Qazi Dairies

Dairy Farm Management System

SPROJ Report

LUMS
A Not-for-Profit University

Furqan Athar	2022-10-0070
Khawaja Junaid	2022-10-0072
Abdullah Saleem	2022-10-0125
Saad Qadeer	2022-10-0209

Advisor: Sir Waqar Ahmad

Co-Advisor: Sir Basit Shafiq

School of Science and Engineering
Lahore University of Management Sciences

Submission Date: 23-April-2022

Acknowledgment and Dedication

We (Group 04) would like to express our special gratitude towards our supervisor and mentor Sir Waqar Ahmad and our Co-Supervisor Sir Basit Shafiq for their guidance, feedback, and support they offered at every step of the project, without which this project would have not been possible.

Furqan Athar

Khawaja Junaid

Abdullah Saleem

Saad Qadeer

Date: 24th May, 2022, Tuesday.

Certificate

I certify that the senior project titled “Add project title here” was completed under my supervision by the following students:

Furqan Athar

Khawaja Junaid

Abdullah Saleem

Saad Qadeer

and the project deliverables meet the requirements of the program.

---------- Date:

Advisor (Signature)

---------- Date:

Co-advisor (if any)

Table of Contents

List of Table and Figures 8

1. Introduction 9

   a. Introduction 9

   b. Objective and Scope 10

   c. Development Methodology 10

   d. Contributions 11

2. System Requirements 12

   a. System Actors – Table 2(a) 12

   b. Functional Requirements – Table 2(b) 13

   c. Non-functional Requirements – Table 2(c) 15

3. System Architecture 16

   a. Architecture Diagram 16

   b. Architecture Description 17

      1. Presentation Layer 17

      2. Business Layer 17

      3. Persistence Layer 18

      4. Database Layer 18

   c. Justification of the Architecture 19

   d. Tools and Technologies 20

4. Requirements Specifications 23

   a. Use Cases 24

      1. Registering Dairy Farm 25

      2. Add/Edit managers 26

      3. Add Animal 27

4. Filter Daily Income and see trends 28
5. Add Inventory Category 29
6. Add items via Inventory Category 30
7. Edit Existing Inventory 31
8. Edit/Add/Remove Employee(s)(data) 32
9. Add daily milk production 33
10. Filter Milk Productions 34
b. Class Diagram 35
Description of Class Diagram 36
c. Sequence Diagrams 38
1. Login for Super Admin (Product Owner) 38
2. Change Access for Tenants 38
3. Register Account for Dairy Farm 39
4. Addition of team member/manager to team 39
5. Update Account Information of a dairy farm owner 40
6. Add new Animal 40
7. Revenue and Revenue Trends 41
8. Add new categories to Inventory 41
9. Add Items to inventory 42
10. Edit/Delete – Use Inventory items 42
5. Software Development Methodology and Plan 43
a. Software Process Selection 43
b. Gantt Chart 45
6. Database Design and Web Services 46
a. Database Design 46

Diagram 46
Description 47
b. API Specification 48
7. System User Interface 49
a. Domain Registration 49
b. Farm Registration and other details 50
c. Login Screen 51
d. Dashboard 52
e. Adding Animals 53
f. Adding Milk Production 54
g. Adding Customers 56
h. Inventory System 58
i. Milk Supply 59
j. Daily Farm Expenditure 61
k. Assigning Work Roles and Building a team 62
8. Project Security 64
a. Project Threats 64
b. Potential Losses 64
c. Security Controls 65
d. Static and Dynamic Security Scanning Tools 65
9. Risk Management 66
Potential Risks and Mitigation Strategies 66
10. Testing and Evaluation 68
11. Conclusion 70
a. Summary 70

Brain Storming and Idea finalization 70
Finalizing the Framework 70
Documentation 70
Division of Labor 70
Sprint Plan 71
Feedback and Setting Deliverables 71
b. Challenges 71
The Hybrid Nature of Communication 71
Finding Free Services 72
Dependencies and Packages 72
Debugging 72
c. Future Works 73
12. Deployment Guidelines 74
13. Review checklist 75
14. References 76

List of Table and Figures

Tables

System Actors ……Table 2(a)

Functional Requirements……Table 2(b)

Non-Functional Requirements……Table 2(c)

Risks and Mitigation Strategies……..Table 9(a)

Figures

Architecture Diagram……....Fig 3A

Use Case Diagram…….Fig 4.1

Class Diagram……...Fig 4.2

Sequence Diagrams…….. Sec 4(c)

Gantt Chart……Fig 5.1

Database Design……Fig 6.1

1. Introduction

a. Introduction

In attempts to modernize the dairy industry in Pakistan, one of the greatest producers and consumers of milk and other dairy products in the region, our proposed and developed system will present a solution to manage the dairy farms better, optimize sales, and help in better record keeping.

Furthermore, multiple dairy farm owners will have the opportunity to set up their accounts on our management system, as we aim to develop a multi-tenant software, which will aid them in getting more customers and for the better management and record-keeping of their farms. The system will allow the customer to keep track of daily milk production, sales, expenses, and maintenance. Moreover, the system will also provide an interface for the dairy farm customers to check their delivery and monthly invoice.

Potential users of our management system include dairy farm owners (who wish to digitize their records and keep an updated track of their farms) and the customers who want to order dairy products, be it for domestic or commercial use.

The primary purpose of this product will be to provide multiple tenants a way to see various trends in their milk production, expenses, and income streams-allowing them to make better decisions for the future.

The target users will be able to use our platform in the form of a Web Application.

The domain of the Application

- Dairy Farms Sector of Pakistan

Target Users of the Application

- Dairy Farm Owners
- Dairy Farm Employees
- Dairy Products Customers

Type of Application

- Web Application

b. Objective and Scope

In this subsection, please define the objectives of your application, why you chose to develop this application and how it would impact/enhance the business operations in the selected domain.

As stated in the former section, Pakistan is one of the largest producer and consumers of dairy products in the world; we felt that this market was largely untapped in the digital world. Dairy Farms were operating in every locality and were providing dairy products to millions of people around them without any form of organization, record keeping, or optimization. For example, no record was kept of how many liters of milk were being produced and how much was being sold. And Local Dairy Farm owners were only capable of delivering at a small commutable distance; hence their sales were significant, and they were not generating a large amount of revenue. We felt that the conditions of the local dairy farms could be improved by providing them with a digital solution and allowing them to become capable of increasing their outreach and reaching more customers. In addition to this, they would be able to perform better record keeping and keep track of their sales, production, stocks etc.

c. Development Methodology

The development methodology that we used to develop this project was Agile (Scrum) The Justification, which has been discussed in more detail in section 15.

d. Contributions

Our project aimed to digitalize the local dairy industry of Pakistan, since technology is scarce at the level of the local dairy farm owners, we have helped them incorporate technologies and use it to fulfill their tasks that they had problems with previously such as data logging, data insights from the production or supply streams of the farm and managing different entities across the farm. Previously the local farmer was not able to visualize their milk production and had to rely on manual methods to extract trend from the tabular data, our website portal allows them to visualize their production data for their farm, enable them to see the revenue trends which are crucial for them to thrive in the dairy market given the large number of competitors operating in the space. Apart from the data visualization the farm owners can manage their employees and manage access controls to them based on their roles, they can also maintain custom item inventories for which they do not need a separate place to log their production, subsequently other major logistics can be maintained through their portal.

The website is better from its competitors in various ways primarily the use cases and the fulfillment has been made out from the farm owner directly, so the portal is customer centric and seeks to automate tasks for them via the technology accessible with simplistic design, so they are able make best use of the website. Since it is a multi-tenant system there are different farms that can maintain their portals on the website and stay at par with their competitors. The portal's consistency makes it easier for the farm owners to navigate through the website and perform their tasks. The portal seeks to take part in most operations at the farm of different types such as for animal, farm items and the workers which makes it a one place for all their essential operations without hassle.

2. System Requirements

System requirements identifies the functionality that is needed by a system in order to satisfy the customer's requirements. The system requirements will cover the system actors, the functional requirements, and the nonfunctional requirements.

a. System Actors – Table 2(a)

Actor Name	Description
Product Owner	The product owner will be present at the top of the hierarchical structure. The Product Owner will ultimately decide which dairy farm owners can set up their accounts on their product.
Admin	Admins are the dairy farm owners/Companies themselves. A dairy farm owner will be able to set up their account/dashboard on the main product interface and is also known as the tenant admin.
Manager	The tenant manager will be the person that would be next in authority after the farm owner. The manager will be the one overlooking the logistics of the farm and is also known as the tenant manager.
Customer	The customer will be the actor that will be present at the lowest level of this hierarchical structure. The customer is the one who will get the milk supply and will be able to check his portal daily.

b. Functional Requirements – Table 2(b)

Sr#	Requirement
1	As a Product Owner, I want to manage which dairy should access my management services.
2	As an admin, I want to register my company into the provided service.
3	As an admin, I want to add multiple admins/managers in my team, and they should be able to login using their own details.
4	As an admin, I want to deactivate the manager's role in my company.
5	As an admin, I want to change my company details from the dashboard.
6	As an admin/manager, I want to add/edit/remove details of each animal added into our dairy farm and need to view information page of those animals
7	As an admin/manager, I want to keep track of the status of each cow. Their date of birth, vaccination dates, medication, and other dates.
8	As an admin/manager, I want to manage different categories in my inventory like medicines, animal feed and other inventory items.
9	As an admin/manager, I want to add items under my categories.
10	As an admin/manager, I want that if some medicine is given to an animal, then that record should be visible under that respective item present in my inventory. That record should be in chronological order.
11	As an admin/manager, I want that if the stock of an item gets updated, it's record should be maintained and should be visible under the item's detail.
12	As an admin/manager, I want to add/edit/remove employees' data into my system.
13	As an admin/manager, I want to add customers' data into my system and maintain their statuses like active/inactive.
14	As an admin/manager, I want to maintain/add daily expenses like food expenses and medication expenses or through animal purchase etc. and want to specify, if that item used should be decreased from my inventory or not.

15	As an admin/manager, I want to see the list and details of all the daily expenses and should be able to edit them.
16	As an admin, I want to see the trend, using graphs, of daily expenses and want to filter expenses based on specified date or the range of dates.
17	As an admin/manager, I want to maintain/add/edit daily income, that could either be through milk distribution or by selling animals.
18	As an admin, I want to see the trend, using graphs, of daily income and want to filter them based on specified date or the range of dates.
19	As an admin/manager, I want to add/edit daily milk production of each milking animal on a single page, and I also want this data to be saved under each animal information card.
20	As an admin/manager, I want to see the trend of milk production of each animal based on date range specified.
21	As an admin/manager, I want to see the record of milk supplied to a customer on the customer detail page and filter the record between specified range of dates.
22	As an admin/manager, I want to apply filters on areas and check how many and which customers are in that area which are getting milk supply.
23	As an admin/manager, I want to compare the average milk production of each animal in between the selected dates so that I can know which animal is performing well.
24	As a customer, I want to login to my portal and see the milk supplied to me in a specific month or on a specific date.
25	As an admin, at the end of each month, I want to select the customers to whom I supplied the milk and send/release them the invoice of the month and should also be able to change the status of invoice paid or not.
26	As a customer, I want to see/download the invoice released to me and change the status of invoice on my portal.
27	As an admin, I want that if someone updated the status of invoice on his portal, I should be notified regarding this.

c. Non-functional Requirements – Table 2(c)

Sr#	Requirements
1	The system should not use extensive memory, not more than 1 GB memory at its time of execution.
2	The system should not fail more than 3 times for a given week and in case of failure must be available within 5 minutes.
3	Each page must load within 2 seconds after the user has made a request.
4	System should be secure against threats essentially the OWASP security standards which include input validation such as validation of data length or range, not disclose sensitive error information such as system details, input sanitation etc.
5	Privacy standards according to the GDPR be followed i.e. Not withholding personal data without the user's consent, limitation of data storage that would mean no extra data is collected from the user.
6	The system should scale and be able to handle more traffic, 500 users at peak times that would be from 10am to 5pm.
7	For the Usability of the system, the user must be able to easily navigate such as not going through a series of pages to make a payment slip request not more than 4 clicks for any request in the navigation.
8	The system should be expandable from the product owners' side so that new features may be added such as a personalized data analysis graph.
9	The help documentation for the system should be released in the commonly understood language English and Urdu.
10	The system should cater for faulty inputs and must load or prompt the user for a correct input such as a price calculator accepting numbers and not letters.
11	The system should require authentications for all resources and information that are intended to be public, the least privilege principle will be followed, and access granted would allow users to fulfill their purpose. As needed, access rights can be given or withdrawn and in case of failure, access controls should fail securely.

3. System Architecture

Our system is based on a layered architecture pattern which includes a presentation layer, business layer, persistence layer and a database layer. As our system is a multi-tenant system hence multiple dairy farms interact with the presentation layer. The multi-tenant system needed to be made in such a way that users have access to only their own data and their data remain safe and secure from other users. That is why a layered architecture was used so that the data is kept safe under multiple layers and all the layers have checks in place to allow only the right users access to their data.

a. Architecture Diagram

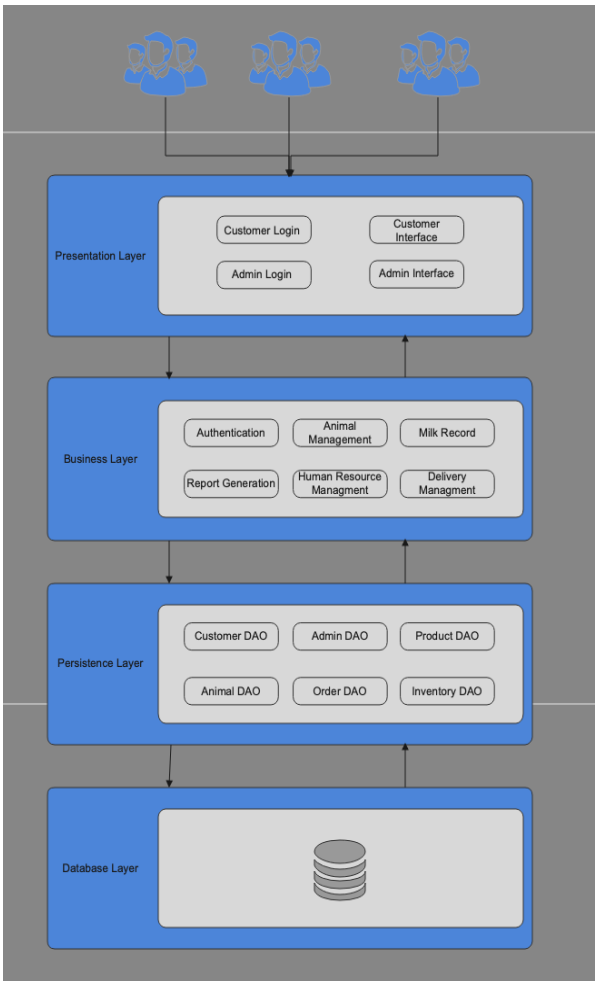


Fig 3A

b. Architecture Description

Details of different layers in the architecture diagram are given below:

1. Presentation Layer

The presentation layer is composed of the components that the user directly interacts with which are the user interface components. Our system has four basic user interfaces: Customer Login Portal, Admin Login Portal, Customer's Interface, and Admin's Interface. Both the login portals have almost the same architecture and design and communicate with the Authentication component in the business layer to allow customers and admins access to the software. The customer's user interface is user-friendly and has links to access the customer's monthly/annual reports/invoices. It encapsulates all the customer's requirements and is an independent component that can be modified and altered on its own without influencing other components. The admin user interfaces envelop all the administration requirements and lets the managers and the owners of the dairy farm perform their administrative tasks online such as generating monthly and annual reports and updating delivery and animal records.

2. Business Layer

The business layers consist of the logical components of the software which are used to implement the logic of the system. Our business layer includes six basic components which are further divided into subcomponents. The six basic components are: authentication, animal management, milk records, report generation, human resource management and delivery management. Authentication communicates with the login components in the presentation layers to authenticate users to grant them access to the portal. Animal management, milk records and human resource management components communicate with the admin interface component in the presentation layer and only the admin of a dairy farm can access these features to manage the resources of his/her dairy farm. The remaining two components, reports generation and delivery management, communicate with both, the customer's interface, and the admin's interface. The admin can generate reports for the entire dairy farm for a year or month or day. On the other hand, the customer can generate reports for deliveries made to that specific customer in a month or year (basically the invoice for that time). Delivery management interacts with the admin portal who updates the deliveries made to a customer daily and the customer can view the deliveries however the customer wouldn't be able to make changes to deliveries.

3. **Persistence Layer**

Persistence layer consists of the components that encapsulate the methods used to access the data from the database. There are basically six data access objects (DAO) which are used to access the data from the database. Customer DAO is used to access all the data related to the customer, admin for admin data, product for product data (milk in our case), animal for animal data, order for order details and inventory DAO fetches data from the inventory in our database. These data access objects contain methods/functions to fetch the relevant data from the database.

4. **Database Layer**

Database is the final layer in our architecture which contains all the data of the dairy farm. As our software is a multi-tenant system hence there was a need to secure the data so that data of one dairy farm remains safe from the other users and data theft. That is why the database layer is the last layer in our system so that all the layers above have checks in place to safely store the data and only allow access to the right user to the data.

c. Justification of the Architecture

The software that we aim to develop is multi-tenant software which means that multiple clients can use our software for their dairy farms. To satisfy our customers' needs, we needed to make sure that the data of our customers is safe and secure. The data should not only be secure from intruders but also from other clients. Our system manages not only the dairy farm but also keeps track of the expenses and the income from the dairy farm, hence there was a dire need to keep the data of the clients safe. To achieve this, we decided to build our system on the layered architecture pattern.

Not only this, but our choice of architecture also helps in the implementation of the non-functional requirements. First, it guarantees security and reliability which is the main concern of the users of any software these days. It also achieves performance trademarks. The software is maintainable as it is divided into layers which are further divided into components. If modifications are made to one layer or component, it does not adversely affect other layers or components. This not only ensures maintainability but also helps when upgrades need to be made to the system.

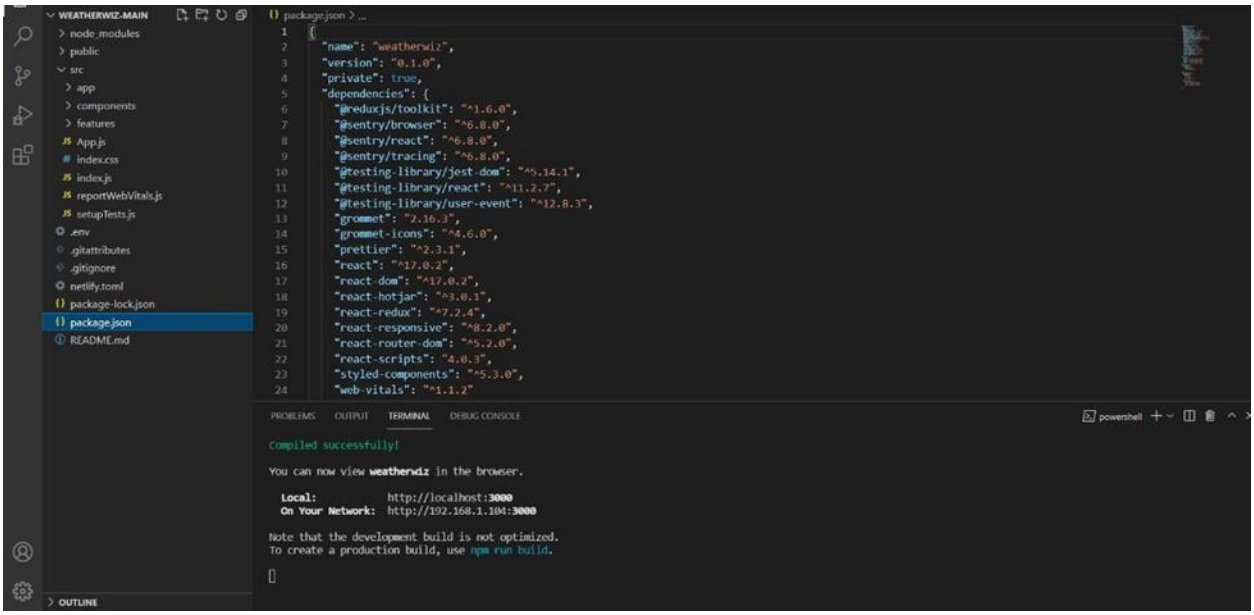
Layered architecture has many advantages such as the guarantee of security, changeability and isolated layers which help in the upgradation of the software, but it also has some drawbacks. Scalability is difficult to achieve with layered architecture. This is because layered apps tend to take on monolithic properties which means that if we need to scale the app, we'll need to scale up the entire application. Also, the cost of the software increases as we add more layers which affects the budget of our customers. Despite all these concerns, the main motivation for us to choose this kind of architecture is the security of the data which has become a great concern for millions of the users of the internet.

d. Tools and Technologies

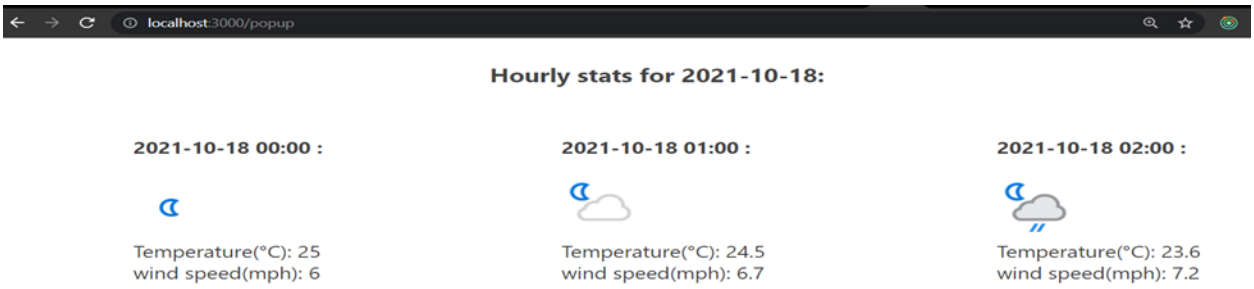
The tools and technologies have been used in the development of our system are as follows:

- ReactJS version react@16.14.0 for front end development of the webapp.
- Front end tools like Redux sagas redux@4.0.4 for state management and API call management.
- React-router react-router@5.2.0 a tool to navigate between components.
- NodeJS version v14.17.6 for backend development.
- MongoDB a NoSQL serverless database MongoDB4.4
- Heroku has been used to deploy and host our website
- Google Drive for document management.
- GitHub for code logistics.

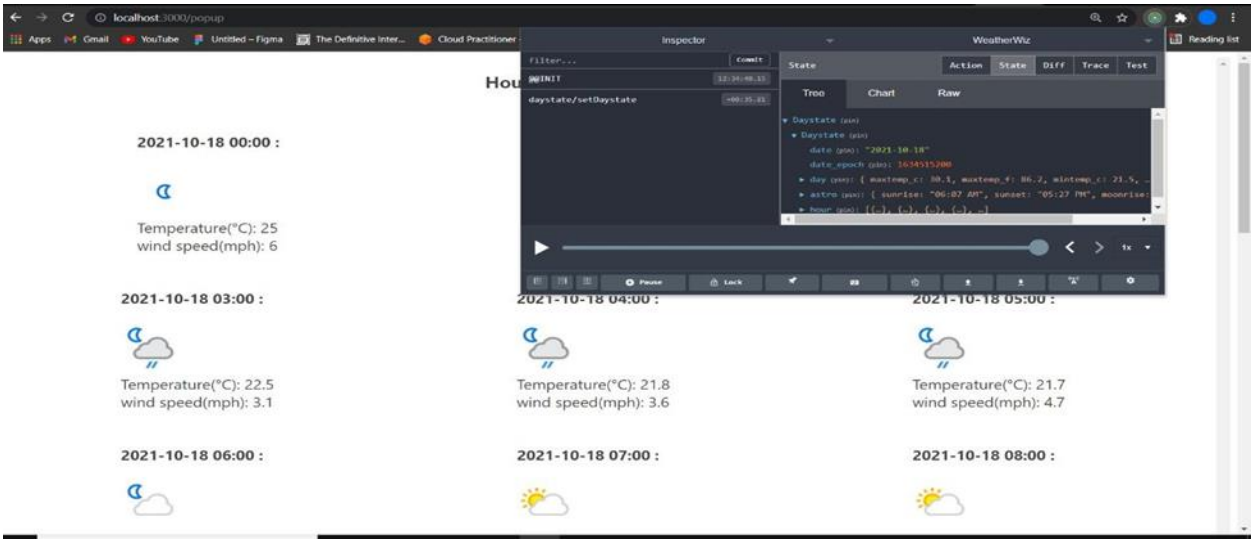
ReactJS app running below are the packages installed which show ReactJS, redux, and router dom from VScode.



The path shows the change in components have taken via “React-router”

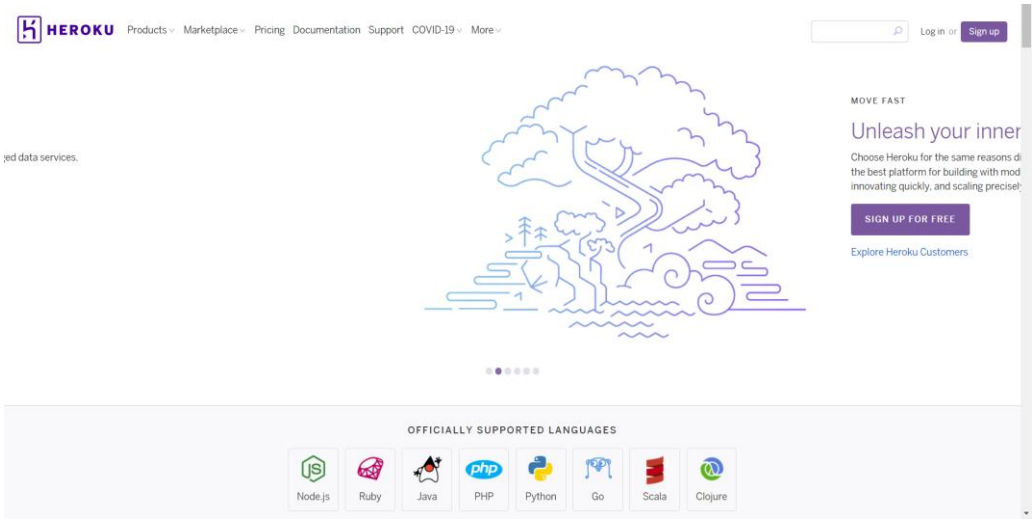


The React redux store responsible for state management is logging states fetched from the API call or updated within the component.



Final Deployment Platform

For now, we have deployed our platform on Heroku, the reason being that it is free. However, for future work on the project the application can be deployed on Amazon AWS due to better and more efficient services.



4. Requirements Specifications

Requirement Specifications that are usually stated in the System Requirements Specification document what the software will do and how it will be expected to perform. To show what functionality our system is capable of performing and the general structure of our system we make use of:

- **Use Cases**

  The use cases tell every single task that can be performed by the system along with how the working of that task will proceed.

- **Class Diagrams**

  In software engineering, a class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.

- **Sequence Diagrams**

  UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

a. Use Cases

Use case diagrams of our system using standard UML notation.

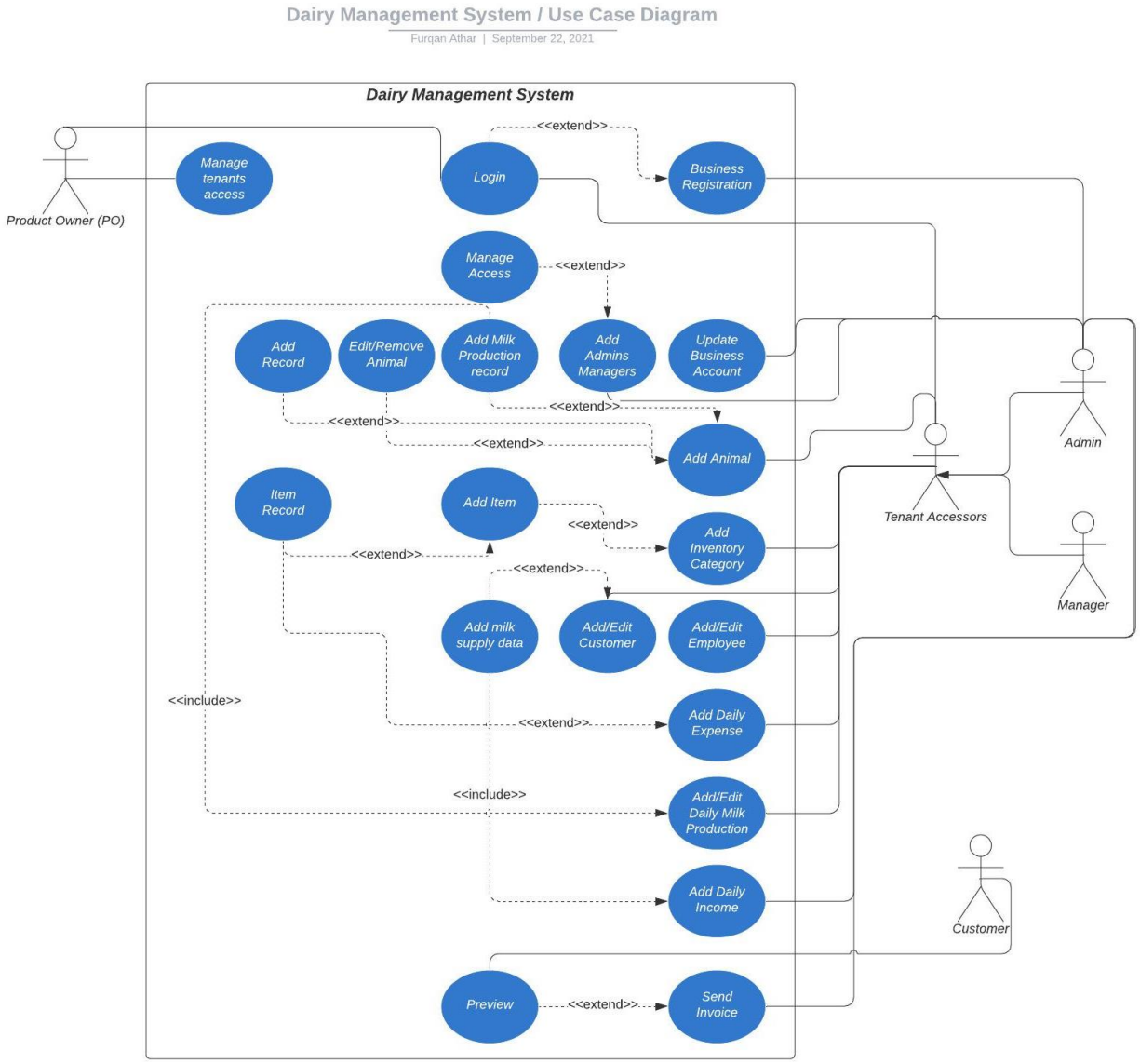


Figure 4.1

1. Registering Dairy Farm

Identifier	UC-003
Purpose	The dairy farm owner wants to register his dairy farm to use our service as an admin.
Pre-conditions	
Post-conditions	Dairy farm will be registered as a tenant to use service

Step #	**Typical Course of Action**
1.	On the registration page, the admin first provides details for his account credentials
2.	After that, the admin will provide details regarding farm name and business domain.
3.	The admin will click “Register”, the account details will be sent to the server for processing.
4.	If registration is successful, the system will redirect the admin to the dashboard.

Step #	**Alternate Courses of Action**
1.	

Step #	**Exception Paths**
1.	If any of the unique dairy information conflicts with the already present dairy, the system will display the error message.

2. Add/Edit managers

Identifier	UC-004
Purpose	The admin wants to add/edit admins and managers and want to manage their access
Pre-conditions	Complete UC-003
Post-conditions	New admins and managers will be added into the management team.

Step #	Typical Course of Action
1.	The admin will go to the admin/manager addition page.
2.	The admin will fill out the information like name, email and other details.
3.	The admin will select the role for that person (Admin/Manager)
4.	The admin will select the person status as well like allow access or restrict.
5.	If successful, a success message will be displayed.

Step #	Alternate Courses of Action
1.	If he wants to restrict someone's access immediately after adding, he can change his status to inactive/restrict.
Step #	**Exception Paths**
1.	If any error occurs, an error message will be displayed.

3. Add Animal

Identifier	UC-006
Purpose	The admin/manager wants to add details of new animals into their system. The page with animal details will be called an animal card
Pre-conditions	Complete UC-003
Post-conditions	Animal detail will be added into the tenant system.

Step #	**Typical Course of Action**
1.	The admin/manager will click the 'Add Animal' button.
2.	The admin/manager will add the animal details tag, name, image, date of birth etc.
3.	The admin/manager will choose animal type like Milking, Child, Heifer etc
4.	The admin/manager will select the animal status like Active, Inactive, Dead etc
5.	The admin/manager will click "Save".
6.	If successful, the system will display a success message.

Step #	**Alternate Courses of Action**
1.	

Step #	**Exception Paths**
1.	In step 5, if any of the unique details conflicts with any of the already present animals, the system will display an error message.

4. Filter Daily Income and see trends

Identifier	UC-018
Purpose	The admin wants to filter the daily income.
Pre-conditions	Complete UC-003 and UC-013
Post-conditions	The filtered daily income list will be displayed.

Step #	**Typical Course of Action**
1.	When the list of all daily incomes will be displayed, the admin can select a specific date or a range of dates from the input form.
2.	The graph, displayed by the system, will show the trend of total daily incomes in the range of dates selected.
3.	The system will display results on runtime after filtration.

Step #	**Alternate Courses of Action**
1.	

Step #	**Exception Paths**
1.	On date selection, if the record is not present, the system will display the message that says, “No Record Found”.

5. Add Inventory Category

Identifier	UC-008
Purpose	The admin/manager wants to maintain/add inventory using categories.
Pre-conditions	Complete UC-003
Post-conditions	Categories will be added into the inventory

Step #	**Typical Course of Action**
1.	On the inventory's category addition page, the admin/manager will first select the option from whether the quantities of items in that category will change on a regular basis or not?
2.	The admin/manager selects the "regular basis" option
3.	The system will ask whether the items to be included are Volume based or Weight based.
4.	The admin/manager will add the category name
5.	The admin/manager will click on "Save"
6.	The system will add a category into the inventory.

Step #	**Alternate Courses of Action**
1.	In step 2, the admin/manager can select the 'Not on Regular basis option and proceed to step 4 directly.
2.	After completing step **6,** the admin/manager can click on 'Delete'
3.	The system will delete the category from the inventory.
Step #	**Exception Paths**
1.	In step 5 , if any of the information of a new category conflicts with any of the already present categories, the system will display an error message.

6. Add items via Inventory Category

Identifier	UC-009
Purpose	The admin/manager wants to add items into the category with their stocks
Pre-conditions	Complete UC-003 and UC-008
Post-conditions	Items will be added into the category

Step #	Typical Course of Action
1.	The admin/manager will click on the category name and then will click on the 'Add Item' button.
2.	If the 'Weight' category was clicked, the admin/manager will enter the name and weight (in kg).
3.	The admin/manager will click on "Save".
4.	The system will display a confirmation popup which will say "Are you sure? You'll not be able to change this item's name again."
5.	If the admin/manager selects "Yes", the request will be sent to the server and on successful completion, a success message will be displayed.

Step #	Alternate Courses of Action
1.	In step 2, if the 'Volume' category is selected, the admin/manager will add volume (in ml) instead of weight for the item.
2.	In step 2, if the 'Miscellaneous' category is selected, the admin/manager will add the discrete value of quantity for the item.
3.	In step 5, if the 'No' option is selected, the admin/manager will be redirected to the step 2.
Step #	**Exception Paths**
1.	In step 3, if the admin/manager tries to add an item that is already present in that category, the system will display an error message and he'll move back to step 2.

7. Edit Existing Inventory

Identifier	UC-010
Purpose	The admin/manager wants to edit/delete the item in inventory having some exceptions.
Pre-conditions	Complete UC-003, UC-008 and UC-009
Post-conditions	Items will be edited/removed from the inventory.

Step #	**Typical Course of Action**
1.	On the inventory's category page, the admin/manager will click "Edit" corresponding to an item.
2.	Admin/Manager will update the stock manually.
3.	The admin/manager will click on "save"
4.	The system will update and add the record of inventory updates under that item's details.
5.	If successful, a system will display a success message.

Step #	**Alternate Courses of Action**
1.	During step 1, 2 or 3, the admin/manager will 'Delete' corresponding to the item.
2.	The system will delete the item, if that item has no record.
Step #	**Exception Paths**
1.	In alternate step 2, the system will not allow the delete event to happen if that item has some record and the system will display an error message.

8. Edit/Add/Remove Employee(s)(data)

Identifier	UC-011
Purpose	The admin wants to add/edit/remove employee's data from the management system.
Pre-conditions	Complete UC-003
Post-conditions	The employee will be added to the system's database.
Step #	**Typical Course of Action**
1.	The admin/manager will click the 'Add Employee' button.
2.	The admin/manager will add the employee's salary.
3.	The admin/manager will add the employee's data like image, date of birth and other things.
4.	The admin/manager will add the CNIC of that employee.
5.	The admin/manager will click "Save".
Step #	**Alternate Courses of Action**
1.	In step 1, the admin/manager will click the 'Edit Employee' button and move to step 2 for editing.
2.	In step 1, the admin/manager will click 'Delete'.
3.	The system will display a confirmation message.
4.	If the admin/manager clicks 'Yes' and the system will verify.
5.	The system will display a success message if there are no records for that employee.
Step #	**Exception Paths**
1.	In alternate step 4, if the employee has some record present, the system will display an error message because an employee having some history must not be deleted.
2.	In step 5, the system will display an error message if the CNIC or any record conflicts with any already present employee.

9. Add daily milk production

Identifier	UC-015
Purpose	The admin/manager wants to add/edit daily milk production of each cow.
Pre-conditions	Complete UC-003 and UC-006
Post-conditions	Daily milk production of each cow will be recorded into the database.

Step #	**Typical Course of Action**
1.	The admin/manager will click ‘Add Daily Milk Production’ button
2.	The admin/manager will add milk records for both morning and evening for each active animal.
3.	The admin/manager will click “Save”

Step #	**Alternate Courses of Action**
1.	In step 1, the admin/manager will click on the ‘Edit Milk Record’ button and will follow step 2.
2.	In step 2, the admin/manager will add the total milk production of all animals in a single will.
3.	The system will automatically divide that total into each active animal.
Step #	**Exception Paths**
1.	In step 3, the system will display an error message if the record on that date is already in the system.

10. Filter Milk Productions

Identifier	UC-016
Purpose	The admin/manager wants to filter the daily milk production.
Pre-conditions	Complete UC-003 and UC-015
Post-conditions	The filtered daily milk production list will be displayed.
Step #	**Typical Course of Action**
1.	When the list of all daily milk production will be displayed, the admin/manager can select a specific date or a range of dates from the input form.
2.	The system will display the graph that will show the trend of total milk production in the range of dates selected.
3.	The system will display the results on runtime after filtration.
Step #	**Alternate Courses of Action**
1.	
Step #	**Exception Paths**
1.	On date selection, if the record is not present, a message will be displayed that says, “No Record Found”.

b. Class Diagram

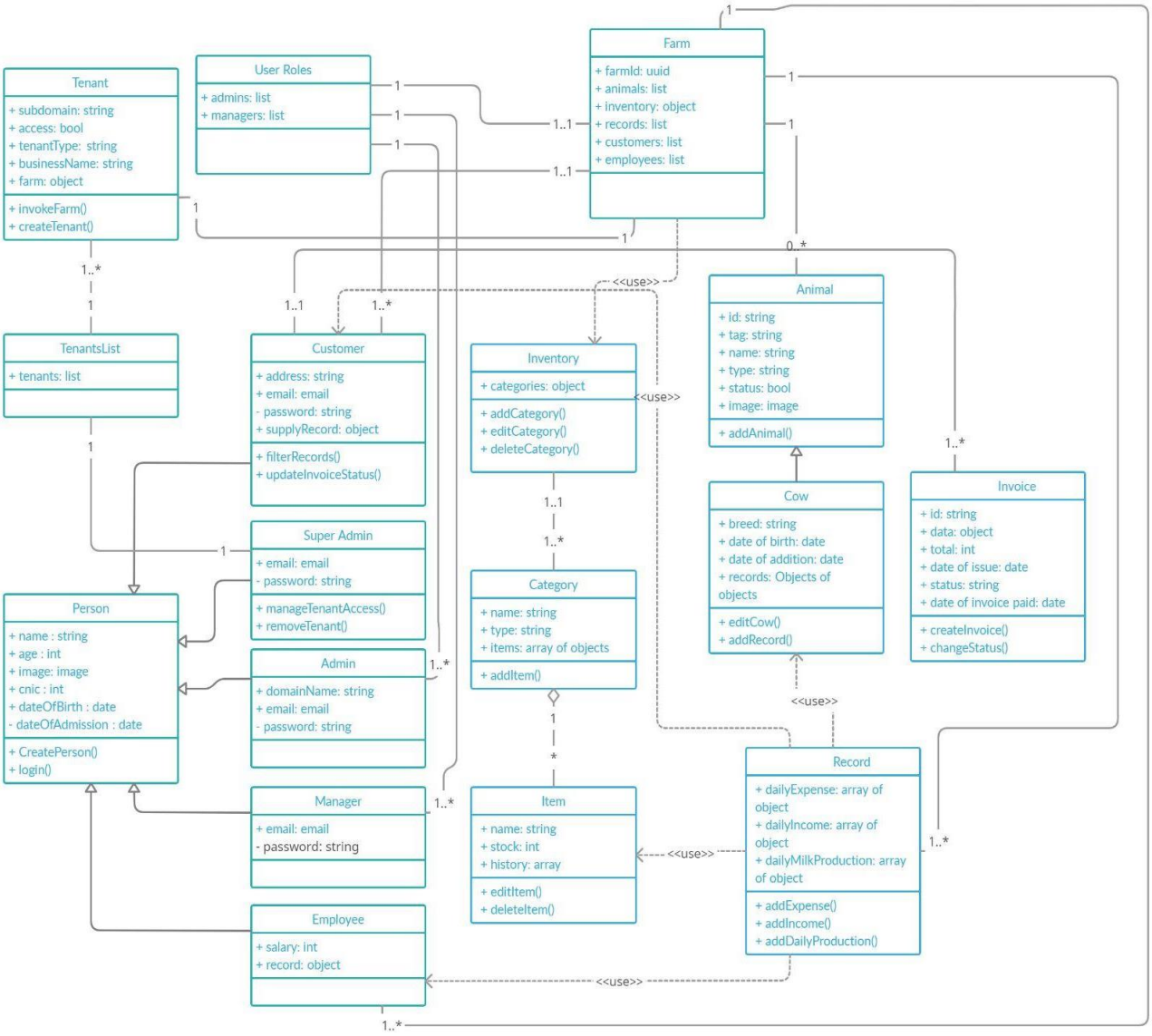


Figure 4.2

Description of Class Diagram

1. Person: It'll be the person class which contains basic attributes for a person to exist. Other persons will inherit this class for their common attributes.

2. Super Admin: Super Admin is the owner of the product i.e., Dairy Management System. He'll be able to manage access to multiple tenants from his dashboard and, he can restrict them from using their business accounts.

3. Tenant: his class will have the information of a single tenant account and the associated single farm. Each tenant will have a single farm associated with it.

4. TenantList: This will contain a list of all tenants in our database and the super admin will have access to all of them and can restrict the access of any of them.

5. Farm: The farm will be associated with a tenant, and it'll consist of all other classes that are associated with it like animals, inventory, and other things.

6. UserRoles: This class will have a list of all admins and managers that are in the farm and can have multiple admins and managers in it.

7. Admin: The tenant admin – will contain basic attributes inherited from the Person Class.

8. Manager: Manager class will inherit the person class for its existence.

9. Employee: Employee class will inherit the person class and it also contains the salary attribute as well that is necessary for him to exist. It also has records associated with him that store how much money he has been paid and on which date.

10. Inventory: Inventory class will have multiple categories and it provides the function to add those new categories.

11. Category: Category class will have the type which specifies that if it is a Volume based category or Weight based or the category whose items' quantities cannot change with consumption.

12. Item: This will store the details of items and provide methods like edit and delete and delete will only work if this item's record is empty.

13. Animal: This class is the base class for animals and has basic attributes that are common among different animals.

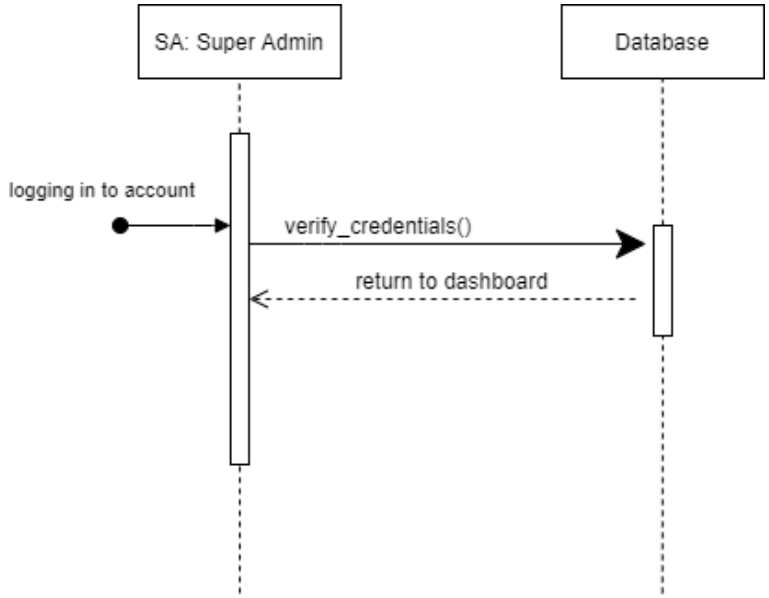
14. Cow: This class will have attributes that are for cows and inherit the Animal Class as well. This class will also have records objects as an attribute which stores different records like milk production record, medication record, vaccination record etc.

15. Record: This is the most important class of all, and it depends on various other classes for its existence. If we want to add daily expense, we must refer to this class and that depends on the items class if we want to add the expense of the item present in the inventory i.e., Medicine. Also, if we want to add daily income, it depends on the Customer Class because if we want to add the milk supplied to a customer then it depends on the existence of this Customer Class instance. Likewise, for the milk production record of animals, we want the instance of Cow Class and hence dependency occurs.

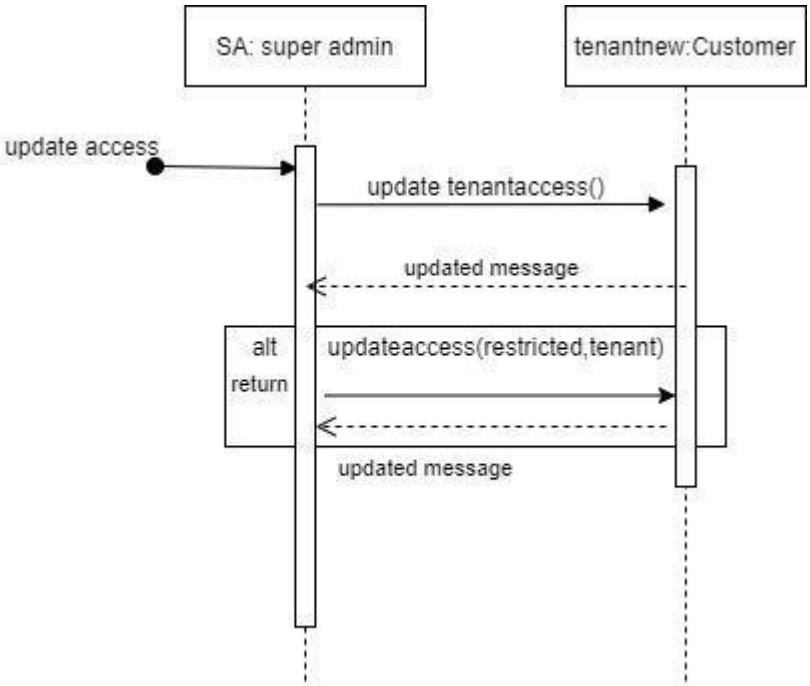
16. Invoice: Invoice class is for charging a customer on the unpaid supplies of milk and a single customer can have multiple invoices.

c. Sequence Diagrams

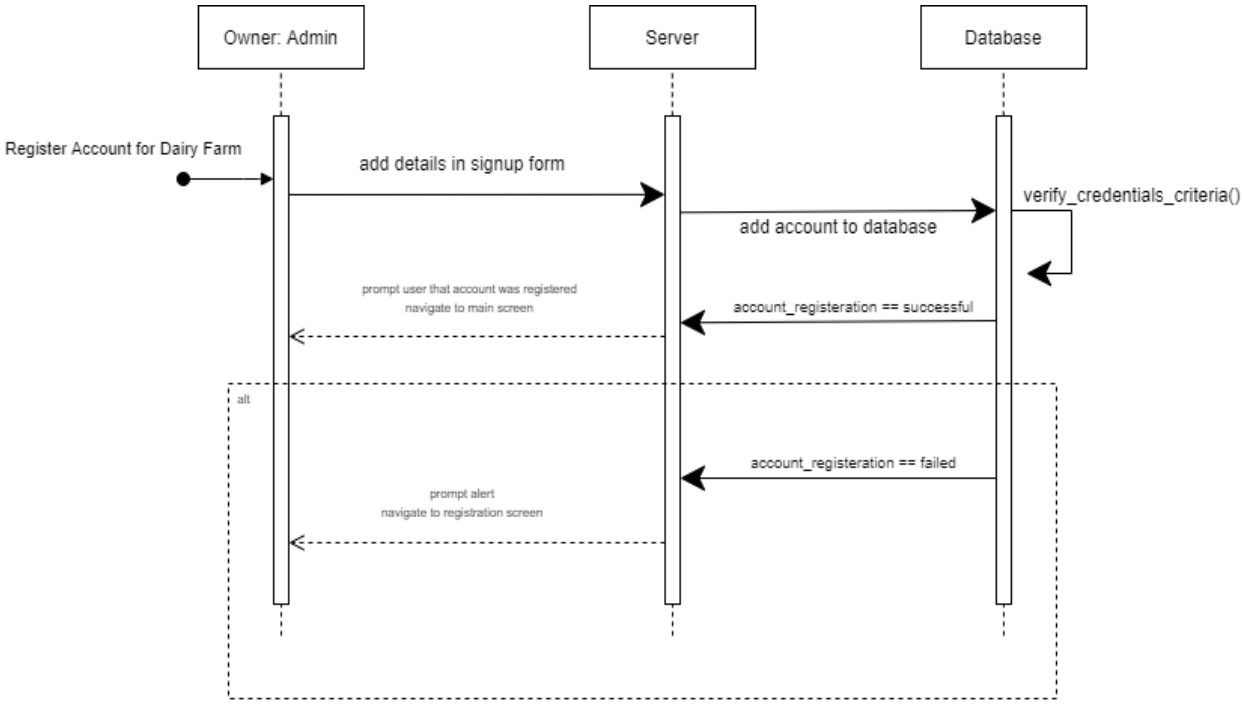
1. Login for Super Admin (Product Owner)



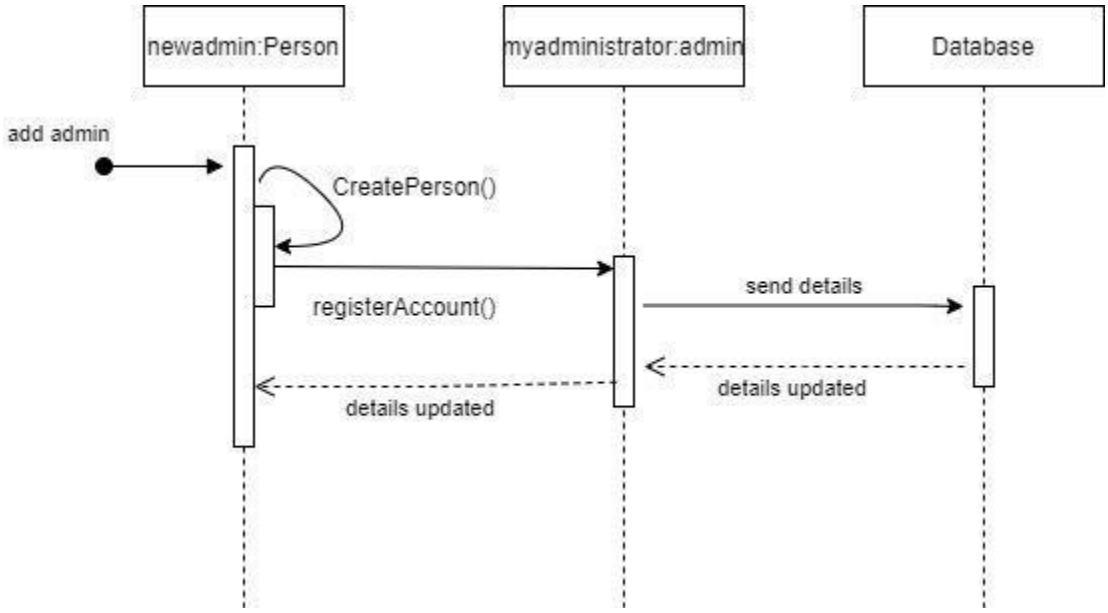
2. Change Access for Tenants



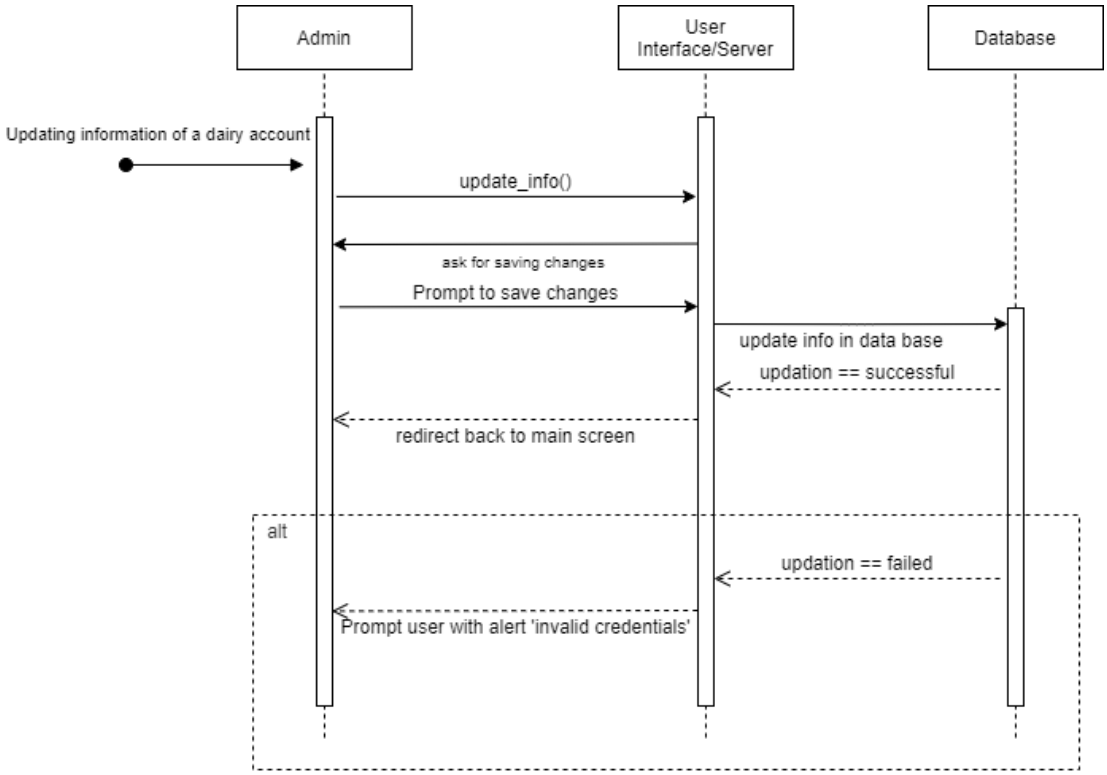
3. Register Account for Dairy Farm



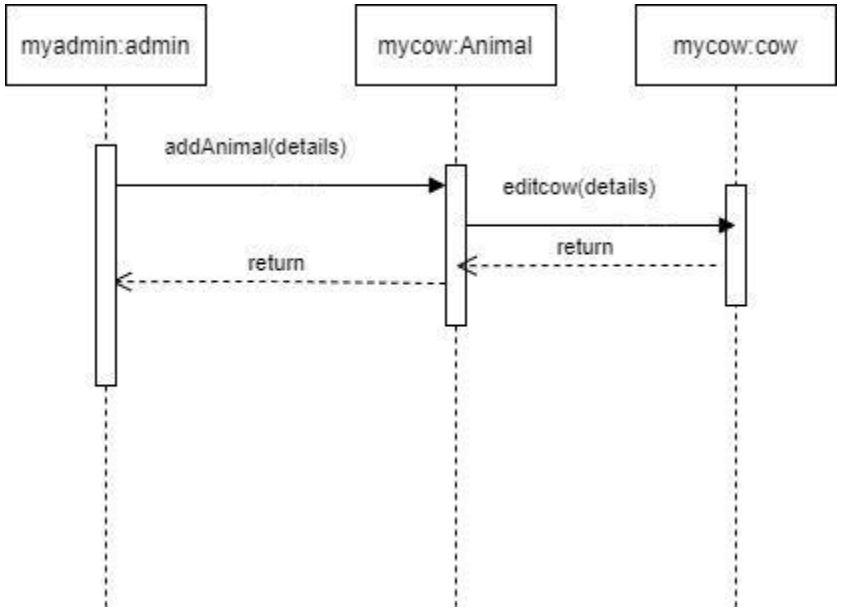
4. Addition of team member/manager to team



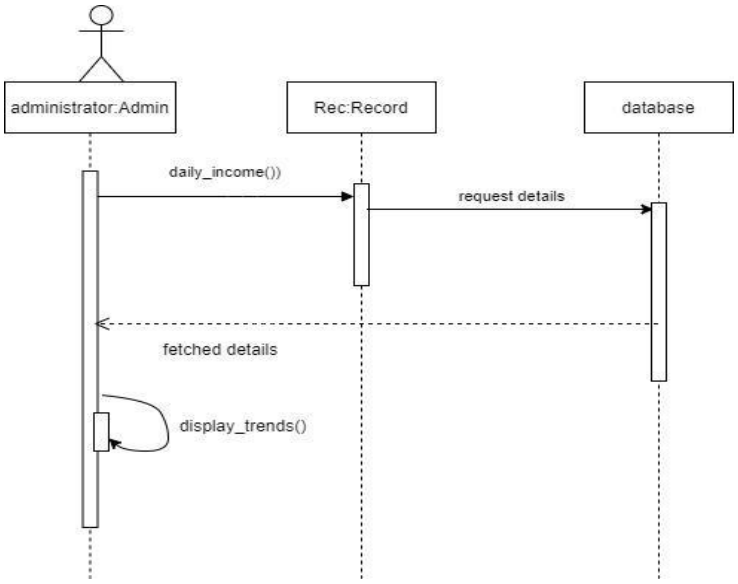
5. Update Account Information of a dairy farm owner



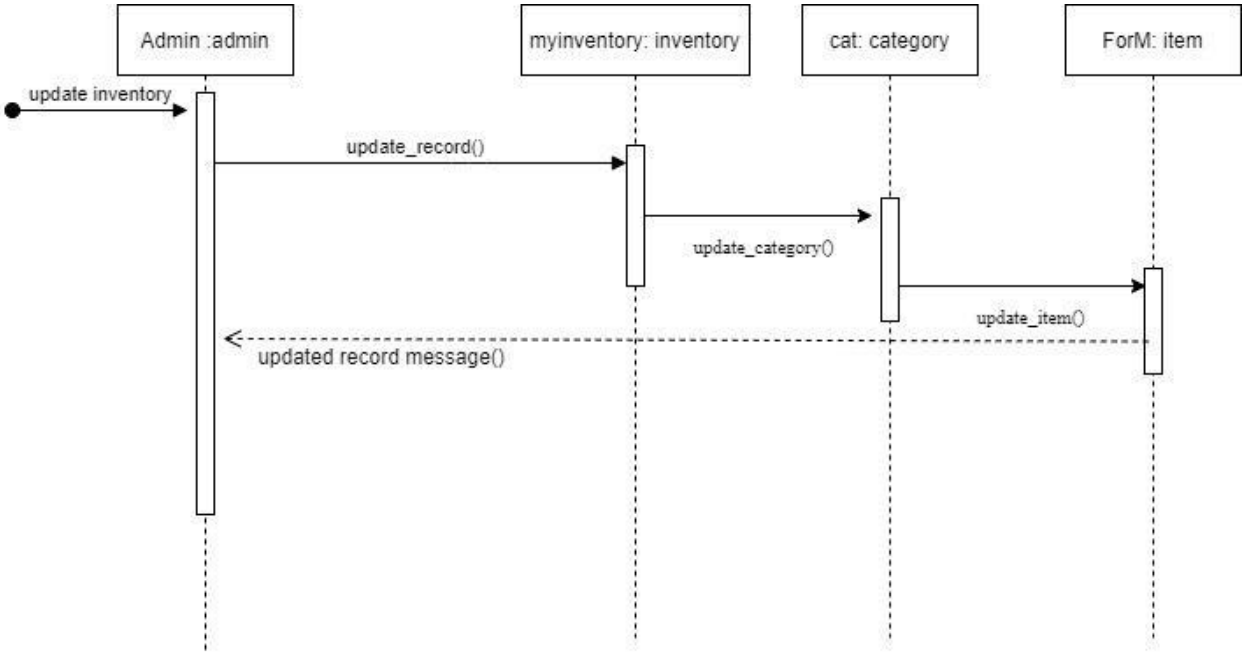
6. Add new Animal



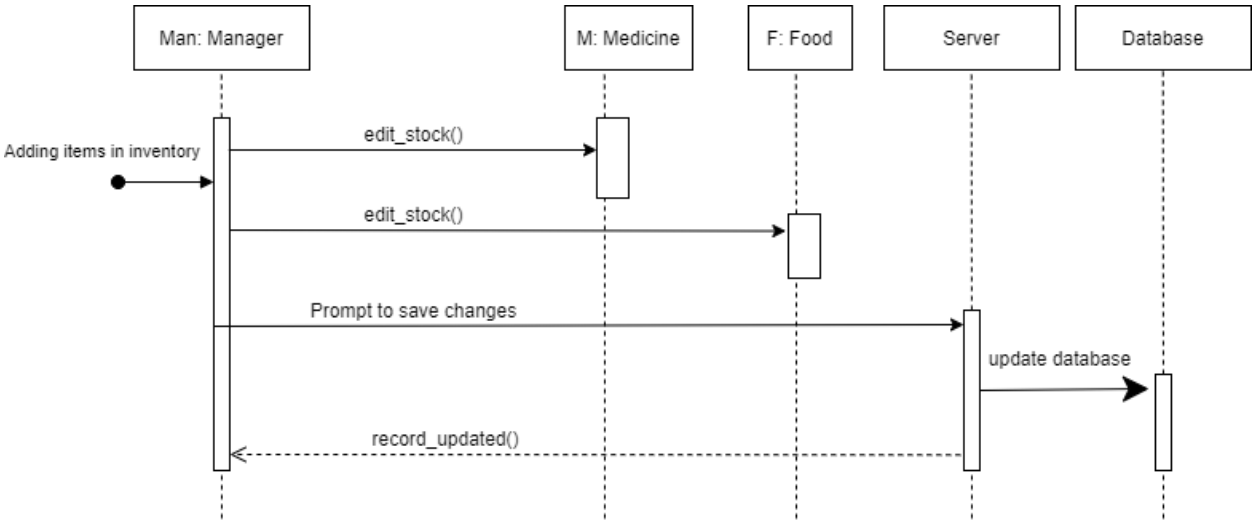
7. Revenue and Revenue Trends



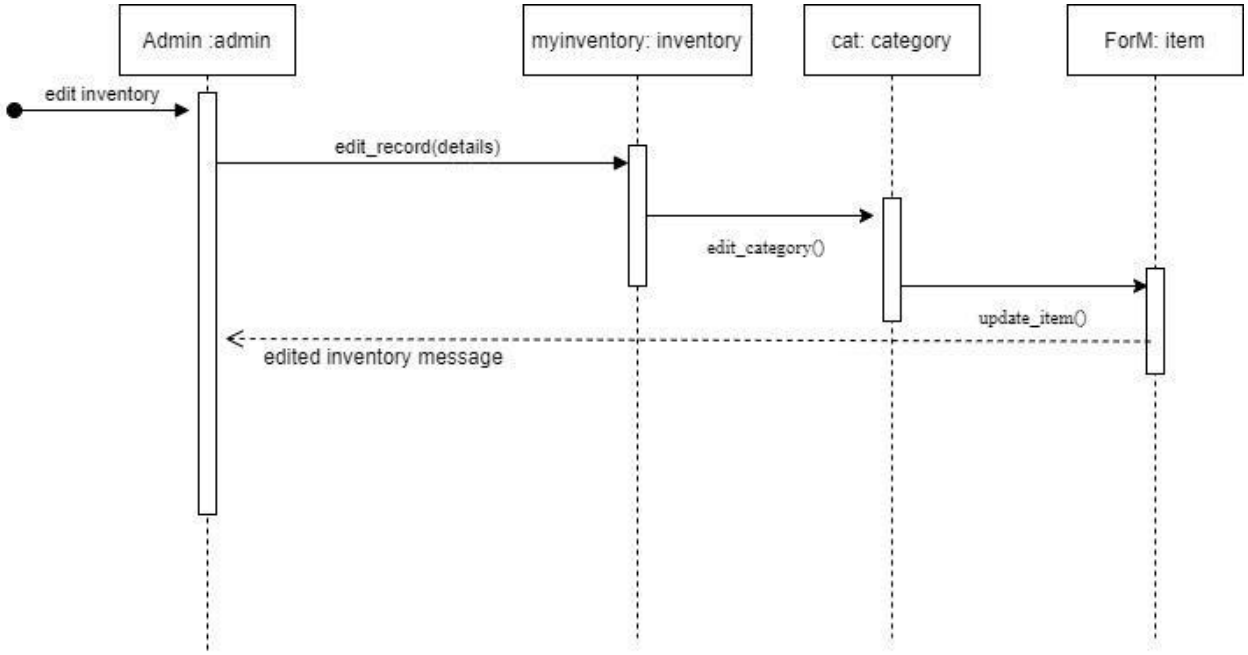
8. Add new categories to Inventory



9. Add Items to inventory



10. Edit/Delete – Use Inventory items



5. Software Development Methodology and Plan

Software Development Methodology and Planning are crucial steps before the team starks formally working on the project. The team must be clear about what type of development plan they want to make use of, keeping in mind the time and financial constraints and the room of error allowed during the development phase. There are various set plans a team can choose from 2 of which have been stated below along with their pros and cons.

a. Software Process Selection

Waterfall:

- **Pros**
    - Simple, well defined, and straight forward phases which are sequential; this helps in better and more efficient management of the project
    - Since each phase is well documented and the flow is sequential, each phase must be completed properly before moving on to the next phase. This allows for more efficient interphase flow of data.
    - There is more clarity about the project phases for the development team and easier to keep track for them.
- **Cons:**
    - This method can be troublesome for a project where the goal of the project is not clearly defined, and requirements are changing frequently.
    - Many businesses do not have the same requirement so updating makes it hard for the teams to manage.
    - The working system gets ready at a very late stage in the development life cycle

Agile:

- **Pros:**
    - The Agile methodology offers more flexibility than the waterfall. Allows more room for improvisation even in the later phase, as you can move back and forth between phases, during each iteration, there is an opportunity to constantly refine and reprioritize the overall product backlog.
    - The Agile methodology is much faster. Using the Agile methodology, the product can reach the market faster. This can be used to gain feedback from the users based on the minimum viable product, that feedback can be incorporated into the MVP and a more user centered product can be produced.
- **Cons:**
    - Moving back and forth between phases excessively may cause the market launch of complex products to suffer and take up more time. A very good level of collaboration and communication is required for inter phase communication between the development team to keep up. At times it gets difficult to maintain these standards.
    - For most clients there is a need for a known deadline, but with agile methodology it is hard to predict when a deliverable will be due.

Software Process for our Project:

Chosen process: Agile(scrum) methodology

For our project we chose to go with the agile scrum methodology, taking in considerations from the project contextual analysis. Agile methodology enables work to be done in continuous sprints, so the product is getting better iteratively, and bugs are taken care of in these sprints. It adjusts well to requirements change given our product is in its early stages of build so a prototype can be shown to the client that can help them better assess requirements.

A team size of 4 people would be more suitable to the agile methodology, since it enables more collaboration effectively and delivers a product in a short span of time. The agile methodology follows a flow in which there is a prototype product produced so there is no pressure to rush through phases such as in the waterfall model. An iterative process would mean that the product has many components that have been made in the earlier stages that can be used for later stages as well, there will be high availability of components.

b. Gantt Chart

ID	Task Name	Start	Finish	Duration (Weeks)
1	Prototype Development	19-Oct-21	8-Nov-21	2.1 w.
2	Architecture and Design	9-Nov-21	17-Nov-21	1.0 w.
3	Testing Strategy	18-Nov-21	26-Nov-21	0.9 w.
4	Exam Break	26-Nov-21	28-Dec-21	3.3 w.
5	Sprint-1 (Use Cases 1-7)	31-Dec-21	28-Jan-22	3.0 w.
6	Sprint-2 (Use Cases 2-14)	1-Feb-22	1-Mar-22	3.0 w.
7	Sprint-3 (Use Cases 15-21)	2-Mar-22	30-Mar-22	3.0 w.
8	Sprint-4 (Use Cases 22 - 27)	31-Mar-22	28-Apr-22	3.0 w.
9	Integrated System Testing	29-Apr-22	9-May-22	1.0 w.

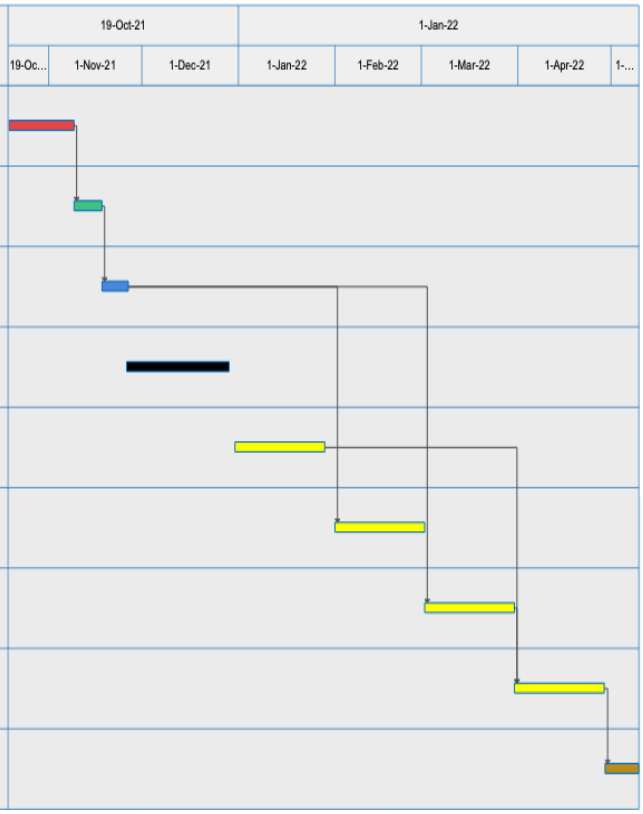


Fig 5.1

Team member names who have worked on each task: For our project all the team members were working together in all the phases of the project be it testing/development. The work was divided based on use cases. Although initially the use cases were divided amongst the team members, however since the work was integrated, we allowed room for greater flexibility and team members could coordinate with each other where they required assistance from other members to complete a certain task or even if they wish to hand over the complete task to other members.

6. Database Design and Web Services

Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data to the database model.

a. Database Design

Diagram

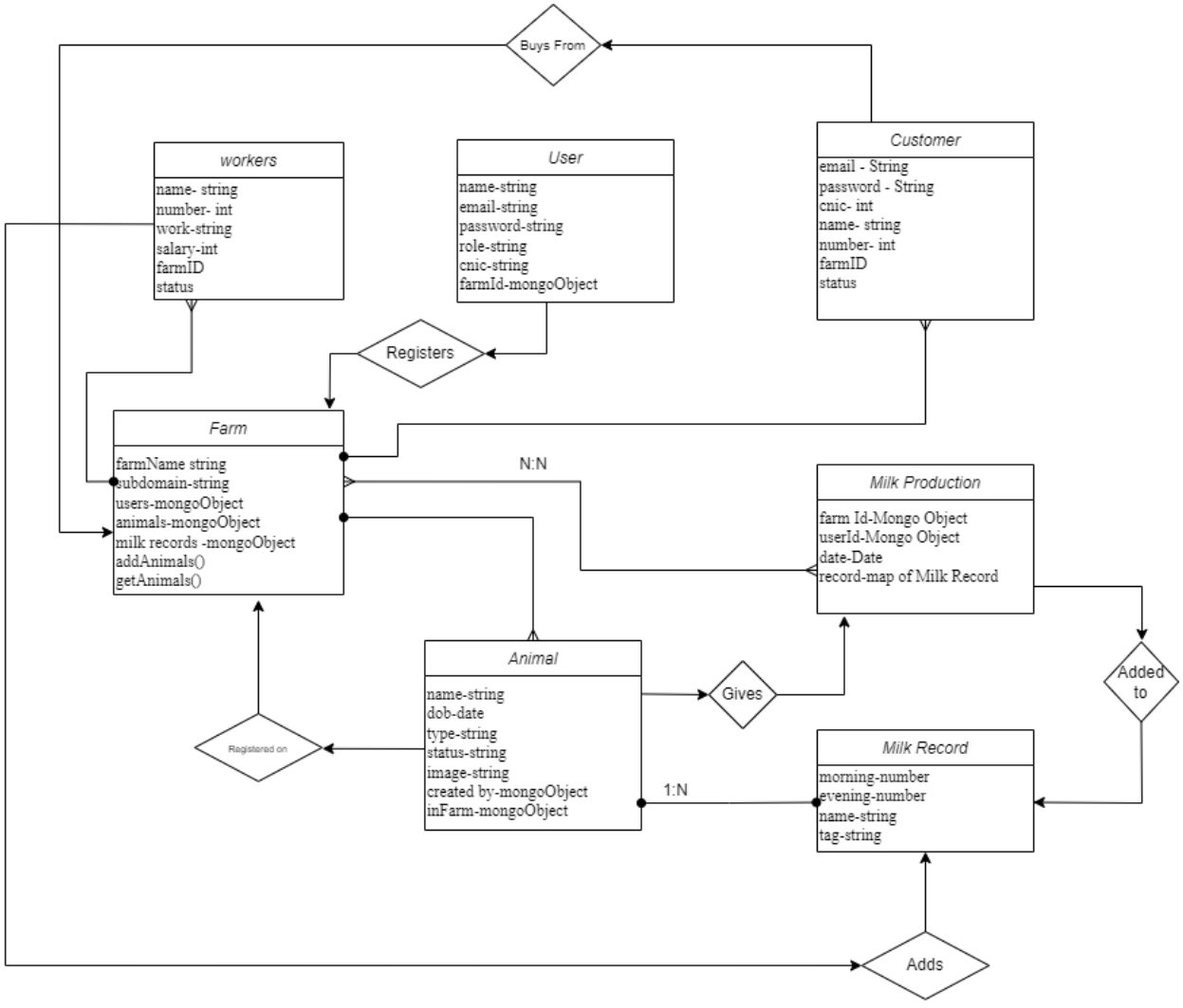


Figure 6.1

Description

Animal:

This data model stores information for farm animals in a particular farm with a unique tag for each animal.

- **name-string**: This adds a name for the animal for ease of identification.
- **tag-string**: a unique id associated with each animal
- **dob-date**
- **type-string**: If the animal is a cow, heifer or etc.
- **status-string**: if the cow is milking, or not milking or active.
- **image-string**: This adds the picture of the animal to the database
- **created by-mongoObject:** stamped by the database
- **inFarm-mongoObject**: stamped by the database

An animal may have multiple milk production records for every different day. But for a single date, there will only be one record, having the evening and morning production.

Farm:

Each tenant can register their farm with a unique business domain name.

- **farmName - String**: Name of the farm
- **subdomain - String**: CNIC of the tenant that is creating the farm
- **users - Mongo Object:** Users on the farm
- **animals - Mongo Object:** Animals on the farm
- **milk records - Mongo Object:** Milk Production record of the animals
-

A farm can have only 1 Owner (1-1).

An Owner may have multiple farms present in the database. (1-N).

Similarly, a farm may have multiple animals (1-N).

A farm may have a cumulative milk production record linked to their respective farm id.

User:

- **name - String:** Name of the user
- **email - String:** Email address of the user used to register the account
- **password - String:** Password for account access
- **role - String:** Role of the user either manager/ tenant/farm owner
- **cnic - String:** The CNIC Number of the user
- **farmId – Mongo Object:** A schema for the database

The User Class is a parent class, it can have multiple subclasses, making it a (1 - N) mapping. Subclasses of Admin, Manager, Employee and Customer inherit from the User Class. Our Model assumes that there is only one super admin, in that case the relation would become (1-1).

Milk Records

- **morning - number:** The amount of milk produced in liters in the morning.
- **evening - number:** The amount of milk produced in liters in the evening.
- **name - string:** The name assigned to an animal, may or may not be unique.
- **tag - string:** The tag is a unique id assigned to an animal, must be unique.

Milk Record are for a single entry per day. This Object is used to cumulate the Milk Production object, provide detailed information regarding milk production over a longer period.

Milk Production:

- **farmId - Mongo Object:** To keep track of milk production of a certain farm
- **userId - Mongo Object:** An object to allow for a multi-tenant system, (yet to be deployed)
- **date - Date Object:** Date to keep track of milk production on a certain day.
- **record - Object of type Milk Record:** Schema mapping of object Milk Record

b. API Specification

```
import nodemailer from "nodemailer";
import sendgridTransport from "nodemailer-sendgrid-transport";
const transporter = nodemailer.createTransport(
  sendgridTransport({
    auth: {
      api_key:
        "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",
    },
  })
);
```

We have used Sendgrid by Twilio backend emailer API to send login credentials to the registered used to add an extra layer of security. The manager that will register an employee, the login credentials for their account will be sent to their respective email, using those credentials they will be able to login and use their account.

For Future use, this API can also be used for marketing and advertising purposes to send emails to potential customers.

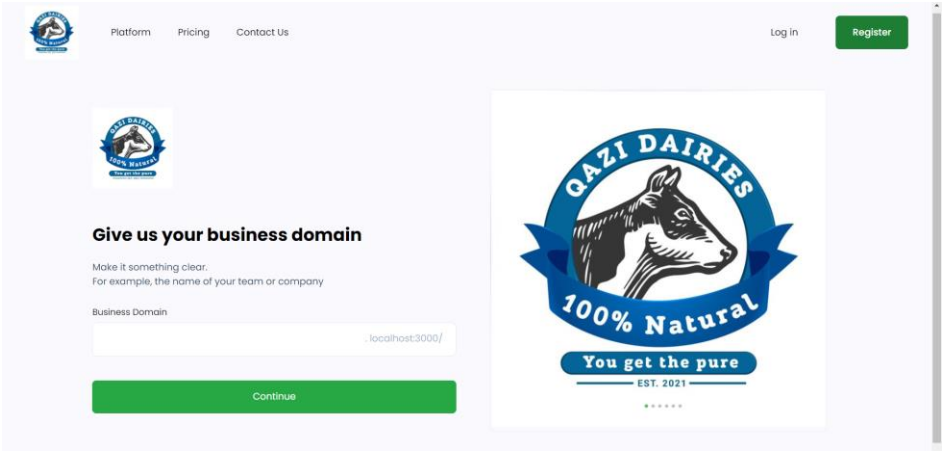
Note: We have only made use of one external API for the project.

7. System User Interface

Dairy Farm Management system was a project that was sponsored by a Dairy Farm Start-up called Qazi Dairies. The project was therefore initially set to cater to use cases that the client had requested, however after pitching our ideas to the client and pointing out some functionality that could lead to an increase in sales the client agreed for a fully functional platform with added functionality. The following figures show the high-level functionality of our system to the end user and how the system is to be operated – Use cases of Medium/Low priority such as search, filtering etc. have not been included.

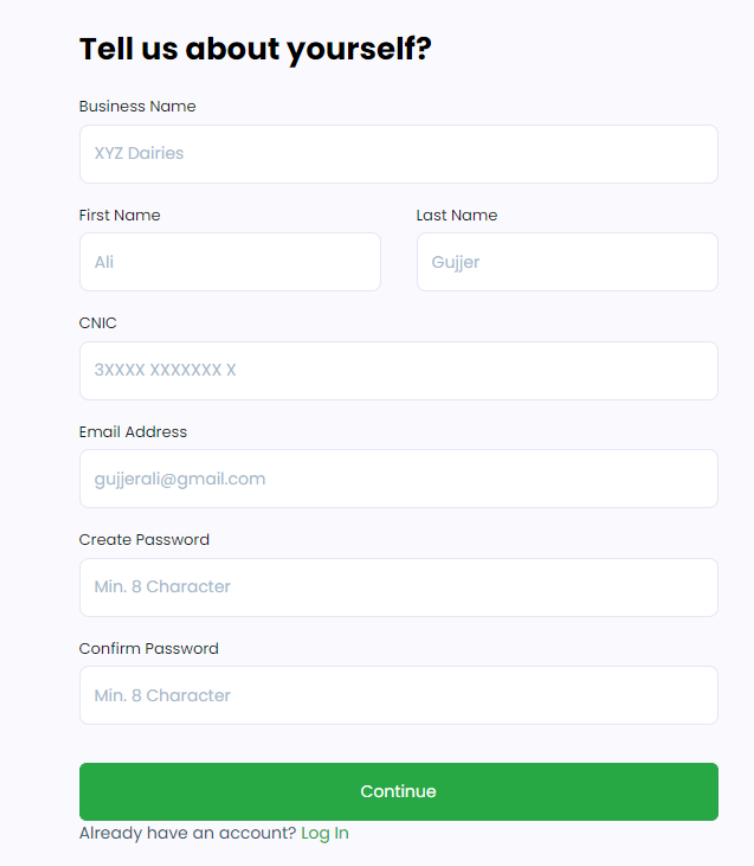
a. Domain Registration

Farm owners need to be assigned a domain before they can register their farm.



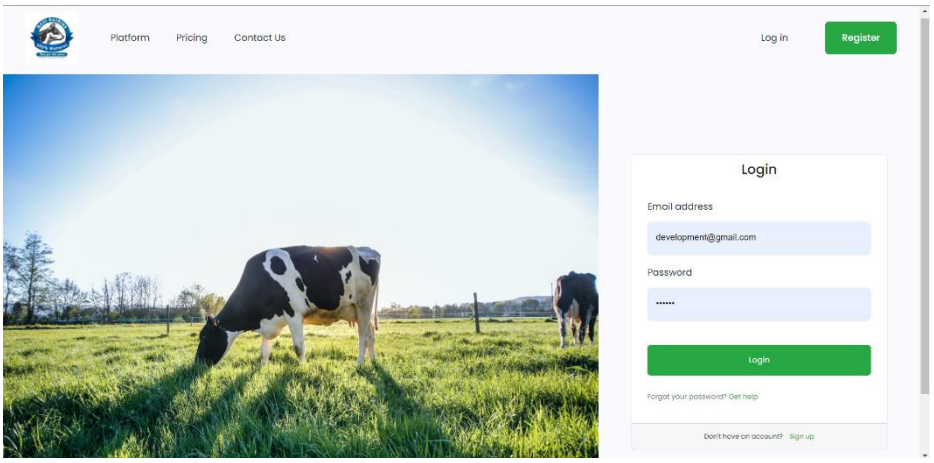
b. Farm Registration and other details

Farm owners after registering their domain need to add other details of their farm to complete the registration of their account.



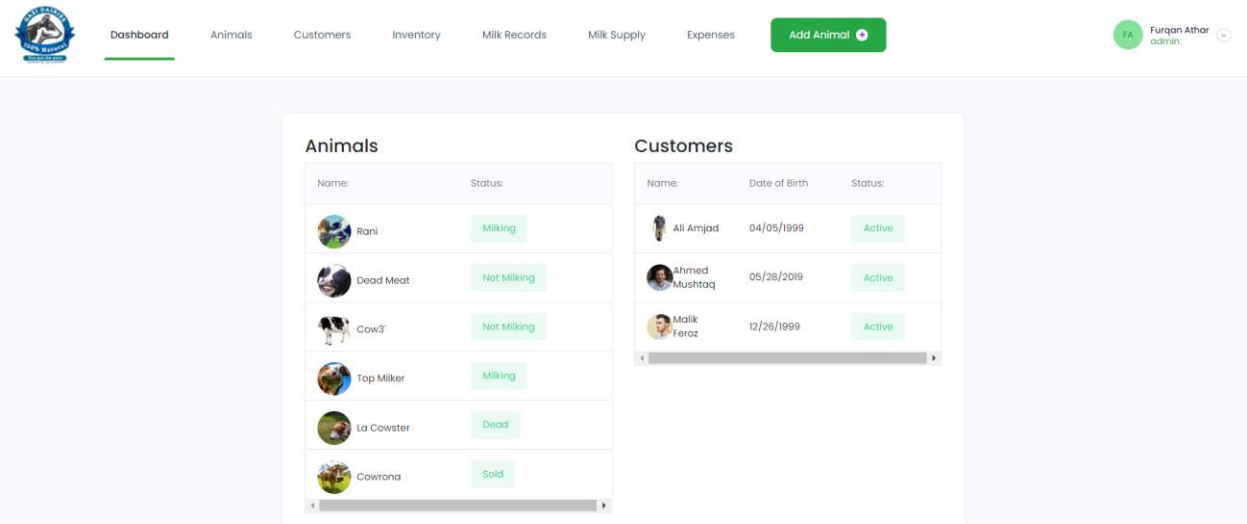
c. Login Screen

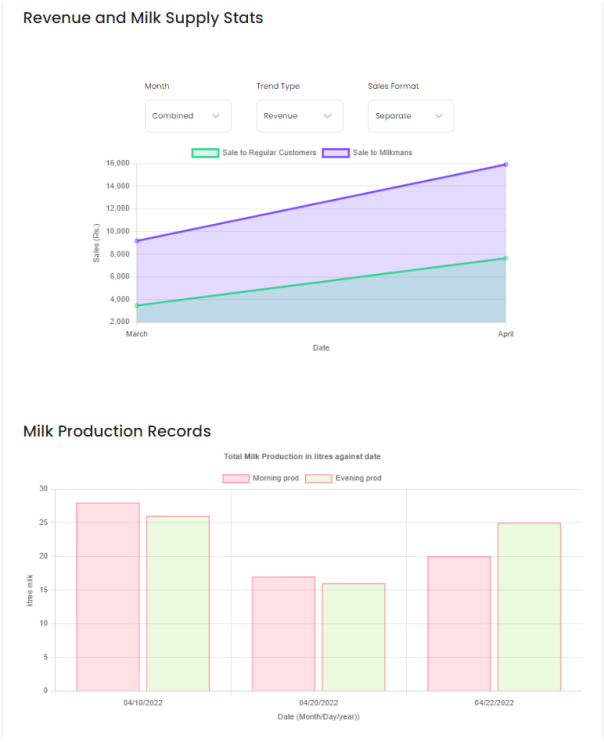
After completing the registration process, the farm owner can now access their farm using the login password they have created.



d. Dashboard

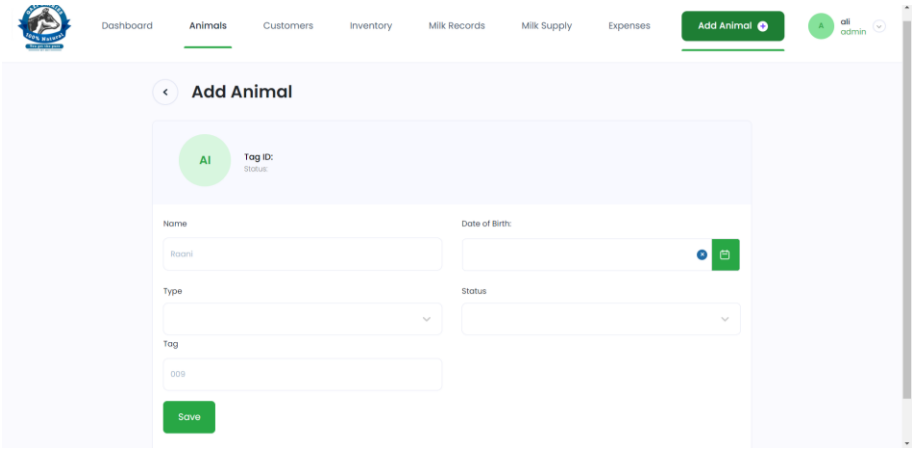
On login the dairy farm admin is navigated to the system dashboard where they can see the over all summarized form of their data and other visualizations.



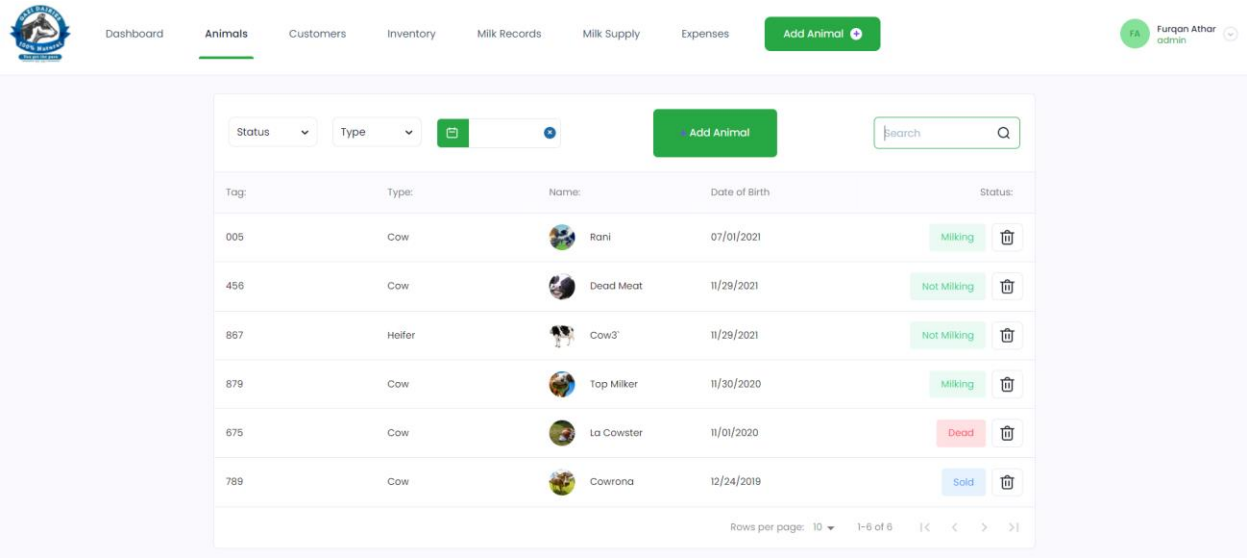


e. Adding Animals

Farm workers that have been registered by the admin can add animals/livestock on the farm to the farm database. After filling the below farm the animal is added to the farm record of animals.

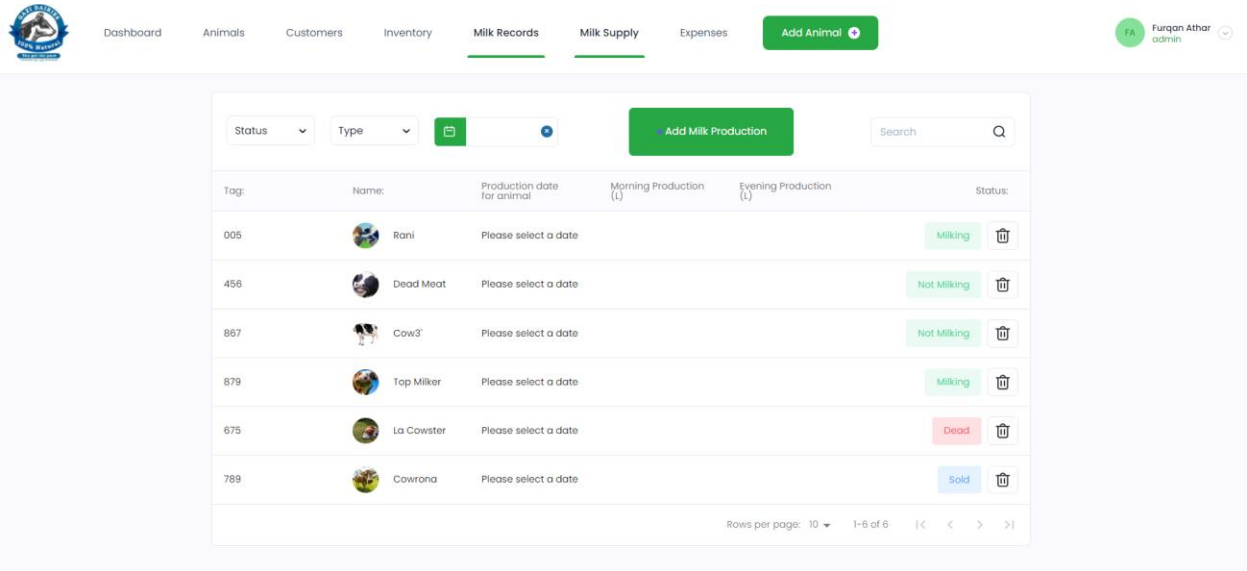


After being added to the database the animals will appear in the animal page home screen.

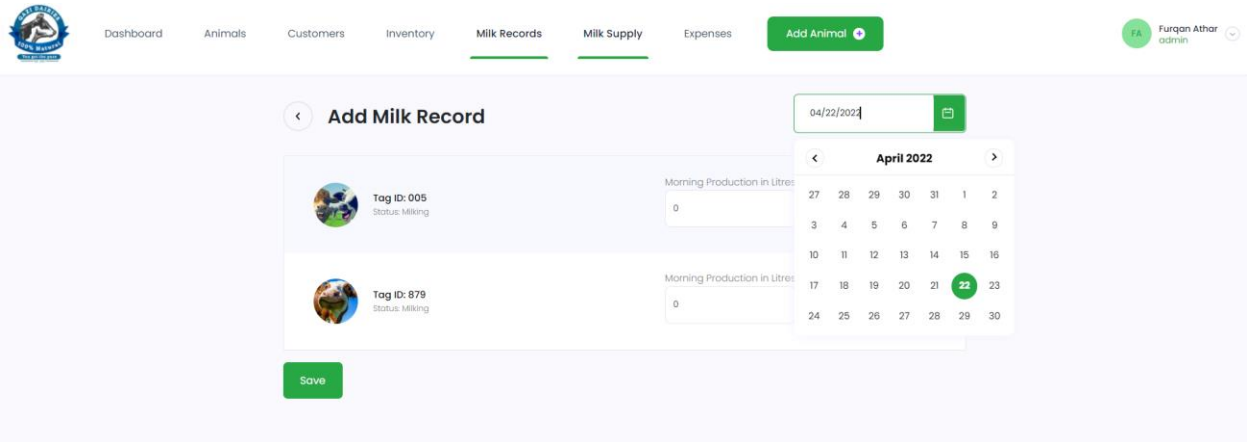


f. Adding Milk Production

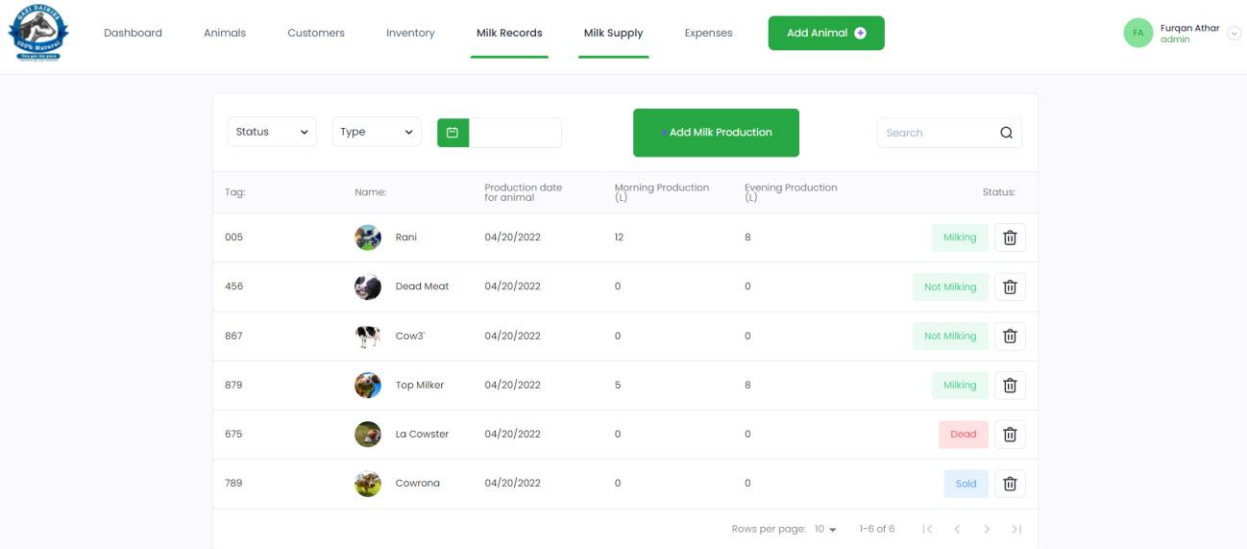
Access the milk record page from the nav bar and navigate to “Add milk record” to add milk record for various animals that are of the milking status.



After the animal has been added their morning and evening milk production can be added for a specific date.

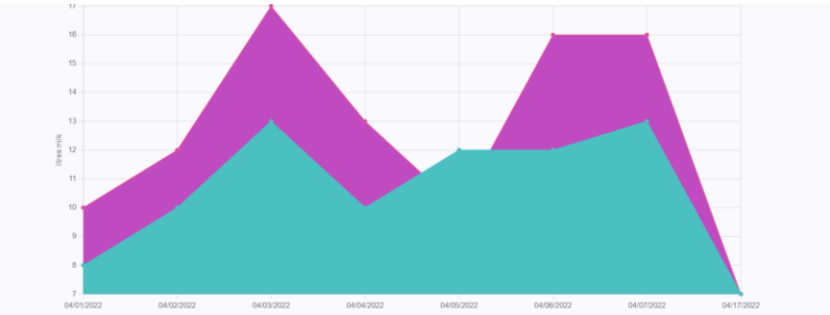


Milk Production can then be check at the dashboard or at the Milk Records Page by navigating back.



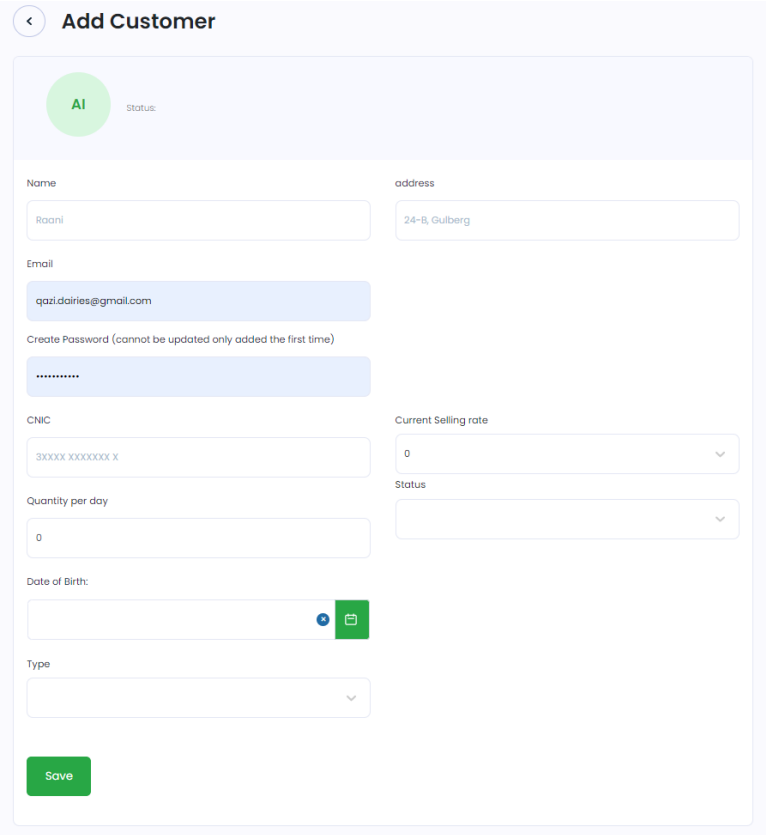
After having added the animals and their relevant production, we can visualize the production on the dashboard from the bar graph to view various trends.

We can also compare milk production of various animals.

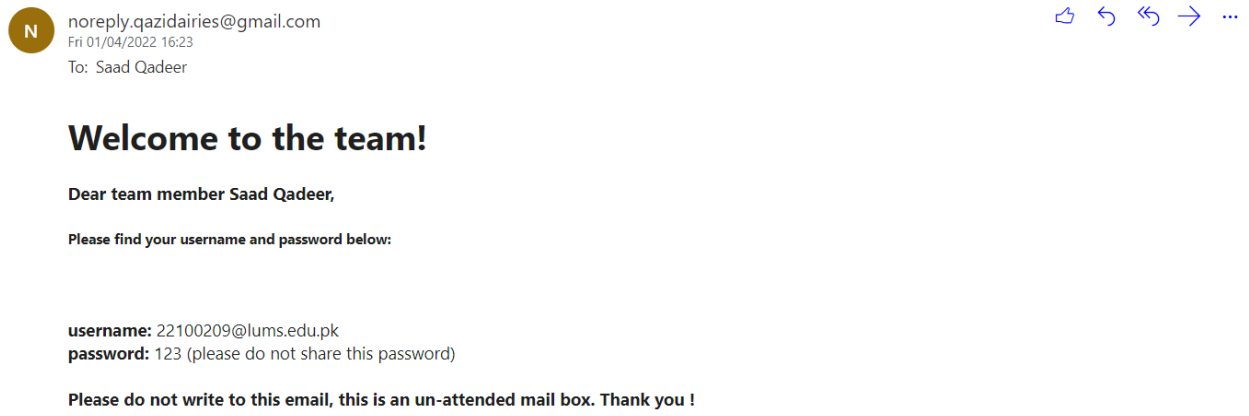


g. Adding Customers

By navigating to the customers page and clicking on add new customer we can add customers to the database.



An email is sent from the back end to the customers registered email and they can use the respective credentials to login to their account.



N

noreply.qazidairies@gmail.com
Fri 01/04/2022 16:23
To: Saad Qadeer

Welcome to the team!

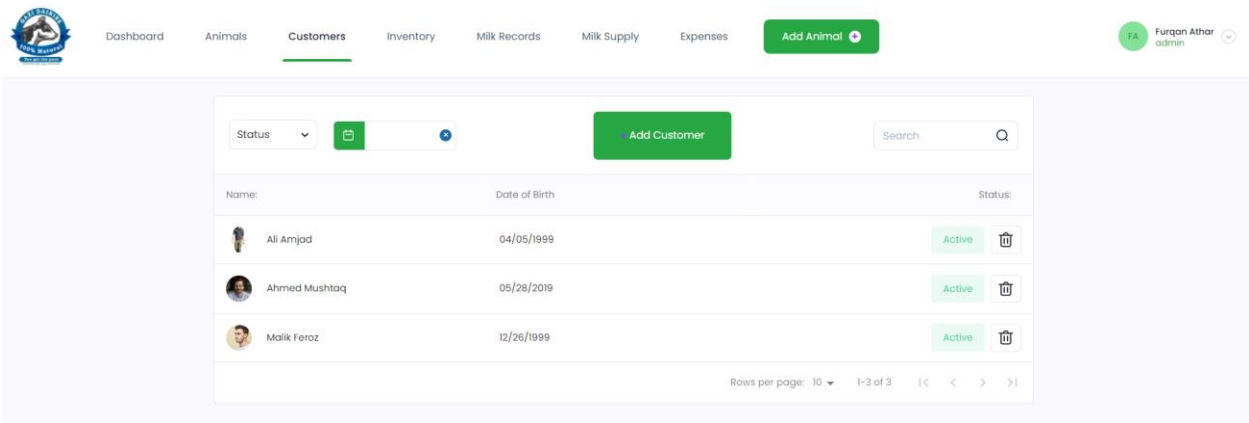
Dear team member Saad Qadeer,

Please find your username and password below:

username: 22100209@lums.edu.pk
password: 123 (please do not share this password)

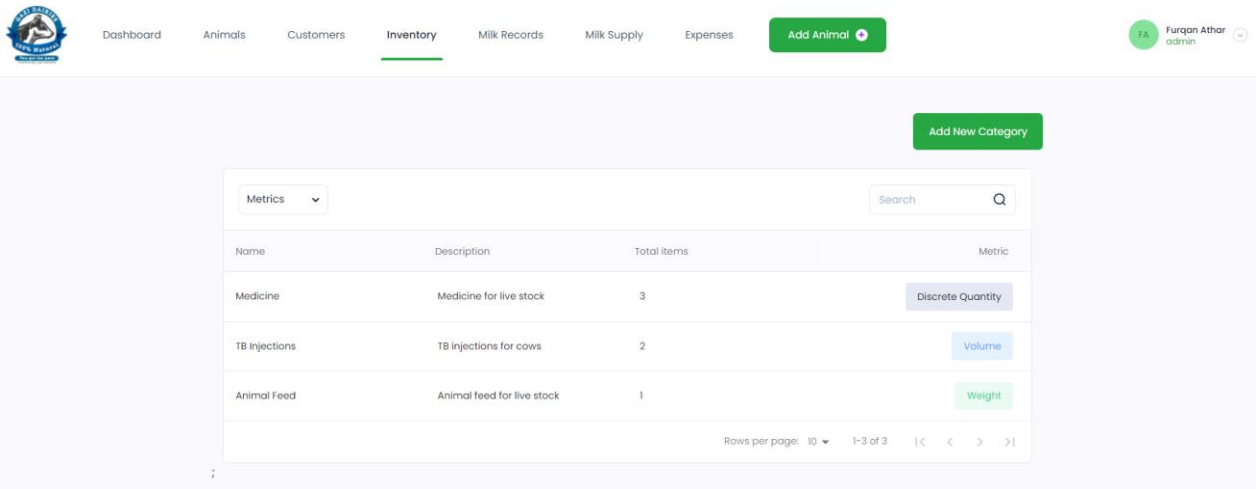
Please do not write to this email, this is an un-attended mail box. Thank you !

Navigating back to the customers page we can see the database with the added customers

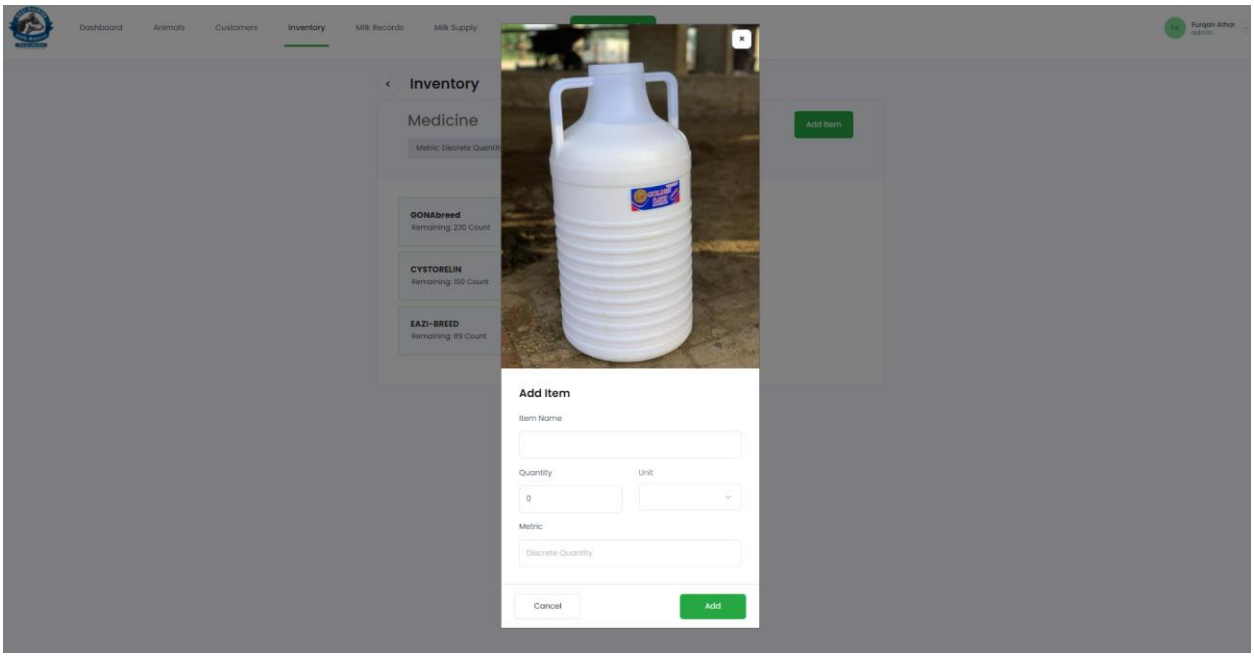


h. Inventory System

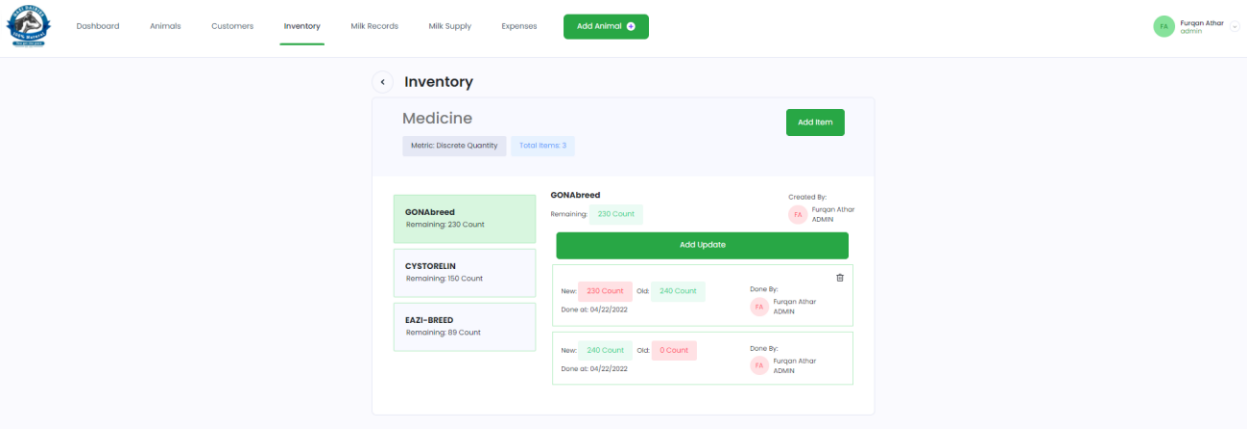
Navigate to the inventory page from the nav bar. A new Inventory category can be defined by pressing the Add new category



By pressing the required category new items can be added to the inventory-After filling the required details in the form below.

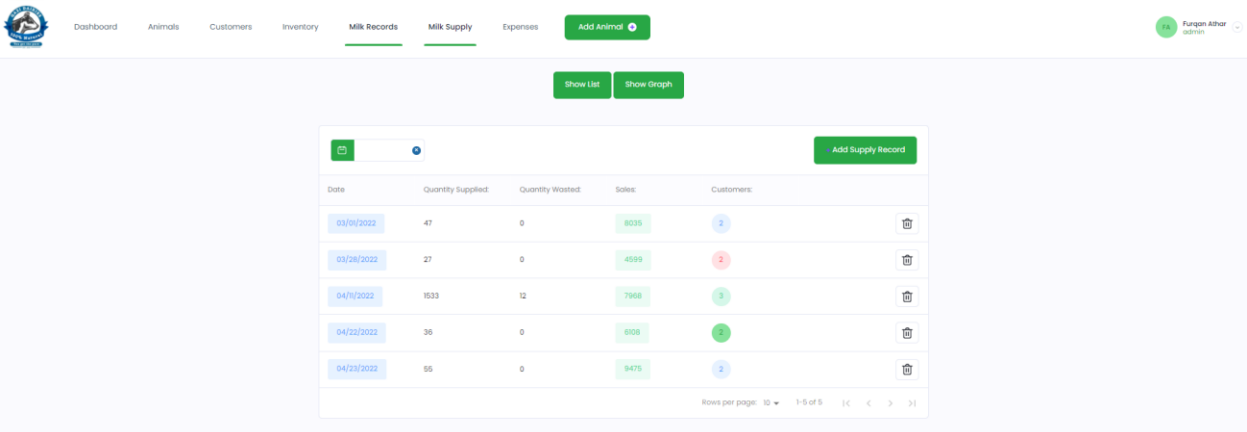


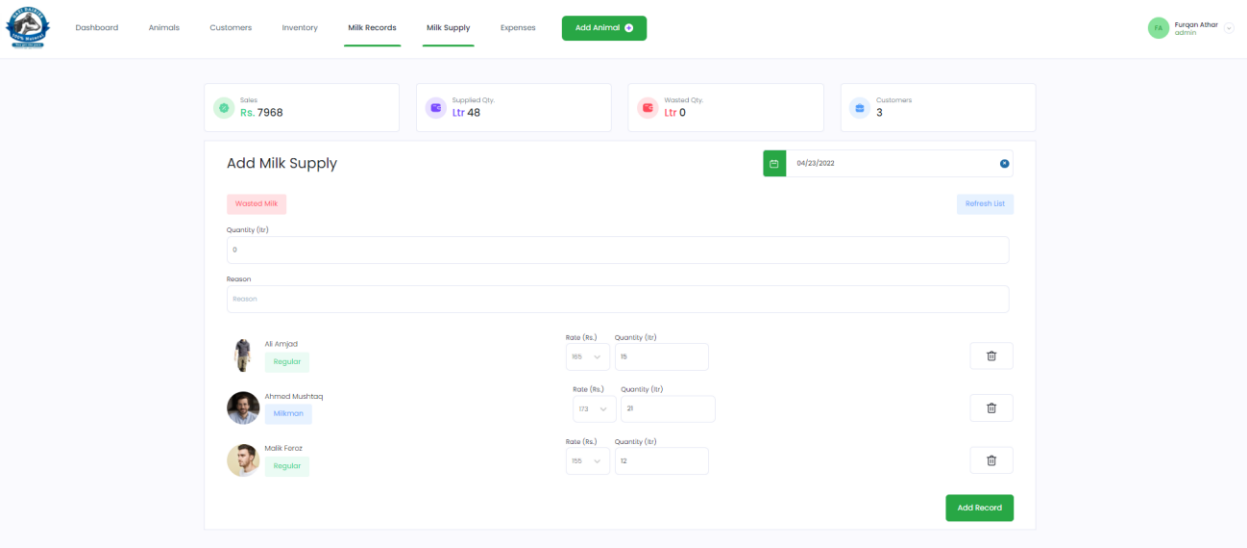
The required items can now be added or used from the inventory. Usage history can also be viewed.



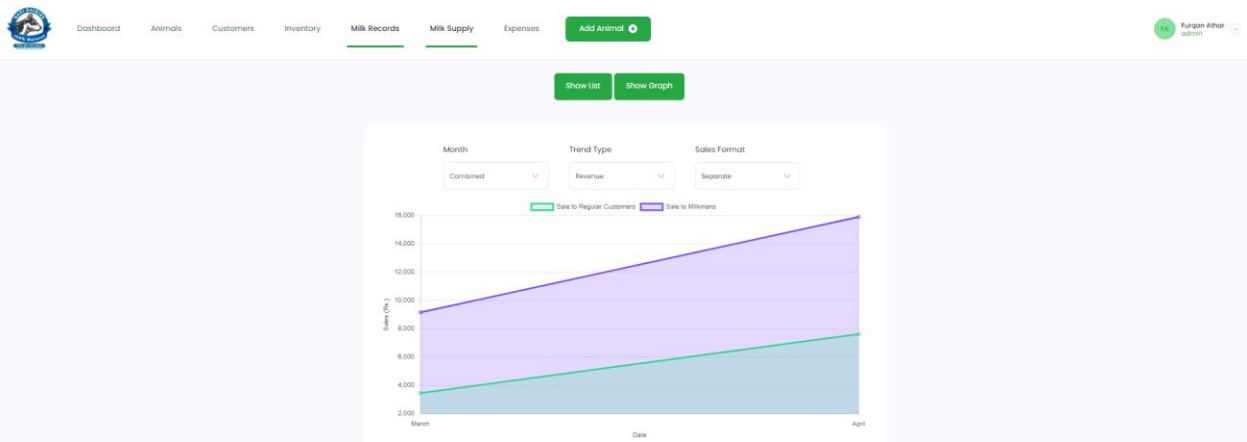
i. Milk Supply

Milk being supplied to various customers including regular customers and milk men-the record of that can been seen in the milk supply page.



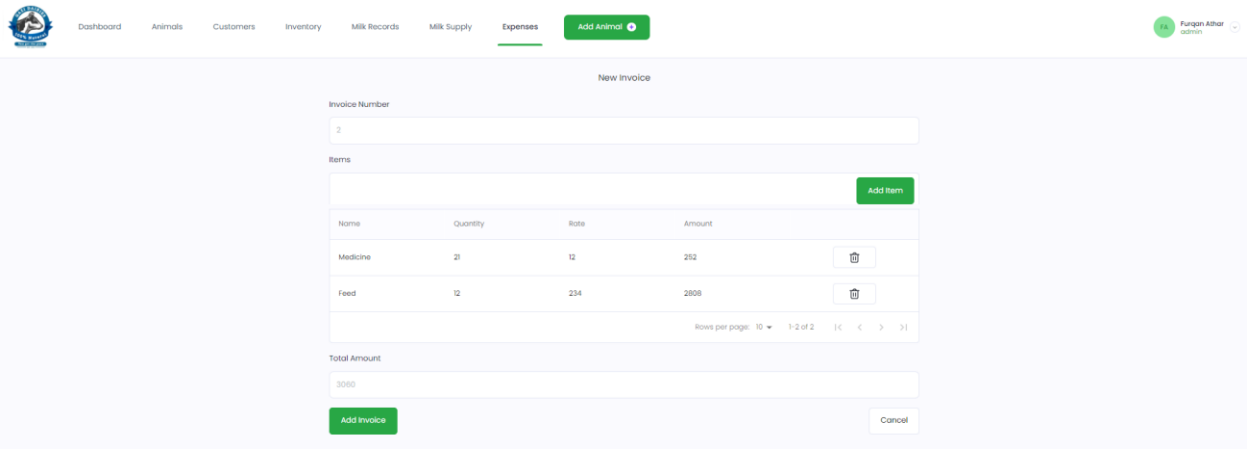


These records can now be views as various visualizations

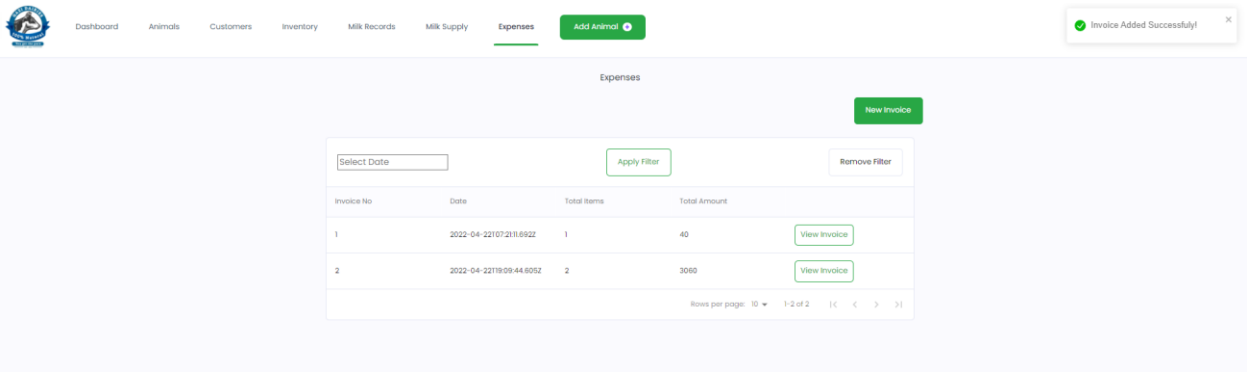


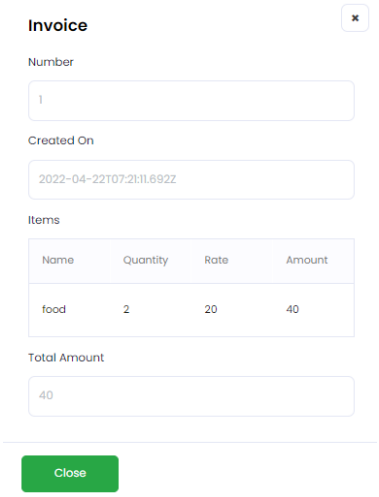
j. Daily Farm Expenditure

By navigating to the daily expenses page the farm employees can add the daily expenditure of the farm to the data base.



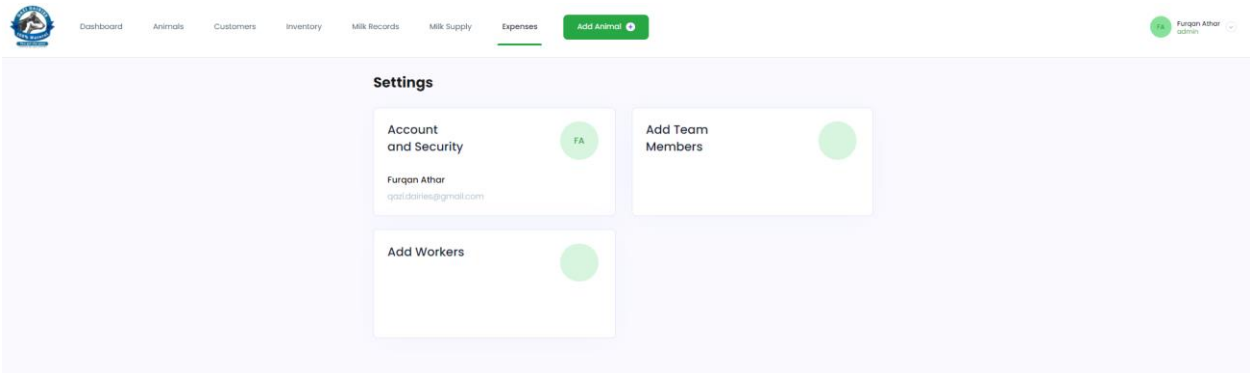
Invoices added can later be viewed for any required reasons by the admin/manager.

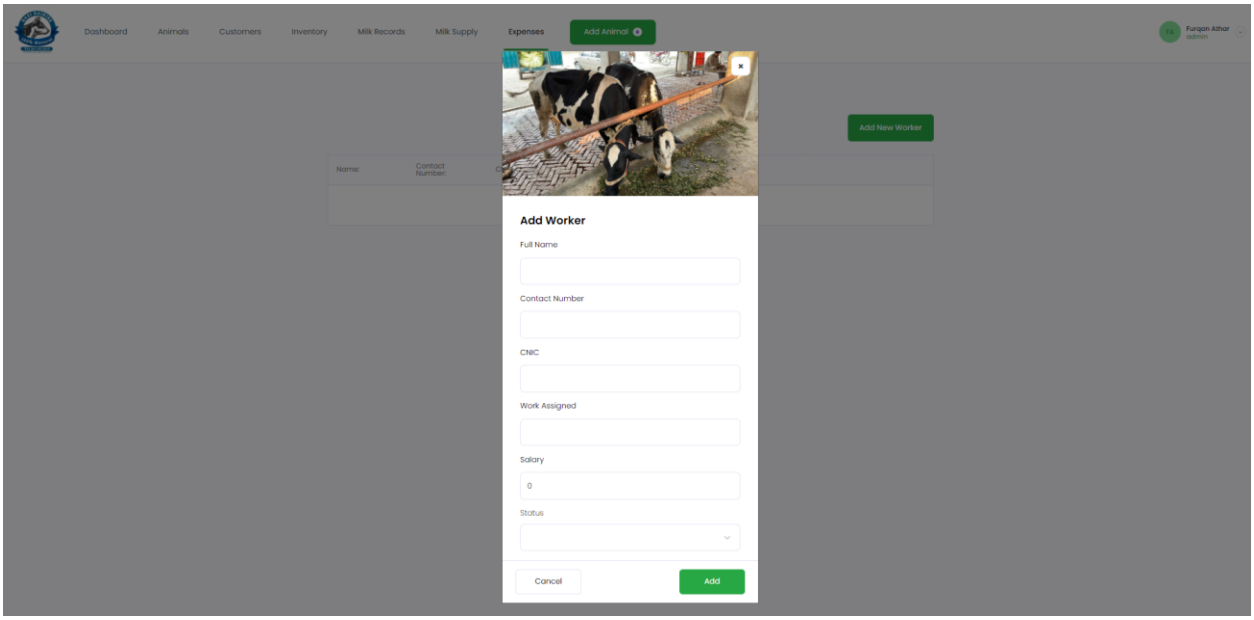


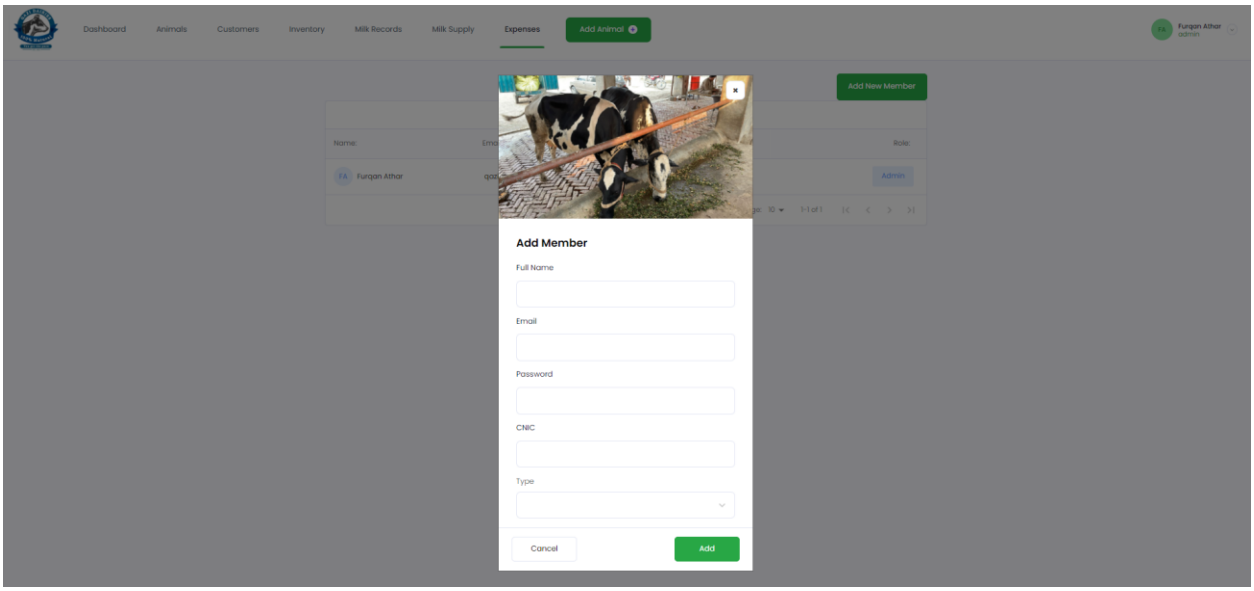


k. Assigning Work Roles and Building a team

The admin of the farm can assign managerial roles to their workers and even hire farm employees. They can also add members to their team by filling out a similar detailed form for each member like that of the customer functionality (See figure 17.13 and 17.14).




Dashboard
Animals
Customers
Inventory
Milk Records
Milk Supply
Expenses
Add Animal
Furqan Athar
admin
Add New Worker
Name:
Contact Number:
Add Worker
Full Name
Contact Number
CNIC
Work Assigned
Salary
0
Status
Cancel
Add


Dashboard
Animals
Customers
Inventory
Milk Records
Milk Supply
Expenses
Add Animal
Furqan Athar
admin
Add New Member
Name:
Role:
FA
Furqan Athar
Admin
Add Member
Full Name
Email
Password
CNIC
Type
Cancel
Add

8. Project Security

When working on a software engineering project, it is necessary to ensure that certain practices are followed to make sure that the end-product i.e., the application/website is secure and free from any type of vulnerabilities. Leaving weak spots in the code means that that it can be exploited by hackers and can leave to sensitive date being stolen.

We read through the OWASP Top 10 Web Applications security risk and filtered out three of the most important security risks that apply to our application. According to OWASP, “Companies should adopt this document and start the process of ensuring that their web applications minimize these risks. Using the OWASP Top 10 is perhaps the most effective first step towards changing the software development culture within your organization into one that produces more secure code.”

a. Project Threats

- A02:2021 – Cryptographic Failures
- A03:2021 – Injection Attacks
- A07:2021 – Identification and Authentication Failures

b. Potential Losses

- Injection
    - Loss of Data
- Cryptographic Failures
    - Loss of Sensitive Data leading to Loss of Business and Bad reputation
- Identification and Authentication Failures
    - Loss of Integrity

c. Security Controls

- Input Validation to deal with Injection Attacks
- Data Classification, Data Encryption to deal with Cryptography failures
- Multi Factor Authentication and User Roles to deal with Identification and Authentication Failures

The mentioned controls are preventive/protective in nature.

d. Static and Dynamic Security Scanning Tools

- **Static Application Security Testing (SAST)**
  - White Box Testing
  - We have complete information about the system
  - Examine Source Code (Static) to detect and report weaknesses

- **Dynamic Application Security Testing (DAST)**
  - Black Box Testing
  - Examine Application in Running State
  - run on operating code to detect issues with
    - Interfaces
    - Requests
    - Responses

9. Risk Management

Potential Risks and Mitigation Strategies

Sr.	Risk Description	Mitigation Strategy
1.	Scope variations, when the project tends to overrun the agreed upon time.	Reduction, the scope changes can be reduced by taking in account the changes that are possible along the way with the client this includes agile methodology which gives time for the client to reflect, after small iterations.
2.	Inadequate productivity of the development team causing them to fall behind the work schedule and not meeting milestones/targets that were set initially.	One possible way to deal with such situations is to first set targets that are realistic and achievable within the given time frame. Another way can be for the development team members to keep check on each other and meet up at fixed periods. Furthermore, group member screening can also be done, to make sure that all the members are almost of the same caliber and to assign tasks based on the skill set of each member. To achieve more organization a manager role can be assigned to keep check and balance.
3.	Imprecise estimation, arises since a timeframe must be given to the client and other estimations.	Reduction, Developers can consider the uncertainty involved in the estimations and certain allocation for estimations such as time frames. Reengineering process so more interaction with the people needed from the client's organization.
4	Lack of a clear product vision. This is a potential risk that may occur if the development team and or client are unclear on what they want for the final product. This may also occur if they are not on the same page regarding the final product outcome.	A mitigation technique to deal with this may be too frequently ask questions from the client regarding the final product they want, meet up with the client, show the client the product developed in the early stages and what the development team is planning to do in the coming time. To make the development of the product roll swiftly there needs to be a clear final product vision about what you want to achieve.

5	Involvement with the end user, making software's for client's users (external user) is difficult since they are opposed to change.	Avoidance, test with users using a beta version of the software or have feedback mechanism that considers the opinions of the end user, conduct meetings with the end user to understand the design decisions.
6	Over simplistic or Over complicated design of the product. This problem may arise after the product has been developed. The client may find the product design to be over simplistic (the product does not offer an adequate number of features) or over complicated (the feature is so much that they overwhelm the user)	. The mitigation strategy to deal with this is to start with a minimum viable product, launch it, get feedback from the user and integrate the received feedback into the product. This can be even done repeatedly to further enhance the product.
7	Poor quality code, it would entail poor coding practices, released without testing, and rushed.	Reduction, following coding good practices, conduct unit testing for the developed components of the code. There can be a dedicated manager to monitor the code base and along with frequent code reviews.
8	Inadequate engagement/involvement of the client with the development team which may lead to a gap between what the client wants and what the developer team thinks the client wants.	Keep the client more involved/ engaged with the development team. The client should be able to clearly put forth their expectations of the product and the development team must be able to convey to the client effectively and efficiently what is achievable in what time span and at what cost.
9	Lack of adequate resources, developers can leave a project at a point in time, creating problems for the project's completion.	Avoidance, Proper documentation of the code base can be maintained, or considering having new team members with a training period.
10	Disagreement between the client stakeholder and the development team.	More client involvement and keeping them informed. Both the clientele and development team need to be vocal and expressive about their concerns and ideas. Either of them should not feel that they are left out and that their ideas and opinions are not being taken under consideration.

Table 9(a)

10. Testing and Evaluation

For any software engineering-based project, both the testing and evaluation phases are one that are crucial to the performance and reputation of the final product. It is also important to ensure the necessary industry standards are being implemented in the relevant product. Based on the results of the test the product is then to be evaluated and required changes should be made to the code to cover the shortcoming of the product.

Launching the final version of product with little to no testing means that the App may react in unexpected ways when in the hands of the end user, this may ultimately lead to the user not wanting to use the app-bad reputation for the company/product.

After each sprint vigorous testing should take place to find various bugs and to see if all the corner cases for user defined input have been handled properly or not.

Testing for our project was done manually. Various test cases were created for both empty, valid, and invalid inputs-it was then observed how the system responded to the specified inputs. Along with that other corner cases were also checked for each use cases.

Testing files and documents for all the sprints can be found in the respective GitHub repository of the project.

For automated testing we had initially planned to use Selenium: https://www.selenium.dev/

Sample Test Cases

Step No	User actions	Inputs	System response	Expected Outputs	Actual Output	Test Result (Pass/Fail)	Comments
1	Admin Navigates to Milk Supply Page from Nav Bar	Click on milk supply in Nav Bar	System opens the milk supply page	-	-		
View Farm Statistics							
T01							
2	Click on Show Graph	-	System Navigates to the Statistics Page and displays the statistics in the form of various graphs	Required Graph	Required Graph	PASS	
T02							
3	Select the Month from the drop down menu at the top	-	System now shows stats for the requested month	Required Graph	Required Graph	PASS	
T03							
4	Select the metric to display, i.e. from the top drop down menu	Select Quantity from drop down menu	System now shows the stats for quantity of milk sold	Required Graph	Required Graph	PASS	
T04							
5	Select the type of graph from top drop down menu	Select Combined from drop down menu	System now shows the stats for milkmen and regular customer combined to show the overall quantity sold, or the overall revenue generated	Required Graph	Required Graph	PASS	

Step No	User actions	Inputs	System response	Expected Outputs	Actual Output	Test Result (Pass/Fail)	Comments
1	Admin navigates to the expenses page	-	system opens the expenditure page of the farm	-	-		
2	Click on new invoice	-	System opens the expenditure form	System opens form	System opens form		
3	Click on add item	-	System opens the invoice form	System opens form	System opens form		
Add Invoice							
T01							
4	Add required details Click Add Item Click Add Invoice Click Contiue on Reaffirmation Message	Name: Live Stock Feed Qty: 30 Kg Rate: 55 Rs/Kg	System calculates invoice amount and adds invoice to record Amount = 1650	System Display Success Message Invoice Added Sucessfully	System Display Success Message Invoice Added Sucessfully	PASS	
T02							
5	Add required details Click Cancel	Name: Live Stock Feed Qty: 30 Kg Rate: 55 Rs/Kg	System Closes form	-	-	PASS	
T03							
6	Add required details Click Add Item Click Add Invoice Click Contiue on Reaffirmation Message	Name: Live Stock Feed Qty: - Kg Rate: - Rs/Kg	System does not invoice to expenses	System Display Error Message No items added	System Display Error Message No items added	PASS	
Delete Item from Invoice							
T01							
7	Click on the delete button next to required item in invoice to remove entry	-	System Removes Item from Invoice list	Success Message Item Removed Successfully	Success Message Item Removed Successfully	PASS	

11. Conclusion

a. Summary

Brain Storming and Idea finalization

The first step that we performed when approaching our Senior Year Project was the brain storming part. We needed to think of ideas that were interesting but at the same time they should be practical and to some extent innovative as well. One of our group members mention about a family run business, we decided to build a management system for that business. One reason for doing so was that would be able to secure a possible sponsor and the other was we felt that this was a relatively untapped market possible digitization of this market could lead to more future projects in the agriculture and farm sector of Pakistan.

Finalizing the Framework

After the idea had been finalized, we began thinking about the framework, being CS students, we wanted to go with a framework that we had never had the opportunity of working with before so that this year long project would be one that was filled with learning opportunities. We decided to use the MERN stack. One because it would be a great learning opportunity, none of the team members had previously used this stack and second this was a relatively popular framework in the industry.

Documentation

What we have learned through the course of this entire project is the importance of documenting everything we do before we even start working on the development phase. The documentation worked as a guideline, after the documentation phase was complete, we had a clear image about what we wanted the final product to look like. This would serve as our blueprint; we could now start working along the lines of these blueprints and make changes wherever needed along the course of the project.

Division of Labor

This was a year-long project, spread over the span of two semesters, so division of Labor was a necessary phase. Multiple meetings were conducted to divide the work amongst all the group members equally so that no other group member would have to face considerably more burden than the rest of the team. The work was divided based on use cases, use cases were equally divided

amongst the group members and each member working on the project had to do the full stack development of that use case.

Sprint Plan

After we had our finalized use cases along with the work divided amongst the team members, we needed to plan our sprints so we could spread the work evenly over the remaining time we had to complete the project along with managing workload of the other courses

Feedback and Setting Deliverables

To make sure that we were doing the work on time and were on the right track we had biweekly meetings with Sir Waqar: He would set deliverables that we needed to complete before the next meeting. His valuable feedback allowed to see what we might be doing right/wrong and make up for our shortcomings during the entire project: We were able to learn about the importance of feedback when working on large projects. Getting timely feedback is important as you can incorporate changes during the early stages of development, if the feedback stage only comes at the end, after development has been completed it becomes nearly impossible to find bugs and make respective changes in the project code.

b. Challenges

Elaborate the issues and challenges, both technical and non-technical, faced during this project and how you have addressed them.

We faced several challenges over the span of the project. Some of them where technical while others were non-technical. This was our 2 major and hence we did not have a lot of experience managing these issues however it was a great learning experience, and we were able to learn some essential Do's and Don'ts. Some of the challenges that we were faced with are listed below:

The Hybrid Nature of Communication

Most of our group meeting with the team members and with Sir Waqar were through an online medium. When communicating through such a medium at times it becomes difficult to convey our thoughts and messages. And when you have multiple hour-long meetings with your group members it often becomes frustrating as well.

Finding Free Services

When came the time for deployment and hosting our software, we were met with a new trouble-finding hosting services that were free and efficient. We could not find a single service that allowed some-what efficient in terms of memory and speed for hosting our website. We had to make use of whatever we could find. Even when using our student packages, such as GitHub student or using the student scholarship for digital oceans we had to provide credit card information to use them-none of our group members had an active credit card that could be used.

Dependencies and Packages

Another issue that we were faced with was that all of our group members had install dependencies and packages of different versions. Some were using dependencies that were install 2 years ago for a course project other were using newer versions. Due to this mismatch the code often ended up crashing. However, we wasted a lot of our time trying to get this issue resolved as we did not think that this would even be an issue but turns out when packages are updated some changes, such as change in the name of library functions, are made. Even the site we were deploying to at one point updated their dependencies and it stopped matching the version of our project which caused even more errors when deploying the app.

Debugging

Debugging was another challenge. What we have realized is that debugging is a headache of its own. After spending hours coding, we had to spend hours debugging as well.

Although there were many challenges and hurdles along the way, nevertheless these were all a learning experience of their own. Which each challenge we overcame, we made sure not to repeat the mistakes and got a chance to get our hands dirty before stepping into the industry.

c. Future Works

Some of the possible future that can be applied to our applications are listed below:

1. Building a customer side interface to place orders

2. Extending the application to other in farm management areas

3. Extending the application to include other areas of the agriculture sector in attempts to digitalize the agricultural sector of Pakistan

4. Adding a marketplace for Dairy Farms to order and purchase inventory items

5. Adding quality assurance page for the customers to tell them about the composition of the dairy products and let them know of the quality and assurance tests that were performed to ensure that a healthy and safe final product is being delivered to their door.

6. Implementing a more advanced inventory system

7. Addition of payment gateways.

12. Deployment Guidelines

GitHub Link: https://github.com/FurqanAthar/P04-Dairy-Farm-Management-System

The Deployment of the project was done on Heroku

We used the following process to deploy our project.

- Add a Procfile to the local repository
- Make sure the front end is connected to the backend

```
npm run server
```

- Create App repository on Heroku

```
heroku create qazidairies
```

- Initialize local repository

```
git init
```

- Push latest version of the code to the local git repository

```
git add .
```

- Commit changes

```
git commit -m “deploy project”
```

- Deploy to Heroku using git

```
Heroku git:remote -a qazidairies
git push heroku master
```

Finally, access the app on Heroku and add the configuration variables (config vars) to the project settings - Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.

13. Review checklist

Internal Review of the final report was performed by the following team members in the respective stated sections of this report.

Chapter/Section Name	Reviewer Name(s)
Sections 1-10	Saad Qadeer, Furqan Athar
Sections 11-17	Abdullah Saleem, Khawaja Junaid

14. References

[1] https://owasp.org/www-project-top-ten/

[2] https://owasp.org/Top10/A02_2021-Cryptographic_Failures/

[3] https://owasp.org/Top10/A03_2021-Injection/

[4] https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

[5] https://en.wikipedia.org/wiki/Class_diagram

[6] https://about.gitlab.com/

[7] https://www.invicti.com/

[8] https://www.selenium.dev/

[9] https://nodejs.org/en/

[10] https://www.mongodb.com/cloud/atlas/register2

[11] https://dashboard.heroku.com/

[12] https://reactjs.org/