**Introduction - 500-600 words**

Project overview

* General introduction to the project.
* A project summary.
* Example:  
  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

* What we want to achieve.
* Provide a solution for automated retrieval of raw data (images).
* Provide a means for the automated classification of images (positive/negative sightings of drop bears).
* Store data of positive/negative sightings in a database.
* Develop a solution for end-users (users and researchers) to interact with the data.

Project scope

* How we go about it.
* Deploy a Cloud based solution for running the client's machine learning model on.
* Deploy a Cloud based database for storing the output from the client’s model.
* Develop interface solutions (website and application) for users to interact with the data.
* Describe the overall order of execution upon the camera traps triggering.
* Assumptions and constraints to the project.

Key milestones

* The schedule in terms of the 3 monthly milestones, what will be completed at these milestones.
* Project deliverables on project closure.

Identification of stakeholders

* Dr Client - Main stakeholder for the project.
* University of New England.
* State and Federal level project funders.
* Drop Bear Protection Society of Australia.
* Software Project Manager.
* Developers.
* 5 test users.

**Technical solution - 500-600 words**

Allocation:

* David - Website/app development, Overview of solution (related to goals/scope)
* Dan - General AWS integration of components
* Joel - Database/ML integration (heavily reliant on ML output)
* Justification of technical solutions (add additional comments for your technical components here).

Overview of solution

* Camera sends 3 images to the email address, the solution must then send these images to AWS and save some metadata, which will then be stored in a database and processed by the machine learning model for classification. Upon classification, the app/website is updated.

Major components (Refer to bottom of doc for more detail on major components)

* What we’re using, how it works, etc.
* A detailed and in-depth explanation of how the major components will work and interact with each other.
* Components:
  + Website for users (sort by postcode, sign up for email alerts).
  + Website for researchers (sort/filter various information on dropbear sightings).
  + Phone application that runs on the latest iOS and Android build (cross platform).
  + Amazon Web Services cloud-based solution to process images received using the machine learning model.
  + AWS needs to parse images received.
  + AWS is used for website hosting.
  + Cloud-based database connection to AWS for storage of images and metadata.
  + Full documentation of all code and solutions.
* Tabulate the costs of each component (the table doesn't contribute to word count so we can be as verbose as we want here).
* Figures and/or diagrams that describe how the technical solution will work, and the general flow of user interaction. (The stakeholders have virtually no technical background so figures would help a lot).

Justification of technical solution

* Why are we using it?

**Development and project management plan - 400-500 words**

Project methodology

* Development environment built around Agile or a subset of Agile (Scrum).
* Agile works well with small teams (like this project!).
* Diagram of our development process.
* User Stories from Dr Client.

Required resources

* Software project manager.
* Human resources.
* Computer resources.
* Physical logistics.
* Networking and internet services.
* A table that allocates funding to required resources.

Development team member roles

* Heavily related to the Project Management roles in Agile.
* Team Leader – Facilitates the development team’s organisation. Acts as an interface between the Team and external organisation (Dr Client).
* The SPM - overseer of the developers (redundant with team leader?)
* Team members:
  + Mr Web Guy - does all the website, app and database development
  + Mr Cloud Guy - does all the Cloud-based coding and implementation of API.
* Customer/user – Dr Client. Essential to the team from the beginning. Defines what the project will deliver, user requirements and priorities. Provides continual explanations and feedback. Ideal for customer work with the team, in this case Dr Client will be working with us via milestones (every 3 months).

Development team composition

* Website/App/MySQL database developer.
* AWS developer.

Programming languages and environments

* HTML/JavaScript/CSS for website development.
* PostgreSQL for database management.
* Java or Kotlin for Android development.
* Swift for iOS development.
* Python for TensorFlow/AWS API.

**Preliminary execution schedule - 400-500 words**

Include all major components in your proposed solution

* Greater detail on what we’re going to develop and in what order. Essentially an outline of the execution schedule.
* Example schedule:
  + **Phase 1 - Month 0-3**: Create and deploy the backend system. Including an AWS based database and the integration of the prediction model into the database and our project. Some form of API that pulls photos from the email accounts to the model instance on AWS.
  + **Phase 2 - Month 3-6**: Create website front-end and integration to our database and model. Allow a user to add an email and postcode to receive updates (emailing feature may not be fully implemented). Create a portal for researchers to sign up and query database.
  + **Phase 3 - Month 6-9**: Create apps for iOS and Android that allow push notifications. Complete any features of the website that have not been completed.
  + **Phase 4 - Month 9-12**: Field test where all cameras will be implemented, and the system will run. Any issues and bugs to be ironed out.

Must include component dependency relationships

* Some form of Gantt chart or network map to demonstrate what components must be finished before another can begin.

**Additional/optional sections - 200~ words**

Testing

* Should be included in our report.

User training

* Only 5 test users - do we really need user training?

Deployment

* Prototype proof-of-concept so may not be required.

Maintenance

* Dr Client strongly suggested that maintenance would not be required as we’re simply developing a prototype proof-of-concept.

**Additional comments**

* Research TensorFlow and AWS to get a better understanding of how these components can be linked together for the technical report.
* Detailed goals/scope:

Develop a Cloud-based API using AWS to retrieve raw data (the images, and timestamp) from each camera trap email address and hand it over to the TensorFlow machine learning model for classification.

Develop a separate API that retrieves classification and confidence level output from the TensorFlow machine learning model and store the information, along with the image and metadata, in an SQL Cloud-based database using AWS.

Develop a two-facing website catering to the general public and researchers. The general public will be able to input a postcode and receive back the number of dropbear sightings within a specific radius, as well as offer notifications via email by utilising a public user’s email address and postcode. For researchers, the website will be invite-only in which researchers must fill in a web-form that is manually verified. Once access has been granted, researchers will be able to find information on dropbear sightings using various sorting and search options (GPS location of the camera trap, sighting time, classification and the confidence level).

Develop a mobile phone application for Android and Apple iOS. The application will be cross platform and provide an identical experience for both platform users. It will work identical to the website; however, it will offer phone alerts rather than email alerts.

* Database information (Amazon Aurora):
  + Amazon Aurora - proprietary technology designed for AWS.
  + Supports both Postgres and MySQL (works as if either one of those DB).
  + Cloud optimised and offers vastly improved performance over MySQL and Postgres.
  + Storage automatically grows incrementally when you need it, starting at 10GB (cost effective).
  + High availability and self-healing.
  + Automatic fail-over, backup and recovery.
  + Strong security with in-flight SSL and encryption. Backups, snapshots and replicas are encrypted.
  + Cost: Standard instance (large) 1 year upfront - $1704.
  + Running costs (storage, IO, backup) -
  + Camera trap taking 3 images every minute \* 12 camera traps = 18.9 million images per year.
  + 500kb per image \* 18.9 million = 10TB (within Aurora limit)
  + $0.133 per GB a month at 10TB (including metadata) = $1,330 per month and
  + $15,960 per year
  + $15,960 + 1 billion I/O requests = $18,160 per year
  + Total cost $18,160 + 1704 = $19,864 or approx. $20,000 allocated to the database.

**Useful links**

Example report: <https://cs.uwaterloo.ca/~apidduck/se362/Assignments/A2/fordexample.pdf>Email retrieval: <https://aws.amazon.com/getting-started/projects/setup-email-receiving-pipeline/>TensorFlow information: <https://www.tensorflow.org/learn>

Tensorflow example: <https://www.youtube.com/watch?v=HS7U6IugXmE>SQL relational database (good for sorting/filtering): <https://aws.amazon.com/rds/aurora/>WTF does Methodology refer to: <https://blog.planview.com/top-6-software-development-methodologies/>

Using AWS Simple Email Service to intercept incoming emails:  
<https://www.youtube.com/watch?v=nxXIpPZzMd0>