

ASSIGNMENT

**Of**

**OBJECT ORIENTED PROGRAMMING FOR SOFTWARE DEVELOPMENT (2219)**

**Implementation of a To Do List application**

**Present By:**

**Harika Pavuluri (PAV22197997)**

**Submitted To:**

**Lecturer Edward Francis Bird**

**MSc Computer Science 2023/024**

**UNIVERSITY COLLEGE BIRMINGHAM**

**Date: 17th January, 2024**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| S.NO | TOPIC | PAGE NO |
| 1 | INTRODUCTION | 3 |
| 2 | ANALYSIS AND DESIGN   * USE CASE DIAGRAMS * CRC CARDS * CLASS DIAGRAM * SEQUENCE DIAGRAM * FLOW CHART | 4-10 |
| 3 | IMPLEMENTATION   * PROJECT SETUP * GITHUB LINK * SCREENSHOTS | 11-26 |
| 4 | TESTING AND VALIDATION   * IMPORTANCE OF TESTING * TEST CASES | 27-29 |
| 5 | EVALUATION   * FUNCTIONAL REQUIREMENTS * PLANNING AND EFFICIENCY * CODE ELEGANCE * APPLICATION USABILITY | 30-31 |
| 6 | REFERENCES | 32 |

**1.0 INTRODUCTION:**

According to the academic team, a To-Do List application would help undergraduate students who are currently having trouble prioritizing their tasks and assignments. Among the application's features are user accounts, sign in and out list tasks, Add, Edit, and Delete tasks, and Label tasks (e.g., by priority, subject, type, etc.) I started by setting up a Django project and constructing a virtual environment to develop this app. I then created a data model that describes the connections between lists and to-do items. Throughout the process of creating a Django to-do list application, I had frequently used the helpful Run server command in Django to make sure everything was operating as it should. Even before my web pages are complete, this can be helpful. I then created my own web pages to highlight the application. They take the form of templates in Django. Skeletal HTML pages, known as templates, are capable of being filled with actual application data. Templates does not provide much logic; they just select which template to display and what information should be provided. To execute that logic, Views are required. The logic of the application, code views, templates for creating and updating lists will include all naturally belong in Django's views. I now know how to link your pages and send them the necessary data using the Django URL dispatcher. I then went ahead and created extra views and templates. Lastly, I created, edited, and removed to-do lists and items to test the new user interface. The project's objective is to ensure that the students can keep a record of their activities and prioritize them so they will know exactly what must be done and when to finish.

**2.0 ANALYSIS AND DESIGN**

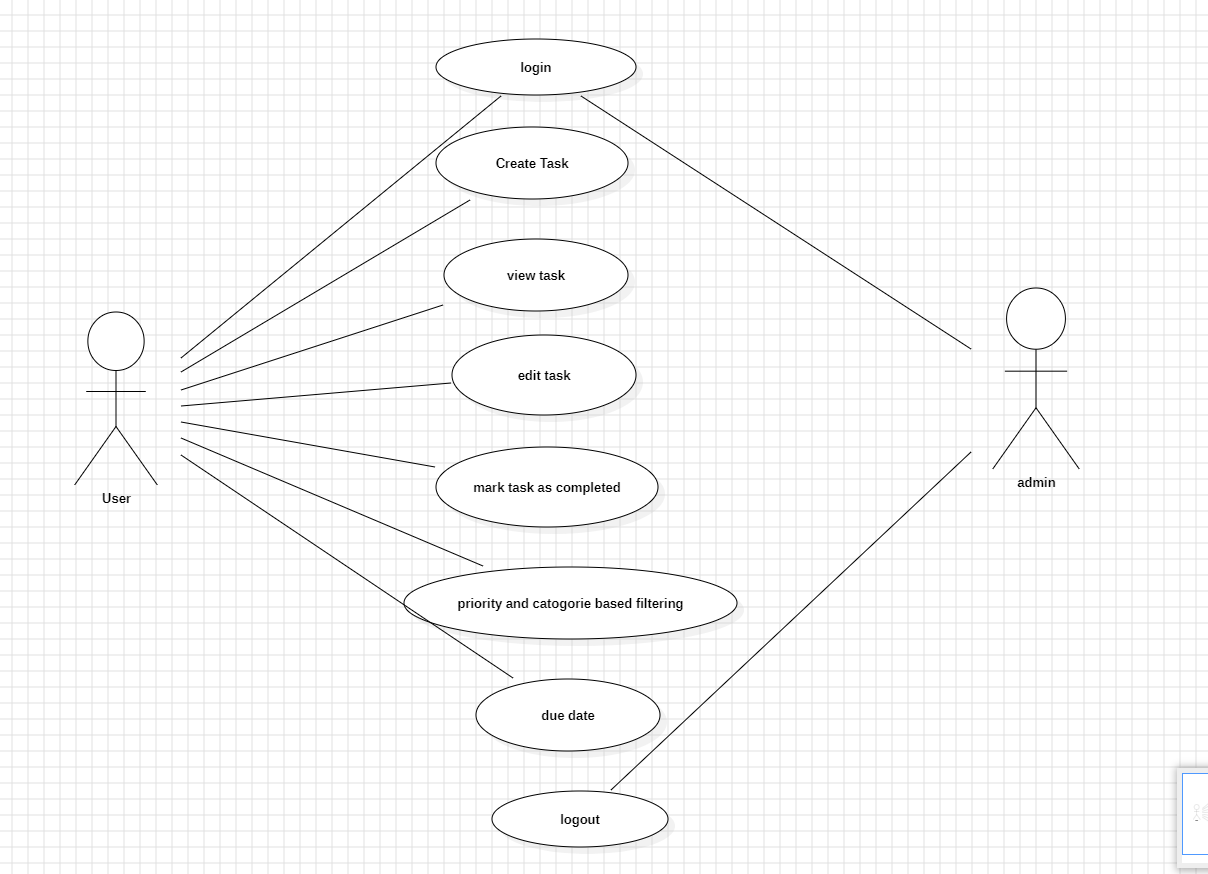
**UML (Unified Modelling Language) Schematics:**

An object-oriented software-intensive system is being developed, and its artifacts are specified, visualized, modified using the UML. A system's architectural blueprints can be visualized in a standard manner using UML, which includes components like:

* Performers
* Business procedures
* Parts
* Actions
* Statements in Programming Languages
* Database Organizations

**2.1 Use-Case Diagrams:**

Use cases describe the subject's provided behaviour without mentioning its internal organization. These actions, which entail communications between the subject and the actor, could alter subject's condition and how it interacts with its surroundings. Our application contains different actions like login, creating task, viewing task, adding task, marking task as completed, priority and category-based filtering, due date, and then logging out from the application. The user gets access to all actions, but the admin gets access to see the users in the application, tasks created by them. Based on the operations performed, these actions will vary from one operation to the other.

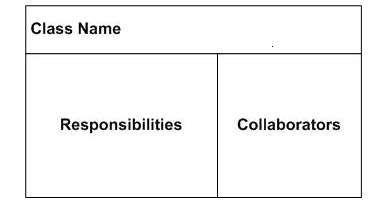


**2.2 CRC Cards**

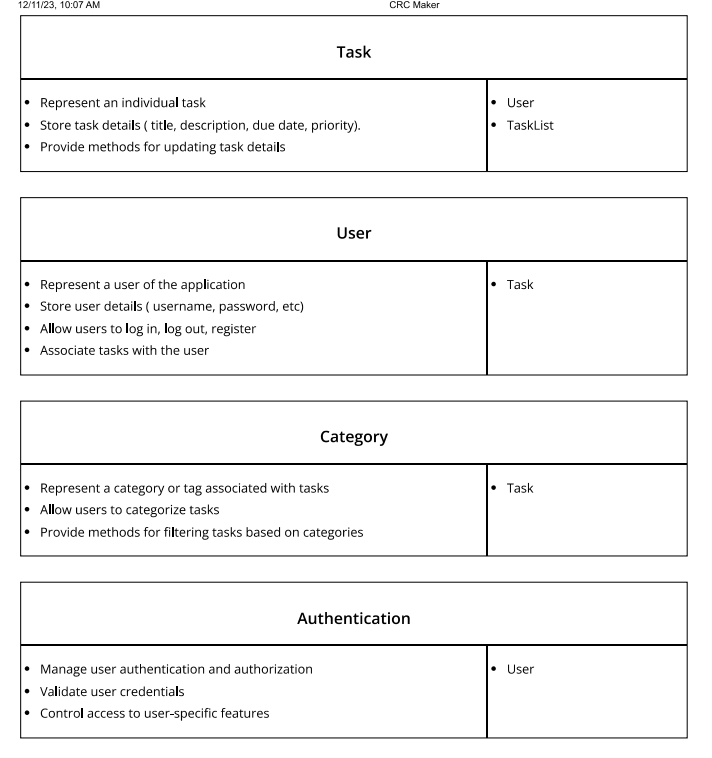
Three components comprised of a set of standard index cards make up the Class Responsibility Collaborator (CRC) concept.

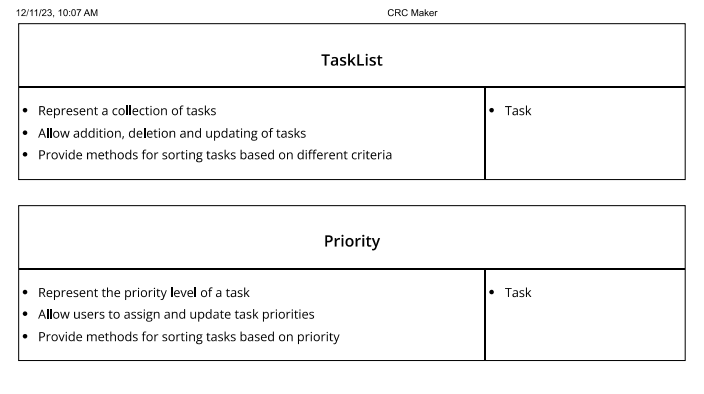
* Class Name: A class is a grouping of related items.
* Responsibilities: This is knowledge or action that students take on.
* A class that works with another class to carry out its duties is a collaborator.

**CRC Card Layout**



**CRC CARDS FOR TO-DO-LIST APPLICATION**

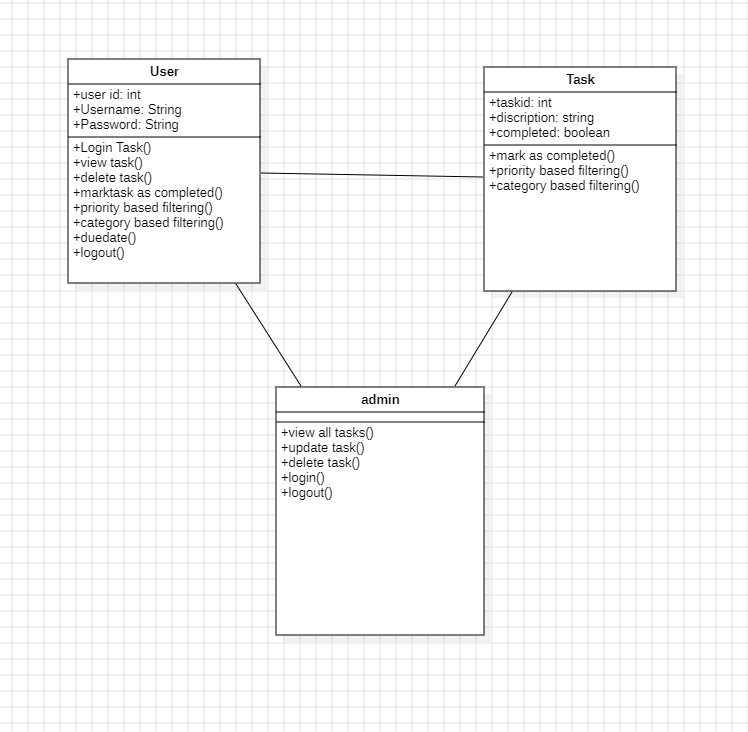




* CRC cards contains different classes like User, Task List, priority, Category.
* The different responsibilities of User include user register, user login, creating a task, editing a task, deleting a task.
* Responsibilities of Task list include list of tasks, addition, deletion of tasks and methods for sorting tasks.
* Responsibilities of Priority includes task priority, permissions for user to assign and update priorities.
* Responsibilities of Authentication includes user authentication and user authorization, validation of user credentials.

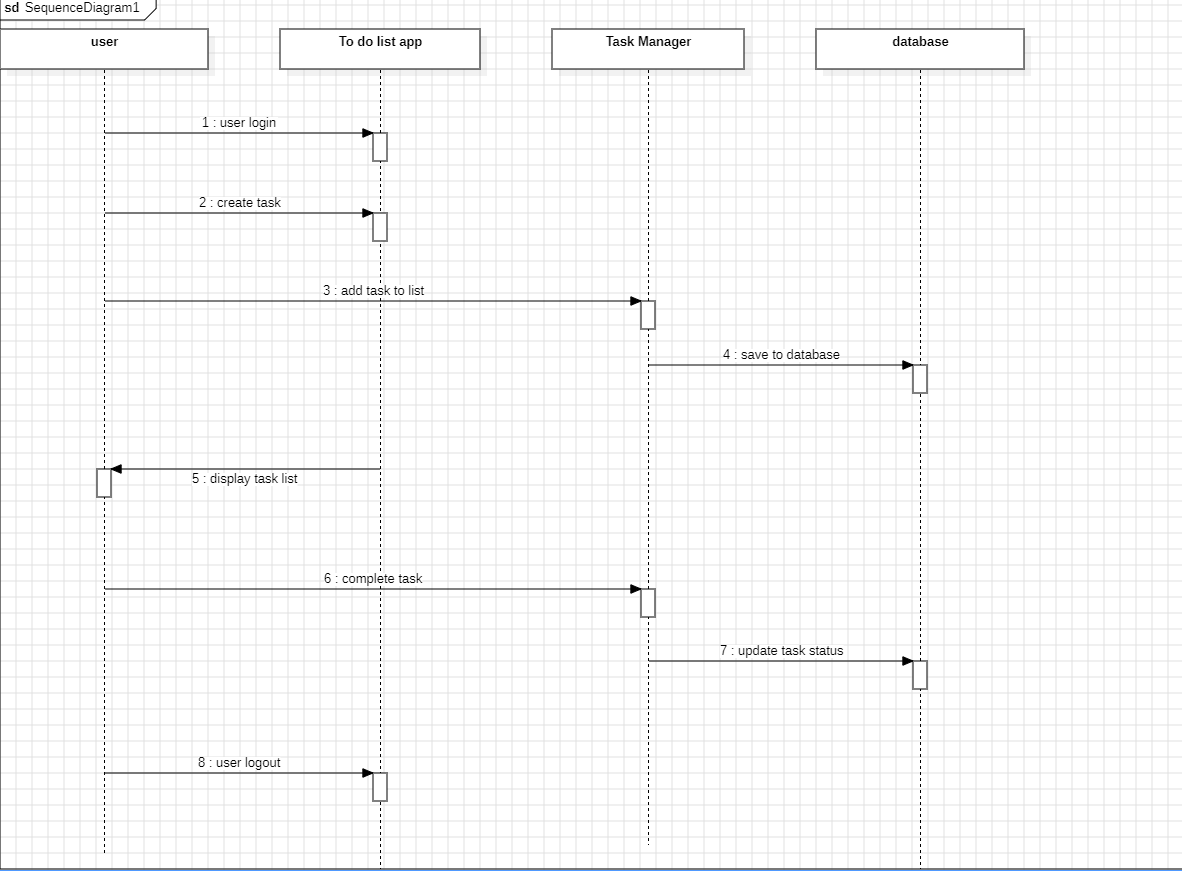
**2.3 Class Diagram**

The fundamental component of object-oriented modelling is class diagram. The user class consists of the user ID and username, which is in string format, and it contains different operations like login task, view task, delete task, mark task as complete, priority-based filtering, due date, and logout. The task contains the task ID, description, and completed fields. It marks the completed tasks, filters the tasks based on priority, and categorizes the tasks based on filtering. Admin gets access to register the user, log in the user, log out the user, delete the user, add the tasks to a user, delete the task, and update the task. Admin cannot see the password created by the user because of security issues, he can only see the user id and the tasks created by the user.



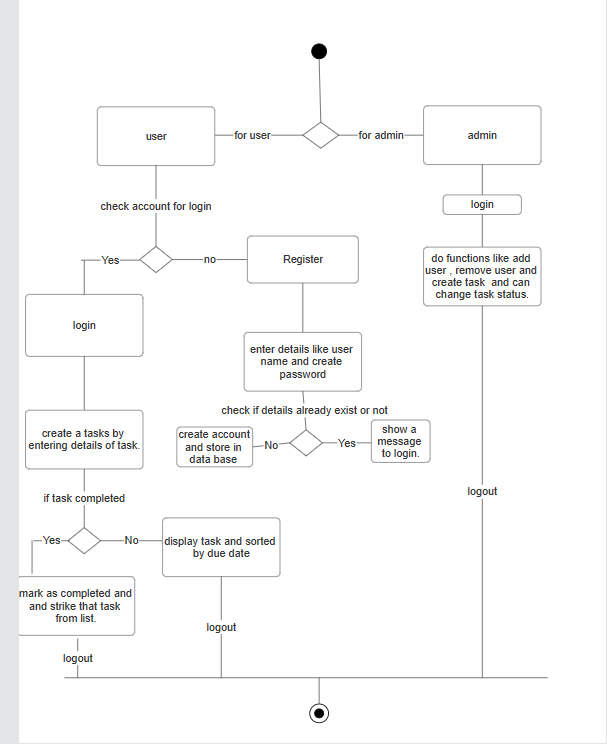
**2.4 Sequence Diagrams:**

It shows how the items interact. In our application the interaction will be between the user and the application. The user will engage with the application by logging into the application with his credentials. Then, the user creates a task, which leads to interaction with the task manager, and the data is saved to the database. The user completes the task and updates the task to the database. After the completion of the task then, he can log out from the application, and the database will be updated with all the information.



**2.5 Flow Chart**

* A flowchart is a graphic representation of a computer program, system, or process. They are extensively used in many different sectors to plan, analyse, document, and communicate—in simple, understandable diagrams—often complicated processes.
* They are used to show off the structure of the code.
* See how code is being executed within a program.
* Display a website or application's structure.
* Recognize how users interact with a software or website



* The user will register to the application if he is new to the application and these details will be stored in the database and can log in with those credentials.
* After logging into the application, he can create a task by entering details of task like title, description, priority of the task, its category, due date of the task and if the task is completed, he can mark the task as completed.
* The admin can view the groups, users and the tasks created by them, but he doesn’t have access to the user's passwords.
* After the work is completed, the user can log out from the application.

**3.0 IMPLEMENTATION**

**3.1 Establishing the project.**

**Step 1: Install Django and configure your virtual environment.**

* To establish a virtual environment, use the subsequent command.

python –m venv venv

* Turn on the virtual setting.

.\venv\Scripts\activate

* Use the following command to launch the command prompt and install Django.

pip install django

## 

## **Step 2: Build a Django To-Do App**

### Using the following commands, create a directory as the new application's root and cd into it, making sure that the syntax matches the operating system.

### cd desktop

### django-admin start project Todo list

### Go to the terminal and type the following command to create an application.

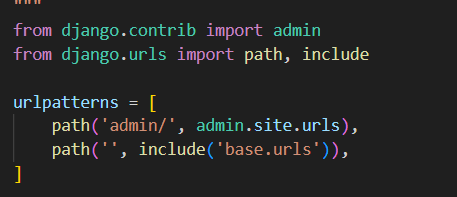
python manage.py start app base

### Step3: Set Up Your Initiative

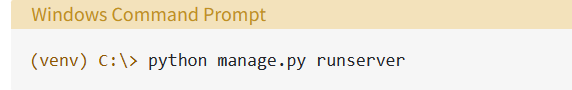
### An array called INSTALLED\_APPS contains a list of app names, beginning with Django. contrib will be visible to us. To meet our needs, Django offers these applications and it had installed them by default. However, our app name is absent in the list. Therefore, the base must be added to the array INSTALLED\_APPS as an item:

### 

* By default, databases are configured to utilize the SQLITE3 database.
* Urls.py, which manages URL lookup at the project level, is the second file in the project folder that must be changed.



* Now, you will generate that file for the app-level URL setup. Create a new file in your editor, then save it as urls.py in the base directory:
* Now that your project and app are set up and functioning, not much happens with them. This command will help us to display our application.



* After editing all the templates, views of the application the same URL will be generated and then if we navigate to the URL we can see our To-do-list application.

A screen shot of a computer

Description automatically generated

* The URL for admin is <http://127.0.0.1:8000/admin>. On clicking this URL we will be using Django admin where you can see the number of users using your application, create a user, delete a user or create tasks for a user.

**Step 4: Define Your Data Models**

### Open your editor and choose the models.py file. Put your data models code in writing.

### Your whole data model is defined in the file models.py.

### Save the files models.py and its two model classes now.

### Step 5: Create the Database

### Manage.py provides these two subcommands, migrate, and make migrations, which aid in automating the process of maintaining the alignment between your code's data model and the real database structure.

### By using make migrations, you may inform Django that you would like to document the modifications you made to the application's data model.

* You may execute instructions against the database with the migrate command to implement those changes.

## 

**Step 6: Create the Django Views**

* A view is a piece of Python code that instructs Django on how to move between pages and what information should be sent for display.
* The element that most nearly resembles an HTML page is the template.

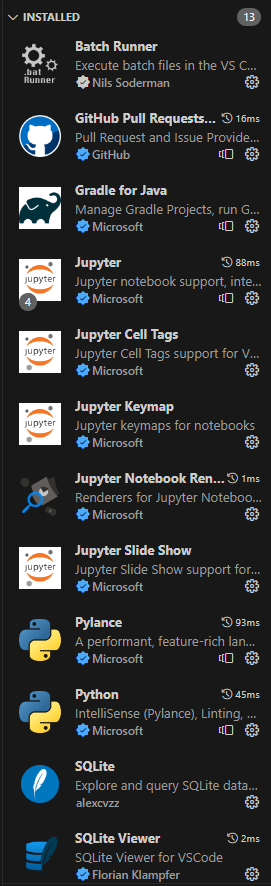
## **Step 7: Delete To-Do Lists and Items**

* To enable users to remove individual items or a list, you will need to add links to the forms in this step. These instances are also handled by the generic views that Django offers.

## **Step 8: Use Your Django To-Do List App**

* The full to-do list app has been developed by us.
* Use the Python mange.py run server command to launch the development server once more. You must fix any issues that the console shows before moving on. If not, go to http://localhost:8000/ in your browser.

**Extensions installed in VS code for project implementation.**



**3.2 GITHUB LINK:**

https://github.com/harikapavuluri/todo\_list\_application.git

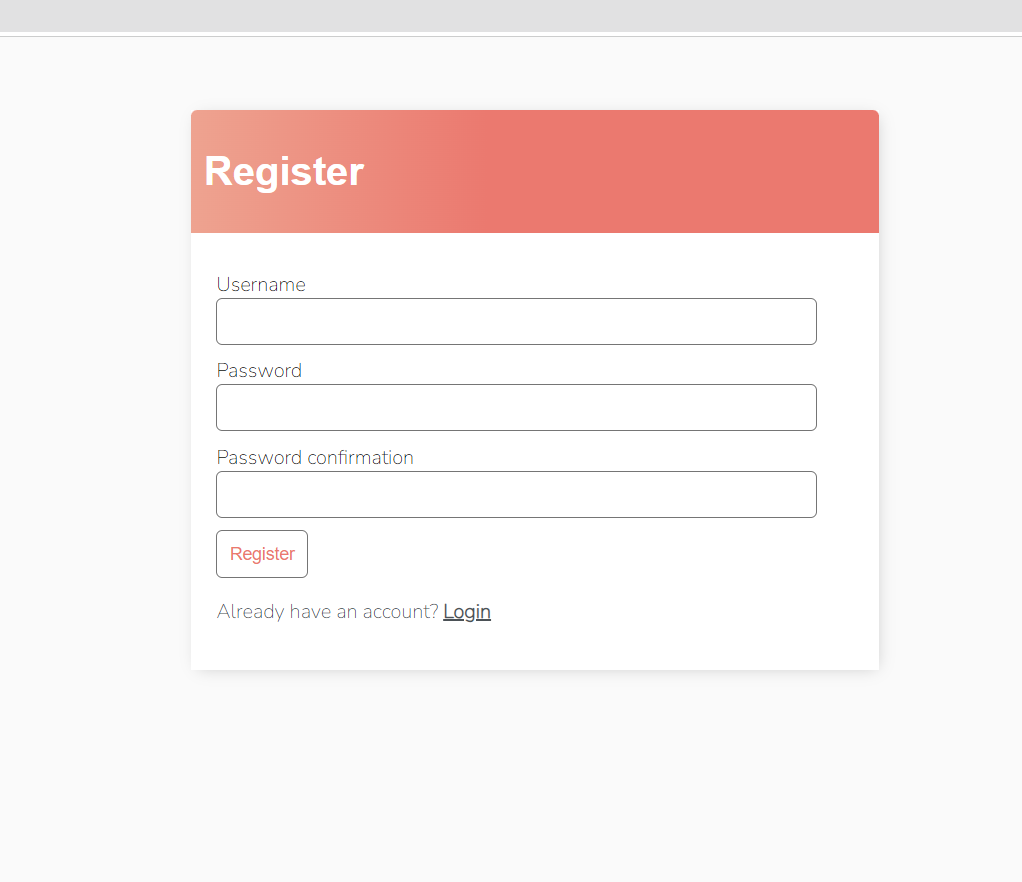
**3.3 SCREENSHOTS OF WORKING APPLICATION**

* Once the URL is clicked, the user will be redirected to this page.

A screenshot of a login form

Description automatically generated

* If the user is new to the application, he will register by entering user id and password.

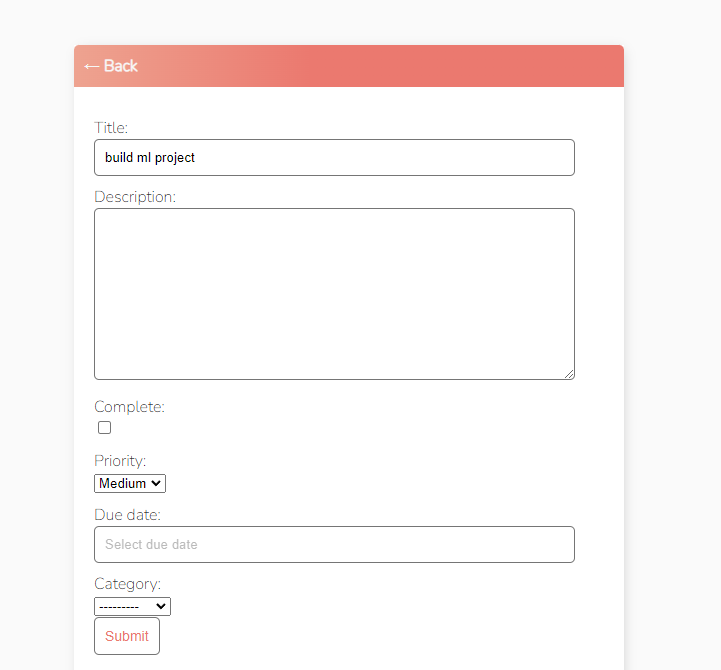


* After successful registration he will be redirected to the application page.

A screenshot of a task

Description automatically generated

* The user will create a new task.



* When a task is completed, the user marks it as complete, and the number of incomplete tasks will be displayed.

A screenshot of a computer

Description automatically generated

* The tasks are displayed in the order of due date and priority. The task whose due date is approaching appears first and if both tasks have same due date, then priority of the tasks is checked.
  + Edit a task

A screenshot of a computer

Description automatically generated

* Clicking on edit gives us this interface.

A screenshot of a computer

Description automatically generated

* Searching functionality can be used to search a task in a list of tasks.

A screenshot of a computer

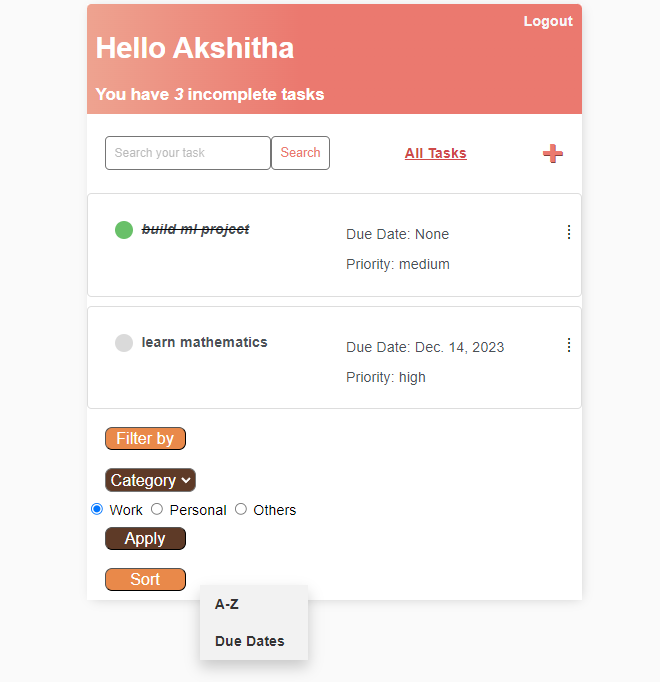
Description automatically generated

* Clicking on delete gives us this prompt.

A screenshot of a computer

Description automatically generated

* This will filter and sort the tasks based on the user’s requirements.



* For example, Filtering the tasks based on work category gives us tasks which are related to work.

A screenshot of a computer

Description automatically generated

* This shows the tasks which are of high priority as I have filtered the tasks based on priority and selected high as priority.

A screenshot of a computer

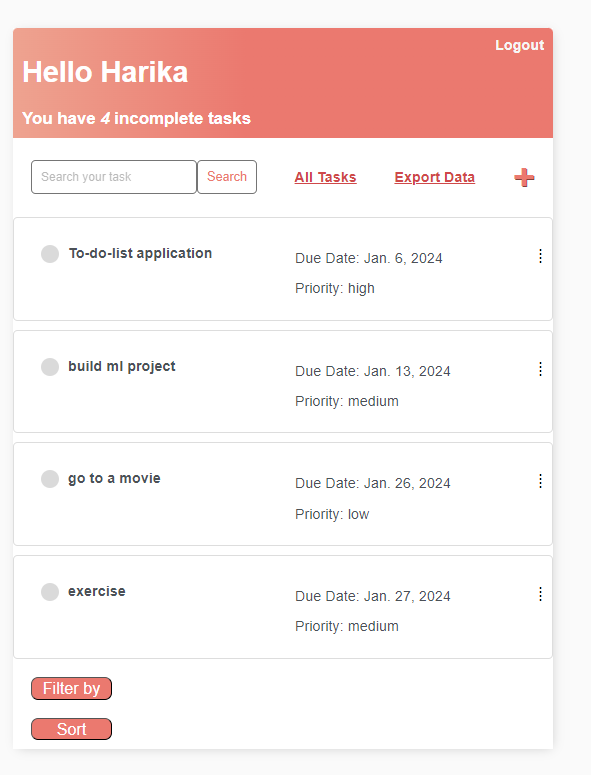
Description automatically generated

* Sorting tasks alphabetically sorts the tasks based on the first letter of the title.

A screenshot of a computer

Description automatically generated

* Sorting tasks based on due dates gives the list of tasks in ascending order.



* Export data is added to application which exports data from application to file.

A screenshot of a computer

Description automatically generated

* I have exported the data to text format, so if we click on it we get output as

A screenshot of a computer program

Description automatically generated

**4.0 TESTING AND VALIDATION**

**IMPORTANCE**

Testing is a procedure that finds program errors. To know whether the software is operating as intended, testing involves running the program through a series of test cases and analysing the results. The process of running software with the goal of identifying errors is called testing.

**4.1 TEST CASES**

The data explains about the test cases in our project. It gives the information about the activity performed by the user and whether the expected output is achieved by the user or not.

User Register – If the user is new to the application, he can create an account by entering the user id and password. Password and username specifications should be met for successful registration.

User Login – After successful registration the user will be redirected to the login page.

Create Task – User creates the task by entering some information like title of the task, description, priority of the task, type of task and due date. Clicking on submit will help user to add the task in application.

Edit task – An icon will be displayed after each task to edit the details in the task.

Mark as Completed – The user can mark the task as completed and the application displays the number of incomplete tasks to the user.

Delete task – Deleting a task gives us prompt to make sure that the user is ready to delete a task or not.

|  |  |  |  |
| --- | --- | --- | --- |
| CASE ID | DESCRIPTION | TEST STEPS | OUTPUT |
| 1 | USER REGISTER | If the user is a new user, they will register by entering user id and password. | Pass |
| 2 | USER LOGIN | Open log in page by entering valid username and password and click on login button. | Pass |
| 3 | CREATE TASK | You can create a new task by giving title and description to the task. | Pass |
| 4 | EDIT TASK | The task created can be also edited | Pass |
| 5 | VIEW TASK | View the details of the task by clicking on the view button. | Pass |
| 6 | MARK TASK AS COMPLETE | If the task is completed mark the task as completed | Pass |
| 7 | DELETE TASK | The user can delete the task if the task is not required. | Pass |

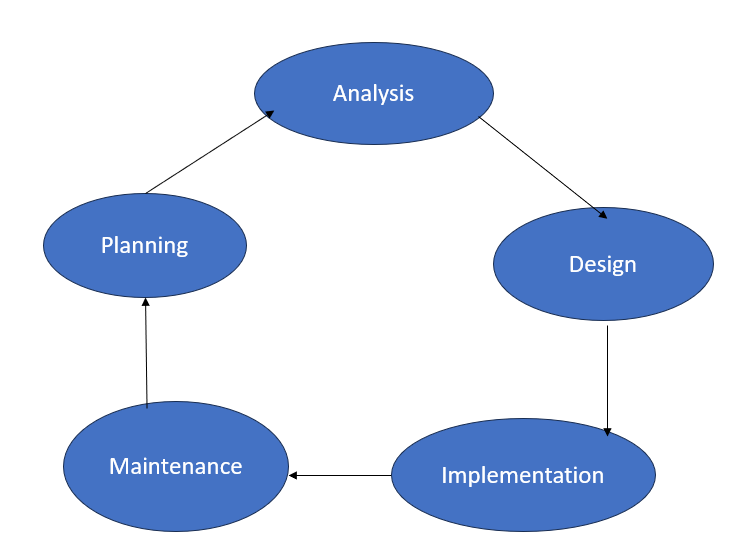
These are the detailed steps which I have followed for testing the application

A screenshot of a computer screen

Description automatically generatedA group of text boxes

Description automatically generated

**5.0 EVALUATION:**

****

For the To-do-list application the problem statement is first analysed, the requirement in our application is seen carefully and then design is drawn for that, the design phase is important as it explains the steps which are performed at each stage. Then this design is implemented carefully and maintained. We further plan according to the implementation and design of the application. In this evaluation phase we can find the accuracy of our application, errors which we might get and the difference between final output and the expected output. I have drawn a flow chart for the application based on the requirement and then proceeded for the further steps.

**5.1 Functional Requirements**

**a. Software Requirements**

* Operating System – Any Operating system that supports Python (such as Windows, Linux, macOS)
* Language – Python 3.8 or above
* Framework - Django 4.1.7
* Web Browser - Google Chrome / Microsoft Edge / Firefox etc.,​

**b. Hardware Requirements**

* Ram – Computer with at least 4GB of RAM recommended.
* A hard drive with at least 5gb of space
* A modern CPU which consists of Intel or ARM processors.

**5.2 Planning and Efficiency**

**Strengths:**

* Clear project planning.
* Efficient task management.

**5.3 Code Elegance**

Strengths:

* Consistent coding style.
* Use of appropriate design patterns.

Weaknesses:

* Limited comments for complex sections.

**5.4 Application Usability**

Strengths:

* Intuitive user interfaces
* Quick task creation and management

Weaknesses:

* Limited customization options for user preferences.

**6.0 REFERENCES**

[1] Python Software Foundation. (2022) Python Language Reference. Available at: https://docs.python.org/3/reference/

[2] Django Software Foundation. (n.d.). Views. [Online]. docs.djangoproject.com. Available at: <https://docs.djangoproject.com/>

[3] Todo App Development Team. (2022) AwesomeToDoApp Repository. Available at: <https://github.com/ToDoApp/awesome-todo-app>

[4] [Belal ali al-maytami](https://www.researchgate.net/profile/Belal-Ali-Al-Maytami?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uIn19) ,[Pingzhi Fan](https://www.researchgate.net/scientific-contributions/Pingzhi-Fan-2119797045?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uIn19) (2019) 'A Comparative Study of Task Scheduling Algorithms in To-Do List Applications'. IEEE Transactions on Software Engineering, 45(2), 78-92.

[5] [Eline Jongmans](https://www.researchgate.net/profile/Eline-Jongmans?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uIn19), [Maud Damperat](https://www.researchgate.net/profile/Maud-Damperat?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uIn19) (2020) 'Strategies for Enhancing User Experience in To-Do List Applications'. User Experience Magazine, 25(4), 45-56.

[6] Irfan Ahmed, Ali Kashif Bashir (2021) 'An In-depth Security Analysis of Mobile To-Do List Applications'. Journal of Cybersecurity Research, 10(1), 112-128.