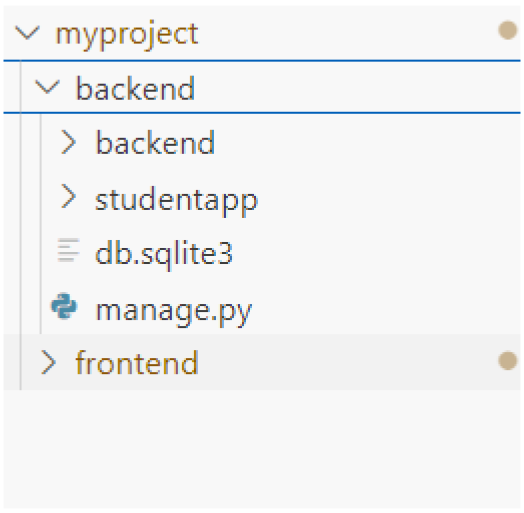
GitHub link - link

Backend Development using Django Rest Framework



In backend/backend, added below in settings.py

```
INSTALLED_APPS = [
    'rest_framework', -> for rest apis
    'studentapp', -> my app
    'corsheaders' -> for enabling cors
]
```

To enable cors

```
MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
]
```

For connecting to postgresql

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'studentdb',
        'USER': 'postgres',
        'PASSWORD': 'pgsql',
        'PORT': '5432'
    }
}
```

Frontend endpoint included in allowed origins, so all the crud operations occurs when requested front end application

```
CORS_ALLOWED_ORIGINS = [
    'http://localhost:3000',
]
```

In backend/backend, urls.py added my studentapp in the urlpatterns

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('',include('studentapp.urls')),
]
```

In backend/studentapp,

Below is the model used

```
class Student(models.Model):
    name=models.CharField(max_length=100)
    branch=models.CharField(max_length=4)
    sem=models.IntegerField()
    dob=models.DateField()
    address=models.CharField(max_length=100)
    cgpa=models.DecimalField(max_digits=4,decimal_places=2)
```

serializer for conversions

```
class StudentSerializer(serializers.ModelSerializer):
    class Meta:
        model = Student
        fields = '__all__'
```

In views.py, below are views developed using api_views using functions.

Student_list function for creating and searching based on branch

```
@api_view(['POST','GET'])
def student_list(request):
        if request.method == 'POST':
            serializer = StudentSerializer(data=request.data)
            if serializer.is_valid():
                serializer.save()
                return
Response(serializer.data,status=status.HTTP_201_CREATED)
            return
Response(serializer.errors,status=status.HTTP_400_BAD_REQUEST)

        elif request.method == 'GET':
            queryset = Student.objects.all()
            value = request.query_params.get('branch')
            serializer =
StudentSerializer(queryset.filter(branch=value),many=True)
            return Response(serializer.data,status=status.HTTP_200_OK)
```

student_operations function for get by student id, delete and update details

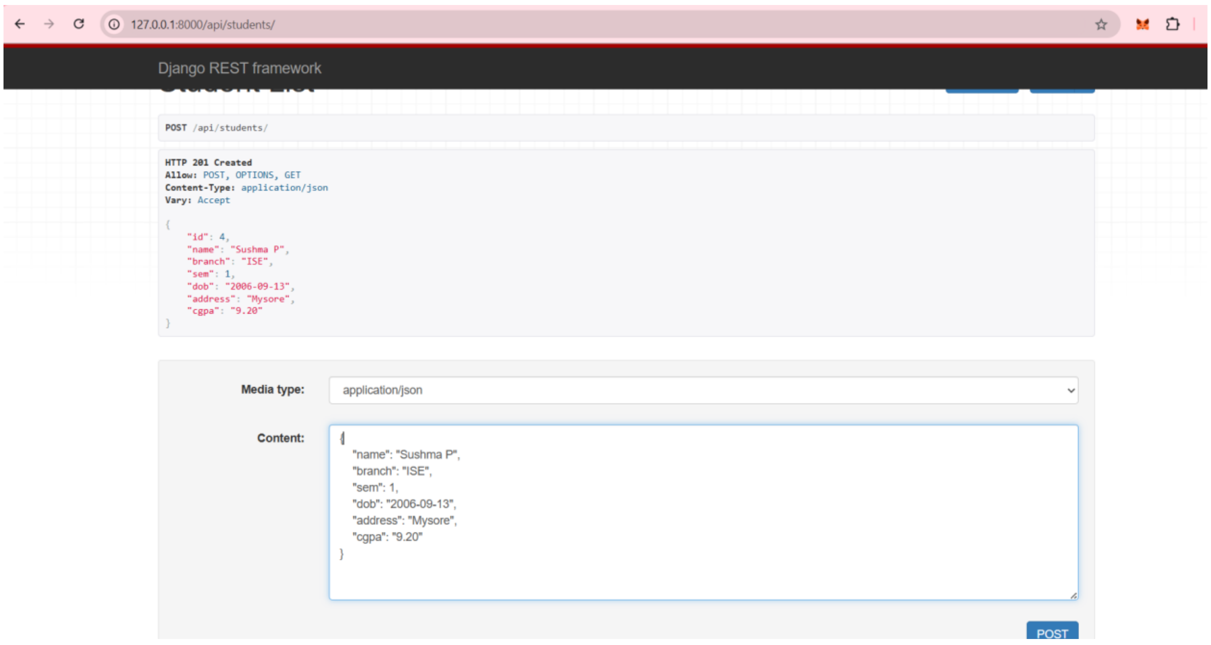
```
@api_view(['GET','DELETE','PUT'])
def student_operations(request,id):
    try:
        stud = Student.objects.get(id=id)
    except:
        return Response(status=status.HTTP_404_NOT_FOUND)

    if request.method == 'GET':
        serializer = StudentSerializer(stud)
        return Response(serializer.data,status=status.HTTP_200_OK)

    if request.method == 'PUT':
        serializer = StudentSerializer(stud,data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data,status=status.HTTP_200_OK)
        return
Response(serializer.errors,status=status.HTTP_500_INTERNAL_SERVER_ERROR)

    if request.method == 'DELETE':
        stud.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

In urls.py,

```
urlpatterns = [
    path('api/students/<int:id>',views.student_operations,name='student_operat
ions'),
    path('api/students/',views.student_list,name='student_list'),
]
```

For get,put and delete operations, all are using 1st api endpoints.

For create and search using branch as filter, 2nd api endpoint is used.


127.0.0.1:8000/api/students/

Django REST framework

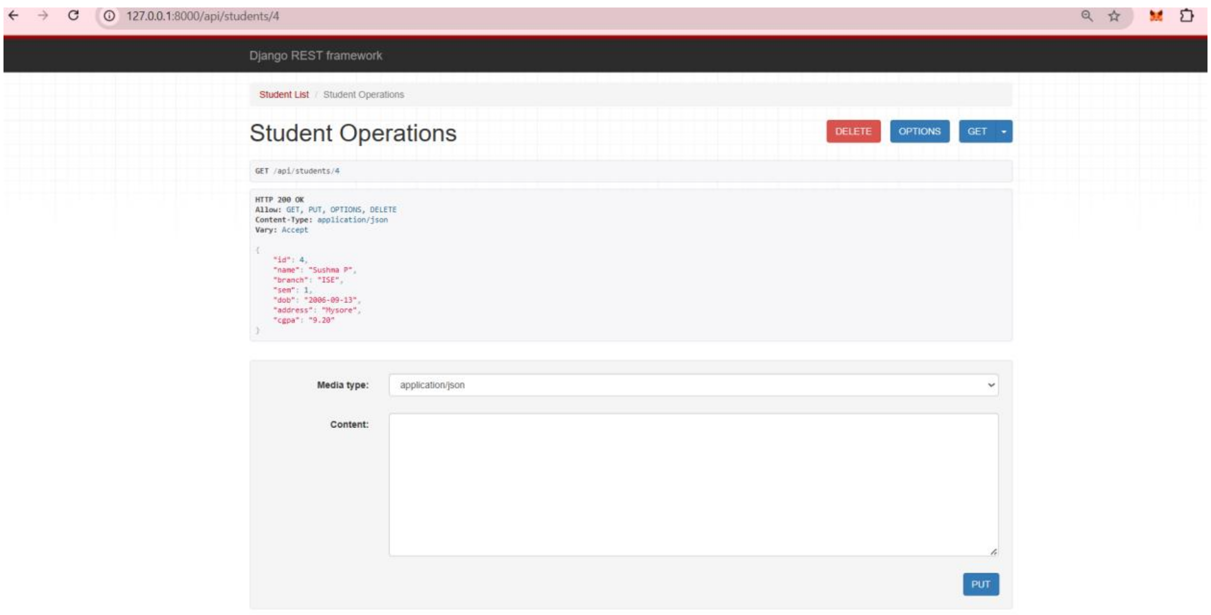
```
POST /api/students/
```

```
HTTP 201 Created
Allow: POST, OPTIONS, GET
Content-Type: application/json
Vary: Accept

{
    "id": 4,
    "name": "Sushma P",
    "branch": "ISE",
    "sem": 1,
    "dob": "2006-09-13",
    "address": "Mysore",
    "cgpa": "9.20"
}
```

Media type: application/json

Content:

```
{
    "name": "Sushma P",
    "branch": "ISE",
    "sem": 1,
    "dob": "2006-09-13",
    "address": "Mysore",
    "cgpa": "9.20"
}
```


127.0.0.1:8000/api/students/4

Django REST framework

Student List / Student Operations

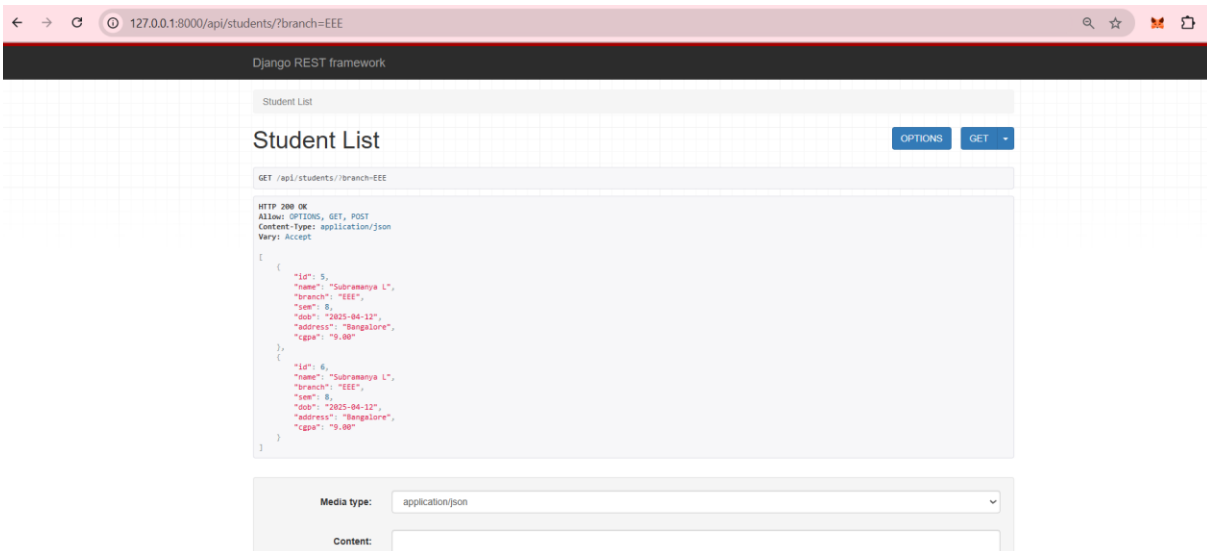
Student Operations

```
GET /api/students/4
```

```
HTTP 200 OK
Allow: GET, PUT, OPTIONS, DELETE
Content-Type: application/json
Vary: Accept

{
    "id": 4,
    "name": "Sushma P",
    "branch": "ISE",
    "sem": 1,
    "dob": "2006-09-13",
    "address": "Mysore",
    "cgpa": "9.20"
}
```

Media type: application/json

Content:



Student List

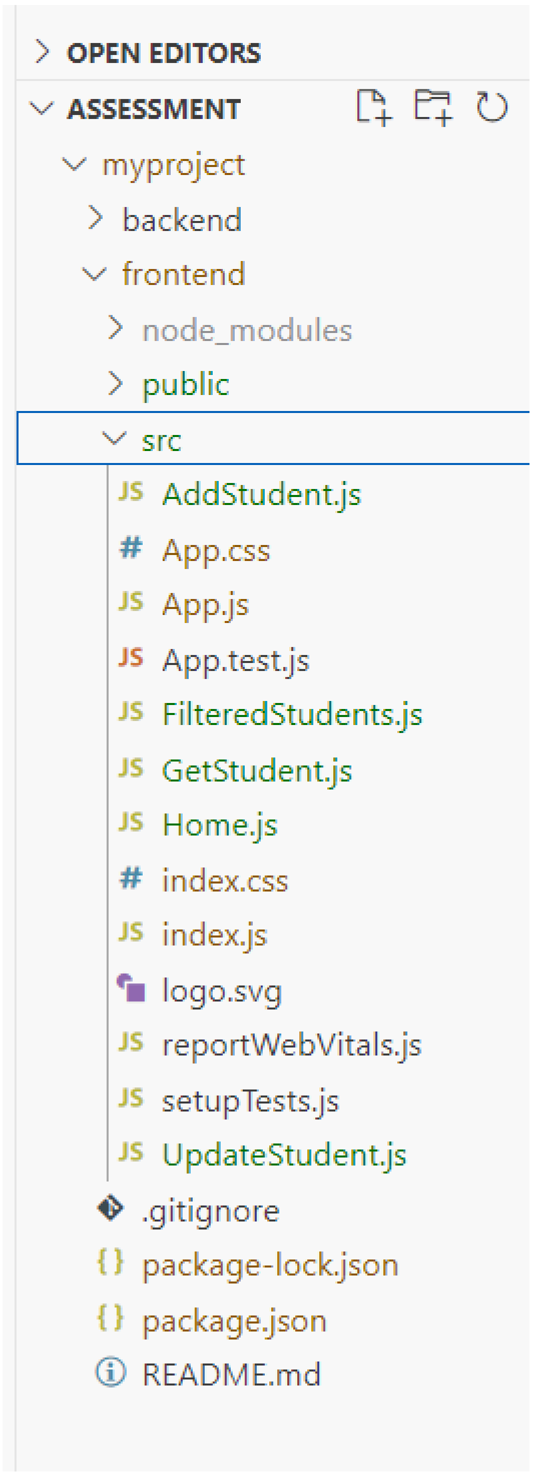
```
GET /api/students/?branch=EEE
```

```
HTTP 200 OK
Allow: OPTIONS, GET, POST
Content-Type: application/json
Vary: Accept

[
    {
        "id": 5,
        "name": "Subramanya L",
        "branch": "EEE",
        "sem": 8,
        "dob": "2025-04-12",
        "address": "Bangalore",
        "cgpa": "9.00"
    },
    {
        "id": 6,
        "name": "Subramanya L",
        "branch": "EEE",
        "sem": 8,
        "dob": "2025-04-12",
        "address": "Bangalore",
        "cgpa": "9.00"
    }
]
```

Media type: application/json

Content:

Frontend using React



```
myproject
  backend
  frontend
    node_modules
    public
    src
      AddStudent.js
      App.css
      App.js
      App.test.js
      FilteredStudents.js
      GetStudent.js
      Home.js
      index.css
      index.js
      logo.svg
      reportWebVitals.js
      setupTests.js
      UpdateStudent.js
    .gitignore
    package-lock.json
    package.json
    README.md
```

In app.js, defined all the routes and nav links

```
<nav className="navbar navbar-expand-lg navbar-dark bg-dark fw-bolder">
            <Link className="navbar-brand" to='/home' >Students App</Link>
            <ul className='navbar-nav'>
```

```
              <li className='nav-item'>
                <Link className='nav-link' to="/add" >Add Student</Link>
              </li>
              <li className='nav-item'>
                <Link className='nav-link' to="/get" >View Student</Link>
              </li>
              <li className='nav-item'>
                <Link className='nav-link' to="/filter" >Filter</Link>
              </li>
            </ul>
          </nav>


<Routes>
          <Route path='/' element={<Home/>} />
          <Route path='/home' element={<Home/>} />
          <Route path='/add' element={<AddStudent/>} />
          <Route path='/get' element={<GetStudent/>} />
          <Route path='/update/:id' element={<UpdateStudent/>} />
          <Route path='/filter' element={<FilteredStudents/>} />
        </Routes>
```

Below are the components used. Used bootstrap for styling and axios to connect with backend and do crud operations

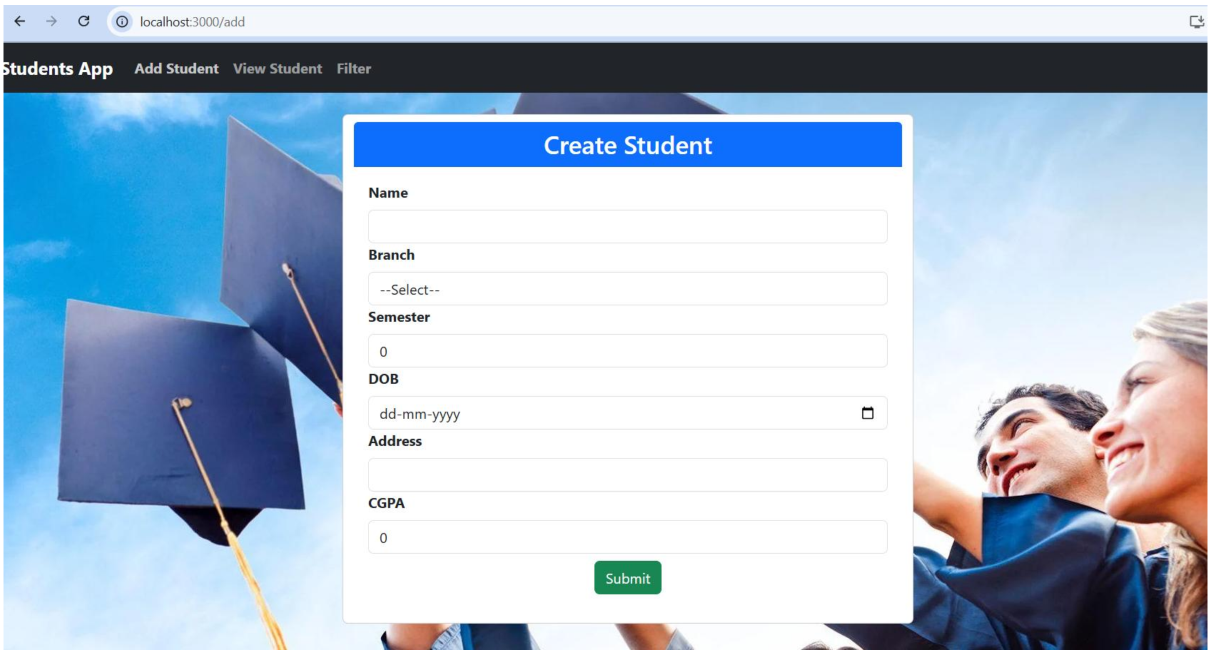
Home

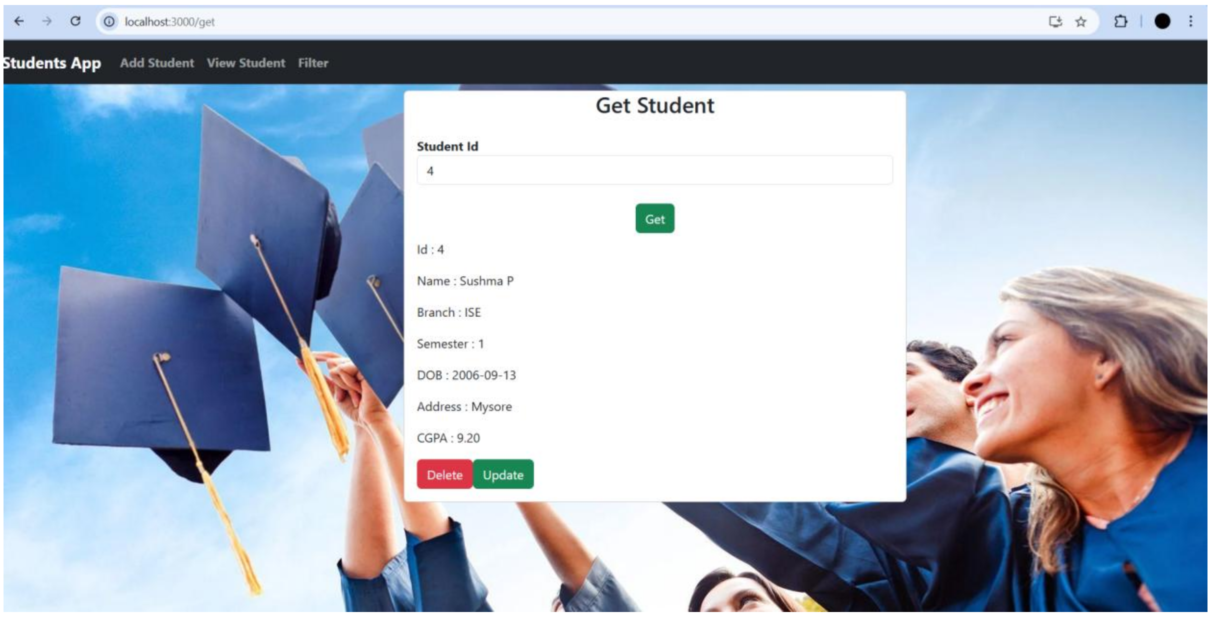
AddStudent – has a form to fill data and submit. Success message is provided on insertion

GetStudent – takes student id as input and provides all details. Link for update and delete is provided

UpdateStudent – Updates student details.

FilteredStudents – To search/filter based on branch is enabled and displayed them in a table format.





Id : 4

Name : Sushma P

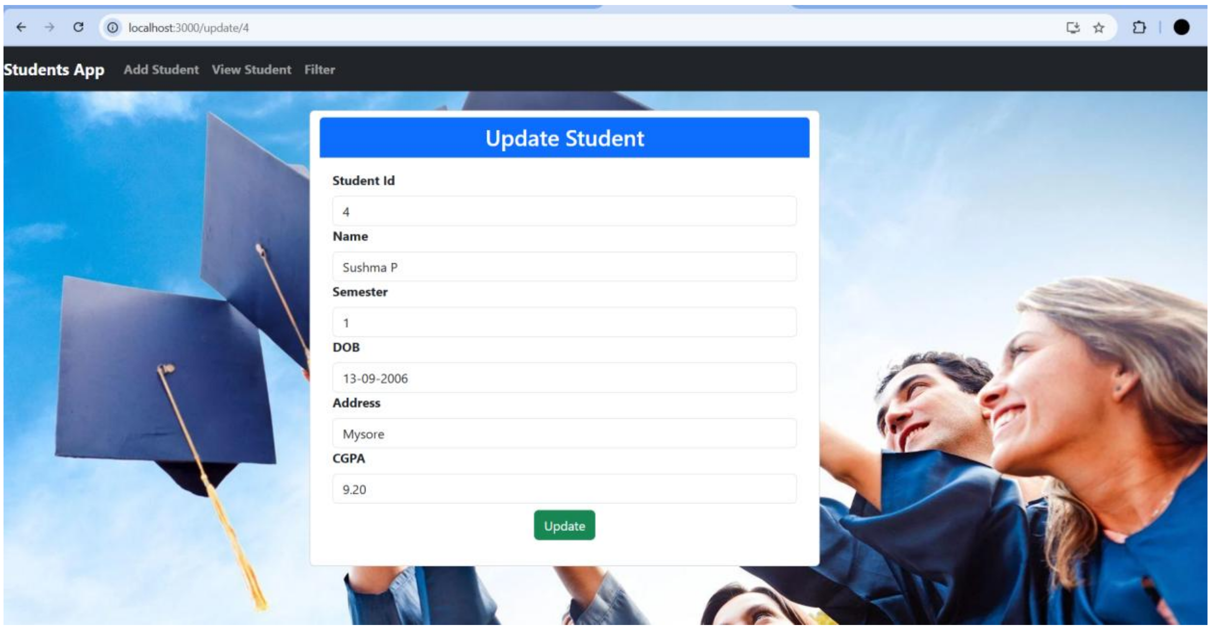
Branch : ISE

Semester : 1

DOB : 2006-09-13

Address : Mysore

CGPA : 9.20


localhost:3000/update/4

Students App   Add Student   View Student   Filter

Update Student

Student Id: 4

Name: Sushma P

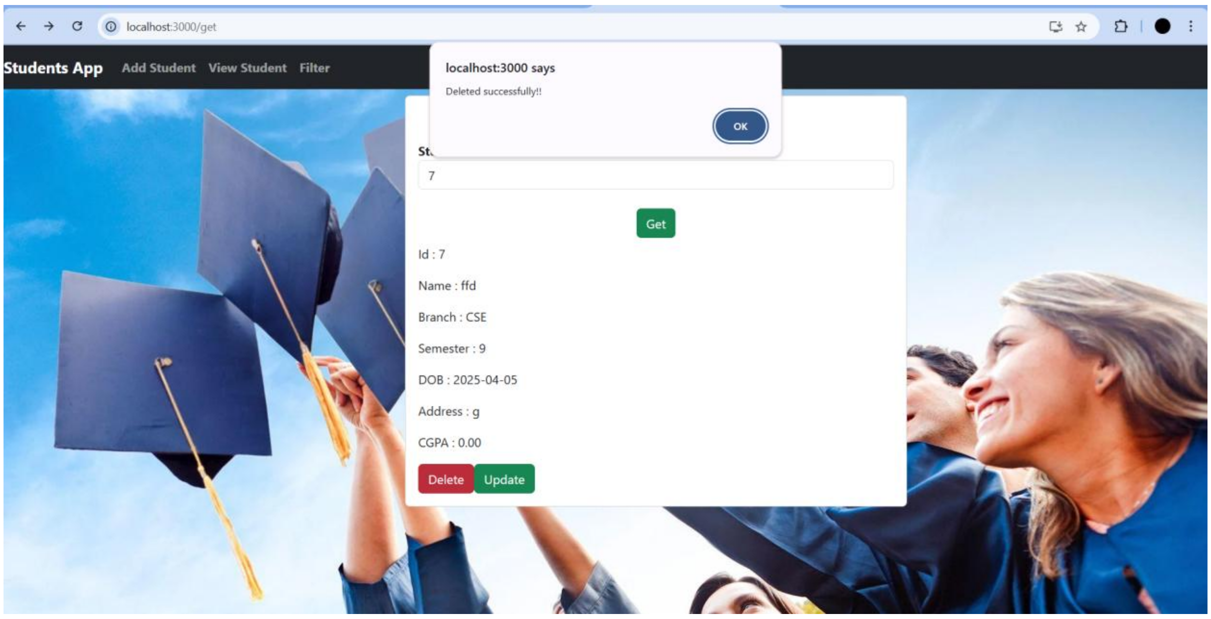
Semester: 1

DOB: 13-09-2006

Address: Mysore

CGPA: 9.20

Update


localhost:3000/get

Students App   Add Student   View Student   Filter

localhost:3000 says

Deleted successfully!!

OK

7

Get

Id : 7

Name : ffd

Branch : CSE

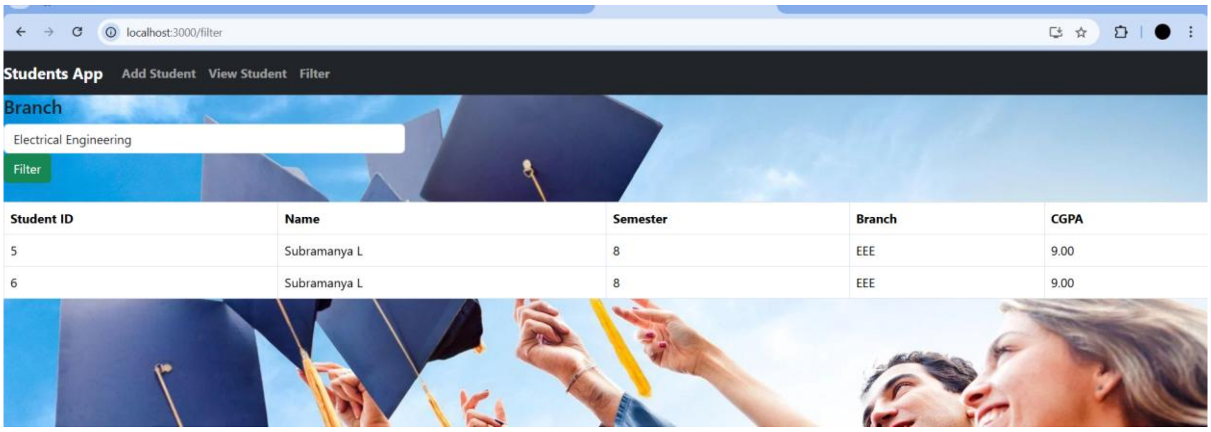
Semester : 9

DOB : 2025-04-05

Address : g

CGPA : 0.00

Delete   Update


localhost:3000/filter

Students App   Add Student   View Student   Filter

Branch

Electrical Engineering

Filter

Student ID	Name	Semester	Branch	CGPA
5	Subramanya L	8	EEE	9.00
6	Subramanya L	8	EEE	9.00