Bilkent University
Department of Computer Engineering

Senior Design Project

Final Report

CONTENTA

GROUP #T2325

Ömer Oktay Gültekin | 21901413 | oktay.gultekin@ug.bilkent.edu.tr
Oğuz Kuyucu | 21902683 | oguz.kuyucu@ug.bilkent.edu.tr
Barış Tan Ünal | 22003617 | tan.unal@ug.bilkent.edu.tr
Mert Ünlü | 22003747 | mert.unlu@ug.bilkent.edu.tr
Alperen Utku Yalçın | 22002187 | utku.yalcin@ug.bilkent.edu.tr

Supervisor: Fazlı Can
Innovation Expert: Tağmaç Topal
Instructors: Mert Bıçakçı & Atakan Erdem
13.05.2024

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

ABSTRACT

Contenta revolutionizes consumer-product interactions through advanced image recognition, swiftly identifying allergens, harmful ingredients, and nutrition facts to calculate EU-standard Nutri-Scores from product labels. Unlike existing apps, *Contenta*'s product-independent approach eliminates constraints imposed by absent barcodes or outdated databases. Personalized features integrate individual health profiles, empowering users to make informed choices. Initially tailored for English, *Contenta* seamlessly adapts to other languages, ensuring global accessibility. Future plans include expansion into cosmetics and cleaning products, advancing its mission to promote healthier, informed consumption worldwide.

This detailed report outlines the comprehensive development and deployment blueprint for *Contenta*, covering detailed architecture, design, development, and implementation specifics. It delves into the integration of key technologies such as Flutter, Amazon Web Services (AWS), Command R+, Python, and Jira, offering insights into the application's technical foundation. Furthermore, it provides an extensive coverage of test cases, maintenance plans, analysis elements, and teamwork dynamics, culminating in a conclusion outlining future prospects and enhancements. A user manual and installation guide, along with a glossary and references, complete this exhaustive examination of Contenta's development journey.

Keywords: packaged food, human health, ingredient analysis, Nutri-Score, informed consumption, image-processing, AWS.

TABLE OF CONTENTS

LIST OF FIGURES & TABLES

1.0 INTRODUCTION

Purchasing packaged goods has become a routine for many individuals, and with increasing consumer awareness, there's a growing desire to understand the contents of the products they consume. Some consumers find themselves carefully scrutinizing product labels, particularly the ingredients section, to make informed choices based on their individual needs and preferences. It's crucial for consumers to be mindful of potential allergens or substances that may pose health risks, even in products deemed safe by regulatory bodies.

However, understanding ingredient labels can be challenging, especially for those with limited knowledge of specific substances, their origins, or potential health implications. As long as someone is not an expert like a medical doctor or chemist, those chemical names in the label are hard to understand and interpret. Customers who have intolerances to specific substances, who could be seriously harmed if they were to ingest a particular product, are particularly impacted by this problem [1]. This difficulty is particularly impactful for individuals with intolerances or dietary restrictions, such as those avoiding certain ingredients like some animal products, corn syrup, gluten, or others. Additionally, there are lesser-known ingredients that may pose potential risks, further complicating the decision-making process for consumers. Besides, people with astigmatism may not easily read the label because the writings are too small for people who have astigmatism.

Fig. 1: Rear face of light raspberry yogurt [2].

Given the complexity of ingredient labels and the inherent risks of consuming certain substances, consumers may inadvertently purchase products that could harm their health, despite their best efforts to read labels carefully. For example, an ingredient like "Acesulfame Potassium," commonly found in sugar-free or ‘light’ products, may pose health risks such as those related to pregnancy and cancer, as indicated by research [2][3]. Awareness of such hazards is crucial for consumers who prioritize their well-being and wish to make informed choices about the products they consume. *Contenta* is an app designed to address these challenges by providing users with transparency regarding the contents of the products they consume. By offering quick and efficient access to information about product ingredients, *Contenta* aims to empower users to make more informed decisions about their everyday consumption.

This report outlines the requirement details, starting with an explanation of its purpose and design goals. It provides a detailed description of the final software architecture for *Contenta* as a standalone application. The report also discusses the implementation details and reviews the test cases to demonstrate expected system behavior and results in various scenarios. Additionally, it addresses constraints, standards, and details of the teamwork process involved in the development of *Contenta*. Finally, it gives a brief demonstration of the user manual and an installation guide.

2.0 REQUIREMENT DETAILS

Contenta is a mobile application designed to analyze food packaging for consumers. It consists of two main functionalities: package scanning with dietary analysis, and informative blogs. The application caters to individual users and expert bloggers. Users will download the app to scan food packages and receive detailed analyses regarding the compatibility of ingredients and additives with their personal dietary preferences, which they’ve configured in the app. Users can also see the Nutri-Score of the packaged food that they scanned, whose formula was created by Sante Publique France [4].

For the blogging functionality, food experts can submit educational content which becomes available to all app users. This aims to increase user engagement and provide valuable dietary and health information, enriching the user experience. Upon scanning a food item, the app will cross-reference the detected ingredients with the user's dietary settings and deliver a compatibility report. This feature supports various dietary needs, such as paleo, vegan, or keto diets. The application will alert users about any unsuitable ingredients based on their dietary settings. The app requires secure handling of personal dietary preferences and health-related data, adhering to data protection laws like GDPR and KVKK [5]. Security measures will be robust to maintain user confidentiality and data integrity.

Fig. 2: Conceptual drawing of a cosmetic product with a focus on the ingredients section [6].

In summary, the *Contenta* serves dual purposes: providing personalized dietary information through a scanning technology and fostering a community of informed eaters through its blogging feature. This dual approach not only enhances the individual's eating habits but also builds a platform for shared knowledge about health and food science.

3.0 FINAL ARCHITECTURE & DESIGN DETAILS

3.1 FRONTEND ARCHITECTURE

While structuring our Flutter application's frontend architecture, we have adopted the Model-View-Controller (MVC) design pattern to create a well-organized and efficient codebase. This approach divides our application logic into three separate layers where each is fulfilling a specific role to ensure smooth operation and functionality.

At the core of our architecture are the model classes, responsible for managing data within the application. These classes handle data storage, retrieval, and manipulation, promoting code reusability and modularity. By segregating data-related operations into dedicated model classes, we maintain separation of concerns and facilitate codebase maintenance and scalability.

The view layer focuses on rendering the user interface and presenting data to users. Using Flutter's extensive widget library, we've crafted an engaging and interactive interface that smoothly integrates with the application's logic. Through thoughtful design and composition, we enhance the user experience while adhering to UI development best practices.

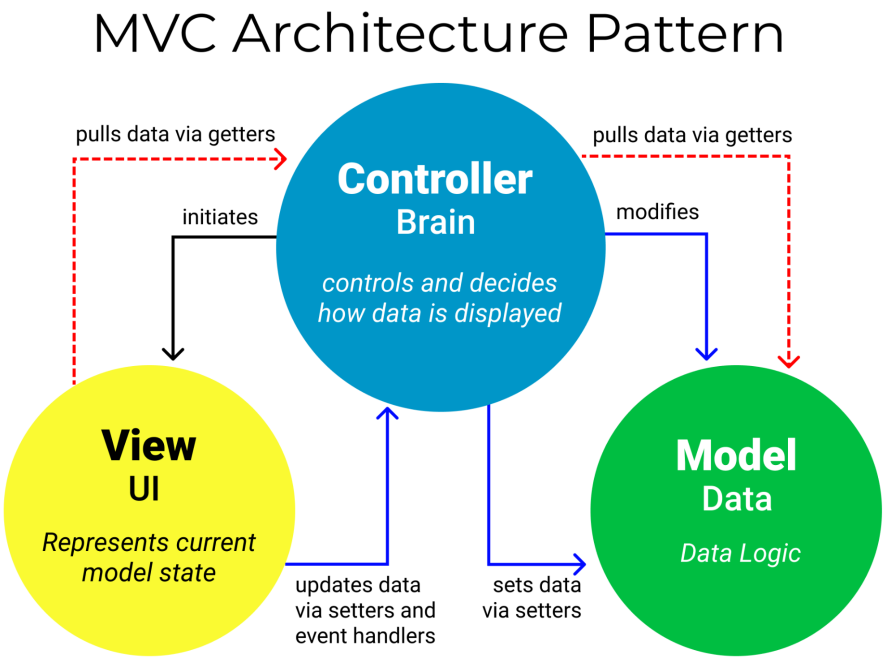

Fig. 3: Model-View-Controller Pattern [7].

Serving as the man in the middle, between the model and view layers, the controller layer manages data flow and user interactions. Controllers ensure the integrity and coherence of the application's state by mediating between user input and data manipulation. By adopting the GetX framework for state management, we embrace a reactive programming model that promotes a single source of truth and facilitates seamless state propagation. Our architectural approach prioritizes code reuse and modularity. Encapsulating analysis results and related components within dedicated model classes fosters a modular and extensible codebase conducive to iterative development. In addition, strategic use of Flutter's widgets enables us to create a dynamic and responsive user interface adaptable to different devices and interactions.

3.2 BACKEND ARCHITECTURE

The backend architecture on AWS maximizes the use of Amazon Cognito User groups for efficient user sign-in management. User access is regulated by Identity and Access Management (IAM) System Roles directly assigned to Cognito user groups. API Gateway, accessed through secure HTTPS connections, serves as the gateway to app functionalities. Security measures are bolstered with AWS WAF protection for API Gateway against vulnerabilities like DDOS attacks. To handle fluctuations in demand, API endpoints autonomously scale to meet user requirements, enhancing app scalability.

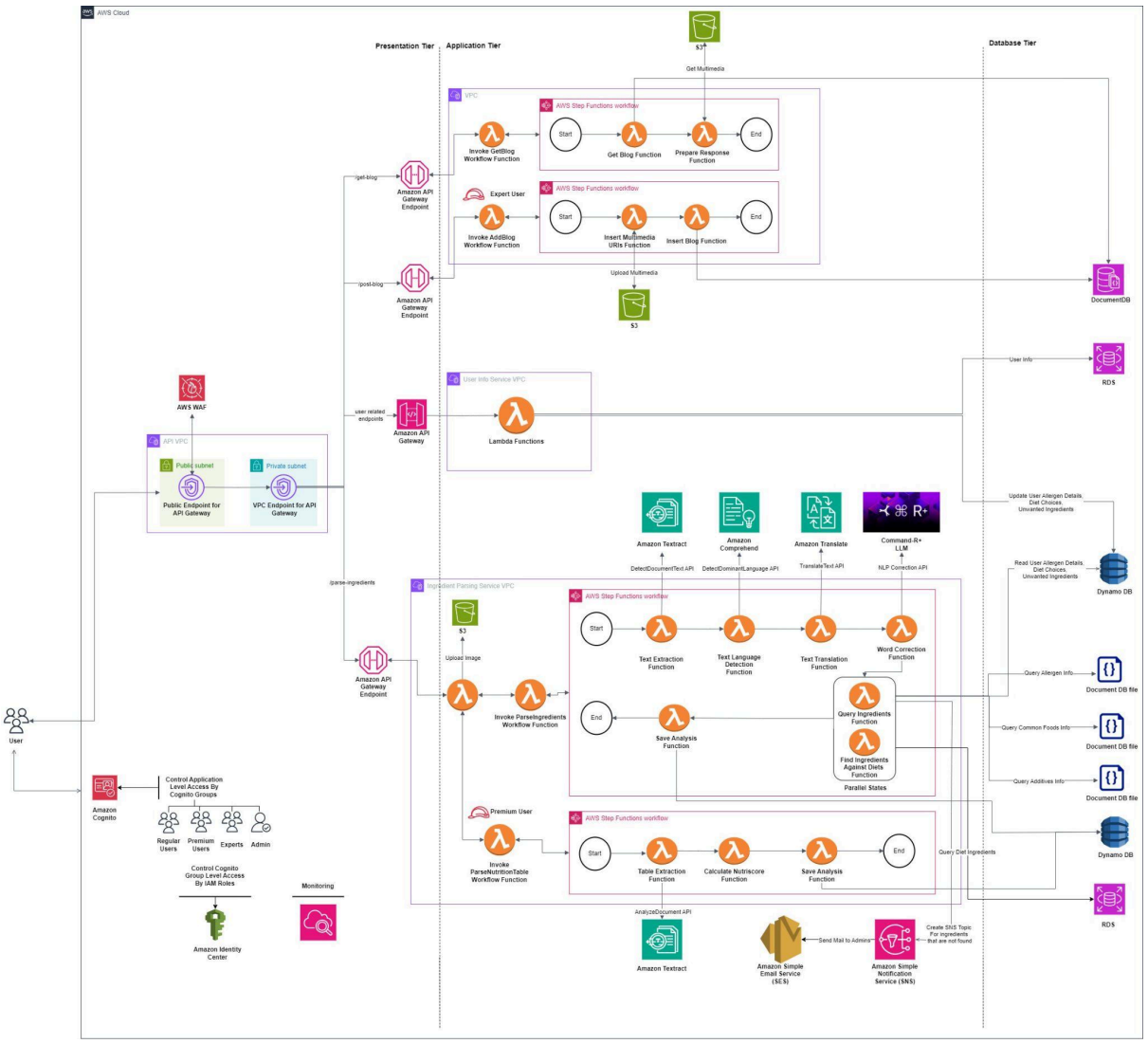

Fig. 4: Contenta's AWS backend pipeline.

At the core of the app's functionality lies Step Functions, orchestrating the main app pipeline. Lambda functions, operating serverlessly, execute sequentially or in parallel as per requirements. This pipeline harnesses Amazon Textract for text extraction from images, Comprehend for language detection, Translate for text translation if necessary, and Cohere's Command-R Plus LLM model for ingredient

extraction. Parallel states cross-reference extracted ingredients with database information. Unmatched ingredients trigger notifications via Amazon Simple Notification Service (SNS), prompting admin intervention to update databases. Amazon CloudWatch archives pipeline logs, creating metrics for error detection and automatic notification to admins.

Image storage utilizes Simple Storage Service (S3). Depending on use cases, DynamoDB, MySQL in Relational Database Service (RDS), and DocumentDB are employed. AWS security measures include restricting access to private resources within a Virtual Private Network (VPC) and safeguarding database credentials within lambda function environment variables. Access to the AWS architecture is channeled exclusively through defined API endpoints, requiring requisite security credentials for entry.

4.0 DEVELOPMENT & IMPLEMENTATION DETAILS

After conducting a thorough requirements analysis, we have discussed the most suitable technologies to use for *Contenta*. We have come up with the following:

4.1 FLUTTER

Opting for Flutter as the development framework for *Contenta*'s frontend offers numerous advantages tailored to the project's needs. Firstly, Flutter's open-source nature aligns with *Contenta*'s principle of transparency and accessibility, allowing for collaboration and innovation within the developer community. By enabling the creation of cross-platform applications from a single codebase, Flutter facilitates the development process for *Contenta*, saving time and resources while ensuring consistency across different platforms. This multi-platform capability is particularly advantageous for *Contenta*, as it aims to reach users across various devices and operating systems, including Android and iOS.

Flutter's rich set of customizable UI widgets and fast rendering engine empower developers to create visually appealing and responsive user interfaces, enhancing the user experience of *Contenta*'s app. Additionally, Flutter's hot reload feature enables quick iteration and experimentation during the development phase, facilitating rapid prototyping and iteration to refine *Contenta*'s features and functionality.

4.2 AMAZON WEB SERVICES (AWS)

Contenta strategically makes use of Amazon Web Services (AWS) to power its backend infrastructure, harnessing a suite of services tailored to support its diverse needs. Firstly, AWS offers a

highly scalable and reliable cloud computing environment as they guarantee availability from 99.9% to 99.999% [8].

AWS Lambda functions serve as the backbone of *Contenta*'s serverless architecture, enabling the execution of code without the need to provision or manage servers, ensuring scalability and cost-efficiency. By deploying Python scripts as AWS Lambda functions, *Contenta* can perform tasks such as Nutri-Score calculation and data processing, while dynamically scaling resources based on demand.

Amazon RDS (Relational Database Service) and DynamoDB play pivotal roles in *Contenta*'s data management strategy, providing scalable and reliable solutions for storing and accessing structured and unstructured data, respectively. RDS offers *Contenta* a managed relational database solution, ensuring high availability, durability, and automated backups for critical data related to user profiles, product information, and transactions. DynamoDB, on the other hand, supports *Contenta*'s need for fast and flexible NoSQL database capabilities, enabling rapid access to data for features like real-time ingredient parsing and analysis.

Amazon Textract emerges as a key component of *Contenta*'s text extraction pipeline, empowering the app to extract structured data from scanned documents and images with ease and accuracy. By integrating Textract into its workflow, *Contenta* integrates processes such as ingredient parsing and label analysis, enhancing user experience and efficiency. Moreover, *Contenta*'s reliance on AWS services extends to data storage and retrieval, with Amazon S3 serving as a scalable object storage solution for housing multimedia content, such as product images and user-generated content. This smooth integration with S3 enables *Contenta* to store and retrieve data securely and reliably, ensuring optimal performance for its users.

Overall, AWS serves as a robust foundation for *Contenta*'s backend infrastructure, providing a comprehensive set of services that support its core functionalities, from serverless computing and data management to text extraction and storage. By utilizing AWS, *Contenta* ensures scalability, reliability, and performance, enabling the app to deliver a seamless and efficient experience for its users.

4.3 FIREBASE

Initially, the decision to integrate Google’s Firebase into *Contenta*'s backend infrastructure was driven by its reputation for offering a comprehensive suite of backend services, including real-time database, authentication, cloud functions, and cloud messaging, tailored for mobile and web applications.

Firebase provided *Contenta* with a convenient and scalable solution for managing user data, facilitating authentication, and enabling real-time updates, which were essential features for the application's functionality and user experience.

However, as *Contenta*'s development progressed and the decision was made to establish an AWS pipeline for backend operations, it became evident that using Amazon services would offer better compatibility, scalability, and integration with the chosen infrastructure. By transitioning away from Firebase and adopting Amazon RDS (Relational Database Service) and DynamoDB for database management, *Contenta* could smoothly integrate backend operations into its AWS pipeline, ensuring efficient data storage, retrieval, and scalability while utilizing the broader ecosystem of AWS services for enhanced functionality and performance. This strategic shift from Firebase to Amazon services aligns with *Contenta*'s goal of optimizing compatibility, scalability, and performance within its AWS-based infrastructure, ensuring a cohesive and streamlined backend architecture that supports the application's evolving needs and requirements.

4.4 GOOGLE ML KIT

Initially, the decision to utilize Google ML Kit for *Contenta*'s text extraction and parsing needs was driven by its reputation for providing powerful machine learning capabilities tailored for mobile applications. Google ML Kit offered a convenient and user-friendly solution for implementing text recognition and understanding functionality within the app, allowing *Contenta* to extract and parse information from product labels with ease. Its integration with Firebase, Google's mobile development platform, provided additional benefits such as cloud-based processing and smooth deployment across Android and iOS devices.

However, as *Contenta*'s development progressed and the decision was made to build a pipeline on AWS, it became apparent that using Amazon services would offer better compatibility and integration with the chosen infrastructure. Amazon Textract emerged as a compelling alternative to Google ML Kit, offering robust text extraction capabilities with native integration with AWS services. By transitioning to Amazon Textract, *Contenta* could smoothly integrate text extraction into its AWS pipeline, ensuring efficient processing and scalability while utilizing the broader ecosystem of AWS services for enhanced functionality and performance. Additionally, the decision to incorporate Command R+ for parsing and understanding labels further reinforced the compatibility with AWS, enabling seamless integration with the chosen infrastructure and ensuring a cohesive development process for *Contenta*.

4.5 COMMAND R+

Selecting Command R+ for *Contenta*'s crucial tasks of ingredient list and nutritional facts table parsing serves as the backbone of the project for several compelling reasons. Firstly, Command R+ offers exceptional performance and reliability, optimized specifically for complex tasks and long-context interactions. Its ability to handle multi-step tool use (agents) makes it particularly well-suited for *Contenta*'s requirements, allowing for intricate workflows and sophisticated analyses. Moreover, Command R+'s extensive multilingual capabilities align with *Contenta*'s goal of catering to a diverse user base, ensuring that it can effectively process and generate content in various languages. Additionally, Command R+ excels in retrieval augmented generation (RAG), a feature crucial for *Contenta*'s tasks such as ingredient parsing, where it can ground its responses based on supplied document snippets, enhancing the reliability and relevance of generated information. Furthermore, Command R+'s training on a vast corpus of diverse texts, coupled with its focus on excelling in critical business use-cases, ensures that it can deliver high-quality results for *Contenta*'s users.

4.6 PYTHON

Python's inclusion in *Contenta*'s development stack for scripting purposes, such as Nutri-Score calculation and web scraping from the FDA website for the additive database, is strategic due to its versatility, simplicity, and extensive ecosystem of libraries and tools. Python's readability and ease of use make it ideal for rapidly prototyping and developing scripts, allowing *Contenta* to efficiently implement essential functionalities like nutri-score calculation with minimal overhead. Additionally, Python's rich collection of libraries, including NumPy and pandas, facilitates data manipulation and analysis, essential for tasks like nutri-score calculation and data extraction from the FDA website. Moreover, Python's web scraping libraries, such as BeautifulSoup and Scrapy, enable us to retrieve and parse data from the FDA website efficiently while populating the dataset of the *Contenta*, supporting the creation and maintenance of the additive database. We used the json library at the preparation stage of the dataset. Lastly, we used the re library for regular expressions to prioritize E numbers in synonyms.

Integrating Python scripts into *Contenta*'s AWS pipeline further enhances the application's functionality and scalability. By using AWS Lambda, *Contenta* can execute Python scripts in a serverless environment, eliminating the need to manage infrastructure and scaling resources dynamically based on demand. This serverless architecture aligns with AWS best practices, ensuring cost-effectiveness and operational efficiency for *Contenta*'s backend operations. Additionally, Python's compatibility with AWS services like S3 and DynamoDB simplifies data storage and

retrieval within the AWS ecosystem, enabling integration with other components of *Contenta*'s infrastructure.

4.7 JIRA

We effectively manage our project workflow and task allocation using Jira, utilizing its powerful project management features to simplify collaboration and track progress. By utilizing Jira's agile boards, we organize work into epics, breaking down tasks into manageable subtasks assigned to team members. Each task is meticulously defined with clear assignees, reviewers, and due dates, ensuring accountability and transparency throughout the development process. This structured approach facilitates efficient task allocation, fosters collaboration among team members, and enables timely delivery of project milestones. Overall, Jira serves as a central hub for our project management, empowering the team to effectively coordinate efforts, prioritize tasks, and achieve project objectives in a systematic and organized manner.

5.0 TEST CASES

This section demonstrates some crucial test cases that we have performed.

5.1 FUNCTIONAL TEST CASES

Test ID: TF01
Type: Functional
Title: Registration and Login
Test Result: Success
Covered Requirement(s):

- *Sign Up:* A user can sign up with an email and password.
- *Login*: A user can log in from different devices by using email and password.
- *Forgot Password:* A user can get an email to change their password whenever they forget it.

Entry Criteria: The application is up and running and the user is not logged in.

Test Steps	Outcome
User clicks the register button.	Registration form is displayed.
User enters the first name, last name, email, password, weight, height, diet preferences, and allergies.	Confirmation mail is sent.
User confirms the confirmation mail.	User gets registered and the main page is

	displayed.
User presses the sign off button from the sidebar.	User is signed off and the login page is displayed.
User enters their email and password.	User is logged in and the main page is displayed.
User presses the sign off button from the sidebar.	User is signed off and the login page is displayed.
User clicks the “forgot password” button.	A temporary password is sent to the user's mailbox.
User logs in by entering their email and the temporary password.	User is logged in and the main page is displayed.

Test ID: TF02

Type: Functional

Title: Change Password and Delete Account

Test Result: Success

Covered Requirement(s):

- *Delete Account:* A user can delete their account. Their personal information will be deleted in this case.
- *Change Password:* When logged in, a user can change their password by using the old password.

Entry Criteria: The application is up and running and the user is logged in.

Test Steps	Outcome
User clicks change password button.	The user is redirected to the change password page which demands current password and new password.
User types old password and new password.	If the current password is wrong, an error which informs the user will appear. If current password is right, the password will be changed
User clicks delete account button.	A page which demands the current password appears.
User types his/her current password and clicks delete account button	If the password is right, the user will be asked “are you sure” If the current password is wrong, an error which informs the user will appear.

User clicks “I am sure” button	User’s account along with all related information will be deleted.

Test ID: TF03

Type: Functional

Title: Premium Membership

Test Result: N/A

We have not implemented premium membership functionality yet, therefore TF03 is not applicable.

Covered Requirement(s):

- *Become Premium Member:* Users can become a premium member by subscribing.
- *Terminate Subscription:* A premium user can stop his/her subscription.

Entry Criteria: User is logged in and is not a premium member.

Test Steps	Expected Outcome
User clicks the “become a premium member” button.	A page which informs user about the advantages of premium members will appear.
User clicks the continue button.	User is redirected to the payment page.
User types his/her credit card information.	Payment API processes the information and returns the result. If payment is successful, the membership is upgraded, if not, an error which informs the user appears.
User clicks terminate subscription button.	User is redirected to “terminate subscription” page which asks the user “are you sure”
User clicks yes.	The subscription is terminated and no more payment will be taken from the user.

Test ID: TF04

Type: Functional

Title: Add Allergy / Unwanted Ingredient

Test Result: Success

Covered Requirement(s):

- *Add Allergy:* A user can add an allergy from the allergies list. The application will notify the user if the user scans a food that contains an ingredient that may trigger the allergy.
- *Add Unwanted Ingredient:* A user can select an additive from the list and get a warning when they scan food ingredients and the application detects the ingredient.

Entry Criteria: User is logged in.

Test Steps	Outcome
User clicks the add allergy button.	List of allergies will appear.
User chooses an allergy and clicks the done button.	The allergy is added.
User scans an ingredient which may potentially trigger the user's allergy/allergies.	User is warned.
User clicks the add unwanted ingredient button.	List of ingredients with a search bar appears.
User chooses an ingredient and clicks the done button.	Unwanted ingredient is added.
User scans an ingredient which has an unwanted ingredient.	User is warned.

Test ID: TF05

Type: Functional

Title: Scan Ingredients

Test Result: Success

Covered Requirement(s):

- *Scan Ingredients:* A user can scan food's ingredients and then select the desired language area.. The application will detect if there are any ingredients that may trigger the user's allergies. The application will warn users if there are any potentially harmful substances in the product.

Entry Criteria: User is logged in.

Test Steps	Outcome
User clicks the scan ingredients button.	The camera is open.
User takes a photo of the package's ingredient part.	The image appears with a cropping option.
User crops the photo and gets rid of irrelevant parts and clicks the done button.	The application shows the ingredients safety level and provides all other relevant information.

Test ID: TF06

Type: Functional

Title: Scan Nutrient Facts Table

Test Result: Success

The latest version of *Contenta* is doing TF05 and TF06 simultaneously. Therefore, there is no need to separate them as different functional requirement tests. However, since we have separated them in the Detailed Design Report, we did not want to delete TF05 or TF06 or merge them into a new functional requirement for standardization purposes.

Covered Requirement(s):

- *Scan Nutrition Facts Table:* A premium member can scan the food nutrients part in the label. The application will detect the Nutri-Score (A, B, C, D, or E) of the food which will be calculated according to EU health legislation.

Entry Criteria: User is logged in and user has premium subscription.

Test Steps	Outcome
Premium user clicks scan nutrient facts button.	The camera is open.
User takes a photo of package's nutrient information part	The image appears with a cropping option.
User crops the photo and gets rid of irrelevant parts and clicks the done button.	The application shows the Nutri-score.

Test ID: TF07

Type: Functional

Title: Promote / Demote User

Test Result: Success

Covered Requirement(s):

- *Promote User*: An admin can promote a user to an expert.
- *Demote User*: An admin can demote an expert to a user.

Entry Criteria: The user is logged in and should have admin authorization.

Test Steps	Outcome
Admin clicks the view candidate experts button.	The user list which shows the users who applied to become an expert appears.
Admin selects a user and clicks the approve button.	User is promoted to an expert.
Admin clicks the view experts button.	List of experts appears.
Admin selects an expert and clicks the demote button.	Expert is demoted to user.

Test ID: TF08

Type: Functional

Title: Read Blog

Test Result: Success

Covered Requirement(s):

- *Read Blog*: Users can read blogs about specific ingredient(s) or nutrient facts which are written by experts.

Entry Criteria: The user is logged in.

Test Steps	Outcome
User clicks the view blogs button.	List of blogs with a search bar appears.
User selects a blog.	The content of the blog appears.
User clicks the back button while reading a blog.	List of blogs with a search bar appears.

Test ID: TF09

Type: Functional

Title: Add / Remove Bookmark

Test Result: Success

Covered Requirement(s):

- *Bookmark:* A user can bookmark a blog article.
- *Remove Bookmark*: A user can remove the bookmark of a blog article.

Entry Criteria: User is logged in.

Test Steps	Outcome
User clicks the view blogs button.	List of blogs with a search bar appears.
User selects a blog.	The content of the blog appears. At the top of the page, the bookmark icon appears. If the blog is already in the bookmarks, inside of this button will be colored, else white.
User clicks the bookmark button.	If the blog is in bookmarks, it is removed. If it is not in the bookmarks, it is added to bookmarks.

Test ID: TF10

Type: Functional

Title: Write Blog

Test Result: Success

Covered Requirement(s):

- *Write Blog*: Experts who are confirmed by admin can write blog articles about some ingredients or nutrition facts.

Entry Criteria: User is logged in and has expert authorization.

Test Steps	Outcome
An expert clicks the write blog button.	A page which demands a title and text appears.
Expert types title and text and clicks the submit button.	The blog is submitted and becomes available to all users.

5.2 NON-FUNCTIONAL TEST CASES

Test ID: TN01

Type: Non-Functional

Title: Performance Testing

Test Result: N/A

Covered Requirement(s):

1. *Account:* Simulate the Sign up, Log in, and Log out actions with only one as well as multiple users in the app.
2. *Scan Ingredients:* Simulate scanning on different ingredients. Simulate scanning ingredients with only one, as well as multiple users using the application, recording the times at which their results are shown on the screen.
3. *Blog Posts:* Simulate adding a blog to the system on one, as well as multiple accounts, and compare the amount of time it takes for them to show up in the blogs tab.

Entry Criteria: The application is running and the user is an expert type.

Test Steps	Expected Outcome	Result
Sign up, log in, log out with one user, and note its time. Sign up, log in, log out with multiple users at the same time, and compare the amount of time it took.	The time taken should be close enough that the case with multiple people should be less than two times the single-person entry time.	Sign up: ~15s Log in: ~8s Log out: ~3s
Scan ingredients of different products and record the time it	The time it takes should be close enough (that the time it takes for one product	Scanning: ~7s Analysis: ~10s

takes for various products.	should be less than two times of time it took for any other product scan).	
Scan an ingredient on a single device and later scan it on multiple devices and note the time it took on both trials.	The time it takes should be relatively similar, similar to the previously expected outcomes.	Trail #1: 10s Trail #2: 9s Trail #3: 10s Trail #4: 11s
Add a blog on one account, later add blogs on multiple accounts, and note the time both processes take for all blog posts to appear on the blogs page.	The time it takes should be relatively similar.	Adding: ~10s Loading: ~2s

---

Test ID: TN02

Type: Non-Functional

Title: User Interface Adaptability

Covered Requirement(s):

1. *Account:* Simulate sign up, log in, sign out, delete account actions and verify that all buttons, and fields are clickable and all text fields are readable.
2. *Scanning:* Simulate scanning an ingredient action, going to different pages, and pressing all buttons.
3. *Blogs*: Simulate sending a blog, editing, and reading blog functionality.

Entry Criteria: The application is running.

Test Steps	Expected Outcome	Result
The user Signs up, Logs in, Logs out, and deletes their account.	the UI is visible, interactable, and functional.	Success
A customer user who is logged in, scans a product, views its contents, and presses the links or buttons.	the UI is visible, interactable, and functional.	Success
An expert user who is logged in writes a blog, sends, and edits a blog. Then read multiple blogs to take a general look at the blogs page.	the UI is visible, interactable, and functional.	Success

---

Test ID: TN03

Type: Non-Functional

Title: Data Access Security

Covered Requirement(s):

1. *Sign up:* Simulate signing up with different and same-named accounts, and simulate different password situations (acceptable, not acceptable).
2. *Log in:* Simulate logging in to multiple accounts, entering the correct/wrong password and correct/wrong username.

Entry Criteria: The application is running.

Test Steps	Expected Outcome	Result
Sign up to account A, then try signing up with A's credentials, then try signing up with A's username but a different password, then sign up to B's account.	The UI should give appropriate warnings (and not sign up) on the 2nd and 3rd cases. The application should not allow these cases. The other cases should sign users up.	Success
Log in to account A, log out, then try logging in to account A with a false password, try logging in with a different username and right password, and log in with account B.	The UI should give appropriate warnings (and not log in) on the false password and false username cases. The application should not allow these cases. The other cases should allow log in.	Success

6.0 MAINTENANCE PLAN & DETAILS

Our maintenance strategy is streamlined to ensure that our application remains efficient, user-centric, and scalable. The plan involves the following key components:

We will continuously gather feedback on the accuracy of ingredient scans. Based on user input, we will update our database to reflect the most current and accurate information about food ingredients. This ensures our database stays relevant and useful. Regularly scheduled updates will help us refine our ingredient listings and add new data as food products evolve. This process requires at most 1 part time working person because our database is ingredient based, not product specific.

Utilizing AWS as our cloud provider ensures high availability ranging from 99.9% to 99.999%, accommodating user demands without service interruptions. Therefore, we do not foresee any problems about availability of our services [8].

We initially launched Contenta in English, we are planning to expand our application's language support to include Turkish. This will make our app accessible to a broader audience and enhance user

engagement by providing a localized experience. Future updates will incorporate language-specific features to cater to diverse user groups, enhancing the overall user experience.

A dedicated technical support team will be available to address any issues or queries from users. This team will provide prompt solutions and guidance, ensuring user satisfaction and smooth app functionality. We will actively monitor user feedback to continually improve the application. This includes adjusting features, fixing bugs, and enhancing usability based on users' experiences and suggestions.

Regular monitoring of system performance is essential to identify and resolve any potential bottlenecks or inefficiencies. A dedicated team will regularly test the functionalities of the application and detect some bottlenecks and some potential inefficiencies. Techniques such as caching, efficient database queries, and optimizations in code will be employed in case of detection of an inefficiency.

Ensuring the security of user data and system integrity is a priority. We will implement rigorous security measures, including regular security audits and updates to safeguard against vulnerabilities. We are using AWS which guarantees security. We also delete data of a user as soon as they want to delete their account in order not to violate data protection laws of Turkey. Data encryption and secure data storage practices will be enforced to protect sensitive user information and comply with data protection regulations.

By adhering to this maintenance plan, our project aims to deliver a robust, scalable, and user-friendly platform that adapts to user needs and technological advancements. This strategy not only supports ongoing technical maintenance but also aligns with our goals for future growth and expansion.

7.0 OTHER ANALYSIS ELEMENTS

In this section of the report, we will explain the other analysis elements of *Contenta*.

7.1. CONSIDERATION OF VARIOUS FACTORS IN ENGINEERING DESIGN

- **Public Health (10/10):** Contenta's main objective is about advancing public health and fostering well-being by informing the user about ingredients and additives of a packaged food. The platform endeavors to empower users with the means to make informed and health-conscious decisions regarding their diets and lifestyles, aiming to make a substantial contribution to overall public health. Carefully crafted features within the application, such as ingredient scanning and health insights, focuses on positively influencing users' health

outcomes. Therefore, it is given a 10 in public health factor.

- **Global and Cultural Factors (2/10):** *Contenta*'s influence on global and cultural aspects remains somewhat constrained. Although the application holds the potential to extend beyond geographical boundaries, its primary focus lies in health-conscious consumption, a feature that may not universally apply. Divergent cultural nuances regarding dietary preferences and health practices may lead to variations, contributing to a lower rating of 2 in global and cultural factors.

- **Social Factors (3/10):** *Contenta*, is not a social media platform. Its primary emphasis is on individual health rather than social interactions. The application's design places a premium on user privacy and tailored experiences, constraining its direct impact on social dynamics. Consequently, it is given a rating of 3 in the social factors category.

- **Environmental Factors (4/10):** Although *Contenta* does not explicitly tackle environmental concerns, it indirectly tackles environmental goals by promoting the adoption of plant-based diets among users. The advocacy for vegan or vegetarian lifestyles has the potential to contribute to a diminished environmental footprint and carbon footprint. Consequently, *Contenta* is assigned a moderate rating of 4 in the environmental factors category.

- **Economic Factors (6/10):** *Contenta* exhibits substantial potential to play a pivotal role in supporting the local economy, primarily through its advertising and subscription models. The platform's support for local businesses and the prospective creation of employment opportunities align with economic factors. It may encourage local businesses to produce healthy products. Contenta's capacity to generate revenue and contribute to local economies gets an above average rating of 6 in this category.

To sum up, *Contenta*'s engineering design is shaped by its fundamental goal of advancing public health. Although its reach on global and cultural aspects is confined, it causes moderate influence on social and environmental factors, while demonstrating significant impact on economic considerations. The concise table below offers a condensed overview of these considerations and their corresponding effects:

Table 1: Factors - Effect Rating Table

Factor	Effect Rating
Public Health	10/10

Global and Cultural Factors	2/10
Social Factors	3/10
Environmental Factors	4/10
Economic Factors	6/10

7.1.1 CONSTRAINTS

There are lots of constraints to bear in mind while developing and maintaining *Contenta*, but the following three are probably the most major ones.

- **Legal Considerations and Privacy**: *Contenta* must prioritize legal considerations and privacy standards to align with data protection regulations. Given the involvement of scanning and processing user-generated content, upholding intellectual property rights, adhering to content distribution policies, and safeguarding user data privacy are imperative. Rigorous adherence to legal frameworks establishes trust with users and safeguards the application's reputation.

- **Marketability and User Engagement:** A critical challenge lies in ensuring *Contenta*'s marketability, which hinges on addressing user adoption and retention. Encouraging users to download and retain the app requires strategic implementation of engaging features, personalized content recommendations, and regular updates to enhance user retention. Exploring the marketability of a premium version with extra features can create additional revenue streams and bolster user commitment.

- **Maintainability and Database Management:** *Contenta*'s maintainability is very important, especially considering potential database growth due to storing users' scanning history. An efficient database management system is essential to handle the expanding volume of data seamlessly. Implementing regular maintenance procedures and optimization strategies will sustain the application's smooth operation, mitigating performance bottlenecks and enhancing the overall user experience. Maintenance of the additive and ingredients table is easy because of *Contenta*'s approach not being product based.

7.1.2 STANDARDS

The development of the *Contenta* mobile application strictly adheres to globally recognized engineering standards, ensuring a robust software life cycle driven by uncompromising quality. This endeavor aligns with the ISO/IEC/IEEE International Standard for Systems and Software Engineering – Software Life Cycle Processes, placing emphasis on the need for systematic and well-defined processes throughout development, testing, and maintenance phases.

When it comes to crafting software requirements, we rely on the ISO/IEEE Recommended Practice for Software Requirements Specifications as a foundational guide. This standard not only guarantees the clarity, completeness, and consistency of software requirements documentation but also promotes effective communication among stakeholders, fostering a comprehensive understanding of project specifications. In the implementation of the project, Effective Dart standard is followed so that code maintenance will be easy.

In the domain of modeling and design, our project makes use of the widely accepted Unified Modeling Language (UML) 2.5.1. This standard is instrumental in visualizing, specifying, constructing, and documenting software artifacts, providing the project team with a universal language for clear communication and a shared understanding of the system architecture.

Additionally, the structure of the analysis and requirements report, including this one, aligns with the rigorous IEEE report writing guidelines. This adherence ensures a logical and coherent presentation of information, facilitating a comprehensive understanding of the project's progress and requirements. While maintaining commitment to IEEE guidelines, a small deviation from the traditional two-column structure has been implemented to enhance readability and accessibility for a broader audience.

7.2 ETHICS & PROFESSIONAL RESPONSIBILITIES

Protecting user privacy and not violating data protection laws are fundamental principles that guide every aspect of *Contenta*'s operation and design. We are committed to implementing powerful data protection measures while users utilize the scanning feature, ensuring that their sensitive information is managed with the highest level of care. In order to ensure data protection and do not violate data protection law, we used AWS which is secure and does not violate Turkey’s data protection law because it stores data in Turkey [9].

Additionally, it’s crucial to recognize that the *Contenta* app may not deliver perfectly accurate results at all times. In keeping with ethical standards and to maintain user trust, Contenta informs users that it

may not give completely accurate results. This policy of transparency not only adheres to our dedication to responsible development practices but also ensures that users are fully aware of the application's limitations. This fosters a relationship based on trust and integrity, which is essential in our project.

7.3 TEAMWORK DETAILS

In the successful completion of this project, dynamic teamwork and collaboration were essential. This section delves into the intricate details of how our team effectively pooled resources, skills, and expertise to achieve our common objectives. Below, there is a rough sketch of how much time we spent on each task.

Table 2: Task - Man-Hour - Assignee Table

TASK	MAN-HOUR	ASSIGNEE
Initial Idea Gathering & Development	150	Alperen, Barış, Mert, Oğuz, Ömer
Market Analysis	50	Alperen, Barış, Oğuz
Requirements Analysis	90	Alperen, Barış, Oğuz
Project Specifications Report	70	Alperen, Barış, Mert, Oğuz, Ömer
Analysis and Requirements Report	160	Alperen, Barış, Mert, Oğuz, Ömer
Dataset Population	200	Alperen, Barış, Oğuz
Ingredient Analysis Logic	90	Ömer
Detailed Design Report	110	Alperen, Barış, Mert, Oğuz, Ömer
Frontend Screens	220	Alperen, Mert, Ömer
AWS Pipeline Build	160	Ömer
Dataset Customization & Maintenance	40	Oğuz
Nutri-Score Calculation Script	30	Barış
Frontend - Backend Integration	80	Mert, Ömer
Testing & Optimization	90	Barış, Oğuz, Ömer

In total, approximately 1530 man-hour work was put in from five team members while developing Contenta as a mobile application from the beginning until now.

7.3.1 CONTRIBUTING AND FUNCTIONING EFFECTIVELY ON THE TEAM

In our project, we emphasized an equitable distribution of workload while keeping our academic obligations in mind. Our collective grasp of the project's demands facilitated mutual support within the team. Regularly scheduled meetings provided a forum for task allocation and planning our next steps. Besides, we used jira to ensure good organization. Each member's work is reviewed by others to ensure quality and consistency. Responsibilities were assigned based on individual strengths, interests, and expertise to guarantee balanced participation. API research and integration were a joint effort involving Mert, Ömer, Barış, Oğuz, and Alperen. Ömer specifically managed the AWS architecture's design and implementation. Mert and Alperen concentrated on front-end development to enhance user interface. Barış and Ömer collectively tackled server-side development. Alperen, Barış, and Oğuz were responsible for populating the database, with Oğuz further automating this process and making necessary customizations. Report compilation was evenly distributed among all team members, ensuring comprehensive documentation of our project's progress and insights. This approach not only ensured a balanced workload but also fostered a supportive and flexible project environment, allowing each member to contribute effectively while managing individual and collective schedules.

7.3.2 HELPING CREATING A COLLABORATIVE & INCLUSIVE ENVIRONMENT

In our team, we all knew each other before starting the project, which made collaboration smoother. We prefer meeting in person because it enhances our communication and teamwork. Even when we don't have specific tasks, we stay in touch to ensure everyone is involved and informed. We're committed to supporting each other, not just in individual tasks but in whatever the team needs. We regularly update each other on our progress, sharing both successes and challenges, so that everyone is in the loop and can pitch in when necessary. This approach keeps us aligned and prepared to tackle any issues together. Although we're friends, we make sure to listen to everyone's opinions in the project. This creates a friendly and supportive atmosphere where everyone feels valued and eager to contribute.

7.3.3 TAKING LEAD ROLE AND SHARING LEADERSHIP ON THE TEAM

In our team, we use a flexible style of leadership that lets us tap into each person's strengths. We don't have just one leader. Instead, we make big decisions together in our regular meetings, making sure everyone's opinions are considered.

We assign tasks based on what each team member is good at and interested in. For example, Ömer, who knows a lot about cloud computing and AWS services, leads our efforts in that area. His

knowledge helps us use the best methods for our cloud setup. In front-end development, Alperen takes the lead because of his ability to create user-friendly designs, which makes our project more appealing and easier to use.

Barış is in charge of our documentation because of his skills in writing well organized documents. He organizes and keeps our project files in order, ensuring they are clear and thorough. For managing our database, Oğuz steps up because he's skilled with regular expressions, making sure the dataset meets app's and server's requirements.

Our leadership approach changes as our project develops. This flexibility not only helps us improve in our specialties but also creates a learning environment where everyone can try new roles and grow their leadership skills. By rotating leadership based on needs and individual abilities, we've built an effective team dynamic.

7.3.4 MEETING OBJECTIVES

To complete our project, we organized several meetings throughout the semester, both with instructors and among ourselves.

7 February 2024: We held the first meeting of the semester on our own to decide what to do during the semester and which methods to adopt. We changed our initial decision about database population, which was to semi-manually populate a database using Python scripts and regular expressions with data from the Food Drug Administration. We decided to look for an API for ingredients and additives. All 5 members attended.

9 February 2024: There was a meeting with Dr. Bıçakçı. Dr. Bıçakçı explained the expectations from the Detailed Design Report, the number of expected test cases, and the qualities of the test cases, etc. All 5 members attended.

3 March 2024: We met to ensure we were all on the same page about the test cases and the detailed design report. Ömer proposed using AWS for the server side because of its enhanced availability. We also made work allocations for the detailed design report. All 5 members attended.

7 March 2024: There was a meeting with Dr. Bıçakçı. We informed Dr. Bıçakçı about the progress of our API search. We were not able to find any relevant API. Dr. Bıçakçı informed us that we could create mock data as long as it is realistic for the CS Fair and the demo. All 5 members attended.

19 March 2024: We requested a meeting with the innovation expert, Mr. Topal. We informed him that we wanted to use an API and could not find any relevant APIs for ingredients and additives. He indicated that even if we found one, we would need to ensure its availability at all times. A service failure during the demo or CS Fair could result in a catastrophe. All 5 members attended.

19 March 2024: After the meeting with Mr. Topal, we organized a meeting among ourselves. We decided to adopt the idea of populating the dataset semi-manually instead of continuing the search for an API. All 5 members attended.

25 March 2024: We decided the format of the database tables and wrote some sample scripts to extract FDA's data. Alperen, Oğuz, and Barış attended since they were responsible for dataset population.

2 April 2024: There was a meeting with Dr. Bıçakçı. We informed Dr. Bıçakçı about our change of decision regarding our dataset and our plans to use AWS. We discussed some current problems like translation. Dr. Bıçakçı proposed three ideas: we could use Amazon's chatbot, we could add additional fields for Turkish names, or we could run the application only in English. Dr. Bıçakçı said the test cases were relevant and sufficient and he explained the expectations for the demo. All 5 members attended.

4 April 2024: We organized a meeting before the Ramadan Eid to review our progress. Mert showed the progress on the frontend, Ömer demonstrated progress on AWS and the backend, and Alperen, Oğuz, and Barış showed progress on the dataset.

2 May 2024: There was a meeting with Dr. Bıçakçı. We organized a mock presentation and demo. Dr. Bıçakçı shared his suggestions about our mock presentation and demo. He highlighted that we should allocate less time to the presentation and more time to the demo.

7.4 NEW KNOWLEDGE ACQUIRED AND APPLIED

In this project, our team acquired a lot of new knowledge, particularly in the areas of cloud computing and mobile app development. Before this project, none of us had hands-on experience with AWS or cloud systems. However, through Bilkent University's cloud computing course, Ömer gained expertise in AWS, which allowed him to effectively build the server-side architecture for our project. Similarly, the client side presented a learning curve with Flutter, a framework none of us had used before.

Alperen, Mert, and Ömer took the initiative to learn Flutter and successfully used it to develop the client side of our application. Additionally, Oğuz and Barış expanded their technical skills by learning how to extract information from JSON files and delving into Python's web-scraping libraries. These new skills were critical in enhancing their ability to manage data interactions and content extraction for the project. This project not only advanced our technical skills but also enriched our practical knowledge, preparing us for future challenges in the tech industry.

8.0 CONCLUSION & FUTURE WORK

The *Contenta* app represents a significant advancement in consumer health technology, addressing the need for transparency and education in food consumption. By enabling users to instantly scan food labels and receive detailed, easy-to-understand information about additives and diet compatibility, *Contenta* empowers individuals to make healthier, more informed dietary choices. The integration of expert-written blogs further enhances the app's value, providing users with reliable knowledge on nutrition and well-being. Throughout the development process, the team has successfully overcome technical and design challenges, resulting in a robust and user-friendly application. The positive feedback received during the testing phase confirms the app's effectiveness and user satisfaction, indicating a strong potential for widespread adoption.

Looking ahead, the project has several exciting avenues for expansion and improvement. Firstly, we aim to make some frontend Improvements. Continued refinement of the user interface will enhance usability and aesthetic appeal, making the app even more engaging for users. We aim to include cosmetic products as well. Utilizing the similar data and architectural requirements, the app will expand its scanning capabilities to include cosmetic products. This will broaden our user base and meet the growing demand for transparency in cosmetic ingredients. We will automate the expert authentication process. Streamlining the process for verifying the credentials of content contributors will ensure the integrity and trustworthiness of the information provided, while also allowing for more scalable content production. We plan to integrate a payment API for premium features like Nutri-Score and therefore commercialize the project.: Integration of a payment system will facilitate the introduction of premium features, enabling a subscription-based revenue model. These enhancements and expansions will not only improve the functionality and appeal of *Contenta* but also ensure its long-term viability and success in the health tech market.

9.0 INSTALLATION GUIDE

Below, you may find the installation guide for *Contenta*. Along with the prerequisites and commands you should run.

Prerequisites:

- Flutter Framework: Flutter framework version 3.19 must be installed on your system.
- Dart Language: Dart language version 3.3 should be installed on your system.
- Android Studio: If running the project on an Android device or emulator, Android Studio must be installed on your system.

Installation Guide:

1. Clone the Repository:

Clone the Contenta GitHub repository into your local machine using the following command:

```
git clone <https://github.com/MrKty/Contenta_v2>
```

2. Install Dependencies:

Navigate to the project folder and run the following command to install project dependencies:

```
flutter pub get
```

3. Build the Project:

Run the following command to build the project:

```
flutter build
```

4. Connect Device or Emulator:

To run the project, ensure that a device is connected through USB with debug mode enabled or an Android emulator is created using Android Studio.

5. Select Device and Run:

Run the following command to list available devices:

```
flutter devices
```

Choose the desired device or emulator from the list and run the project by executing the following command:

```
flutter run
```

Additional Notes:

- Ensure that your system meets the hardware and software requirements for running Flutter and Android Studio.
- Make sure to have a stable internet connection during the installation process to download dependencies.
- For troubleshooting and additional assistance, refer to the Flutter documentation and community forums.

10.0 GLOSSARY

- EU: European Union
- KVKK: Kişisel Verilerin Korunması Kanunu (Personal Data Protection Law of Türkiye)
- GDPR: General Data Protection Regulation
- Nutri-Score: Nutrition Score (a.k.a 5-Colour Nutrition Label)
- UI: User Interface
- FDA: Food Drug Administration
- AWS: Amazon Web Services
- E Numbers: Europe Numbers (A number used in the European Union to identify permitted food additives).

REFERENCES

[1] M. Janani, P. Selvasekaran, M. Lokanadham, and R. Chidambaram, "Food and food products associated with food allergy and food intolerance – An overview," *Food Research International*, vol. 138, Part B, pp. 109780, 2020, ISSN 0963-9969, [Online]. Available: https://doi.org/10.1016/j.foodres.2020.109780.

[2] Calderone, Julia. “24 foods that artificial sweeteners are hiding in”. *Business Insider*. Accessed: Mar. 13, 2024. https://www.businessinsider.com/surprising-foods-that-artificial-sweeteners-are-hiding-in-2016-1.

[3] “Acesulfame Potassium”. Center for Science in the Public Interest. Accessed: Nov. 16, 2023. https://www.cspinet.org/article/acesulfame-potassium.

[4] “Nutri-Score”. Santé Publique France. Accessed: May 12, 2024. https://www.santepubliquefrance.fr/determinants-de-sante/nutrition-et-activite-physique/articles/nutri-score.

[5] Türkiye, TBMM. (2016, Mar. 24). Kişisel Verilerin Korunması Kanunu. Accessed: Nov. 16, 2023. https://www.mevzuat.gov.tr/mevzuat?MevzuatNo=6698&MevzuatTur=1&MevzuatTertip=5.

[6] Lapidos, Rachel. “What Derms Want You to Know About ‘Controversial’ Skin-Care Ingredients”. *Well+Good*. Accessed: Dec. 08, 2023. https://www.wellandgood.com/cosmetic-ingredient-review/

[7] Rafael D. Hernandez. “The Model View Controller Pattern – MVC Architecture and Frameworks Explained”. freeCodeCamp, Accessed: 12 May 2024. https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/

[8] "High Availability and Scalability on AWS," Amazon Web Services. Accessed: May 11, 2024. https://docs.aws.amazon.com/whitepapers/latest/real-time-communication-on-aws/high-availability-and-scalability-on-aws.html.

[9] Tuğçe İçözü, "Amazon, KVKK Onayı Aldı," *Webrazzi*, Mar. 05, 2021. Accessed: May 11, 2024. https://webrazzi.com/2021/03/05/amazon-kvkk-onay/.