

SKILLVERTEX MINOR PROJECT

ISHAN CHAKRABARTI

Introduction:

The goal of this project is to create a “To-Do List” using Django Web Framework. Using this app, the user can set up a to-do list. Within the list, a schedule for their daily tasks can be created. This is achieved by allowing the users to add, modify and delete items from the list. A custom “due date” can also be set, although the default time frame is one week.

Requirements:

- Any modern – day PC
- Python 3.6 or higher
- Any Code Editor (PyCharm , VSCode, Notepad++, etc.)
- Internet access
- Django (version 3.9 or higher)

Project Overview:

This project uses Django (and a bit of HTML) to create the to-do app. The steps involved are as follows:

1. Creating a Virtual Environment & Installing Django:
 - i) Open Command Prompt
 - ii) Create a virtual environment by typing: “python -m venv venv”
 - iii) Activate the virtual environment by typing: “cd venv/Scripts/activate”
 - iv) Install Django by typing: “python -m pip install django”
2. Creating a superuser:
 - i) Use the command: `python manage.py createsuperuser` to create credentials for Django at <http://127.0.0.1:8000/admin/>
3. Creating the to-do app:
 - i) Use the command: `django-admin startproject todo_project` . to create a manage.py file
 - ii) Use the command: `django-admin startapp todo_app` to create an application
 - iii) The following files are of importance to the to-do app: models.py, urls.py, settings.py and views. py.
 - iv) models.py: It defines the database of a web application by declaring tables, fields, behavior of particular fields and the meta–data of the database.
 - v) urls.py: It contains the tuple “urlpatterns” which defines the mapping between URLs and views.
 - vi) settings.py: It holds all the configuration values required for the application to work.
 - vii) views.py: It receives a HTTP request and generates a HTTP response
 - viii) An HTTP request is sent to urls.py. It gets forwarded to the appropriate class in views.py. The views.py file exchanges data with models.py and along with the template for that particular page, an HTTP response is generated.

4. Databases:

- i) Use the command: `python manage.py makemigrations` to record changes made to files such as `views.py` and `models.py`
- ii) Use the command: `python manage.py migrate` to save the changes made and put these into effect.

5. Templates:

- i) The following HTML templates have been used to create the to-do list:
 - a) front.html: This file contains the HTML code for the front page of the to-do app.
 - b) index.html: This file displays the available lists, or the message: "No lists have been created!", if there are no available lists. It also contains a button to create a new list.
 - c) todo_list.html: This file displays the available items within a particular list, or the message: "Empty List!", in case there are no items in the list. It has button for adding new items to the list, as well as deleting the list.
 - d) todoitem_confirm_delete.html: This file shows a confirmation page to confirm the deletion of an item from the list.
 - e) todoitem_form.html: This file contains the code for the page that allows the user to add or delete an item from a list. It also contains a "Cancel" button to return to the List view page.
 - f) todolist_confirm_delete.html: This file shows a confirmation page to confirm the deletion of a list.
 - g) todolist_form.html: This file contains the code for the page that allows the user to delete a list.

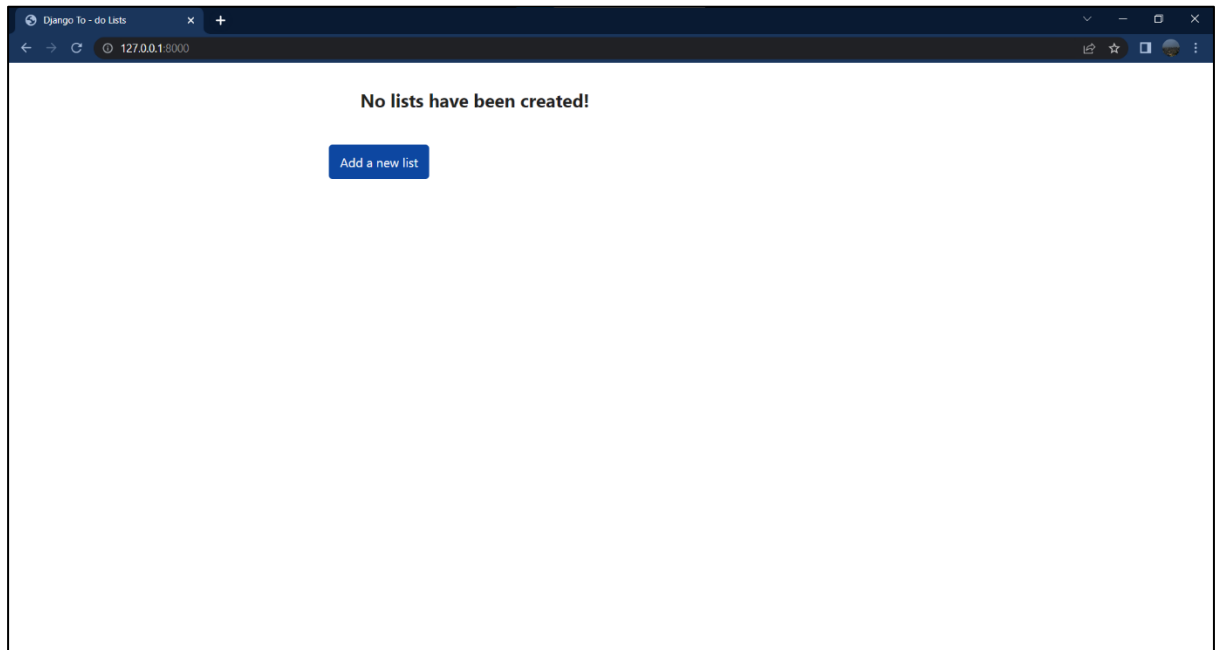
6. Running the Code:

- i) The code is run using the command: `python manage.py runserver`
- ii) The link <http://127.0.0.1:8000/> is used to view the final project

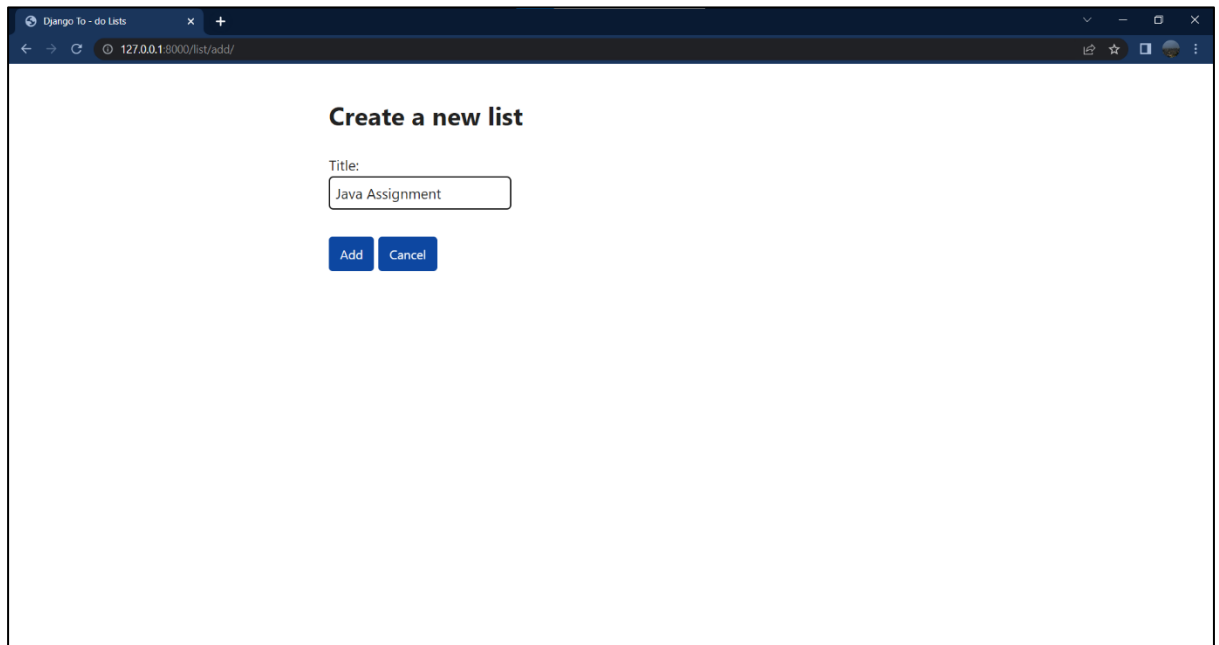
Results:

The outcome of this project has been shown below:

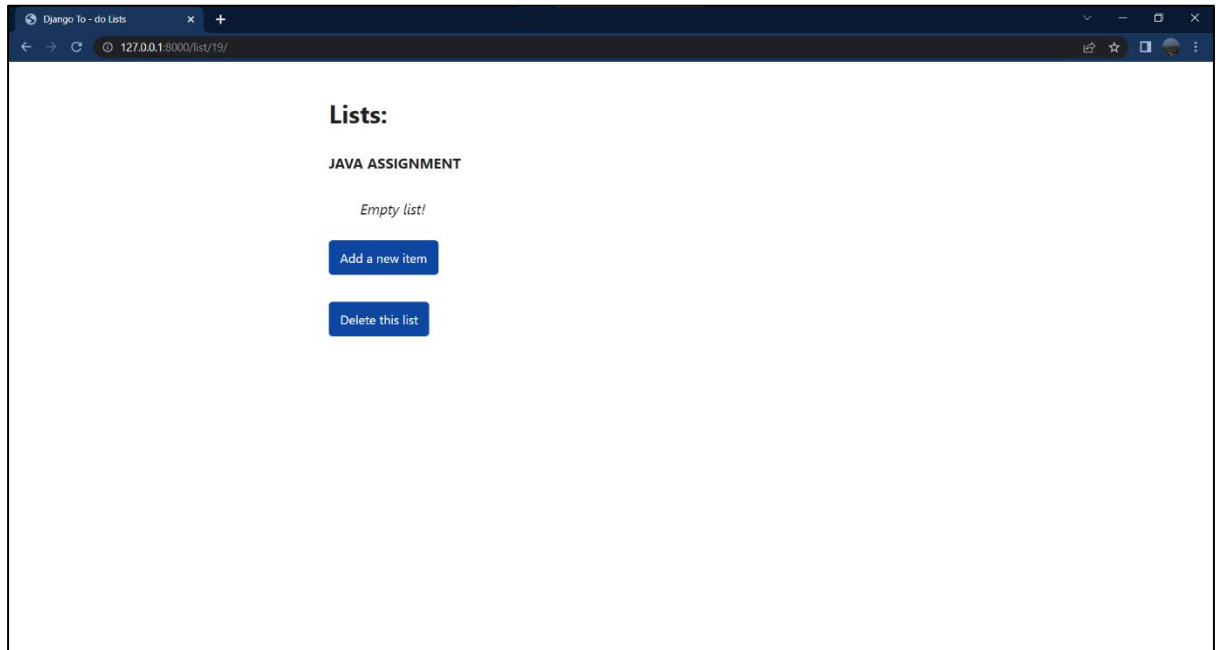
1. Front Page (No Lists have been created):



2. Creating a new List:



3. Empty List:



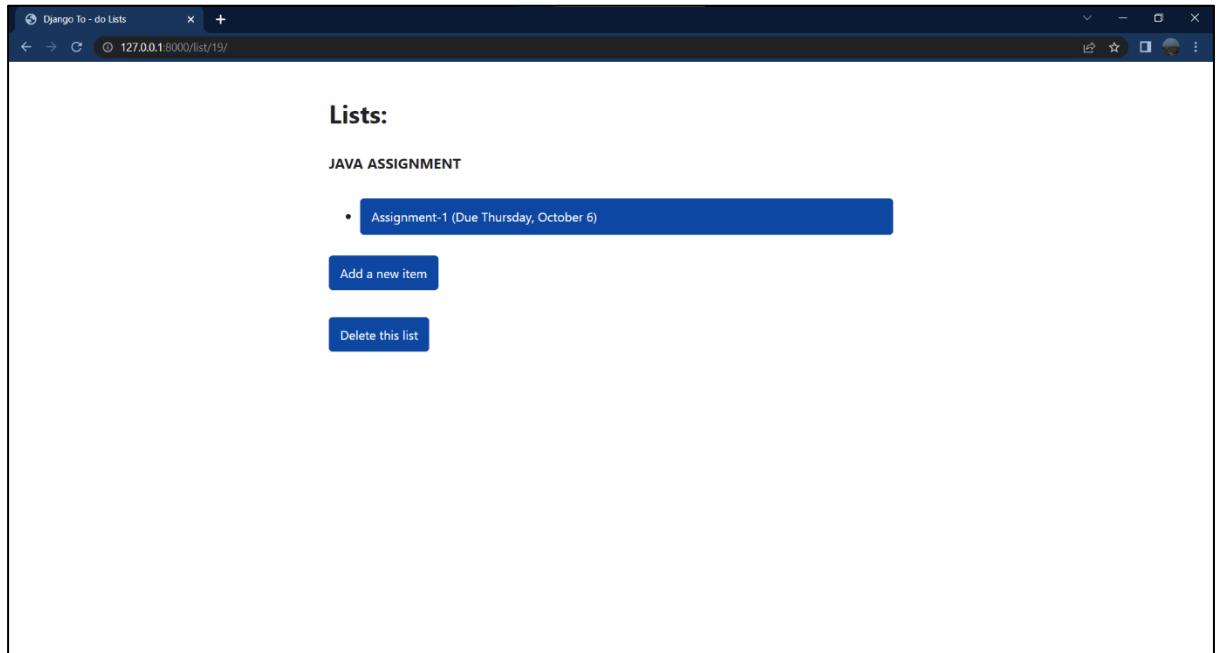
4. Adding an item to the List:

A screenshot of a web browser window showing the 'Add a new item' form. The browser's address bar displays the URL `127.0.0.1:8000/list/19/item/add/`. The form is titled **Add a new item** and contains the following fields:

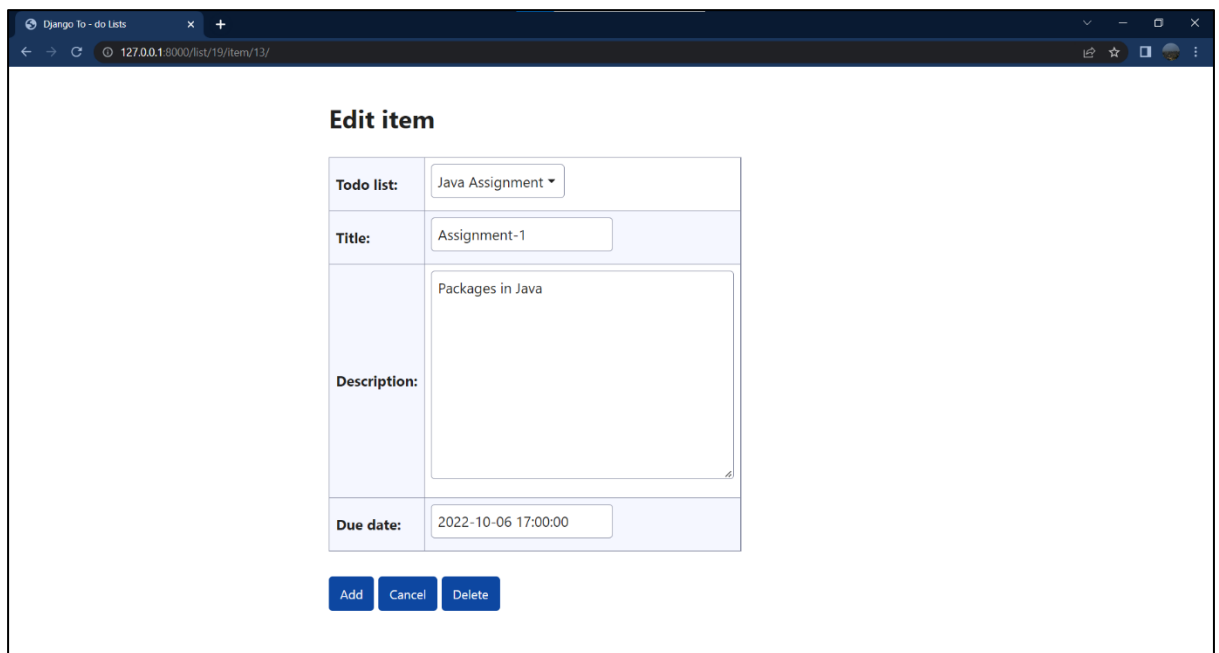
- Todo list:** A dropdown menu with 'Java Assignment' selected.
- Title:** A text input field containing 'Assignment-1'.
- Description:** A text area containing 'Packages in Java'.
- Due date:** A date and time input field containing '2022-10-06 17:00:00'.

At the bottom of the form, there are two blue buttons: **Add** and **Cancel**.

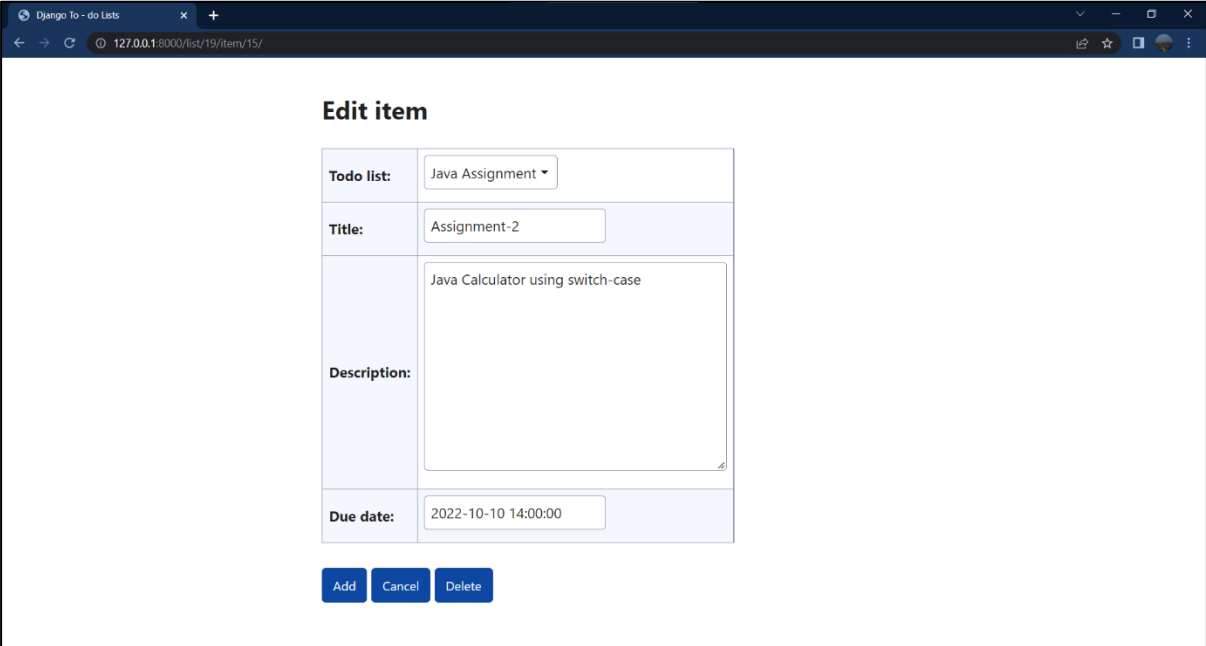
5. Item has been added to the List:



6. Edit view for item:



7. Adding another item:

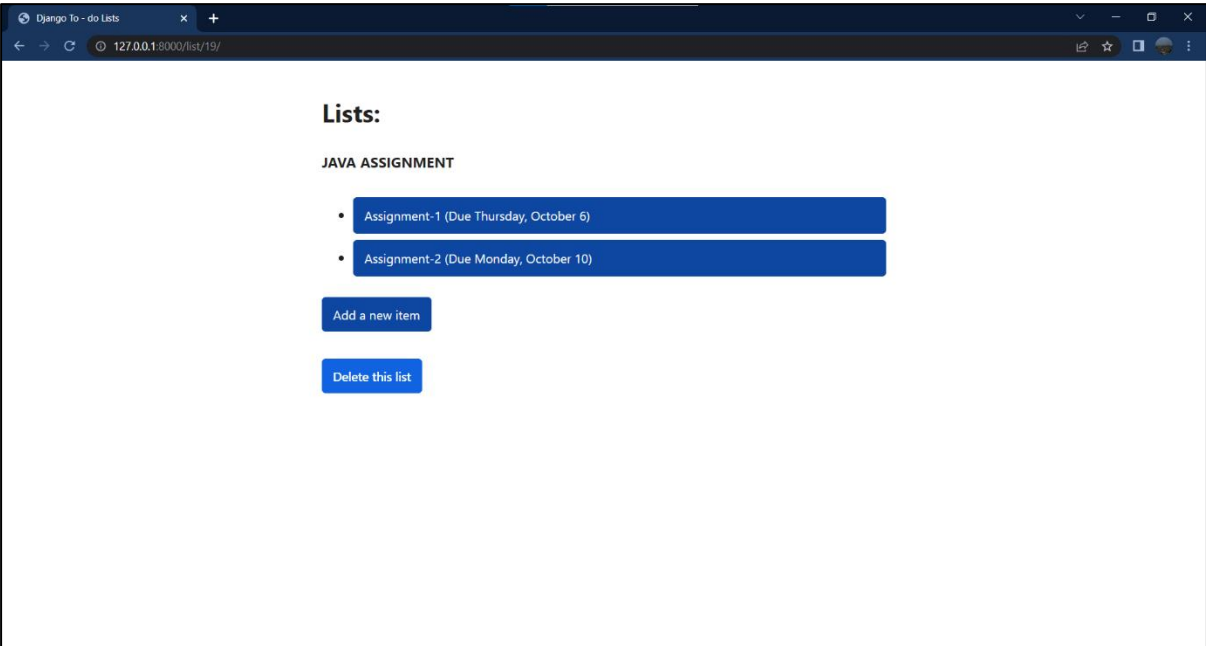


Edit item

Todo list:	Java Assignment
Title:	Assignment-2
Description:	Java Calculator using switch-case
Due date:	2022-10-10 14:00:00

[Add](#) [Cancel](#) [Delete](#)

8. List view after adding second item:



Lists:

JAVA ASSIGNMENT

- Assignment-1 (Due Thursday, October 6)
- Assignment-2 (Due Monday, October 10)

[Add a new item](#)

[Delete this list](#)

9. Deleting the first item:

i) Edit view:

Edit item

Todo list:	Java Assignment ▾
Title:	Assignment-1
Description:	Packages in Java
Due date:	2022-10-06 17:00:00

[Add](#) [Cancel](#) [Delete](#)

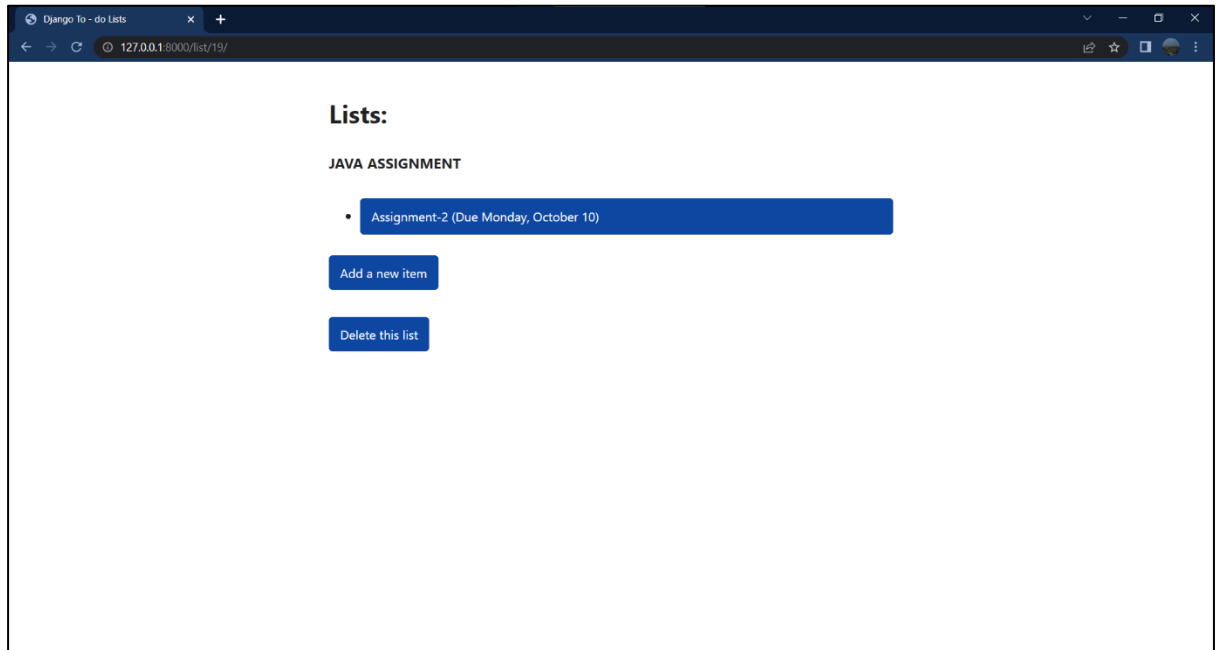
ii) Confirmation page:

Delete Item

Delete item **Assignment-1** from the list **Java Assignment**?

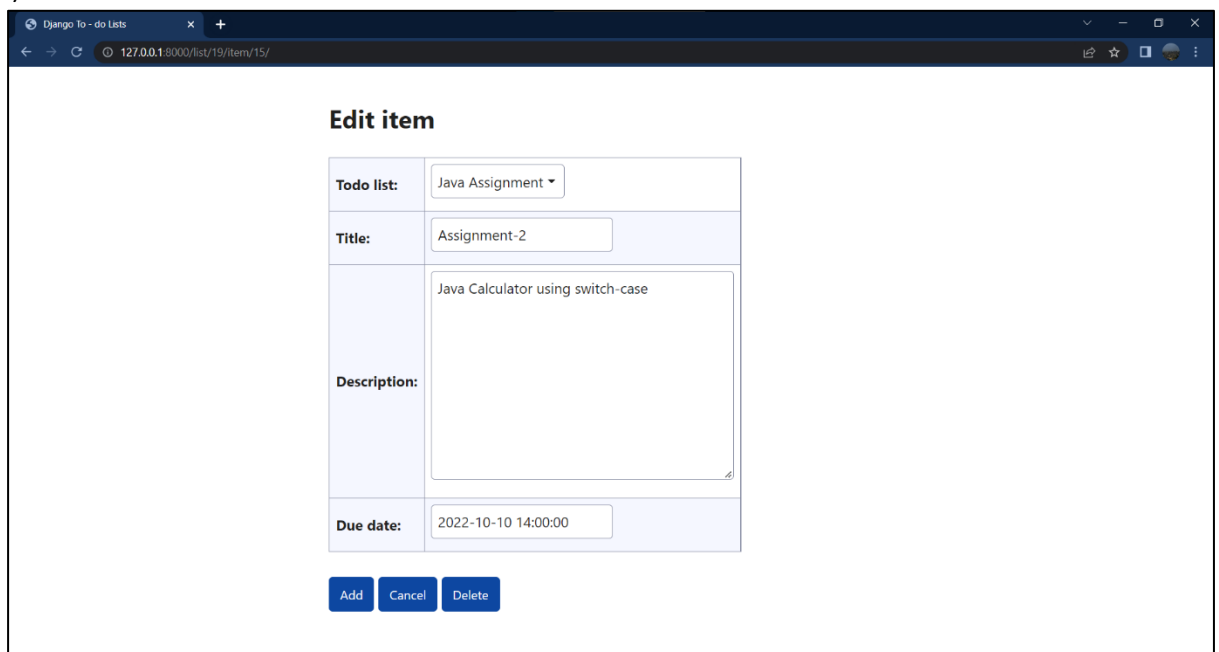
[Yes](#) [No](#)

10. List view after first item has been deleted:



11. Modifying Description and Due date for second item:

i) Before modification:



ii) Modifying Description:

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000/list/19/item/15/'. The page title is 'Django To - do Lists'. The main content area is titled 'Edit item' and contains a form with the following fields:

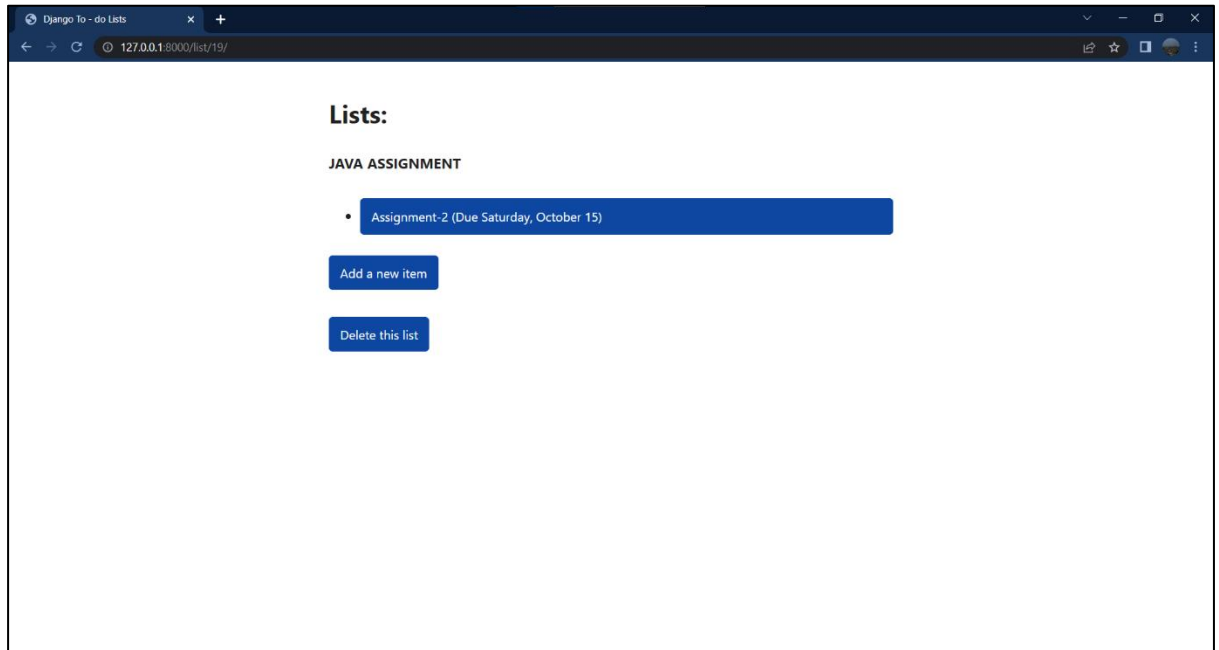
- Todo list:** A dropdown menu with 'Java Assignment' selected.
- Title:** A text input field containing 'Assignment-2'.
- Description:** A large text area containing 'Java Calculator without using switch case'. This field is highlighted with a blue border.
- Due date:** A date and time input field containing '2022-10-10 14:00:00'.

At the bottom of the form, there are three buttons: 'Add', 'Cancel', and 'Delete'.

iii) Modifying Due date:

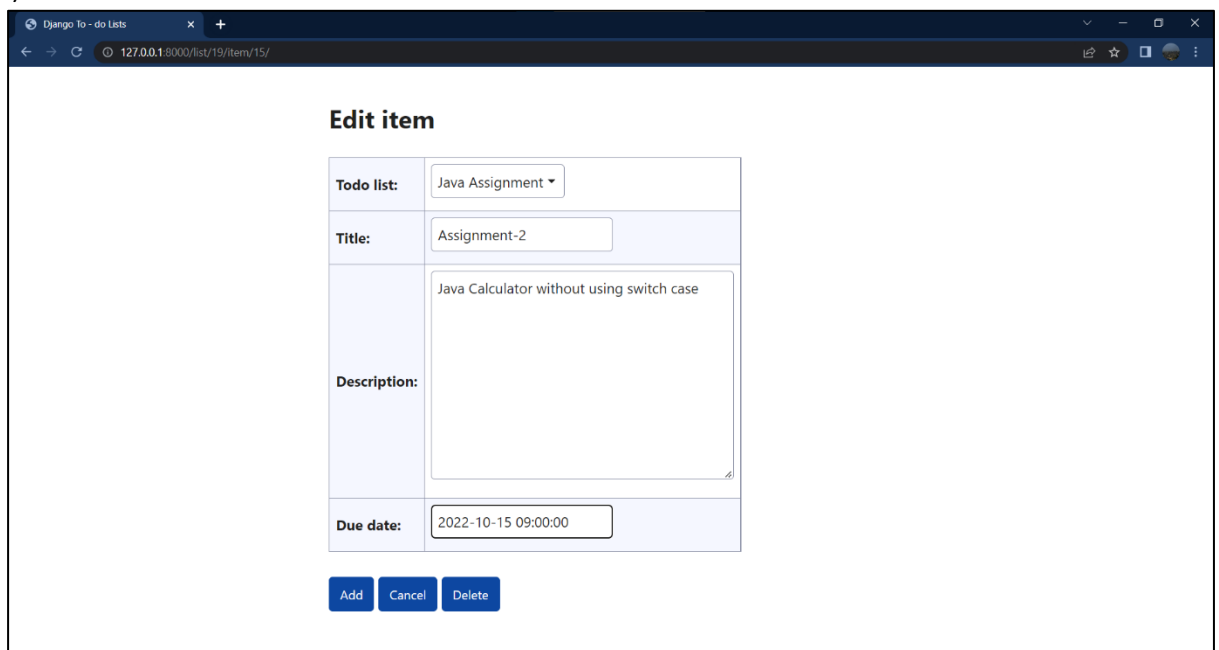
The screenshot shows the same web browser window as the previous one, but the 'Due date' field is now highlighted with a blue border, indicating it is the focus of the modification. The 'Due date' field now contains '2022-10-15 09:00:00'.

12. List view after modifying second item:

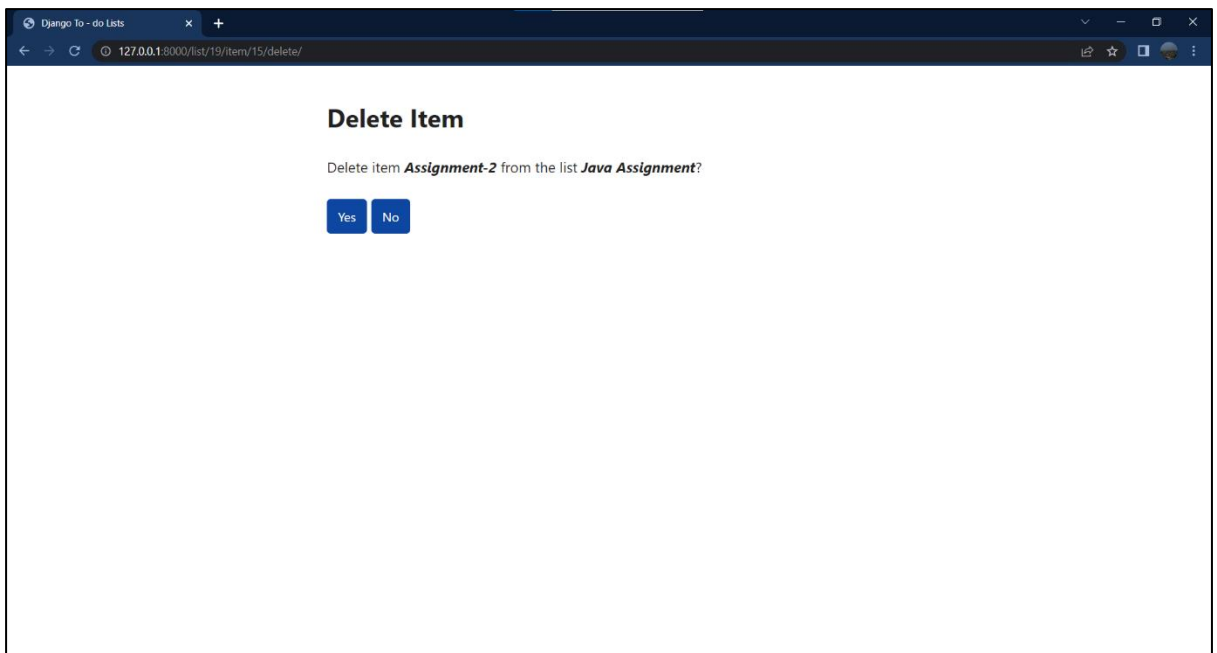


13. Deleting the second item:

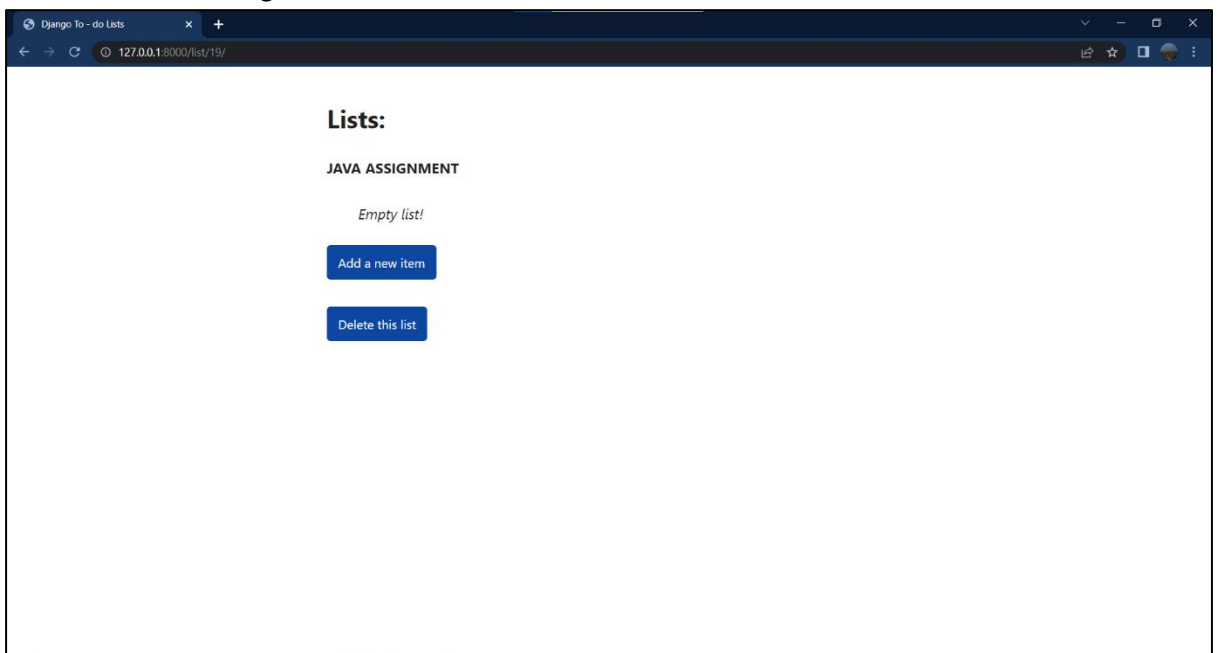
i) Edit view:



ii) Confirmation page:

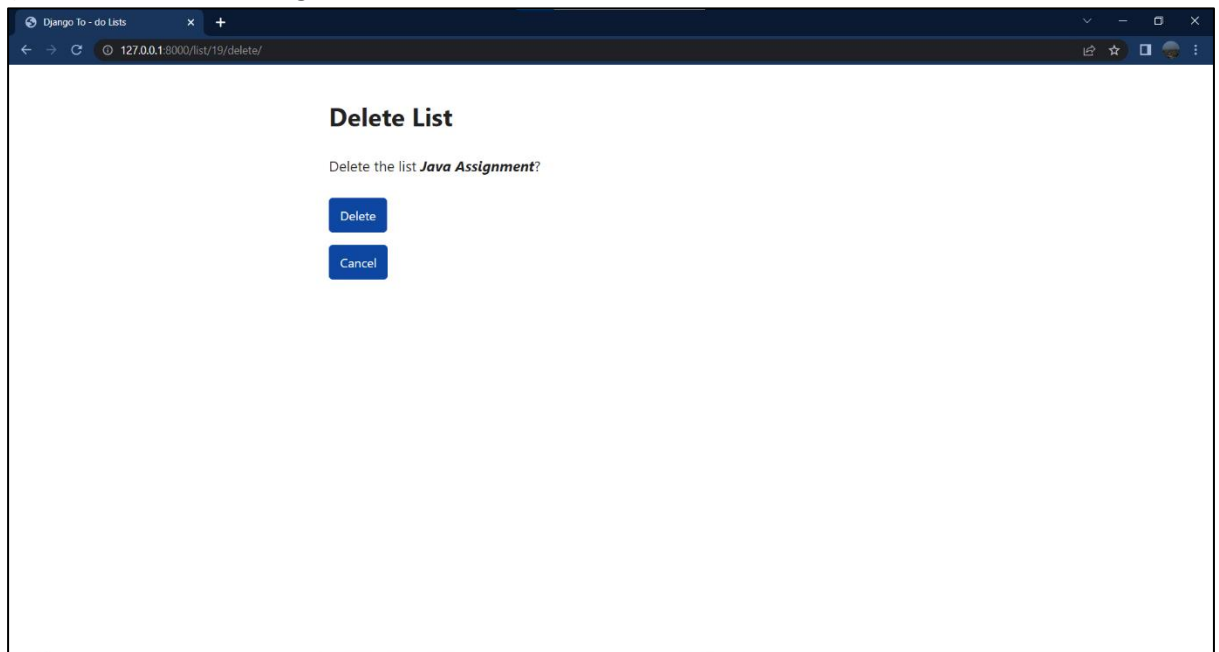


14. List view after deleting the second item:

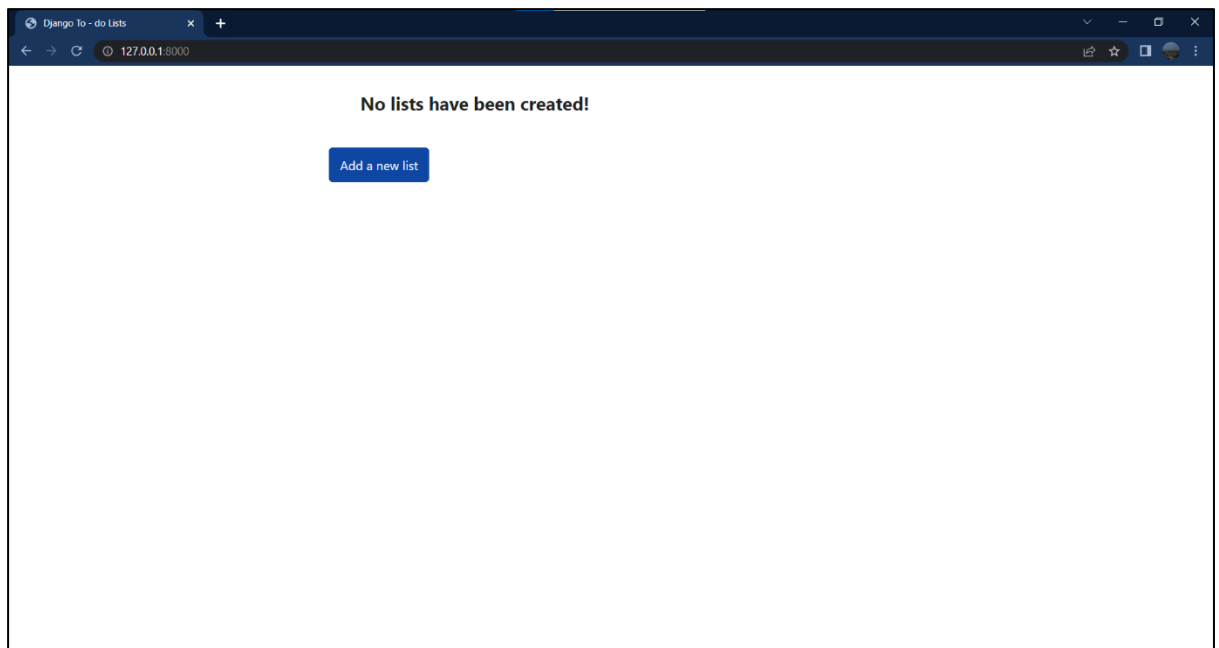


15. Deleting the List:

i) Confirmation Page:



ii) List view after deleting the List:



Conclusion:

Thus, a to-do app that can be used to create, modify and delete tasks, as well as provide a custom time frame has been created. This app can be used both personally (for daily chores, fitness goals, homework, etc.) and commercially (for recording employee timings, assigned tasks and setting target completion dates).