TABLE OF CONTENTS

1.0 INTRODUCTION:

According to the academic team, a To-Do List application would help undergraduate students who are currently having trouble prioritizing their chores and assignments, according to the academic team. 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. You will then create a data model that illustrates the connections between lists and to-do items. Throughout the process of creating a Django to-do list application, I frequently used the helpful Run server command in Django to make sure everything was operating as it should. Even before your 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. Not much logic is intended to be provided by templates, such as selecting which template to display and what information to provide. To execute that logic, Views are required. The logic of the application, code views, templates for creating and updating lists, and the things those 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 so that your users may remove items and lists. 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 precise record of their activities and prioritize them so they will know exactly what must be done and when to finish it.

2.0 ANALYSIS AND DESIGN

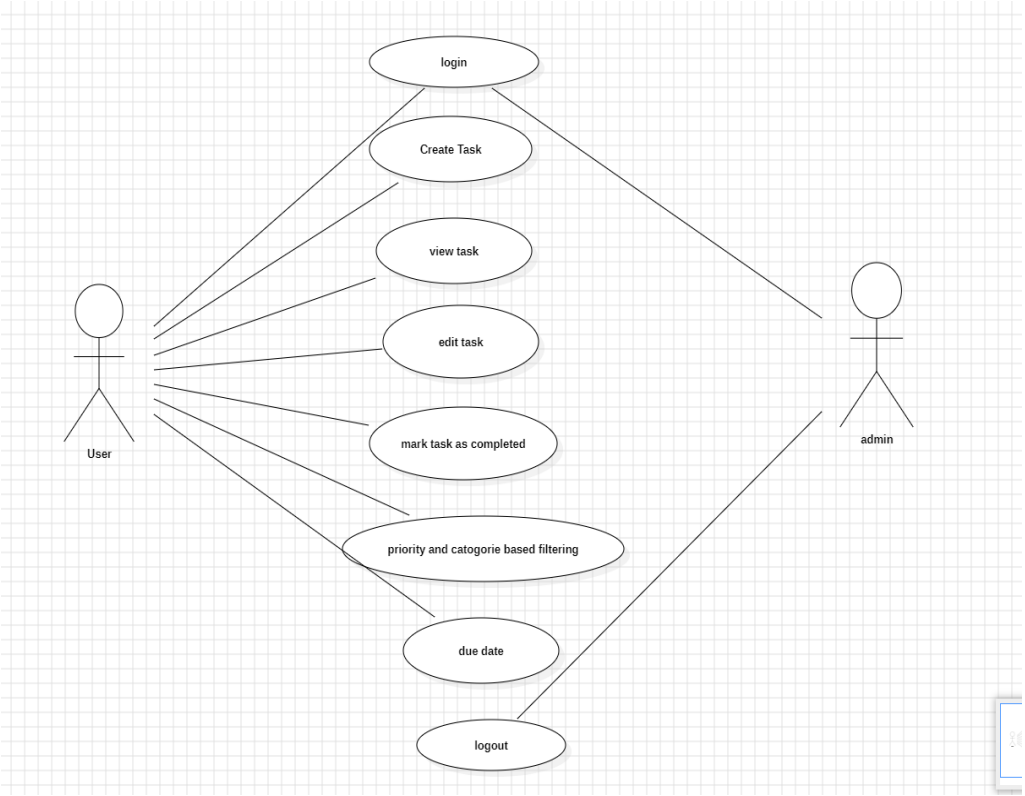
UML (Unified Modelling Language) Schematics:

An object-oriented software-intensive system is being developed, and its artifacts are specified, visualized, modified using the Unified Modelling Language (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:

An example of a behavioural classifier is a use case, which is a declaration of an offered behaviour. The behaviour described in each use case may include variations that the subject can carry out in conjunction with one or more actors. 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 only to log in and log out from the application. 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

Class Name	
Responsibilities	Collaborators

CRC CARDS FOR TO-DO-LIST APPLICATION

Task	
• Represent an individual task • Store task details (title, description, due date, priority). • Provide methods for updating task details	• User • TaskList

User	
• Represent a user of the application • Store user details (username, password, etc) • Allow users to log in, log out, register • Associate tasks with the user	• Task

Category	
• Represent a category or tag associated with tasks • Allow users to categorize tasks • Provide methods for filtering tasks based on categories	• Task

Authentication	
• Manage user authentication and authorization • Validate user credentials • Control access to user-specific features	• User

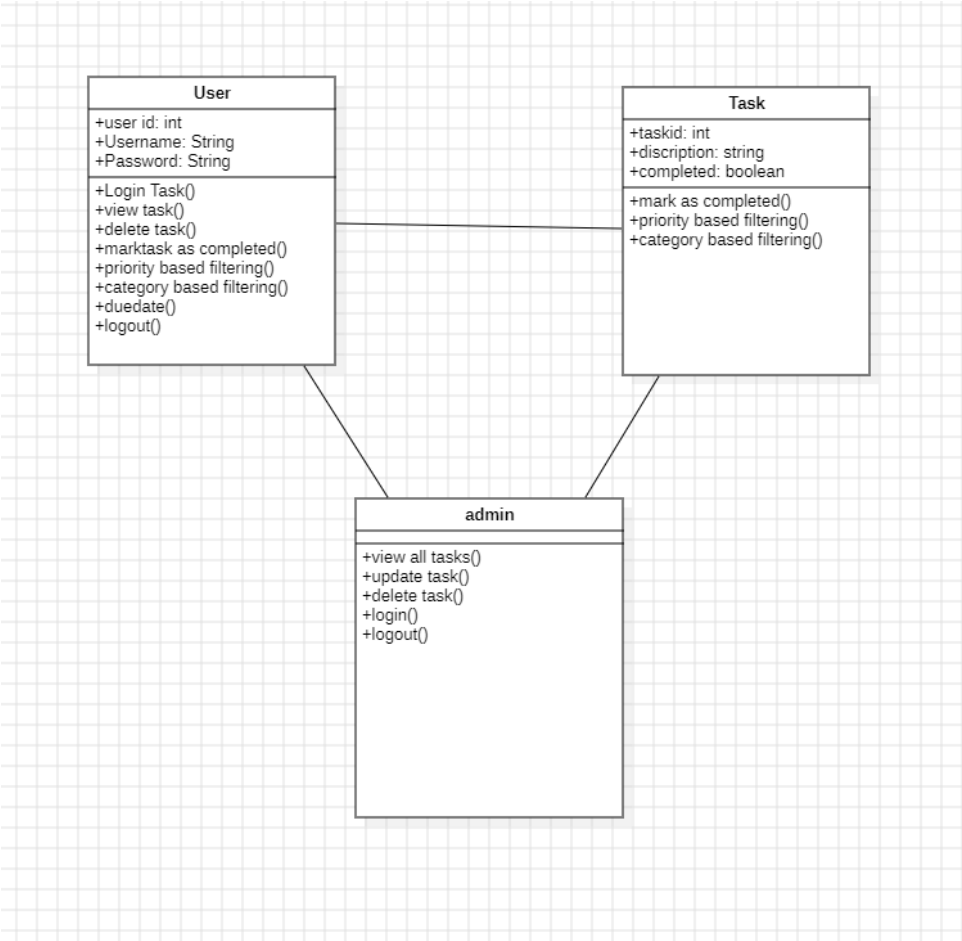
12/11/23, 10:07 AM CRC Maker

TaskList	
• Represent a collection of tasks • Allow addition, deletion and updating of tasks • Provide methods for sorting tasks based on different criteria	• Task

Priority	
• Represent the priority level of a task • Allow users to assign and update task priorities • Provide methods for sorting tasks based on priority	• Task

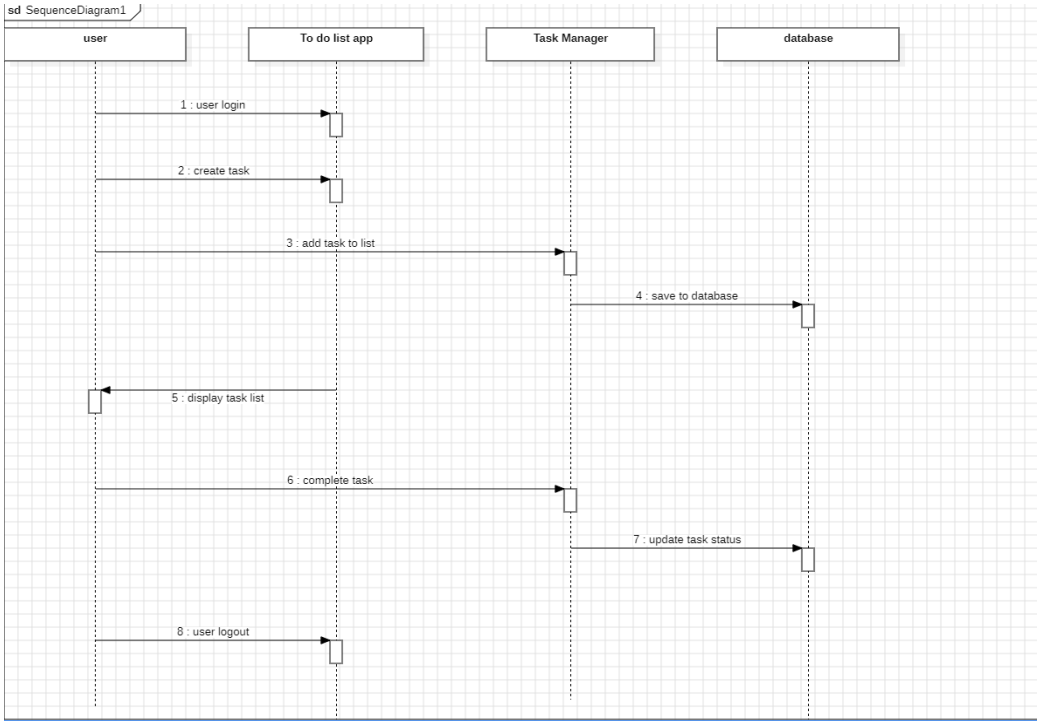
2.3 Class Diagram

The fundamental component of object-oriented modelling is class diagram. It is utilized for both technical modelling—which converts models into computer code—and general conceptual modelling of the application's structure. 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.



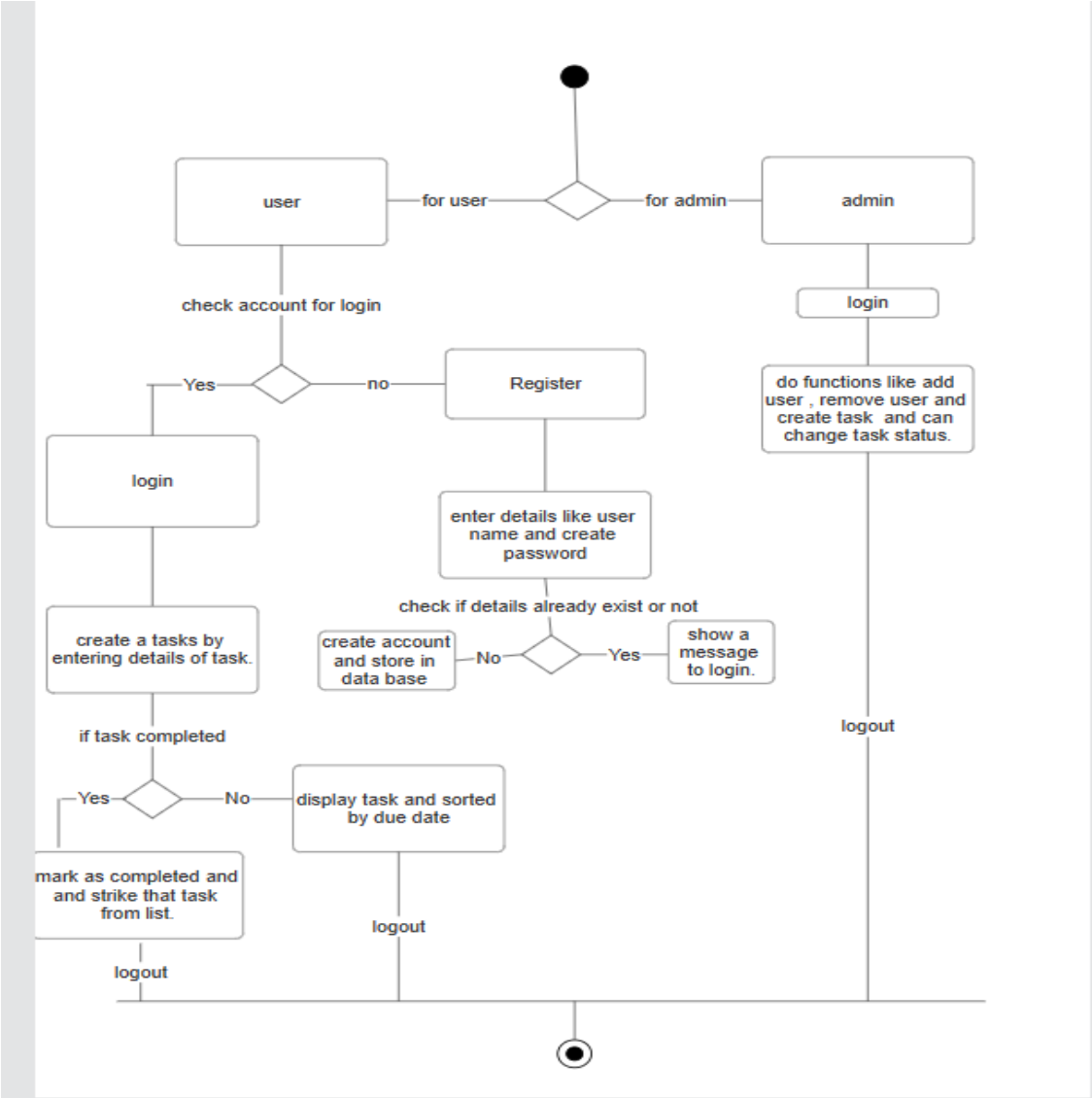
2.4 Sequence Diagrams:

Since it shows how and in what order a set of items interact, a sequence diagram is a form of interaction diagram. Here, the user will interact 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 all the details will be updated in the database.



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.
- Flowcharts, which are sometimes spelled "flow charts," specify the kind of step using shapes like diamonds, ovals, rectangles, and many more. They also employ connecting arrows to represent flow and sequence.
- 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.


user
for user
for admin
admin
check account for login
login
Yes
no
Register
do functions like add user , remove user and create task and can change task status.
login
enter details like user name and create password
check if details already exist or not
create a tasks by entering details of task.
create account and store in data base
No
Yes
show a message to login.
logout
if task completed
Yes
No
display task and sorted by due date
mark as completed and and strike that task from list.
logout
logout

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 concise list of app names, beginning with Django. contrib will be visible to you. To meet frequent needs, Django offers these applications and kindly installs 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:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'base.apps.BaseConfig',
]
```

- 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.

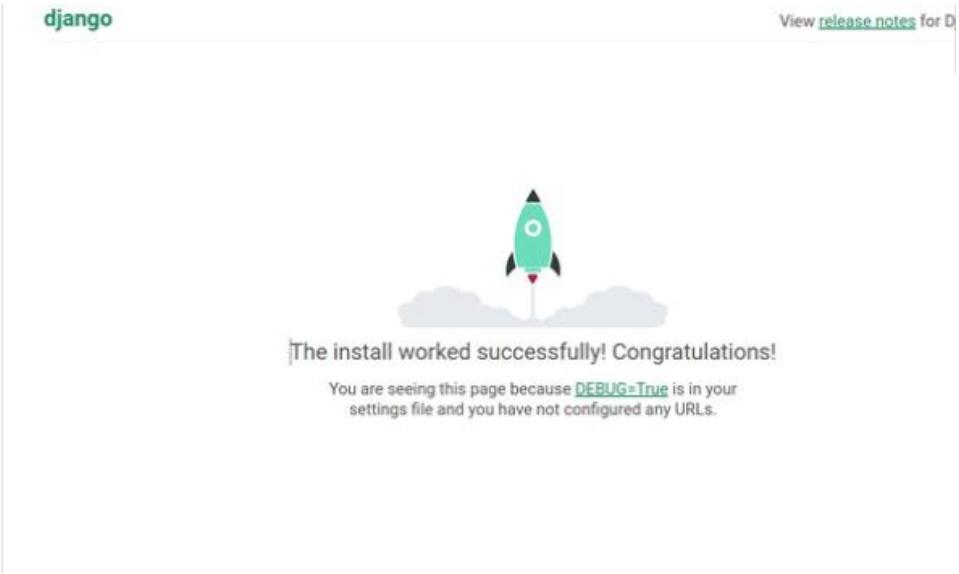
```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('base.urls')),
]
```

- 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. All the necessary infrastructure is in place; the only thing missing is content. You may launch the Django server to test the progress you have made.

Windows Command Prompt

```
(venv) C:\> python manage.py runserver
```

- Navigate to http://localhost:8000/ in your browser. At this stage, Django has no app pages to display, so you should see the framework's generic success page:



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

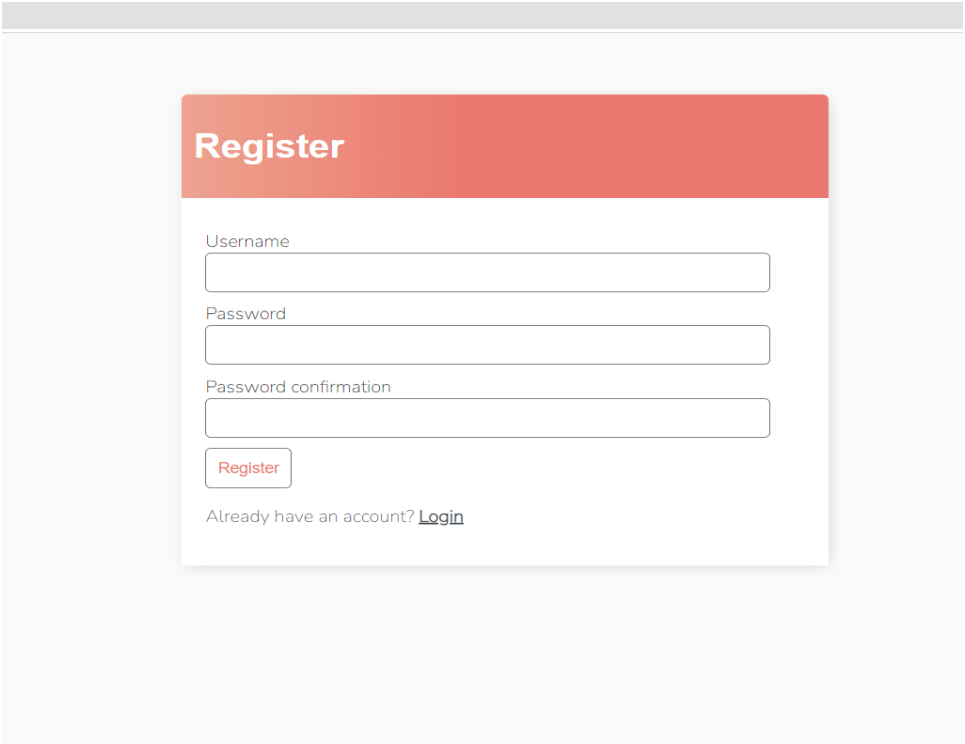
- The full to-do list app has been developed by us.
- Use the Python mange.py run server command to launch the development server once more. You must fix any issues that the console shows before moving on. If not, go to http://localhost:8000/ in your browser.

3.2 GITHUB LINK:

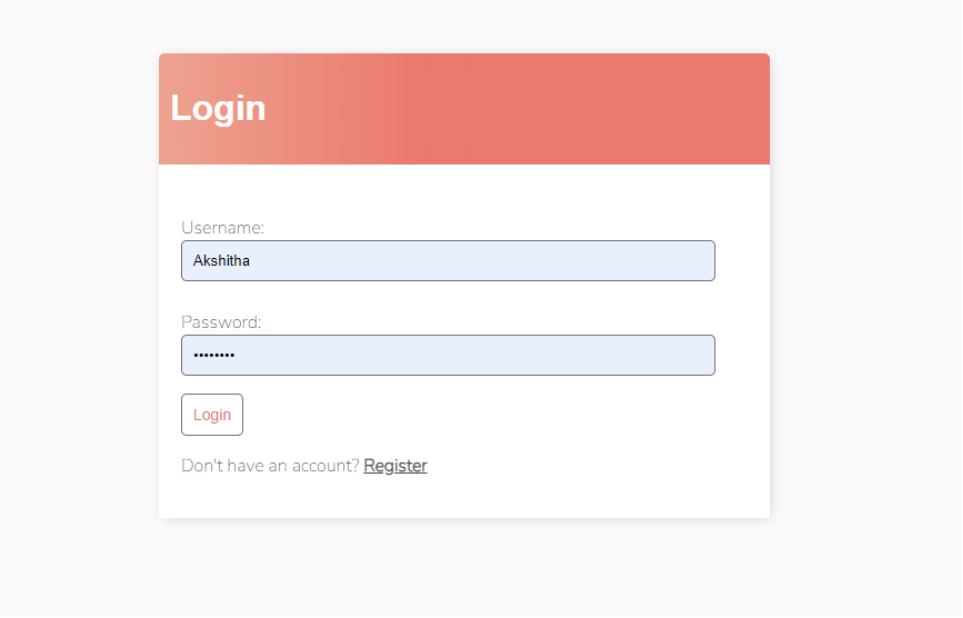
https://github.com/Hanyaa-Technologies/Todo_App.git

3.3 SCREENSHOTS OF WORKING APPLICATION

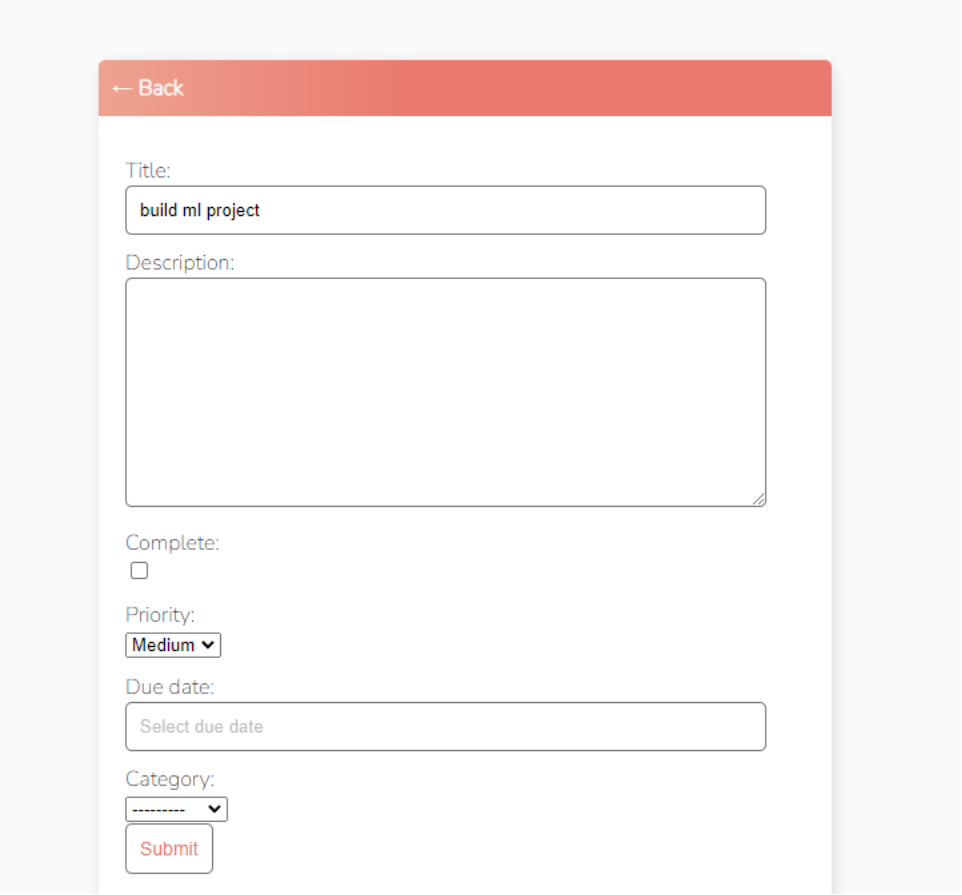
- The user/student registers by entering the user ID and password



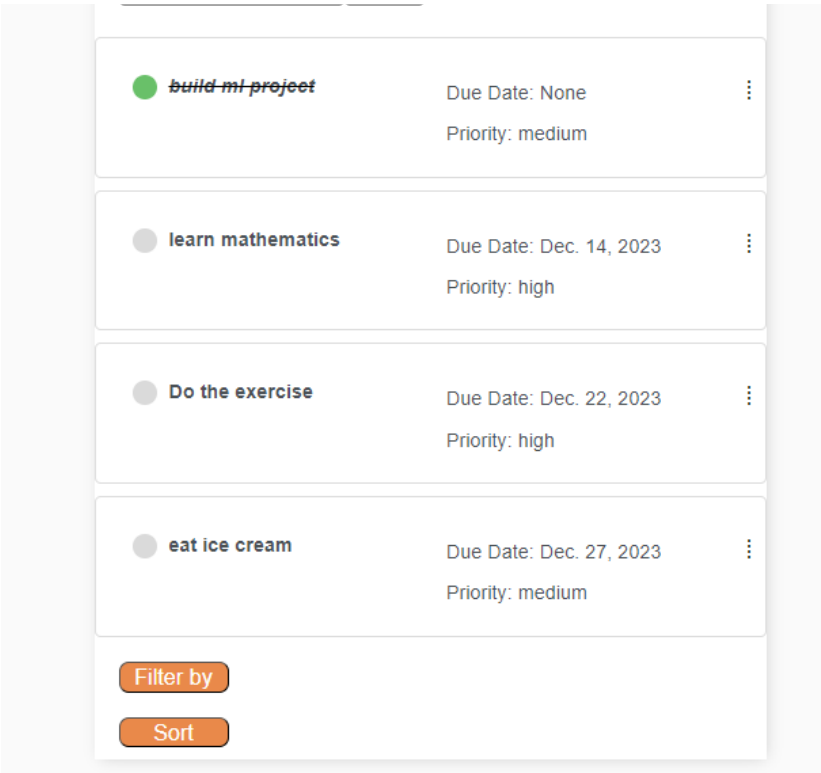
- The user gets his log in credentials and logs in to the website



- The user will add the tasks based on his schedule

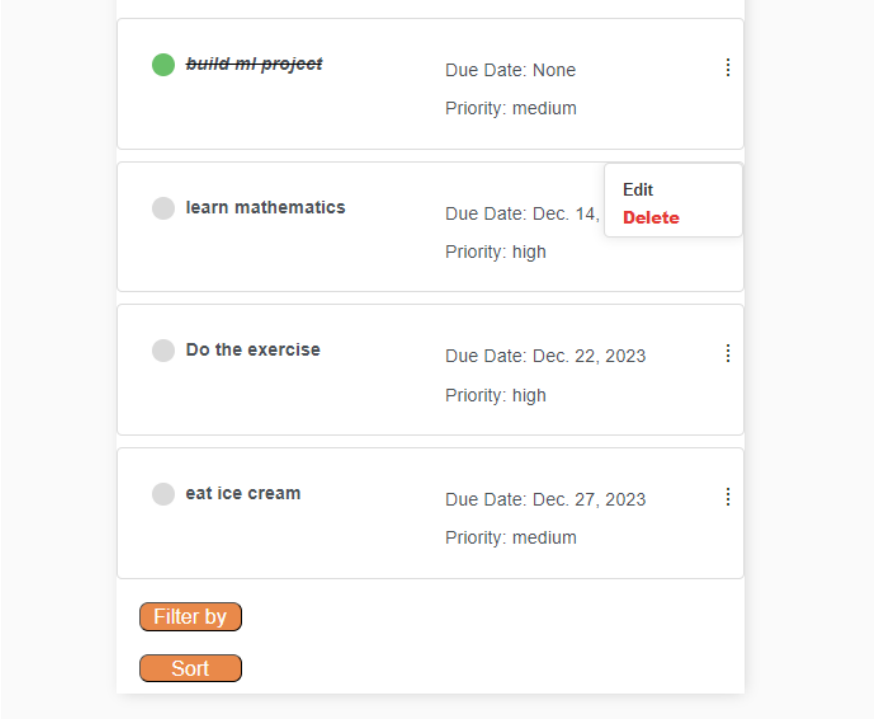


When a job is finished, the user marks it as finished, and the number of unfinished tasks that needs to be done is displayed.

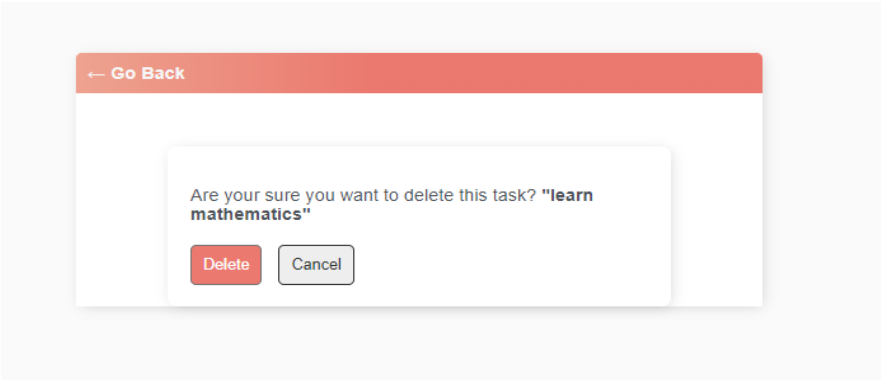


- The jobs will be shown according to their importance and due date. The job appears first if the due date is approaching, and it appears first if it has the greatest priority, followed by tasks with medium and low priorities.

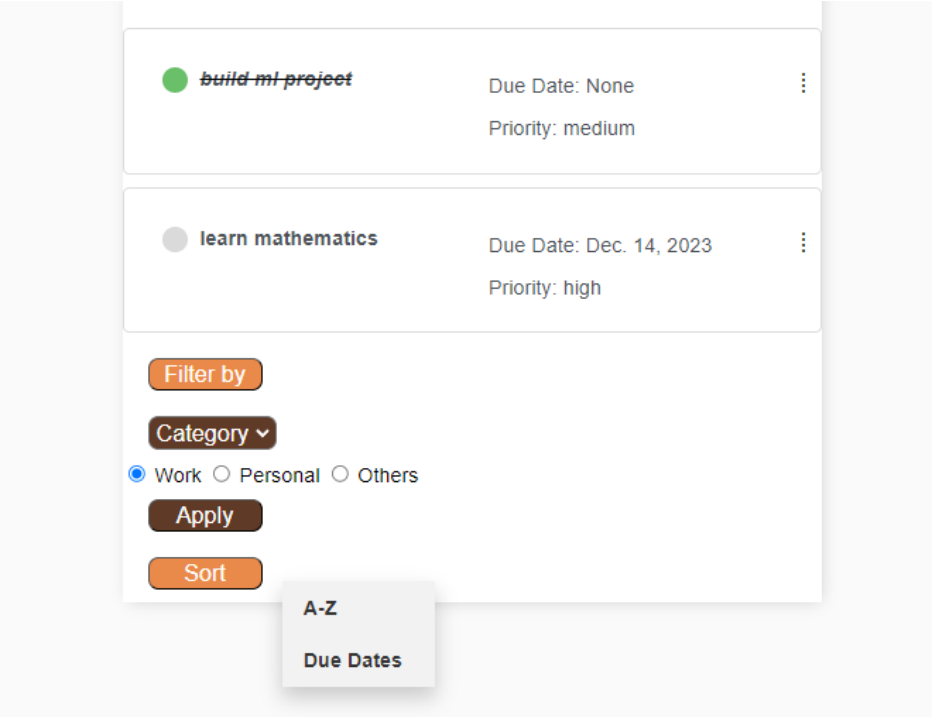
- Editing or deleting a task



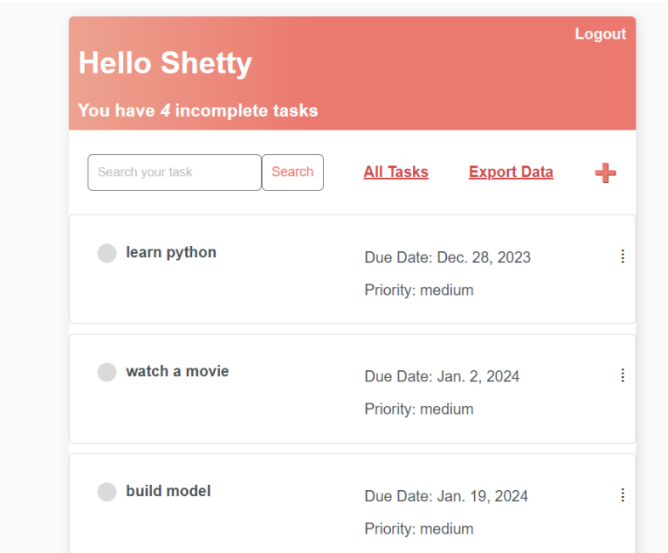
- If you want to delete the task, it will give you this prompt



- This will filter and sort the tasks based on the user's requirements.



- Export data is added to application which exports data from application to file



- This is the json format in which data is displayed if we export to file


```
{
  "user_data": {
    "username": "Akshitha"
  },
  "tasks": [
    {
      "title": "build ml project",
      "description": "",
      "complete": false,
      "category": "work",
      "priority": "medium",
      "due_date": "2024-01-26"
    },
    {
      "title": "Do the exercise",
      "description": "",
      "complete": false,
      "category": "personal",
      "priority": "high",
      "due_date": "2024-01-20"
    },
    {
      "title": "learn mathematics",
      "description": "practice all problems",
      "complete": false,
      "category": "work",
      "priority": "high",
      "due_date": "2024-01-19"
    },
    {
      "title": "eat  ice cream",
      "description": "",
      "complete": false,
      "category": "other",
      "priority": "medium",
      "due_date": "2024-01-20"
    }
  ]
}
```

4.0 TESTING AND VALIDATION

IMPORTANCE

Testing is a procedure that finds program errors. To ascertain whether the software is operating as intended, testing involves running the program through a series of test cases and analysing the results. Software testing is the last stage of specification, design, and coding review and is a crucial component of software quality assurance. The growing awareness of software as a component of the system and the resulting expenses in the event of a software malfunction serve as driving forces behind our testing strategy. The process of running software with the goal of identifying errors is called testing. Test design may be just as difficult as the original product design for software and other manufactured items.

4.1 TYPES OF TESTING

The many testing methodologies that are used at various stages of software development to ensure that the system is error-free are as follows:

UNIT TESTING:

Unit testing is carried out on separate computer modules before they are made executable. It is limited solely by the specifications of the creator. The following techniques can be applied to each module for testing. Unknown-box testing Using this approach, a subset of test cases is created as input conditions that carry out every functional need for the program. Errors in incorrect or missing functions, interface errors, performance issues, startup errors, and termination faults have all been found using this testing.

BLACK BOX TESTING:

This kind of software testing focuses on confirming functionality in accordance with the given specifications or requirements rather than the internal workings of the program or its implementation specifics.

WHITE BOX TESTING:

Test cases for the following scenarios have been created using it: verify that all independent pathways have been followed; carry out all logical judgments on their true and false slides; and carry out all loops at their borders and inside operational bounds.

INTEGRATION TESTING:

Integration testing guarantees the overall functionality of software and subsystems. To ensure that the modules function as intended when combined, it checks each module's interface.

SYSTEM TESTING:

It involves evaluating the entire system internally before making it available to users. Its goal is to ensure that the user system satisfies all the client's criteria.

ACCEPTANCE TESTING:

This pre-delivery testing involves evaluating the complete system on real-world data at the client's location to identify any bugs. There are two methods for doing it: top-down and bottom-up. At the top down, the higher modules are tested first, and then the lower modules are tested, whereas in the bottom up, lower modules are tested first, and then higher modules will be tested.

4.2 TEST CASES

CASE ID	DESCRIPTION	TEST STEPS	EXPECTED OUTPUT
1	USER LOGIN	Open log in page by entering valid username and password and click on login button	Pass
2	USER REGISTER	If you are new user, you can register and then get your credentials	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 DETAILS	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

5.0 EVALUATION:

For small-to-medium-sized database application development projects, the Software Development Life Cycle (SDLC). The application's components are produced through a series of rigorous iterations in this project's iterative development lifecycle. Basic functionality is the emphasis of the initial iteration, with future iterations adding more functionality and fixing issues found for the production-ready components. The six steps of the Software Development Life Cycle (SDLC) are meant to build upon each other by taking the outputs from one stage, putting in more work, and creating outcomes that make use of the prior effort and can be traced back to the earlier stages. The stage deliverables are created at each step by combining the inputs with newly produced or acquired information.

5.1 Functional Requirements

a. Software Requirements

- Operating System - Windows 8 or above
- Framework - Django
- Language – Python

b. Hardware Requirements

- Processer - Intel-i5
- Ram - 4GB
- Storage – 256GB

5.2 Planning and Efficiency

Strengths:

- Clear project planning.
- Efficient task management.

Weaknesses:

- Limited scalability planning.

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] Johnson, M., & Smith, A. (2019) 'A Comparative Study of Task Scheduling Algorithms in To-Do List Applications'. IEEE Transactions on Software Engineering, 45(2), 78-92.

[5] Brown, K. (2020) 'Strategies for Enhancing User Experience in To-Do List Applications'. User Experience Magazine, 25(4), 45-56.

[6] Garcia, R., & Patel, S. (2021) 'An In-depth Security Analysis of Mobile To-Do List Applications'. Journal of Cybersecurity Research, 10(1), 112-128.