x

Belize Cattle Tracker

Report 1: System Specifications

Group Members:
Joanne Y.
Chimezirim A.
Rene S.
Miguel A.
Juan S.
Cameron T.

Submitted To:
CMPS4131
Mr. Manuel Medina
April 6, 2022

Contribution Breakdown

SIGNATURE BLOCK			
Statement	I did my share of the work, and I have a general understanding of the contents of the assignment.		
Team Member	**Contribution**	**Signature**	**Date**
Joanne Yong	Project Management. Did Problem Statement, Glossary, Preliminary Designs, Plan of Work, Hardware Requirements. Assisted with Use Cases, Traceability Matrix, Fully Dressed Description. Did Data Model and Persistent Data Storage, Class Diagrams, Data Types and Operation Signatures, User Interface Design and Implementation, and Project Management. Assisted with Interaction Diagrams, conceptual model, Algorithms, Concurrency, and Data Structures.		05/16/2022
Miguel Avila	Assisted with Problem Statement. Did System Specifications. User Effort Estimation Identifying Subsystems Did Data Structures and Design of Test.		05/16/2022
Rene Sanchez	Did Problem Statement. Did Casual Description Assisted with Use Case Diagrams Assisted with System Sequence Diagrams Did Subsystems Did Architectural Styles Did Conceptual Model, Interaction Diagrams, User Interface Design and Implementation,		05/16/2022
Juan Carlos Sarabia	Did traceability matrix Fully Dressed Description Assisted with Identifying Sub System Did Connectors and Network Protocols. Assisted with Conceptual Model (Domain Models).		05/16/2022
Chimezirim Amagwula	Did System Specifications/Requirements. Assisted with User Effort Estimation. Assisted with Traceability Matrix. Assisted with Problem Statement. Did Global Control Flow. Assisted with Project Management. Did Conceptual Model, System Operation Contracts, Traceability Matrices, Interaction Diagrams, Concurrency.		05/16/2022

Cameron Tillet	Did the Use Case Diagrams Did the System Sequence Diagrams Did the Architectural Styles Did the Mapping Subsystems to Hardware Algorithms		05/16/2022

Responsibility Matrix

Note: See point allocations in the table below.		Team Member Name					
		Joanne	Rene	Miguel	Chimeziri m	Juan	Cameron
Responsibility Level	Project management (*10 points*)	95%			5%		
	Sec.1: Customer Statement of Requirements (*9 points*)	80%	15%		5%		
	Sec.2: System Requirements (*6 points*)		20%	40%	40%		
	Sec.3: Functional Requirements Specification (*30 points*)	6%	29%		15%	25%	25%
	Sec.4: User Interface Specs (*15 points*)	50%		20%	30%		
	Sec.5: System Architecture (15 *points*)	15%	25%	15%	15%	15%	15%
	Sec.6: Domain Analysis *(25 points)*	27.5%	27.5%		27.5%	17.5%	
	Sec.7: Interaction Diagrams *(30 points)*	30%	35%		35%		

	Sec.8: Class Diagrams & Interface Specification *(15 points)*	80%			20%		
	Sec.9: Algorithms & Data Structures *(12 points)*	15%		32.5%	32.5%		20%
	Sec.10: User Interface Design & Implementation *(4 points)*	50%	50%				
	Sec.11: Test Case Design*(15 points)*	20%	30%	50%			
	Sec.12: Plan of Work (*19 points*)	100%					

Responsibility Allocation Chart

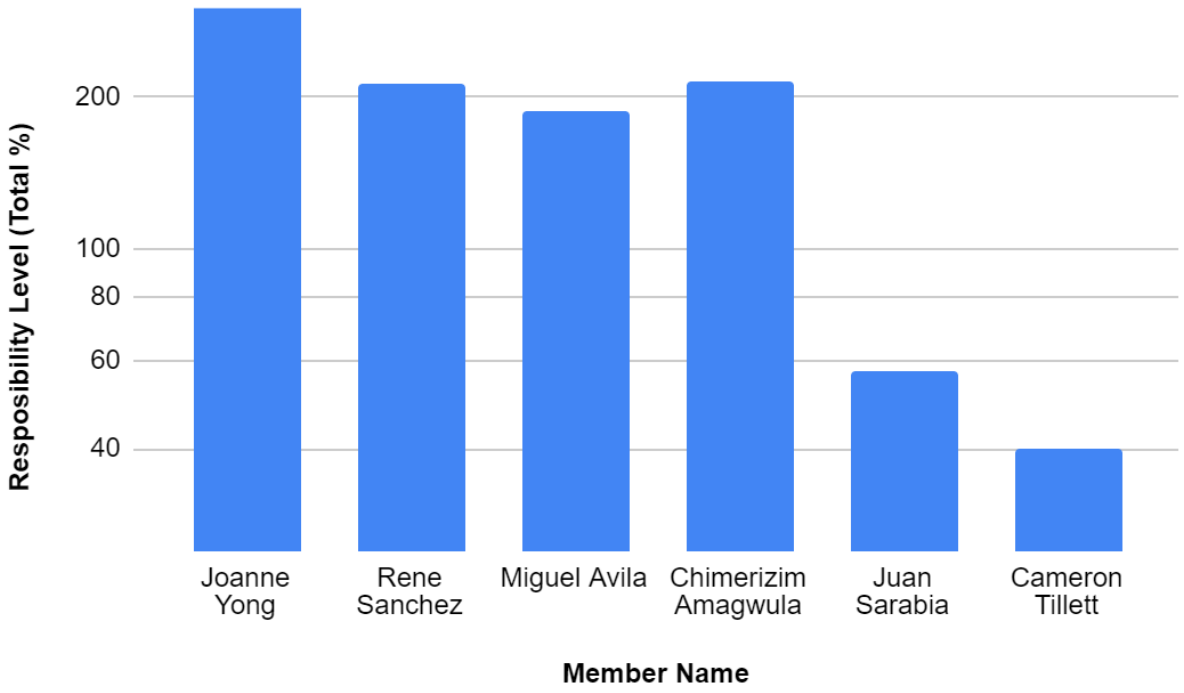


Table of Contents

Contribution Breakdown **2**

Responsibility Matrix **3**

Table of Contents **5**

Customer Statement Requirements **6**
Problem Statement *6*
Glossary of Terms *11*

System Requirements **16**
Enumerated Functional Requirements *16*
Enumerated Nonfunctional Requirements *18*
On Screen Appearance Requirements *18*

Functional Requirements Specification **20**
Stakeholders *20*
Actors and Goals *20*
Casual Description *24*
Use Case Diagram *27*
Traceability Matrix *28*
Fully Dressed Description *29*
System Sequence Diagram *32*

User Interface Specification **34**
Preliminary Designs (MOCKUPS) *34*
User Effort Estimation *38*

System Architecture **39**
Identifying Subsystems *39*
Architecture Styles *46*
Mapping Subsystems to Hardware *48*
Connectors and Network Protocols *48*
Global Control Flow *48*
Hardware Requirements *49*

Plan of Work **50**
Gantt Chart *50*

Breakdown of Responsibilities **51**

References **53**

Analysis and Domain Modeling

Conceptual Model

(I) Concept Definitions

Concept Definition for CreateCattleProfile(UC1)

Responsibility Description	Type	Concept Name
RS.1 Coordinate actions of concept associated with the use case delegate the work to other concepts.	D	Controller
RS.2 Verifies if the user is a farmer.	D	Verifier
RS.3 System displays farmer portal.	K	PortalDisplay
RS.4 Select option to create cattle profile.	D	ProfileCreator
RS.5 Enter cattle information.	D	EnterInfo
RS.6 Selects the save option.	D	SaveInfo
RS.7 Database stores cattle information.	D	Database Connection
RS.8 Notifies farmers that information was successfully added.	D	Notifier

Concept Definition for ViewProductDetails(UC2)

Responsibility Description	Type	Concept Name
RS.1 Coordinate actions of concept associated with the use case delegate the work to other concepts.	D	Controller
RS.2 User enters roleID into the form.	K	Interface Page
RS.3 Medium for customer to scan QR Code	K	Scanner
RS.4 QR Code is scanned using the Medium	K	QR Code
RS.5 Prepare a database query that best matches the QR Code and retrieve the records from the database.	D	DatabaseRetriever
RS.6 Database provides product details.	D	Database Connection

RS.7 Render the retrieved records for sending to the actor's web browser for display.	K	DisplayReport

Concept Definition for AddProductInformation(UC5)

Responsibility Description	Type	Concept Name
RS.1 Coordinate actions of concept associated with the use case delegate the work to other concepts.	D	Controller
RS.2 User enters roleID into the form.	K	Interface Page
RS.3 Verifies if the user is a packaging manager.	D	Verifier
RS.4 System displays packaging portal.	K	PortalDisplay
RS.5 System prompts for the search filter criteria	K	SearchRequest
RS.6 Prepare a database query that best matches the actor's search criteria and retrieve the records from the database.	D	DatabaseRetrieve
RS.7 Render the retrieved records for sending to the actor's web browser for display.	K	DisplayReport
RS.8 Select add option for displayed document.	D	AddInfo
RS.9 Enter packaging information.	D	EnterInfo
RS.10 Selects the save option.	D	SaveInfo
RS.11 Store saved info into database.	D	DatabaseStore
RS.12 Notifies Actor that information was successfully added	D	Notifier
RS.13 Selects the generate QR code option.	D	GenerateQR
RS.15 System prompts the React library to generate a new unique QR code.	K	QRRequest
RS.16 Prepare a database query that best matches the QR's data requirements and retrieve the records from the database.	D	QRDataRetrieve
RS.17 Render the retrieved QR for sending to the actor's web browser for display.	D	DisplayQR

Concept Definition for GenerateReport(UC7)

Responsibility Description	Type	Concept Name
RS.1 Coordinate actions of concept associated with the use case	D	Controller

delegate the work to other concepts.		
RS.2 User enters roleID into the form.	K	Interface Page
RS.3 Verifies if the user is a BAHA admin.	D	Verifier
RS.4 System displays the BAHA admin portal.	K	PortalDisplay
RS.5 Select option to Generate Reports.	D	ReportGenerator
RS.6 System displays search form .	K	SearchFormDisplay
RS.7 Select search filter/cattle category.	D	FilterSearch
RS.8 System prompts for the search filter criteria.	K	SearchRequest
RS.9 Prepare a database query that best matches the actor's search criteria and retrieve the records from the database.	D	DatabaseRetrieve
RS.10 Render the retrieved records for sending to the actor's web browser for display.	K	DisplayRecords
RS.11 Select cattle id.	D	SelectID
RS.12 Render the retrieved profile/report for sending to the actor's web browser for display.	K	DisplayReport

Concept Definition for AddSlaughterInformation(UC8)

Responsibility Description	Type	Concept Name
RS.1 Coordinate actions of concept associated with the use case delegate the work to other concepts.	D	Controller
RS.2 User enters roleID into the form.	K	Interface Page
RS.3 Verifies if the user is a slaughterhouse manager.	D	Verifier
RS.4 System displays the slaughterhouse portal.	K	PortalDisplay
RS.5 System prompts for the search filter criteria	K	SearchRequest
RS.6 Prepare a database query that best matches the actor's search criteria and retrieve the records from the database.	D	DatabaseRetrieve
RS.7 Render the retrieved records for sending to the actor's web browser for display.	D	DisplayReport

RS.8 Select add option for displayed document.	D	AddInfo
RS.9 Enter slaughter information.	D	EnterInfo
RS.10 Selects the save option.	D	SaveInfo
RS.11 Notifies Actor that information was successfully added	D	Notifier
RS.12 Store saved info into database.	D	DatabaseStore

(II) Association Definitions

Association Definition for CreateCattleProfile(UC1)

Concept Pair	Association Description	Association Name
Verifier ↔ PortalDisplay	System verifies the role ID of an actor and displays their portal.	Verifies
PortalDisplay↔ PortalCreator	When the system displays the actor's portal the actor can create a cattle profile.	Presents Portal
PortalDisplay↔ InterfacePage	The portal renders and displays on the Interface Page.	Shows Interface Page
ProfileCreator↔ EnterInfo	Before a cattle profile is created, the user has to enter in the cattle info for that account.	Creates Profile
EnterInfo↔Database Connection	When the actor presses the save button after the cattle info is entered, the information gets stored to a table in the database.	Initiates Database
SaveInfo↔Notifier	The actor is prompted with a success message when the save button is selected.	Notifies Success
Database Connection↔Notifier	The system saves the entered cattle information into the database. After that, it notifies the actor with a confirmation message.	Notifies Success

Association Definition for ViewProductDetails(UC2)

Concept Pair	Association Description	Association Name
Controller ↔ Database	Connects to the database	Connects Database

Connection		
Scanner ↔ Display Report	Once the QR code is scanned, the record is retrieved from the database	Retrieves Record
Database Connection ↔ Display Report	Provides data that matches the actor's QR code	Provides Data
Display Report↔InterfacePage	The report info renders and displays on the Interface Page.	Shows Interface Page

Association Definition for AddProductInformation(UC5)

Concept Pair	Association Description	Association Name
Controller ↔ DatabaseRetrieve	Controller passes search requests to Database Connection.	Conveys Request
PortalDisplay↔ SearchRequest	When the portal is displayed, the actor can click the search request button to search for a criteria (cattleID).	Search Data
DatabaseRetrieve ↔ DisplayReport	Provides data that matches the Actor's search criteria.	Provides Data
Display Report↔InterfacePage	The report info renders and displays on the Interface Page.	Shows Interface Page
AddInfo↔ EnterInfo	When the add button is selected, the actor can enter in packaging information.	Enters Info
AddInfo↔ SaveInfo	When the save button is selected, the system will save the added information and push the data to the database.	Saves Data
SaveInfo ↔ DatabaseStore	Once the information has been saved, the information is placed into a table in the database.	Stores Data
GenerateQR↔QRRequest	There should be a prompt for a QR code to be generated once the user selects the option for it.	Selects Info

QRRequest↔QRRetrieve	Once the QR code has been generated, the accurate information should be retrieved from the database.	Retrieves Data
QRRetrieve↔DisplayQR	The Information that has been retrieved from the Database is displayed on the device.	Displays DataVeri

Association Definition for GenerateReport(UC7)

Concept Pair	Association Description	Association Name
Controller ↔ DatabaseRetrieve	Controller passes search requests to Database Connection.	Conveys Request
Portal Display ↔ Search Request	When the portal is displayed, the actor can click the generate reports button to search for cattle and retrieve reports about them.	Search Data
DatabaseRetrieve ↔ DisplayRecords	Provides data that matches the Actor's search criteria.	Provides Data
Display Records↔InterfacePage	The report info renders and displays on the Interface Page.	Shows Interface Page
Display Records ↔ Select ID	When the records are displayed, the actor can click a cattle id to access the animal's profile/report.	Search Report
DatabaseRetrieve ↔ DisplayRecords	Provides data that matches the Actor's search criteria.	Provides Data

Association Definition for AddSlaughterInformation(UC8)

Concept Pair	Association Description	Association Name
Controller ↔ DatabaseRetrieve	Controller passes search requests to Database Connection	Conveys Request
Verifier ↔ PortalDisplay	System verifies the role ID of an actor and displays their portal.	Verifies
PortalDisplay↔	The portal renders and displays on the Interface	Shows Interface Page

InterfacePage	Page.	
PortalDisplay↔ SearchRequest	When the portal is displayed, the actor can click the search request button to search for a criteria (cattleID).	Search Data
SearchRequest ↔DataabaseRetreive	Once a search request is entered, the system will be prompted to retrieve the records from the database.	Retrieves Data
DatabaseRetrieve ↔ DisplayReport	Provides data that was retrieved from the database and displays the report to the actor on a newly rendered page.	Provides Data
Display Report↔InterfacePage	The report info renders and displays on the Interface Page.	Shows Interface Page
AddInfo↔ EnterInfo	When the add button is selected, the actor can enter in slaughter information.	Enters Info
AddInfo↔ SaveInfo	When the save button is selected, the system will save the added information and push the data to the database.	Saves Data
SaveInfo ↔ DatabaseStore	Once the information has been saved, the information is placed into a table in the database.	Stores Data

(III) Attribute Definitions

Attribute Definition for CreateCattleProfile(UC1)

Concept	Attributes	Attribute Description
Verifier	User ID	Used to identify the Actor's identity, which will determine what operations that can be performed in turn
	UserRole	Determines the Actor's role
Database Connection	User's Identity	Prepare a Query that retrieves based on the class that is selected
Notifier	Data Status	Used to show a message that the Actor's Data has been successfully added to the Database.

ProfileCreator	Cattle ID	Used to Create the Cattle ID, that will be added to the Database.

Attribute Definition for ViewProductDetails(UC2)

Concept	Attributes	Attribute Description
QR Code	QR ID	Determines the ID of the QR Code that was scanned
	Class Information	Identifies the class of the QR code that was scanned
Scanner	Medium Type	Used to Determine what kind of Device is being used to scan QR Code

Attribute Definition for AddProductInformation(UC5)

Concept	Attributes	Attribute Description
Verifier	User ID	Used to identify the Actor's identity, which will determine what operations that can be performed in turn
	UserRole	Determines the Actor's role
Search Request	Search Criteria	Needed to Prepare a Database Query to return a result that matches the Actor's search.
AddInfo	CattleInfo	Used to Add information about a specific Cattle into the Database. Can be broken down into more Attributes such as "cut of meat", "breed" and etc.
SaveInfo	ClassID	Used to update the cattle Data in the Database after new information has been added.
Notifier	Data Status	Used to show a message that the Actor's Data has been successfully added to the Database.
QRDataRetrieve	QR ID	Used to Identify the proper QR information so that the accurate information can be displayed on the screen.

Attribute Definition for GenerateReport(UC7)

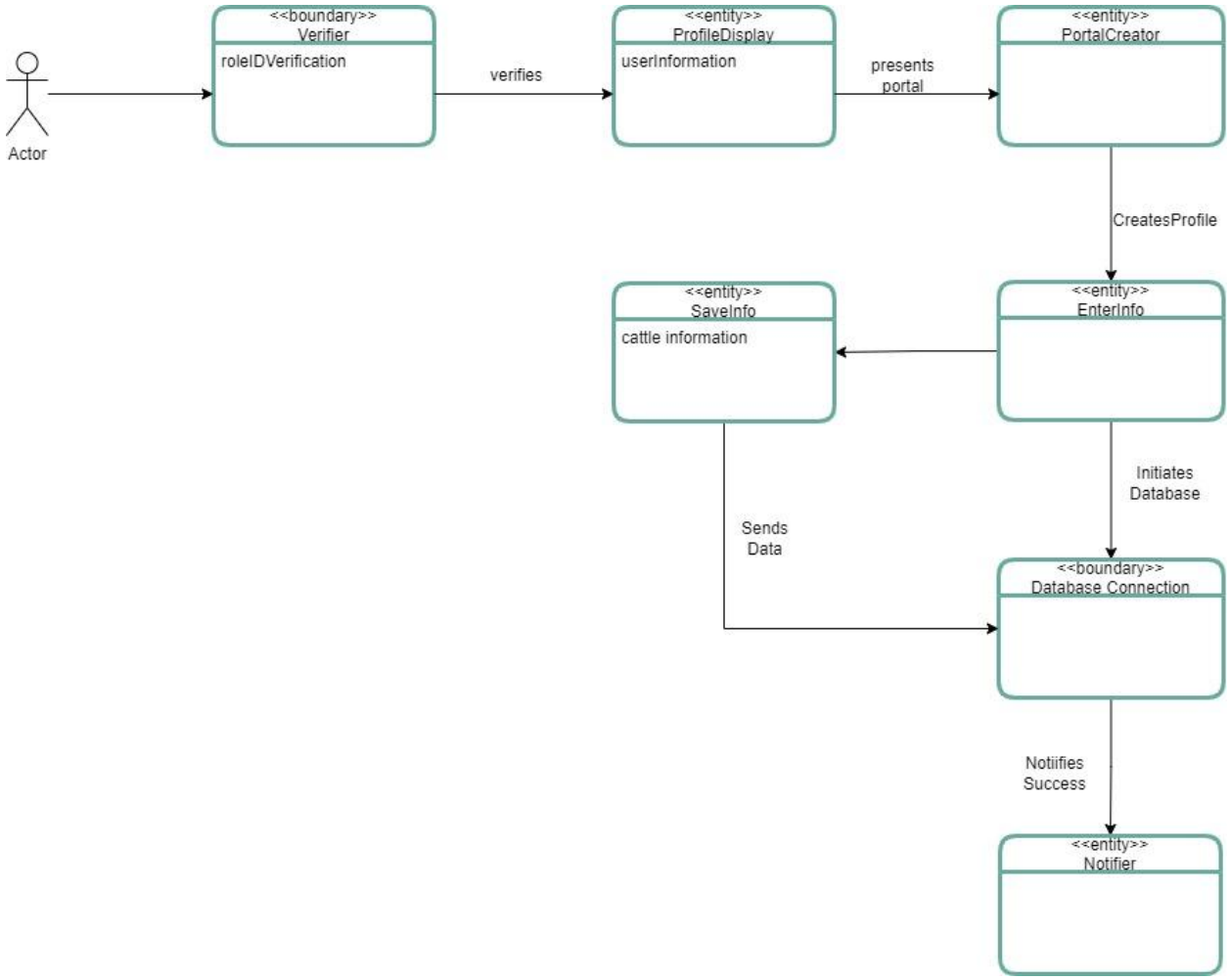
Concept	Attributes	Attribute Description
Verifier	User ID UserRole	Used to identify the Actor's identity, which will determine what operations that can be performed in turn Determines the Actor's role
Filter Search	Filter Request	Used to Determine what information to filter out
Search Request	Search Criteria	Needed to Prepare a Database Query to return a result that matches the Actor's based on the filter provided.
DatabaseRetrieve	ClassID	Used to Identify the cattle information for the specific class
SelectID	Cattle ID	Needed to extract information from the Database in order to match the Cattle ID that was entered, which is then displayed on the screen.

Attribute Definition for AddSlaughterInformation(UC8)

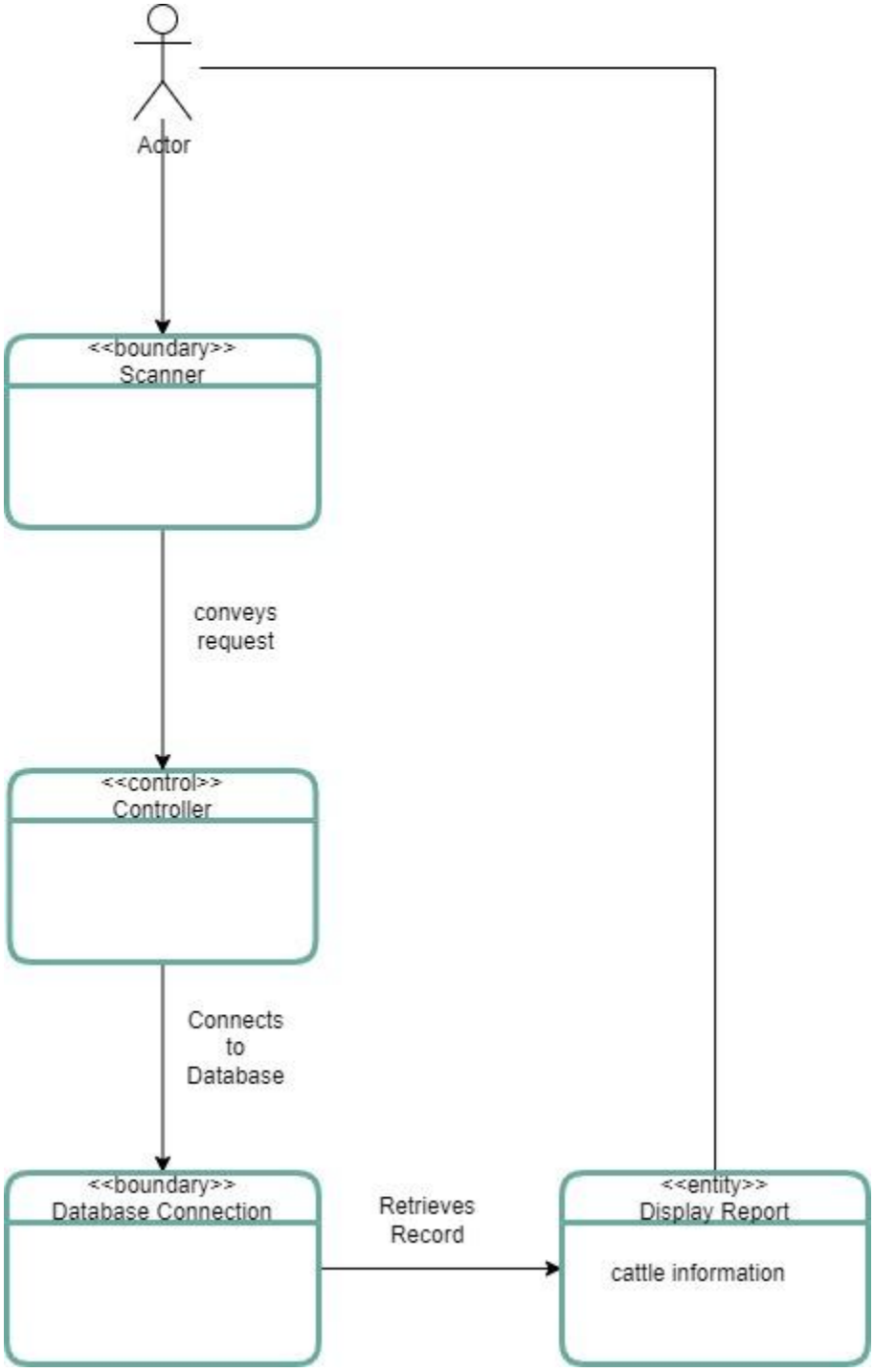
Concept	Attributes	Attribute Description
Verifier	User ID UserRole	Used to identify the Actor's identity, which will determine what operations that can be performed in turn Determines the Actor's role
Search Request	Search Criteria	Needed to Prepare a Database Query to return a result that matches the Actor's search.
AddInfo	CattleInfo	Used to Add information about a specific Cattle into the Database. Can be broken down into more Attributes such as "cut of meat", "breed" and etc.
SaveInfo	ClassID	Used to update the cattle Data in the Database after new information has been added.

Domain Models

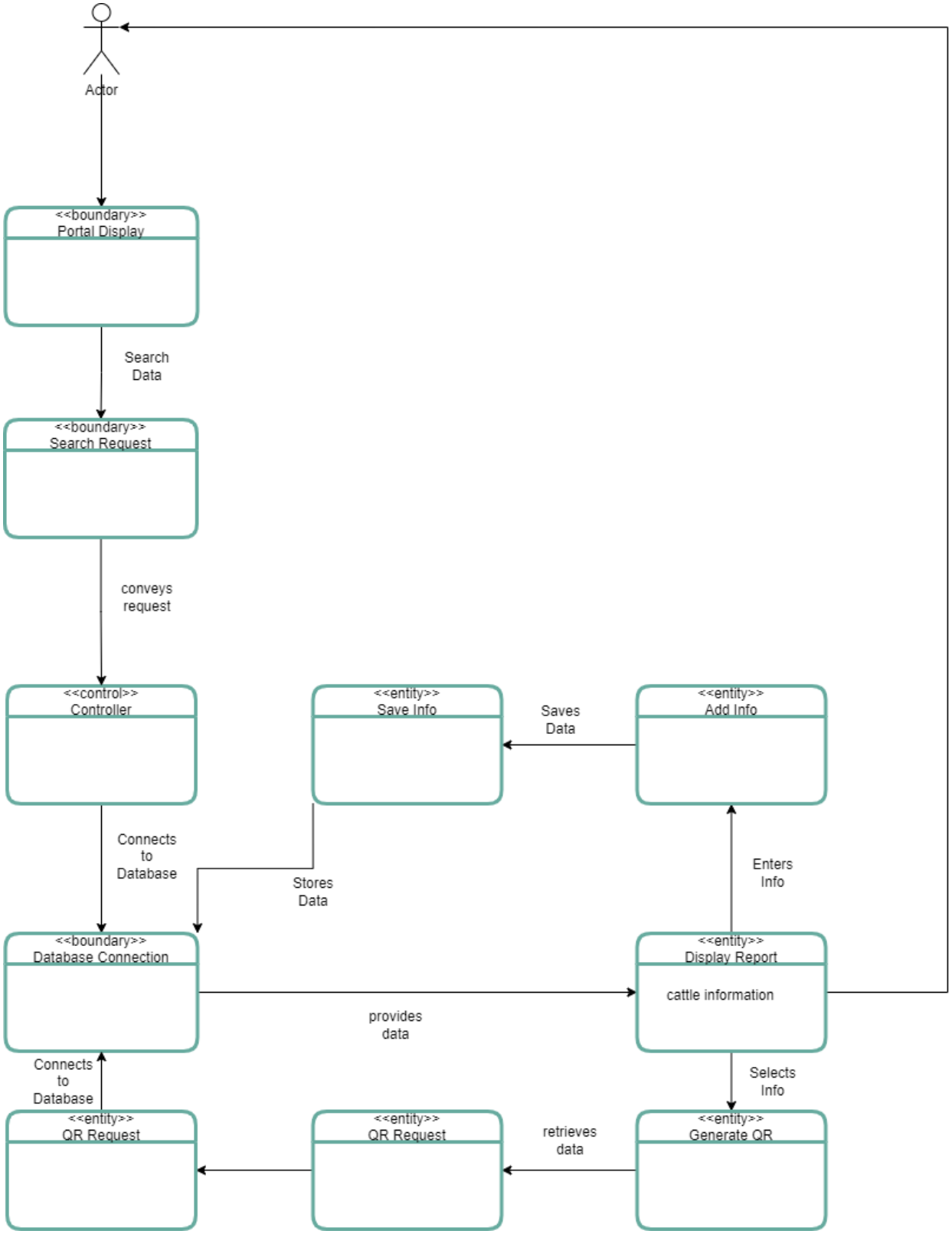
Domain Model of UC-1 CreateCattleProfile



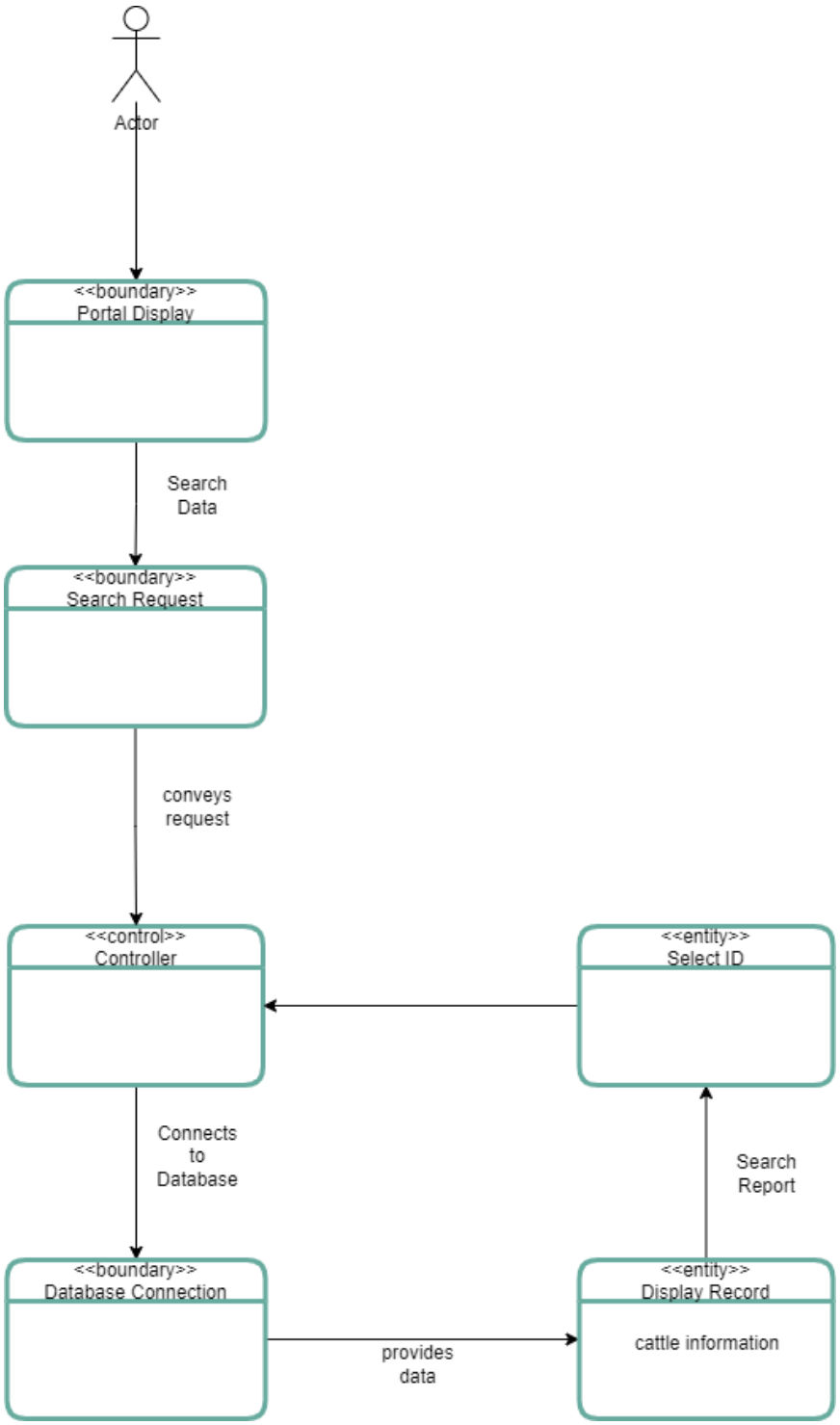
Domain Model of UC-2 viewProductDetails



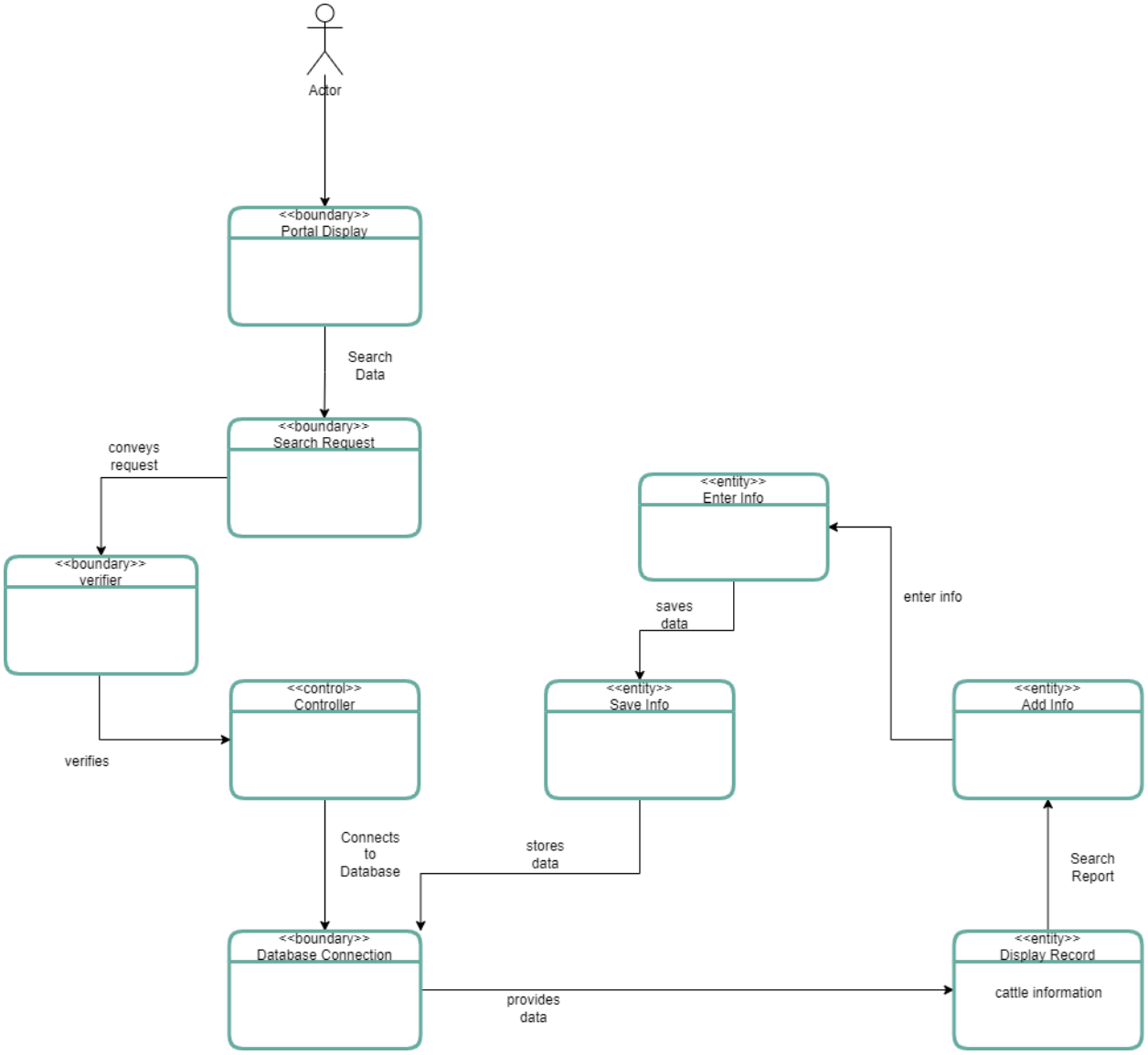
Domain Model of UC-5 addProductInformation



Domain Model of UC-7 generateReport



Domain Model of UC-8 addSlaughterInformation



(IV) Traceability Matrix

Domain Concepts	PW	Controller	Verifier	Portal Display	Profile Creator	EnterInfo	SaveInfo	DatabaseConnection	Notifier	Scanner	QR Code	DisplayReport	SearchRequest	DatabaseRetreive	AddInfo	DatabaseStore	GenerateQR	QRRequest	QR Data Retrieve	DisplayQR	ReportGenerator	SearchFormDisplay	Filter Search	SearchRequest	CriteriaSearch	Select ID	DisplayRecords
UC1	9	x	x	x	x	x	x	x	x						x	x												
UC2	11	x						x		x	x	x		x			x	x	x	x								x
UC3	3	x												x	x	x												
UC4	6	x											x	x								x		x		x	x	
UC5	8	x	x	x		x	x	x	x				x	x	x	x	x	x	x			x		x		x		
UC6	2	x								x		x					x	x	x	x								x
UC7	9	x	x	x		x	x	x			x	x	x	x	x		x	x	x		x	x	x	x	x	x	x	x
UC8	11	x	x	x		x	x	x	x	x	x	x	x	x	x	x						x		x		x		
UC9	7	x				x	x	x					x	x		x			x			x		x		x		

Each priority use case corresponds to ideas that are used to ensure that the use case performs what it is supposed to do. The Database Connection is used in almost every use case since it is critical to the overall application's operation. The reason for this is that most of our tasks include the system retrieving and storing data in a database. Because the search concept allows a user to access a specific cow profile through its cattleID, it is critical for those use cases that need information to be updated to a cattle profile. In certain circumstances, the notifier is also used to provide success or completion notifications to the user (for example, when adding information to a livestock profile).

System Operation Contracts

System Contracts for CreateCattleProfile

Contract Name	verifyInfo
Operation	**VerifyInfo(string info)**
Cross Reference	**Use Cases: Create Cattle Profile**
Responsibilities	**Verify that user credentials is valid**
Type	**System**
Exceptions	**Credentials were entered incorrectly**
Pre-Conditions	**Information needs to be entered.**

Post-Conditions	**Information is either valid or Invalid, If it is Valid, then user gains access to the system, else the user is prompted to re-enter credentials**

Contract Name	createInfo
Operation	**createInfo(string info)**
Cross Reference	**Use Cases: Create Cattle Profile**
Responsibilities	**Creates a New Cattle profile which is then added to the database**
Type	**System**
Exceptions	**No information is entered or The wrong type of information is entered in the wrong field.**
Pre-Conditions	**Information needs to be entered to be saved**
Post-Conditions	**Information has been received and successfully added to the Database and the Actor is notified**

System Contracts for ViewProductDetails

Contract Name	recordScan
Operation	**recordScan(scan: QRCode)**
Cross Reference	**Use Cases: ViewProductDetails**
Responsibilities	**Parses the QRCode so that the information can be retrieved from the database and displayed**
Type	**System**
Exceptions	**None**
Pre-Conditions	**Scanner is readily accessible because the user has the camera app open.**
Post-Conditions	**QRCode information is successfully parsed and displayed to the user.**

Contract Name	receiveQRInfo()

Operation	**receiveQRInfo()**
Cross Reference	**Use Cases: ViewProductDetails**
Responsibilities	**Receives QRCode information from the scanner**
Type	**System**
Exceptions	**Unsuccessful Scan (User scans QRCode Improperly, eg: Only captures half of the QR Code)**
Pre-Conditions	**User has already scanned QRCode**
Post-Conditions	**Information has been received and verified**

System Contracts for AddProductInformation

Contract Name	displayRecords
Operation	**displayRecords()**
Cross Reference	**Use Cases: Add Product Information**
Responsibilities	**Display Information that matches the Actor's Search Criteria**
Type	**System**
Exceptions	**No information has been Entered/ Enters Criteria that doesn't exist in the Database**
Pre-Conditions	**Information has been entered.**
Post-Conditions	**Information has been validated and displayed for the Actor to view.**

Contract Name	notifyUser
Operation	**NotifyUser()**
Cross Reference	**Use Cases: Add Product Information**
Responsibilities	**Notifies the User that the information has been successfully added**

Type	**System**
Exceptions	**Incorrect Information entered (Enters a numerical value in a field that requires text values)**
Pre-Conditions	**User has clicked the Add button**
Post-Conditions	**Information has been validated and successfully inserted**

System Contracts for GenerateReport

Contract Name	filterResult
Operation	**filterResult()**
Cross Reference	**Use Case: GenerateReport**
Responsibilities	**Filters the information stored in the database based on the category provided by the Actor.**
Type	**System**
Exceptions	**Invalid Filter Category Provided**
Pre-Conditions	**Filter category has been selected**
Post-Conditions	**Information has been extracted from the database and is displayed to the screen**

Contract Name	displayReport
Operation	**displayReport()**
Cross Reference	**Use Case: GenerateReport**
Responsibilities	**Displays the retrieved profile/report to the actor's web browser for display**
Type	**System**
Exceptions	**N/A**
Pre-Conditions	**Cattle ID has been selected**

Post-Conditions	**Cattle ID was validated and the accurate information is displayed for the user to view**

System Contracts for AddSlaughterInformation

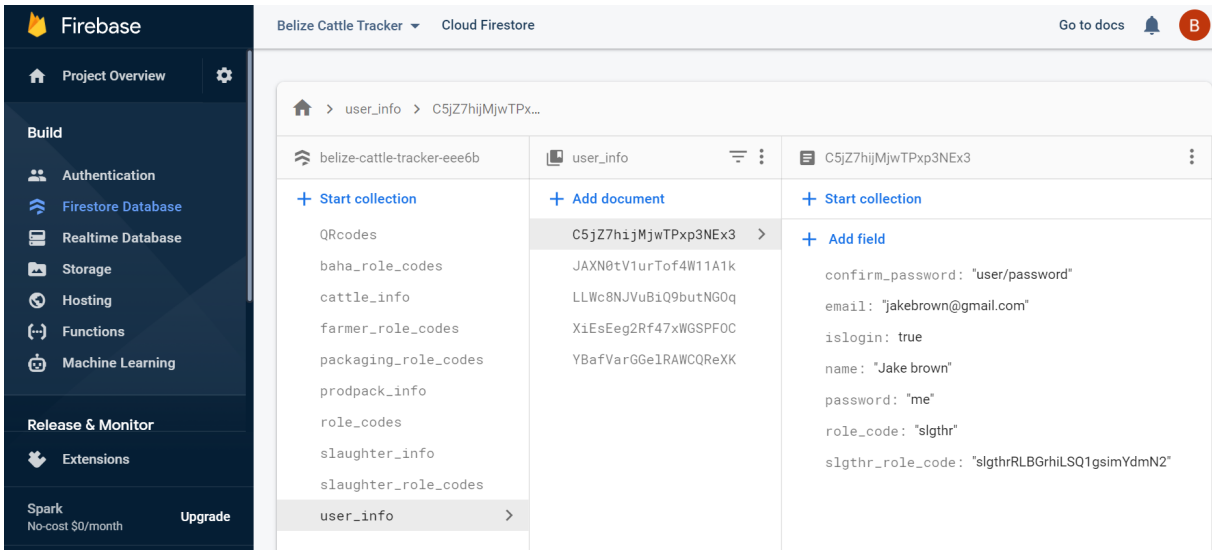
Contract Name	retrieveInfo
Operation	**retrieveInfo()**
Cross Reference	**Use Cases: AddSlaughterInformation**
Responsibilities	**Retrieves the accurate information that matches the user's input from the database.**
Type	**System**
Exceptions	**Invalid search criteria provided**
Pre-Conditions	**Search criteria needs to be inputted to the form**
Post-Conditions	**Once the criteria has been validated and the appropriate data has been extracted, it is displayed on screen.**

Contract Name	updateInfo
Operation	**updateInfo()**
Cross Reference	**Use Cases: AddSlaughterInformation**
Responsibilities	**Updates the database based on the information provided by the Actor**
Type	**System**
Exceptions	**No information is given**
Pre-Conditions	**createInfo from Use Case CreateCattleProfile needs to have been executed**
Post-Conditions	**The database is updated and the information is successfully added. The Actor is notified that the information has been added.**

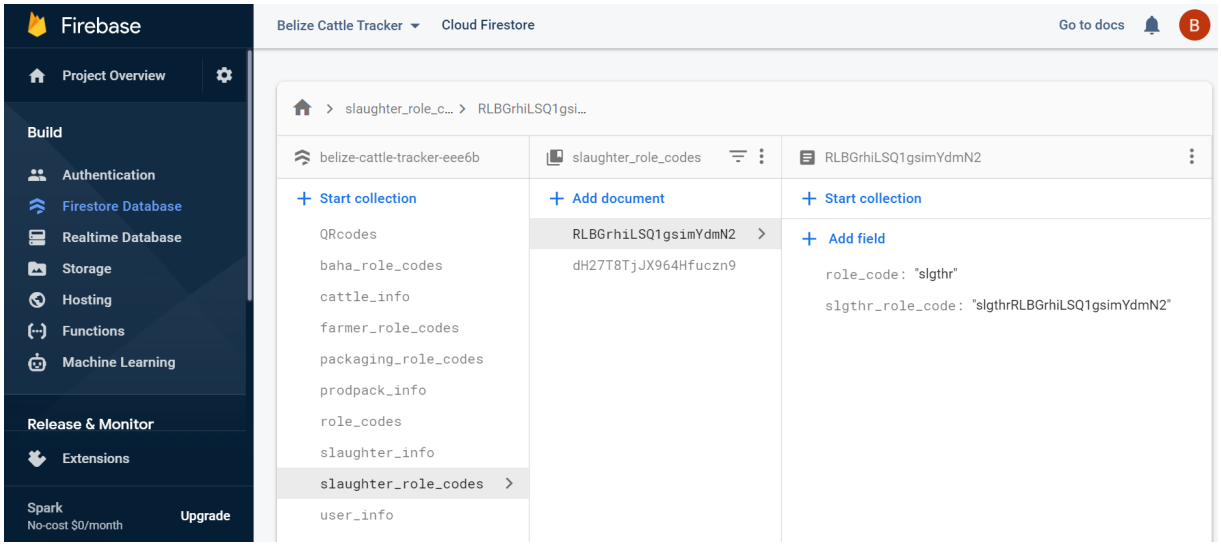
Data Model and Persistent Data Storage

The Belize Cattle Tracker consists of various scenarios where data has to be retrieved, passed, and displayed. These retrieved and stored data must outlive a single execution of the system as new data will be added and updated to previously available data, because of this, the data will be stored in a cloud-hosted database (Firebase). Since it is a noSQL database, it allows for data to be stored and synced between different fields in real time. Being a non relational database, SQL queries such as JOINS cannot be used to link tables together so document references and api links have to be included in the code of the function; though it can sometimes be linked through similar fields in the different tables (Delaney, 2018). Moreover, in regards to Firebase, a collection would be considered a table in a relational database, a document would be considered a table attribute and a field would be considered a table row/tuple. (Additionally, Firebase is also a schema-less database. Below is our Firebase Setup. Collections are linked through the "cattleid" field.

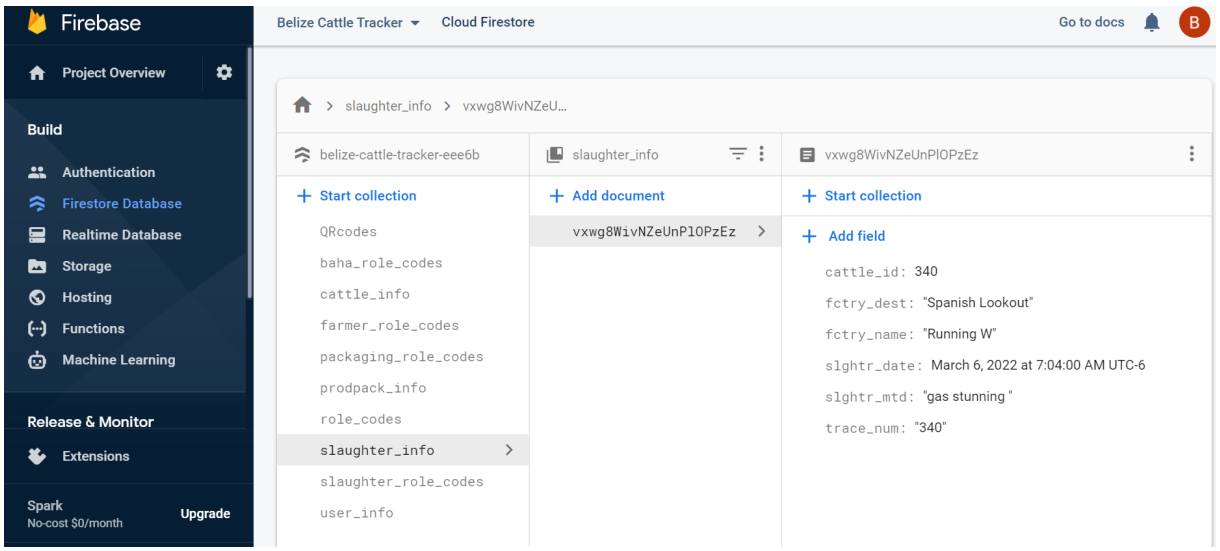
"Schema" for user_info document to store login information.



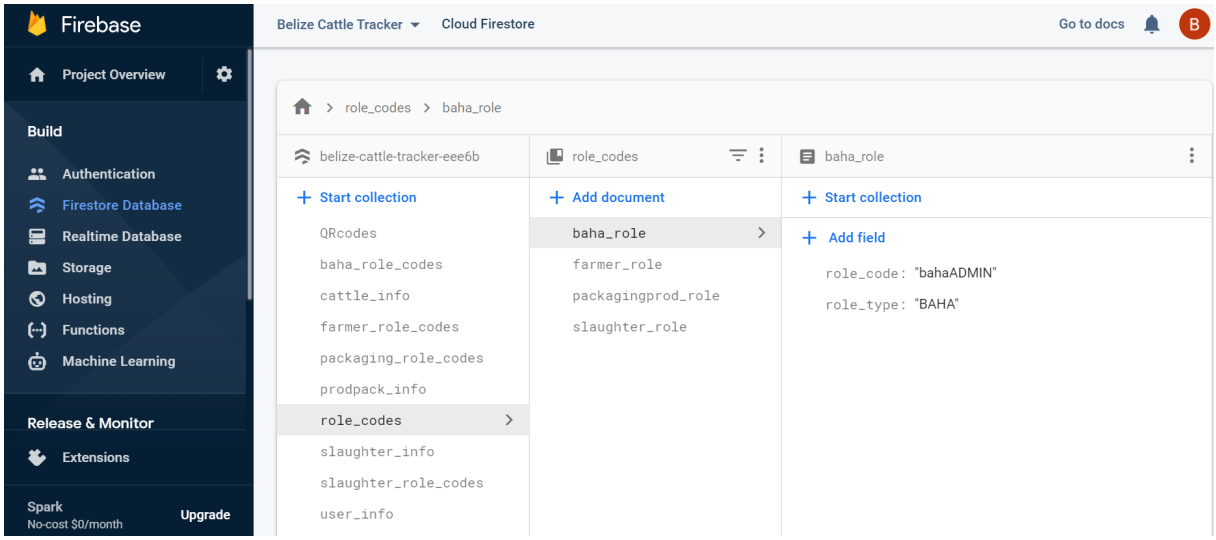
"Schema" for slaughter_role_codes document to store unique role codes for different slaughterhouse managers who have signed up.



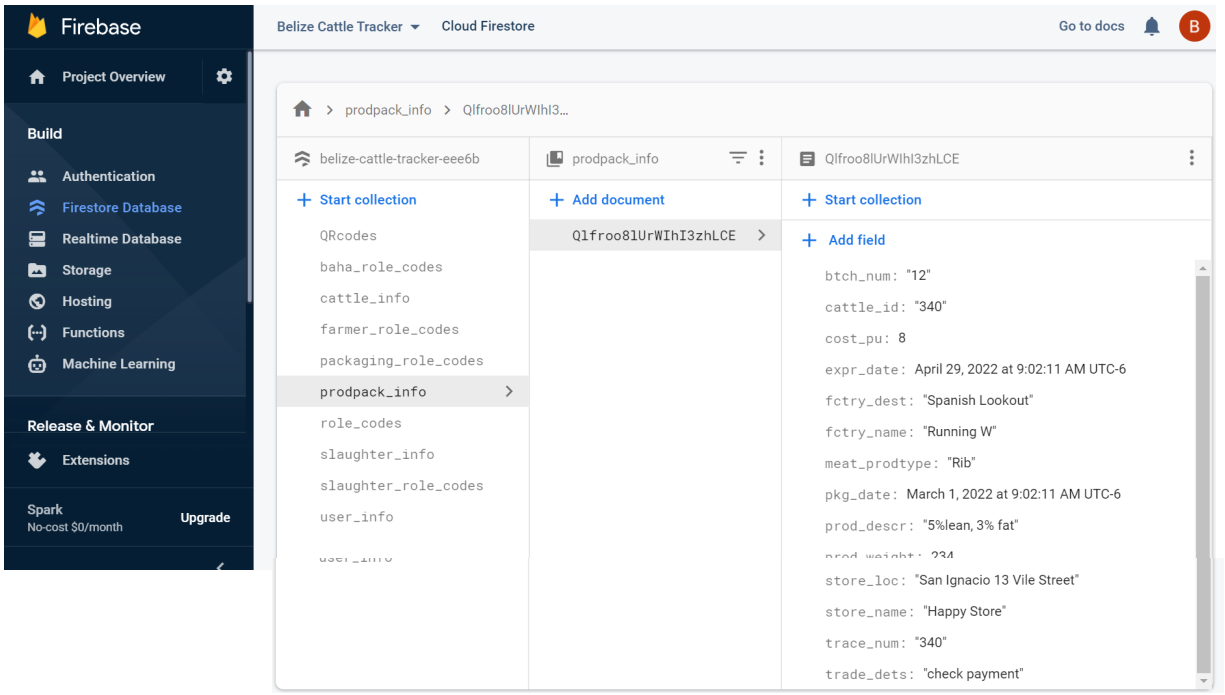
"Schema" for slaughter_info document to store slaughter info that was added on to the cattle info.



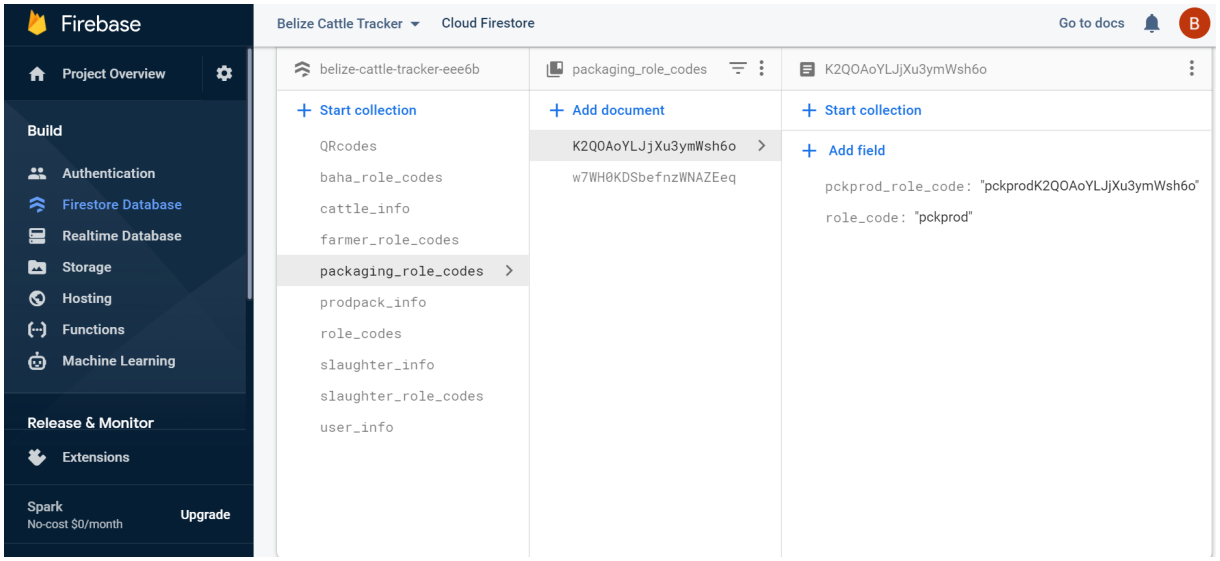
"Schema" for role_codes document to store role types and their code.



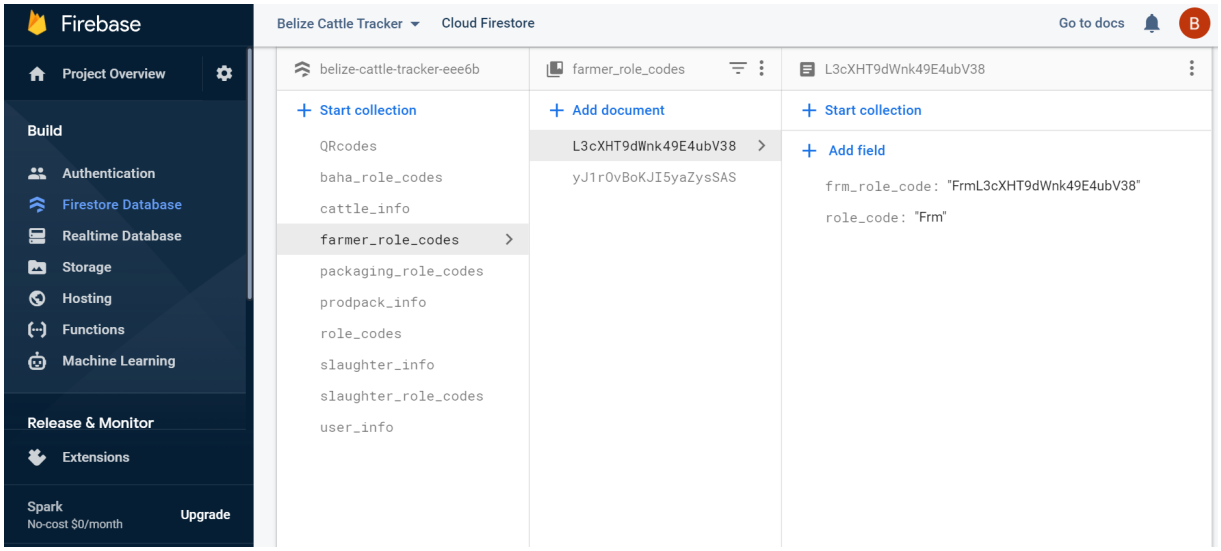
"Schema" for prodpack_info document to store the product info that was added on to the cattle and slaughter info.



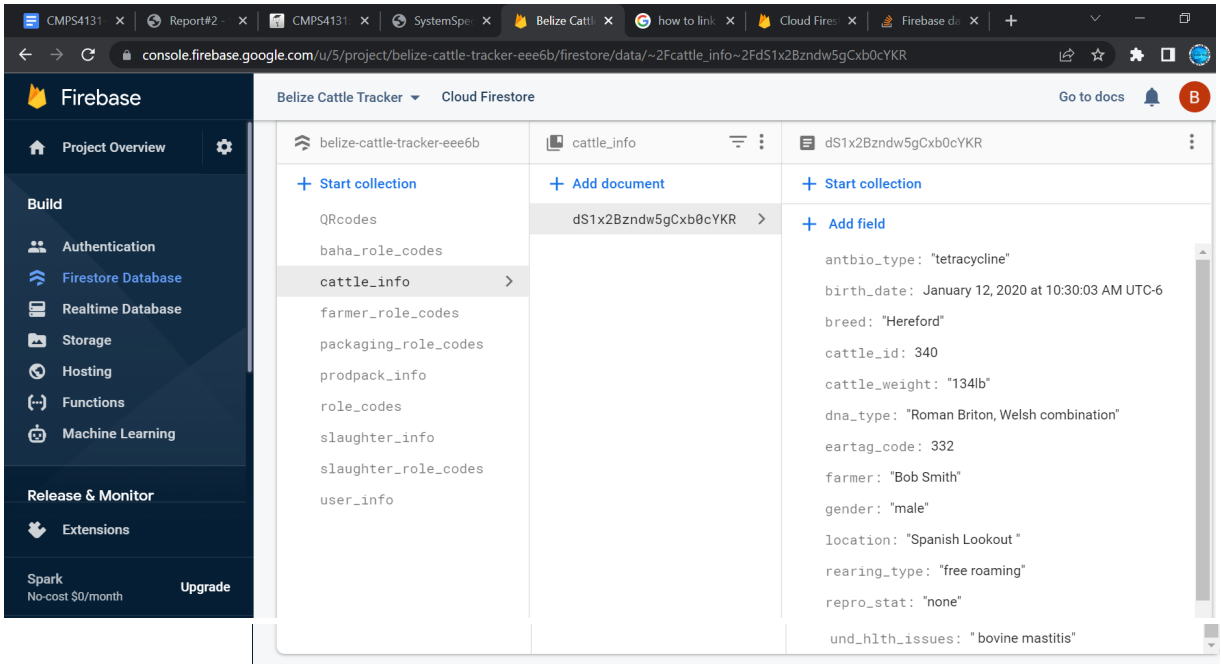
“Schema” for packaging_role_codes document to store the unique role codes for different packaging managers who have signed up.



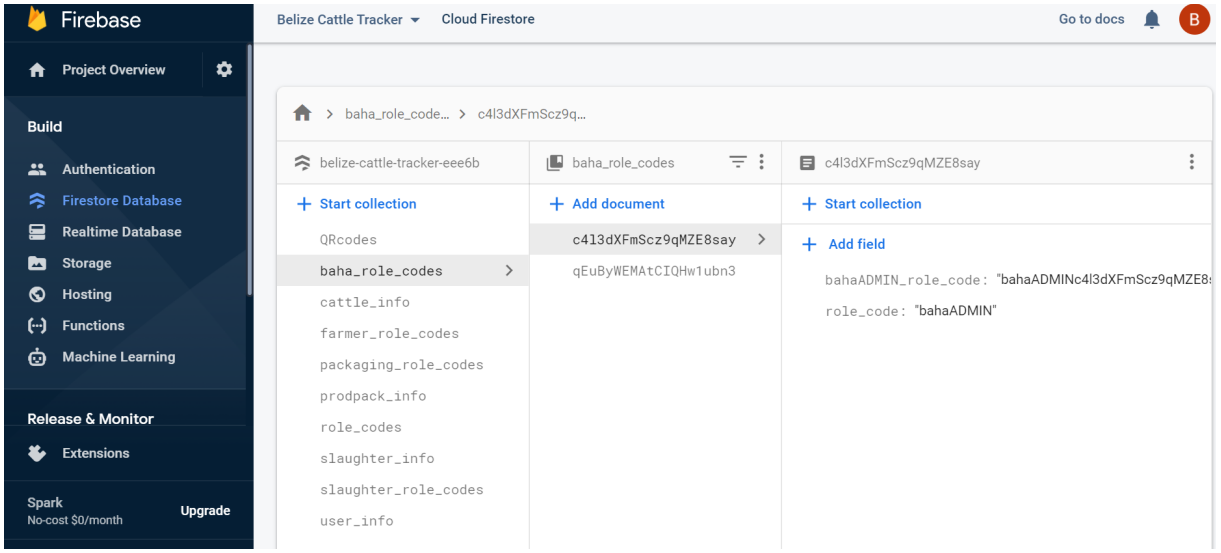
“Schema” for farmer_role_codes document to store the unique role codes for different farmers who have signed up.



“Schema” for cattle_info document to store the information that was added by the farmer in order to create or update a cattle profile.

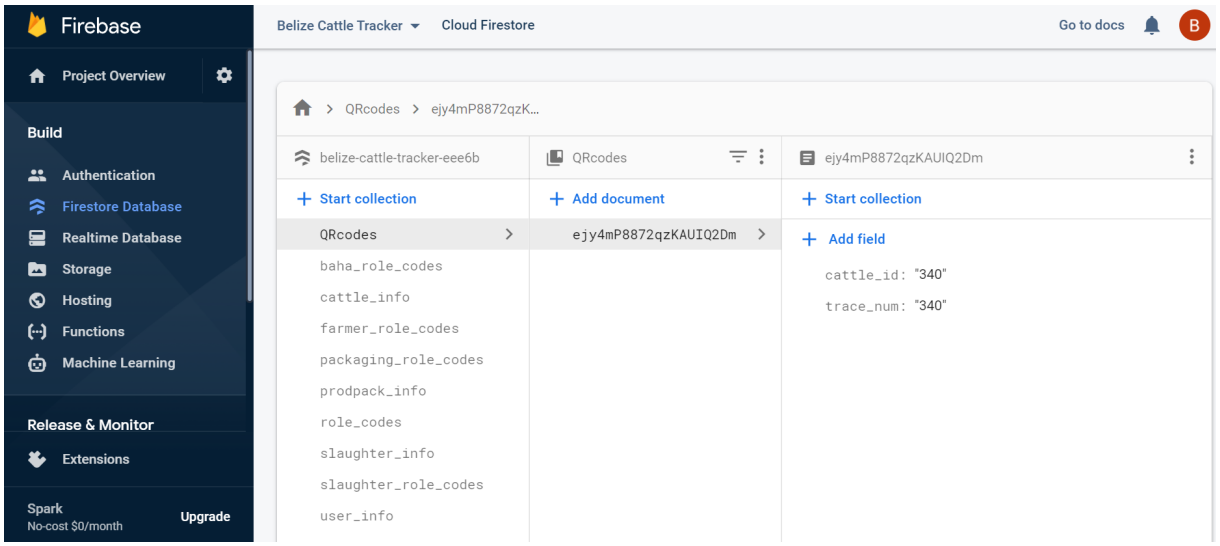


"Schema" for baha_role_codes document to store the unique role codes for the BAHA admins.

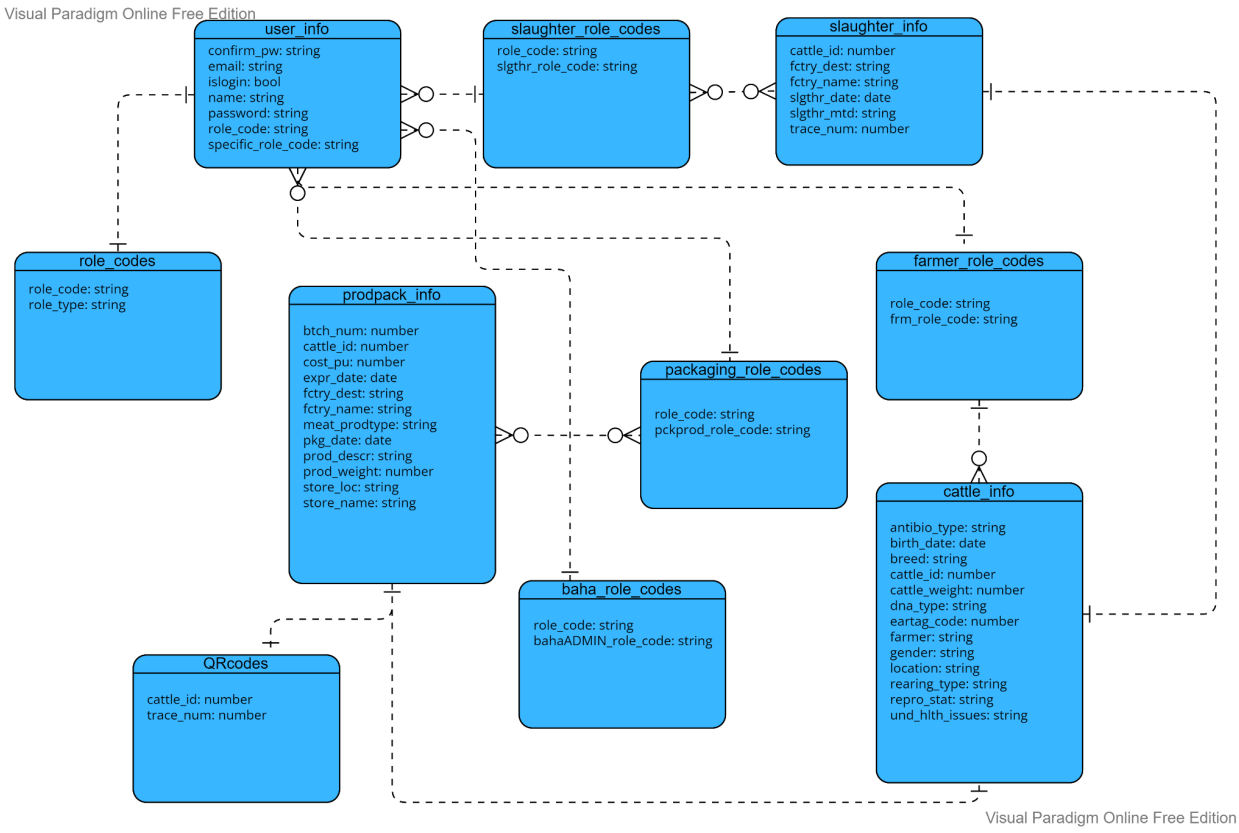


"Schema" for QRcodes document to store the uniquecattleid and tracenum for a specific cattle. This table will be used to make the collectionGroup function (in React code) easier to pull

information for a specified trace number. Eg: packaging. cattle, and slaughter info that has the same tracenum value will be pulled and linked to a generated QR code.



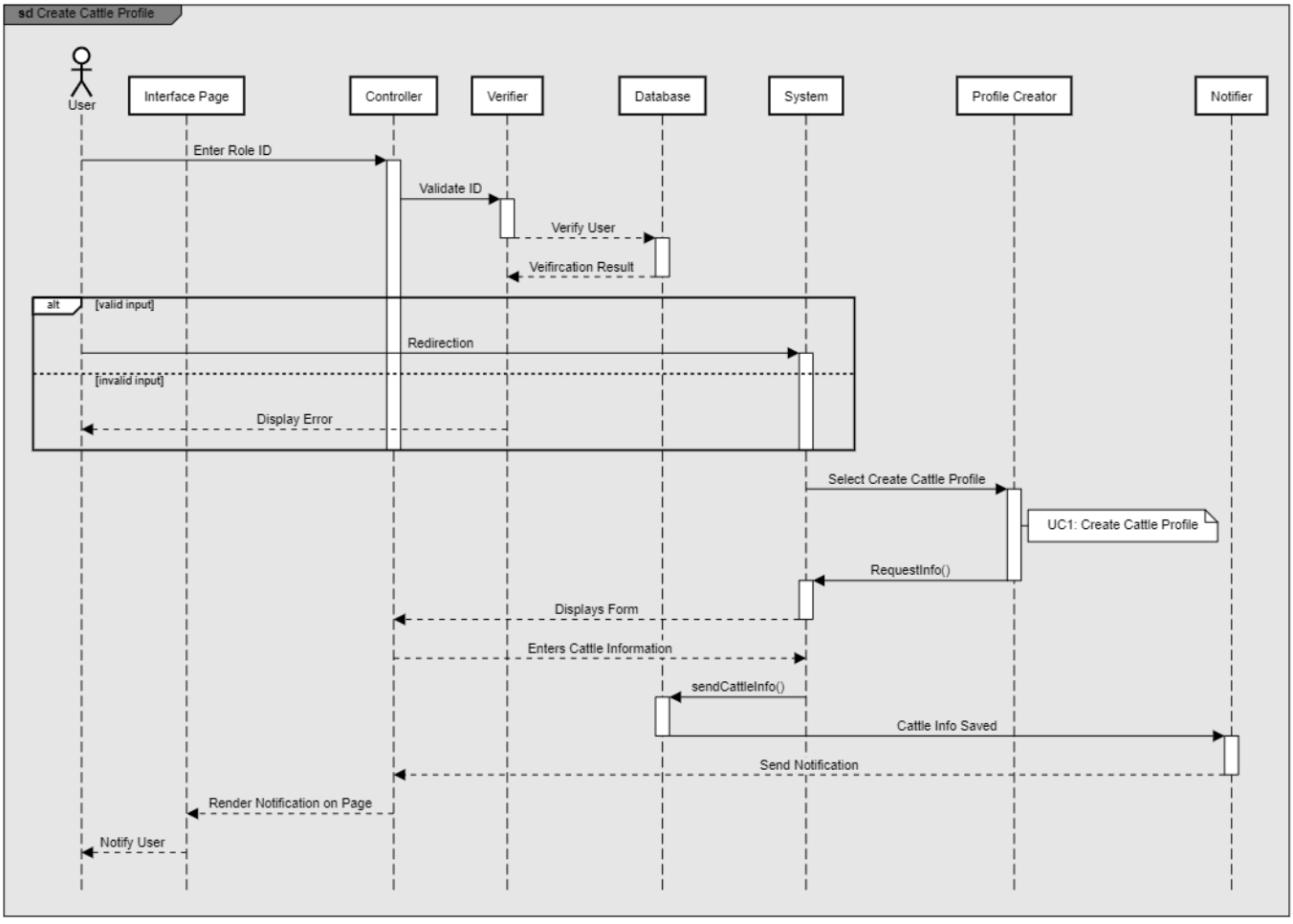
Below is a clear representation of what the **database schema** works/looks like with the assumption that the Firestore Database is in terms of a relational SQL based database.



Interaction Diagrams

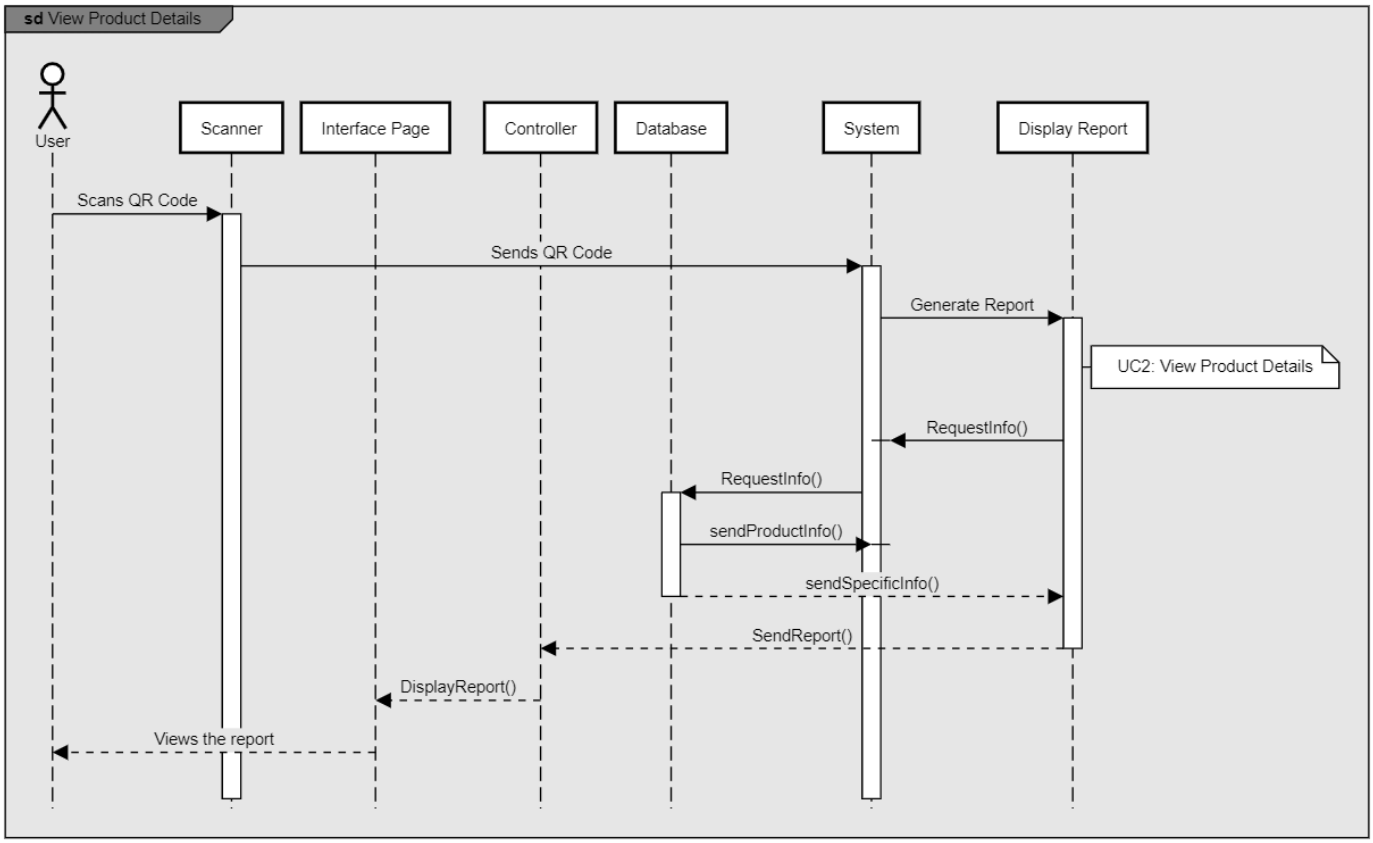
Sequence Diagram for UC 1 - Create Cattle Profile

Create Cattle Profile is responsible for creating and notifying a farmer of a successful creation of a cattle profile. A farmer can enter their ID to verify their role and specify their portal. The CONTROLLER then sends the farmer role ID to the VERIFIER to confirm the farmer ID from the database. When the farmer ID is retrieved from the database, the system will be notified and the FARMERPORTAL will be displayed. From there, the farmer can select CREATECATTLEPROFILE which would initiate the system's PROFILECREATOR to display the CATTLEBIRTHINFOFORM. The farmer can then enter and save the CATTLEBIRTHINFO which would push the data to the DATABASE. After the information is pushed to the database, the farmer receives a successful confirmation message from the NOTIFIER, letting them know that the profile was successfully created. The concept(s) (User Interface, Database, Product Profile) evolved into (Interface Page, Controller, Verifier, Database, System, Profile Creator and Notfier) respectively.



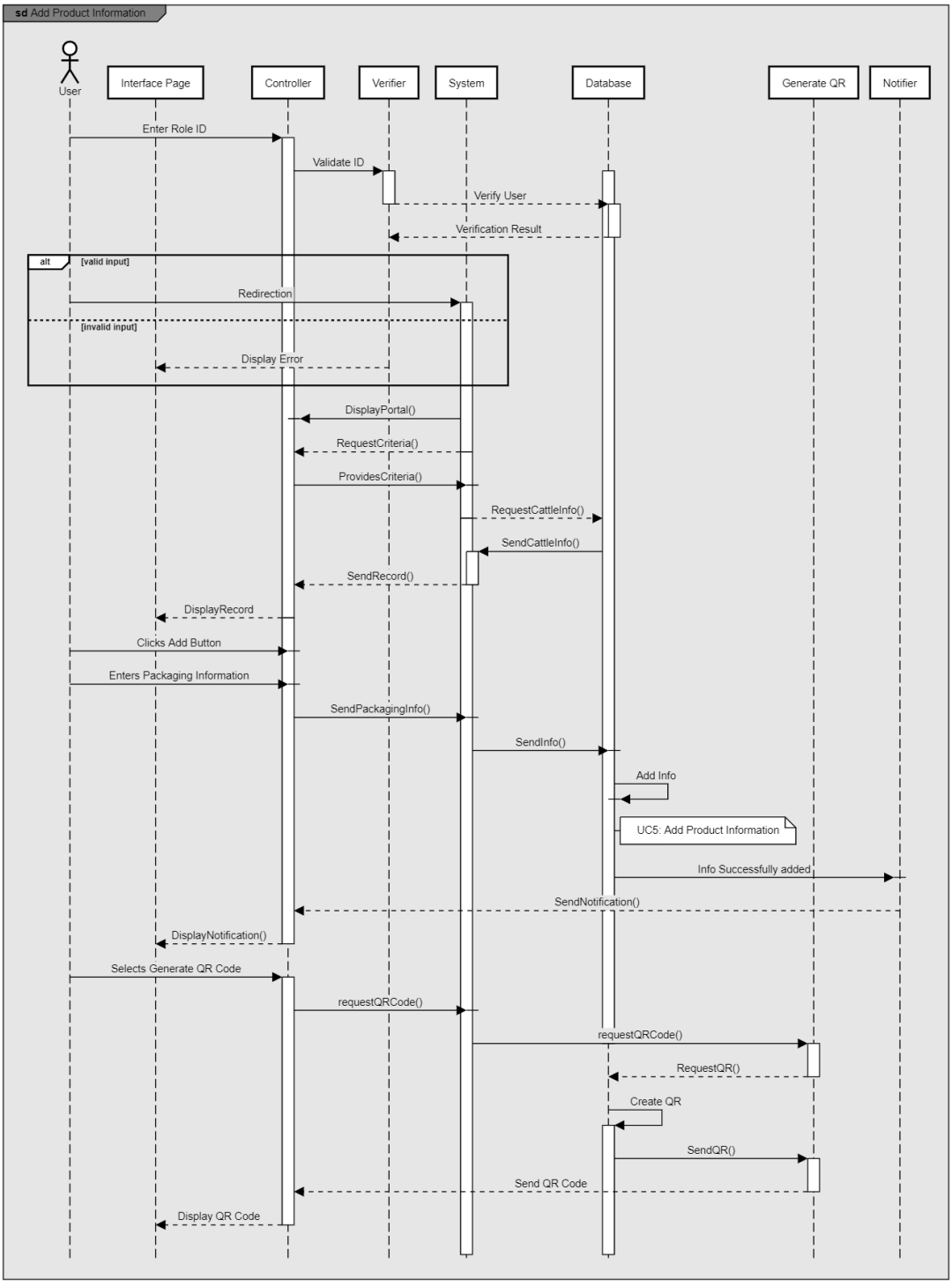
Sequence Diagram for UC 2 - View Product Details

View Product Details is responsible for displaying the product details file to the customer when the QR code is scanned. A customer can use any form of smart device medium that has a camera with the ability to scan QR codes. When the customer scans the QR code, the system gets prompted to GENRATEREPORT. From there, the system will invoke DISPLAYREPORT, which will query the DATABASE for the required information. After retrieving the data from the DATABASE, the DISPLAYREPORT delivers the file containing the necessary records to the CONTROLLER, allowing the INTERFACEPAGE to DISPLAYREPORT. Finally, the consumer may access the PRODUCTDETAILSREPORT via the INTERFACEPAGE. The concept(s) (User Interface, Database, Product Profile) evolved into (Scanner, Interface Page, Controller, Database, System, and Displasy Report) respectively.



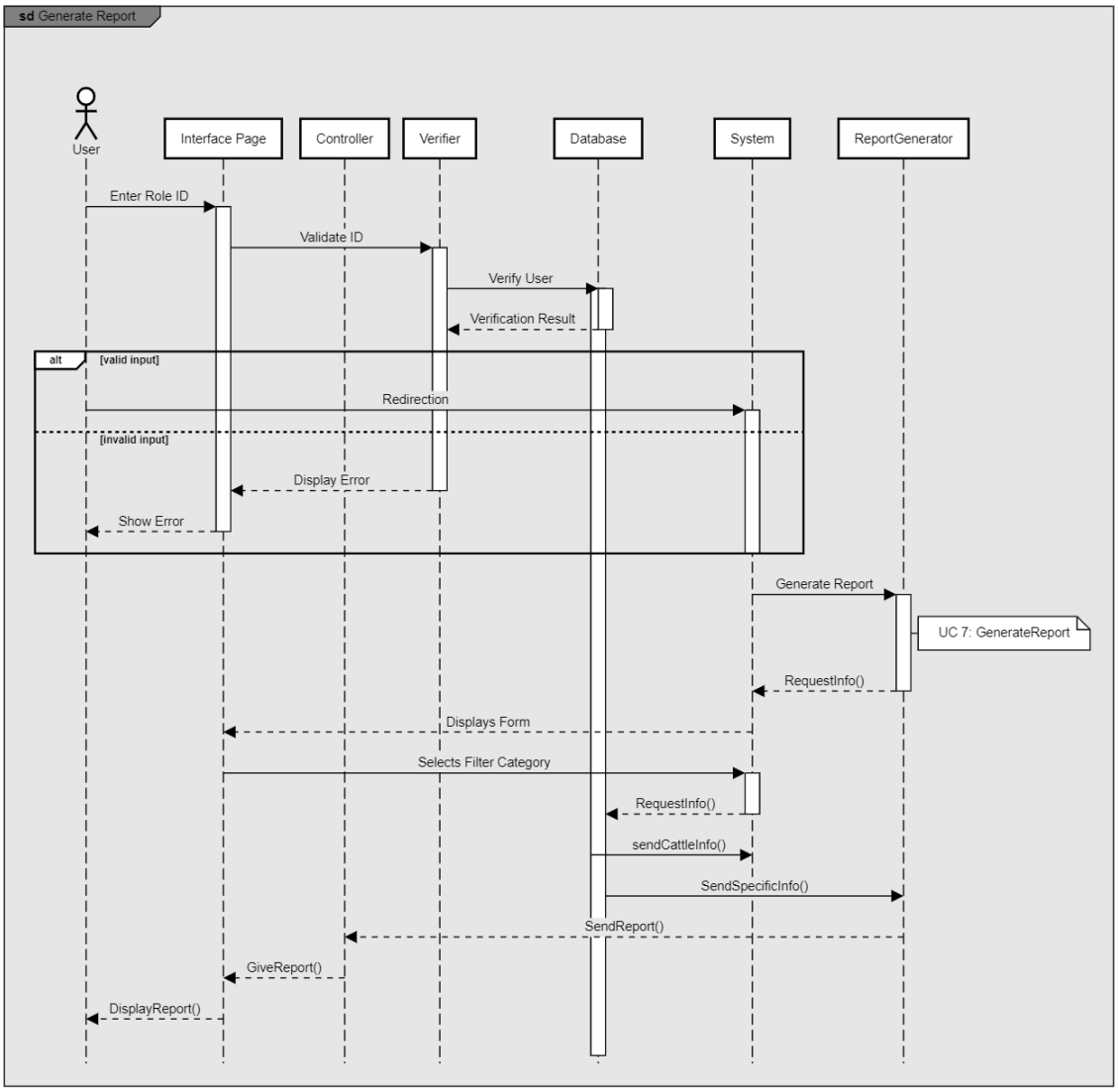
Sequence Diagram for UC 5 - Add Product Information

Add Product Information is responsible for adding and notifying a package manager of a successful addition of packaging information to a cattle profile. A packaging manager can enter their ID to verify their role and specify their portal. The CONTROLLER then sends the packaging manager role ID to the VERIFIER to confirm the packaging manager's ID from the database. When the packaging manager ID is retrieved from the database, the system will be notified and the PACKAGING PORTAL will be displayed. From there, the packaging manager can search for a specific cattle ID which will enable the SYSTEM to request for the cattle information from the DATABASE. The DATABASE will send the requested information to the SYSTEM, which will send the information to the INTERFACEPAGE where it will be displayed to the USER. The packaging manager can then add packaging information to that cattle profile and save the PACKAGINGINFO which would push the data to the DATABASE. After the information is pushed to the database, the packaging manager receives a successful confirmation message from the NOTIFIER, letting them know that the information was successfully added. The packaging manager can then select to generate a QR code, this would prompt the SYSTEM to REQUESTQRCODE. The SYSTEM will invoke GENERATEQR to request a QR code from the DATABASE. The DATABASE will create the QR code and send it to GENERATE QR which will send the QR code to the CONTROLLER. The CONTROLLER will send it to the INTERFACEPAGE where it will be displayed to the packaging manager/USER. The concept(s) (User Interface, Database, Product Profile) evolved into (Interface Page, Controller, Verifier, System, Database, Generate QR and Notfier) respectively.


sd Add Product Information
User
Interface Page
Controller
Verifier
System
Database
Generate QR
Notifier
Enter Role ID
Validate ID
Verify User
Verification Result
alt
[valid input]
Redirection
[invalid input]
Display Error
DisplayPortal()
RequestCriteria()
ProvidesCriteria()
RequestCattleInfo()
SendCattleInfo()
SendRecord()
DisplayRecord
Clicks Add Button
Enters Packaging Information
SendPackagingInfo()
SendInfo()
Add Info
UC5: Add Product Information
Info Successfully added
SendNotification()
DisplayNotification()
Selects Generate QR Code
requestQRCode()
requestQRCode()
RequestQR()
Create QR
SendQR()
Send QR Code
Display QR Code

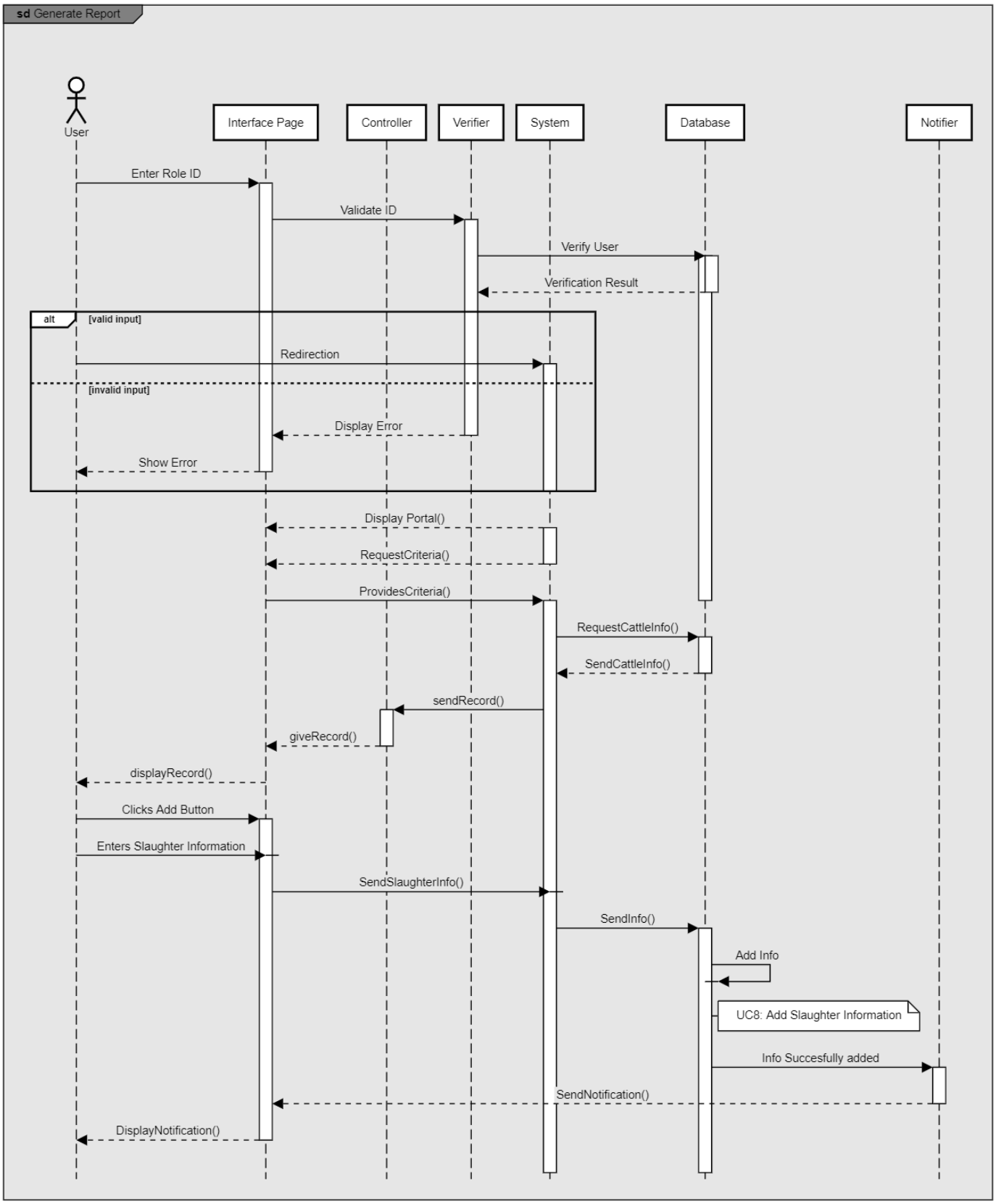
Sequence Diagram for UC 7 - Generate Report

Generate Report is responsible for generating a report of an existing cattle profile. A BAHA Admin can enter their ID to verify their role and specify their portal. The CONTROLLER then sends the BAHA admin role ID to the VERIFIER to confirm the admin ID from the database. When the BAHA admin ID is retrieved from the database, the system will be notified and the BAHAADMIN PORTAL will be displayed. From there, the admin can select GENERATEREPORTS which would initiate the system's REPORTGENERATOR to display the GENERATEREPORTFORM. The USER will select the search criteria to find a cattle profile. This will prompt the system to request the information from the DATABASE. The DATABASE sends the specific cattle information to the REPORTGENERATOR, which will send the report to the CONTROLLER. The CONTROLLER will give the report to the INTERFACEPAGE where it will be displayed to the BAHA Admin/USER. The concept(s) (User Interface, Database, Cattle Profile) evolved into (Interface Page, Controller, Verifier, Database, System, Report Generator) respectively.



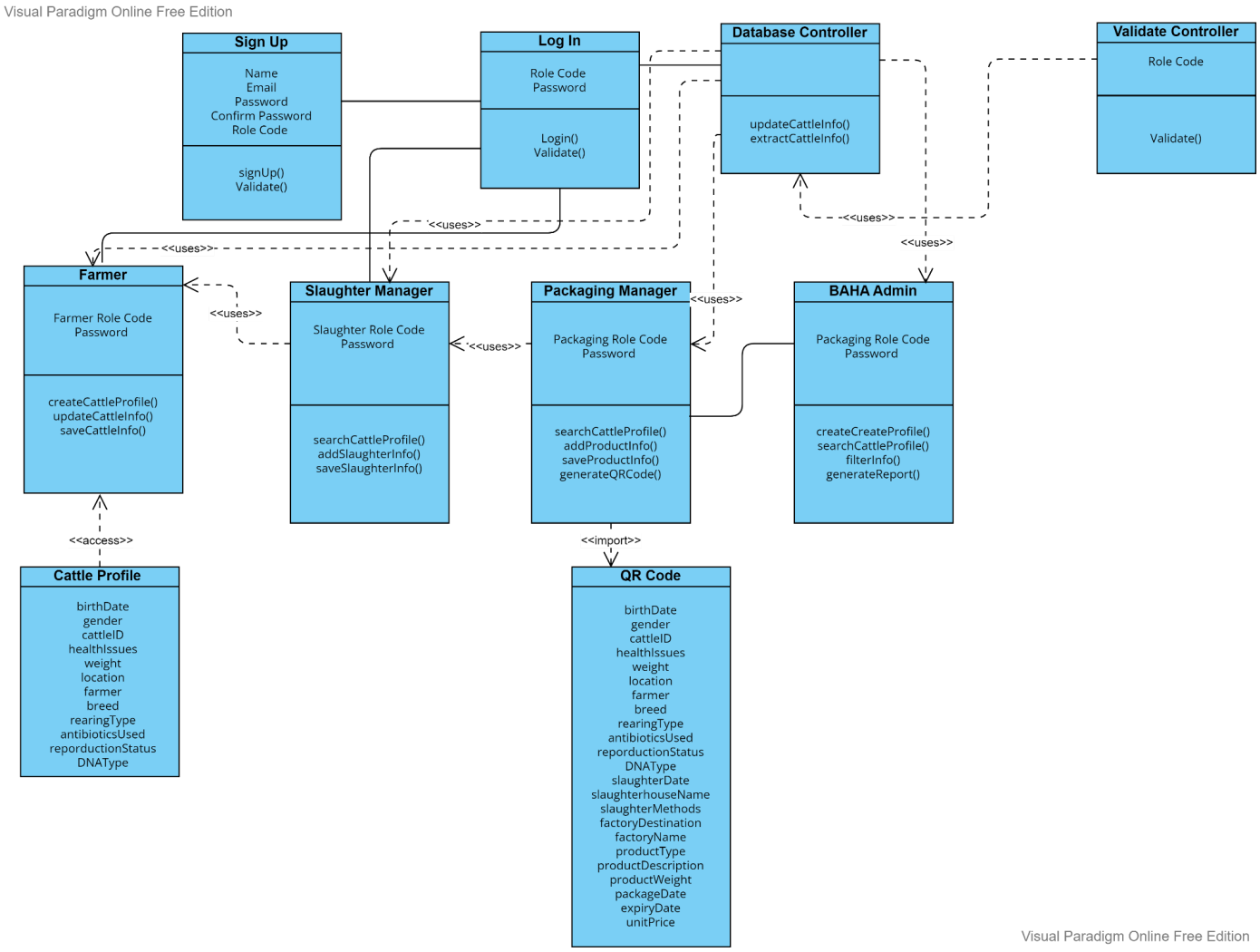
Sequence Diagram for UC 8: Add Slaughter Information

Add Slaughter information allows a slaughterhouse manager to add slaughter information to previously uploaded CATTLEBIRTHINFO and to provide a completion message when the information is successfully entered. A slaughterhouse manager can input their ID to confirm their role and choose their portal. The slaughter role ID is subsequently sent to the VERIFIER, who confirms the slaughter manager ID from the DATABASE. The SYSTEM will be informed and the SLAUGHTERHOUSEPORTAL will be displayed when the slaughter manager ID is obtained from the database. The slaughterhouse manager may then search for a cattle profile, which will cause the system to request a criterion (specific cattleID). When a SEARCHREQUEST is issued, it offers the criteria to the SYSTEM, which will obtain the necessary information from the DATABASE associated with the cattleID. The DATABASE then returns the cattle information to the SYSTEM, which then transmits the records to the CONTROLLER. After that, the CONTROLLER will DISPLAYCATTLEINFO on the INTERFACEPAGE where the slaughterhouse manager can press the ADDBUTTON and input SLAUGHTERINFO. After entering this, the SLAUGHTERINFO is stored to the SYSTEM and DATABASE (during this process UC8 is initiated). The NOTIFIER will be notified after the SLAUGHTERINFO has been successfully uploaded to the DATABASE. The slaughterhouse manager receives a successful confirmation message from the NOTIFIER on the INTERFACEPAGE, indicating that the process had completed successfully. The concept(s) (User Interface, Database, Product Profile) developed into (Interface Page, Controller, Verifier, Database, System, and Notifier) respectively.

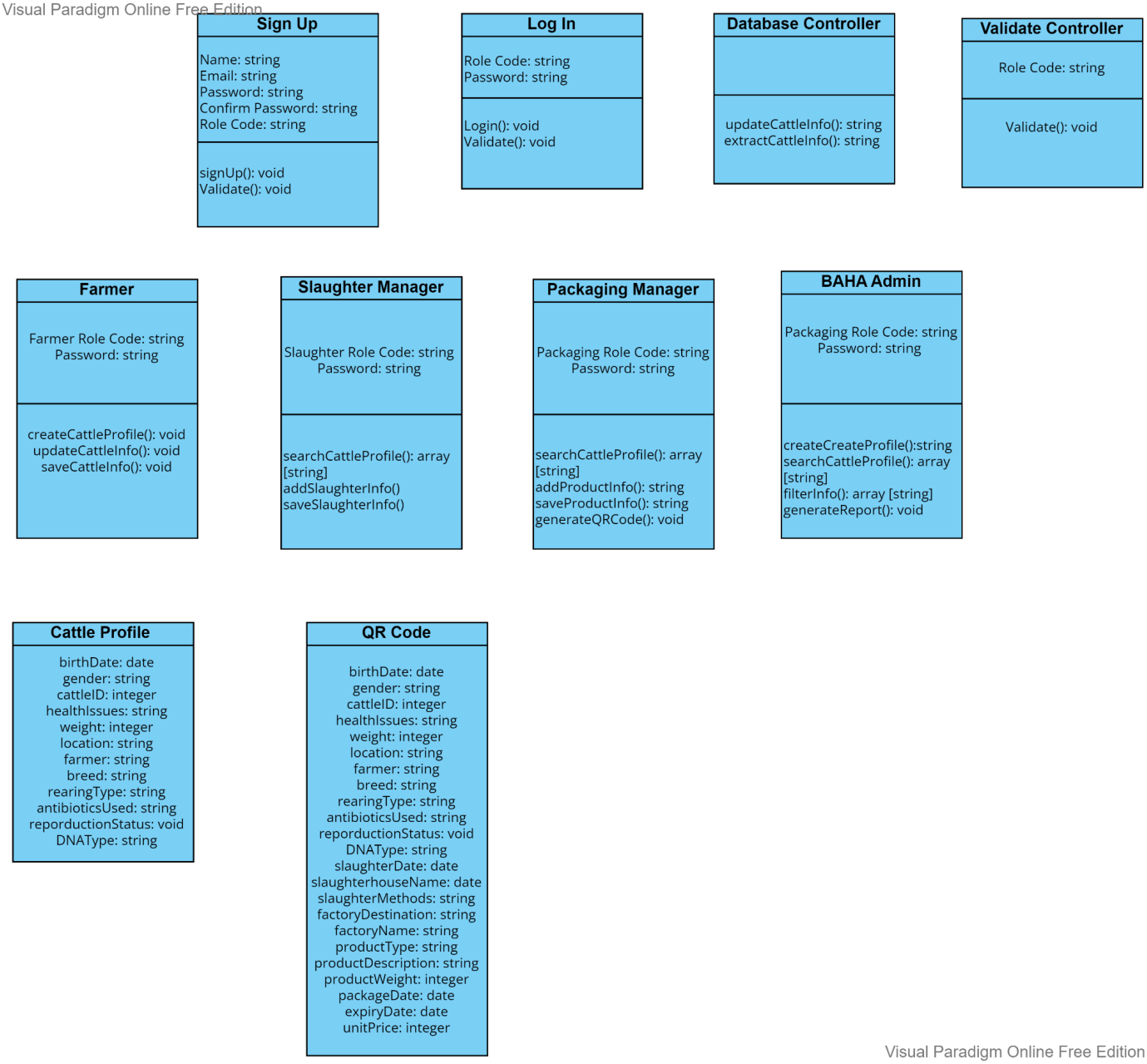

sd Generate Report
User
Interface Page
Controller
Verifier
System
Database
Notifier
Enter Role ID
Validate ID
Verify User
Verification Result
alt
[valid input]
Redirection
[invalid input]
Display Error
Show Error
Display Portal()
RequestCriteria()
ProvidesCriteria()
RequestCattleInfo()
SendCattleInfo()
sendRecord()
giveRecord()
displayRecord()
Clicks Add Button
Enters Slaughter Information
SendSlaughterInfo()
SendInfo()
Add Info
UC8: Add Slaughter Information
Info Succesfully added
SendNotification()
DisplayNotification()

Class Diagram and Interface Specification

Class Diagram



Data Types and Operation Signatures



Traceability Matrix

Domain Concepts	Sign Up	Login	Database-Controller	Validate-Controller	Farmer	Slaughter Manager	Packaging Manager	BAHA Admin	Cattle Profile	QR Code
Controller	x	x	x	x	x	x	x	x	x	x
Verifier	x	x		x					x	x
PortalDisplay	x	x	x	x	x	x	x	x	x	x
ProfileCreator			x		x				x	
EnterInfo					x	x	x	x	x	x
SaveInfo			x		x	x	x	x		
DatabaseConnection	x	x	x		x	x	x	x		
Notifier			x						x	x
Scanner							x			
QR Code							x		x	x
DisplayReport			x				x		x	x
SearchRequest					x	x	x	x		
DatabaseRetreive			x							
AddInfo			x							
DatabaseStore			x							
GenerateQR							x	x		x
QRRequest			x							x
QRDataRetreive			x							x
DisplayQR							x	x		x
ReportGenerator			x					x		
SearchFormDisplay				x	x	x	x	x		
Filter Search			x					x		
SearchRequest				x	x	x	x	x		
CriteriaSearch				x	x	x	x	x	x	
Select ID				x	x	x	x	x	x	
DisplayRecords				x	x	x	x	x		x

Sign Up -

- Deals with the sign up functionalities when a manager signs up to use the Website Application.
- Not Derived from any of the Domain Concepts however it works alongside VERIFIER in assisting to ensure that the user enters the right credentials. Also works alongside INTERFACE PAGE which collects user inputs and displays the page for viewing.

Log in -

- Deals with the login functionalities when a manager signs in to perform an operation
- Not Derived from any of the Domain Concepts however it works alongside VERIFIER in assisting to ensure that the user enters the right credentials. Also works alongside INTERFACE PAGE which collects user inputs and displays the page for viewing.

Database Controller

- Evolved from the Concept DATABASECONNECTION which handles the connection between the other Domain concepts and the database

Validate Controller

- Evolved from the Concept VERIFIER which handles the validation of user role ID and password by comparing values in the Database with the User's Input.

Farmer

- Concerned with handling different functionalities such as CREATECATTLEPROFILE(), which creates a new profile for a newly registered cattle, as well as UPDATECATTLEINFO(), which updates the information for an already created Cattle profile and lastly SAVECATTLEINFO(), which saves the newly updated information in the database.

Slaughter Manager

- Concerned with handling different functionalities such as SEARCHCATTLEPROFILE(), which searches for a cattle profile using a unique identifier such as an ID, as well as ADDSLAUGHTERINFO(), which adds information about the slaughter of an already created cattle profile. Works after UPDATECATTLEINFO() has been implemented by the Farmer and lastly SAVESLAUGHTERINFO(), which saves the newly updated information in the database.

Packaging Manager

- Concerned with handling different functionalities such as SEARCHCATTLEPROFILE(), which searches for a cattle profile using a unique identifier such as an ID, as well as ADDPRODUCTINFO(), which adds information about the packaging of an already created cattle profile. Works after ADDSLAUGHTERINFO() has been implemented by the Slaughter Manager, another functionality that is being performed by this class is SAVEPRODUCTINFO(), which saves the newly updated information in the database and lastly GENERATEQRCODE(), which simply generates a QR Code from the cattle the user has saved in the Database.

Cattle Profile

- Concerned with the handling and storing of several details about the cattle that is being created by the Farmer

QR CODE

- Concerned with the handling and storing of several details about the cattle that is being created by all the managers. This information will be displayed to the customer when they scan the QR Code.

Algorithms and Data Structures

Algorithms

The Belize Cattle Tracker demonstrates mainly search and storing algorithms. Unlike other applications, the algorithms done by the Belize Cattle Trackers don't require complex formulas or arithmetic operations.

I. Create Profile

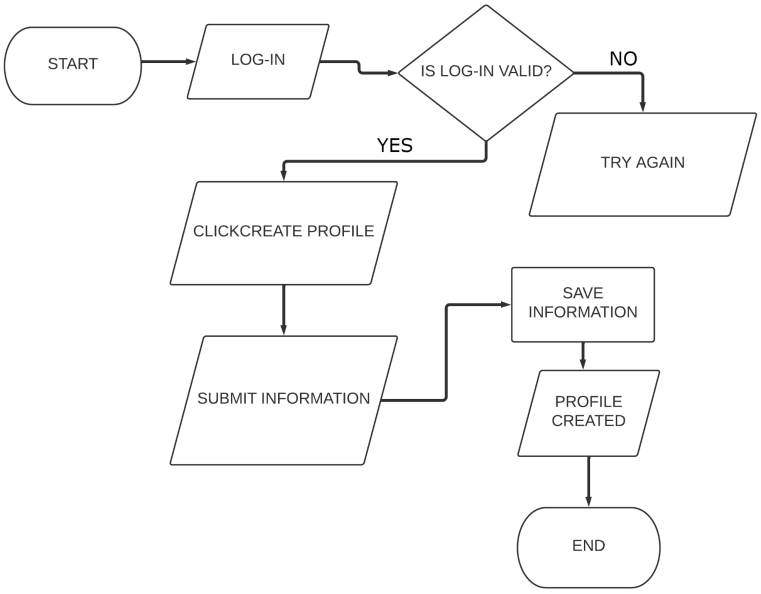


Figure 1. CreateProfile activity diagram

II. view Product Details

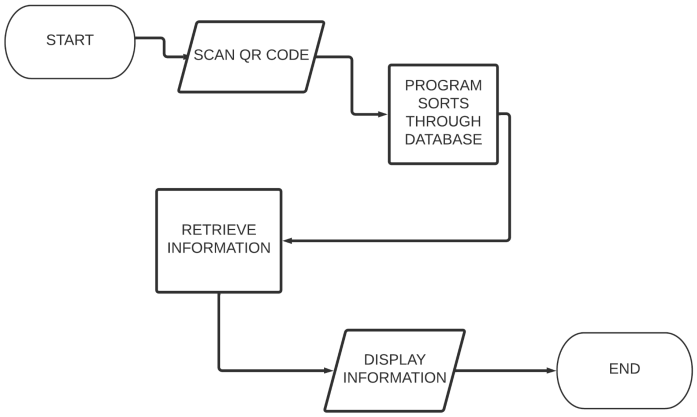


Figure 2. viewProductDetails activity diagram

III. Add Product Information

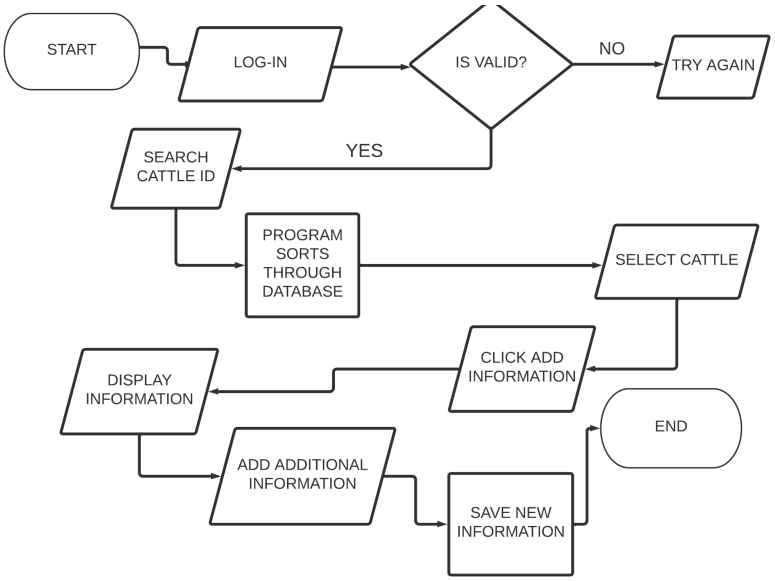


Figure 4. addProductInformation activity diagram

IV. Generate Report

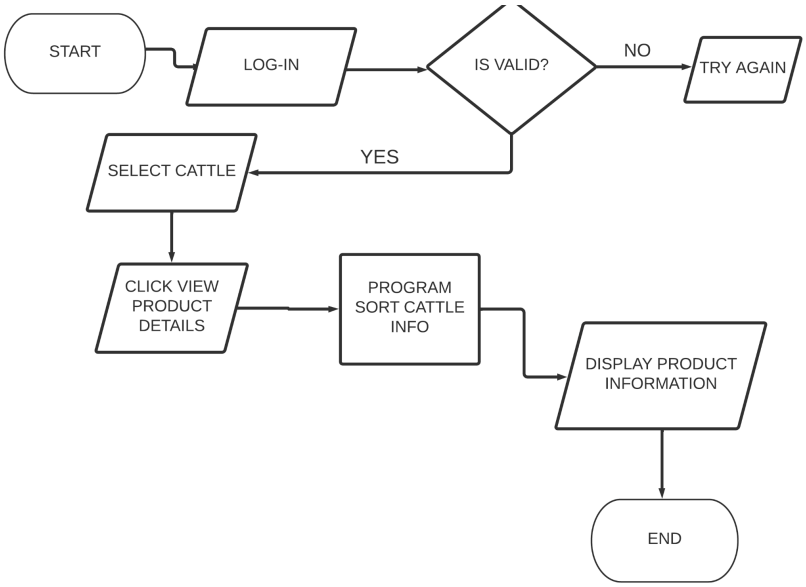


Figure 5. generateReport activity diagram

V. Add Slaughter Information

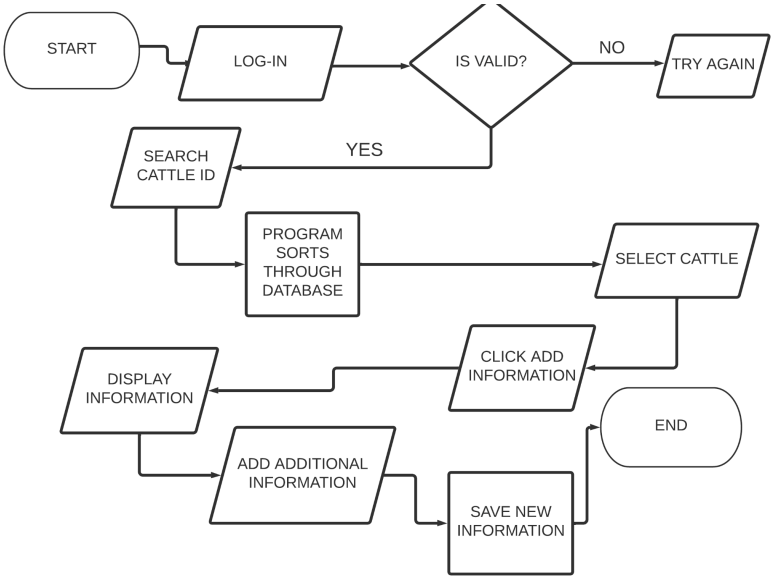


Figure 6. addSlaughterInformation activity diagram

Data Structures

The data structure that the Belize Cattle Tracker utilizes the most are arrays. Arrays are data structures that consist of a collection of elements, which are then identified by their index or key. In our case, the array's main purpose will be to hold whatever information is pulled from the database during searches. Not only do arrays provide much more flexibility in Javascript, in arrays:

- ArrayList can expand and contract dynamically.
- Elements can be added to or removed from a specific place.
- Data can be easily accessible in Array.
- Methods for Array in Javascript are very useful, for instance filter, map, pop, push, and more.
- Arrays in Javascript can contain a mix of different data types.

Concurrency

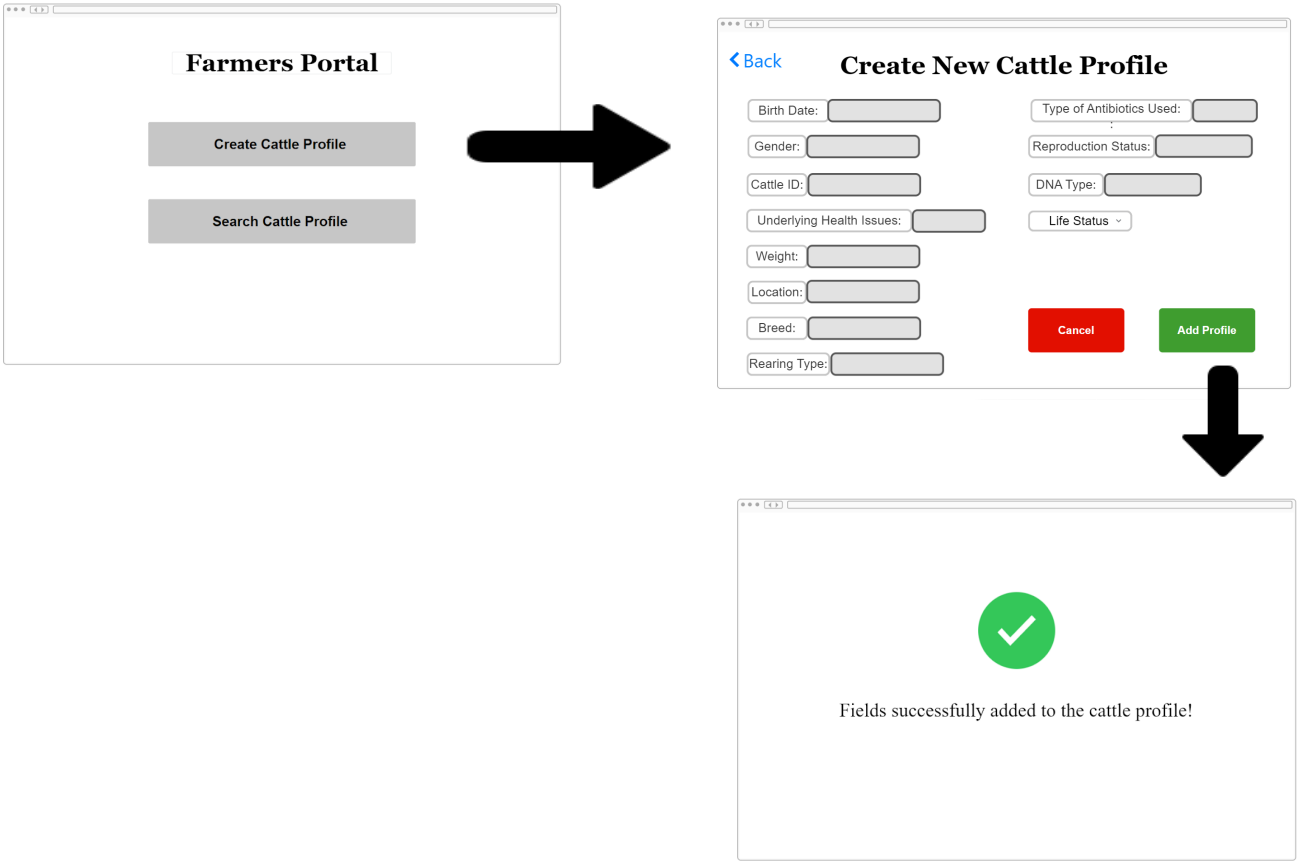
This system will use multithreading in the event that more than one user is on the website at the same time. Due to the fact that users would be able to change and fully interact with the system all at the same time, concurrency would be needed as these would be a variety of different functions taking place at the same time. By implementing multi-threading, the users would be able to simultaneously work on the system at the same time. A Farmer can be logged in at the same time as a Packaging manager and make real time changes to the database and website. In order to handle and synchronize threads and requests in real time, The server/database would be responsible for doing so. The most vital features that would need to be synchronized in our system would be the Different Roles performing insertions into the database in real time as this is vital to ensuring that there is no room for error.

User Interface Design and Implementation

Modifications and Implementation of Screens

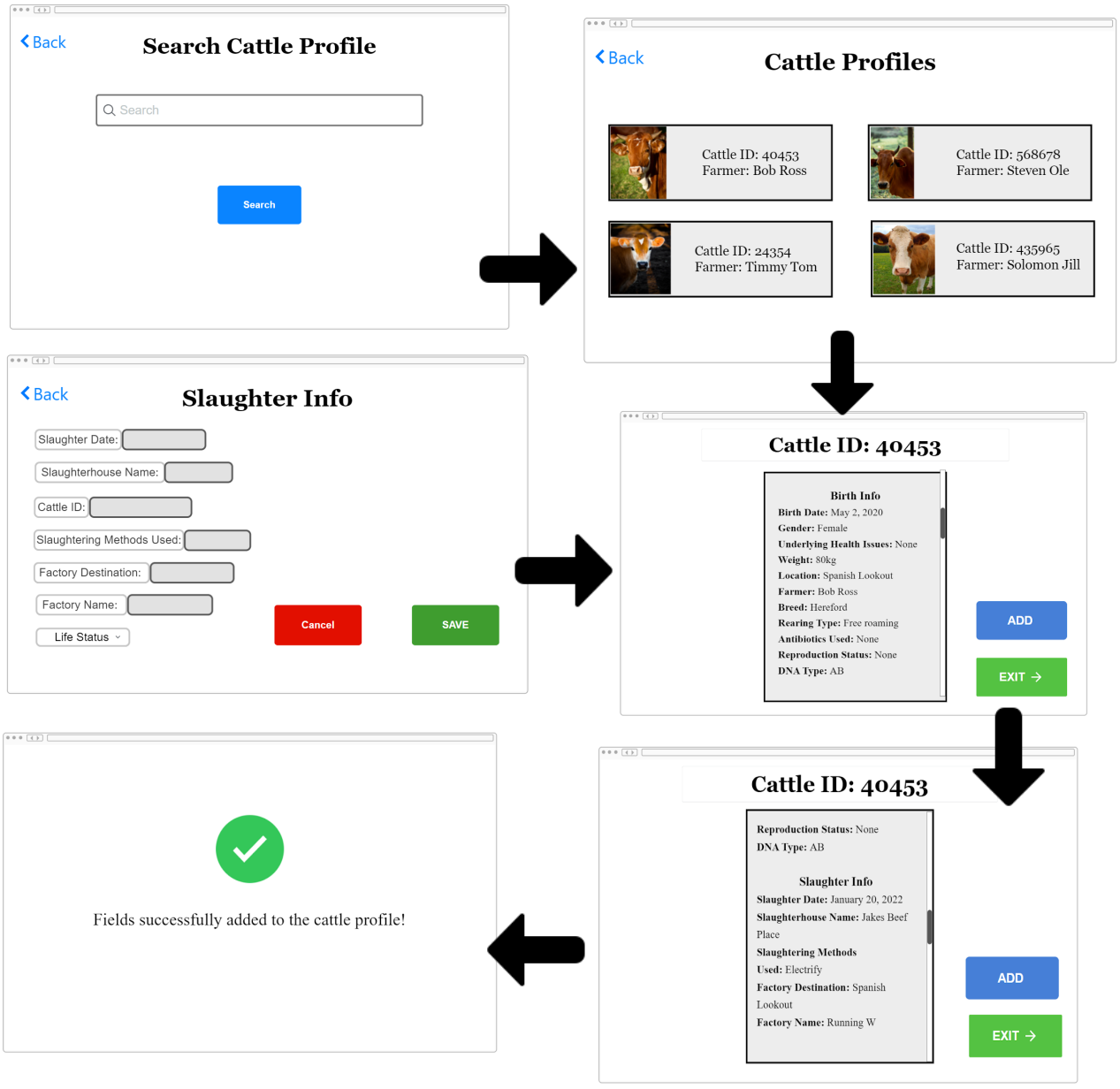
Screen MockUps

UC-1 Create Cattle Profile



For the Create New Cattle Profile screen for UC1, a “Cancel” button was added to the interface so that farmers/BAHA can terminate their operations in case they do not want to enter in the records. A “Back” button was also added to the interface to allow the farmer to go back to a previously opened interface. Adding these assists with abiding by the user control and freedom heuristic “Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action without having to go through an extended process (Nielson, 2020).” These types of heuristics should always be considered when developing the screens that store data because users can make mistakes sometimes on the system and they should have a way to exit the problem. Another heuristics that this abides by is the consistency and standards heuristic, “Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform and industry conventions (Nielson, 2020)..” Systems should always include a “Cancel” operation if there is an “Add” operation because industry standard applications always have to consider that users can make mistakes when operating with the system.

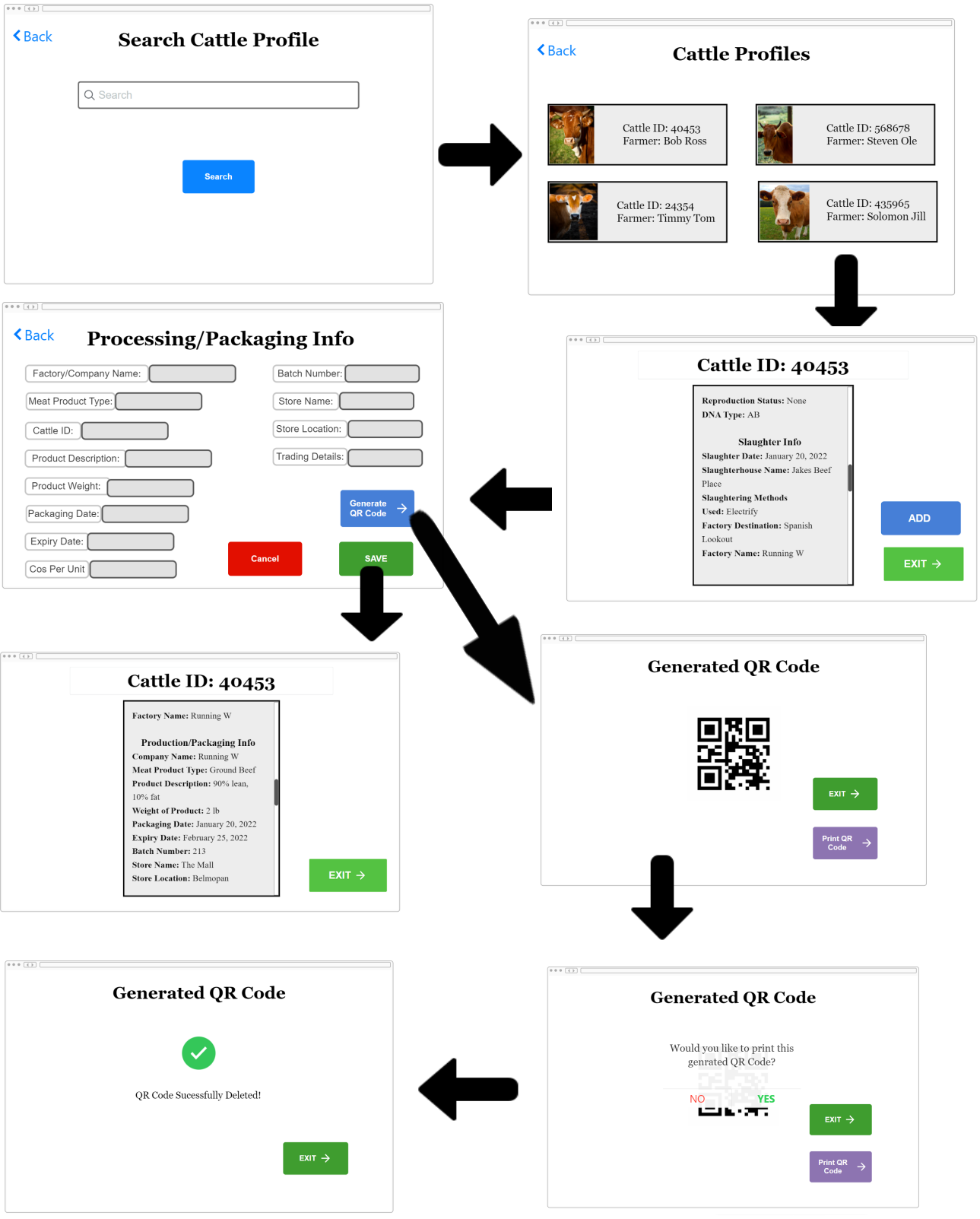
UC-8 Add Slaughter Information



For the Search Cattle Profile screen for UC8, a “Search” button was added to the interface so that admin users (farmers, BAHA, packaging and slaughterhouse managers) can click on the button to search for a cattle profile by its ID. In case they do not want to use their “Enter” button on their keyboard, this can serve as another option to provide user control and freedom. A “Back” button was also added to the Cattle Profiles interface and Slaughter Info interface to allow the farmer to go back to a previously opened interface. Adding these assists with abiding by the user control and freedom heuristic “Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action without having to go through an extended process (Nielson, 2020).” Additionally, once again, to abide by the consistency and standards

heuristic. a “Cancel” button was added to the Slaughter Info interface so that slaughterhouse managers can terminate their operations in case they do not want to enter in the records.

UC-5 Add Product Information

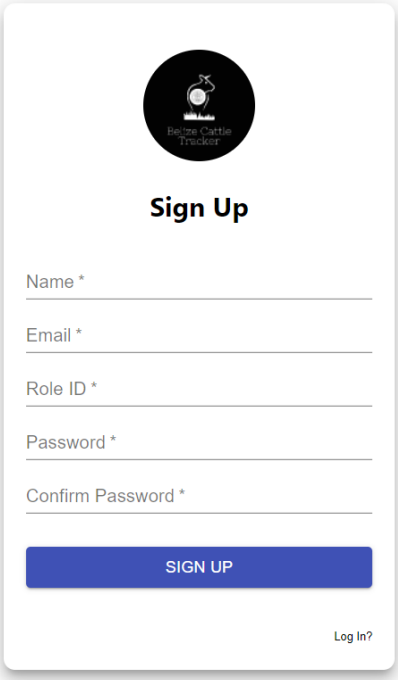


For the Search Cattle Profile screen for UC8, a “Search” button was added to the interface so that admin users (farmers, BAHA, packaging and slaughterhouse managers) can click on the button to search for a cattle profile by its ID. In case they do not want to use their “Enter” button on their keyboard, this can serve as another option to provide user control and freedom. A “Back” button was also added to the Cattle Profiles interface and Packaging/Processing Info interface to allow the farmer to go back to a previously opened interface. Adding these assists with abiding by the user control and freedom heuristic “Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action without having to go through an extended process (Nielson, 2020).” Additionally, once again, to abide by the consistency and standards heuristic. a “Cancel” button was added to the Slaughter Info interface so that slaughterhouse managers can terminate their operations in case they do not want to enter in the records.

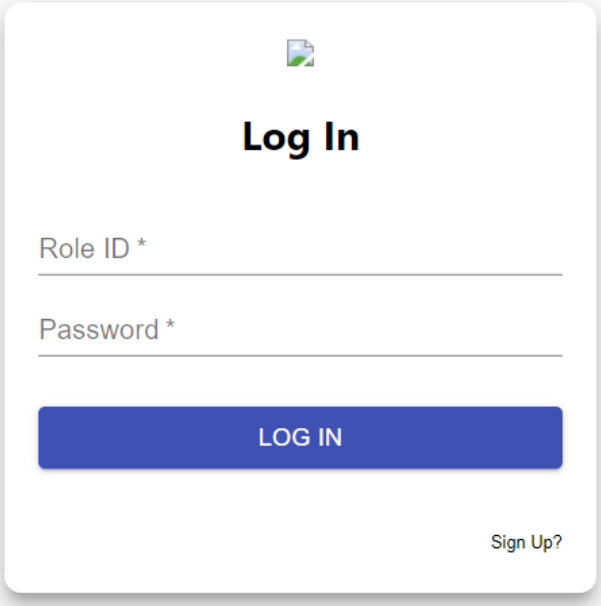
Implemented System Screens from Screen MockUps

Below are the implemented screens created based on the mockups and priority use cases. The screen format/layout is quite different as the MUI library from React was used to make the website mobile and web friendly; should in case the different admins would like to enter in the data from their mobile devices.

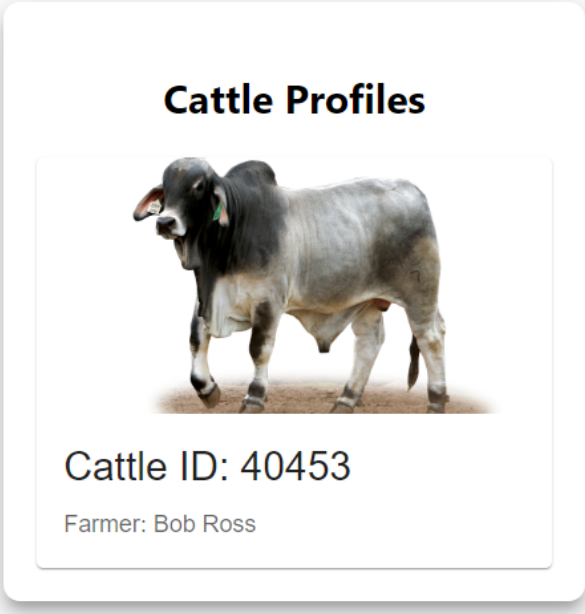
Sign Up Screen



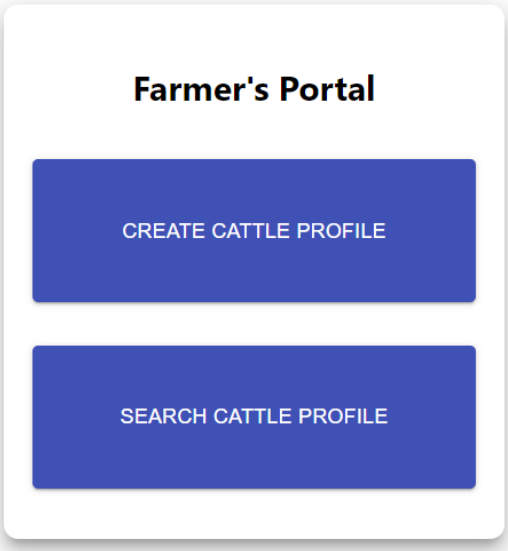
Log In Screen



Cattle Profile Screen

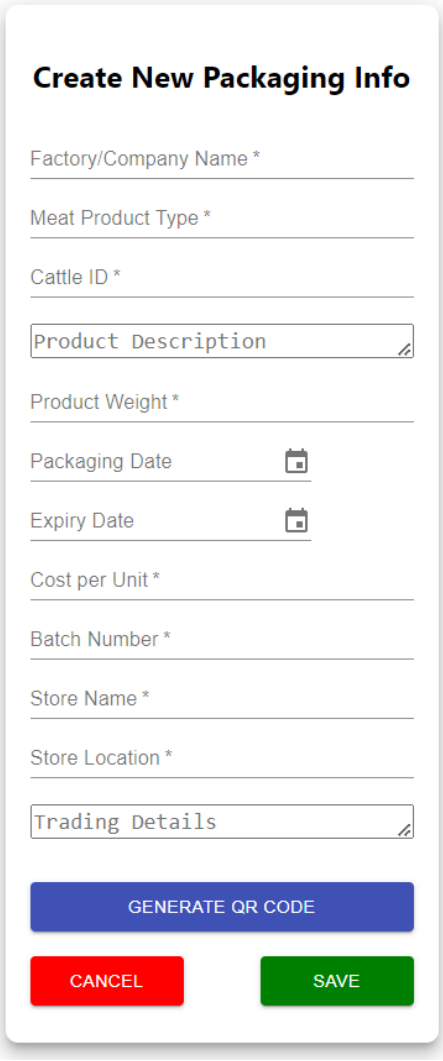


Farmer's Portal Screen



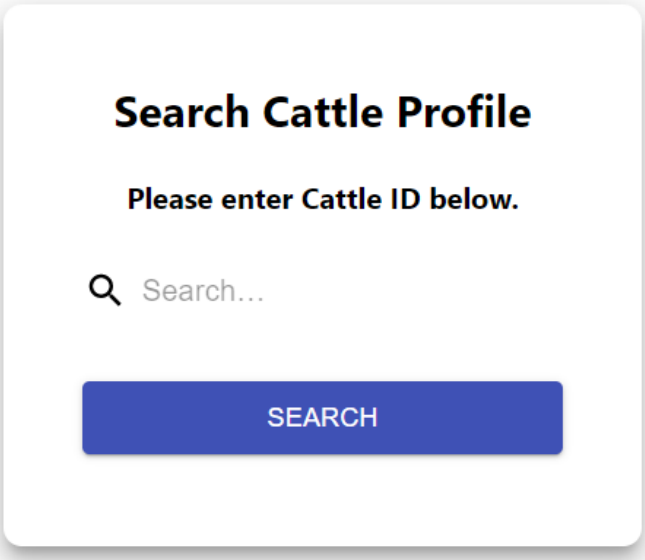
New Cattle Profile Screen

New Packaging Info Screen



New Slaughter Info Screen

Search Cattle Screen



Ease of Use

In terms of "ease of use," the system is based on both "click" and "touch" interactions between the user and the system, making it suitable for both desktop and mobile devices. The Belize Cattle Tracker System is simple to use and gives the user a sense of control, particularly when entering information about cattle or meat products. It allows the user to choose whether to exit a page or proceed to another operation (generating QR code) by clicking or touching buttons that provide the options. The pages have been designed to be as straightforward as possible, with page names, buttons, field labels, and other features condensed to provide users with only the most important information to avoid confusion. All of the system's pages have a consistent design and format (where styling and alignments are of similar structure), and complementary color schemes. This allows users to stay focused on the work at hand and avoid being sidetracked. Instead of having the user complete repeated operations, the system helps them to perform their duties more effectively, and it also offers feedback to them. For example, while generating a cattle profile, the system alerts the user that the profile was successfully created. Users who are familiar or new with mobile or desktop programs should find the Belize Cattle Tracker System to be simple to use, with little to no documentation required. Because the majority of the duties on the system include filling out forms and operating the system by clicking or touching buttons.

Design Tests

Overview of Design Tests

Functions that were used for the test cases were functionals that played a crucial part in our priority requirements and our system as a whole. There will never be a case where 100% of a project can have regression testing fully automated. Programming an automated script to handle the vast number of alternatives would be extremely tough. And that's even if it's possible. It would very certainly take far too long to justify the time savings. For the time being, that is. This is one work that humans are better suited to since they can think, react to, and compute all of the factors that machines can't yet handle.

TestCase1

Test Case	TC-1
Use Case being tested	**CreateCattleProfile**
Criteria for success/fail:	**Test that farmer can create cattle profile and was posted to the database**
Input Data:	**Alphanumeric**
Test Procedure:	
Step 1: Call function AddCattleInfo(info) with valid data.	**Success**

Step 2: Call function AddCattleInfo(info) with invalid data.	**Fail**
Step 3: Call function getCattleInfo(id) with valid data.	**Success**
Step 3: Call function getCattleInfo(id) with invalid data.	**Fail**

This Test Case will mainly test the functions of UC-1 (CreateCattleProfile). It uses Req 11, Req 10.

TestCase2

Test Case	TC-2
Use Case being tested	**ViewProductDetails**
Criteria for success/fail:	**Test that QRCode to appropriate URL**
Input Data:	**QRCode**
Test Procedure:	
Step 1: Call function ReadQRCode(info) with valid QRCode	**Success**
Step 1: Call function ReadQRCode(info) with invalid QRCode	**Fail**

This Test Case will mainly test the functions of UC-2 (ViewProductDetails). It uses Req6, Req2, Req 10.

TestCase5

Test Case	TC-5
Use Case being tested	**AddProductInformation**
Criteria for success/fail:	**Test that the information for add product was posted to the database**
Input Data:	**Alphanumeric**
Test Procedure:	
Step 1: Call function	**Success**

addProduct(info) with valid data	
Step 2: Call function addProduct(info) with invalid data	**Fail**
Step 3: Call function getProductInfo(id) with valid data.	**Success**
Step 4: Call function getProductInfo(id) with invalid data.	**Fail**

This Test Case will mainly test the functions of UC-5 (AddProductInformation) with the aid of UC-1 (CreateCattleProfile). It uses Req4, Req 10.

TestCase7

Test Case	TC-7
Use Case being tested	**Generate Report**
Criteria for success/fail:	**Test that database query that best matches the actor's search criteria and retrieve the records from the database**
Input Data:	**Alphanumeric**
Test Procedure:	
Step 1: Call function filterProductInfo(info) with valid data	**Success**
Step 2: Call function filterProductInfo(info) with invalid data	**Fail**
Step 3: Call function getCattleInfo(Id) with valid data	**Success**
Step 4: Call function getCattleInfo(Id) with invalid data	**Fail**

This Test Case will mainly test the functions of UC-7 (GenerateReport) with the aid of UC-1 (CreateCattleProfile), UC-5 (AddProductInformation). It uses Req8, Req 10,Req 15.

TestCase8

Test Case	TC-8
Use Case being tested	**AddSlaughterInformation**
Criteria for success/fail:	**Test that information was added to the database**
Input Data:	**Alphanumeric**
Test Procedure:	
Step 1: Call function addSlaughterInfo(info) with valid data	**Success**
Step 2: Call function addSlaughterInfo(info) with invalid data	**Fail**
Step 3: Call function getSlaughterInfo(Id) with valid data	**Success**
Step 4: Call function getSlaughterInfo(Id) with invalid data	**Fail**

This Test Case will mainly test the functions of UC-8 (AddSlaughterInformation) with the aid of information from UC-1 (CreateCattleProfile), UC-5 (AddProductInformation), and UC-7 (GenerateReport). It uses Req 4, Req 10.

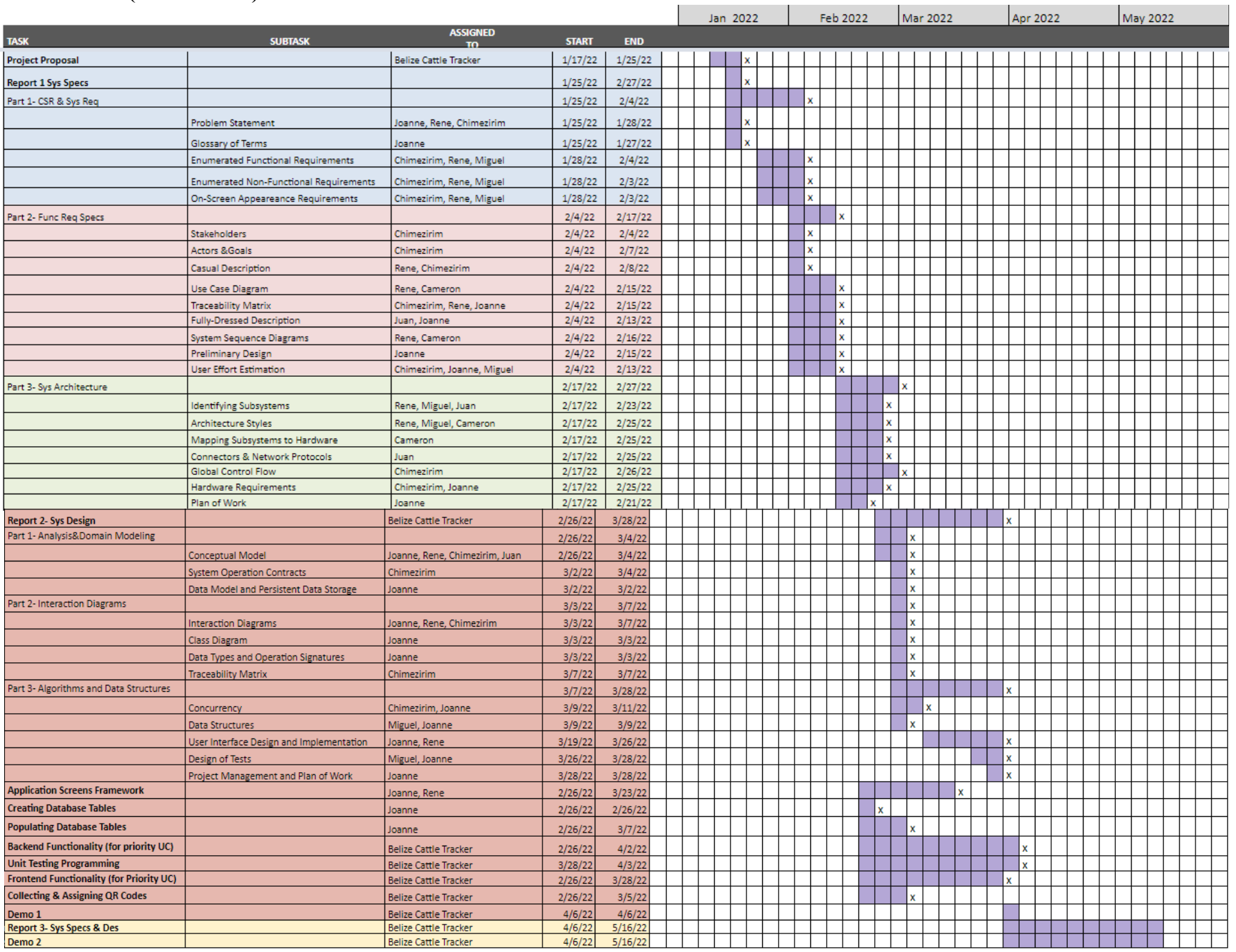
Project Management and Plan of Work

Issues Encountered

Some issues that were encountered while compiling this report was finding the proper time to work on the project together as all the members work and have daily responsibilities that they have to take care of, which means that we struggled in finding the time needed to meet and establish a deadline or make vital decisions like what programming languages we would be using or who would be doing what. There was also poor time management in certain phases of the Report process because of external factors. We were able to counter this problem and find a solution for it by making a github repo, which all the members were able to clone and work on when they had time to do so and also setting flexible deadlines. Another problem that was encountered was coming up with the classes that would be tested. Since our project scope was mostly limited to extracting information from the Database and displaying it to the user as well as Inserting information into the database and since the React already comes with a QR library, coming up with a QR Code seemed to be made much easier and so we had trouble with that aspect of the project. However we were able to counteract this problem by doing further research

as to how we could properly unit test the functionality that we already had available in our code. Another problem was finding an IDE that we would use to conduct the unit test especially since we were generally unfamiliar with how unit testing works, however after further research, we would be able to come to the conclusion that we would just continue using VSCode as it was the one that we were most comfortable with. In general even though we did encounter some difficulties with the coding process and getting firebase to connect to our react code, most of these issues were resolved after putting our heads together and working on it together to come up with a solution.

Plan of Work (Gantt Chart)



TASK	SUBTASK	ASSIGNED TO	START	END
Project Proposal		Belize Cattle Tracker	1/17/22	1/25/22
Report 1 Sys Specs			1/25/22	2/27/22
Part 1- CSR & Sys Req			1/25/22	2/4/22
	Problem Statement	Joanne, Rene, Chimezirim	1/25/22	1/28/22
	Glossary of Terms	Joanne	1/25/22	1/27/22
	Enumerated Functional Requirements	Chimezirim, Rene, Miguel	1/28/22	2/4/22
	Enumerated Non-Functional Requirements	Chimezirim, Rene, Miguel	1/28/22	2/3/22
	On-Screen Appeareance Requirements	Chimezirim, Rene, Miguel	1/28/22	2/3/22
Part 2- Func Req Specs			2/4/22	2/17/22
	Stakeholders	Chimezirim	2/4/22	2/4/22
	Actors &Goals	Chimezirim	2/4/22	2/7/22
	Casual Description	Rene, Chimezirim	2/4/22	2/8/22
	Use Case Diagram	Rene, Cameron	2/4/22	2/15/22
	Traceability Matrix	Chimezirim, Rene, Joanne	2/4/22	2/15/22
	Fully-Dressed Description	Juan, Joanne	2/4/22	2/13/22
	System Sequence Diagrams	Rene, Cameron	2/4/22	2/16/22
	Preliminary Design	Joanne	2/4/22	2/15/22
	User Effort Estimation	Chimezirim, Joanne, Miguel	2/4/22	2/13/22
Part 3- Sys Architecture			2/17/22	2/27/22
	Identifying Subsystems	Rene, Miguel, Juan	2/17/22	2/23/22
	Architecture Styles	Rene, Miguel, Cameron	2/17/22	2/25/22
	Mapping Subsystems to Hardware	Cameron	2/17/22	2/25/22
	Connectors & Network Protocols	Juan	2/17/22	2/25/22
	Global Control Flow	Chimezirim	2/17/22	2/26/22
	Hardware Requirements	Chimezirim, Joanne	2/17/22	2/25/22
	Plan of Work	Joanne	2/17/22	2/21/22
Report 2- Sys Design		Belize Cattle Tracker	2/26/22	3/28/22
Part 1- Analysis&Domain Modeling			2/26/22	3/4/22
	Conceptual Model	Joanne, Rene, Chimezirim, Juan	2/26/22	3/4/22
	System Operation Contracts	Chimezirim	3/2/22	3/4/22
	Data Model and Persistent Data Storage	Joanne	3/2/22	3/2/22
Part 2- Interaction Diagrams			3/3/22	3/7/22
	Interaction Diagrams	Joanne, Rene, Chimezirim	3/3/22	3/7/22
	Class Diagram	Joanne	3/3/22	3/3/22
	Data Types and Operation Signatures	Joanne	3/3/22	3/3/22
	Traceability Matrix	Chimezirim	3/7/22	3/7/22
Part 3- Algorithms and Data Structures			3/7/22	3/28/22
	Concurrency	Chimezirim, Joanne	3/9/22	3/11/22
	Data Structures	Miguel, Joanne	3/9/22	3/9/22
	User Interface Design and Implementation	Joanne, Rene	3/19/22	3/26/22
	Design of Tests	Miguel, Joanne	3/26/22	3/28/22
	Project Management and Plan of Work	Joanne	3/28/22	3/28/22
Application Screens Framework		Joanne, Rene	2/26/22	3/23/22
Creating Database Tables		Joanne	2/26/22	2/26/22
Populating Database Tables		Joanne	2/26/22	3/7/22
Backend Functionality (for priority UC)		Belize Cattle Tracker	2/26/22	4/2/22
Unit Testing Programming		Belize Cattle Tracker	3/28/22	4/3/22
Frontend Functionality (for Priority UC)		Belize Cattle Tracker	2/26/22	3/28/22
Collecting & Assigning QR Codes		Belize Cattle Tracker	2/26/22	3/5/22
Demo 1		Belize Cattle Tracker	4/6/22	4/6/22
Report 3- Sys Specs & Des		Belize Cattle Tracker	4/6/22	5/16/22
Demo 2		Belize Cattle Tracker	4/6/22	5/16/22

Project Progress Description

The milestones and deadlines were adjusted in response to input from our project developers and the review procedure outlined in the management processes. This project was completely managed using WhatsApp chats and a GitHub repository set up for the team. When modifications were made, they were recorded in the revision history table located at each commit's position in the repository. As stated in reports 1 and 2, group meetings gave input that was used to regulate progress reports and the construction and implementation of user screens. From the project's start date until the present, our group holds virtual meetings over Zoom on one day of the week (typically Fridays and Saturdays) to discuss the project's progress and documentation.

From our initial idea and intended design, we wanted to implement a camera library (from React) on the public (customer) side of the website so that users can scan meat product QR codes through it, but after some research and experimenting we realized that it was deemed inefficient as the camera from the camera library is flawed for some mobile devices. We then realized that the QR codes could have been scanned with the normal camera app on a user's phone and the user would immediately be redirected to the site containing the product details. This was not only more convenient but it is more time efficient for the user because the user wouldn’t have to access our public website in order to scan.

Possible new milestones include creating, designing and implementing screens and functionality for:

1. A portion/interface where farmers can only view their created cattle profiles.
2. A better filtering system for BAHA so that they can filter out and generate reports based on different things such as DOB of cattle, farmer name, etc. instead of just by cattleID.
3. An implementation of the non-functional requirement where farmers, slaughterhouse managers, and packaging managers get terminated from adding new information to the cattle profile after the cattle/produce is handed off to the next stage.

Key accomplishments formulated until present for the project:

- Portals for some of the admin users (Eg: farmer, BAHA)
- The adding of information from each stage. (Eg: Farmer→ Slaughterhouse→ Packaging)
- Unique generating of QR codes
- Information connection between Firebase database and to the specified QR code.
- QR code scanner can now scan a QR code generated by us
- Basic functionalities of the system (log in, sign up, authentication)

Breakdown of Responsibilities

Name	Did	Doing	Will Do
Joanne	R1: Project Management. Did Problem Statement., Glossary, Preliminary Designs, Plan of Work. Assisted with Use Cases, Traceability Matrix, Fully Dressed Description, Hardware Requirements. Updated Report 1 based on feedback R2: Did Data Model and Persistent Data Storage, Class Diagrams, Data Types and Operation Signatures, User Interface Design and Implementation, and Project Management. Assisted with Interaction Diagrams, conceptual model, Algorithms, Concurrency, and Data Structures.	Next phase report, attending meetings, project managing, system framework, unit testing, database management.	System framework and design, create and populate database tables, assist with front end functionality, debugging.
Chimeziri m	R1: Did System Specifications/Requirements & Global Control Flow.. Assisted with User Effort Estimation, Traceability Matrix, Problem Statement, Hardware Requirements, Project Management. Updated Report 1 based on feedback R2: Did Conceptual Model, System Operation Contracts, Traceability Matrices, Interaction Diagrams, Concurrency.	Next phase report, attending meetings, QR generation, unit testing.	Front end and back end functionalities, populate database, generate and assign QR codes.
Rene	R1: Did Problem Statement, Casual Description, Identifying Subsystems, Architectural Styles. Assisted with Use Case Diagrams, System Sequence Diagrams. Updated Report 1 based on feedback	Next phase report, attending meetings, system framework, unit testing.	System framework and design, assist with front end functionality.Populating database tables.

	R2: Did Conceptual Model, Interaction Diagrams, User Interface Design and Implementation,		
Miguel	R1: Assisted with System Specifications, User Effort Estimation, Identifying Subsystems. R2: Did Data Structures and Design of Test.	Next phase report, attending meetings, backend functionality, unit testing.	Front end and back end functionalities, populate database, generate and assign QR codes. Debugging.
Juan	R1: Assisted with Traceability Matrix, Did fully dressed Descriptions, Connectors & Network Protocols. R2: Assisted with Conceptual Model (Domain Models).	Next phase report, attending meetings, database connections, unit testing.	Front end and back end functionalities, populate database, generate and assign QR codes.
Cameron	R1: Did Architectural Styles and Mapping subsystems to Hardware Assisted with Sequence Diagrams and Use Case Diagrams. R2: Algorithms	Next phase report, attending meetings.	Front end and back end functionalities.

References

Tuck, K. (2020, February 26). *Regression testing can never be fully automated*. Kevin Tuck. Retrieved March 31, 2022, from https://kevintuck.co.uk/regression-testing-can-never-be-fully-automated/

World Leaders in Research-Based User Experience. (n.d.). *10 usability heuristics for user interface design*. Nielsen Norman Group. Retrieved April 3, 2022, from https://www.nngroup.com/articles/ten-usability-heuristics/

Delaney, J. (2018, August 24). *Join collections in Firestore*. Fireship.io. Retrieved April 3, 2022, from https://fireship.io/lessons/firestore-joins-similar-to-sql/