**Lab Exercise 6 – Working with Advanced Widgets in PySide6 (TreeView, TableView, ComboBox)**

**Objective:**

This lab focuses on using **TreeView, TableView, and ComboBox** in PySide6 to build interactive applications with structured data. You will learn how to:

* Display hierarchical data using **QTreeView**.
* Show tabular data using **QTableView** with a data model.
* Use **QComboBox** to let users select options and update data dynamically.

**1. Learning Outcomes**

By the end of this lab, you will:

✅ Use **QTreeView** to display hierarchical data.  
✅ Implement **QTableView** with a custom model.  
✅ Connect **QComboBox** to update TableView dynamically.  
✅ Work with **Qt’s Model/View architecture** for handling data efficiently.

**2. Project Setup & Requirements**

**Project Structure**

pyside6\_advanced\_widgets/

│── main.py # Python script for UI logic

│── requirements.txt # Dependencies (PySide6)

**Install Dependencies**

Ensure **PySide6** is installed:

pip install PySide6

**3. Step 1: Implement UI with Advanced Widgets (main.py)**

This script creates a **QTreeView**, **QTableView**, and **QComboBox** to interact with structured data.

import sys

from PySide6.QtWidgets import QApplication, QWidget, QVBoxLayout, QTreeView, QTableView, QComboBox

from PySide6.QtGui import QStandardItemModel, QStandardItem

class AdvancedWidgetsDemo(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle("Advanced Widgets in PySide6")

self.setGeometry(100, 100, 600, 400)

layout = QVBoxLayout()

# ComboBox Setup

self.comboBox = QComboBox()

self.comboBox.addItems(["Fruits", "Vegetables"])

self.comboBox.currentTextChanged.connect(self.updateTableView)

layout.addWidget(self.comboBox)

# TreeView Setup

self.treeView = QTreeView()

self.treeModel = self.createTreeModel()

self.treeView.setModel(self.treeModel)

layout.addWidget(self.treeView)

# TableView Setup

self.tableView = QTableView()

self.tableModel = self.createTableModel("Fruits") # Default category

self.tableView.setModel(self.tableModel)

layout.addWidget(self.tableView)

self.setLayout(layout)

def createTreeModel(self):

"""Creates a tree structure with parent-child relationships."""

model = QStandardItemModel()

rootNode = model.invisibleRootItem()

# Fruits Category

fruits = QStandardItem("Fruits")

fruits.appendRow(QStandardItem("Apple"))

fruits.appendRow(QStandardItem("Banana"))

fruits.appendRow(QStandardItem("Cherry"))

# Vegetables Category

vegetables = QStandardItem("Vegetables")

vegetables.appendRow(QStandardItem("Carrot"))

vegetables.appendRow(QStandardItem("Broccoli"))

vegetables.appendRow(QStandardItem("Spinach"))

# Add categories to root

rootNode.appendRow(fruits)

rootNode.appendRow(vegetables)

return model

def createTableModel(self, category):

"""Creates a table model based on the selected category."""

model = QStandardItemModel(3, 2) # 3 Rows, 2 Columns

model.setHorizontalHeaderLabels(["Item", "Price ($)"])

data = {

"Fruits": [("Apple", "1.2"), ("Banana", "0.8"), ("Cherry", "2.5")],

"Vegetables": [("Carrot", "0.5"), ("Broccoli", "1.0"), ("Spinach", "1.5")],

}

for row, (name, price) in enumerate(data[category]):

model.setItem(row, 0, QStandardItem(name))

model.setItem(row, 1, QStandardItem(price))

return model

def updateTableView(self, selected\_category):

"""Updates the TableView based on ComboBox selection."""

self.tableModel = self.createTableModel(selected\_category)

self.tableView.setModel(self.tableModel)

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

window = AdvancedWidgetsDemo()

window.show()

sys.exit(app.exec())

**4. Running the Application**

Run the Python script:

python main.py

✅ **TreeView** displays hierarchical categories (Fruits & Vegetables).  
✅ **TableView** shows tabular data, changing based on the ComboBox selection.  
✅ **ComboBox** updates TableView dynamically based on the selected category.

**5. Key Concepts in This Exercise**

✅ **QTreeView** → Displays hierarchical data using QStandardItemModel.  
✅ **QTableView** → Uses a data model to display structured tabular data.  
✅ **QComboBox** → Changes TableView data dynamically based on user selection.  
✅ **Model/View Architecture** → Efficiently manages large UI data updates.