The demo we were provided walked you through how to set up a simple to do list web application using Node JS, Angular, MongoDB and HTML/CSS. Following a long was simple enough though it was a little outdated it was simple enough to find the solutions to get the application functional. From there it was up to us how we wanted to enhance the project. I went for a more simple but practical approach implementing categories for each to do list item you added as well as making each category collapsible and providing a simple check box to cross it off your list because everyone enjoys crossing an item off your to do list.

Using Angular 17 I did also have to implement the process using the*, ng new todoapp --no-standalone* command to fully set up the application from the video. I think the main issue I faced was the delete button when I had multiple items and pressed the button on one item it would delete another item in that category due to conflicting ID numbers. So I switched from the ID method implemented in the YouTube tutorial and integrated each item using MongoDB’s ID system that was unique for each item using *const { ObjectId } = require('mongodb');* . This took most of my time to rework the code to where it would work and taught me a lot on how to handle the functions to access the database and integrate it not only into the back end but front-end applications as well. I also reworked the back-end process for connecting to the database using the more contemporary jargon of async and await, to make connecting easier. I also implemented error conditions to further give feedback if something was not working properly. And then played with the CSS a little bit to add color and easier to understand formatting.

I think this has been my first successful application with no problems and I hope to build on it over the next few months to maybe someday launch something like this of my own. I know what I have now is primitive but with the skills I used in the project it was nice to go full stack in developing this application. I look forward to honing my JavaScript skills as well as utilizing databases such as MongoDB and playing around with different ideas this project has sparked.

**Index.JS**

var Express = require("express");

var MongoClient = require("mongodb").MongoClient;

const { ObjectId } = require('mongodb');

var cors = require("cors");

const multer = require("multer");

var app = Express();

app.use(cors());

app.use(Express.urlencoded({ extended: true }));

app.use(Express.json());

var CONNECTION\_STRING = "mongodb+srv://admin:ENI6yfKViJpAiDM4@cluster0.hsox5.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0";

var DATABASENAME = "todoappdb";

var database;

const PORT = 5038;

async function connectToDatabase() {

    try {

        const client = await MongoClient.connect(CONNECTION\_STRING);

        database = client.db(DATABASENAME);

        console.log("MongoDB Connection Successful");

        } catch (error) {

        console.error("MongoDB Connection failed", error);

        }

    }

app.listen(PORT, async () => {

    await connectToDatabase();

});

app.get('/api/todoapp/GetNotes', async (request , response) => {

    try {

        const result = await database.collection("todoappcollection").find({}).toArray()

        response.send(result);

    } catch (error) {

        response.status(500).send(error);

    }

});

app.post('/api/todoapp/AddNotes', multer().none(), async (request,response) => {

    try {

        const count = await database.collection("todoappcollection").countDocuments({});

        await database.collection("todoappcollection").insertOne({

            description: request.body.newNotes, //user input note

            category: request.body.category, //user input category

            completed: request.body.completed === 'true' //checkmark for completion

        });

        response.json("Added Successfully");

    } catch (error) {

        console.error(error);

        response.status(500).send(error);

    }

});

app.delete('/api/todoapp/DeleteNotes', async (request, response) => {

    try {

        const id = request.query.id

        const result = await database.collection("todoappcollection").deleteOne({

        \_id: new ObjectId(id)

    });

    if (result.deletedCount === 0) {

        response.status(404).json( "Note not found" );

    } else {

        response.status(200).json( "Delete Successful" );

    }

} catch (error) {

    console.error(error);

    response.status(500).send(error);

    }

});

app.put('/api/todoapp/UpdateNote', async (request,response) => {

    try {

        const result = await database.collection("todoappcollection").updateOne(

            { \_id: new ObjectId(request.body.id) },

            { $set: { completed: request.body.completed === 'true'} }

        );

        response.json("Update Successful");

    } catch (error) {

        console.error(error);

        response.status(500).send(error)

    }

});

**App.Component.ts**

import { HttpClient, HttpClientModule } from '@angular/common/http';

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  standalone: false,

  styleUrl: './app.component.css'

})

export class AppComponent {

  title = 'ToDoApp1';

  readonly APIUrl='http://localhost:5038/api/ToDoApp'

  constructor(private http:HttpClient){}

  notes: any = [];

  groupedNotesCat: { [key: string]: any [] } = {};

  collapsedCategories: { [key:string]: boolean } = {};

  ngOnInit() {

    this.refreshNotes();

  }

//refresh on any updates to site

  refreshNotes() {

    this.http.get(this.APIUrl+"/GetNotes").subscribe({

      next: data => {

      this.notes=data;

      this.groupedNotesCat = this.groupNotesByCategory(this.notes);

    },

    error: (error) => {

      console.error("Error getting notes:", error);

      this.notes = []; //fallback

    }

  });

}

//add note function to send to Mongo

  addNotes() {

    var newNotes=(<HTMLInputElement>document.getElementById("description")).value;

    var newCategory = (<HTMLInputElement>document.getElementById("category")).value;

    const formData = new FormData();

    formData.append("newNotes", newNotes);

    formData.append("category", newCategory);

    formData.append("completed", "false");

    this.http.post(this.APIUrl+ '/AddNotes', formData).subscribe({

      next: () => {

      alert('Note Added');

      this.refreshNotes();

      (<HTMLInputElement>document.getElementById("description")).value = '';

      (<HTMLInputElement>document.getElementById("category")).value = '';

    },

    error: (error) => {

      console.error("Error Adding Note:", error);

      alert("Failed to add note");

    }

  });

}

//delete notes for Mongo

  deleteNotes(id:string){

    this.http.delete(this.APIUrl+ '/DeleteNotes?id='+ id).subscribe({

      next: () => {

      alert('Note deleted');

      this.refreshNotes();

    },

      error: (error) => {

        console.error("Error deleting", error);

        alert("Failed to delete note");

      }

    });

  }

//group by category

groupNotesByCategory(notes: any[]): { [key: string]: any []} {

  const grouped: { [key: string]: any[]} = {};

  for(let note of notes) {

    if (!note.category) {

      note.category = 'Uncategorized'; //fallback if no category

    }

    if (!grouped[note.category]) {

      grouped[note.category] = [];

    }

    grouped[note.category].push(note);

  }

    return grouped;

}

//collapsible categories

toggleCategory(category: string) {

  this.collapsedCategories[category] = !this.collapsedCategories[category];

  }

//checkmarktoggle

toggleComplete(note: any) {

  note.completed = !note.completed; //boolean

  this.http.put(this.APIUrl + '/UpdateNote', {

    id: note.\_id,

    completed: note.completed }).subscribe(data => {

    console.log('Note updated!');

  });

}

}

**App.component.html**

<div class="card">

<div class="header"> To Do App </div>

<!-- add todo list note -->

<input id="description" placeholder="What's on your agenda? ">

<!-- add category to the note -->

<select id="category">

  <option value="">--Select Category--</option>

  <option value="Personal">Personal</option>

  <option value="Work">Work</option>

  <option value="School">School</option>

  <option value="Home">Home</option>

</select>

<button (click) = "addNotes()"> Add Notes</button>

</div>

<!--category boxes-->

<div class="grid-container">

<div class="category-box" \*ngFor="let category of groupedNotesCat | keyvalue">

  <div class="category-header" (click)="toggleCategory(category.key)" style="cursor:pointer;">

    <span class="arrow" [class.collapsed]= "collapsedCategories[category.key]" >&#9660;</span>

    {{ category.key }}

  </div>

<!--notes only show inside category-->

  <div \*ngIf="!collapsedCategories[category.key]">

    <div class="note-item" \*ngFor="let note of category.value">

      <input type="checkbox" [checked]="note.completed" (change)="toggleComplete(note)">

      <span [ngStyle]="{'text-decoration': note.completed ? 'line-through' : 'none'}">

      \* {{note?.description}}</span>

        <button (click)="deleteNotes(note.\_id)">Delete Note</button>

    </div>

</div>

</div>

</div>

**App.component.css**

/\* Header \*/

.header {

    font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

    background: #ccc;

    color: black;

    padding: 50px;

    font-size: xx-large;

    font-weight: 600;

}

/\* background \*/

body {

    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

    background: #111111;

    margin: 0;

    padding: 20px;

}

/\* card layout \*/

.card {

    background: whitesmoke;

    text-align : center;

    padding: 20px;

    border-radius: 12px;

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

    margin-bottom: 20px;

}

/\* input field \*/

input, select {

    text-align: center;

    padding: 10px;

    margin: 8px 5px;

    border: 1px solid #ccc;

    border-radius: 8px;

    width: 300px;

}

/\* add note button \*/

button {

    padding: 10px 12px;

    background-color: #006847;

    color: white;

    border: none;

    border-radius: 8px;

    cursor: pointer;

    margin-left: 8px;

    transition: background-color 0.3s ease;

    white-space: nowrap;

}

button:hover {

    background-color: red;

}

/\* grid for the boxes \*/

.grid-container {

    display: grid;

    grid-template-columns: repeat(2, 1fr); /\* 2 columns 2 rows \*/

    gap: 10px;

    margin-top: 10px;

}

/\* box \*/

.category-box {

    background:white;

    padding: 4px;

    border-radius: 12px;

    box-shadow: 0 4px 8px rgba(0,0,0,0.1);

}

/\* category header \*/

.category-header {

    background-color: #8F8F8C;

    padding: 10px;

    margin-bottom: 10px;

    font-weight: bold;

    font-size: 18px;

    border-radius: 8px;

    cursor: pointer;

    display: flex;

    align-items: center;

    justify-content: center;

}

/\* arrow rotate \*/

.arrow {

    display: inline-block;

    margin-right: 10px;

    transition: transform 0.3s ease;

}

.arrow.collapsed {

    transform: rotate(-90deg);

}

/\* Note Item \*/

.note-item {

    display: flex;

    justify-content: space-between;

    gap: 0px;

    padding: 0 0 8px 12px;

    background-color: #f9f9f9;

    border-radius: 6px;

    margin-bottom: 6px;

}

.note-item span {

    flex-grow: 1;

    margin-left: 0;

    text-align: left;

    white-space: nowrap;

}

.note-item button {

    flex-shrink: 0;

}

**App.module.ts**

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { HttpClientModule } from '@angular/common/http';

@NgModule({

  declarations: [

    AppComponent

  ],

  imports: [

    BrowserModule,

    HttpClientModule

  ],

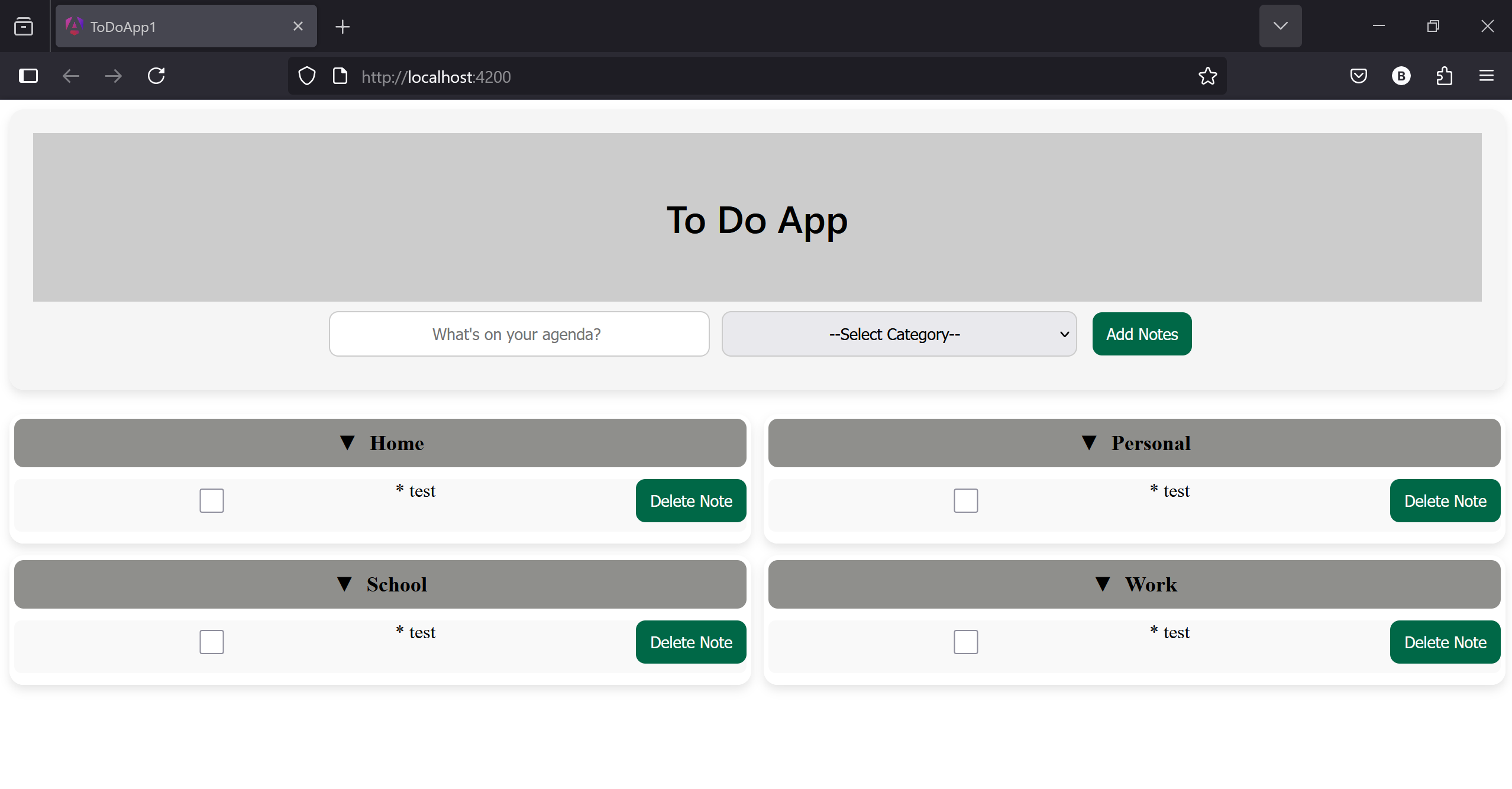
  providers: [],

  bootstrap: [AppComponent]

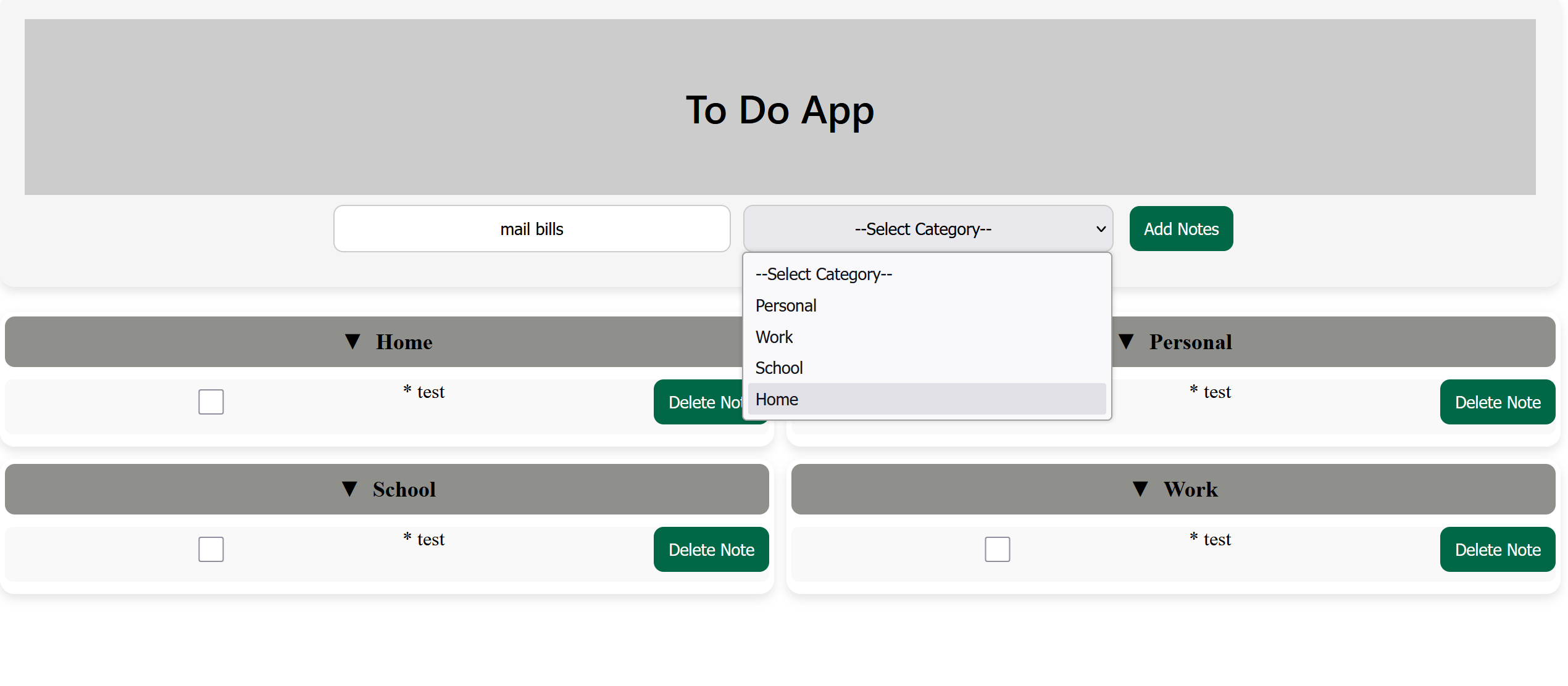
})

export class AppModule { }

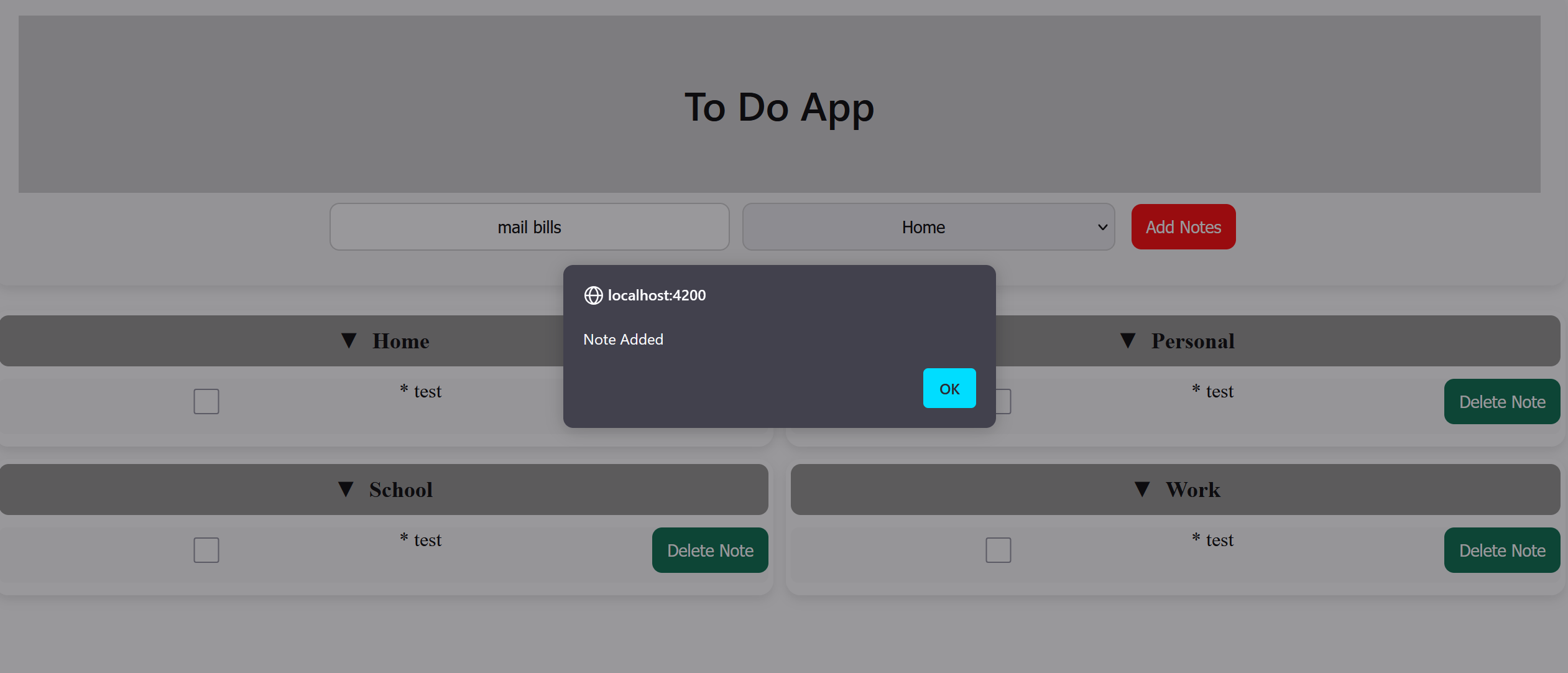
**Code outpu**

****

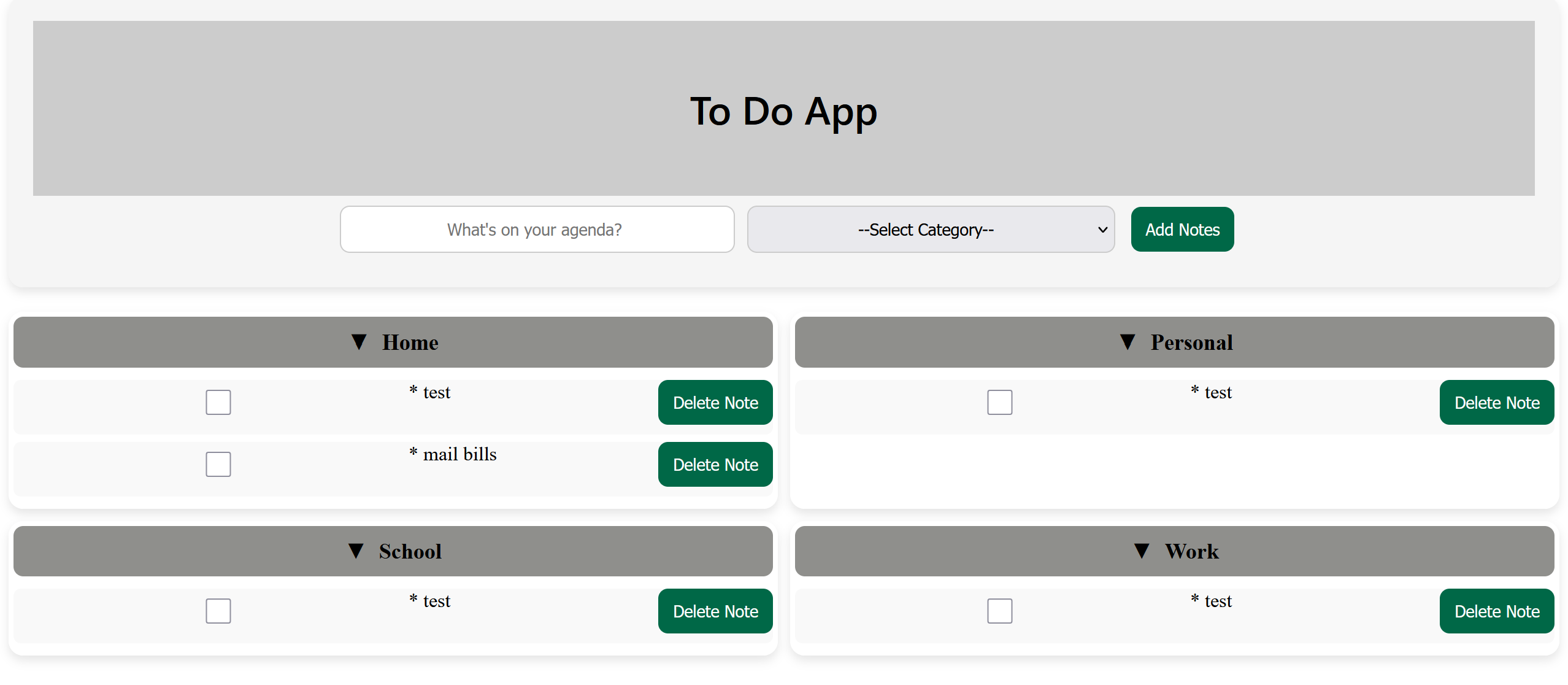
**Shows the main page when loading in**

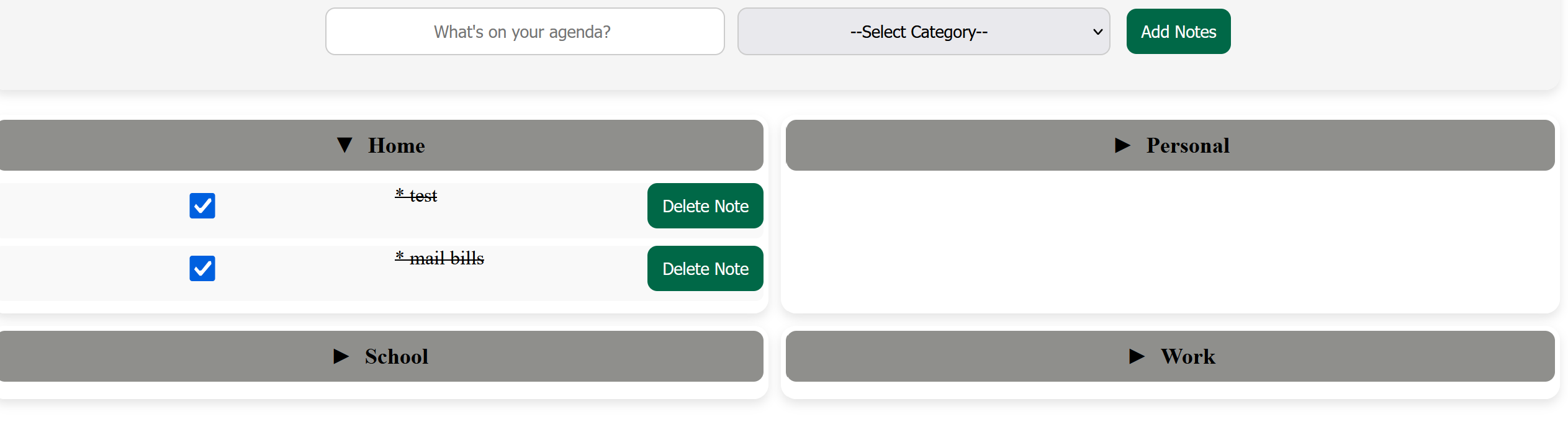
****

**Process of adding item to do list. Enter what needs to be done and then select relevant category.**

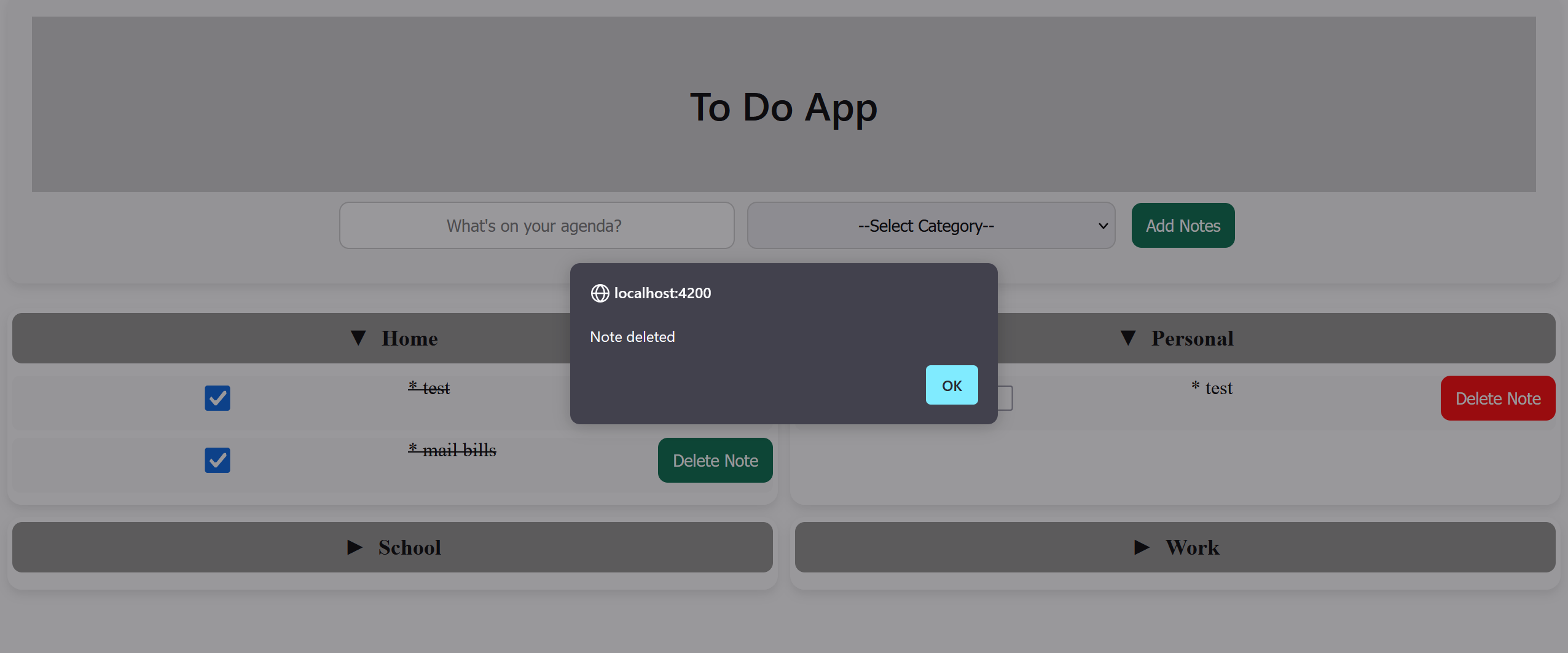


Adding the note to the list

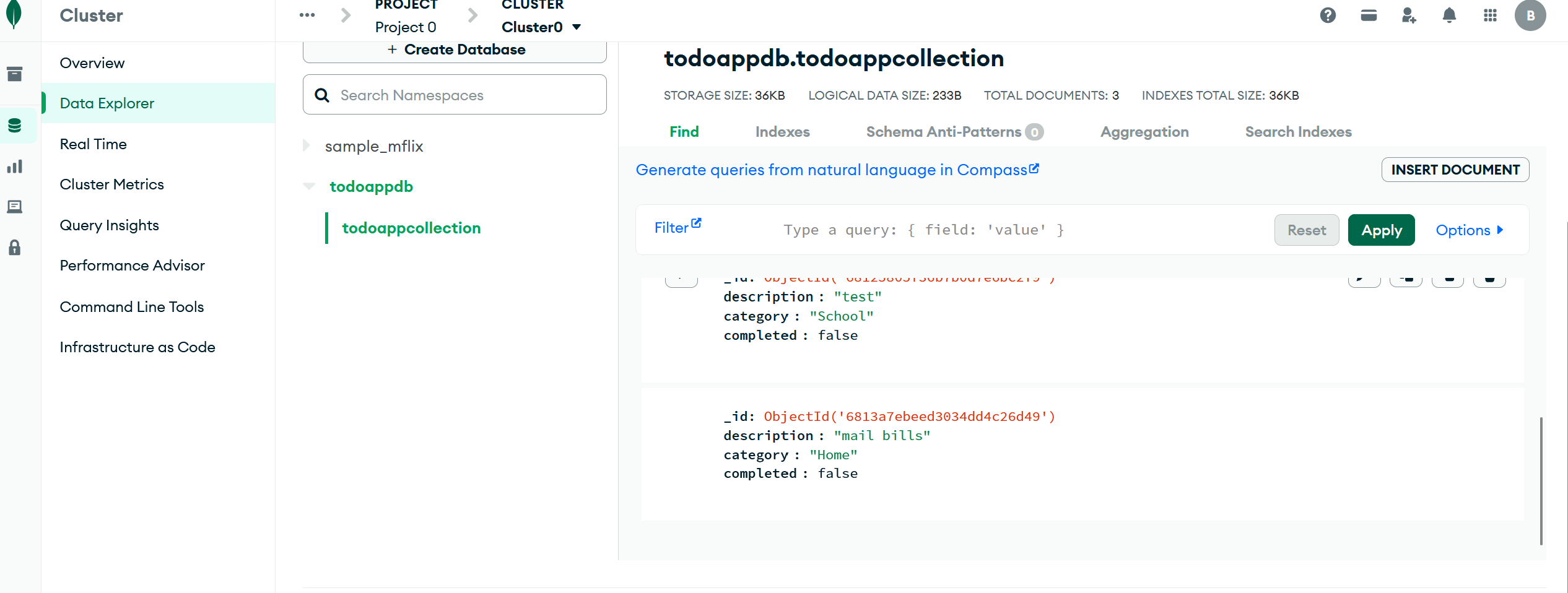




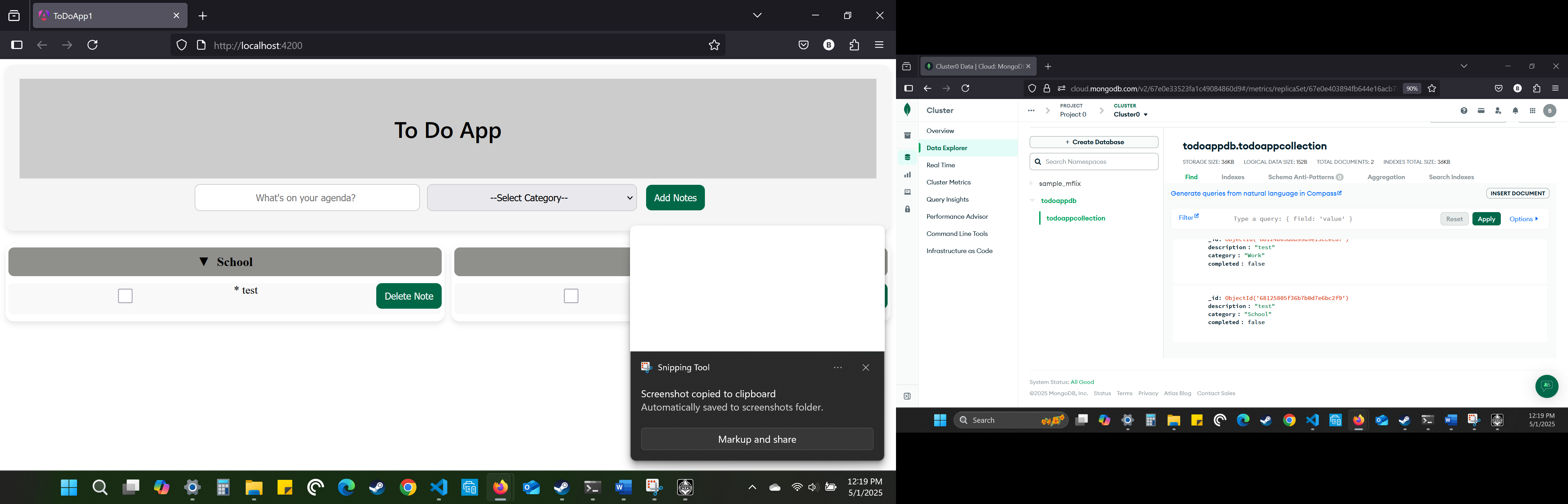
Checking items off and collapsing categories that you are not working on



Showing note be deleted



Showing mail bills item in MongoDB



Showing mail bills deleted in MongoDB