

**Back-end Web Development**

**1st, Semester**

**Jan 2023**

**Continuous Assessment 4**

**Module Title:** Back-end Web Development

**Module Code:** BSC30922

**Assessment Type:** Individual Project **Weighting:** 25%

**Maximal Possible Mark:** 100 marks

**Allocation date:** 28/3/2023

**Submission date:** 18/4/2023



# Assignment Description

Create a new full stack web application using Python, Flask and MongoDB that can perform CRUD operations, Follow the on-line tutorial in the link below to create a template App. Do the following tasks to extend the App:

Extend the App to use a new feature for the ToDo App and/or Use a new Data object for the App

Extend the App by adding a unit test to test the data entry form(s)

Optional for higher grade: Extend the App to add login authentication.

|  |  |
| --- | --- |
| [5%] | Create working development environment with Python, Flask and MongoDB showing hello world app working. |
| [5%] | Follow [How To Use MongoDB in a Flask Application | DigitalOcean](https://www.digitalocean.com/community/tutorials/how-to-use-mongodb-in-a-flask-application) tutorial to create the Flask ToDo App that stores data in MongoDB. |
| [10%] | Document the code with information on what each code item is doing |
| [10%] | Document the testing of the function and the assertions being used. |
| [20%] | Create a readme file with screen shots of App working |
| [40%] | Extend the App to use a new feature for the ToDo App and/or Use a new Data object for the App |
| [10%] | Optional for higher grade: Extend the App to add login authentication. |

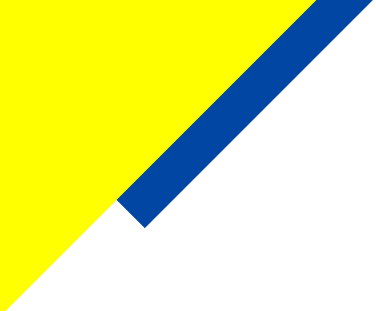
# Additional Information

* This project is an individual project, and you have all the freedom to work on a topic of your interest.
* All the code and the report will be checked for plagiarism, so make sure everything is referenced using comments. The code that is not referenced properly will not be marked and will be reported for plagiarism.
* Outside of special circumstances, any code submitted to GitHub after the deadline – will not be marked.
* Any other technical details that might come up during this project should be clarified directly with your lecturer.



# ASSIGNMENT COVER SHEET 2022/2023

|  |  |  |
| --- | --- | --- |
| **LECTURER NAME:Geoff Wright** | | |
| **STUDENT NAME:Usman Zia** | | **STUDENT ID:24087** |
| **PROGRAMME:Bs Computing** | **STAGE/YEAR:3** | |
| **MODULE NAME:Back-End Web Development** | | |
| **ASSIGNMENT NO. & TITLE:CA4** | | |
| **GROUP (Names & Student IDs):Usman Zia** | | |
| * All assignments must be submitted through Moodle by the agreed submission date. * Only submit a signed hardcopy to your lecturer if requested and you must arrange to hand it directly to the Lecturer and must include this cover sheet. * No submissions will be accepted at Reception. * Late assignments will incur a penalty unless a learner has documented personal mitigating circumstances or if they have been granted an extension. All such documentation must be submitted at least 7 days prior to the submission date. For late submission the assessment grade will be reduced by 10% for each day that the assessment is late. After   day five, the assignment will not be accepted. | Date Received: (Lecturer use only) | |
| **MODE OF SUBMISSION: SOFTCOPY ☐ HARDCOPY ☐** | | |
| **COMPONENTS OF SUBMISSION:**  **(e.g. no and type of pieces submitted, no of pages in a report, disk included?)** | | |
| DECLARATION: I declare that:   * *By uploading this work to Moodle I automatically declare that this work is entirely my own and that I have acknowledged all materials and sources used in its preparation;* * *I have not copied in part or whole or otherwise plagiarised the work of anyone else and have not knowingly allowed others to plagiarise my work in this way.* * *I understand that plagiarism is a serious offence and that I am bound by Dorset College policy on Academic Integrity. I understand that I may be penalised if I have violated the policy in any way.* * *This assignment has not been submitted for any other course or module at Dorset College or any other institution, without authorisation by the relevant lecturer(s).* * *I have read and abided by all of the requirements set down for this assignment.*   SIGNATURE\* …………Usman Zia…………………………… DATE ……10/04/2023…………………………  (\* if this is a group assignment, each member of the group must sign). | | |

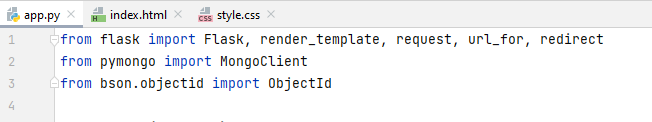
 

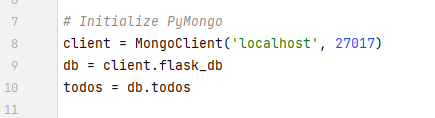


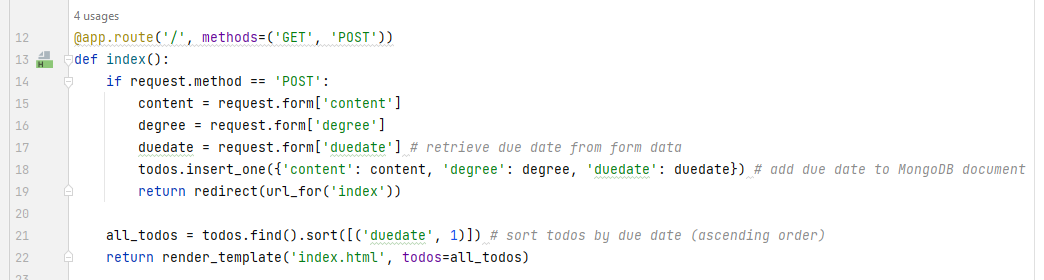
# ASSIGNMENT SOLUTION

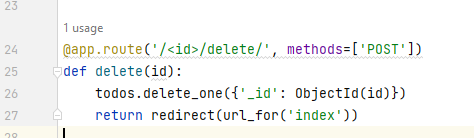
**App.py:**

**I have started this project by creating a new project in PyCharm and firstly I’d like to explain my code and what each line of code is doing**

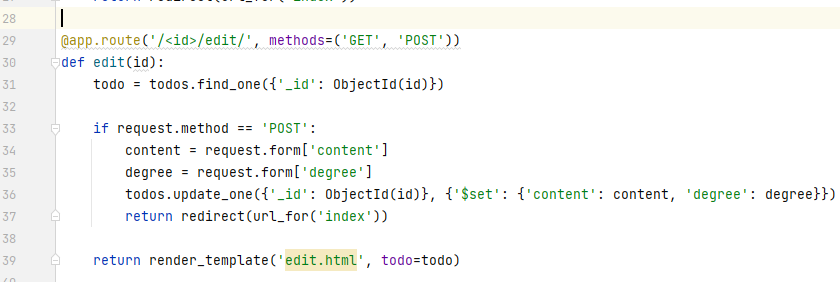
1. **The screenshot attatched below has lines which import the necessary modules and libraries, and initialize the Flask application.**
2. **Then for connecting to the MongoDB database running on localhost:27017, and set up a reference to the todo application collection.**

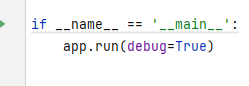


1. **Next is the route for the home page, which displays a list of todo activities. If the request method is POST, the code extracts the todo information from the form data and adds it to the todos collection in the MongoDB database. Then, it redirects back to the home page. If the request method is GET, it retrieves all the todos from the todos collection and sorts them by due date. Finally, it renders the index.html template and passes the sorted todos as context.**
2. **Next I added the route handles which will delete a todo item. It receives an id parameter in the URL and deletes the corresponding document from the todos collection in the MongoDB database. Then, it redirects back to the home page.**



1. **This screenshot below has route handler which will editing a todo item. It receives an id parameter in the URL and retrieves the corresponding document from the todos collection in the MongoDB database. If the request method is POST, it updates the document with the new information from the form data and redirects back to the home page. If the request method is GET, it renders the edit.html template and passes the todo as context.**



1. **Last but not least this runs the Flask application in debug mode if the script is executed directly.**

**Index.html:**

**For the HTML template for this app's main page. There is a head section contains meta tags, page title and then some css for its design screenshot attatched below.**

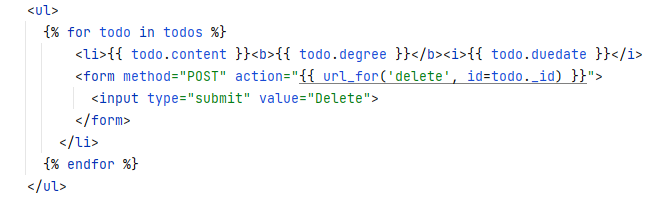


**Then there is main body of html which starts with an h1 tag for the app title. Then, there is a form that allows the user to add a new to-do item to the list. It contains three input fields: a text field for the to-do item content, a select field for the priority level, and a date field for the due date.**



**Below the form, there is a horizontal line with <hr> tag.**

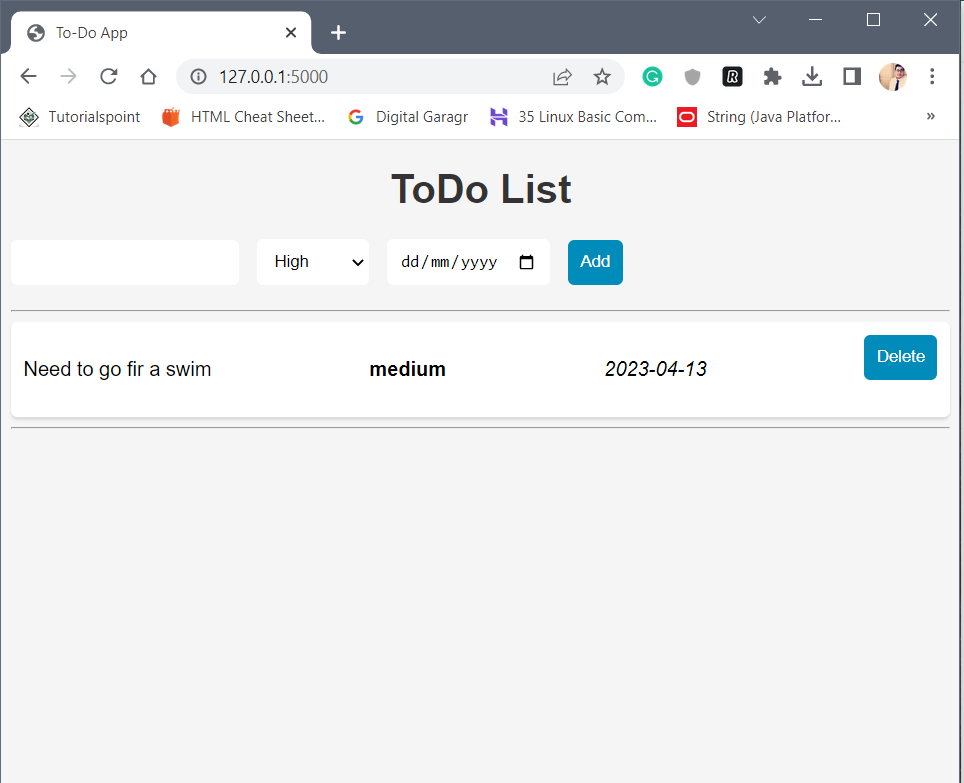
**Then, there is an unordered list that contains all the existing to-do items. For each item, there is an <li> tag that displays the content, priority level, and due date. Each item also includes a form with a delete button that allows the user to remove the item from the list.**



**Finally, there is another horizontal line at the bottom of the list as mentioned above it has a <hr> tag. Just to mention that this template uses Flask's templating engine and Jinja syntax to dynamically display the data from the Flask application. The curly braces with percent signs denote Jinja syntax that tells the template how to render the data from the Flask app.**

**Testing:**

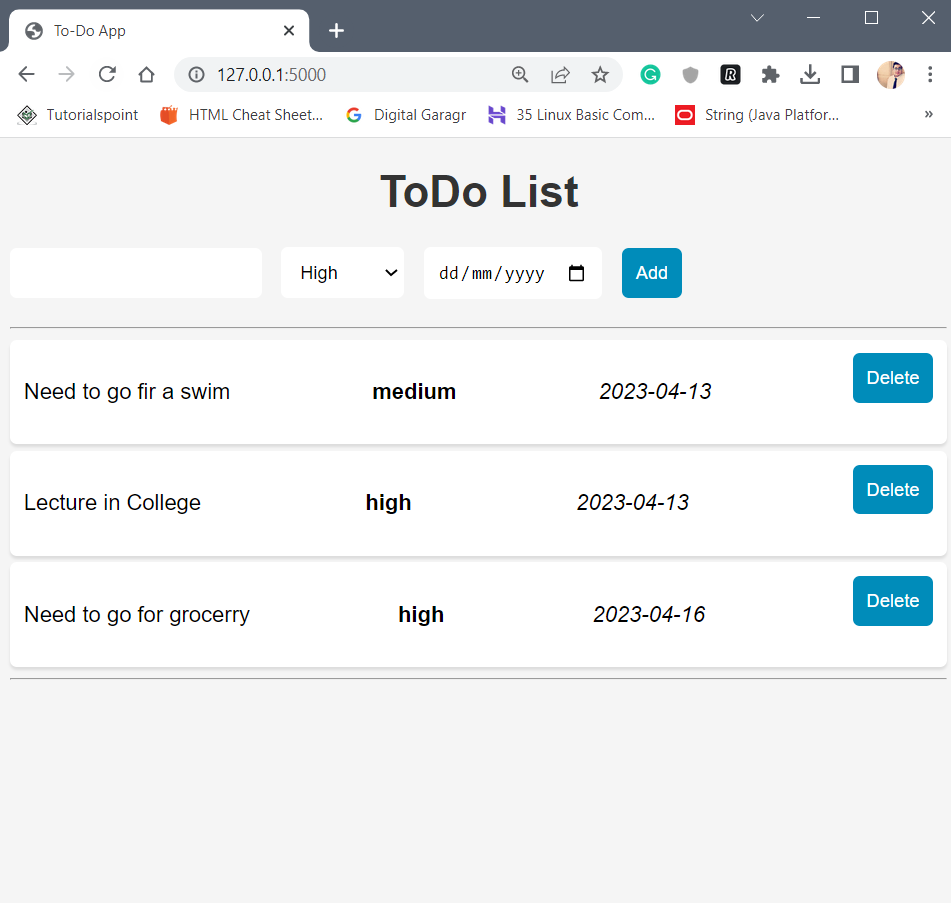
**In order to test it I start the Flask server and browsed** [**http://localhost:5000/**](http://localhost:5000/) **and opened the browser which display a form where you can add a new to-do item by entering a content, selecting a degree, and providing a due date. I Filled in the form and submit it by clicking the "Add" button.**



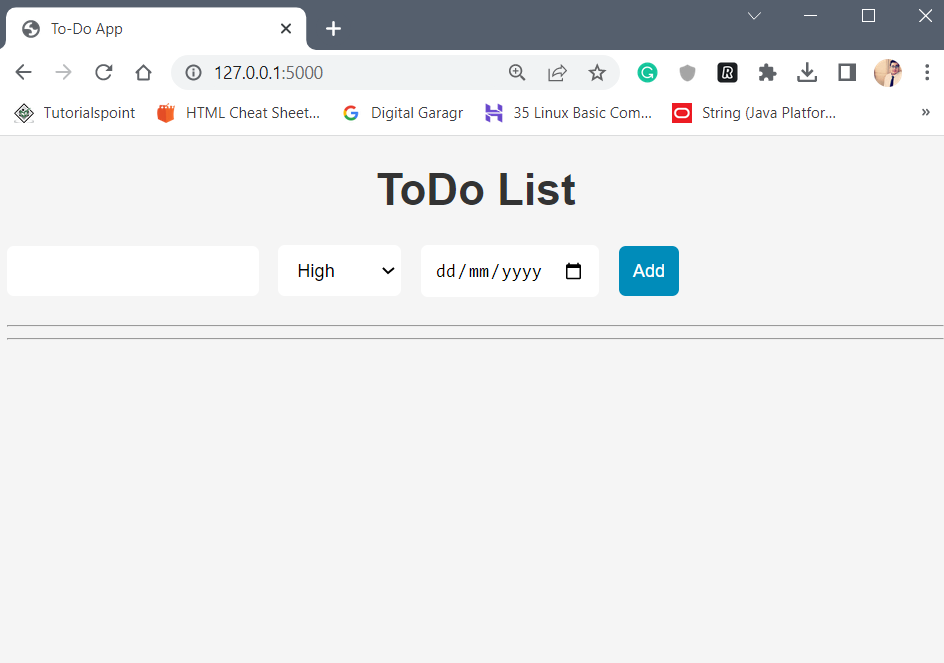
**Looking at above screenshot it an be verified that a new to-do item is added to the list of existing items on the web page with correct content, degree, and due date.**

**Then in order to make sure delete button is functional I created and deleted multiple enteries just to make sure if it is working or not, after testing and all it was functional after all.**

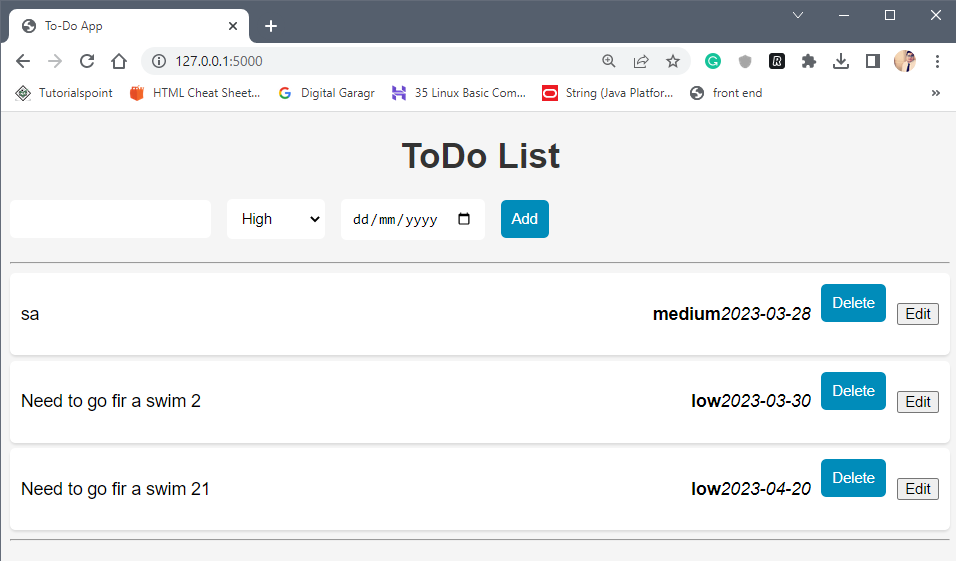
**With data:**



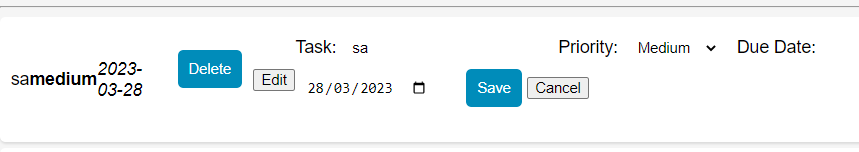
**After Deleting Date:**



**Then I added Edit option as well in this which looks as follows**

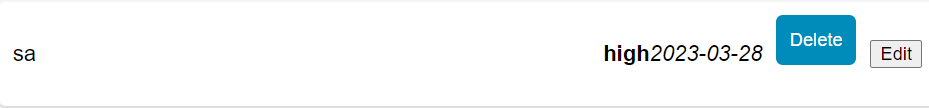


**On clicking Edit button is shows following options**



**Each button is functional and is updating in real time as it can be seen on screenshots**

**Firstly I’d change priority from medium to high,**



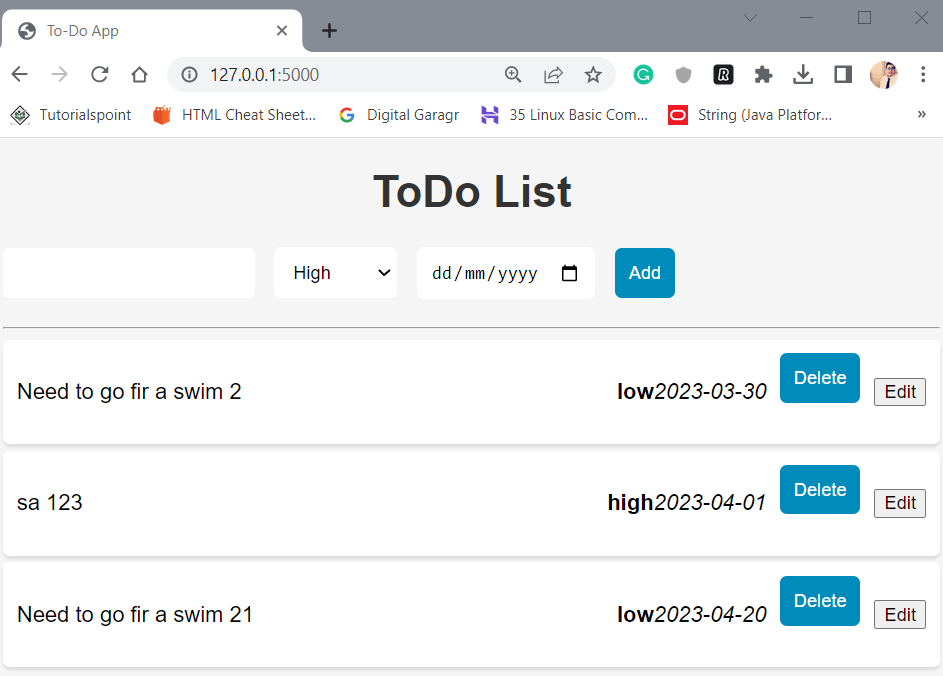
**Secondly I’ve changed date,**



**Thirdly I’ve change its title/task,**



**Now whole page looks like following**



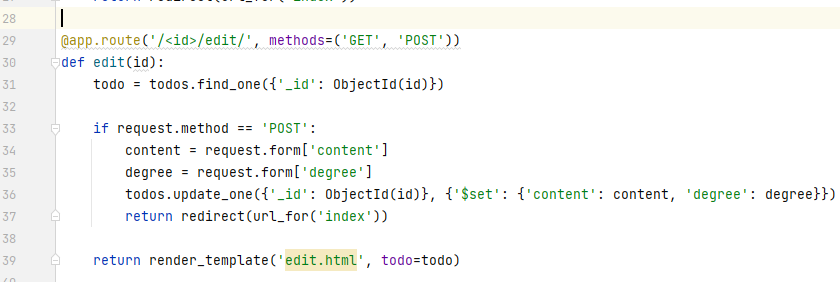
**Hence The application is functional and is in perfect order.**

**Changes After Testing:**

**The only thing I noted at this point was that Edit/Update option was missing so I added that functionality as well.**

**Changes with app.py:**

**From**



**To**



**This code above has a modified Flask route for handling the updating of a to-do item in a MongoDB database event. The route takes the unique identifier of the to-do item as an argument.**

**Final Look of All pages:**

**App.py:**

from flask import Flask, render\_template, request, url\_for, redirect  
from pymongo import MongoClient  
from bson.objectid import ObjectId  
  
app = Flask(\_\_name\_\_)  
  
*# Initialize PyMongo*client = MongoClient('localhost', 27017)  
db = client.flask\_db  
todos = db.todos  
  
  
@app.route('/', methods=('GET', 'POST'))  
def index():  
 if request.method == 'POST':  
 content = request.form['content']  
 degree = request.form['degree']  
 duedate = request.form['duedate'] *# retrieve due date from form data* todos.insert\_one({'content': content, 'degree': degree, 'duedate': duedate}) *# add due date to MongoDB document* return redirect(url\_for('index'))  
  
 all\_todos = todos.find().sort([('duedate', 1)]) *# sort todos by due date (ascending order)* return render\_template('index.html', todos=all\_todos)  
  
  
@app.route('/<id>/delete/', methods=['POST'])  
def delete(id):  
 todos.delete\_one({'\_id': ObjectId(id)})  
 return redirect(url\_for('index'))  
  
  
@app.route('/<id>/edit/', methods=('GET', 'POST'))  
def edit(id):  
 todo = todos.find\_one({'\_id': ObjectId(id)})  
  
 if request.method == 'POST':  
 content = request.form['content']  
 degree = request.form['degree']  
 todos.update\_one({'\_id': ObjectId(id)}, {'$set': {'content': content, 'degree': degree}})  
 return redirect(url\_for('index'))  
  
 return render\_template('edit.html', todo=todo)  
  
  
@app.route('/<id>/update/', methods=('GET', 'POST'))  
def update(id):  
 todo = todos.find\_one({'\_id': ObjectId(id)})  
  
 if request.method == 'POST':  
 content = request.form['content']  
 degree = request.form['degree']  
 duedate = request.form['duedate']  
 todos.update\_one({'\_id': ObjectId(id)}, {'$set': {'content': content, 'degree': degree, 'duedate': duedate}})  
 return redirect(url\_for('index'))  
  
 return render\_template('update.html', todo=todo)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

**index.html:**

<!doctype html>  
<html>  
 <head>  
 <meta charset="UTF-8">  
 <title>To-Do App</title>  
 <style>  
body {  
 font-family: Arial, sans-serif;  
 background-color: #f5f5f5;  
}  
  
h1 {  
 text-align: center;  
 color: #333;  
}  
  
form {  
 margin-bottom: 20px;  
}  
  
ul {  
 list-style-type: none;  
 padding: 0;  
 margin: 0;  
}  
  
li {  
 background-color: #fff;  
 margin-bottom: 5px;  
 padding: 10px;  
 border-radius: 5px;  
 box-shadow: 0px 2px 2px rgba(0,0,0,0.1);  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
}  
li .edit-btn {  
 margin-left: 10px;  
}  
  
li button[type="submit"] {  
 margin-left: 10px;  
}  
  
li span {  
 flex-grow: 1;  
}  
  
input[type="text"],  
select {  
 padding: 10px;  
 border-radius: 5px;  
 border: none;  
 margin-right: 10px;  
}  
  
input[type="submit"],  
li button[type="submit"] {  
 padding: 10px;  
 border-radius: 5px;  
 border: none;  
 background-color: #008CBA;  
 color: #fff;  
 cursor: pointer;  
}  
  
input[type="submit"]:hover,  
li button[type="submit"]:hover {  
 background-color: #006F8F;  
}  
  
input[type="date"] {  
 padding: 10px;  
 border-radius: 5px;  
 border: none;  
 margin-right: 10px;  
}  
  
 </style>  
 </head>  
 <body>  
 <h1>ToDo List</h1>  
  
<form method="POST" action="{{ url\_for('index') }}">  
 <input type="text" name="content">  
 <select name="degree">  
 <option value="high">High</option>  
 <option value="medium">Medium</option>  
 <option value="low">Low</option>  
 </select>  
 <input type="date" name="duedate"> *<!-- new input field for due date -->* <input type="submit" value="Add">  
</form>  
  
  
 <hr>  
 <ul>  
 {% for todo in todos %}  
 <li>  
 <span>{{ todo.content }}</span>  
 <b>{{ todo.degree }}</b>  
 {% if todo.duedate %}  
 <i>{{ todo.duedate }}</i>  
 {% endif %}  
 <form method="POST" action="{{ url\_for('delete', id=todo.\_id) }}">  
 <button type="submit">Delete</button>  
 </form>  
 <button class="edit-btn" onclick="*showEditForm*('{{ todo.\_id }}')">Edit</button>  
  
 <form class="edit-form" id="edit-form-{{ todo.\_id }}" method="POST" action="{{ url\_for('update', id=todo.\_id) }}" style="display: none;">  
 <label for="edit-content-{{ todo.\_id }}">Task:</label>  
 <input type="text" id="edit-content-{{ todo.\_id }}" name="content" value="{{ todo.content }}" required>  
  
 <label for="edit-degree-{{ todo.\_id }}">Priority:</label>  
 <select id="edit-degree-{{ todo.\_id }}" name="degree">  
 <option value="high" {% if todo.degree == 'high' %}selected{% endif %}>High</option>  
 <option value="medium" {% if todo.degree == 'medium' %}selected{% endif %}>Medium</option>  
 <option value="low" {% if todo.degree == 'low' %}selected{% endif %}>Low</option>  
 </select>  
  
 <label for="edit-duedate-{{ todo.\_id }}">Due Date:</label>  
 <input type="date" id="edit-duedate-{{ todo.\_id }}" name="duedate" value="{% if todo.duedate %}{{ todo.duedate }}{% endif %}">  
  
 <button type="submit">Save</button>  
 <button type="button" onclick="*hideEditForm*('{{ todo.\_id }}')">Cancel</button>  
 </form>  
 </li>  
 {% endfor %}  
 </ul>  
 <script>  
 function *showEditForm*(id) {  
 document.getElementById('edit-form-' + id).style.display = 'block';  
 }  
  
 function *hideEditForm*(id) {  
 document.getElementById('edit-form-' + id).style.display = 'none';  
 }  
 </script>  
 <hr>  
 </body>  
</html>

**Conclusion:**

**To conclude a proper working To-Do Web based app is been created using Flask, MongoDb, Html, Css, Python. This app works perfectly and has a nice css and is easy to use.**

**Refrence:**

[**https://moodle.dorset.ie/course/view.php?id=427#section-2**](https://moodle.dorset.ie/course/view.php?id=427#section-2)

[**https://github.com/Geoff-Wright**](https://github.com/Geoff-Wright)

[**https://www.digitalocean.com/community/tutorials/how-to-use-mongodb-in-a-flask-application**](https://www.digitalocean.com/community/tutorials/how-to-use-mongodb-in-a-flask-application)

[**https://www.mongodb.com/docs/atlas/troubleshoot-connection/**](https://www.mongodb.com/docs/atlas/troubleshoot-connection/)

[**https://www.mongodb.com/**](https://www.mongodb.com/)

# End of document

Directors: Hugh Hughes B.Sc. Helen Hughes TDip.lCTS. Hugh T Hughes B.A. H.Dip.Ed. Computer Training Specialists Ltd. Registered in the Republic of Ireland No. 178322

**7/8 Belvedere Place, D1 +353 (1) 8309677** [**m info@dorset.ie**](mailto:minfo@dorset.ie) **e** [**www.dorset.ie**](http://www.dorset.ie/)