**Chapter 1**

**Case Study**

**Voter Grievance Redressal Mobile Application – Sahaaya**

In the present time people are beginning to lose their faith in the democratic system where the people themselves volunteer to conduct elections by spending their own money to choose their representatives and later rant in despair as there is nobody to listen to the problems they face in their neighborhood.

The problems may span across a number of categories such as water, electricity, roads, waste management etc. The aim of our application, **Sahaaya,** which is a mobile application is, to group together these problems under the departments that are meant to address the respective problems.

The user is provided with a number of categories such as roads, sanitation, waste management, water, electricity, corruption etc. under which he/she could raise an issue. This complaint is recorded under that particular category and the concerned authorities are notified. If the issue is not resolved in a reasonable period of time, the overall score of that constituency goes down.

This app will also rank constituencies and their respective leaders based on the number of issues they have addressed in a period of time. Thus, people can know how effectively his/her representative is carrying out the work. ­­­

**Dart and Flutter**

**Dart** is an object-oriented, class-based programming language optimized for building user interfaces. It is developed by Google and is used to build mobile, desktop, server, and web applications.

**Flutter** is Google’s UI toolkit for building beautiful, natively compiled applications for mobile, web, and desktop from a single codebase. Flutter apps are written in the Dart language and make use of many of the language's more advanced features.

**Chapter 2**

**Design Pattern**

A design pattern is a general repeatable solution to a commonly occurring problem in software design. Many design patterns have gone into the design and development of our application. Most often used design patterns are **Façade** and **Template** patterns.

**Template Method**

The Template Method is a pattern intended to help one abstract out a common process from different procedures. According to the **Gang of Four**, the intent of the template method is to:

**“**Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Redefine the steps in an algorithm without changing the algorithm's structure**“**.

**Key Features**

**Intent**

Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Redefine the steps in an algorithm without changing the algorithm's structure.

**Problem**

There is a procedure or set of steps to follow that is consistent at one level of detail, but individual steps may have different implementations at a lower level of detail.

**Solution**

Allows for definition of sub steps that vary while maintaining a consistent basic process.

**Participants and Collaborators**

The Template Method consists of an abstract class that defines the basic Template Method classes that need to be overridden. Each concrete class derived from the abstract class implements a new method for the Template.

**Consequences**

Templates provide a good platform for code reuse. They also are helpful in ensuring the required steps are implemented. They bind the overridden steps together for each Concrete class, and so should only be used when these variations always and only occur together.

**Implementation**

Create an abstract class that implements a procedure using abstract methods. These abstract methods must be implemented in subclasses to perform each step of the procedure. If the steps vary independently, each step may be implemented with a Strategy pattern.

**Gang of Four Reference**

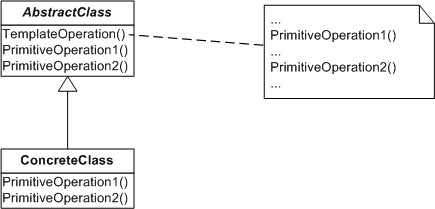


Fig 2.1

**Example Implementation**



Fig 2.2

**Implementation of Template Method in Sahaaya Application**

**Use Case 1: Checking for internet connectivity**

In our application we need to check if the user has an active internet connection in his mobile across various screens of the application. Thus we have to repeatedly keep calling a method which checks if the user is connected to the internet and display the respective alert message.

Class **NetworkCheck** is an abstract class that defines two methods namely **checkInternetStatus()** and **showAlert()** that when implemented checks if the user is connected to the internet.

This abstract class is extended by the **CheckStatus** class that implements the above mentioned methods. Other classes such as **HomePage** , **Statistcs**, **NewGrievance.dart** create objects of the **CheckStatus** class and call the respective methods thus depicting a **has-a relationship**. If the user is not connected to the internet the function **showAlert()** displays the appropriate message. If the user is connected to the internet then the Splash Screen animation (the first screen that is displayed when the user opens the app) is executed.

**Use Case 2: Implementing animation through the use of Animated Widget class**

Widgets are the building blocks in a flutter application. In order to implement animations in our app we have made use of the built in Animated Widget class provided by the flutter framework.

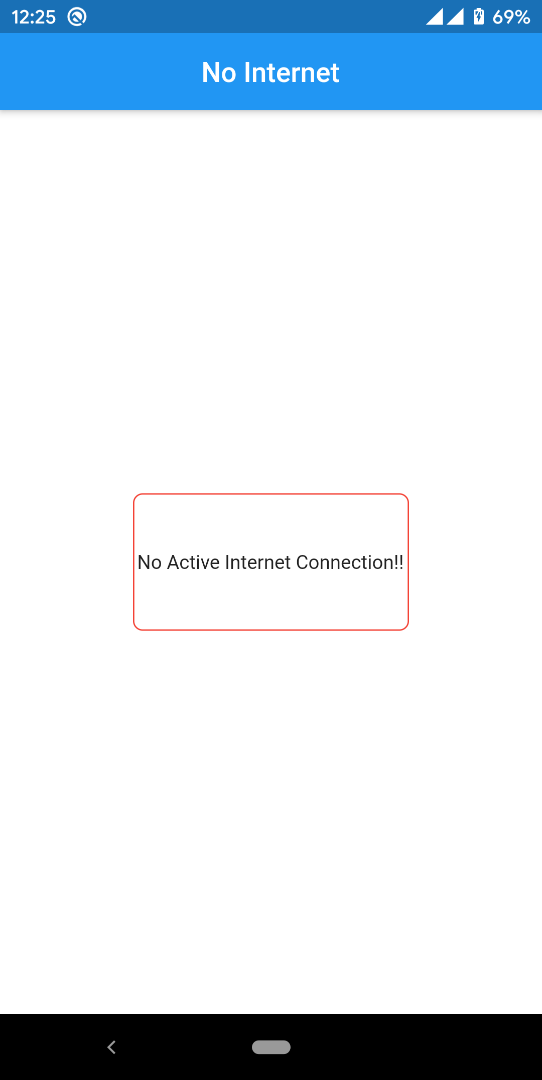
AnimatedWidget class is an abstract class that defines only one method to be implemented which is the Widget build() method that displays the specified widget on to the screen. Here class SplashScreen extends the AnimatedWidget class and thus executes the Splash Screen animation.

**Chapter 4**

**Implementation and Output**

Implementation of the Template design pattern in dart programming language is present in the codes folder.

**Output**



Splash Screen when internet is active Alert message when internet is not active

Corresponding video attached Internet.mp4 Corresponding video attached NoInternet.mp4