PRACTICAL:

functionality to add a student to a session from scratch:

· In src\models\students.ts:

```
import client from '../database'
import bcrypt from 'bcrypt'
import config from '../config'

const salt_rounds  = config.salt, pepper = config.pepper



export type student = {
    id?: number
    first_name: string
    last_name: string
    password: string
}

export const hashing = (password: string) => {
    const salt = parseInt(salt_rounds as unknown as string, 10)
    return bcrypt.hashSync(`${password}${pepper}`, salt)
}



//add student
    async add(
```

```
    s: student
  ): Promise<{ id: number; first_name: string; last_name: string }> {
    try {
      const sql = 'INSERT INTO students (first_name, last_name, password) VALUES($1, $2, $3) RETURNING id, first_name, last_name ;'
      const conn = await client.connect()

      const hash = hashing(s.password)

      const result = await conn.query(sql, [
        s.first_name,
        s.last_name,
        hash,
      ])

      conn.release()
      return result.rows[0]
    } catch (err) {
      throw new Error(
        `Could not add new student ${s.first_name} ${s.last_name}. Error: ${err}`
      )
    }
  }
```

· In src\handlers\students_handler.ts :

```
import { NextFunction, Request, Response } from 'express'
```

```
import { studentModel } from '../models/students'
import jwt from 'jsonwebtoken'
import config from '../config'
import errorMiddleWare from '../middleware/errorMiddleware'


const student_model = new studentModel()




export const add = async (
    req: Request,
    res: Response,
    next: NextFunction
) => {
    try {
        const newstudent = await student_model.create(req.body)
        const token = jwt.sign(
            { student: newstudent },
            config.token as unknown as string
        )
        res.json({
            status: 'success',
            student: newstudent,
            token: token,
            message: 'Student added Successfully',
```

```
        })
    } catch (error) {
        next(errorMiddleWare(error as Error, req, res))
    }
}
```

- In src\routes\api\students_route.ts :

```
import { Router } from 'express'

import * as handler from '../../handlers/students_handler'

const routes = Router()

routes.route('/').post(handler.add)

export default routes
```

THEORETICAL:

Explain how to get a unique list of rows in a SELECT statement

- We can use ``` SELECT DISTINCT ``` query to get only unique values in a column
- Note that ```DISTINCT``` only operates on a single column.

EXAMPLE:

```sql
SELECT DISTINCT category FROM products
```

NAME: Fatma Yasser Sabry

LinkedIn: https://www.linkedin.com/in/fatma-yasser-68b784202/

GitHup: https://github.com/tmtm8976

Ahmed Abu Qahf [LinkedIn][GitHub][Dev.to]

Theoretical

To get a unique list of rows in a `SELECT` statement, use the keyword `DISTINCT` before the column name as below:

```
SELECT
   DISTINCT column_name
FROM
   table_name;
```

Practical [Code]

Mina Sameh Wadie [LinkedIn][Github]

Theoretical

```
SELECT * FROM table WHERE (id,  aRow) IN (
        SELECT id, aRow FROM user GROUP BY id,  aRow HAVING count(*) = 1
);
```

Practical [Github]