Save Farmers

Team: Save Farmers

Dillon Pikulik

Eric Vo

Nick Burlakov

Eastern Washington University

December 5, 2024

Version 4.0

Github: https://github.com/Sanmeet-EWU/cscd-488-490-project-save-farmers

Figma:
https://www.figma.com/design/shMiHT8N7REikxoaUDINU9/Save-Farmers?node-id=0-1&t=NkgSp0bFDW9XTU2Q-

Table of Contents

List of Figures

1 Executive Summary

The Save Farmers App is a marketplace for farmers. LTIMindtree presented this app to us. This app's purpose is to aid farmers in selling crops in a faster and more seamless way. Farmers have a lot of difficulty with crops from growing them to selling them. There is a lot of time and effort that goes into the whole process. We with Save Farmers are here to eliminate some of the hassles with selling the crops. Farmers need to sell crops fast to make space for new crops and so crops don't spoil. We are developing an app that can show buyers what a farmer has to sell with all the necessary information so that the buyer can make decisions fast and so the farmer can sell faster. With collaboration with LTIMindtree and farmers, we hope to make farmers' lives better. Farmers live very hard-working lives and we aim to make it easier for them in at least one part of the process.

Features that this app will include to help farmers will be having a large community of buyers that will buy the farmer's crops, an easy and seamless way to post, and a very interactive way to look at data of past posts on the profile page. For the buyers, we will have an excellent search integration, a large number of farmers posting their crops, and a profile page that will showcase past purchases that can aid in recontacting farmers for more crops.

2 Introduction

2.1 Purpose and Scope

This document provides a comprehensive overview of the Save Farmers App, a marketplace designed to address the challenges farmers face in selling their crops efficiently. The purpose of this document is to outline the system's objectives, define its scope, clarify specialized terms, and provide references for the tools and frameworks used in the project's development.

The Save Farmers App aims to simplify the process of selling crops by connecting farmers directly with buyers in a seamless and efficient manner. This document serves as a guide for understanding the system's design, functionality, and future directions while providing a basis for collaboration among stakeholders.

The project encompasses the design and development of a mobile application that facilitates the posting, searching, and transaction of crops. It targets two primary user groups: farmers (sellers) and buyers. The app's boundaries include features for user authentication, crop postings, search and filter functionalities, and profile management. External integrations, such as messaging or payment systems, are not part of the initial development phase but are considered for future enhancements.

2.2 Definitions, Acronyms, and Abbreviations

1. **Save Farmers App**: The system under development that connects farmers and buyers.
2. **LTIMindtree**: The project's sponsor organization.
3. **ER Diagram**: Entity-Relationship Diagram, used to design the database schema.

2.3 References

1. **GitHub Repository**: Save Farmers Project
2. **Figma Designs**: Save Farmers Figma Designs
3. **Cypress**: End-to-end testing tool used for navigation and functionality validation (Cypress.io).

4. **Firebase/SQLite**: Database solutions considered for the app's backend.
5. **Ionic Framework**: The framework used for this project to make the apps.

This document establishes the foundation for developing a robust solution aimed at helping farmers while simplifying the sales processes.

3 System Overview

3.1 Product Perspective

The Save Farmers App fits within the broader ecosystem of agricultural technology solutions aimed at improving farmer livelihoods. It addresses inefficiencies in traditional crop marketing processes, such as delayed sales and spoilage, by offering a centralized platform where farmers can post their crops and buyers can search for them. The app draws inspiration from platforms like Facebook Marketplace and Central Dispatch but tailors its functionality to the unique needs of the agricultural sector. It operates without a built-in payment system or in-app messaging.

3.2 Product Features

The app includes the following features, aligned with its functional requirements:

- User Login (FR-1): Allows farmers and buyers to create and manage accounts.
- Crop Posting (FR-2): Enables farmers to list their crops for sale with details like price, quantity, and location.
- Crop Search and Filters (FR-3, FR-6): Buyers can search for specific crops and apply filters to help their results.
- Profile Page (FR-4): Users have dedicated profiles displaying transaction histories and past posts.
- Post Editing (FR-5): Farmers can update existing posts to ensure accurate information for buyers.

- Viewing the Listings (FR-3): Detailed crop listings provide essential data to help buyers make informed decisions.

3.3 User Classes and Characteristics

- Farmers (Sellers):
    - Primary users are responsible for posting crops for sale.
    - Require features like account creation, crop posting, and editing.
- Buyers:
    - Secondary users who search for and purchase crops.
    - Use features like search, filters, and transaction history.

3.4 Assumptions and Dependencies

- Assumptions:
    - Users have access to a smartphone with an internet connection to use the app.
    - Farmers are familiar with basic app navigation and data entry.
- Dependencies:
    - Firebase or SQLite: Backend database for storing user data, posts, and transaction histories.
    - Cypress: Tool for validating app functionality during testing phases.

4 User Stories and Scenarios

4.1 User Story 1: Crop Post

U1: As a farmer, I want to post my specific crops for sale, so that I can quickly sell the crops.

Scenario 1: Successful crop post

- Given I am on the crop post page
- When I enter the crop name, price, location, time, etc
- And I click the "Post" button

- Then I should see the listing posted

Scenario 2: Unsuccessful crop post

- Given I am on the crop post page
- When I enter the crop name, price, time, etc
- And I click the "Post" button
- Then I should see the message "Missing An Input On The Post"

4.2 User Story 2: Crop Posts

U2: As a farmer, I want to see all of my posts of crops for sale, so that I can see what crops I have already posted.

Scenario 1: Having posted crops

- Given I am on the “My Crops” page
- Then I should see all of my crop postings

Scenario 2: Not having any posted crops

- Given I am on the “My Crops” page
- Then I should see no crops posted

4.3 User Story 3: Editing Crop Post

U3: As a farmer, I want to edit the post of my specific crops that are currently for sale, so that there is no miscommunication between me (the farmer) and the buyer.

Scenario 1: Successfully deleted crop post

- Given I am on the “My Crops” page
- When I find that crop I want to delete
- And I click the "Delete" button on that crop post
- Then I should see that the listing is no longer on the “My Crops” and “Search” pages

Scenario 2: Editing crop post

- Given I am on the “My Crops” page
- When I find that crop I want to edit
- And I click the "Edit" button on that crop post
- And I changed the field that needed editing
- And I click the "Save" button on that crop post
- Then I should see that the listing has changed on the “My Crops” and “Search” pages

4.4 User Story 4: Crop Search

U4: As a buyer, I want to search for specific crops by name, so that I can quickly find and purchase the crops I am interested in.

Scenario 1: Successful crop search

- Given I am on the crop “Search” page
- When I enter the crop name "Tomatoes" in the search bar
- And I click the "Search" button
- Then I should see a list of available "Tomatoes"
- And I should see their prices, quantities, and seller information

Scenario 2: No crops found

- Given I am on the crop “Search” page
- When I enter the crop name "Dragon Fruit" in the search bar
- And I click the "Search" button
- Then I should see the message "No results found for 'Dragon Fruit'"
- And I should remain on the crop search page

4.5 User Story 5: View Crop Details

U5: As a buyer, I want to view detailed information about a specific crop, so that I can make an informed purchasing decision.

Scenario 1: Viewing crop details

- Given I have searched for "Tomatoes"
- And I see a list of available "Tomatoes"
- When I click on a specific listing for "Tomatoes"
- Then I should be taken to the crop details page
- And I should see information such as price, quantity, seller location, and delivery options

Scenario 2: Incomplete crop details

- Given I am on the crop details page for "Tomatoes"
- When the seller has not provided full details (e.g., no quantity listed)
- Then I should see the message "Quantity information not available"
- And I should still be able to view other available information

4.6 User Story 6: Account

U6: As a buyer or seller, I want to be able to create and have an account, so that I can see my profile.

Scenario 1: Making an account

- Given I need to make an account
- When I click “Sign Up”
- Then I can create an account
- And login to use the app’s features

Scenario 2: Looking at account

- Given I want to look at past transaction
- Then I click on the “Profile” button
- Then I will be in my account
- And I can see the past transactions

5 Functional Requirements

Functional Requirement:	**Description/Specification:**
[FR-1(U6)] User Login	Users should be able to create an account and log in.
Priority	Level 0 (Essential)
[FR-2(U1)] Sellers Posts	Sellers should be able to make posts of the products they are selling.
Priority	Level 0 (Essential)
[FR-3(U4,5)] Seeing Posts	Any user should be able to see the posts that are made by the sellers.
Priority	Level 0 (Essential)
[FR-4(U2,6)] Profile Page	All users should have a profile page that has a history of past transactions.
Priority	Level 1 (High Priority)
[FR-5(U3)] Sellers Editing Posts	Sellers should be able to edit their existing posts.
Priority	Level 1 (High Priority)
[FR-6(U4,5)] Search Filters	All users should be able to make search filters to find what they are looking for easier and faster.
Priority	Level 2 (Medium Priority)

6 Use Case Diagrams and Use Cases

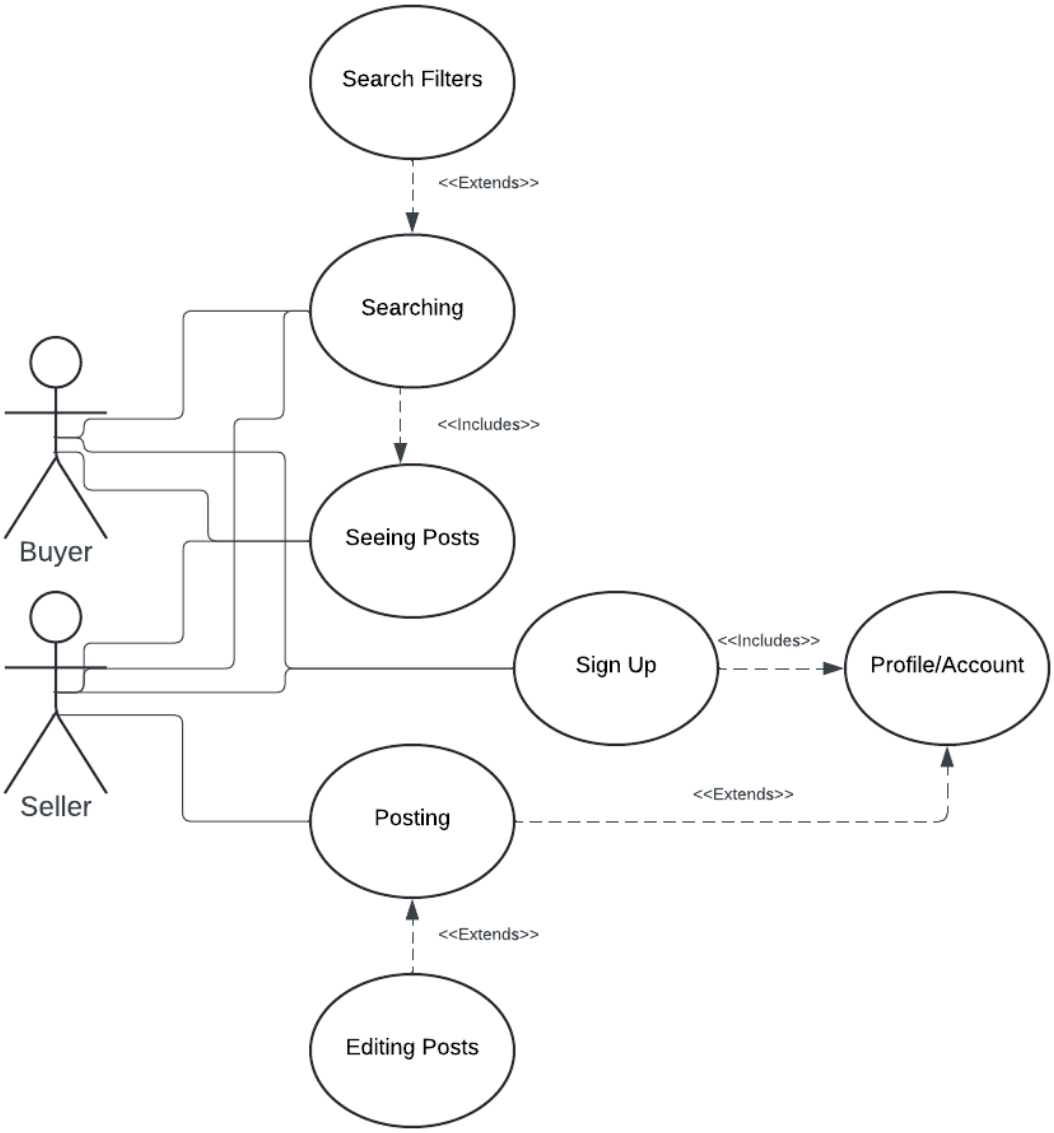

Fig. 1 (Use Case Diagram)

6.1 Use Case 1: Sign Up

- Actors: Seller and Buyer
- Preconditions: None
- Postconditions: Will have an account
- Main Flow:
    - Click Sign Up

    - Puts in information about self
    - Now has an account
- Alternative Flow: If not all information is provided, then an account can not be made.
- Related Requirements: FR-1

6.2 Use Case 2: Profile/Account

- Actors: Seller and Buyer
- Preconditions: Sign Up
- Postconditions: Can post and use the account
- Main Flow:
    - Click Profile Button
    - Now they can see their profile
    - And they can use the feature of the app
- Alternative Flow: Without having the profile they can't use the many different features of the app.
- Related Requirements: FR-1, FR-4

6.3 Use Case 3: Posting

- Actors: Seller
- Preconditions: Has Account
- Postconditions: Can post their product
- Main Flow:
    - Click "Post"
    - Fill in information
    - Then click "Submit"
- Alternative Flow: If they don't fill out all of the information the post will be incomplete and won't post.
- Related Requirements: FR-2, FR-4

6.4 Use Case 4: Searching

- Actors: Seller and Buyer
- Preconditions: Has Account
- Postconditions: Can search
- Main Flow:
    - Click Search
    - Will see all available products
    - And can search within them
- Alternative Flow: If no products are available, you won't see anything.
- Related Requirements: FR-3

6.5 Use Case 5: Search Filters

- Actors: Seller and Buyer
- Preconditions: Has Account
- Postconditions: Can search with filters
- Main Flow:
    - Click Search
    - Click Filter
    - Make Filter
    - Click Apply
    - Now they will have their search filtered
- Alternative Flow: If no posts are made, you won't see anything.
- Related Requirements: FR-3, FR-6

6.6 Use Case 6: Seeing Posts

- Actors: Seller and Buyer
- Preconditions: Has Account
- Postconditions: Can see sellers' posts
- Main Flow:
    - Click Posts
    - Now you will see all available posts
- Alternative Flow: If no posts are made, you won't see anything.
- Related Requirements: FR-3

6.7 Use Case 7: Editing Posts

- Actors: Seller
- Preconditions: Has Account
- Postconditions: The user can edit their posts
- Main Flow:
    - Click Profile
    - Then find the post you want to edit
    - Click Edit
    - Then edit the post
    - Then Click Submit
- Alternative Flow: If you don't want to edit the post you don't have to do anything.
- Related Requirements: FR-5

7 Non-functional Requirements

Non-Functional Requirement:	**Description:**

[NFR-1] Usability	The system shall be easy to navigate for all users, regardless of technical skill level.
[NFR-2] Availability	The system shall have a consistent uptime so that users can trust the app and use it reliably.
[NFR-3] Scalability	The system shall utilize a modular architecture, enabling individual components to be upgraded or replaced without impacting the overall system functionality.
[NFR-4] Performance	The system shall load pages within 2 seconds under normal traffic conditions.

8 Traceability Matrix

Functional Requirement:	**Use Case:**	**User Story:**	**Priority:**
FR-1: User Login	UC-1: Sign Up	US6: As a user, I want to create an account and log in.	Level 0
FR-2: Sellers Posts	UC-3: Posting	US1: As a seller, I want to post my crops up for sale.	Level 0
FR-3: Seeing Posts	UC-6: Seeing Posts	US4, US5: As a user, I want to see all public crop postings.	Level 0
FR-4: Profile Page	UC-2: Profile/Account	US2, US6: As a user, I want to see my transaction history.	Level 1
FR-5: Seller Editing Posts	UC-7: Editing Posts	US3: As a seller, I want to edit my existing posts.	Level 1
FR-6:Search Filter	UC-5: Search Filters	US4, US5: As a user, I want to be able to search for specific posts.	Level 2

9 Evaluation of Existing Solution

According to our project description, I have 2 existing apps/websites that I can evaluate. Facebook Marketplace is a marketplace where any user of Facebook can post anything for sale, with exceptions. Our app will be more geared toward farmers but use the same concepts. Farmers will be able to post what they have for sale and give a price, and a description of necessary information. Just like Facebook Marketplace, we will have no internal payment options but Facebook does use a messaging app Messenger which could be good to incorporate later into our app. This way we could put out less public information on our app such as phone numbers.

Another website that I will look at is Center Dispatch. That website is geared towards car transport. Central dispatch has brokers that post the cars that need to be moved and shippers that move those cars. On the post the information about what vehicle, price, location, days, and contact number are provided to the shippers. The contact of the shipper to broker and payment are all done outside the website itself. Our app will look more like Central Dispatch. Implementation of payment and messaging within the app is not a priority for us at this moment.

In both of these websites/apps, there is a big issue with user authentication, spam, and resellers. In Facebook Marketplace there is no real way to avoid this and it is solely up to the user to trust the other. While in Central Dispatch some ways/questions could be provided to prove the legitimacy of a shipper/broker outside of the app.

These 2 examples are going to be a big source of inspiration for our app since our sponsor has given us a lot of creative freedom. First keeping in mind the requirements, then implementing useful features and possibly creating our own to benefit farmers.

10 System Architecture Design

10.1 Architecture Diagram

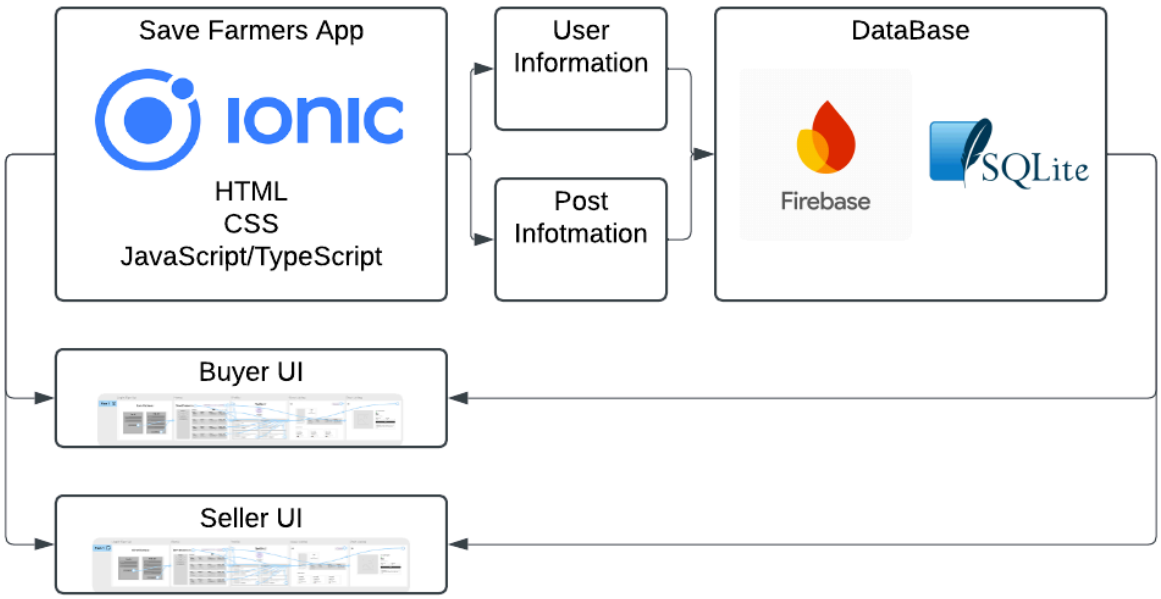

Fig. 2 (Architecture Diagram)

There are two users, the buyer and the seller (Farmers). These users will be able to access the Save Farmers App from Android and IOS (Apple). The app will use a Firebase or Sqlite database (currently using Sqlite). This app will be created first in a local environment therefore we have no web connection and database certainty.

10.2 Behavioral Diagram (Activity Diagram)

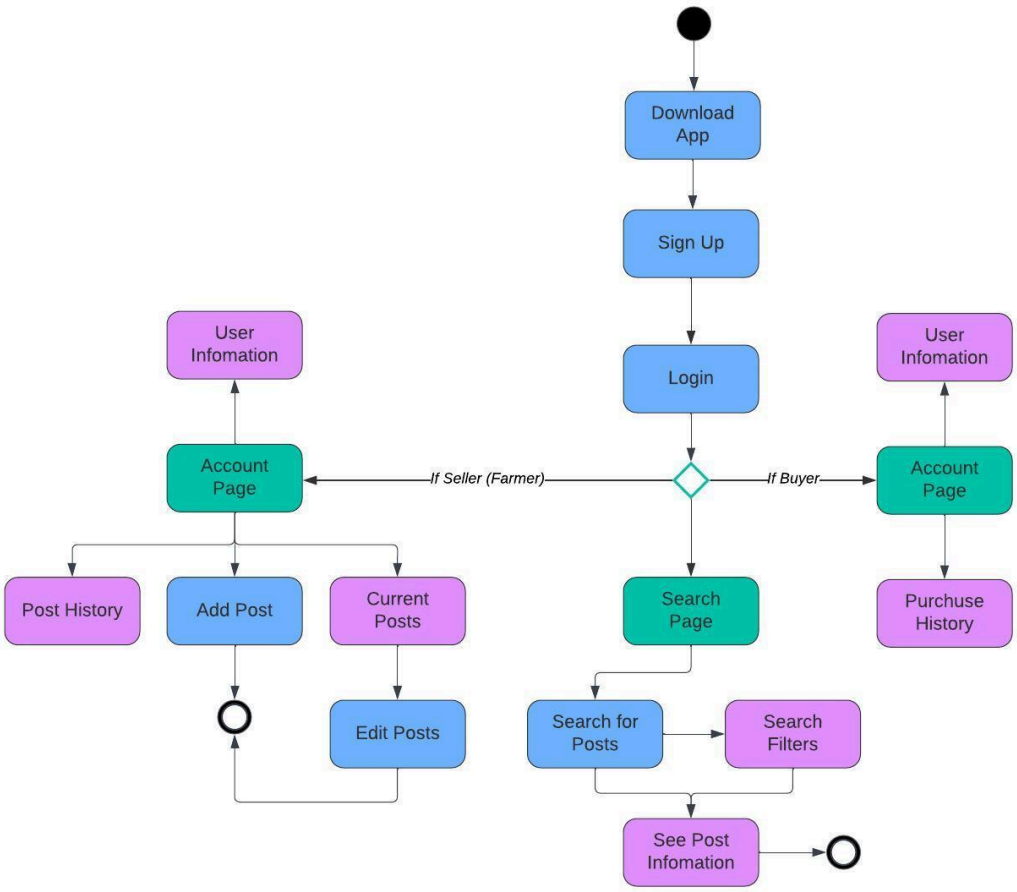

Fig. 3 (Activity Diagram)

A user will download the Save Farmers App and proceed to make an account. Once an account is made the user can log in. If the user is a seller and has a seller account they can access their account page. The seller can also make posts and edit them. A seller can see their post history and user information on the account page. If you are a buyer and have a buyer account you can access your account page and see user information and purchase history. Both buyers and sellers can access the search page, search for posts, and make filters. After a user finds the post that they want, they can contact the seller through the given information in the post.

10.3 Structural Diagram (Class Diagram)

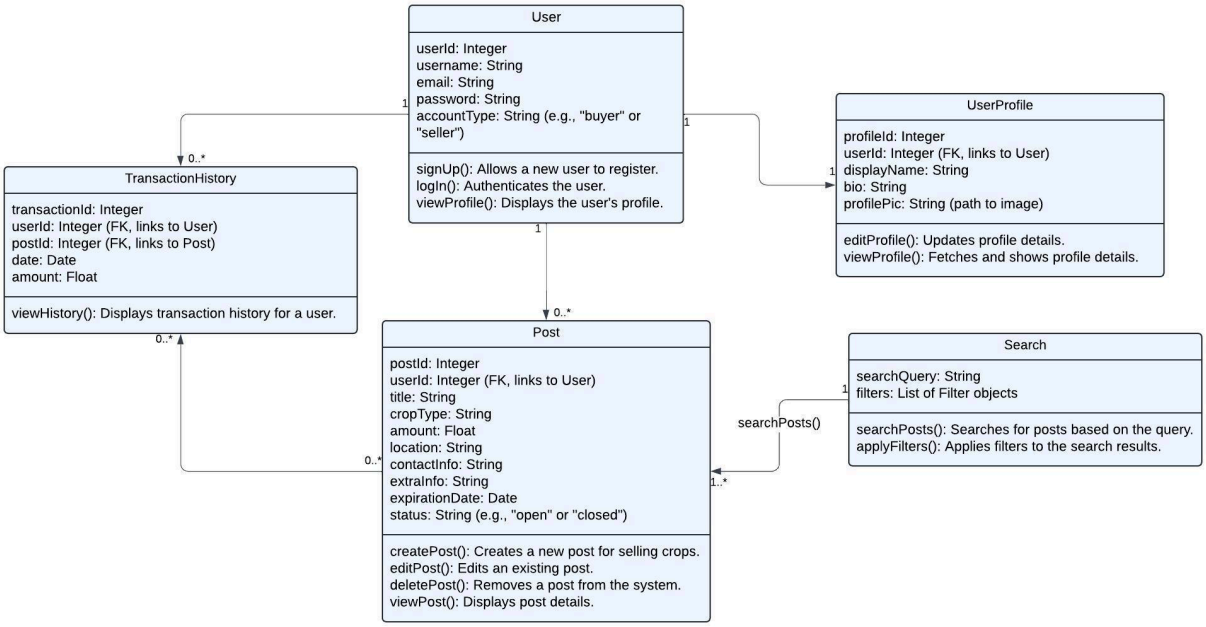

Fig. 4 (Class Diagram)

A User object creates a corresponding UserProfile object upon account creation. The viewProfile() method of User fetches details from UserProfile. Users with a seller account type can create, edit, and delete Post objects. The Post class is associated with the User through the userId foreign key. The TransactionHistory class tracks all transactions a user is involved in, with a foreign key reference to both User and Post objects. Users can view their transaction history through the viewHistory() method. The Search class interacts with the Post class to retrieve posts based on a searchQuery and applied filters. This interaction facilitates buyers to find relevant crop listings.

11 User Interface Design

11.1 Rough Draft Mockup

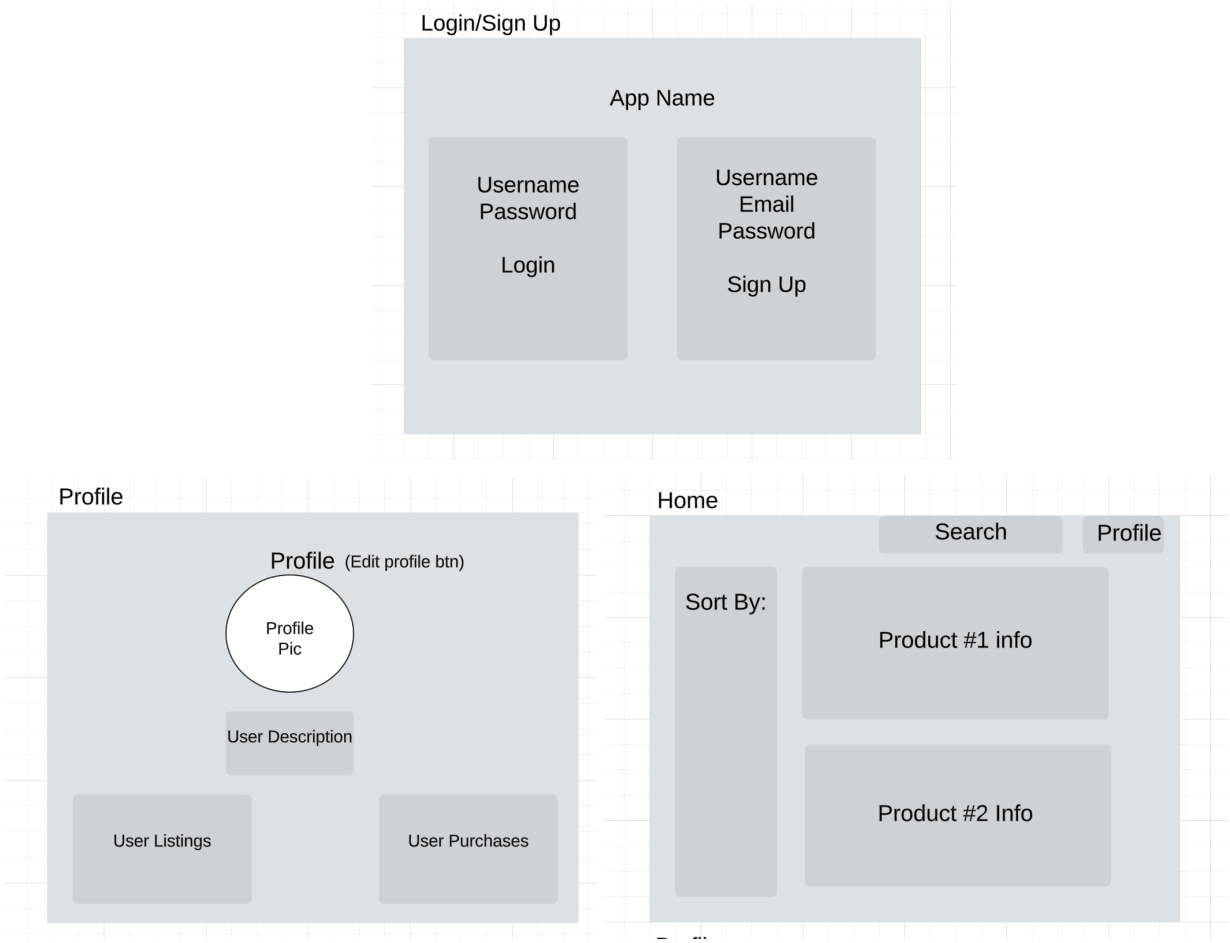

Fig. 5 (Rough Draft Mockup)

Our initial mockup was a hand drawn sketch that covered the basics; a screen to login/sign up, view listings, and view user profile. Using this basic design we digitized our first wire frame and proceeded to create a more detailed daft in Figma that would cover our most important elements.

11.2 Figma Design

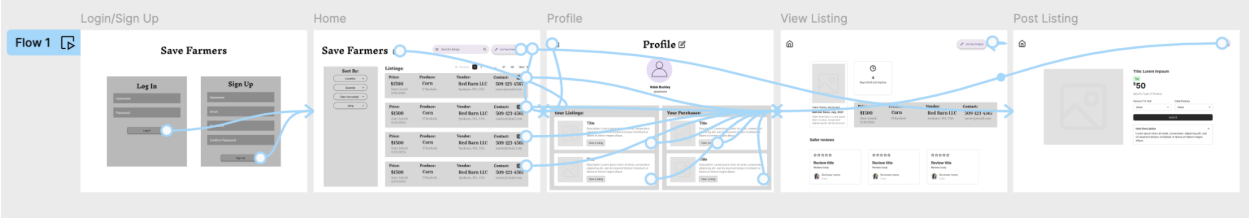

Fig. 6 (Figma Design)

To view the original file you can click here. In addition to the original screens of login/sign up, view listings, and view user profile, we also added a post listing and view listing screen.

11.3 Initial UI Features

The **Login/Sign Up** screen allows users to either log in with a username and password or sign up with additional fields like email and password confirmation. The **Log In** button directs users to the **Home** screen upon a successful login, as does the **Sign Up** button.

Fig. 7 (Figma Login/Sign Up)

The **Home** screen lists cards with details such as price, product type, vendor name, location, and contact info, along with options to sort listings by location, quantity, or date harvested. Users can

also search listings using a search bar. Each listing card has a clickable button that takes users to the **View Listing** screen for the selected item.

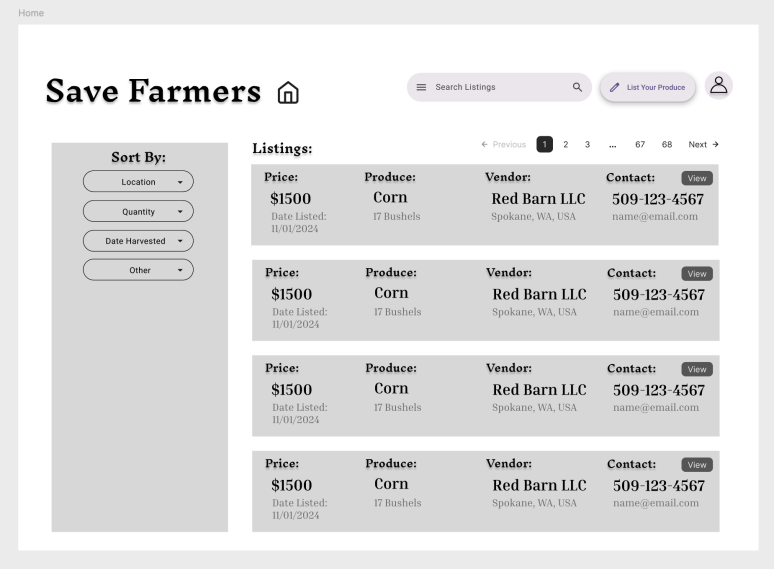

Fig. 8 (Figma Home)

In the **Profile** screen, users see their profile picture, name, and email, along with tabs for "Your Purchases" and "Your Listings," where they can manage items they've bought or listed. The **Edit Profile** button allows users to update their information within the Profile screen. In the **Your Purchases** and **Your Listings** sections, each item includes a **View Listing** button that directs users to the **View Listing** screen for that specific product. Additionally, there is a **Post New Listing** button that takes users to the **Post Listing** screen to create a new listing.

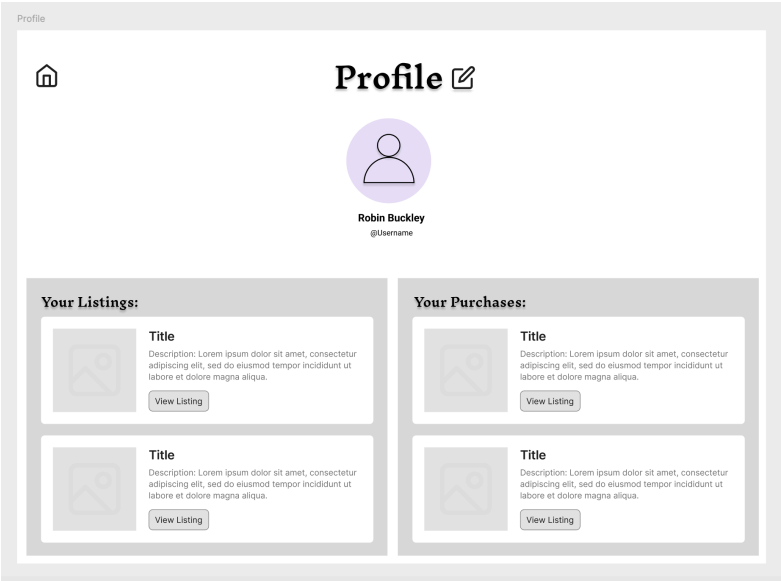

Fig. 9 (Figma Profile)

The **View Listing** screen provides a detailed look at a product with an image, title, price, quantity, stock info, vendor location, and expiration date, as well as product descriptions and seller reviews. There is a **Back** button that returns users to the **Home** screen, as well as a profile button that redirects users to the **Profile** screen.

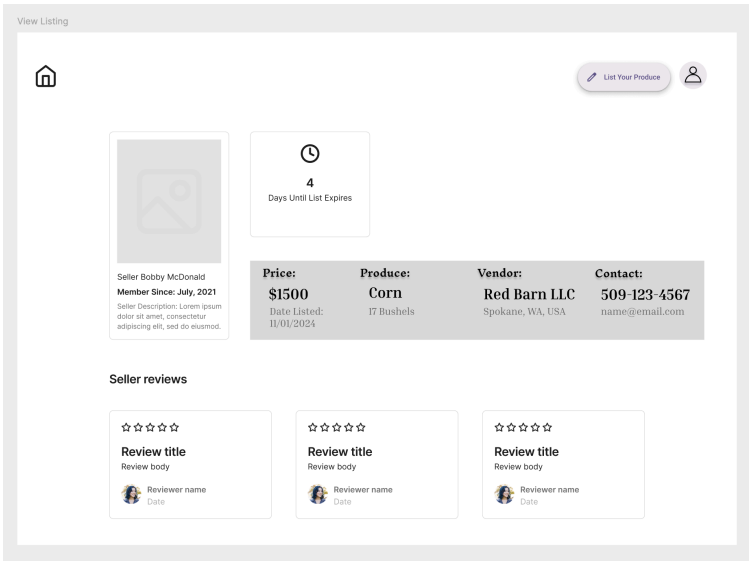

Fig. 10 (Figma View Listing)

The **Post Listing** screen allows users to create a new listing by entering a title, price, quantity, description, and uploading an image. Users can complete the listing process by clicking the **Submit** button, which publishes their listing and redirects them back to the Profile screen under “Your Listings” so they can view their newly created post.

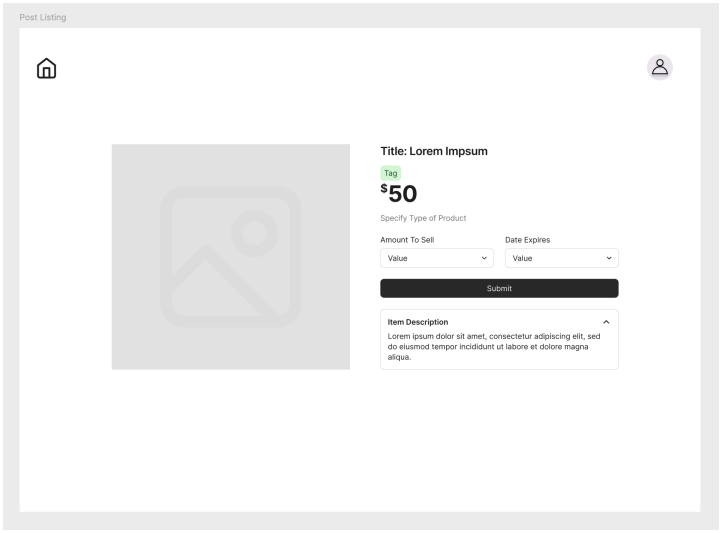

Fig. 11 (Figma Post Listing)

12 Data Design

12.1 Data Model

Entities:

- **User:** Represents an individual registered in the system.
- **Post:** Represents a posting of goods (like crops) created by users.

Relationships:

- **User to Post:** Zero-to-Many.

12.2 Database Schema (ER Diagram)

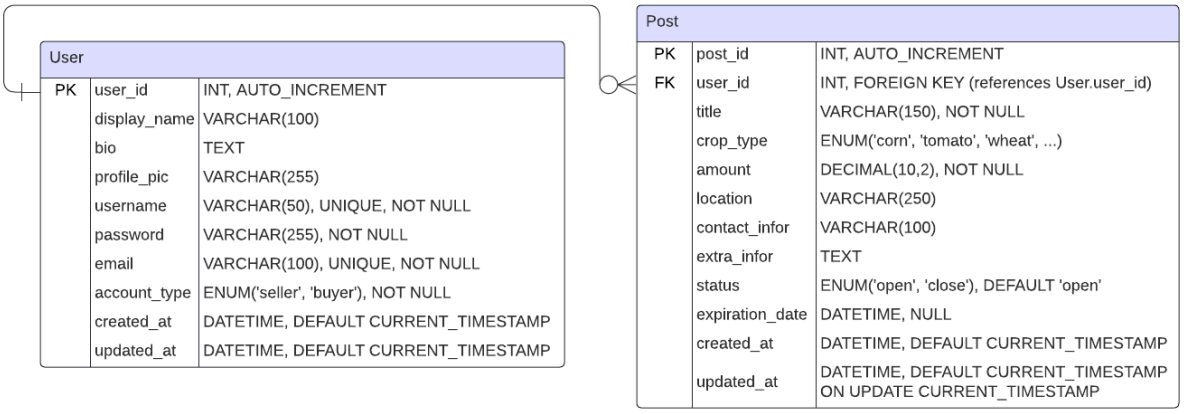

Fig. 12 (ER Diagram)

12.3 Sample Data (SQL Inserts)

Sample Data for **User** Table:

```
INSERT INTO
User (username, email, password, account_type)
VALUES
('john_doe', 'john@example.com', 'password123', 'buyer'),
('jane_smith', 'jane@example.com', 'securepass', 'seller');
```

Sample Data for **Userprofile** Table:

```
INSERT INTO
UserProfile (user_id, display_name, bio, profile_pic)
VALUES
(1, 'John Doe', 'Bio of John Doe', 'path/to/john_pic.jpg'),
```

(2, 'Jane Smith', 'Bio of Jane Smith', 'path/to/jane_pic.jpg');

Sample Data for **Post** Table:
INSERT INTO
Post (user_id, title, crop_type, amount, location, contact_info, extra_info, expiration_date, status)
VALUES
(2, 'Fresh Corn for Sale', 'corn', 100.00, 'New York', 'jane@example.com', 'Fresh, organic corn.', '2024-12-31', 'open'),
(2, 'Tomato Crop for Sale', 'tomato', 150.00, 'California', 'jane@example.com', 'High-quality tomatoes.', '2024-11-30', 'open');

13 Prototype Development

13.1 Current Implemented Functionalities

Currently, we have implemented the Login, Sign up, Profile, and Home pages. These pages do not have a backend. We have a working local database using Sqlite. This database is setup to work in Ionic. On the database branch in Github on the Profile page, there are 3 field inputs that will take those inputs, store them in the database, and output them on the same page. This database is for testing that it is working.

13.2 Login/Sign Up Pages:

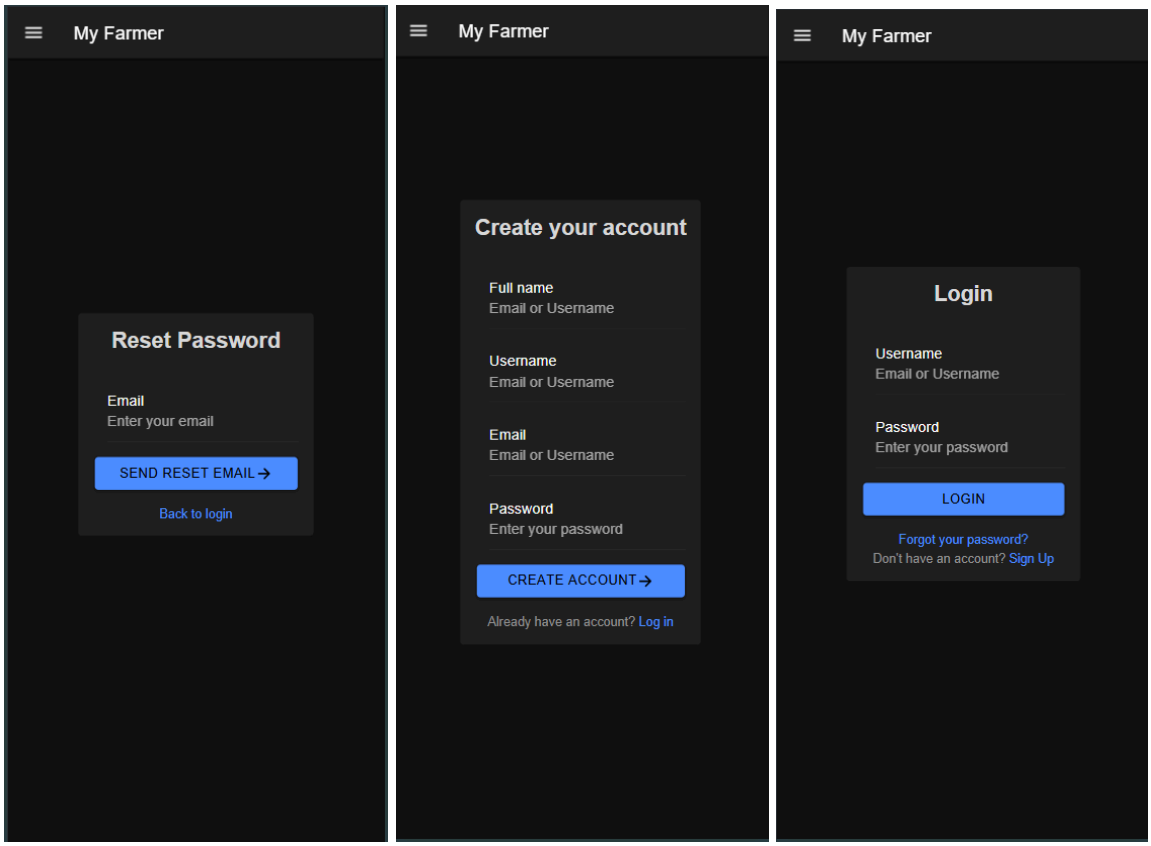

Fig. 13 (Login/Sign Up Pages)

Using the Ionic framework we made these pages. They do not reflect the full design that we are going for but are enough to help other parts of the app be created.

13.3 Profile Page:

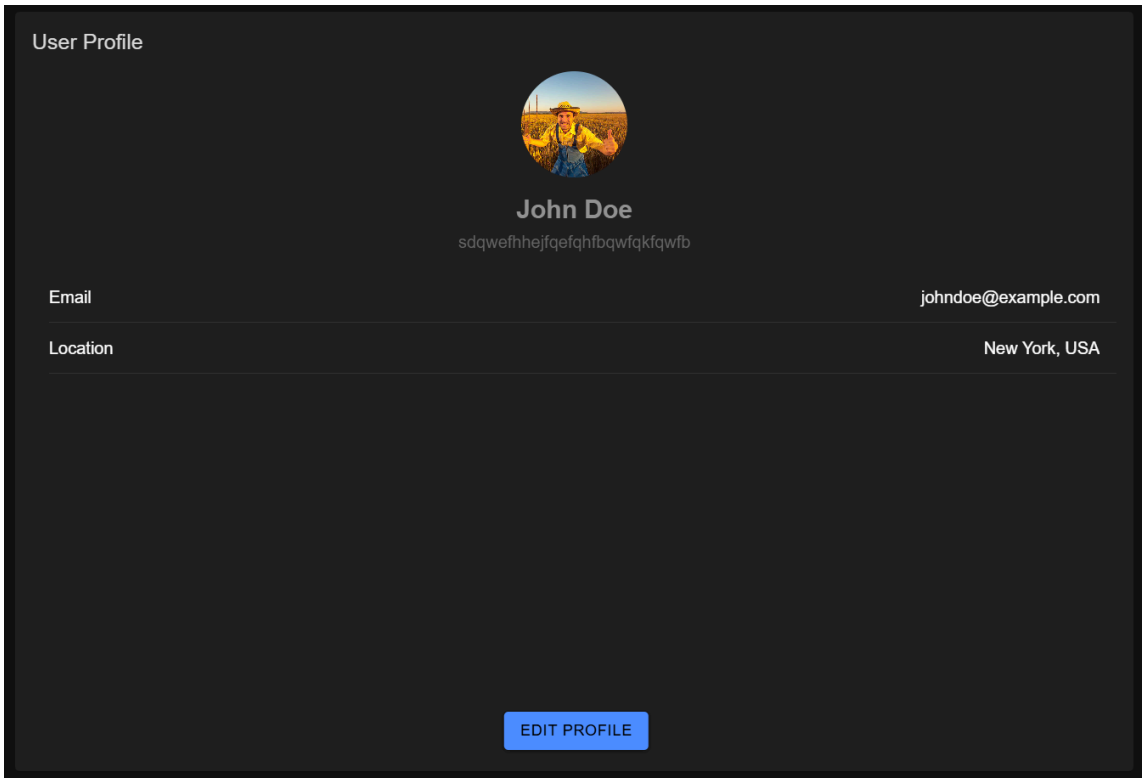

Fig. 14 (Profile Page)

Using the Ionic framework we made this page. It does not reflect the full design that we are going for but is enough to help other parts of the app be created.

13.4 Home Page/User Posts Page:

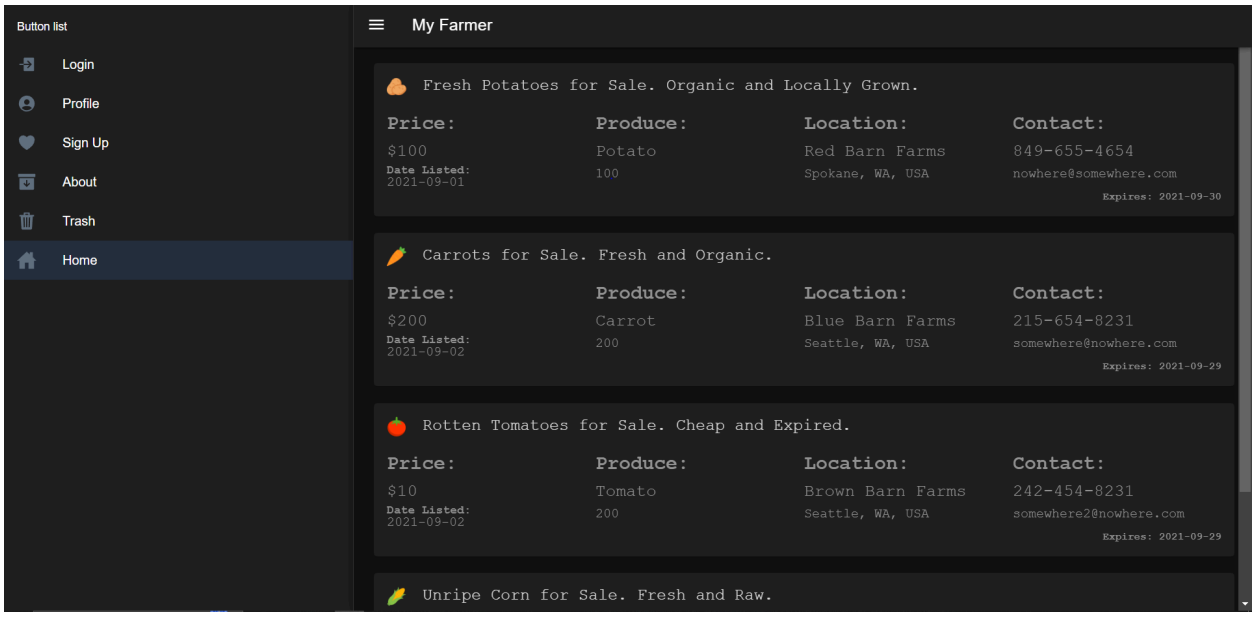

Fig. 15 (Home Page/User Posts Page)

This page is where we put most of our time into. We designed the posts to be easily readable and convey all the information needed. This design is the direction that we are heading into.

13.5 Sqlite Database in Ionic:

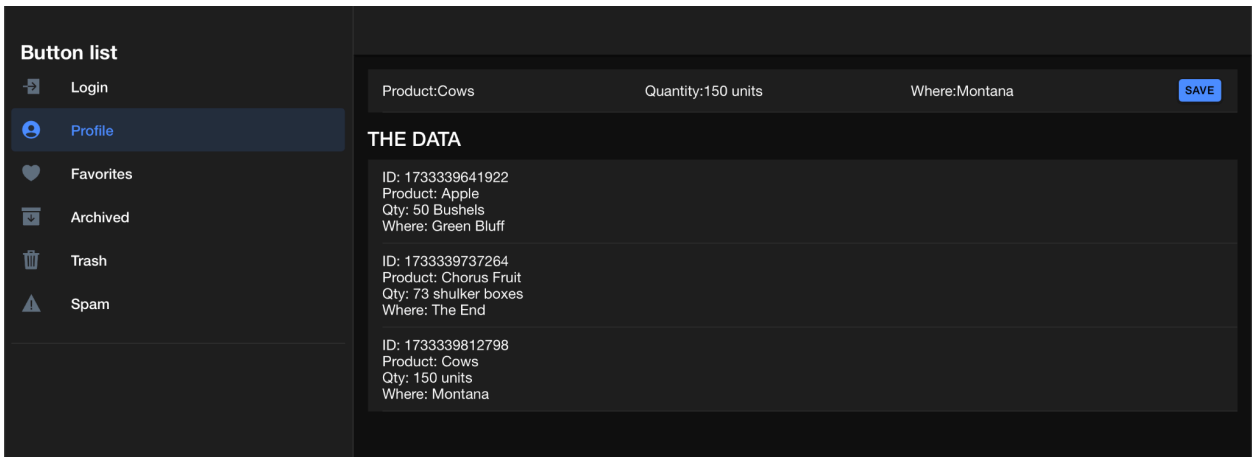

Fig. 16 (Mock Database UI)

This is the current mock database input and display that we have set up for testing the functionality of the database.

```
//do an insert on db
const addItem = async() => {
  try {

  //losad db
  await db.value?.open();
  //query db
  const respInsert = await db.value?.query(
    'INSERT INTO test7 (id,name,qty,loc) VALUES (?,?,?,?)',
    [Date.now(),inputName.value,inputQuantity.value,inputLocation.value]
    );
  console.log(`res: ${JSON.stringify(respInsert)}`);

  await db.value?.close();
  await loadData();

} catch (error) {
    alert((error as Error).message+"      ADD");
  }
};
```

```
onIonViewDidEnter(async () => {
  // validate the connection
  sqlite.value = new SQLiteConnection(CapacitorSQLite)
  const ret = await sqlite.value.checkConnectionsConsistency();
        const isConn = (await sqlite.value.isConnection("db_vite", false)).result;
        // let db = null;
        if (ret.result && isConn) {
          db.value = await sqlite.value.retrieveConnection("db_vite",false);
        } else {
          db.value  = await sqlite.value.createConnection("db_vite", false, "no-encryption", 1, false);
        }

    loadData();
});

// closing connection
onIonViewWillLeave(async() => {
  await sqlite.value?.closeConnection("db_vite",false);

});
```

```
        // minipulate the database
        await db.open();
        console.log(`db: db_vite opened`);
        const queryCreateTable = `
          CREATE TABLE IF NOT EXISTS test7 (
          id INTEGER PRIMARY KEY NOT NULL,
          name TEXT NOT NULL,
          qty TEXT NOT NULL,
          loc TEXT NOT NULL
         );
        `

        const respCT = await db.execute(queryCreateTable);
        console.log(`res: ${JSON.stringify(respCT)}`);

        await sqlite.closeConnection("db_vite",false);
```

```
// do a select on db
const loadData = async() => {
  try {
  //losad db
  await db.value?.open();
  //query db
  const respSelect = await db.value?.query('SELECT * FROM test7');
        console.log(`res: ${JSON.stringify(respSelect)}`);

  await db.value?.close();
  items.value = respSelect?.values;

  } catch (error) {
    alert((error as Error).message+"     LOAD");
  }

};
```

Fig. 17 (Database Functions)

These are the functions for, loading, validating, inserting, and creating the database. There is more code that goes into the setup of the database. The database is Sqlite using Capacitor in Ionic.

14 Testing and Validation

14.1 Test Plan

Our testing methodology focused on verifying the core functionalities of the app through end-to-end testing via Cypress. We determined that this early into development the key features that needed testing were navigation and page rendering.

14.2 Test Cases and Results

The tests focus on validating the navigation functionality of the app, ensuring that users can smoothly move between different pages. The App Navigation test suite includes the following

scenarios:

savefarmer > tests > e2e > specs > TS test.cy.ts > ...

```
describe('App Navigation', () => {
  it('loads the home page', () => {
    cy.visit('/');
    cy.contains('Button list');
  });

  it('navigates to Profile page', () => {
    cy.visit('/');
    cy.contains('Profile').click();
    cy.url().should('include', '/Profile');
  });

  it('navigates to Login page', () => {
    cy.visit('/');
    cy.contains('Login').click();
    cy.url().should('include', '/Login');
  });

  it('navigates to Home page', () => {
    cy.visit('/');
    cy.contains('Home').click();
    cy.url().should('include', '/Home');
  });
});
```

Fig. 18 (Test Cases)

1. **Loads the Home Page**: Verifies that the app's home page is accessible and the "Button list" header is rendered.
2. **Navigates to Profile Page**: Confirms that clicking the "Profile" button correctly redirects to the /Profile page.
3. **Navigates to Login Page**: Ensures that the "Login" button redirects to the /Login page as expected.
4. **Navigates to Home Page**: Checks that the "Home" button leads back to the /Home page.

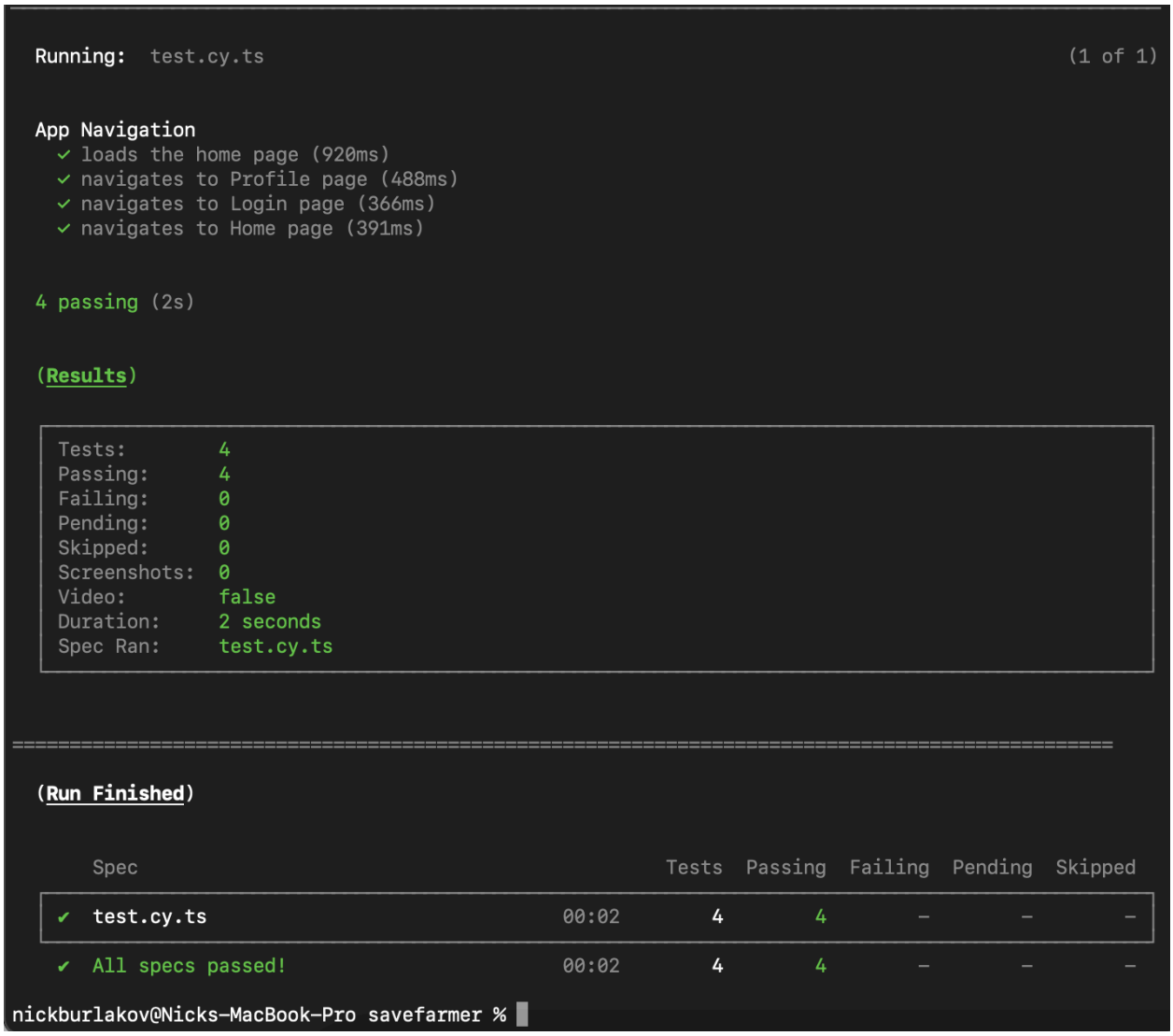

Fig. 19 (Successful Test Validation)

These tests confirm proper functionality of the navigation menu and ensure the app responds correctly to user interactions. If a test were to fail, Cypress would notify us of the failed case and generate a screenshot of the error from a user perspective, making debugging easier.

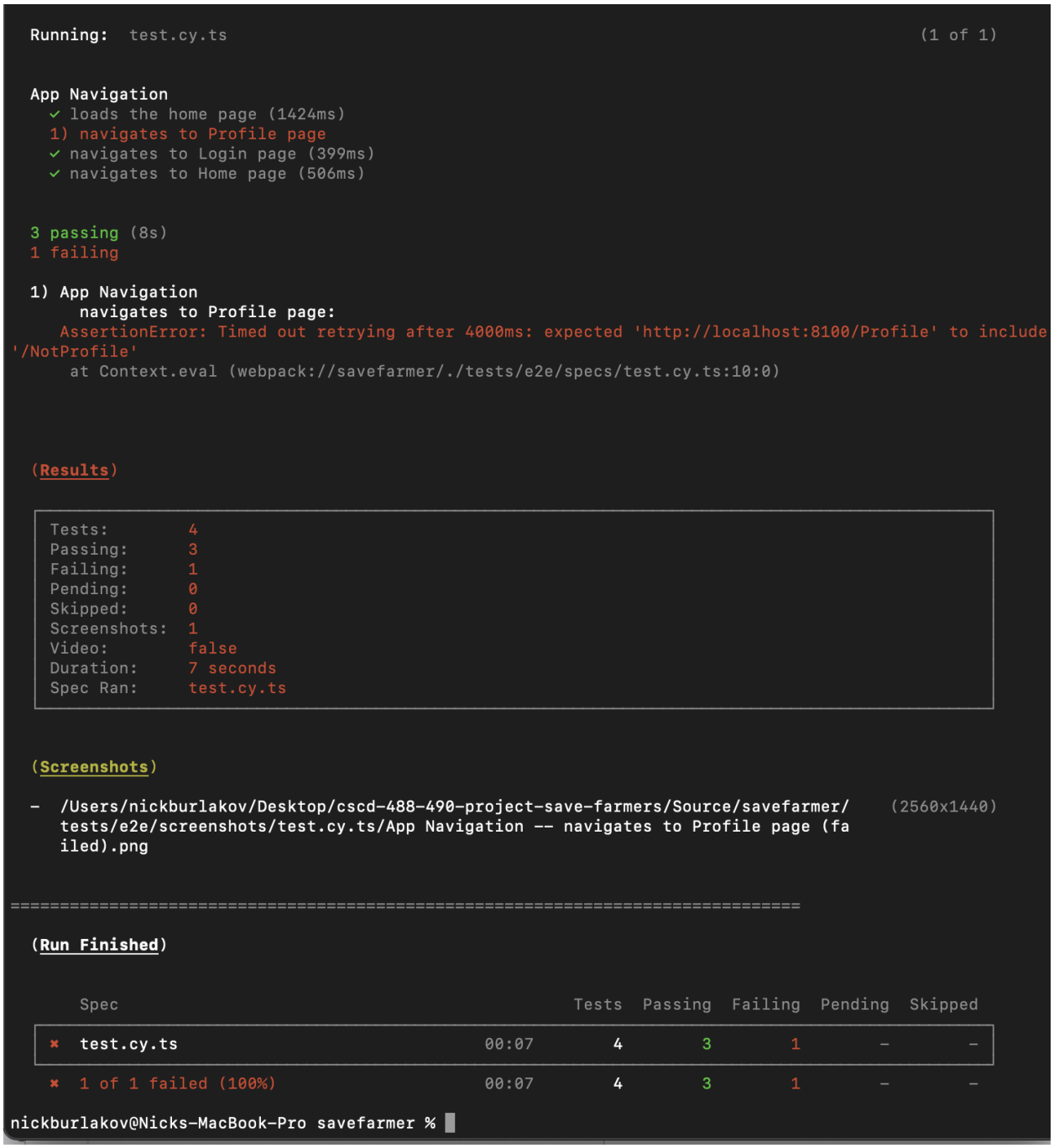

Fig. 20 (Failed Test Validation)

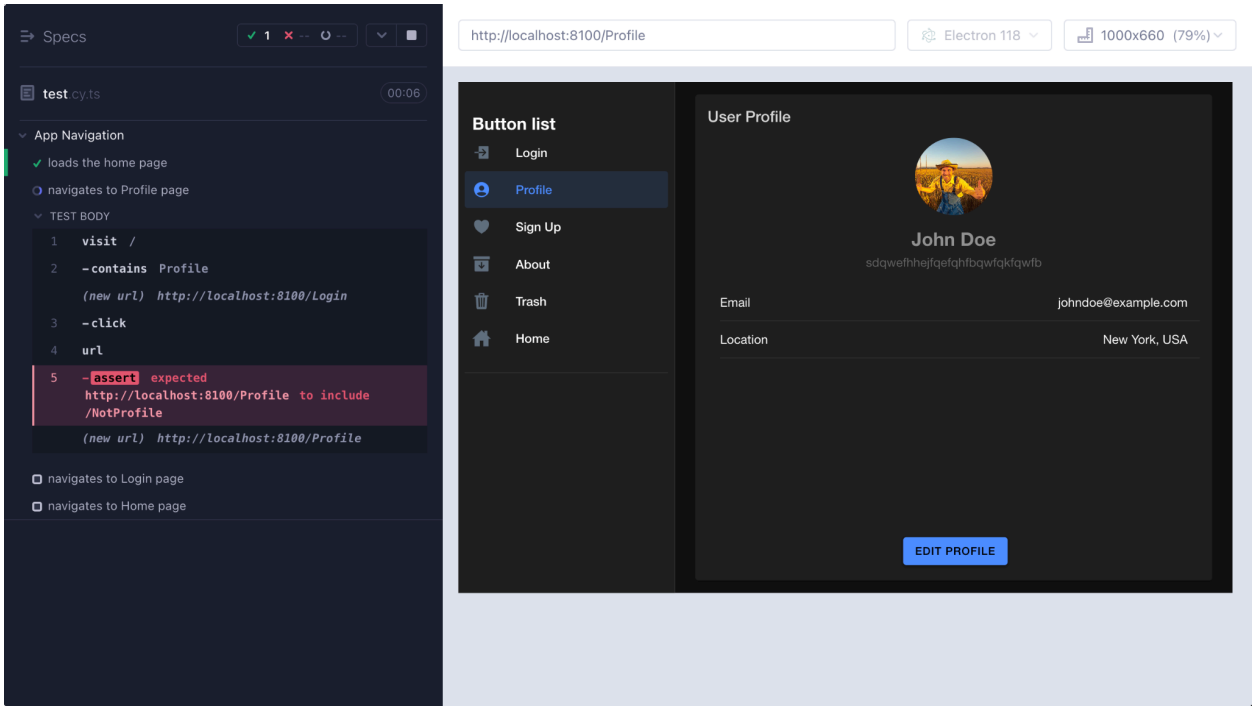

Fig. 21 (Failed Instance Snapshot)

The test results highlight how important automated testing is in ensuring a seamless user experience. By validating the navigation functionality of the app, we can confirm that key interactions work as intended. The failed test for the Profile page underscores the value of Cypress's detailed reporting and snapshot features, which make it easier to pinpoint and resolve issues. These tests provide confidence in the app's ability to handle user navigation effectively, while also emphasizing the need for ongoing testing to maintain functionality and user satisfaction.

15 Conclusion and Future Work

15.1 Summary

We are ending the quarter with the Save Farmers App in its early prototype development phase. Despite being in its infancy, the project has achieved significant milestones:

1. **Initial Framework Development:** A prototype skeleton has been created, focusing on core functionalities such as user registration, crop listing, and basic search capabilities.
2. **Stakeholder Collaboration:** Regular feedback from sponsors and potential users has informed the app's early design choices.
3. **Team Coordination:** The team has laid a solid foundation for scalable development.

Lessons Learned:

- Starting with a focused Minimum Viable Product ensures essential features are prioritized.
- Effective communication with and stakeholders is critical for aligning technical solutions with user needs.
- Early identification of technical constraints helps streamline the development process.

15.2 Future Enhancements

To expand the functionality and reach of our app, we can implement the following:

1. **Integrated Messaging System:** Adding an in-app chat feature to facilitate direct communication between buyers and sellers, reducing reliance on external channels.
2. **Payment Integration:** Enabling secure in-app payment options to streamline transactions and build trust among users.
3. **Rating and Review System:** Allowing users to rate and review sellers and buyers to enhance credibility and transparency.
4. **Localization Support:** Introducing multilingual support and region-specific settings to cater to a diverse user base.
5. **Advanced Analytics for Buyers and Sellers:** Offering insights such as popular crops, price trends, and buyer preferences to aid in decision-making.
6. **Mobile Notifications:** Adding real-time alerts for new listings, updates to favorites, and responses to posts.
7. **Community Features:** Introducing forums or groups where farmers and buyers can share advice, tips, and collaborate.

16. Individual Contributions

This section highlights the specific contributions of each team member to the *Save Farmers App* project.

- **Dillon Pikulik**
    - **Role:** Team Leader, Database, and Communications.
    - **Contributions:**
        - Communications with the team and sponsor.
        - Setup of the database.
        - Make (with the whole team), double-check, and turn in documents.

- **Nicholas Burlakov**
    - **Role:** UI Designer, Tester
    - **Contributions:**
        - Designed the initial user interface.
        - Conducted functional testing of the prototype to identify and resolve bugs in early development stages.
        - Provided feedback on usability and ensured the interface aligns with the user stories and target audience needs.

- **Eric Vo**
    - **Role:** Ionic Framework Frontend
    - **Contributions:**
        - UI implementation in the current prototype
        - Writing documentation
        - Designing database scheme
        - Setting up Ionic

Each team member played a pivotal role in shaping the foundation of the Save Farmers App. The collective effort has set the stage for further development and success.