Farming App - Built with the Ionic 3 Framework

Gary Mannion
Derrick Conway

B.Sc.(Hons) in Software Development

February 7, 2019

Final Year Project

Advised by: Patrick Mannion

Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)

Contents

List of Figures

Abstract

For our final year project, we were looking to create an industry stander application that would help farmers in day to day life. There has been a lot of talk in recent weeks about the escalating fodder crisis and the negative impact it is having on farmers and their livestock. For months, farmers have faced difficult farming conditions due to persistent cold and wet weather. Farmers usually purchase enough fodder, dried hay or feed given to cattle and livestock, to last until the spring when the grass begins to grow, and animals can begin to eat that instead. As a team we want to design a free web application where the user can enter data and store it, do out calculations for the fodder months. There is also AI section for keeping track of the herd in the calving months, tagging section for keeping track of new born animals, section for keeping track of the medicine used on the herd throughout the year. We have created this application with a client and server pulling data from databases. We created a three-tier application, using Mongo Db and Firebase as our Data Tier, NodeJS for our Logic Tier and Ionic 3 for our Presentation Tier. Adding specific features such as, adding Al, Tagging, Feed, Madison and creating a message board for farmers to group together and find solution's to problems there are encountering. it was our objectives by gearing our app specifically for farmers.

Authors:

- **Derrick Conway**
- **Gary Mannion**

Acknowledgements

We would like to acknowledge our project supervisor Patrick Mannion for his help and supervision during the creation of this appliction.

We would also like to thank John Healy, Head Lecturer of the Applied Project and Minor Dissertation module.

Chapter 1

Introduction

When choosing our project we wanted to pick something that was relevant to current everyday life. We wanted a project that also highlighted our existing skills and allowed us to learn new skills and develop as part of a team. With this criteria in mind we started brain storming ideas for our project.

We eventually decided on a App called herd list, herd list would be a farming fodder app. We recognized the current fodder crisis throughout Ireland and thought it would benefit farmers and farmers alike to have a specific fodder app for the farming population. Currently farmers in Ireland can find themselves in difficult situations for up to 1 or 2 months before finding proper fodder for animals. This is quiet common for farmers and with our app they would only be able for to calculate fodder for the winter and spring months and keep track of tagging, medicine, AI.

Having a fodder app as our starting point we started to research other apps and websites in the farming category. The main one we focused on was the Heard Watch. Heard Watch would be the most popular farming app in Ireland and it provided us with a great deal of insight on what our app would need to have and also on what heard watch lacks for the farming market.

We wanted a app that had essentially the same functionality as Heard Watch but which was geared more towards the stress points of farming We did this by focusing on what stresses farmers the most, We also created a forum for farmers to get in touch with one another, to either get information from one another or to give information, These would compliment the normal functionality of a accommodation app such as, publish ads (either fodder or rent land), search lists, view ads or messages and land for rent and edit and delete your personal ads. This would also require log in and registration functionality.

After deliberating over the objective of our application, we then turned to what technology we would use. We wanted to use a array of technologies

to show our capabilities but also gain more knowledge and become better programmers over all. We settled on a 3 tier structure containing a Ionic 3 presentation tier (front end), NodeJs logic tier (middleware) and Mongo/-firebase database data tier (back end).

From what we hoped to achieve from creating this app was the knowledge of new technology, personal growth through team building and of course to produce something of substance and something we could be proud of.

Chapter 2

Context

The general context of our application revolves around farmers having a platform which is geared towards tracking fodder, and there heard. farmers will be able to calculate amounts of fodder for the winter months and use the app to keep track of the tagging of new born animals, track medicines for animals throughout the year. And store AI information to which heard animal are in calf and how far alone they are gone and when they are due.

- Feed
- AI
- Medicine
- Tagging

2.1 Objectives

The main objective of our application is to help farmers who may be struggling with the current fodder crisis in Ireland. We also wanted to make it easier for farmers and farmers alike, who are finding it difficult to keep track fodder, AI, Medicine and Tagging. The following is a list of the main pages in our application along with the objectives for each page.

1. **Login/Register Page** Our application has a login and a register page. The register page allows the user to securely register with our application while the login page allows the user to securely login to our application. Once logged in, the user has access to extra features not accessible to unregistered users.

2. **Forgot-Password Page** The objective of the forgot-password page is to allow a user who has forgotten their password to enter their email address. They will then receive an email containing a link which will allow them to create a new password.

3. **Home Page** The objective of the homepage is that it is a base of navigation for the application. The homepage will also display if a user is logged in or not.

4. **Feed** The objective of this feed page is to display is to calculate out how much feed you need for the winter months by entering the number of animals on the farm and calculate out how much is needed.

5. **Tagging** The objective of this tagging page is to keep track of the heard animals calving by inputting the details of birth,gender and breed of the animal. This will allow the user to keep track in an easy way of filling in form details at there own personal time..

6. **AI** The objective of this AI page is to keep track of witch hear animals are in calf and how far gone they are and witch are not in calf this will allow the user to keep track of heard and when they are to calf buy filling in form details as it is being read out.

7. **Medicine** The objective of this medicine page is to keep track of all the animals medicine that have been taken, show when it was taken, how much was taken. This will allow the user to keep track of heard medicine buy filling in form details.

8. **Message Board** The objective of the message board is to have a place where farmers can communicate with each other, helping themselves and other farmers with problems and place ads for stock.

9. **My Ads Page** The objective of the Ad's page is to allow a user to view a list of their published ads. In here they can delete these ads or they have the option to edit their published ads.

2.2 Project Links

Link to Repository

- https://github.com/Gazza1996/Final-Year-Project-Applied-Diss

2.3 Chapters Review

This paper is broken down into different chapters, ranging from planning, design and development. The following sub-sections give a brief overview of each chapter in this paper.

2.3.1 Methodology

In this Chapter we discuss some of the methodologies we used during the various stages of creating the farming app. In this section we discuss Agile, version control, testing and sprints.

2.3.2 Technology Review

In the Technology Review chapter we review the various technologies we used in our project. We review back and front end technologies, development tools and version control.

2.3.3 System Design

In this chapter we give a detailed explanation of the overall system architecture and design of farmers app. We explain why we chose these specific technologies and how we implemented them into our system design.

2.3.4 System Evaluation

In this chapter we evaluate our system in the areas of Robustness, Testing, Scalability, Results against Objectives and Limitations.

2.3.5 Conclusion

In the conclusion we summaries our project against our goals and objectives. We review our application from the various stages and speak about possible future development.

Chapter 3

Methodology

In this chapter we discuss the methodologies used in our project. A methodology is just a way to plan and control the development process of a piece of software. There are various methodologies to choose from including Extreme Programming, Rapid Application Development or Waterfall but for this project we use Agile as our main methodology.

3.1 Agile Development

During the life cycle of our project we attempted to use an Agile like approach to the research, design and implementation stages of the project. In the initial stages of the project we discussed various methodologies we could use, for example Waterfall but we decided on Agile because of waterfalls lack of flexibility. Agile offers continual improvement, flexibility and incremental delivery of the software.

During the research and development process of our project we used a Scrum like approach. Scrum is an Agile method in which a development cycle is carried out in what are known as sprints. Section 3.4 and 3.5 contains more information on each sprint of the research and development phase of the project.

Throughout the life cycle of the project we held weekly meetings. In these meetings we would plan and discuss the features of our application. Before each sprint of our project, we used these meetings to plan our sprint cycle. Results from these meetings were then added as to-dos and added to a Git-Hub projects page. Git-Hub projects board help us prioritize and organize our work load. We found this to be an especially useful tool. Once an is-

sue was created it was added into the To-do phase. Then, when we started working on a problem it got added to in-progress, then finally into the completed section once the task was done. A snapshot of our Git-Hub Projects page can be found in Figure !!NEED TO INSERT IMAGE OF TODO LIST!!

We also had weekly contact with our project supervisor. Our supervisor was instrumental in advising us on our progress and helping us with issues with the project.

3.2 Version Control

Throughout the life cycle of our project we used GitHub. GitHub is a hosting service for version control. We created a GitHub repository, adding all the members of the team to work as collaborators. We found GitHub to be an extremely useful tool in the research and development of Farmers app. Even though we primarily used GitHub to manage source code, we also took advantage of some collaboration features that GitHub offers. We used wikis to track what we learned in the research stages of our project. A wiki is a website where multiple users can modify content directly from their browser. We used GitHub's task management tool when a task was set and then used this tool to track that tasks progress. We found this tool particularly useful as it helped break down the workload into small manageable tasks.
As a team, working with source code on GitHub proved to be its most valuable feature. Git-Hub allows all members of the team to see all the commits made to the repository. We can see every update, previewing every change that was made and even rolling back a commit if necessary.

3.3 Testing

For our application, We needed to make a decision on how we would test the application as we did not choose to use a framework (i.e. J-unit) to test our application, we did however decide to use both white and black box testing.

3.3.1 White Box Testing

White box testing is a way of testing software where the tester can see the internal workings of the software. White box testers have full knowledge of the internal makeup of the software and are usually software developers themselves. In our application, to white box test, one team member would write

a feature (e.g. login) and then another team member would test that functionality. That team member would present input, follow the path through the code and then examine the output. A diagram of white box testing can be found in Fig 3.3.1

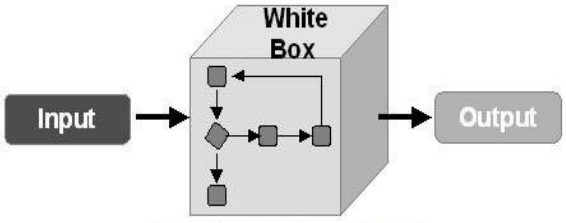

Figure 3.3.1: White Box Testing

3.3.2 Black Box Testing

Black box testing is a way of testing a piece of software without knowing the internal functionality of the software being tested. A black box tester has no knowledge of the internal design, structure or implementation of the software and are often not programmers themselves. In our application, to black box test, we would ask friends to use our application e.g. asking them to login to the application and to maneuver through the application and then reporting their experience. With this type of testing, the tester is trying to break your application looking for faults. We found this type of testing especially useful as it gave us an insight into the way a user would use our application, something we previously would not have thought of when developing this farming application. A diagram of black box testing can be found in Fig 4.1.4

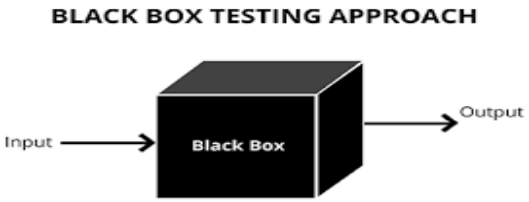

Figure 3.3.2: Black Box Testing

The following two sections contain information about the various sprints we undertook in the research.

3.4 Sprints

In this section we shale be discussing the sprites, Each of the sprints we completed during the research of our project gave us the opportunity to learn as a team and adapt to the various needs of the project. Each sprint represents a vital aspect of the application, i.e., creating three tier architecture, accessing native phone capabilities through Cordova and learning how to integrate all these features together. We prioritized the research sprints in the order as follows.

3.4.1 Sprint 1 Review Project

This was our first sprint and it was worked on by all members in our team. we researched simple app in which he explains the process of building a simple review app using three tier technology. On the front end a user can write a small review and then add a score. That information is then sent to a node server which stores the information in a MongoDB database. We felt it was very important that all members understood the process involved in connecting these three layers together so we all worked on this sprint together.

3.4.2 Sprint 2 Scanner

3.4.3 Sprint 3 Image

3.4.4 Sprint 4 Login/Register

In this sprint, implemented by Patrick, a small mobile application was developed where login and registration with Firebase was implemented. We are using Firebase in our farming application to securely authenticate our users with email and password registration and login. The user also has the option of logging into our app via Facebook.

3.5 Development Sprints

3.5.1 Sprint 1 Create Pages

From the result of team meetings, we had decided on the number of pages our application would require. In the first development sprint, we created all the pages that we would need for our application, And added a few extra features to the application if we had time to do these extra features example, scanner and camera war the extra features. The sprint was completed by the members of this project and this formed the base skeleton of the farming application.

3.5.2 Sprint 2 Menu Navigation

The next step was to implement a menu into our app connecting all the pages together. The result of this sprint was so that each page could now be navigated to, by either the main menu or by a button on a page. This sprint was completed by the members of the project.

3.5.3 Sprint 3 Login/Registration

For this sprint it was now time to implement the registration and login functionality to the farming application. This was successfully done by a member of the project that implemented this sprint as he had previous knowledge of implementing this functionality with Firebase from a previously completed research sprint.

3.5.4 Sprint 4 Implementing the Back End

In this sprint we implement our NodeJS server running on one of our AWS instances. This sprint was completed by members of the project and in it we set up our API routes along with MongoDB to store text data in our database. The text data being stored would be the data from a created listing i.e. breed, tag number, Date of birth ect...

3.5.5 Sprint 5 Feed Calculator

In this sprite we worked on the how we would use the feed page, we decided to use it as a calculator to calculate out the amount of feed witch would be required by the amount of herd animals on the farm and working out the calculations for the pit and bales that will be required for the winter months

ahead. We first implemented the functionality on the front end. Allowing the user to enter the amount needed for the future months, and also show the amount of feed the farmer currently has.

3.5.6 Sprint 6 Tagging

In this sprite we worked on the how we would use the tagging page, we decided to use it as a way of keeping note of the heard animals calving and if there was any complications. We first implemented the functionality on the front end allowing the user to input data of the animal and storing it, the features of this page are to input the animal tag number, breed, date of birth and if there was any complication on the animal giving birth.

3.5.7 Sprint 7 Medicine

In this sprite we worked on the how we would use the medicine page, we decided to use it as a way of keeping track of the heard animals that have received medicine and note why they needed the medicine and how much was giving and the dates it was giving to the animal, also note if there was any complications. We first implemented the functionality on the front end allowing the user to input data of the animal and storing it,

3.5.8 Sprint 8 AI

In this sprite we worked on the how we would use the AI page, we decided to use it as a way of keeping track of the heard animals that are in calf, We first implemented the functionality on the front end allowing the user to input data of the animal and storing it, the information would be how far alone they war, and track with one are not in calf as of yet. so the farmer can track how many to expect for the winter and spring months.

3.5.9 Sprint 9 Message Board

The next task, implemented by the team, was the Message board. The result of this sprint was the creation of a forum in which users can communicate with each other by creating and posting messages. we first set up the backend to store these messages in MongoDB database and then displays the stored messages to the user. Message output includes the users name, message and date created.

3.5.10 Sprint 10 Camera

This sprint was worked on by all members of our team because taking and storing images is one of the prominent features of farm application. We first implemented the functionality on the front end. This allows a user to take a photo using the camera on their device or by selecting a photo from their devices internal storage. We then implemented our back-end API to store those images mapping each image/s with its unique ad id.

3.5.11 Sprint 11 Scanner

This sprint was worked on by all members of our team as we had no experience in the scanner side of it, the scanner is to scan the ear tag of the animal and show the information of the animal. this was unsuccessful as there war complications. We could scan the tag but needed a scanner to read the information from the tag, cause of this we could not get our hands on one as there war expensive and not many had them.

3.5.12 Sprint 12 Search

One of the last features of our application is the search functionality. Implemented by team members, user can narrow down the tag number information in the tagging page, medicine page and AI page. We first implemented the functionality on the front end allowing the user to input data of the animal and searching it.

Chapter 4

Technology Review

In this section, we will cover the overall design and architecture of our Application. We will do this through code snippets and visual aids to help you get an understanding of the Application design. The System Design chapter will be broken up into three parts. Data Tier represented by our Databases Mongo DB and Firebase, Logic Tier represented by NodeJS servers, Presentation Tier which will be represented by Ionic 3 App and our deployment of our app.

4.1 Data Tier

The data tier represents the overall stored data and gives the ability to store, access, update and delete certain data. For our application we have two databases. One database is used to store our user accounts and the other is used to store our users information and messages.

4.1.1 Firebase Authentication

We chose Firebase to handle our user accounts as it was established in the ionic community for being a good database for login and registration functionality. The Firebase Authentication SDK provides methods for creating and managing users who use email and password to log on to the system. Firebase uses tokens to allow users to login/logout and to register to an Application. The firebase documentation states that the listener receives notifications in the following:

1. When the FirebaseAuth object completes the initialization and the user is already authorized in the previous session or is redirected from the input stream of the other authentication provider.

2. When the user logs in (the current user is installed).
3. When the user exits (the current user becomes null).
4. When the access token of the current user is updated. This can happen if the token expired.
5. The user re-authenticates
6. The user changes their password

If this is the case. Firebase will issue new tokens to the new user and make the old token used as invalid. A great part of Firebase Authentication is that if a user changes its details i.e password, they are automcatically logged out of all devices they have accessed with the old account details.

4.1.2 Mongo DB

We chose Mongo DB to store our clients ads, host images and forum messages as it offers a lot of flexibility. When developing a product no matter how much planning one does, requirements can always be subject to change. With this in mind we chose Mongo as it would allow us to change the structure of database as the project evolved. Mongo allows you to download and host your own Mongo DB server for free. This download comes with its own terminal which you can use to query your database which was very effective tool during development.

4.1.3 PouchDB

We both agreed to use PouchDB as the Database for our medicines page in the application. We decided on this as to have a varied option of Databases as we are also using MongoDB in another area of our System. PouchDB is simply just an API that we use in the System design so we can make a call to our actual Database on CouchDb, which I will discuss further after this. PouchDB is a very effective API to use with ionic as it can be imported through our Node command Prompt and all we need is the name of our CouchDB database and where it is being stored on the LocalHost.

4.1.4 Apache CouchDB

Apache CouchDB lets you access your data where you need it. The Couch Replication Protocol is implemented in a variety of projects and products

that span every imaginable computing environment from globally distributed server-clusters, over mobile phones to web browsers.

Store your data safely, on your own servers, or with any leading cloud provider. Your web- and native applications love CouchDB, because it speaks JSON natively and supports binary data for all your data storage needs.

The Couch Replication Protocol lets your data flow seamlessly between server clusters to mobile phones and web browsers, enabling a compelling offline-first user-experience while maintaining high performance and strong reliability. CouchDB comes with a developer-friendly query language, and optionally MapReduce for simple, efficient, and comprehensive data retrieval. [1]

,

Figure 4.1.4: Apache CouchDB

4.2 Logic Tier

4.2.1 NodeJS

NodeJS is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). NodeJS was developed by Ryan Dahl in 2009 and its latest version is v0.10.36. NodeJS is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. NodeJS uses an event-driven, non-blocking I/O model that makes it lightweight and efficient,

perfect for data-intensive real-time applications that run across distributed devices which makes it perfect for use in our Application. [2]

4.2.1.1 npm

NodeJS Package Manager is the package manager that is part of the NodeJS. Now, instead of installing each library separately, we can run one command and install everything in one go.

```
npm install
```

When you run the command, npm will look for the package.json file in the current folder. If found, it will install each library from the list. [3]

4.3 Presentation Tier

4.3.1 Ionic

4.3.2 Angular

4.4 Languages

4.4.1 Typescript

4.4.2 HTML

4.4.3 SCSS

4.4.4 JavaScript

4.4.5 LaTeX

4.5 Software and tools used

4.5.1 Visual Studio Code

4.5.2 GitHub

4.5.3 OverLeaf

Chapter 5

System Design

Chapter 6

System Evaluation

Chapter 7

Conclusion

Chapter 8

Appendix

Project Source Code Link:

https://github.com/Gazza1996/Final-Year-Project-Applied-Diss

Bibliography

[1] "Apache couchdb," 2019. Available: http://couchdb.apache.org/, [Accessed: 7- Februrary- 2019].

[2] "Nodejs," 2019. Available: https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm, [Accessed: 7- Februrary- 2019].

[3] "npm," 2019. Available: https://docs.npmjs.com/about-npm/, [Accessed: 7- Februrary- 2019].