Automatic Number Conversion System

Program Documentation & Project Documentation

Technology: Flutter (Dart)  
Target Platforms: Android & iOS

GROUP MEMBERS

MOSES WANDURI-CT100/G/21575/24

BRIAN WARUKARI-CT100/G/22327/24

ANGELA WANJERI-CT100/G/22315/24

JOY WAIRIMU-CT100/G/21997/24

JOY ANN NJOKI-CT100/G/21919/24

# **Program Documentation**

## **1. Introduction**

This program converts numbers between various numeral systems automatically (e.g., Decimal ↔ Binary ↔ Octal ↔ Hexadecimal). The system is built using Flutter to ensure cross-platform compatibility.

## **2. Program Structure**

- main.dart (number\_system.dart): Entry point of the application and contains UI + logic.  
- ui/: Contains user interface screens.  
- widgets/: Custom reusable widgets.  
- services/conversion\_service.dart: Contains logic for number conversions.  
- models/: Defines number input and output structures.

## **3. Key Functions**

convertToBinary, convertToOctal, convertToHexadecimal, convertFromBinary, convertFromOctal, convertFromHexadecimal.  
Main conversion logic is implemented in \_convertNumber().

## **4. User Interface Flow**

1. User inputs a number.  
2. Selects the base to convert from.  
3. Selects the base to convert to.  
4. Presses 'Convert' button.  
5. Result is displayed on screen.

## **5. Main Code File (number\_system.dart)**

import 'package:flutter/material.dart';  
  
void main() {  
 runApp(const MyApp());  
}  
  
class MyApp extends StatelessWidget {  
 const MyApp({Key? key}) : super(key: key);  
  
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 title: 'Number Converter',  
 theme: ThemeData(  
 primarySwatch: Colors.deepPurple,  
 fontFamily: 'Inter',  
 useMaterial3: true,  
 ),  
 home: const NumberConverterScreen(),  
 );  
 }  
}  
  
class NumberConverterScreen extends StatefulWidget {  
 const NumberConverterScreen({Key? key}) : super(key: key);  
  
 @override  
 \_NumberConverterScreenState createState() => \_NumberConverterScreenState();  
}  
  
class \_NumberConverterScreenState extends State<NumberConverterScreen> {  
 final TextEditingController \_inputController = TextEditingController();  
 String \_fromBase = '10';  
 String \_toBase = '2';  
 String \_result = '';  
 String \_error = '';  
  
 final List<Map<String, String>> \_bases = [  
 {'value': '10', 'label': 'Decimal (Base 10)'},  
 {'value': '2', 'label': 'Binary (Base 2)'},  
 {'value': '8', 'label': 'Octal (Base 8)'},  
 {'value': '16', 'label': 'Hexadecimal (Base 16)'},  
 ];  
  
 void \_handleConversion() {  
 setState(() {  
 \_error = '';  
 \_result = '';  
 });  
  
 if (\_inputController.text.trim().isEmpty) {  
 setState(() {  
 \_error = 'Please enter a number.';  
 });  
 return;  
 }  
  
 try {  
 int? decimalValue = int.tryParse(\_inputController.text, radix: int.parse(\_fromBase));  
  
 if (decimalValue == null) {  
 setState(() {  
 \_error = 'Invalid input for base $\_fromBase.';  
 });  
 return;  
 }  
  
 String convertedValue = decimalValue.toRadixString(int.parse(\_toBase));  
  
 setState(() {  
 \_result = convertedValue.toUpperCase();  
 });  
 } catch (e) {  
 setState(() {  
 \_error = 'Conversion failed. Please check your input and bases.';  
 });  
 }  
 }  
  
 void \_handleReset() {  
 setState(() {  
 \_inputController.clear();  
 \_fromBase = '10';  
 \_toBase = '2';  
 \_result = '';  
 \_error = '';  
 });  
 }  
  
 @override  
 void initState() {  
 super.initState();  
 \_inputController.addListener(\_handleConversion);  
 }  
  
 @override  
 void dispose() {  
 \_inputController.removeListener(\_handleConversion);  
 \_inputController.dispose();  
 super.dispose();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: const Text('Number Converter', style: TextStyle(color: Colors.white)),  
 backgroundColor: Colors.deepPurple,  
 ),  
 body: SingleChildScrollView(  
 padding: const EdgeInsets.all(24.0),  
 child: Center(  
 child: ConstrainedBox(  
 constraints: const BoxConstraints(maxWidth: 400),  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.stretch,  
 children: [  
 const Text(  
 'Convert numbers between decimal, binary, octal, and hexadecimal.',  
 textAlign: TextAlign.center,  
 style: TextStyle(fontSize: 16, color: Colors.black54),  
 ),  
 const SizedBox(height: 24),  
 TextField(  
 controller: \_inputController,  
 decoration: const InputDecoration(  
 labelText: 'Enter Number',  
 border: OutlineInputBorder(  
 borderRadius: BorderRadius.all(Radius.circular(12.0)),  
 ),  
 ),  
 keyboardType: TextInputType.text,  
 ),  
 const SizedBox(height: 16),  
 Row(  
 children: [  
 Expanded(  
 child: DropdownButtonFormField<String>(  
 value: \_fromBase,  
 onChanged: (String? newValue) {  
 setState(() {  
 \_fromBase = newValue!;  
 \_handleConversion();  
 });  
 },  
 items: \_bases.map<DropdownMenuItem<String>>((Map<String, String> base) {  
 return DropdownMenuItem<String>(  
 value: base['value'],  
 child: Text(base['label']!),  
 );  
 }).toList(),  
 decoration: const InputDecoration(  
 labelText: 'From Base',  
 border: OutlineInputBorder(  
 borderRadius: BorderRadius.all(Radius.circular(12.0)),  
 ),  
 ),  
 ),  
 ),  
 const SizedBox(width: 16),  
 Expanded(  
 child: DropdownButtonFormField<String>(  
 value: \_toBase,  
 onChanged: (String? newValue) {  
 setState(() {  
 \_toBase = newValue!;  
 \_handleConversion();  
 });  
 },  
 items: \_bases.map<DropdownMenuItem<String>>((Map<String, String> base) {  
 return DropdownMenuItem<String>(  
 value: base['value'],  
 child: Text(base['label']!),  
 );  
 }).toList(),  
 decoration: const InputDecoration(  
 labelText: 'To Base',  
 border: OutlineInputBorder(  
 borderRadius: BorderRadius.all(Radius.circular(12.0)),  
 ),  
 ),  
 ),  
 ),  
 ],  
 ),  
 const SizedBox(height: 24),  
 if (\_error.isNotEmpty)  
 Container(  
 padding: const EdgeInsets.all(16),  
 decoration: BoxDecoration(  
 color: Colors.red.shade50,  
 borderRadius: BorderRadius.circular(12),  
 border: Border.all(color: Colors.red.shade200),  
 ),  
 child: Text(  
 \_error,  
 textAlign: TextAlign.center,  
 style: TextStyle(color: Colors.red.shade700, fontWeight: FontWeight.w500),  
 ),  
 ),  
 if (\_error.isEmpty)  
 Container(  
 padding: const EdgeInsets.all(16),  
 decoration: BoxDecoration(  
 color: Colors.deepPurple.shade50,  
 borderRadius: BorderRadius.circular(12),  
 border: Border.all(color: Colors.deepPurple.shade200),  
 ),  
 child: Column(  
 children: [  
 Text(  
 'Converted Value:',  
 style: TextStyle(color: Colors.deepPurple.shade600, fontSize: 14),  
 ),  
 const SizedBox(height: 8),  
 Text(  
 \_result.isNotEmpty ? \_result : '...',  
 style: const TextStyle(fontSize: 28, fontWeight: FontWeight.bold),  
 textAlign: TextAlign.center,  
 ),  
 ],  
 ),  
 ),  
 const SizedBox(height: 16),  
 ElevatedButton(  
 onPressed: \_handleReset,  
 style: ElevatedButton.styleFrom(  
 backgroundColor: Colors.grey.shade200,  
 foregroundColor: Colors.black87,  
 padding: const EdgeInsets.symmetric(vertical: 16),  
 shape: RoundedRectangleBorder(  
 borderRadius: BorderRadius.circular(24.0),  
 ),  
 ),  
 child: const Text('Reset'),  
 ),  
 ],  
 ),  
 ),  
 ),  
 ),  
 );  
 }  
}

## **6. Error Handling**

Invalid input for the selected base is caught with try-catch. Error messages are displayed in the UI.

## **7. Testing**

Unit tests validate conversion methods. UI testing validates inputs and dropdowns.

## **8. Deployment**

Deployed on Android (APK/AAB) and iOS (IPA). Requires Flutter SDK 3.0+, Dart 2.18+.

# **Project Documentation**

## **1. Project Overview**

The Automatic Number Conversion System assists students, developers, and engineers in converting numbers across numeral systems. It eliminates manual calculations by providing instant and accurate conversions.

## **2. Objectives**

- Build a cross-platform mobile app for number conversion.  
- Support Decimal, Binary, Octal, and Hexadecimal.  
- Provide a user-friendly and interactive UI.

## **3. Scope**

Includes input validation, real-time conversion, copy-to-clipboard, and responsive UI.

## **4. Functional Requirements**

FR1: Accept user input in any numeral system.  
FR2: Automatically convert to all other numeral systems.  
FR3: Display results clearly.  
FR4: Handle invalid input gracefully.

## **5. Non-Functional Requirements**

Performance, usability, compatibility (Android 8+, iOS 13+), reliability.

## **6. System Architecture**

Frontend: Flutter Widgets (Material UI).  
Backend: Dart logic in number\_system.dart.  
Data Layer: Stateless.

## **7. Tools & Technologies**

Flutter 3.0+, Dart 2.18+, Android Studio / VS Code, Git & GitHub.

## **8. Code Reference**

The main file is number\_system.dart (see Program Documentation).

## **9. Project Deliverables**

Fully functional Flutter app (APK & IPA), source code, documentation, test reports.

## **10. Future Enhancements**

Roman numerals, dark mode, web support, voice input.

## **11. Conclusion**

The system simplifies number conversions, improving learning efficiency and ensuring accessibility via Flutter’s cross-platform support.