**Scope of Work – CITS3200 – Group 24**

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.

* 1. **Constraints**

There are multiple constraints that will determine how our team approaches the project. They have been divided into two categories, system constraints which are determined by the requirements of the finished system and development constraints which are factors which will affect the development process.

* + 1. **System Constraints**
* **Accessibility**: The system should only be accessible by those who have permission and there must be a process for granting access to new users.
* **Usability:** The system should present the data in an easy and intuitive visual interface.
* **Data compatibility:** There is no current system in place however there is a large amount of existing data from different sources which the system must be capable of storing. The system must also be compatible with other data types which may be needed in the future.
* **Data management:** The system will need to allow for data to be created, updated, read and deleted.
* **Capacity:** The system will need the capacity to store various data dating back to 2009 and for 30-40 current users, the system will also need to plan for future growth.
* **Availability**: The system will need to be available at all times or as close to it as practically possible.
* **Accountability:** The system will need to log changes within the system so that any changes by a user must be accountable.
* **Redundancy:** The system will need to allow for changes to be undone if required or for the recovery of lost data.
* **Security:** There will be private and confidential information stored in the system, this must be secured as much as possible.
* **Maintenance:** The system will need to be designed so that it can be maintained by whoever has control of it in the future with as little technical knowledge as possible. If any changes need to be made to the system, it should be easy to determine how to effect those changes on the system.
* **Device compatibility**: There are no standard devices from which the system will be accessed from. The system will therefor need to be compatible with as many devices as possible and allow for a consistent experience across all of them.
* **Existing system compatibility:** It is not clear as of yet how the system will be deployed but it may need to be compatible with UWAs existing hosting solution or at least flexible.
  + 1. **Development Constraints**
* **Timeframe:** The project will need to completed over one semester which will limit options, especially with areas of the project which will require additional learning. As it is being completed alongside other full-time study requirements, the number of available workhours will be limited and varying.
* **Existing knowledge**: Our team comes from a variety of different backgrounds and varying levels of familiarity with web development. We will have to carefully select frameworks and libraries in order to maximise use of our existing knowledge and cut down on learning time.
* **Project requirements**: In addition to the development of the product, the team will have to consider assessment requirements for the Professional Computing unit. Requirements such as work needing to be evidenced by GitHub commits will rule out certain development pathways.
* **Team** **member** **coordination**: It will be a challenge to coordinate work with team members who will have differing schedules and availabilities. Assigning roles and responsibilities so that everyone has the chance to contribute equally despite different areas of knowledge will be a challenge and these must also be rotated often. In addition, not all team members are located in Perth and having to coordinate work with remote team members will offer different challenges.
* **Budget:** There is no budget allocated to the development of the project so any option must be available without cost. It is also important to consider whether there are any costs associated with products when they are no longer used for personal or educational use.
  1. **System Model**
     1. **Scenarios**

There are many possible scenarios but the following have been selected to represent intended use and necessary requirements and presented in user-story format.

1. As a **potential user**, I want to have a clear pathway of gaining access to the repository so I can use its functions.
2. As a **potential** or **registered user**, I want only the required personal data to be stored and any confidential information that I input should be kept secure.
3. As a **registered user**, I want to be able to have a way to reset or change my login details in the scenario in which they have been forgotten or that I require for them to be changed.
4. As a **registered user**, I want to be able to upload, update, view or delete a file so that the information required is stored in the repository and it shouldn’t be a tedious process.
5. As a **registered user**, I want to be able to locate a required file quickly so I can perform the needed action on it.
6. As a **registered user**, I want the data to be sorted logically so that I can locate files related to what I require even though I may not know specifically what I need.
7. As an **admin user**, I want to control who has access to the repository so that data is only able to be accessed and modified by those who are authorised to.
8. As an **admin user**, I want the ability to restore or override data modifications and deletions in the event they were performed accidentally or maliciously.
9. As an **admin user**, I want the ability to monitor changes and track accountability so that in the event of an unintended use of the system I can prevent it or investigate what happened.
10. As an **admin user**, I want the ability to change parts of the system as they are needed and it shouldn’t be too complicated to do so.
    * 1. **Use Case Models**

There will be 3 types of actors who will interact with the system:

* **Potential Users**: Site visitors who have not yet been granted access to the repository.
* **Registered Users**: Approved users who have access to the repository.
* **Admin Users**: Users who have elevated permissions, who control access.

**3.5.2.2 Use Cases**

Diagram

Description automatically generated

**3.5.3 Object Models**

**3.5.3.1 Data Dictionary**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** | **Example** |
| UserID | Integer |  | Identifier for each user | 1 |
| FirstName | Text | 20 | First name of user | John |
| LastName | Text | 20 | Last name of user | Smith |
| Email | Text | 30 | Email address of user | jsmith@email.com |
| Phone | Text | 10 | Contact number of user | 0123456789 |
| Username | Text | 20 | Username for logging in | jsmith |
| Password | Text | 20 | Password of user, stored as hash value | 5e88489… |
| Role | Text | 20 | Role of user at UWA/Ridgefield | Researcher |
| Admin | Boolean | 1 | Whether user has admin permissions | False |
| FileID | Integer |  | Identifier for each file | 2 |
| File | BLOB |  | Actual file stored |  |
| Filename | Text | 20 | Name of the file | Example.docx |
| Modified | Timestamp |  | When file was last modified | 14/08/2022 19:00 |
| PaddockId | Integer |  | Paddock identifier | 1 |
| PaddockName | Text | 20 | Name of paddock | Paddock1 |
| Year | Integer |  | Year that file is associated with | 2022 |
| TagId | Integer |  | Tag identifier | 1 |
| TagName | Text | 20 | Keyword which can be associated with file | Soil |

**3.5.3.2 Class Diagrams**

**Diagram

Description automatically generated**

**3.5.4 Dynamic Models**

**Diagram

Description automatically generated3.5.4.1 Activity Diagram**

**3.5.5 User Interface**

Graphical user interface

Description automatically generated with low confidence

Graphical user interface, application

Description automatically generated