```
#!/usr/bin/python3
import gi
gi.require_version('Gtk','3.0')
from gi.repository import Gtk
from setting import Setting
import subprocess

class MainWindow(Gtk.Window):
    def __init__(self):
        Gtk.Window.__init__(self)
        self.nginx = 'nginx'
        self.php = 'php7.2-fpm'
        self.mysql = 'mysql'
        self.jsonSetting = Setting().fetchSetting()

        self.importObject()
        self.initial(self.jsonSetting["nginxPortEntry"])
        self.statusL()
        # self.headerLogoStatus()
        self.switchStatusText()
        self.switchButton.set_active((self.statusAllProcess()))
        self.switchButton.connect("notify::active", self.switchToggle)
        self.settingButton.connect("clicked",self.onSettingClicked)
        self.mainWin = self.builder.get_object('mainWin')
        self.mainWin.connect('delete-event',self.quit)

        self.mainWin.show_all()

        Gtk.main()

    def switchStatusText(self):
        if(self.statusAllProcess()):
            self.switchStatus.set_text("Server Started")
            self.markup(self.switchStatus, '<b>Server Started</b>')
        else:
            self.switchStatus.set_text("Server Stopped")


    def startStop(self):
        if self.jsonSetting["startWithGemp"]:
            self.startAllProcess()
        else:
            print("Error: StartWithGemp")

    def onSettingClicked(self, widget):
        self.mainWin.set_opacity(0.5)
        self.setting = Setting()
        self.setting.main()
        print("back")
        self.backFromSetting()

        # print("Back")
    def backFromSetting(self):
        self.mainWin.set_opacity(1)
        self.mainWin.connect('delete-event',self.quit)
        if self.reloadAllServices(True):
            print("All Service Reloaded")
        else: print("Reloading Error")

    def reloadAllServices(self, widget):
        return (self.processControl(self.nginx, "restart") and 
self.processControl(self.php, "restart") and self.processControl(self.mysql,
"restart"))

    def initial(self, jsonSetting):
        self.osStatus.set_text(self.osStatusCheck())
        self.uidStatus.set_text(subprocess.getstatusoutput('whoami')[1])
        self.linkButton.set_label('Open Web')
        if self.processControl(self.nginx, 'status'):
            self.linkButton.set_uri("http://localhost:{}".format(jsonSetting))
        else:
            self.linkButton.set_uri('#')
        self.switchButton.set_active(self.statusAllProcess)

    def osStatusCheck(self):
        return subprocess.getstatusoutput('cat /etc/*-release')[1].split('\n')
[3].split('=')[1].replace('"',"")

    def importObject(self):
        self.builder = Gtk.Builder()
        self.builder.add_from_file('gui/main.glade')

        self.logo = self.builder.get_object("logo")
        self.reloadButton = self.builder.get_object("reloadButton")
        self.reloadButton.connect("clicked", self.reloadAllServices)
        self.switchButton = self.builder.get_object('switchButton')
        self.switchStatus = self.builder.get_object('switchStatus')
        self.nginxStatusL = self.builder.get_object('nginxStatusL')
        self.mysqlStatusL = self.builder.get_object('mysqlStatusL')
        self.phpStatusL = self.builder.get_object('phpStatusL')
        self.osStatus = self.builder.get_object('osStatus')
        self.uidStatus = self.builder.get_object('uidStatus')
        self.ipStatus = self.builder.get_object('ipStatus')
        self.msgText = self.builder.get_object('msgText')
        self.msgText.set_no_show_all(True)
        self.warningBox = self.builder.get_object("warningBox")
        self.warningBox.set_no_show_all(True)
        self.linkButton = self.builder.get_object('linkButton')
        self.settingButton = self.builder.get_object("settingButton")
        self.nginxdot = self.builder.get_object("nginxdot")
        self.mysqldot = self.builder.get_object("mysqldot")
        # self.mysqldot.set_no_show_all(True)
        self.phpdot = self.builder.get_object("phpdot")

        # self.warningBox = self.builder.get_object('warningBox')
        # self.warningBox.set_visible(False)

    def installServices(self, service):
        # if (service == 'php7.1-fpm'):
        #     status = subprocess.getstatus
        pass


    def statusAllProcess(self):
        if ((self.processControl(self.nginx,'status')) and
(self.processControl(self.php, 'status')) and (self.processControl(self.mysql,
'status'))):
            self.logo.set_from_file('gui/colorHeader.png')
            return True
        else:
            self.logo.set_from_file("gui/blackHeader.png")
            return False


    def switchToggle(self, switch, gparam):
        if switch.get_active():
            if self.startAllProcess():
                self.logo.set_from_file('gui/colorHeader.png')
                print("All Process Started")
            else:
                print("Error while Starting All Process")
        else:
            if self.stopAllProcess():
                    print("All process Stoped")
                    self.logo.set_from_file("gui/blackHeader.png")
            else:
                print("Error while stopping Process")
        self.switchStatusText()
        self.statusL()
        # self.switchButton.set_active(not(self.statusAllProcess()))

    def startAllProcess(self):
        # if not((subprocess.call(['sudo','service','nginx','start']) and subprocess.call(['sudo','service','php7.0-fpm','start']) and subprocess.call(['sudo','service','mysql','start'])))
        if (self.processControl(self.nginx,'start') and self.processControl(self.php, 'start') and self.processControl(self.mysql, 'start')):
            return True
        else:
            print("StartAllProcess Error")
            return False

    def stopAllProcess(self):
        if (self.processControl(self.nginx,'stop') and self.processControl(self.php, 'stop') and self.processControl(self.mysql, 'stop')):
            return True
        else:
            print("stopAllProcess Error")
            return False

    def statusL(self):
        if self.processControl(self.nginx, 'status'):
            self.markup(self.nginxStatusL, '<b>Nginx</b>')
            self.nginxdot.set_from_file("gui/16green.png")
        else:
            self.markup(self.nginxStatusL, 'Nginx')
            self.nginxdot.set_from_file("gui/16red.png")

        if self.processControl(self.php, 'status'):
            self.markup(self.phpStatusL, '<b>PHP</b>')
            self.phpdot.set_from_file("gui/16green.png")
        else:
            self.markup(self.phpStatusL, 'PHP')
            self.phpdot.set_from_file("gui/16red.png")

        if self.processControl(self.mysql, 'status'):
            self.markup(self.mysqlStatusL, '<b>MySQL</b>')
            self.mysqldot.set_from_file("gui/16green.png")
        else:
            self.markup(self.mysqlStatusL, 'MySQL')
            self.mysqldot.set_from_file("gui/16red.png")

    def markup(self, object, text):
        object.set_markup(text)
```

```python


    def processControl(self, service, signal):
        status = (subprocess.getstatusoutput('sudo systemctl {} {}'.format(signal, service))[0])
        if (status == 1):
            print('Process {} is Masked\nUnmasking service...!'.format(service))
            if(self.processControl(service,'unmask')):
                print("unable to Unmask...!")
            else:
                print("Unmasking {} Sucessufull".format(service))
        elif (status == 0):
            print("ProcessControl", status, service)
            return True
        elif (status in [4,5]):
            if (self.installServices(service)):
                print("{} install Sucessfully".format(service))
            else:
                print("Error in installing...")
        else:
            print("Process: {} {}".format(service, status))
            return False

    def quit(self, signal, s):
        print("stopWithGemp", self.jsonSetting["stopWithGemp"])
        if self.jsonSetting["stopWithGemp"]:
            self.stopAllProcess()
        else:
            print("Error: stopWithQuit")
        print("Quiting")
        self.mainWin.close()
        Gtk.main_quit()

```