Technical Report – Group Project

COSC310 – Software Project Management

Team Crocodile

Joel Foster

Daniel Hemmings

David Newman

Word Count: 1,993

Contents

Introduction

Project overview

The time and effort required to manually sort and classify images is significant and painstaking. The collection of field data and the processing of said data needs to be automated and streamlined in such a way that the database of information can dynamically update as new data is collected. These efficiency gains would allow end users to easily access up to date data as they require it. The client has provided their predictive model for the automatic classification of drop bears across various camera trap deployments. The aim of this project is to deploy this model (via a Cloud based platform) and to develop a means for users to interact with the output prediction data.

Goals of the project

- Develop an AWS Cloud solution to retrieve raw data from deployed camera traps to be processed by a trained TensorFlow machine learning model and stored in an SQL database.
- Develop the required website and mobile phone application to inform the general public about dropbear sightings and assist in dropbear research.

Project scope

To meet the user requirements for this project the following software components will be developed.

- Cloud-based API using Amazon Web Services (AWS) to retrieve raw data from email generated by pre-deployed camera traps, to then be processed by a pre-trained TensorFlow machine learning model.
- A secondary Cloud-based API using AWS to retrieve various classification data from the TensorFlow machine learning model, to be stored in an SQL Cloud-based database on the same AWS platform.

- A two-facing website to facilitate the database information for the general public and researchers. On the public level, it will allow the retrieval of dropbear sightings based on a specified postcode. On the researcher level, it will provide an approval-only web portal for researchers to apply for, which will grant access to various information on dropbear sightings.
- A cross-platform smart phone app for Android and Apple iOS with identical functionality as the general public website.
- Email alerts and smart phone notification functionality for postcode sightings.

Assumptions

- The project team has the required experience to meet user requirements and deliver the project on time.
- The client will convene with the development team every 3 months.
- The budget allocated provides enough for additional human and computer resources if required.

Constraints

- Additional funding is not available for the project.
- Existing software components cannot be modified.
- Team members time on the project is limited to 40 hours per week, Monday to Friday, for the duration of the project.

Key Milestones

The milestones are set to be delivered every three months. These meetings will be used to showcase our current advancements in the project and allow Dr Client the ability to take part in the direction of the project. This will be achieved through up-to-date training of Dr Client, with the ability for him to ask questions and discuss relevant changes.

Table 1: Milestones and timeframes

Project Milestone	**Project Artifact**	**Timeframe**
Project start		TBD
Milestone 1	Phase 1 delivery	+ 3 months
Milestone 2	Phase 2 delivery	+ 3 months
Milestone 3	Phase 3 delivery	+ 3 months
Milestone 4	Phase 4 delivery, Product completion	+ 3 months

Please note: Phase details, as seen in Table 1, can be found in the preliminary execution schedule.

Identification of Stakeholders

We understand the following stakeholders as part of the project:

- Dr Client
- University of New England
- State and Federal level project funders
- Drop Bear Protection Society of Australia
- Software Project Manager
- Development Team
- 5 test users

Technical Solution

Overview of Solution

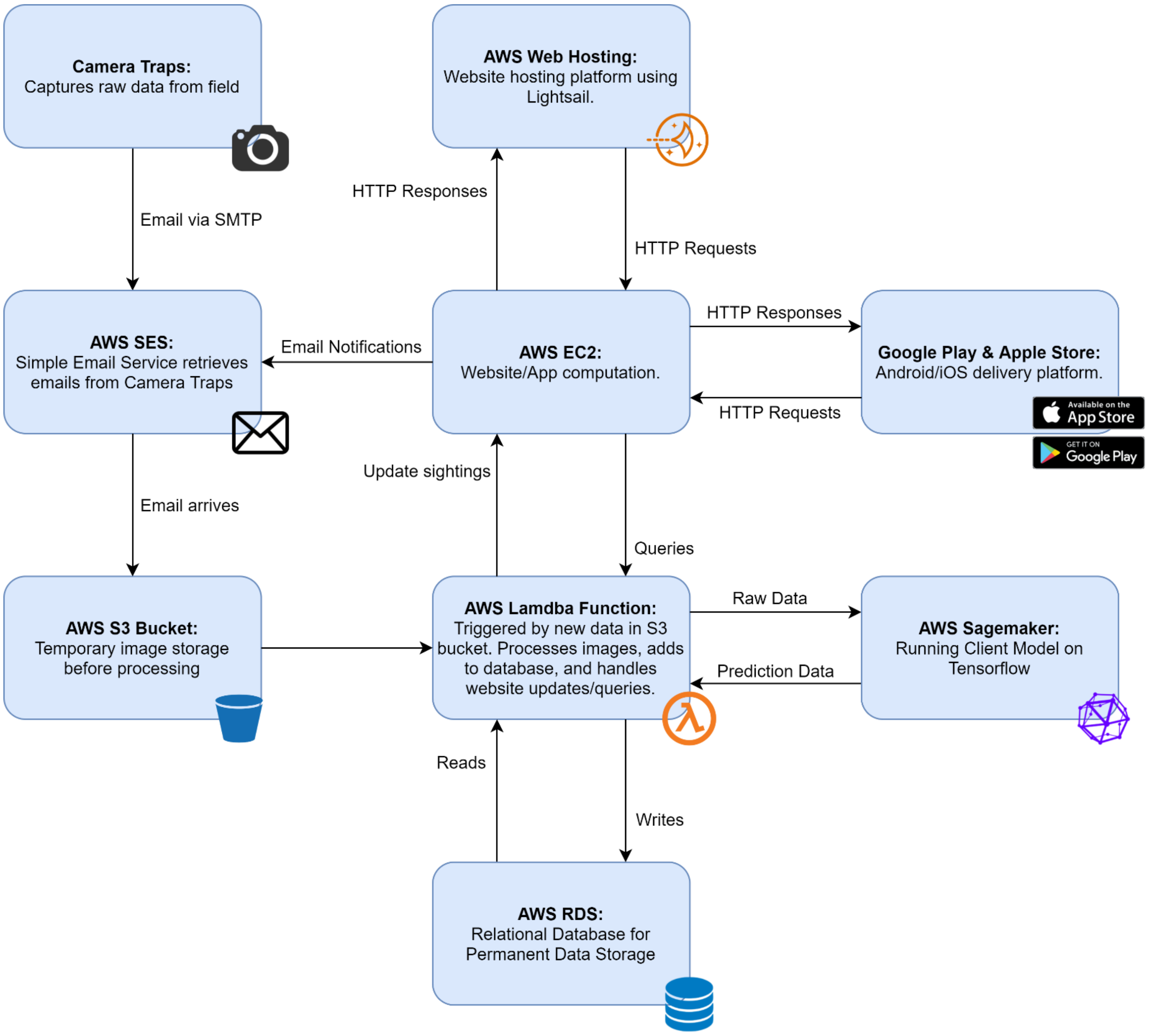
Camera Traps:
Captures raw data from field
AWS Web Hosting:
Website hosting platform using Lightsail.
Email via SMTP
HTTP Responses
HTTP Requests
AWS SES:
Simple Email Service retrieves emails from Camera Traps
Email Notifications
AWS EC2:
Website/App computation.
HTTP Responses
HTTP Requests
Google Play & Apple Store:
Android/iOS delivery platform.
Available on the App Store
GET IT ON Google Play
Email arrives
Update sightings
Queries
AWS S3 Bucket:
Temporary image storage before processing
AWS Lamdba Function:
Triggered by new data in S3 bucket. Processes images, adds to database, and handles website updates/queries.
Raw Data
Prediction Data
AWS Sagemaker:
Running Client Model on Tensorflow
Reads
Writes
AWS RDS:
Relational Database for Permanent Data Storage

Figure 1 - Overview of Proposed Solution - Cloud Processing

Major Components

- AWS Simple Email Service (SES):
  AWS Cloud based Email service is capable of intercepting emails coming in from the field camera traps.

- AWS Lambda Function:
  A script written in Python that is called to run based on a customisable trigger event, such as when AWS SES receives an email from the field. Custom script would be able to pull out any necessary metadata from email, such as timestamps, and retrieve the photo attachments. Script would then store this information into our S3 Bucket.

- AWS S3 Bucket:
  AWS Cloud storage, would act as an intermediary storage option for our data. Raw data from the Camera Trap emails could be stored, and then passed directly into AWS SageMaker for processing. This data could then be moved to our AWS Aurora database for permanent storage.

- AWS SageMaker:
  AWS machine learning service would allow us to deploy the client Tensorflow prediction model on a GPU instance (for greater processing requirements). The model would be fed with raw data from our S3 Bucket, and output prediction data in the form of a classification, and a confidence interval along with an output image, highlighting a potential Dropbear sighting (if one was detected).

- AWS Aurora:
  A relational database, using MySQL, would be the most suitable to organise and store the images and their data. This service provides automatic scaling of both storage and computation and faster speeds than some of their other offerings, providing less maintenance in the future as the system grows.

- Website (Vue.js framework)
  A modern and responsive website built for the general public and researchers. Built with Vue.js, it will allow public users to input a postcode and return positive sightings and sign up for email notifications. A web form and login option will be available for researchers. The researcher functionality will be separate from the public functionality and will include the ability to search, filter, and sort with various parameters and present the data in a table, which will be achieved by the backend with relational algebra. The frontend will be built with Vue.js and the backend will be built with PHP. HTTP requests use the RESTful API. Database management using SQL. Hosting handled by AWS.

- Android and iOS cross-platform app
  A smart phone app built for the general public with identical functionality of the public website. It will allow users to search by postcode and return positive sightings. Users will also be able to set a postcode and receive phone notifications when a new positive sighting is stored to the database. The Android app will be built using Kotlin, and the iOS app will be built using Swift. Cross-platform functionality will be done through interaction with the AWS Web Server, as shown in Figure 2 below.

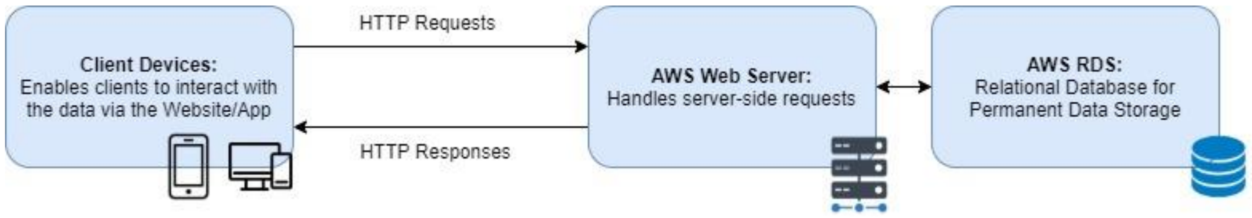
HTTP Requests
Client Devices:
Enables clients to interact with the data via the Website/App
AWS Web Server:
Handles server-side requests
AWS RDS:
Relational Database for Permanent Data Storage
HTTP Responses

Figure 2 - Client Side Interaction

Table 2: Estimate of first year AWS pricing

Service	**Details (Estimated)**	**AUD/Month ($/m)**
AWS Simple Email Service (SES)	100 triggers per camera per day = 1200 emails/day @ 1.5 MB	$45.64
AWS Lambda Function	36,000 emails a month @ 10 second processing time	$17.30
AWS S3 Bucket	~2 GB storage per day = 1 TB a year	$41.71
AWS SageMaker	2 seconds processing per image, 2 hours needed per day @ $0.32/hr	$19.31
AWS Aurora	With 3 TB storage	$2581.35
AWS EC2 Web Hosting	Website/App computation	$142.38
AWS LightSail	Hosting for the website	$120.21
SSL Certificate	Signed certificate for security	$75
	TOTAL (per month)	**$3042.90**
	TOTAL (initial year)	**$36,514.80**

Note: Prices will grow dynamically with use, dependant on added camera traps and access to servers by users/researchers.

Justification of Technical Solution

Utilising only AWS components allows for simple and straightforward integration between the various components of the proposed solution. For example:

- AWS SES and Lambda Functions allow for easier photo retrieval.
- AWS SES will also be used as an easy way to send notification emails for users who sign up on the website.
- AWS SageMaker contains pre-defined containers for running a Tensorflow model.
- AWS can dynamically scale various services are required.

- Lambda calls and AWS SageMaker are only costing money when they are running, no need to create an instance and pay for it unless it’s working.
- AWS Aurora allows a fast and efficient relational database with strong scalability.
- AWS Web Hosting has vast platform support, scalability and flexible pricing models.
- Our Android developer is experienced in Kotlin and Swift application development.
- Our website developer is a full stack web developer experienced in Vue.js, PHP, and SQL.

Development and Project Management Plan

Project Methodology

Strong documentation of the components and client feedback are two important aspects to the development of this project, and we will need to strike a balance between the two. As we are a smaller development team and have relatively short milestones to demonstrate working components, the Agile methodology is a good fit for our project. Agile places a strong emphasis on both working components and their corresponding documentation, which is required to meet the goals of the project.

The core method within Agile that our team will be using is Feature-Driven Development (FDD). FDD is an iterative model built around a five step process, as shown in Figure 3. The overall model is formed in the first two phases, and 75% of development time is spent in the 4th and 5th iterative phases where features are designed and developed. FDD’s strengths lie in quickly producing working components that can be integrated into the development environment and its ability to adapt to changes from the client. Our focus will be on regular builds to facilitate continuous integration, which will assist in demonstrating the project to the client during milestones.

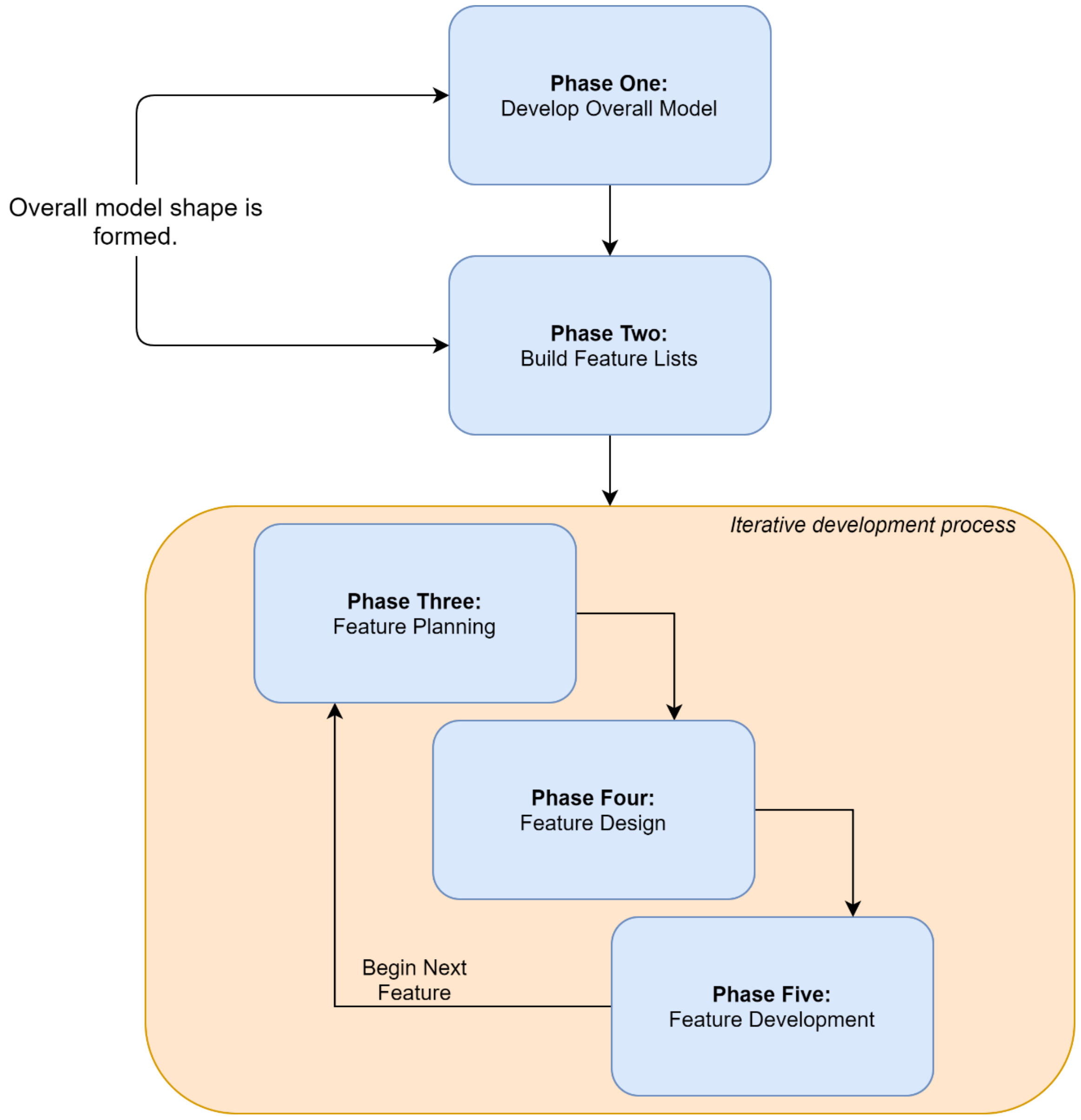

Figure 3 – Feature-Driven Development Process

Required Resources

The resources required for our project fall into the following two main categories:

<u>Human Resources:</u>

- SPM/Team Leader
- Development Team
- End users/prototype testers

The skillsets required by each of the Human Resources are outlined in greater detail below.

<u>Physical Resources:</u>

- Computer Resources (workstations, development software etc.)
- Office space and associated equipment (desks, chairs etc.)
- Networking and Internet services

As we are a start-up, each of the team will be working remotely from a home office setup. Each developer has an established work area equipped with a workstation, internet connectivity and appropriate furniture. The majority of communication between developers will be using applications like Zoom and Slack and the sharing of the code base will be via GitHub.

In terms of scheduling our resources, the entire development team will be required to work on the project throughout the duration. The project execution schedule (detailed later in this document) allows our multi-skilled team to be actively developing the various components throughout the projects lifecycle.

Project Team Member Roles

As the project is built on the fundamental Agile methodology principles, the roles within development team are derived heavily from traditional Agile team roles.

- SPM/Team Leader: Our project facilitator, will be responsible for organising the development team, interfacing with Dr Client, and acting as a mentor role within the development team. As we are a small team, the Team Leader will also be required to get hands on with development, but that is secondary to their role.
- Development Team: Hands on programmers and developers, responsible for building for the system that will be delivered to the client.
- Product Owner/Stakeholder Representative: As the Product Owner and a key stakeholder, Dr Client's insight into the requirements, and progress of the project are vital to the project's overall success.

Development Team Composition

Our Development Team will be composed of 2 developers, each with a specific skillset that will allow us to bring all the individual core components of the project together. Note, the SPM/Team Leader role is a dual Manager/Developer role, their focus is not entirely on development.

- Cloud/AWS Developer: Responsible for building the components of the project that run via AWS.
- Web/Database/Application Developer: Responsible for building the web services required (including the integration of a Database) and for the development of the two mobile phone applications.

While we have outlined very specific roles, our development team members are multi-skilled and will be working closely with each other on each component as they are built and integrated into the project.

Programming Languages and Environments

Table 3: Languages and Frameworks

Language/Framework	**Component(s)**
Vue.js	Website Development: Front-end
PHP	Website Development: Back-end
MySQL	Database Management
Kotlin	Android Application
Swift	iOS Application
Python	AWS Components (Model Deployment, Lambda Functions)

Preliminary Execution Schedule

Scheduling of Major Components

Table 4: In-depth phase details

Phase	Timeframe	Details
1	Month 0-3	• Create and deploy back-end system to receive and store images: ○ AWS SES set up to receive emails from one initial camera before adding support for the rest. ○ Set up S3 server to auto-retrieve and store images from an incoming email. ○ Create a database using AWS Aurora. ○ Use an AWS Lambda server to retrieve images stored on S3 server and store on a database with necessary image data. • Website/App backend: ○ Set up a basic AWS EC2 web server that allows querying of database through a basic website frontend. ○ Create a similar proof-of-concept for both an iOS and Android application.
2	Month 3-6	• Use AWS SageMaker to run the image prediction algorithm: ○ Using our Lambda service to now collect an image, run it through the algorithm and store in the database with additional information – classification and accuracy in confidence rating. • Batch processing of already collected data: ○ If possible, begin the process of running all external

		data that has already collected through our system and adding to the database. • Expand the website/application: ○ Allow a user to search a postcode and return sightings in the past 24 hours. Incorporate similar feature to applications. ○ Create a portal for researchers to sign up and query database once accepted.
3	Month 6-9	• Complete backend integration between services: ○ Complete links between all services as per Figure 1, including setting up SES to allow for outgoing emails (used for website notifications). • Complete website development: ○ Finalise frontend design of website ○ Allow a user to sign up for notifications using an email and a postcode, including a trigger to send new notification emails when a new sighting is added. • Complete applications for iOS and Android: ○ Permit the use of push notifications for a postcode when a new sighting is added to the database. ○ Show sightings in a postcode over a 24-hour period.
4	Month 9-12	• Field testing: ○ All twelve cameras will be implemented into the system. ○ Final testing for all systems that will help us find any issues or bugs that need to be rectified. ○ Any final adjustments as agreed upon by all stakeholders.

We will have four main phases to coincide with our milestone meetings with Dr Client. They are designed to show constant progress, allowing us to keep Dr Client up-to-date and trained in using the system from the start. This also allows any changes to be dealt with early in the project.

Testing will be conducted early and often. There will be an internal test suite to ensure all code is sturdy and faults are kept to a minimum. As well as this, we will be incorporating our five test users from Phase 1 as soon as the database is able to be queried through a proof-of-concept website and applications, as per Table 4.

As we are employing the use of Feature-Driven Development, documentation will be relied upon heavily, with the benefit of future-proofing the project. The documentation will be used to communicate important information internally, so will be thorough and written for clarity.

Component Dependency Relationships

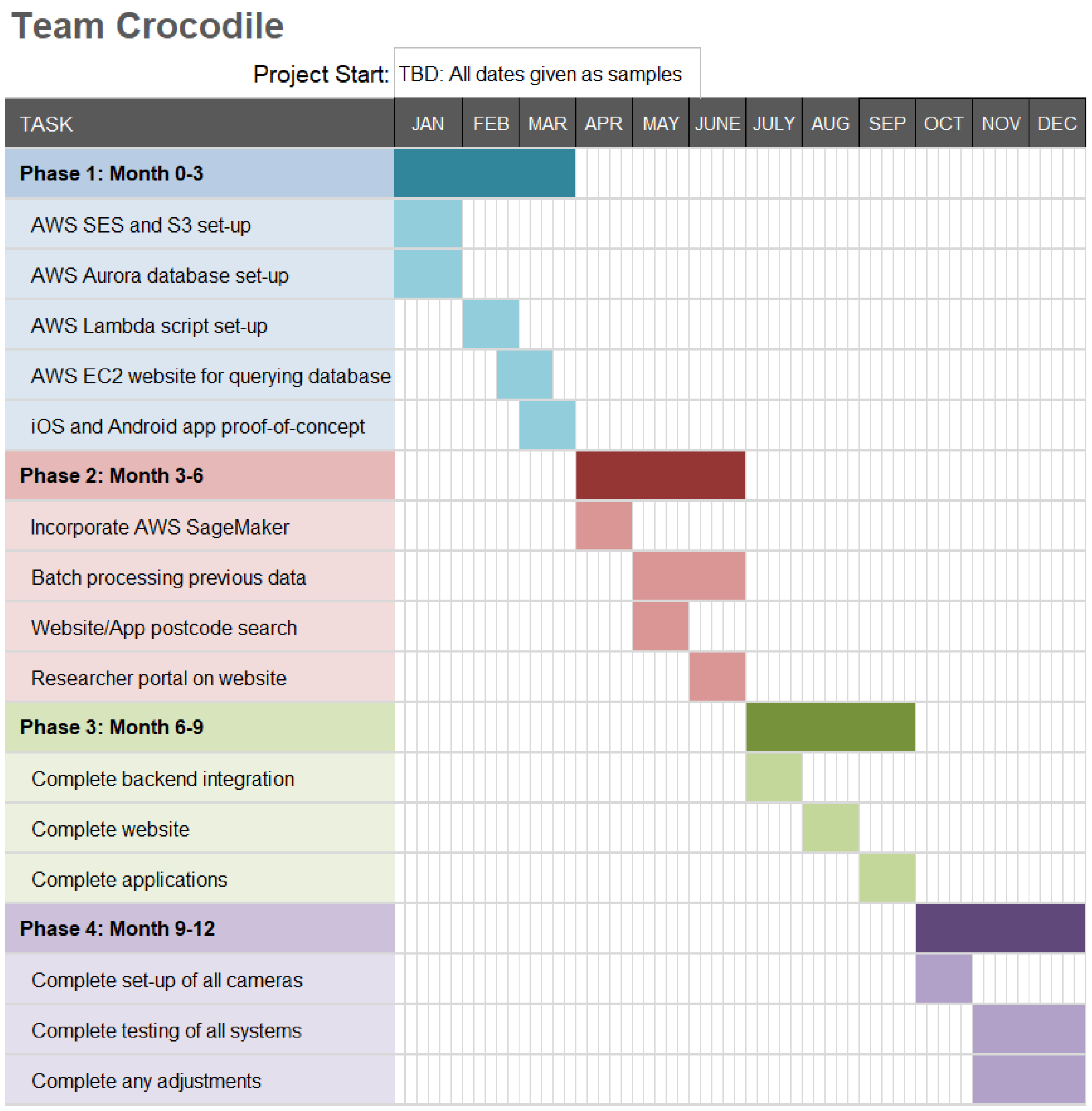

Figure 4 - Gantt Chart based on our four phases

Figure 4 details our one-year project plan to ensure no delays in completion of the project. The final three months, during the field trial phase, will allow us ample time to complete all testing and adjustments, while also allowing leeway to complete any unforeseen delays or completing any extra tasks that have been agreed upon by the stakeholders.