Deliverable 5

Final Project Report

Control Alt Defeat
T13A

Software Engineering
Design Workshop

Table of Contents

Overview

Hare is an app that allows a user to take a picture (or upload one) of a packaged products' ingredients label and obtain useful insights and information into the contents of the item. It targets users who want or need to be particularly aware of what's inside products around them (for example, those with allergies). As the general market applies to many if not all consumers, we aim for the application to be very user friendly and provide a great user experience. In Deliverable 1 to 4, we demonstrated how this can be designed, implemented, and demonstrated a working prototype.

Revision History

24/04/2019	0.5	Draft
26/04/2019	1.0	Submission

Group Members

z5161502	Michael Lloyd
z5160635	Negar Bolourchi
z5182793	Ravija Maheshwari
z5158357	Julie Duan
z5207123	Jack Callander

1 Scope & Specification

The scope of our application is limited to the breadth of the course and the resources the group had availalbe during design and development. Our application's scope is small, and specifically limits itself as a useful tool to provide information about consumer product packaging for the general consumer. This section clearly outlines the features our application provides, and some user stories to provide insight into how the features are available in normal usage.

Features

1) Guests and members can upload picture of a packaged good and obtain a list of ingredients
2) Guests and members can upload pictures of foreign goods, or packaged goods in a foreign language, and obtain an ingredients list in English.
3) Guests and members can obtain descriptions and brief overviews of the ingredients in a list from a packaged product.
4) Ingredients obtained from packaged goods must be categorised such that a user is able to infer more information about how they are used, or their purpose in the product.
5) Members can filter categories of ingredients for highlighting.
6) Members can highlight certain ingredients in packaged products when obtaining the ingredients list.
7) Members can save ingredients lists from a packaged product somewhere for later viewing.
8) Members can save and load their preferences/filters, highlighting and ingredient lists.
9) Members can infer serious health implications that may result from normal consumption of a listed ingredient.
10) Members can to search for allergens or ingredients that may have a specific health implication for them.
11) A guest can register and log in to become a member and a member can log out to become a guest.

User Stories

User Story 1

Feature: Upload picture of packaging (Feature 1)
As a Guest
So that I can find information about ingredients in the product
I want to upload a picture of the ingredients on a product
Scenario: Take a picture, or upload an image of an ingredients list to receive information about each ingredient GIVEN: I am on the default landing page
AND: I have not logged in
WHEN: I click the scan image icon
THEN: I am on the upload page with camera preview
WHEN: I take a picture of the product ingredients
AND: I click submit
THEN: I see a page with details about the ingredients in the picture

User Story 2

Feature: Scan packaged item with Camera (Feature 2, Feature 10)
As a Member
So that I can view categorised information about the usage or purpose of ingredients in a product.
Scenario: Take a picture of an ingredients list to receive information about each ingredient
GIVEN: I am on the default landing page
AND: I am logged in
WHEN: I click the scan image icon
THEN: I should be on the upload page with a camera preview
WHEN: I take a picture of the product ingredient details
AND: I click submit
THEN: I see a page with details about the ingredients in the picture

User Story 3

Feature: Log In (Feature 11)
As a Member
So that I can save my previous scans, enable ingredient highlighting, and compare multiple products.
Scenario: Log In
GIVEN: I am on any page within the web app
AND: I am not logged in
WHEN: I click the 'Log In' button
THEN: I should be on a Login page
WHEN: I correctly enter my credentials
THEN: I should be on the user preferences page, logged in

User Story 4

Feature: Log Out (Feature 11)
As a Member
So that I can return to the default landing page of the web app.
I want to log out of the app
Scenario: Log Out
GIVEN: I am on any page within the web app
AND: I am logged in
WHEN: I click the 'Logout' button
THEN: I am taken to the default landing page and am no longer logged in

User Story 5

Feature: Select pre-set highlighting preferences for ingredients (Feature 4, Feature 6)
As a Member
So that I can easily identify ingredients of common preferences
I want to choose a pre-set to highlight these ingredients
Scenario: Select a pre-set in my ingredient's preferences
GIVEN: I am logged in
AND: I am on the user settings page
WHEN: I scroll to 'Highlighting Preferences'
AND: click on any of the sample preference settings
THEN: Future ingredient details will display selected ingredients with custom colour using a pre-set

User Story 6

Feature: Define custom highlighting rules for viewing ingredient (Feature 5, Feature 8)
As a Member
So that I can easily identify ingredients
I want to create a pre-set to highlight chosen ingredients or ingredient categories
Scenario: Create a highlighting rule set in my preferences
GIVEN: I am logged in
AND: I am on the user settings page
WHEN: I scroll to 'Highlighting Preferences'
AND: click 'Customise'
WHEN: I choose my ingredient and colour
THEN: Future ingredient details will display selected ingredients

User Story 7

Feature: Save ingredient lists for later viewing (Feature 7)
As a Member
So that I can view the ingredients of saved items easily
I want to have easy access to frequently bought products
Scenario: I want to check the ingredients of an item I've previously scanned
GIVEN: I am logged in
AND: I am on the user homepage
THEN: I should see a list of previously scanned items
AND: I have previously saved the ingredients of the item
WHEN: I click on my saved item list
THEN: I see the ingredients of item in the list I click on

External Data Sources

Our application requires a number of external sources of information for the application to work without large development on our part. Namely, we need to be able to recognize ingredients from an image, and then get information about those ingredients to pass back to the user. These sources are:

OpenFDA's Database API [1]
The OpenFDA database API is an easy access public database with information ranging from drug datasets to packaged product codes and label information. The database is maintained by the Food and Drug Administration (FDA) in the United States which has a credible reputation in regulating product packaging and labels. For this reason, we rely on the dataset of ingredients and substances listed to be possible in packaged foods, interacting with it inside of our own database, and occasionally re-referencing the OpenFDA API.

This source is particularly important, because it provides us with a primary reference point for how to interpret the ingredients on food labels, as well as providing us we very accessible lists of the categories and types that may exist.

DrugBank.ca Database API[2]
The Canada-based DrugBank API was used to complete our ingredient and substance data-sets, as this particular database offers several query terms that OpenFDA does not. Notably, there is a large number of synonyms and descriptions for substances contained in the OpenFDA database that OpenFDA does not provide. This means that the Canadian DrugBank API is essential for completing substance profiles that are sent to the user.

MediaWiki Official Wikipedia API[3]
We have included the official Wikipedia API as a supporting source of information for common ingredients and substances in packaged products. We also intend to reference where we get information directly to the user, so that they can follow a link to see more themselves. This should greatly improve the user experience as it allows us to cover all the most common substances they are likely to find in products with well written descriptions and outgoing links.

Google Vision Optical Character Recognition API[4]
Google Vision offers an Optical Character Recognition (OCR) API that can be used to return interpreted text in an image through only a REST API. It has access to several other Google Cloud services (such as Google AutoML) that allow for the categorization and parsing of the information once it is read, and returns a JSON file as a response to the caller with information ranging from different lines read and the x, and y coordinates for which it read them.

This offers some great advantages to our project over directly running an OCR service inside of our back-end or use another service as this particular service is well maintained, optimized and efficient. We have tested latency, character accuracy, and a variety of different lighting conditions with this API and concluded it to be superior to many alternatives available.

Google Languages Translation API[5]
The Translation API provides a simple programmatic interface for translating an arbitrary string into any supported language on Google. Language detection is also available in cases where the source language is unknown. We observe extremely low latency when using this API, like with Google Vision's OCR, and consider it necessary to meet one of our primary problem statements – allowing people to see the ingredients in a product when the packaging is in a foreign language.

2 Technology & Implementation

Software Stack

Our front end has a number of requirements we deem desirable for optimal use. Namely: the Front-End needs to support cross-compatibility among devices, from using a computer's browser to iOS and Android running applications. The Front-End should be modular enough to enable development to happen in an appropriate time-frame, and it should handle some level of programmatic responsiveness.

On the other end, the Front-End does not need to contain any logic involved with OCR, parsing, database handling or making external API requests.

ReactJS[6]
React, commonly misinterpreted as a framework, takes many of the roles that a traditional Front-End framework does. It serves as a library for extending upon several of its base class components, and then is often wrapped with a number of bundling and pre-processors for the code it's written with to provide an actual Frameworks' features.

A large advantage of React is that it supports programmatic events and responses directly linked to the components placed on the page. This means that routing and event management can be directly inside of the JSX components and does not need to be directed to an events container or dispatcher (used as a personal library for handling the interactions of users). It is also written for JavaScript, which the development team is familiar with – making it attractive as a framework.

As React is completely independent of the source it is served from, and bundles/processes all of the scripts necessary to run the page before hosting, we can completely decouple the Front-End framework from the Back-End framework, and bridge communication between the two with a custom REST API. This results in lower coupling but does not affect the level of cohesion that architecture has during development.

Babel/JSX Pre-processing[7]
Babel is a pre-processor for JavaScript that allows a variant of code called "JavaScript X" or "JSX" to be written. This comes bundled into the default React standard but is required for writing higher order JavaScript to HTML. It allows components to be directly declared as pseudo-HTML inside of JavaScript, then upon pre-processing compiled to both HTML and JavaScript separately.

UI element libraries[8]
Alongside Babel, we used Grommet (version 2) as a library for pre-written JSX components. As Grommet is standardized, offers icon libraries and a variety of maintained and well managed event handling functions, we considered this ideal for building the website. It allows for rapid prototyping, similar to its' counterparts such as Bootstrap, or Material Design.

Express Framework for NodeJS
After decoupling our Front-End from our Back-End, focus can be given on how the logic, data storage, external API access, and view generation should be managed. We have identified that there should be 4 primary sections of this:

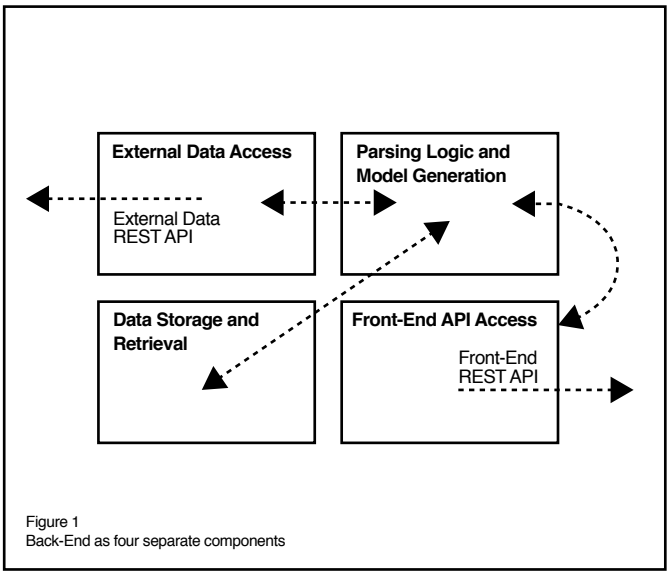


Figure 1
Back-End as four separate components

We expect that the Controller will manage requests from the Front-End and to external API's, and that the Model will collate and generate a page's necessary information container then have the Controller deliver it. The Database should work alongside the Model to collate information that is necessary for the App to run (namely, user profiles and authentication details) as well as assisting information for events on the websites (product pages, ingredient lists and substance details)

NodeJS[9]
To assist the development team, and to maintain cohesion on the language-level of the stack, we opted to use JavaScript to create the Back-End's logic, meaning that we inevitably chose NodeJS, a runtime environment for JavaScript outside of the Browser, to host the server on. Node allows for elaborate asynchronous architecture to exist and can become particularly useful in small web-applications.

It can measurably be shown to work well with applications such as Hare, and scalable upon use[10]. While the technical details of running a single-thread environment may seem limiting, they allow for event handling with call-backs and promises to be used – appropriate for an app that is designed to be responsive.

Node also has very large and well supported package libraries, with the Node Package Manager (npm), which allows third party tools and support to be used to assist development as there is a large body of documentation for them on the internet.

ExpressJS[11]
Express is a framework used for running and hosting Back-End applications. It runs as a package (module) in NodeJS and can handle several key requirements we have for our applications controller. Namely, Express can handle routing requests, HTML requests and responses, and can provide a REST API, or interact with one.

As Express is a well-maintained framework that runs well with NodeJS, this is an appealing and ideal choice for development. While some of its features require the use of asynchronous programming, it can be used to serve bundled Front-End code, receive requests and respond with objects to the Front-End through an embedded REST API. While compared to Frameworks such as Flask, it may add extra layers of complexity to the architecture of the stack, in our case it opens up compatibility with JSON file database handling.

Express does not have any error handling or testing capabilities alone, so our team has opted to using a combination of Express-JS and "Jest", a JavaScript testing library for running automated unit tests before hosting and compilation.

Chosen Architecture

With all of our chosen components for development, we can finally review the components involved in the entire application. Namely, devices, browsers, and external API's. The entire architecture of our application when including them appears as:

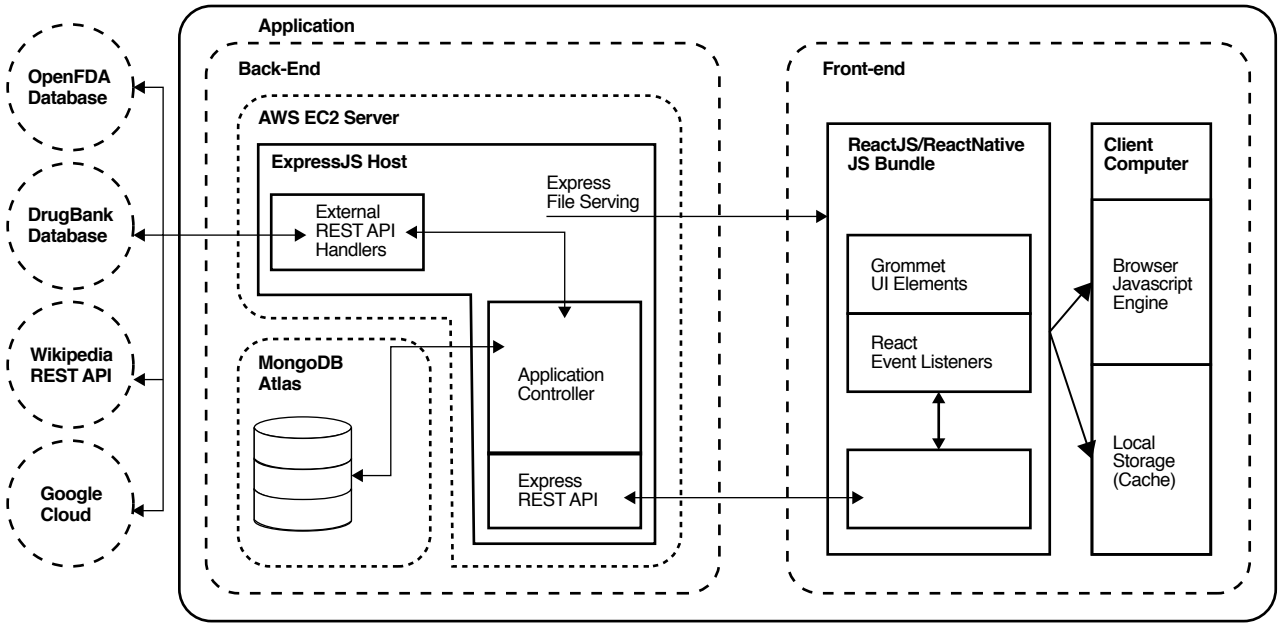


Figure 3
Diagram software architecture

Key Point Summary

React can significantly reduce the amount of redundant code on the front end, by simplifying commonly used snippets of HTML and JS into 'components'. These provide an object-oriented view of objects on a page and allow pages to be constructed more efficiently.

Babel is a JavaScript transpiler that converts ES6 or ES7 JavaScript into a backwards compatible version of JavaScript in current and older browsers or environments, avoiding errors occur by incompatibility. Babel also has extensions for transpiling JSX for React, which is a very important feature, since it allows us to intuitively construct components with HTML-like syntax and avoids the use of cumbersome React syntax for creating elements.

Grommet is a React-based web UI framework providing accessibility, modularity, responsiveness, and theming in a neat package. It offers several layout components, component APIs, and visualisation tools with a matching set that doesn't require another library. It helps to create a UI with consistent look, which also provides robust UX design capabilities.

It is efficient for Node.js to build fast, scalable network applications, as it's capable of handling a huge number of simultaneous connections with high throughput, which equates to high scalability. Node.js operates on a single-thread, using non-blocking I/O calls, allowing it to support tens of thousands of concurrent connections, remaining lightweight and efficient in the face of data-intensive real-time applications that run across distributed devices.

With advanced techniques for creating routes, such as URL parameter parsing and middleware, Express allows us to very easily add advanced functionality to our REST API, while keeping our code modular and maintainable. The AWS Elastic Computer Cloud service would allow us to gracefully scale our service across multiple servers in multiple regions, and even automate this to adjust for demand. It also handles virtual networks between our instances, reliability and backups, and elegant management of servers using server images.

MongoDB is a document-oriented database. It is easy to access documents by indexing. Hence, it provides fast query response,100 times faster than the relational database. MongoDB is also a schema-less database, which makes it much easier to evolve data structures compared to running ALTER TABLE statements on databases

3 Software Design

Use Cases

UC01 Scan Product
User scans a product, then uploads it

UC02 View Scan Result
User views the results of scanning a product.

UC03 Log In
User clicks “Log in” and logs in

UC04 Register
User clicks “Register” and registers for an account

UC05 Log Out
User clicks “Log Out” and logs out

UC06 Change Highlighting
User clicks “Change Highlighting” to change preferences

UC07 Save Product
User clicks “Save” on a product page

UC08 View Saved Products
User clicks on profile and “Saved Products”

UC09 Load Product
User clicks on a product, after UC09

Actor 01 Guest User
Users who intend to use the application, but did not log in.

Actor 02 Registered User
Users who intend to use the application, and have logged in (and registered).

Actor 03 OpenFDA
The US Food and Drug Administrations open source database API.

Actor 04 DrugBank (.ca)
The Canadian DrugBank database API.

Actor 04 Google Cloud (Vision/Translate)
Google’s cloud services, notably their OCR image recognition API service, and their text-to-test language translation service.

Actor 05 MongoDB Database
A hosted instance of a MongoDB Database, with pre-set configurations on document structure.

Actor 06 Authentication Agent
Back-End service to check credentials, and store them.

Use Cases (Diagrams)

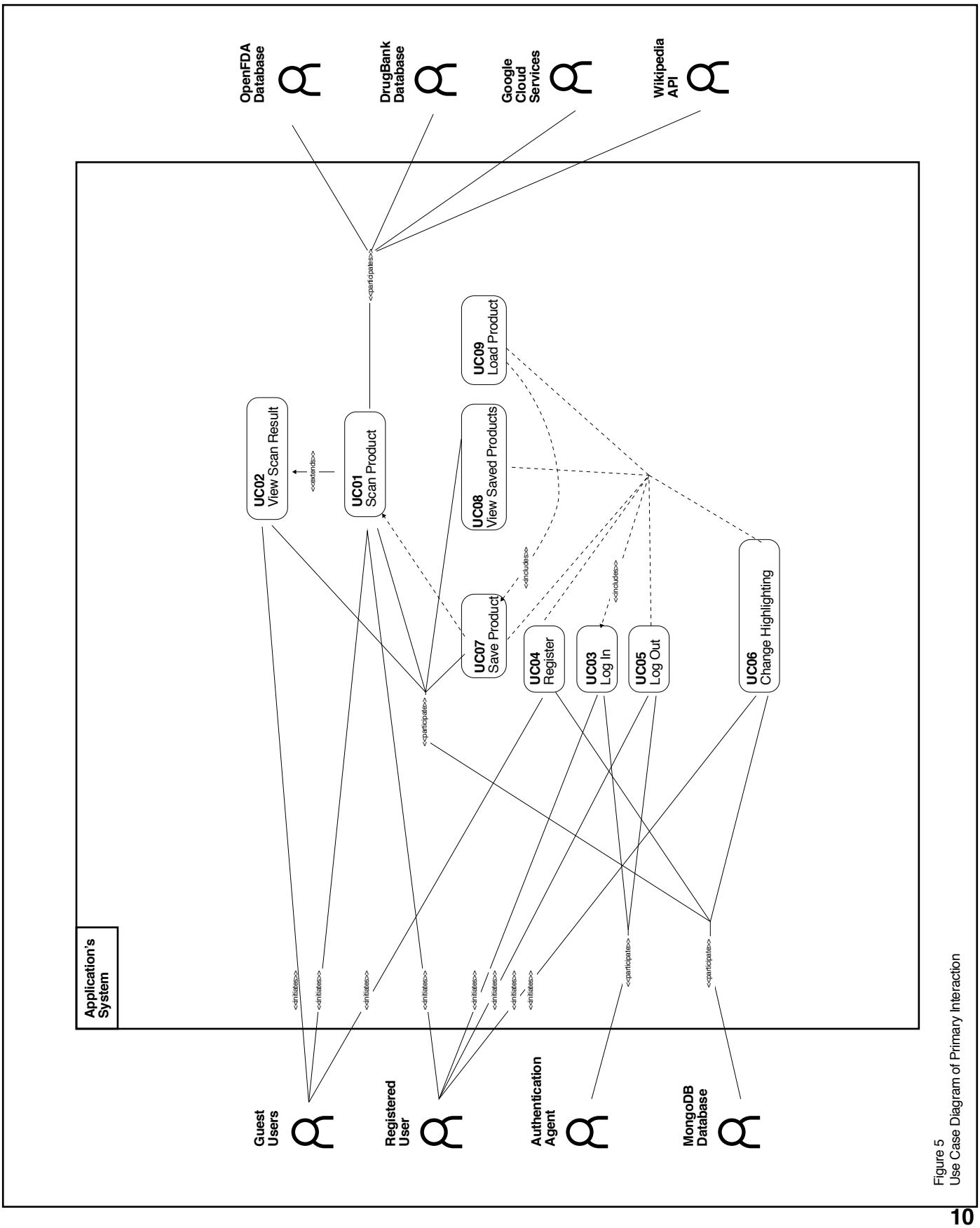


Figure 5
Use Case Diagram of Primary Interaction

Sequence Diagrams

UC01 Scan Product

User scans a product, then uploads it

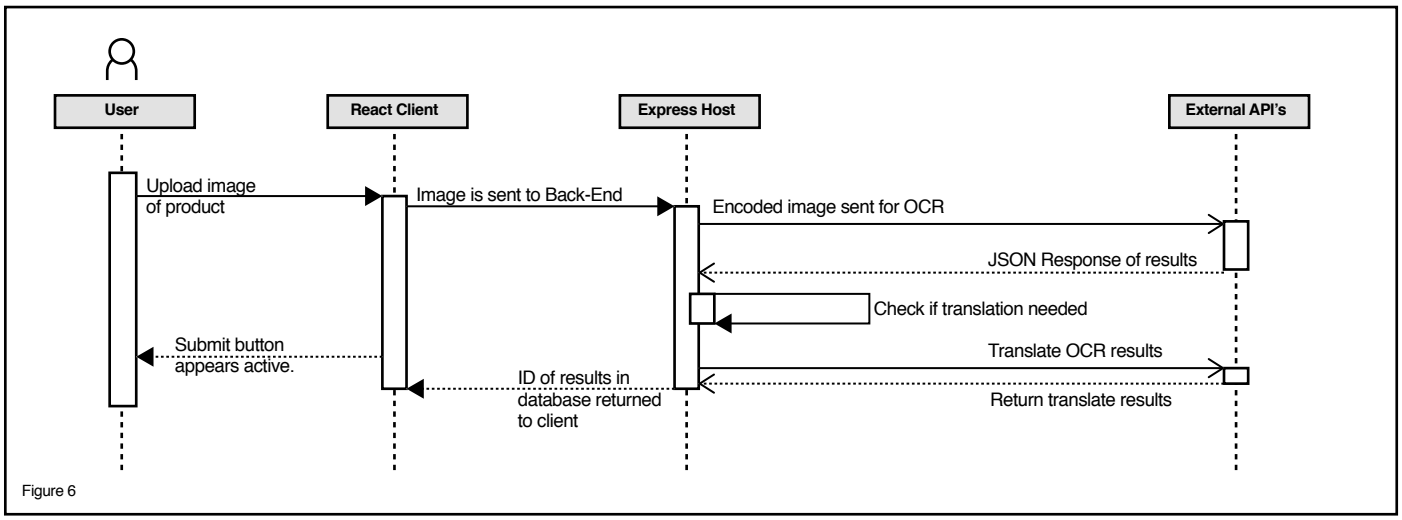


Figure 6

UC02 View Scan Results

User clicks on an ingredient to find out more information

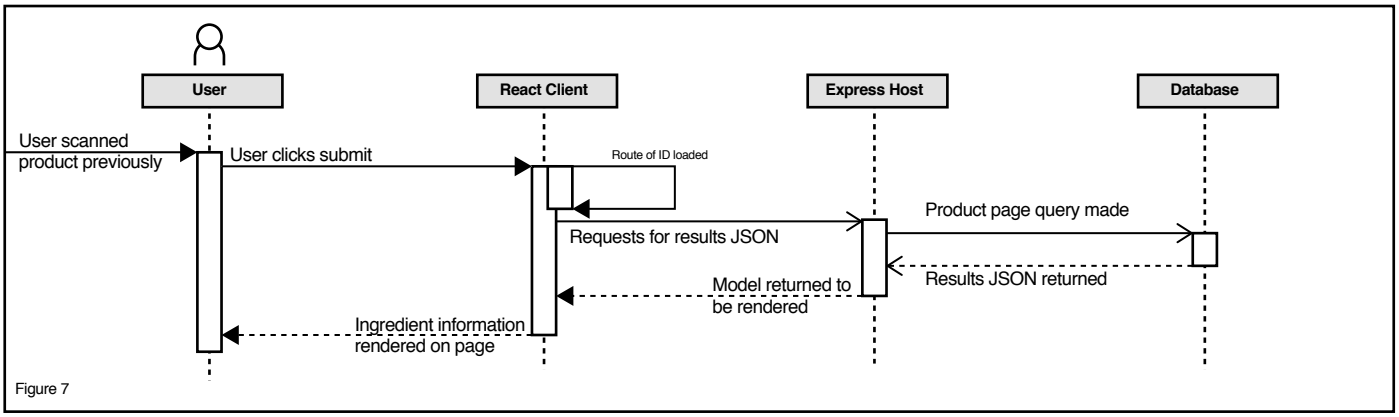


Figure 7

UC03 Log In

User clicks “Log in” and logs in

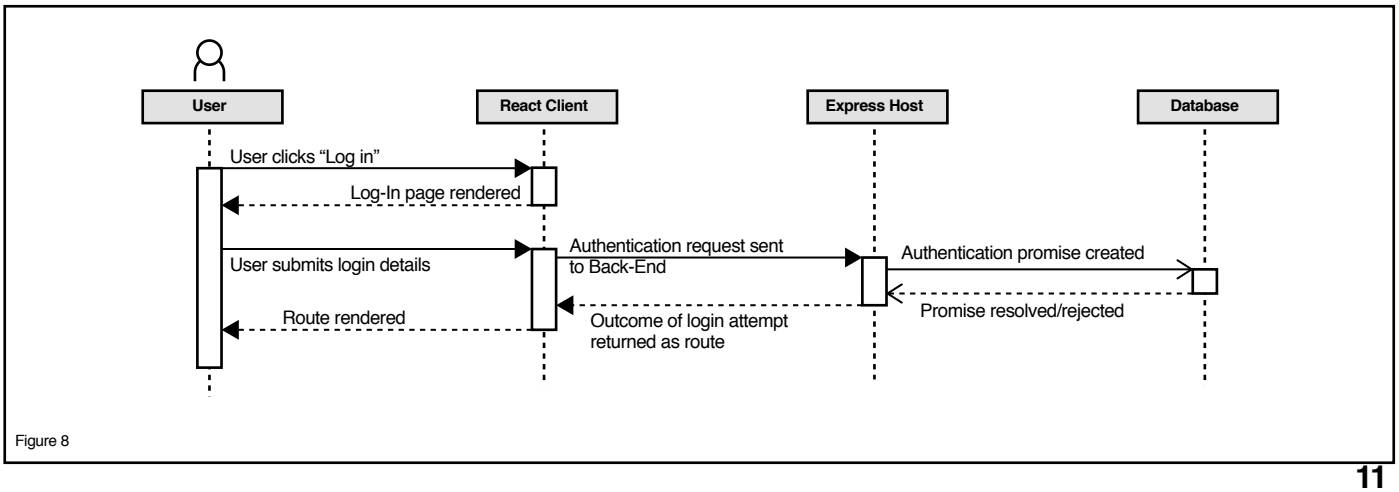
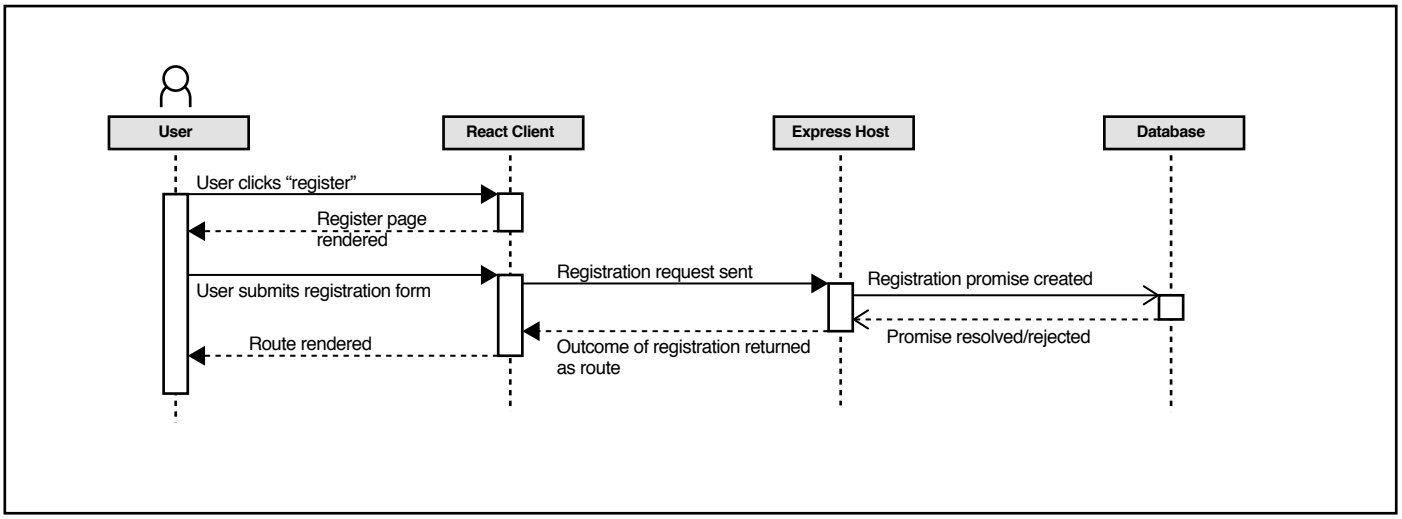


Figure 8

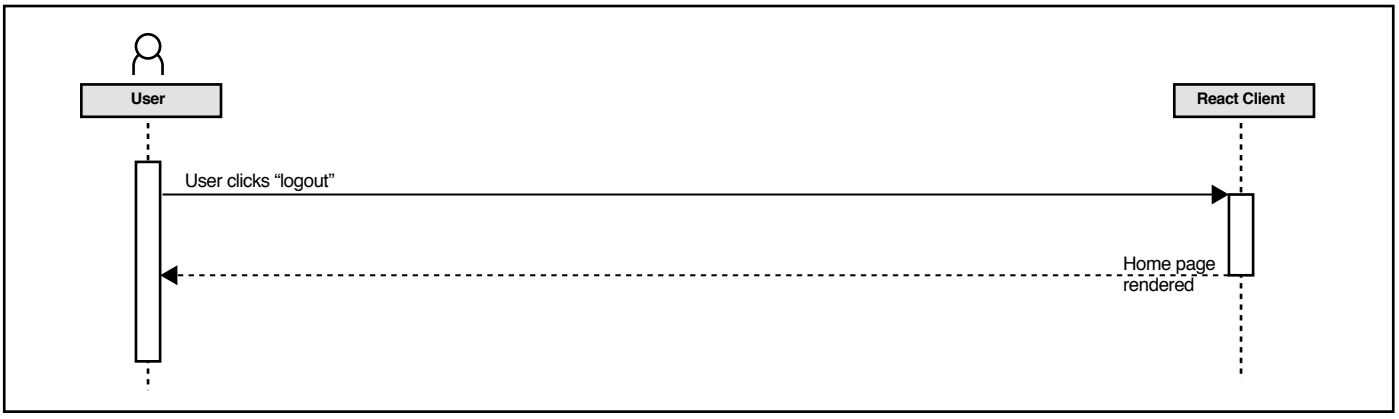
UC04 Register

User clicks “Register” and registers for an account



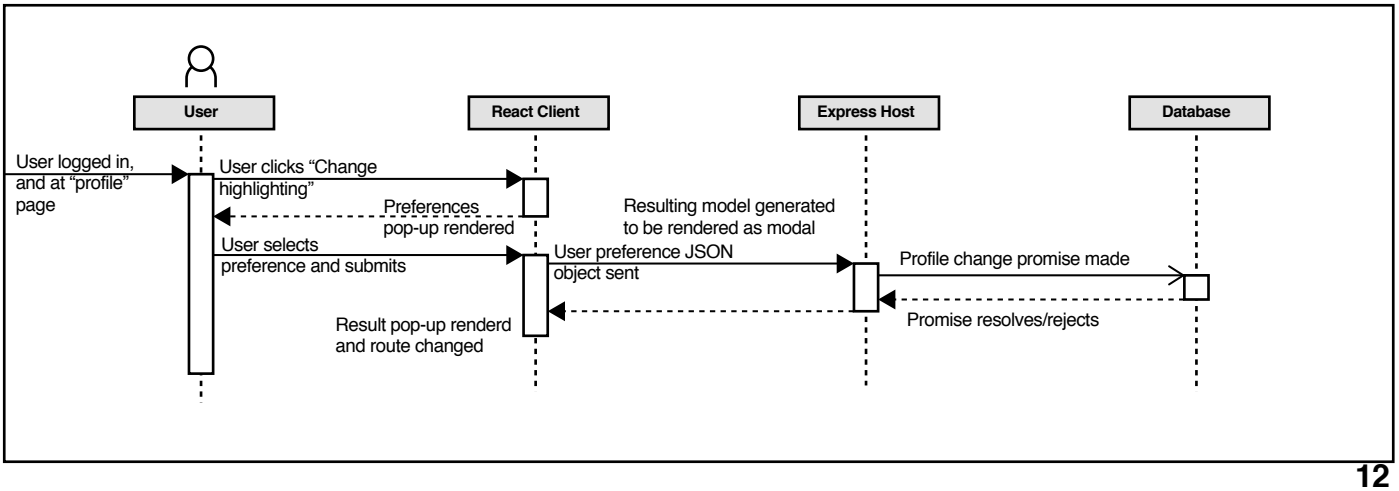
UC05 Log Out

User clicks “Log Out” and logs out



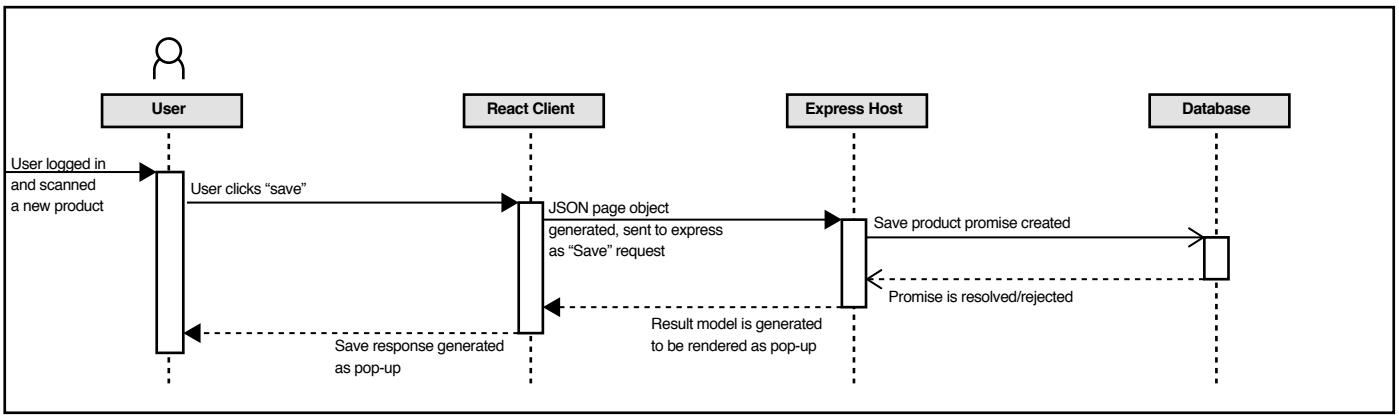
UC06 Change Highlighting

User clicks “Change Highlighting” to change preferences



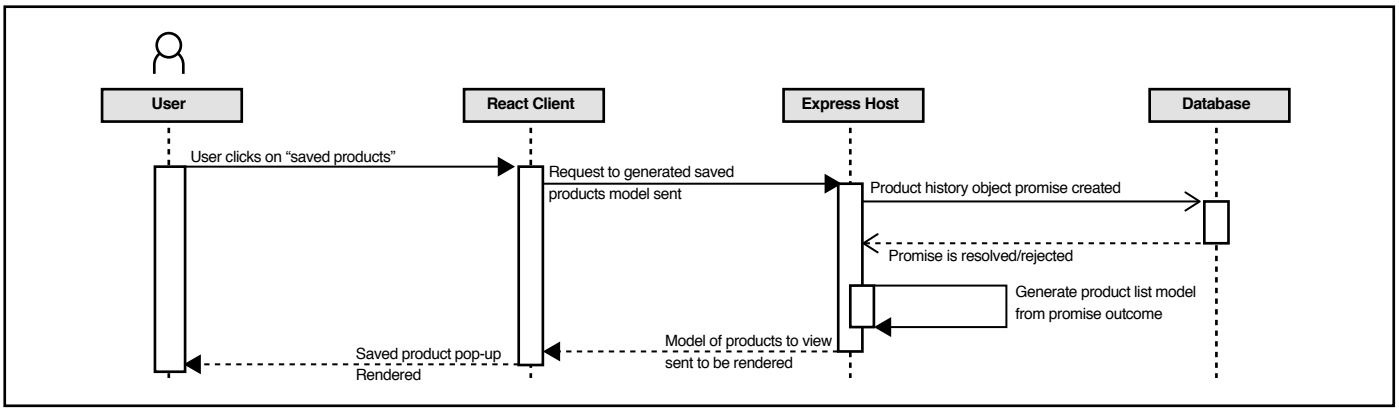
UC07 Save Product

User clicks on the save icon.



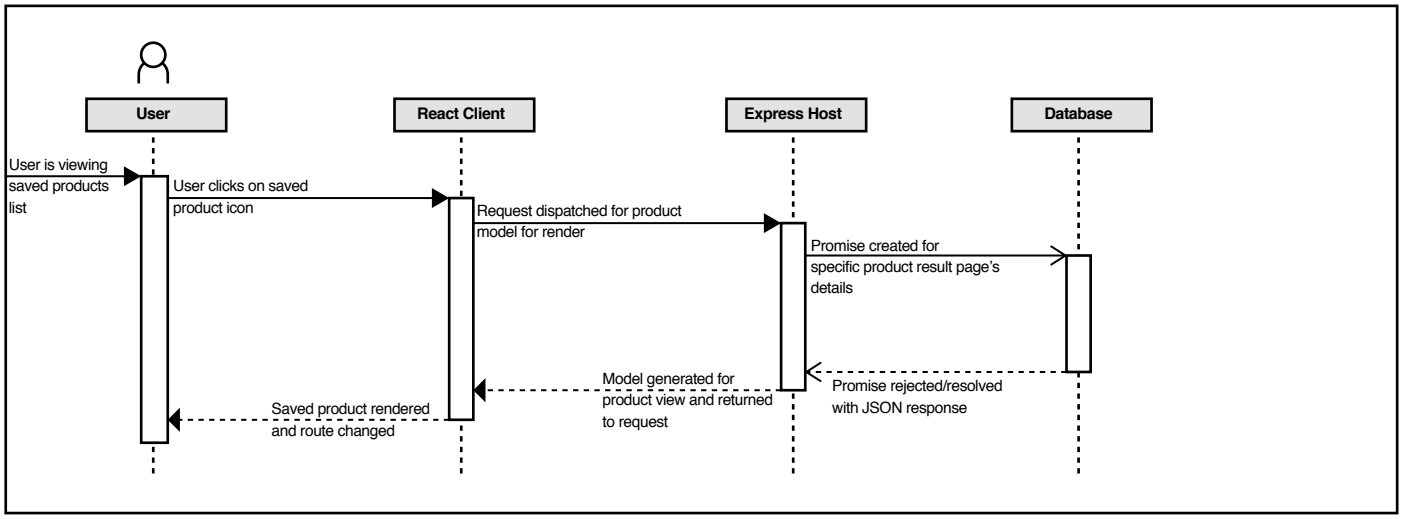
UC08 View saved products

User views all of their saved products



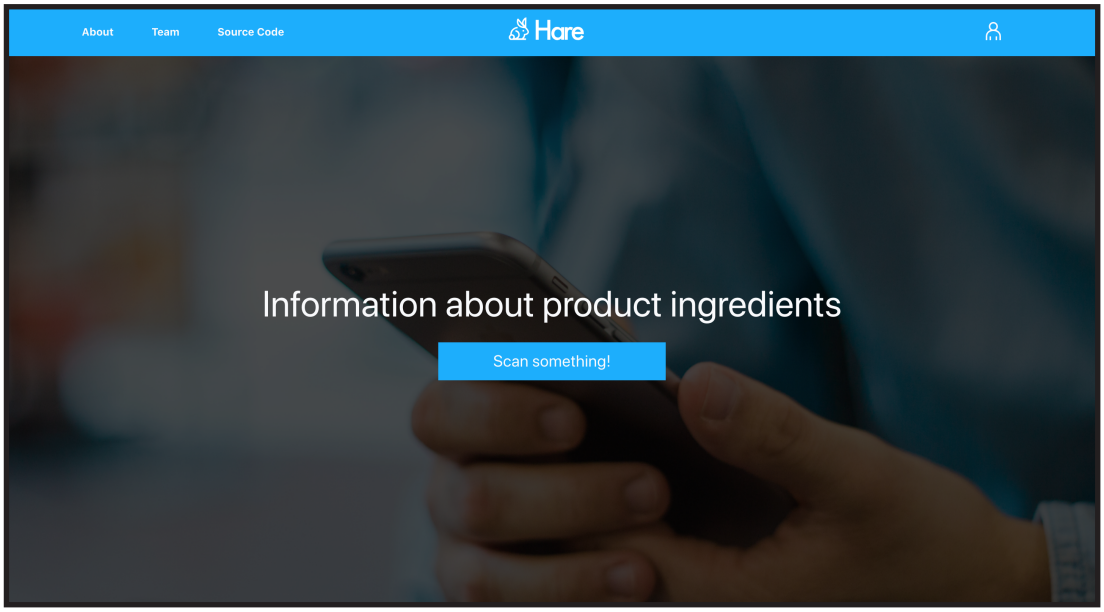
UC08 Load Product

User clicks on a product, after UC09 , or during UC07

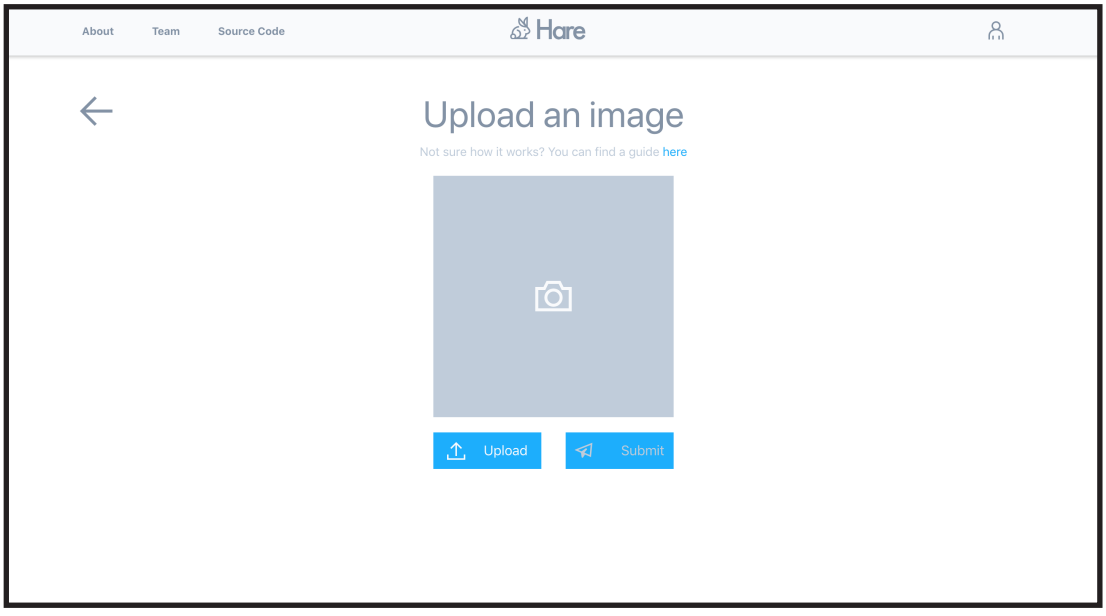


4 Interface Design

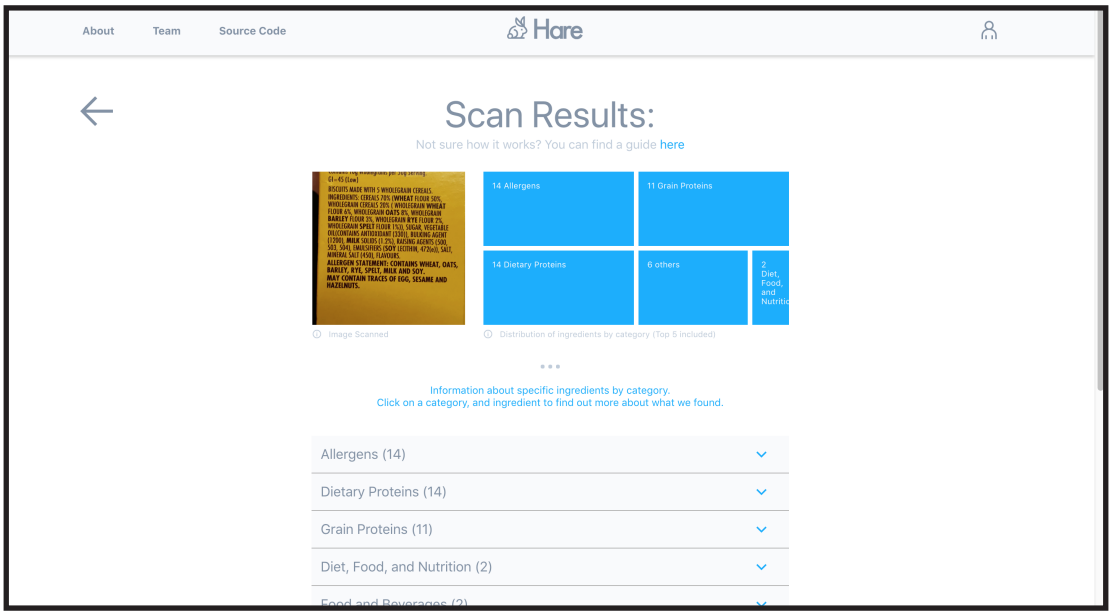
GUI Design (Screenshots)



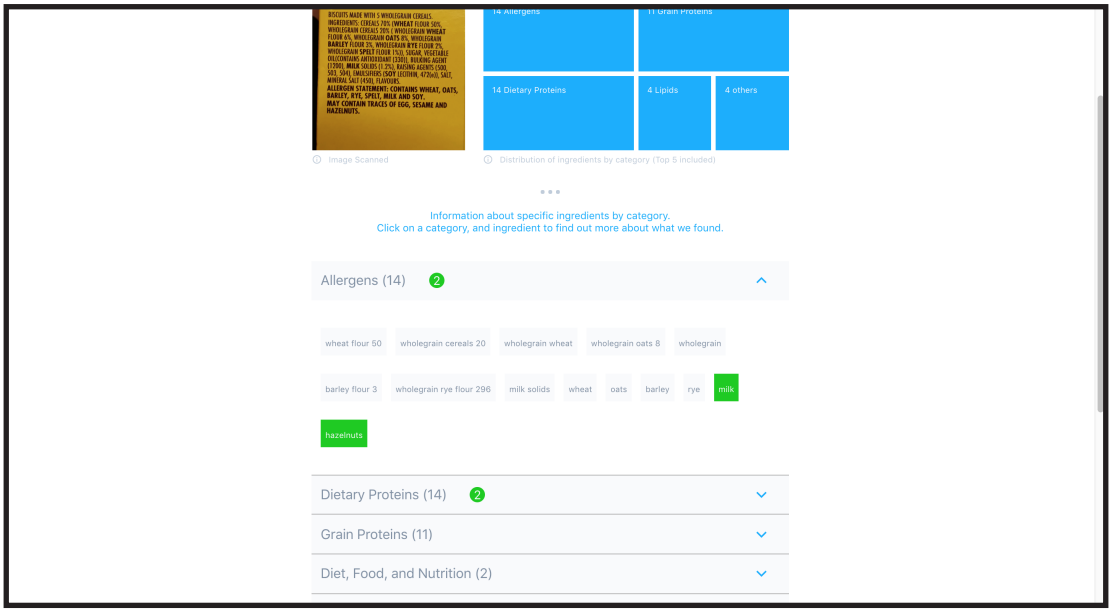
Homepage
/ Landing Page



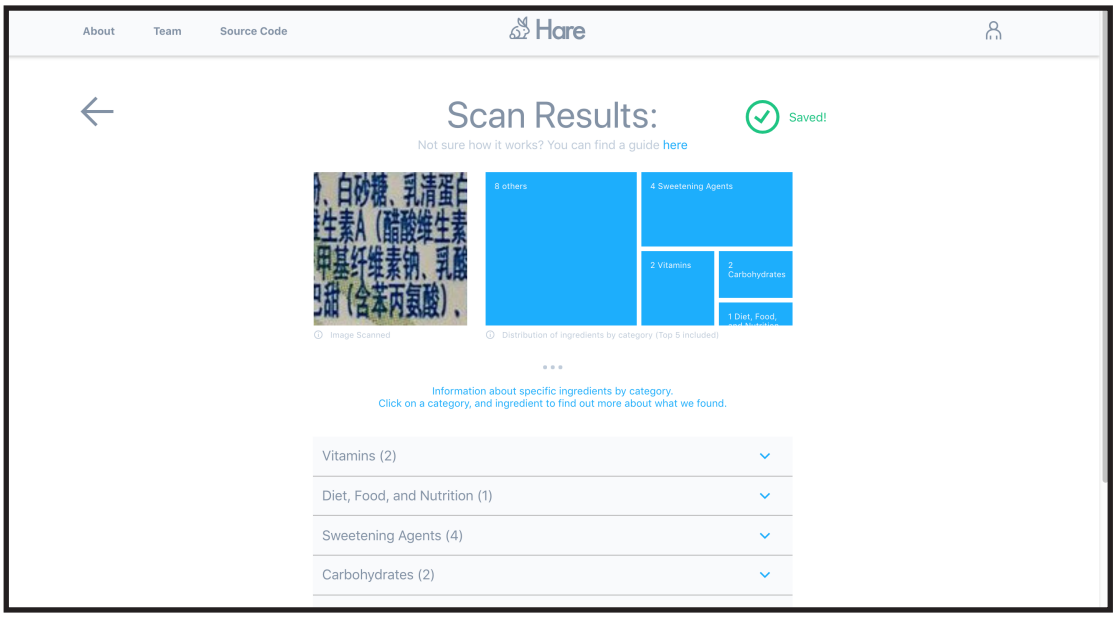
Upload page



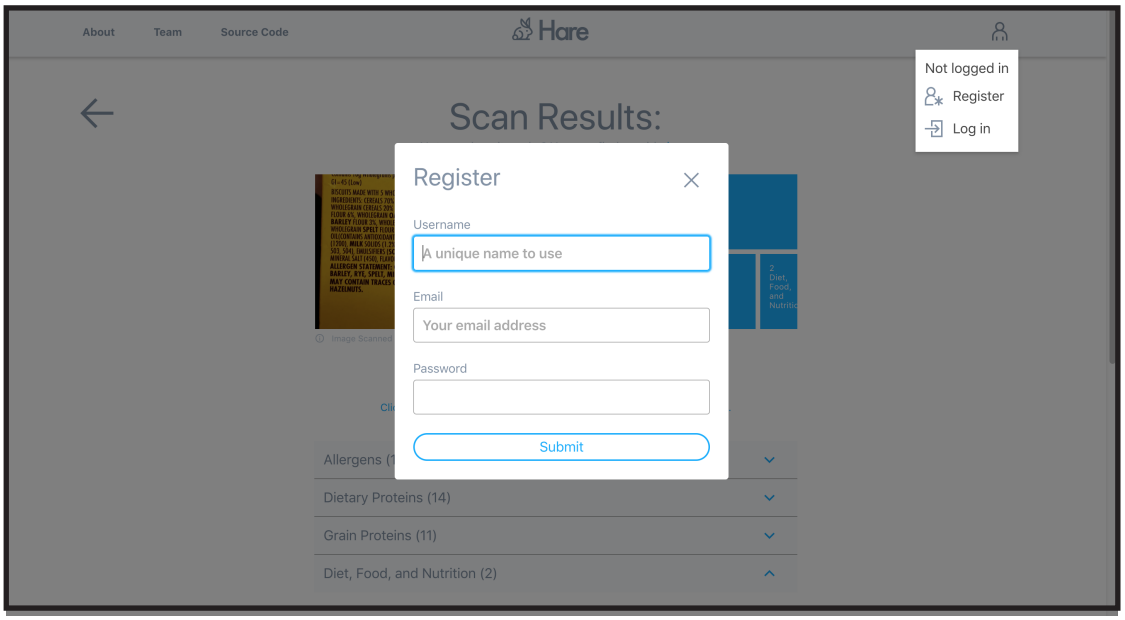
Scan Result Page (Example)



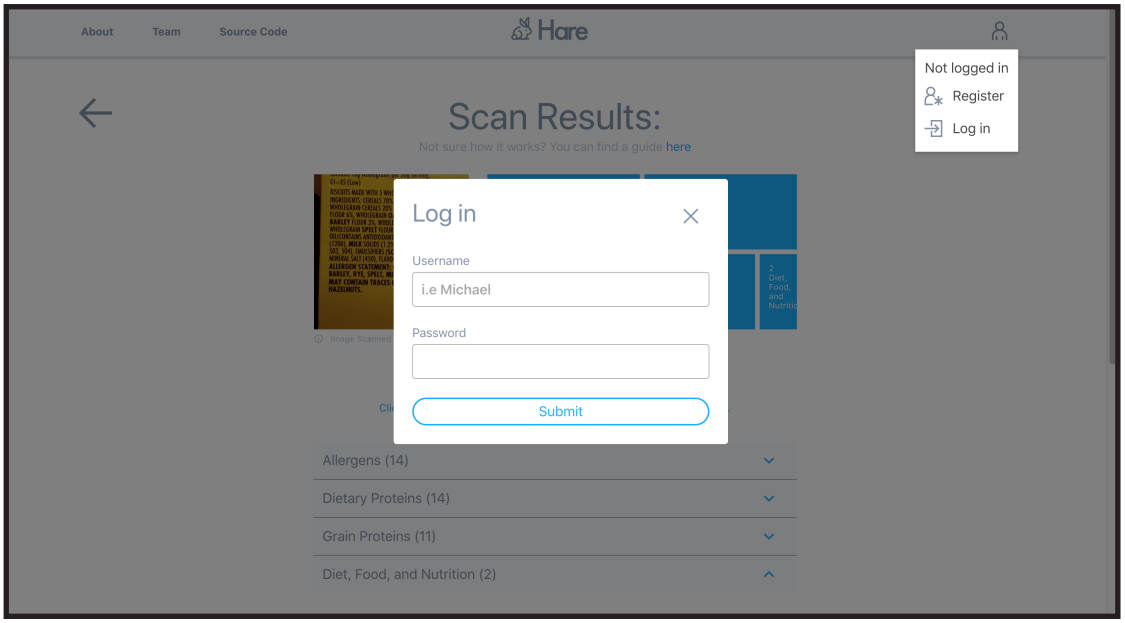
Scan Result Page With Highlighting (Example)



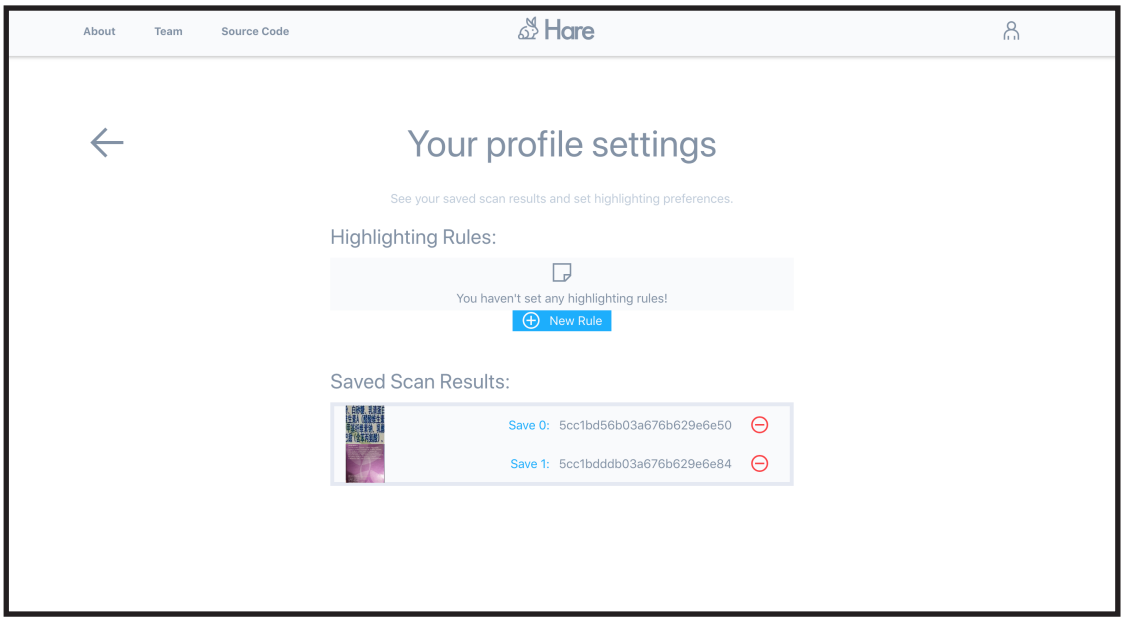
Scan Result Page For Foreign Product Packaging



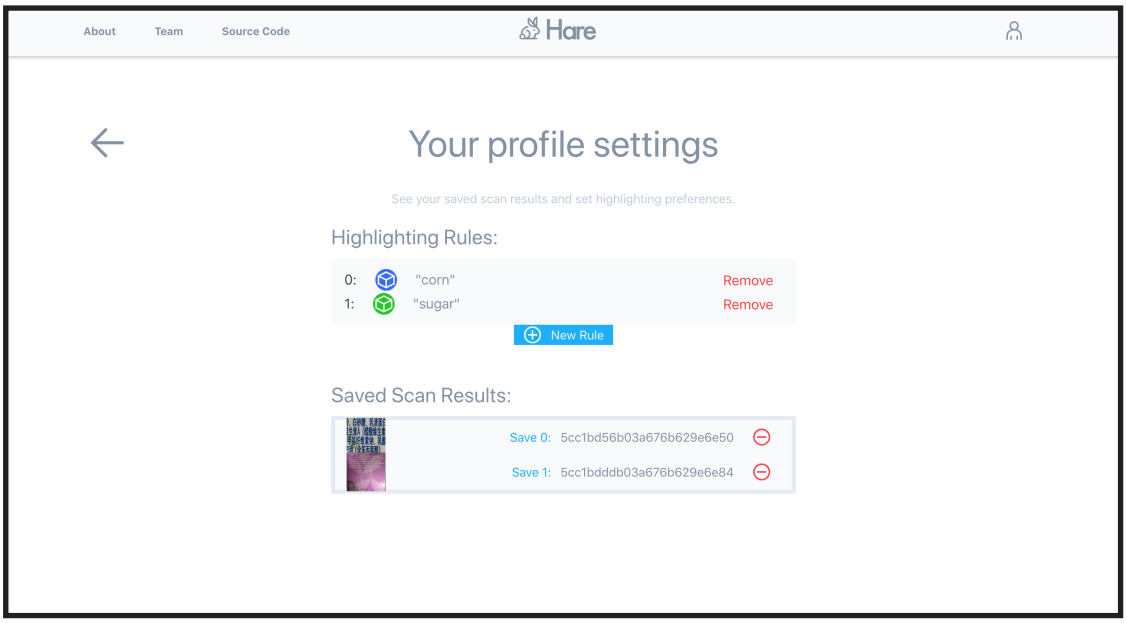
Registration Panel



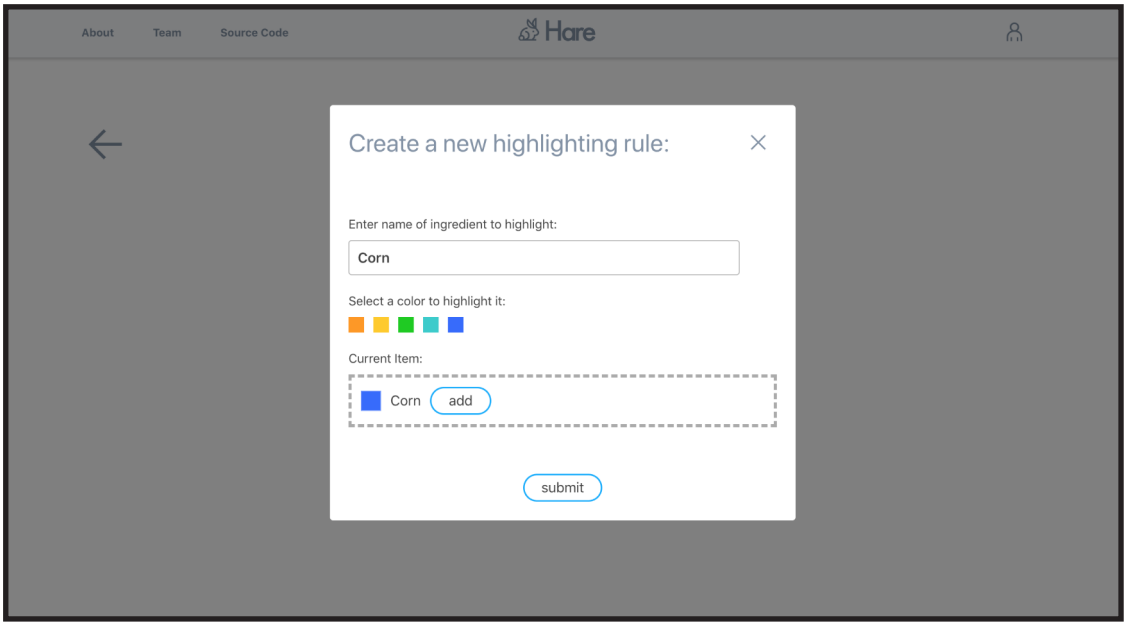
Log-In Panel



Profile Page (Example)



Profile Page With Preferences (Example)



New Preferencing Panel (Example)

Final demonstration, and the rest of the application can be viewed at www.senghare.xyz

Control
Alt
Defeat

Requirements
and Design
Workshop