Final Report

INDY 09 Blue - Pure Dine

CS 4850-01 - Spring 2024

Sharon Perry - 4/27/24

Ibrahima Guye, Mohamed Gothany,

Mohammed Shaikh

Git: https://github.com/Indy-9-Blue-Pure-Dine/blue-pure-dine

Website: http://indy09blue.github.io

Roughly 2800 lines of code were produced for this project, and five tools were used.

Table of Contents

1.0 Introduction

1.1 Overview

Finding foods that cater to specific dietary laws like Halal and Kosher can be a challenge, especially for travelers or those in unfamiliar locales. The PureDine app aims to address this challenge by enabling users to search for products and providing them with brands and items that specifically adhere to their dietary restrictions.

1.2 Project Goals

The web application will feature user profiles where individuals are able to lookup certain brands and foods to see if they follow restrictions, allowing for an inclusive experience. The goal of the web app Is to provide users with the opportunity to look up certain products and check if it aligns with their dietary needs. Another goal is to create the app with a user-friendly platform to empower and boost the confidence of shoppers with dietary restrictions.

1.3 Definitions and Acronyms

Definitions
Kosher: Refers to Jewish dietary restrictions
Vegan: Individuals who do not consume any food sourced from animals or use products derived from animals
Halal: Refers to Islamic dietary restrictions
Vegetarian: Individuals who do not eat meat
Acronyms
API: Application Programming Interface
UX: User Experience
UI: User Interface

1.4 Assumptions

The first assumption is that users are connected to the internet and can operate any operating system a. Another is that users have previous experience with similar technology and familiarity with certain app functions (navigating between pages, searching products, exiting web pages, etc.).

2.0 Design Constraints

2.1 Environment

The PureDine app will run on any mobile device and is compatible with popular browsers such as Apple, Samsung, Google. Any mobile device with camera capabilities will be able to successfully run the application.

The backend system will be hosted on FireBase, which uses cloud architecture for scalability and stability. Firebase will be used to store user information such as profile and saved preferences. Firebase also

The application's frontend will be built with JavaScript (JS) to provide dynamic and interactive user experiences.

2.2 User Characteristics

The PureDine web app's target customers include people who observe certain dietary regulations like Halal and Kosher, as well as travelers, tourists, and locals looking for food products that meet their dietary restrictions.Users' technical skills might vary, ranging from casual internet users to highly experienced persons.

2.3 System

The PureDine web app will be developed to manage a high volume of concurrent user queries while being responsive and reliable during peak usage periods.

3.0 Functional Requirements

3.1 Login With Password & User Authentication

3.1.1

Account Creation: Users can choose their screen names, passwords, and email addresses when registering. Email verification is a component of the registration process to increase security. After successfully registering, users will receive an email address with a verification link that they must click to activate their account.

3.1.2

Login: The clients can login through their email and password. Adding "Remember Me" function to relieve login stress for returning users.

3.1.3

Password Recovery: The password recovery process can be started by the user in the case that they forget their password. A link to reset their password will be given to the email address they registered with, which needs to be entered. After that, users can reset their password and get back into their accounts by clicking the link.

3.2 User Profile Management

3.2.1

Profile Customization: Users can specify their food preferences by adding options such as Halal, Kosher, Vegan, and Vegetarian to their profile. Users can adjust

their search results by stating their dietary needs in their profiles, ensuring that only products that fit their specific dietary requirements appear.

3.2.2

Search History: The software preserves a safe copy of the users' search history, making it easy for them to retrieve previously seen content. By enabling the rapid retrieval of previous searches, this feature improves user experience by saving time and effort.

3.2.3

Favorites: Users have the option to save their favorite products and brands for convenience. By bookmarking their favorite products, users may easily identify products they frequently buy or want to repurchase, speeding up their browsing experience.

3.3 Beneficial Application

3.3.1

Product Search:Users can search for items by name, brand, or category, making it easier to find products that match their dietary needs. The search engine is intended to interpret and accommodate diverse manifestations of dietary laws, resulting in accurate and relevant search results for users with special dietary constraints.

3.3.2

Advanced Filters: In addition to the basic search feature, users can enhance their search results using complex filters. These thorough filters enable users to narrow their search results based on specific parameters such as dietary preferences, certifications, nutritional attributes, and so on. By allowing users to apply advanced filters, the platform enhances the granularity and accuracy of search results, resulting in a more efficient and gratifying browsing experience.

3.4 Product Information and the Verification Tests

3.4.1

Detailed Product Information: Each product page will have all essential information, including ingredients, dietary certifications, user reviews, and ratings. Users can make informed judgments based on their dietary preferences and needs after receiving a complete review of product details.

3.4.2

Certification Verification: To ensure credibility, product pages will feature links or confirmation stamps proving the authenticity of dietary certifications such as Halal, Kosher, Vegan, and Vegetarian. These verification techniques give users confidence that the commodities correspond to the defined dietary restrictions and guidelines, fostering trust and openness on the platform. Furthermore, measures to promote and install technologies such as diesel soot filters contribute to significant reductions in urban air pollution, displaying a commitment to environmental sustainability.

3.5 Homepage and Navigation

3.5.1

Display Home Page: Home page will show up a clean and user-friendly interface via which one can easily access search, dietary filters, and previous searches while at the same time enjoying a great user experience.

3.5.2

Navigation Menu: We'll make a responsive navigation menu so that users can quickly flip between sections. Users will be able to readily access the application's many features and details with the help of the menu, which will have subheadings like Home, Search, Favorites, Profile, and About Us.

4.0 Non-Functional Requirements

4.1Performance

The system is optimized for a quick start during which it's able to run up to its power in only 5 seconds. It performs under active operation in the same way, and this is so even as there are many users consuming large data sizes at the same time. In this case, the system does not slow down or crash. Regardless of the amount of data being processed or the number of concurrent users interacting with the system, this performance guarantees a flawless user experience.

4.2 Capacity

It should be possible for the system to support a sizable number of concurrent user sessions without seeing a noticeable drop in performance.

4.3 Usability

Users with different degrees of technical expertise should be able to easily navigate and use the user interface.

4.4 Reliability

We will ensure that the system is always accessible anytime the user has an internet connection to guarantee successful launch of the app. Users profile once set will be safely secure for you to visit and make edits when needed. Also, expect to work offline as well by seeing the application work fully offline. Also, the camera access also permits

the scanner to operate in a consistent fashion, with the scanning always done correctly, through the correct scan of the barcodes of the food items.

5.0 External Interface Requirements

5.1 Hardware Interface Requirements

There are no specific hardware requirements. Users can use this on any mobile device if they're able to connect to the internet.

5.3 Software Interface Requirements

The user interface should be responsive and adaptive to multiple screen sizes, such as PCs, tablets, and smartphones. Visual design aspects should be consistent and visually appealing, hence improving the entire user experience.

5.4 Communication Interface Requirements

Email communication is required if the users forget their passwords. HTTP protocol will be used for the Application in conjunction with Google Firebase. API communication to OpenFoodFacts is also used to retrieve nutrition data.

6. Detailed System Design

Most components described in the System Architecture section will require a more detailed discussion. Other lower-level components and subcomponents may need to be described as well. Each subsection of this section will refer to or contain a detailed description of a system software component. The discussion provided should cover the following software component attributes:

6.1. Frontend

The website will provide manual and social authentication options to make users sign in effortlessly and fast. With regards to manual logins, we securely deal with and keep user credential information, including usernames and passwords. For those who like to log in via social networking, our site is integrated with the most common platforms, such as Google and Yahoo, so users can access our services with their existing social media accounts. Manual Authentication Process: Users who choose to log in manually will be verified by the Firebase Authentication service directly. Once the users are verified, they are given access to various features of the website. Our website makes use of the OAuth protocols for the social media logins so that users can log in through their social network accounts. This method guarantees a secure transfer of user data between social platforms and our website, enabling simpler logging in processes but at the same time providing high security.

6.2. Backend

Our site is supported by a Firestore NoSQL database which is a very useful tool for storing and managing the user profile information including the personal data and a

scanned products history. Firestore guarantees real-time synchronization across each user’s device, giving access to the information at any time. Firestore enables creating, updating, and deleting the user profile information. Additionally, it takes the responsibility for authentication of new data, including email and phone number updates, and then incorporates these changes into user profiles. The site has the history of the last ten scanned items stored in the Firestore. The scanned products are shown with their picture, title, and dietary alignment status (check or "x"), making it very easy for the user to review them. Cloud functions specifically designed for this purpose evaluate the product data against the user's dietary preferences. The compliance is then signified by a check or an "X."

6.3. API

The process involves sending HTTP requests to the OpenFoodFacts API, decoding the data returned from the API, and empowering the user experience with product details and features that enhance dietary compliance. Once the OpenFoodFacts API responds, the function at the back end uses the JSON data to isolate the crucial product information. This process guarantees that users get the data that is precise and relevant to the barcodes they scan, which end up being used to make informed decisions based on their dietary needs.

7.0 Analysis

In the analysis phase of the "INDY 09 Blue - Pure Dine" project, we made a full analysis on the motivation factors, did appraisals, and set parameters in the development process that will guide our activities. It afforded to us a framework which every step of the way was based on our objectives and user requirements.

7.1 Motivation

Goal Alignment: The primary goal behind the "Pure Dine" was to ease the process of authenticating and searching the food items which are proficient to match specific dietary restrictions.The expanding trend of dietary-specific information underscores the substantial market need that the proposed app is geared towards closing.

User Empowerment: We are determined to develop an easy to use interface and offer instantaneous statistics to our users, which will enable them to make informed decisions, regardless of the health related dietary restrictions like allergies as well as diets like vegan and vegetarian.

Technological Innovation: By using advanced technologies such as React Native and Firebase, this project aims to push the boundaries of what really is possible when it comes to dietary management apps. Hopefully setting a new standard for integration and

7.2 Pros/Cons

Advantages:

- User-Centric Design: The app is user-friendly as it was thoughtfully designed so that it is accessible to tech-savvy people and even those who are not, thus that makes it universal, and the availability increases that particular base of the users.
- Real-Time Updates: Insertion of OpenFoodFacts as the core engine implies that product information is both wide-ranging and moreover is frequently real-time, allowing users to access freshest data available.
- Scalability: The utilized approach of cloud backed backend architecture provides an extraordinary scalability which makes it possible to handle more users and more data requests without any decline in performance.

Disadvantages:

- **Integration Complexity**: Integration of such sophisticated technology for multiple dietary databases and user preferences is data demanding which requires robust data handling, this can expose the apps to higher risks of errors and bugs during the initial phases of roll out.
- **Dependency on Third-Party APIs:**The App function partly depends on the external API which might be caused by data unavailability or data security issues.
- **Complexity in Maintenance:** Maintenance entailed continuous monitoring, keeping track of the data flow, and remaining up-to-date, thus, was complicated by the integration of many technologies.

7.3 Guidelines

Development Standards: To ensure a steady and high-quality performance of the app, we are strictly committed to the most advanced coding standards and the latest development principles. The repetition of code reviews and the MVC pattern creating modular and maintainable code is being always carried out.

Performance Benchmarks: The app is fashioned for achieving certain performance goals, such as rapid loading and proficient data side-by-side, to guarantee a smooth experience for the user even during high traffic volumes.

Future Adjustments: Further, users' involvement through continuous feedback, allows them to drive implementation decisions. So then we can adapt to acknowledge user needs and stay updated with technological innovations.

8.0 Maintenance and Validation

8.1 Testing Strategies

Before launching the application, we must ensure that all tests are passing and optimal. Many testing tools exist and specifically we will use the one on Expo-Go, which is a built-in testing function for each component of the software. This will assist us with determining the performance details of the software application. The testing method that we will use is the functional testing which tests the components and functions to ensure the tasks are being performed in a correct manner. Finally we will determine the performance of the application based on speed, app reliability, and optimal scaling.

8.2 Maintenance Process

When an error occurs in the application or it crashes, the problem must be identified and fixed expeditiously as they may affect other functions of the application. This can be done by continuously testing our software and taking feedback from users in the reviews section as they give unseen feedback sometimes about crashes or performance speeds. Our goal is to identify these issues and address the bugs before they take effect. The current method at the moment is agile however we may go with the waterfall or intertwine them depending on issues.

8.3 Functional Testing

The overall testing involves making our own assessments on each feature in correspondence to the requirements. The goal of functional testing is to observe and note each component and decide if they were successful or not. Testing involves: Navigation to profile , Login and register, Navigation to Barcode Scanner, History Page, Map Feature for Restaurants

Comprehensive Test:

Requirements	**Pass/Fail**
Navigation to profile	**PASS**
Log and Register	**PASS**
Navigation to Barcode Scanner	**PASS**
History Page	**PASS**
Map Feature for	**FAIL**

Restaurants	

9.0 Design

9.1 System Architecture

The project architecture was carefully selected to be able to meet scalability and performance requirements. This system utilizes a client-server architecture which is the backend hosted on Firebase, receiving the input data from the front-end immediately and performing the task without taking time.

9.1.1 Frontend Architecture

We used React as the foundation for the frontend and we executed javascript methods to get data. This decision was determined by the processes, in which React plays a fundamental role. The architecture of React components is designed in such a way that it enables re-rendering and state management which can be used for dynamic application. Responsiveness of this application allows it to adapt to different types and the sizes of the devices it is used on, its useful features being CSS Flexbox and Grid system.

9.1.2 Backend Architecture

Firebase takes care of everything as a backend platform, it gives the users integrated cloud services like Firestore for database management, Firebase authentication for safety and security, and Firebase Cloud Functions for running the server side operations. It motivates the limit of the workload for the server and fully automates the scaling process at the application demand.

9.2 Data Flow Diagram

We made a particular diagram (DFD) for solving problems of data flow among system components. By means of this flow chart, the data trip is highlighted from where app users put in information, this data is transferred to Firebase services and then back to the user interface in the same way. Several procedures such as user authentication, product information, and history records that are included in the stated activities.

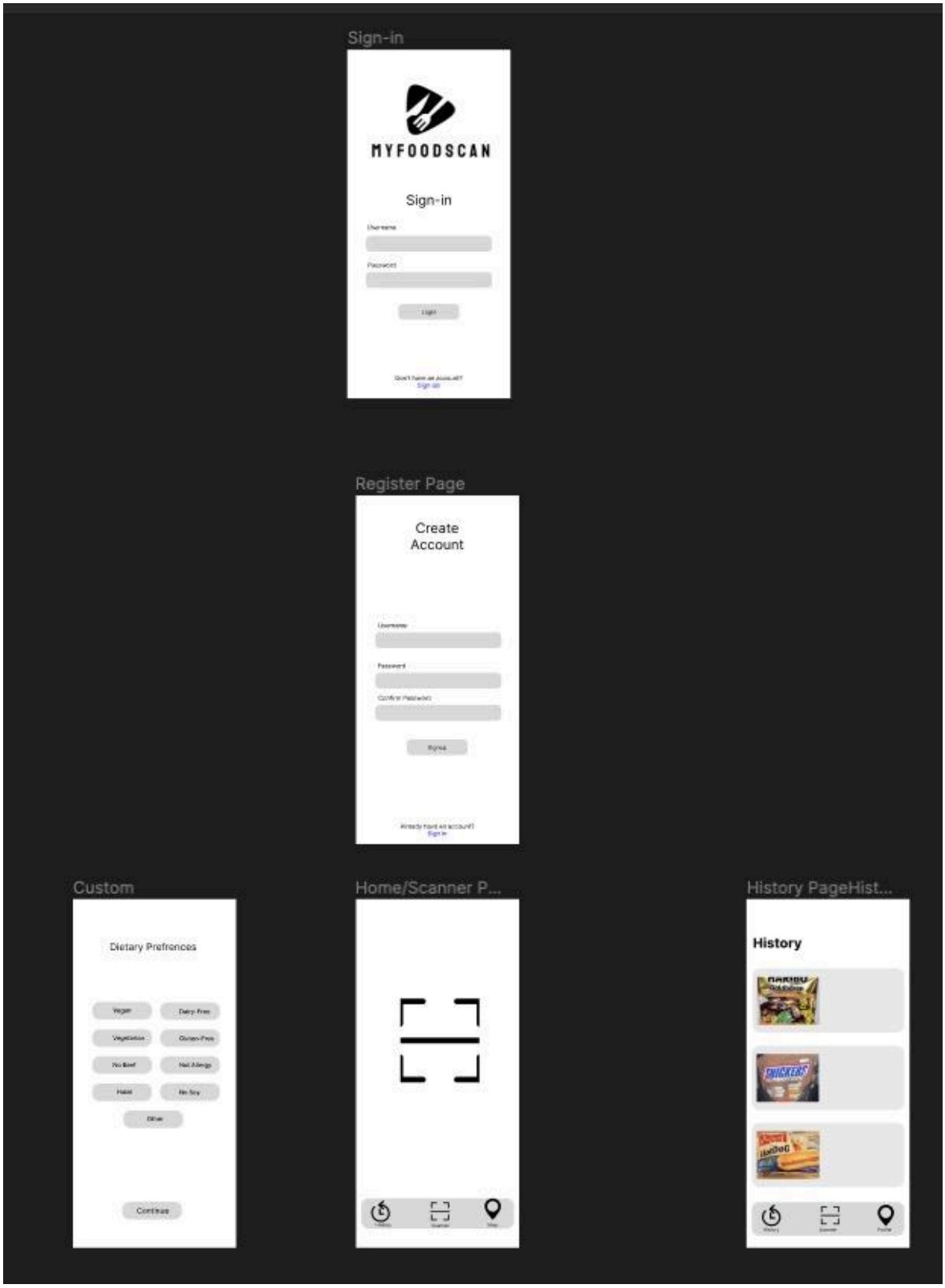

Figure 1 layout

10.0 Development

10.1 Major Components of Technology:

Developing the app we chose to use React Native, a JavaScript-based framework which solved the issue of supporting iOS and Android with one single codebase. This was advantageous to us because it enabled us to build projects faster using pre-existing libraries. Moreover, we took the benefit of Expo platform as an open-source tool to assist us in testing, debugging and to reduce the delivery time as well. Expo not only displays the app for us but also has it in real time preview mode.

10.1.1 Account Creation and User Login

The approach used data storage with Firebase Firestore and user authentication with Firebase Authentication for the account creation and user login capabilities. This kind of flow helped to store the user's basic information such as name and email, while providing a variety of authentication sources like either an email/password or OAuth platform such as Google or Facebook, for easier access.

10.1.2 Dietary Selection Interface

The dietary selection interface was created to allow users to identify their dietary preferences among many that are available which could include Halal, Kosher, Gluten-Free, Nut-Free, or Vegan. The process of tailoring software to the user's needs was built on this feature, which utilized conditional logic to adapt the user interface according to the user selections, thus making the experience custom and relevant for every user.

10.1.3 Product Scanning Interface

The React Native Camera library we use works as an interface to the device’s camera, making possible the barcode scanner feature of the app. By scanning the product, the app then interacts with OpenFoodFacts API which is then used to fetch all information about the product. This uninterrupted connection agrees in the fact that consumers will get correct and time data relating to one's dietary compliance.

10.1.4 Product Details Overview

Once a product is scanned, a detailed pop-up is displayed which presents information such as product images, ingredients, dietary compliance. This module is designed to provide a comprehensive overview, helping users make informed dietary choices quickly.

10.1.5 Search History Management

Our app stores information achieving this by using firestore all user search history of the products that were searched including their name, image, and timestamp. In the attempt to maintain the popularity of the IoT app, we

implemented the LRU (Least Recently Used) cache strategy but not all recent events, which makes the app fast and responsive.

10.1.6 User Profile Customization

User's profile section is run by Firestore and Firebase Auth. It enables the consumers to be sure that their information regarding diet and preferences remain safe as they can update their details subsequently. Integration with the Expo Image-Picker module helped create a convenient interface for users who wished to upload or change their profile pictures and so it made the application better and more personalized for the users.

10.2 Challenges Encountered:

During development, we faced several challenges:

10.2.1 Dietary Selection Syncing

A significant challenge was ensuring that dietary selections made during the registration process were accurately reflected in the user's profile. We addressed this by improving state management across the app.

10.2.2 Barcode Scanning

Being first, the scant reliability of the function devoted to barcode scanning during low-light conditions was the significant problem from the very beginning. We effectively tuned the lens settings and digital image processing algorithms that yielded a more efficient product.

10.2.3 Real-Time Data Rendering

Just one problem at the beginning, the difficulties with real-time querying of the search histories due to the delays. We improved our code through the use of React Native's optimized code lifecycle and Firestore's real-time data synchronization infrastructure, hence, greatly decreasing latency.

10.2.4 Profile Customization

We ran into trouble with the current users profile pics not updating in real time, this because of the incompatibility problem with the previous version of the Expo Image-Picker library. Updating the library to a newer version and revising our implementation strategy we managed to get rid of these problems so now users are able to react to our system hassle-free.

11.0 Challenges and Resolutions

In the course of the app's development our team was approached with a number of problems across different app components. The problems were solved through innovative approaches and adaptation that are shared below.

11.1 Questionnaire Interface

Challenge: The synchronization between user-chosen dietary restrictions with their profiles in real-time was the crucial challenge for the interface of the questionnaire.

Resolution: For this, we developed a better state management logic within React and used additional hooks that prompt an update of all the components as soon as any change is made without delay. Such assures that any choice from a user will be reflected right away in their profile settings.

11.2 Product Scanning Functionality

Challenge: The outdated library that was employed at the initial implementation of the barcode scanning feature led to unstable performance and errors.

Resolution: We moved to the improved React Native Camera for barcode detection and some customized algorithms to enhance the recognition accuracy even when the lighting conditions vary or camera angles change.

11.3 Product Details Display

Challenge: The greatest challenge was in bringing the vital product information including images, ingredients, and dietary compliance into the pop-up display which often resulted in data alignment and timely retrieval challenges and affecting the customer experience.

Resolution: We optimized database queries and restructured the information architecture so that the critical information loads first. This was then added by a live update feature which uses immediate content refresh as the data fetch is complete.

11.4 History Log Updates

Challenge: The history page faces difficulties because of cache asynchronous updating as it is not up to date with recently scanned items and data updates.

Resolution: To resolve this, we introduced more efficient event-driven data-handling using Firebase Cloud Functions that now makes the history log up-to-date. So the interface is responsive.

11.5 Inclusion of the Mapping Functionality

Challenge: One of our visionary goals is to integrate a mapping feature as a means of allowing the users to effortlessly access stores selling products matched to their special dietary needs. In spite of the fact that real-time mapping was compatible with our database, the complexity of data mapping across

diverse formats increased the cost and was too processing-intensive for real-time location processing.

Resolution: After several trails we decided to postpone this feature for a future update. This decision was made to ensure the stability and performance of the current app features were not compromised. The integration challenges came from the need to sync a lot of location data with user profiles and product availability in real time, which needed a more sophisticated backend architecture incorporating geospatial databases.

12.0 Version Control

We made a GitHub account for the Senior Project. This account serves as the repository for all code and documentation related to our project.Github Link:

https://github.com/Indy-9-Blue-Pure-Dine/blue-pure-dine

13.0 Conclusion

The "INDY 09 Blue - Pure Dine" initiative presents a new remarkable opportunity to make a difference for people having special dietary requirements. Our team has been working on building an application that is easy to use and encompasses all of the essential features such as linking dietary requirements and access for the foods as well as it is a delightful shopping platform for customers with specific preferences in their selection of Halal, Kosher, Vegan, & Vegetarian foods.

The main competitive advantage of our app is its powerful database, the intuitive UI, and the set of advanced filtering tools which provide an opportunity to easily pick up products that are appropriate for diet adherence. With technologies like Firebase for backend services and the OpenFoodFacts API supporting real-time product information our app has become both trustworthy and able to scale as it serves many more customers.

We have tackled a lot of things and as a result, we got priceless lessons. Among the major obstacles was to make our application user friendly and perform optimally when multiple configuration requests are concurrently submitted. This was made possible by consistently testing the codebase and making iterative improvements in our infrastructure and codes thus striking a balance between performance and functionality.

This project does not only meet its preliminary objectives, but it is the first step that leads to future innovations in the tech-food business. We feel proud about our collaborative deeds and we are optimistic about an improved life of specialized dietary needs individuals, which our application is going to contribute positively to.