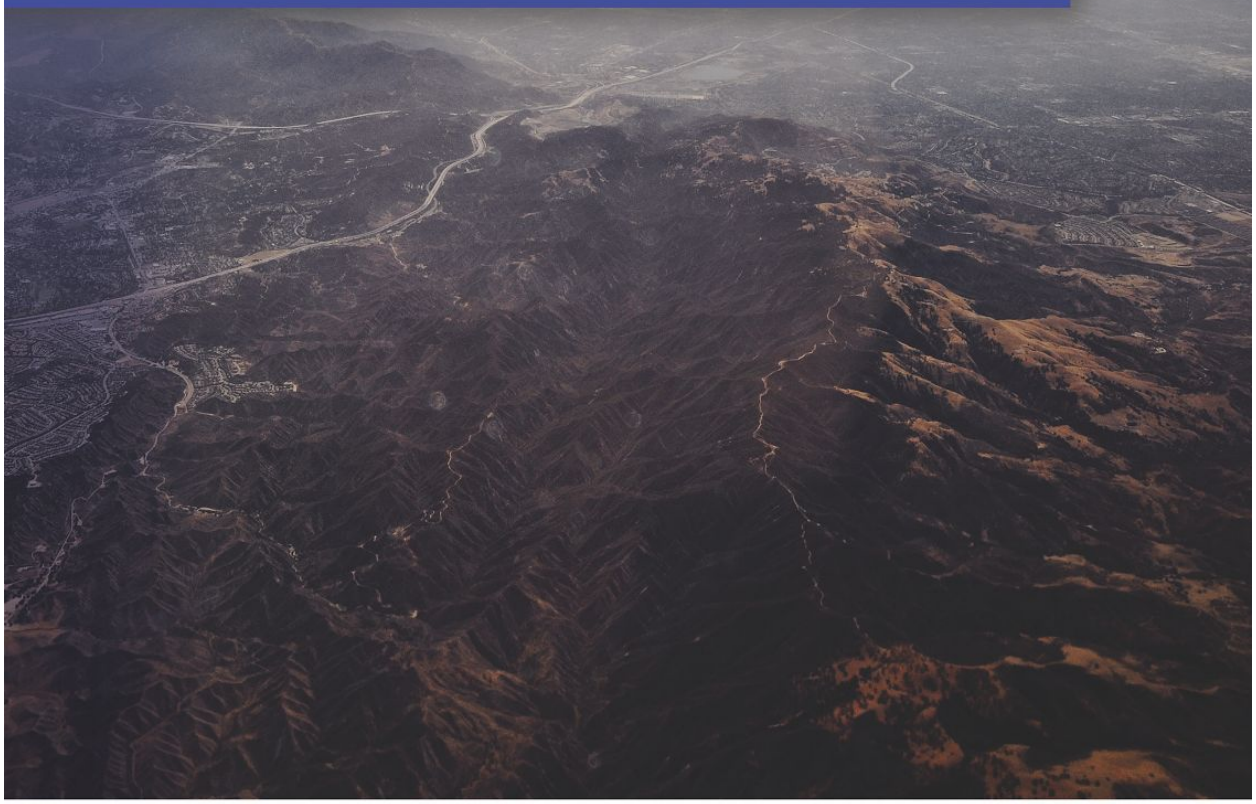
AUT

PROJECT PROPOSAL
SERVERLESS CMS
AWS LAMBDAλ

Version 4.0



TEAM KITSUI
BULLETPROOF
MISSION CRITICAL CLOUD

Team Members

Miguel Saavedra 021 124 5734 hdf3153@autuni.ac.nz
Software Development

Adam Campbell 021 255 6332 npw8877@aut.ac.nz
Software Development, Computer Science

John Cave 022 109 1268 mxy3233@aut.ac.nz
Networking and Security

Christopher Threadgold 027 380 5215 mxy3233@aut.ac.nz
Computer Science

Supervisor

Waqar Hussain
whussain@aut.ac.nz

Client BulletProof

James Burton
021 405 877
james@cloud.house

Version Control

v1.0	15/03/2016	Creation	All members
v2.0	20/03/2016	Revision	All members
v3.0	21/03/2016	Proofreading and formatting	Christopher Treadgold
v4.0	21/032016	Implementing supervisor feedback, Editing	All members
v4.1	30/05/2016	Adding Roles to stakeholder table	Miguel Saavedra

Table of Contents

Terms of Reference

Cloud House has been acquired recently by Bulletproof and is in the process of merging with its parent company, so from this point on we will refer to our client as Bulletproof.

Bulletproof is one of the leading Amazon Web Services (AWS) Premier Consulting Partners within the Asia-Pacific region. They build professional solutions using AWS/consult in the use of AWS for various businesses across Australasia. They also offer a number of other services, including, but not limited to:

- Migration of services to the cloud.
- Optimisation of scaling and speed for cloud services.
- Assuring stability and security of cloud services with compliance to industry standards.
- Custom AWS solution development.

This project was conceived by Jordan Grieg, co-founder and Chief Technical Officer (CTO) of Cloud House (currently transitioning to a role within bulletproof) in order to capitalise on a gap in the market. That is, that there is no well supported Content Management System (CMS) that runs on AWS Lambda. The reason this is considered a worthwhile undertaking is that use of AWS Lambda rather than a server will reduce costs to businesses and individuals for hosting websites. It will also provide an easier way to develop and deploy simple websites due to not having to deal with setting up a server. The primary goal of this project is to get more people using AWS, and to have Bulletproof be the company that made it possible, drawing more customers to them in the long run.

In order to achieve our goal we plan to make the CMS open-source, the aim being to utilise the skill and interest of the open-source community on GitHub.
James Burton is a corporate handyman at Bulletproof meaning his job fills many gaps within the business including marketing, human resources and website content management. James studied at the University of Otago majoring in Business Commerce as well as working at FutureTech before he moved to Cloud House/Bulletproof. He is is our main contact with Bulletproof.

Jordan Grieg will also be assisting us, focusing more on the technical aspects of the project.

Project Rationale

Bulletproof currently provides custom AWS solutions where clientele can utilize cloud servers for hosting, testing and development. The advantages of their product are known throughout the industry. They provide easy to use, well-documented web API's with great flexibility. This makes it easy for their clients to build new applications for AWS or to migrate existing applications to the cloud.

AWS provides rapidly scalable, high performance cloud services that are reliable and highly secure. AWS is extremely cost effective with not nearly as much overhead as using traditional servers since customers are billed for only the AWS resources they use such as compute power and storage. All of these features scale up to even powerhouse web applications such as Netflix and Apple's icloud, both of which run purely on AWS.

Through meetings with the client we have outlined the following reasons to pursue this project:

- A CMS tool using AWS Lambda would have lower overhead and setup time than a server-based approach due to code only being run as needed with no need to set up entire new servers as demand increases, or remove servers as demand decreases.
- A CMS tool using AWS Lambda even has lower overhead and setup time compared to the usual cloud server approaches such as AWS EC2. This is because even with cloud servers new instances still have to be spun up and down as demand shifts up and down.
- The Client wishes to encourage use of AWS and considers this project a great way to get people using AWS both through the people who use the CMS and through people who contribute to its open-source development. This increase in AWS users should result in more customers for Bulletproof.

Project Plan

We have created a Gantt Chart to plan our project (see artefact. 1). We are assuming that all team members will contribute 10 hours per week to the project. Once each team member has completed training in AWS we will begin sprinting. There will be a total of of seven two week sprints for development, followed by one for client acceptance testing. Providing that testing is successful we will have successfully created and delivered a Minimum Viable Product(MVP) by the standards of Bulletproof.

Remaining time after this delivery until the end of the semester will be spent eliciting feedback from open-source developers about the future of the project, as well as working with interested parties to

deploy the software and spreading the word about our project online through channels such as blogs and forums.

Scope and Objectives

Bulletproof has an overall idea of what they want but does not have all of the details. We as the project group have been given a fair amount of flexibility in influencing these details as we are expected to both design and create the Lambda based CMS.

Though it would be possible to elicit the entirety of the requirements before starting on design and creation of the project, doing so could take several months, severely limiting the time we would have to actually make the CMS. Therefore we have decided to begin right away and adapt to the changing requirements as they are made apparent.

With our decision to accept that much of the requirements will be unknown at this stage, we are currently limited in regards to how specific the project scope can be. As such we have been fairly broad in defining scope elements and will refine them as a more concrete idea develops of what we are creating. Despite this we hope this section can still provide valuable insight into where we are heading and what we must complete to get there.

Main Objectives

The overall goal of this project is to provide a functioning AWS Lambda CMS for Bulletproof. The CMS must be able to be used to create a blog and company website with basic functionality. Bulletproof wishes for the CMS to be considered, by their own definitions, an MVP (Minimum Viable Product) by its completion. Some things that should be easy to implement with the CMS are:

- Blog post implementation.
- Adding custom web pages.
- Login page.
- Administrative dashboard linked to certain accounts.
- Membership dashboard for each member.

Security

We will be designing and developing our own security model for authentication and authorization of the AWS CMS. The purpose of implementing our own security model is due to the lack of pre built security modules provided by Amazon. We will be using industry standard one way hashed and salted password authorization for providing access to administrative functionality.

Extension Objectives

Should we produce a functioning MVP CMS and still have additional time to spend, we would spend any additional time adding functionality such as:

- Payment processing support.
- Localization options e.g. multiple language support.
- Pluggable modules for quick additions of website functionality.

Deliverables

- Scrum reflection, at the end of each sprint.
- Burndown chart, updated at the end of each sprint.
- Scrum Product backlog, updated at the beginning of sprints and as needed.
- A functioning website created using the CMS such as a company website or blog.
- Git repository of product.
- Meeting minutes, at least weekly.
- Product wiki.
    - Documentation for users.
    - Development details for us and for open source developers.
    - On release we will open the wiki to modification so that ongoing development can be documented by open source developers.

Deadline

The project should reach the point of being considered an MVP on or before 14/09/16 as specified by Bulletproof. Project should be wrapped up by 02/11/16.

Project Approach

As detailed at the beginning of the Scope and Objectives section we have a limited understand of requirements currently as well as a short time frame from project start to project completion, and so we have decided an agile approach is best. Specifically, we have decided to use Scrum due to its ability to excel in environments of high uncertainty as well as at the suggestion of Bulletproof. In addition Bulletproof is extremely knowledgeable in Scrum, meaning we can take this as an opportunity to enhance our Scrum knowledge with their assistance.

Waterfall vs Scrum

The waterfall methodology was an option we considered when discussing methodologies, however we opted not to use it. A primary reason for this is that the waterfall method requires the gathering of all requirements before moving on to the design phase, which would take several months, and is this not viable for this project due to our short time frame. Scrum does not have a strict requirements gathering phase and so is more suitable an approach for our project than waterfall. Some other reasons we consider the waterfall methodology to be unsuitable are:

- Having to follow each phase of the waterfall methodology strictly creates a lot of overhead time where we must make sure each phase has been closed appropriately, giving us less time to work on the actual work that needs to be done in each phase.
- The waterfall methodology is well known for being more susceptible to scope creep than agile methodologies due to the lower level of interaction with the client during creation of the product. With strict deadlines due to AUT we can not afford to let scope creep cost us additional time.

Kanban vs Scrum

Kanban, like Scrum, is an agile methodology and as such has similar advantages over the waterfall methodology as Scrum. Unlike Scrum, Kanban is an ongoing process that is not broken down into time units, and as such doesn't have an explicit time or reflection and acceptance testing with the client. As a team we highly value getting feedback from the client, and so we opted for the methodology with the more explicit feedback mechanism.

Phases

Scrum does not have phases in a traditional sense, with units of progress called sprints. Sprints can range anywhere from one week to several months but tend to be on the shorter end of that scale so as to iterate quickly. Quick iterations help keep projects flexible, allowing for changing requirements. We have opted for a sprint length of 2 weeks both at the suggestion of bulletproof and from discussions internally.

Tasks

We will keep a product backlog to monitor and select tasks. Team members will select tasks at the beginning of sprints or once they have finished their current tasks. We aim to limit tasks so that they can be completed in a number of days to a week, with tasks exceeding this being split into several

smaller tasks. Tasks will be added at the beginning of each sprint and will be prioritised (based on estimated importance) by the client with additional tasks added as required. The number of tasks completed compared to the number of tasks that we predicted to be completed during that sprint will be what is used to create and update the burndown chart at the end of each sprint.

To ensure no one gets stuck on a task, and that tasks are being completed on time, we will have a daily standup meeting. This meeting will take place in person at least once per week, with the remaining standups being held using an online communication tool such as Google Hangouts or Slack.

Skills and Knowledge

This project team is comprised of four members, all of whom are confident in their ability to work as a team and conduct themselves professionally at all stages of the project. Three members of the team are from software development and computer science backgrounds, with the fourth member of our team from a Networks and Security background.

Research Phase

The first stage of the project will be a market research phase. We have to look into what users of AWS would desire in this CMS and what could convince them to use it. Some of the skills required by the team are:

Skill	Team member with skill
Software Design	All Members
Analyse current CMS solutions	John
Requirements Engineering	Chris, Miguel, Adam
Effective Communication	All Members

Development Phase

The next stage of the project will be when we code and ship the project, this will require strong software engineering and development skills, the skills required include:

Skill	Team member with skill
Programming Knowledge	All Members
Solutions Engineering	All Members
SEO - Search Engine Optimisation	All Members have some knowledge
Understanding of the HTTP protocol	John
AWS Skills	No Members

Programming Platforms

The following programming platforms will be used throughout the project and all team members who do not have them will have to develop them during the project.

Skill	Team member with skill
Node.JS	No Members
AWS Lambda	No Members
HTML	All Members
Bootstrap	John, Chris, Miguel
Database Administration	All Members
Git	Chris, Adam, Miguel

Upskilling

The team has most of the skills that will be needed. The exceptions to this are the Node.js programming language and knowledge of the Amazon Web Services stack. To fulfill the need for AWS knowledge we will be using resources provided by Bulletproof over the two weeks from the twenty fourth of March until the 12th of April. This overlaps with the time we will spend designing the end product as it will be an individual learning exercise. During this time the team will also apply themselves to learning how to use the Node.js platform, building on their knowledge of Javascript to learn this framework.

Estimation Of Costs

For this project, the client will not be charged for the production of the software they have requested, instead the costs associated with the production will be evaluated in “Man Hours” (“Man-hour | Define man-hour at Dictionary.com”, 2016). There will however be a monetary figure listed within the Cost Estimates table, to indicate the hours dedicated by the team and supervisor.

Summary of Projected Costs

As we will be assessing the cost for this project in terms of hours worked, and having an end date for the production of the CMS being September 14th 2016. We have calculated a total of 14 weeks available for design, development, testing, and handover to the client.

The client has offered to pay each member $30 per hour, till we have gained certification in AWS development after which the pay will increase to $50 per hour.

Software Costs

To reduce software costs, we will use as many free tools as possible. The only exceptions being Atlassian’s JIRA (“JIRA Software - Issue & Project Tracking for Software Teams | Atlassian”, 2016) and Atlassian’s Confluence (“Confluence - Team Collaboration Software | Atlassian”, 2016), each of which cost of $10 a month, for a total cost of $20 per month.

JIRA will be the main tool for the creation, management, and processing of requirements for this project. The Team will use JIRA Agile (a part of JIRA with features specific to Agile methodologies) for creation and management of the Scrum product backlog. It will also be used for other Scrum artefacts (burndown chart, velocity tracking).

Confluence is a secondary tool provided by Atlassian, which is a tool available at an additional cost as a part of JIRA (see cost estimates). The Team will be using Confluence for the creation of project management artefacts, as a document repository, and as an internal Wiki for project info.

Development Costs

Each member of the team will require access to AWS, the cost of using AWS will be provided by Bulletproof who will be requesting $5000 worth of computation time from Amazon. This cost will be enough cover the costs for all phases of the project.

GitHub ("How people build software - GitHub", 2016) will be our online repository for project code, as well as being a place to download a copy of the CMS tool upon project completion. GitHub is free.

Selenium ("Selenium - Web Browser Automation", 2016) will be the web browser based automated test platform for the project. Selenium will be used to record a set of given instructions (interactions) with the CMS via a web browser, and will allow the team to create a set of records which can be run at any time to assert expectations from web page interactions.

Hardware Costs

For this project, the Team will be taking advantage of the fact that each member already owns a suitable device for development and testing (laptops, desktops, etc). As such, Bulletproof will not incur the cost of providing development machines.

Upskilling Costs

Bulletproof has agreed to provide each team member with access to an online learning facility (Linux Academy) for 2 months. Access to Linux Academy costs $157NZD per month for 4 users, which Bulletproof has agreed to cover.

Bulletproof has also agreed to cover the cost of the AWS Certified Developer (Associate level) certification as a part of the upskilling process, a one off cost of $224NZD per member.

The training period will be over of 2 weeks. This will mean 1 less sprint is available for the actual production of the project, however the training is necessary for the success of the project.

The Team will also be required to learn node.js, Bootstrap and a few other tools to successfully complete this project, part of which will take place during the training period of 2 weeks. This will be done using free online tutorials.

Figures

Cost Estimates

Descriptor	Quantity	Cost	Total
Supervisor			
Waqar Hussain	108	$147.00	$15,876.00
Developer Training			
Adam Campbell	20	$30.00	$600.00
Miguel Saavedra	20	$30.00	$600.00
John Cave	20	$30.00	$600.00
Chris Treadgold	20	$30.00	$600.00
Developers			
Adam Campbell	280	$50.00	$14,000.00

Miguel Saavedra	280	$50.00	$14,000.00
John Cave	280	$50.00	$14,000.00
Chris Treadgold	280	$50.00	$14,000.00
Software			
JIRA	8	$10.00	$80.00
Confluence	8	$10.00	$80.00
Hardware			
Devices to access project tools	4	$0.00	$0.00
Training (Lambda)	2	$214.00	$214.00
Certification	4	$224.00	$896
		Subtotal	$75,536.00
	GST	15%	$11,330.40
		Total	$86,866.40

Risk Matrix

Risk	Likelihood	Impact on Project	Response Plan
Misunderstanding requirements	1	High	Communicate with the client frequently, ensuring that any features we implement fit with their vision of the finished product.
Minor team conflicts	2	Medium	Introduce more structure to the discussion, note points down one after the other and make decisions from a more neutral standpoint.
Finished product has a number of bugs	3	High	Utilize quality assurance methods and extensive software testing before deployment.

Lack of external Client Communication	4	High	Ensure the client is updated frequently on the project progress using slack and holding as frequent meetings as possible.
Lack of Team Communication	5	High	Remind team members that are not communicating the impediments factors of the project and rules within the team agreement.
Lack of learning resources	6	Medium	Make sure the client provides sufficient learning resources in a timely manner.
Planning project deliverables poorly	7	High	Setting tasks with a buffer of 1.5 times longer than our initial estimation of its duration.
Change Supervisor	8	Low	The new Supervisor will swiftly be informed of the project and the current status with use of the project charter and various other scrum and extreme programming artefacts.
Change/Loss of Team member(s)	9	High	Assess leaving team member's workload and decide whether to cut scope, request an extension or divide work amongst available members.

Quality Assurance Plan

Procedures	How	Why
Development Phase		
Sprint Acceptance Testing	Presenting the product to the client at the end of each sprint with a sprint review to get feedback on what we've implemented.	To ensure that the product meet the clients requirements.
Integration Testing	Throughout the course of the development of a feature through a sprint integration testing occurs when that feature is being merged into the stable main branch.	To ensure that the main branch is stable and working features are implemented.
Bug Protocols	1. The Bug name, details and priority are added in the intra-sprint issue	To make bug fixing as simple and transparent as possible.

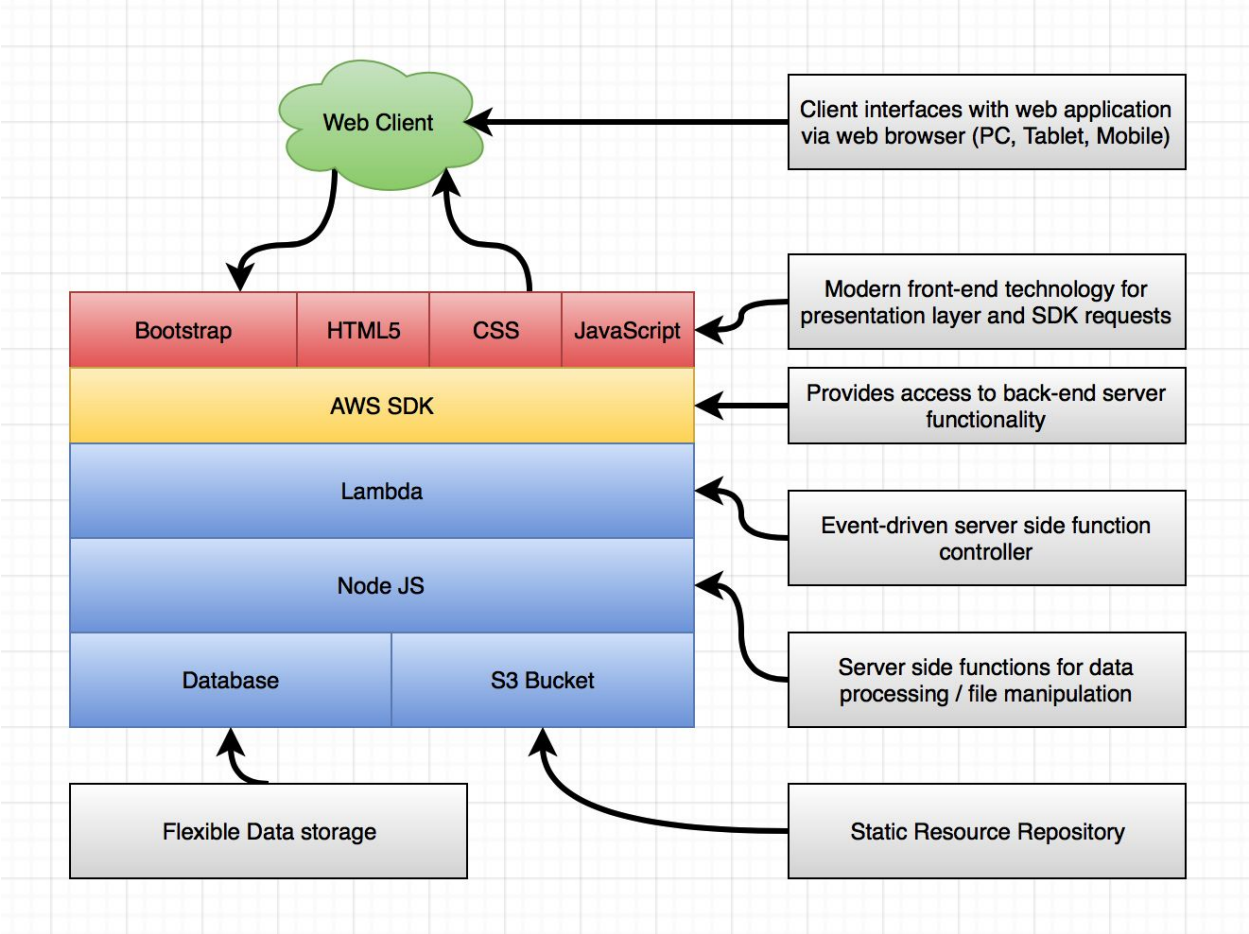
	log. The bug is rated high if the bug impedes progress, medium if it affects the functioning of the program in a significant way, and low otherwise. 2. Bugs in the issue log are fixed.. 3. If all bugs are not fixed before the beginning of the next sprint, add the remaining bugs to the overall issue log.	

Documentation Protocols	Document appropriately and update the project wiki frequently.
	
	- Comment all code
	  - The functionality it performs
	  - The Parameter use
	- When recording on the online wiki the functional use of the deliverable products functions are briefly explained, what parameters are used and what other functions does it interact with.
	- Each class file will have a change log, where each change must be logged with the Editor's name, date and what has been changed.
		To increase maintainability and make development of the product easier.

Coding Style	Each developer will have to adopt similar coding and documenting styles. We will be following the coding standards found in Felixge's node-style-guide ("Js Node Coding Style - Github" - 2016).	To keep code readable and understandable.
Github Integrity	Team members should only make a pull request once all branches in the local repository have been merged to to the local main. This main branch will only be updated once changes have been approved by another developer and the Lead developer.	Prevents problems being introduced to the main branch.

Github Branch Management	The main branch will be managed by the Lead developer meaning that they are the only person that is allowed to accept pull requests.	To reduce conflict between different merged code.
Github Merging Practice	All pull requests are overseen by the lead and secondary developers, who then decide to approve or deny merging that branch.	This allows second person to spot bugs or problems with the code that one person may have not seen on their own.
Github Commit Practice	Every commit to a branch a developer is working on has to contain comments that have the functional aspects of the user story and current medium to high priority bugs.	This will allow for readability when tracking versions of the product, making it easier to find where a bug may have originated.
Testing Phase/End of Sprint		
Unit testing	Selenium will be our tool to automate written unit tests that are for stress test and help check for logical and syntax errors within the code.	This will reduce the time it takes compared to manually testing the code, leaving more time for other things.
Exploratory testing	➢ Reviewing the logical program path ➢ Finding where it may fail ➢ Making a small statement of what a the test case will accomplish ➢ Testing to achieve the result	Exploratory testing allows our developers to create further tests to ensure the resilience and security of the product
Post Sprint		
Showcasing	Amended Deliverables will be shown to the client at the end of each sprint for client correspondence and feedback.	This will make sure that the product meets the needs of the client.

Basic Project Architecture



This is a high level conceptual model of the client interaction with the AWS CMS application. It contains a subset of the services provided by AWS which are required to achieve success within this project. These services mentioned are based on a subset of the model of Amazon's Web Application Hosting architecture (“Amazon Web Application Hosting”, 2016).

Access to the back-end server side code will be managed using the AWS JavaScript SDK (“AWS SDK for JavaScript in the Browser”, 2016). The JavaScript SDK will allow client interactions with the front end to request event triggers through Lambda, calling server side code for data processing. Being a web application, this will in turn result in a web page being sent back to the client.

Stakeholder Register

Bulletproof

Name	Role	Phone	Email	Background	Influence
James Burton	Community Manager	021 405 877	james@cloud.house	Art Director, Business Development, Marketing, Design	Moderate
Jordan Grieg	CTO/Founder	09 869 2888	jordan@cloud.house	Solutions Architect, Cloud Infrastructure	High
Scott Judson	CEO/Founder	09 869 2888	scott@cloud.house	Management, Cloud Services, Mobile Applications	High

Research and Development Team

Name	Role	Phone	Email	Background
Waqar Hussain	Supervisor	-	whussain@aut.ac.nz	Lecturer

Name	ID Number	Phone	Role	Email	Major(s)
Adam Campbell	1311607	021 255 6332	Lead Developer	npw8877@aut.ac.nz	Software Development, Computer Science
Miguel Saavedra	13826904	021 124 5734	Team Leader, Scrum Master, Developer	hdf3153@autuni.ac.nz	Software Development
Chris Treadgold	1399164	027 380 5215	Developer	hfv6430@autuni.ac.nz	Computer Science
John Cave	1324776	022 109 1268	Lead Tester, Developer	mxy3233@aut.ac.nz	Networking and Security

Glossary of Terms

Term	Definition
AWS	Amazon Web Services; a selection of useful services packaged for use by programmers for their Cloud needs.
Lambda	An AWS service that allows code to be run without the programmer needing to manage the server it runs upon.
CMS	Content Management System - a tool used by technical and non-technical people to easily create and manage website(s).
Github	A free, hosted software collaboration tool used by open-source projects to maintain their source code.
SEO	Search Engine Optimisation; the process of making a website easily understandable to robots such as those used to find Google results.
Repository	Central location where data is stored

References

Google Docs - create and edit documents online, for free. (2016). Retrieved March 20, 2016 from https://www.google.co.nz/docs/about/

Js Node Coding Style - (2016). Retrieved Retrieved March 21, 2016
From https://github.com/felixge/node-style-guide

Microsoft Word | Document and Word Processing Software. (2016). Retrieved March 20, 2016, from https://products.office.com/en-us/word

How people build software - GitHub. (2016). Retrieved March 20, 2016, from https://github.com/

JIRA Software - Issue & Project Tracking for Software Teams | Atlassian. (2016). Retrieved March 20, 2016, from https://www.atlassian.com/software/jira

Confluence - Team Collaboration Software | Atlassian. (2016). Retrieved March 20, 2016, from https://www.atlassian.com/software/confluence

Slack: Be less busy. (2016). Retrieved March 20, 2016, from https://slack.com/is

Selenium - Web Browser Automation. (2016). Retrieved March 20, 2016, from http://www.seleniumhq.org/

Man-hour | Define man-hour at Dictionary.com. (2016). Retrieved March 20, 2016, from http://www.dictionary.com/browse/man-hour

AWS SDK for JavaScript in the Browser. (2016). Retrieved March 20, 2016, from https://aws.amazon.com/sdk-for-browser/

Amazon Web Application Hosting. (2016). Retrieved March 20, 2016, from http://media.amazonwebservices.com/architecturecenter/AWS_ac_ra_web_01.pdf

What is Exploratory testing in software testing?. (n.d.). Retrieved March 20, 2016, from http://istqbexamcertification.com/what-is-exploratory-testing-in-software-testing/

Disclaimer

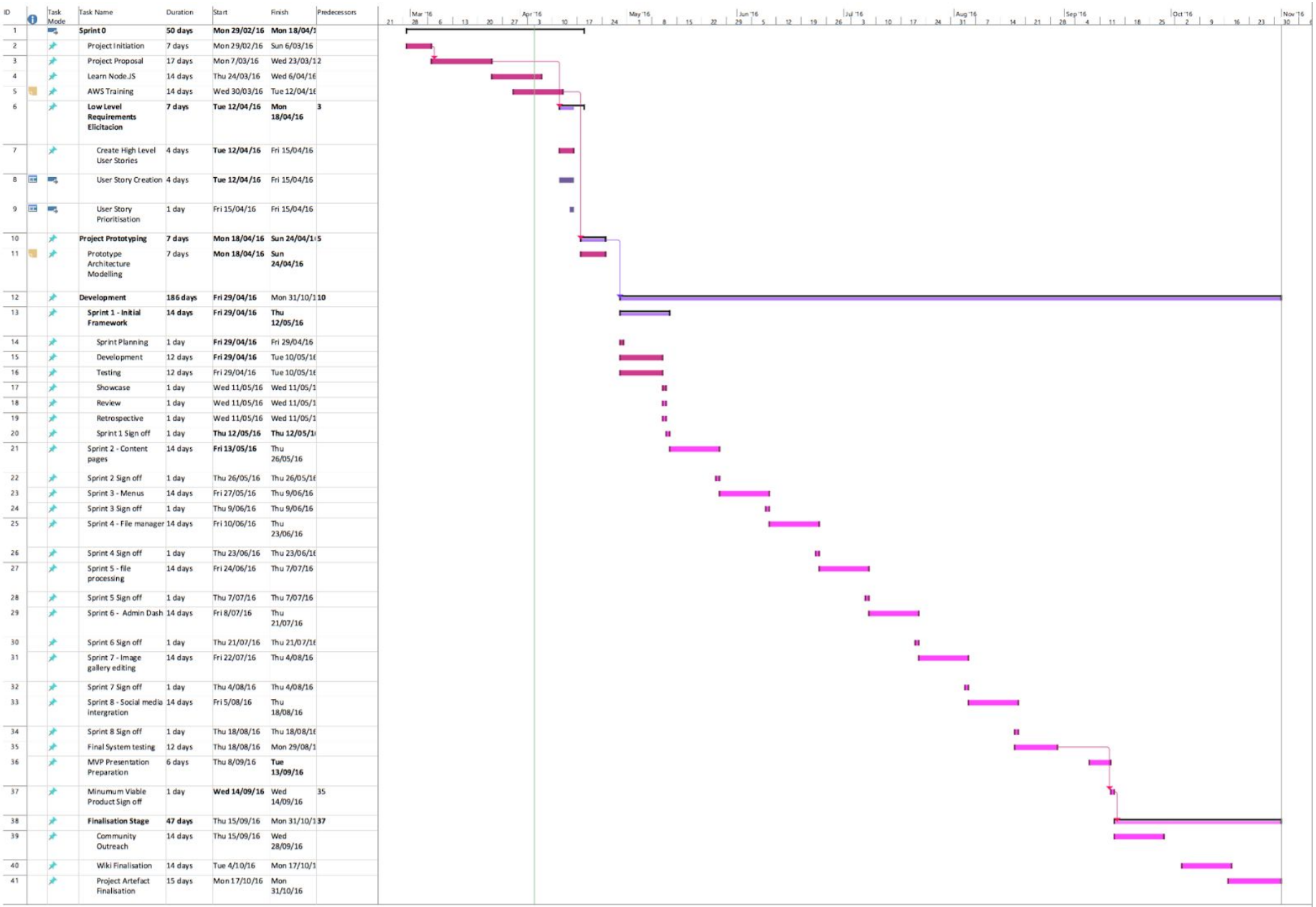
Clients should note the general basis upon which the Auckland University of Technology undertakes its student projects on behalf of external sponsors:

While all due care and diligence will be expected to be taken by the students, (acting in software development, research or other IT professional capacities), and the Auckland University of Technology, and student efforts will be supervised by experienced AUT lecturers, it must be recognised that these projects are undertaken in the course of student instruction. There is therefore no guarantee that students will succeed in their efforts.

This inherently means that the client assumes a degree of risk. This is part of an arrangement, which is intended to be of mutual benefit. On completion of the project it is hoped that the client will receive a professionally documented and soundly constructed working software application, some part thereof, or other appropriate set of IT artefacts, while the students are exposed to live external environments and problems, in a realistic project and customer context.

In consequence of the above, the students, acting in their assigned professional capacities and the

Auckland University of Technology, disclaim responsibility and offer no warranty in respect of the "technology solution" or services delivered, (e.g. a "software application" and its associated documentation), both in relation to their use and results from their use.



ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		**Sprint 0**	**50 days**	**Mon 29/02/16**	**Mon 18/04/1**	
2		Project Initiation	7 days	Mon 29/02/16	Sun 6/03/16	
3		Project Proposal	17 days	Mon 7/03/16	Wed 23/03/1	2
4		Learn Node.JS	14 days	Thu 24/03/16	Wed 6/04/16	
5		AWS Training	14 days	Wed 30/03/16	Tue 12/04/16	
6		**Low Level Requirements Elicitacion**	**7 days**	**Tue 12/04/16**	**Mon 18/04/16**	3
7		Create High Level User Stories	4 days	**Tue 12/04/16**	Fri 15/04/16	
8		User Story Creation	4 days	**Tue 12/04/16**	Fri 15/04/16	
9		User Story Prioritisation	1 day	Fri 15/04/16	Fri 15/04/16	
10		**Project Prototyping**	**7 days**	**Mon 18/04/16**	**Sun 24/04/1**	5
11		Prototype Architecture Modelling	7 days	**Mon 18/04/16**	**Sun 24/04/16**	
12		**Development**	**186 days**	**Fri 29/04/16**	Mon 31/10/1	10
13		**Sprint 1 - Initial Framework**	**14 days**	**Fri 29/04/16**	**Thu 12/05/16**	
14		Sprint Planning	1 day	**Fri 29/04/16**	Fri 29/04/16	
15		Development	12 days	**Fri 29/04/16**	Tue 10/05/1	
16		Testing	12 days	Fri 29/04/16	Tue 10/05/1	
17		Showcase	1 day	Wed 11/05/16	Wed 11/05/1	
18		Review	1 day	Wed 11/05/16	Wed 11/05/1	
19		Retrospective	1 day	Wed 11/05/16	Wed 11/05/1	
20		Sprint 1 Sign off	1 day	**Thu 12/05/16**	**Thu 12/05/1**	
21		Sprint 2 - Content pages	14 days	**Fri 13/05/16**	Thu 26/05/16	
22		Sprint 2 Sign off	1 day	Thu 26/05/16	Thu 26/05/1	
23		Sprint 3 - Menus	14 days	Fri 27/05/16	Thu 9/06/16	
24		Sprint 3 Sign off	1 day	Thu 9/06/16	Thu 9/06/16	
25		Sprint 4 - File manager	14 days	Fri 10/06/16	Thu 23/06/16	
26		Sprint 4 Sign off	1 day	Thu 23/06/16	Thu 23/06/1	
27		Sprint 5 - file processing	14 days	Fri 24/06/16	Thu 7/07/16	
28		Sprint 5 Sign off	1 day	Thu 7/07/16	Thu 7/07/16	
29		Sprint 6 - Admin Dash	14 days	Fri 8/07/16	Thu 21/07/16	
30		Sprint 6 Sign off	1 day	Thu 21/07/16	Thu 21/07/1	
31		Sprint 7 - Image gallery editing	14 days	Fri 22/07/16	Thu 4/08/16	
32		Sprint 7 Sign off	1 day	Thu 4/08/16	Thu 4/08/16	
33		Sprint 8 - Social media intergration	14 days	Fri 5/08/16	Thu 18/08/16	
34		Sprint 8 Sign off	1 day	Thu 18/08/16	Thu 18/08/1	
35		Final System testing	12 days	Thu 18/08/16	Mon 29/08/1	
36		MVP Presentation Preparation	6 days	Thu 8/09/16	**Tue 13/09/16**	
37		Minumum Viable Product Sign off	1 day	**Wed 14/09/16**	Wed 14/09/16	35
38		**Finalisation Stage**	**47 days**	Thu 15/09/16	Mon 31/10/1	37
39		Community Outreach	14 days	Thu 15/09/16	Wed 28/09/16	
40		Wiki Finalisation	14 days	Tue 4/10/16	Mon 17/10/1	
41		Project Artefact Finalisation	15 days	Mon 17/10/16	Mon 31/10/16	

Project: MS Project Document
Date: Tue 5/04/16

Task
Split
Milestone
Summary
Project Summary
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
External Tasks
External Milestone
Deadline
Progress
Manual Progress