1.5 Core Functional and Non-functional Requirements

Functional Requirements:

Category	Description	Priority	Dependency(s)
1. The system must:	1. Be able to identify products and their ingredients in a store	High	
	2. Be able to fit a wide range of different dietary restrictions	High	
	3. Identify and flag product ingredients (direct and indirect associations) specified by users	High	
	4. Have an extensive database of products and allergens	High	
	1. Receive regular updates to the database		
2. The system should:	1. Be able to quickly identify products	High	FR 1.1, 1.2
	2. Include common allergens and dietary choices (e.g vegan) in the database	High	FR 1.4
	3. Clearly display details about flagged products and highlight the ingredients that clash with user preferences	High	FR 1.2, 1.3, 1.4
	4. Allow the user to specify pre-configured groups of ingredients to avoid	Medium	FR 1.1, 1.2
	1. Allow the user to create personal profiles with specifications, protected by passwords		
	5. Allow users to look back at previous scans	Medium	FR 1.2, 1.4
	1. Allow users to report products and update database		
3. The system could:	1. Have a list feature to be used for shopping	Medium	FR 1.1
	2. Be able to group/sort purchased products	Low	FR 1.1
4. The system (most likely) will not:	1. Be able to suggest cheaper alternatives	Low	
	2. Be able to suggest more nourishing alternative products or potential accompanying products	Low	FR 1.1
	3. Be able to suggest healthier alternative products	Low	FR 1.1

Non-Functional Requirements:

Category	Description	Priority	Dependency(s)
1. Performance:	1. The product scanning process should be fast and responsive, providing results within a reasonable timeframe	High	FR 1.1, 2.1
	2. In-app functions should be low-latency	High	
2. Reliability	1. The system should be robust and stable with minimal failures	High	
	2. It should be able to handle various product labels in different conditions (e.g damaged packaging)	High	FR 1.1, 2.1
	3. The system should be highly accurate in product recognition and information retrieval	High	
	4. The app should use error-handling mechanisms for cases where product information is not available	High	
3. Usability	1. The product scanning feature should be quick, accurate, and easy to use	High	FR 1.1, NFR 1.1, 2.3
	2. The app should have an intuitive and user-friendly interface to accommodate users with varying levels of technological expertise	High	
	1. Navigation within the app should be easy and straightforward		
4. Compliance	1. The app should comply with relevant data protection and privacy regulations (e.g. GDPR)	High	
	2. It should adhere to food industry standards for ingredient labelling and information	High	
5. Security	1. User data, especially flagged ingredients and dietary preferences must be stored securely and protected from unauthorised access	High	FR 1.4, 2.4.1, NFR 4.1
	1. Hashing and encryption should be used to store user data securely		
	2. Data storage should comply with relevant data protection regulations		
	2. The app should employ secure communication protocols to transmit data between the user's device and the server	High	
	3. The app should be secure against unauthorised access and data breaches	High	FR 1.4
	1. The database must be secure against common attacks (e.g SQL injection)		

NFR Continued...

6. Accessibility	1. The system should be usable by individuals with impaired vision or other disabilities 1. It should comply with accessibility standards (e.g WCAG) 2. The app could support screen readers and provide alternative text for images and buttons.	High Medium	NFR 3.1, 3.2
7. Maintainability	1. The codebase should be well-documented and modular, allowing for ease of maintenance and updates 2. Regular updates and patches should be delivered to address bugs, security vulnerabilities, and improve functionality	High Medium	
8. Compatibility	1. The app should be compatible with a wide range of mobile devices and operating systems (iOS, Android) 2. The app should be regularly updated to maintain functionality with newer operating systems and devices	High Medium	 NFR 7.2
9. Interoperability	1. The app should be designed to work seamlessly with other health and wellbeing apps or platforms 2. It should be able to integrate with external databases or APIs to access the most up-to-date information about food products	High Medium	 FR 1.4, NFR 5.2
10. Scalability	1. The system should be able to handle a growing user base and an expanding database of flagged ingredients without significant degradation in performance 2. The app architecture could be designed to accommodate potential future expansions and updates	Medium Medium	FR 1.4
11. Performance Monitoring	1. Tools and processes could be implemented for monitoring the app's performance and optimising resource usage	Medium	
12. Backup and Recovery	1. The app could regularly backup user data to prevent data loss in case of system failures 2. There could be a robust recovery plan in place to minimise downtime and data loss in the event of unexpected incidents	Medium Low	

1.6 Conflict Management

In our approach to prioritising functional requirements, the adoption of the MoSCoW technique (Must-Have, Should-Have, Could-Have, Won't-Have) played a pivotal role in simplifying our decision-making process. This method not only streamlined the prioritisation workflow but also provided an intuitive framework for addressing conflicts among the functional requirements. Complementing this, we enhanced precision by assigning a secondary priority level (high, medium, or low) to each requirement, offering a nuanced perspective on project objectives. Extending this systematic approach to non-functional requirements, we organised them using their assigned priority levels as well as their category labels. This strategy proved effective, ensuring that essential qualities such as performance, security,