**Lab Exercise 4 – Model-View-Controller (MVC) in Qt/QML/PyQt**

Objective: In this lab exercise, you will learn how to implement the Model-View-Controller (MVC) pattern in a Qt/QML/PyQt application for a simple To-Do List.

**Instructions:**

**Part 1: Model**

Open a Python IDE or text editor.

Create a Python class TodoModel that represents the Model for the To-Do List. This class should manage the list of tasks (To-Do items).

class TodoModel:

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

self.tasks = []

def add\_task(self, task):

self.tasks.append(task)

def get\_tasks(self):

return self.tasks

def remove\_task(self, task):

if task in self.tasks:

self.tasks.remove(task)

**Part 2: View**

Create a QML file named TodoView.qml. This file defines the View for the To-Do List.

import QtQuick 2.15

import QtQuick.Controls 2.15

ApplicationWindow {

visible: true

width: 400

height: 400

title: "To-Do List"

ListView {

id: listView

width: parent.width

height: parent.height

model: ListModel {

ListElement { task: "Task 1" }

ListElement { task: "Task 2" }

}

delegate: Item {

width: listView.width

height: 50

Text {

text: model.task

anchors.centerIn: parent

}

}

TextField {

id: taskInput

width: parent.width

placeholderText: "Add a task..."

onAccepted: {

var task = taskInput.text

if (task !== "") {

listView.model.append({ task: task })

taskInput.text = ""

}

}

}

}

}

**Part 3: Controller**

Create a Python class TodoController that represents the Controller for the To-Do List. This class should interact with the Model and connect it to the View.

class TodoController:

def \_\_init\_\_(self, model, view):

self.model = model

self.view = view

# Connect the view's add task signal to the model's add\_task method

self.view.add\_task\_signal.connect(self.model.add\_task)

# Connect the view's remove task signal to the model's remove\_task method

self.view.remove\_task\_signal.connect(self.model.remove\_task)

**Part 4: Integrating Model, View, and Controller**

Create a Python script to launch the application and connect the Model, View, and Controller.

import sys

from PyQt5.QtCore import Qt, pyqtSignal

from PyQt5.QtGui import QGuiApplication

from PyQt5.QtQml import QQmlApplicationEngine

from PyQt5.QtWidgets import QApplication

from PyQt5.QtQuick import QQuickView

from PyQt5.QtCore import QObject

from model import TodoModel

from controller import TodoController

class TodoView(QObject):

add\_task\_signal = pyqtSignal(str)

remove\_task\_signal = pyqtSignal(str)

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

app = QApplication(sys.argv)

# Create Model, View, and Controller

model = TodoModel()

view = TodoView()

controller = TodoController(model, view)

# Load QML file

engine = QQmlApplicationEngine()

engine.rootContext().setContextProperty("todoView", view)

engine.load("TodoView.qml")

sys.exit(app.exec\_())

**Part 5: Running the Application**

* Run the Python script to launch the To-Do List application.
* Observe how tasks can be added and removed from the list using the user interface.

Conclusion: In this lab exercise, you implemented the Model-View-Controller (MVC) pattern in a simple To-Do List application using Qt/QML/PyQt. This exercise demonstrates how to structure an application by separating concerns, making it easier to manage and extend.