Software Engineering
Software Requirements Specification (SRS) Document

Ingredient Intel

Mar – 26 – 2024
V.0.1.1
Joseph Cramer, Justin Evans, Pradhy Kothapalli

By adding our names to this paper we acknowledge that the Academic Integrity Policy governs our academic activities. WE HAVE ABIDED BY THE UNCG ACADEMIC POLICY ON THIS ASSIGNMENT.

Table of Contents

1. Introduction

1.1. Purpose

The purpose of ingredient intel is to reduce the difficulty in reading, understanding, and extracting information from ingredient lists on food products. By doing this we additionally hope to make consumers more knowledgeable about what they eat and more healthy because of that.

1.2. Document Conventions

The purpose of this Software Requirements Document (SRD) is to describe the client-side and developerview requirements for the Ingredient Intel web and android App. In it we will detail the requirements for a successful client experience and the requirements on the back end and for the developer-view in order for the service to run smoothly and require little technical knowledge to use. These requirements will include a description of the different user types and their levels of access and services provided to them. They will also include a system wide description of the software, database, and framework requirements and the metrics to determine software success such as performance, safety, and functionality.

1.3. Definitions, Acronyms, and Abbreviations

PostgreSQL	Open-source relational database management system.
.HTML	Hypertext Markup Language. This is the code that will be used to structure and design the web application and its content.
Django	A web framework that will be used with python to direct web traffic, interact with our database, and generally control website operations.
MVC	Model-View-Controller. This is the architectural pattern that will be used to implement our system.
VS Code	Our development environment where our code will be written for the website, backend, and database, and the interactions between them.
Python	An open source scripting language that will be used to write our application.
Android Studio	The android development platform that will allow us to develop and test our android application.
API	Application Programming Interface. We will use one to allow developer interaction with our database and will sync our database with the FDA's API for recalls, updates on ingredients, etc.

1.4. Intended Audience

The intended audience for the whole of this SRS document includes our development team, the UNCG CS department, the US FDA, and other interested computer scientists. The functional requirements and general description are intended for any users with companies and researchers as the main target.

1.5. Project Scope

The goal of the software is to provide a simple user interface to access a robust database which will provide a wide range of information on ingredients and products for companies, researchers, and normal users. This will benefit the goals of the government in providing transparency for safety information of ingredients reducing operational costs and difficulty of regulating food safety. Additionally, it will also serve as a service for consumers helping them stay informed about safety risks of ingredients, recalls of food items, and reduce the difficulty of avoiding certain ingredients that may be allergens or high risk for certain individuals.

Itemized the benefits are:

- For Business o Increased customer satisfaction o Reduced negative health impacts of products
    - o Ease of compliance with regulation and transparency with the government to obviate potential issues

- For Government o Reduced regulatory costs o Increased regulatory effectiveness o Increased consumer awareness about products and ingredients
    - o Fast reacting and simple system for communication with consumers and companies

- For Consumers o Time saving when shopping for dietary restrictions o Reduced confusion with product ingredients, health effects, and recalls o Increased ability for people with dietary restrictions to try new products

- For Researchers o Increased ability for new findings to be communicated to consumers o Simplified systematic process for updating food safety information o Simple access to previous research about food ingredients
    - o Simple access to product database which would allow for better data for correlation or impact studies

1.6. Technology Challenges

There will not be any challenges on the hardware technology side of this product. Some limitations may be put in place to increase access for people have limited access to modern cell phones.

1.7. References

Alred, F., Brusaw, C., and Oliu, W. (2003). Handbook of Technical Writing (7th ed.). Boston: Bedford/St. Martin’s.

FDA API. Open.fda.gov. (n.d.). https://open.fda.gov/apis/

Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.

2. General Description

2.1. Product Features

Ingredient Intel (INI), a revolutionary app for Android and web, empowers you to make informed food choices. Scan any product's barcode to access clear ingredient breakdowns, potential allergens, and health effects. Set dietary preferences and get alerts for unwanted ingredients. Companies benefit too, with easy ingredient management, recall communication, and transparency tools. Research institutions also play a role with access to contribute scientific notes, keeping consumers informed about the latest health findings. With the database connected to the FDA's API, Ingredient Intel prioritizes safety by automatically flagging recalled products.

- Barcode scanning feature to get nutritional facts(get)
- Searching items by ingredients and by the actual product(get)
- Researcher feature to add and remove information on ingredients as new research is done(post)
- Company feature add products and the ingredients in the said product

Some of the products features will include barcode scanning capabilities and search bar implementation for looking into ingredients. Another feature will be the ability for researchers to change ingredients disclaimers on unwanted ingredients as new research comes out. Companies will also be able to add their products to database with their ingredients also included.

2.2. User Class and Characteristics

Our application is intended to be accessible to anyone with access to the internet and our app is designed to provide access to any with an android. Beyond that only basic internet operation skills and the ability to read English are needed.

For researchers and companies a slightly more in depth understanding of computers is required to set up an account and for some feature a basic understanding of SQL will be useful though not necessary for any core features.

2.3. Operating Environment

Ingredient Intel is being planned to be on Android Studios for item scanning as well as a website for searching specific ingredients.

2.4. Constraints

The biggest challenge of making this software will be generating the ingredients database and scaling as businesses add products. Additionally getting widespread adoption with companies will be time taking and require legislative action.

2.5. Assumptions and Dependencies

The software will be dependent upon Android Studios as well as Django to create and execute the MVC in VSCode, software will also be dependent on the FDA's public API for product recalls (RES) and on PostgreSQL open source database.

3. Functional Requirements

3.1. Primary

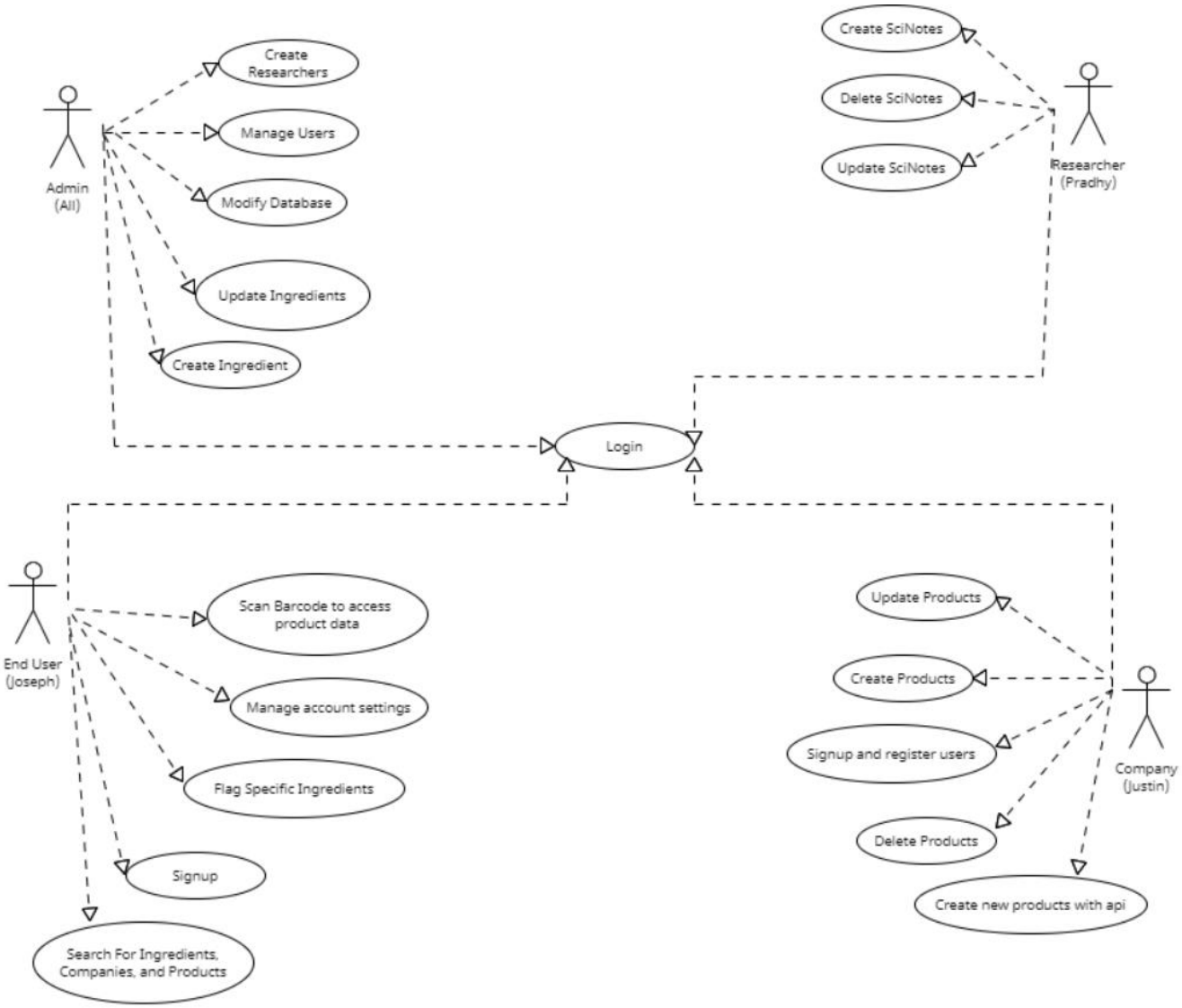
- FR0: The system should be able to scan barcodes accurately using the device's camera. It should then be able to decode the information and retrieve the details associated with the product.
- FR1: The system should maintain a database of ingredients, and support functionality for adding, updating and deleting information as needed. The database of ingredients includes names, categories, potential health effects and allergen information.
- FR2: The system should be able to send alerts and notifications regarding product recalls, and other relevant updates. It should also allow users to customize their alerts based on dietary restrictions, allergens, etc.

3.2. Secondary

- Develop tools to categorize ingredients into groups to make it easier for users to find.
- Integrate with external APIs to enhance accuracy and send alert notifications to users promptly.
- Ensure there is an error handling mechanism to detect issues related to barcode scanning, DB operations, etc.

3.3. Use-Case Model

3.3.1. Use-Case Model Diagram



3.3.2. Use-Case Model Descriptions

3.3.2.1. Actor: Super User(All)

- Modify database infrastructure: super user has the ability to delete, modify, or create anything within the database including to modify the database servers layout, software.
- Modify Front End: Super users will have the ability to directly modify the code behind the app to change functionality or appearance.
- Create Research API Keys: Super users will be able to generate API keys for researchers and research institutions.

3.3.2.2. Actor: End User(Joseph)

- Scan Barcode: The user can upload the ingredients to the DB by scanning the barcode through the app.
- Receive Alerts: If there is a recall on a specific product, then the user gets an alert with more information.

- Search Database: By using a simple search feature within the app or the website users can learn more about products or ingredients.

3.3.2.3. Actor: Company (Justin)

- Modify ingredient list: If there is a change in the list of ingredient for a product, The company can modify the ingredient list.
- Communicate recall: If the company identifies a product that needs to be recalled, the can communicate that with the users.

3.3.2.4. Actor: Researcher(Pradhy)

- Publish research notes: Researchers can publish scientific notes regarding specific ingredient. For example: Potential health effects and other concerns.
- Ingredient analysis: Researchers can conduct analysis on specific ingredients or the combination or ingredients and share their findings with users through the app. For example if an item was found to increase risk of cancer, they could add that warning onto the ingredient.

3.3.3. Use-Case Model Scenarios

3.3.3.1. Actor: Super User (All)

- Use-Case Name: Modify Database
  - Initial Assumption: The super user has the super user role and access to the login for the database.
  - Normal: The super user will log into the database and correct any errors or issues with the data.
  - What Can Go Wrong: n/a
  - Other Activities: Data can be copied, deleted, modified. The system can be shut down or a new system created. Users can be removed, permissions modified, etc.
  - System State on Completion: The database is updated and any issues are resolved on the back end and also changes propagate and can be viewed by the end user.

- Use-Case Name: Create New API Key
  - Initial Assumption: The superuser has access to the superuser role in the database and access to the admin API page.
  - Normal: The superuser logs into the admin API page and clicks the button the create a new research API key, they are then prompted to enter superuser authentication for the database and the API key is added to the researcher's profile.
  - What Can Go Wrong: Incorrectly entered passwords could cause setup to fail.
  - Other Activities: The superuser may also delete research API keys by searching for the name and using the delete button, confirming with password.
  - System State on Completion: The profile of the researcher or research institution will now have the API key within their profile.

- Use-Case Name: Modify Front End

- Initial Assumption: Superuser has modify access to codebase.
- Normal: Superuser logs into codebase and makes edits to the code, either saving them to a branch while in development or propagating changes to the system.
- What Can Go Wrong: Errors in the code could cause a system crash, accidental propagation of incomplete code could cause unexpected behavior.
- Other Activities: Changes in code can be rolled back.
- System State on Completion: Either updated system wide or new code stored in off-main branch.

3.3.3.2. Actor: End user (Joseph)

- Use-Case Name: Scanning barcodes
  - Initial Assumption: The user opens the app and scans the barcode.
  - Normal: The app gets the list of ingredients and if it is not already in the database, then adds it.
  - What Can Go Wrong: Bad lighting can cause the barcode to not be scanned properly.
  - Other Activities: The user can search the database manually for the product. If the product is not in the database, then the user can add it manually.
  - System State on Completion: The app successfully retrieves or updates the information from the database and displays the list of ingredients for the product.

- Use-Case Name: Create special search and notification settings
  - Initial Assumption: The user created an account and is logged in
  - Normal: The user selects any dietary restrictions they might have and saves the changes.
  - What Can Go Wrong: Due to a glitch or other issues, the user could be prevented from saving changes to their dietary restrictions. This would cause the user to get false information and could be dangerous.
  - Other Activities: The app could provide recommendation or alerts based on their dietary preferences.
  - System State on Completion: The user's preferences are saved and applied to their account successfully.

3.3.3.3. Actor: Research Institution (Pradhy)

- Use-Case Name: Access/modify existing research studies
  - Initial Assumption: The researcher signs in and navigates to the research dashboard.
  - Normal: The researcher selects create a new scientific note and enters relevant info to create new note.
  - What Can Go Wrong: The form is filled out incorrectly or the research study itself has inaccurate or outdated information. Additionally, any links used as citations may reference pages that are no longer active.
  - Other Activities: The researcher can view and modify their existing notes.
  - System State on Completion: The researcher successfully accesses/modifies the research study and the information is updated in the database.

- Use-Case Name: Ingredient analysis
    - Initial Assumption: The researcher navigates to the analysis dashboard on the web-platform.
    - Normal: The researcher uses the SQL query tool to extract information about products, ingredients, or companies from the database.
    - What Can Go Wrong: Malformed SQL queries could fail or in some extreme case researcher may cause unauthorized changes to the database.
    - Other Activities: If the query tool does not have functionality for the specific query the researcher would like to perform direct SQL code can be ran through the interface.
    - System State on Completion: Database information has been extracted without modification and a file download of the results has been sent to the researcher.

3.3.3.4. Company(Justin)

- Use-Case Name: Update product information
    - Initial Assumption: An employee at the company accesses the web platform and updates the ingredient information.
    - Normal: The employee logs in, and updates the product as needed.
    - What Can Go Wrong: The company employee enters incorrect/misleading information and uploads it to the database.
    - Other Activities: The employee should add an image or pdf of the new product label or other nutritional content.
    - System State on Completion: The ingredient information is successfully updated in the database.

- Use-Case Name: Communicate ingredient or product recall
    - Initial Assumption: The company representative logs in to the Ingredient Intel platform.
    - Normal: The representative navigates to the recall management section. The representative constructs the notification and sends it to the consumers.
    - What Can Go Wrong: The user does not have the notifications on or does not get notifications. The effected products should not be shown until the issue is solved.
    - Other Activities: The platform could track user responses/acknowledgement to the recall notification to make sure the user is aware.
    - System State on Completion: The recall notification is successfully communicated to users through all the Ingredient Intel platforms.

4. Technical Requirements

4.1. Interface Requirements

4.1.1. User Interfaces

The user will connect via login, after that the user can either type in ingredients to look up in our database via a search bar, or click a camera option so they can scan the bar code of an item, bringing them to its nutrition facts, recall info, etc.

4.1.2. Hardware Interfaces

In all instances the software will need internet to function. The device has to be able to fully interact with our webpage. Devices that can use the service will be android products, as well as most computers. In order to use the barcode scanning feature users will need to have the app downloaded and have access to a functioning camera.

4.1.3. Communications Interfaces

The software will need to be communicating with the internet, as well as PostgreSQL as the local database, HTTP, and lastly the open FDA API that we are using for recall and food safety information.

4.1.4. Software Interfaces

On the backend we will be using PostgreSQL for our database, and we will be using Django with python to connect our front end and back end along with hosting and controlling our website including using some functionality for designing the frontend.

5. Non-Functional Requirements

5.1. Performance Requirements

- NFR0(R): The global database should not use more than 1 TB of memory.
- NFR1(R): The rest of the app including the database should not exceed the use of 150 MB of memory.
- NFR2(R): New users will be able to easily scan a barcode and get nutrition facts in less than 5 minutes.
- NFR3(R): Experienced users can scan a bar code or look up items by ingredients in less than 1 minute.

5.2. Safety Requirements

Research studies listed for specific ingredients will include disclaimers when necessary to prevent consumer panic.

5.3. Security Requirements

- NFR4(R): All user data will be encrypted and stored locally.
- NFR5(R): The system will have a sign in feature to verify users.
- NFR6(R): Database write access will be limited with unique API keys.

5.4. Software Quality Attributes

5.4.1. Availability

The app will be free to use, meaning that anyone can use the app as they see fit. The app will only be available on Android for the time being but a limited version may also be accessed from the web.

5.4.2. Correctness

In general the onus of correctness falls on the companies and researchers. By restricting access or with fines companies or researchers will be required to provide accurate information. Users will be able to report any potential inaccuracies that can then be manually checked for accuracy.

5.4.3. Maintainability

Researchers and companies will update information about ingredients and products respectively. Our team would handle any bug fixes or updates needed along with managing data storage and service availability.

5.4.4. Usability

Software will maintain reasonable loading and response times and will include features for accessibility like high contrast or magnification.

5.4.5. Portability

The application will be available on android, making it very portable and easy to use so long as there is access to the internet which is necessary for accessing the database.

5.5. Process Requirements

5.5.1. Development Process Used

We are utilizing the waterfall development process.

5.5.2. Time Constraints

The main time constraint is our delivery date, as well as certain milestones we must reach along the way. Specifically:

- Feb-27-2024 Product prototype completion
- Apr-30-2024 Project completion

5.5.3. Cost and Delivery Date

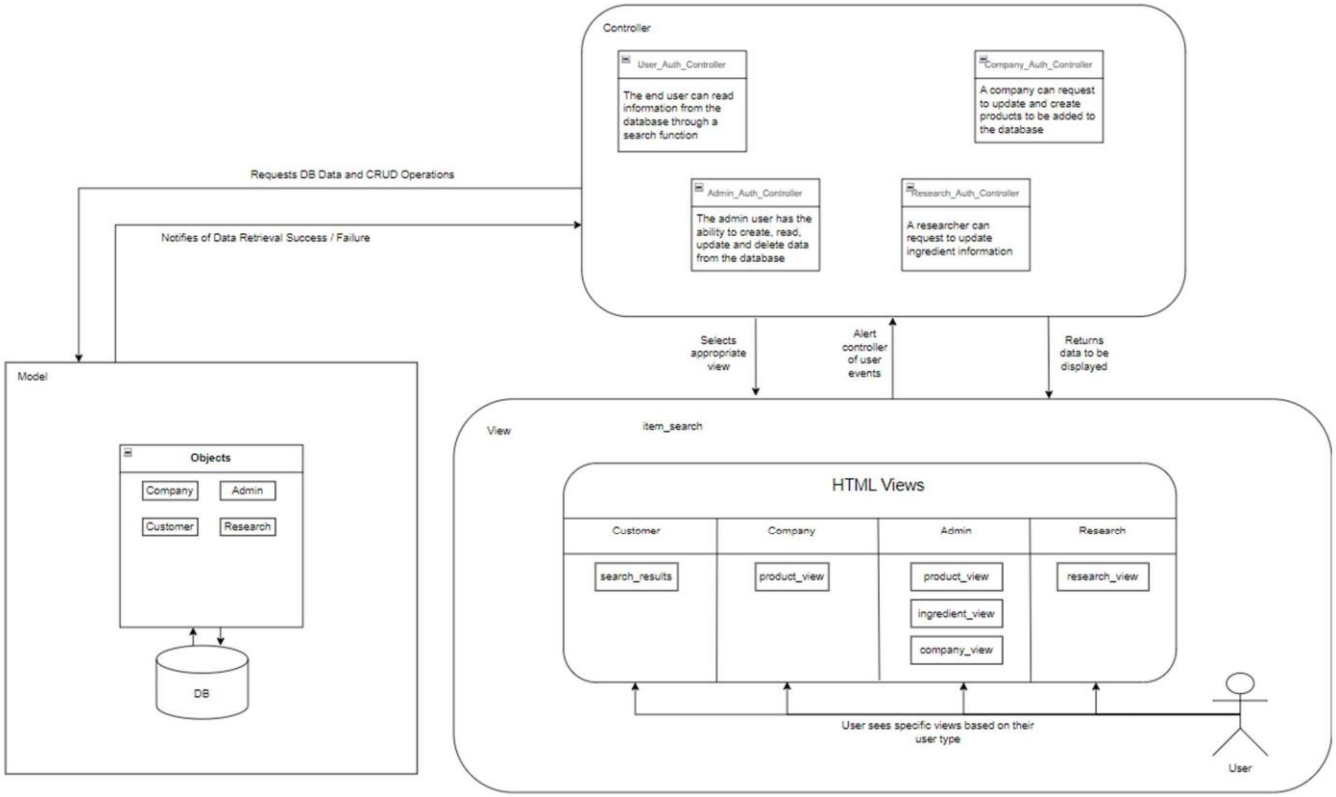
The delivery date is planned for later April 2024. Expected cost is $15,000 in work time, and a hosting cost that will vary based off server costs changing rapidly currently due to AI development but we estimate a hosting cost of $20,000/year when running at full capacity. Maintenance costs should be fairly low and will contained within the purview of the existing FDA contractors.

5.6. Other Requirements

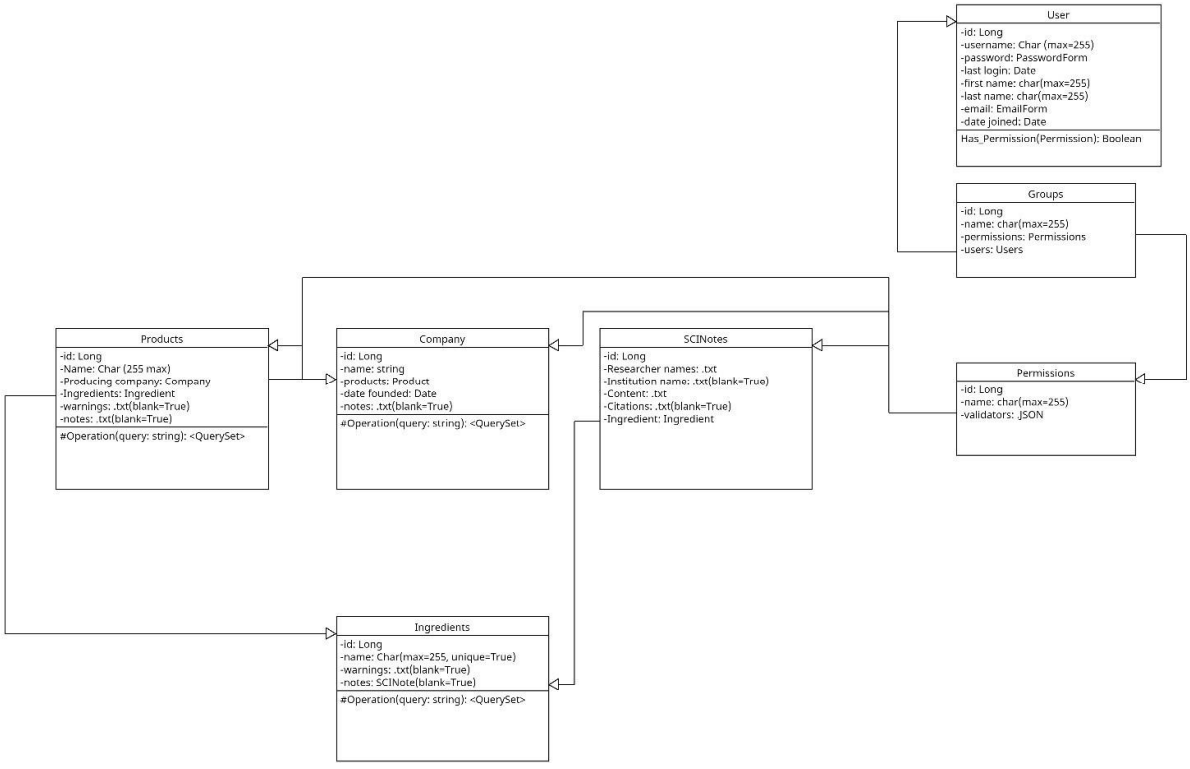
Post product development integration with FDA services and additional transition work to be done after approval by FDA.

6. Design Documents

6.1. Software Architecture

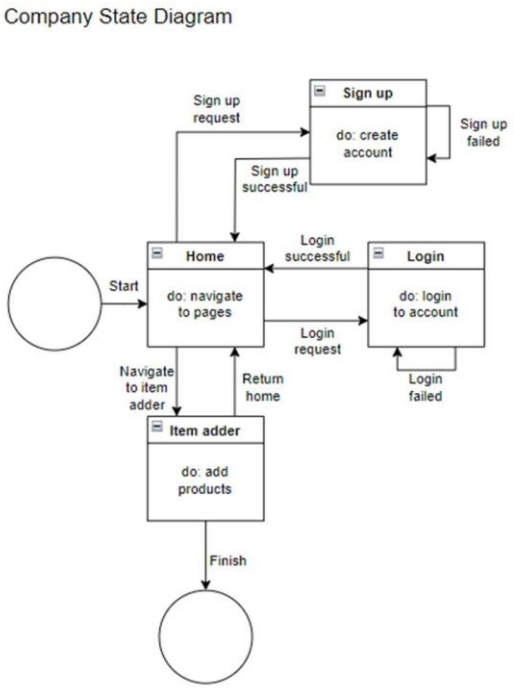


6.2. High-Level Database Schema



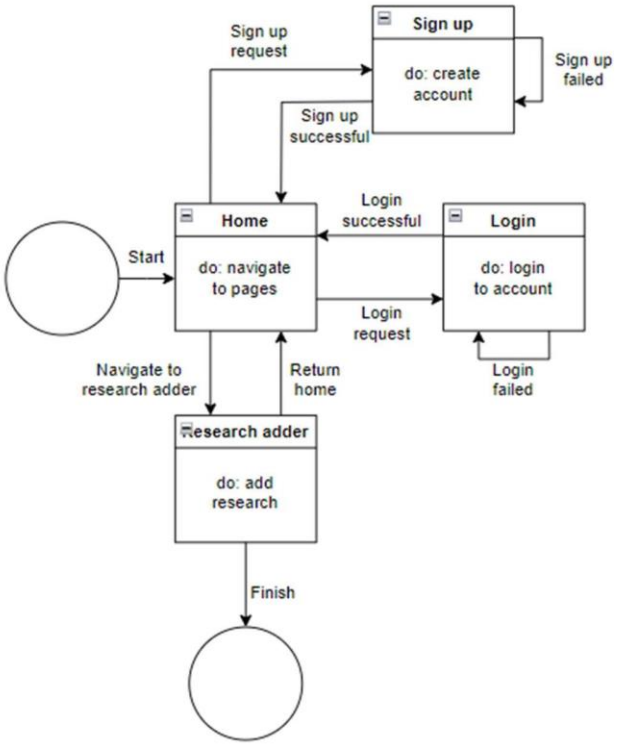
6.3. Software Design

6.3.1. State Machine Diagram: Company (Justin Evans)



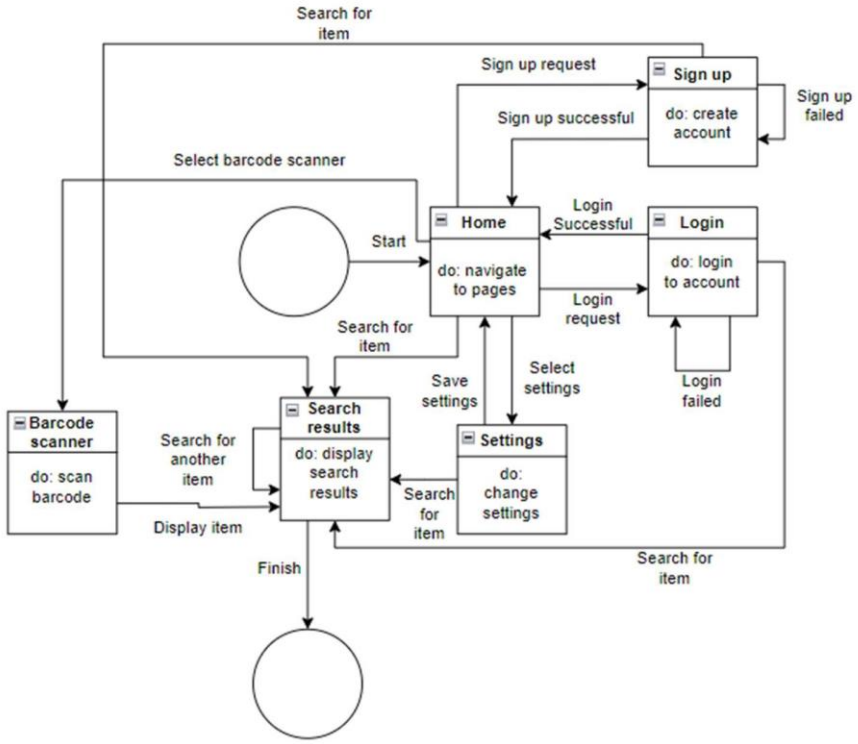
6.3.2. State Machine Diagram: Researcher (Justin Evans)

Researcher State Diagram

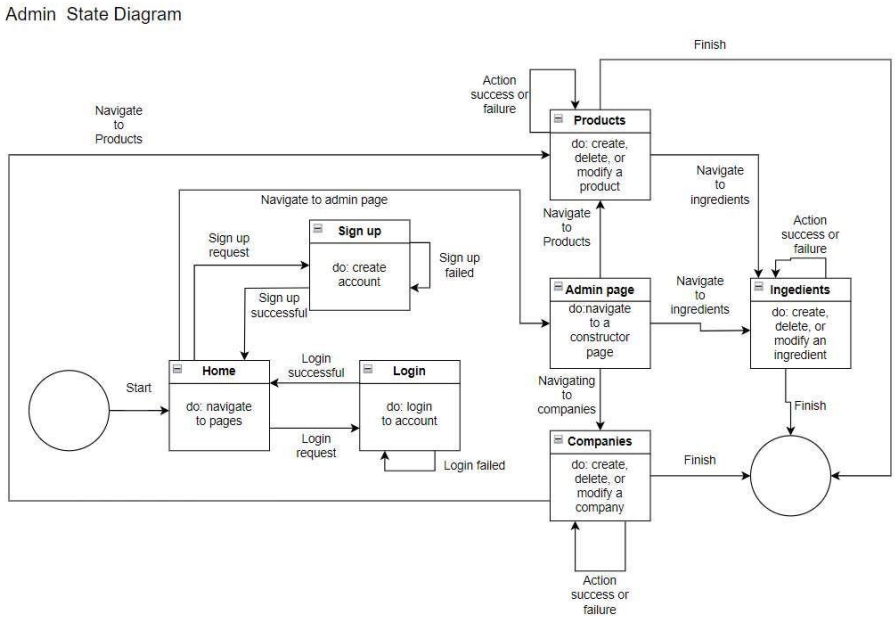


6.3.3. State Machine Diagram: Customer (Justin Evans)

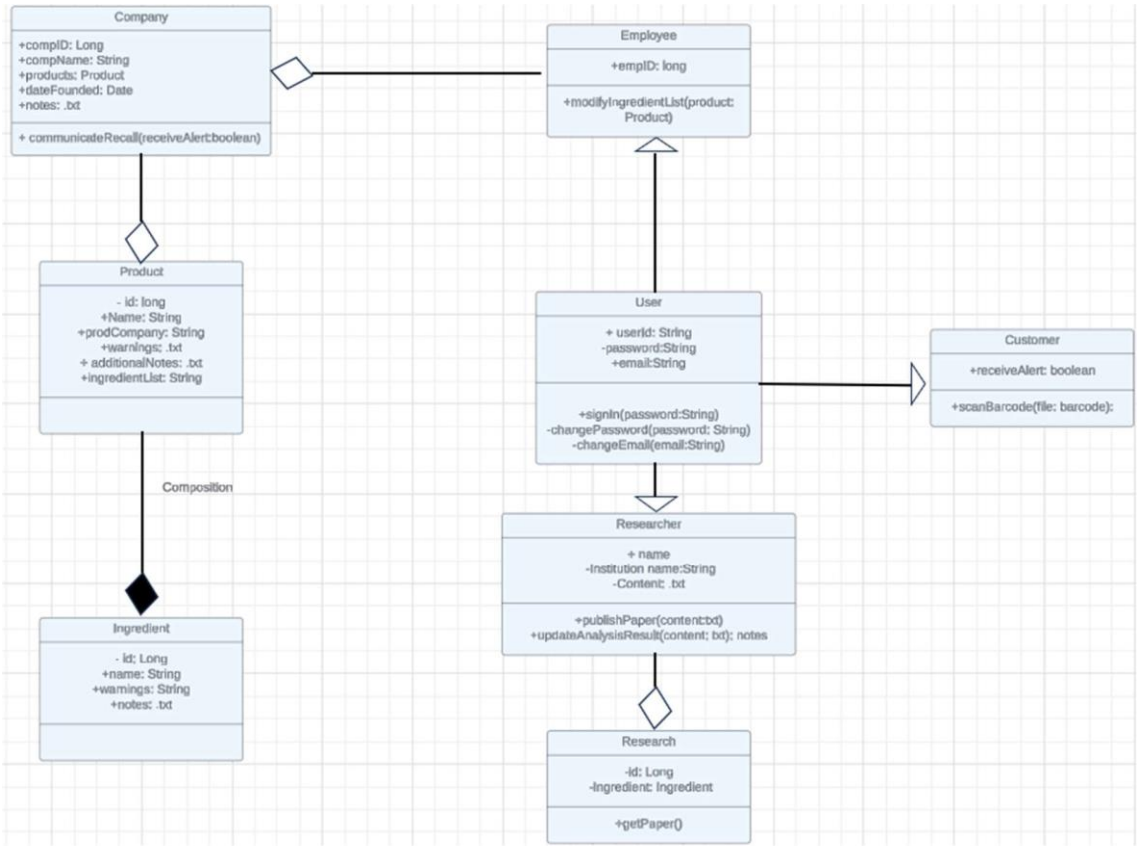
Customer State Diagram



6.4.4. State Machine Diagram: Admin (Justin Evans)



6.4. UML Class Diagram



7. Scenario

7.1. Brief Written Scenario with Screenshots