1. **General Goals**

The main objectives of the project involve developing a database/repository to house data collected from research and commercial activities, conducted by students, researchers and the Farm manager, at the UWA Farm Ridgefield. The data being stored can have various formats and should be accessible by those with an account within the application.

1. **Current System**

Currently, the data that has been collected thus far, has not been coordinated and/or made accessible to everyone operating at the farm. Thus, a central system is required for data to be viewed easily by any of the users.

1. **Proposed System**
   1. **Overview**

The application will aim to tackle two main problems:

* Management of Ridgefield Data:
  + Data should be sorted based on the year, from 2009 onwards.
  + There should be a capability to add data to any year as needed by a user.
* Management of Ridgefield Paddock Data:
  + Files stored on the repository will contain data on each paddock at the UWA Farm Ridgefield, including data on yield, cropping history, grazing history, soil data and tree planting history.
  1. **Functional Requirements**

The application should have the following functional requirements:

* User interface to upload and download data.
* User interface to monitor edits/changes to ensure data is not accidently deleted or incorrectly edited.
* Back up data for a period of time.
* User login and profile, including a name, email address, mobile phone number and role (UWA staff, UWA student, other etc.).
* Security of data/personal details.
  1. **Non-Functional Requirements**

The application should have the following non-functional requirements:

* Performance
* Scalability
* Reliability
* Security
* Usability
* Data Integrity
  + 1. **User Interface and Human Factors**
* Type of user:
  + Users with different sets of skills, from novice to expert, will use the application.
* Number of users:
  + It is expected that the system will have around 40 – 50 users but likely scale to 100 users.
* Type of training required for users:
  + Training would involve teaching a user how to create a profile, how to sign in, how to upload/download data.
* Usability:
  + The system should be easy to learn since users will have a range of expertise with online applications.
* Error prevention:
  + Edits/changes should be tracked by the application to prevent deletion of data from the database/repository.
* Input/output devices:
  + No peripherals will be required for the current application.
    1. **Documentation**

The required documentation and audience are as follows:

* Git source code of the application, tools/hardware requirement, setup instructions: used by future developers of the application.
* Basic app and team information: used by user.
  + 1. **Hardware Considerations**

In terms of hardware, the following considerations will be needed:

* Browser compatibility: application should work on most browsers without issues.
* Laptop/computer compatibility: application should work on the most basic of laptops/computer system requirements.
  + i3 or equivalent processor.
  + Minimum 2 GB RAM .
  + No auxiliary storage necessary.
    1. **Performance characteristics**

Most of the performance issues will be related directly to the speed of upload and download of files into/from the repository:

* Uploading and downloading data will be affected by the bandwidth.
* Ideally, at most, only a few users will be uploading and downloading data at once thus low-level multi-threading can be implemented if needed.
  + 1. **Error Handling and Extreme Conditions**

The application/repository will be able to prevent and handle errors in the following ways:

* Constraints on input:
  + For example, when selecting the data for specific years, options can be listed as opposed to direct user input, which can prevent invalid operations and reduce errors.
* Displaying error messages:
  + Ensuring errors are communicated to the user or developer are of utmost importance and can prevent any issues from occurring beforehand.
  + For example, having an error message appear when the email inputted into the sign-in page is incorrect or already in use can help prevent duplicated data and/or incorrect inputs to be stored within the database.
  + Displayed error messages can also be customised depending on the type of error e.g. ‘Invalid Input’, ‘Try Again Later’ or ‘Please contact support’.
* Exception handling:
  + The application/repository should have a series of exception handlers within the code that allow for errors to be caught effectively.
  + Basic exception handlers include ‘ObjectDoesNotExist’, ‘FieldDoesNotExist’ and ‘ValidationError’. A statement or error message can be printed/displayed to notify the user or developer for these issues.

* + 1. **System Interfacing**

The application/repository will not rely on external systems, inputs or outputs. The user will interact solely with the application and the information contained within. Restrictions on inputs into the repository will be quite limited since the application will store data of different formats, including word files, excel spreadsheets etc.

* + 1. **Quality Issues**

Requirements for reliability of the repository will be as follows:

* Well-designed application: the architecture of the repository should be intuitive and the features should be easily accessible.
* Security: Sensitive data may be stored within the repository and hence, it is important to ensure that hacks or breaches to the data stored do not occur and/or are minimised.
* Speed: the application and the subsequent pages e.g. sign-in page, sign-up page, main page etc. should not take too long to load.

Portability of the repository is important in the event that users are operating on different hardware systems. Thus, the portability of the application should allow the repository to be accessed on:

* Different browsers.
* Different computer/laptop devices.
  + 1. **System Modifications**

Most of the future modifications will be either visual changes to improve overall ease of access to features or the addition of quality-of-life features including:

* Search function: allows for searching of different files within the repository, likely based on keywords or tags.
* Visual aid: allowing for changes to the colours used in the application to accommodate disabilities e.g. colour blind mode.
* Database expansion: depending on the number of users, the database may need to be expanded into a data warehouse to allow for more information to be stored.
  + 1. **Physical Environment**

The target equipment, which includes laptops and computers, will operate either on the farm or remotely. However, the environmental conditions should not be very abnormal, although some differences may occur when operating on the farm as opposed to remote operation, when using the equipment.

* + 1. **Security Issues**

The application/repository should be impervious to data breaches or hacks and should have the following features:

* Authentication: verifying the identity of a user.
  + Hashing and salting: process to convert passwords into fixed-length values that cannot be reverse engineered.
* Authorisation: access to data or permissions to edit sensitive information should be granted to specific users and not the general public.
* Prevention of SQL injection: preventing private information to be accessed from SQL attacks into the database.
* Prevention of cookie poisoning: ensuring that cookies are not altered by third parties to steal personal data from the database.
  + 1. **Resource Issues**

It is expected that the database will be saved automatically by the system every few months to ensure that in the event that the current database/repository is compromised, a backup will still exist to allow user access to previous versions of the data.