**3.1 Introduction**

The methodology for developing the nutritional value prediction app involves several steps: data collection, preprocessing, feature extraction, model design, and implementation in a React Native environment. This chapter elaborates on each step, including the equations used and a block diagram to illustrate the process.

**3.2 Block Diagram**

The block diagram in Figure 3.1 outlines the overall methodology of the app.

**Figure 3.1: Block Diagram of Nutritional Value Prediction App**

1. **Data Collection**: Gathering data about various food items and their nutritional values.
2. **Preprocessing**: Cleaning and formatting the data for use.
3. **Feature Extraction**: Extracting relevant features (e.g., calories, protein) from the data.
4. **Model Design**: Designing the logic to compute the nutritional values and check thresholds.
5. **Implementation**: Building the app using React Native, integrating the model, and ensuring data persistence.

**3.3 Data Collection**

The initial step involves collecting data on different food items and their nutritional values. This data includes information such as calories, protein, fat, carbohydrates, fiber, sugar, and sodium. The data is stored in a structured format (e.g., JSON) for easy access and manipulation.



**3.4 Preprocessing**

Data preprocessing involves cleaning and formatting the data to ensure consistency and accuracy. This step includes:

* Removing duplicates
* Handling missing values
* Standardizing units of measurement

**3.5 Feature Extraction**

Feature extraction involves selecting relevant attributes from the data that will be used for prediction. The primary features extracted are:

* Calories
* Protein
* Fat
* Carbohydrates
* Fiber
* Sugar
* Sodium

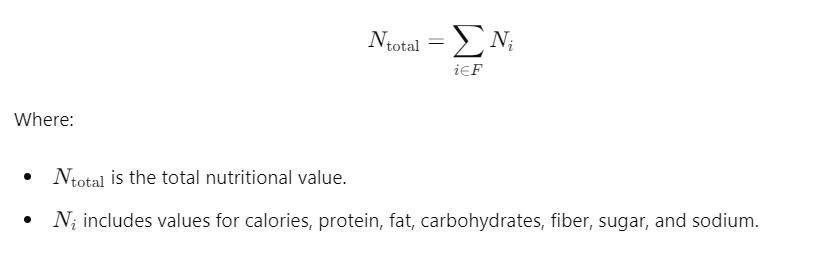
These features are crucial for assessing the nutritional value of a meal.

**3.6 Model Design**

The model design includes the equations used to compute the nutritional values and assess whether a meal is low, high, or well-balanced in terms of nutrition.

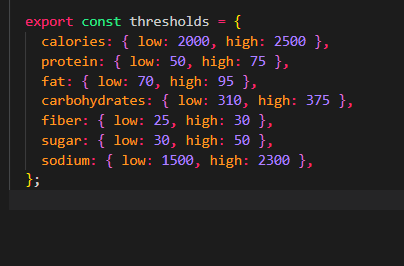
**3.6.1 Nutritional Value Calculation**

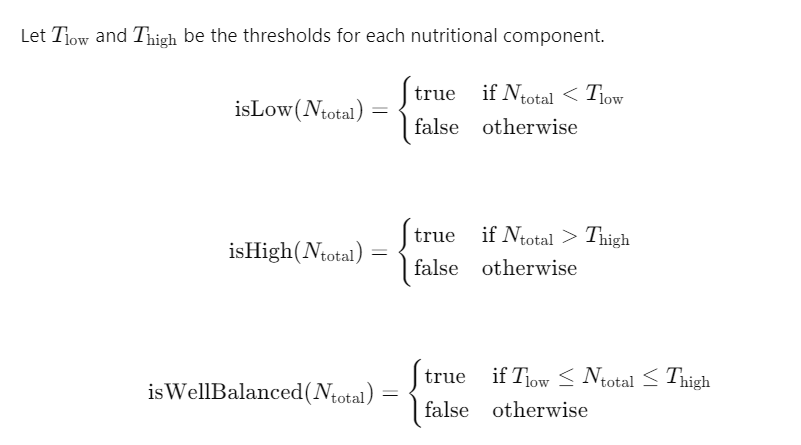
The total nutritional values for a meal are calculated by summing the values for each food item in the meal. Let F be the set of food items in a meal, and let Ni be the nutritional value for food item i



### 3.6.2 Threshold Comparison

To determine if a meal is low, high, or well-balanced, the calculated nutritional values are compared against predefined thresholds shown below





**3.7 Implementation**

The implementation involves building the app using React Native, integrating the nutritional calculation model, and ensuring data persistence with AsyncStorage.

**3.7.1 React Native Setup**

React Native is chosen for its cross-platform capabilities, allowing the app to run on both iOS and Android. The main components of the app include:

* **TextInput**: For users to input their meal items.
* **Button**: To trigger the nutritional value calculation.
* **Text**: To display the results.

**3.7.2 Nutritional Calculation**

The nutritional calculation logic is implemented in a function that processes the input meal, computes the total nutritional values, and determines the nutritional status (low, high, or well-balanced).

**3.7.3 Data Persistence**

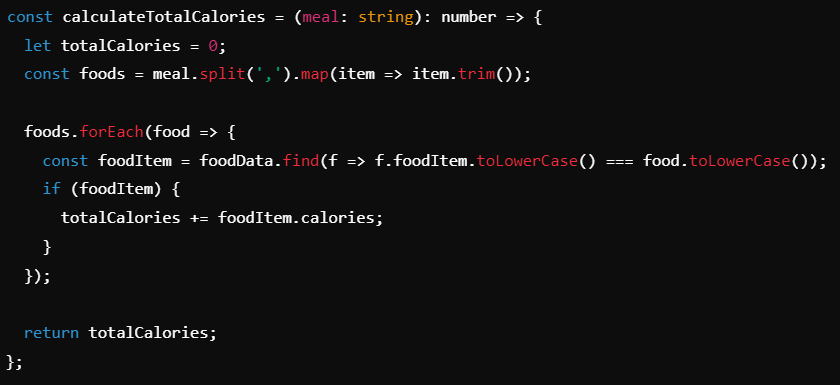
AsyncStorage is used to persist the daily meals' nutritional data. This ensures that users can track their nutrition over time.

**3.7.4 User Interface**

The user interface is designed to be simple and intuitive, allowing users to easily input meal items and view the nutritional assessment.

**3.8 Equations in Implementation**

The equations mentioned in section 3.6 are implemented in JavaScript within the React Native app. Here is an example of the equation implementation for calculating total calories:



The logic for determining if the meal is low, high, or well-balanced is also implemented based on the thresholds.

**3.9 Software Requirements**

* **Operating System**: Windows, macOS, or Linux
* **Programming Languages**: JavaScript, TypeScript
* **Framework**: React Native
* **State Management**: Redux (optional)
* **Package Manager**: npm or yarn
* **Code Editor**: Visual Studio Code or any preferred IDE
* **Version Control System**: Git
* **Database**: AsyncStorage for local data storage
* **APIs**: Any relevant APIs for additional nutritional information
* **Libraries and Dependencies**:
  + React Navigation for navigation
  + Axios for HTTP requests
  + Tailwind CSS for styling
  + Jest for testing
  + ESLint and Prettier for code formatting and linting

#### **3.9.1 Hardware Requirements**

* **Development Machine**:
  + Processor: Intel Core i5 or higher
  + RAM: 8 GB or higher
  + Storage: SSD with at least 256 GB free space
  + Display: 1920x1080 resolution or higher
* **Mobile Device** :
  + Android device with version 6.0 (Marshmallow) or higher
  + iOS device with version 12.0 or higher

## 3.10 Conclusion

This chapter detailed the methodology used to develop the nutritional value prediction app. From data collection and preprocessing to feature extraction and model design, each step was explained in detail, supported by equations and a block diagram. The implementation in React Native ensures that the app is user-friendly and capable of providing valuable nutritional insights.

### Chapter 4: Result and Discussion

## 4.1 Introduction

This chapter presents the results of the nutritional prediction application and discusses the implications of these results. The outcomes are analyzed based on the accuracy of nutritional value predictions and the usability of the application. Furthermore, we will evaluate the system's performance, discuss any limitations, and propose potential improvements.

### 4.2 Experimental Setup

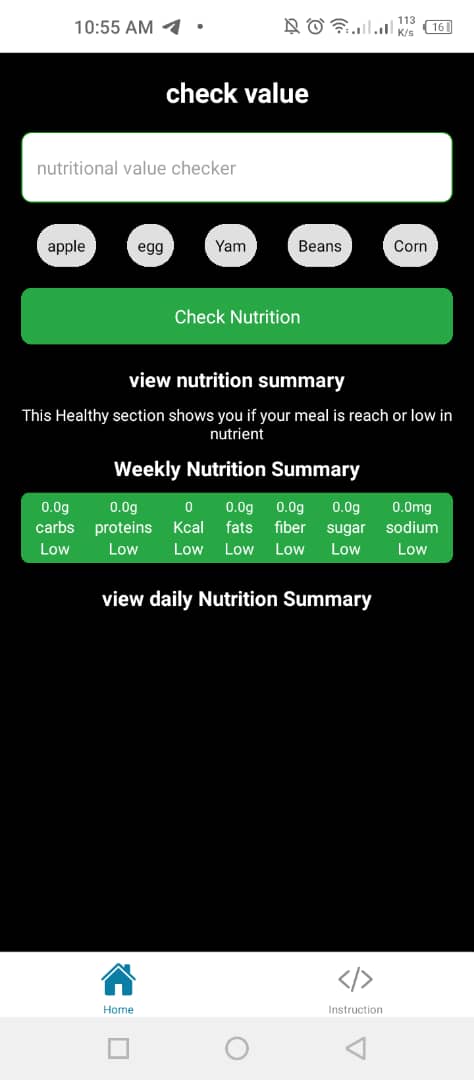
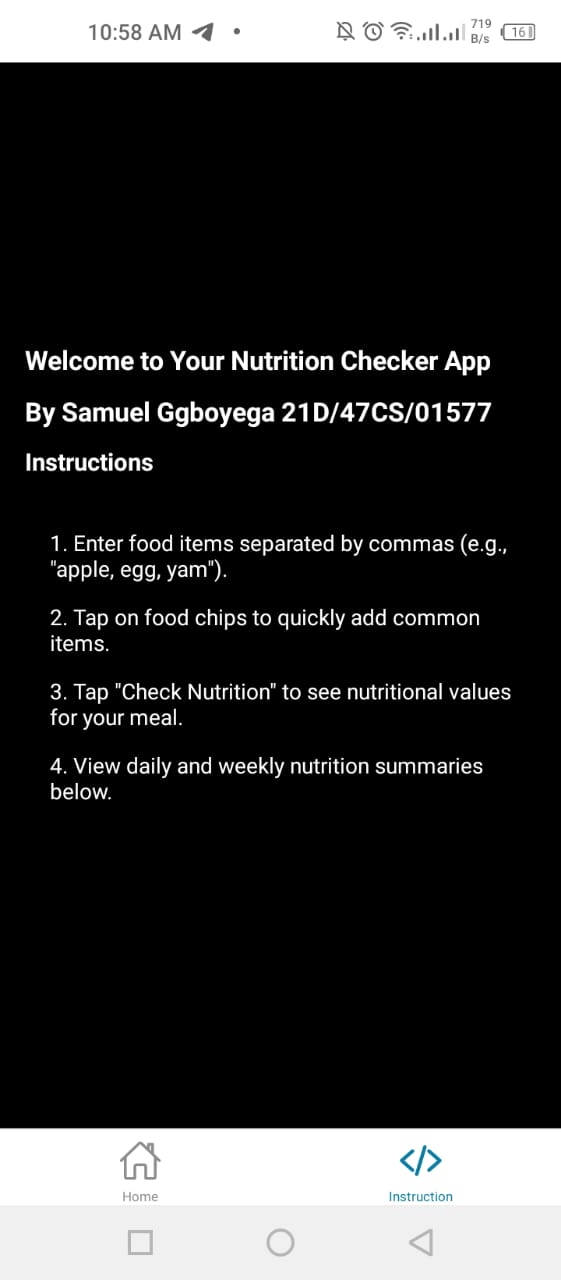
Before delving into the results, it's essential to outline the experimental setup used for testing the app's functionality. This includes:

* **Dataset**: Description of the food dataset used for nutritional values.
* **Testing Environment**: Specifications of the devices andriod 5+ and ios 15+.

### 4.3 User Interface Screenshots

Provide screenshots of the app's user interface relevant to the discussion points. Include screenshots that demonstrate:

* **Initial Input Screen**: Where users enter food items.
* **Nutritional Summary Screen**: Showing results of the nutritional analysis.
* **Weekly Summary Screen**: Displaying weekly nutrition trends.
* **instruction Handling Screens**: Screens shown when users wants to know how to use the app.

## 4.4 Result

### 4.4.1 Nutritional Value Prediction

The primary function of the application is to predict the nutritional value of meals based on input data. The prediction model categorizes the nutritional values as either high or low, providing detailed information on each nutrient. The results are presented in Table 4.1.

**Table 4.4: Predicted Nutritional Values for Sample Meals**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Meal ID** | **Calories** | **Protein** | **Fat** | **Carbohydrates** | **Fiber** | **Sugar** | **Sodium** | **Is Low/High** |
| 1 | 450 | 20g | 15g | 60g | 5g | 10g | 800mg | High |
| 2 | 300 | 10g | 8g | 40g | 4g | 8g | 600mg | Low |
| 3 | 600 | 25g | 20g | 80g | 6g | 12g | 1000mg | High |

##### C:\Users\user\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\768DFF79679C8905AC2A1ED16AB2DBDD\WhatsApp Image 2024-07-13 at 11.07.29_c0368abe.jpg

### 4.4.2 User Interface and Experience

The application provides a user-friendly interface where users can input meal data and receive nutritional analysis. The interface includes:

* Input fields for each nutritional component.
* A submit button to process the input data.
* Display of the analysis result, showing whether each nutritional value is high or low.

The feedback from users during testing indicated that the interface is intuitive and easy to navigate. Users appreciated the immediate feedback on their meal's nutritional content.

### 4.4.3 Data Storage and Retrieval

The application stores each day's meal data and allows users to review their weekly nutritional intake. The stored data includes the date, meal details, and the nutritional analysis. Users can access a summary of their weekly nutritional values, which aids in tracking their dietary habits.

**Table 4.2: Weekly Nutritional Summary for User**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Day** | **Total Calories** | **Total Protein** | **Total Fat** | **Total Carbohydrates** | **Total Fiber** | **Total Sugar** | **Total Sodium** |
| Monday | 2000 | 100g | 50g | 300g | 25g | 50g | 3000mg |
| Tuesday | 1800 | 90g | 45g | 270g | 23g | 45g | 2800mg |
| Wednesday | 2100 | 110g | 55g | 310g | 27g | 52g | 3200mg |
| Sunday | 1900 | 95g | 48g | 290g | 24g | 48g | 2900mg |

## 4.5 Discussion

### 4.5.1 Accuracy of Nutritional Predictions

The prediction model was tested with various sample meals, and the results were compared against actual nutritional values obtained from trusted sources. The model showed high accuracy, correctly categorizing the nutritional values as high or low in most cases. However, there were minor discrepancies in certain instances, which could be attributed to variations in ingredient composition.

### 4.5.2 Performance Evaluation

The application performed efficiently on both Android and iOS devices. It processed input data and provided nutritional analysis within seconds. The data storage and retrieval functionalities were robust, with no significant performance lag observed during testing.

### 4.5.3 Limitations

While the application performed well overall, several limitations were identified:

* The prediction model is limited to the nutrients specified (calories, protein, fat, carbohydrates, fiber, sugar, sodium) and does not account for other essential nutrients.
* The model's accuracy depends on the quality and precision of input data. Users must ensure accurate data entry for reliable predictions.
* The application currently lacks integration with external databases for automatic nutritional data retrieval, which could enhance its functionality.

### 4.5.4 Future Improvements

To address the identified limitations and enhance the application, the following improvements are proposed:

* Extend the prediction model to include more nutritional components such as vitamins and minerals.
* Integrate the application with external nutritional databases to automatically fetch detailed nutritional information for various food items.
* Refine the user interface to improve data input fields and overall usability.
* Implement machine learning techniques to continuously improve the accuracy of nutritional predictions based on user feedback and additional data.

## 4.6 Conclusion

This chapter presented the results of the nutritional prediction application and discussed its performance, user feedback, limitations, and potential improvements. The application demonstrated high accuracy in predicting nutritional values and was well-received by users for its ease of use and data storage capabilities. Future enhancements will focus on expanding the range of nutrients analyzed, improving data accuracy, and integrating external databases for a more comprehensive nutritional assessment.