VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

ADVANCED PYTHON PROGRAMMING

LAB ASSESMENT 21

FLUTTER APP DEVELOPMENT

NAME: M.GOKULESH.

REG NO: 22MIC0102.

EXPENSE TRACKER APP USING FLUTTER

MAJOR TAKEAWAY: Create Read Update Delete [CRUD Operations] experimentation in Flutter.

main.dart

```
import 'package:flutter/material.dart';
import 'package:project_23/pages/about_application_page.dart';
import 'package:project_23/pages/application_settings_page.dart';
import 'package:project_23/pages/balance_sheet_page.dart';
import 'package:project_23/pages/loading_page.dart';
import 'package:flutter/services.dart';
import 'package:project_23/pages/statistics_page.dart';
import 'package:project_23/pages/transactions_history_page.dart';
import 'package:project_23/primary.dart';
void main() async{
  WidgetsFlutterBinding.ensureInitialized();
  runApp(const ExpenseTracker());
}
class ExpenseTracker extends StatefulWidget{
  const ExpenseTracker({super.key});
```

```
  @override
  State<ExpenseTracker> createState() => _ExpenseTrackerState();
}
class _ExpenseTrackerState extends State<ExpenseTracker> {
  _ExpenseTrackerState();
  @override
  Widget build(BuildContext context){
    (){SystemChrome.setEnabledSystemUIMode(SystemUiMode.edgeToEdge);
      SystemChrome.setSystemUIOverlayStyle(
        const SystemUiOverlayStyle(
            statusBarColor: Colors.transparent,
            systemNavigationBarColor: Colors.transparent,
            systemNavigationBarDividerColor: Colors.transparent));}();
    return MaterialApp(
      routes:{
        '/':(context)=>const LoadingPage(),
        '/expense_tracker':(context)=>const ExpenseTracker(),
        '/primary_page':(context)=>const Primary(),
        '/app_information_page':(context)=>const AboutApplication(),
        '/app_settings_page':(context)=>const ApplicationSettings(),
        '/income_page':(context)=>const BalanceSheetPage(index:0),
        '/balance_sheet_funds_flow_page':(context)=>const BalanceSheetPage(),
        '/expense_page':(context)=>const BalanceSheetPage(index:2),
        '/stats_settings_page':(context)=>const StatisticsPage(index:0),
        '/statistics_pie_page':(context)=>const StatisticsPage(),
        '/statistics_graph_page':(context)=>const StatisticsPage(index:2),
        '/transaction_history_page':(context)=>const TransactionsHistory(),
      },
      debugShowCheckedModeBanner: false,
    );
  }
}
```

primary.dart

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:project_23/pages/home_page.dart';
import 'package:project_23/pages/balance_sheet_page.dart';
import 'package:project_23/pages/individual_transaction_page.dart';
import 'package:project_23/pages/loans_given_page.dart';
import 'package:project_23/pages/loans_taken_page.dart';
import 'package:project_23/pages/statistics_page.dart';
import 'package:project_23/pages/transactions_history_page.dart';
void main() async{
  WidgetsFlutterBinding.ensureInitialized();
  runApp(const Primary());
}
```

```
class Primary extends StatefulWidget{
  const Primary({super.key});
  @override
  State<Primary> createState() => _PrimaryState();
}
class _PrimaryState extends State<Primary> {
  final GlobalKey<ScaffoldState> _scaffoldKey=GlobalKey<ScaffoldState>();
  _PrimaryState();
  int pageIndex=1;
  List<StatefulWidget> pages=const
[BalanceSheetPage(),HomePage(),StatisticsPage()];
  PageController pageController=PageController(initialPage:1);
  @override
  Widget build(BuildContext context){
    return MaterialApp(
      home:Container(
        color:Colors.green,
        child: Scaffold(
          key: _scaffoldKey,
          drawer: Drawer(
            child:Container(
              padding:const EdgeInsets.fromLTRB(2.0,10.0,0.0,0.0),
              child: ListView(
                children: <Widget>[
                  DrawerHeader(
                    decoration: const BoxDecoration(
                      image: DecorationImage(image:
AssetImage('assets/images/loading_screen_image.png'),
                      fit: BoxFit.fill,opacity: 0.35)
                    ),
                    child: Column(
                        mainAxisAlignment: MainAxisAlignment.center,
                        children: [
                          Text('Expense Elixir',
                            style:GoogleFonts.montserrat(
                              fontSize:30.0,
                              fontWeight:FontWeight.w300,
                              color:Colors.black,
                            )),
                          const SizedBox(height:5.0),
                          Text('Elixir For Your Expense Enigma',
                              style:GoogleFonts.montserrat(
                                fontSize:15.0,
                                fontWeight:FontWeight.w300,
                                color:Colors.black,
                              )),
                        ],
                      ),
```

```
                ),
                ListTile(
                    leading:const
ImageIcon(AssetImage('assets/icons/income_icon.png')),
                    title:const Text('Income'),
                    onTap:()=>Navigator.pushNamed(context,'/income_page'),
                ),
                ListTile(
                    leading:const
ImageIcon(AssetImage('assets/icons/balance_sheet_page_icon.png')),
                    title:const Text('Funds Flow'),
                    onTap:()=>Navigator.pushNamed(context,'/balance_sheet_fu
nds_flow_page'),
                ),
                ListTile(
                    leading:const
ImageIcon(AssetImage('assets/icons/expense_icon.png')),
                    title:const Text('Expense'),
                    onTap:()=>Navigator.pushNamed(context,'/expense_page'),
                ),
                ListTile(
                    leading:const
ImageIcon(AssetImage('assets/icons/settings_icon.png')),
                    title:const Text('Stats Settings'),
                    onTap:()=>Navigator.pushNamed(context,'/stats_settings_p
age'),
                ),
                ListTile(
                    leading:const
ImageIcon(AssetImage('assets/icons/pie_page_icon.png')),
                    title:const Text('Pie Chart'),
                    onTap:()=>Navigator.pushNamed(context,'/statistics_pie_p
age'),
                ),
                ListTile(
                    leading:const
ImageIcon(AssetImage('assets/icons/graph_page_icon.png')),
                    title:const Text('Graph Chart'),
                    onTap:()=>Navigator.pushNamed(context,'/statistics_graph
_page'),
                ),
                ListTile(
                  leading: const
ImageIcon(AssetImage('assets/icons/transaction_history_icon.png')),
                  title: const Text('Transaction History'),
                  onTap:()=>Navigator.pushNamed(context,'/transaction_histor
y_page')
```

```
                ),
                const Divider(
                  thickness: 2.5,
                  color: Colors.black38,
                ),
                ListTile(
                  leading:const
ImageIcon(AssetImage('assets/icons/info_icon.png')),
                  title:const Text('About App'),
                  onTap:()=>Navigator.pushNamed(context,'/app_information_pa
ge')
                ),
                ListTile(
                  leading:const
ImageIcon(AssetImage('assets/icons/settings_icon.png')),
                  title:const Text('App Settings'),
                  onTap:()=>Navigator.pushNamed(context,'/app_settings_page'
),
                )
              ],
            ),
          ),
        ),
        backgroundColor: Colors.transparent,
        body:Stack(
          children: <Widget>[PageView(
            scrollDirection:Axis.vertical,
            onPageChanged: (index){
              setState(()=>pageIndex=index);
            },
            controller:pageController,
            children: pages,
          ),
            Positioned(
              left: 10,
              top: 41,
              child: IconButton(
              icon: const Icon(Icons.menu),
              onPressed: () => _scaffoldKey.currentState!.openDrawer()
              )
            ),
          ],
        ),
        bottomNavigationBar: Container(
          padding:const EdgeInsets.fromLTRB(0.0, 0.0, 0.0,0.0),
          decoration:const BoxDecoration(
            border: Border(top: BorderSide(color: Colors.black38),),
          ),
```

```
            child: BottomNavigationBar(
              elevation: 0,
              items: const<BottomNavigationBarItem>[BottomNavigationBarItem(
                icon:
ImageIcon(AssetImage('assets/icons/balance_sheet_page_icon.png')),
                label:'Balance Sheet',
              ),
                BottomNavigationBarItem(
                  icon:
ImageIcon(AssetImage('assets/icons/home_page_icon.png')),
                  label:'Home',
                ),
                BottomNavigationBarItem(
                  icon:
ImageIcon(AssetImage('assets/icons/statistics_page_icon.png')),
                  label:'Statistics',
                ),
              ],
              currentIndex:pageIndex,
              selectedItemColor: Colors.white,
              backgroundColor: Colors.green,
              onTap: (index){
                setState(()=>pageIndex=index);
                pageController.animateToPage(pageIndex,duration:
                const Duration(milliseconds:200,),curve:Curves.linear);
              },
            ),
          ),
        ),
      ),
      debugShowCheckedModeBanner: false,
      routes:{
        '/expense_page':(context)=>const BalanceSheetPage(index:2),
        '/transactions_history_page':(context)=>const TransactionsHistory(),
        '/individual_transaction_page':(context)=>const
IndividualTransactionPage(),
        '/loans_taken_page':(context)=>const LoansTakenPage(),
        '/loans_given_page':(context)=>const LoansGivenPage(),
      }
    );
  }
}
```

budget_model.dart

```
```

```dart
class BudgetModel{
  late double budget;
  late String bDTime;
  late String mode;
  BudgetModel({required this.budget,required this.bDTime,required this.mode});
  Map<String,dynamic> toMap(){
    Map<String,String> map={};
    map['budget']=budget.toString();
    map['bDTime']=bDTime;
    map['mode']=mode;
    return map;
  }
  BudgetModel.fromMap(Map<String,dynamic> map){
    budget=map['budget'];
    bDTime=map['bDTime'];
    mode=map['mode'];
  }
}
```

stat_settings_model.dart

```dart
class StatSettingsModel{
  String? tQuery;
  String? tTime;
  StatSettingsModel(this.tQuery,this.tTime);
  Map<String,dynamic> toMap(){
      Map<String,dynamic> map={};
      map['typeQuery']=tQuery;
      map['tTime']=tTime;
      return map;
  }
  StatSettingsModel.fromMap(Map<String,dynamic>map){
    tQuery=map['tQuery'];
    tTime=map['tTime'];
  }
}
```

transaction_model.dart

```dart
class TransactionModel{
  int? id;
  String? tType;
  String? tSubType;
  String? fType;
  String? amount;
  String? dTime;
```

```
  String? sord; //source or destination
  String? desc;
  String? sAmount;
  String? dateReadable;
  TransactionModel({required this.id,required this.tType,required
this.tSubType,required this.fType,
    required this.amount,required this.dTime,required this.sord,required
this.desc,});
  Map<String,dynamic> toMap(){
    Map<String,dynamic> map={};
    map['id']=id;
    map['tType']=tType;
    map['tSubType']=tSubType;
    map['amount']=amount;
    map['fType']=fType;
    map['dTime']=dTime;
    map['sord']=sord;
    map['desc']=desc;
    return map;
  }
  TransactionModel.fromMap(Map<String,dynamic> map){
    id=map['id'];
    if(map['tType']!=null){
      tType=map['tType'];
    }
    if(map['tSubType']!=null) {
      tSubType = map['tSubType'];
    }
    if(map['amount']!=null) {
      amount = map['amount'].toString();
    }
    if(map['fType']!=null){
      fType=map['fType'];
    }
    if(map['dTime']!=null) {
      dTime = map['dTime'].toString();
    }
    if(map['sord']!=null){
      sord=map['sord'];
    }
    if(map['desc']!=null){
      desc=map['desc'];
    }
    if(map['SUM(amount)']!=null){
      sAmount=map['SUM(amount)'].toString();
    }
    else{
      sAmount='0';
```

```
    }
    if(map['SUBSTR(dTime,1,10)']!=null){
      dateReadable=map['SUBSTR(dTime,1,10)'];
    }
    else{
      dateReadable="";
    }
  }
}
```

Funds_class.dart

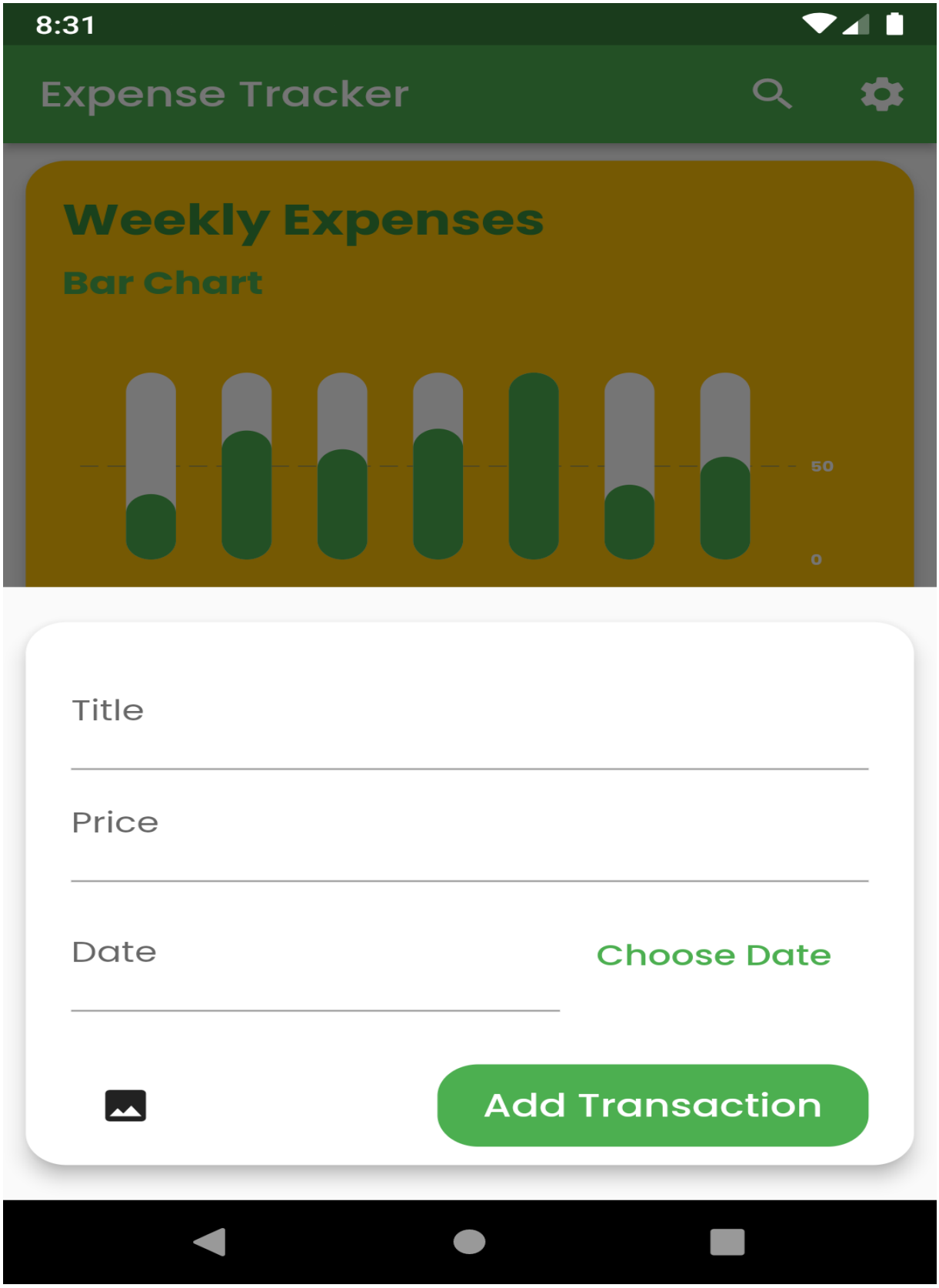
```
abstract class Income{
  String tType='Income';
  late String fundsType;
  late String dateTime;
  late num amount;
}
class Inflow extends Income{
    String? source;
    String? description;
    late String tSubtype;
    Inflow({required String fundsType,required String dateTime,required
amount,this.source,this.description}){
      tSubtype=runtimeType.toString();
      this.fundsType=fundsType;
      this.dateTime=dateTime;
      this.amount=amount;
    }
}
class PrizeMoney extends Income{
  String? source;
  String? description;
  late String tSubtype;
  PrizeMoney({required String fundsType,required String dateTime,required
amount,this.source,this.description}){
    tSubtype=runtimeType.toString();
    this.fundsType=fundsType;
    this.dateTime=dateTime;
    this.amount=amount;
  }
}
class LoanReturns extends Income{
  String? source;
  String? description;
  late String tSubtype;
  LoanReturns({required String fundsType,required String dateTime,required
amount,this.source,this.description}){
```

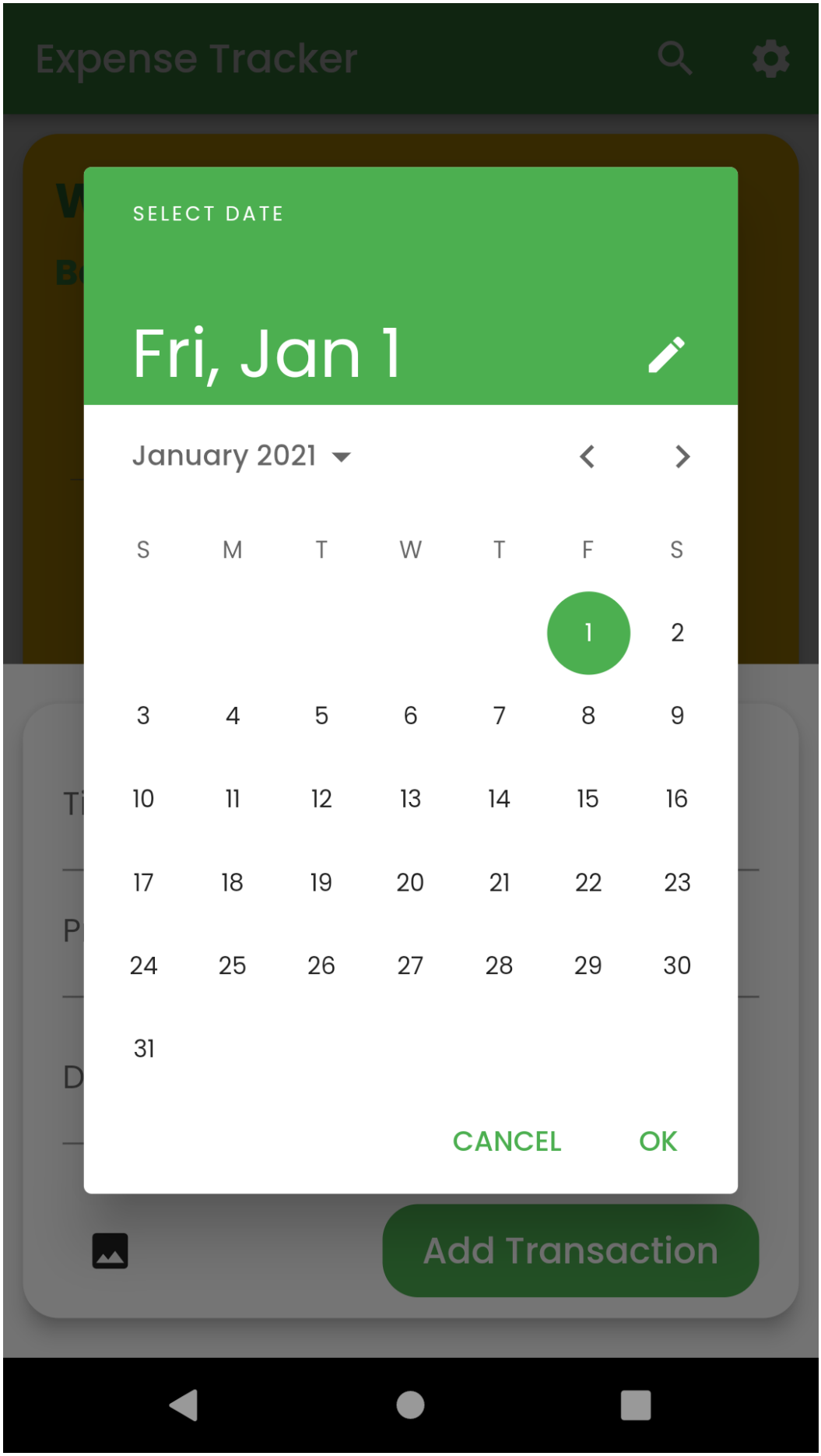
```
    tSubtype=runtimeType.toString();
    this.fundsType=fundsType;
    this.dateTime=dateTime;
    this.amount=amount;
  }
}
class LoanTaken extends Income{
  String? source;
  String? description;
  late String tSubtype;
  LoanTaken({required String fundsType,required String dateTime,required
amount,this.source,this.description}){
    tSubtype=runtimeType.toString();
    this.fundsType=fundsType;
    this.dateTime=dateTime;
    this.amount=amount;
  }
}
class InvestmentReturns extends Income{
  String? source;
  String? description;
  late String tSubtype;
  InvestmentReturns({required String fundsType,required String
dateTime,required amount,this.source,this.description}){
    tSubtype=runtimeType.toString();
    this.fundsType=fundsType;
    this.dateTime=dateTime;
    this.amount=amount;
  }
}
class Salary extends Income{
  String? source;
  String? description;
  late String tSubtype;
  Salary({required String fundsType,required String dateTime,required
amount,this.source,this.description}){
    tSubtype=runtimeType.toString();
    this.fundsType=fundsType;
    this.dateTime=dateTime;
    this.amount=amount;
  }
}
abstract class Expense{
  String tType='Expense';
  late String fundsType;
  late String dateTime;
  late num amount;
}
```

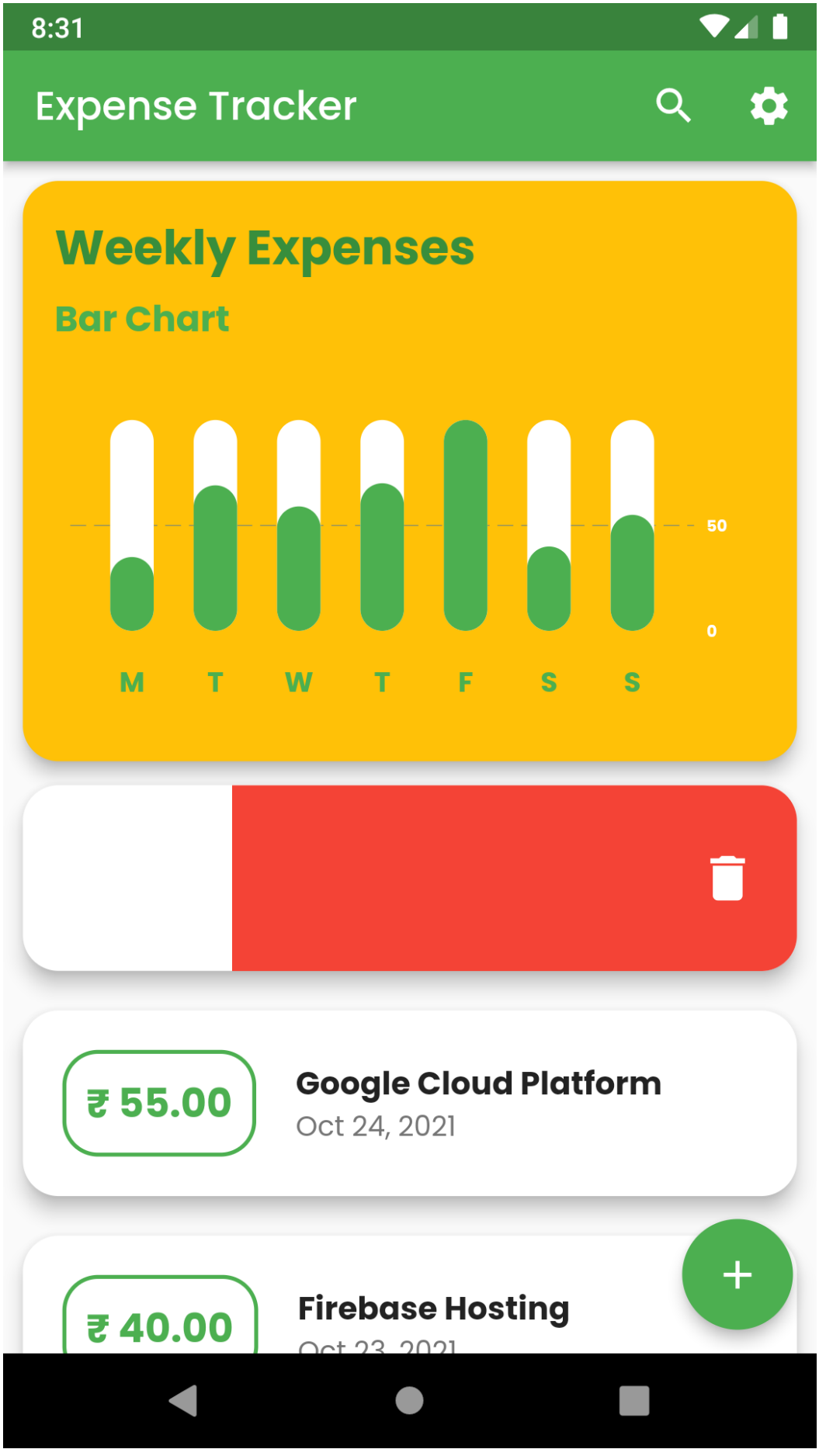
```
class FoodAndBeverage extends Expense{
  String? destination;
  String? description;
  late String tSubtype;
  FoodAndBeverage({required String fundsType,required String dateTime,required
amount,this.destination,this.description}){
    tSubtype=runtimeType.toString();
    this.fundsType=fundsType;
    this.dateTime=dateTime;
    this.amount=amount;
  }
}
class StationarySupplies extends Expense{
  String? destination;
  String? description;
  late String tSubtype;
  StationarySupplies({required String fundsType,required String
dateTime,required amount,this.destination,this.description}){
    tSubtype=runtimeType.toString();
    this.fundsType=fundsType;
    this.dateTime=dateTime;
    this.amount=amount;
  }
}
class EntertainmentAndLeisure extends Expense{
  String? destination;
  String? description;
  late String tSubtype;
  EntertainmentAndLeisure({required String fundsType,required String
dateTime,required amount,this.destination,this.description}){
    tSubtype=runtimeType.toString();
    this.fundsType=fundsType;
    this.dateTime=dateTime;
    this.amount=amount;
  }
}
class Subscription extends Expense{
  String? destination;
  String? description;
  late String tSubtype;
  Subscription({required String fundsType,required String dateTime,required
amount,this.destination,this.description}){
    tSubtype=runtimeType.toString();
    this.fundsType=fundsType;
    this.dateTime=dateTime;
    this.amount=amount;
  }
}
```

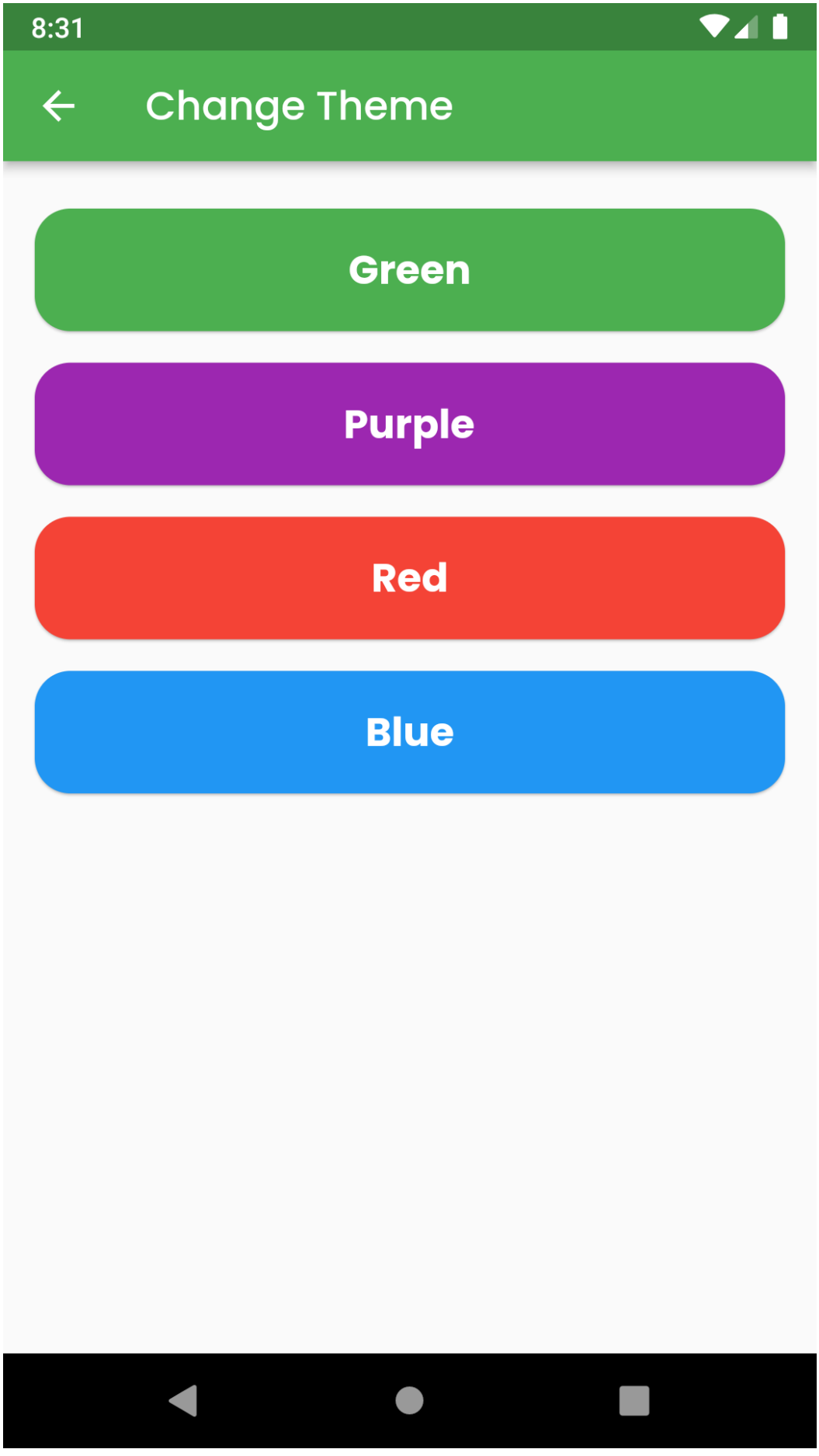
```
class LoanReturned extends Expense{
  String? destination;
  String? description;
  late String tSubtype;
  LoanReturned({required String fundsType,required String dateTime,required
amount,this.destination,this.description}){
    tSubtype=runtimeType.toString();
    this.fundsType=fundsType;
    this.dateTime=dateTime;
    this.amount=amount;
  }
}
class LoanGiven extends Expense{
  String? destination;
  String? description;
  late String tSubtype;
  LoanGiven({required String fundsType,required String dateTime,required
amount,this.destination,this.description}){
    tSubtype=runtimeType.toString();
    this.fundsType=fundsType;
    this.dateTime=dateTime;
    this.amount=amount;
  }
}

}
```

OUTPUT SCREENSHOTS:


8:31
Expense Tracker
Weekly Expenses
Bar Chart
50
0
Title
Price
Date
Choose Date
Add Transaction


Expense Tracker
SELECT DATE
Fri, Jan 1
January 2021
S M T W T F S
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
CANCEL
OK
Add Transaction


8:31
Expense Tracker
Weekly Expenses
Bar Chart
50
0
M
T
W
T
F
S
S
₹ 55.00
Google Cloud Platform
Oct 24, 2021
₹ 40.00
Firebase Hosting


8:31
Change Theme
Green
Purple
Red
Blue