Project report

Psarros Filippos

A full analysis of the project is provided in the Recipe_app_report.pdf file.

Project Description

This project involves the development of a mobile application for recording and managing cooking recipes, using the Flutter framework and the local Hive database. The main goal of the app is to allow users to add, view, sort, and delete recipes while providing a pleasant and functional user interface, supporting both light and dark modes.

The application is implemented with a focus on modern UI/UX design. It supports image input from the user's gallery, features a star rating system, and allows sorting of recipes based on preparation time, difficulty, or rating. The app is built following best Flutter practices, and Hive ensures persistent local data storage without the need for an external connection or server.

The application integrates several techniques, such as:

- Use of Provider for dynamic theme switching
- Cards with responsive design and ripple effects
- Swipe-to-delete functionality with Undo option via SnackBar
- Safe handling and display of user data

This is a complete mobile application combining frontend technologies and local storage, offering users a pleasant and practical tool to organize their favorite recipes.

Project Implementation

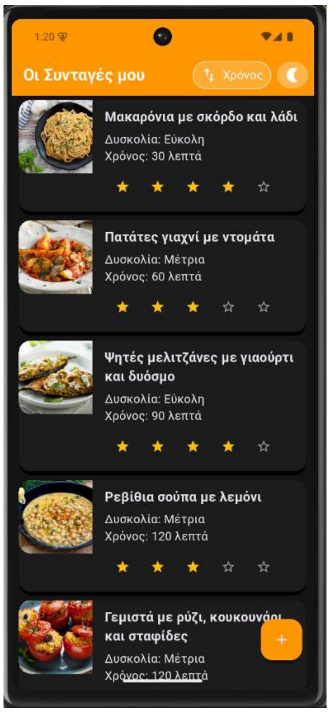
To implement this project, Flutter and Android Studio were installed. An Android device emulator was used for testing. The Flutter extension was added to VS Code, where the application was developed in the Dart programming language. A new Flutter project was created, and the code was run directly on the device for real-time preview and debugging.

Code Functionality and Results

Initially, the application structure was created, with each feature implemented in separate files as shown below:

- In lib/models/, the recipe.dart model was created, defining the core structure of each recipe. It includes fields for:
    - Title

    - Detailed description
    - Preparation time in minutes
    - Difficulty ("Easy", "Medium", "Hard")
    - Image path (local or URL)
    - Rating (0 to 5 stars)
    - Ingredients
- The main.dart file sets up the MaterialApp, defines routing between screens, and handles theme selection with ThemeMode.
- In lib/screens/, the home_screen.dart file was created. It contains:
    - A Switch for toggling between light and dark themes
    - A PopupMenuButton for sorting recipes
    - A ListView that displays all recipes stored in Hive
    - Each recipe is rendered using a custom RecipeCard widget
- Deleting a recipe is supported via a swipe gesture (Dismissible) with a SnackBar and Undo functionality.



Μακαρόνια με σκόρδο και λάδι
Δυσκολία: Εύκολη
Χρόνος: 30 λεπτά
★ ★ ★ ★ ☆

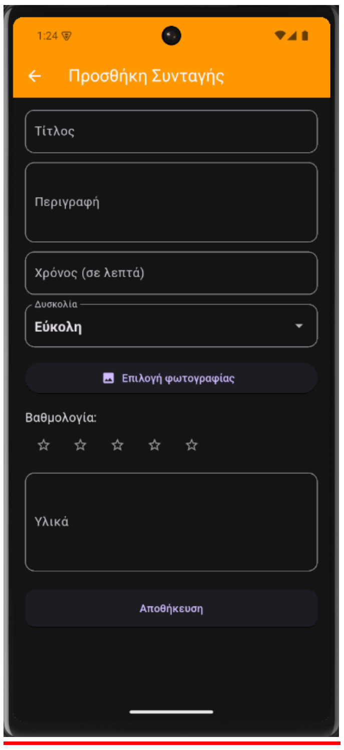
Πατάτες γιαχνί με ντομάτα
Δυσκολία: Μέτρια
Χρόνος: 60 λεπτά
★ ★ ★ ☆ ☆

Ψητές μελιτζάνες με γιαούρτι και δυόσμο
Δυσκολία: Εύκολη
Χρόνος: 90 λεπτά
★ ★ ★ ★ ☆

Ρεβίθια σούπα με λεμόνι
Δυσκολία: Μέτρια
Χρόνος: 120 λεπτά
★ ★ ★ ☆ ☆

Γεμιστά με ρύζι, κουκουνάρι και σταφίδες
Δυσκολία: Μέτρια
Χρόνος: 120 λεπτά

The file add_recipe_screen.dart was created in lib/screens, where the user can add a new recipe with custom details. Specifically, it includes a form using Form and TextFormField, a dropdown menu to select the difficulty level, and an image picker implemented with ImagePicker, allowing the user to choose a photo from the device’s gallery after granting the necessary permissions. The screen also supports adding ingredients and selecting a rating between 1 and 5 stars.



1:24

Προσθήκη Συνταγής

Τίτλος

Περιγραφή

Χρόνος (σε λεπτά)

Δυσκολία

Εύκολη

Επιλογή φωτογραφίας

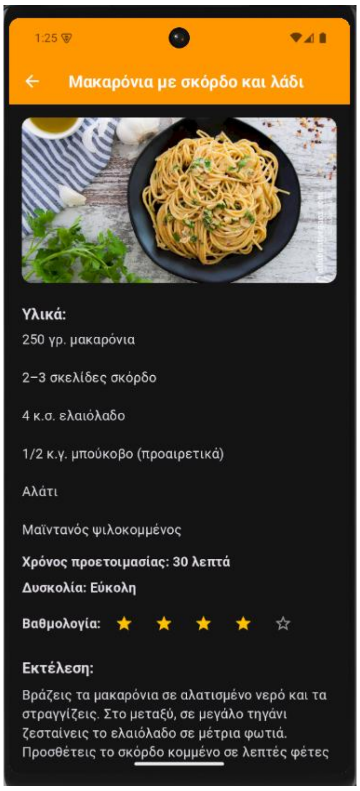
Βαθμολογία:

☆ ☆ ☆ ☆ ☆

Υλικά

Αποθήκευση

The file recipe_detail_screen.dart was created in lib/screens, which displays the details of a recipe (title, ingredients, instructions, rating). It also includes the use of Hero animation for the recipe image and a multi-line title in the AppBar with dark-colored font styling.



1:25

Μακαρόνια με σκόρδο και λάδι

Υλικά:

250 γρ. μακαρόνια

2–3 σκελίδες σκόρδο

4 κ.σ. ελαιόλαδο

1/2 κ.γ. μπούκοβο (προαιρετικά)

Αλάτι

Μαϊντανός ψιλοκομμένος

Χρόνος προετοιμασίας: 30 λεπτά

Δυσκολία: Εύκολη

Βαθμολογία: ★ ★ ★ ★ ☆

Εκτέλεση:

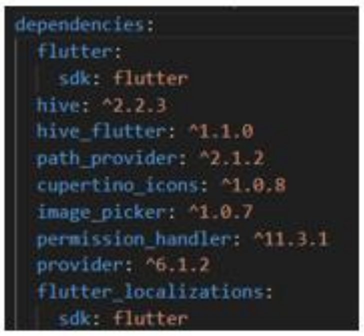
Βράζεις τα μακαρόνια σε αλατισμένο νερό και τα στραγγίζεις. Στο μεταξύ, σε μεγάλο τηγάνι ζεσταίνεις το ελαιόλαδο σε μέτρια φωτιά. Προσθέτεις το σκόρδο κομμένο σε λεπτές φέτες

The file recipe_card.dart was created in lib/widgets, which displays each recipe on the home screen. It features a rounded card using RoundedRectangleBorder, an InkWell ripple effect when tapped, and shows the following information: title, difficulty, rating, preparation time. It also includes a Hero animation for the recipe image.

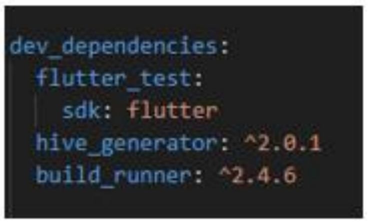
The file star_rating.dart was also created in lib/screen, which displays star icons and renders the rating with corresponding colors (amber/grey).

Position	**Icon**	**Color**
< rating	Icons.star	Colors.amber
≥ rating	Icons.star_border	Colors.grey

For storing data on the device, Hive was used, and the following steps were required: Adding dependencies in the pubspec.yaml file.



```yaml
dependencies:
  flutter:
    sdk: flutter
  hive: ^2.2.3
  hive_flutter: ^1.1.0
  path_provider: ^2.1.2
  cupertino_icons: ^1.0.8
  image_picker: ^1.0.7
  permission_handler: ^11.3.1
  provider: ^6.1.2
  flutter_localizations:
    sdk: flutter
```



```yaml
dev_dependencies:
  flutter_test:
    sdk: flutter
  hive_generator: ^2.0.1
  build_runner: ^2.4.6
```

Modification of the Recipe model for Hive, where the Recipe class is properly structured using Hive annotations such as @HiveType and @HiveField, in order to enable it to be stored.

Then, the command flutter pub run build_runner build is executed to generate the recipe.g.dart file, which is required to support the storage of objects.

Hive is initialized in main.dart for local storage, where the RecipeAdapter is registered and the recipes and settings boxes are opened. The code uses Provider to manage the theme (light/dark), and through MaterialApp, the available screens (home and add recipe) are defined.

This way, Hive enables storing and retrieving recipes locally on the user's device.

Saving recipes in the AddRecipeScreen: When the form (title, description, prep time, difficulty, ingredients, photo, and rating) is submitted, a Recipe object is created and stored locally in Hive under the 'recipes' box. This ensures that the recipe is saved on the device.

Loading Recipes in the HomeScreen

The HomeScreen is the main screen of the application and displays all the recipes that have been stored locally using Hive in the 'recipes' box. Upon launch, the recipes are loaded from Hive and sorted based on the selected option (prep time, rating, or difficulty). The user can add a new recipe by navigating to the AddRecipeScreen, view recipe details, delete a recipe with an undo option, and switch between light and dark themes using the Provider. Hive ensures persistent local storage of data directly on the device.

SWIPE DELETE

Swipe-to-delete functionality was implemented, allowing the user to delete a recipe permanently from both the UI and the Hive database by swiping a recipe card. Within the ListView.builder, each recipe card is wrapped in a Dismissible widget.

Element	**Description**
Dismissible	Enables swipe action for each list item
key	Must be unique, e.g., title + index
box.deleteAt(index)	Deletes the recipe from the Hive database
recipes.removeAt(index)	Removes the recipe from the visible list
SnackBar	Displays a confirmation message after deletion

An undo option was also added. If a user accidentally deletes a recipe, they have 3 seconds to restore it. The recipe is temporarily removed, and a SnackBar appears with an “UNDO” button. If the user taps “UNDO” within the time limit, the deletion is canceled. Otherwise, the recipe is permanently removed from Hive.

Functionality	**Description**
recipes.removeAt()	Temporarily removes the recipe from the screen
SnackBarAction	Button for undoing the deletion
Future.delayed	Waits for 3 seconds before final deletion
if (!recipes.contains())	If undo wasn’t triggered, deletes from Hive

SORTING

A sorting dropdown was added to the top-right corner of the screen (in the AppBar), offering the following options:

- Prep Time
- Rating
- Difficulty

The sorting changes the display order of recipes in real time. An extra feature was added: the sort options are now accessed through an icon for cleaner UI.

Component	**Description**
PopupMenuButton	Displays the sort options in a popup menu
OutlinedButton.icon	A button with both an icon and label
RoundedRectangleBorder	Makes the button have rounded corners
swap_vert	Icon used to represent sorting (↑↓)
foregroundColor: Colors.white	Displays the button in white on a blue AppBar

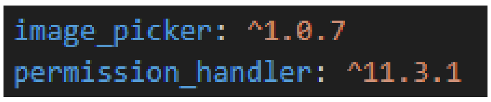
DEVICE IMAGE GALLERY PERMISSIONS

Permission handling was implemented to allow the app to access the user's device gallery and import images into the application. This allows the user to select photos from their phone's storage and attach them to a recipe.

This was achieved by adding the necessary dependencies in pubspec.yaml:

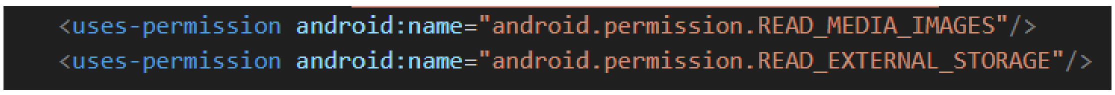
- image_picker
- permission_handler

These packages enabled requesting runtime permission and selecting images from the gallery.



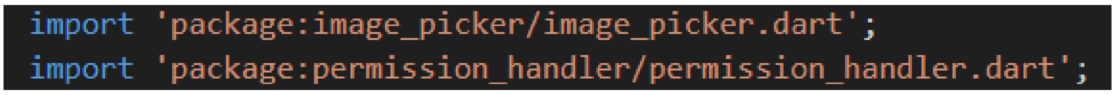
```
image_picker: ^1.0.7
permission_handler: ^11.3.1
```

Additionally, permission to access the device was granted by adding the following lines to the android/app/src/main/AndroidManifest.xml file:



```
    <uses-permission android:name="android.permission.READ_MEDIA_IMAGES"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

Also, in the add_recipe_screen, the imagePath field was changed from String to XFile? selectedImage;
and the following imports were added:



```
import 'package:image_picker/image_picker.dart';
import 'package:permission_handler/permission_handler.dart';
```

Finally, a Future function was created, which uses the _pickImage() method to request permission to access photos. If permission is granted, it opens the gallery allowing the user to select an image. If an image is selected, it saves the image and its path. If the permission is denied, an error message is displayed.

Step	**Description**
permission_handler	Requests access to device photos
image_picker	Allows the user to pick an image
XFile	Contains the image path for storage/preview
Image.file(File(path))	Displays the selected image inside the app

Then, the recipe_card.dart code was updated to show the selected image, or display a placeholder icon if no image was selected.

State	**What it Displays**
imagePath.isEmpty	image_not_supported icon
imagePath.startsWith('/')	Image from device (via Image.file)
Else (e.g. assets/...)	Image from assets (via Image.asset)

DARK/LIGHT MODE THEMING INTEGRATION

Theming functionality was also added, allowing the user to automatically switch between light and dark mode based on system settings. This was implemented as follows:

- First, the file app_theme.dart was created in lib/theme/, where two theme styles are defined: light and dark.
- Then, the main.dart code was updated to integrate theming support across all screens of the application.

The file theme_notifier.dart was also created inside lib/theme/. Each time the theme changes, the new preference is saved and automatically reloaded the next time the app is launched.

Additionally, the user's dark/light theme selection is stored using Hive, ensuring that the chosen theme persists across sessions.

An extra feature was also added: a **custom switch** that uses icons (Icons.nightlight and Icons.sunny) instead of a simple toggle.

Extra Features

Rounded Cards with Elevation:
Visually appealing cards with rounded corners and elevation (shadow), giving them a raised appearance from the background. This adds visual depth and clearly separates each recipe on the main list.

Hero Animation:
Smooth image transition from the recipe card to the detail screen using Flutter's Hero widget.

Ripple Effect (InkWell):
A wave-like ripple effect appears when a user taps a UI element like a recipe card, providing instant visual feedback.

Custom Page Transition (Slide):
Instead of the default screen transition, the app uses sliding transitions for a smoother navigation experience.

Shadows:
Widgets such as cards are rendered with drop shadows to make them look more realistic and elevated.

Greek Localization (locale el_GR):
The app supports Greek language and formatting for dates and numbers.

Clean Aesthetic (Elevation, Spacing):
The design includes well-thought-out spacing and elevation to maintain a clean and professional look.

Responsive Layout:
The layout adjusts smoothly to different screen sizes and devices.

Use Case Scenarios / App Flow

1. The user launches the app.
2. They see a list of saved recipes (or an empty screen if none exist).
3. They tap the + button to add a new recipe:

    - Fill in the title, description, and ingredients
    - Select an image from the device gallery
    - Set the preparation time and star rating
4. They save the recipe.
5. They return to the home screen where the new recipe appears in the list.
6. They can now:
    - Tap to view recipe details
    - Swipe to delete a recipe
    - Sort recipes by difficulty, time, or rating
    - Toggle between dark and light mode using the custom switch.