MPL EXP-2

Ansh Sarfare D15A - 49

Aim: To Design Flutter UI by including common widgets

Theory:

Common Widgets used in Flutter app:

1. **Scaffold:** Provides the basic visual structure for a screen, including app bars, bodies, and bottom navigation.
2. **AppBar:** A widget that is displayed at the top of the screen.
3. **Text:** Displays an immutable piece of text.
4. **TextField:** Allows the user to enter text.
5. **Image.network:** Displays an image from a URL.
6. **SizedBox:** A box with a specified size.
7. **ListView.builder:** Creates a scrollable list of widgets that are built on demand.
8. **Card:** A panel with slightly rounded corners and a shadow.
9. **InkWell:** A rectangular area of a UI that responds to touch.
10. **Column:** Arranges its children in a vertical array.
11. **Row:** Arranges its children in a horizontal array.
12. **Padding:** Inserts space around another widget.
13. **Align:** Aligns its child within itself.
14. **Container:** A widget that combines common painting, positioning, and sizing widgets.
15. **CircularProgressIndicator:** A widget that indicates that a task is in progress.
16. **BottomNavigationBar:** A bar at the bottom of the screen for selecting different destinations.
17. **BottomNavigationBarItem:** An item in a bottom navigation bar.
18. **SingleChildScrollView:** A box in which a single widget can be scrolled.
19. **AnimatedOpacity:** Animates the opacity of a widget.
20. **Center:** Centers its child within itself.
21. **MaterialPageRoute:** A route that replaces the entire screen with a platform-adaptive transition.
22. **Navigator.push:** Method to push a route onto the navigator's stack.
23. **Icon:** A graphical icon widget.
24. **InputDecoration:** Styles the visual appearance of a TextField.
25. **OutlineInputBorder:** A border for TextField with a rectangular outlinE.

Code:

main.dart:

```
import 'package:flutter/material.dart';
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'widgets/recipe_detail_screen.dart';
import 'models/recipe.dart';
import 'dart:async';
import 'login_screen.dart';

void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Tasty',
      theme: ThemeData(
        scaffoldBackgroundColor: Colors.grey[900],
        appBarTheme: AppBarTheme(
          backgroundColor: Colors.grey[850],
          titleTextStyle: TextStyle(
            color: Colors.white,
            fontSize: 20,
            fontWeight: FontWeight.bold,
          ),
              textTheme: TextTheme(
          bodyMedium: TextStyle(color: Colors.white),
          titleLarge: TextStyle(
            color: Colors.white,
            fontWeight: FontWeight.bold,
          ),
        elevatedButtonTheme:
ElevatedButtonThemeData(
          style: ElevatedButton.styleFrom(
            backgroundColor: Colors.amber,
            foregroundColor: Colors.black,
          ),
        ),
      ),
      home: MyHomePage(),
    );
  }
}
class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() =>
_MyHomePageState();
}
class _MyHomePageState extends
State<MyHomePage> {
  List<Recipe> _recipes = [];
  String _searchTerm = '';
  Recipe? _randomRecipe;
  Timer? _timer;
  double _opacity = 1.0;
  Map<String, List<Recipe>> _areaRecipes = {};
  List<String> areas = [
    'Indian',
    'Canadian',
    'Italian',
    'Chinese',
    'Mexican',
    'Thai'
  ];
  bool _isLoadingAreaRecipes = true;
  int _selectedIndex = 0;
  @override
```

```
 void initState() {
   super.initState();
   fetchRandomRecipe();
   fetchAreaRecipes();
   _timer = Timer.periodic(Duration(seconds: 5),
(Timer timer) {
     setState(() {
       _opacity = 0.0;
     });
     Future.delayed(Duration(milliseconds: 500), () {
       fetchRandomRecipe();
       setState(() {
         _opacity = 1.0;
       });
     });
 @override
 void dispose() {
   _timer?.cancel();
   super.dispose();
 }
 Future<void> fetchRandomRecipe() async {
   final response = await
http.get(Uri.parse('https://www.themealdb.com/api/js
on/v1/1/random.php'));
   if (response.statusCode == 200) {
     final data = json.decode(response.body);
     if (data['meals'] != null) {
       setState(() {
         _randomRecipe =
Recipe.fromJson(data['meals'][0]);
       });
     }
   } else {
     print('Failed to load random recipe');
   }
 }
 Future<void> fetchAreaRecipes() async {
   setState(() {
     _isLoadingAreaRecipes = true;
   });
   Map<String, List<Recipe>> tempAreaRecipes =
{};
   for (String area in areas) {
     final response = await
http.get(Uri.parse('https://www.themealdb.com/api/js
on/v1/1/filter.php?a=$area'));
     if (response.statusCode == 200) {
       final data = json.decode(response.body);
       if (data['meals'] != null) {
         List<Recipe> recipes = (data['meals'] as
List).map((json) => Recipe.fromJson(json)).toList();
         tempAreaRecipes[area] = recipes;
       }
     } else {
       print('Failed to load $area recipes');
     }
   }
   setState(() {
     _areaRecipes = tempAreaRecipes;
     _isLoadingAreaRecipes = false;
   });
 }
 Future<void> fetchRecipes() async {
   if (_searchTerm.isEmpty) {
     setState(() {
       _recipes = [];
     });
     return;
   }

   final response = await http.get(Uri.parse(

'https://www.themealdb.com/api/json/v1/1/search.php
?s=$_searchTerm'));
```

```
    if (response.statusCode == 200) {
      final data = json.decode(response.body);
      if (data['meals'] != null) {
        setState(() {
          _recipes = (data['meals'] as List)
              .map((json) => Recipe.fromJson(json))
              .toList();
        });
      } else {
        setState(() {
          _recipes = [];
        });
      }
    } else {
      throw Exception('Failed to load recipes');
    }
  }
  List<Recipe> get filteredRecipes {
    return _recipes.where((recipe) =>

recipe.strMeal.toLowerCase().contains(_searchTerm.t
oLowerCase()) ||
        recipe.ingredients.any((ingredient) =>

ingredient.name.toLowerCase().contains(_searchTer
m.toLowerCase()))).toList();
  }
  Future<Recipe?> fetchRecipeDetails(String
recipeName) async {
    final response = await
http.get(Uri.parse('https://www.themealdb.com/api/js
on/v1/1/search.php?s=$recipeName'));
    if (response.statusCode == 200) {
      final data = json.decode(response.body);
      if (data['meals'] != null &&
data['meals'].isNotEmpty) {
        return Recipe.fromJson(data['meals'][0]);
      }
    } else {
      print('Failed to load recipe details for
$recipeName');
    }
    return null;
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Row(
          children: [
            Padding(
              padding: const EdgeInsets.only(right: 8.0),
              child: Image.asset(
                'assets/tasty_logo.png', // Path to your asset
                width: 30, // Adjust size as needed
                height: 30, // Adjust size as needed
              ),
            ),
            Text(
              'Tasty',
            ),
          ],
        ),
        centerTitle: false, // Align title to the left
      ),
      body: Column(
        children: [
          Padding(
            padding: const EdgeInsets.all(16.0),
            child: TextField(
              style: TextStyle(color: Colors.white),
              decoration: InputDecoration(
                hintText: 'Search recipes...',
        hintStyle: TextStyle(color:
Colors.grey[500]),
        fillColor: Colors.grey[800],
        filled: true,
        border: OutlineInputBorder(
         borderRadius:
BorderRadius.circular(30.0),
         borderSide: BorderSide.none,
        ),
        prefixIcon: Icon(Icons.search, color:
Colors.white),
       ),
       onChanged: (value) {
        setState(() {
         _searchTerm = value;
         fetchRecipes();
        });
     ),
     Expanded(
      child: _searchTerm.isEmpty
        ? SingleChildScrollView(
       child: Column(
        crossAxisAlignment:
CrossAxisAlignment.start,
        children: [
         AnimatedOpacity(
          opacity: _opacity,
          duration: Duration(milliseconds: 500),
          child: SizedBox(
           height: 300,
           child: _randomRecipe != null
             ? InkWell(
            onTap: () {
             Navigator.push(
              context,
              MaterialPageRoute(
               builder: (context) =>
RecipeDetailScreen(recipe: _randomRecipe!),
              ),
             );
            },
            child: Card(
             color: Colors.grey[800],
             child: Column(
              crossAxisAlignment:
CrossAxisAlignment.stretch,
              children: [
               Expanded(
                child: Image.network(
                 _randomRecipe!.strMealThumb,
                 fit: BoxFit.cover,
                 errorBuilder: (context, error,
stackTrace) {
                  return Container(
                   color: Colors.grey[300],
                   child: Center(
                    child:
Icon(Icons.error_outline),
                   ),
               ),
               Padding(
                padding: const
EdgeInsets.all(8.0),
                child: Align(
                 alignment: Alignment.center,
                 child: Text(
                  _randomRecipe!.strMeal,
                  style: TextStyle(
                   color: Colors.white,
                   fontSize: 18,
                   fontWeight: FontWeight.bold,
                  ),
               )
            : Center(child:
CircularProgressIndicator()),
          ),
        ),
        SizedBox(height: 20),
        if (!_isLoadingAreaRecipes)
          ...areas.map((area) {
            return Column(
              crossAxisAlignment:
CrossAxisAlignment.start,
              children: [
                Padding(
                  padding: const
EdgeInsets.symmetric(vertical: 8.0),
                  child: Text(
                    area,
                    style: TextStyle(
                      color: Colors.white,
                      fontSize: 20,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ),
                SizedBox(
                  height: 250,
                  child: ListView.builder(
                    scrollDirection: Axis.horizontal,
                    itemCount:
_areaRecipes[area]?.length ?? 0,
                    itemBuilder: (context, index) {
                      final recipe =
_areaRecipes[area]![index];
                      return InkWell(
                        onTap: () async {
                          Recipe? detailedRecipe =
await fetchRecipeDetails(recipe.strMeal);
                          if (detailedRecipe != null) {
                            Navigator.push(
                              context,
                              MaterialPageRoute(
                                builder: (context) =>
RecipeDetailScreen(recipe: detailedRecipe),
                              ),
                            );
                          } else {
                            // Handle the case where
detailed recipe is not found
                            print("Detailed recipe not
found");
                          }
                        },
                        child: Container(
                          width: 200,
                          margin: EdgeInsets.only(right:
10),
                          child: Column(
                            crossAxisAlignment:
CrossAxisAlignment.stretch,
                            children: [
                              Expanded(
                                child: ClipRRect(
                                  borderRadius:
BorderRadius.circular(10),
                                  child: Image.network(
                                    recipe.strMealThumb,
                                    fit: BoxFit.cover,
                                    errorBuilder: (context,
error, stackTrace) {
                                      return Container(
                                        color:
Colors.grey[300],
                                        child: Center(
                                          child:
Icon(Icons.error_outline),
                              ),
                              Padding(
                                padding: const
EdgeInsets.all(8.0),
                                child: Text(
                                  recipe.strMeal,
                                  style: TextStyle(
                                    color: Colors.white,
                                    fontWeight:
FontWeight.bold,
                                  ),
    textAlign: TextAlign.center,
                      ),
                  );
                }).toList()
              else
                Center(child:
CircularProgressIndicator()),
            ],
          ),
        )
          : ListView.builder(
          itemCount: filteredRecipes.length,
          itemBuilder: (context, index) {
            final recipe = filteredRecipes[index];
            return Card(
              color: Colors.grey[800],
              child: InkWell(
                onTap: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) =>
RecipeDetailScreen(recipe: recipe),
                    ),
                  );
                },
```

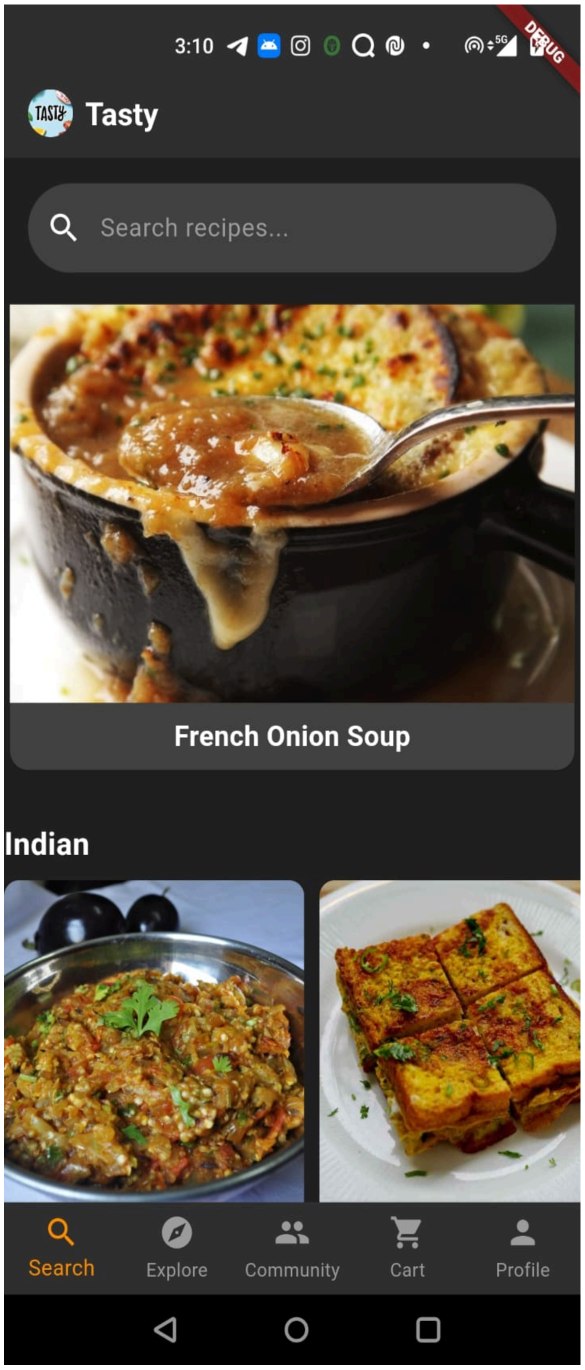
```
              child: Padding(
                padding: const EdgeInsets.all(8.0),
                child: Column(
                  children: [
                    Image.network(
                      recipe.strMealThumb,
                      width: double.infinity,
                      height: 250,
                      fit: BoxFit.cover,
                      errorBuilder: (context, error,
stackTrace) {
                        return Container(
                          width: double.infinity,
                          height: 250,
                          color: Colors.grey[300],
                          child: Center(
                            child: Icon(Icons.error_outline),

                    ),
                    SizedBox(height: 8),
                    Align(
                      alignment: Alignment.center,
                      child: Text(
                        recipe.strMeal,
                        style: TextStyle(
                            color: Colors.white,
                            fontSize: 18,
                            fontWeight: FontWeight.bold
                        ),
      ),
      bottomNavigationBar: BottomNavigationBar(
        items: const <BottomNavigationBarItem>[
          BottomNavigationBarItem(
            icon: Icon(Icons.search),
            label: 'Search',
          ),
          BottomNavigationBarItem(
          icon: Icon(Icons.explore),
          label: 'Explore',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.people),
          label: 'Community',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.shopping_cart),
          label: 'Cart',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.person),
          label: 'Profile',
        ),
      ],
      currentIndex: _selectedIndex,
      selectedItemColor: Colors.amber[800],
      unselectedItemColor: Colors.grey[500],
      backgroundColor: Colors.grey[850],
      type: BottomNavigationBarType.fixed,
      onTap: (index) {
        setState(() {
          _selectedIndex = index;
          if (index == 4) {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) =>
LoginScreen()),
            );
          } else {
            // Handle navigation for other tabs
          }
}
```

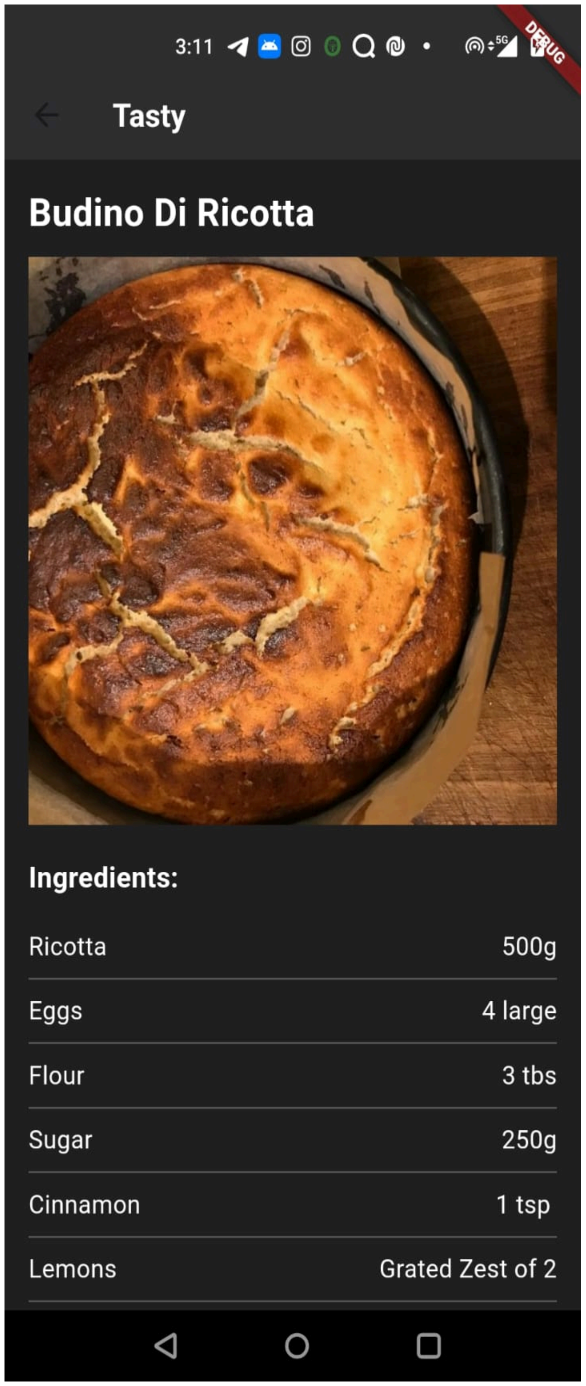
recipe_detail_screen:

```
import 'package:flutter/material.dart';
import '../models/recipe.dart';
import 'package:url_launcher/url_launcher.dart';

class RecipeDetailScreen extends StatelessWidget {
  final Recipe recipe;

  RecipeDetailScreen({required this.recipe});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Tasty'),
      ),
      body: SingleChildScrollView(
        padding: EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment:
CrossAxisAlignment.start,
          children: [
            Text(
              recipe.strMeal,
              style: TextStyle(fontSize: 24, fontWeight:
FontWeight.bold),
            ),
            SizedBox(height: 10),
            Image.network(
              recipe.strMealThumb,
              width: double.infinity,
              fit: BoxFit.cover,
              errorBuilder: (context, error, stackTrace) {
                return Container(
                  height: 200,
                  width: double.infinity,
                  color: Colors.grey[300],
                  child: Center(
                    child: Icon(Icons.error_outline),
            ),
            SizedBox(height: 20),
            Text(
              'Ingredients:',
              style: TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
            ),
            SizedBox(height: 10),
            ListView.builder(
              shrinkWrap: true,
              physics: NeverScrollableScrollPhysics(),
              itemCount: recipe.ingredients.length,
              itemBuilder: (context, index) {
                final ingredient = recipe.ingredients[index];
                return Column(
                  children: [
                    Padding(
                      padding: const
EdgeInsets.symmetric(vertical: 8.0),
                      child: Row(
                        mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                        children: [
                          Text(
                            ingredient.name,
                            style: TextStyle(fontSize: 16),
                          ),
                          Text(
                            ingredient.measure,
                            style: TextStyle(fontSize: 16)
                      ),
                    ),
                    Divider(
                      color: Colors.grey[700], // Dull grey
line
                      height: 1,
```

```
            ),
            SizedBox(height: 20),
            Text(
              'Instructions:',
              style: TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
            ),
            SizedBox(height: 10),
            Text(
              recipe.strInstructions,
              style: TextStyle(fontSize: 16),
            ),
            SizedBox(height: 20),
            if (recipe.strYoutube != null &&
recipe.strYoutube!.isNotEmpty)
              ElevatedButton(
                onPressed: () async {
                  final Uri url =
Uri.parse(recipe.strYoutube!);
                  if (await canLaunchUrl(url)) {
                    await launchUrl(url);
                  } else {

ScaffoldMessenger.of(context).showSnackBar(
                      SnackBar(content: Text('Could not
launch ${recipe.strYoutube}')),
                    );
                  }
                },
                child: Text('Watch on YouTube'),
              ),
          ],
        )
```

Output:

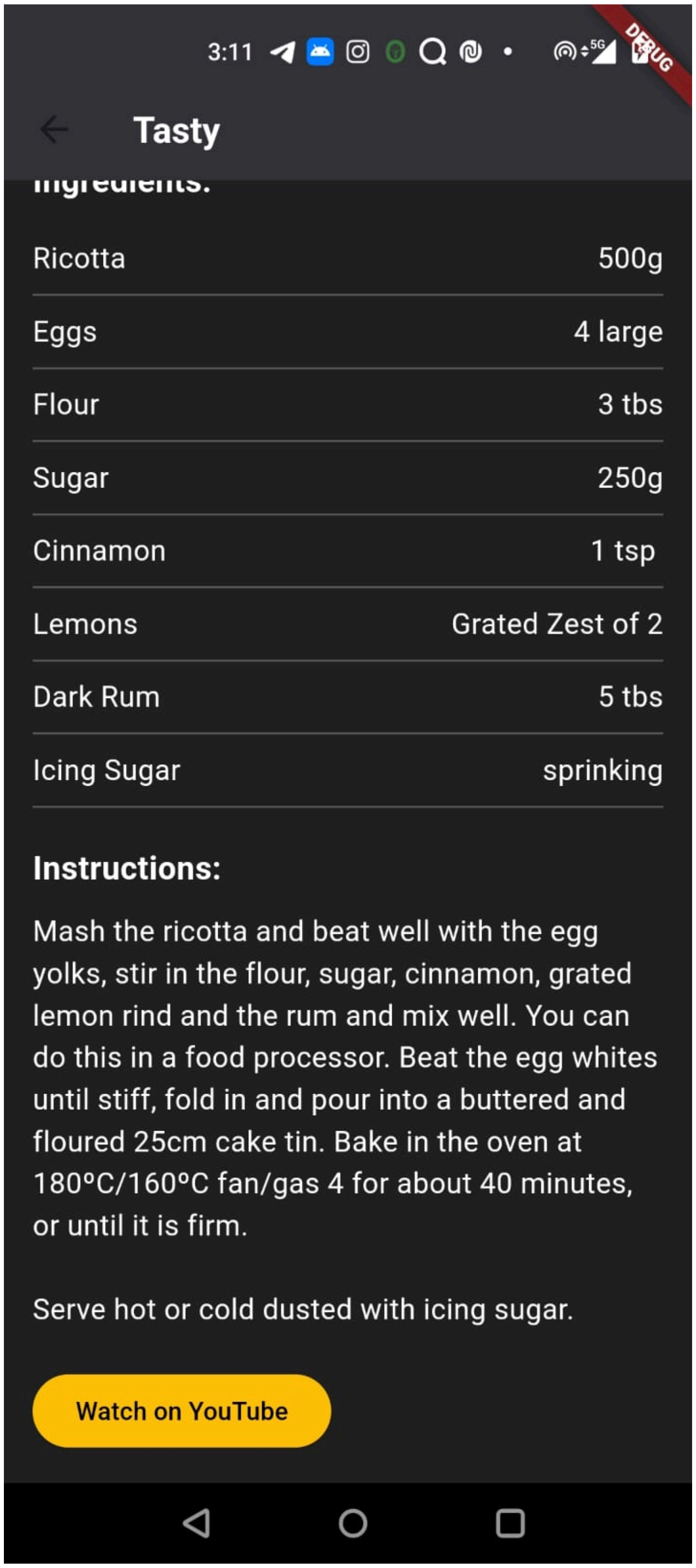



Tasty

Budino Di Ricotta

Ingredients:

Ricotta	500g
Eggs	4 large
Flour	3 tbs
Sugar	250g
Cinnamon	1 tsp
Lemons	Grated Zest of 2



Tasty

Ingredients:

Ricotta	500g
Eggs	4 large
Flour	3 tbs
Sugar	250g
Cinnamon	1 tsp
Lemons	Grated Zest of 2
Dark Rum	5 tbs
Icing Sugar	sprinking

Instructions:

Mash the ricotta and beat well with the egg yolks, stir in the flour, sugar, cinnamon, grated lemon rind and the rum and mix well. You can do this in a food processor. Beat the egg whites until stiff, fold in and pour into a buttered and floured 25cm cake tin. Bake in the oven at 180°C/160°C fan/gas 4 for about 40 minutes, or until it is firm.

Serve hot or cold dusted with icing sugar.