School of Information and Communication Technology

Griffith University

3020ICT – Industry Affiliates Program

SmartEye Project Development

Project Report

David Gray s2947315

26/05/2017 Trimester 1

RedEye Apps

Nicolas Dijoux

Andrew Lewis

Table of Contents

Executive Summary

RedEye Apps are a software development company specialising in creating cloud based, purpose built, engineering management solutions. Their EDCMS (engineering drawing collaboration & management solutions) help improve the working experience of their clients by simplifying their workflows and allowing them easy organisation of their data and tasks.

SmartEye is RedEye's new development project that will use AI (artificial intelligence) techniques to extract insight and data from client documents. Examples of the insight and data that can be extracted are; essential metadata details, the type of document that has been uploaded, and identification of similar documents already uploaded.

SmartEye will allow RedEye's clients to have greater control over the information contained in their files, giving them the ability to search through uploaded files for specific information and metadata along with allowing them to upload a greater variety of file types.

My part of the project is the data extractor of SmartEye, allowing users to upload files to the RedEye server which will then be converted to images and sent to Google Cloud Vision, with the results returned and stored in RedEye's database. Clients/users will then be able to query RedEye's servers for information on their files stored in the database.

It will require: a database to store transaction and file information; a web server to accept user/client requests; the ability to accept image and pdf files to upload; a file converter that can convert and split pdf files into individual image files to upload; a search feature returning page and document results; and a multithreaded Golang application integrating all these aspects.

The proposed solution for the SmartEye data extractor tool involved creating a multithreaded Golang application that can accept file uploads and return data requests along with an API solution for SmartEye that will allow users to make requests to the program via a web application.

The SmartEye program utilises both Google Golang and external libraries. The SmartEye data extractor uses a web application as a front end for the users/clients which allows them to upload image files, upload pdf files to split, or search for documents and pages in the database. This web application will send requests to the API to either return database results, or upload the given files, sending them to Google Cloud Vision, returning a set of metadata results which is then stored into the SmartEye database.

The project uses GitHub as a repository to store all updates of the program throughout its creation.

The main issue that occurred during the project was my illness during Sprint 4. This was resolved using a push back plan created that moved tasks from Sprint 4 to Sprint 5. This plan was a solution described in RedEye's risk management plan for the project.

Due to extensive initial planning, estimation and scheduling, the SmartEye data extractor tool was completed on time, with all deliverables, objectives and success criteria being met. The final tool consists of a web application for the user/client to interact with, and a Golang server and program which takes requests and manipulates the data from the requests to return results to the user or receive results from Google Cloud Vision. The final project met all RedEye standards and was approved by Nicolas Dijoux (industry supervisor).

1 PROJECT PLAN

1.1 INTRODUCTION

1.1.1 Project Overview

The industry partner for this project is RedEye Apps. RedEye Apps are a software development and engineering company who create cloud based, purpose built, engineering management solutions. They create EDCMS (engineering drawing collaboration & management solutions) to help improve the working experience of their clients.

The products that RedEye apps produces allows clients to increase their productivity by: organising their drawings, workflows and tasks; allowing for mark-ups and approvals on those drawings, workflows, and tasks; and being able to securely share the drawings, workflows and tasks.

The industry partner contact is Nicolas Dijoux who is a part of the DMS team of RedEye who are building a cloud based solution that manages engineering data and drawings that allows asset owners and their suppliers to access data from many different devices in any location.

In the engineering landscape there are very few ways to easily view, access, and retrieve all the data from drawings and documents using a computer or mobile device. SmartEye is a new development by RedEye which aims to help combat this problem so that engineering firms are able to access all the information they need, from the files they upload or have access to, from any device.

The SmartEye project is a new cloud based solution that will let asset owners and suppliers upload various file types to the RedEye database. These files will be uploaded to Google Cloud Vision which will extract data from these files and return the extracted metadata to the RedEye database. The metadata will then be stored so it can be accessed by RedEye and its clients. The SmartEye solution will allows clients to have greater control over information in their files, as well as allow clients to search through all their uploaded files for specific information.

This final solution of this project will be integrated with the current RedEye DMS system to give clients and asset owners the ability to upload more varied file types to store engineering information, and change their physical based documentation systems to digital in an easier manner.

1.1.2 Project Objectives

The objective of this project is to create a cloud software solution that will be able to handle pdf and text document inputs, which will be uploaded to Google Cloud Vision which will return data that will need to be stored in the RedEye DMS database to be accessed by RedEye and respective clients. This will help the company fulfil the needs of its clients who are currently using old, paper based file systems convert to a modern digital system, increasing the accuracy and efficiency of their file/data storage and giving them increased and easier access to this data.

This software solution will be made in accordance to RedEye's software guarantee of "Easy + Relevant + Fast + Secure = Essential". This means that the end product will: be using innovative software designs and takes; be easy to use and the code easy to read; runs efficiently and processes requests promptly; have minimal external security risks. Integrating this into the product design will make it essential for the clients to use in their daily processes.

1.1.3 Proposed Solution Design and Requirements

Proposed database solution design:

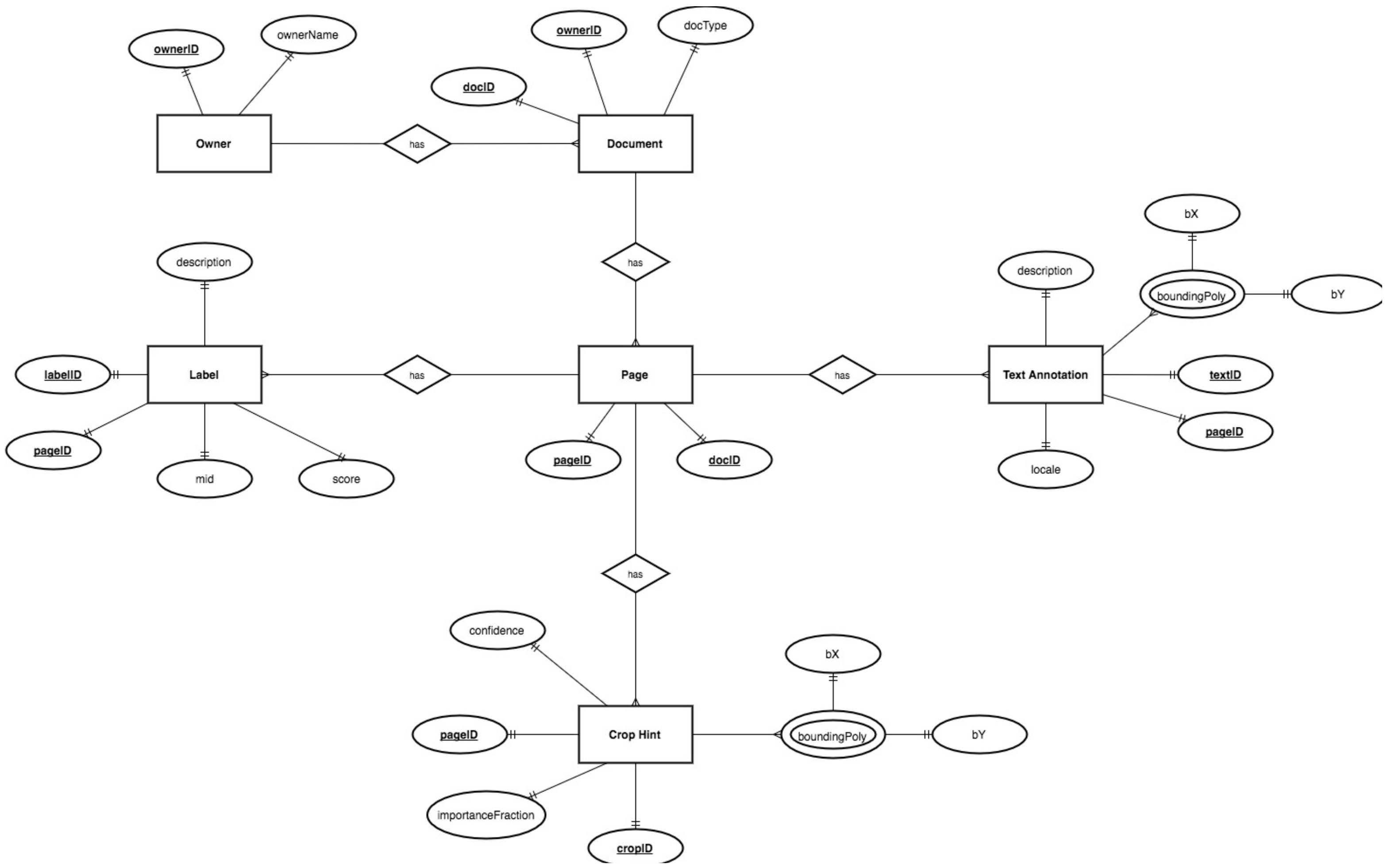


Figure 1 (Database design)

The database must be able to store: the owners of the documents; the documents; the pages relating to the documents; the labels on each page; the text annotations and their locations on each page; and the crop hints and their locations on each page.

Proposed main solution design:

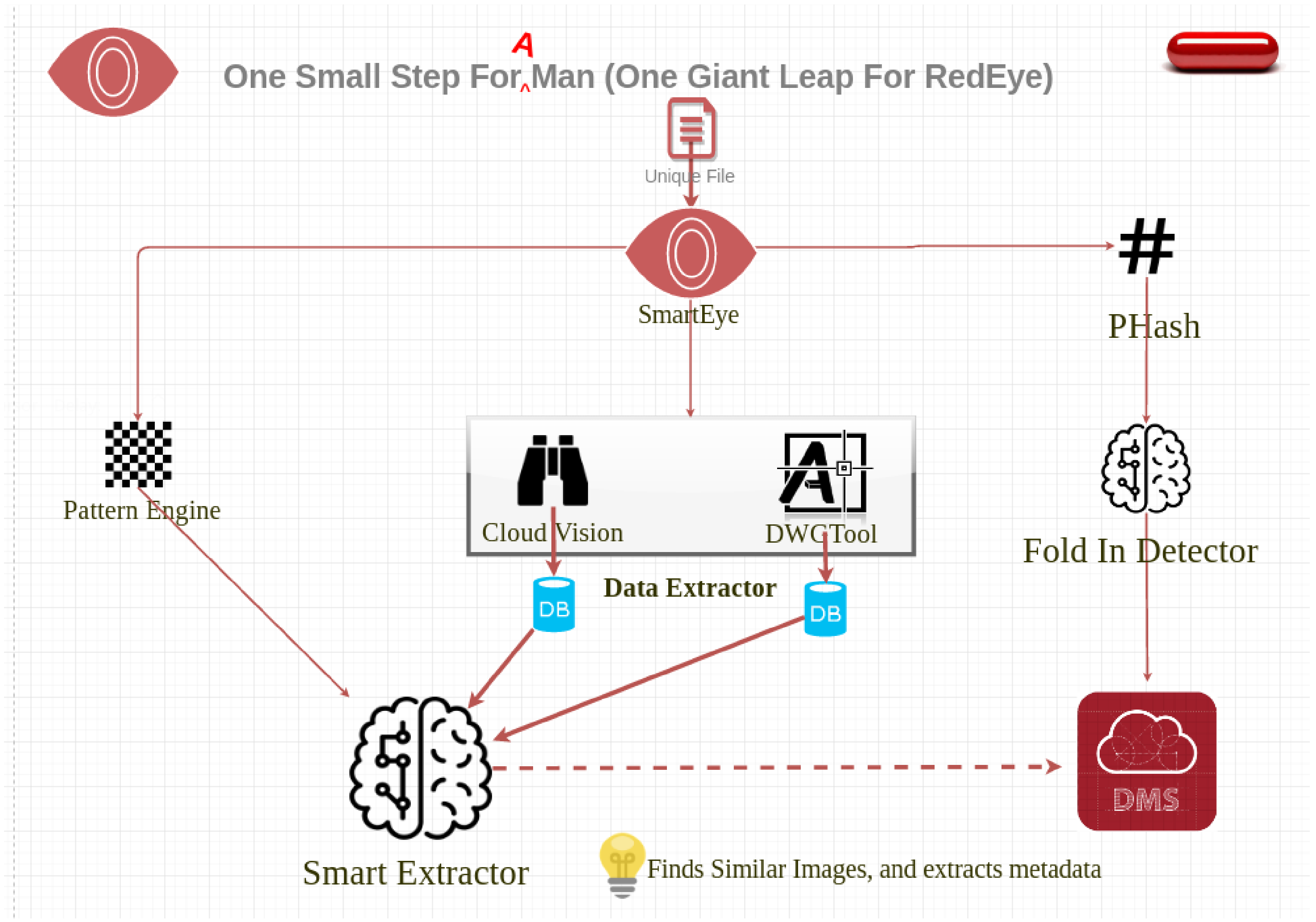


Figure 2 (SmartEye design)

SmartEye (main solution + database) requirements:

Objective	Requirements
Upload various document/image file types	● Allow upload of pdf, excel, png, jpg, and doc files ● Securely store data of uploaded files using hashing system
SmartEye database creation	● Create reusable database schema ● Have documents and pages tables ● Have relevant data types and tables for all potential Cloud Vision returned data
Connect SmartEye database to Golang program	● Have secure account set-up between SmartEye and program ● Create Golang database schema with Goose and SQLBoiler
Connect Golang program to Google Cloud Vision and DWG Tool	● Set-up secure data transfer system between program and Google Cloud Vision ● Have data and account encryption for connection between database, Google Cloud Vision, and DWG Tool
Retrieve and parse json image/document metadata from Google Cloud Vision	● Upload secure files from program to Google Cloud Vision ● Retrieve sent back metadata from Google Cloud Vision ● Store json metadata in program ● Parse json metadata and store relevant information into Golang and SQL database
Have hashing system for image data storage and searching	● Use hashing system for storing data information and ids of image ● Allow searching of images through ids
Integrate DMS access to SmartEye	● Allow secure access between Golang SmartEye program and current DMS solution ● Allow DMS solution to search and upload images to SmartEye

1.1.4 Project Deliverables

Planned Deliverable	Deliverable Description	Sprint #
Project breakdown report created	Plan project deliverables, objectives, risk assessment, change control management, reviews, and retrospectives.	1
Golang IDE and libraries set up	Install Gogland IDE along with Docker, PostgreSQL, SQLBoiler, and Goose	1
SQL database for project designed and created	Read list of database requirements and create an ERD diagram outlining them and from that create a database schema in SQL.	1
Document and page instances created in the database using Golang	Use SQLBoiler library with Golang to add a document and pages table and insert data into these tables using Golang.	1
Sample Cloud Vision data stored into database	Create sample Cloud Vision data sets and store this data into the database using Golang.	1
Sprint 1 deliverables unit tested	Break down reports of each part of the software created in the sprint.	1
Sprint 1 retrospective/review Report	Go through deliverables of Sprint 1 and analyse the successes/failures and adapt future sprints/methodologies to the results.	2
Google Cloud Vision account set up	Create and add funds to Google Cloud Vision account.	2
PDF files split and converted to images using Golang.	Take a PDF file, and convert each page of the PDF into individual images using Golang.	2
Returned data from Google Cloud Vision stored correctly after image upload	Set up program with Golang to be able to send an image to Google Cloud Vision and store the returned data into the database.	2
Log files created and recycled using Golang	Create logs of program and save them to a log file and recycle them if necessary.	2
Sprint 2 deliverables unit tested	Break down reports of each part of the	2

	software created in the sprint.	
Sprint 2 retrospective/review Report	Go through deliverables of Sprint 2 and analyse the successes/failures and adapt future sprints/methodologies to the results.	3
Golang application converted to multi-threaded version	Take current created project and convert it to have multithreaded capabilities using Golang.	3
Program performance monitored and tool reliability ensured	Check for various errors, or programing hanging, and store all logs of program.	3
Golang program able to send multiple files to Google Cloud Vision at one time and store the results	Set up program with Golang to be able to send multiple items to Google Cloud Vision and store the returned data into the database.	3
API design research documentation and final API design chosen	Look up various API designs and libraries for Golang and create document of pros and cons to each to decide which one to use later on.	3
Sprint 3 retrospective/review Report	Go through deliverables of Sprint 3 and analyse the successes/failures and adapt future sprints/methodologies to the results.	4
API implementation documentation	Document created and used API for project.	4
Implement API into project	Integrate API with Golang project.	4
API tests completed and documented	Use Postman or other similar API testing methods to test the API. Document results.	4
UI/UX mock-up of front end for program	Create a draft mock up for the user interface that the client will see when using the program.	4
UI/UX research documentation	Choose technologies/programs to use to create user front end for example, Aurelia with Bootstrap.	4
Sprint 4 retrospective/review report	Go through deliverables of Sprint 4 and analyse the successes/failures and adapt future sprints/methodologies to the results.	5

Program front end solution implemented	Set-up front end of program so that users can interact with an easy to use client.	5
Front end solution unit tested	Break down individual components of front end and test them to make sure everything works as intended	5
Sprint 5 retrospective/review report	Go through deliverables of Sprint 5 and analyse the successes/failures and adapt future sprints/methodologies to the results.	6
Deliver project	Finalise anything left of project to complete and complete documentation of project.	6

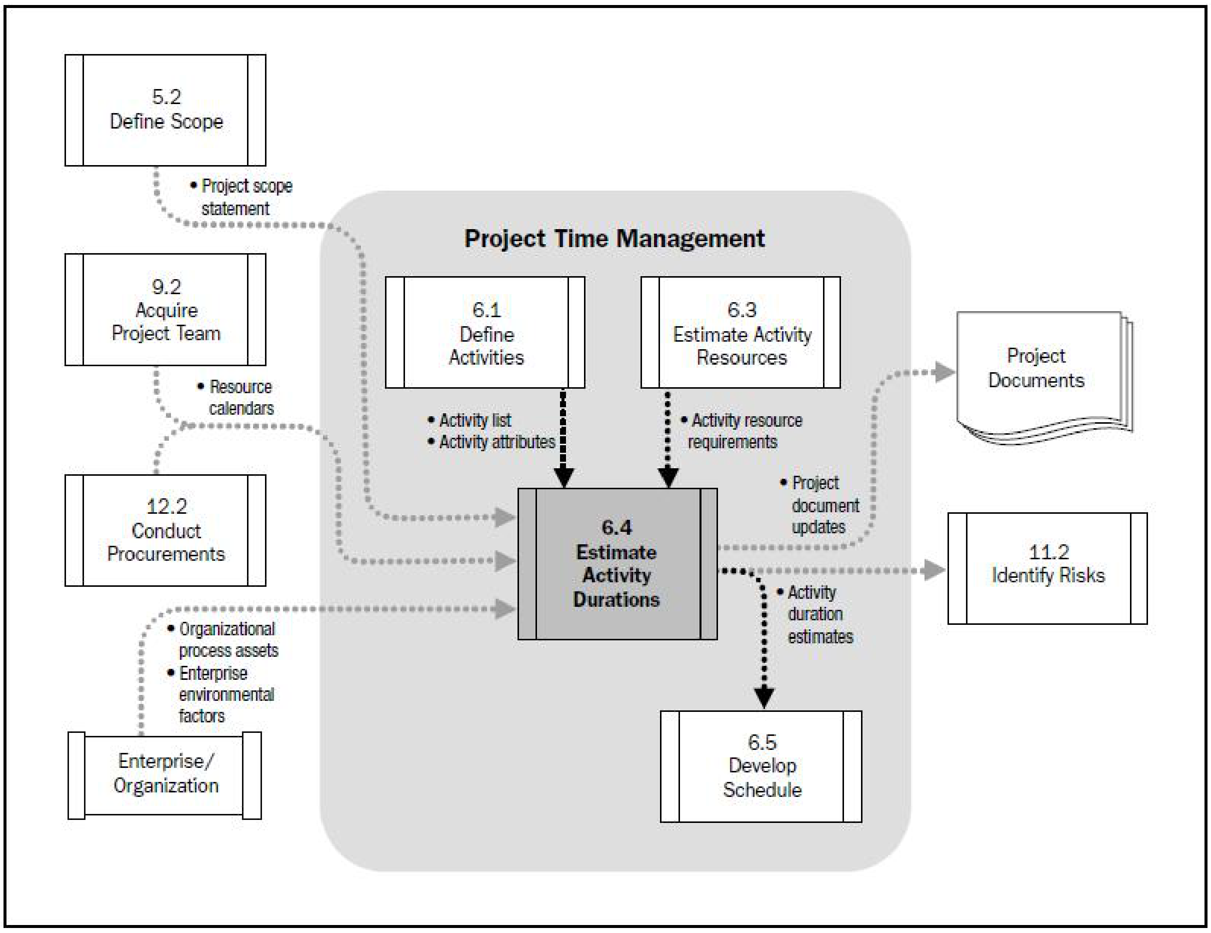
1.1.5 Work Packages and Dependencies

Figure 3 (WBS)

1.1.6 Resource Requirements and Allocation and Project Schedule

Resource estimates for this project were completed using historical project analysis and story point counting. By using previous data, some estimates could be made based on similar tasks of this project. An outline for the process of how the development of the resource and time estimation requirements were made is shown below. This diagram (Figure 4) displays how different control, risk, report, and activity management reviews are collected and used to create a list of user stories to break down project resource requirements, estimates, and achieve an accurate project schedule.

Figure 4 (Resource breakdown) [CITATION Pro13 \l 1033]



In agile based projects, story points are commonly used on user stories and project deliverables/objectives in order to accurately estimate the resources and effort each activity will require. Story points do not indicate the time taken to complete a user story/deliverable. The deliverables/requirements are written as user stories which are given complexity ratings which give a value that represents the many interrelated parts of the project. The value is determined based on time, effort, risk, and potential resources used.

The story point system was used for calculating the effort for each of the tasks/user stories outlined in the project breakdown. From that, the tasks were broken down into sprints and it was made sure that each sprint does not contain a significantly more story points than others in order to have an evenly distributed work timeline and have a lower risk of overestimating the work capable of being completed in each sprint which would cause a work backlog and potentially derail the scheduled timeline of the project.

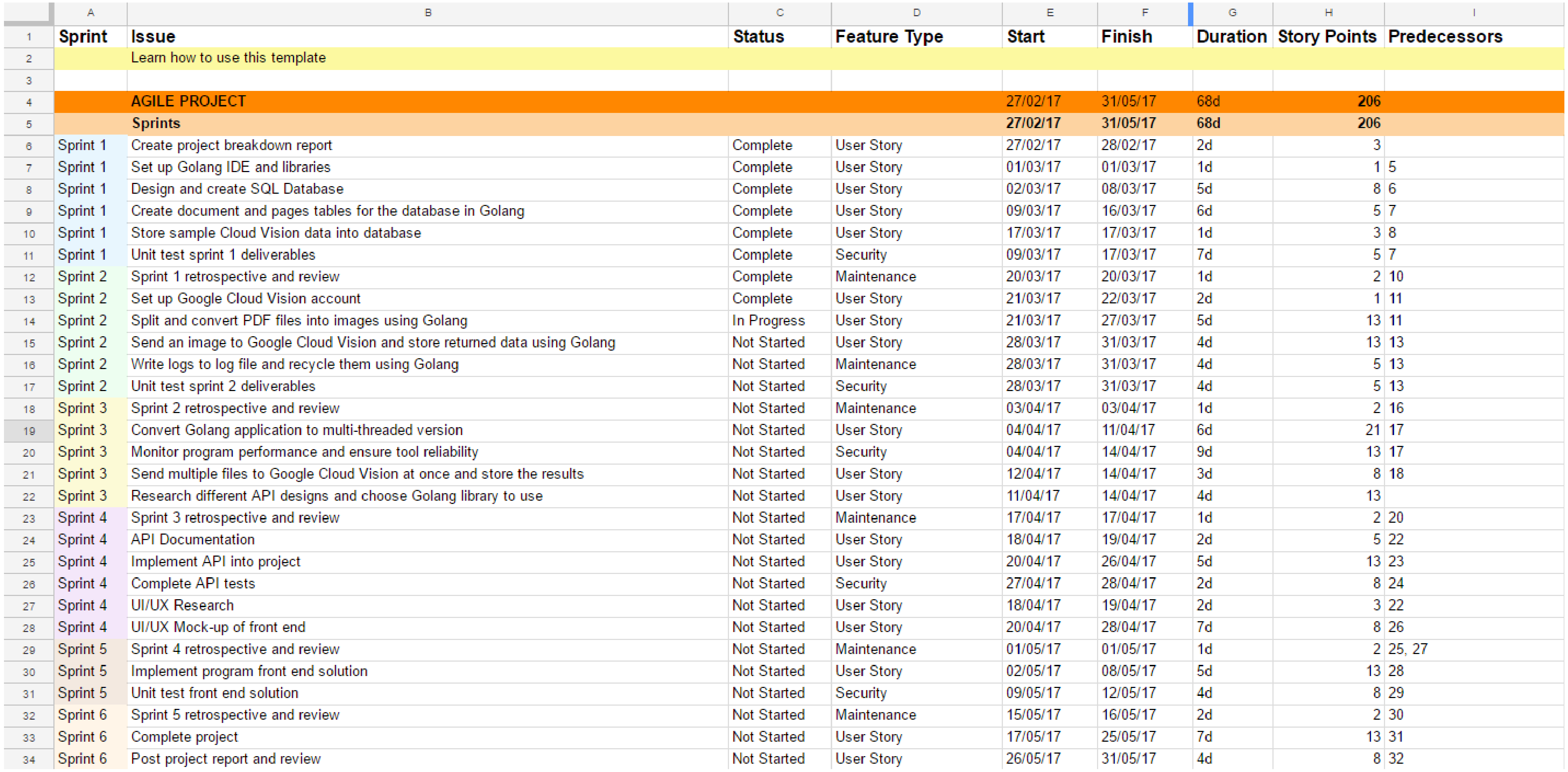
The story point system in this project made use of the Fibonacci number sequence with the maximum potential value for a task being 21.

Each sprint, its tasks, the effort required to complete a task and project as a whole, its timeline, and each deliverables predecessors (deliverables that need to be completed before that deliverable can start) have been documented and are shown in both the table and diagram below (Figure 5 and 6).

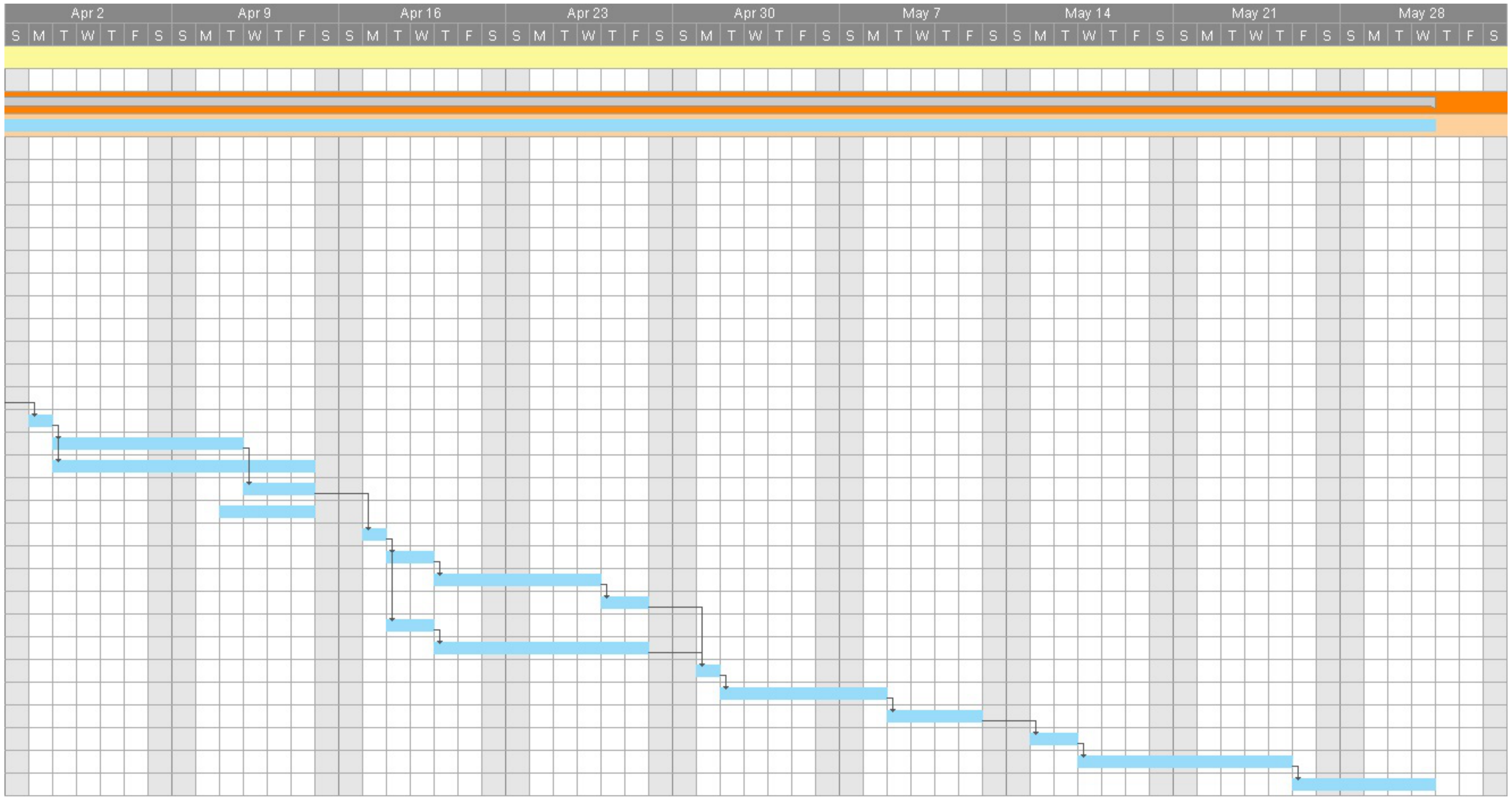
These tables and diagrams will be updated throughout the project to reflect any changes that may have been made.

Figure 5 (Sprint log, schedule, story points, and critical path)

Figure 6 (Gantt chart)



	A	B	C	D	E	F	G	H	I
1	**Sprint**	**Issue**	**Status**	**Feature Type**	**Start**	**Finish**	**Duration**	**Story Points**	**Predecessors**
2		Learn how to use this template							
3									
4		AGILE PROJECT			27/02/17	31/05/17	68d	**206**	
5		**Sprints**			**27/02/17**	**31/05/17**	**68d**	**206**	
6	Sprint 1	Create project breakdown report	Complete	User Story	27/02/17	28/02/17	2d	3	
7	Sprint 1	Set up Golang IDE and libraries	Complete	User Story	01/03/17	01/03/17	1d	1	5
8	Sprint 1	Design and create SQL Database	Complete	User Story	02/03/17	08/03/17	5d	8	6
9	Sprint 1	Create document and pages tables for the database in Golang	Complete	User Story	09/03/17	16/03/17	6d	5	7
10	Sprint 1	Store sample Cloud Vision data into database	Complete	User Story	17/03/17	17/03/17	1d	3	8
11	Sprint 1	Unit test sprint 1 deliverables	Complete	Security	09/03/17	17/03/17	7d	5	7
12	Sprint 2	Sprint 1 retrospective and review	Complete	Maintenance	20/03/17	20/03/17	1d	2	10
13	Sprint 2	Set up Google Cloud Vision account	Complete	User Story	21/03/17	22/03/17	2d	1	11
14	Sprint 2	Split and convert PDF files into images using Golang	In Progress	User Story	21/03/17	27/03/17	5d	13	11
15	Sprint 2	Send an image to Google Cloud Vision and store returned data using Golang	Not Started	User Story	28/03/17	31/03/17	4d	13	13
16	Sprint 2	Write logs to log file and recycle them using Golang	Not Started	Maintenance	28/03/17	31/03/17	4d	5	13
17	Sprint 2	Unit test sprint 2 deliverables	Not Started	Security	28/03/17	31/03/17	4d	5	13
18	Sprint 3	Sprint 2 retrospective and review	Not Started	Maintenance	03/04/17	03/04/17	1d	2	16
19	Sprint 3	Convert Golang application to multi-threaded version	Not Started	User Story	04/04/17	11/04/17	6d	21	17
20	Sprint 3	Monitor program performance and ensure tool reliability	Not Started	Security	04/04/17	14/04/17	9d	13	17
21	Sprint 3	Send multiple files to Google Cloud Vision at once and store the results	Not Started	User Story	12/04/17	14/04/17	3d	8	18
22	Sprint 3	Research different API designs and choose Golang library to use	Not Started	User Story	11/04/17	14/04/17	4d	13	
23	Sprint 4	Sprint 3 retrospective and review	Not Started	Maintenance	17/04/17	17/04/17	1d	2	20
24	Sprint 4	API Documentation	Not Started	User Story	18/04/17	19/04/17	2d	5	22
25	Sprint 4	Implement API into project	Not Started	User Story	20/04/17	26/04/17	5d	13	23
26	Sprint 4	Complete API tests	Not Started	Security	27/04/17	28/04/17	2d	8	24
27	Sprint 4	UI/UX Research	Not Started	User Story	18/04/17	19/04/17	2d	3	22
28	Sprint 4	UI/UX Mock-up of front end	Not Started	User Story	20/04/17	28/04/17	7d	8	26
29	Sprint 5	Sprint 4 retrospective and review	Not Started	Maintenance	01/05/17	01/05/17	1d	2	25, 27
30	Sprint 5	Implement program front end solution	Not Started	User Story	02/05/17	08/05/17	5d	13	28
31	Sprint 5	Unit test front end solution	Not Started	Security	09/05/17	12/05/17	4d	8	29
32	Sprint 6	Sprint 5 retrospective and review	Not Started	Maintenance	15/05/17	16/05/17	2d	2	30
33	Sprint 6	Complete project	Not Started	User Story	17/05/17	25/05/17	7d	13	31
34	Sprint 6	Post project report and review	Not Started	User Story	26/05/17	31/05/17	4d	8	32


Apr 2
Apr 9
Apr 16
Apr 23
Apr 30
May 7
May 14
May 21
May 28
S M T W T F S

1.1.7 Reference Materials

REFERENCES

Project Management Institute. (2013). *A Guide to the Project Management Body of Knowledge* (5th ed.). Project Management Institute.

All images used in this report (apart from tables, sprint images, and schedule/gantt chart images) are taken from the Project Management Body of Knowledge 5th edition.

1.1.8 Definitions and Acronyms

Acronym/Term	Definition
IDE	Integrated development environment - a software application that facilitates computer programmers for software development
WBS	Work breakdown structure - a structured list of tasks/deliverables to be completed
DMS	Document management system
Sprint	A set of tasks scheduled to be completed with a small span of time - usually involved within the iterative process of the SCRUM Agile methodology
User story	A description of something a client/user needs in the product
Story points	Used for calculating the effort required to complete a user story or task
Front end	The part of the program the user interfaces with
Unit test	Breaking down the software into its components and testing them for reliability and security
UI/UX	User interface/User experience
API	Application programming interface - a set of subroutine definitions, protocols, and tools for building application software.
SQL	Structured query language - used to create databases and perform actions upon them

Software library	A collection of pre-written code, classes, methods, procedures, data and more.

1.2 TECHNICAL PROCESSES

1.2.1 Methods, Tools and Techniques

Methods to be used during project:

Method	Description
Agile SCRUM Methodology	Overarching planning of project completion
Sprints	Breaking down tasks and project iterations
Retrospectives and reviews	Review project sprints and deliverables to improve processes and the project
Control/Risk/Scope/Schedule Monitoring	Continuous monitoring of these aspects throughout the project creation in order to reduce chance of project failure

The methods described above will be used throughout the project and they make up the backbone of how and what will be done. The SCRUM methodology is the underlying principle of the processes to complete the project and it is made up of sprint cycles which are iterations of the project and after each sprint there will be retrospectives and reviews of the work that has been completed.

Tools to be used during project:

Tool	Description
Gogland GO IDE	IDE used for project creation
iTerm2 Terminal	Replaces OSX Bash terminal, used for project installation and testing
Docker Virtual Machine	Used with iTerm to run project database and testing
SQLBoiler for GO	Used with Gogland and GO for database functionality
Goose Database Migration System	Used with PostgreSQL to convert database schema into GO usable code
Google Drive	Used for documentation and reports

Atlassian Jira	Used for documenting tasks, work log time, and task completion status
Slack	Instant messaging service for use between team members
Google Cloud Vision	For image metadata gathering
Github	For storing code and project revisions
PostgreSQL	Database management system
Smartsheet	Sprint breakdown and scheduling tool
Creately	WBS creation tool
MacBook Pro 2015	Main computers used for creation of project

Techniques to be used during project:

Technique	**Description**
Meetings	Used to schedule, update, review and renew progress and deliverables of the project
Brainstorming	Used for project breakdown, sprint reviews, risk identification, and most other analysis and review tasks. Involves meeting with a group of individuals with varying backgrounds in the company to create an ideas backlog for the problem at hand.
Historical analysis	Using previous project and company data to analyse the similarities, and differences between them and the current project and what can be done better in the current project over the previous ones.
Risk management	Using risk identification strategies such as brainstorming and historical analysis, a list of risks can be formed. From this list, a further analysis of the ways the risks can be mitigated and controlled is completed with the results documented.
Change management	Using change identification strategies such as brainstorming and historical analysis, a format of how to identify and deal with change is formed. From this format, a request system, and review system is also formed.

Client testing	Potential clients of the project are asked to review test builds of the project during production in order to make sure that their needs are understood and met with the product.
Code reviews	Pull requests and bug testing by both myself and team members on each other's code
Change reviews and requests	Formal hierarchy used to process change reviews and requests
Unit testing	To test each section of code for functionality, reliability, and security before the deliverable can be handed off
Risk matrix priority list	An updated list of most likely risks, their potential effects, and how to mitigate them
Product backlog	List of priority of tasks and user stories to complete in current sprint

1.2.2 Quality Management

Both a general and a software specific quality management plan will be followed during the lifetime of this project. The general quality management plan involves dealing with quality planning, assurance, control and improvement. The specific quality management plan involves with dealing with quality issues of each sprint.

The quality plan is the outline of how the quality processes will take place. This involves a list of RedEye individuals, clients, asset holders, and client liaisons to commune with during different stages of the project for example, dealing with clients and the RedEye DMS team to work out the scope and feasibility of the project in accordance to what the clients need. It also involves: a plan of when contact needs to be made with the groups on this list and what their roles are; the ways quality will be assessed within the project by both myself, the RedEye team, and the clients; the plans to control quality to make sure that the product is fast, easy, relevant and secure; and how to improve the quality through each stage of development.

The quality of this project is determined by: the deliverables being completed; the deliverables being completed on time; the code for each deliverable is easy to read by the other members of the RedEye DMS team and can be easily reused; the code does not allow for any major external security flaws to occur; how does the code perform when completing unit tests in terms of both speed and reliability; the client has determined the deliverables are acceptable to their needs; and that there is consistent communication between clients, RedEye team members, and myself throughout the project lifecycle.

Each sprint will consist of a report and a retrospective of the development outcomes using the quality definitions given above. A review will be held between myself and my industry advisor to see where improvements can be made to maximise the quality of the project production and to make sure that the project is still within the scheduled timeline, scope and budget.

The outcomes from the report and retrospective review will help to improve the subsequent sprints by giving us the knowledge of what did and did not work, reducing the chance of repeating similar mistakes and allowing all those involved to adapt to the suggested changes.

1.2.3 Software Development Life Cycle Model

The software development life cycle model used in this project is the SCRUM Agile SDLC, which makes use of an iterative development consisting of numerous sprints.

The SCRUM Agile SDLC is a software development model that is based around breaking a project up into multiple iterations where solutions and requirements evolve and are met through collaboration between self-organising teams. It involves frequent adaption and inspection of each iteration in order to achieve a result of high quality in the shortest manner of time possible. In the SCRUM Agile SDLC, each iteration has heavy client/user interaction in order to make sure that all the requirements are being met and to see if there are any changes that need to be made. The approach emphasises the importance of adapting to changing circumstances and requirements and the development of working software over significant documentation as the information technology landscape is a rapidly changing field. This means that it is harder to plan long term projects in this field. Due to this, methodologies that have short turn arounds or that involve short term project iterations are preferred.

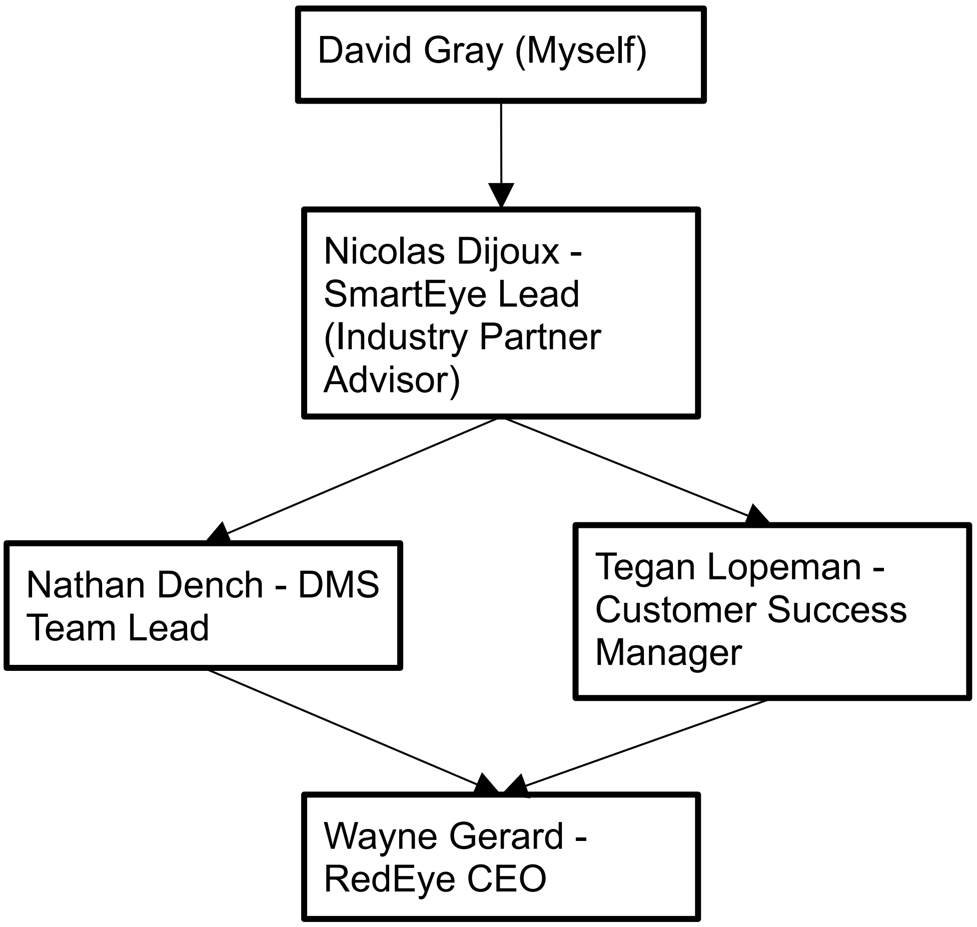
Sprints are small sets of deliverables that will be completed in approximately 2 weeks. The sprints within this project will consist of small tasks working towards the final goal of the SmartEye system base which will be able to analyse documents and return metadata of those documents. It will start with a project breakdown, and more development and software tasks will be added to each following sprint once everything that is required from the previous sprint has been completed. At the end of each sprint cycle a retrospective will be performed to see what happened, what went wrong and what was done well. This retrospective review will be used to improve subsequent sprints.

The reasons why the SCRUM Agile SDLC model was chosen for this project are: the Agile model allows for changes in the project plan to be adapted to in a quicker fashion as the iterative cycles are short and deliverables and processes can be refined after each iteration and the initial plan of the project has the potential for change in mind; the SCRUM Agile model allows for increased work quality over non-iterative SDLCs as the project is broken down into smaller, more manageable parts which are repeatedly iterated upon, reviewed and tested; it also allows for higher levels of collaboration between team members than non-iterative SDLCs; it allows for a better estimate of when deliverables will be completed and handed to the client as the sprints are short periods of time and do not contain a large amount of tasks in to be able to complete them in time and with high quality standards. This means that the iteration periods are more manageable and are less likely to cause overflow; and the company has had historical experience and success using the SCRUM Agile model, allowing for a faster turnaround in project preparation and more familiarity with the processes used and required with the SCRUM Agile SDLC.

1.3 MANAGERIAL PROCESSES

1.3.1 Project Organisational Structure

Figure 7 (Organisation structure)


David Gray (Myself)
Nicolas Dijoux - SmartEye Lead (Industry Partner Advisor)
Nathan Dench - DMS Team Lead
Tegan Lopeman - Customer Success Manager
Wayne Gerard - RedEye CEO

The project organisational structure involves regular communication between myself and my industry advisor, who communicates with the DMS team leader and RedEye customer success manager. The DMS team leader and customer success manager frequently update the RedEye CEO of the progress of all current projects undertaken by RedEye apps.

In the end, the SmartEye project will be used alongside the DMS project from RedEye so that clients have additional functionality to the workflow suites that they purchase from RedEye Apps.

1.3.2 Project Success Criteria

Success Criteria	Deliverables/Objectives Completed	Metrics	Priority
SQL Database created using PostgreSQL and SQLBoiler and has capability to store all	• Design and create SQL Database • Create document and pages tables for the database in Golang	On time Secure Full functionality Easy to use/read Essential	Mandatory

relevant data returned from Google Cloud Vision	• Store sample Cloud Vision data into database	Requirements met	
Golang program able to send files to Google Cloud Vision	• Set up Google Cloud Vision account • Split and convert PDF files into images using Golang • Send an image to Google Cloud Vision and store returned data using Golang	On time Secure Full functionality Easy to use/read Essential Requirements met	Mandatory
Golang program able to receive json data returned from file upload to Google Cloud Vision	• Send an image to Google Cloud Vision and store returned data using Golang • Send multiple files to Google Cloud Vision at once and store the results	On time Secure Full functionality Easy to use/read Essential Requirements met	Mandatory
SQL database integrated and working with Golang program	• Convert Golang application to multi-threaded version • Send multiple files to Google Cloud Vision at once and store the results	On time Secure Full functionality Easy to use/read Essential Requirements met	Mandatory
Sprint reviews and reports completed and documented	• All sprint retrospectives/reviews	On time Essential Requirements met	Mandatory
Cloud Vision json data stored into database using Golang program	• Send multiple files to Google Cloud Vision at once and store the results • Send an image to Google Cloud Vision and store returned data using Golang • Convert Golang application to multi-threaded version	On time Secure Full functionality Easy to use/read Essential Requirements met	Mandatory
API research/creation completed for Golang program	• API Documentation • Implement API into project • Complete API tests	On time Secure Full functionality Easy to use/read	Mandatory

		Essential Requirements met	
UI/UX front end mock-up satisfies RedEye's/clients requirements	● UI/UX Research ● UI/UX Mock-up of front end	On time Easy to use/read Essential Requirements met	Desirable
UI/UX front end integrated with main program without security issues	● Implement program front end solution ● Unit test front end solution	On time Secure Full functionality Easy to use/read Essential Requirements met	Desirable
All major software deliverables have been security tested, reviewed, and unit tested	● All sprint unit tests ● Monitor program performance and ensure tool reliability	On time Secure Full functionality Easy to use/read Essential Requirements met	Mandatory
Solution delivered on time, and within budget	● Complete project ● All sprint retrospectives/review s	On time Secure Full functionality Easy to use/read Essential Requirements met Within budget	Desirable
Post project report/review completed and discussed	● Post project report and review	On time Essential Requirements met Review discussed	Mandatory

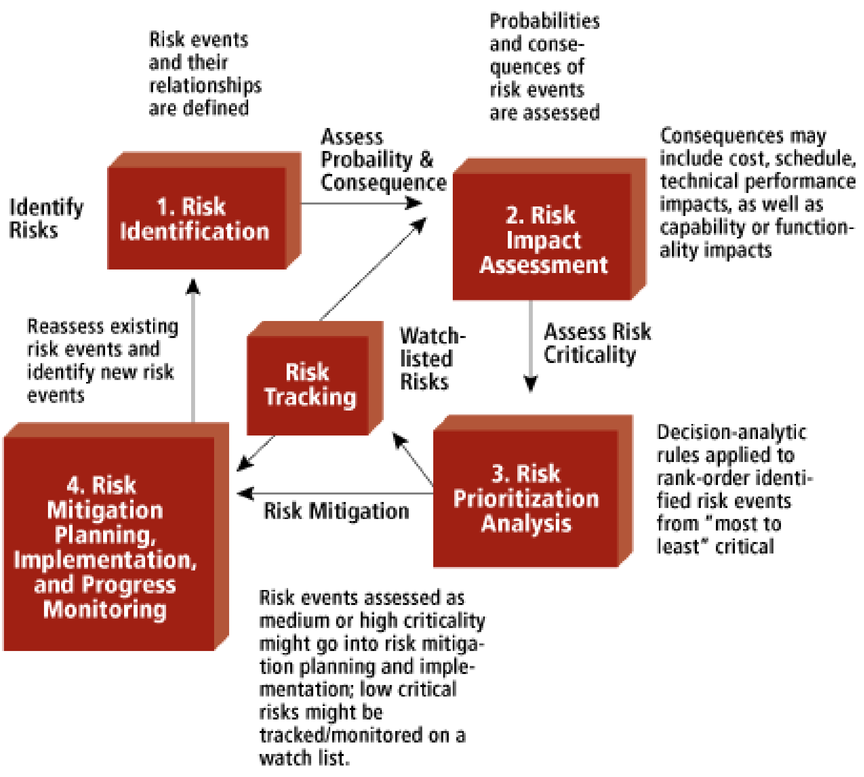
1.3.3 Risk Management

As with most software development project undertakings, the main areas for risk are in the managing the time, scope, and financial requirements. This is because it can be extremely hard to estimate the actual resources required to make the task at hand feasible to complete which also makes it crucial to try and get as accurate an estimate as possible.

A variety of techniques have been used to identify all potential risks in this project. They are: brainstorming - getting a group of individuals from various technical backgrounds (business, IT, management) to think about the various potential risks; using historical information from previous projects and the risks that arose there; and SWOT analysis - where the strengths, weaknesses, opportunities and threats of the project are identified by RedEye team members with varying backgrounds.

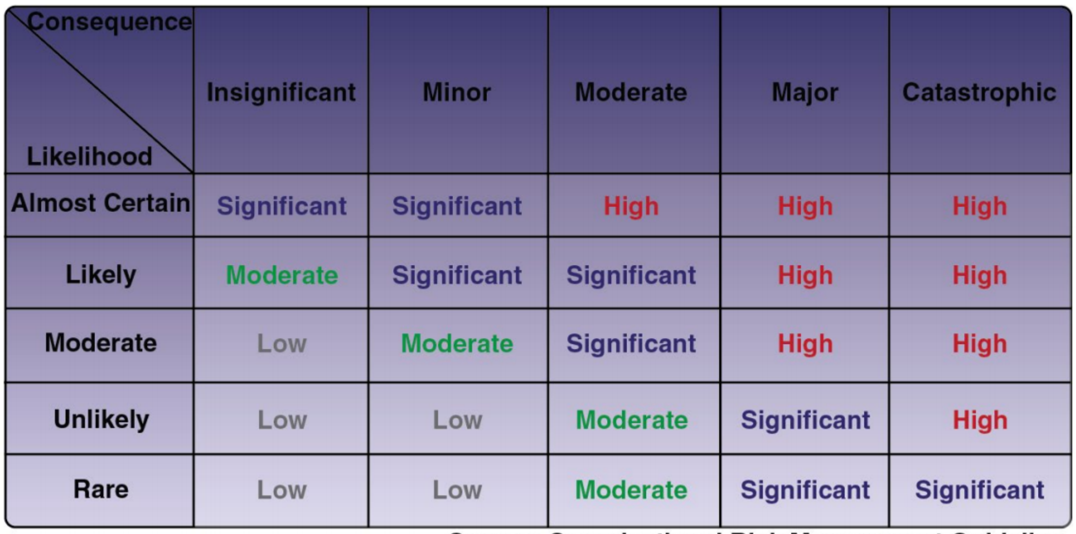
Below is a diagram (Figure 8) of how the risks were identified, analysed, assessed and documented.

Figure 8 (Risk identification process) [CITATION Pro13 \l 1033]


Risk events and their relationships are defined
Probabilities and consequences of risk events are assessed
Identify Risks
1. Risk Identification
Assess Probaility & Consequence
2. Risk Impact Assessment
Consequences may include cost, schedule, technical performance impacts, as well as capability or functionality impacts
Reassess existing risk events and identify new risk events
Risk Tracking
Watch-listed Risks
Assess Risk Criticality
4. Risk Mitigation Planning, Implementation, and Progress Monitoring
Risk Mitigation
3. Risk Prioritization Analysis
Decision-analytic rules applied to rank-order identified risk events from "most to least" critical
Risk events assessed as medium or high criticality might go into risk mitigation planning and implementation; low critical risks might be tracked/monitored on a watch list.

A risk matrix (Figure 9) has been used on the identified risks to prioritise the risks based on their potential impact, and chance of occurring.

Figure 9 (Risk impact matrix) [CITATION Pro13 \l 1033]



Consequence / Likelihood	Insignificant	Minor	Moderate	Major	Catastrophic
Almost Certain	Significant	Significant	High	High	High
Likely	Moderate	Significant	Significant	High	High
Moderate	Low	Moderate	Significant	High	High
Unlikely	Low	Low	Moderate	Significant	High
Rare	Low	Low	Moderate	Significant	Significant

Source: Organisational Risk Management Guideline

After analysing the potential for each identified risk, each risk was reviewed and ideas were brainstormed, along with using historical analysis of similar risks, in order to create avoidance strategies and potential responses to these risks.

A planned response has been made for each potential major risk and the monitoring strategy that will be used during the project involves using some of the previously stated risk identification techniques and re-analysing the priority list of risks after each sprint and updating/adding the values accordingly.

The main potential risks within this project are:

ID	Category	Risk Description	Probability	Impact	Avoidance Strategies	Response
R1	Schedule	Project Delivery Late	Moderate	Major	Constant monitoring of project scope, change and risk	Agree on new project delivery date, with appropriate checks in place
R2	Schedule/Management	Schedule and resource delays	Moderate	Major	Constant monitoring of project scope, and resource allocation	Agree on new project delivery dates, and split resource allocation

						using 80/20 rule
R3	Design/Scope	Specification requirements changes	Likely	Major	Break down project and requirements into as much detail as possible	Perform change control analysis and update scope/timeline of project if necessary
R4	Design/Management	Data loss, or security risks with testing development builds of project	Moderate	Catastrophic	Make sure all potential security issues are identified and put security checks in these locations	Remove parts of program with security risks and rewrite them to fix these issues
R5	Communication	Miscommunication between individuals/groups about tasks/requirements/updates	Moderate	Major	Schedule and plan communication methods and meetings	Have communications review meeting to understand what is going wrong
R6	Scope/Budget	Project requirements go out of scope/budget, or are not feasible to be completed	Moderate	Major	Set clear budget and scope restrictions and change management control measures	Reassess current schedule, scope and budget and realign the current scope and budget to fit

						requireme nts

1.3.4 Change Control

The change control process for this project involves a management hierarchy that will assess and manage each change proposal.

If a change needs to be requested, there is an order as to who must be informed before the change can go through. If I find that there is a change required during a sprint or after a sprint review, I must alert Nicolas Dijoux (my industry supervisor) who will either acceptor decline the change or refer the potential change to Nathan Dench (RedEye DMS team leader). Nathan Dench will review the change and either accept or decline it or refer the potential change to Wayne Gerard (RedEye CEO) for final review. If a change has been acknowledged and approved that was not requested by myself, it follows down the same chain to me. The team and project workflow and sprint log is updated to reflect the approved changes.

The main way to identify potential needs for change during the project will be through each sprint review and retrospective. Each sprint review and retrospective will break down how all the work during that sprint progressed, what was good/bad, and what needs to be, or can be improved, removed, or altered.

To determine whether or not a change is feasible or necessary we go through these questions: what is the impact of the change on the project; is the change necessary for project completion; does the suggested change fall within the scope of the project; what are the benefits of the change; what is the ease of implementing the change; how long will it take to implement the change and how will it affect the completion time of the project; what are the costs of the change (time, finance, and other costs); and what are the risks associated with implementing the change.

2 POST-PROJECT REVIEW

2.1 SUCCESS CRITERIA

Success Criteria	**Deliverables/Objectives**
SQL Database created using PostgreSQL and SQLBoiler and has capability to store all relevant data returned from Google Cloud Vision	• Design and create SQL Database • Create document and pages tables for the database in Golang • Store sample Cloud Vision data into database

This criteria was the basis of the whole project and needed to be completed and data secure before any other work on the project could commence. The design of the database was discussed and documented between myself and Nicolas Dijoux. All the initial database requirements were mapped and created as an SQL database. Unit tests were created to store sample Cloud Vision data and had a coverage of 100% for the code tested. All aspects of this success criteria were completed.

Success Criteria	**Deliverables/Objectives**
Golang program able to send files to Google Cloud Vision	• Set up Google Cloud Vision account • Split and convert PDF files into images using Golang • Send an image to Google Cloud Vision and store returned data using Golang

The next stage of the project required the ability to send data from the program to Google's API and Cloud Vision service. A Cloud Vision account was set up along with an associated API key in order to connect to Google Cloud Vision from the Golang program. A PDF splitter was implemented using the ImageMagick external library to convert PDF files into individual images that can be uploaded to Google Cloud Vision. Unit tests involving PDF splitting, image uploading, and data storing were created and run with all tests passing. The success criteria was successfully met.

Success Criteria	**Deliverables/Objectives**
Golang program able to receive json data returned from file upload to Google Cloud Vision	• Send an image to Google Cloud Vision and store returned data using Golang • Send multiple files to Google Cloud Vision at once and store the results

This success criteria involved successfully parsing the returned data from an uploaded file to Google Cloud Vision in order to split relevant pieces of information into the appropriate section of the database. Structs were created in the program to parse the JSON data returned from Cloud Vision. These structs were called to parse the data when returned data is unpacked. The parsed data is then stored into the database. Multiple file uploading and the ability to parse and receive multiple Cloud Vision responses was implemented into the project using batch request uploading and receiving. These deliverables were unit tested extensively until 100% coverage was reached. All success criteria objectives were met.

Success Criteria	**Deliverables/Objectives**
SQL database integrated and working with Golang program and convert program to multi-threaded application	• Convert Golang application to multi-threaded version • Send multiple files to Google Cloud Vision at once and store the results

This success criteria was about increasing the efficiency of the program by converting it to a multithreaded version so that multiple requests can be sent and handled at once. Go routines and functions were added to the parts of the application that sent and received data from/to Google Cloud Vision so that multiple files could be sent and multiple responses could be received at the same time. This functionality was extensively tested using unit tests in order to see if all files were sent, all data was received, and all data was being correctly stored in the database. Unit tests showed full coverage for this section of the program. All success criteria deliverables were met.

Success Criteria	**Deliverables/Objectives**
Sprint reviews and reports completed and documented	• All sprint retrospectives/reviews

Sprint objectives and deliverables were documented and reviewed between myself and Nicolas Dijoux after each sprint in order to update and create the finalised objectives and deliverables for the next sprint. Whilst the sprints were outlined from the start, multiple revisions and updates needed to be made throughout the project due to project stalls, problems, and other unforeseen circumstances. Clear communication was had between both parties and all of the necessary deliverables and objectives required for this success criteria were achieved.

Success Criteria	Deliverables/Objectives
API research/creation completed for Golang program	• API Documentation • Implement API into project • Complete API tests

An API needed to be created for the program so that users can connect to a remote server and send requests to the program and the program can complete tasks based on these requests. The API acts as a middleman between the user and the program so that the user cannot directly access, affect and modify the underlying program. An API was developed using the Gin Gonic external library, creating a RESTful API meaning it allows for HTTP requests to GET, PUT, POST and DELETE data. Middleware within the API is created to bridge the functionality and logic of the program to the requests given. The created API was then extensively unit tested to make sure all previous functions were still running as expected and produced the correct results. All the API deliverables and objectives were completed, meeting all the given success criteria.

Success Criteria	Deliverables/Objectives
UI/UX front end mock-up satisfies RedEye's/clients requirements	• UI/UX Research • UI/UX Mock-up of front end

An initial mock-up of the UI/UX was required before it could be created and implemented with the program. Using Adobe Photoshop and GIMP, a mock-up design was created to be implemented using HTML5 and CSS3. Functionality was the primary concern over form as the main aspect of the project was the transferring of data between the user of the program and RedEye. With this in mind, the design process was not extensive and focused on the essentials that the user will require when interacting with the program. The design process involved myself, Nicolas Dijoux, and Jared Sager (UI/UX Specialist at RedEye). All design and UX requirements were met.

Success Criteria	Deliverables/Objectives
UI/UX front end integrated with main program without security issues	• Implement program front end solution • Unit test front end solution

The design of the previous success criteria was implemented using HTML5 and CSS3. JQuery and Ajax were used to allow for requests between the web pages and the program's functions. The UI/UX was tested by myself, Nicolas Dijoux, and other RedEye team members to make sure it worked correctly on multiple devices. The UI/UX success criteria was fully met.

Success Criteria	Deliverables/Objectives
All major software deliverables have been security tested, reviewed, and unit tested	• All sprint unit tests • Monitor program performance and ensure tool reliability

Benchmarking and performance tools were created to analyse the operations of each function of the SmartEye program. These tools analysed the failure rate, upload rate, and overall speed of each aspect of the program. Along with these tools, unit tests were created to test each function of the program to ensure full security and reliability of the code. All unit test and benchmarking deliverables were completed, with all success criteria being met.

Success Criteria	Deliverables/Objectives
Solution delivered on time, and within budget	• Complete project • All sprint retrospectives/reviews

The final SmartEye project was completed on time and within budget. All sprint retrospectives were completed and reviewed with data to be used for future projects.

Success Criteria	Deliverables/Objectives
Post project report/review completed and discussed	• Post project report and review

The project report and post project review have been completed and submitted on time. Reports have been reviewed and checked by myself and Nicolas Dijoux for consistency and accuracy.

2.2 ESTIMATE REPORTING

2.2.1 Analysis of planned vs actual schedule

This project consisted of six, two week sprint periods which contained various amounts of tasks based on the estimated effort required to complete the tasks. The planned schedule document shown in 1.1.7 accurately portrays how the project progressed throughout its creation. There were very few issues that occurred during the project that caused any major variances in the project schedule.

The first few weeks in Sprint 1 focused on the creation of the initial project report and initial planning, scheduling and estimating for the project. (As seen in 1.1.7 tasks 5-10). Initial database design and testing was also a part of this phase. The first sprint was completed with no issues or hiccups in scheduling. Sprint 1 started on 27-02-17 and concluded on 17-03-17.

Sprint 2 focused on creating the basis of the project in Golang, and creating a basic application that could connect to Google Cloud Vision and be able to send and receive data using Google API requests. A PDF splitter was also created as Google Cloud Vision only accepts images to be sent so document files had to be split and converted to image files. Sprint 2 was completed as planned with no interruptions. Sprint 2 started on 20-03-17 and concluded on 31-03-17.

Sprint 3 involved increasing the efficiency of the current program by converting it to a multithreaded application that could send and receive multiple pieces of data at once. This consisted of researching Go Routines and Go Channels which is how Golang implements multithreading and interprocess communication. After the research was implementing and testing the Go Routines/Channels in the current program so that images sent to Google Cloud Vision would be split into batches of multiple files to send at once, and then parsing the returned batch responses correctly and storing the returned data. The first week of this sprint was focused on research and initial creation and testing of the Go Routines and Channels whilst the second week was bug testing and finalising the multithreaded parts of the SmartEye application. During this process, the research and initial conversion of the application to be multithreaded took a day longer than expected. This was due to multiple problems with Gogland IDE and how it was running the multiple processes when the main program was executed. It was inconsistent as to where it would store the data in memory and updating the program solved this issue. During the bug/unit testing stages of each sprint, a slack day is allocated to recover from situations like this. This slack day ended up not being required due to all bug and unit testing being completed in a single day. Sprint 3 started on 03-04-17 and concluded on 14-04-17.

Sprint 4 was planned to include the research and creation of the API for the SmartEye project along with the research and mock-up of a front end solution for the user for SmartEye. However, during this sprint there were unforeseen circumstances that forced some tasks to be spread over the following sprints as they could not be completed in time. This was due to myself being ill and bedridden from 16-04-17 to 22-04-17. Due to this, I caught up on the missed days of work during that week by going into the RedEye offices an extra 2 days in the following week. This allowed me to mostly catch up on the work that I had missed and not let the project schedule increase in time. During this week, myself and Nicolas Dijoux went through all the tasks of the sprint to make sure all priority tasks would be completed with the others being pushed back into later sprints. We reviewed the project and sprint schedules and determined that the UI/UX tasks could be pushed into Sprint 5 as I would be able to get assistance in this area from the UI/UX expert in RedEye, Jared Sager. This change was updated and reflected in the project schedule and Sprint 4 review. Sprint 4 started on 17-04-17 and concluded on 28-04-17.

With the circumstances that occurred in Sprint 4, Sprint 5 now consisted of researching, creating and implementing the front end solution of the SmartEye project that the clients/users will use. This involved using Bootstrap, HTML5, CSS3, and jQuery to create a web application that would be able to send requests to the server via the API that was created for SmartEye. As the main aspect of the project was the functionality of the program, the UI aesthetic was not focused on during this part. A UI base was created using Bootstrap and standard UI principles and methodologies. The integration of the web app to the SmartEye API to correctly send and receive request responses was the most important part of this sprint. The first week of this sprint dealt with getting the web app set-up whilst the second week focused on unit and bug testing the web app to make sure everything was working correctly and securely. There were no issues that caused any delays in the project schedule during this sprint. Sprint 5 started on 01-05-17 and concluded on 12-05-17.

Sprint 6 was the finalising of the project, making sure everything was completed and up to RedEye standard, along with completing the project reports and post project reviews. All sprints were successfully completed with all deliverables and objectives being finished. The project was completed on time and within budget. Sprint 6 started on 15-05-17 and concluded on 26-05-17.

2.2.2 Analysis of planned vs actual size

Due to extensive initial project planning and reviewing, the final project scope and size was similar to the estimated project scope and size. There were a few small changes that had to be made to the sprint scheduling due to illness on my part and there was a small change in the requirements for the UI/UX of the web application of the SmartEye application, but there were no major changes that were required.

The sprint scheduling change due to my illness went through the process of meeting between myself and Nicolas Dijoux to identify the problem, figure out solutions to the problem, and help adjust the project schedule to cater for the planned changes. Nicolas Dijoux then notified Nathan Dench (DMS RedEye team leader) about the changes.

As this change was caused by unforeseen circumstances and adjustments had to be made due to lost time, the full change identifications methods were not required. Only the project scheduling and reviewing aspects were needed.

With the UI/UX requirements changes, during Sprint 4 when it became aware to us that we would not have time to start the research and mock-ups for the UI/UX, myself and Nicolas Dijoux set aside time to identify and review what could be done to mitigate this issue so that the project could be finished on time. During the problem identification and review, potential solutions were discussed such as reducing functionality requirements, or getting Jared Sager (RedEye UI/UX Expert) to complete the initial UI/UX development. The final solution that was implemented was the reduction of UI aesthetic and form requirements, prioritising application functionality as my project was mainly software development based. Along with this, the UI/UX aspects were pushed into Sprint 5 as Sprint 5 and 6 tasks were mainly the finalising of the SmartEye project which did not have high estimated effort values.

These changes were successfully made and the project was able to be completed on time whilst still passing all of the project success criteria.

2.2.3 Analysis of planned vs actual effort

The planned effort described in section 1.1.6 and figures 5 and 6 shows that the project had a total estimated story point count of 206 points with each sprint varying between 23-59 points. The initial planning phases also required an adjustment to the RedEye working environment which was not documented as a task in itself so a smaller set of story point tasks was allocated to that time.

Sprints 5 and 6 were also given smaller story point tasks as a precaution for any work/project changes or scheduling problems. This would allow for story points that could not be completed in previous sprints, that were still required, to be pushed into those sprints whilst still being able to finish the whole project on time.

Sprints 2, 3 and 4 had the largest amounts of story points allocated to them as those sprints would be in a period where I had adapted to the work situation at RedEye and would be used to the amount of work required to complete given tasks. Sprint reviews and retrospectives were extremely important during this time as I had to make sure that I was be on top of the current work/project situation and that Nicolas Dijoux would be confident in my ability to keep working at this current rate to finish the project on time.

As stated in section 2.2.2, in Sprint 4 there were UI/UX requirements changes that were necessary due to myself being ill. This meant that 10 story points of effort from Sprint 4 were shifted into Sprint 5. This process went smoothly due to the initial story point allocation allowing story point overflow to push over into Sprint 5 and Sprint 6 as they had smaller story point tasks scheduled initially.

The estimated story point efforts estimated for each task and sprint were accurate in comparison to the final project outcomes. Each sprint was completed on time, successfully meeting all required deliverables, objectives and success criteria. Sprints 1 and 3 were completed 1 day earlier than expected allowing for sprint retrospectives and reviews to be started earlier. All other sprints concluded on their scheduled finish dates.

2.3 REVIEW OF TECHNIQUES

Techniques	Review
Meetings	This technique was mainly used during initial report planning and estimating, sprint planning, sprint reviews and retrospectives, team gatherings, and the final report review. Meetings were effective in updating the members involved of the status, progress, and future work. They were also used for code and project reviews to make sure everything is up to RedEye standard and to make changes if necessary.
Code reviews	Code reviews were used for every project update or added function. My code could not be added to the master repository until it was reviewed by Nicolas Dijoux. This was to ensure code quality and consistency. This was effective as it kept my code standard high and it could be easily read and used by the other RedEye team members.
Brainstorming	Brainstorming was used during meeting, project planning, and code/project problem solving. It was useful to help aid myself in finding solutions to unorthodox problems that others had not faced before. Brainstorming with others also helped with this as it would add an additional perspective to the problem giving me a different view on the issue, potentially aiding in finding solutions to problems that I could not.
Historical analysis	Historical analysis was used during the initial project planning and estimating and during project problem solving. It was useful in the initial project planning and estimating phase as it helped to get more accurate estimations of the effort and time requirements to complete some tasks along with setting a standard for report elements and requirements. This was done by comparing previously completed tasks and their details, to the current planned tasks. Using it during project creation helped myself and Nicolas Dijoux solve problems if RedEye had faced similar issues before, as a similar approach could be used to solve the current issues.
Risk management	Risk management was used in the initial planning phase and throughout the project's creation to deal with any issues that arose. Some risk management techniques used were: creating a risk register of the top potential risks that could happen; a risk matrix to map the potential risks to; SWOT analysis to describe the strengths, weaknesses, opportunities and threats of a situation; brainstorming potential risks; and using historical analysis of previous projects for risks that have occurred

	before. Risks were identified, assessed for their potential impact, prioritised by the most likely/highest impact, and mitigation/avoidance strategies were created for each risk. This allowed for myself and RedEye to be as prepared as possible for any risk or problem that occurred during the project's creation.
SWOT Analysis	SWOT Analysis was used during the risk and change management phases of the project. It was used to identify the pros and cons for given situations or prospects to see if we should take that approach or find a different solution. SWOT was useful to eliminate various brainstormed solutions as their negative consequences heavily outweighed the positives that would be gained.
Risk Matrix	The risk matrix was used to place each potential risk on a chart based on their impact on the project, and the likelihood of the problem occurring. This was extremely useful as it allowed us to prioritise our risk register and focus on creating solutions to and mitigating damage from the most important/likely problems first. Problems that were not prioritised would receive less attention as their impact or chance of occurring could be worked around or would not affect the project in a major way.
Risk Register	The risk register was used to keep a running list of the current most likely/highest impact problems that could occur during the project's creation. This register was consistently updated as the risks involved could change at any point during the project. This allowed us to be as ready as possible for the most likely issues that could occur.
Change management	A change management plan was put into place with a management hierarchy and review process created to organise any requested and implemented changes to the project. This was useful to keep track of all changes proposed and to have a systematic process on the way changes would be reviewed and accepted or declined, and then implemented if they are accepted.
Client testing	Client testing was a technique used for quality assurance, testing, and making sure the program was meeting their needs. It was useful in helping us determine the aspects of the program that had bugs/issues, or required further working on and fixing, along with the features that were working as intended and were appreciated by the users.
Change reviews and requests	Change requests and reviews went through a hierarchy management process for approvals. This was useful in order to determine the necessity of the suggested changes and it also provided multiple opinions and points of views on the changes as each member in the hierarchy chain has a different role/expertise within RedEye. The reviews and requests made sure only changes that are required or would improve the project with minimal project schedule/budget impact are implemented.

Unit testing	Unit testing was extensively used throughout the creation of the SmartEye project. It was extremely important in order to test the reliability and security of the code after each function is added or modified. Program stability, reliability, security and speed are key aspects of RedEye software so quality assurance and unit testing are always performed. Unit testing on the code is performed by myself and Nicolas Dijoux to ensure that the program works the same in multiple environments.
Product backlog	The product backlog is a technique used in agile software development as a means to show tasks that are still required to be completed and in what order they will be completed. They were used in this project during each sprint iteration to list the tasks that are currently being worked on and what is going to be worked on next. They were also used when pushing back the UI/UX tasks from Sprint 4 to the product backlog of Sprint 5. They were useful in helpful myself and Nicolas Dijoux keep up to date with the current progress of the SmartEye data extractor.

2.4 QUALITY ACHIEVEMENTS

Quality Measures

Deliverables/Objectives completed:

Each sprint had objectives and deliverables that needed to be completed and all tasks were completed during their planned sprint schedules apart from the UI/UX tasks in Sprint 4 which were pushed back into Sprint 5 and completed there. Each deliverable/objective was reviewed by Nicolas Dijoux, other RedEye team members, and clients to make sure it was of acceptable standard for RedEye before it could be marked as completed and I could move on to the next set of tasks.

Completed on time and within budget:

Each sprint had an allocated period of two weeks and had a set amount of tasks and budget to that needed to be completed and adhered to. The project started on 27-02-17 and was scheduled to be completed on 26-05-17. The project was successfully completed by the scheduled due date, having been finalised on 25-05-17 with final reviews occurring on 26-05-17. Each sprint's tasks outlined in figures 5 and 6 were completed during that period of time, excluding the UI/UX tasks from Sprint 4, which were completed during Sprint 5.

Code easily readable and reusable:

The SmartEye program code that I developed was consistently reviewed by Nicolas Dijoux and other RedEye DMS team members to ensure that it adhered to their standards and that the code was structured, commented, and easily readable and could be reused by any team member. This process was done by myself submitting pull requests through GitHub to the master branch, and the RedEye team members would look over my pull requests and add comments to my code suggesting changes. Once my code was up to standard, it was merged with the master branch. This ensured that all parts of my code was of high quality.

Code/Program is secure:

As the program deals with storing and retrieving data from a database, it must be secure from SQL injections or attacks. All potential user inputs and database queries were sanitised to stop this from occurring. The server also accepts file uploads and to stop potential security threats from the uploaded files, files are scanned and file sizes and file types are restricted.

Program is reliable and fast:

Benchmarking tests were performed at each stage of project development to make sure it was able to handle mass queries/tasks whilst still performing at a speed that RedEye considered fast for the users/clients. Golang and Gogland IDE has built in benchmarking tools that were used for this. The benchmarking tools tested both speed and reliability of the code making sure that everything is working as intended and passed all tests.

Program meets clients requirements:

The client requirements that were elicited in the initial project plan report were described as the deliverables, objectives and success criteria in sections 1.1.2, 1.1.3, 1.1.4 and 1.3.4. Users/clients were satisfied that each of their requirements that was initially stated were met during quality assurance and client tests. These tests were given to users/clients through out the project but were emphasised during the final stages of the project when the web application was built.

Communication standards high:

Communication between myself, Nicolas Dijoux, and other RedEye team members was frequent using multiple communication avenues such as instant messaging services, meetings, and informal conversations. This kept everyone informed of the status of the project along with the changes/updates that were required before the next task can be started or my project could be merged to the master repository. This was important as it helped to reduce any potential obscurity or questions about the project and its progress. Communication was also high between RedEye and potential users/clients of the project with regular feedback received from the users/clients as to their opinions on the product and if their requirements are being met.

Risks minimised:

No major risks arose during the project. The only minor risk that occurred was myself being ill for a week, causing the UI/UX tasks from Sprint 4 to be pushed into Sprint 5 where they were completed. This risk was quickly averted by following the risk management protocol that was created by having this situation mapped out in our risk priority register with solutions already suggested along with alerting Nicolas Dijoux and other RedEye members of the situation that was at hand.

2.5 LESSONS LEARNED

What did you learn?

In my time at RedEye apps working on the SmartEye data extractor tool there were many lessons that I learned about myself and work ranging from general work life, to my mentality.

Working in a team

As this is was the first team I had worked in in an IT business environment, it took a small period of time to adjust to this new setting. Interacting with the developers, businessmen, sales teams, project managers and more each required a different approach as everyone had different technical backgrounds. This meant that I had to be aware at all times to the audience that I was speaking/presenting to. The same situations occurred when dealing with external clients and users of RedEye projects and SmartEye. Being able to adapt quickly in these situations was a very important skill that I acquired during my time at RedEye.

Start up environment

The start up environment is very different to large, established, company environments. Start up companies have smaller teams with individuals usually performing multiple roles within the company. Along with this, the hierarchy structure is mostly flat and everyone speaks to each other as equals, allowing for more work to be done as ego does not play much part in the workplace.

What I learned from this is how important each person's role is within a smaller company and that consistent communication between team members is required at all times in order for everything to function at its optimum. In an environment like this it also emphasised that the workplace can be a mixture of serious and fun surroundings, allowing for team members to recharge when required but also work hard when necessary.

Sprints and scheduling

This project used an agile SDLC model that utilised sprints for its workflows during the project's lifetime. The sprint model helped me with scheduling and breaking down certain tasks and their feasibility to be completed within the allocated time. I learnt how much work I could complete within each two week schedule along with always keeping track of progress and updates for sprint reports and reviews so that at the end of each sprint, we could learn from the good and bad allowing us to improve for subsequent sprints.

Project estimation and Task efforts

Having significant initial planning allowed us to more accurately estimate the project timeline and individual task efforts. Using this along with the risk management control methods that were set up, the project was able to be completed without any major issues occurring.

This showed me the absolute importance of getting the initial planning reports as accurate as possible. It helped to reduce stress and major workload changes throughout the project's progression which allowed the project to be finished on time and within budget.

Unit and bug testing

The team members at RedEye showed me the importance of always unit and bug testing as much as possible on all new pieces of code as any minor issue that may not occur at one time, but could potentially do so at another, could end up significantly affecting the system it is integrated in. Unit testing for a minimum of 75% coverage was a requirement to make sure that the code is as reliable as possible.

Researching solutions, libraries, languages etc

As I was using Golang for my project, it was a challenge to try and find solutions to problems that occurred as there is minimal documentation, error support, and code support due to it being a newer and not widely used language. This meant I had to learn and adapt quickly in order to find solutions to obscure problems or program issues as it was unlikely that the issue had been documented before. Brainstorming, trial and error, and getting others opinions helped in these situations.

I also learnt that extensive researching is crucial in finding the best routes to take when choosing the correct external libraries to use, and the most intuitive way to write an API to connect my program to a web server. Researching the pros and cons for both of these problems allowed me to choose the correct libraries for my programs functionality, and write my API to be easily connected to and used by other RedEye team members.

Meeting requirements/deliverables

Due to having short sprint times it was crucial to make sure all requirements and deliverables were met otherwise the whole project's schedule could quickly spiral out of control. Not meeting requirements/deliverables would force these tasks to be pushed back into other sprints, increasing the workloads of the subsequent sprints. I learnt that it was always important to keep on time and to consistently communicate progress with other team members so they know where you are up to and if you are struggling, they can help in some way.

Code reviews

Working with the RedEye DMS team helped me with understanding what are some better practices when it comes to reviewing and setting out code. After each major function for the project was completed, myself, Nicolas Dijoux, and a few other RedEye team members would sit down with each other and analyse the code to see what was done well, what could be done better, what needs to be changed, and what can be merged with master. This process helped me with each new iteration of code as I knew what elements they were looking for, and what they did and did not like due to inefficiency, it being more difficult to integrate with the main system, or it being harder to read.

Taking feedback on board

With the consistent communication channels, code reviews and sprint retrospectives and reviews, I received numerous amounts of feedback from many different people both good and bad. I learnt the importance of keeping a clear mind and realising that every bit of feedback should be looked at logically to analyse what needs to or does not need to be changed. The temptation to take negative feedback poorly can be high so it is crucial that you view it as a chance to improve and that they are just trying to help you do so. A goal of mine throughout this project was to improve myself and my code and I believe this helped me achieve that.

What would you do the same again and why?

Extensive planning

When the project first started, I questioned Nicolas Dijoux and other team members as to the amount of estimation and planning that we were completing before the project started. By the end, I was extremely glad that we put that much time and effort into the initial project report and planning as the estimations for time and effort were extremely accurate and which allowed myself and the others to have confidence that we would be able to finish and meet all our deliverables, success criteria, and objectives on time and within budget.

The planning period lasted a few days and involved a variety of different techniques such as brainstorming and SWOT analysis. The planning also included creating risk and change management protocols that would be taken if there are any issues or changes required to the project. This helped when I was ill during the project, allowing me to push back a few tasks from Sprint 4 into Sprint 5.

Had we not done this extensive planning, my illness could have caused major time and scheduling issues for the project due to missed deliverables/objectives in a sprint due to not having solutions created if a situation like that occurred.

Heavy researching

Various aspects of the project had to be researched heavily to make sure the best options going forward were chosen. This included: deciding on what external libraries to use; what API format to use and what is required in the API; best IDE and resources to use for making the program and running the database/server; integrating and using Google Cloud Vision and Google's API; what techniques to use for change/risk management; and what communication methods and scheduling is best for the team.

Choosing the correct external libraries to use is extremely important as you have to make sure that the external library covers your needs, is easily readable and implemented, has documentation for its use, and has a long life cycle and will not be removed in the near future allowing us time to build our own custom library to perform the functions required.

API designs can be extremely varied and what was chosen was a RESTful API approach which allows sending requests to and from the web server to our program. This was the easiest and most relevant API model to use with the SmartEye data extractor tool as the tool only requires the user to upload files and make requests on the SmartEye database.

Using the correct IDE and resources is important to make your development life much simpler and easier as they allow you to focus on your coding rather than the platform/resources/interfaces you are using. RedEye had completed significant researching on this matter prior to make sure that I would be able to get the project started as quickly as possible.

It was important to fully understand the components of Google's API and Google Cloud Vision that I would be using so that I could utilise those elements efficiently within my program. Researching this was crucial in my understanding of how data was sent and received to Google's services so I could code my project in a way that would be able to encode and decode requests and responses in the correct manner.

Identifying potential risks and getting the risk and change management procedures correct is a vital process in the initial planning of any project. Extensively covering these aspects allows you to be as prepared as possible for any unfortunate circumstances or unforeseen changes that can occur during the project. This is something that should always be done in any project of any size as it can help save a project from potential disaster or failure.

Planning out communication methods between team members is important in ensuring maximum cohesion between team members as everyone is updated of current tasks and project progress and issues can be caught earlier as everyone is aware of what is happening and what each person is doing. Planning these communication methods helped with setting up and scheduling reviews, retrospectives, meetings, and general conversations about the project between myself and other team members.

For all these reasons stated above, heavy researching before, during and after the project is extremely useful and I would do the same thing again on future projects that I undertake.

Significant unit/bug testing

Unit testing is something that should always be done in programming based projects but in the past I have been sluggish with keeping up with unit testing on each major function of my projects.

RedEye and Nicolas Dijoux required me to write unit tests for each function of my project, emphasising the importance of having reliable code in a project like this as it will be integrated into a much larger system and any unreliable section of code can cause the whole system to fail.

Whilst creating unit tests and bug testing can take up a large portion of time, it is better to take that time in the beginning to help reduce potential serious consequences that could come from the project having problems at a later point in time.

Code reviewing

Code reviewing was set up for each pull request I made in GitHub to merge my code with the master repository.

The code reviewing process consisted of myself, Nicolas Dijoux, and other RedEye team members going over my code line by line making sure that what I was writing was up to RedEye standard and could be easily read and implemented by the other team members.

This helped to ensure the consistency and quality of my code as they would let me know of what I could improve on and why, allowing me to make the required changes and write future code in that manner.

The code reviewing process was extremely useful in helping me learn what other developers look for in code and how I should try and develop my code in future situations. Code reviewing is something I will use in the future as everyone can always improve their code, and the coding style of different companies can vary so consistency checking is a must.

Efficiency monitoring and benchmarking

Efficiency monitoring and benchmarking were tools that I used throughout to test my code for its reliability, security, speed, and performance. They were extremely useful in helping me determine problematic functions that caused issues in any of those areas and also in helping me decide between different ways to solve problems as they would give different benchmark results in those areas.

Efficiency monitoring and benchmarking is something that I will always use in any project to make sure my program is running at its optimum. It is also extremely useful for charting and displaying results to other team members and users/clients, giving them viewable insights into the program.

What would you do differently, how and why?

More organisation between myself and Nicolas Dijoux for working days

During this project I had organised my working days for each week in the week prior to let Nicolas Dijoux know when I would be coming in and potentially requiring his assistance on the project. Unfortunately there were a few cases where I did this and he replied with the affirmative but did not show up on those days due to prior obligations already organised.

In these circumstances, I should have asked initially if he would also be in on the days that I notified him that I would be coming in, but Nicolas Dijoux should have also informed me of his prior obligations so that I would not potentially waste my time by not being able to continue working due to being at a roadblock, requiring his help to continue.

Get more involved with other teams of RedEye to know all their products and external team members more

Whilst working at RedEye, I mostly spent my time working with and around the DMS team which is only one section of the whole RedEye team. Due to this I did not get a chance to get to know many members of RedEye outside of my team and fully immerse myself in the RedEye culture.

In order to get to know the whole workplace and environment better, in the future at any other workplaces that I enter I will make sure to go around, learn, and understand what the other team members are doing which will help me embrace the work's culture along with familiarising the rest of the teams of my presence/personality and to also increase my business and social network for potential future opportunities.

Increase documentation of sprint reporting/retrospectives

Sprint reporting and retrospectives were completed at the end of each sprint during the project. The reports and retrospectives documented the work that was done, how it was done, what went well, what did not, and what could be improved for the next sprint.

Everything was documented extensively for the report side, however in the retrospective, the review of why things did not work as well as we believed they should have and the lessons learnt from that were not detailed enough. To me, the why is important in understanding the reasons for things being changed and to be able to make the necessary improvements in the future. The RedEye processes and documentation mainly focused on the how and what can be improved without going through the why in detail. I believe that going through why things should be done would help with future projects and endeavours as it gives you the full context to the previous situations and from there you can adapt it to your current situation.

This is definitely something that I would change in future work projects.

Have more people knowledgeable in Golang or use a different language

My supervisor, Nicolas Dijoux, was the only individual within RedEye who was extensively familiar with the Golang language which did cause some issues throughout the project's development.

As Golang is a relatively new programming language, the resources for help are limited which meant when I had issues that were not documented online, I had to ask Nicolas Dijoux for assistance. Problems arose when Nicolas Dijoux did not come into work the same days that I did, or he had other tasks that needed completion. This put blocks on the progress of my work at certain stages as I could not continue forward until these issues were solved.

To solve this issue, RedEye and I could try and find outside experts in Golang to help the company develop the project alongside myself and Nicolas Dijoux, or we could have used a different programming language that has more online documentation that can be used to help solve these issues.

3 APPENDICES: PROJECT DELIVERABLES

A user manual for users/clients and a technical manual for developers/SmartEye have been included as SmartEye_User_Manual.docx and SmartEye_Technical_Manual.docx.

All relevant source code project files and unit tests have been included and commented so new users/readers can understand what each section of the code does.