Ministerul Educaţiei, Culturii și Cercetării al Republicii Moldova
Universitatea Tehnică a Moldovei
Departamentul Ingineria Software și Automatică

RAPORT

Lucrare de laborator nr. 2, varianta II
Disciplina: PAM
Tema: Utilizare avansata de componente UI. Realizare app dupa design definit complex.

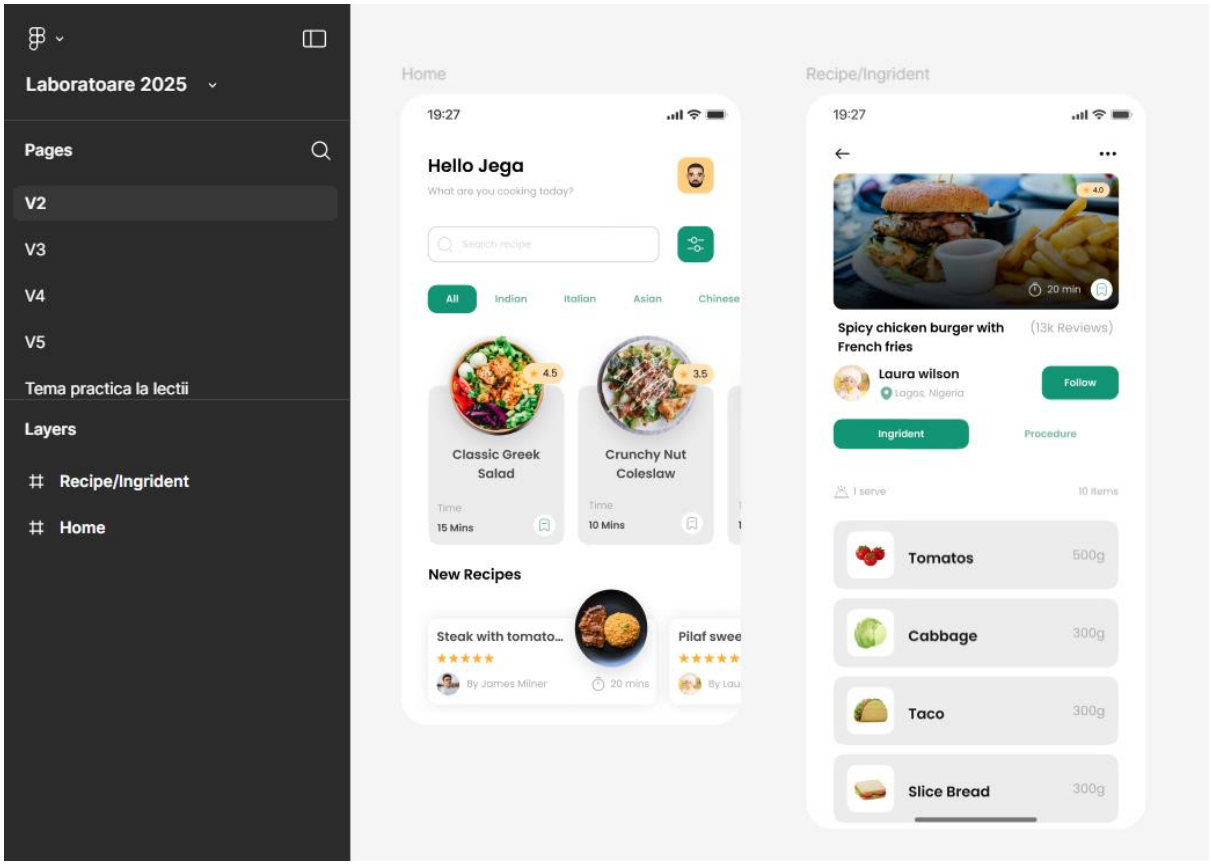
A efectuat: Mandiș Nichita TI-224

A verificat :
Buza D.

Chișinău 2025

Sarcina de lucru: Varianta 2.

Utilizare avansata de componente UI. Realizare app dupa design definit complex.
Designul prezentat în Figma:



Mersul lucrării:

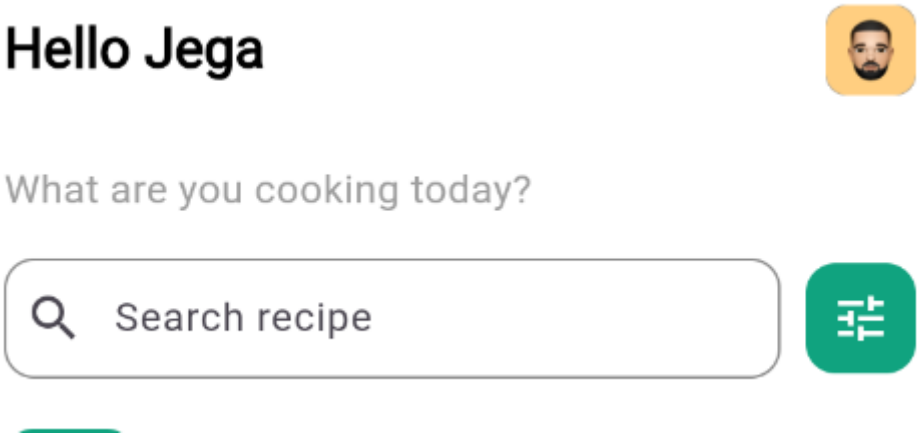
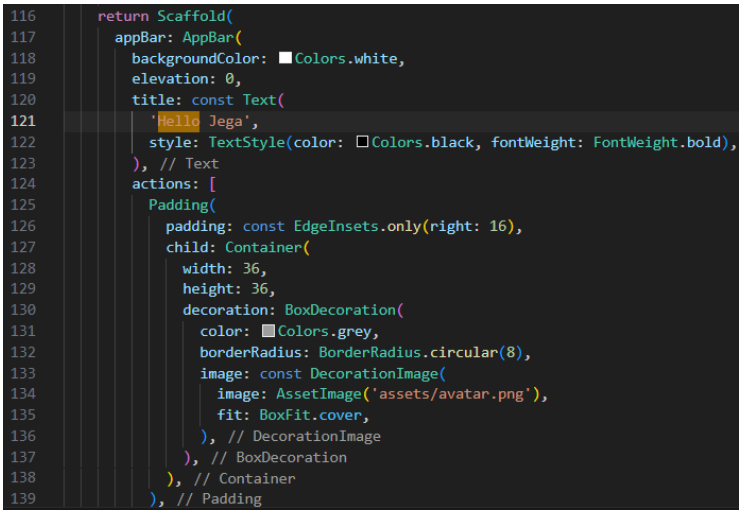


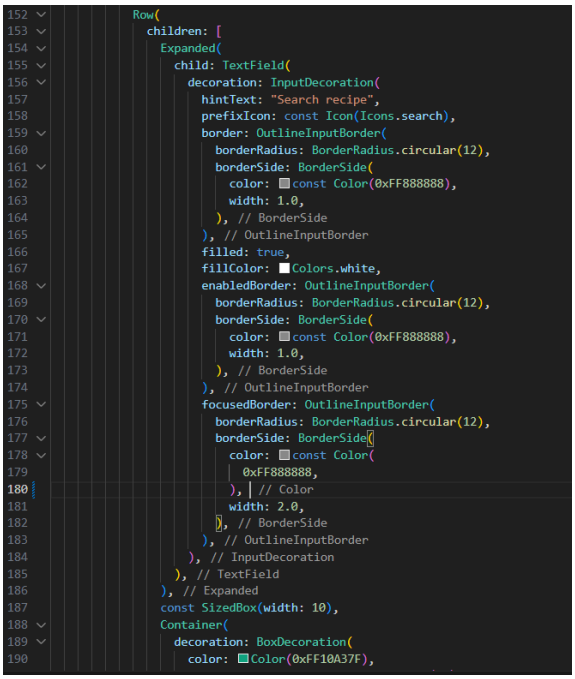
Figura 1 – Secțiunea cu search bar



```
return Scaffold(
  appBar: AppBar(
    backgroundColor: Colors.white,
    elevation: 0,
    title: const Text(
      'Hello Jega',
      style: TextStyle(color: Colors.black, fontWeight: FontWeight.bold),
    ), // Text
    actions: [
      Padding(
        padding: const EdgeInsets.only(right: 16),
        child: Container(
          width: 36,
          height: 36,
          decoration: BoxDecoration(
            color: Colors.grey,
            borderRadius: BorderRadius.circular(8),
            image: const DecorationImage(
              image: AssetImage('assets/avatar.png'),
              fit: BoxFit.cover,
            ), // DecorationImage
          ), // BoxDecoration
        ), // Container
      ), // Padding
```

Figura 1.2 – Titlul + imaginea cu avatar-ul user-ului

În acest fragment de cod sunt utilizate elemente pentru crearea unei interfețe grafice. Scaffold este structura de bază a unei pagini în Flutter și oferă cadrul principal în care pot fi plasate componente precum bara de sus, conținutul sau butoanele de navigare. AppBar reprezintă bara de titlu aflată în partea superioară a ecranului; în acest caz, are un fundal alb, fără umbră (elevation: 0) și conține un titlu și o acțiune în partea dreaptă. Text este utilizat pentru afișarea textului „Hello Jega”, având un stil personalizat — culoare neagră și text îngroșat. În interiorul actions, este folosit un Padding pentru a adăuga spațiu la dreapta, iar un Container cu proprietăți de BoxDecoration afișează un avatar, aplicând o culoare gri, colțuri rotunjite și o imagine din fișierul local assets/avatar.png. Astfel, aceste elemente lucrează împreună pentru a construi o bară de aplicație estetică și funcțională.



```
Row(
  children: [
    Expanded(
      child: TextField(
        decoration: InputDecoration(
          hintText: "Search recipe",
          prefixIcon: const Icon(Icons.search),
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(12),
            borderSide: BorderSide(
              color: const Color(0xFF888888),
              width: 1.0,
            ), // BorderSide
          ), // OutlineInputBorder
          filled: true,
          fillColor: Colors.white,
          enabledBorder: OutlineInputBorder(
            borderRadius: BorderRadius.circular(12),
            borderSide: BorderSide(
              color: const Color(0xFF888888),
              width: 1.0,
            ), // BorderSide
          ), // OutlineInputBorder
          focusedBorder: OutlineInputBorder(
            borderRadius: BorderRadius.circular(12),
            borderSide: BorderSide(
              color: const Color(
                0xFF888888,
              ), // Color
              width: 2.0,
            ), // BorderSide
          ), // OutlineInputBorder
        ), // InputDecoration
      ), // TextField
    ), // Expanded
    const SizedBox(width: 10),
    Container(
      decoration: BoxDecoration(
        color: Color(0xFF10A37F),
```

Figura 1.3 – SearchBar + butonul cu filtre

În acest fragment de cod sunt utilizate mai multe elemente UI pentru a construi secțiunea principală a paginii. SingleChildScrollView permite derularea conținutului pe

ecran, astfel încât toate elementele din interiorul său pot fi vizualizate chiar și pe ecrane mici. Proprietatea padding adaugă un spațiu uniform în jurul conținutului. În interior se află un Column, care aliniază elementele vertical, iar crossAxisAlignment: CrossAxisAlignment.start asigură alinierea lor la stânga. Primul element este un Text care afișează întrebarea „What are you cooking today?” cu o dimensiune mică și culoare gri. SizedBox este folosit pentru a adăuga spațiu între elemente. Urmează un Row, care aliniază orizontal un TextField și un Container cu o pictogramă. TextField este câmpul de căutare, având un InputDecoration ce include un indiciu (hintText), o pictogramă de căutare (prefixIcon), un fundal alb și margini rotunjite. OutlineInputBorder definește conturul câmpului atât pentru starea activă (focusedBorder) cât și pentru cea normală (enabledBorder). Alături, Container are un fundal verde, colțuri rotunjite și conține o pictogramă de tip Icon(Icons.tune), reprezentând un buton de filtrare. Toate aceste elemente împreună creează o interfață clară și modernă pentru introducerea și filtrarea rețetelor.

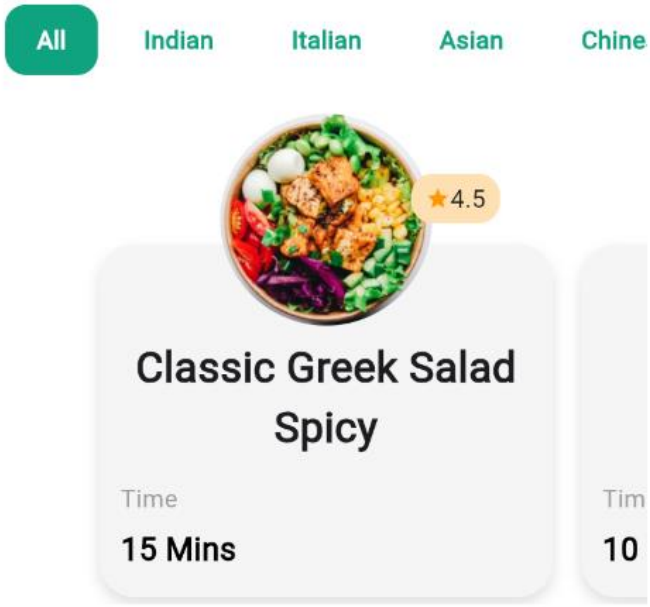
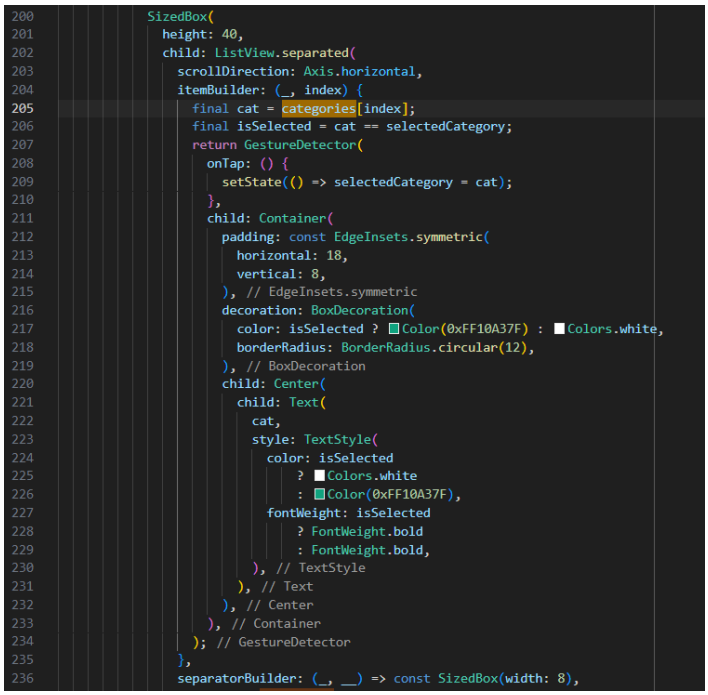


Figura 2 – Secțiunea cu meniul principal, cu butoane pentru categorii

```
final List<Map<String, dynamic>> recipes = [
  {
    'title': 'Classic Greek Salad Spicy',
    'time': '15 Mins',
    'rating': 4.5,
    'category': 'Italian',
    'image': 'assets/salad1.png',
    'author': "Klaus Schimmer",
    'authorImage': 'assets/author_image.png',
    'location': "Lagos, Nigeria",
    'ingredients': [
      {"Tomatos": "500g"},
      {"Cabbage": "300g"},
      {"Taco": "300g"},
      {"Slice Bread": "300g"},
    ],
  },
  {
    'title': 'Crunchy Nut Coleslaw',
    'time': '10 Mins',
    'rating': 3.5,
    'category': 'Asian',
    'image': 'assets/salad2.png',
    'author': "Carl Stein",
    'authorImage': 'assets/author_image.png',
    'location': "Lagos, Nigeria",
    'ingredients': [
      {"Tomatos": "500g"},
      {"Cabbage": "300g"},
      {"Taco": "300g"},
      {"Slice Bread": "300g"},
    ],
  },
  {
    'title': 'Steak with tomato sauce',
```

Figura 2.1 – Lista declarată cu toate elementele meniului (recete)



```
SizedBox(
  height: 40,
  child: ListView.separated(
    scrollDirection: Axis.horizontal,
    itemBuilder: (_, index) {
      final cat = categories[index];
      final isSelected = cat == selectedCategory;
      return GestureDetector(
        onTap: () {
          setState(() => selectedCategory = cat);
        },
        child: Container(
          padding: const EdgeInsets.symmetric(
            horizontal: 18,
            vertical: 8,
          ), // EdgeInsets.symmetric
          decoration: BoxDecoration(
            color: isSelected ? Color(0xFF10A37F) : Colors.white,
            borderRadius: BorderRadius.circular(12),
          ), // BoxDecoration
          child: Center(
            child: Text(
              cat,
              style: TextStyle(
                color: isSelected
                    ? Colors.white
                    : Color(0xFF10A37F),
                fontWeight: isSelected
                    ? FontWeight.bold
                    : FontWeight.bold,
              ), // TextStyle
            ), // Text
          ), // Center
        ), // Container
      ); // GestureDetector
    },
    separatorBuilder: (_, __) => const SizedBox(width: 8),
```

Figura 2.2 – Butoanele cu categorii

În acest fragment de cod este creat un rând de butoane orizontale care reprezintă categoriile de rețete, utilizând un ListView.separated. Acesta afișează o listă derulabilă

orizontal, unde fiecare element este separat de un spațiu de 8 pixeli, adăugat prin separatorBuilder. Containerul este inclus într-un SizedBox cu înălțimea de 40 de pixeli, pentru a controla dimensiunea întregului rând. În interiorul funcției itemBuilder, se obține fiecare categorie din lista categories, iar prin variabila isSelected se verifică dacă categoria curentă este cea selectată. Fiecare element este înfășurat într-un GestureDetector, care detectează atingerea utilizatorului și, prin metoda setState, actualizează categoria selectată. Vizual, fiecare categorie este reprezentată printr-un Container cu margini rotunjite și culori diferite în funcție de selecție: verde pentru cea activă și alb pentru celelalte. Textul din interiorul butonului își schimbă, de asemenea, culoarea în alb când este selectat și în verde când nu este.

```
243             SizedBox(
244               height: 280,
245               child: PageView.builder(
246                 controller: PageController(viewportFraction: 0.75),
247                 itemCount: filtered.length,
248                 itemBuilder: (_, index) {
249                   final recipe = filtered[index];
250                   return GestureDetector(
251                     onTap: () {
252                       Navigator.push(
253                         context,
254                         MaterialPageRoute(
255                           builder: (_) => RecipeDetailScreen(recipe: recipe),
256                         ), // MaterialPageRoute
257                       );
258                     },
259 
260                     child: AnimatedContainer(
261                       duration: const Duration(milliseconds: 300),
262                       margin: const EdgeInsets.symmetric(
263                         horizontal: 8,
264                         vertical: 6,
265                       ), // EdgeInsets.symmetric
266                       child: Stack(
267                         clipBehavior: Clip.none,
268                         children: [
269                           Positioned(
270                             top: 70,
271                             left: 0,
272                             right: 0,
273                             child: Container(
274                               width: 250,
275                               height: 200,
276                               decoration: BoxDecoration(
277                                 color: Colors.grey.shade100,
278                                 borderRadius: BorderRadius.circular(20),
279                                 boxShadow: [
```

Figura 2.3 – Containerul cu scroll pentru elementele meniului

În acest fragment de cod este creat un carusel interactiv de rețete folosind PageView.builder, care permite derularea orizontală a cardurilor. Acesta este plasat într-un SizedBox cu înălțimea de 280 pixeli pentru a controla spațiul ocupat pe ecran. Proprietatea viewportFraction: 0.75 face ca fiecare pagină să ocupe 75% din lățimea ecranului, oferind un efect vizual modern cu următoarea și precedenta rețetă parțial vizibile. Fiecare card este creat dinamic din lista filtered, care conține rețetele filtrate.
Fiecare element este înconjurat de un GestureDetector, care detectează atingerea utilizatorului și, la apăsare, deschide o pagină nouă — RecipeDetailScreen — cu detaliile rețetei selectate, prin intermediul Navigator.push.

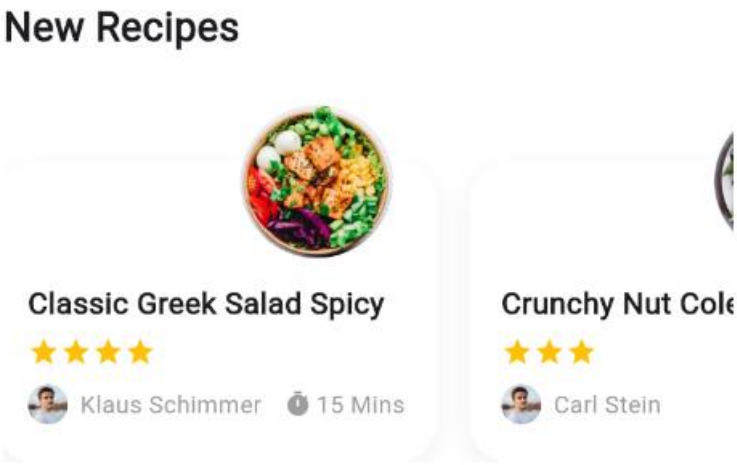


Figura 3 – Secțiunea “New Recipes”

```
380          const SizedBox(height: 30),
381          const Text(
382            "New Recipes",
383            style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
384          ), // Text
385          const SizedBox(height: 12),
386
387          SizedBox(
388            height: 190,
389            child: ListView.builder(
390              scrollDirection: Axis.horizontal,
391              itemCount: recipes.length,
392              itemBuilder: (_, index) {
393                final r = recipes[index];
394
395                return GestureDetector(
396                  onTap: () {
397                    Navigator.push(
398                      context,
399                      MaterialPageRoute(
400                        builder: (_) => RecipeDetailScreen(recipe: r),
401                      ), // MaterialPageRoute
402                    );
403                  },
404                  child: Stack(
405                    clipBehavior: Clip.none,
406                    children: [
407                      Container(
408                        width: 220,
409                        margin: const EdgeInsets.only(right: 16, top: 40),
410                        decoration: BoxDecoration(
411                          color: Colors.white,
412                          borderRadius: BorderRadius.circular(20),
413                          boxShadow: [
414                            BoxShadow(
415                              color: Colors.black.withOpacity(0.05),
416                              blurRadius: 10,
417                              offset: const Offset(0, 4),
418                            ), // BoxShadow
```

Figura 3.1 – Secțiunea “New Recipes” cu scroll container

În acest fragment de cod este construită o secțiune intitulată „New Recipes”, care afișează o listă orizontală de carduri cu rețete noi. Secțiunea începe cu un Text evidențiat printr-un stil îngroșat și o dimensiune a fontului de 20, urmat de un spațiu vertical (SizedBox(height: 12)) pentru separare vizuală. Lista de rețete este creată cu ajutorul unui ListView.builder cu derulare orizontală, ce generează dinamic fiecare element din lista recipes.

Fiecare card de rețetă este un GestureDetector, care permite navigarea către o pagină de detaliu (RecipeDetailScreen) atunci când utilizatorul atinge cardul. Cardul propriu-zis este realizat cu un Container alb, rotunjit la colțuri și cu umbră subtilă, oferind un aspect elegant.

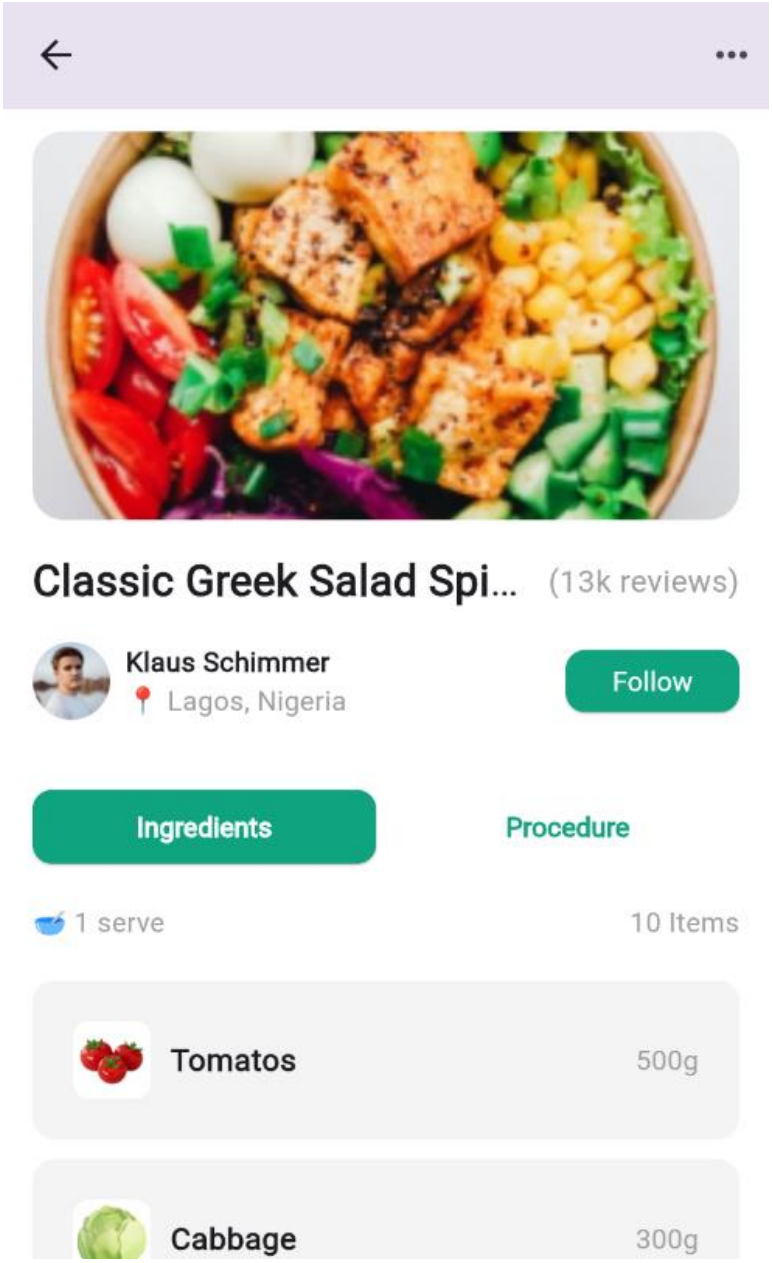
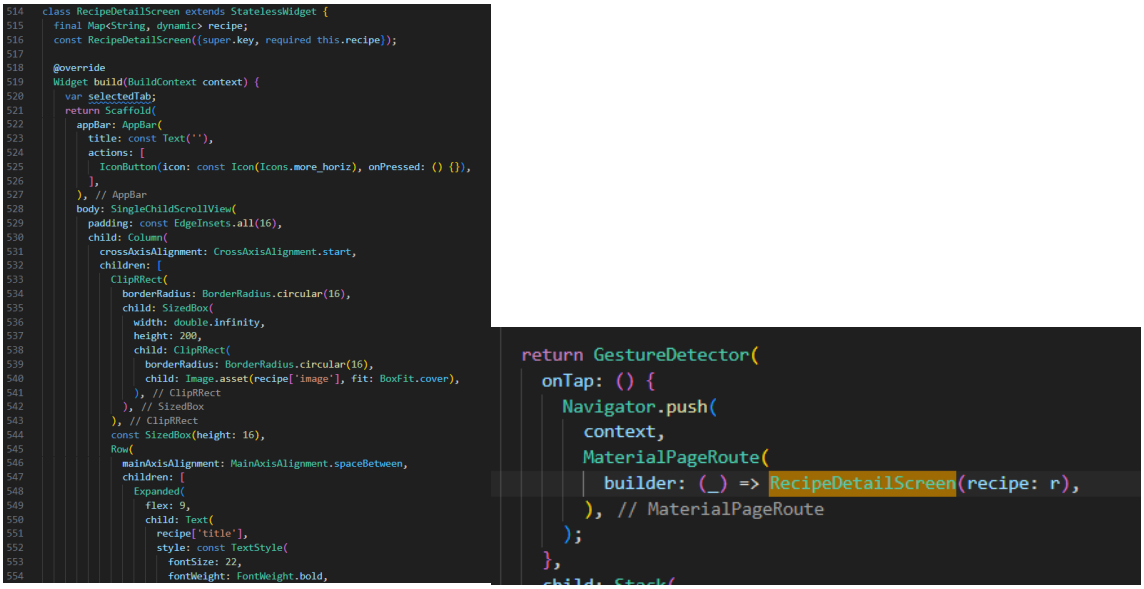


Figura 4 – Pagina cu detalii



```
class RecipeDetailScreen extends StatelessWidget {
  final Map<String, dynamic> recipe;
  const RecipeDetailScreen({super.key, required this.recipe});

  @override
  Widget build(BuildContext context) {
    var selectedTab;
    return Scaffold(
      appBar: AppBar(
        title: const Text(''),
        actions: [
          IconButton(icon: const Icon(Icons.more_horiz), onPressed: () {}),
        ],
      ), // AppBar
      body: SingleChildScrollView(
        padding: const EdgeInsets.all(16),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            ClipRRect(
              borderRadius: BorderRadius.circular(16),
              child: SizedBox(
                width: double.infinity,
                height: 200,
                child: ClipRRect(
                  borderRadius: BorderRadius.circular(16),
                  child: Image.asset(recipe['image'], fit: BoxFit.cover),
                ), // ClipRRect
              ), // SizedBox
            ), // ClipRRect
            const SizedBox(height: 16),
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                Expanded(
                  flex: 9,
                  child: Text(
                    recipe['title'],
                    style: const TextStyle(
                      fontSize: 22,
                      fontWeight: FontWeight.bold,
```

```
return GestureDetector(
  onTap: () {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (_) => RecipeDetailScreen(recipe: r),
      ), // MaterialPageRoute
    );
  },
```

Figura 4.1 – Pagina cu detalii și acțiunea OnTap

Acest fragment de cod Flutter definește ecranul detaliat al unei rețete în aplicație, prin intermediul clasei RecipeDetailScreen, care este un StatelessWidget și primește un Map<String, dynamic> cu datele rețetei. Structura principală este oferită de Scaffold, care conține AppBar cu un buton de acțiune (more_horiz) și un SingleChildScrollView pentru a permite derularea verticală a întregului conținut.

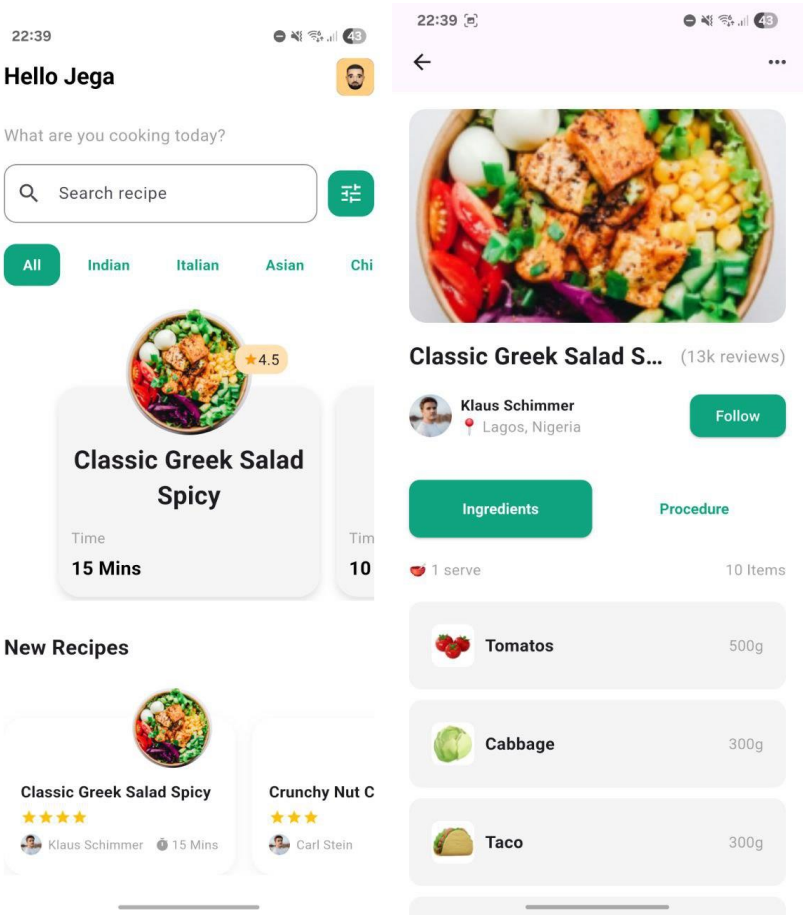


Figura 5 – Rezultatul final

Concluzie

Aplicația reușește să combine funcționalitatea și estetica într-un mod intuitiv și atractiv pentru utilizator. Prin utilizarea eficientă a widgeturilor precum Scaffold, AppBar, ListView, PageView, Container, Row, Column, TextField și ElevatedButton, interfața oferă o navigare ușoară între secțiuni, afișează informațiile rețetelor clar și permite interacțiuni dinamice, cum ar fi selectarea categoriilor, vizualizarea detaliilor rețetelor și derularea imaginilor într-un carusel. Designul responsiv, elementele vizuale suprapuse și butoanele interactive contribuie la o experiență plăcută și modernă, demonstrând modul în care Flutter facilitează dezvoltarea rapidă a aplicațiilor mobile complexe, cu o interfață estetică și funcționalitate completă.