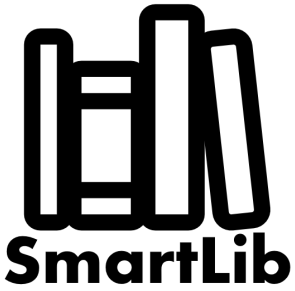


PROJECT PLAN

Version *1.6*
23/2/2021

Team Xeon

Document Number: SL-05

REVISION HISTORY

Version	Date	Author	Description of Changes	Approved by
1.0	16/02/2021	All project members	Outline of Project Plan	Kenny Voo Tze Rung
1.1	17/02/2021	Kenny Voo Tze Rung	Software Development Lifecycle and Process Definition added Work Breakdown Structure added	Kenny Voo Tze Rung
1.2	20/02/2021	Kenny Voo Tze Rung	Monitoring & Reporting Mechanism added	Kenny Voo Tze Rung
1.3	21/02/2021	Kong Hou Jing	Introduction added	Kenny Voo Tze Rung
1.4	22/02/2021	Zeyu	Project Estimates added Produce Checklist added Best Practice Checklist added	Kenny Voo Tze Rung
1.5	22/02/2021	Wilson, Boon Shuan	Schedule added Risk Management added Quality Assurance added Monitoring & Control Added	Kenny Voo Tze Rung
1.6	23/02/2021	Irsyad	Activity Dependencies added Work Package Details added	Kenny Voo Tze Rung

Table of Contents

1 Introduction **3**

1.1 Project Overview 3

1.2 Project Description and Scope 3

2 Project Organization **4**

2.1 Team Structure 4

2.2 Roles and Responsibilities 4

2.3 Team Communication 5

3 Process Definition **6**

3.1 Lifecycle Model 6

4 Schedule **7**

4.1 Activity Dependencies and Schedule 7

4.2 Work Breakdown Structure 8

4.3 Work Packages 9

4.4 Activity Dependencies 9

4.5 Work Package Details 11

5 Project Estimates **16**

5.1 Code Size Estimation using Function Points 16

5.1.1 Unadjusted Function Points 16

5.1.2 Adjusted Function Points 17

5.1.3 Lines of Code 18

5.2 Efforts, Duration and Team Size Estimation 18

5.2.1 Distribution of Effort 19

5.3 Cost Estimates 20

6 Product Checklist **21**

7 Best Practice Checklist **22**

8 Risk Management **23**

9 Quality Assurance **25**

10 Monitoring & Control **27**

1 Introduction

1.1 Project Overview

SmartLib is an intelligent system aimed to improve the experience at the libraries across Nanyang Technological University. Its main purpose is to address the issue of locating empty seats/booths at the libraries as well as mitigating seat hogging. SmartLib enables library users, mainly students, to check and view the availability of individual seats in the library. In addition, students will be able to reserve a seat for a short period of time before walking over to the library. This ensures that they can secure a seat for them to do their work and avoid unnecessary disappointment.

1.2 Project Description and Scope

SmartLib is a library seat monitoring and reservation system using human detection. It is based on the Internet Of Things model where our cameras will send the necessary data to our cloud for data processing, feeding back the necessary output to our web interface which users interact with.. Cameras will be strategically placed in the library to capture images of the seats, which will be mapped to the virtual seats in our system. These images are captured at a defined rate and will be sent to the back-end server for image processing and human detection. The availability of these seats will be determined by processing the collected data via machine learning and their status will be sent to the database through the internet.

The target audience for this system will be mainly students and the librarians. Students will be able to access our system through our web interface using any device with a web browser as well as the kiosks located in the library. They can view the status of the seats without logging in but may only reserve them once they have logged in to the system using their student account. Seats will be reserved for a short period of time, i.e. 15 mins, to allow sufficient time for the students to head over to the reserved seats. The purpose of the reservation is to eliminate the risks of multiple people heading over to the same seat, leading to disappointment and time wastage.

Librarians will be using an administrator account to access their librarian web interface which includes administrator features. Librarians will be notified if any seats are left unattended (hogged seats) for a prolonged period of time, i.e. 60 minutes. They will then be able to verify and take necessary actions, for example confiscation of belongings to free up space for other users during peak periods. Librarians will also be able to add/ remove library seats, as well as overriding the status of the seats for maintenance or in special events, such as spacing out seats due to the need of social distancing in the event of a pandemic outbreak.

The design objective are listed as:

(1) Seat occupancy detection using camera
(2) Check availability of seats
(3) Reserve a seat
(4) Seat management
(5) Alert the librarian
(6) Data analysis

2 Project Organization

2.1 Team Structure

The following is the list of executive roles, as required by CMM level 3.

- Project Manager: Kenny Voo Tze Rung
- Lead Developer: Wilson Tai
- Software Engineering Team: Kong Hou Jing , Irsyad , Zeyu
- Quality Assurance Manager: Boon Shuan
- Quality Assurance Engineer: Wilson Tai
- Release Engineer: Boon Shuan

2.2 Roles and Responsibilities

Project Manager: Kenny Voo

- Oversees project progress
- Approves and executes project plan
- Assigns tasks and reports status of project to team members
- Manages and motivates team members
- Represents the team to the outside world

Lead Developer: Wilson Tai

- Overall Technical Lead
- Responsible for Product Release
- Coordinate team's schedule
- Ensure Effective Communication between team members

Front End Developer: Irsyad, Hou Jing

- Implement the visual elements of the final product
- Ensure the technical feasibility of the UI/UX designs
- Optimize application for maximum speed and scalability
- Assure all user input is validated before sending it to the backend

Back-End Developer: Zeyu

- Responsible for Server-side application logic
- Integrate their work with the front-end side
- Design and implementation of data storage solution

QA Manager: Boon Shuan

- Oversee the overall product and process quality
- Recording, analysing and distributing statistical information

- Supervising QA engineer

QA Engineer: Wilson Tai

- Reviewing quality specifications and technical design documents to provide timely and meaningful feedback
- Create detailed and comprehensive test-cases
- Coordinating quality test activities

Release Engineer: Boon Shuan

- Create baselines, build and integrate changes for delivery
- Manage release of product
- Maintain and monitor software builds

2.3 Team Communication

Team Xeon communication channels are as followed:

- Weekly online meetings (Zoom)
- Whatsapp group
- Collaborative workspace (Trello)

3 Process Definition

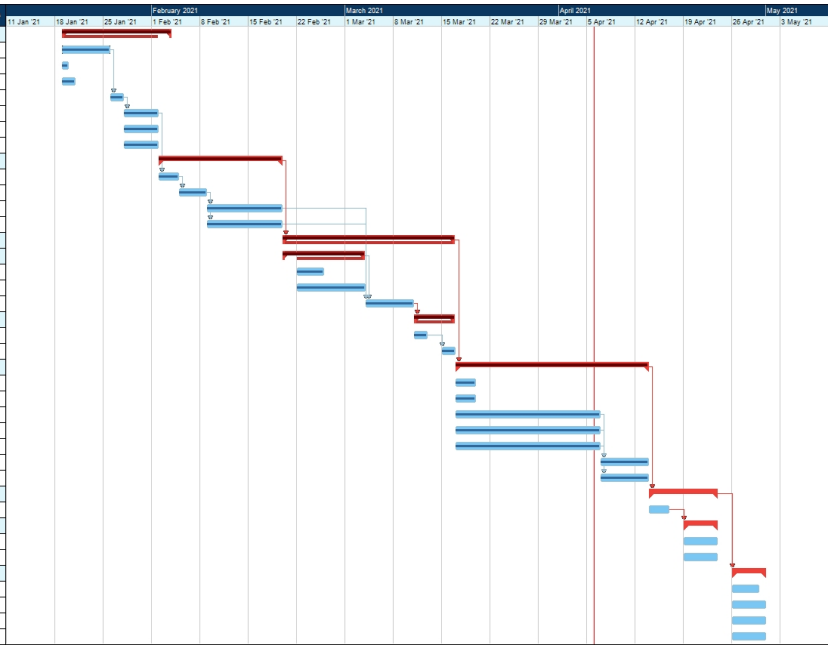
3.1 Lifecycle Model

Team Xeon will be implementing the Agile methodology in this project. Due to the short timeline, Agile will be a better choice than the traditional lifecycle model as it provides the team greater flexibility if there's any requirement changes by the stakeholder. It allows the team to develop the product incrementally and revise to improve the product based on the feedback gathered. This will reduce the risk of the team of failing to deliver the project before the deadline. The duration of this project will last for 3 months starting from 19th of January 2021 to 30th April 2021.

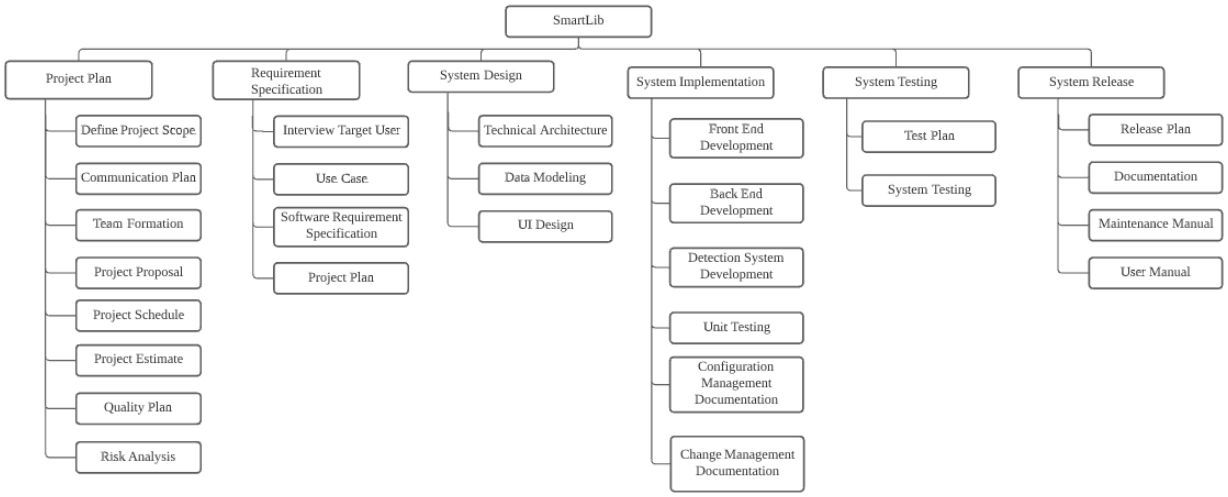
4 Schedule

4.1 Activity Dependencies and Schedule

		Tas...	Task Name	Duration	Start	End	Predecessors
1	✓		**Project Plan**	**12 days**	**19/1/2021**	**3/2/2021**	
2	✓		Define Project Scope	5 days	19/1/2021	25/1/2021	
3	✓		Communication Plan	1 day	19/1/2021	19/1/2021	
4	✓		Project Estimate	2 days	19/1/2021	20/1/2021	
5	✓		Project Schedule	2 days	26/1/2021	27/1/2021	2
6	✓		Project Proposal	3 days	28/1/2021	1/2/2021	5
7	✓		Quality Plan	3 days	28/1/2021	1/2/2021	
8	✓		Risk Analysis	3 days	28/1/2021	1/2/2021	
9	✓		**Requirement Specification**	**14 days**	**2/2/2021**	**19/2/2021**	
10	✓		Interview Target User	3 days	2/2/2021	4/2/2021	6
11	✓		Use Case	2 days	5/2/2021	8/2/2021	10
12	✓		Software Requirement Spe...	9 days	9/2/2021	19/2/2021	11
13	✓		Project Plan	9 days	9/2/2021	19/2/2021	11
14	✓		**System Design**	**17 days**	**20/2/2021**	**16/3/2021**	**9**
15	✓		**UI Design**	**8 days**	**20/2/2021**	**3/3/2021**	
16	✓		Dialog Map	4 days	22/2/2021	25/2/2021	
17	✓		Wire Frame	8 days	22/2/2021	3/3/2021	
18	✓		Technical Architecture	5 days	4/3/2021	10/3/2021	12,13,15
19	✓		**Data Modeling**	**4 days**	**11/3/2021**	**16/3/2021**	**18**
20	✓		ER Diagram	2 days	11/3/2021	12/3/2021	
21	✓		Set Up Database	2 days	15/3/2021	16/3/2021	20
22	✓		**System Implementation**	**20 days**	**17/3/2021**	**13/4/2021**	**14**
23	✓		Configuration management ...	3 days	17/3/2021	19/3/2021	
24	✓		Change Configuration mana...	3 days	17/3/2021	19/3/2021	
25	✓		Front End Development	15 days	17/3/2021	6/4/2021	
26	✓		Back End Development	15 days	17/3/2021	6/4/2021	
27	✓		Detection System	15 days	17/3/2021	6/4/2021	
28	✓		Integration	5 days	7/4/2021	13/4/2021	25,26,27
29	✓		Unit Testing	5 days	7/4/2021	13/4/2021	25,26,27
30			**System Testing**	**8 days**	**14/4/2021**	**23/4/2021**	**22**
31			Test Plan	3 days	14/4/2021	16/4/2021	
32			**System Testing**	**5 days**	**19/4/2021**	**23/4/2021**	**31**
33			White Box Testing	5 days	19/4/2021	23/4/2021	
34			Black Box Testing	5 days	19/4/2021	23/4/2021	
35			**System Release**	**5 days**	**26/4/2021**	**30/4/2021**	**30**
36			Release Plan	4 days	26/4/2021	29/4/2021	
37			Documentation	5 days	26/4/2021	30/4/2021	
38			Maintenance Manual	5 days	26/4/2021	30/4/2021	
39			User Manual	5 days	26/4/2021	30/4/2021	



4.2 Work Breakdown Structure



4.3 Work Packages

The entire project work is broken down by the important phases of the software development life cycle. They include the following:

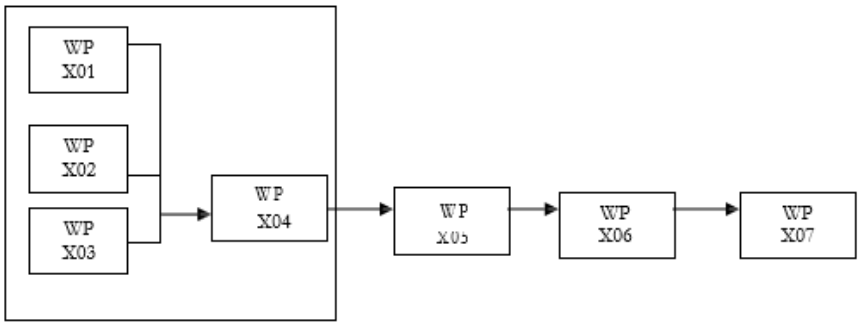
1. Project Plan
2. Requirement Specification
3. User Interface
4. Technical Architecture
5. Data Modeling
6. Coding & Unit Testing
7. Integration & Quality Assurance

4.4 Activity Dependencies

The following table describes the dependencies of the deliverable work packages:

Work Package #	Work Package Description	Duration	Dependencies
X01	Project Plan	9 days	--
X02	Requirement Specification	9 days	--
X03	User Interface	9 days	--
X04	Technical Architecture	14 days	X01,X02,X03
X05	Data Modeling	8 days	X04
X06	Coding & Unit Testing	23 days	X05
X07	Integration & Quality Assurance	30 days	X06

The following Activity Network Diagram describes the above in more graphical detail:



Note that work package X05 is dependent on all work packages encapsulated by the larger boxes linked to its left. For instance, WP X05 may not start until WP X01- X04 has been finished.

4.5 Work Package Details

The tables below describe each Work Package in detail. The names of each team members assigned to each work package will be bolded where they will be responsible in coordinating and ensuring that the activities encapsulated in each Work Packages are met.

Project	SmartLib
Work Package	1— Project Plan (1 of 7)
Assigned To	**Kenny Voo Tze Rung**
Effort	8 Days
Start Date	2 February 2021
Purpose	Provide an overview of the project.
Inputs	None.
Activities	This work package includes providing a brief overview of the project, its objectives, and a set of proposed project deliverables throughout the development of the software cycle. The people responsible for this work package will also be transcribing ideas brought up in the group meeting discussion into a formal report.
Outputs	A written document of the Project Plan.

Project	SmartLib
Work Package	2 — Requirement Specification (2 of 7)
Assigned To	**Wilson Tai**
Effort	8 Days
Start Date	2 February 2021
Purpose	To establish a common understanding between the customer and the software project team of the customer's requirements to be addressed by the project.
Inputs	Customer's Requirements

Activities	This work package includes identifying and interviewing customers, writing and inspecting customer requirements.s.
Outputs	A written document of the Requirement Specification.

Project	SmartLib
Work Package	3 — User Interface (3 of 7)
Assigned To	**Kong Hou Jing, Irsyad**
Effort	8 Days
Start Date	2 February 2021
Purpose	To build a user friendly interface for both users to interact with
Inputs	User Information
Activities	This work package includes getting user information, identifying interface actions, displaying the dialog between system and user and displaying the result of request.
Outputs	User Interface

Project	SmartLib
Work Package	4 — Technical Architecture (4 of 7)
Assigned To	**Wilson Tai**
Effort	14 Days
Start Date	12 February 2021
Purpose	To do the high level architecture design.
Inputs	Project Plan, Requirement Specification, User Interface
Activities	This work package includes identifying architecture of software system, components and relationships between them, deciding on the software and hardware infrastructures, deciding the language used to implement the software, and addressing the design topics including maintainability, portability, and reusability.

	High level design entails defining the architecture of the software system and identifying the various components and how they are inter-related to and interactive with each other. Designers also need to decide on the software and hardware infrastructures, such as what operating system on which the software is built, the language used to implement the software, and so on. Design topics including maintainability, portability, and reusability will be addressed here as well.
Outputs	A written document of the High Level Design and Architectural Specification.

Project	SmartLib
Work Package	5 — Data Modeling (5 of 7)
Assigned To	**Zeyu**
Effort	8 Days
Start Date	4 February 2021
Purpose	To build the project's database
Inputs	Project Plan, Requirement Specification, User Interface, Technical Architecture
Activities	This work package includes analyzing the data flow relationships, entity relationships.
Outputs	A written document of the Data Modeling.

Project	SmartLib
Work Package	6 — Coding & Unit testing (6 of 7)
Assigned To	**Kenny Voo Tze Rung, Kong Hou Jing, Irsyad, Zeyu, Boon Shuan, Wilson**

Effort	23 Days
Start Date	17 March 2021
Purpose	To implement the system as per the requirements specification, conduct preliminary testing and crafting other associated documents.
Inputs	Project Plan, Requirement Specification, User Interface, Technical Architecture, Data Modelling
Activities	This work package includes implementing the modules according to the design specifications noted in the Specification document, preparing unit test plans and examining the different paths through the modules.
Outputs	A written document of the Unit Test Report, source code and header files.

Work Package	7 — Integration & Quality Assurance (7 of 7)
Assigned To	**Boon Shuan, Wilson Tai**
Effort	35 Days
Start Date	15 April 2021
Purpose	To perform black box testing, white box testing to check for logical errors, identify and fix syntactical errors produced during the implementation of the system. Evaluation of the overall project performance to provide confidence that the project satisfies the relevant quality standards.
Inputs	Project Plan, Requirement Specification, User Interface, Technical Architecture, Coding & Unit testing, Data Modelling, Coding & Unit testing
Activities	This work package includes examining issues such as system performance and integrity. Metrics are used to develop strategies for improving the software process, and thus, improving the quality of the end product.
Outputs	A written document of the Test Report and Quality Assurance Report.

5 Project Estimates

5.1 Code Size Estimation using Function Points

We calculated unadjusted function point based on the complexity of functions provided by this system. Code size is then estimated by adjusted function point.

5.1.1 Unadjusted Function Points

SmartLib supports the following proposed functions:

Student:

- Save new student account to the database (Registration)
- Retrieve login information from the database
- View the availability of individual seats in the library
- Make seat reservation
- View past booking information

Administrator:

- Modify the seat status and update to external database
- Alert if the seat is hogged
- Generate report for the librarians

Detection System:

- Detect and update the seat status to external database (python)

The measure of unadjusted function points is based on five primary component elements of these functions: Inputs, Outputs, Inquiries, Logical Files, and Interfaces. Each element ranges from Low Complexity, Medium Complexity to High Complexity. The detailed evaluation of the complexity is as follows:

Element	**Complexity**	**Detail**
Inputs	Low	Registration
	Low	Make Seat Reservation
	Medium	Modify seat status
Outputs	Medium	Generate report
	Low	Alert if the seat is hogged
Inquiries	Low	View the availability of individual seats in the library
	Low	Login
	Low	View Past Booking Information
Logical Files	Low	Registration
	Low	Make Seat Reservation

	Low	Admin update seat status
	Medium	Detection system update seat status (python)
Interfaces	Low	modify and update seat status to external database
	Low	detect and update seat status to external database(python)

Calculation of Unadjusted Function Points:

Characteristic	**Low**		**Medium**		**High**	
Inputs	2	× 3	1	× 4	0	× 6
Outputs	1	× 4	1	× 5	0	× 7
Inquiries	3	× 3	0	× 4	0	× 6
Logical Files	3	× 7	1	× 10	0	× 15
Interfaces	2	× 5	0	× 7	0	× 10
Unadjusted FP	50		19		0	
Total=L+M+H	69					

5.1.2 Adjusted Function Points

Influence Factors	**Score**	**Detail**
Data Communications	4	Application is more than a front-end, and supports more than one type of teleprocessing communications protocol.
Distributed Functions	4	Distributed processing and data transfer are online and in both directions.
Performance	3	Response time or throughput is critical during all business hours. No special design for CPU utilization was required. Processing deadline requirements with interfacing systems are constraining.
Heavily used	2	Some security or timing considerations are included.
Transaction rate	4	Daily peak transaction period is anticipated.
On-line data entry	5	More than 30% of transactions are interactive data entry
End-user efficiency	2	Four to five of the efficiency designs are included
On-line data update	3	Online update of major internal logical files is included.
Complex processing	1	Any one of the complex components
Reusability	4	The application was specifically packaged and/or documented to ease re-use, and the application is customized by the user at source code level.
Installation Ease	1	No special considerations were stated by the user *but* special setup is required for installation.

Operational Ease	1	Effective start-up, back-up, and recovery processes were provided, but no operator intervention is required (count as two items).
Multiple sites	0	User requirements do not require considering the needs of more than one user/installation site.
Facilitate change	3	Flexible query and report facility is provided that can handle complex requests, for example, *and/or* logic combinations on one or more internal logical files (count as three items).
Total score	37	
Influence Multiplier = Total score × 0.01 + 0.65 = 37 × 0.01 + 0.65 = 1.02		
Adjusted FP = Unadjusted FP × Influence Multiplier = 69× 1.02 = 70.38		

Scoring (0 – 5)
0 = No influence
1 = Insignificant influence
2 = Moderate influence
3 = Average influence
4 = Significant influence
5 = Strong influence

5.1.3 Lines of Code

According to Quantitative Software Management, each Function Point requires 53 lines of code if the application is implemented using Javascript and 24 lines of code if the application is implemented with python.

Therefore, we have: **Lines of Code** =54.78 FP × 53 LOC/FP +15.6x 24LOC/FP= 3277.74**LOC**

5.2 Efforts, Duration and Team Size Estimation

To estimate the effort and duration required for the project, we use function points as the basis to calculate Effort, Duration, Team size and finally the schedule. The estimates are expanded to account for project management and extra contingency time to obtain the total average effort estimates. From these averages, the duration of each work package in working days is estimated based on the following calculations.

- Working days include 5 days in a week.
- Effort = Size / Production Rate = (3277.74 LOC) / (39 LOC/PD)[1] = 84PD

- Duration = 3 × (Effort) $^{1/3}$ = 3 × (84) $^{1/3}$ =13.14Days
- Initial Schedule = 13.14 / 5 days in a week = 2.628 Weeks
- Team size = 84 PD / 13.14 D =6.39 P = 7 Persons
- Working hours include 8 hours in a working day.
- Total person-hours (PH) = 84 PD × 8 hours = 672 PH

Based on the estimation, this project will take around 2 weeks to complete provided that there's 7 people in the team working 5 days per week. For our case, if we assume that every one of our team members commits 1 day of effort per week, which means that this project will take around 13 weeks which is 3 months to complete. Therefore, the project will be planned for 3 months.

5.2.1 Distribution of Effort

1990's Industry Data	**Work Package**	**Distribution**	**Estimates**
Preliminary Design 18 %	Project Plan	9%	60.48
	Requirement Specification	9%	60.48
Detailed Design 25 %	User Interface	7%	47.04
	Technical Architecture	11%	73.92
	Data Modeling	7%	47.04
Code & Unit Testing 26 %	Code & Unit testing	21%	141.12
	Online Documentation	5%	33.6
Integration & Test 31 %	Integration & Quality Assurance	31%	208.32
	Extrapolated total effort		672
	2% for project management		13.44
	3% for contingency		6.25
	Total effort		691.69

These duration estimates are based on the assumption that each team member works an equal amount on any given work package.

[1] Lines of code per Person Day statistics based on Industrial Benchmarks, 1997: 31 LOC/PD for United States; 62 LOC/PD for Canada

5.3 Cost Estimates

These estimates are based on the estimated project schedule of 3 months.

Hardware:

Developer workstations:

6 - Dell Precision Workstation 3930	
Intel Xeon 6-core 2.4GHz processor	
16 GB RAM	$6,000.00
256 GB SSD storage	
10 - Wifi Camera	$1190.00
	Total $7190.00

Software:

Software License Provided by Third Party:

Microsoft Visual Studio Code	$0.00
Firebase Cloud Database	$1,500.00

Other Resources:

Staff:

Project Manager	$15,000
Team Members (System Developers and Quality & Release Managers)	$45,000
	Total $60,000

Stationary:

Paper, photocopying and other miscellaneous cost	$50

Total:

$68,740

The customer will supply the required hardware and software necessary to run the back-end and

the front-end web server. Team Xeon is not responsible in any way for supplying said systems. SmartLib's hardware and software responsibilities relate only to our own development needs to accomplish the project we have been asked to complete, and which has been described in the introduction section of this document. SmartLib will also demonstrate the completed product.

6 Product Checklist

The plan is that the items listed below will be delivered on the stated deadlines.

Deliverables	**Estimated Completion Date**	**Final Dateline**
Project Proposal	30th January 2021	2nd February 2021
Use Case Model	30th January 2021	2nd February 2021
System Requirement Specification	13th February 2021	16th February 2021
Quality Plan	14th February 2021	16th February 2021
Project Plan	27th February 2021	2nd March 2021
Risk Management	27th February 2021	2nd March 2021
Prototype Demo	27th February 2021	2nd March 2021
Prototype related items; Code, documents, Powerpoint Slides or Video Clips	27th February 2021	2nd March 2021
Design Report on Software Maintainability	4th April 2021	6th April 2021
Configuration Management Plan	4th April 2021	6th April 2021
Change Management Plan	4th April 2021	6th April 2021
Release Plan	4th April 2021	6th April 2021
Presentation Slides (Project Introduction and Summary)	4th April 2021	6th April 2021
Test Plan	4th April 2021	6th April 2021
Test Cases and Requirements Test Coverage Report	4th April 2021	6th April 2021
CMMI Level 2 Definition	4th April 2021	6th April 2021

7 Best Practice Checklist

Practice	9
Document what we do; all documentation must be in a standardized format.	
Pay attention to requirements, check for ambiguity, completeness, accuracy, and consistency. The requirement documentation must contain a complete functional specification.	
Keep it simple. Complexity management is one of the major challenges. Strive to: • Minimize interfaces between modules, procedures and data. • Minimize interfaces between people, otherwise exponential communication cost • Avoid fancy product functions, design as long as the functionality meets the customer requirements	
Require Visibility. We must see what we build otherwise we can measure the progress and take management action. This includes: the manager must have good communication with his or her employees; require developers to make code available for review; review design for appropriateness.	
Plan for continuous change. We must: • All manuals designs, test, source code should have revision numbers and dates revision history comments, change marks to indicate the changes • New revisions should be approved before being made and checked for quality and compliance after being made • Use a configuration management system and make processes • Required maintenance	
Don’t underestimate. We must be careful to obtain accurate estimates for: time, effort, overhead, meeting time, and especially effort on integration, testing, documentation and maintenance.	
Code reviews are a much more efficient method to find software defects. Plan and manage code reviews between team members	
Software testing will use both black box and white box testing. It will involve unit, functional, integrating and acceptance testing.	

8 Risk Management

Besides the general risk management, the following risks have been identified for the SmartLib project:

More changes to requirements than anticipated
Impact Severity: High
Probability: 25%
Impacts: Depending on the stage at which changes occur, could range from needing to update the requirements documentation to needing to do a complete redesign.
Risk Reduction: Be rigorous in eliciting requirements. Make customers aware of potential repercussions of requirement changes.

Specification Delays
Impact Severity: High Probability: 15%
Impacts: Delay in finalizing the specification will push the schedule for all following stages of the project.
Risk Reduction: Monitor progress of specification carefully.

System size underestimated
Impact Severity: Moderate
Probability: 30%
Impacts: More work will need to be spent on design and coding; could negatively impact schedule. Risk Reduction: Update estimates often as the project progresses.

Staff leaving before project complete
Impact Severity: Extreme
Probability: 5%
Impacts: There would be more work for remaining employees, and any specialized skills or knowledge would be lost.
Risk Reduction: Offer benefits and incentives to staff.

Problems coordinating within group
Impact Severity: Moderate
Probability: 40%
Impacts: Members may be unaware of what is expected of them; managers may not be able to measure progress; portions of projects not completed.
Risk Reduction: Follow communication plans as documented in section 2.3

Underestimation of funding due to changes in price
Impact Severity: High Probability:

25%
Impacts: Depending on the stage at which changes occur, we could seek other cheaper alternatives or request for additional funding.
Risk Reduction: Always request quotations from multiple companies so that we could always have more than one option when the previously selected option is no longer available.

Specification
Staff Unavailability
Impact Severity: High Probability: 20%
Impacts: Delay in work completion and slowing down the progress of the project.
Risk Reduction: Assign two or more people in a team that focus on a particular work assignment (e.g. Development Team for coding and Documentation team for work documentation purposes).

System Performance Issue
Impact Severity: High
Probability: 5%
Impacts: Clients' business activities would be affected due to the performance issues encountered
Risk Reduction: Create a progressive benchmark to ensure the performance is consistent with the client's expectations.

Error during the presentation
Impact Severity: Medium
Probability: 30%
Impacts: Client's time would be wasted due to the failure of execution by the software and reducing the client's faith in our ability.
Risk Reduction: Multiple testing should be conducted regularly before the deadline and development log has to be updated consistently.

Uncertainty on how to use the interface/software
Impact Severity: Extreme
Probability: 5%
Impacts: If the UI designed was not user-friendly, the client wouldn't be able to use it properly
Risk Reduction: Create a guidebook or provide training for the user to familiarize themselves with the system.

Poor Estimation of project timelines

Impact Severity: Moderate
Probability: 50%

Impacts: Developers have to overwork at the later stages of the project, affecting the quality of the end product or overshooting the project deadline.

Risk Reduction: Developers have to work hand in hand with the Project Manager to estimate the project timelines to ensure that the project is estimated correctly.

9 Quality Assurance

The project will achieve the quality assurance by following the standard set by the company. The specific procedures and details shall be provided in the Quality Plan.

Specific test procedures and details shall be provided in the Module/System Test Plan.

In addition, the SmartLib shall make use of two testing methodologies:

- **Unit Testing** involves testing system components individually.
- **In-Place Testing** involves testing of the whole system as a unit.

Furthermore, these methodologies will be used to test these important aspects of the SmartLib:

- **System Function** will be tested to ensure that software flaws are eliminated, and
- **Algorithmic Function** will be tested to ensure that heuristic aspects of the project (such as differentiating humans from objects in the human detection system) so as to provide a more accurate result for the end user, minimizing detection errors.
- **User Friendliness** will be tested to ensure that SmartLib users are able to navigate through the app interfaces easily with little to no relearning/recap required, increasing its familiarity.

The team's methodology makes broad use of realistic test cases. Detailed test data is an important part of the final project delivery. Although the team's client is expected to furnish and enter data regarding flights, cities, airports, and the other aspects involved in the Airline Agent, the team shall provide a comprehensive and detailed subset of this data for testing purposes. The team will validate code and heuristic result ranking technology using realistic scenarios. In addition, extreme cases (such as departure from a city without an airport nearby) will be used to ensure that the system behaves correctly in degenerate cases.

Software quality consists of the following attributes:

1) **Functionality**
    a) Do the functions of the application match customer requirements?
    b) Are the functions implemented correctly, and do they interact with other components properly?
2) **Reliability**
    a) Is the application able to perform under specific conditions?
    b) How often do bugs and rogue test cases appear? How are they handled?
3) **Usability**
    a) Can the users understand the application intended usage easily?
4) **Efficiency**
    a) Are good coding techniques followed?

5) **Maintainability**
    a) Can bugs be dealt with easily?
    b) Can improvements or fixes be implemented onto the application efficiently?
6) **Portability**
    a) Can the system adapt to changes in its environment? How easy can the system be accessed?

The QA team, along with the Project Manager should ensure that these guidelines are followed and ensure quality code is produced whilst developing the application.

We shall make use of two testing methodologies:

- **White Box Test -** test performed by our Quality Assurance team to ensure the high quality product developed for the clients. This may include:
  Test Automation with software testing tools like Selenium, to ensure all features perform their purpose without any error or bug.
- **Black Box Test -** Beta test or test performed by the users who know what the software is all about but unaware how the input is being processed and output is produced. This may include:
  Getting the users to come down to the library to use and explore the application from the library kiosk with the implemented software, to test the use cases. This will help achieve user friendliness and high customer satisfaction.

10 Monitoring & Control

Many procedures are required in order to be able to successfully monitor the progress of a software project. Some of the most important are:

Quantitative measurement of resource consumption: Estimates of the SmartLib's resource requirements, primarily in terms of human resources, can provide a quantitative measurement of project progress when compared to progress in terms of project milestones. The percentage estimates of each milestone's resource requirements provided in this document allow for easy progress tracking.

Identification of major project risks: Early identification of major risks to the project allows for placement of preventative measures before problems can develop. Major risks have been identified in the Risk Management section of this document, along with the measures being taken to avoid them.

Regular reviews of project progress: Throughout the duration of the SmartLib project, the team shall meet weekly to review the progress of all project tasks, including management, planning, analysis, development, and testing.

Timeline Planning and task decomposition: This document outlines an estimated timeline for the project. A reasonably accurate timeline can be assembled by hierarchically decomposing tasks into measurable subcomponents and estimating requirements for each. At the same time, this decomposition can assist in task assignment and balancing. Throughout the implementation phase, these subcomponents can allow for fine-grained measurement of progress. Project subcomponents and timeline estimates are included in the Estimates and Work Breakdown Structure sections of this document.