## Time spent:

Mock service and data - 1.5 hrs

Development and refactoring - 9 hrs

Finalising build for submission, and summary doc - 1 hr

## Summary:

The application was developed using the most recent version of Angular v5.2 via the Angular CLI (<https://www.npmjs.com/package/@angular/cli>) toolset, designed to assist with rapid development of Angular single page applications. Packages were installed and managed using Node Package Manager (NPM).

Bootstrap CSS library (<https://www.npmjs.com/package/bootstrap>) was implemented to assist with prototyping, and to make use of its responsive components for accessibility on a range of devices and device widths.

Git was used for version control throughout development, in combination with a private GitHub repository.

GitHub pages was then used to host the application, as static hosting is free with each repository and no server back end was required of the application. Additionally, it afforded fast-deployments of app builds through the use of another CLI tool (<https://github.com/angular-schule/angular-cli-ghpages>).

A data access service was made to generate the mock data in memory on initialisation of the service. Class models were set up for each of the required entities: Miller, Farm, and Paddock, and the mock-data was populated using the FakerJS library (<https://www.npmjs.com/package/faker>).

The DataService class was declared as an injectable at the application level, rather than per-component, to ensure a singleton service would be accessible across all app components.

CRUD operations that return promises were used in the data-access service to mimic asynchronous interaction with a HTTP Web-Service / API.  
  
Issues, interpretations, and decisions made:

The requirements requested the use of a MVC/MVVM framework for the application, which was taken into consideration initially and the project was going to be developed using Angular 1/AngularJS. This was overlooked when an early change was made to use the latest version of Angular, which does not technically follow MVC/MVVM by design (and instead uses a component-based architecture like that of React).

An issue was encountered early with how to implement the mock data service. The initial interpretation of the requirement was that the app should both generate and modify data in memory only, and this is the method that was implemented. Upon later reflection it became apparent that the data could have possibly been generated as a separate JSON file and then retrieved and stored in-memory, to provide a more consistent dataset for testing. Due to time constraints, it was decided that the original implementation was sufficient.

Entity array indexes were used as URL parameters for farms and millers, due to other identifier properties being generated at random by the application. Referencing the Farm’s ‘code’ field as it’s identifier meant that the code (a randomly generated UUID) would be re-generated on page refresh and the URL reference would no longer be valid.

Initial setup was fast, as NPM and angular-cli handled much of the overhead with generating initial project structure and management of required packages. Despite this, development took longer than expected in several areas due to unfamiliarity with Angular 5 and the component-based architecture in general. Also, as a result, there are some areas of the application that do not make use of the benefits this architecture provides. For example, the forms and validation required for adding or editing a farm are currently duplicated within the add-farm and edit-farm components. A better approach would likely be to extract both into a single component and re-use in both situations. Due to time and availability constraints on the project, such areas were not revisited before submission.

Two changes were made shortly before submission to fix an issue encountered when attempting to run the application locally in the browser (from [file:///](NULL)). An existing ticket for this issue was found on GitHub, lodged against the angular repository, here: <https://github.com/angular/angular/issues/13948>. The base href in index.html was replaced with a document.write script, and hash routing was enabled for angular routing, which successfully resolved the issue.

## Steps to run the application:

Hosted application: <https://taulelei.github.io/code-assignment-farms/>  
Hosted repository: <https://github.com/taulelei/code-assignment-farms>

* A hosted version of the application has been provided in the 'Hosted application' link above.
* A ZIP file of the built project has been provided (and can be accessed , opening the index.html file in a browser will run the application without any local HTTP server required.
* Alternatively, if any issues are encountered, the project can be served from a local HTTP server by performing the following steps:

1. Ensure NodeJS is installed on the local machine
2. Clone the repository from the 'Hosted repository' GitHub link provided above
3. Open CLI and change to project directory
4. Run command: *npm install*
5. Run command: *ng serve --open*

A browser window should now open to <http://localhost:4200/> with the application running.