AMITY INTERNATIONAL SCHOOL
VASUNDHARA SECTOR - 6

Computer Science Project

'NoterPy: Notes & Tasks Management System'

Name: Devansh Singh

Class: XII – B

Roll Number:

INDEX

CERTIFICATE

This is to certify that Devansh Singh of class XII B, Amity International School, Vasundhara Sector - 6, roll number – has successfully completed his project in computer practical for the AISSCE as prescribed by CBSE in the academic year 2020 – 2021.

External Examiner **Internal Examiner**

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher Ms. Halina Gupta as well as our principal Ms. Sunila Athley who gave me the golden opportunity to do this wonderful project, which also helped me in doing a lot of research and I got to learn a lot of new things, I am really thankful to them.

Secondly, I would like to thank my parents who helped me a lot in finalising this project within the limited time frame.

ABOUT PYTHON

Python is an interpreted, object-oriented, high level programming language with dynamic semantics. It's high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for rapid application development.

Python is simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

REQUIREMENTS

- Processor:
  Intel Core i5 4300M/ Ryzen 5 3550 H (4 cores)

- RAM:
  8GB DDR4 RAM

- Disk Space:
  4 GB+, SSD preferred

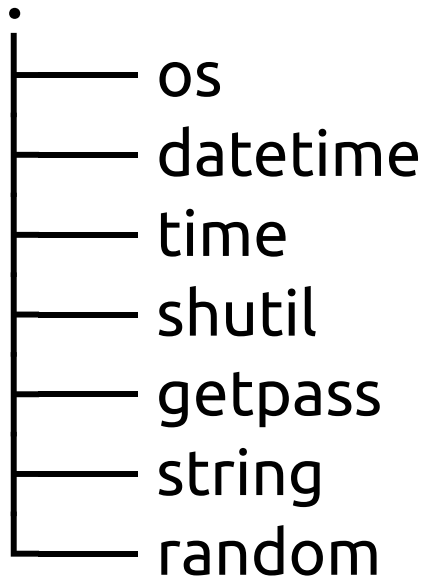
- Operating Systems:
  - Windows 7/8/10
  - Linux Distros
  - macOS

- Python 3.8 (or Later)

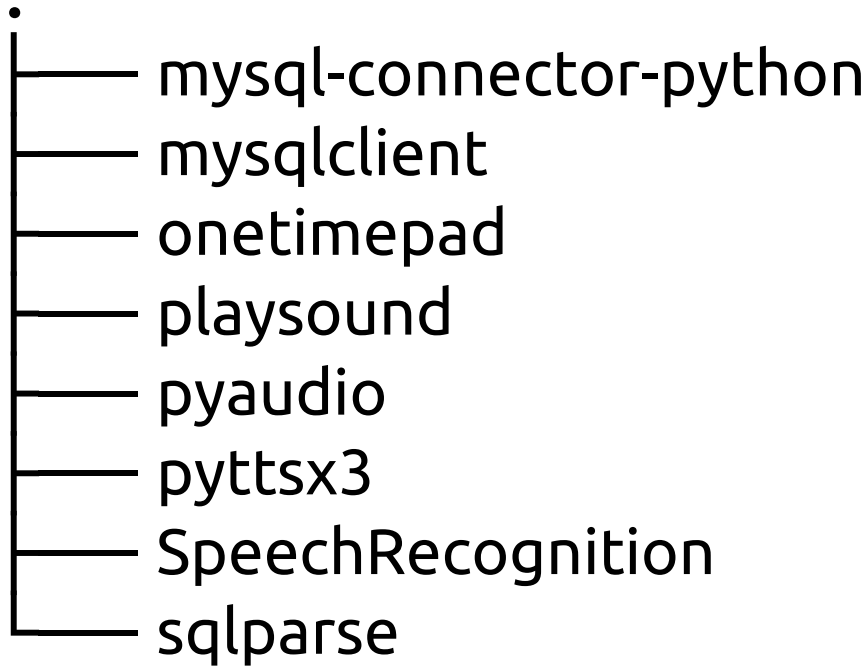
- MySQL setup

MODULES USED

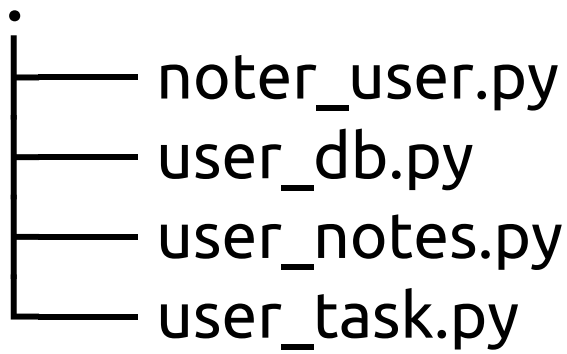
In-built Modules



External Modules



Self-Made Modules



- noter_user.py
  module for maintaining user's information

- user_db.py
  module for maintaining database in MySQL for a user

- user_notes.py
  module for maintaining notes of a user, in the directory notes

- user_task.py
  module for maintaining tasks in the to-do-list of a user, in the directory to-do-list

SOURCE CODE

- **noter.py**
  (main executable file)

```
import noter_user as nu
import user_task as ut
import user_notes as un
import time
import datetime
import getpass

ASCII = '''
  _   _       _              _____
 | \ | |     | |            |  __ \
 |  \| | ___ | |_ ___ _ __| |__) |   _
 | . ` |/ _ \| __/ _ \ '__|  ___/ | | |
 | |\  | (_) | ||  __/ |  | |   | |_| |
 |_| \_|\___/ \__\___|_|  |_|    \__, |
                                   __/ |
                                  |___/
'''

main_menu = '''
+--------------------+
|Main Menu           |
+--------------------+
|1. My to-do list    |
|2. My notes         |
|3. Settings         |
|4. About            |
|5. Exit             |
+--------------------+
'''

task_menu = '''
+---------------------------+
|To-do List                 |
+---------------------------+
|1. Show to-do list         |
|2. Add a task              |
|3. Add a task by speaking  |
|4. Remove a task           |
|5. Update a task           |
|6. Speak the to-do list    |
|7. Return to main menu     |
+---------------------------+
'''
```

```
note_menu = '''
+---------------------------+
|Notes                      |
+---------------------------+
|1. Show list of notes      |
|2. Show a note             |
|3. Add a note              |
|4. Add a note by speaking  |
|5. Remove a note           |
|6. Update a note           |
|7. Speak a note            |
|8. Return to main menu     |
+---------------------------+
'''

settings_menu = '''
+-------------------------+
|Settings                 |
+-------------------------+
|1. Change user name      |
|2. Change user password  |
|3. Remove user           |
|4. Return to main menu   |
+-------------------------+
'''

print(ASCII)
time.sleep(1)
current_date = datetime.datetime.now()
print(current_date.strftime("%A, %d %B %Y"))
print()

flag = False
name = ''

while True:
	username = input('Enter name: ')
	password = getpass.getpass(prompt = 'Password: ')
	print()

	if nu.User.check(username, password) == True:
		print(f'Hello, {username}\nHope you\'re having a nice
day!\n')
		flag = True
		name = username
		break
	elif nu.User.check(username, password) == "wrong pass":
		user_prompt = input("Wrong password, terminating
program\n")
		break
	else:
		user_prompt = input('Name not found in record, new user?
Y/N: ')
		print()

		if user_prompt.lower() == 'y':
			if nu.User.new(username, password) == False:
```

```
                                print('Username already exists, try
another name\n')
                        else:
                                print(f'Welcome, {username}\n')
                                flag = True
                                name = username
                                break
                elif user_prompt.lower() == 'n':
                        print('Thankyou for using NoterPy :)\n')
                        break
                else:
                        print('Choose a valid option\n')

if flag == True:
        while True:
                print(main_menu)
                print()
                user_prompt = input('Enter your choice: ')
                print()

                if user_prompt == '1':
                        while True:
                                print(task_menu)
                                print()
                                user_prompt_list = input('Enter your
choice: ')
                                print()

                                if user_prompt_list == '1':
                                        ut.Task.show(name)
                                elif user_prompt_list == '2':
                                        task = input('Enter the task: ')
                                        ut.Task.add(name, task)
                                elif user_prompt_list == '3':
                                        print('The next 10 seconds audio
will be taken in as input\n')
                                        task = ut.Speech.speechToTask()

                                        if task != False:
                                                print('Did you say: ' + task
+ '?' + '\n')
                                                user_input = input('Y/N: ')

                                                if user_input.lower() ==
'y':
                                                        ut.Task.add(name,
task)
                                                else:
                                                        print('Returning
back to main menu\n')
                                                        break
                                        else:
                                                print('No audio found\n')
                                elif user_prompt_list == '4':
                                        number = input('Enter task number:
')
                                        ut.Task.remove(name, number)
                                elif user_prompt_list == '5':
```

```
                                        number = input('Enter task number:
')
                                        task = input('Enter the new task:
')
                                        ut.Task.update(name, number, task)
                                elif user_prompt_list == '6':
                                        ut.Speak.taskToSpeech(name)
                                elif user_prompt_list == '7':
                                        break
                                else:
                                        print('Choose a valid option')
                                print()

                        elif user_prompt == '2':
                                while True:
                                        print(note_menu)
                                        print()
                                        user_prompt_list = input('Enter your
choice: ')
                                        print()

                                        if user_prompt_list == '1':
                                                un.Notes.showList(name)
                                        elif user_prompt_list == '2':
                                                name_of_note = input('Enter name of
note: ')
                                                un.Notes.show(name, name_of_note)
                                        elif user_prompt_list == '3':
                                                name_of_note = input('Enter name of
note: ')
                                                content = input('Enter content: ')
                                                un.Notes.add(name, name_of_note,
content)
                                        elif user_prompt_list == '4':
                                                name_of_note = input('Enter name of
note: ')
                                                print('The next 30 seconds will be
taken in as input\n')
                                                content = un.Speech.speechToNote()

                                                if content != False:
                                                        print('Did you say: ' +
content + '?' + '\n')
                                                        user_input = input('Y/N: ')

                                                        if user_input.lower() ==
'y':
                                                                un.Notes.add(name,
name_of_note, content)
                                                        else:
                                                                print('Returning
back to main menu\n')
                                                                break
                                                else:
                                                        print('No audio found\n')
                                        elif user_prompt_list == '5':
                                                name_of_note = input('Enter name of
note: ')
```

```
                                        un.Notes.remove(name, name_of_note)

                                elif user_prompt_list == '6':
                                        name_of_note = input('Enter name of
note: ')
                                        content = input('Enter content to
be updated: ')
                                        un.Notes.update(name, name_of_note,
content)
                                elif user_prompt_list == '7':
                                        name_of_note = input('Enter name of
note: ')
                                        un.Speak.noteToSpeech(name,
name_of_note)
                                elif user_prompt_list == '8':
                                        break

                                else:
                                        print('Choose a valid option\n')
                                print()

                        elif user_prompt == '3':
                                while True:
                                        print(settings_menu)
                                        print()
                                        user_prompt_list = input('Enter your
choice: ')
                                        print()

                                        if user_prompt_list == '1':
                                                new_name = input('Enter the new
name of user: ')
                                                password = getpass.getpass(prompt =
"Password: ")
                                                if nu.User.update(name, new_name,
password) == False:
                                                        print('Username already
exists, try another name\n')
                                                else:
                                                        print('User name changed
successfully\n')
                                        elif user_prompt_list == '2':
                                                password = getpass.getpass(prompt =
"Current Password: ")
                                                new_password =
getpass.getpass(prompt = "New Password: ")
                                                nu.User.change_pass(name, password,
new_password)
                                        elif user_prompt_list == '3':
                                                con = input('All user data will be
deleted. Do you want to continue? Y/N: ')

                                                if con.lower() == 'y':
                                                        password =
getpass.getpass(prompt = "Password: ")
                                                        nu.User.delete(name,
password)
                                                elif con.lower() == 'n':
```

```
                                        print('Returning to main
menu')
                                        break
                                else:
                                        print('Choose a valid
option\n')
                        elif user_prompt_list == '4':
                                break
                        else:
                                print('Choose a valid option\n')
                        print()

                elif user_prompt == '4':
                        print('Thankyou for using NoterPy :)\n')
                        break

                else:
                        print('Choose a valid option\n')
```

- noter_user.py

```
import os
import user_db as udb
import shutil

class User:

	def check(name, password):
		if udb.User.check(name, password) == True:
			return True
		elif udb.User.check(name, password) == "wrong pass":
			return "Wrong Password"
		return False

	def new(name, password):
		if udb.User.insert(name, password) == False:
			return False
		else:
			try:
				os.system(f'cmd /c "cd notes & mkdir
{name}"')
				file = open('./to-do-
list/{}.txt'.format(name), 'a')
				file.close()
			except:
				pass
			return True

	def delete(name, password):
		if User.check(name, password) == False:
			print('User not found\n')
		elif User.check(name, password) == "Wrong Password":
			print("Wrong Password\n")
		else:
			try:
				shutil.rmtree(f'./notes/{name}')
				os.remove(f'./to-do-list/{name}.txt')
			except:
				pass
			udb.User.remove(name)
			print('User removed successfully\nTerminating
Program\n')
			exit()

	def update(old_name, new_name, password):
		if User.check(old_name, password) == False:
			print('User not found\n')
		elif User.check(old_name, password) == "Wrong Password":
			print("Wrong Password\n")
		else:
			if udb.User.update(old_name, new_name) == False:
				return False
			else:
				try:
```

```
                                        os.rename(f'./notes/{old_name}',
f'./notes/{new_name}')
                                        os.rename(f'./to-do-
list/{old_name}.txt', f'./to-do-list/{new_name}.txt')
                                except:
                                        pass
                                return True

        def change_pass(name, old_password, new_password):
                if User.check(name, old_password) == False:
                        print('User not found\n')
                elif User.check(name, old_password) == "Wrong Password":
                        print("Wrong Password\n")
                else:
                        udb.User.change_password(name, new_password)
                        print('Password changed succesfully\n')
```

• user_db.py

```
from datetime import datetime
import random
import string
import mysql.connector as mc

connectMySQL = mc.connect(host='localhost', user='root',
password='root')
cursor = connectMySQL.cursor(buffered = True)
cursor.execute('create database IF NOT EXISTS noterpy')
cursor.execute('use noterpy')
cursor.execute('create table IF NOT EXISTS users(name varchar(30),
password varchar(30), cryptkey varchar(10))')

class User:

    def generate_pass():
           password_char = string.ascii_letters + string.digits +
string.punctuation
           password = ''
           for i in range (10):
                  password += random.choice(password_char)
           return password

    def show_all_users():
           cursor.execute('select name from users')
           return cursor.fetchall()

    def check(name, password):
           sql = f'select password from users where name = "{name}"'
           cursor.execute(sql)
           result = cursor.fetchone()
           if result == None:
                  return False
           elif result[0] == password:
                  return True
           elif result[0] != password:
                  return "wrong pass"
           return False

    def insert(name, password):
           result = User.show_all_users()
           for i in result:
                  if i[0] == name:
                         return False

           cryptkey = User.generate_pass()
           sql = f'insert into users values ("{name}", "{password}",
"{cryptkey}")'
           cursor.execute(sql)
           cursor.execute(f'create table IF NOT EXISTS {name}(date
varchar(10), log varchar(100), time varchar(10))')

           date = datetime.now().strftime("%d/%m/%Y")
```

```
            time = datetime.now().strftime("%X")
            cursor.execute(f'insert into {name} values ("{date}",
"new user {name} created", "{time}")')
            connectMySQL.commit()
            return True

    def remove(name):
            sql = f'delete from users where name="{name}"'
            cursor.execute(sql)
            connectMySQL.commit()

            date = datetime.now().strftime("%d/%m/%Y")
            time = datetime.now().strftime("%X")
            cursor.execute(f'insert into {name} values ("{date}",
"removed user {name}", "{time}")')

            file = open(f'./logs/{name}.txt', 'a')
            cursor.execute(f'select * from {name}')
            result = cursor.fetchall()
            file.write(str(result))
            file.close()

            sql = f'drop table {name}'
            cursor.execute(sql)
            connectMySQL.commit()

    def update(old_name, new_name):
            result = User.show_all_users()
            for i in result:
                    if i[0] == new_name:
                            return False
            try:
                    sql = f'update users set name="{new_name}" where
name="{old_name}"'
                    cursor.execute(sql)

                    sql = f'rename table {old_name} to {new_name}'
#update user name in user's logs table
                    cursor.execute(sql)

                    date = datetime.now().strftime("%d/%m/%Y")
                    time = datetime.now().strftime("%X")
                    cursor.execute(f'insert into {new_name} values
("{date}", "updated username from {old_name} to {new_name}",
"{time}")')
                    connectMySQL.commit()
            except:
                    pass
            return True

    def change_password(name, new_password):
            sql = f'update users set password="{new_password}" where
name = "{name}"'
            cursor.execute(sql)
            date = datetime.now().strftime("%d/%m/%Y")
            time = datetime.now().strftime("%X")
            cursor.execute(f'insert into {name} values ("{date}",
"updated password", "{time}")')
```

```
            connectMySQL.commit()

    def crypt_key(name):
            sql = f'select cryptkey from users where name = "{name}"'
            cursor.execute(sql)
            key = cursor.fetchone()
            return key[0]

class Logs:

    def add_note(name, name_of_note):
            date = datetime.now().strftime("%d/%m/%Y")
            time = datetime.now().strftime("%X")
            cursor.execute(f'insert into {name} values ("{date}",
"added a new note {name_of_note}", "{time}")')
            connectMySQL.commit()

    def update_note(name, name_of_note):
            date = datetime.now().strftime("%d/%m/%Y")
            time = datetime.now().strftime("%X")
            cursor.execute(f'insert into {name} values ("{date}",
"updated note {name_of_note}", "{time}")')
            connectMySQL.commit()

    def delete_note(name, name_of_note):
            date = datetime.now().strftime("%d/%m/%Y")
            time = datetime.now().strftime("%X")
            cursor.execute(f'insert into {name} values ("{date}",
"deleted note {name_of_note}", "{time}")')
            connectMySQL.commit()

    def add_task(name):
            date = datetime.now().strftime("%d/%m/%Y")
            time = datetime.now().strftime("%X")
            cursor.execute(f'insert into {name} values ("{date}",
"added a new task", "{time}")')
            connectMySQL.commit()

    def update_task(name, number):
            date = datetime.now().strftime("%d/%m/%Y")
            time = datetime.now().strftime("%X")
            cursor.execute(f'insert into {name} values ("{date}",
"updated task number {number}", "{time}")')
            connectMySQL.commit()

    def delete_task(name, number):
            date = datetime.now().strftime("%d/%m/%Y")
            time = datetime.now().strftime("%X")
            cursor.execute(f'insert into {name} values ("{date}",
"deleted task number {number}", "{time}")')
            connectMySQL.commit()
```

• user_notes.py

```
import os
import onetimepad as ot
import pyttsx3
import speech_recognition as sr
import user_db as udb


class Notes:

    def checkFolder(name):
           return os.path.isdir('./notes/{}'.format(name))

    def showList(name):
           if Notes.checkFolder(name) == False:
                  os.system(f'cmd /c "cd notes & mkdir {name}"')
                  print('You have no notes!\n')
           elif os.listdir('./notes/{}'.format(name)) == []:
                  print('You have no notes!\n')
           else:
                  obj = os.listdir('./notes/{}'.format(name))
                  for i in range (len(obj)):
                         print('---> ' + str(i + 1) + ". " +
obj[i][:-4])

    def show(name, name_of_note):
           if os.path.isfile('./notes/{}/{}.txt'.format(name,
name_of_note)) == False:
                  print('Note not found\n')
           elif os.listdir('./notes/{}'.format(name)) == []:
                  print('You have no notes!\n')
           else:
                  file = open('./notes/{}/{}.txt'.format(name,
name_of_note), 'r')
                  password = udb.User.crypt_key(name)
                  obj = file.read().strip()
                  obj = ot.decrypt(obj, password)
                  print(obj)

    def add(name, name_of_note, content):
           if Notes.checkFolder(name) == False:
                  os.system(f'cmd /c "cd notes & mkdir {name}"')
           file = open('./notes/{}/{}.txt'.format(name,
name_of_note), 'a')
           password = udb.User.crypt_key(name)
           content = ot.encrypt(content, password)
           file.write(content+'\n')
           file.close()
           udb.Logs.add_note(name, name_of_note)
           print('Note made succesfully\n')

    def remove(name, name_of_note):
           if os.path.isfile('./notes/{}/{}.txt'.format(name,
name_of_note)) == False:
```

```
                print('Note not found\n')
            elif os.listdir('./notes/{}'.format(name)) == []:
                print('You have no notes\n')
            else:
                os.remove('./notes/{}/{}.txt'.format(name,
name_of_note))
                udb.Logs.delete_note(name, name_of_note)
                print('Note removed succesfully\n')

    def update(name, name_of_note, content):
            if os.path.isfile('./notes/{}/{}.txt'.format(name,
name_of_note)) == False:
                print('Note not found\n')
            elif os.listdir('./notes/{}'.format(name)) == []:
                print('You have no notes\n')
            else:
                file = open('./notes/{}/{}.txt'.format(name,
name_of_note), 'w')
                password = udb.User.crypt_key(name)
                content = ot.encrypt(content, password)
                file.write(content+'\n')
                file.close()
                udb.Logs.update_note(name, name_of_note)
                print('Note updated succesully\n')

class Speak:

    def noteToSpeech(name, name_of_note):
            engine = pyttsx3.init()
            engine.setProperty('rate', 150)

            if os.path.isfile('./notes/{}/{}.txt'.format(name,
name_of_note)) == False: #if task list is empty
                engine.say('Note not found!')
                engine.runAndWait()
            else:
                file = open('./notes/{}/{}.txt'.format(name,
name_of_note), 'r')
                obj = file.read().strip()
                password = udb.User.crypt_key(name)
                obj = ot.decrypt(obj, password)
                engine.say(obj)
                engine.runAndWait()

class Speech:

    def speechToNote():
            r = sr.Recognizer()
            with sr.Microphone() as source:
                try:
                                audio = r.record(source,
duration=30)
                                converted =
r.recognize_google(audio)
                                converted = converted.lower()
                                return converted
                except:
                        return False
```

- **user_task.py**

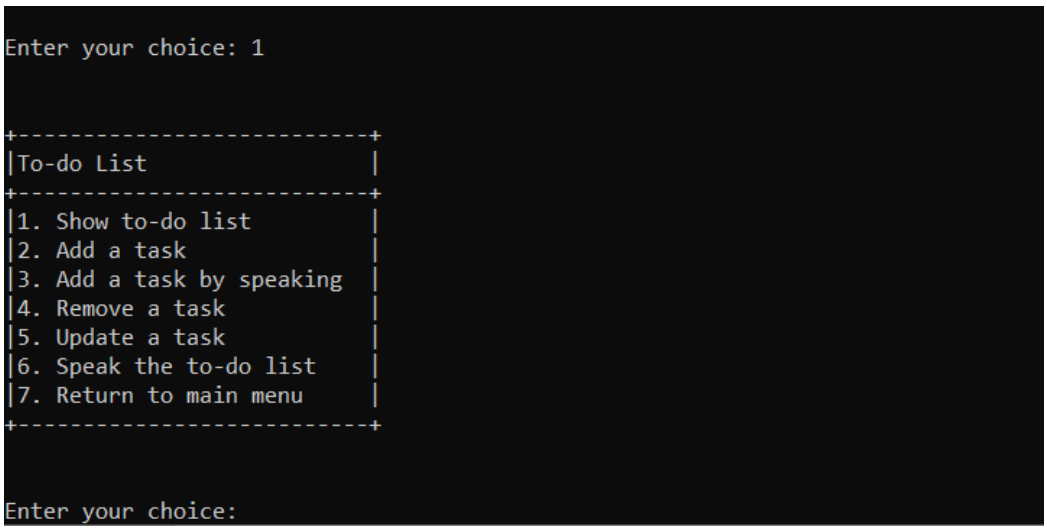
```
import os
import onetimepad as ot
import pyttsx3
import speech_recognition as sr
import user_db as udb

class Task:

	def check(name):
		try:
			file = open('./to-do-list/{}.txt'.format(name),
'r')
			obj = file.read()
			if len(obj)>0:
				return True
			return False
		except FileNotFoundError:
			file = open('./to-do-list/{}.txt'.format(name),
'a')

	def show(name):
		if Task.check(name) == False:
			print('No tasks in the to-do list!\n')
		else:
			file = open('./to-do-list/{}.txt'.format(name),
'r')
			obj = file.read().splitlines()
			password = udb.User.crypt_key(name)
			for i in range (len(obj)):
				obj[i] = ot.decrypt(obj[i], password)
				print('---> ' + str(i+1) + '. ' + obj[i] +
'\n')

	def add(name, task):
		file = open('./to-do-list/{}.txt'.format(name), 'a')
		password = udb.User.crypt_key(name)
		task = ot.encrypt(task, password)
		file.write(task + '\n')
		file.close()
		udb.Logs.add_task(name)
		print('Task was successfully added\n')

	def remove(name, number):
		file = open('./to-do-list/{}.txt'.format(name), 'r')
		new_file = open('./to-do-list/new.txt', 'a')
		obj = file.read().splitlines()
		if int(number) > len(obj):
			print('Task not found\n')
		else:
			for i in range (len(obj)):
				if i != int(number) - 1:
					new_file.write(obj[i]+'\n')
				elif i == int(number) - 1:
```

```
                                pass
                        file.close()
                        new_file.close()
                        os.remove('./to-do-list/{}.txt'.format(name))
                        os.rename('./to-do-list/new.txt', './to-do-
list/{}.txt'.format(name))
                        udb.Logs.delete_task(name, number)
                        print('Task was successfully removed\n')

        def update(name, number, new_task):
                file = open('./to-do-list/{}.txt'.format(name), 'r')
                new_file = open('./to-do-list/new.txt', 'a')
                obj = file.read().splitlines()

                if int(number) > len(obj):
                        print('Task not found\n')
                else:
                        for i in range (len(obj)):
                                if i == int(number) - 1:
                                        password = udb.User.crypt_key(name)
                                        new_task = ot.encrypt(new_task,
password)
                                        obj[i] = new_task
                                        new_file.write(obj[i]+'\n')
                                else:
                                        new_file.write(obj[i]+'\n')
                        file.close()
                        new_file.close()
                        os.remove('./to-do-list/{}.txt'.format(name))
                        os.rename('./to-do-list/new.txt', './to-do-
list/{}.txt'.format(name))
                        udb.Logs.update_task(name, number)
                        print('Task was successfully updated\n')

class Speak:

        def taskToSpeech(name):
                engine = pyttsx3.init()
                engine.setProperty('rate', 150)

                if Task.check(name) == False:
                        engine.say('No tasks in the to-do list!')
                        engine.runAndWait()
                else:
                        file = open('./to-do-list/{}.txt'.format(name),
'r')
                        obj = file.read().splitlines()
                        password = udb.User.crypt_key(name)
                        for i in obj:
                                i = ot.decrypt(i, password)
                                engine.say(i)
                                engine.runAndWait()
```

```
class Speech:

        def speechToTask():
                r = sr.Recognizer()
                with sr.Microphone() as source:
                        try:
                                audio = r.record(source, duration=10)
                                converted = r.recognize_google(audio)
                                converted = converted.lower()
                                return converted
                        except:
                                return False
```

OUTPUT SCREENS

- Program



```
Enter your choice: 2


+--------------------------+
|Notes                     |
+--------------------------+
|1. Show list of notes     |
|2. Show a note            |
|3. Add a note             |
|4. Add a note by speaking |
|5. Remove a note          |
|6. Update a note          |
|7. Speak a note           |
|8. Return to main menu    |
+--------------------------+


Enter your choice:
```

```
Enter your choice: 3


+-------------------------+
|Settings                 |
+-------------------------+
|1. Change user name      |
|2. Change user password  |
|3. Remove user           |
|4. Export logs           |
|5. Return to main menu   |
+-------------------------+


Enter your choice:
```

- MySQL Database

```
mysql> show tables;
+--------------------+
| Tables_in_noterpy |
+--------------------+
| devansh            |
| pratham            |
| preetika           |
| ridham             |
| samridh            |
| users              |
+--------------------+
6 rows in set (0.04 sec)

mysql>
```

```
mysql> select * from users;
+----------+------------+------------+
| name     | password   | cryptkey   |
+----------+------------+------------+
| devansh  | amity@123  | qRbDzFNK<  |
| ridham   | tippa      | d#H-3`EN?G |
| pratham  | jeetopper  | ?4^,KcuW!z |
| samridh  | mirzapurop | O4xH@#*Y>e |
| preetika | codpr0     | uk~[&HuJ+W |
+----------+------------+------------+
5 rows in set (0.00 sec)

mysql>
```

```
mysql> select * from devansh;
+------------+--------------------------+----------+
| date       | log                      | time     |
+------------+--------------------------+----------+
| 10/12/2020 | new user devansh created | 17:04:01 |
| 10/12/2020 | added a new task         | 17:04:13 |
| 10/12/2020 | updated password         | 17:04:26 |
| 10/12/2020 | updated password         | 18:22:57 |
| 10/12/2020 | added a new task         | 18:23:23 |
| 10/12/2020 | added a new note dilemma | 18:31:15 |
| 10/12/2020 | updated password         | 19:44:30 |
| 11/12/2020 | updated password         | 12:57:18 |
| 11/12/2020 | added a new note bored   | 13:06:37 |
| 11/12/2020 | updated password         | 14:31:33 |
+------------+--------------------------+----------+
10 rows in set (0.00 sec)

mysql>
```

BIBLIOGRAPHY

- https://github.com
- https://geeksforgeeks.com
- https://stackoverflow.com
- https://docs.python.org/3/
- https://dev.mysql.com/doc/
- https://tutorialspoint.com


```
Thankyou for using NoterPy :)
```