
# APPLICATION MANAGER

## (Ubuntu based Linux OS)

A Project Report

Submitted as part of the Course

**OPERATING SYSTEMS**

**CSE 2005**

**School of Computer Science Engineering**

**VIT Chennai**

**Winter Semester: 2020-2021**

**Course Faculty: Dr.Rishikeshan C A**

Submitted By

Akshara M - 19BPS1061

Aravind - 19BPS1110

Balaji R - 19BCE1744

## ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, Dr. Rishikeshan C A, School of Computer Science and Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to Dr. Jagadeesh Kannan, Dean of the School of Computer Science and Engineering, VIT Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our Head of the Department Dr. Neelanarayanan for his support throughout the course of this project. We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

# INDEX

## ABSTRACT

The project we worked on is an Application manager to easily install, update and remove software in Ubuntu-based Linux Operating Systems. Application Management (AM) is actually the lifecycle process for software applications, covering how an application operates, its maintenance, version control, and upgrades from cradle to grave. Application management services are an enterprise-wide endeavor providing governance designed to ensure applications run at peak performance and as efficiently as possible.

In this manner, Application manager acts as a service operation function that manages and supports applications and key stakeholders who provide operational proficiency or technical expertise through the lifecycle.

We have used Tkinter for the front end which is a python framework which is Python's de-facto standard GUI (Graphical User Interface) package. It is a thin object-oriented layer on top of Tcl/Tk. Tkinter is not the only GuiProgramming toolkit for Python. It is however the most commonly used one.For back end we have used python programming language .We have used SQL for storing application data which is used to communicate with the database. It is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database.

This project allows users to install,update,delete,search for applications or packages at a common interface to facilitate operations at ease.We are developing this application manager on Ubuntu based Linux.

# INTRODUCTION

Applications Manager is an application performance monitoring software that provides deep visibility into the performance and user experience of your business-critical applications and infrastructure components. It helps you to isolate and resolve performance issues across your entire application stack quickly - from the URL to the line of code - with minimal overhead, before your customers are affected. Organizations of all sizes use Application Manager to improve and automate IT and DevOps processes, ensure optimal end-user experience, and deliver better business outcomes. It is the job of the Application Manager to identify the software, then acquire the software, prepare the servers, install the software, configure it, load data into it, tune it, upgrade it, and repair it when there are problems.

An application manager (app manager) is programming for overseeing the installation, patching and updating and perhaps access of software applications. An application manager can be used to monitor a software application's performance and alert administrators if there is a problem. Some application managers for enterprise IT also provide context-aware network access control. In such a scenario, if a device fails to comply with the app manager's security policies, the device is refused access to applications.Application management is related to application lifecycle management, which involves the tracking of different software versions.

## OBJECTIVES

The operations of this application manager are to perform the following functions:

1) Check for Software Updates.
2) View Installed Applications.
3) Search for an Application.
4) Install an Application.
5) Remove Application.
6) Remove Application with Configuration Files.
7) Update an Application.
8) Update all Applications.
9) Remove Obsolete/Unwanted Packages.
10) Upgrade Distribution

# GOALS

Managing multiple applications task in one place in a Linux OS is a hectic task as for now, we wanted to address that issue and we worked on creating an Application manager to easily install, update and remove software in one place ,with this application all the processes can be done on a simple touch of a button so we need not remember the commands require to do the process and helps the user to save time and when applications are efficiently managed, more IT resources are available to focus on new business challenges and competitive issues.Additionally, effectively managed applications are more reliable and less prone to failure that could lead to loss of functionality. Thus, application management can reduce the risk of downtime and improve overall business continuity.Application management can provide an enhanced end-user experience, which not only increases productivity but also helps accelerate the adoption of new applications or features

## PROJECT DESCRIPTION

## LIST OF MODULES

**1. Check for Software Updates**

When you run the updates, it updates all the packages installed via apt. This means updating Ubuntu will update the core operating system, Linux kernels as well as the applications installed from the software center (if they were apt packages) or installed using apt command.

**2. View Installed Application**

This will list all the packages that have been installed using apt. It will also list the packages that were installed as a dependency. Which means that not only you'll have the applications you installed, you'll also have a huge list of libraries and other packages that you didn't install directly.

**3. Search for an Application**

This is to find the name of a specific application that we want to install. The apt-cache command is available to show the information stored in the repositories. The way this information is gathered is via a kind of a cache or temporary memory as it is a temporary combination of the content of different software source.

**4. Install an Application**

When Ubuntu Software launches, search for an application, or select a category and find an application from the list.Select the application that you want to install and click Install.You will be asked to authenticate by entering your password. Once you have done that the installation will begin. The installation usually finishes quickly.

**5. Remove Application**

Find the application that you want to remove by using the search box or by looking through the list of installed applications. Select the application and click Remove. Confirm that you want to remove the application.

**6. Remove Application with Configuration File**

Search for package name and Fully remove the program along with config files using apt.Then we need to Confirm deletion if asked.

**7. Update an Application**

Check the updates you want to install. We need to select the app to update.Click the Install Updates button.Enter your (Sudo) password.Click Ok.This module updates application when we press update.

**8. Update all Application**

This module updates all applications on Ubuntu.User has to enter the password as this is a sudo operation.The user has to give yes ‘y’ key to see the updation of all Applications

**9. Remove Unwanted Package**

This module will remove all unused packages (orphaned dependencies). While the explicitly installed packages will remain.Press “Y” and Enter to remove all Applications.

**10. Upgrade Distribution**

This module focuses on upgrading all installed packages of Ubuntu version 18.04.Here also the user would be pressing ‘Y’ inorder to upgrade all packages.

# TOOLS AND SOFTWARE USED

**Ubuntu based Operating System**

Ubuntu is a Linux distribution based on Debian and composed mostly of free and open-source software. Ubuntu is officially released in three editions: Desktop, Server, and Core for Internet of things devices and robots

**Linux Pre installed**

Linux is an open source operating system (OS). An operating system is the software that directly manages a system's hardware and resources, like CPU, memory, and storage. The OS sits between applications and hardware and makes the connections between all of your software and the physical resources that do the work.

**Python Compiler(Visual Studio) with required packages**

A computer program that translates code written in one programming language into another is called a compiler. Python leads the faction of the fastest growing programming languages. As such, there is no scarcity to Python compilers that can cater to varying project needs.

**Front End: Tkinter Python Framework**

Python has a lot of GUI frameworks, but Tkinter is the only framework that's built into the Python standard library. Tkinter has several strengths. It's cross-platform, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run.

**Back End: SQL and Python**

The Python programming language has powerful features for database programming. Python supports various databases like SQLite, MySQL, Oracle, Sybase, PostgreSQL, etc. Python also supports Data Definition Language (DDL), Data Manipulation Language (DML) and Data Query Statements. The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard.

# IMPLEMENTATION DETAILS

## SOURCE CODE

```
from os import remove
import tkinter as tk
from tkinter import ttk
from control import Control
import time

class UI:
    def __init__(self):
        self.control = Control()
        self.root = tk.Tk()
        self.root.title("Application manager")
        self.sv = tk.StringVar(self.root)
        self.option_var = tk.StringVar(self.root)
        self.info_string = ""

        self.canvas1 = tk.Canvas(self.root, width = 1000, height = 350)


        label2 = tk.Label(self.root, text='Application u want to search:')
        label2.config(font=('helvetica', 14))
        self.canvas1.create_window(350, 50, window=label2)

        self.sv.trace("w", lambda name, index, mode, sv=self.sv: self.callback(sv))
        self.entry = tk.Entry(self.root, textvariable=self.sv)
        self.canvas1.create_window(650, 50, window=self.entry)

        self.hint_label = tk.Label(self.root, text=self.info_string)
        self.hint_label.config(font=('helvetica', 10))
        self.canvas1.create_window(500, 155, window=self.hint_label)

        button1 = tk.Button(self.root, text='Update System', command=self.updatesys)
        self.canvas1.create_window(300, 250, window= button1)

        button2 = tk.Button(self.root, text='Remove obselete', command=self.remove_garbage)
        self.canvas1.create_window(500, 250, window=button2)

        button3 = tk.Button(self.root, text='Upgrade distribution', command=self.upgrade)
        self.canvas1.create_window(700, 250, window= button3)

        button4 = tk.Button(self.root, text='About', command=self.about)
        self.canvas1.create_window(400, 300, window=button4)

        button5 = tk.Button(self.root, text='Exit', command=self.exit)
        self.canvas1.create_window(600, 300, window=button5)
```

```
    def callback(self, sv):
        self.info_string = ""
        self.hint_label.configure(text = self.info_string)
        self.control.printall(sv.get())
        if(len(self.control.list)!=0):
            self.option_var.set(self.control.list[0][0]) # default valu
            options = ttk.Combobox(self.root, textvariable= self.option
elf.control.list[:max(len(self.control.list), 10)]], width= 50)
            options.current()
            #w = tk.OptionMenu(self.root, self.option_var, *[i[0] for i
ommand=self.option_changed)
            self.canvas1.create_window(500, 120, window=options)
            self.canvas1.pack()

            self.install_button.destroy()

            self.update_button = tk.Button(self.root, text='Update', co
            self.canvas1.create_window(300, 200, window= self.update_bu
            self.canvas1.pack()

            self.remove_button = tk.Button(self.root, text='Uninstall',
            self.canvas1.create_window(700, 200, window=self.remove_but
            self.canvas1.pack()
        else:
            self.option_var.set(['No results found'])
            self.install_button = tk.Button(self.root, text='Install',
            self.canvas1.create_window(500, 200, window=self.install_bu
            self.canvas1.pack()

            self.update_button.destroy()
            self.remove_button.destroy()



    def exit(self):
        exit(0)

    def updatesys(self):
        self.info_string = "Updating system"
        self.hint_label.configure(text = self.info_string)
```

```
    def upgrade(self):
        self.info_string = "Upgrading system"
        self.hint_label.configure(text = self.info_string)
        self.control.update()
        self.info_string = "Upgraded system"
        self.hint_label.configure(text = self.info_string)


    def remove_garbage(self):
        self.info_string = "Removing obselete..."
        self.hint_label.configure(text = self.info_string)
        self.control.auto_remove()
        self.info_string = "Removing obselete..."
        self.hint_label.configure(text = self.info_string)

    def install(self):
        self.info_string = "Installing..."
        self.hint_label.configure(text = self.info_string)
        self.control.install(self.option_var.get())
        self.info_string = "Installed"
        self.hint_label.configure(text = self.info_string)

    def remove(self):
        self.info_string = "Uninstalling..."
        self.hint_label.configure(text = self.info_string)
        self.control.remove(self.option_var.get())
        self.info_string = "Uninstalled"
        self.hint_label.configure(text = self.info_string)

    def update(self):
        self.info_string = "Updating..."
        self.hint_label.configure(text = self.info_string)
        self.control.updateApp(self.option_var.get())
        self.info_string = "Updated"
        self.hint_label.configure(text = self.info_string)

    def about(self):
        self.info_string = "Hello world! This is an application manager for Ubuntu b
        self.hint_label.configure(text = self.info_string)


if __name__ == "__main__":
    app = UI()
    app.root.mainloop()
```

## Control

```
import os
import re
import sys
import sqlite3
from db import db


def prints():
    os.system("ls")

class Control:

    def __init__(self):
        self.update_status = 0
        self.upgrade_status = 0
        self.db = db()
        self.list = []
        self.getAll()
        self.password = "12345"

    def getAll(self):
        op = os.popen("dpkg --list")
        lt = op.read()
        lt = lt.split("ii")[1:]

        final  = []
        i = 0
        for each in lt:
            final.append(re.split(r'\s{2,}', each))
            lt = re.split(r'\s{2,}', each)
            if len(lt)==5:
                self.db.insert(lt[1], lt[2], lt[3], lt[4])
                self.list.append([lt[1], lt[2], lt[3], lt[4]])
            #i+=1
            #if i>10:
            #    break
        print("successfully got all application details")

    def printall(self, string):
        self.list = self.db.get(string)


    # Method to remove given package.
    def remove(self, name):
```

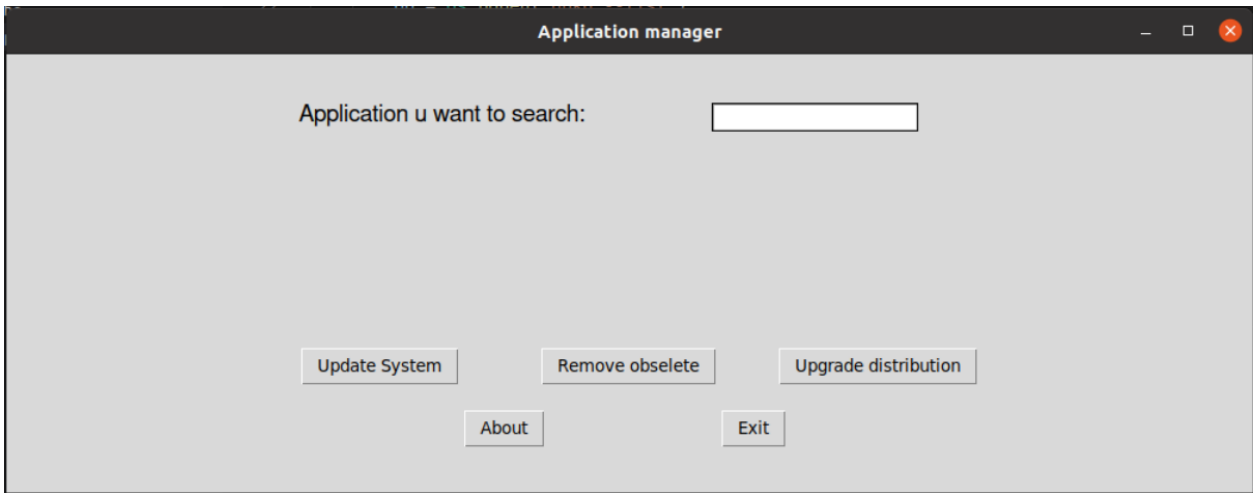
```
43      # Method to remove given package.
44      def remove(self, name):
45          command = "sudo apt-get remove " + name
46          os.system(command)
47
48      # Method to fully upgrade the system
49      def dist_upgrade(self):
50          if self.upgrade_status ==1:
51              os.system("sudo apt full-upgrade ")
52          else:
53              self.upgrade()
54              self.dist_upgrade()
55
56      # Method to remove obsolete/ non essential packages.
57      def auto_remove(self):
58          os.system("sudo apt-get autoremove")
59
60      # Method to install given package.
61      def install(self, name):
62          command = "sudo apt-get install " + name
63          if self.update_status != 1:
64              self.update()
65          os.system(command)
66
67      # To update a given application
68      def updateApp(self, name):
69          command = "sudo apt-get upgrade " + name
70          if self.update_status != 1:
71              self.update()
72          os.system(command)
73
74
75      # Method to remove given package and associated files like configuration, logs etc.
76      def sremove(self, name):
77          command = "sudo apt-get --purge remove " + name
78          os.system(command)
79
80
81      # TO upgrade all packages
82      def upgrade(self):
83          if self.upgrade_status == 1:
```

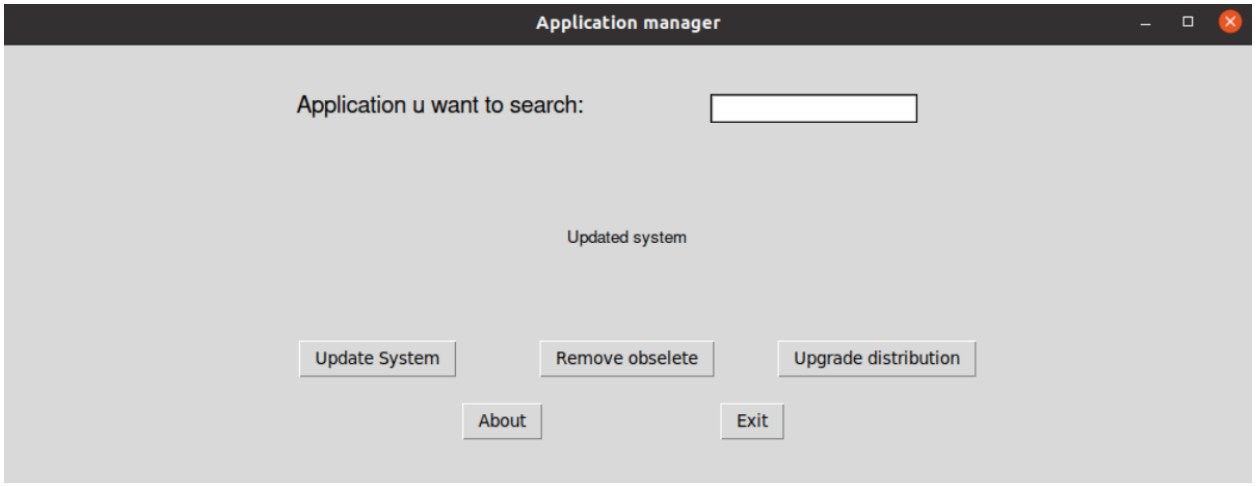
```
        if self.upgrade_status == 1:
            response = os.system("sudo apt-get upgrade ")
            if response == 0:
                self.upgrade_status = 1
            else:
                self.update()
                self.upgrade()


    # Method to update system. Equivalent to apt-get update.
    def update(self):
        response = os.system("sudo apt-get update")
        if response == 1:
            self.update_status = 1

```

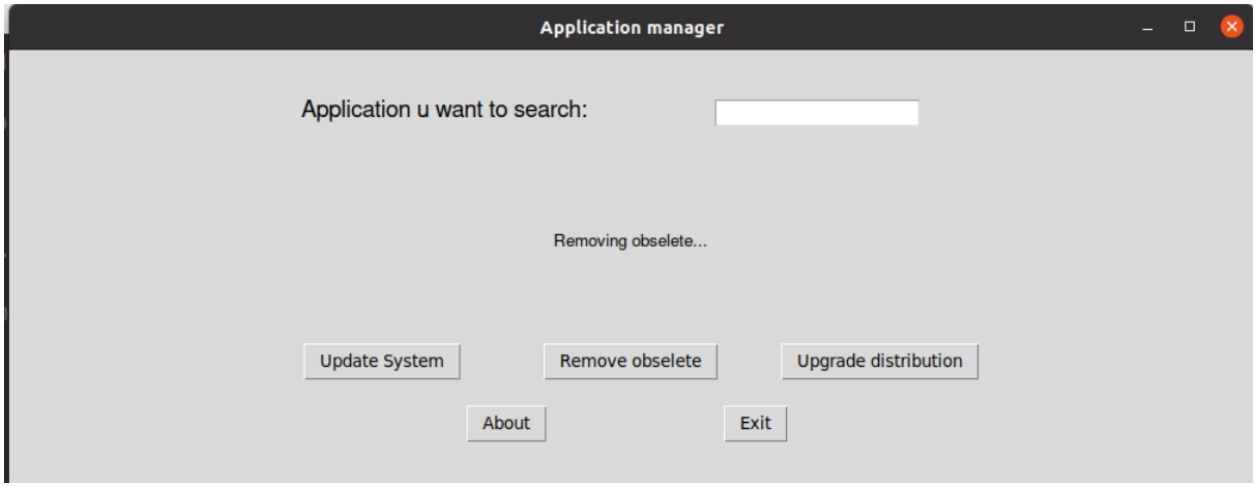
## SQL

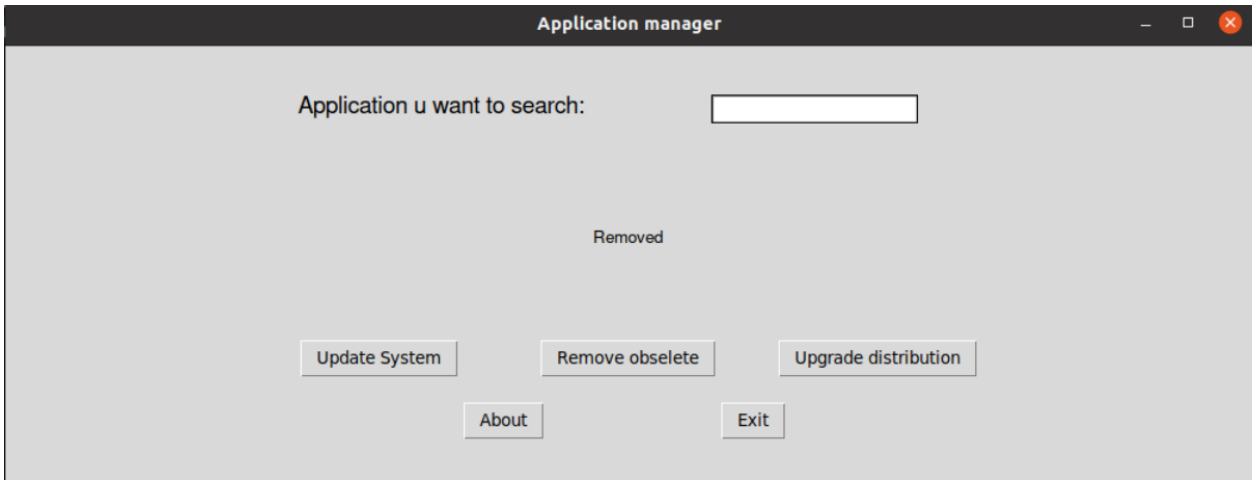
```
import sqlite3

class db:
    def __init__(self):
        self.conn = sqlite3.connect('test.db')
        self.c = self.conn.cursor()
        if(self.conn is not None):
            self.c.execute("DROP TABLE APPS")
            self.c.execute('''CREATE TABLE IF NOT EXISTS APPS
                    (NAME TEXT NOT NULL,
                    VERSION TEXT NOT NULL,
                    ARCH TEXT,
                    DESC TEXT,
                    PRIMARY KEY(NAME));''')
            self.conn.commit()
        else:
            print("Error! cannot create the database")

    def __del__(self):
        self.conn.close()

    def insert(self, name, version, arch, desc):
        self.c.execute("INSERT INTO APPS (NAME,VERSION,ARCH,DESC) VALUES (?, ?, ?, ? )", [name, version, arch, desc])
        self.conn.commit()
        pass

    def get(self, name = ""):
        lt = []
        name = "%"+name+"%"
        cursor = self.c.execute("SELECT NAME, VERSION, DESC FROM APPS WHERE NAME LIKE (?)", [name])
        for each in cursor:
            lt.append([each[0], each[1], each[2]])
        return lt

    def update(self):
        self.conn.cursor.execute("UPDATE COMPANY set SALARY = 25000.00 where ID = 1")

    def delete(self):
        self.conn.cursor.execute("DELETE from COMPANY where ID = 2;")


```

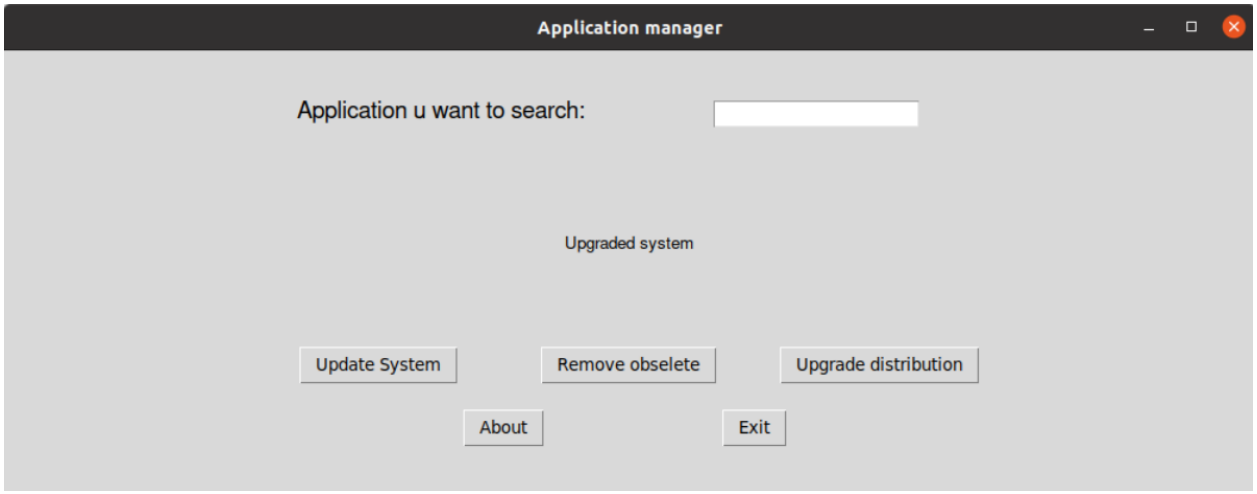
# SCREENSHOTS

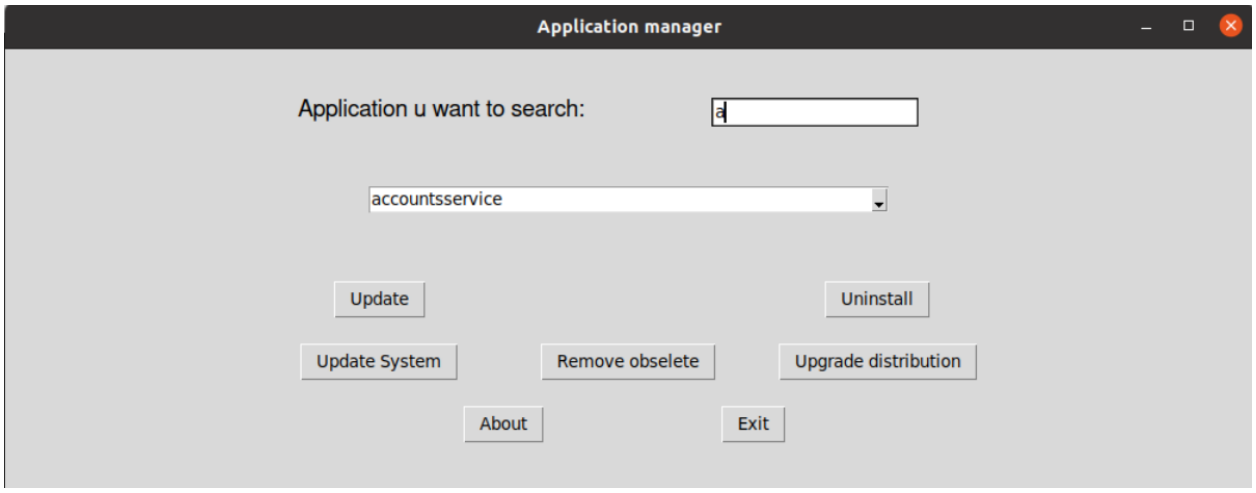


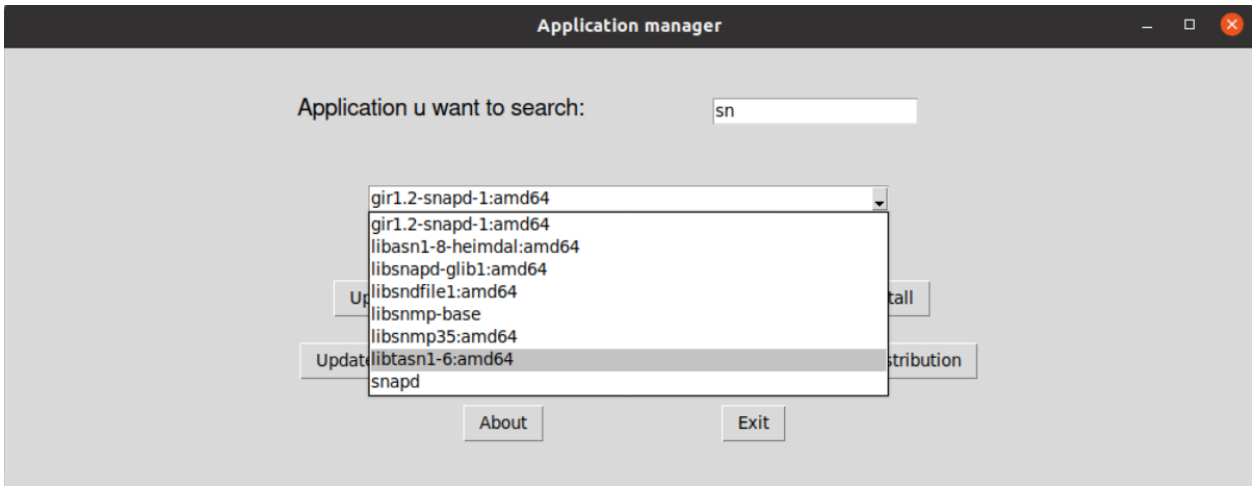



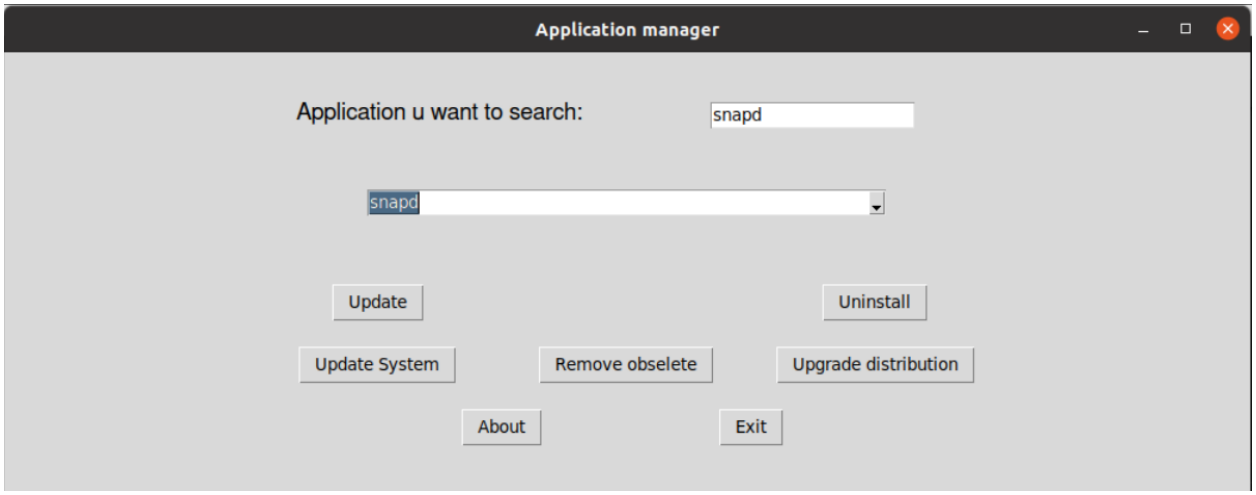



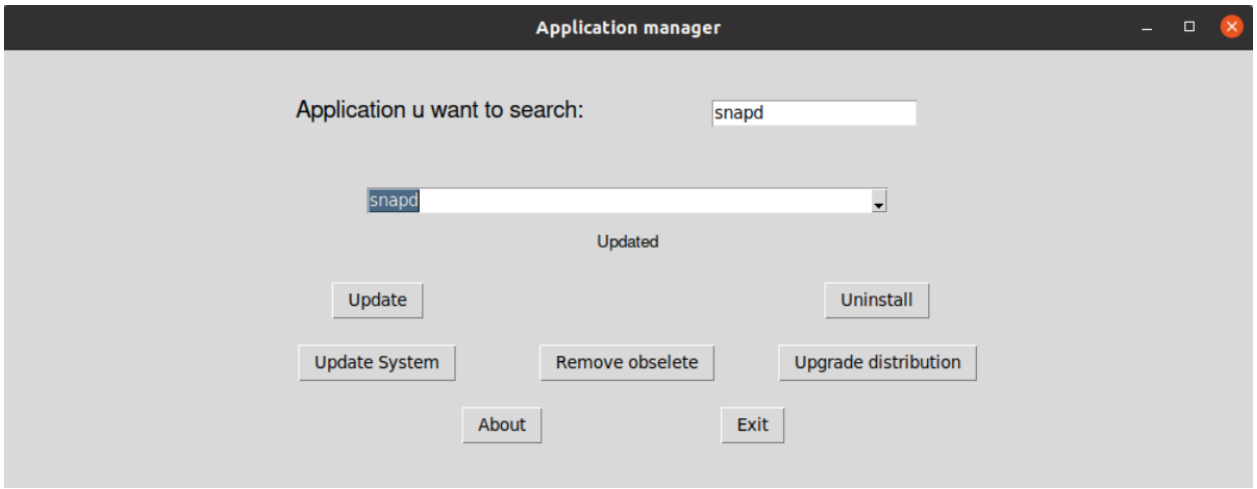

Application manager
Application u want to search:
Removed
Update System
Remove obselete
Upgrade distribution
About
Exit

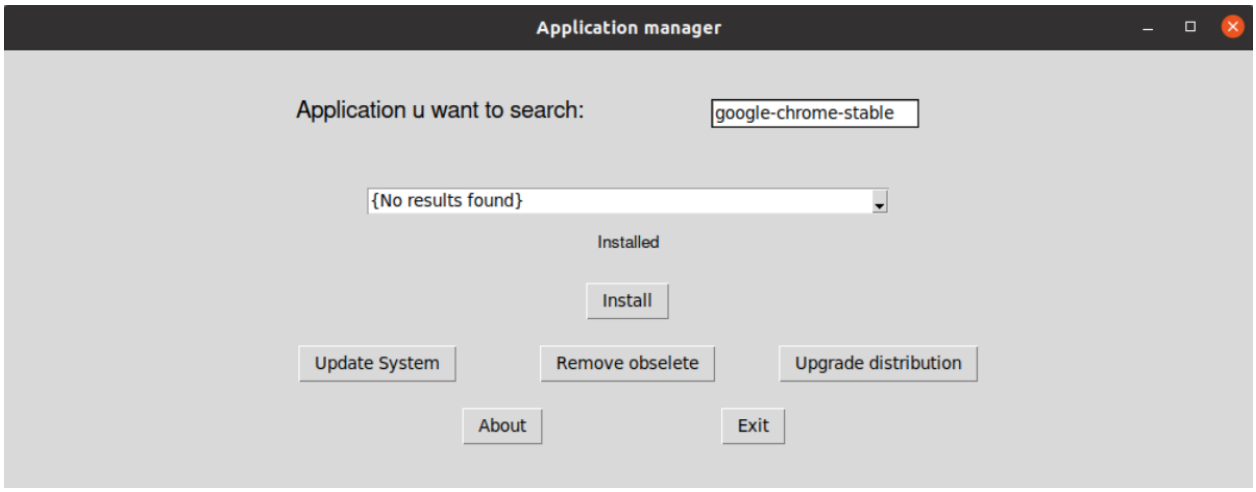

Application manager
Application u want to search:
Upgraded system
Update System
Remove obselete
Upgrade distribution
About
Exit

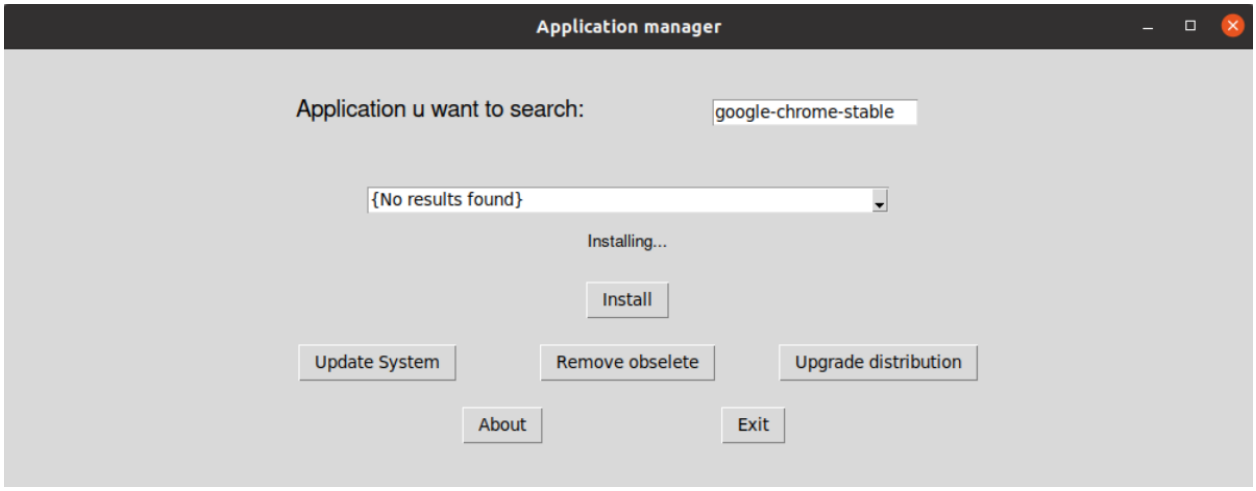

Application manager
Application u want to search:
a
accountsservice
Update
Uninstall
Update System
Remove obselete
Upgrade distribution
About
Exit

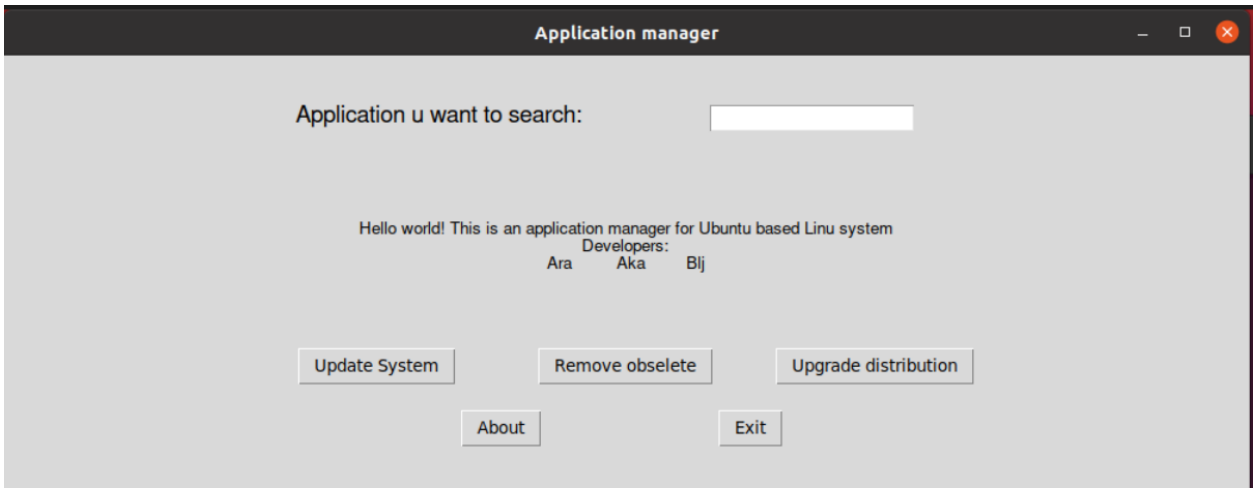

Application manager
Application u want to search:
sn
gir1.2-snapd-1:amd64
gir1.2-snapd-1:amd64
libasn1-8-heimdal:amd64
libsnapd-glib1:amd64
libsndfile1:amd64
libsnmp-base
libsnmp35:amd64
libtasn1-6:amd64
snapd
About
Exit


Application manager
Application u want to search:
snapd
snapd
Update
Uninstall
Update System
Remove obselete
Upgrade distribution
About
Exit


Application manager
Application u want to search:
snapd
snapd
Updated
Update
Uninstall
Update System
Remove obselete
Upgrade distribution
About
Exit


Application manager
Application u want to search:
google-chrome-stable
{No results found}
Installed
Install
Update System
Remove obselete
Upgrade distribution
About
Exit


Application manager
Application u want to search:
google-chrome-stable
{No results found}
Installing...
Install
Update System
Remove obselete
Upgrade distribution
About
Exit


Application manager
Application u want to search:
Hello world! This is an application manager for Ubuntu based Linu system
Developers:
Ara Aka Blj
Update System
Remove obselete
Upgrade distribution
About
Exit

# METHODOLOGY

We have got a list of applications using shell commands and updated it in SQL.We have created a user interactive front end using Python framework is Tkinter.This had options to delete,update,browse,install applications or set of applications.When the user gives in his/her command then appropriate actions are taken by the application manager.Users can remove obsolete packages that is the packages that are not used by the users.They can also upgrade their Operating Systems.

Users need not memorize long commands for updating,deleting or installing applications.Application manager helps users to use commands at a common place.

# CONCLUSION

We have implemented the listed modules using python.This application Manager eliminates the work of remembering various commands for installing,deleting,deleting and updating. So we came up with a solution of creating an application manager which can do the functions of installing , uninstalling and updating with just a click. We have also created a user-friendly GUI for the user to understand easily also because keeping it simple was our motto and we have successfully accomplished it. Basically once the user clicks install by searching a particular application the system searches for the application inside the system if it couldn't find it , the system automatically installs the desired application. Similarly for updating ,when a user gives an update they need to search the desired application in the search box and then need to click on update and finally the updated application message can be seen by the user.

# FUTURE WORK

As we have covered the basic functionalities for an application in our project, in addition to that we can also add a few other functionalities like restoring previous versions, checking for updates automatically, notifying the user regarding updates and giving a report on their past updation dates. So many other features can also be added and a good looking GUI can also be done to attract people. These are the features that can be implemented as a future work of this project.

## REFERENCES


https://www.reallinuxuser.com/linux-command-line-for-beginners-how-to-simply-find-and-install-applications/

https://github.com/luong-komorebi/Awesome-Linux-Software

https://github.com/KrispyCamel4u/SysMonTask

https://codex.cs.yale.edu/avi/os-book/

https://www.os-book.com/OS9/