**Todoist**

*A*

*Mini Project Report*

*Submitted in partial fulfilment of the*

*Requirements for the award of the Degree of*

**BACHELOR OF ENGINEERING**

**IN**

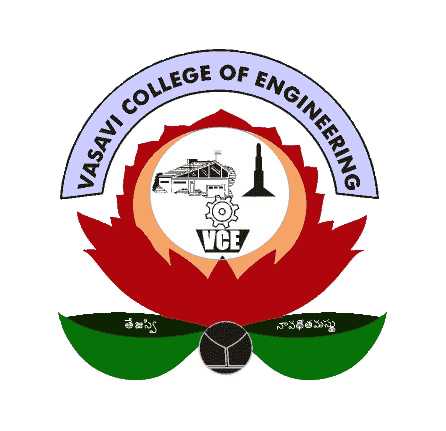
**INFORMATION TECHNOLOGY**

By

**Modagala Nikhil Narasimha 1602-20-737-025**

**Madhyanapu Pavan Kalyan 1602-20-737-027**

**Rayella Srujan Teja 1602-20-737-046**



Department of Information Technology

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE(Affiliated to Osmania University and Approved by AICTE)

Ibrahim Bagh, Hyderabad-31

2022

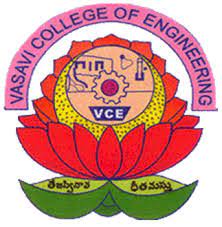
**Vasavi College of Engineering (Autonomous)**

**ACCREDITED BY NAAC WITH 'A++' GRADE**

**(Affiliated to Osmania University and Approved by AICTE)**

**Hyderabad-500 031**

**Department of Information Technology**



DECLARATION BY THE CANDIDATE

We, **M. Nikhil Narasimha, M. Pavan Kalyan, and R. Srujan Teja,**

bearing hall ticket numbers, **1602-20-737-025, 1602-20-737-027 and 1602-20-737-046,** hereby declare that the project report entitled “**Todoist**” is submitted in partial fulfilment of the requirement for the award of the **Degree Bachelor of Engineering** **in Information Technology**.

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

**M. Nikhil Narasimha**

**1602-20-737-025**

**M. Pavan Kalyan**

**1602-20-737-027**

**R. Srujan Teja**

**1602-20-737-046**

ACKNOWLEDGMENT

We extend our sincere thanks to Dr. S. V. Ramana, Principal, Vasavi College of Engineering for his encouragement.

We express our sincere gratitude to Dr. K. Ram Mohan Rao, Professor & Head, Department of Information Technology, Vasavi College of Engineering, for introducing the Mini-Project module in our curriculum, and also for his suggestions, motivation, and co-operation for the successful completion of our Mini Project.

We also want to thank and convey our gratitude towards our mini project coordinators Rajyalakshmi and L.Divya, for guiding us in understanding the process of project development & giving us timely suggestions at every phase.

We would also like to sincerely thank the project reviewers for their valuable inputs and suggestions.

**ABSTRACT**

Every day we have thousands of thoughts running in our head, we feel like to grasping and trying to remember everything we’ve thought, but thoughts seem to slip away (it’s obvious) here comes the TO-DO LIST, as soon as we write down the thoughts scrambling around our head, we feel a sense of relief, your brain stops stressing about remembering everything, we can become more organized.

One of the most important reasons for keeping a to-do list is the organization. Organizing your tasks with a list can make everything much more manageable and make you feel grounded. Seeing a clear outline of your completed and uncompleted tasks will help you feel organized and stay mentally focused.

Our To do list application helps you organize your daily tasks

It has a simple GUI build using Tkinter and with support of sqllite3

It has features like Adding Tasks, removing tasks, date adding, productivity chart …etc.

**Table of contents**

|  |  |  |
| --- | --- | --- |
| **SNO** | **Contents** | **Pg No.** |
| 1. | Abstract and information | **6** |
| 2. | Technology | **8** |
| 3. | Proposed work | **9** |
| 4. | Results | **31** |
| 5. | Conclusion and future work | **38** |
| 6. | References | **39** |

**ABSTRACT AND INFORMATION**

1. **Information about project domain in general**

*Our ToDoist Project is a GUI and Database based application*

GUI is a Graphical Interface that is a visual representation of communication presented to the user for easy interaction with the machine. GUI means Graphical User Interface. It is the common user Interface that includes Graphical representation like buttons and icons, and communication can be performed by interacting with these icons rather than the usual text-based or command-based communication.

A database is a collection of data, usually stored in electronic form. A database is typically designed so that it is easy to store and access information.

A good database is crucial to any company or organization. This is because the database stores all the pertinent details about the company such as employee records, transactional records, salary details etc. In our case to retrieve or save the previously executed data we used a database

Need of a To-Do List Application

Every day we have thousands of thoughts running in our head, we feel like to grasping and trying to remember everything we’ve thought, but thoughts seem to slip away (it’s obvious) here comes the TO-DO LIST, as soon as we write down the thoughts scrambling around our head, we feel a sense of relief, your brain stops stressing about remembering everything, we can become more organized.

Our to do list application is a productivity tool helping people enhance their productivity.

1. **List of features specific to your project[what you prioritized and wanted to achieve through this project]**
2. **User - friendly UI**
3. **Easy to use**
4. **Productivity chart**
5. **Dark theme**
6. **Adding dates**

*Our product being a productivity tool, we wanted to help people make their life.*

*Hassle free by helping them organize their part of life.*

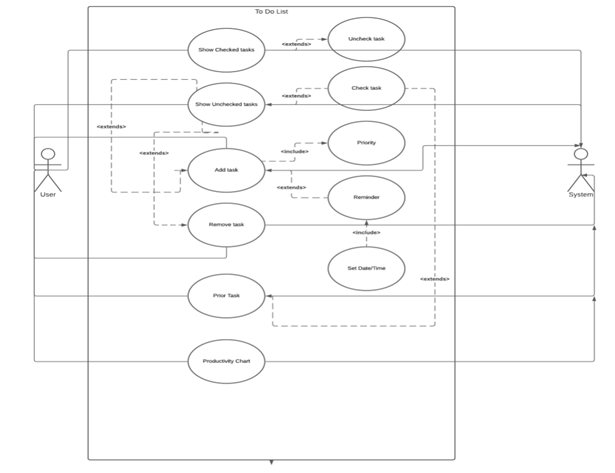
**TECHNOLOGY**

1. **Software Requirement (Our Side)**
   1. Pycharm Application with Python software pre-installed
   2. Installation of Modules like Tkinter, Numpy, Matplotlib, and Sqllite3
   3. User will need any basic Windows version like XP to run this application
2. **Hardware Requirements**
   1. Memory : 500MB
   2. Graphics : Graphic card is not mandatory
   3. CPU : Intel Core 2 Duo U7600 2.00Ghz or Above
   4. OS : XP or above

**PROPOSED WORK**

**A. Design**

1. **USE CASE DIAGRAM**



**Use Case Descriptions**

Name: Add Task

Actors: User

Description: Allows to create a new task

Pre-Conditions: None

Post-Conditions: A Task is created

|  |  |
| --- | --- |
| User | System |
| 1.Clicks the Add Task Button |  |
|  | 2.Text field appears with format |
|  | - |
| 3.User Enters the respective info if needed |  |

Name: Remove tasks

Actors: User

Description: Allows to remove/delete a task

Pre-Conditions: The task must be existing

Post-Conditions: The existing task will be deleted

|  |  |
| --- | --- |
| User | System |
| 1.Clicks the Remove task button |  |
|  | 2.The System deletes the task |

Name: Show unchecked tasks

Actors: User

Description: Displays the Unchecked tasks

Pre-Conditions: Unchecked tasks existing

Post-Conditions: The Unchecked tasks will be displayed

|  |  |
| --- | --- |
| User | System |
| 1.Clicks the Show Unchecked tasks button |  |
|  | 2.The unchecked tasks are shown  And today’s tasks are appeared(if exist) |
| 3.User can check off the tasks  Or add new tasks |  |
|  | 4.The respective operations are performed by system |

Name: Productivity Chart

Actors: User

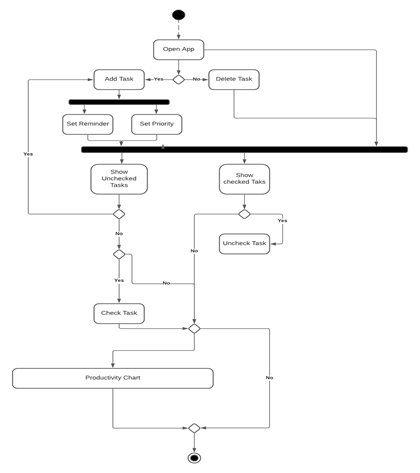
Description: Displays the Productivity chart i.e., analysis of checked vs unchecked tasks.

Pre-Conditions: No of tasks shouldn’t be zero

Post-Conditions: The graph is displayed

|  |  |
| --- | --- |
| User | System |
| 1.Clicks the Productivity Chart button |  |
| -- | 2.The Graph is displayed |

ii. Activity Diagram



**B. implementation**

1. Module wise code

**Tkinter Code**

def showtask():

if(len(l1)==0):

tkmb.showinfo(title="Tasks",message = "The Tasks List is Empty !!")

else:

top = Toplevel()

top.title("Tasks")

top.configure(bg="#d1b3ff")

lblTasks\_today = Label(top, text="Today's tasks are", font=("comicsans", 10), fg="#000000")

lblTasks\_today.place(x=150, y=0)

lblTasks= Label (top,text="The Tasks are",font=("comicsans",10), fg="#000000")

lblTasks.place(x=160,y=175)

T = Text(top, height=10, width=30)

T.place(x=90,y=200)

T\_today = Text(top, height=8, width=30)

T\_today.place(x=90, y=35)

#lblTasks.pack()

top.geometry("400x400")

ctt = 0

today\_tasks = []

#T\_today.insert(tk.END, " Today")

for i in l1:

task\_date = l1[i]

x\_dt = datetime.strptime(task\_date,"%d/%m/%y")

if(x\_dt.day==datetime.today().day and x\_dt.month==datetime.today().month and x\_dt.year==datetime.today().year):

ctt = ctt + 1

i = str(ctt) + ") " + i + " " + l1[i] + "\n"

today\_tasks.append(i)

if(len(today\_tasks)==0):

T\_today.insert(tk.END,"\n\n That's so empty in here \n \U0001F5D1")

else:

for i in today\_tasks:

T\_today.insert(tk.END, i)

ct = 0

for i in l1:

ct = ct+1

i = str(ct)+") "+i+" "+l1[i]+"\n"

T.insert(tk.END, i)

T.configure(state='disabled')

T\_today.configure(state='disabled')

def get(event):

try:

task\_Entered = AddTask\_Entry.get()

splitting = task\_Entered.split(',')

x\_dt1 = datetime.strptime(splitting[1],'%d/%m/%y')

#task\_date = date1

#print(str(x\_dt.day)+ "/"+ str(x\_dt.month) + "/"+str(x\_dt.year))

if (task\_Entered == ""):

tkmb.showwarning(title="Warning !", message="Please give a task name")

else:

top\_added = Toplevel()

top\_added.title("Task")

top\_added.configure(bg="#cce6ff")

lbladded = Label(top\_added, text="\n\n"+splitting[0].upper()+" Added !!\n\n Make time to do it !!",font=("Calibri", 12), fg="#262626",bg="#cce6ff")#,img="java.png")

lbladded.pack()

top\_added.geometry("200x180")

l1[splitting[0]] = splitting[1]

except:

tkmb.showwarning(title="Warning !", message="Please enter a valid date")

def delete(event):

task\_Rem = AddTask\_Entry.get()

if (task\_Rem == ""):

tkmb.showwarning(title="Warning !", message="Please give a task name")

else:

if (task\_Rem not in l1):

tkmb.showwarning(title="Warning !", message="Task DNE")

else:

tkmb.showinfo(title="Task ",message ="GG !! Task "+task\_Rem+" is checked off")

l2.append(task\_Rem)

del l1[task\_Rem]

def removeTask():

global RemTask

AddTask\_Entry.place(x=225,y=150)

lblenter.place(x=70,y=150)

lblenter.config(text="Enter the task to be deleted :")

#AddTask\_Priority.destroy()

AddTask\_Entry.bind('<Return>',delete)

temp=0

def addTask():

global temp

temp+=1

global lblenter

lblenter = Label(root, text=" Enter the task: ")#.place(x=25,y=150)

lblenter.place(x=65,y=150)

if(checkvar1.get()==1):

lblenter.configure(bg="#4d4d4d")

else:

lblenter.configure(bg="#cce6ff")

global AddTask\_Entry

AddTask\_Entry = tk.Entry(root, width=30)

AddTask\_Entry.place(x=180, y=150)

AddTask\_Entry.insert(END,'Task\_Name,Date(DD/MM/YY)')

AddTask\_Entry.bind('<Return>', get)

def chart():

try:

x = len(l1)

y = len(l2)

if(x ==0 and y==0):

tkmb.showinfo(title="Productivity !!", message="No tasks yet ")

else:

if(y>=x):

tkmb.showinfo(title="Productivity !!", message="GG !! Tasks Checked are more than tasks unchecked \U0001F44F")

pie = np.array([x, y])

mylbl = ["Unchecked", "Checked"]

plt.pie(pie, labels=mylbl)

plt.legend()

plt.title("Productivity pie")

plt.show()

except:

tkmb.showwarning(title="Productivity !!", message="Unknown error occurred !!")

dta=0

def database():

ans=tkmb.askyesno("Edho okati","Are you sure you want to delete the data?")

if ans==1:

global dta

dta=1

conn = sqlite3.connect('Todotasks.db')

c = conn.cursor()

c.execute('DELETE FROM todo\_list;', )

conn.commit()

conn.close()

conn2 = sqlite3.connect('Todotasksckd.db')

c2 = conn2.cursor()

c2.execute('DELETE FROM todo\_listckd;', )

conn2.commit()

conn2.close()

else:

dta=0

def showcheck():

if(len(l2)==0):

tkmb.showinfo(title="Tasks", message="The Tasks List is Empty !!")

#lblTasksckd = Label(topck,text = "The Tasks list is Empty",font=("Calibri", 10), fg="#595959")

#lblTasksckd.place(x=43,y=57)

#topck.geometry("200x200")

else:

topck = Toplevel()

topck.title("Tasks")

topck.configure(bg="#d1b3ff")

lblTasksckd = Label (topck,text="The Checked tasks are",font=("comicsans",10))

lblTasksckd.pack()

Tckd = Text(topck, height=20, width=30)

Tckd.place(x=90,y=60)

lblTasksckd.pack()

topck.geometry("400x400")

ct = 0

for i in l2:

ct = ct+1

i = str(ct)+") "+i+"\n"

Tckd.insert(tk.END, i)

Tckd.configure(state='disabled')

try:

button\_add = tk.Button(root, text="Add Task", width=20, command=addTask, bg="#d9d9d9") # ,font=myFont)

button\_add.place(x=170, y=200)

button\_Remove = tk.Button(root, text="Remove Task/Check Task", width=20, command=removeTask, bg="#d9d9d9")

button\_Remove.place(x=170, y=250)

button\_ShowUn = tk.Button(root, text="Show Checked tasks", width=20, bg="#d9d9d9", command=showcheck)

button\_ShowUn.place(x=170, y=400)

button\_ShowTask = tk.Button(root, text="Show Tasks", width=20, bg="#d9d9d9", command=showtask)

button\_ShowTask.place(x=170, y=300)

button\_ShowChart = tk.Button(root, text="Productivity Chart", width=20, bg="#d9d9d9", command=chart)

button\_ShowChart.place(x=170, y=350)

button\_data = tk.Button(root, text="Clear Database", width=11, bg="#cce6ff", command=database)

button\_data.place(x=400, y=400)

except:

tkmb.showwarning(title="Warning ",message="Unknown error occurred")

def toggle():

global lblenter

if(checkvar1.get()==1):

root.configure(bg="#4d4d4d")

if temp>0:

lblenter.configure(bg="#4d4d4d",fg="#e6e6e6")

lblTodo.configure(bg="#4d4d4d",fg="#e6e6e6")

chkbtn1.configure(bg="#4d4d4d",fg="#e6e6e6")

button\_data.config(bg="#4d4d4d",fg="#f2f2f2")

elif(checkvar1.get()==0):

root.configure(bg = "#cce6ff")

if temp>0:

lblenter.configure(fg = "#404040", bg = "#cce6ff")

lblTodo.configure(fg = "#404040", bg = "#cce6ff")

chkbtn1.configure(fg = "#404040", bg = "#cce6ff")

button\_data.config(bg = "#cce6ff",fg="#262626")

global checkvar1

checkvar1 = IntVar()

global chkbtn1

chkbtn1 = Checkbutton(root, text="Dark", variable=checkvar1, onvalue=1, offvalue=0, height=2, width=2,command=toggle)

chkbtn1.place(x=420,y=70)

chkbtn1.configure(fg = "#404040", bg = "#cce6ff")

**Matplotlib**

def chart():

try:

x = len(l1)

y = len(l2)

if(x ==0 and y==0):

tkmb.showinfo(title="Productivity !!", message="No tasks yet ")

else:

if(y>=x):

tkmb.showinfo(title="Productivity !!", message="GG !! Tasks Checked are more than tasks unchecked \U0001F44F")

pie = np.array([x, y])

mylbl = ["Unchecked", "Checked"]

plt.pie(pie, labels=mylbl)

plt.legend()

plt.title("Productivity pie")

plt.show()

except:

tkmb.showwarning(title="Productivity !!", message="Unknown error occurred !!")

**Database**

conn = sqlite3.connect('Todotasks.db')

c = conn.cursor()

c.execute("""CREATE TABLE if not exists todo\_list(list\_item CHAR(30),task\_date CHAR(30));""")

conn.commit()

conn.close()

conn2 = sqlite3.connect('Todotasksckd.db')

c2 = conn2.cursor()

c2.execute("""CREATE TABLE if not exists todo\_listckd(list\_itemckd CHAR(30));""")

conn2.commit()

conn2.close()

conn = sqlite3.connect('Todotasks.db')

c = conn.cursor()

c.execute("SELECT list\_item FROM todo\_list")

records = c.fetchall()

c.execute("SELECT task\_date FROM todo\_list")

records\_date=c.fetchall()

conn.commit()

conn.close()

for i in range(len(records)):

data1 = records[i]

data2 = records\_date[i]

frm1 = data1[0]

frm2 = data2[0]

l1[frm1] =frm2

conn2 = sqlite3.connect('Todotasksckd.db')

c2 = conn2.cursor()

c2.execute("SELECT \* FROM todo\_listckd")

records\_ckd = c2.fetchall()

conn2.commit()

conn2.close()

for i in records\_ckd:

l2.append(i[0])

conn = sqlite3.connect('Todotasks.db')

c = conn.cursor()

c.execute('DELETE FROM todo\_list;', )

items = l1

conn2 = sqlite3.connect('Todotasksckd.db')

c2 = conn2.cursor()

c2.execute('DELETE FROM todo\_listckd;', )

items\_ckd = l2

if(dta==0):

for item in items:

item = str(item)

item\_dt = str(items[item])

c.execute("""INSERT INTO todo\_list VALUES (:item,:item\_dt)""",

{

'item': str(item),

'item\_dt': str(item\_dt)

})

conn.commit()

conn.close()

for itemckd in items\_ckd:

itemckd = str(itemckd)

c2.execute("""INSERT INTO todo\_listckd VALUES (:itemckd)""",

{

'itemckd':str(itemckd)

})

conn2.commit()

conn2.close()

GITHUBLINK: https://github.com/Nikhil1357/Mini-Project\_Todoist

**c) Testing**

Add Task: **Graphical user interface

Description automatically generatedGraphical user interface, application, Word

Description automatically generated**

Graphical user interface, application

Description automatically generated

Remove Task:

Graphical user interface

Description automatically generated

Graphical user interface, website

Description automatically generated

Graphical user interface, website

Description automatically generated

Show unchecked tasks:

**Graphical user interface

Description automatically generated**

Text

Description automatically generated

A picture containing text

Description automatically generated

Show checked tasks:

**Graphical user interface

Description automatically generated**A picture containing graphical user interface

Description automatically generated

Productivity Chart:

Graphical user interface, application

Description automatically generated

Chart, pie chart

Description automatically generated

**RESULTS**

**Graphical user interface

Description automatically generated**

**Graphical user interface

Description automatically generated**

Main Window After Clicking Show Tasks

**Graphical user interface

Description automatically generated**

**Graphical user interface

Description automatically generated**

After Clicking Add Task After toggling Dark button

A picture containing text

Description automatically generatedGraphical user interface, application, Word

Description automatically generated

After entering a task After clicking on show tasks

Text

Description automatically generatedGraphical user interface, application, Word

Description automatically generated

Adding a new task After clicking on show tasks

Graphical user interface

Description automatically generatedGraphical user interface, application

Description automatically generated

Error occurred while giving an invalid input Main Window

A picture containing graphical user interface

Description automatically generatedGraphical user interface, website

Description automatically generated

dark themed window removing/checking off task

Graphical user interface, application

Description automatically generatedGraphical user interface, website

Description automatically generated

Trying to delete a task which does not exist Adding a new task

Graphical user interface, application

Description automatically generatedText

Description automatically generated

Viewing tasks removing task

A picture containing graphical user interface

Description automatically generated

Graphical user interface, application

Description automatically generated

After clicking show checked tasks After clicking Productivity Chart

Chart, pie chart

Description automatically generated

A graph image is shown

Graphical user interface

Description automatically generated

After clicking on Clear Database

**ADDITIONAL KNOWLEDGE GAINED**

One of the best ways to get into programming is by doing projects. Projects can be one of the most efficient ways to learn as they force you to apply the skills while learning them. This, in fact, aids in retention and increases the usefulness of the skills learned.

We in our To do list project have learned much more than what our college has taught in college we had used 4/5 very influential modules (Like Numpy, Maplotlib...), other than the one’s taught in the college.

We’ve learnt about constructing Graphical User Interfaces(GUI). The GUI will make user interaction with application simpler and elegant. We implemented GUI using Tkinter.

We’ve learnt Tkinter and Sqlite3 which we applied thereafter in the project, using Tkinter we were able to learn the GUI implementation in python, we learned about different widgets, events, design ...etc, which made our user interaction more elegant and easier. Using Sqllite3 we were able to store and retrieve data in a faster way and made sure the previous data is not washed out.

By doing this project we were able to explore new domains like GUI and Database

**Conclusion and future work**

* In this Mini Project we have successfully implemented a To-do List application using Python and its Libraries Tkinter and Sqllite3 especially
* Now a days we are bombarded with lots of tasks to do, we often get lost and forgot most of the things we had to in the first place, so using a ToDolist will enhance productivity and help you organize better
* In our application the user can enter the tasks, delete the tasks, add a respective date to view daily tasks, view the productivity chart…etc.
* The tasks entered will remain forever even though the application is killed and ran again.

### Everything in our project performed as planned and our team meets the goals outlined in the description. We designed a To Do List using python.

### We are highly indebted to our faculty who continuously supported us throughout the journey.

### Overall, this project helped our programming skills a lot and introduces us to the real world projects we could face in our careers.

**Future Work**

### We would like to add a login system, Calendar which shows tasks of respective days, notifications, notes, recurring tasks, and develop the GUI to the newer levels

**REFERENCES**

* [**Python Tkinter Docs**](https://docs.python.org/3/library/tk.html)
* [**GUI in Python, Tkinter by FreeCodeCamp**](https://www.youtube.com/watch?v=YXPyB4XeYLA)
* [**Python Tkinter tutorial - Geeks for Geeks**](https://www.geeksforgeeks.org/python-gui-tkinter/)