# Lab 15

Instructor: Sidra Khatoon E-mail: skhatoon[@uit.edu](mailto:ad@uit.edu)

# Objective

# The objective of lab is exploring global state management and create a cart by using provider.

# .

**Student Information**

|  |  |
| --- | --- |
| **Student Name** |  |
| **Student ID** |  |
| **Date** |  |

**Assessment**

|  |  |
| --- | --- |
| **Marks Obtained** |  |
| **Remarks** |  |
| **Signature** |  |

# Objective

# The objective of lab is exploring global state management and create a cart by using provider.

# Instructions

You have to perform the following tasks yourselves. Raise your hand if you face any difficulty in understanding and solving these tasks. **Plagiarism** is an abhorrent practice and you should not engage in it.

# How to Submit?

Submit lab work using Teams.

**Global State Management:**

Global state, is state that needs to be accessible from multiple parts of the app. For example, if you have a shopping cart that needs to be accessed from various screens, you’ll need a way to manage that state globally.

Managing global state can become complex, especially as your app grows. This is where tools like Provider come in handy, making it easier to share and manage state across the entire app.

**State Management with Provider:**

Provider is a package that simplifies state management by allowing you to share data across your app and rebuild UI parts when the data changes. It’s perfect for beginners because it’s both easy to use and powerful enough to handle more complex cases as your app scales.

**Core Concepts**

**ChangeNotifier**: This class helps notify listeners (widgets) when the state changes. It’s the core of most state management solutions in Flutter.

**Provider:** This widget is used to make an instance of ChangeNotifier available throughout the widget tree.

**Consumer:** A widget that listens to Provider and rebuilds whenever the state it depends on changes.

**Creating an App with Global State Management**

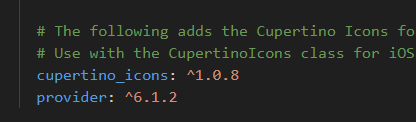
Let’s build an example of Counter, and this state will be managed globally using Provider.

you’ll need to set up the project and include the Provider package. Let run command in flutter terminal

flutter pub add provider

flutter pub get

After these command provider package in included in pubspec.yaml. You can check your file.



A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

**Add Item to List**

Let’s build an example where users can add items to a list, and this state will be managed globally using Provider.

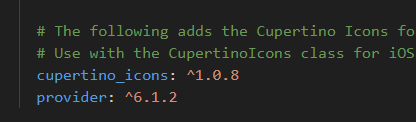
**Step 1: Setting Up the Project**

you’ll need to set up the project and include the Provider package. Let run command in flutter terminal

flutter pub add provider

flutter pub get

After these command provider package in included in pubspec.yaml. You can check your file.



**Step 2: Creating the State Model**

We’ll start by creating a state model that will manage the list of items.



**Explanation**: In the ItemModel class:

\_items: A private list that holds the state of our items.

addItem: Adds a new item to the list and notifies listeners that the state has changed.

removeItem: Removes an item from the list by its index and notifies listeners.

**Step 3: Providing the State Model**

Next, we wrap our app with ChangeNotifierProvider to provide the ItemModel to all widgets in the app.



**Explanation:**

* The ChangeNotifierProvider makes the ItemModel available to the entire widget tree.
* Any widget in the tree can now access and modify the global state.

**Step 4: Building the UI**

Now, let’s build the UI where users can add and remove items from the list.

**** A screen shot of a computer code

Description automatically generated

**Explanation:**

* TextField: Captures the item to be added.
* ElevatedButton: Triggers the addItem method in ItemModel to add the item to the list.
* Consumer: Listens to changes in ItemModel and rebuilds the ListView whenever an item is added or removed.

# Assessment:

# Create a Flutter app for managing a simple list of tasks. Use the Provider package to manage the list of tasks, where each task has a title, description, and a completion status.

# Expected Output:

# A basic task management app where:

# A list of tasks can be added and displayed.

# Each task has a checkbox to mark it as completed.

# The task list updates immediately upon adding or toggling tasks.