PHENIKAA UNIVERSITY

FACULTY OF ELECTRICAL ELECTRONIC ENGINEERING

OBJECT-ORIENTED PROGRAMMING

Final Project:

ToDo List Application

Lecturer: **Msc. Vu Hoang Dieu**

Full name: Pham Dinh Dat

Group: 3

Student code: 20010736

Hanoi, May 2023

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1. INTRODUCTION

Life with many thing force us to get them done. Sometimes, we can feel overwhelmed by a hectic schedule, feel like at the rock bottom as missing the deadlines, procrastinating when everything is too much to do. At that moment, planning is the best way to help us get out of this situation. As Antoine de Saint-Exupéry said: “A goal without a plan is just a wish”, or Dale Carnegie had a quote: “An hour of planning can save your 10 hours of doing” [1]. The best way to plan is making a list of whatever you have to complete. This is prioritized list of all tasks that we need to carry out [2]. To-do lists are crucial since it help us to be much better organised, less procrastinated and more productive. It can not help us to make progress in leaps and bounds but we can gradually improve the working proficiency. However, not everyone can make a suitable to-do list and prioritize the more important task.

Inspired by this issues, our team has made an application to help user create their own to-do lists. The application allows users to enter all tasks, delete tasks, sort tasks as their importance and urgency, etc. Besides, the application has user-friendly interface which can help them to plan in an easy way and do not make the planning become a pressure.

Chapter 2. LITERATURE REVIEW

1. Python

1.1. Python Object Oriented Programming

(This part is mostly referenced in a document on website of GeeksforGeeks [3])

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently.

In Python, object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming. The main concept of OOPs is to bind the data and the functions that work on that together as a single unit so that no other part of the code can access this data.

Generally, there are several OOPs concepts in Python mentioned below.

1.1.1. Classes

A class is a collection of objects. A class contains the blueprints or the prototype from which the objects are being created. It is a logical entity that contains some attributes and methods.

1.1.2. Objects

- The object is an entity that has a state and behavior associated with it.
- An object consists of:
  - State: It is represented by the attributes of an object. It also reflects the properties of an object.
  - Behavior: It is represented by the methods of an object. It also reflects the response of an object to other objects.
  - Identity: It gives a unique name to an object and enables one object to interact with other objects.

1.1.3. Inheritance

Inheritance is the capability of one class to derive or inherit the properties from another class. The class that derives properties is called the derived class or child class and the class from which the properties are being derived is called the base class or parent class. The benefits of inheritance are:

- It represents real-world relationships well.
- It provides the reusability of a code. We don't have to write the same code again and again. Also, it allows us to add more features to a class without modifying it.
- It is transitive in nature, which means that if class B inherits from another class A, then all the subclasses of B would automatically inherit from class A.

1.1.4. Encapsulation

Encapsulation is one of the fundamental concepts in object-oriented programming (OOP). It describes the idea of wrapping data and the methods that work on data within one unit. This puts restrictions on accessing variables and methods directly and can prevent the accidental modification of data. To prevent accidental change, an object's variable can only be changed by an object's method. Those types of variables are known as private variables.

1.1.5. Polymorphism

Polymorphism in object-oriented programming (OOP) refers to the ability of objects of different classes to be treated as objects of a common superclass. It allows objects to be processed in a generic manner, without the need to know their specific types.

In Python, polymorphism is supported through method overriding and method overloading i.e. a subclass defines a method with the same name as a method in its superclass and the subclass method overrides the implementation of the superclass method.

1.1.6. Abstraction

It hides unnecessary code details from the user. Also, when we do not want to give out sensitive parts of our code implementation and this is where data abstraction came. Data Abstraction in Python can be achieved by creating abstract classes.

1.1.7. Decorators

decorators are a way to modify the behavior of functions or classes without directly changing their source code. Decorators are implemented using the concept of higher-order functions, which take one or more functions as arguments and return a new function. They provide a convenient and reusable way to add functionality to existing code.

Decorators are denoted by the @ symbol followed by the decorator function or class name, placed immediately before the function or class definition. When the decorated function or class is called or instantiated, the decorator is applied to it.

1.2. PyQt5

PyQt5 is the latest version of a GUI widgets toolkit developed by Riverbank Computing. It is a Python interface for Qt, one of the most powerful, and popular cross-platform GUI libraries. PyQt5 is a blend of Python programming language and the Qt library. So, in this project, we used PyQt5 to build and design the TodoList App.

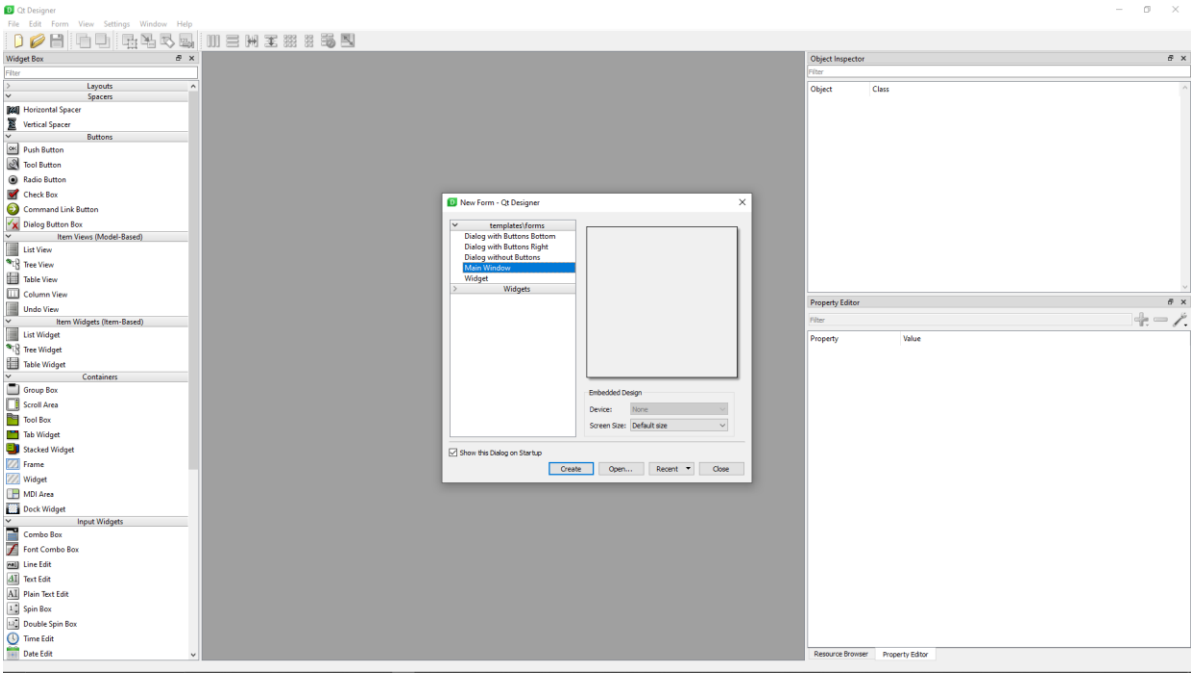


Fig. 1 Create a new application with Qt Designer

Chapter 3. SOLUTIONS/ METHODS

1. Approach

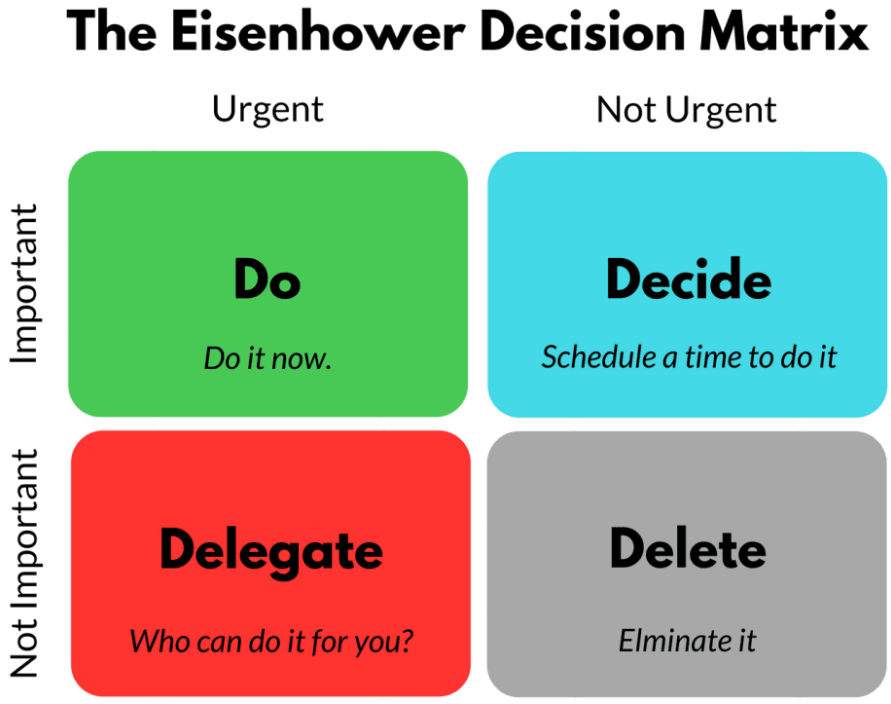


Fig. 2 The Eisenhower Matrix, a productive tool that helps prioritize tasks based on their urgency and importance, dividing them into four categories: urgent and important, important but not urgent, urgent but not important, and not urgent nor important.

The working priciple of this application is based on The Eisenhower Matrix [3] as Fig. 2. The Eisenhower Matrix, also known as the Eisenhower Decision Matrix or the Urgent-Important Matrix, is a productivity tool that helps individuals prioritize tasks and manage their time effectively. It was popularized by former U.S. President Dwight D. Eisenhower, who used a similar framework to organize his workload. The matrix categorizes tasks into four quadrants based on their urgency and importance:

- Important and Urgent: These are tasks that require immediate attention and have a significant impact on your goals or responsibilities. They should be dealt with promptly and personally.
- Important but Not Urgent: These tasks are important for long-term goals and have a higher value, but they don't require immediate action. They should be scheduled and given dedicated time for completion.
- Urgent but Not Important: These tasks are often distractions or interruptions that demand immediate action but don't contribute much to your goals. Whenever possible, delegate or eliminate these tasks to focus on more important activities.
- Not Urgent and Not Important: These tasks are time-wasters and distractions that provide little or no value. They should be eliminated or minimized to free up time for more meaningful endeavors.

By using the Eisenhower Matrix, individuals can gain clarity on their priorities and make informed decisions on how to allocate their time and effort more effectively, ultimately enhancing productivity and achieving desired outcomes.

Similarly, we arrange the order of all tasks with the four task evaluation criteria as this matrix. All tasks were labeled with the level of importance and urgency. It is done in a way that users will check whether this task is essential or not. Regarding urgency, users have to enter the due date, the system has to calculate the remaining time from the current time and compare it with a threshold. If it is larger than the threshold, it is labeled as an urgent task, and vice versa. With the support of 2 labels – importance and urgency, we can divide all of them into four groups as aforementioned. We stack these four groups in order “do”, “decide”, “delegate”, and “delete” and display them on the to-do list board to remind users of what they have to complete in prioritized order.

2. Implementation

2.1. Application's function

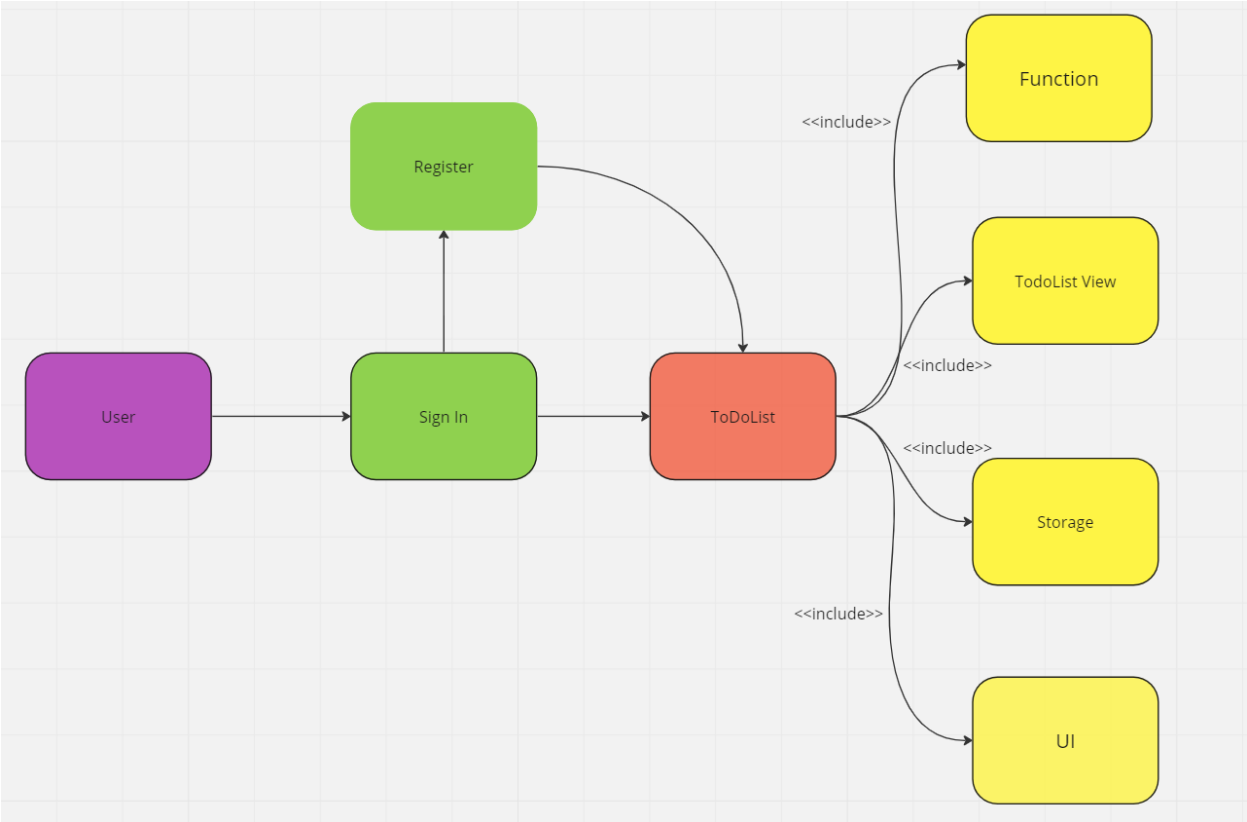


Fig. 3 The first initial use case diagram.

Before creating the TodoList Application, we firstly create a diagram that presents the idea and planning user story. We created some initial block as a "backbone" of the user story. Then, we managed to the use cases to keeping all possible actions.

In the first initial use case diagram, when user open the app, the first thing they do is sign in their account, if they have not any account, they can register a new account. Before signing in, the user comes to the TodoList interface that includes the functions, TodoList View, Storage, and UI. Having a initial use case diagram, then, we developed and added more requirements and extends on the use case diagram. Finally, we have the last use case diagram as Fig. 4

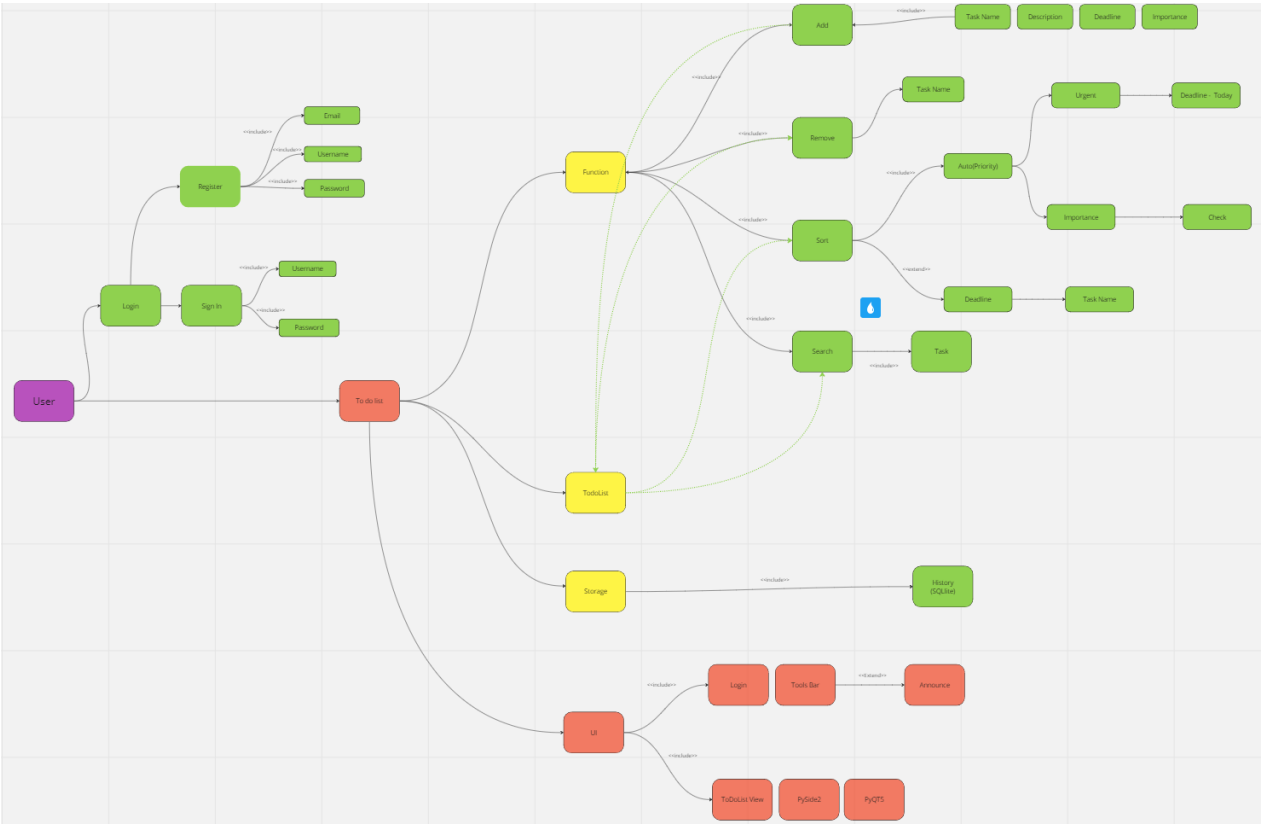


Fig. 4 The use case diagram of TodoList App

With login task, we have two main classes – Register and Login as Fig. 5.

Regarding Register class, we need to list 3 strings – username, email, and password to '.json' file. Before doing it, we need to make sure all information is valid by creating a class called CheckValidation. Information of each user was saved on a saved '.json' file which was stored on a folder called 'user_infor'. First of all, we need to load this folder to work with information of all users. Then, we get the enterd information and check the validation by 3 functions – 'check_username', 'check_email', and 'check_password'. 'check_username' function will check the entered string included at least 4 charaters and existed before or not by retrieval all account and check all identical information. To check valid password, we defined valid password contains at least a number and an uppercase letter. A valid email contains "@" and "." characters. We created a class called **Register:** that has three attributes: username(string), email (string or integer that stores email addresses of the user), and password(any). The class have 2 methods:

- **check_valid** (username: str, email: any) – the function load over to check username and password correctly. If existing username or email return False, else the function will create a new account on database.

- **register** (username: str, email: any, password: any) – Before checking the valid account, the function will create a file .json or .txt to store all information of user (ID, username, email, password)..

With Login class which inherits some properties of Register class we just need to check two entered strings – username and password whether they are existed. If it is true, return True. We created a class called **Login** that has two attributes: username (string that username when user create a new account) and password(any). The class Login have 2 methods:

- **check_valid**(username: str, password: any) – the function load over database to check username and password correctly. If valid return True, else return False.
- login (username: str, password: any) – if valid username, login to account and return True.

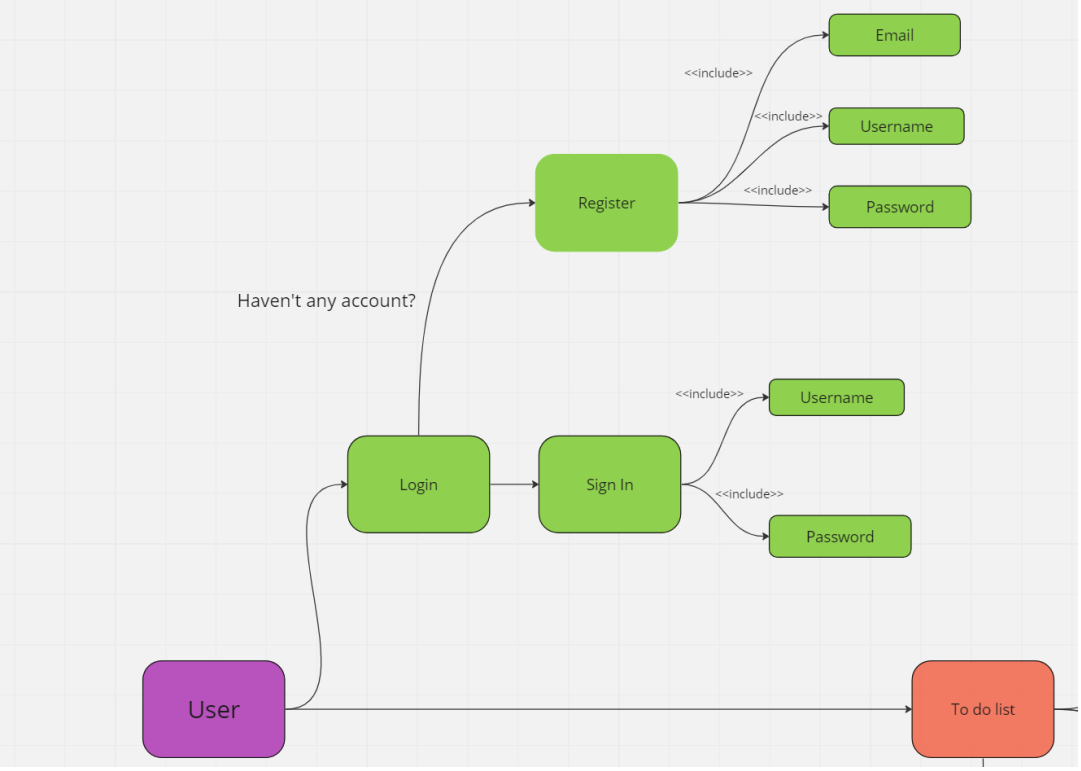


Fig. 5 Login use case diagram

In the functions of the app, we add more functions such as add task, delete task, sort tasks and search task. These functions require some implementation from users. TodoList is an important task in this project. It plays a role as process and management the application. In this project, we create a python file called function.py. In this file, we create a class named ToDoList that has many methods:

- **create_connect**(self, **db_file**) – this method is used to create the connection with SQL database. In my application, we used sqlite3 as a database which store all data of the user.
- **sql_database(**self, db_folder) – this method is used to create the database to store all tasks or any change information of user. The database is SQL table with named "**Tasklist**" and the columns are TASK, DECSRIPTION, DEADLINE, IMPORTANCE. If this method is called, it will generate a file .db which is the database about Todolist of user storing in this file.
- task_completed_database(self, **db_Folder)** – this method is to store all completed task in Todolist, the tasks which is completed are stored in SQL table with the same as we create a new database.
- get_all_task(self, db_Folder) – this method to getting all tasks that stored in SQL database.
- add_task(self, **db_Folder**) – by heritance class UI_MainWindow, this method will get and extract the input of the app such as: taskname, description, deadline, importance. This information is inserted into the database when the user wants to add new tasks.
- delete_task(self, db_Folder, db_folder_cpl) – there are two arguments: db_Folder(path to the database storing current tasks), db_folder_cpl(path to the database storing completed tasks). When the user completes the tasks, they chose taskname field then click on "deleted task" button. The delete_task function will be called and remove this task from list task. The task which was removed is stored in the completed task database.
- search_task(self) – when user want to search their task, they just write on the search task engine. This function gets text in the search engine and looks for this task name in current list task then return the task with index of row and view in task view.
- sort_task(self, db_folder, threshold, sort_type= "sort by deadline") – there are three arguments: db_folder(path to the database storing current tasks), threshold(integer that setting threshold to calculate sort task), sort_type(string that set type of sort algorithms). This function will get all information of Todolist (Taskname, Description, Deadline, Importance) by using SQL query. The list of tasks will be sorted by one of two algorithms. If sort_type = "sort by deadline", the function returns the list of tasks with the day to deadline increasing, else if sort_type = "sort by urgent", that means we use the Eisenhower Matrix combine with two field deadline and importance. The threshold argument is used to set the threshold of deadline. The function will return a ranking task list using Eisenhower Matrix.

- display_task(self, db_folder) – db_folder(path to current task database). This function will connect to database and get all information of all tasks in current task database then show in table views. When adding task , delete task, search task or sort task, this function is usually called to show current tasks in TodoList.

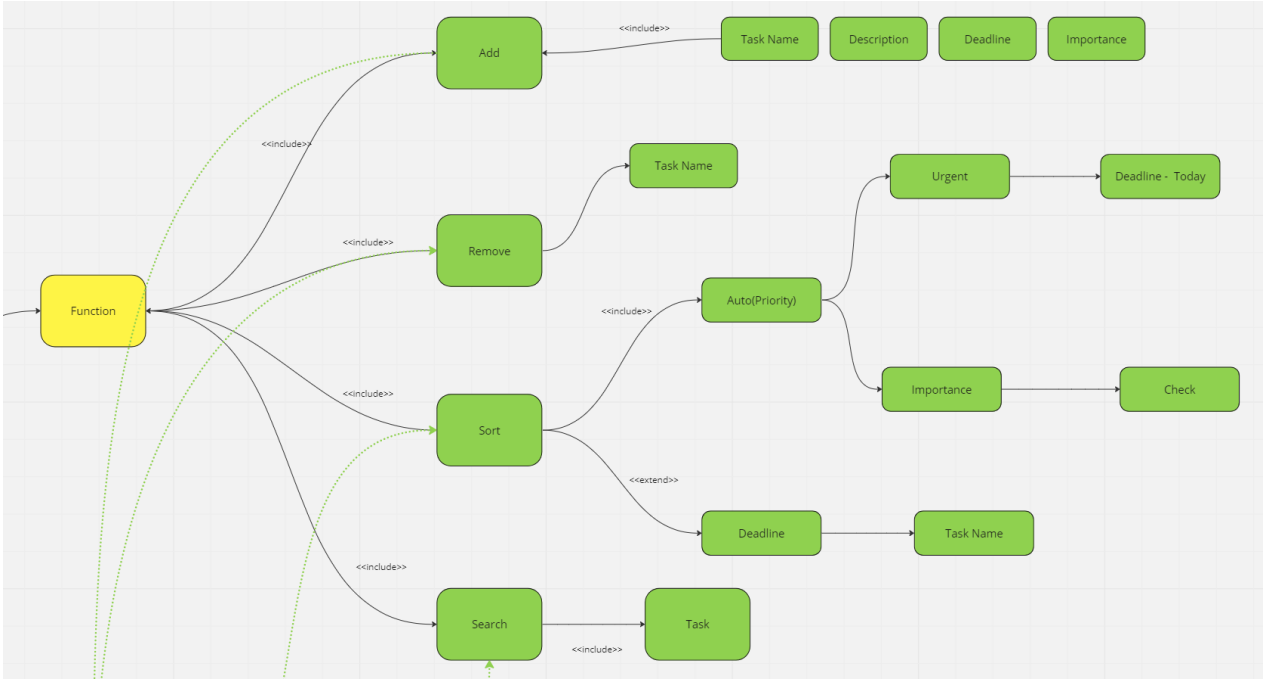


Fig. 6 The functions of the app

2.2. Interface

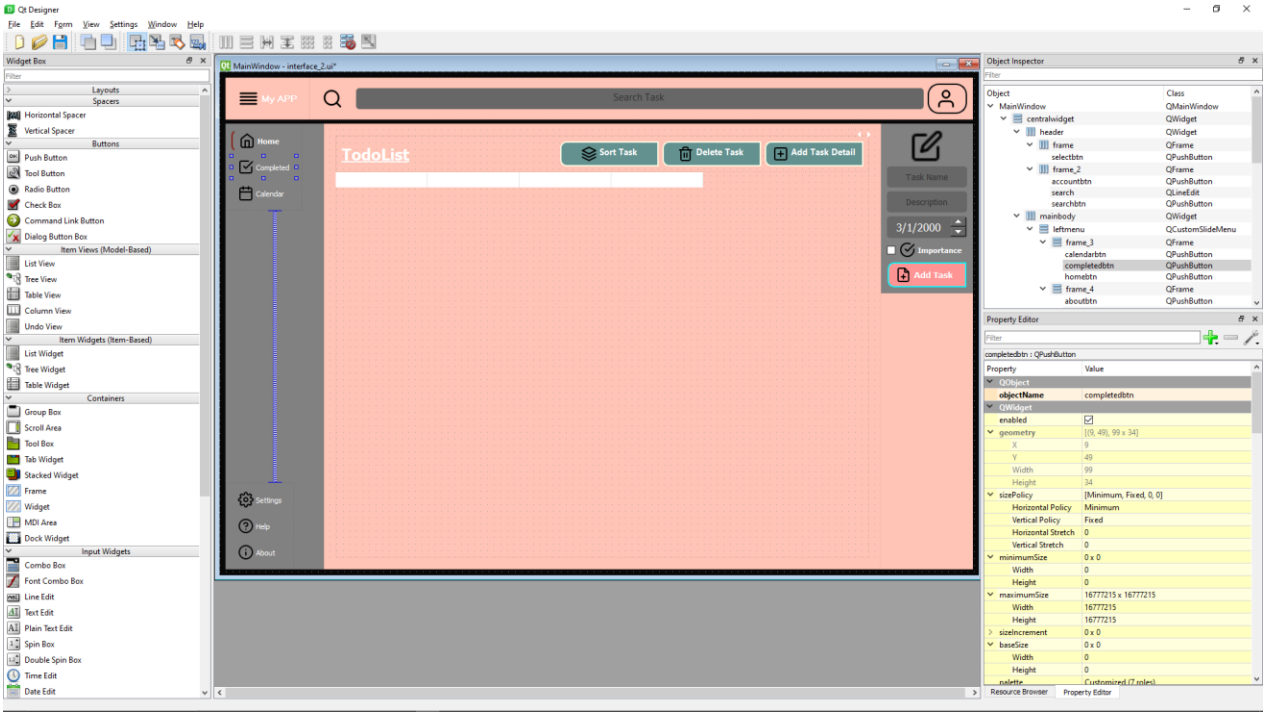


Fig. 7 Our TodoList App is designed by Qt Designer

We started to build the application by using the Qt Designer toolbox. This is a tool designer that you can easily design your application. By using this tool, we created templates, customized them, and set the functionality of each element in the application. When you want to view in python code, you can choose “Form” then select “View Python code”, it will generate a python code and you can run it with command line.

main.py: we create a main python file to run all programs. In this file, we create class named MainWindow. This class is inherited from class QMainWindow provided by PySide2 library. In this class, the class UI_MainWindow in file “ui_inteface.py” that exported from the file .ui when we edit by Qt Design tool. The task functions are called in this class. When the user edits or add new tasks, the functions above will be called and executed.

Chapter 4. RESULTS

1. Interface

Our application includes 3 main tabs: login tab, register tab, and home tab as Fig. 8, Fig. 9, and Fig. 10.

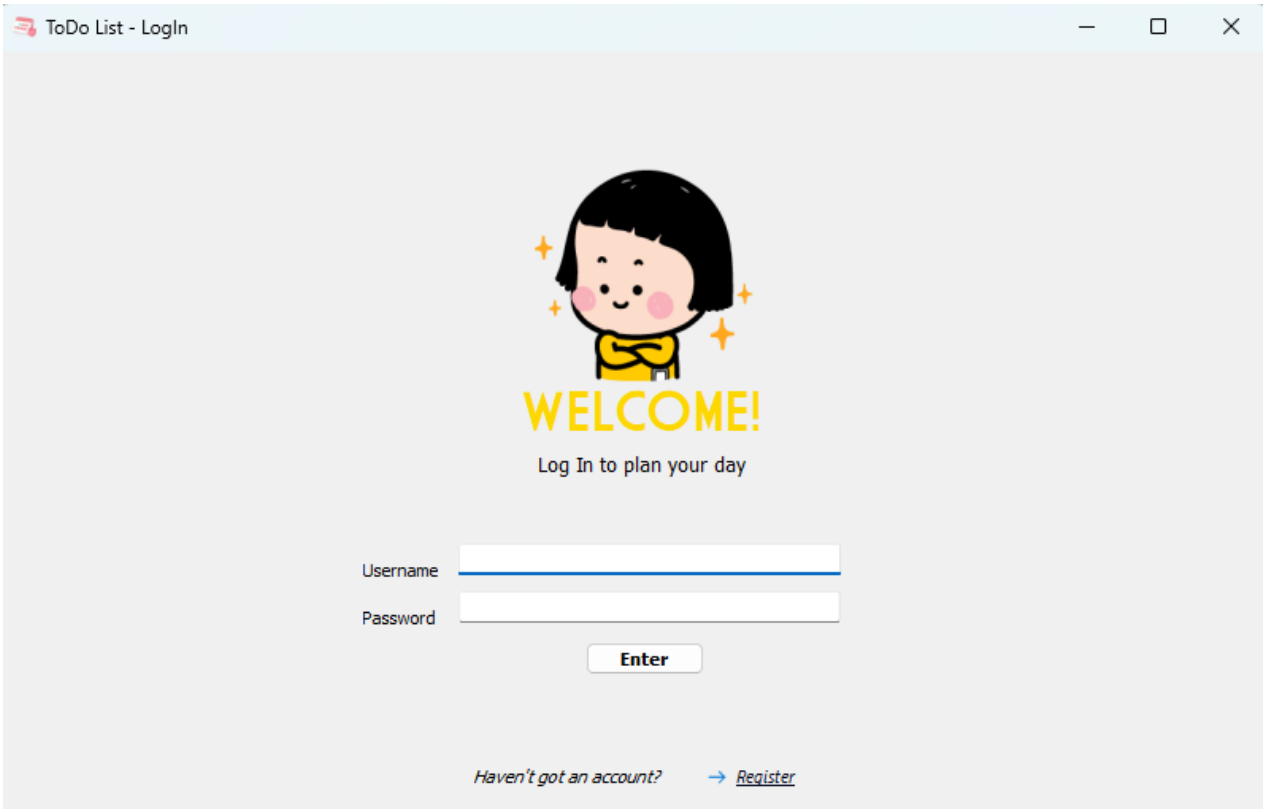


Fig. 8 Login tab where we have to enter username and password to access to the home tab. If we don't have any account, we can move to Register tab by clicking on Register button

This tab includes 2 empty boxes to enter users' information – username and password to identify which account will be planned. 'Enter' button is used to confirm that user wants to login with the information they entered before in the 2 above boxes. 'Register' button to direct user to the Register tab if they want.

Regarding the Register tab in Fig. 9, it is the place users can create their own account to have a right to access this application. There are 3 empty boxes to enter the users' information. Users can make the entered password visible with 'show password' button. It also includes 2 direct buttons – 'sign instead' if user want to back login tab and 'create' if they want to confirm the registration.

The most important is the home tab we user can do all functions add task, delete task, add task details, search task and sort tasks. Moreover, all of their tasks can be sorted automatically.

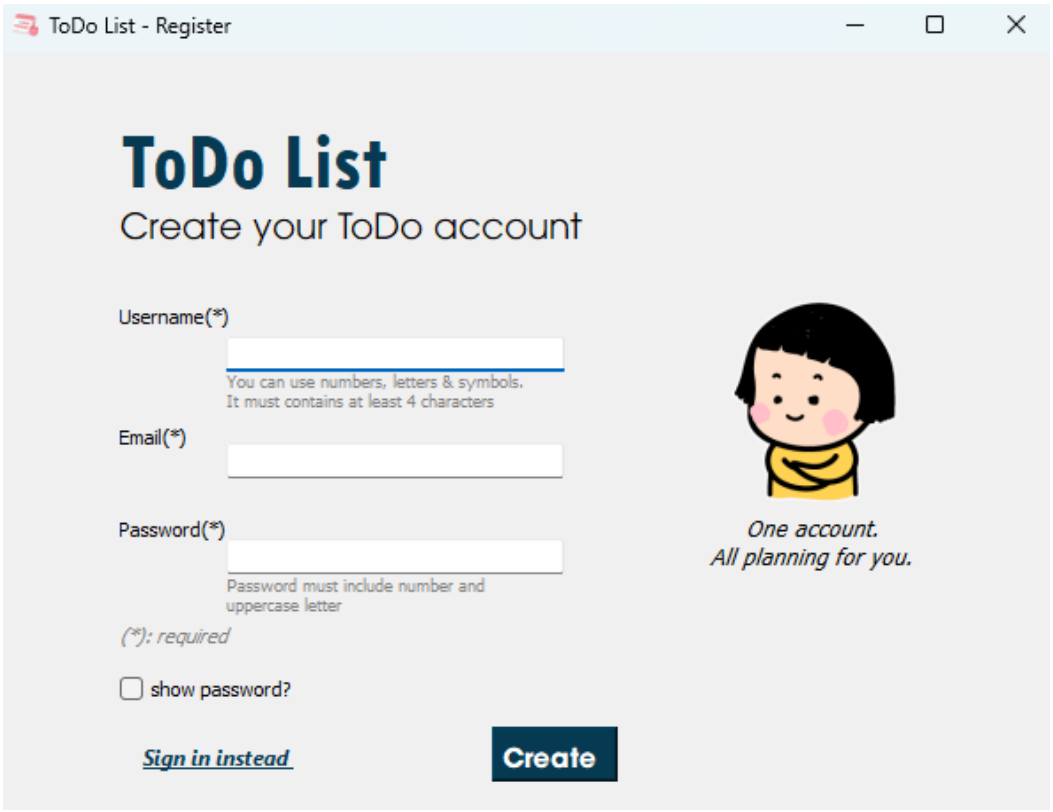


Fig. 9 Register tab where we create user account

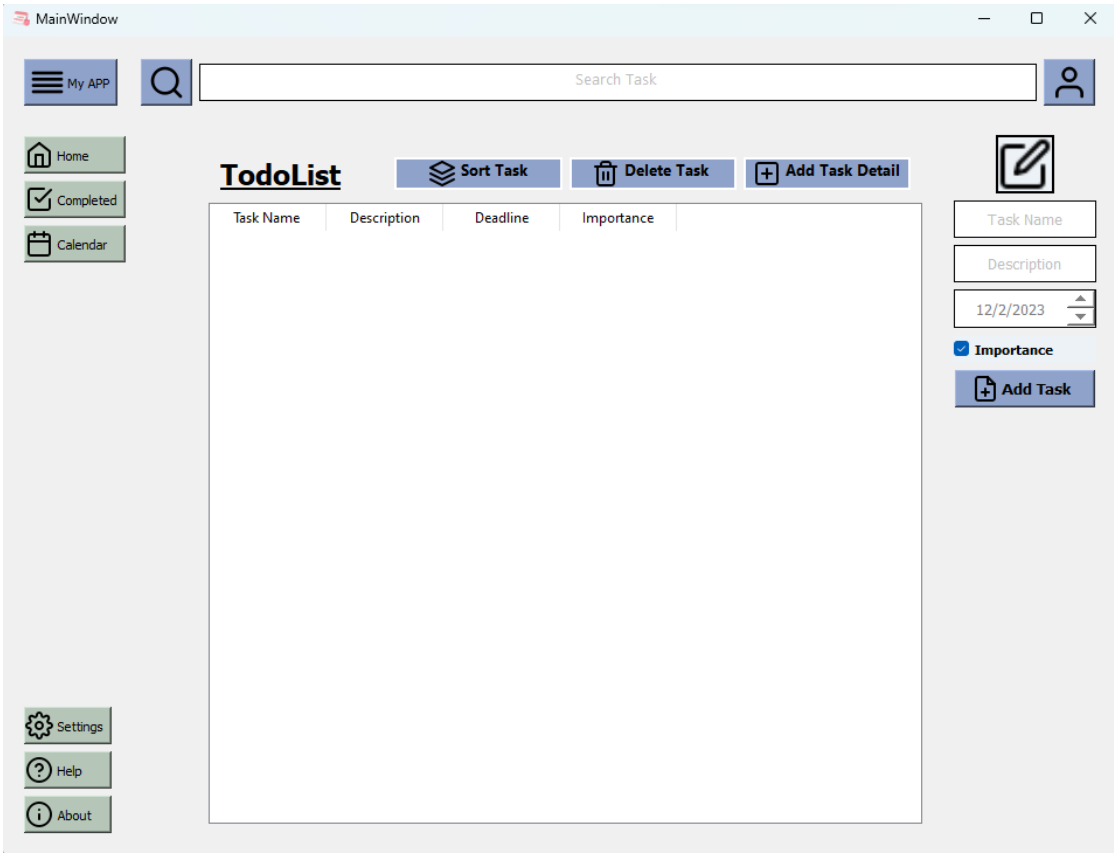


Fig. 10 Home interaface where we can plan all tasks with functions such as add task, delete task, sort tasks

2. User instructions

2.1. LogIn

In the Login tab, you can see in the display interface, it has the word "WELCOME" at the center of screen with a cute icon. When you want to login to your account, you must fill in two fields: Username and password. The username field is the name of the account that is the name when you register an account. The username must be the same as the username when you register. Clicking 'Enter' to confirm. The Password field is the Password when you register. Like Username, your Password must be the same as the password when you register in your account. If your account existed on the system, after clicking 'enter', you can immediately redirect to the home of ToDo List application.

2.2. Register

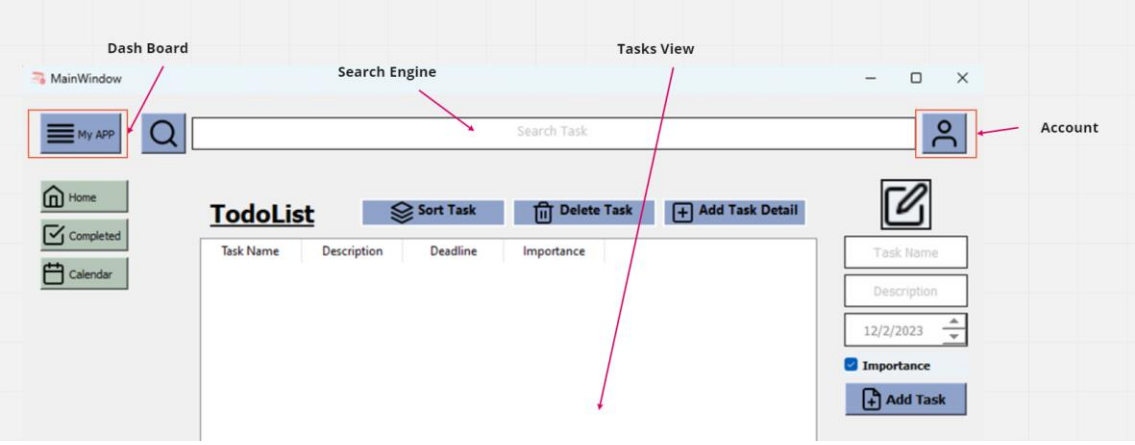
If you did not get any account, please click on 'register' to create one. The Register interface appears, and you can see some required fields to fill in. In this tab, please enter your information such as username, email adress and passwords. You should be carefull in this step since your entered information has to satisfy some requirements to create account successfully.

Firstly, your username has to be unique means it has not existed on the system before. It can be numbers, letters, and symbols and it must contains at least 4 charaters. In the **Email** field, you put in this with your email, and it require standard style of email like: k14-air@gmail.com. Next, your password must include number and uppercase letter. Additionally, you can recheck your password whether it is what you mean by clicking on 'show password button. You have to notice that you have to fill in all boxes to create an account.

After filling in all, clicking 'create' to completing your process, you have just created a new account!

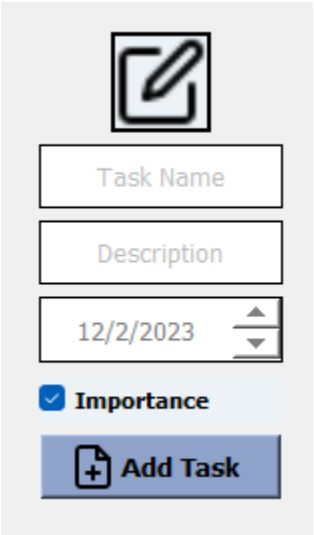
If you want to login, you can click on 'sign in instead'.

2.3. ToDo List



When you sign in or register successfully, you will be moved to the main interface. In this tab, you can plan all tasks with functions as add task, delete task…. Firstly, when the main interface tab appears, you can immediately see the current tasks on the center of the tab we call it "tasks view" (if you have no task or a new account, it will show the blank tables). In the top center of the tab, there is a search task engine. On the top right of the main interface, it has an account icon which shows the information of the user.

In the tasks view, you will see all information of all tasks. You can see the task name, deadline, the description of the tasks and the importance of the tasks. Besides, there are some functional buttons to edit the tasks like "Sort tasks", "Delete task" and "Add task".



When you want to add a new task, click on the "Add task detail" button. A new window appears, on the right side of the window, you can see a combo box to enter your task, task description, and due date. Your task name is placed on "task name" field, the description of the task is placed on the "description" field. You can set the deadline of the task in the next button. And click on the "importance" button it the task is necessary to complete. Then click on "Add task" button, the new task will be already shown in the task's tables.

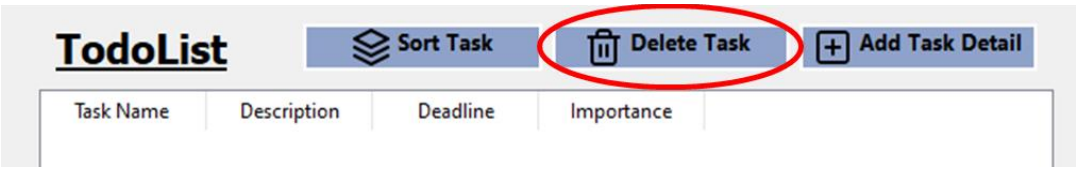


Fig. 11 Delete task

Next, to delete this task out of the list when you completed, click on to this task and click on 'Delete Task' button as Fig. 11. To make this task easier, you can search the task automatically without any manual search with search task bar as Fig. 12. When you want to search for a task, put a task name on the search task engine then click on the search icon, the task view will show the information of the task.

Fig. 12 Search task

The most useful function you need to know in this application is sort task function. With this function, you can receive a list of your to-do tasks in prioritized order. To use this, you just need to press 'sort task' button as Fig. 13.

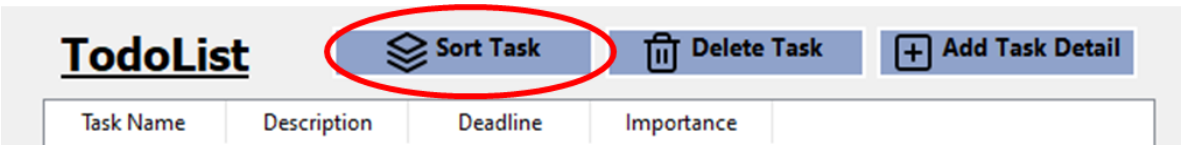


Fig. 13 Sort tasks

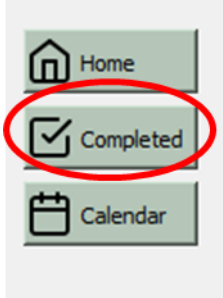


Fig. 14 Completed tasks

Besides, On the top left of the main interface, you can see the "My APP" button. There are some function buttons such as 'setting', 'help', 'about' to get additional information about app. When you click on this button, a new window will appear, you can chose some option to view in the tasks view such as "Completed" which retrieval your completed tasks, "Calendar" which shows the date of the day or just calendar, "Settings", "Help" and "About" buttons are the optional functions that you want to create some setting on your app or you want to know more about information and the user manual. If you want to back ToDo List, press 'home' button of the same combo box.

Chapter 5. CONCLUSIONS AND RECOMMENDATIONS

1. Conclusions

We have built an application which solves the planning problem raised on the introduction part. This app can help users plan their day by adding their tasks, automatically sorting tasks as its priority. With a user-friendly, the user can easily interact with the application. The implementation part also applies concepts of object-oriented programming.

2. Recommendations

On the other hand, our project also has some drawbacks such as the limitation of function. To develop this project, we will implement more functions such as alarm clock, Pomodoro timer, and more functions of sorts – sort by deadline, sort manually, etc. Besides, we also make this application executable with installation and optimizing database management.

Chapter 5. APPENDIX

In this section, we will show some concepts of Object-oriented Programming which are shown in our project.

1. Objects/Classes

We created four main classes: TodoList, Login, Register and Ui_MainWindow, and some others which play a different role to support the main classes.

For example:

```
class TodoList():
    def __init__(self, arg):...

    def create_connect(self,db_file):...

    def create_table(self, connect, create_table_sql):...


    def sql_database(self, db_Folder):...

    def task_completed_database(self, db_Folder):...

    def get_all_tasks(self, db_Folder):...

    def add_task(self, db_Folder):...
```

The TodoList class is a simple and easy-to-use way to create a to-do list application in Python. It provides all the basic functionality that you need to manage a to-do list, such as adding, deleting, searching for, sorting tasks and retrieval completed tasks. It provides organization and structure by grouping related data and behavior together. Classes promote modularity and reusability, allowing the todo list component to be used in different parts of the application or future projects. Separation of concerns separates the logic for managing the todo list from the rest of the code, enhancing maintainability. Classes enable extensibility, allowing the addition of new features without impacting the existing codebase. Lastly, using classes improves code readability by modeling real-world entities and making the code easier to understand and maintain.

2. Inheritance

Inheritance in OOP means When a class derives from another class. The child class will inherit all the public and protected properties and methods from the parent class. In addition, it can have its own properties and methods. An inherited class is defined by using the extends keyword.

```
# add account to users folder when user register
class Register:
    def __init__(self):
        self.accounts = []
        self.id = 1
```

```
    def check_infor(self, username:str, email:str, password:str):# check
validation of  user information

    def add_account(self, username:str, email:str, password:str):# add new
acount to the storage database


#login your account
class LogIn(Register):
    def __init__(self, username, password):
        super().__init__()
        self.__username = username
        self.__password = password
        self.accounts = Register().accounts


    def check_infor(self, user):
        if self.get_username in str(user) and self.get_password in str(user):
            return True


    def login(self):
        users = os.listdir('users_infor')
        for user in users:
            self.check_infor(user=user)
```

In this code, class `LogIn` inherit the `Register` class to get all accounts which we retrieval the entered information in.

3. Encapsulation

Encapsulation is a way to restrict the direct access to some components of an object, so users cannot access state values for all of the variables of a particular object. Encapsulation can be used to hide both data members and data functions or methods associated with an instantiated class or object.

```
class LogIn(Register):
    def __init__(self, username, password):
        super().__init__()
        self.__username = username
        self.__password = password
        #self.accounts = Register().accounts

    def check_infor(self, user):
        if self.get_username in str(user) and self.get_password in str(user):
            return True


    def login(self):
```

```
        users = os.listdir('users_infor')
        for user in users:
            self.check_infor(user=user)


    def get_password(self):

    def set_password(self, password):

    def get_username(self):

    def set_username(self, username):
```

In this code, we used private attributes to demonstrate encapsulation and used set and get function to interact with them.

4. Abstraction

Abstraction is the concept of object-oriented programming that “shows” only essential attributes and “hides” unnecessary information. The main purpose of abstraction is hiding the unnecessary details from the users.

```
from abc import ABC, abstractmethod

class ToDo(ABC):

    @abstractmethod
    def create_connect(self):
        pass

    @abstractmethod
    def create_table(self):
        pass

    @abstractmethod
    def task_completed_database(self):
        pass

    @abstractmethod
    def get_all_tasks(self):
        pass

    @abstractmethod
    def add_task(self):
        pass
```

```
    @abstractmethod
    def delete_task(self):
        pass

    @abstractmethod
    def search_task(self):
        pass

    @abstractmethod
    def sort_task(self):
        pass

    @abstractmethod
    def display_task(self):
        pass


class TodoList(ToDo):

    def __init__(self, arg):
        super(TodoList, self).__init__()
        self.arg = arg


    def create_connect(self,db_file): # create connect to database

    def create_table(self, connect, create_table_sql): #create table SQL
database

    def sql_database(self, db_Folder): make a SQL database

    def task_completed_database(self, db_Folder):# create a completed tasks
database

    def get_all_tasks(self, db_Folder): # get all current tasks


    def add_task(self, db_Folder): #add new tasks and saved to database
```

class ToDo includes some essential methods which are used in class TodoList and hides other unnecessary methods. It shows the abs tractional property of methods.

5. Polymorphism

The literal meaning of polymorphism is the condition of occurrence in different forms. Polymorphism is a very important concept in programming. It refers to the

use of a single type of entity (method, operator or object) to represent different types in different scenarios.

```
class Register:
    def __init__(self):
        self.accounts = []
        self.id = 1
    def check_infor(self, username:str, email:str, password:str):# check
validation of  user information
        check_valid = CheckValidation(username, email, password)
        check_email = check_valid.check_email()
        check_username = check_valid.check_username()
        check_password = check_valid.check_password()
        return check_username, check_email, check_password


class LogIn(Register):
    def __init__(self, username, password):
        super().__init__()
        self.__username = username
        self.__password = password
        self.accounts = Register().accounts


    def check_infor(self, user):
        if self.get_username in str(user) and self.get_password in str(user):
            return True
```

Both class Register and LogIn, have check_infor() function, but with class Register, it takes responsibility for check validation of user information, with the class LogIn, it is used to check existence of the account.

REFERENCES

[1] "12 AMAZING QUOTES ABOUT PLANNING TO LIVE BY," [Online]. Available: https://diaryofajournalplanner.com/planning-quotes/#:~:text=Report%20Ad-,%E2%80%9CA%20goal%20without%20a%20plan%20is%20just%20a%20wish.%E2%80%9D,by%20Antoine%20de%20Saint%2DExup%C3%A9ry.. [Accessed May 2023].

[2] t. M. T. C. Team, "To-Do Lists," [Online]. Available: https://www.mindtools.com/aug8p7g/to-do-lists. [Accessed May 2023].

[3] GeeksforGeeks, "Python OOPs Concepts," 25 May 2023. [Online]. Available: https://www.geeksforgeeks.org/python-oops-concepts/. [Accessed 27 May 2023].

[4] LUXAFOR, "The Eisenhower Matrix: Time and Task Management Made Simple," [Online]. Available: https://luxafor.com/the-eisenhower-matrix/. [Accessed May 2023].