

A REPORT
ON
UNIVERSITY MANAGEMENT APP USING ANGULAR

By

Name of the student

Enrolment/Registration No.

Pothala Pavan Kumar

190141

Prepared in the partial fulfillment of the
Practice School II Course

AT

EAIESB SOFTWARE SOLUTIONS PVT LTD, HYDERABAD

A Practice School II Station of



BML MUNJAL UNIVERSITY

(June - July 2021)

BML MUNJAL UNIVERSITY

PRACTICE SCHOOL – III

JOINING REPORT

Date: 03-08-2021

Name of the Student	Pothala Pavan Kumar
Name and Address of the Practice School – II Station	EAIESB SOFTWARE SOLUTIONS PVT LTD, HYDERABAD
Date of Joining PS-II station as per offer letter	26 st May 2021
Actual date of reporting to PS-II station	26 st May 2021
Department Allocated	Angular and Python
Name and Designation of the Industry Guide/ Industry Mentor for the Project	Name: MR.Vamsi Arkadu, MR.Akhil Kumar Designation: Programme Analyst At EAIESB SOFTWARE SOLUTIONS PVT LTD, HYD
Industry Mentor Contact No. (If available)	9866847636
Industry Mentor E-mail Address (Compulsory)	akhil.kumar@eaiesb.com , vamsi.arkadu@eaiesb.com

ACKNOWLEDGMENT

It is indeed a great pleasure for me to present this PS-2 training report on EAIESB solutions. As a part of the curriculum of the Bachelor of Technology (Computer Science and Engineering) degree.

I would like to express my honest gratitude to my Mentors and Instructors at EAIESB Solutions: Mr. Vijay , Mr. Vamsi, Mr. Akhil Kumar for helping me and uplifting us to complete the Internship program. I would also like to express my special thanks of gratitude to Vice Chancellor and Professor Maneek Kumar, Dean, SOET and Dr.Pradeep Arya, coordinator of PS-II program who gave us this wonderful opportunity to do this .

Signature of Student
Pothala Pavan Kumar

ABSTRACT

Objective:

Building an student management system using angular for UI , FastAPI for writing the services and Neo4j as Database. In this project we were asked to learn how to create CRUD Operations using Angular and Cypher Query Language, Neo4j graph DB database. We need to use Angular's Reactive Forms service for managing the user submitted data in our web application.

Results:

Successfully Created an University Management Website with an interactable and userFriendly user interface .

Here is the Demo Video link of the website:

<https://drive.google.com/file/d/11AC5XcdDQlnYY9qnzxxVbGPZbaITVkqL/view?usp=sharing>

<https://drive.google.com/file/d/1tOEp3jn4ArKubLVfzrOYmZE7ByoLvffj/view?usp=sharing>

Conclusion:

Now website is completely live. It Can be used by any university to manage all the things where they can simply do the CRUD operations and manage all type of details in a simpler and graphical way.

TABLE OF CONTENTS

CONTENTS	Pg.no
1. Introduction of Organization	07
2. Overview of the organization	07
3. Plan of your internship	08
4. Background of the problem	09
5. Methodology	10
6. Outcomes	26
7. References	27
8. Installation's	27
9. Figure's Table	29

A BRIEF INTRODUCTION OF THE ORGANIZATION'S BUSINESS SECTOR

EAIESB is a Middleare solutions and develment company based out of India. EAIESB deigns, develops and deploys custom solutions that automate complex business proceses. Oracle Recognizd EAIESB with Oracle BPM Excellence Award winnr for the Year 2014. EAIESB is focud on the execution, delivery and support of enterprise software for Insurance, technology, communications, Bio-Technology, Retail, Banking and industrial anufacturing companies. With solution centers in the Unites States, UK, Romnia and India, EAIESB has a tremendous base of eperience leveraging Mule Soft, Oracle and Tibco technology and applications on behalf of its customers.

WEBSITE: <http://www.eaiesb.com>

OVERVIEW OF THE ORGANIZATION

INNOVATION

As trusted sources who move and inspire people with novative ideas, EAIESB turns these ideA into reality. Our innovatiVe implementation Won the award from OracLe.

TRUSTED ADVISORY

With YeaRs of experience ,and deep subject matter expertise, EAIESB team takes pride in serving as trusted advisOrs for Customers.

AWARD WINNING SOLUTIONS

EAIESB solutions-focused approach has resulted in winning awards for EAIESB And its Customers.

MIDDLEWARE EXPERTS

EAIESB has proven expertise and project experience in middleware tools like MuleSoft, Oracle FusioN & TIBCO stacks. Migration Experts from SeeBeyond, Sun products, Oracle Fusion, TIBCO, and MuleSoft vice versa.

PLAN OF YOUR INTERNSHIP

To build an Dynamic University Management Website:

- Install Neo4j and start writing queries for sample static data.
- Building the UI of the app:
 - Making the HomePage Components in Angular.
 - Making the sidenav Bar
 - Making the components for each of the component display for schools,Departments,Programmes etc
- Building the web framework to create Api's with Python using FastApi for connecting the database and fetching the data from the database
- Create the Service files in the angular for fetching to the webpage and FastApi.
- Implementing the routing part
- Displaying the data that is retrieved from the database in the components as required.
- Making a Graphical Representation

BACKGROUND AND DESCRIPTION OF THE PROBLEM

Problem Statement:

You must create an angular website with dynamic data that should be using the neo4j database with using the fastapi web framework for building the API.

The website should include the following features.

- All the data should be dynamically fetched from the database.
- It should have an container where it can display all the nodes present in Neo4j.
- It should have an container where it should display all the schools present in the university and should also have an button to add any new node by filling the post form.
- All the data should be fetched from the forms reactively and when the user clicks on submit form the data should be posted in the Neo4j database and user must be able to see the nodes in database.
- There should also be an form for posting the relation between two nodes and the relation between two nodes should be reflected in the database.
- The UI should have an container where it should display all the programmes present in a particular school. All the crud operation for posting were same as like for Posting schools and connecting the relationships.
- The same should follow for the department section whenever the user clicks on any programme.
- The main part of the Project is displaying the students and faculty members for each division.
- At the start of the Page the container should display all the present.
- Whenever the user clicks on school it should display the students related to a particular school
- The same should continue to the display the students and faculty whenever the user click on any programme and department.

- At the end , when the student displays a particular student or faculty it should display the complete details of the student.
- Lastly We need to make an new component where all the above data should be in a graphical representation way.
-

METHODOLOGY

Task 1:

Prototyping the Application and environment setup:

First, validating the idea by doing a low-level implementation (Proof of Concept) of the components and architecture involved in the project.

- Research and collecting the information required for the UI Implementation and concepts that include the angular components and lifecycle hooks.
- Getting a better understanding of the stages involved in the project. For example: setting up the server and database, create-edit-delete the user posts.
- Environment setup - Installing node.js and necessary dependencies through npm, setting up server.
- Setting up Database through NEO4J - to manage the data in the cloud and connecting it to my http server .
- Installing all the required modules and libraries for FastApi which is a web framework for building API's.
- All the dependenics and nodeModules required for the angular should be installed.
- Installing the **bootstrap** for Angular is also necessary.

```
<!-- JavaScript Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-gtEjrD/SeCtm"

<link rel="stylesheet" href="/assets/css/bootstrap.min.css">
<link rel="stylesheet" href="/assets/css/quicksand.css">
<link rel="stylesheet" href="/assets/css/style.css">
<link rel="stylesheet" href="/assets/css/fontawesome-all.min.css">
<link rel="stylesheet" href="/assets/css/fontawesome.css">
<link rel="stylesheet" href="/assets/css/animate.min.css">
<link rel="stylesheet" href="/assets/css/chartist.min.css">
<link rel="stylesheet" href="/assets/css/jquery-jvectormap-2.0.2.css">
<link rel="stylesheet" href="/assets/js/calendar/bootstrap_calendar.css">
<link rel="stylesheet" href="/assets/css/nice-select.css">
```

Fig:1 BootStrap Links

Dependencies:

```
@ package.json > {} dependencies
1 {
2   "name": "bmu3",
3   "version": "0.0.0",
4   "scripts": {
5     "ng": "ng",
6     "start": "ng serve",
7     "build": "ng build",
8     "watch": "ng build --watch --configuration development",
9     "test": "ng test"
10  },
11  "private": true,
12  "dependencies": {
13    "@angular/animations": "~12.1.1",
14    "@angular/common": "~12.1.1",
15    "@angular/compiler": "~12.1.1",
16    "@angular/core": "~12.1.1",
17    "@angular/forms": "~12.1.1",
18    "@angular/platform-browser": "~12.1.1",
19    "@angular/platform-browser-dynamic": "~12.1.1",
20    "@angular/router": "~12.1.1",
21    "leader-line-new": "^1.1.9",
22    "ngx-leader-line": "0.0.3",
23    "rxjs": "~6.6.0",
24    "tslib": "^2.2.0",
25    "zone.js": "~0.11.4"
26  },
27  "devDependencies": {
```

Fig:2 Dependences

BOOTSTRAP DEPEDENCY:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\Users\PAVAN KUMAR\Desktop\bmu3> ng add @ng-bootstrap/ng-bootstrap
```

Fig:3 Bootstrap Dependency

ANGULAR APP CREATION:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\Users\PAVAN KUMAR\Desktop\bmu3> ng new BMU-app
```

Fig:4 Angular App Creation

Task 2:

Building the pages layout using HTML, CSS, and Bootstrap

HomePageView:

1. SideBar (Nodes Display)

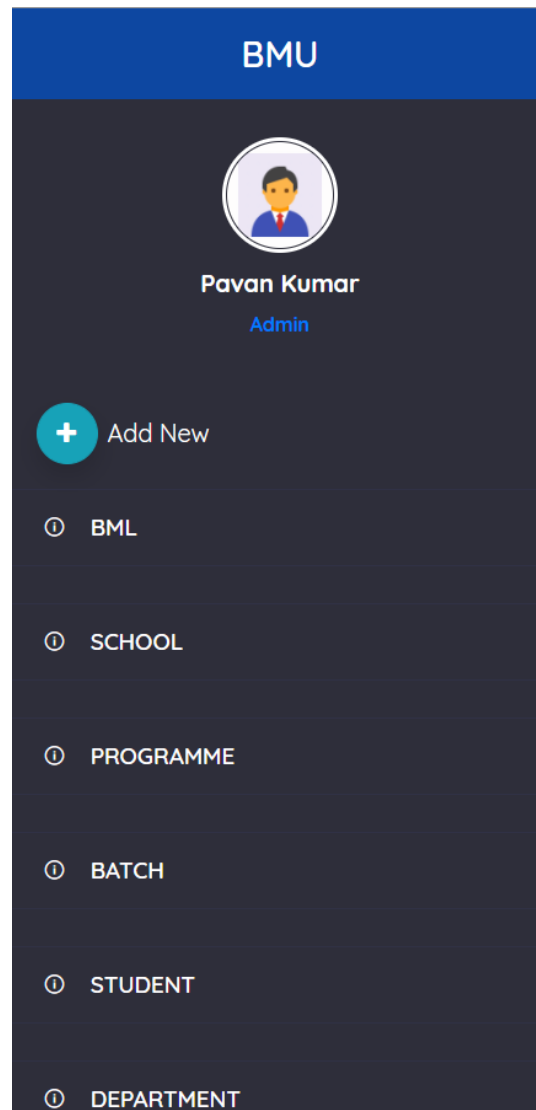


Fig:5 SideBar UI

2. Relationships View

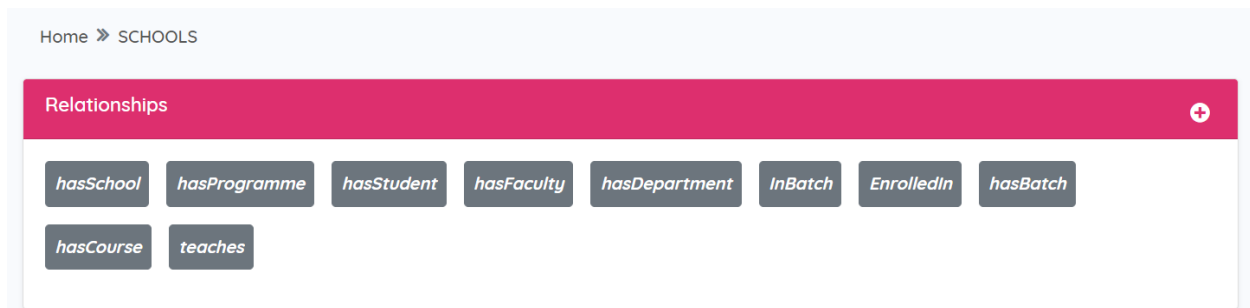


Fig:6 Relationship View

3. SCHOOLS View

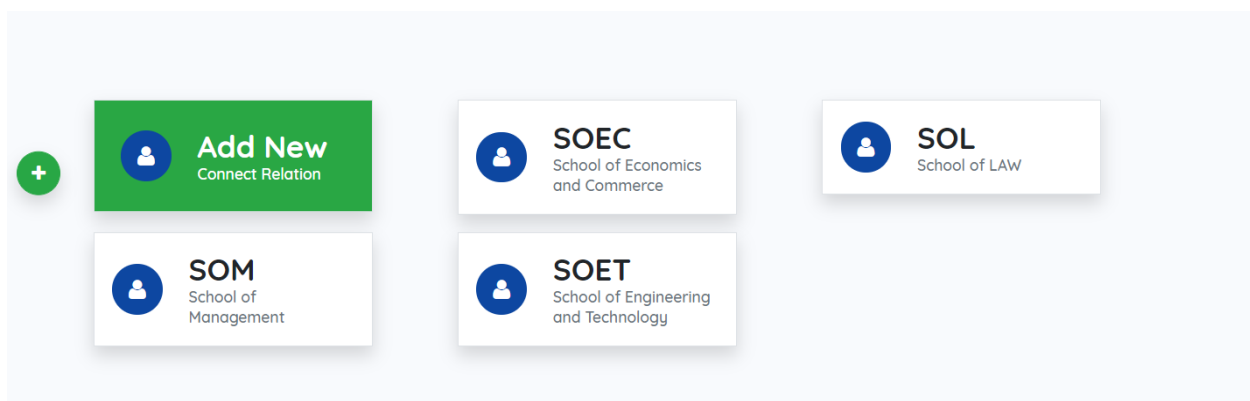


Fig:7 School's View

4. Programmes Display (It appears only when any user selects any school)

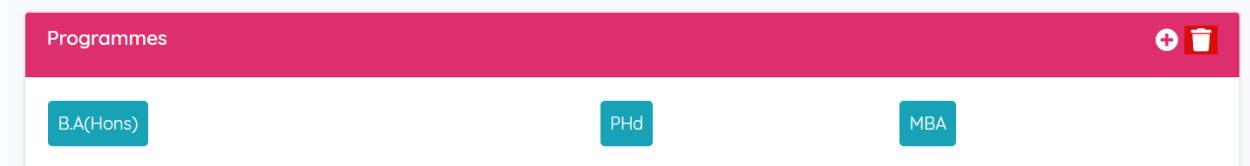


Fig:8 Programme's View

5. Departments Display(Appears when any user selects any Programme)

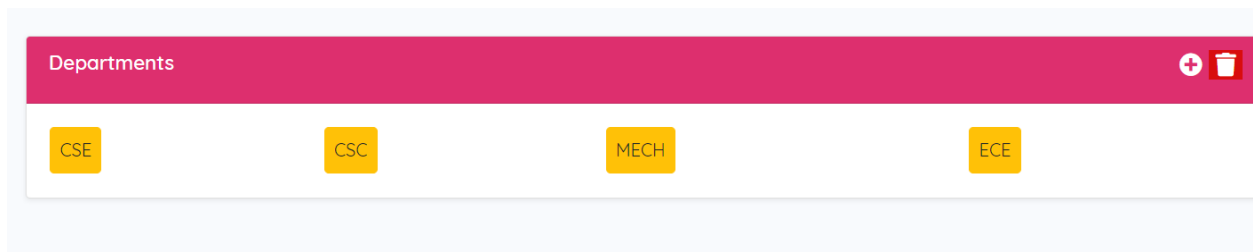


Fig:9 Departments View

6. Students View

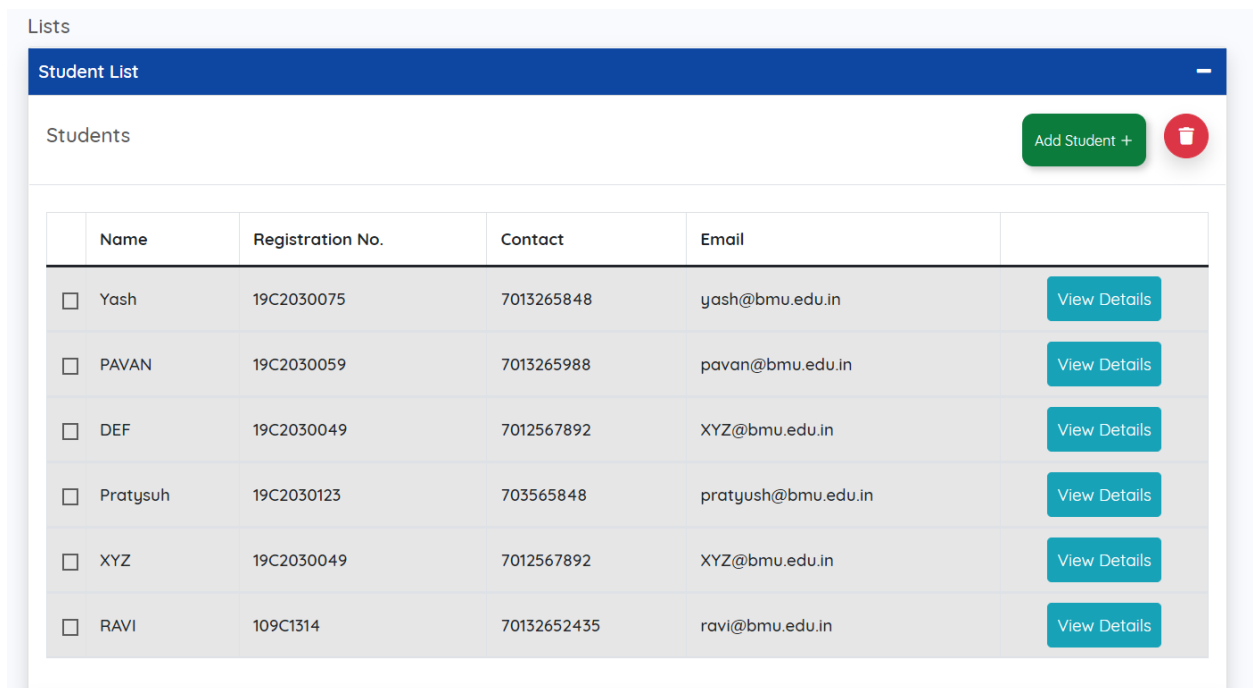


Fig:10 Students

7. Faculty View


Faculty List				
Faculty			Add Faculty +	
	Name	Contact	Email	Course
<input type="checkbox"/>	VIJAY	970417993	vijay@eaiesb.com	NEO4J
<input type="checkbox"/>	AKHIL	8247896715	akhil@eaiesb.com	PYTHON3.0
<input type="checkbox"/>	AKHIL	8247896715	akhil@eaiesb.com	JAVA
<input type="checkbox"/>	VAMSI	9866847636	vamsi@eaiesb.com	JAVA
<input type="checkbox"/>	Rajesh	7012354789	rajesh@bmu.edu.in	NEO4J

Fig:11 Faculty

8. Course view



Courses					
Course Title	Credits	Instructor	Duration		
PYTHON3.0	3	AKHIL	Half-Sem		
JAVA	3	AKHIL	Half-Sem		
JAVA	3	VAMSI	Half-Sem		
NEO4J	3	VIJAY	Half-Sem		
NEO4J	3	Rajesh	Half-Sem		

Fig:12 Courses

9.Student Details

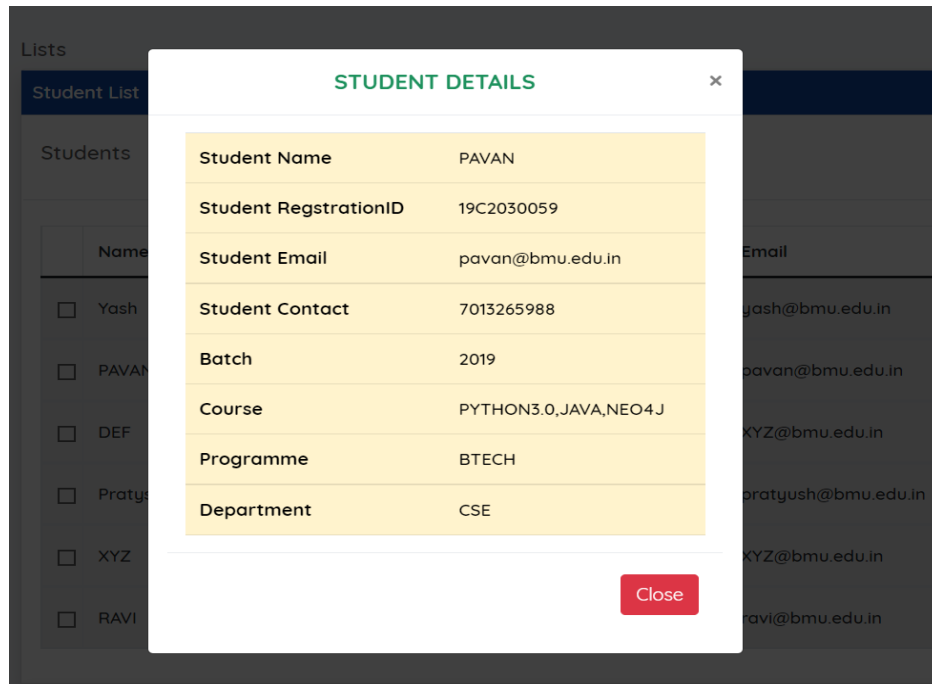


Fig:13 Student Details

Task 3:

DATABASE CREATION

- Generating static data in neo4j for testing out .
- Making all the nodes and relations between nodes.
- The overall database would look like this.

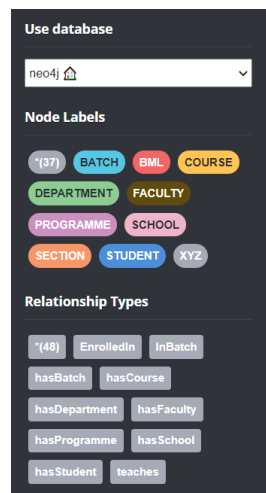


Fig:14 Neo4j Nodes And Labels

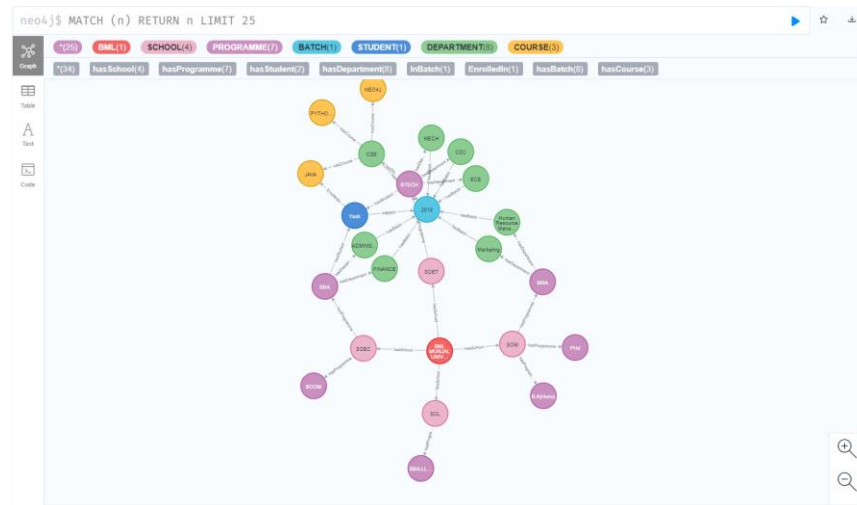


Fig:15 Neo4j

Task 4:

Building the FastApi webFramework for building the API's ,All the end points are documents in the FastApi docs which helps in testing the api and check whether the data is being fetched and retrieved.

1. *Establish connections between neo4j and fastApi using neo4j drivers*
(Authentication of the user is mandatory to establish a connection)

```
from neo4j import GraphDatabase

app = FastAPI()
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

graphdb = GraphDatabase.driver(uri="bolt://localhost:7687", auth=("neo4j", 'qwerty'), max_connection_lifetime=200)
session = graphdb.session()
```

Fig:16-1 FastAPI

2. Performing the CRUD operation with the endpoints

```
@app.get('/getNode/{Node}', tags=["Nodes"])
async def getNamesOfNodes(Node:str):
    q2 = '''MATCH (n:%s) RETURN collect(n.Name) as Nodes '''%(Node)
    result = session.run(q2)
    data = result.data()
    json_data = jsonable_encoder(data)
    print(json_data)
    return (json_data)

@app.get('/getNodebyRelation/{node}/{nodeName}/{relationName}', tags=["Nodes"])
async def getNamesOfNodes(node:str,nodeName:str,relationName:str):
    q2 = '''MATCH (n:%s),(k) where n.Name = $name AND (n)-[:%s]->(k) RETURN collect(k.Name) as Nodes '''%(node,relationName)
    map = {"name":nodeName}
    result = session.run(q2,map)
    data = result.data()
    json_data = jsonable_encoder(data)
    print(json_data)
    return (json_data)

class Name(BaseModel):
    name:str
```

Fig:16-2 Fast API

3. FastApi docs

FastAPI 0.1.0 OAS3
/openapi.json

Nodes			^
GET	/getAllLabels	Getallnodes	▼
GET	/getNodeNames	Getnodenames	▼
GET	/getNode/{Node}	Getnamesofnodes	▼
GET	/getNodebyRelation/{node}/{nodeName}/{relationName}	Getnamesofnodes	▼
DELETE	/deleteNode/{Node}/{Name}	Deletenode	▼
PUT	/update/{Node}	Updatenodes	▼
POST	/createNode/{Node}	Createnode	▼

Relations

GET	/getAllRelationNames	Getallrelations types	⌵
GET	/getrelations/{node}	Getrelations	⌵
DELETE	/relation/{relationship}	Getrelations	⌵
GET	/getrelation/{node}/{name}	Getspecificrelations	⌵
POST	/createRelation/{node1}/{node2}/{relationship}	Postrelations	⌵

STUDENTS

GET	/getAllStudents	Getallstudents	⌵
GET	/getdetails/{stdname}	Getstudentdetails	⌵
GET	/getStudentsbyProgramme/{Programme}	Getstudentbyprogramme	⌵
GET	/getstudentsbyCourse/{Course}	Getstudentbycourse	⌵
GET	/getstudentsbyDept/{Department}	Getstudentbyprogramme	⌵

FACULTY

GET	/getAllFaculty	Getallfaculty	⌵
GET	/getfacultydetails/{name}	Getfacultydetails	⌵
GET	/getFacultybyProgramme/{Programme}	Getfacultybyprogramme	⌵
GET	/getfacultybyCourse/{Course}	Getfacultybyprogramme	⌵
GET	/getfacultybyDept/{Department}	Getfacultybydept	⌵
GET	/getfacultybySchool/{School}	Getfacultybyschool	⌵
POST	/createFaculty/{name}	Createfaculty	⌵

COURSES

GET	/getCourses/{Student}	Getcourse	⌵
GET	/getAllCourses	Getcourses	⌵
POST	/Course/{courseName}/{courseCredits}	Getcourses	⌵

Fig:17 FastAPI doc's

4. Testing the Api's

The screenshot shows an API testing tool interface. At the top, the method is 'GET' and the URL is '/getdetails/{stdname}' with the description 'Getstudentdetails'. Below this, the 'Parameters' tab is active, showing a table with columns 'Name' and 'Description'. A parameter 'stdname' is listed as 'required', with a value 'PAVAN' entered in the input field. Below the table are 'Execute' and 'Clear' buttons. The 'Responses' section shows the 'Curl' command, the 'Request URL' as 'http://127.0.0.1:5000/getdetails/PAVAN', and the 'Server response' with a status code of '200'. The 'Response body' is a JSON object containing student details.

```
curl -X 'GET' \
  'http://127.0.0.1:5000/getdetails/PAVAN' \
  -H 'accept: application/json'
```

```
http://127.0.0.1:5000/getdetails/PAVAN
```

```
http://127.0.0.1:5000/getdetails/PAVAN
```

```
200
```

```
{
  "Student": {
    "Email": "pavan@bmu.edu.in",
    "RegistrationId": "19C2830059",
    "Name": "PAVAN",
    "Contact": "7813265988"
  },
  "Batch": "2019"
},
{
  "Course": {
    "PYTHON3.0",
    "JAVA",
    "NEO4J"
  },
  "Program": {
    "Name": "BTECH"
  }
}
```

Fig:18 Api Testing

Task 5:

In this particular task, I finally started connections using Angular and HttpClientModule

- Building the webserver using HttpClient.
- Connecting server to API
- Wrote the services for all the endpoints .

- *Angular Service file*

```

> app > services > main > main.service.ts > MainService > getAllStudents
1  import { Injectable } from '@angular/core';
2  import { HttpClient, HttpHeaders } from '@angular/common/http'
3
4  @Injectable({
5    providedIn: 'root'
6  })
7  export class MainService {
8
9    constructor(private http: HttpClient) { }
10
11
12    getAllNodes() {
13      return this.http.get("http://127.0.0.1:5000/getAllLabels")
14    }
15
16
17    getNodeNames(name:any){
18      return this.http.get(`http://127.0.0.1:5000/getNode/${name}`)
19    }
20
21    // getAllStudents
22    getAllStudents():any{
23      return this.http.get("http://127.0.0.1:5000/getAllStudents")
24    }
25

```

Fig:20 Service.ts file

Task 6:

In this task, started making the forms for posting all kind of information.

1. *Post Node*

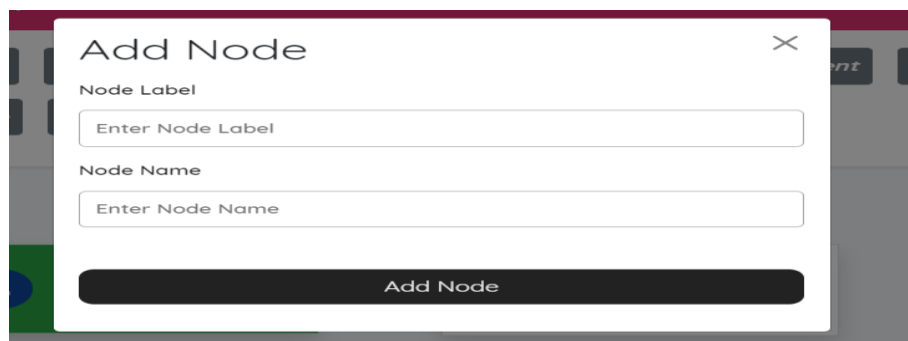


Fig:21 Add Node

2.Relationship Post

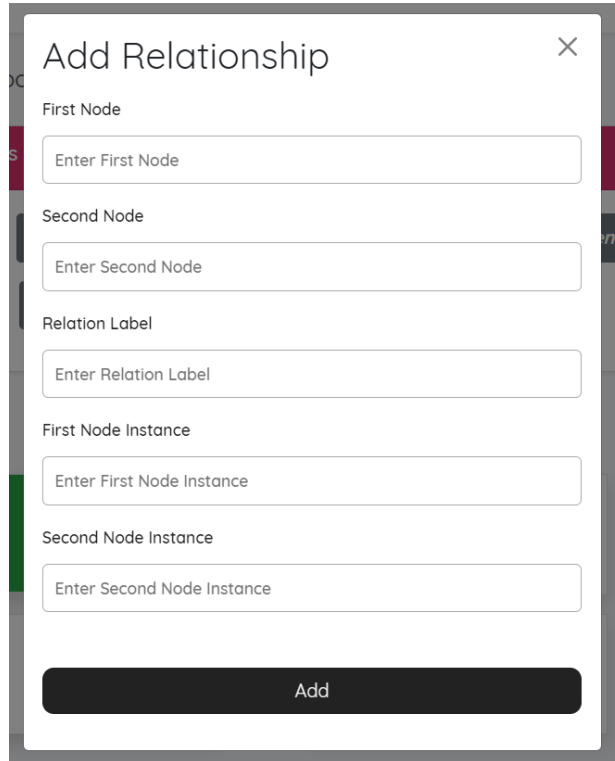
A screenshot of a mobile application form titled "Add Relationship". The form has a close button (X) in the top right corner. It contains six input fields: "First Node" with placeholder "Enter First Node", "Second Node" with placeholder "Enter Second Node", "Relation Label" with placeholder "Enter Relation Label", "First Node Instance" with placeholder "Enter First Node Instance", "Second Node Instance" with placeholder "Enter Second Node Instance", and a large black "Add" button at the bottom.

Fig:22 Add Relation

3.Add Student

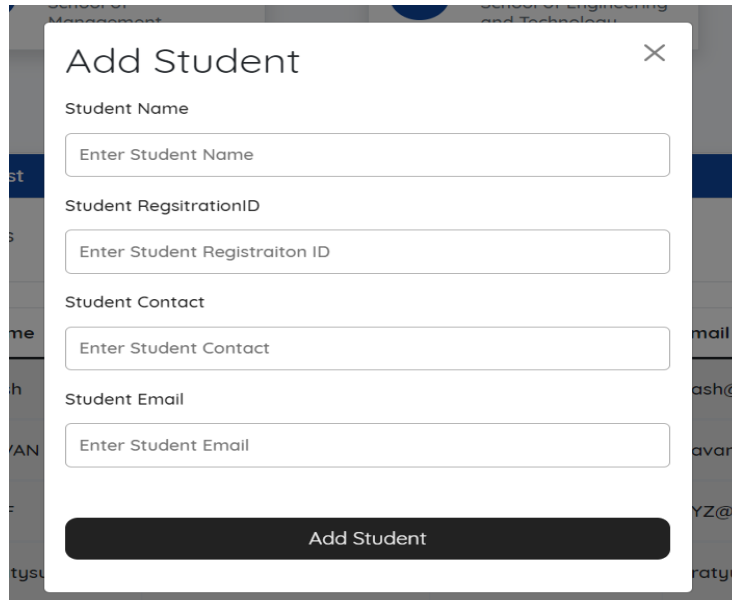
A screenshot of a mobile application form titled "Add Student". The form has a close button (X) in the top right corner. It contains five input fields: "Student Name" with placeholder "Enter Student Name", "Student RegistrationID" with placeholder "Enter Student Registratoin ID", "Student Contact" with placeholder "Enter Student Contact", "Student Email" with placeholder "Enter Student Email", and a large black "Add Student" button at the bottom.

Fig:23 Add Student

4. Add Course

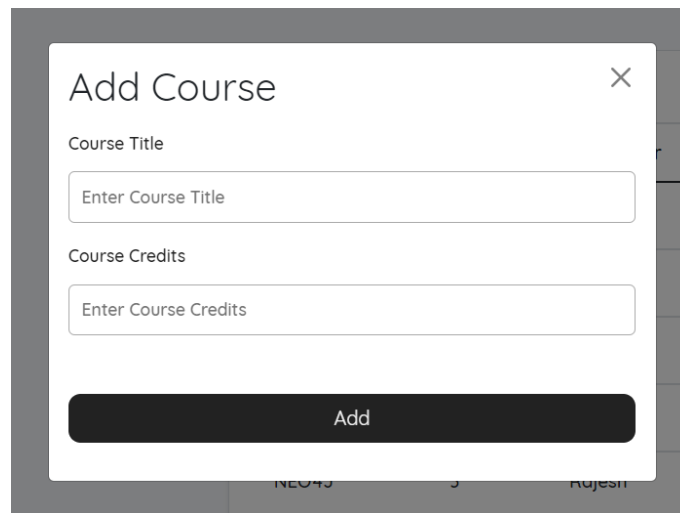
A modal dialog box titled "Add Course" with a close button (X) in the top right corner. It contains two input fields: "Course Title" with a placeholder "Enter Course Title" and "Course Credits" with a placeholder "Enter Course Credits". At the bottom is a dark "Add" button.

Fig:24 Add Course

Task 7:

Fetching the data from forms and posting them using the services

```
postStudent(data:any,name:any):any{
  console.log(data)
  const config = { headers: new HttpHeaders().set('Content-Type', 'application/json')}
  return this.http.post(`http://127.0.0.1:5000/createStudent/${name}`,data,config)
}

postFaculty(data:any,name:any):any{
  console.log(data)
  const config = { headers: new HttpHeaders().set('Content-Type', 'application/json')}
  return this.http.post(`http://127.0.0.1:5000/createFaculty/${name}`,data,config)
}

postNode(label:any,data:any):any{
  console.log(data)
  const config = { headers: new HttpHeaders().set('Content-Type', 'application/json')}
  return this.http.post(`http://127.0.0.1:5000/createNode/${label}`,data,config)
}

postRelation(data:any,node1:any,node2:any,relationship:any){
  console.log(data)
  const config = { headers: new HttpHeaders().set('Content-Type', 'application/json')}
  return this.http.post(`http://127.0.0.1:5000/createRelation/${node1}/${node2}/${relationship}`,dat
}
```

Fig:25 Services CodeView

Task 8:

*Started working on **Graphical User InterFace** implementation of the website.*

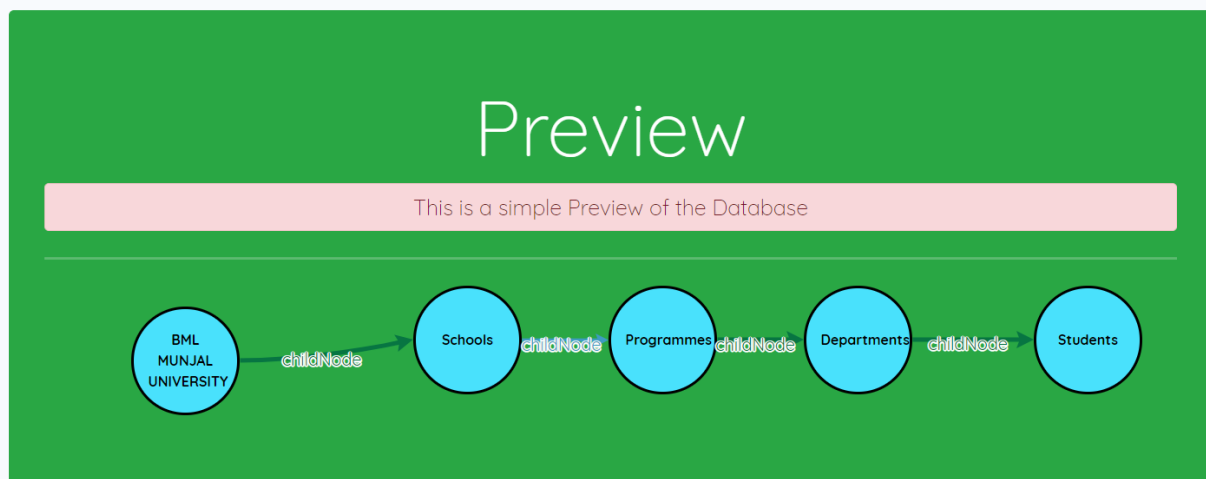
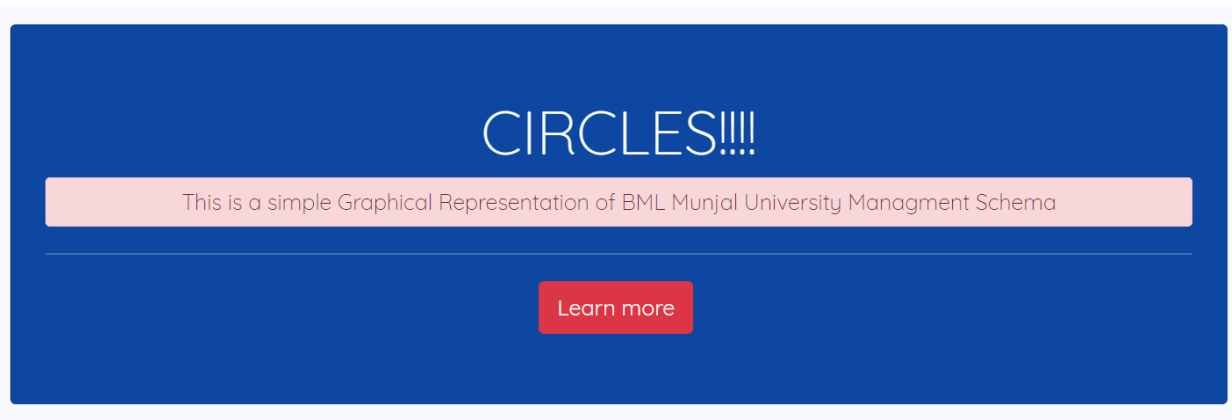


Fig:26 Graphical UI-1

- *All the view are simultaneously display after selecting the required .*

Please Select The School

SOET

Please Select The Programme

BTECH

Please Select The Department

CSE

Load Chart

Fig:27 Graphical UI-2

- *Student FlowChart*

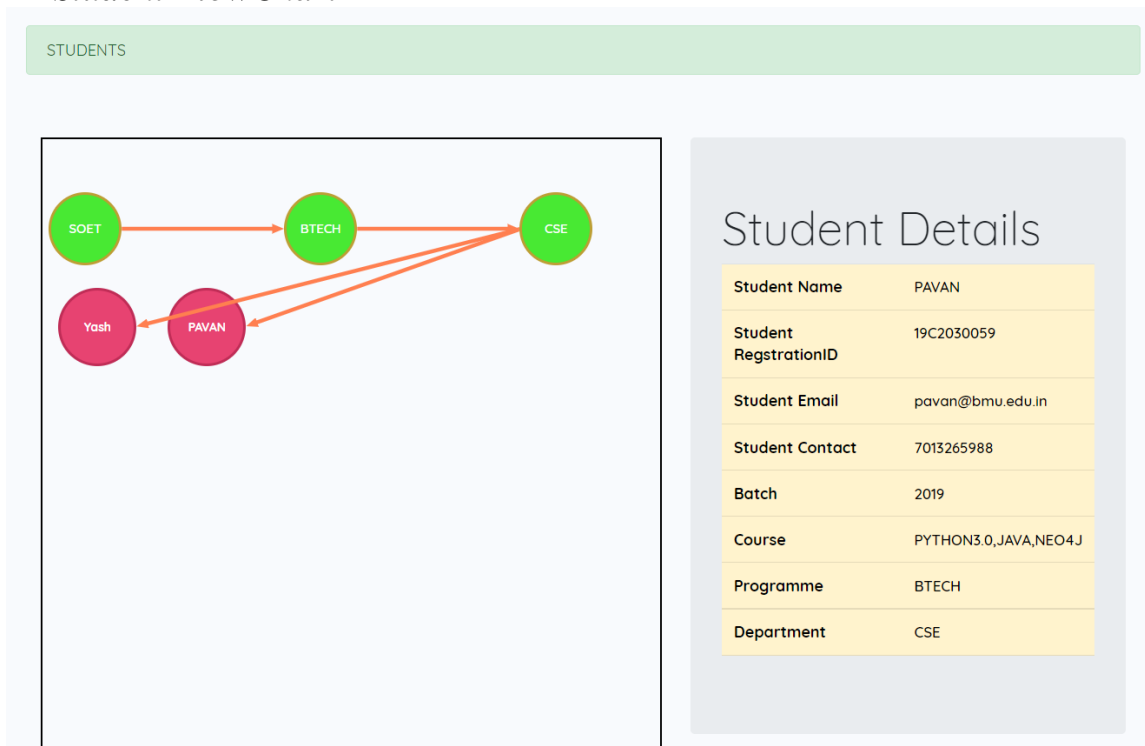


Fig:28 Student Nodes

- *Faculty FlowChart*

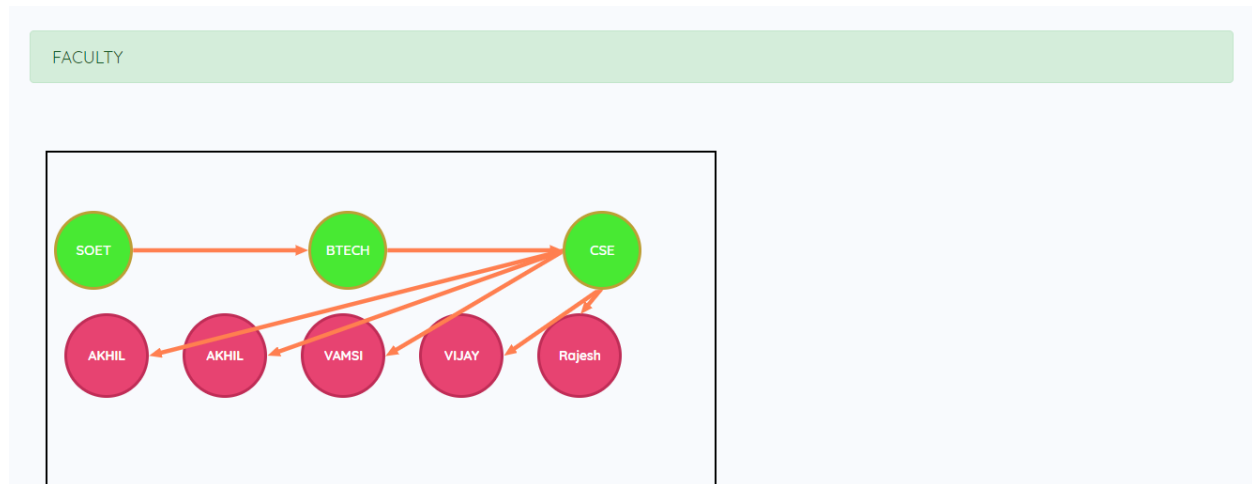


Fig:29 Faculty Nodes

That's the End.

OUT COMES

- ☐ Any University can have an userfriendly access to all types of classifications .
- ☐ It can have an complete details of each student and each faculty .
- ☐ It helps to know the interest of students.
- ☐ It can easily add new Members to the database.
- ☐ User can search for students and faculties
- ☐ Will have an graphical Represntation of all students and facultiy members.
- ☐ User will finally have a complete access for each and everything in the university.which is very crucial and important thing.

REFERENCES

- <https://fastapi.tiangolo.com/>
- <https://angular.io/docs>
- <https://neo4j.com/docs/>
- <https://www.udemy.com/course/building-an-imdb-clone-with-python-flask-and-neo4j/>
- <https://getbootstrap.com/docs/4.1/components/alerts/>

INSTALLATION'S

ANGULAR:

- Go to <https://nodejs.org/en/download/> and click on the Windows/MacOS installer. The Node.js setup window will pop up.
- In the destination folder window, pick any location of your choice and click next or simply skip this part by directly clicking next.
- In the custom setup window, simply click next.
- Finally, click install. After the installation is complete, click Finish.
- To test if the installation was successful, open your command prompt and type:

➤ **node -v**

➤ **npm -v**

- To install and setup angular in our system, first we will install Angular CLI(command line interface).

Run the following command in your command prompt:

➤ **npm install -g @angular/cli**

- To check if @angular/cli is successfully installed, run the following command in your command prompt:

➤ **ng version**

- Create a folder named 'angular' in your pc and open command prompt through that folder.
- Create your angular application using the 'ng new' command followed by the name of your application as shown below:

➤ **ng new firstAngularApp**

- It will then ask whether to add Angular routing. Press 'y' (yes).
- It will then ask which stylesheet format you would like to use. Select CSS and press enter.
- In your firstAngularApp folder, all required source files & node modules will be created.
- From your firstAngularApp folder, open cmd or just from your code editor terminal, enter the following command to run the application:

➤ **ng serve -o**

- It will perform all the build processes and give you the localhost server (<http://localhost:4200/> is the default server) where your application is running.

NEO4J:

Go to <https://neo4j.com/download/> and click the download button to download Neo4J Desktop.

- You will then be asked to fill details required to link the Neo4J desktop application with your email.

After filling the details required, click Download Desktop button to start the download.

- On the screen, you will also see an activation key. Copy that key
- Once the download completes, you simply run the installer .exe file.
- In the Installation Options window, choose the installation location of Neo4J desktop on your system. On clicking next, the installation will begin.
- Once the installation is complete, run the desktop application. A License Agreement window will open. Accept the terms & conditions.
- Then, it will ask where we want to store application data. It is preferred to use the default location.
- The next step is Software Activation. Paste the Activation key which we copied earlier in the software key box. If you do not have an activation key, enter your name, email & organization details on the left to receive the activation key.
- Once we click on Activate, the Neo4J Desktop application will start and we can start building the projects and databases.

FIGURES Table

<i>FIGURE DESCRIPTION</i>	PAGE NO
BootStrap Links	10
Dependenices	11
Bootstrap Dependency	11
Angular App Creation	11
SideBar UI	12
Relationship View	13
School's View	13
Programme's View	13
Departments View	14
Students	14
Faculty	15
Courses	15
Student Details	16
Neo4j Nodes And Labels	16
Neo4j	17
FastAPI	17
FastAPI doc's	18
Api Testing	20
Service.ts file	21

Add Node	21
Add Relation	22
Add Student	22
Add Course	23
Services CodeView	23
Graphical UI-1	24
Student Nodes	25
Faculty Nodes	26