the questionnaire survey (Kesavan et al. 2015)



**Motivation**

* Popular Construction Project management tools such as Oracle Primavera currently in the market are costly and sophisticated and therefore require additional expertise for their use and maintenance. Hence their usage and benefits are limited mostly to large-scale Construction Projects.
* The motivation behind this project is to improve and simplify Construction Project Management and Project collaboration in Sri Lanka make with a software system that is more accessible, user-friendly, and cost-effective for stakeholders, especially in low-rise to medium-rise projects thus contributing to the reduction in prevalent delays and associated financial losses faced by stakeholders involved.

## 1.3 Problem in brief

* Construction projects are complex and require close coordination and collaboration among Stakeholders involved during each phase of a project.
* In low-rise to medium-rise projects with tight budget constraints, requests, and approvals for Construct-specific processes such as Work inspections, Interim Payments, Material Inspections etc., are commonly issued and exchanged via email and printed documents due to a lack of adopting proper communication and issue tracking systems. This can lead to an exchange of a large number of emails that can be cumbersome to organize while having the potential to be overlooked. Increased paperwork can also be difficult to maintain.
* The transparency of the progress of the project can be limited due to the lack of effective, easily interpretable, real-time project progress updating and monitoring.

These factors contribute towards Construction project delays and the resulting financial losses.

## 1.4 Aim & Objectives

### 1.4.1 Aim

This project aims to design and develop a Web-Application that can streamline and simplify Project Management, Communication, and Collaboration between stakeholders involved in a construction project while increasing overall transparency of progress.

### 1.4.2 Objectives

The objectives of this project are as follows:

1. To conduct a critical review of the current state of software tools used in Construction Project Management.
2. To identify software technologies that can be leveraged to address the problem.
3. To design and implement a software solution to address the problem.
4. To validate and evaluate the proposed software solution.
5. To prepare and validate the documentation and training material.
6. To reduce overall complexity in construction project management with interfaces designed for increased user-friendliness
7. To reduce delays by implementing clear workflows and notification channels
8. To democratize the use of construction project management applications among stakeholders in projects of any scale and budget.

## 1.5 Scope

1. **Justification**: There is a lack of user-friendly, cost-efficient solutions in the marketplace that addresses the need to digitize and automate workflows that are unique to construction projects.
2. **Scope** **Description**: The software solution will be used to manage and communicate only information and data that is produced by the input of stakeholders involved in the project.
3. **Business** **Objectives**: The solution will increase the velocity in which projects progress with reduced delays, because of swift approval processes. This in turn, increases project success rates with deadlines which benefits all stakeholders financially as result of incurring fewer losses that usually arise due to project timelines being stretched further than initially planned.
4. **Project** **Exclusions**: Additional enhancements such as delay prediction using weather analysis and recommendations based on economic trends are planned for future releases and are intentionally not included in this project given the time constraints, current technical feasibility**,** and priority placed on more business-critical features that will add immediate value.
5. **Constraints***:* Given the timeline for the completion of the project, more emphasis is given on implementing the key functionalities that will make the most immediate impact on the end-user. And so, various functionality available in high-end systems will not be included in the scope of this project, at least in the initial versions used by the client, but will be documented as change requests to be designed and implemented in future iterations.
6. **SWOT Analysis:**

Table 2: SWOT Analysis

|  |  |
| --- | --- |
| **Strengths**   * User-friendliness: Increased usability for non-technical users * Reports and Updates are real-time and easy to interpret * Cloud-based and accessible from anywhere in the world | **Weaknesses**   * Functionality is limited to addressing critical needs and workflows * Requires stable internet connectivity as the application is web-based |
| **Opportunities**   * There is a lack of affordable and user-friendly project management tools that cater to the workflows specific to construction projects * Due to the COVID pandemic, there is a need for remotely accessible, automated processes that eliminate manual paper-based document handling and signature verification | **Threats**   * Current established software solutions can integrate similar functionality. * Insufficient data protection and security practices can lead to liability |

## 1.6 Deliverables

* Final Working product – The Construction Project Management Web Application
* Final Documentation – Software Requirement Specification, System Design, Project Plan, System Architecture Diagram etc.
* System Validation Documentation - Reports of testing. This will include unit testing, functionality testing, user-acceptance testing etc.
* User Training Guide – Video based Instructions on the usage of the software system.

## 1.6.1 Software Requirements Specification



SOFTWARE REQUIRemENTS SPECIFICATION

Constructify – Construction Project Management Application



## Purpose

Construction projects require digitized, automated project management workflows to be executed effectively and efficiently in a timely manner.

The purpose of the software system is to provide online workflows for project management and automated notifications between stakeholders involved in low-medium rise construction projects.

**Major Functions:**

* Authenticating users by using the assigned usernames and passwords
* Creating, Reading, Updating and Deleting records for Project Tasks, Delays, Material Inspection Requests, Work Inspection Requests, Payment Requests, and Variation requests.
* Notifying users of project updates via SMS and email notifications
* Printing PDF reports pertaining to each project module
* Sending PDF reports to all stakeholders via email
* Managing and sharing files and images related to project items
* Viewing project progress via dynamic charts and infographics
* Creating and managing users
* Collaborating and communicating with stakeholders in the project via chat features
* Role-Based access control

**Intended Audience (Construction project stakeholders):**

* Clients: Individuals paying for the project
* Consultants: Quantity surveyors, Quality assurance engineers and Architects representing the client
* Contractors: Engineers, Procurement officers, Architects etc., responsible for building the construction project.

## Scope

**Procedures:**

* Contractors create Task, Delay, Material Inspection, Work Inspection, and Payment Request records and submit the relevant supporting multimedia files or documentation
* Consultants view the Task, Delay, Material Inspection, Work Inspection, and Payment Request records created by the contractor and update the records to have them approved, marked for resubmission, rejected etc., while submitting the supporting documentation.
* Contractors create Variation Request records and submit the supporting documentation
* Consultants view the Variation Request records submitted by the Contractors and have them marked as approved or rejected while submitting the supporting documentation.
* All Stakeholders can generate PDF reports of the overall Task Status as well reports of individual records (Tasks, Delays etc.)
* All stakeholders of the project receive notifications of project updates via SMS and email
* The user accounts of all stakeholders are managed by the client
* Each stakeholder has a role in the system through which access to functionalities is controlled with RBAC.

**The online web-based construction project management system provides several benefits to users:**

* The application can be accessed from anywhere in the world
* Installation of the application is not required as it is web-based and can be accessed via any web browser. This eases the onboarding of users onto the application.
* Intuitive, easy to use interfaces with emphasis on simplicity increases usability. This is important, especially for clients with limited computer literacy having limited exposure to similar applications

**The online web-based construction project management system helps users accomplish key objectives:**

* Drastically reducing delays that arise from poor communication channels and workflows via back-and-forth emails. This is done by providing a clear record of requests and their related documentation along with instant notifications of said records being sent to every stakeholder in the project.
* With a single source of truth in the form of dynamic charts and infographics via dashboards and comprehensive PDF reports, the possibility for information misinterpretation is eliminated, thus ensuring accountability of each stakeholder of the project.
* Helps prevent duplication of project requests and approvals
* Requires minimal administrative effort to maintain the system, thus reducing the cost of usage.

## System Risks

**Mentioned below are risks associated with the system:**

* Risk of application failure due to malfunctions on the cloud infrastructure.
* Lack of internet connectivity can disrupt access to the system, thereby causing delays in sending out critical requests and reports
* Inadequate data protection and cyber security mechanisms can lead to a data breach.

## Functional Requirements

1. Client:
2. View all project updates (Tasks, Delays, Material inspections, Work inspections, Payment requests, Variation requests)
3. Print PDF reports
4. Email PDF reports to stakeholders
5. Create, update, and delete Stakeholder User accounts
6. View Media and Documents submitted by contractors and consultants
7. Chat with all stakeholders via a group chat
8. Consultant:
9. View all project updates (Tasks, Delays, Material inspections, Work inspections, Payment requests)
10. Print PDF reports
11. Email PDF reports to stakeholders
12. Approve Variation Request records
13. Update/approve records of Tasks, Delays, Material inspections, Work inspections, Payment requests
14. Update Documentation for Delays, Material inspections, Work inspections, Payment requests, and Variation Requests.
15. Chat with consultants in a private chat
16. Chat with all stakeholders via a group chat
17. Contractor:
18. Create and view all project updates (Tasks, Delays, Material inspections, Work inspections, Payment requests)
19. Delete project updates (Tasks, Delays, Material inspections, Work inspections, Payment requests)
20. Update the status of project tasks
21. Create Variation Request records
22. Print PDF reports
23. Email PDF reports to stakeholders
24. Update Documentation for Delays, Material inspections, Work inspections, Payment requests, and Variation Requests
25. Chat with other contractors via a private chat
26. Chat with all stakeholders via a group chat

## Non- Functional Requirements

**01. Hardware requirements.**

* Processor Intel (Pentium 4 or newer)
* Operation System: Linux (64-bit Ubuntu 18.04+ or newer)
* 4GB RAM minimum
* 250GB hard disk storage minimum.
* Internet connection.

**02. Software requirements.**

* Windows 7 operation system or newer.
* Google Chrome Internet browser.

**03. Application security.**

* Form data sent from the user browser to the application server is encoded
* User passwords saved in the database are encrypted
* With RBAC enabled, users are only allowed to perform tasks for which they are authorized
* Application data on disk hosted in Azure cloud VMs are encrypted at rest
* Using SSL certs for the Domain that is used for hosting the application enables encryption of data transmitted between the clients’ browsers and the web server hosting the application

**04. Data safety.**

* Regular backups of the SQLite Data file can be scheduled to avoid data loss.

**05. User experience.**

* Intuitive interfaces with fast loading times are implemented.

## 1.6.2 Project Plan

The Agile-Scrum based project plan Gannt chart used for the build project:

Figure 1Timeline

Description automatically generated.6.2.1: Project Gannt chart

## 1.7 Literature Review

1. **Appraisal of existing Construction project Management Tools**

Upon research into the functionalities of popular construction management tools in the market such as GanttPRO, Buildertrend, etc. (Top 19 Construction Project Management Software in 2022 - Reviews, Features, Pricing, Comparison - PAT RESEARCH: B2B Reviews, Buying Guides & Best Practices, 2022), it was made apparent that Project Tracking especially is implemented effectively with the existing software solutions. Other Popular construction project management tools such as Oracle Primavera come with some collaborative features and are widely used in large enterprise projects and require additional expertise for their management and maintenance which significantly increases the cost of operation.

However, most of the tools mentioned above cater more to the project management and tracking needs of the contractors and do not include functionalities for simplified, effective management of requests and approvals specific to construction projects, that flow between all stakeholders including the Client, QA Team, Quantity surveyors, Contractors etc.

1. **Key differentiating factor and justification for the build project**

Taking the above factors into consideration, this software build project seeks to fill in gaps and address most of construction project related delays that arise from ineffective approval and reviewal processes. By providing essential collaborative functionality required in construction project management software via workflows and interfaces designed with emphasis on minimized complexity and effective notification features, we can ensure that all stakeholders are in the loop on all project events and progress.

With the current intended user demographic being stakeholders in low to medium-rise construction projects, the functional and non-functional requirements have been boiled down to features that cater to the specific needs of such. The barrier to entry for the use of tools for clients with relatively low computer literacy is mitigated with intuitive interfaces, simple tables, basic PDF reports and charts for Information analysis.

In the requirement gathering phase, the client for the project ( a chartered Quantity Surveyor) with extensive experience using cutting-edge project management software tools like Oracle Primavera and MS projects revealed that they lack adequate automation and notification features for various construction project-specific approvals, delay reporting, and request workflows.

This was one of the key points that led to the justification and decision to implement this build project with its overarching goal being to address delays and associated financial losses that are directly correlated to untimely approvals and acknowledgment of requests between stakeholders. The use of this software system is expected to have a positive, quantifiably demonstratable impact in the cost management of construction projects.

# 2. Analysis

1. **User Stories**

**User: Contractor**

Figure 2.1:User Stories: Contractor

|  |  |  |  |
| --- | --- | --- | --- |
| As a contractor, I want to be able to update Project Related Delays with notifications sent to stakeholders | As a contractor, I want to able to submit relevant documents and request approvals for payments from the consultant | As a contractor, I want to be able to request work inspection requests from the QA team so that I can get prompt approvals to progress with the project. | As a contractor, I want to be able to request material inspection requests from the QA team so that I can get prompt approvals to make the payments to suppliers and progress with the project swiftly. |
| As a contractor, I want to be able to create and update variation requests so that all changes are recorded and organized | As a contractor for the project, I want to be able to track the project status and other important information via graphs and charts in a dashboard so that I’m kept in the loop on all activities and events. | As a contractor, I want to be able to track and maintain a clear record of submitted photos related to all project tasks | As a contractor, I want to be able to update and track project status using a simple interface so that all stakeholders are made aware of project progress. |

**User: Consultants (Quantity Surveyor, Quality Assurance team, Architect)**

Figure 2.2:User Stories: Consultant

|  |  |  |  |
| --- | --- | --- | --- |
| As a quality assurance engineer, I want to able to promptly approve or comment on material inspection requests made by contractors so that the project may proceed with minimal delays | As a quality assurance engineer, I want to be able to promptly approve or comment on work inspection requests made by contractors so that the project may proceed with minimal delays | As a consultant for the project, I want to be able to view weekly and monthly reports on project status. | As a quantity surveyor I want to be able to promptly approve or comment on Interim payment applications made by the contractor and send out notifications to the client so that the payment is made promptly to avoid delays |
| As a quality assurance engineer, I want to able to promptly approve or comment on material inspection requests made by contractors so that the project may proceed with minimal delays | As a consultant for the project, I want to be able to track the project status and other important information via graphs and charts in a dashboard so that I’m kept in the loop on all activities and events. | As a consultant for the project, I want to be able to view all task-related photos | As a consultant, I want to be able to generate reports and have them shared with all stakeholders in the project |

**User: Client**

Figure 2.3:User Stories: Client

|  |  |  |  |
| --- | --- | --- | --- |
| As the client for the project, I want to be able to track the project status and other important information via graphs and charts in a dashboard so that I’m kept in the loop on all activities | As the client for the project, I want to be promptly notified of any Interim Payment approvals so that I can do the necessary payment transfers and avoid delays | As the client for the project, I want to be notified of all important project-related events via email and other notification methods. | As the client for the project, I want to be able to easily manage users on the application |

1. **Use Cases: Use Case Diagram**

The use cases enabled by the software system are illustrated in the below Use Case Diagram:

Figure 2.4:Use case diagram

Diagram

Description automatically generated

1. **Requirement Engineering**

Consists of the feasibility study and requirement elicitation process:

**Feasibility Study**

**1. Technical Feasibility:** The software solution has been deemed technically feasible using Python as the programming language with the Flask Application Framework due to its extensibility in using a range of libraries and plugins that enable a range of features. Maintainability of the software solution is ensured with Python’s ease of readability coupled with appropriate modularity.

**2. Operational Feasibility:** The software solution has been deemed operationally feasible due to the emphasis given on user-friendliness while reducing overall complexity in its use by providing the essential features required in construction project management. This eliminates any barrier to its use, especially for non-technical users. The solution is web-based and facilitates collaboration from any location in the world, thus ensuring accessibility.

**3. Economic Feasibility:** The cost of maintenance of the software solution is minimized to a great extent due to the Modularity and reduced complexity that the Flask Application framework provides. The use of costly third-party solutions for certain functionality is mitigated with the usage of open-source libraries available for Flask. Furthermore, Application hosting costs can be minimized to a great extent using Cloud providers Like Heroku and Azure that provide Application hosting as PaaS that charges according to usage patterns. PaaS services also mitigate the overall complexity of manually maintaining servers. Feature updates are cost-effective due to the availability of affordable, technically competent Python Developers with experience using the Flask Framework.

1. **Requirement Elicitation and Analysis**
2. Using an Agile-Scrum approach, requirement elicitation for this build project was done with regular liaison with the client. The functionalities required in the construction project management software were determined as user stories that clearly illustrated the specific user requirements and use cases.
3. Using the strategy of reverse brainstorming, firstly, the problems or user stories of the client were analyzed on a high level. Secondly, the features that address the stories were identified and prioritized for implementation. Thirdly, the correct approach to actualizing these features to achieve to desired functionality was determined. Fourthly and fifthly, the required functionality and their constituent components were identified.

An example for requirements determined for an identified problem is provided below:

Figure 2.5:Reverse brainstorming board

Timeline

Description automatically generated with low confidence

1. The above board also helps in identifying the specific functional requirements that will be included in a more detailed manner in the S.R.S.
2. **Analysis Outcomes**

**Points that were identified in the analysis process:**

1. The technical, operational, and economic feasibility was identified which led to the decision to proceed with the project to address the issues faced in the construction project management domain.

2. The level of complexity of the project as well as the scope for future maintenance was identified. The choice of software programming language, framework, and libraries was made accordingly.

3. User-friendliness is of paramount importance as the software solution will be used by clients of varying computer literacy. Hence, emphasis is paid to making the UI of the Web Application as intuitive and simple as possible.

4. With prompt responses to requests and approvals between stakeholders making a significant impact on the success of a construction project, the importance of effective notification systems was identified which led to the decision to implement email and SMS-based notifications.

5. Most Modules share the similar workflows, so emphasis is made on designing code to be reusable.

Overall, the analysis process was successful in satisfactorily revealing the key areas that the build project should be addressing.

# 3. Design

**Design Pattern**

The web application will be implemented using the MVC (Model,View,Controller) design pattern which increases code modularity and maintainability.

Figure 3. 1: MVC Architecture

Diagram

Description automatically generated

This design pattern logically isolates the source code and functionality across Business logic, Database queries, and Presentation. This increased modularity allows for developers to independently maintain the codebase.

## 3.1 Design Techniques

1. **Software Process model**

The Project will be decomposed into small-manageable modules with each module being individually designed, developed, tested, validated, and deployed with the client’s involvement. Every Increment will be developed in iterations within Sprints. This will allow for any required changes to be implemented and tested early. The **Scrum Methodology/Process Model** which is a combination of iterative and incremental development will be chosen for this project.

The Each sprint would consist of the following

1. Plan the Functionality to be implemented based on priority
2. Design the functionality
3. Implement the module
4. Unit Tests to be carried out for validation
5. The Feature is integrated with the rest of the application and is deployed in a testing environment for integration testing
6. The integrated feature is further tested and reviewed by the client
7. The feature is committed to the main GIT branch

Figure 3.1. 1: Agile-Scrum Methodology

Diagram

Description automatically generated

1. **Database Design: ERD Diagram**

The below ERD diagram depicts the entities that exist in the database for the application as well as their inter-relationships and attributes:

Figure 3.1. 2: Entity-relationship diagram

Diagram

Description automatically generated

1. **Activity Diagram: Interim payment application approval process**

The below Activity diagram is a behavioral diagram in UML that describes dynamic aspects of the system. In this case the workflow for an interim payment application is presented:

Figure 3.1. 3: Activity Diagram

Diagram

Description automatically generated

1. **Sequence Diagram: Work inspection request approval process**

The below sequence diagram depicts the interactions between objects arranged in time sequence.

In this case, the object calls made in a work inspection request submission is presented:

Figure 3.1. 4: Sequence Diagram

Diagram

Description automatically generated

1. **Flowchart: Project Delay EOT Approval Process**

The below Flowchart depicts the process of Delay and EOT management between the Contractor and Consultant:

Figure 3.1. 5: Flowchart

Diagram

Description automatically generated

## 3.2 System Overview

1. **System architecture**

The application is designed using a monolithic system architecture where the entire codebase, though split into several modules is run on a single VM. The system architecture of the web application includes a web server such as Apache or Nginx which accepts HTTP requests from clients, a WSGI such as Gunicorn that forwards requests to and from the Python- Flask application, and the on-disk SQLite database file that is managed by SQLAlchemy. (The webserver, WSGI, source code and database files all reside in the same VM)

Figure 3.2. 1Diagram

Description automatically generated: System Architecture Diagram

1. **Data Structures used in the code base**
2. Integer
3. String
4. Python Dictionary
5. Char
6. Python class objects
7. **Data Model**

The application uses the SQLAlchemy Object Relational Mapper (ORM) tool that translates Data Models which are written as Python classes into tables in relational SQL databases while also translating python function calls into SQL statements.

Figure 3.2. 2: Model implemented as a python class

Text

Description automatically generated

The basic flow of python objects and SQL statements that are processed and translated by the SQLAlchemy object relational mapper:

Diagram, polygon

Description automatically generatedFigure 3.2. 3: Object relational mapper flow

1. **Wireframes**

The initial Wireframe for the application dashboard:

Figure 3.2. 4Chart

Description automatically generated: Wireframe for the application dashboard

The initial Wireframe for the project delay management page:

Graphical user interface

Description automatically generatedFigure 3.2. 5: Wireframe for the Delay Management page

Initial version of the project management dashboard based on the wireframes:

Figure 3.2. 6Graphical user interface, website

Description automatically generated: Initial Version: Dashboard

The Latest iteration of the project management dashboard:

Figure 3.2. 7: Final version: Dashboard

Graphical user interface

Description automatically generated

# 4. Product Implementation

1. **Backend Programming Language and Framework**

A picture containing logo

Description automatically generated

The short learning curve and ease of maintenance, along with a wide range of libraries that simplify and abstract key functionality, made Python the choice for this build project.

Text

Description automatically generated

Flask, a micro web framework written in Python, is chosen as the backend framework for the project. Being easy to learn along with other significant benefits such as having high compatibility with the latest technologies and opensource libraries, leaving a smaller code base, high framework performance, and providing high scalability for simple web applications made this the ideal framework for the project. It also comes with an integrated debugger which significantly speeds up and simplifies unit tests.

1. **Frontend Programming Language and Framework**

A picture containing text, clipart

Description automatically generated Logo

Description automatically generated

The frontend code of the application will be written using HTML, CSS and JavaScript with styling and UI components provided by the Bootstrap Framework.

1. **Libraries used**

One of the key benefits of using the Flask framework is its functional extensibility with powerful libraries that are actively maintained by the open-source community. Below is the list of libraries used for this web application:

**SQLAlchemy**: Python SQL toolkit and Object Relational Mapper that provides the ability to interact with SQL-based databases while adapting to simple Python code.

**Flask**: Python microservice-based web application framework

**Flask-Login:** Provides user session management as well as authentication.

**BCrypt:** Provides functionality to encrypt and decrypt sensitive information such as user passwords that are stored in and accessed from databases.

**Email-validator:** Enables email validation in forms

**Flask-Mail:** Enables sending of emails via the SMTP protocol

**Flask-WT Forms:** Enables creation of forms and handling form data using python objects.

**DateTime:** Handles date/time formatting

**Jinja2:** Templating engine for python that allows data presentation via HTML formats

1. **Coding Principles**

**Documentation**

Each function has descriptive comments included which makes it easy for developers to maintain and read the code. For an example of well-commented code, see appendix A.

**Simplicity**

Python and the Flask framework provide built-in libraries and functions that enable powerful functionality which reduces the need to write complex functions from scratch. Variable names are written to be easily interpreted. The Jinja Templating engine allows for reusable HTML elements which reduce the overall complexity of maintaining multiple front-end HTML documents while also providing functionality to pass values from the controller layer to the view layer. For an example of the usage of library functions, see appendix B.

**Object-Oriented Programming**

Classes are created for objects like Database tables as well as forms. Each class has its attributes and methods that are called when an instance of the class is created. With Python being an object-oriented programming language, passing objects as parameters to functions is simplified and simplifies the writing of various functionality. For an example of the usage of OOP concepts, see appendix C.

**Modularity**

As the application uses the MVC design pattern, the code base is split into multiple python source code files. each performing its specific function. Forms.py (defining form classes), routes.py (Defining application URIs and backend functionality), models.py (defining database classes), \_\_init\_\_.py (loads most of the libraries and sets application configuration). The HTML, CSS, and JavaScript files are placed in a separate directory which can be accessed by the Python code using the Flask Framework. See appendix D for the file structure of the codebase.

**DRY (Don’t repeat yourself)**

"Don't repeat yourself" is a software development principle aimed at eliminating program pattern recurrence by employing abstractions or data standardization to minimize redundancy. This principle was applied throughout the codebase by eliminating repeated code and instead, creating separate functions that can be called by modules using a single line of code with the required parameters being passed.

1. **Critical discussion of coding issues**

**Libraries**

1. The Flask framework on its own lacks form validation, database abstraction, or any other built-in components and layers which makes application developers heavily dependent on pre-existing third-party libraries to enable such functionality.
2. While the use of third-party libraries in the codebase comes with the obvious benefits of rapid development and reduced code, it does introduce a few disadvantages that should not be overlooked.

Finally, if the library code is updated, for example, to fix a bug or include additional functionality, then the relevant updated libraries must then be re-downloaded with the source code of the application having to go through modification to refer to the new library versions that need to be imported.

1. **Functions implemented using third-party libraries**

A few examples of third-party library functions used throughout the codebase:

1. **Creating records using built-in SQLAlchemy functions:**

Functions used in the example: .session.add(record object)

.session.commit()

Figure 4. 1**A screenshot of a computer

Description automatically generated with medium confidence**: SQLAlchemy functions: Create record

1. **Deleting records using built-in SQLAlchemy functions:**

Functions used in the example: .session.delete(record object)

.session.commit()

Figure 4. 2**Text

Description automatically generated**: SQLAlchemy functions: Deleting records

1. **User authentication using built-in Flask-login functions:**

Functions used in the example: login\_user(user object)

**Text

Description automatically generated**Figure 4. 3: Flask-login functions: log in user

1. **User session termination using built-in Flask-login functions:**

Functions used in the example: logout\_user()

Text

Description automatically generatedFigure 4. 4: Flask-login functions: log out user

1. **Generating PDFs using built-in PDFKit functions:**

Functions used in the example: .from\_string(string,file location)

Figure 4. 5Text

Description automatically generated: PDFKit functions: generation a PDF file

1. **Downloading files using built-in Flask functions:**

Functions used in the example: send\_from\_directory (directory, path, as\_attachment)

Figure 4. 6Text

Description automatically generated:Flask functions: downloading files from local directory

1. **Hashing passwords using built-in Bcrypt functions:**

Functions used in the example:.generate\_password\_hash(plain text password)

.check\_password\_hash(self.password\_hash, password)

Figure 4. 7Text

Description automatically generated:BCrypt:Generating and validating password hashes

1. **Code Structure**

With the application based on the MVC architecture, the codebase was split/modularized into several source code files with each responsible for performing a specific function.

The below illustration of the source file structure demonstrates the decomposition of code into separate source code files/modules:

1. **models.py** (database-related code that forms the model layer)
2. **routes.py** (Business logic-related code that forms the controller layer)
3. **templates** (Contains HTML and CSS files that form the view layer):
4. **forms.py** (classes for forms that to collect user input)
5. **email\_send.py** (All functions responsible for sending email and SMS notifications)
6. **\_\_init\_\_.py** (Code for initializing the flask application and importing base libraries)

Figure 4. 8: Source code structure

Graphical user interface, application

Description automatically generated

# 5. Validation

1. **Testing Strategy**

Chart, sunburst chart

Description automatically generated

Figure 5. 1: Test Strategy

**Unit testing**

Testing starts with unit tests where each function is tested individually using the Flask Debugger and print statements to check function outputs. Defects revealed at this stage can be rectified early on which minimizes the complexity in debugging issues during the later testing phases.

**Integration Testing**

Here, we focus on the validating the design of the modules and functions and the calls that flow between them. For example, functions that implement business logic and the functions used for Database manipulation are subjected to integration testing to verify if data processed by the business logic layer is received and stored in the database by the relevant Database model layer.

**Validation Testing**

Validation tests take place with the client upon completing unit and integration tests for each module in the web application. This ensures that the requirements are well understood and translated into the relevant functionality. Black box testing will be performed here. The module is either marked as completed or requiring further refactoring at this stage, in which case, the module goes through another iteration to address concerns raised by the validation tests.

**System Testing**

The completed application will be subjected to system testing during which the Client will test all functionality that the web application provides in a cloud VM environment to represent an actual client deployment more accurately. The Overall usability, functionality, and accuracy of the application will be tested at this stage. Successful completion marks the application as ready for deployment in a production environment.

1. **Black Box Testing**

Two types of black box testing were performed on the system:

1. **Non-functional Testing**: Usability Testing
2. **Functional Testing**: Testing the Functions of a Module

**Usability testing**

Results of usability testing carried out for all modules:

Table 4: Usability Test Report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tester** | | Kirusanth Siva (Client) | | |
| **Test Description** | | Usability of Module Functions | | |
| **Test Case** | **Module** | **Usability Score (1-10)** | **Suggestions/Areas for Improvement** | **Outcome Result (PASS/FAIL)** |
| 1 | Task Management | 7 | A Slide bar for updating task percentage would be better than entering the number | PASS |
| 2 | Delay Management | 8 | Option to add additional emails to send reports to if needed would be great | PASS |
| 3 | Work inspections | 9 | Workflow is fine. Nothing much to change | PASS |
| 4 | Material Inspections | 9 | Workflow is fine. Nothing much to change | PASS |
| 5 | Payment Requests | 8 | A pop-up modal would be ideal to create new records | PASS |
| 6 | Variation Requests | 8 | A pop-up modal instead of a new screen would be ideal to create new records | PASS |
| 7 | User Management | 9 | Self-password resets for users would reduce the workload on the administrator. The rest works fine. | PASS |

**Functional Testing**

Results of functional testing carried out for the Task Management Module:

Table 5: Functional Test Report (Task Management Module)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tester** | | Kirusanth Siva (Client) | | |
| **Test Description** | | Task Management Module: Functional Tests | | |
| **Test Case** | **Action** | **Expected Outcome** | **Actual Outcome** | **Outcome Result (PASS/FAIL)** |
| 1 | Fill the Task creation form with valid information | Task should be created. Infographics and table are updated. | Task is created. Infographics and tables are updated. | PASS |
| 2 | Create a record with a task start date that is greater than the end date | Form Submission should fail | Form submission failed | PASS |
| 3 | Update Task Status | The Task status on the table should be updated | The Task status on the table is updated | PASS |
| 4 | Upload Task Image | Image should be displayed upon clicking the “View Photos” option | The image, image name and submitted date are displayed | PASS |
| 5 | Delete Task | The task record should be deleted from the table | The task record is deleted from the table | PASS |
| 6 | Email PDF report | Clicking on “Email report to stakeholders” should send an email to all stakeholders with the PDF report containing Task-related details | Emails with Generated PDFs were received by all stakeholders | PASS |

**Evidence:**

Functional testing: Test case 1

Figure 5. 2: Functional test case 1

**Diagram

Description automatically generated with medium confidence**

Functional testing: Test case 1 (passed)

Figure 5. 3Graphical user interface, application, website

Description automatically generated: Functional test case 1 (passed)

Functional testing: Test case 2

Figure 5. 4A picture containing diagram

Description automatically generated: Functional test case 2

Functional testing: Test case 2 (passed)

Figure 5. 5A picture containing graphical user interface

Description automatically generated: Functional test case 2 (passed)

Functional testing: Test case 3

Figure 5. 6Graphical user interface, application, website

Description automatically generated: Functional test case 3

Functional testing: Test case 3 (Passed)

Figure 5. 7Graphical user interface, application, website

Description automatically generated: Functional test case 3 (passed)

Functional testing: Test case 4

Figure 5. 8Graphical user interface, application

Description automatically generated: Functional test case 4

Functional testing: Test case 4 (Passed)

Figure 5. 9A picture containing text, screenshot

Description automatically generated: Functional test case 4 (passed)

Functional testing: Test case 5

Figure 5. 10Graphical user interface, application

Description automatically generated: Functional test case 5

Functional testing: Test case 5 (Passed)

Figure 5. 11Graphical user interface, application, website

Description automatically generated: Functional test case 5 (passed)

Functional testing: Test case 6

Figure 5. 12Graphical user interface, website

Description automatically generated: Functional test case 6

Functional testing: Test case 6 (Passed)

Figure 5. 13Graphical user interface, application, website

Description automatically generatedGraphical user interface, text, application

Description automatically generated: Functional test case 6 (passed)

1. **Client Feedback**

Table 6: Client Feedback Report

| **Feedback: System Testing** | |
| --- | --- |
| **User Experience** | |
| Comments/performance examples  The interface is intuitive and very easy to grasp. It took me minutes to learn how to perform most of the functions on the system. The overall design could improve a bit. | Rating:  [ ] Needs improvement  [ ] Satisfactory  [] Good  [\*] Excellent |
| **Functionality** | |
| Comments/performance examples  The functionality is effective in achieving the required outcomes. Additional functionality that can help in making predictions on delays based on weather patterns and various other metrics would make a significant positive impact on managing delays. | Rating:  [ ] Needs improvement  [ ] Satisfactory  [\*] Good  [ ] Excellent |
| **Scope** | |
| Comments/performance examples  The web application satisfactorily covers all the requirements that were defined under the scope of the project. | Rating:  [ ] Needs improvement  [ ] Satisfactory  [ ] Good  [\*] Excellent |
| **Data Analytics/ Reporting** | |
| Comments/performance examples  The current version of the PDF reports provides a good overview of information that needs to be communicated to stakeholders. There is room for enhancements in the PDF reports. Aside from improving the overall appearance, infographics and additional information could be included. | Rating:  [ ] Needs improvement  [\*] Satisfactory  [ ] Good  [ ] Excellent |
| I hereby state that the feedback provided above, which was determined upon testing and usage of the web application to the best of my ability, is my own and that it accurately reflects the state of the application at the point that it was tested. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Kirushanth Siva BSc(Hons) QS, MAIQS, CQS (Client for the build project) |

# 6. Critical Review & Conclusion

## 6.1 Closing executive summary

Upon completing the build project, a few points for the future maintenance of the software solution were identified:

**Database Management**

Initially, the database management system was planned to be implemented using a DB server such as a Microsoft SQL Server in conjunction with Flask’s SQL Alchemy ORM functionality.

Upon researching with the client and identifying the average amount of CRUD operations occurring in construction project management being relatively low compared to most applications, DB performance was deemed a low-priority quality attribute. Hence, the current iteration of the application will leverage an SQLite DB file with SQL Alchemy for database operations. Data Backups are simplified using this approach as scheduled cronjobs can be used to periodically make encrypted copies/backups of the DB file. Thus, assuring data availability and security.

However, this approach poses a problem when implementing a highly available architecture with multiple VM’s for redundancy as the database files are accessed by the application from within the same VM. Future releases should focus on implementing application connectivity with an external Database server as planned initially.

**Deployment and hosting**

The application is currently designed to run on cloud VMs although containerization of the application using docker was tested successfully. Containerized instances of the application would require an external database as the database records would not persistent upon container restart.

Containerization would enable the application to be hosted using PaaS cloud solutions such as Azure Container instances or Azure App services which abstract and manage the underlying infrastructure required to run the application, thus removing the infrastructure management overhead, allowing developers to focus on improving and optimizing the codebase, not on managing servers. Future releases should focus on implementing application connectivity with an external Database server.

**Summary**

Aside from the two points raised above , the remaining functional and non-functional requirements for the project have been successfully addressed by the build project.

## 6.2 Conclusion

In conclusion, the aims and objectives of the build project which revolve around simplifying construction project management workflows in low to medium-rise construction projects have been successfully achieved with the implemented system having been subjected to various tests and receiving client validation.

The key strengths of the project as identified from the onset, which lie in its simplicity of use and emphasis on implementing simple workflows for functionality that adds the most value to construction projects of low to medium-rise scopes has been maintained.

The Scrum-based approach of requirement elicitation and continuous sprint-based module implementation and validation in liaison with the client has been effective and time-efficient. Key decisions such as overall interface design and process workflows have been implemented as increments and have been validated early in the project. This significantly minimized change requests.

While the client put forward suggestions for additional functionality and enhancements that were outside the initial scope, they were subjected to scrutiny and analysis to determine the justification of effort vs the business value added to the end-user. While not all suggestions were deemed feasible, they were recorded as possible implementations in future releases. This greatly prevented scope creep and helped in completing the build project within the given deadline.

This build project was instrumental in understanding the end-to-end process behind implementing a software solution, from the requirement elicitation and analysis phase that revealed the effective methods to gather requirements, to the designing phase where the requirements were translated into functional specifications with the help of diagrams, to the implementation of the software which revealed best coding practices, to finally validation, which revealed the process behind assuring the quality of software builds through various techniques.

See appendix E and F for Supervisor log sheets and feedback emails respectively.

# References / Bibliography

Armstrong, G. and Gilge, C., 2016. Building a technology advantage - Global Construction Survey 2016. [ebook] KPMG, p.18. Available at: <https://assets.kpmg/content/dam/kpmg/xx/pdf/2016/09/global-construction-survey-2016.pdf>[Accessed 10 November 2021].

Trauner, T., 2009. Construction delays. 2nd ed. Amsterdam: Butterworth-Heinemann, p.266.

Kesavan, M., Gobidan, N. and Dissanayake, P., 2015. Resource Planning to Reduce Project Delays - Sri Lankan Construction Industries. International Journal of Conceptions on Mechanical and Civil Engineering, [online] 3(2), pp.7-11. Available at: <https://www.academia.edu/28359721/Resource\_Planning\_to\_Reduce\_Project\_Delays\_Sri\_Lankan\_Construction\_Industries\_-\_International\_Journal\_of\_Conceptions\_on\_Mechanical\_and\_Civil\_Engineering\_IJCMCE\_Vol.\_3\_Issue.\_2\_August\_2015\_ISSN\_2357-2760\_pp\_7-12\_> [Accessed 5 November 2021].

PAT RESEARCH: B2B Reviews, Buying Guides & Best Practices. 2022. Top 19 Construction Project Management Software in 2022 - Reviews, Features, Pricing, Comparison - PAT RESEARCH: B2B Reviews, Buying Guides & Best Practices. [online] Available at: <https://www.predictiveanalyticstoday.com/top-construction-project-management-software/#:~:text=Project%20Management%20Software-,GanttPRO%2C%20BIM%20360%2C%20Acculynx%2C%20Jonas%20Premier%2C%20Procore%2C,Top%20Construction%20Project%20Management%20Software.> [Accessed 19 January 2022].

# Appendices

*Appendix A*

*Text

Description automatically generated with medium confidence*

*Appendix B*

Text

Description automatically generated

*Appendix C*

**A screenshot of a computer

Description automatically generated with medium confidence**

*Appendix D*

**Graphical user interface, diagram

Description automatically generated**

*Appendix E*

Table

Description automatically generated

*Appendix F*

*Graphical user interface, text

Description automatically generated*