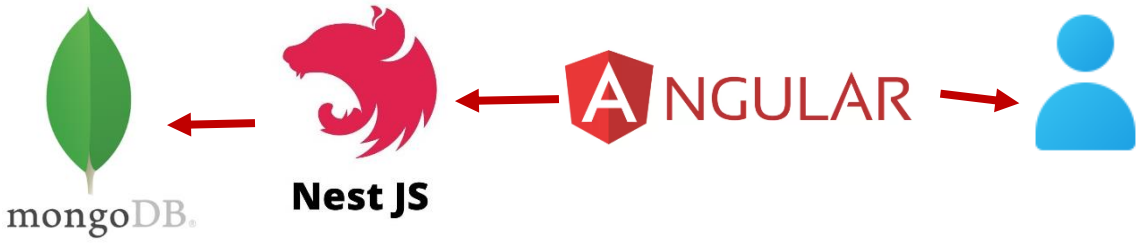
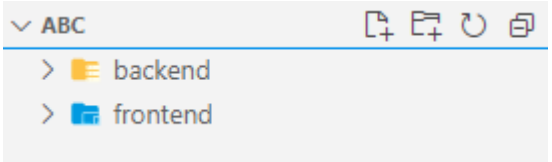


1. Create a NestJS application using the Nest CLI
2. Add MongoDB as the database of choice using the Mongoose library
3. Create an Angular application using Angular Material for the UI
4. Create a model for the data using Mongoose
5. Create controllers and services in NestJS to handle CRUD operations
6. Create front-end components to interact with the API using Angular's HTTP client
7. Test the application to ensure everything is working properly.

Setup NestJS

Install NodeJS at first.

Folder Structure



Open **CMD** in C/wamp/www/**abc/**

Install NestJS CLI in local machine.

```
npm i -g @nestjs/cli
```

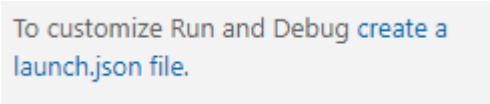
Create a new nest project.

```
nest new backend
```

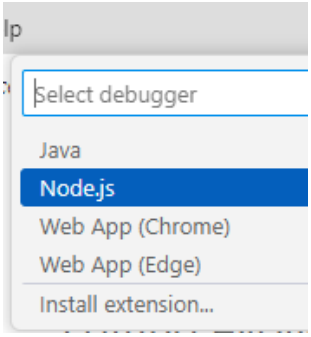
now open the C/wamp/www/abc/backend via VSCode

Create a new debug file (launch.json)

 click 'run and debug' in VSCode side bar menu.



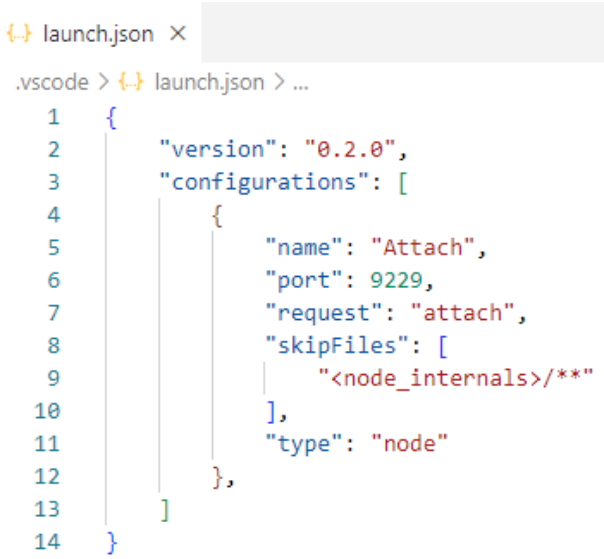
click on create launch.json file



select NodeJs from here

Click on Add Configuration Button ->Select Node.JS: Attach

Remove recent codes and keep only latest codes.



Now run the app on debug mode using cmd

C:\wamp\www\abc\backend> npm run start:debug

Check on server running on browser at localhost:3000

Create a new angular app inside C:\wamp\www\abc as frontend

npm install -g @angular/cli

```
ng new frontend
cd frontend
ng serve
```

check angular app on browser at localhost:4200

install bootstrap

c:\wamp\www\abc\frontend>npm install bootstrap@5.3.0-alpha3

configure bootstrap on angular.json

```
"styles": [
  "src/styles.css",
  "node_modules/bootstrap/dist/css/bootstrap.min.css"
],
"scripts": [
  "node_modules/bootstrap/dist/js/bootstrap.min.js"
]
```

Now replace the content in the frontend>src>app>app.component.html

app.component.html M ×

frontend > src > app > app.component.html > ...

Go to component

```
<nav class="navbar bg-body-tertiary">
    <div class="container-fluid">
      <a class="navbar-brand" href="#">
        <img src="https://getbootstrap.com/docs/5.3/assets/brand/bootstrap-logo.svg"
        Bootstrap
      </a>
    </div>
  </nav>
<router-outlet></router-outlet>
```

Connecting MongoDB with NestJS

Setup MongoDB

Goto mongodb website, and login to atlas

https://www.mongodb.com/cloud/atlas/register

Create a database

Choose your cloud provider, region, and specs.

Build a Database

Name

You cannot change the name once the cluster is created.

abc

Username

chamara

Password

AEzakmi6526560

Create User

My Local Environment

Use this to add network IP addresses to the IP Access List. This can be modified at any time.

Add My Current IP Address

Go to Databases → Browse Collections → Add My Own Data

Create Database

Database name

abc_db

Collection name

students

Additional Preferences

- [ ] Capped Collection
- [ ] Time Series Collection
- [ ] Clustered Index Collection

Cancel Create

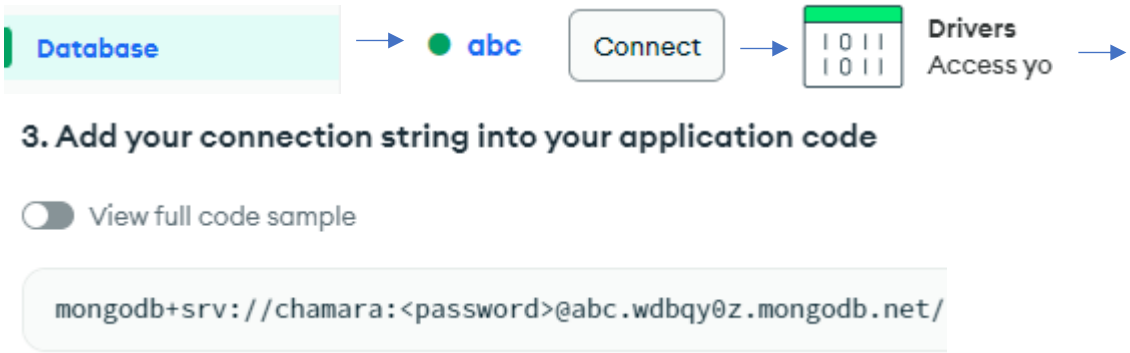
Insert Document

To Collection students

VIEW {} ≡

```
1   _id: 6449e6e01a3e832a10c2ddba          ObjectId
    name: "Chamara"                        String
3   contact: "712345678"                   String
    image: "                               String
```

After Creating database and table, go back to Database using sidebar menu, click connect button



Install mongoose

```
npm i @nestjs/mongoose mongoose
```

Go to backend->src->app.module and set database connection string

```
@Module({
  imports: [
    MongooseModule.forRoot(
      'mongodb+srv://chamara:AEzakmi@6526560@abc.wdbqy0z.mongodb.net/abc_db?retryWrites=true&w=majority',
    ),
  ],
  controllers: [AppController],
```

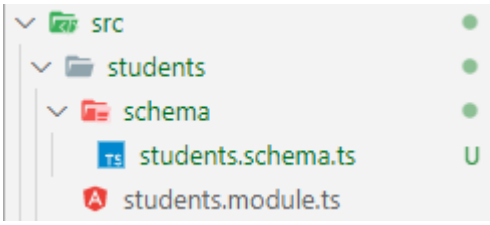
Check the nest app run correctly, npm start

Create new Module for crud operations named students.

backend> nest g mo students

Create an entity which denoted database schema,

backend> nest g cl students/schema/students.schema --flat --no-spec



Create data model class for the collection

backend > src > students > schema > students.schema.ts > ...

```
1  import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';
2  import { Document } from 'mongoose';
3
4  export type StudentsDocument = Students & Document;
5
6  @Schema({ collection: 'students' })
7  export class Students {
8    @Prop()
9    name: string;
10
11   @Prop()
12   contact: string;
13
14   @Prop()
15   image: string;
16 }
17
18 export const StudentsSchema = SchemaFactory.createForClass(Students);
```

Register schema for our future module

backend > src > students > students.module.ts > StudentsModule

```
   You, 35 seconds ago | 1 author (You)
1  import { Module } from '@nestjs/common';
2  import { MongooseModule } from '@nestjs/mongoose';
3  import { StudentsSchema } from './schema/students.schema';
4  import { StudentsService } from './students.service';
5  import { StudentsController } from './students.controller';
6
   You, 35 seconds ago | 1 author (You)
7  @Module({
8    imports: [
9      MongooseModule.forFeature([
10       {
11         name: 'Students', //Mongoose model        You, 35 seco
12         schema: StudentsSchema,
13         collection: 'students',
14       },
15     ]),
16   ],
17   providers: [StudentsService],
18   controllers: [StudentsController],
19 })
20 export class StudentsModule {}
21
```

This is a module class in a NestJS application that imports the MongooseModule and a schema for the "students" collection.

Mongoose is an Object Data Modeling (ODM) library for MongoDB and NestJS provides integration with it through the @nestjs/mongoose module. The purpose of this module is to provide a way to interact with the MongoDB database using Mongoose in a NestJS application.

The MongooseModule.forFeature() method is used to register a Mongoose model in the current module. In this case, the "students" collection schema is being registered under the name "students" using the name property. The schema property defines the schema for the collection and the collection property defines the actual name of the collection in the database.

By importing the StudentsModule in other modules, you can use the registered Mongoose model to perform CRUD (Create, Read, Update, Delete) operations on the "students" collection in the MongoDB database.

Create Service File

Backend>nest g s students --no-spec

This is a TypeScript class definition for a service in a NestJS application that interacts with a MongoDB database through the Mongoose ORM. Here's a breakdown of the code:

The @Injectable() decorator marks the class as a provider that can be injected into other classes or controllers in the NestJS application.

The StudentsService class has a constructor that takes in a Model instance of the Students schema (defined in a separate file) through the @InjectModel decorator. This allows the service to perform CRUD operations on the students collection in the MongoDB database.

The getAll() method is an asynchronous function that returns a Promise that resolves to an array of Students objects. It uses the find() method on the studentsModel instance to retrieve all documents from the students collection.

Overall, this code sets up a service for retrieving all Students documents from a MongoDB database using Mongoose in a NestJS application.

backend > src > students > students.service.ts > ...

```
1  import { Injectable } from '@nestjs/common';
2  import { Students, StudentsDocument } from './schema/students.schema';
3  import { Model } from 'mongoose';
4  import { InjectModel } from '@nestjs/mongoose';
5
6  @Injectable()
7  export class StudentsService {
8    constructor(
9      @InjectModel(Students.name) private studentsModel: Model<StudentsDocument>,
10   ) {}
11
12   async getAll(): Promise<Students[]> {
13     return this.studentsModel.find().exec();
14   }
15 }
```

Create Controller

backend>nest g co students --no-spec

TypeScript class definition for a controller in a NestJS application that handles HTTP requests related to the students resource. Here's a breakdown of the changes:

The method calls this.studentsService.getAll() will ensure that the method calls the getAll() method on the injected StudentsService instance, which will retrieve all Students documents from the database.

With this change, the getAll() method now returns a Promise that resolves to an array of Students objects retrieved from the database.

Overall, this code sets up a controller for handling HTTP GET requests to the /students endpoint and correctly uses the StudentsService to retrieve all Students documents from the database.

backend > src > students > students.controller.ts > ...

```
1  import { Controller, Get } from '@nestjs/common';
2  import { StudentsService } from './students.service';
3
4  @Controller('students')
5  export class StudentsController {
6    constructor(private studentsService: StudentsService) {}
7
8    @Get()
9    async getAll() {
10     return this.studentsService.getAll();
11   }
12 }
```

Now check the browser, localhost:3000 and it should retrieve data from database as a json object

GET http://localhost:3000/students

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

	Key	Value
	Key	Value

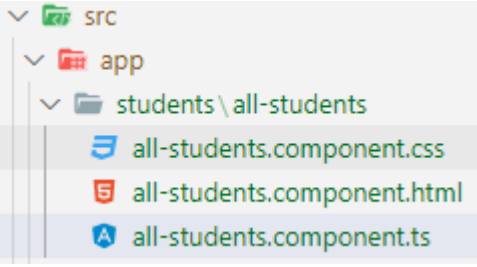
Body Cookies Headers (7) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON


```
1  [
2      {
3          "_id": "6449e6e01a3e832a10c2ddba",
4          "name": "Chamara",
5          "contact": "712345678",
6          "image": "https://img.icons8.com/fluency/240/null/gunter.png"
7      },
8      {
9          "_id": "6449e7a01a3e832a10c2ddbb",
10         "name": "Saroja",
11         "contact": "712345678",
12         "image": "https://img.icons8.com/fluency/240/null/flame-princess.png"
13     }
14 ]
```

Frontend development

frontend>ng generate component students/all-students --skip-tests



app-routing.module.ts × all-students.component.css

frontend > src > app > app-routing.module.ts > ...

You, 4 minutes ago | 1 author (You)

```
import { Component, NgModule } from '@angu
import { RouterModule, Routes } from '@ang
import { AllStudentsComponent } from './st

const routes: Routes = [
  {
    path: '',
    component: AllStudentsComponent
  }
];

```

frontend>ng generate interface students/students

frontend > src > app > students > students.ts > ...

```
export interface Students {
    _id:string;
    name:string;
    contact:string;
    image:string;
}
```

frontend>ng generate service students/students --skip-tests

Add Http Module

students.service.ts U app.module.ts M

frontend > src > app > app.module.ts > AppMod

```
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

frontend > src > app > students > students.service.ts > ...

```
import { Injectable } from '@angular/core';
import {HttpClient} from '@angular/common/http'

@Injectable({
  providedIn: 'root'
})
export class StudentsService {

  constructor(private http:HttpClient) {

  }
}
```

Set URL to get data.

students.service.ts M app.module.ts students.ts all-students.component.c

frontend > src > app > students > students.service.ts > StudentsService

```
})
export class StudentsService {

  constructor(private http: HttpClient) {

  }
  get(){
    return this.http.get<Students[]>("http://localhost:3000/students");
  }
}
```

You, 2 minutes ago • add http module …

Bind received data to the UI

This is an Angular component that retrieves a list of students from a service and assigns them to a class property.

all-students.component.ts (Working Tree) M app.modu

ponent.ts > ...

```
import { Component } from '@angular/core';
import { StudentsService } from '../students.service';
import { Students } from '../students';

@Component({
  selector: 'app-all-students',
  templateUrl: './all-students.component.html',
  styleUrls: ['./all-students.component.css']
})
export class AllStudentsComponent {
  constructor(private studentsService: StudentsService) { }
  students: Students[] = [];
  ngOnInit(): void {
    this.getAll();
  }
  getAll() {
    this.studentsService.get().subscribe((data)=>{
      this.students = data;
    });
  }
```

all-students.component.html M

frontend > src > app > students > all-students > all-students.component.html > ...

You, 26 seconds ago | 1 author (You) | Go to component

```html
<div class="container mt-2">
    <div class="row row-cols-1 row-cols-md-3 g-4">
        <div class="col" *ngFor="let item of students">
          <div class="card">
            <img src="{{item.image}}" class="card-img-top" alt="...">
            <div class="card-body">
              <h5 class="card-title">{{item.name}}</h5>
              <p class="card-text">{{item.contact}}</p>
            </div>
          </div>
        </div>
      </div>
</div>
```

If you got below error when running

```
Access to XMLHttpRequest at 'http://localhost:3000/students' from origin 'http://localhost:4200' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.
Failed to load resource: net::ERR_FAILED
▸ ERROR ▸ HttpErrorResponse
```

Goto backend>main.ts

```typescript
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  app.enableCors();
  await app.listen(3000);
}
bootstrap();
```

Restart both apps.

GET http://localhost:3000/students

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value
Key	Value

Body Cookies Headers (7) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON


```
[
    {
        "_id": "6449e6e01a3e832a10c2ddba",
        "name": "Chamara",
        "contact": "712345678",
        "image": "https://img.icons8.com/fluency/240/null/gunter.png"
    },
    {
        "_id": "6449e7a01a3e832a10c2ddbb",
        "name": "Saroja",
        "contact": "712345678",
        "image": "https://img.icons8.com/fluency/240/null/flame-princess.png"
    }
]
```


Bootstrap

Chamara
712345678

Saroja
712345678

Insert Data

Create new function inside the backend->students.service.ts

```
no usages   new *
16  async create(students: Students) : Promise<Document<?, {}, Studen...  {
17    const newStudents : Document<?, {}, StudentsDocume...  = new this.studentsModel(students);
18    return newStudents.save();
19  }
20 }
```

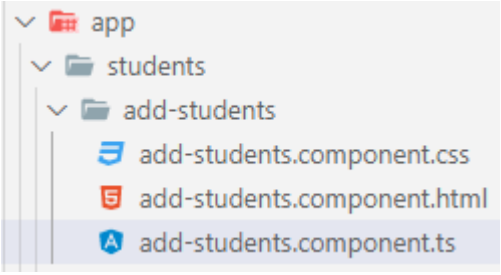
Create new function inside the backend->students.controller.ts

```
no usages   new *
14  @Post()
15  async createStudents(@Body() students: Students) : Promise<Document<?, {}, Studen...  {
16    return this.studentsService.create(students);
17  }
18 }
```

Now try to insert data using postman to the same url.

Create UI form

cmd frontend> ng generate component students/add-students --skip-tests



Open app-routing.module.ts insert new route as add-student

```
const routes: Routes = [
  {
    path: '',
    component: AllStudentsComponent
  },
  {
    path: 'add-student',
    component: AddStudentsComponent
  }
];
```

ng generate interface students/create-or-update-student

create-or-update-student.ts U

frontend > src > app > students > create-or-update-student.ts >

```
export interface CreateOrUpdateStudent {
    name: string;
    contact: string;
    image: string;
}
```

Below code segment creates an HTTP POST request to add a new student to a server using the provided student data.

students.service.ts M

frontend > src > app > students > students.service.ts > ...

```
  get(){
    return this.http.get<Students[]>("http://localhost:3000/students");
  }

  create(student:CreateOrUpdateStudent){
    return this.http.post("http://localhost:3000/students",student);
  }
}
```

add-students.component.ts (Working Tree) M

students > add-students > add-students.component.ts > AddStudentsComponent

```
import { Component, OnInit } from '@angular/core';
import { CreateOrUpdateStudent } from '../create-or-update-student';
import { StudentsService } from '../students.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-add-students',
  templateUrl: './add-students.component.html',
  styleUrls: ['./add-students.component.css']
})
export class AddStudentsComponent implements OnInit {
  constructor(private studentsService: StudentsService, private router: Router) { }

  ngOnInit(): void { }

  students: CreateOrUpdateStudent = {
    name: '',
    contact: '',
    image: ''
  }

  create() {
    this.studentsService.create(this.students).subscribe(() => {
      this.router.navigate(['/']);
    });
  }
}
```

You, 1 minute ago • Uncommitted changes

app.module.ts (Working Tree) M

```
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

add-students.component.html

```
<div class="container mt-2 mb-2">
  <legend></legend>
  <form>
    <div class="mb-3">
      <label for="txtName" class="form-label">Name</label>
      <input type="text" class="form-control" id="txtName" name="name" [(ngModel)]="students.name">
    </div>
    <div class="mb-3">
      <label for="txtContact" class="form-label">Contact</label>
      <input type="text" class="form-control" id="txtContact" aria-describedby="phone" name="contact"
             [(ngModel)]="students.contact">
    </div>
    <div class="mb-3">
      <label for="txtImage" class="form-label">Image</label>
      <textarea class="form-control" id="txtImage" rows="3" name="image" [(ngModel)]="students.image"></textarea>
    </div>
    <button type="button" class="btn btn-primary" (click)="create()">Submit</button>
  </form>

</div>
```

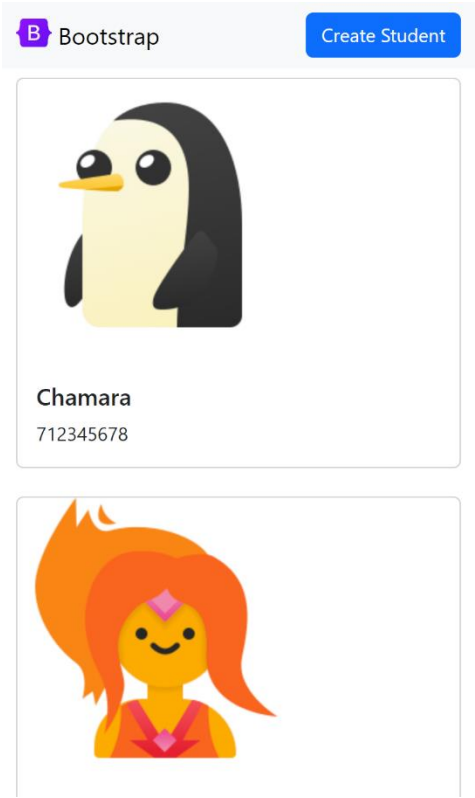
Create a navigation button the page

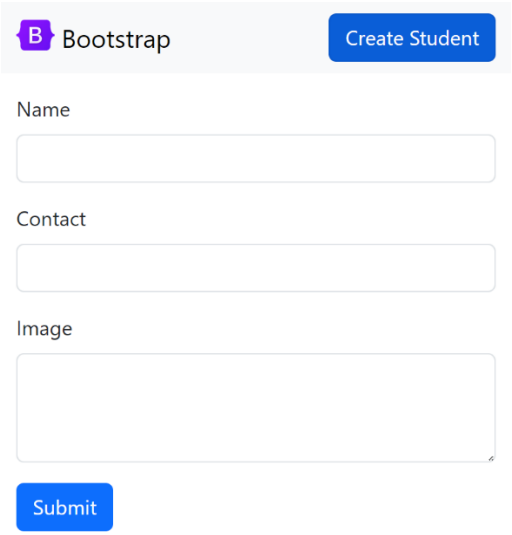
Open app.component.html file, and a link

app.component.html

```
<nav class="navbar bg-body-tertiary">
    <div class="container-fluid">
      <a class="navbar-brand" href="#">
        <img src="https://getbootstrap.com/docs/5.3/assets/brand/bootstrap-logo
        Bootstrap
      </a>
      <a class="btn btn-primary" routerLink="/add-student">Create Student</a>
    </div>
  </nav>
<router-outlet></router-outlet>
```

Output


Bootstrap
Create Student
Chamara
712345678


Bootstrap
Create Student
Name
Contact
Image
Submit