**FACULTY OF SCIENCE, ENGINEERING AND COMPUTING**

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

**BSc DEGREE**

**IN**

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

**PROJECT INTERIM REPORT**

Name: Tuan Saad Ousman

ID Number: K2168332

Project Title: Construction Project Management System

Project Type: Build

Date: 16/02/2022

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 interim report provides information on the construction project management system build project that is currently underway. 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 main 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 regardless of their location.

With a clear central source of truth and communication regarding the Construction project via the Web application, the chances of vital notifications and information being missed 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 by the contractor as well as update any required changes that will then be agreed upon by both parties.

Automated and manual report generation will help inform the stakeholders on the status of completed project tasks, pending approvals, unexpected and expected delays, etc. Digitization of reports and document approvals will reduce the need for paperwork that could 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 SCRUM development methodology.

Emphasis will be made on making the front-end interface as intuitive and user-friendly as possible for the software to be usable by clients with limited computer literacy.

The system will be designed and implemented using HTML, CSS, JavaScript, Python using Python’s Flask Framework for the backend and Bootstrap 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 1](#_Toc95922743)

[1.1 Introduction 1](#_Toc95922744)

[1.2 Background and Motivation 2](#_Toc95922745)

[1.3 Problem in brief 4](#_Toc95922746)

[1.4 Aim & Objectives 5](#_Toc95922747)

[1.4.1 Aim 5](#_Toc95922748)

[1.4.2 Objectives 5](#_Toc95922749)

[1.5 Scope 6](#_Toc95922750)

[1.6 Deliverables 7](#_Toc95922751)

[1.7 Literature Review 8](#_Toc95922752)

[2. Analysis 9](#_Toc95922753)

[3. Design 14](#_Toc95922754)

[3.1 Design Techniques 15](#_Toc95922755)

[3.2 System Overview 18](#_Toc95922756)

[4. Product Implementation 21](#_Toc95922757)

[5. Proof of concept 28](#_Toc95922758)

[6. Validation 30](#_Toc95922759)

[7. Critical Review & Conclusion 31](#_Toc95922760)

[7.1 Closing executive summary 31](#_Toc95922761)

[7.2 Conclusion 32](#_Toc95922762)

[References / Bibliography 33](#_Toc95922763)

[Appendices 34](#_Toc95922764)

**List of Figures/Tables**

[Figure 1: User stories of contractors 9](#_Toc95828241)

[Figure 2: User Stories of consultants 9](#_Toc95828242)

[Figure 3: User Stories of the Client 9](#_Toc95828243)

[Figure 4: Use Case Diagram 10](#_Toc95828244)

[Figure 5: Reverse brainstorming board 12](#_Toc95828245)

[Figure 6: MVC Architecture 14](#_Toc95828246)

[Figure 7: Code file structure 14](#_Toc95828247)

[Figure 8: Scrum Methodology 15](#_Toc95828248)

[Figure 9: Entity relationship Diagram 16](#_Toc95828249)

[Figure 10: Activity Diagram 17](#_Toc95828250)

[Figure 11: System Architecture Diagram 18](#_Toc95828251)

[Figure 12: Model implemented as a python class 19](#_Toc95828252)

[Figure 13: Object relational mapper flow 19](#_Toc95828253)

[Figure 14: Wireframe for the application dashboard 20](#_Toc95828254)

[Figure 15: Wireframe for the delay management page 20](#_Toc95828255)

[Figure 16: Database query declaration using python and SQLAlchemy 23](#_Toc95828256)

[Figure 17: Structure of a database query 23](#_Toc95828257)

[Figure 18: Returning variables to the view layer 23](#_Toc95828258)

[Figure 19: Committing changes to database tables 24](#_Toc95828259)

[Figure 20: SQLAlchemy’s commit function 24](#_Toc95828260)

[Figure 21: Deleting a database record 24](#_Toc95828261)

[Figure 22: Authentication module 25](#_Toc95828262)

[Figure 23: Decorator used check user sessions 25](#_Toc95828263)

[Figure 24: Modifying Delay records 28](#_Toc95828264)

[Figure 25: Creating new Delay records using a Modal 28](#_Toc95828265)

[Figure 26: Project Management dashboard 29](#_Toc95828266)

[Figure 27: Email notification feature in action 29](#_Toc95828267)

[Figure 28: Testing Strategy diagram 30](#_Toc95828268)

[Table 1: Excerpt from the questionnaire survey (Kesavan et al. 2015) 2](#_Toc95828318)

[Table 2: SWOT Analysis 6](#_Toc95828319)

**Glossary of Terms**

**Cloud:** the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user.

**CSS:** a style sheet language used for describing the presentation of a document written in a markup language such as HTML.

**ERD:** a graphical representation that depicts relationships among objects within a database

**Framework:** A software library that provides a fundamental structure to support the development of applications for a specific environment.

**HTML:** The standard markup language for documents designed to be displayed in a web browser.

**Library:** A suite of data and programming code that is used to develop software programs and applications.

**Modularity:** A software design technique that emphasizes separating the functionality of a program into independent, interchangeable modules.

**Module:** A partition or logical separation of system code or design

**Object:** An abstract data type with the addition of polymorphism and inheritance

**PaaS:** Platform as a service (PaaS) is a complete development and deployment environment in the cloud.

**Plugin:** A software component that adds a specific feature to an existing computer program.

# Introduction & Literature Review

## 1.1 Introduction

Managing a construction project and its related activities at any scale demands a great deal of effort put in during its lifecycle. Due to the many concurrent activities like material purchasing, shop drawing approvals, change requests, etc. that take place during a project, the complexity associated with the management and tracking of these tasks increases the demand for systems and processes to for their simplification.

One of the keys to increasing the odds of a project meeting its budget and time deadlines successfully is effective communication and collaboration between the Stakeholders including the client, contractor, and consultant involved in a project.

This is where this 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 in a construction project can be relieved of the burden of manually keeping track of approvals, requests, project progress, payment requests, etc., thus reducing the related delays as well.

With a clear source of truth and real-time notifications, the accountability of stakeholders is made clear as the tasks required to be carried out from their ends are well defined. 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 stakeholders.

## 1.2 Background and Motivation

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 and user-friendly for stakeholders in a construction project especially in low-rise to medium-rise projects thus contributing to the reduction in the 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, shop drawings are commonly issued and exchanged via email and printed documents due to not adopting proper communication and issue tracking systems.

This can lead to an exchange of many emails that can be cumbersome to organize while having the potential to be missed. Increased paperwork can be cumbersome to maintain.

Furthermore, transparency in 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 financial losses that accompany them.

## 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 perform a critical study of 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.

## 1.5 Scope

**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.

**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.

**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 which benefits all stakeholders financially because of improved controlling of expenses.

**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 value immediately.

**Constraints***:* Given the timeline for the completion of the project, more emphasis is given on implementing the key functionalities that will make the most impact on the end-user. The client’s limited exposure to automated project management tools adds significant effort in testing and validating functionalities and userinterfaces*.*

**SWOT Analysis**

|  |  |
| --- | --- |
| Strengths   * User-friendliness: Increased usability for non-technical users * Reports and Updates are automated and easy to interpret * Cloud-based and accessible from anywhere in the world | Weaknesses   * Functionality is limited to address**ing** 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 an increased need for accessibility and automated processes that eliminate manual document handling and signature verification | Threats   * Current established software solutions can integrate similar functionality. * Insufficient data protection practices can lead to liability |

Table 2: SWOT Analysis

## 1.6 Deliverables

* Final Working product – The Construction Project Management Web Application
* Final Documentation – SRS, System Design, Project Plan, System Architecture Diagram etc.
* System Test Logs - Reports of the testing will be included in the system test log. This will include unit testing, user acceptance testing, etc.
* User Manual – Instructions on the usage of the software system, any troubleshooting steps, user restrictions, regulations of usage, and rules.
* Training Guide – Training Videos will be created to define how new users can perform key tasks.

## 1.7 Literature Review

**Appraisal of existing Construction project Management Tools**

Upon research into the features and functionalities of popular construction management tools in the market such as GanttPRO, AccuLynx, 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 the needs of Project Tracking and Project Management were adequately met with current software solutions that cater to most markets. Other Popular construction project management tools such as Oracle Primavera come with advanced collaborative features and are widely used in large enterprise projects and require additional expertise for their management and maintenance which increases the cost of operation significantly.

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.

**Key differentiating factor and motivation behind the build project**

Taking the above factors into consideration, this software build project seeks to 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, 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 of 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 and thus making a positive, demonstratable impact in the cost management of construction projects.

# 2. Analysis

**User Stories**

***Contractor***

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.

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.

As a contractor, I want to be able to track and update variation requests made by the client so that all changes are recorded and organized

Figure 1: User stories of contractors

As a contractor, I want to be able to track and maintain a clear record of submitted construction drawings and shop drawings and their approvals.

As a contractor, I want to be able to track and update variation requests made by the client so that all changes are recorded and organized

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 able to submit relevant documents and request approvals for Interim payments from the client

As a contractor, I want to be able to update Project Related Delays so that all stakeholders are made aware of project status

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

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 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 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 project architect, I want to be able to issue construction drawings to the contractor and approve or comment on any shop drawings that are sent back from the contractor so that the project commences with clear requirements

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

Figure 2: User Stories of consultants

As a consultant for the project, I want to be able to view weekly and monthly reports on project status.

***Client***

As the client for the project, I want to be able to view weekly and monthly reports on project status

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 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 able to track the project status and other important information via a graphs and charts 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 and events.a dashboard so that I’m kept in the loop on all activities and events.

Figure 3: User Stories of the Client

**Use Cases**

Diagram

Description automatically generated

Figure 4: Use Case Diagram

**Requirement Engineering**

**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 essential features. The Maintainability aspect of the software solution is satisfied 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 thus allows for collaboration from any location in the world, thus always 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.

**Requirement Elicitation and Analysis**

Using a Scrum approach, requirement elicitation for this build project was done primarily via regular one-on-one interviews with the client. The functionalities required in the construction project management software were determined as user stories that clearly illustrated the specific user requirements. (Examples can be found in the first section of this chapter)

Using the strategy of reverse brainstorming, firstly, the User Stories were analyzed on a high level. Secondly, the constituent parts that address the stories were scrutinized and prioritized based on their impact on the problem domain. Thirdly, the correct approach to actualizing these parts was determined while boiling down the key components to be implemented for the desired functionality. An example for requirements determined for a problem addressed by the requirement elicitation phase is provided below:

Timeline

Description automatically generated with low confidence

Figure 5: Reverse brainstorming board

The above board helps in identifying the specific functional requirements that will be included in a more detailed manner in the S.R.S.

**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 based on this.

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.

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

# 3. Design

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

Diagram

Description automatically generated

Figure 6: MVC Architecture

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

The below illustration of the source file structure demonstrates the decomposition of code into separate modules such as models (database-related code that forms the model layer), routes (Business logic-related code that forms the controller layer), templates (HTML and CSS files that form the view layer):

Graphical user interface, application

Description automatically generated

Figure 7: Code file structure

## 3.1 Design Techniques

**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 as a Sprint. 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 Major Phases of the project using the Scrum Process Models include:

1. Requirement gathering
2. Planning
3. Sprint Implementation
4. System Testing
5. User Acceptance Testing

Diagram

Description automatically generated

Figure 8: Scrum Methodology

**Database Design: ERD Diagram**

Diagram

Description automatically generated

Figure 9: Entity relationship Diagram

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

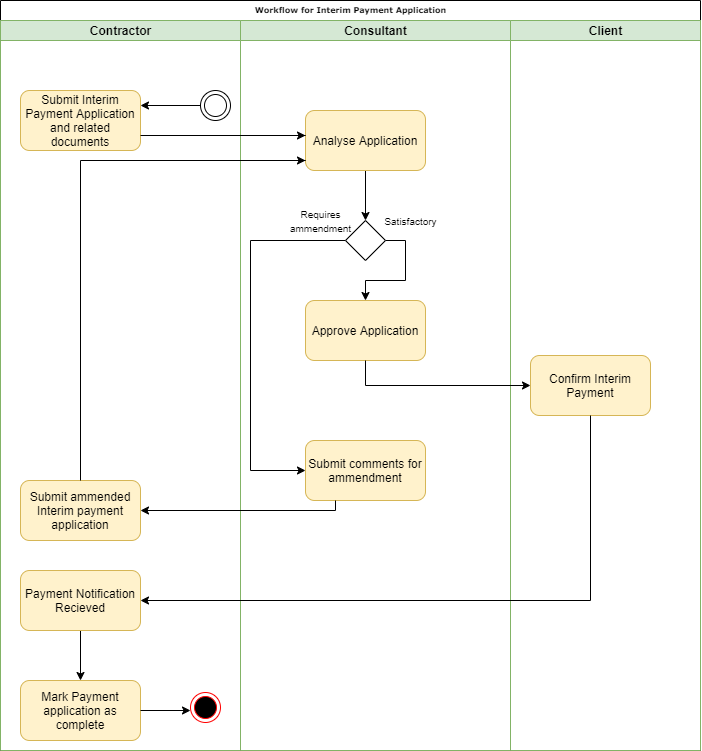


Figure 10: Activity Diagram

## 3.2 System Overview

**System architecture**

The application is designed using a monolithic architecture where the entire codebase, though split into several modules is run on a single VM or container. 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.

Diagram

Description automatically generated

Figure 11: System Architecture Diagram

**Data Structures used in the code base**

1. Integer
2. String
3. Python Dictionary

**Data Model**

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

Text

Description automatically generated

Figure 12: Model implemented as a python class

Figure 13 below depicts the basic flow of python objects and SQL statements that are processed and translated by an object relational mapper.

Diagram, polygon

Description automatically generated

Figure 13: Object relational mapper flow

**Wireframes**

The initial Wireframe for the application dashboard is shown in figure 14 below.

Chart

Description automatically generated

Figure 14: Wireframe for the application dashboard

The initial Wireframe for the project delay management page is shown in figure 14 below.

Graphical user interface

Description automatically generated

Figure 15: Wireframe for the delay management page

# 4. Product Implementation

**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 that the Python programming language provides, made it the choice for this build project with importance given to maintainability and readability.

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 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.

**Frontend Programming Languages 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 Frontend Framework.

**Libraries**

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

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

**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:** Validate emails during sign up

**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

**Functions Used**

With the core functionality of the web application revolving around information storage and retrieval, emphasis was given to the reusability of CRUD functions that take in user input, perform the necessary database queries as well retrieve data to be pushed to the view layer when required.

**Data Retrieval and Presentation**

Text

Description automatically generated

Figure 16: Database query declaration using python and SQLAlchemy

The above homepage() function provides an example of the data query functionality that the SQLAlchemy library provides. Instead of using standard SQL to perform queries, SQLAlchemy’s Object-relational mapper feature allows for inbuilt pythonic functions to be used instead. These functions are then mapped to relevant SQL queries by SQLAlchemy and abstract much of the complexity in embedding SQL within python code.

Diagram

Description automatically generated

Figure 17: Structure of a database query

Once the variables are initialized with the necessary data from the database query, we “render” or return an HTML page to the user’s browser using the below line of code:



Figure 18: Returning variables to the view layer

This returns, the ‘home.html’ page to the user’s browser after the variables, eg: pending\_delays, approved\_delays etc. have been passed to the page(View Layer) using the jinja templating engine.

**Data modification/Inserting**

Database entries are created and modified with the use of standard SQLAlchemy functions.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 19: Committing changes to database tables

In the above code block, the *delay\_to\_create* variable is initialized with the *Delay* Database record object which is instantiated using values returned from user-entered form values from the front-end form functionality that Flask-Forms provides.

The record is inserted into the database table with two simple lines of code:

db.session.add(delay\_to\_create)

db.session.commit()

Figure 20: SQLAlchemy’s commit function

**Data Deletion**

Database entries are Deleted with the use of standard SQLAlchemy functions.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 21: Deleting a database record

In the above *deleteTask(id)* function, the *id* parameter is passed to the function from a front table. The *task\_to\_delete*  variable is initialized using the DB object/record that is generated by a DB query using the *id*  parameter against the *Tasks* table.

**Authentication/User-session creation**

User authentication and session are enabled using the Flask-Login library which enables session cookie insertion and validation throughout the use of the application.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 22: Authentication module

The login\_page() function above takes the form object that is sent to the backend via a POST request. The validate\_on\_submit() function that is provided by the Flask-form library validates the form data using the Email Validator library. The password is validated using the check\_password\_correction() Function that is defined in the model layer. If the user exists and the password validation function returns a True value, then the attempted\_user object queried from the User is passed to the login\_user() function provided by the Flask-Login and a user session and browser cookie are generated. Each page of the web application checks for the user session. This session validation is enabled by using the @login\_required decorator for every route or Web application URI.

Text

Description automatically generated

Figure 23: Decorator used to check user sessions

**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 given with clarity in mind and are easy to interpret. The Jinja Templating engine allows for reusable HTML elements to reduce the overall complexity of maintaining multiple front-end HTML documents while providing functionality to pass values from the controller layer to the view layer with ease. 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 makes it easy to write functionality that enables CRUD operations. 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.

**Critical discussion of coding issues**

**Libraries**

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.

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.

Firstly, 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.

# 5. Proof of concept

Early versions of modules of the application are being implemented and iterated upon feedback from the client. This section includes illustrations of some of the modules that have been implemented thus far.

**Page for maintaining project delays**

The below figure 24 shows the actions that users can take to modify a delay record via a dropdown button.

Graphical user interface, website

Description automatically generated

Figure 24: Modifying Delay records

The below figure 25 shows the modal that users interact with to create new delay records.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 25: Creating new Delay records using a Modal

**Initial version of the project management dashboard**

The below figure 25 showcases the initial basic version of the project management dashboard. Additional components such as Gannt charts to track project tasks will be included in later iterations.

Graphical user interface, website

Description automatically generated

Figure 26: Project Management dashboard

**Email Notifications**

The below figure 27 shows an email notification that was generated by the application upon the deletion of a delay record.

Graphical user interface

Description automatically generated

Figure 27:Demonstration of the Email notification feature

# 6. Validation

**Testing Strategy**

Chart, sunburst chart

Description automatically generated

Figure 28: Testing Strategy diagram

**Unit testing**

Testing starts with unit tests where each module is tested individually using the Flask Debugger and PyTest testing framework. This can be considered white box testing. Defects revealed at this stage can be rectified early on which minimizes the complexity in debugging during the later testing phases.

**Integration Testing**

Here, we focus on 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 manipulation layer.

**Validation Testing**

Since the project is implemented using the Scrum methodology, 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 were well understood and translated into the relevant functionality. 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 application will be subject to system testing which will include testing all functionality that the web application provides as well as stress to determine how the systems cope with the simulated stress that can be expected in an actual production environment.

# 7. Critical Review & Conclusion

## 7.1 Closing executive summary

Upon reviewing the progress made with the build project thus far, a few points were brought to light that indicated a few required adjustments in the overall execution of the project:

**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 to improve database query performance. Upon researching with the client and identifying that 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 application will leverage a 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 of the SQLite DB file. Thus, assuring data availability and security.

**Deployment and hosting**

The application is designed to run on cloud VMs, but upon considering the complexity in manually pushing feature updates and enhancements, containerization of the application using docker images was considered and tested successfully. This enables 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.

**Project management approach**

Thus far, 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, 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. The need for an effective change management system was identified for potential functionality that needs to be integrated into the system in future releases. As such, necessary change management documentation procedures and tracking systems have been implemented.

## 7.2 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 thus far not deviated from what was originally determined. The Scrum software development process has proven to be effective and has resulted in the successful implementation of functionality and workflows addressing client requirements including intuitive user interfaces and simple workflows that facilitate a gradual learning curve which is of utmost priority.

The functionalities/modules implemented so far, including delay and project task tracking, have been tested and validated by the client. The required notification channels including email-based alerts have also been tested and deemed satisfactory.

The key strength of the project as identified from the onset lies in its simplicity of use and emphasis on including just required functionality that adds the most value to low to medium-rise construction projects. The backend code is written purely in Python makes the codebase highly maintainable, easy to read, and the wide range of available third-party plugins increases the extensibility of the application’s functionality while minimizing the amount of code to be written. With the client being a highly qualified subject matter expert in all areas of the construction project management process, the requirement elicitation and engineering process were carried out effectively with minimal ambiguity on both functional and non-functional requirements, thus ensuring that the scope of the project is limited to that which adds the most operational and business value to the end-users.

Limitations of the project currently include the use of limited aesthetics in front-end design with only basic front-end frameworks and components being used. Also, currently, the Flask-login library provides secure authentication and user-session management mechanisms and although this solution adequately fulfills the security needs of the application, consideration should be given to security hardening in future iterations of the application by implementing more secure and robust third-party SaaS solutions such as Auth0 or Okta although this is currently beyond the scope of the build project and is a proposal for future work.

Other proposals for future work include enabling advanced data analytics that could provide actionable insights to all stakeholders in a construction project as well as extending access to the functionalities of the web application via an integrated mobile application.

# 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**