```
import 'package:flutter/material.dart';
import 'package:scoped_model/scoped_model.dart';
import 'ExpenseListModel.dart';
import 'Expense.dart';
void main() {
 final expenses = ExpenseListModel();
 runApp(
 ScopedModel<ExpenseListModel>(
 model: expenses, child: MyApp(),
 )
 );
}
class MyApp extends StatelessWidget {
 // This widget is the root of your application.
 @override
 Widget build(BuildContext context) {
 return MaterialApp(
 title: 'Expense',
 theme: ThemeData(
 primarySwatch: Colors.blue,
 ),
 home: MyHomePage(title: 'Expense calculator'),
 );
 }
}
class MyHomePage extends StatelessWidget {
 MyHomePage({Key key, this.title}) : super(key: key);
 final String title;
 @override
 Widget build(BuildContext context) {
 return Scaffold(
 appBar: AppBar(
 title: Text(this.title),
 ),
```

```
 body: ScopedModelDescendant<ExpenseListModel>(
 builder: (context, child, expenses) {
 return ListView.separated(
 itemCount: expenses.items == null ? 1
 : expenses.items.length + 1, itemBuilder:
(context, index) {
 if (index == 0) {
 return ListTile( title: Text("Total
expenses: "
 + expenses.totalExpense.toString(),
 style: TextStyle(fontSize: 24,fontWeight:
 FontWeight.bold),) );
 } else {
 index = index - 1; return Dismissible(
 key:
Key(expenses.items[index].id.toString()),
 onDismissed: (direction) {

expenses.delete(expenses.items[index]);
 Scaffold.of(context).showSnackBar(
 SnackBar(
 content: Text(
 "Item with id, " +
expenses.items[index].id.toString()
 + " is dismissed"
 )
 )
 );
 },
 child: ListTile( onTap: () {
 Navigator.push( context,
MaterialPageRoute(
 builder: (context) => FormPage(
 id: expenses.items[index].id,
```

```
expenses: expenses,
 )
 ));
 },
 leading: Icon(Icons.monetization_on),
 trailing:
Icon(Icons.keyboard_arrow_right),
 title:
Text(expenses.items[index].category + ": " +

expenses.items[index].amount.toString() + " \nspent on " +
 expenses.items[index].formattedDate,
 style: TextStyle(fontSize: 18,
fontStyle: FontStyle.italic),))
 );
 }
 },
 separatorBuilder: (context, index) {
 return Divider();
 },
 );
 },
 ),
 floatingActionButton:
ScopedModelDescendant<ExpenseListModel>(
 builder: (context, child, expenses) {
 return FloatingActionButton(
 onPressed: () {
 Navigator.push(
 context, MaterialPageRoute(
 builder: (context)
 =>
ScopedModelDescendant<ExpenseListModel>(
 builder: (context, child, expenses)
```

```
{
 return FormPage( id: 0,
expenses: expenses, );
 }
 )
 )
 );
// expenses.add(
 new Expense(
 // 2, 1000, DateTime.parse('2019-04-01
11:00:00'), 'Food'
 )
 );
// print(expenses.items.length);
 },
 tooltip: 'Increment', child: Icon(Icons.add),
 );
 }
 )
 );
 }
}
class FormPage extends StatefulWidget {
 FormPage({Key key, this.id, this.expenses}) : super(key: key);
 final int id;
 final ExpenseListModel expenses;

 @override
 _FormPageState createState() => _FormPageState(id: id,
expenses: expenses);
}
class _FormPageState extends State<FormPage> {
 _FormPageState({Key key, this.id, this.expenses});
 final int id;
```

```
 final ExpenseListModel expenses;
 final scaffoldKey = GlobalKey<ScaffoldState>();
 final formKey = GlobalKey<FormState>();
 double _amount; DateTime _date;
 String _category;
 void _submit() {
 final form = formKey.currentState;
 if (form.validate()) {
 form.save();
 if (this.id == 0) expenses.add(Expense(0, _amount,
_date, _category));
 else expenses.update(Expense(this.id, _amount, _date,
_category));
 Navigator.pop(context);
 }
 }
 @override
 Widget build(BuildContext context) {
 return Scaffold(
 key: scaffoldKey, appBar: AppBar(
 title: Text('Enter expense details'),
 ),
 body: Padding(
 padding: const EdgeInsets.all(16.0),
 child: Form(
 key: formKey, child: Column(
 children: [
 TextFormField(
 style: TextStyle(fontSize: 22),
 decoration: const InputDecoration(
 icon: const
Icon(Icons.monetization_on),
 labelText: 'Amount',
 labelStyle: TextStyle(fontSize: 18)
```

```
 ),
 validator: (val) {
 Pattern pattern = r'^[1-
9]\d*(\.\d+)?$';
 RegExp regex = new RegExp(pattern);
 if (!regex.hasMatch(val)) return
'Enter a valid number';
 else return null;
 },
 initialValue: id == 0 ? ''
 : expenses.byId(id).amount.toString(),
 onSaved: (val) => _amount =
double.parse(val),
 ),
TextFormField(
 style: TextStyle(fontSize: 22),
 decoration: const InputDecoration(
 icon: const
Icon(Icons.calendar_today),
 hintText: 'Enter date',
 labelText: 'Date',
 labelStyle: TextStyle(fontSize: 18),
 ),
 validator: (val) {
 Pattern pattern = r'^((?:19|20)\d\d)[-
/.]
 (0[1-9]|1[012])[- /.](0[1-9]|[12][0-
9]|3[01])$';
 RegExp regex = new RegExp(pattern);
 if (!regex.hasMatch(val)) return
'Enter a valid date';
 else return null;
 },
 onSaved: (val) => _date =
```

```
DateTime.parse(val),
 initialValue: id == 0 ? '' :
expenses.byId(id).formattedDate,
 keyboardType: TextInputType.datetime,
 ),
TextFormField(
 style: TextStyle(fontSize: 22),
 decoration: const InputDecoration(
 icon: const Icon(Icons.category),
 labelText: 'Category',
 labelStyle: TextStyle(fontSize: 18)
 ),
 onSaved: (val) => _category = val,
 initialValue: id == 0 ? '' :
expenses.byId(id).category.toString(),
 ),
RaisedButton(
 onPressed: _submit,
 child: new Text('Submit'),
 ),
 ],
 ),
 ),
 ),
 );
 }
}
```