Approach:

Since the requirement document states that the Application is being used by the scientist for her research work, so it was obvious to think in admin-user way. I divided the users in admin and non-admin. Each user can see their previous entry and make new one. If they don’t want to use the app regularly, they can schedule the data entry. On the other hand, admin should have the privileges to see all the entries made by all the users along with a dashboard when she can perform analysis on data submitted. Along with this only admin should have the access to create new user feature. So I started with simple landing page and started adding components for edit, create dashboard etc.

Technology Used: MERN stack with AG-grid for dashboard reporting/analysis. First decided to build this on MEAN (Mongo, Express, Angular & Node) stack but since this was a small project and Angular being a huge framework would have been overkill for this, so decided to go with ReactJS. Also, I have only worked on hobby/side projects which involved react as Frontend, so it was a good challenging and learning opportunity. Also, in my personal experience small React Apps load much faster than angular. Mongo is any day a good choice for database if simple crud operation are required by application. Node being easy to use with express was first choice to use as middle-tier (api).

High-level Component design:

List/Edit/Delete View

Reporting Dashboard (Admin)

Create User (Admin)

Create/Edit Feed Data

Login Module

Mongo DB (Atlas Cloud)

Navigation Module

NodeJS(API)

Main App

Database model diagram:

**FeedData**

**\_id(ObjectId)  
Username(string)  
feedTime(datetime)  
feedLocation(string)  
foodType(string)  
foodCategory(string)  
numberOfDucks(int)  
foodQuantity(int)**

**User**

**\_id(ObjectId)**   
**username(string)**  
**password(string)**

**Schedule**

**\_id(ObjectId)**   
**username(string)**  
**scheduled(boolean) feedTime(datetime)  
feedLocation(string)  
foodType(string)  
foodCategory(string)  
numberOfDucks(int)  
foodQuantity(int)**

Number of hours spent:

9 hours (plus 3 hours debugging silly Heroku deployment errors).

Future Enhancement/Good to have features:

* Allow admin to remove and update users.
* Allow users to request reset password in case they forget the password.
* A common dashboard showing charts and table (read-only data) to all the users to understand the trends of duck feeding across the globe.
* Restricting the data entry by changing fields to dropdown from input wherever necessary (e.g., food type and category).
* Allowing admin to cancel all the scheduled jobs at one click.
* Allowing non admin users to download their data in excel/csv format.