

# COMP3040 Mobile Device Programming

## Coursework 2 Report

Name: Lee Boon Kah

Student ID: 20297564

Lecturer: Dr Nabil El Ioini

## Table of Contents

1. Project Title .....	3
2. Introduction .....	3
3. Project Planning and Management .....	4
4. Requirement Analysis .....	6
4.1. Functional Requirement .....	6
4.2. Non-Functional Requirement .....	11
4.3. User Story .....	15
4.4. User Acceptance Test .....	16
5. Design and Architecture .....	30
5.1. Overall Architecture .....	30
5.2. Design Principles .....	32
5.3. UML Class Diagram .....	35
6. Implementation .....	36
7. Testing .....	40
8. User Documentation .....	42
9. Deployment .....	54
10. Assumption of the application .....	58

## Project Title: Diet Navigator

### Introduction

The Diet Navigator app addresses the common challenge of maintaining a healthy diet aligned with individual preferences and health considerations. Many people struggle with making informed food choices, particularly those with dietary restrictions or health concerns. The goal of the Diet Navigator app is to provide a personalized platform that educates users on disease prevention and assists them in creating healthy and enjoyable meals tailored to their dietary preferences. The overarching goal is to promote a healthy lifestyle by offering users and administrators a seamless and intuitive experience within the application.

## Project Planning and Management

### Timeline and Milestones:

1. Project Initiation:
  - Defined the problem statement.
  - Researched existing solutions in the market.
  - Identified key features for the Diet Navigator app.
  - Duration: 13<sup>th</sup> November to 18<sup>th</sup> November (6 days)
2. Design and Architecture:
  - Planned the app structure.
  - Designed the User Interface (UI) for the four main pages (Home, Health News, Favourite, Profile).
  - Finalized the overall architecture of the app.
  - Duration: 20<sup>th</sup> November to 25<sup>th</sup> November (6 days)
3. Implementation:
  - Coded the development of the four main pages.
  - Integrated features such as custom diet recommendations, health news fetching, and user profile management.
  - Duration: 26<sup>th</sup> November to 7<sup>th</sup> December (12 days)
4. Testing:
  - Conducted unit testing for individual components.
  - Performed integration testing to ensure seamless interaction between modules.
  - Executed user acceptance testing to verify the app meets user requirements.
  - Duration: 7<sup>th</sup> December to 9<sup>th</sup> December (3 days)
5. Documentation:
  - Prepared comprehensive user documentation.
  - Created deployment instructions for users.
  - Duration: 9<sup>th</sup> December to 12<sup>th</sup> December (4 days)

The Diet Navigator project adhered to the planned timeline and milestones, ensuring a well-organized development process. The initiation phase started at 13<sup>th</sup> November and focused on understanding the problem and defining key features, leading to a clear direction for the project. Subsequently, the design and architecture phase unfolded from 20<sup>th</sup> November to 25<sup>th</sup> November, encompassing the planning of app structure, UI design for main pages, and finalization of the overall architecture.

Implementation began promptly after, with the development of the four main pages and the integration of features like custom diet recommendations, health news fetching, and user profile management. This phase kicked off on 26<sup>th</sup> November and concluded on 7<sup>th</sup> December. Testing commenced on 7<sup>th</sup> December, involving unit testing for individual components, integration testing for seamless module interaction, and user acceptance testing to verify adherence to user requirements. The final phase involved documentation, with comprehensive user documentation and deployment instructions created by 12<sup>th</sup> December.

This timeline and milestone approach enabled the successful development of the Diet Navigator app within the timeframe, meeting the goals of providing a personalized platform for users to make informed and sustainable food decisions.

## Requirements Analysis

### Stakeholders:

1. Users
2. Admin

### Functional Requirements:

1. User Authentication:
  - Users must provide a valid email address and password.
  - The system should authenticate users using Firebase Authentication.
2. Registration Link:
  - Users can navigate to the registration page by clicking the "Create New Account" link.
  - Users can create a new account.
3. Login Button:
  - Users initiate the login process by clicking the "Login" button.
  - The system verifies the provided credentials and grants access if they are valid.
4. Admin Differentiation:
  - The system checks if the logged-in user is an admin based on their email.
  - If the user is an admin, they are redirected to the admin page.
  - If the user is not an admin, they are redirected to the main user page.
5. Google Sign-In:
  - Users can choose to log in using their Google accounts by clicking the Google's icon.

6. Forgot Password:

- Users can reset their passwords by clicking the "Forgot Password" link.
- A dialog allows users to enter their registered email for password reset.
- The system sends a password reset email upon user request.

7. Collect Allergies and Diseases:

- New Users should be able to select their allergies from a predefined list.
- New Users should be able to select any existing diseases or choose "None" if not applicable.

8. Save Details

- Users should have the option to save their selected allergies and diseases.
- The system must validate that both allergies and diseases are selected before saving.

9. Custom Diet Filtering:

- Users should be able to filter recipes based on their custom dietary restrictions.
- The system should consider user-specific allergies and diseases when displaying recipes.

10. Recipe Filtering:

- Users should be able to filter recipes based on different categories, including "Breakfast," "Lunch," "Dinner," "Dessert," and "All."
- The system should display recipes that align with the user's selected filters.

#### 11. Search Functionality:

- Users should be able to search for recipes using a search bar.
- The system should dynamically filter, and display recipes based on the entered search text.
- Users can search for specific health news articles using the provided search functionality.
- The system should update the displayed news articles based on the user's search query.

#### 12. News Retrieval:

- The system must retrieve health-related news articles using an external News API.
- The user should be able to view a list of health news articles upon entering the activity.
- The system should fetch news articles related to the "health" category by default.

#### 13. Navigation Drawer:

- The navigation drawer should provide links to the home page, user profile, about us, and logout functionality.

#### 14. Bottom Navigation:

- The bottom navigation bar should allow users to switch between the home page, health news, favourite recipe, and user profile sections.
- The icon of the section at the bottom navigation should be highlighted when the user is in the respective section.

#### 15. Displaying Favourite Recipes:

- The system should fetch and display a list of favourite recipes for the logged-in user.
- The displayed recipes should be retrieved from the Firebase Realtime Database.



#### 16. Displaying Profile Information:

- The system retrieves and displays user profile information, including the user's email address, profile picture, allergies, and diseases.
- The profile information is fetched from the Firebase Firestore database.
- The user's email address is prominently displayed on the profile screen.

#### 17. BMI calculation:

- Users can calculate their Body Mass Index (BMI) by entering their weight (in kilograms) and height (in centimetres).
- The system calculates BMI using formula:  $BMI = \text{weight} / \text{height}^2$
- The system interprets the calculated BMI to categorize the user's weight status.
- BMI values falling within specific ranges are classified as "Obese," "Overweight," "Normal weight," or "Underweight."

#### 18. Profile Editing:

- Users can edit their profile information, including allergies and diseases.
- The system validates that both allergies and diseases are selected before update the profile information.
- The system allows users to select an image from their device using an image picker library.
- The selected image is uploaded to Firebase Cloud Storage.
- The system associates the uploaded image URL with the user's profile.

#### 19. Food Recipe upload:

- Admin can choose an image for a recipe by selecting the "Select Image" option.
- Admin input recipe details, such as name, description, ingredients, steps, and category.
- The system stores the recipe in the Firebase Realtime Database.

#### 20. Recipe Deletion:

- Admin can initiate the deletion of a recipe.
- Upon recipe deletion, the associated image stored in Firebase Storage is removed.

#### 21. Logout:

- Users and Admin should have the option to log out from the application.
- Logging out should redirect the user to the Login Page.

## Non-Functional Requirements:

### 1. Security:

- Secure handling of user credentials during the authentication process.
- Secure password reset process.
- Secure handling of user credentials and profile information during the authentication and profile retrieval processes.

### 2. Usability:

- Intuitive user interface for entering login credentials.
- Clear navigation to the registration and password reset functionalities.
- The interface should be user-friendly for browsing all the pages.
- Clear feedback messages should be provided when the user perform an action.
- Distinction between regular users and administrators.
- The filtering options (by category, custom diet, and search) should be intuitive and easily accessible to users.
- The user interface should be intuitive for viewing and editing the user's profile.
- The BMI calculator interface should be user-friendly, guiding users to input weight and height easily.
- The BMI result, along with its interpretation, should be presented in a comprehensible manner.

### 3. Performance:

- Efficient authentication process with minimal delay.
- The process of saving details and retrieving previous information should be efficient.

- The recipe filtering and search functionalities should be responsive, providing quick results to users.
- The system should efficiently handle the retrieval and display of filtered recipes based on user preferences.
- The system should efficiently fetch and display health news articles from the external API.
- The system should efficiently handle the retrieval and display of favourite recipes.
- Quick response times are expected during user interactions such as scrolling and switching between sections.
- The BMI calculation process should be swift and responsive.
- Real-time feedback on the user's weight status should be displayed promptly.

#### 4. Reliability:

- The system should handle authentication failures gracefully, providing informative error messages to users.
- The recipe filtering and search functionalities should be responsive, providing quick results to users.
- The system should efficiently handle the retrieval and display of filtered recipes based on user preferences.

#### 5. Maintainability:

- Code maintainability should be emphasized, ensuring that future updates or modifications can be implemented easily.
- The codebase should follow best practices and coding standards for Android development.

#### Constraints:

1. The login functionality relies on Firebase Authentication services.
2. Users must have a stable internet connection for the authentication process.
3. The application relies on Firebase Firestore for storing and retrieving user profile information.
4. The application relies on Firebase Realtime Database for storing and retrieving food recipe information.
5. The application relies on Firebase Storage for storing and retrieving the food picture and the user profile picture.
6. The image upload functionality depends on the user's device capabilities and permissions.
7. Admins must have the necessary permissions to delete recipes and associated images.

#### Assumptions:

1. Users are expected to provide valid email addresses and passwords for authentication.
2. The Firebase Authentication service is assumed to be operational and available during the login process.
3. The user's profile picture is stored in Firebase Storage, and the retrieval process assumes a stable internet connection.
4. User profile information (allergies, diseases) is assumed to be pre-stored in the Firestore database during the user registration process.
5. The dynamic visibility of the bottom navigation and bottom app bar enhances the user experience during recipe browsing.
6. The image upload process may take some time based on the image size and network speed.

#### Additional Insights:

1. Admins are identified based on a specific email address ("admin@gmail.com").
2. The code includes specific actions for admin and regular users during the login process.
3. There is a distinction between the user and admin roles in the system.

## User Story

### User Story for Admin:

As an admin of the Diet Navigator application, I aim to enrich the recipe database by contributing new and exciting recipes. To achieve this, I log into the app using my admin credentials. Once authenticated, I navigate to the recipe upload section, where I can input detailed information about a recipe, including its name, description, ingredients, steps, and category. I have the flexibility to choose an appealing image for the recipe. Upon clicking the "Upload Recipe" button, the system should efficiently store this new recipe in the Firebase Realtime Database, ensuring it becomes accessible to all users. Additionally, the system should confirm the successful upload, providing a seamless experience. After a productive session of uploading recipes, I can conveniently log out, securing the app and its valuable data.

### User Story for Regular User:

As a regular user of the Diet Navigator application, my primary goal is to find and explore recipes that align with my dietary preferences and restrictions. I begin my journey by logging into the app with my user credentials. Once inside, I'm welcomed by a user-friendly interface with intuitive navigation options. The bottom navigation bar allows me to effortlessly switch between sections like the home page, health news, favourite recipes, and my user profile. To personalize my experience, I navigate to the profile section and edit my profile information. I select my allergies from a predefined list, choose any existing diseases, and upload a profile picture for a more personalized touch. The app validates my selections and efficiently updates my details in the Firebase Firestore Database. I can further enhance my journey by using the custom diet filtering option to discover recipes tailored to my dietary restrictions. The search functionality enables me to find specific recipes or health news articles. After exploring and finding some favourite recipes, I can view and manage them in the dedicated "favourites" section. The Diet Navigator app provides a seamless experience, allowing me to navigate, personalize, and engage with recipes and health-related content effortlessly.

## User Acceptance Test

### User Acceptance Test (UAT) Scenario: Registration and Profile Setup

Objective: To verify that users can create a new account and set up their profiles.

#### UAT Steps:

##### 1. Test Case: Registration Link

- Steps:
  1. Open the Diet Navigator app.
  2. On the login page, click on the "Create New Account" link.
  3. Enter the required registration details.
  4. Click on the "Register" button.
- Expected Result:
  - The system should create a new user account using Firebase Authentication.
  - The user should be redirected to the first-time login setup.

##### 2. Test Case: First-Time Login Setup

- Steps:
  1. Complete the registration process.
  2. On the first-time login page, select allergies and diseases from the provided list.
  3. Complete the profile setup process.
  4. Click on the "Submit" button.
- Expected Result:
  - The system should save user details using Firebase Firestore.
  - The user should be redirected to the main user page after completing the initial setup.



## User Acceptance Test (UAT) Scenario: User Authentication

**Objective:** To ensure that users can successfully authenticate into the Diet Navigator application.

### UAT Steps:

#### 1. Test Case: Email and Password Authentication

- Steps:
  1. Open the Diet Navigator application.
  2. On the login page, enters a valid email address and password into the respective input boxes.
  3. Click on the "Login" button.
- Expected Result:
  - The system should authenticate the user and redirect them to the food recipe page.

#### 2. Test Case: Google Sign-In Authentication

- Steps:
  1. Open the Diet Navigator application.
  2. On the login page, click on the Google icon for sign-in.
  3. Enter the valid email address and password.
  4. Click on the "Sign in" button.
- Expected Result:
  - The system should authenticate the user using Google credentials and redirect to the food recipe page.

## User Acceptance Test (UAT) Scenario: Navigation and User Interface

Objective: To ensure that users can navigate seamlessly between different sections of the application and interact with a user-friendly interface.

### UAT Steps:

#### 1. Test Case: Navigation Drawer Links

- Steps:
  1. Login to the Diet Navigator application.
  2. Open the navigation drawer by press the icon that display at the tool bar.
  3. Click on links such as home, user profile, about us, and logout.
- Expected Result:
  - The system should navigate users to the respective pages and functionalities.

#### 2. Test Case: Bottom Navigation

- Steps:
  1. Login to the Diet Navigator application.
  2. Explore the bottom navigation bar.
  3. Click on icons for home, health news, favourite recipes, and user profile.
- Expected Result:
  - The system should switch between sections, highlighting the selected icon in the bottom navigation.

## User Acceptance Test (UAT) Scenario: Recipe Browsing and Interaction

Objective: To ensure that users can effectively browse, filter, and interact with recipes.

### UAT Steps:

#### 1. Test Case: Custom Diet Filtering

- Steps:
  1. Login to the Diet Navigator application as a user.
  2. Select the “Custom Diet” button that allocates below the search bar.
- Expected Result:
  - The system should display recipes that align with the selected dietary restrictions.

#### 2. Test Case: Category-Based Filtering

- Steps:
  1. Login to the Diet Navigator application.
  2. Click on the specific recipe category button.
- Expected Result:
  - The system should display recipes belonging to the chosen category that under the custom diet list.

#### 3. Test Case: Recipe Search

- Steps:
  1. Login to the Diet Navigator application.
  2. Use the search bar to enter the recipe name.
- Expected Result:
  - The system should dynamically filter and display recipes containing the entered keywords.

## User Acceptance Test (UAT) Scenario: Displaying Favourite Recipes

### Objective:

To ensure that users can view their favourite recipes.

### UAT Steps:

#### 1. Test Case: Add Recipe to Favourites

- Steps:
  1. Login to the Diet Navigator app.
  2. Navigate to the recipe details page.
  3. Click on the heart that allocate at the bottom-left of the food recipe.
- Expected Result:
  - The system should add the selected recipe to the user's favourites in the Firebase Realtime Database.

#### 2. Test Case: View Favourite Recipes

- Steps:
  1. Navigate to the "Favourites" section in the app.
  2. Check for the presence of the recipes added to favourites.
- Expected Result:
  - The system should fetch and display a list of favourite recipes for the logged-in user.
  - The displayed recipes should be retrieved from the Firebase Realtime Database.

## User Acceptance Test (UAT) Scenario: Health News Interaction

Objective: To verify that users can access and interact with health-related news articles.

### UAT Steps:

#### 1. Test Case: Default News Display

- Steps:
  1. Login to the Diet Navigator application.
  2. Navigate to the health news section.
- Expected Result:
  - The system should fetch and display health-related news by default.

#### 2. Test Case: News Search

- Steps:
  1. Login to the Diet Navigator application.
  2. Navigate to the health news section.
  3. Use the search functionality to enter specific keywords.
- Expected Result:
  - The system should display news articles containing the entered keywords.

#### 3. Test Case: Navigation to Full Article

- Steps:
  1. Login to the Diet Navigator application.
  2. Navigate to the health news section.
  3. Click on a specific news article.
- Expected Result:
  - The system should navigate to the full article page, displaying the complete content of the selected news article.

## User Acceptance Test (UAT) Scenario: View User Profile

### Objective:

To ensure that users can view their profile information.

### UAT Steps:

#### 1. Test Case: Navigate to Profile

- Steps:
  1. Login to the Diet Navigator app.
  2. Look for an option such as "Profile" in the bottom navigation menu
  3. Click on the "Profile" option.
- Expected Result:
  - The system should navigate the user to a page displaying their profile information.

#### 2. Test Case: Display User Information

- Steps:
  1. On the "Profile" page, check for the display of user information such as email address, profile picture, allergies, and diseases.
- Expected Result:
  - The user's email address should be prominently displayed.
  - The profile picture, if uploaded, should be visible.
  - Allergies and diseases, if selected, should be listed.

### 3. Test Case: Check for Edit Profile Option

- Steps:
  1. On the "Profile" page, look for an option to edit profile information.
  2. Click on the "Edit Profile" or similar option.
- Expected Result:
  - The system should navigate to a page where users can modify their profile details.

### 4. Test Case: Check for Edit Profile Option

- Steps:
  1. On the "Profile" page, look for the icon of BMI calculator.
  2. Click on the BMI calculator icon.
- Expected Result:
  - The system should navigate to a page where users can calculate the BMI.

## User Acceptance Test (UAT) Scenario: BMI Calculation and Profile Feedback

Objective: To verify that users can accurately calculate their Body Mass Index (BMI) and receive prompt feedback.

### UAT Steps:

#### 1. Test Case: BMI Calculation and Interpretation

- Steps:
  1. Login to the Diet Navigator application.
  2. Navigate to the Profile Section.
  3. On the Profile Page, click on the BMI calculator icon.
  4. Enter age, weight (in kilograms) and height (in centimetres).
  5. Click the "Calculate " button.
- Expected Result:
  - The system should accurately calculate BMI using the provided formula and display the result.
  - The system should interpret the calculated BMI, categorizing it as "Obese," "Overweight," "Normal weight," or "Underweight," and provide this information to the user.



## User Acceptance Test (UAT) Scenario: Edit Profile Information

### Objective:

To verify that users can successfully edit their profile information.

### UAT Steps:

#### 1. Test Case: Navigate to Edit Profile

- Steps:
  1. Login to the Diet Navigator app.
  2. Navigate to the profile section.
  3. Look for an "Edit Profile" icon and click on it.
- Expected Result:
  - The system should navigate the user to the page where profile information can be edited.

#### 2. Test Case: Modify Allergies and Diseases

- Steps:
  1. On the "Edit Profile" page, locate the sections for allergies and diseases.
  2. Modify the selected allergies and diseases.
  3. Click on the "Save" button.
- Expected Result:
  - The system should validate the modified allergies and diseases and update the user details in the Firebase Firestore Database.

### 3. Test Case: Change Profile Picture

- Steps:
  1. On the "Edit Profile" page, click on the “upload picture” button.
  2. Choose an image from the device using the image picker library.
  3. Click on “Save” button.
- Expected Result:
  - The system should upload the selected image to Firebase Cloud Storage.
  - The user's profile should now reflect the updated image.

## User Acceptance Test (UAT) Scenario: Admin Authentication and Navigation

Objective: To ensure that admins can authenticate and access admin-specific functionalities.

### UAT Steps:

#### 1. Test Case: Admin Email Authentication

- Steps:
  1. Open the Diet Navigator application.
  2. Enter the admin email address and password.
  3. Click on the "Login" button.
- Expected Result:
  - The system should authenticate the admin and redirect them to the admin page.

#### 2. Test Case: Navigation to Admin Features

- Steps:
  1. Login as an admin.
  2. Navigate to admin-specific functionalities (e.g., recipe upload, recipe deletion).
- Expected Result:
  - The system should grant access to admin-specific features and restrict access for regular users.

## User Acceptance Test (UAT) Scenario: Recipe Upload

### Objective:

To verify that admins can successfully upload new recipes to the Diet Navigator application.

### UAT Steps:

#### 1. Test Case: Accessing Recipe Upload Section

- Steps:
  1. Login as an admin.
  2. Click on the floating button that allocate at the bottom right of the page.
  3. Inside the Recipe Upload section, input recipe details such as name, description, ingredients, steps, and category.
  4. Choose an image for the recipe using the "Select Image" option.
  5. Click on the "Upload Recipe" button.
- Expected Result:
  - The system should store the provided recipe details, including the image, in the Firebase Realtime Database.
  - A success message should confirm the successful upload.

#### 2. Test Case: Verification of Uploaded Recipe

- Steps:
  1. Navigate to the recipes section or a designated area where all recipes are listed.
  2. Look for the recently uploaded recipe.
- Expected Result:
  - The uploaded recipe should be visible in the list of recipes, displaying all the details provided during the upload process.

### 3. Test Case: Logout

- Steps:
  1. After uploading a recipe, locate the logout option in the navigation or settings.
  2. Click on the "Logout" option.
- Expected Result:
  - The system should log out the admin, redirecting them to the login page.

## Design and Architecture

### Overall Architecture:

The Diet Navigator app follows a layered architecture with separation of concerns for data, presentation, and logic layers, incorporating the Model-View-ViewModel (MVVM) pattern. The MVVM layer in the Diet Navigator app plays a crucial role in managing the separation of concerns between the data layer, presentation layer, and the logic layer. MVVM is a design pattern that enhances the organization and testability of the code by introducing a clear distinction between the UI components and the underlying data and business logic.

#### 1. Model-View-ViewModels:

- ViewModel components are responsible for handling UI-related logic, interacting with the data layer, and ensuring that data is presented in a format suitable for the UI components.
- Each major screen or functionality in the app has an associated ViewModel. For example, a RecipeViewModel manages the data related to displaying recipes, handling user interactions, and coordinating with the data layer to fetch and present information.
- It utilizes Firebase services for authentication, storage, and database management.
- **Rationale:** ViewModels centralize UI-related logic, making it easier to manage and test. Each major screen having its ViewModel ensures a modular and maintainable codebase. This is crucial for handling complex UI interactions, such as displaying recipes and managing user profiles.

#### 2. Data Layer:

- **Firebase Authentication:** Handles user login, registration, and password reset.
- **Firebase Firestore:** Stores user profile information (including allergies, diseases)
- **Firebase Realtime Database:** Stores recipe information (including ingredients, instructions, category, and image URL).
- **Firebase Cloud Storage:** Stores food recipe images and user profile pictures.
- **Rationale:** Firebase Authentication is a robust solution for handling user login, registration, and password reset. Its integration ensures secure access to the

app's functionalities, providing a seamless and trustworthy user experience. Firestore and Realtime Database serve as efficient NoSQL databases for storing user profiles and recipe information. Firestore's real-time synchronization capabilities are well-suited for dynamic user data, while Realtime Database is effective for recipe-related information. Storing images in Firebase Cloud Storage aligns with Firebase's ecosystem, providing a reliable and scalable solution for handling media assets associated with recipes and user profiles.

### 3. Presentation Layer:

- **Activities:** Main screens of the app, responsible for displaying UI components and responding to user interactions.
- **Adapters:** Responsible for populating data in RecyclerViews, such as recipes and news articles, connecting the data layer with the UI components.
- **Layouts:** Define the visual structure of screens using XML resources, adhering to the MVVM separation of UI logic and business logic.
- **Rationale:** Activities serve as the main screens, and adapters bridge the gap between data and UI components, ensuring smooth population of data in RecyclerViews. This follows Android's native development patterns and adheres to MVVM principles.

### 4. Logic Layer:

- **ViewModels (MVVM):** the ViewModel components handle UI-related logic, communicate with the data layer, and ensure data is presented in a format suitable for the UI.
- **Services:** Handle complex logic and business rules, such as retrieving recipes, filtering based on user information, and calculating BMI.
- **Models:** Represent data structures, such as Recipe, User, and News Article.
- **Utils:** Helper classes for common tasks, such as image uploading.
- **Rationale:** Models represent data structures, providing a standardized way to handle data. Utils encapsulate common tasks, enhancing code reusability and maintainability.

## Design Principles:

### 1. DRY (Don't Repeat Yourself):

- DRY is aimed at reducing repetition of code, promoting reusability, and making maintenance easier.
- Example:
  - Create utility classes or methods for shared functionalities like image uploading to Firebase Cloud Storage.
- Rationale:
  - DRY reduces redundancy, promoting reusability and easing maintenance. Creating utility classes for shared functionalities, like image uploading, adheres to this principle.

### 2. Encapsulate What Changes:

- Encapsulation involves bundling the data (attributes) and the methods (functions) that operate on the data into a single unit (class).
- Example:
  - Encapsulate the details of Firebase Authentication and Firestore interactions within dedicated classes or services to isolate changes related to these services.
- Rationale:
  - Encapsulation isolates changes related to specific functionalities. Encapsulating Firebase Authentication and Firestore interactions ensures that modifications to these services won't impact other parts of the application.



### 3. Open Closed Design Principle:

- Software entities (classes, modules, functions) should be open for extension but closed for modification.
- Example:
  - Design adapters that can be extended for new data sources (e.g., additional recipe categories) without modifying existing adapter code.
- Rationale:
  - Adhering to the open-closed principle allows for extension without modification. Designing adapters that can be extended for new data sources ensures adaptability to future changes.

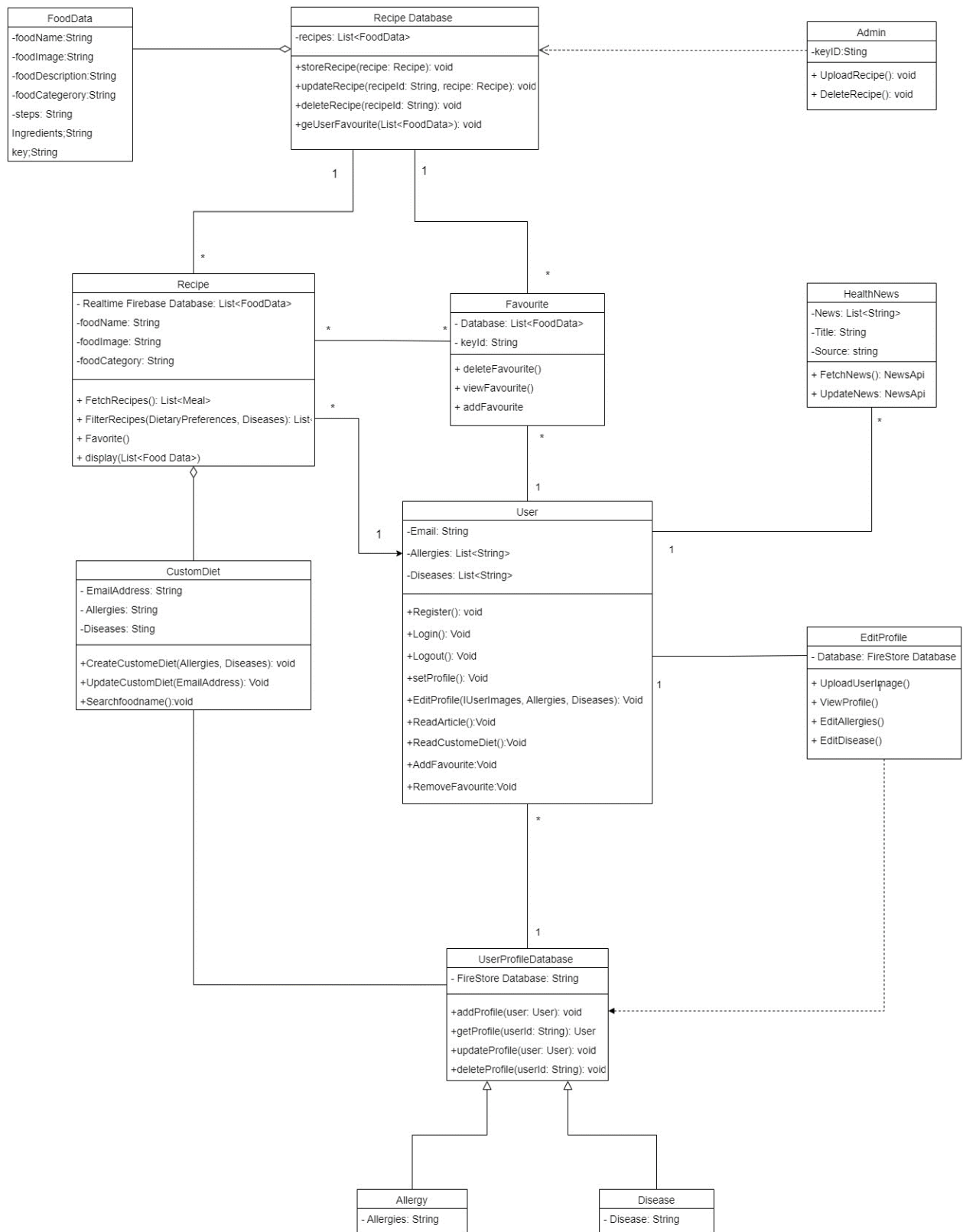
### 4. Single Responsibility Principle (SRP):

- A class should have only one reason to change, meaning it should have only one responsibility.
- Example:
  - Design individual classes or services responsible for specific tasks, such as handling user authentication, managing user profiles, and interacting with Firebase services.
- Rationale:
  - SRP enhances code maintainability. Having individual classes or services responsible for specific tasks, such as authentication or database interactions, ensures a clear and focused responsibility for each component.

## 5. Liskov Substitution Principle (LSP):

- Subtypes must be substitutable for their base types without altering the correctness of the program.
- Example:
  - Ensure that subclasses (e.g., specific recipe categories) can be used interchangeably with their base class (e.g., a generic recipe class) without causing issues in the app's functionality.
- Rationale:
  - LSP ensures that subclasses can be used interchangeably with their base types. Ensuring that specific recipe categories can be used seamlessly with a generic recipe class maintains the app's functionality and flexibility.

## UML Class Diagram



## Implementation

The implementation of the Diet Navigator app involves several major components and modules. Each of these components plays a crucial role in the functionality and user experience of the application.

### 1. Authentication Module:

- Purpose: Manages user authentication and authorization.
- Implementation: Utilizes Firebase Authentication for user registration, login, and password reset functionalities.
- Key Features:
  - Email/password authentication.
  - Google Sign-In for seamless authentication.
  - Firebase Authentication Triggers for handling authentication events.

### 2. User Profile Module:

- Purpose: Handles user profile information and preferences.
- Implementation: Uses Firebase Firestore to store and retrieve user-specific data.
- Key Features:
  - Allows users to set allergies and diseases during the initial setup.
  - Enables users to edit profile information, including allergies, diseases, and profile pictures.
  - Calculates and displays BMI based on user-entered data.

### 3. Recipe Management Module:

- Purpose: Manages the storage and retrieval of recipes.
- Implementation: Utilizes Firebase Realtime Database to store recipe details.
- Key Features:
  - Allows admins to upload new recipes, including name, description, ingredients, steps, and category.
  - Enables users to browse and filter recipes based on dietary preferences.
  - Allows users to mark recipes as favourites.

### 4. Health News Module:

- Purpose: Fetches and displays health-related news articles.
- Implementation: Retrieves data from external APIs or a dedicated backend service.
- Key Features:
  - Displays default health news articles upon entering the health news section.
  - Allows users to search for specific news articles using keywords.
  - Navigates users to full articles for an in-depth view.

### 5. BMI Calculator Module:

- Purpose: Calculates and interprets the Body Mass Index (BMI) for users.
- Implementation: Part of the user profile section, utilizing user-entered height, weight, and age.
- Key Features:
  - Provides real-time BMI calculation.

- Categorizes BMI results (e.g., "Obese," "Overweight," "Normal weight," or "Underweight").
- Offers immediate feedback and health recommendations based on BMI.

#### 6. Navigation Module:

- Purpose: Manages the navigation flow within the app.
- Implementation: Utilizes Android's Navigation Component.
- Key Features:
  - Implements bottom navigation for easy access to home, health news, favourite recipes, and user profile sections.
  - Utilizes navigation drawers for additional navigation options.
  - Ensures seamless transitions between screens.

#### 7. UI Components Module:

- Purpose: Defines the visual structure and components of the app.
- Implementation: Utilizes XML layouts and resources for UI design.
- Key Features:
  - Creates layouts for activities and fragments.
  - Designs RecyclerView adapters for displaying lists of recipes and news articles.
  - Ensures a user-friendly and visually appealing interface.

## 8. Utilities Module:

- Purpose: Provides helper functions for common tasks.
- Implementation: Implements utility classes and functions.
- Key Features:
  - Manages image uploading to Firebase Cloud Storage.
  - Formats data for display.
  - Performs other common and reusable tasks.

## Testing

### 1. Functional Testing:

- Objective: To ensure that the application functions according to specified requirements.
- Results: Functional tests identified issues with the registration process, where some users were unable to complete the first-time login setup.
- Issues and Fixes:
  - Updated the registration process to handle edge cases and varying user inputs.
  - Improved error messages for better user guidance.
  - Conducted additional functional testing to validate the registration flow.

### 2. Usability Testing:

- Objective: To assess the overall user experience and identify usability issues.
- Results: Usability tests highlighted issues with unclear navigation labels, leading to user confusion.
- Issues and Fixes:
  - Updated navigation labels and icons for better clarity.

### 3. User Acceptance Testing (UAT):

- Objective: Confirm that the app meets user requirements and provides a positive user experience.
- Results: UAT scenarios, including registration, authentication, recipe browsing, and profile editing, have been successfully executed.
- Issues and Fixes:



- Addressed UI inconsistencies based on user feedback to enhance the overall user experience.
- Refactored the recipe upload process for admins to ensure a smooth and error-free operation.

#### 4. Security Testing:

- Objective: Identify and address potential security vulnerabilities, especially in user authentication and data storage.
- Results: Security testing has been conducted to ensure secure handling of user credentials, protection against unauthorized access, and secure storage of sensitive data in Firebase services.
- Issues and Fixes:
  - Ensured that Firebase Authentication mechanisms are implemented securely to prevent unauthorized access.
  - Validated the secure transmission of data between the app and Firebase servers.

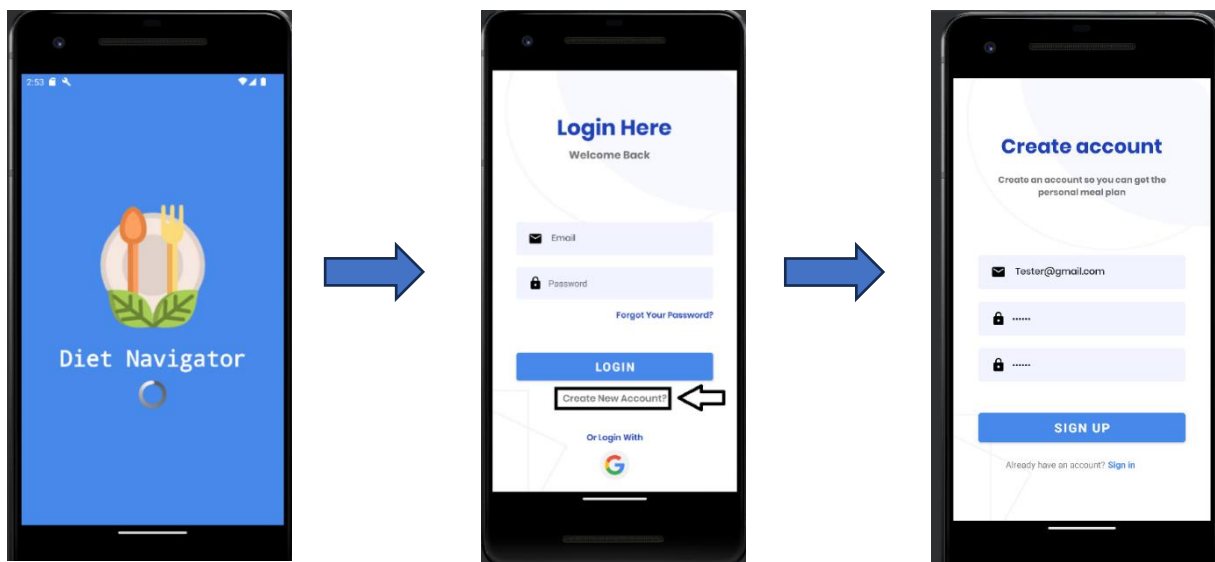
## User Documentation

### Diet Navigator App User Guide:

#### 1. Getting Started

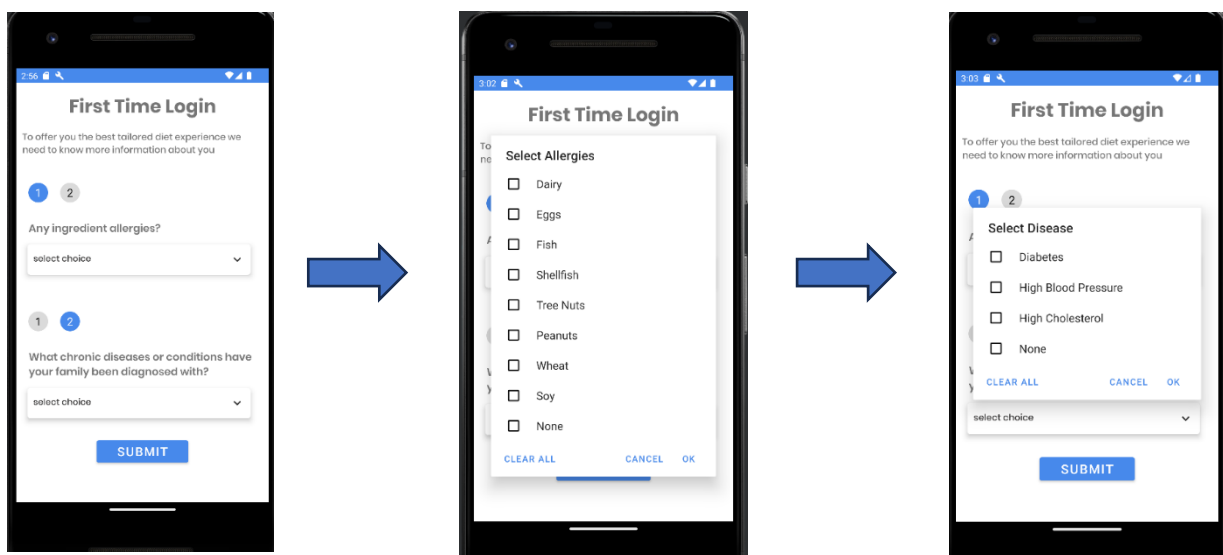
##### 1.1. Creating an Account

- 1.1.1. Open the app and click on the "Create New Account" on the login page.
- 1.1.2. On the Register page, enter a valid email address, password and confirm the password.
- 1.1.3. After completing the registration process, click on the "SIGN UP" button to create your account.



##### 1.2. First Time Login

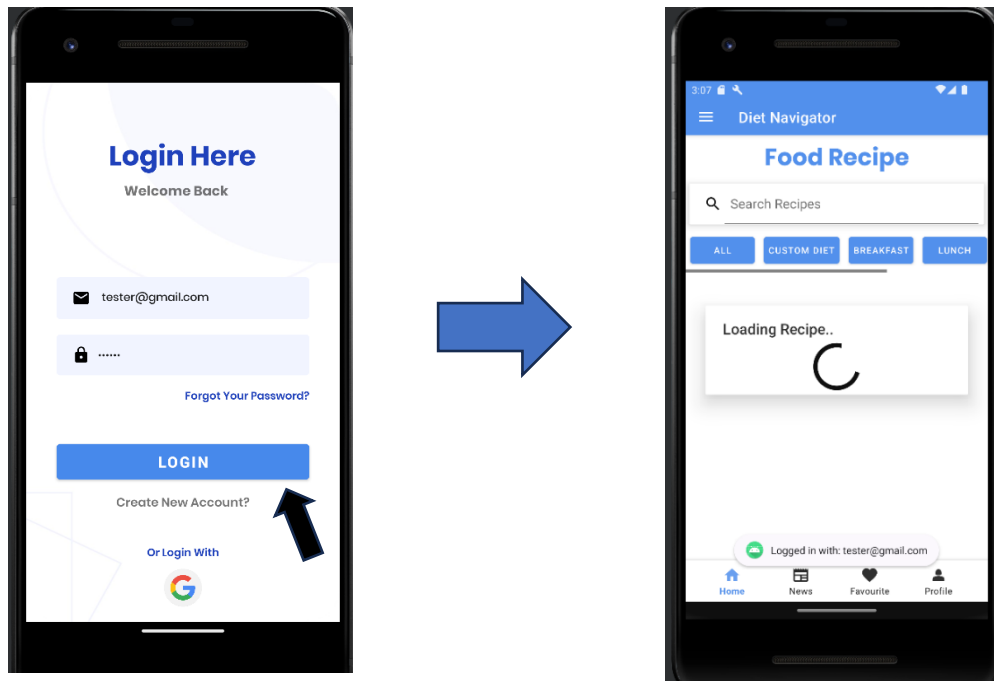
- 1.2.1. Select the allergies, diseases to complete your profile information.
- 1.2.2. Click on "Submit" button to navigate to the main page.



### 1.3. Logging

1.3.1. In the Login page, enter your registered email address and password on the login page.

1.3.2. Click on "Login" to access the app.

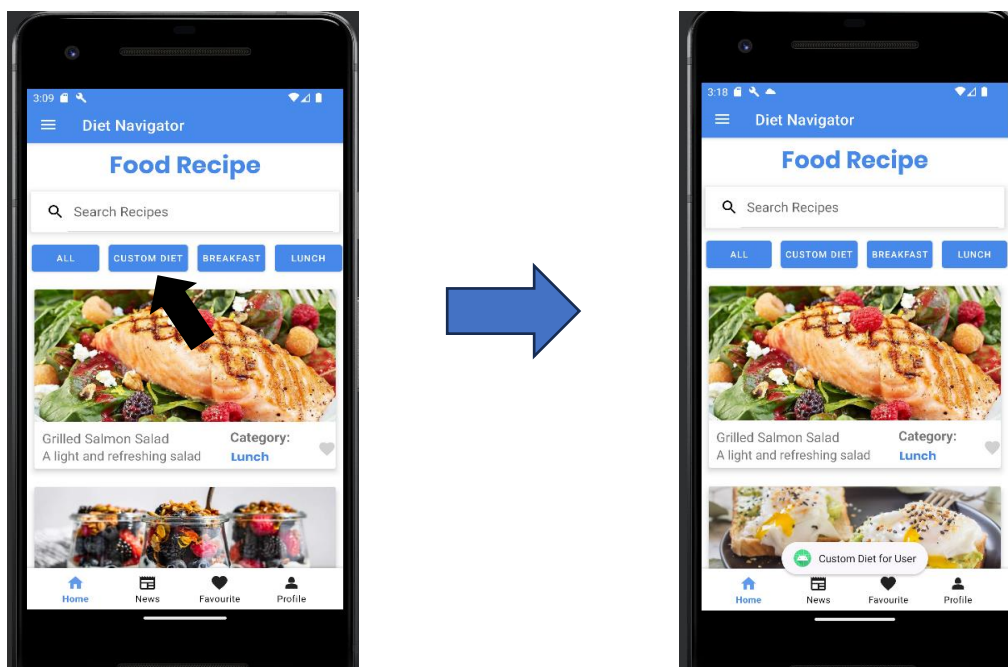


## 2. Discovering Food Recipes

### 2.1. Custom Diet Filtering

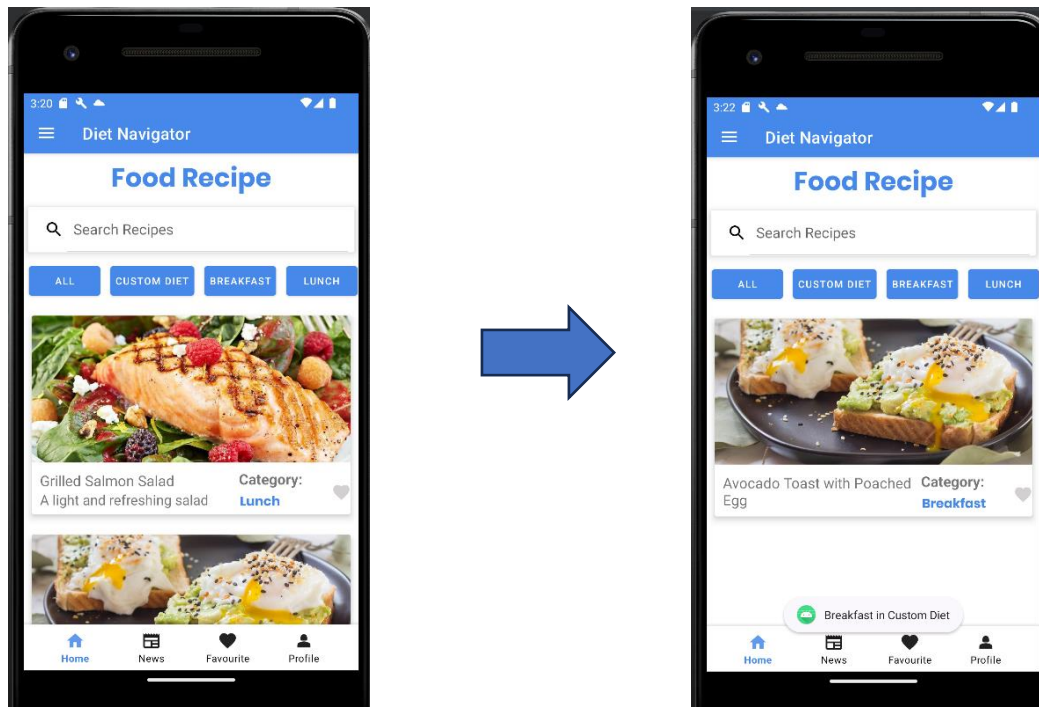
2.1.1. In the recipe section, click on the "Custom Diet" button.

2.1.2. The app will display recipes aligned with your selected criteria.



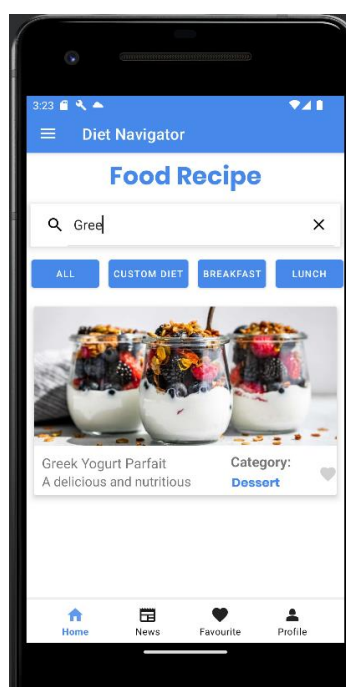
## 2.2. Category-Based Filtering

- 2.2.1. Navigate to the recipe category you are interested in.
- 2.2.2. Click on the specific recipe category button (e.g. Breakfast).
- 2.2.3. Recipes within the chosen category in the custom diet will be displayed.



## 2.3. Recipe Search

- 2.3.1. Use the search bar to enter specific keywords (recipe names).
- 2.3.2. The app will dynamically filter, and display recipes based on your search.

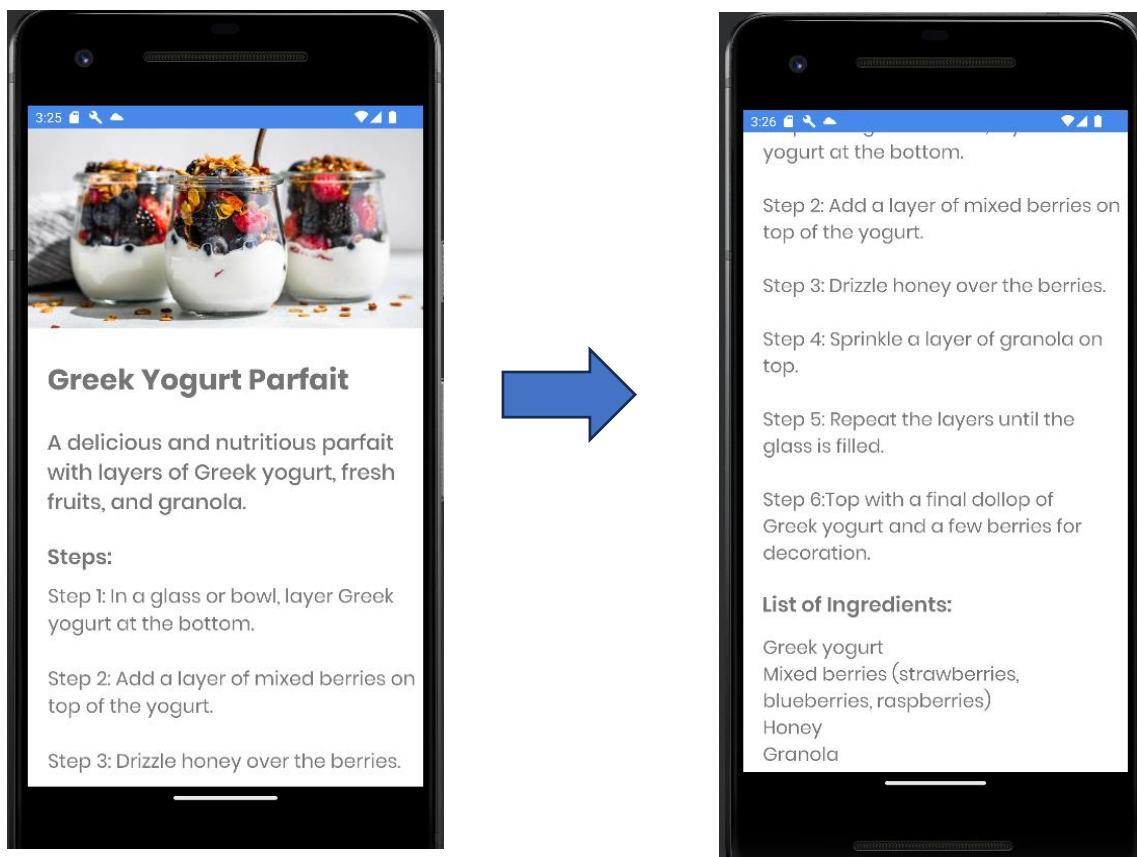


## 2.4. Food Recipe Details

2.4.1. Click on a specific recipe from the displayed list.

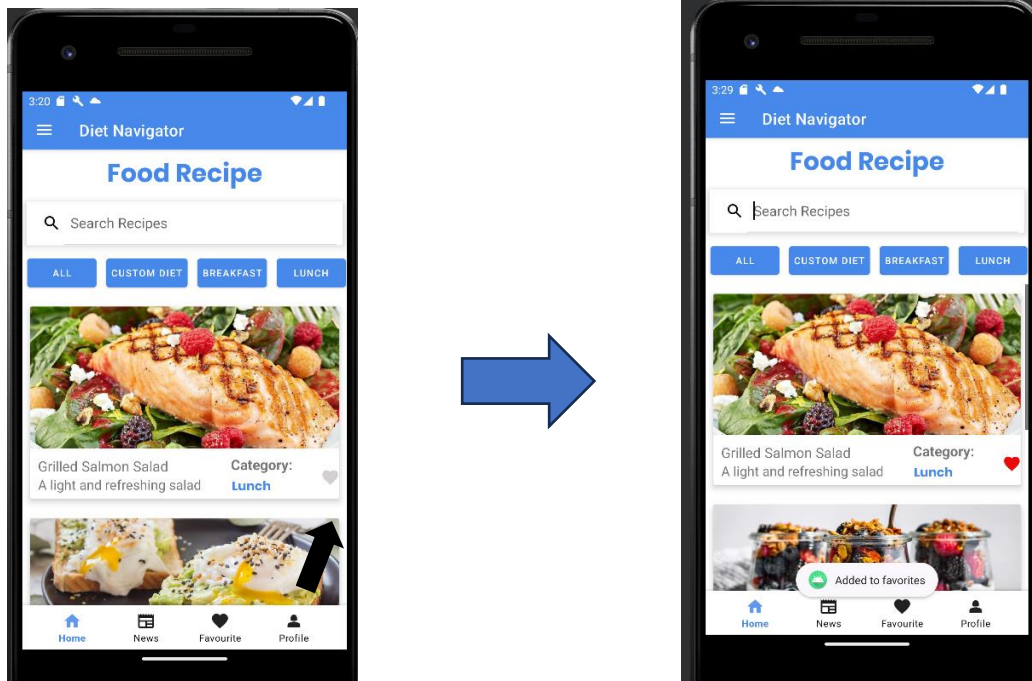
2.4.2. You will be directed to the Food Recipe Details page, which includes the following information:

- i. Food Name: The name of the recipe.
- ii. Food Description: A brief description of the recipe.
- iii. Ingredients: A list of ingredients required for the recipe.
- iv. Steps: Detailed steps to prepare the recipe.
- v. Category: The category to which the recipe belongs.



## 2.5. Adding Recipes to Favourites

2.5.1. Click on the heart icon at the bottom right of the recipe to add it to your favourites.

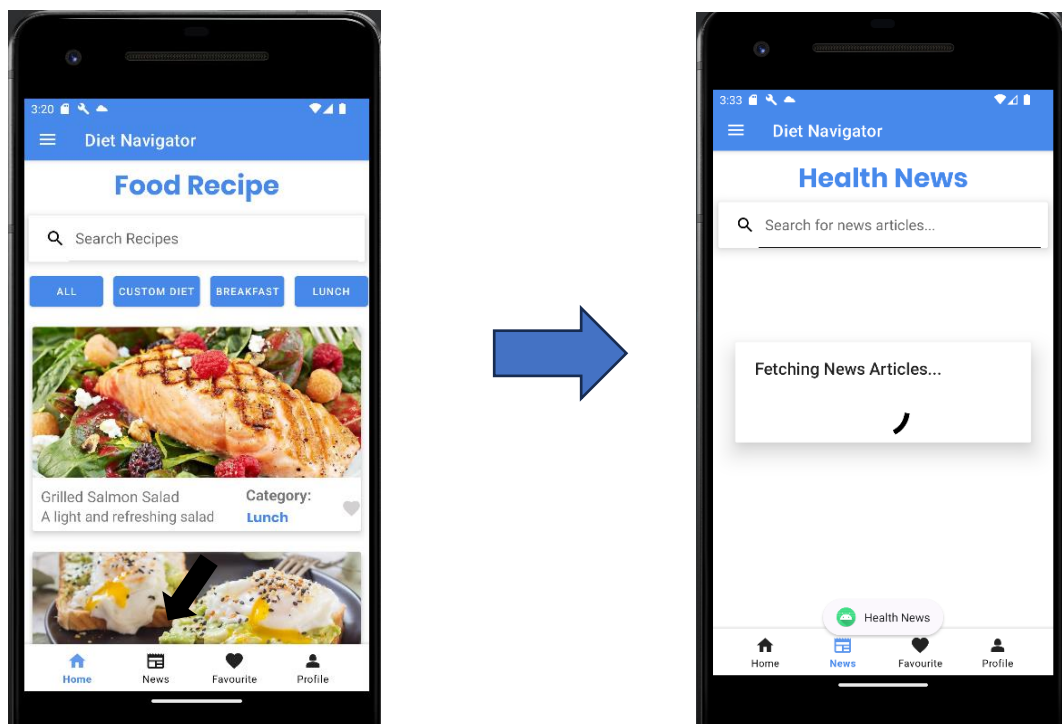


## 3. Health News

### 3.1. Default News Display

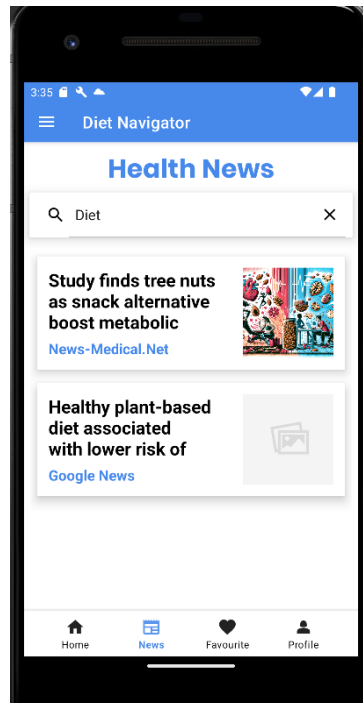
3.1.1. Access the "Health News" section from the bottom navigation.

3.1.2. The app will fetch and display health-related news articles by default.



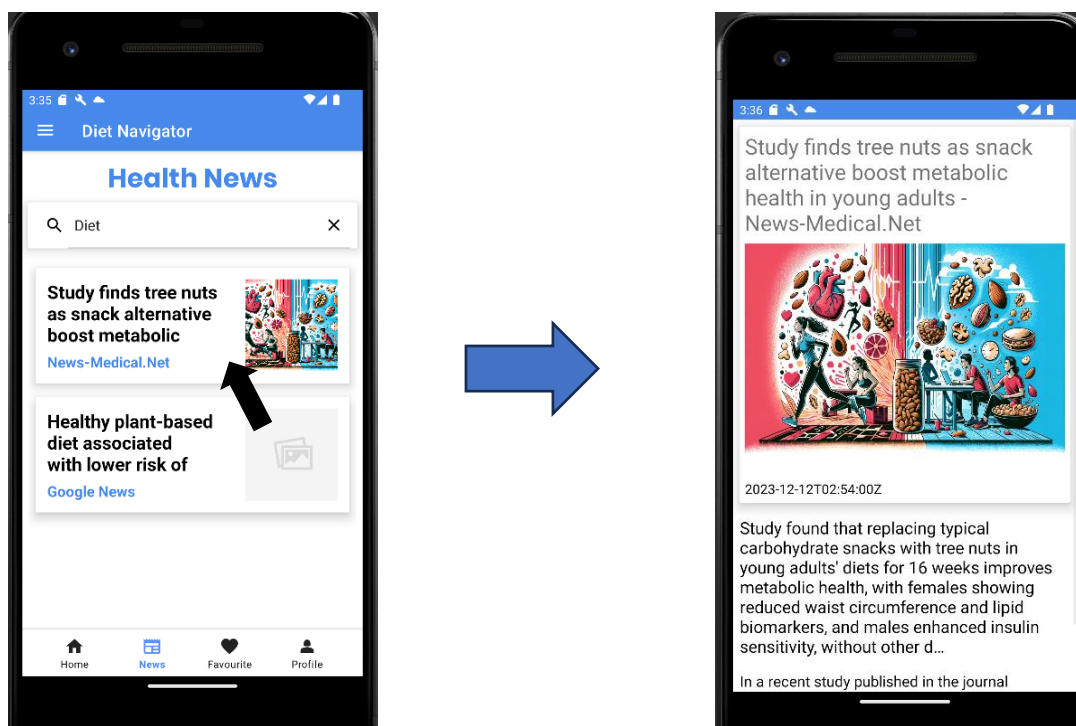
### 3.2. News Search

- 3.2.1. Use the search functionality to enter specific keywords.
- 3.2.2. The app will update the displayed news articles based on your search query.



### 3.3. Navigation to Full Article

- 3.3.1. Click on a specific news article to read the full content.



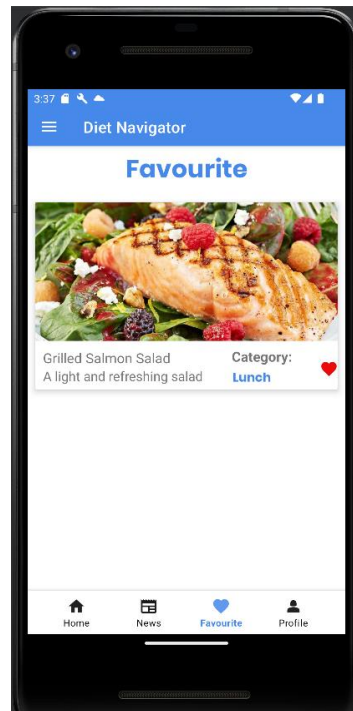


## 4. Favourite Recipes

### 4.1. Viewing Your Favourite Recipes

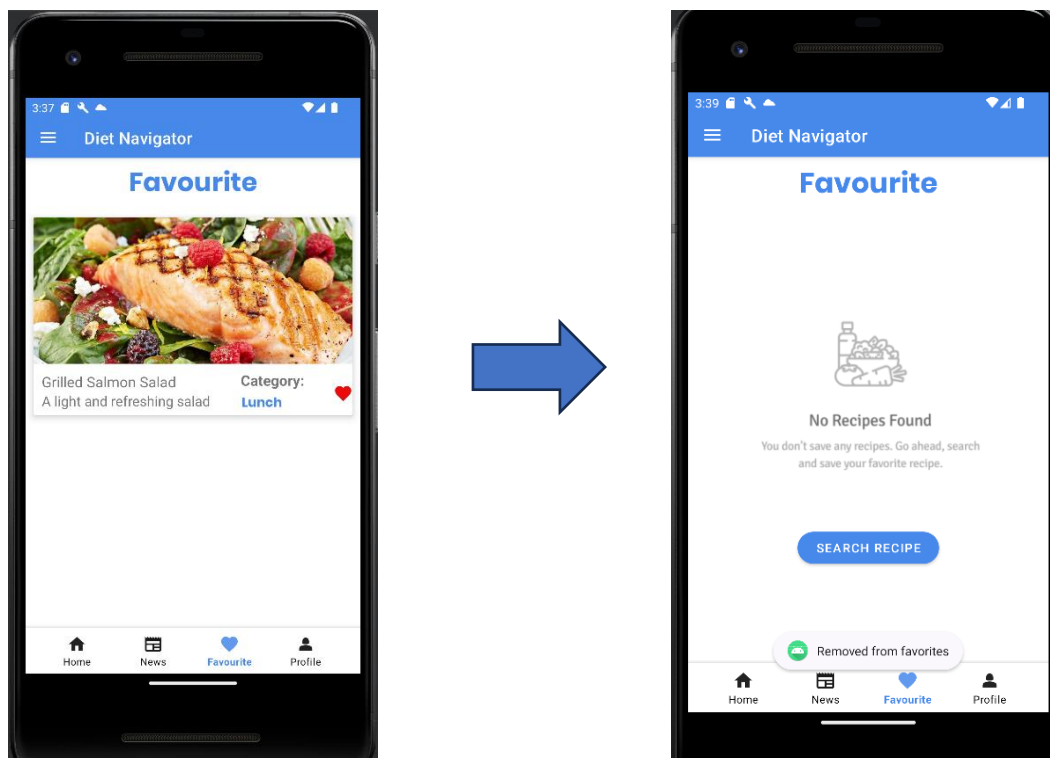
4.1.1. Navigate to the "Favourites" section in the app.

4.1.2. You can view and manage recipes added to your favourites.



### 4.2. Remove Recipes from Favourites

4.2.1. Click on the heart icon to remove the recipe from your favourites.





## 5. User Profile

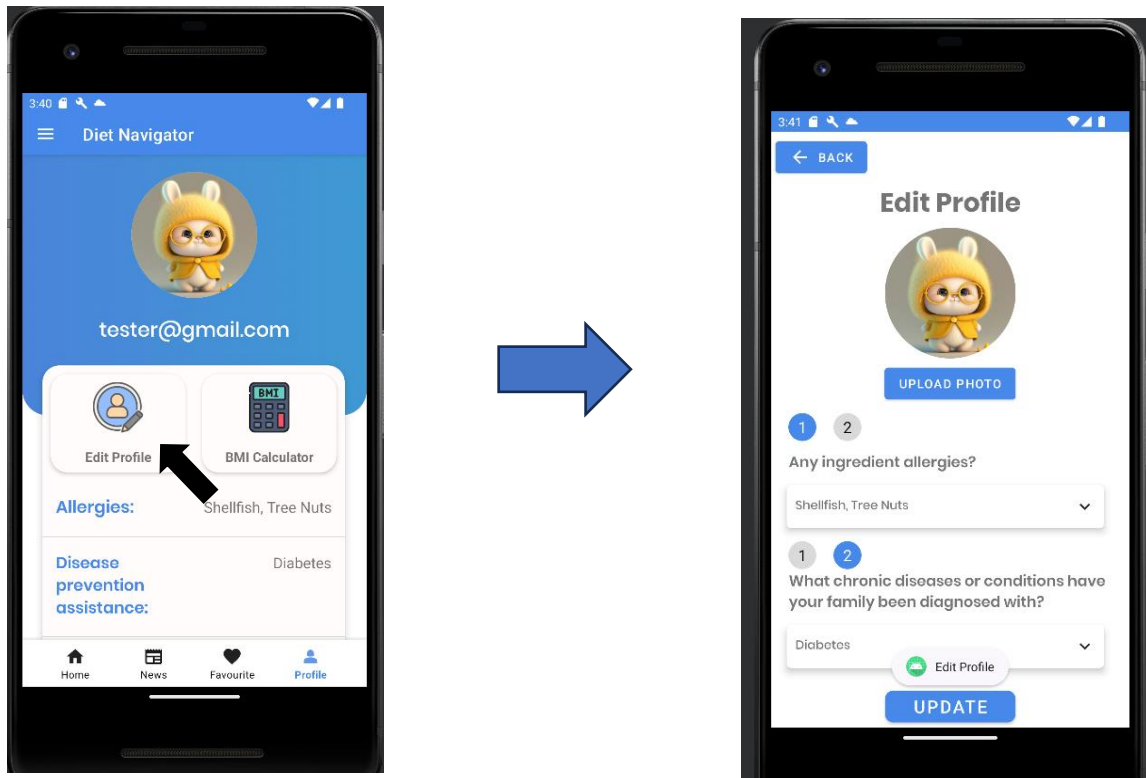
### 5.1. Editing Your Profile

5.1.1. Navigate to the "Profile" section from the bottom navigation.

5.1.2. Click on the "Edit Profile" option.

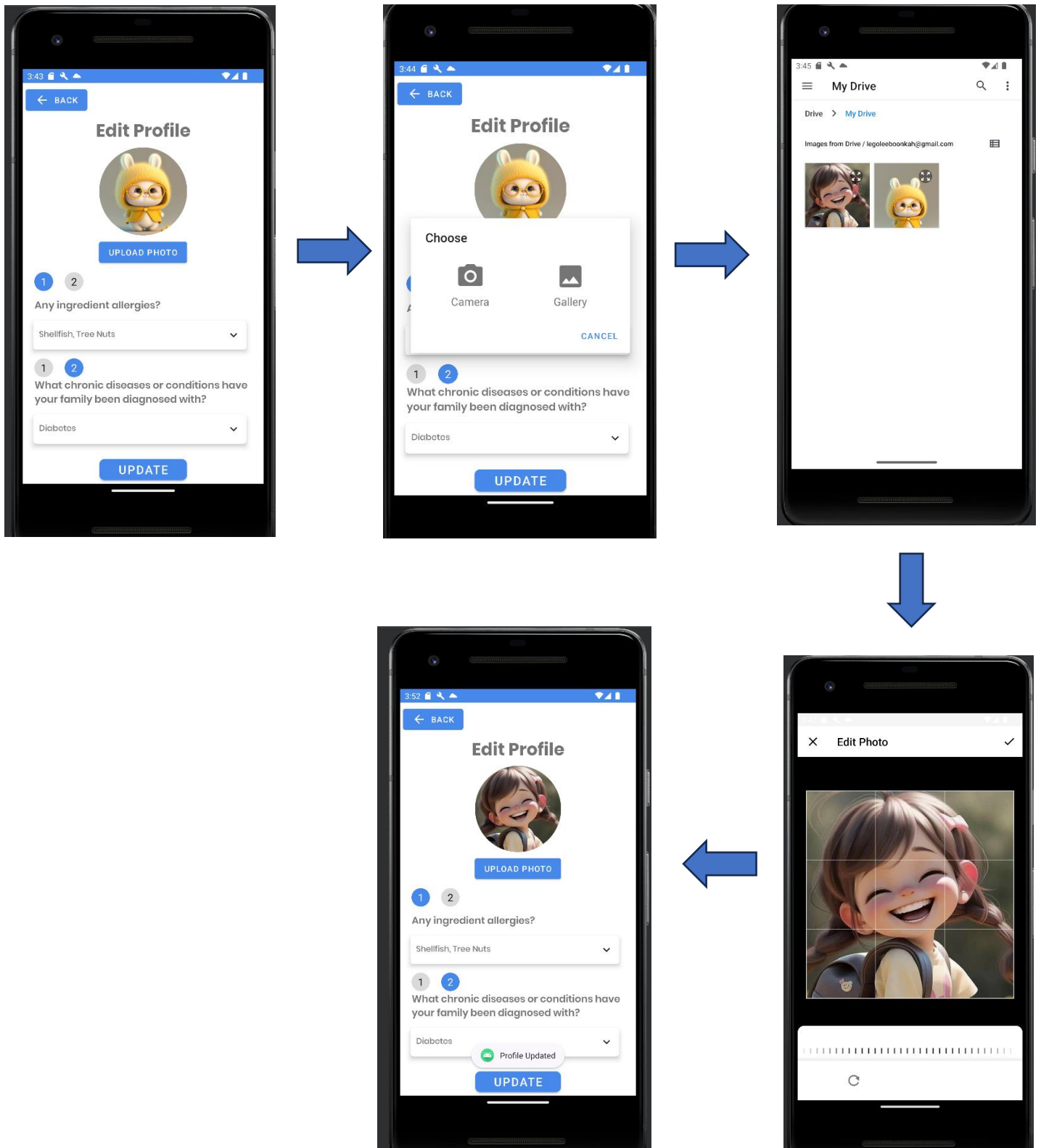
5.1.3. Modify your allergies and diseases as needed.

5.1.4. Click on "Update" to update your profile.



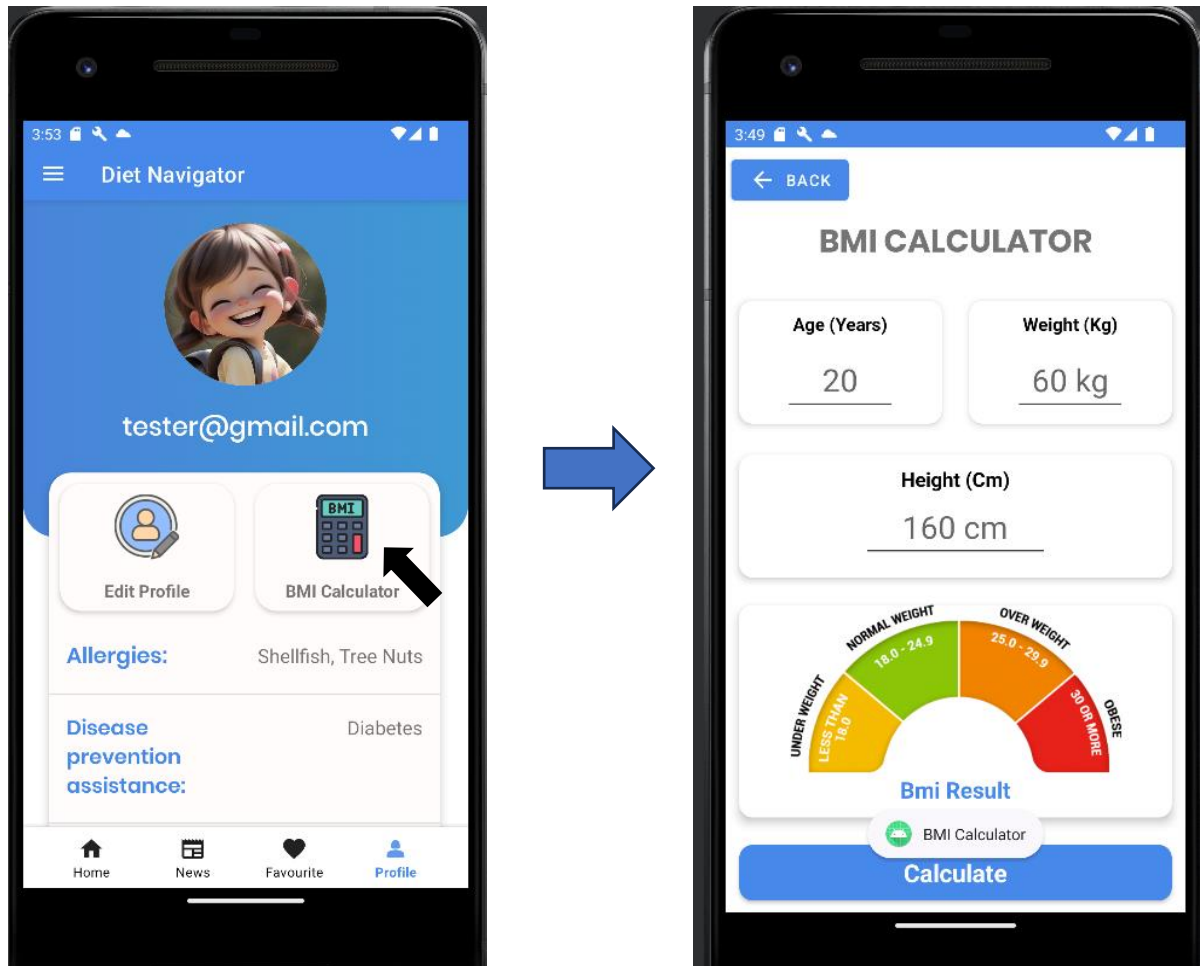
## 5.2. Uploading a Profile Picture

- 5.2.1. On the "Edit Profile" page, click on the "Upload Picture" button.
- 5.2.2. Choose an image from your device using the image picker.
- 5.2.3. Click on "✓" on the top right to upload and set your profile picture.
- 5.2.4. Click on "Update" to update your profile picture.



### 5.3. BMI Calculator

- 5.3.1. On the "Profile" page, click on the BMI calculator icon.
- 5.3.2. Enter your age, weight (in kilograms), and height (in centimetres).
- 5.3.3. Click the "Calculate" button to determine your BMI.



## 6. Admin Features (Admin Users Only)

### 6.1. Recipe Upload

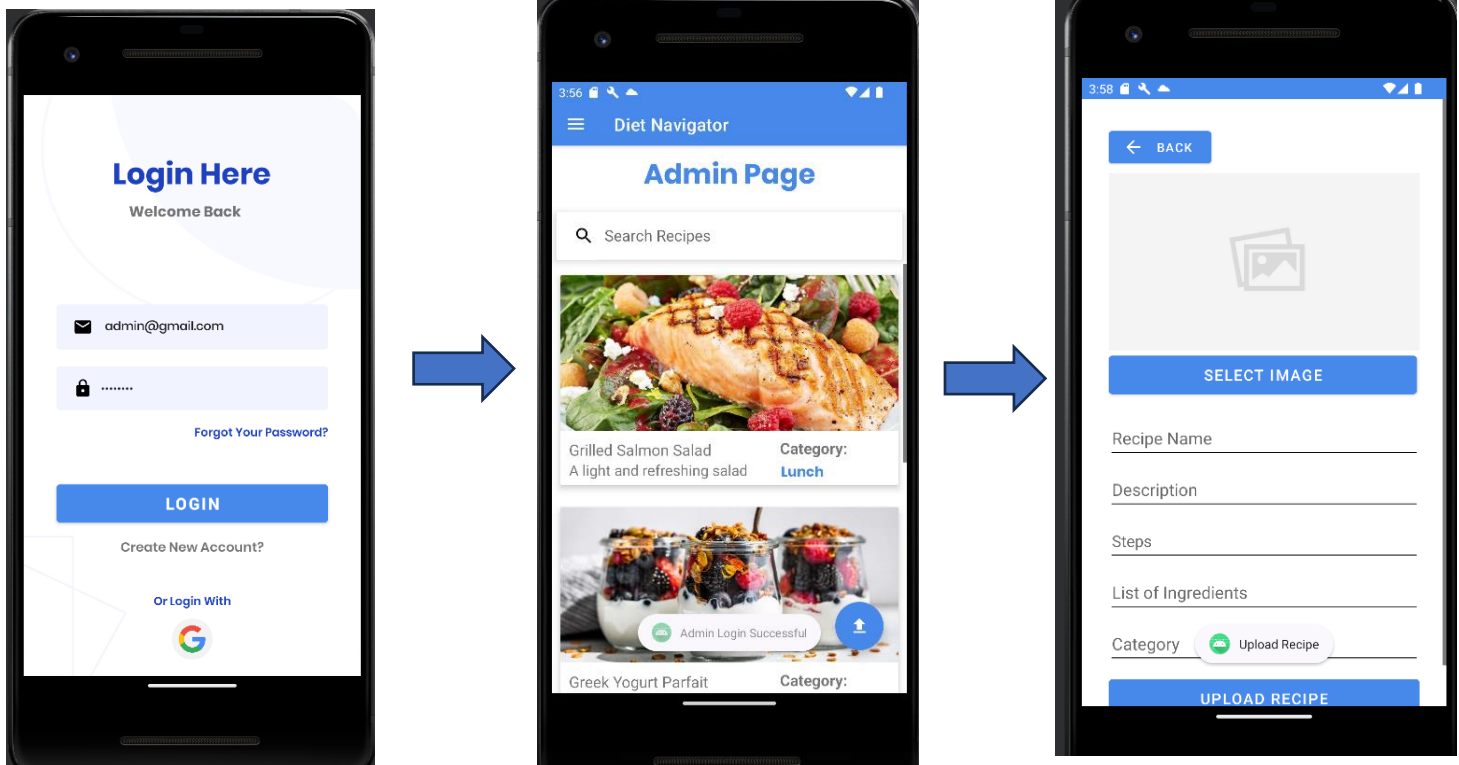
6.1.1. Log in as an admin using the credentials:

- i. Email address: admin@gmail.com
- ii. Password: admin123

6.1.2. Click on the floating button at the bottom right of the page to access the recipe upload section.

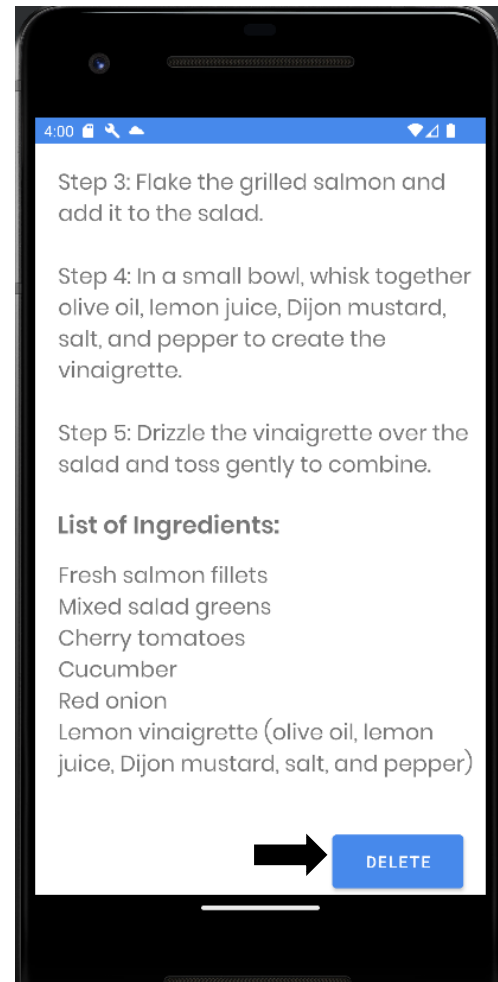
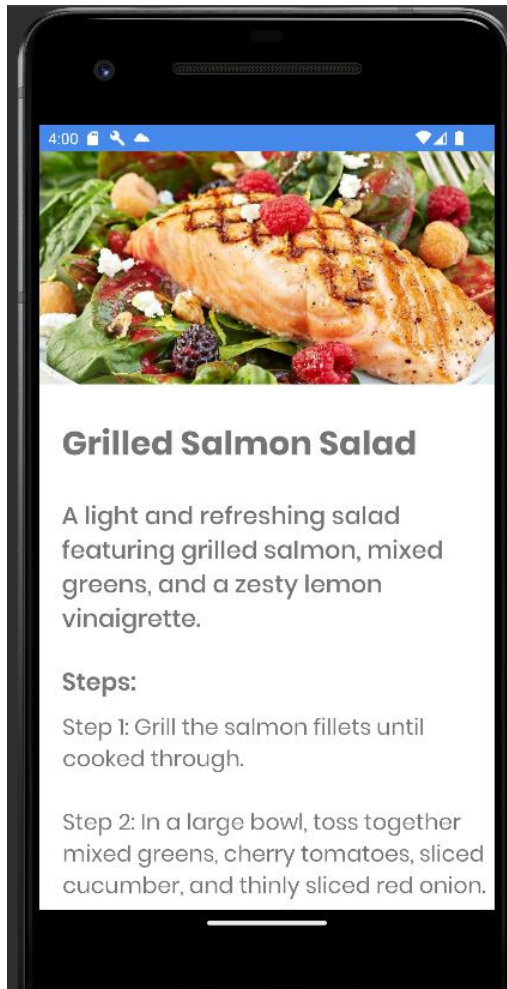
6.1.3. Input recipe details and choose an image for the recipe.

6.1.4. Click on "Upload Recipe" to add it to the database.



## 6.2. Recipe Deletion

- 6.2.1. Log in as an admin.
- 6.2.2. Click on the recipe that you want to delete.
- 6.2.3. Scroll to the bottom of the recipe details page.
- 6.2.4. Click on the “DELETE” button.



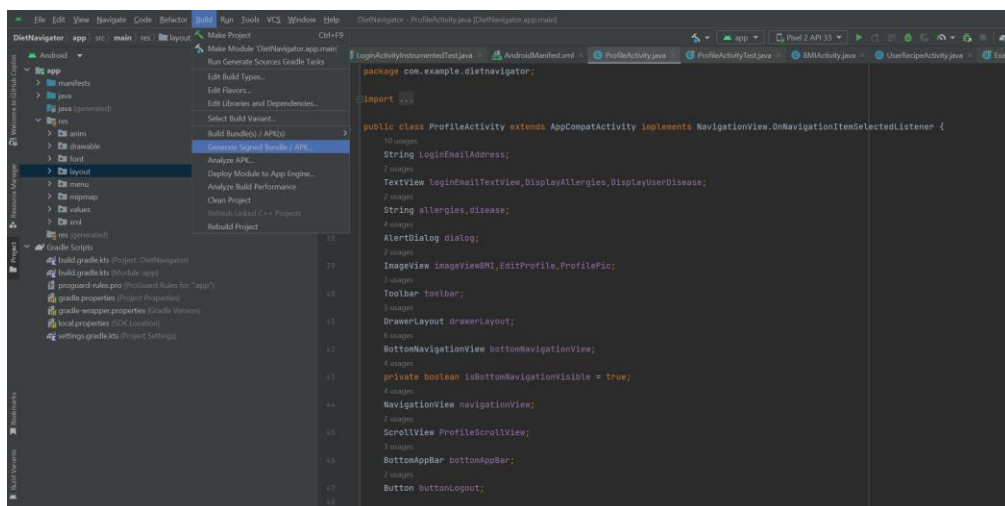
## Deployment

### 1. Build the Release Version of the App:

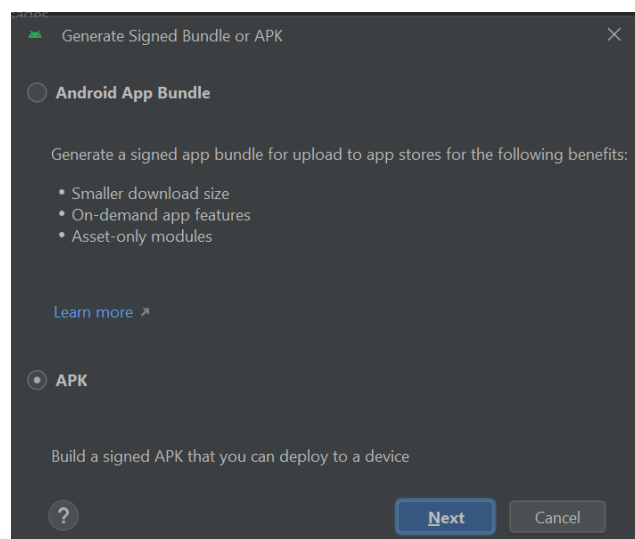
#### Generate Signed Bundle / APK

Generating a signed APK is a crucial step in deploying your Android app to production environments like Google Play. This signed APK file acts as a digital signature, ensuring the app's authenticity and protecting against unauthorized modifications.

- Open the Generate Signed Bundle or APK Dialog:
  1. Navigate to Build > Generate Signed Bundle/APK.

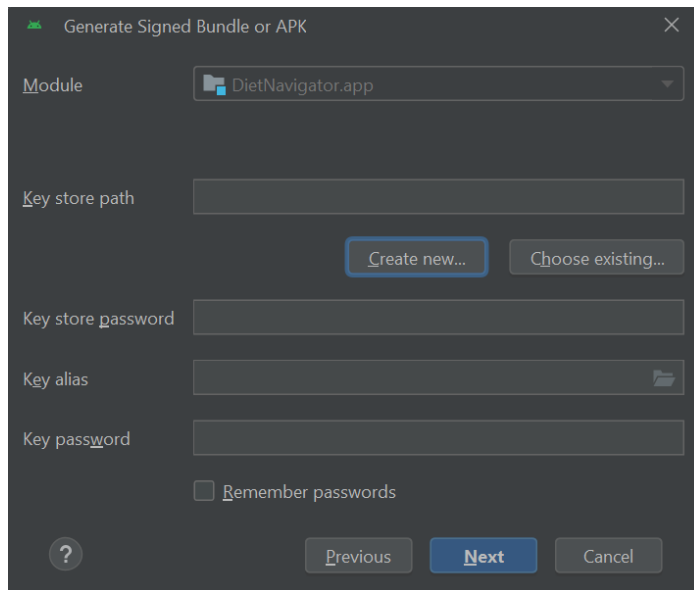


- Select Build Type:
  2. Choose APK
  3. Click Next.



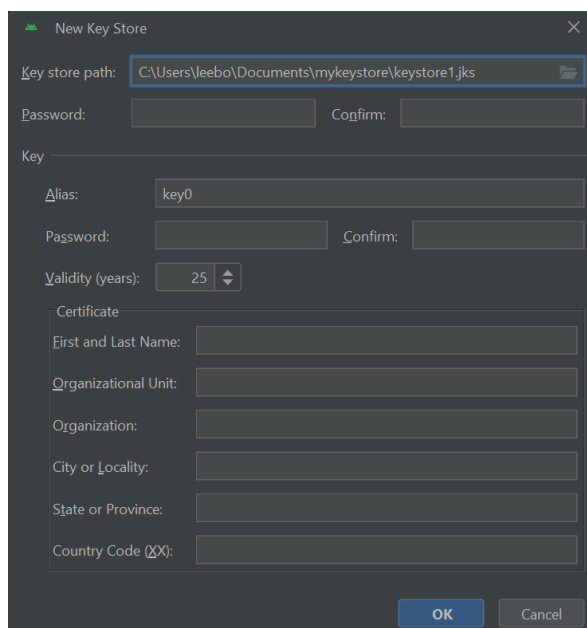
- Select Module:

4. Choose the module you want to generate the signed build for. Usually, this will be your app module.



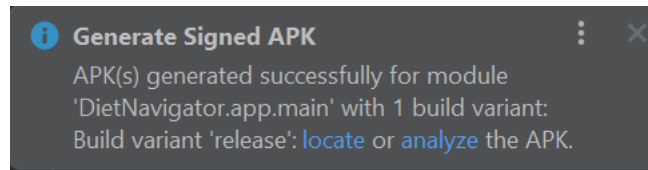
- Choose Keystore or Create a New One:











5. If already have a keystore containing your signing key, select it from the list.
6. If no, click Create New
7. Provide details like keystore location, password, key alias, and key password. Remember to choose strong passwords and keep them safe.
8. Click Next.





- Click Finish to generate the signed APK or App Bundle.

9. The APK file is in the release folder.



	.gradle	29/11/2023 9:37 PM	File folder	
	.idea	10/12/2023 11:22 PM	File folder	
	build	12/12/2023 9:24 PM	File folder	
	libs	26/11/2023 4:21 PM	File folder	
	release	12/12/2023 9:46 PM	File folder	
	src	10/12/2023 11:08 PM	File folder	
	.gitignore	26/11/2023 4:21 PM	Git Ignore Source ...	1 KB
	build.gradle.kts	11/12/2023 10:50 PM	KTS File	2 KB
	google-services	29/11/2023 9:35 PM	JSON Source File	2 KB
	proguard-rules.pro	26/11/2023 4:21 PM	PRO File	1 KB

	app-release.apk	12/12/2023 9:46 PM	APK File	9,871 KB
	output-metadata	12/12/2023 9:46 PM	JSON Source File	1 KB

## 2. Firebase Configuration:

Ensure that project is configured for the production environment. This includes updating your Firebase project settings, authentication settings, and database rules to match the production requirements.

## 3. Set Up Firebase Realtime Database Rules:

Ensure that your Firebase Realtime Database has appropriate security rules for the production environment. Restrict access to data as needed to protect user information.



4. Prepare Firebase Cloud Functions:

Ensure that the functions are set up correctly for the production environment and adjust environment-specific configurations.

5. Prepare Firebase Authentication (Optional):

If the app relies on Firebase Authentication, make sure it's properly configured for production. This may include setting up OAuth providers, configuring sign-in methods, and adjusting security settings.

6. Create Release Notes and Documentation:

Prepare release notes for users, detailing new features, bug fixes, and any other relevant information. Update the user documentation to reflect any changes or improvements in the production version.

7. Distribute the App:

There are various ways to distribute the app to users:

- Google Play Store (Android):
  - i. Create a Developer Account on the Google Play Console.
  - ii. Follow the steps to create a new app listing.
  - iii. Upload the APK file.
  - iv. Set up pricing and distribution settings.
- Apple App Store (iOS):
  - i. Enrol in the Apple Developer Program.
  - ii. Create an App Store Connect account.
  - iii. Follow the steps to create a new app listing.
  - iv. Upload your app using Xcode.

## Assumption of the app

### 1. Valid Email Addresses and Passwords:

- Assumption: Users are expected to provide valid email addresses and passwords during the registration and login processes.
- Rationale: This assumption aligns with standard authentication practices. Registered users typically need to provide accurate and valid credentials for secure access to their accounts.

### 2. Users possess basic knowledge of Android smartphones and app usage:

- Navigating through the app's interface, utilizing functionalities like search and filtering, and understanding the displayed information requires basic knowledge of Android smartphones and app usage.
- Rationale: We assume users are familiar with these aspects for smooth interaction with the Diet Navigator app.

### 3. Firebase Authentication Service Availability:

- Assumption: The Firebase Authentication service is assumed to be operational and available during the login process.
- Rationale: The Diet Navigator app relies on Firebase Authentication for user login, registration, and password reset. Assuming the availability of Firebase Authentication is common in projects utilizing Firebase services.

### 4. Admin Identification:

- Assumption: Admins are identified based on a specific email address ("admin@gmail.com").
- Rationale: To distinguish between regular users and administrators, a straightforward assumption is made that administrators will use a specific email address for authentication.

5. Stable Internet Connection:

- Assumption: Users must have a stable internet connection for the authentication process.
- Rationale: Since the app relies on Firebase services, and internet connectivity is necessary for communication with Firebase Authentication, Firestore, Realtime Database, and Cloud Storage. Online functionality is typical for applications with cloud-based services.

6. Dynamic Visibility of Bottom Navigation:

- Assumption: The dynamic visibility of the bottom navigation and bottom app bar enhances the user experience during recipe browsing.
- Rationale: Dynamically adjusting the visibility of navigation elements is a common practice to provide a more focused and intuitive user interface. It assumes that users benefit from a simplified interface when browsing recipes.

7. Pre-Stored User Profile Information:

- Assumption: User profile information, including allergies and diseases, is assumed to be pre-stored in the Firestore database during the user registration process.
- Rationale: During user registration, it is assumed that users provide information about their allergies and diseases. This information is stored in Firestore, allowing the app to provide personalized recommendations based on user preferences.

8. Image Upload Time Variation:

- Assumption: The image upload process may take some time based on the image size and network speed.
- Rationale: Uploading images to cloud storage can be influenced by factors such as image size and network speed. Users are assumed to be aware that upload times may vary based on these factors.

9. Third-party API Reliability:

- Assumption: The app relies on external APIs for functions like news article retrieval. These APIs are assumed to be reliable and available.
- Rationale: While the app utilizes reliable APIs, unexpected outages or instability may occur.

10. . User Comprehension of Health Information:

- Assumption: Users possess basic understanding of health and nutrition concepts presented in the app.
- Rationale: The app provides information and recommendations, but users are expected to understand and interpret them based on their individual needs and medical advice.

These assumptions are made to provide a realistic context for the functionality and user experience of the Diet Navigator app. It is crucial to continuously monitor and adjust assumptions to ensure the app remains user-friendly, reliable, and accessible.