

Student Progress Tracker

The screenshot shows a web browser window with the URL `127.0.0.1:5000` in the address bar. The page title is "Student Progress Tracker". The navigation bar includes links for "Suggested Sites", "Dashboard", "Students", "Scores", "Analytics", and "Logout". The main content area features a large heading "Welcome to the Student Progress Tracker" and a sub-instruction "Select a student to view reports or visit the dashboard." A blue button labeled "Go to Dashboard" is centered below the text.

Outcome in MS Edge Browser

The screenshot shows the Microsoft Edge browser window with the title "student-tracker". The left sidebar displays the file structure of a Django project named "STUDENT-TRACKER". The "index.html" file is selected in the "OPEN EDITORS" tab, showing its content:

```
app > templates > index.html
1  {% include '_navbar.html' %}
2  <!DOCTYPE html>
3  <html>
4  <head>
5  | <title>Student Progress Tracker</title>
6  | <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
7  </head>
8  <body class="p-4">
9  | <div class="container text-center">
10 | | <h2>Welcome to the Student Progress Tracker</h2>
11 | | <p class="lead">Select a student to view reports or visit the dashboard.</p>
12 | | <a href="/dashboard" class="btn btn-primary mt-3">Go to Dashboard</a>
13 | </div>
14 </body>
15 </html>
```

The bottom status bar shows the terminal output: "(base) PS C:\Users\jagdevsingh\dsanjh.RHINO\student-tracker>". The system tray indicates it's 33°C and Haze.

VS Code view 1

The screenshot shows the Visual Studio Code interface. The left sidebar displays the project structure for 'STUDENT-TRACKER'. The code editor window shows the file 'index.html' with the following content:

```

1  {% include '_navbar.html' %}
2  <!DOCTYPE html>
3  <html>
4  <head>
5      <title>Student Progress Tracker</title>
6      <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
7  </head>
8  <body class="p-4">
9      <div class="container text-center">
10         <h2>Welcome to the Student Progress Tracker</h2>
11         <p class="lead">Select a student to view reports or visit the dashboard.</p>
12         <a href="/dashboard" class="btn btn-primary mt-3">Go to Dashboard</a>
13     </div>
14 </body>
15 </html>
16

```

The terminal tab shows the command 'flask run' being executed, with a warning message:

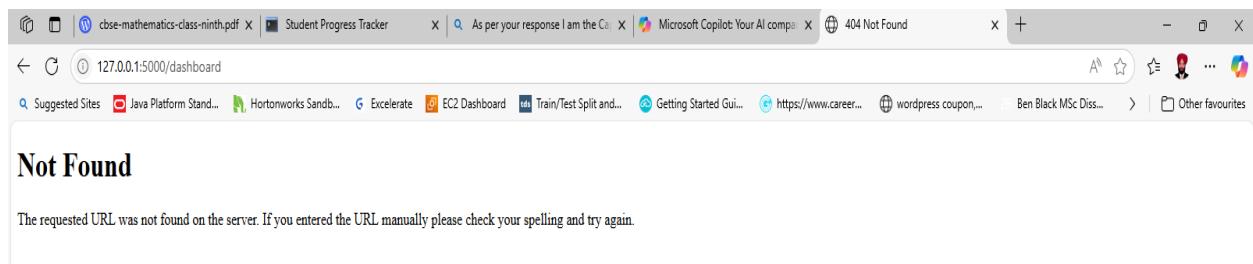
```

(base) PS C:\Users\jagdevsinghdosanjh.RHINO\student-tracker> flask run
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit

```

The taskbar at the bottom includes icons for file operations, search, and various applications like Microsoft Edge, File Explorer, and Task View.

VS Code View 2 (after flask run command execution)



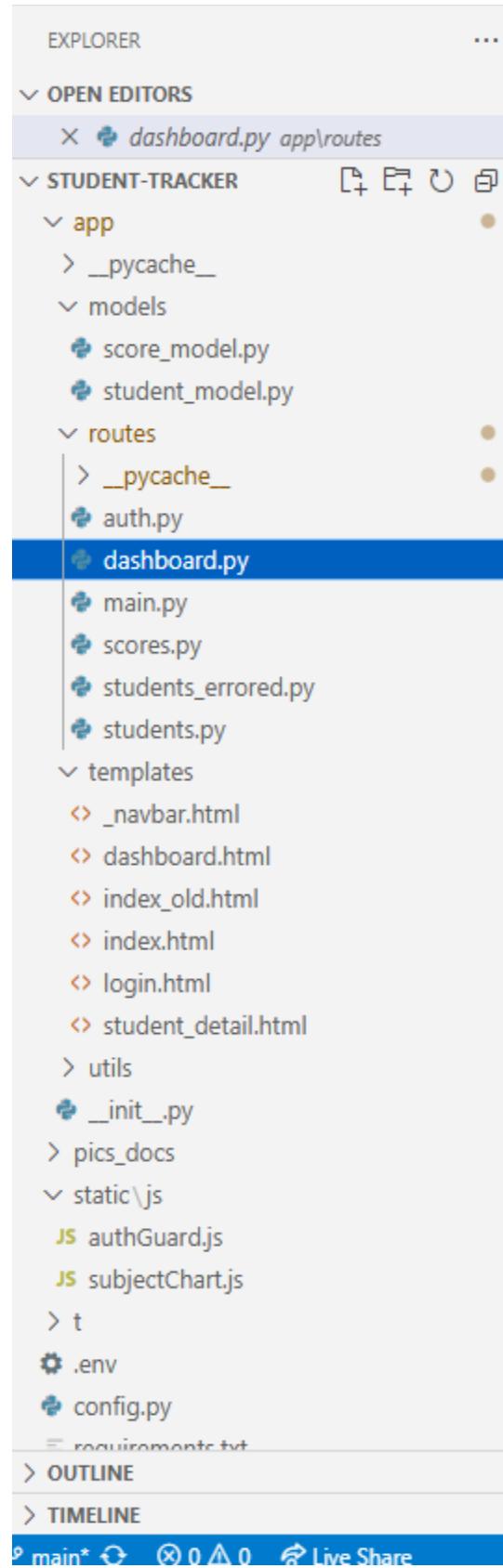
Dashboard Not found when navbar link Dashboard (as per blueprint) clicked to open it.



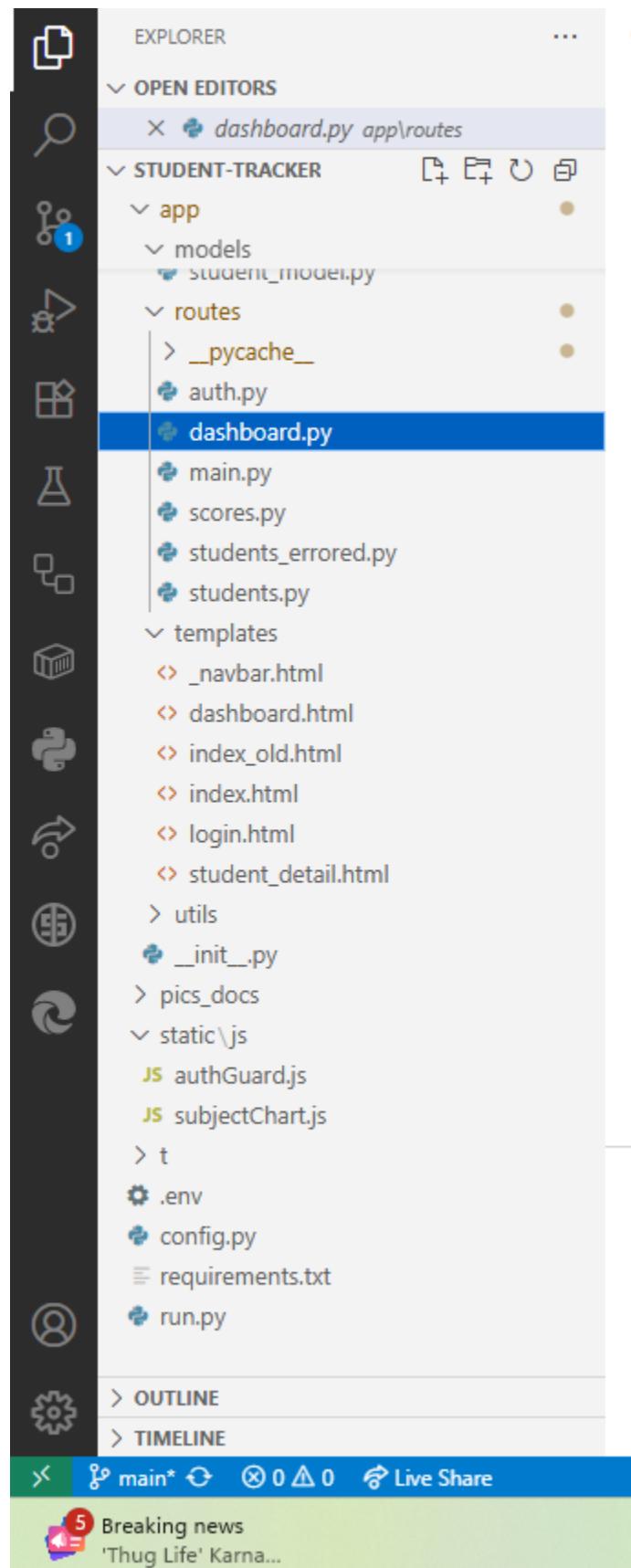
Warning 404 at VS Code Terminal for Dashboard resource

Other resources like Students Scores Analytics are also not working. Just 404 Messages for these resources.

Directory Structure Snapshot 1 Top View



Directory Structure Snapshot 1 Bottom View



Directory Tree (SPT)

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
(base) PS C:\Users\jagdevsinghdosanjh.RHINO\student-tracker> tree
Folder PATH listing
Volume serial number is A2BB-0B67
C:.
└── app
    ├── models
    ├── routes
    │   └── __pycache__
    │       ├── templates
    │       ├── utils
    │       │   └── __pycache__
    │       └── __pycache__
    ├── pics_docs
    ├── static
    └── t
    └── js
```

○ (base) PS C:\Users\jagdevsinghdosanjh.RHINO\student-tracker>

Models directory Contents

```
(base) PS C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\app\models> dir
```

○

```
Directory: C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\app\models
```

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
-a---	17-06-2025 01:41	316	score_model.py
-a---	17-06-2025 02:31	1432	student_model.py

```
(base) PS C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\app\models>
```

Routes Directory Contents

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
● (base) PS C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\app\routes> dir
```

```
Directory: C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\app\routes
```

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
d----	17-06-2025 17:41		__pycache__
-a---	17-06-2025 01:42	717	auth.py
-a---	17-06-2025 17:22	177	dashboard.py
-a---	17-06-2025 00:53	165	main.py
-a---	17-06-2025 00:08	2107	scores.py
-a---	17-06-2025 17:26	7549	students.py
-a---	17-06-2025 01:45	4226	students_errorred.py

```
○ (base) PS C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\app\routes>
```

Templates directory contents

```
PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

(base) PS C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\app\templates> dir

Directory: C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\app\templates

Mode LastWriteTime Length Name
---- ----- ----- 
-a--- 16-06-2025 23:48 1136 dashboard.html
-a--- 17-06-2025 02:22 519 index.html
-a--- 17-06-2025 00:55 1717 index_old.html
-a--- 16-06-2025 23:13 1346 login.html
-a--- 17-06-2025 02:11 1335 student_detail.html
-a--- 17-06-2025 02:08 830 _navbar.html
```

○ (base) PS C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\app\templates> █

Utils Directory Contents

```
PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

(base) PS C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\app\utils> dir

Directory: C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\app\utils

Mode LastWriteTime Length Name
---- ----- ----- 
d---- 17-06-2025 00:34 __pycache__
-a--- 17-06-2025 00:30 1202 auth_utils.py
-a--- 16-06-2025 23:23 614 image_handler.py
-a--- 17-06-2025 00:07 1217 pdf_generator.py
```

○ (base) PS C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\app\utils> █

Complete app folder in student-tracker project

```
PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

(base) PS C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\app> dir

Directory: C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\app

Mode LastWriteTime Length Name
---- ----- ----- 
d---- 16-06-2025 22:40 models
d---- 17-06-2025 00:53 routes
d---- 17-06-2025 00:55 templates
d---- 17-06-2025 00:21 utils
d---- 17-06-2025 17:20 __pycache__
-a--- 17-06-2025 15:06 1264 __init__.py
```

○ (base) PS C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\app> █

static/js contents

```
PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

(base) PS C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\static\js> dir

Directory: C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\static\js

Mode                LastWriteTime        Length Name
----                -----          -----
-a---  16-06-2025    23:50              77 authGuard.js
-a---  16-06-2025    23:37             568 subjectChart.js

○ (base) PS C:\Users\jagdevsinghdosanjh.RHINO\student-tracker\static\js>
```

Files and codes

Files

1. score_model.py : (inside models of app directory)
2. student_model.py : (inside models)
3. auth.py: (inside routes of app directory)
4. dashboard.py: (inside routes)
5. main.py: (inside routes)
6. scores.py: (inside routes)
7. students.py: (inside routes)
8. _navbar.html: (inside templates of app directory)
9. dashboard.html: (inside templates)
10. index.html: (inside templates)
11. login.html: (inside templates)
12. student_detail.html: (inside templates)
13. auth_utils.py: (inside utils of app directory)
14. image_handler.py: (inside utils)
15. pdf_generator.py(inside utils)
16. __init__.py (inside app directory)
17. authGuard.js: (inside static\js)
18. subjectChart.js: (inside static\js)
19. .env : (inside student-tracker project directly)
20. config.py: (inside student-tracker project directly)
21. requirements.txt: (inside student-tracker project directly)
22. run.py : (inside student-tracker project directly)

Codes

1. score_model.py

```
# score_model.py

from datetime import datetime
from bson import ObjectId

class Score:
    def __init__(self, student_id, subject, marks_obtained, total_marks, exam_date,
                 created_at=None, updated_at=None, _id=None):
        self.id = str(_id) if _id else None
        self.student_id = student_id # Should already be ObjectId
        self.subject = subject
        self.marks_obtained = marks_obtained
        self.total_marks = total_marks
        self.exam_date = exam_date
        self.created_at = created_at or datetime.utcnow()
        self.updated_at = updated_at or datetime.utcnow()

    def to_dict(self):
        return {
            "_id": ObjectId(self.id) if self.id else None,
            "student_id": ObjectId(self.student_id),
            "subject": self.subject,
            "marks_obtained": self.marks_obtained,
            "total_marks": self.total_marks,
            "exam_date": self.exam_date,
            "created_at": self.created_at,
            "updated_at": self.updated_at
        }

    @staticmethod
    def from_dict(data):
        return Score(
            student_id=data["student_id"],
            subject=data["subject"],
            marks_obtained=data["marks_obtained"],
            total_marks=data["total_marks"],
            exam_date=data["exam_date"],
            created_at=data.get("created_at"),
            updated_at=data.get("updated_at"),
            _id=data.get("_id")
        )
```

2. student_model.py

```
# student_model.py

from bson import ObjectId

class Student:
    def __init__(self, name, roll_no, email, class_name=None, date_of_birth=None,
                 guardian_contact=None, _id=None):
        self.id = str(_id) if _id else None
        self.name = name
        self.roll_no = roll_no
        self.email = email
        self.class_name = class_name
        self.date_of_birth = date_of_birth
        self.guardian_contact = guardian_contact

    def to_dict(self):
        return {
            "full_name": self.name,
            "roll_number": self.roll_no,
            "class_name": self.class_name,
            "date_of_birth": self.date_of_birth,
            "guardian_contact": self.guardian_contact,
            "_id": ObjectId(self.id) if self.id else None,
            "email": self.email
        }

    @staticmethod
    def from_dict(data):
        return Student(
            name=data.get("full_name"),
            roll_no=data.get("roll_number"),
            email=data.get("email"),
            class_name=data.get("class_name"),
            date_of_birth=data.get("date_of_birth"),
            guardian_contact=data.get("guardian_contact"),
            _id=data.get("_id")
        )
```

3. auth.py

```
# app/routes/auth.py

import jwt
import datetime
from flask import Blueprint, request, jsonify, render_template
from app import mongo
import os
from dotenv import load_dotenv

load_dotenv()

auth_bp = Blueprint('auth', __name__)
SECRET_KEY = os.getenv("SECRET_KEY", "default_secret_key")

# 🔒 Login API (POST)
@auth_bp.route('/login', methods=['POST'])
def login_api():
    data = request.get_json()
    if not data or "username" not in data or "password" not in data:
        return jsonify({"error": "Username and password required"}), 400

    user = mongo.db.users.find_one({"username": data["username"]})
    if not user or user["password"] != data["password"]:
        return jsonify({"error": "Invalid credentials"}), 401

    token = jwt.encode({
        "user_id": str(user["_id"]),
        "exp": datetime.datetime.utcnow() + datetime.timedelta(hours=2)
    }, SECRET_KEY, algorithm="HS256")

    return jsonify({"token": token})

# 🌐 Login Page (GET)
@auth_bp.route('/login', methods=['GET'])
def login_page():
    return render_template('login.html')
```

4. dashboard.py

```
# In dashboard.py
from flask import Blueprint, render_template

dashboard_bp = Blueprint('dashboard', __name__)

@dashboard_bp.route('/dashboard')
def dashboard():
    return render_template('dashboard.html')
```

5. main.py

```
from flask import Blueprint, render_template

main_bp = Blueprint('main', __name__)

@main_bp.route('/')
def home():
    return render_template('index.html')
```

6. scores.py

```
from flask import Blueprint, request, jsonify
from app import mongo
from datetime import datetime
from bson.objectid import ObjectId
from app.utils.auth_utils import token_required
from flask import send_file
from app.utils.pdf_generator import generate_student_report

score_bp = Blueprint('scores', __name__)

@score_bp.route('/', methods=['POST'])
@token_required
def add_score():
    data = request.get_json()
    required = ["student_id", "subject", "marks_obtained", "total_marks",
    "exam_date"]
    if not all(k in data for k in required):
        return jsonify({"error": "Missing fields"}), 400

    score = {
```

```

        "student_id": ObjectId(data["student_id"]),
        "subject": data["subject"],
        "marks_obtained": int(data["marks_obtained"]),
        "total_marks": int(data["total_marks"]),
        "exam_date": data["exam_date"],
        "created_at": datetime.utcnow(),
        "updated_at": datetime.utcnow()
    }

    mongo.db.scores.insert_one(score)
    return jsonify({"message": "Score added"}), 201

@score_bp.route('/student/<student_id>', methods=['GET'])
def get_scores_by_student(student_id):
    scores = mongo.db.scores.find({"student_id": ObjectId(student_id)})
    result = []
    for s in scores:
        result.append({
            "subject": s["subject"],
            "marks_obtained": s["marks_obtained"],
            "total_marks": s["total_marks"],
            "exam_date": s["exam_date"]
        })
    return jsonify(result), 200

@score_bp.route('/report/<student_id>', methods=['GET'])
@token_required
def download_report(student_id):
    student = mongo.db.students.find_one({"_id": ObjectId(student_id)})
    if not student:
        return jsonify({"error": "Student not found"}), 404

    scores = list(mongo.db.scores.find({"student_id": ObjectId(student_id)}))
    pdf_buffer = generate_student_report(student, scores)

    filename = f"{student['full_name'].replace(' ', '_')}_Report.pdf"
    return send_file(pdf_buffer, as_attachment=True, download_name=filename,
                    mimetype='application/pdf')

```

7. students.py

```
# app/routes/students.py

from flask import Blueprint, request, jsonify, render_template, send_file
from bson.objectid import ObjectId
from datetime import datetime
from app import mongo
from app.utils.auth_utils import token_required
from app.utils.image_handler import save_image, get_image, delete_image
import io

student_bp = Blueprint('students', __name__)

# NEW Add a new student
@student_bp.route('/', methods=['POST'])
@token_required
def add_student():
    data = request.form
    photo = request.files.get('photo')

    required_fields = ("full_name", "roll_number", "class_name", "date_of_birth",
"guardian_contact")
    if not all(k in data for k in required_fields):
        return jsonify({"error": "Missing required fields"}), 400

    photo_id = save_image(photo) if photo else None

    student_doc = {
        "full_name": data['full_name'],
        "roll_number": data['roll_number'],
        "class_name": data['class_name'],
        "date_of_birth": data['date_of_birth'],
        "guardian_contact": data['guardian_contact'],
        "photo_id": photo_id,
        "created_at": datetime.utcnow(),
        "updated_at": datetime.utcnow()
    }

    result = mongo.db.students.insert_one(student_doc)
    return jsonify({"message": "Student added successfully", "id": str(result.inserted_id)}), 201

# Fetch all students
@student_bp.route('/', methods=['GET'])
```

```

def get_students():
    students = mongo.db.students.find()
    result = []

    for student in students:
        result.append({
            "id": str(student["_id"]),
            "full_name": student["full_name"],
            "roll_number": student["roll_number"],
            "class_name": student["class_name"],
            "date_of_birth": student["date_of_birth"],
            "guardian_contact": student["guardian_contact"],
            "photo_id": str(student["photo_id"]) if student.get("photo_id") else None,
            "created_at": student["created_at"],
            "updated_at": student["updated_at"]
        })

    return jsonify(result), 200

```

```

# 📸 View student photo
@student_bp.route('/photo/<photo_id>', methods=['GET'])
@token_required
def get_photo(photo_id):
    binary_data, mime, filename = get_image(photo_id)
    if binary_data:
        return send_file(io.BytesIO(binary_data), mimetype=mime or 'image/jpeg',
                        as_attachment=False, download_name=filename or 'photo.jpg')
    else:
        return jsonify({"error": "Photo not found"}), 404

```

```

# ❌ Delete student photo
@student_bp.route('/photo/<photo_id>', methods=['DELETE'])
@token_required
def delete_photo(photo_id):
    if delete_image(photo_id):
        return jsonify({"message": "Photo deleted successfully"}), 200
    else:
        return jsonify({"error": "Photo not found"}), 404

```

```

# ✎ Update student
@student_bp.route('/<student_id>', methods=['PUT'])
@token_required
def update_student(student_id):
    data = request.form
    photo = request.files.get('photo')

```

```

    required_fields = ("full_name", "roll_number", "class_name", "date_of_birth",
"guardian_contact")
    if not all(k in data for k in required_fields):
        return jsonify({"error": "Missing required fields"}), 400

student_doc = {
    "full_name": data['full_name'],
    "roll_number": data['roll_number'],
    "class_name": data['class_name'],
    "date_of_birth": data['date_of_birth'],
    "guardian_contact": data['guardian_contact'],
    "updated_at": datetime.utcnow()
}

if photo:
    photo_id = save_image(photo)
    student_doc["photo_id"] = photo_id

result = mongo.db.students.update_one({"_id": ObjectId(student_id)}, {"$set": student_doc})
if result.matched_count == 0:
    return jsonify({"error": "Student not found"}), 404

return jsonify({"message": "Student updated successfully"}), 200

# ✖ Delete student
@student_bp.route('/<student_id>', methods=['DELETE'])
@token_required
def delete_student(student_id):
    result = mongo.db.students.delete_one({"_id": ObjectId(student_id)})
    if result.deleted_count == 0:
        return jsonify({"error": "Student not found"}), 404
    return jsonify({"message": "Student deleted successfully"}), 200

# 📄 View student profile with scores
@student_bp.route('/<student_id>', methods=['GET'])
@token_required
def view_student(student_id):
    student = mongo.db.students.find_one({"_id": ObjectId(student_id)})
    if not student:
        return jsonify({"error": "Student not found"}), 404

    scores = list(mongo.db.scores.find({"student_id": ObjectId(student_id)}))
    student["_id"] = str(student["_id"]) # Needed for safe Jinja2 rendering
    return render_template('student_detail.html', student=student, scores=scores)

# 📈 View student scores separately

```

```

@student_bp.route('/<student_id>/scores', methods=['GET'])
@token_required
def view_student_scores(student_id):
    scores = list(mongo.db.scores.find({"student_id": ObjectId(student_id)}))
    if not scores:
        return jsonify({"message": "No scores found for this student"}), 404
    return render_template('student_scores.html', scores=scores)

# ✚ Add a score for student
@student_bp.route('/<student_id>/scores', methods=['POST'])
@token_required
def add_student_score(student_id):
    data = request.json
    if not all(k in data for k in ("subject", "score", "date")):
        return jsonify({"error": "Missing required fields"}), 400

    score_doc = {
        "student_id": ObjectId(student_id),
        "subject": data['subject'],
        "marks_obtained": data['score'],
        "total_marks": 100, # default or adjust via UI later
        "exam_date": data['date'],
        "created_at": datetime.utcnow(),
        "updated_at": datetime.utcnow()
    }

    result = mongo.db.scores.insert_one(score_doc)
    return jsonify({"message": "Score added successfully", "id": str(result.inserted_id)}),
201

# 🖍 Update a student's score
@student_bp.route('/<student_id>/scores/<score_id>', methods=['PUT'])
@token_required
def update_student_score(student_id, score_id):
    data = request.json
    if not all(k in data for k in ("subject", "score", "date")):
        return jsonify({"error": "Missing required fields"}), 400

    score_doc = {
        "subject": data['subject'],
        "marks_obtained": data['score'],
        "exam_date": data['date'],
        "updated_at": datetime.utcnow()
    }

    result = mongo.db.scores.update_one(
        {"_id": ObjectId(score_id), "student_id": ObjectId(student_id)},

```

```

        {"$set": score_doc}
    )

    if result.matched_count == 0:
        return jsonify({"error": "Score not found"}), 404

    return jsonify({"message": "Score updated successfully"}), 200

# ✘ Delete a student's score
@student_bp.route('/<student_id>/scores/<score_id>', methods=['DELETE'])
@token_required
def delete_student_score(student_id, score_id):
    result = mongo.db.scores.delete_one({"_id": ObjectId(score_id), "student_id": ObjectId(student_id)})
    if result.deleted_count == 0:
        return jsonify({"error": "Score not found"}), 404
    return jsonify({"message": "Score deleted successfully"}), 200

```

8. _navbar.html

```

<nav class="navbar navbar-expand-lg navbar-dark bg-dark mb-4">
    <div class="container-fluid">
        <a class="navbar-brand" href="/">SPT</a>
        <div class="collapse navbar-collapse">
            <ul class="navbar-nav me-auto">
                <li class="nav-item"><a class="nav-link" href="/dashboard">Dashboard</a></li>
                <li class="nav-item"><a class="nav-link" href="/students">Students</a></li>
                <li class="nav-item"><a class="nav-link" href="/scores/dashboard">Scores</a></li>
                <li class="nav-item"><a class="nav-link" href="/scores/analytics">Analytics</a></li>
            </ul>
            <button class="btn btn-outline-light" onclick="logout()">Logout</button>
        </div>
    </div>
</nav>

<script>
function logout() {
    localStorage.removeItem('token');
    window.location.href = '/login';
}
</script>

```

9. dashboard.html

```
{% include '_navbar.html' %}

<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Dashboard - Student Progress Tracker</title>
  <link rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
<style>
  body {
    padding-top: 20px;
  }
  .container {
    max-width: 800px;
  }
</style>
<script>
  if (!localStorage.getItem('token')) {
    window.location.href = '/login';
  }
</script>
<title>Welcome to Dashboard</title>
</head>
<body>
<div class="container">
  <h1 class="mt-4">Dashboard</h1>
  <p>Welcome to the Student Progress Tracker Dashboard!</p>
  {% comment %} <p>Use the navigation bar to access different sections.</p> {%
  endcomment %}
</div>
<div class="container">
  <h4 class="mb-3">Recent Student Scores</h4>
  <table class="table table-striped">
    <thead>
      <tr>
        <th>Student Name</th>
        <th>Subject</th>
        <th>Score (%)</th>
        <th>Date</th>
      </tr>
    </thead>
    <tbody id="recentScores">
```

```

        <!-- Recent scores will be populated here -->
    </tbody>
</table>
<div class="container">
    <h4 class="mb-3">Subject-wise Performance Overview</h4>
    <canvas id="subjectChart" width="600" height="300"></canvas>
</div>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script>
    async function loadChart() {
        const token = localStorage.getItem('token');
        const res = await fetch('/scores/analytics/subject-averages', {
            headers: { 'Authorization': 'Bearer ' + token }
        });
        const data = await res.json();
        const subjects = data.map(x => x.subject);
        const averages = data.map(x => x.average);

        new Chart(document.getElementById('subjectChart'), {
            type: 'bar',
            data: {
                labels: subjects,
                datasets: [{ label: 'Average %', data: averages, backgroundColor: 'steelblue' }]
            },
            options: { scales: { y: { beginAtZero: true, max: 100 } } }
        });
    }
    loadChart();
</script>
</body>
</html>

```

10. index.html

```

{% include '_navbar.html' %}
<!DOCTYPE html>
<html>
<head>
    <title>Student Progress Tracker</title>
    <link rel="stylesheet"
        href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body class="p-4">
    <div class="container text-center">
        <h2>Welcome to the Student Progress Tracker</h2>
        <p class="lead">Select a student to view reports or visit the dashboard.</p>

```

```

        <a href="/dashboard" class="btn btn-primary mt-3">Go to Dashboard</a>
    </div>
</body>
</html>

```

11. login.html

```

{% include '_navbar.html' %}

<!DOCTYPE html>
<html>
<head>
    <title>Login | SPT</title>
    <link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body class="p-5">
    <div class="container col-md-4">
        <h3 class="mb-4">Teacher Login</h3>
        <form id="loginForm">
            <input type="text" class="form-control mb-2" name="username"
placeholder="Username" required />
            <input type="password" class="form-control mb-2" name="password"
placeholder="Password" required />
            <button class="btn btn-primary w-100">Log In</button>
        </form>
        <div id="msg" class="mt-3 text-danger"></div>
    </div>

    <script>
        document.getElementById('loginForm').onsubmit = async (e) => {
            e.preventDefault();
            const data = Object.fromEntries(new FormData(e.target).entries());
            const res = await fetch('/login', {
                method: 'POST',
                headers: {'Content-Type': 'application/json'},
                body: JSON.stringify(data)
            });
            const result = await res.json();
            if (res.ok) {
                localStorage.setItem('token', result.token);
                window.location.href = '/dashboard'; // redirect to dashboard page
            } else {
                document.getElementById('msg').innerText = result.error || 'Login failed';
            }
        };
    </script>
</body>

```

```
</html>
```

12. student_detail.html

```
{% include '_navbar.html' %}  
<!DOCTYPE html>  
<html>  
  <head>  
    <title>{{ student.full_name }} - Progress Report</title>  
    <link rel="stylesheet"  
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" />  
  </head>  
  <body class="p-4">  
    <div class="container">  
      <h3>{{ student.full_name }}'s Progress Report</h3>  
      <p>  
        <strong>Class:</strong> {{ student.class_name }} &ampnbsp&ampnbsp <strong>Roll  
        No:</strong> {{ student.roll_number }}  
      </p>  
      <p>  
        <strong>Date of Birth:</strong> {{ student.date_of_birth }}  
      </p>  
  
      <hr />  
  
      <h5>Subject Scores</h5>  
      <table class="table table-bordered table-striped">  
        <thead>  
          <tr>  
            <th>Subject</th>  
            <th>Marks Obtained</th>  
            <th>Total Marks</th>  
            <th>Exam Date</th>  
          </tr>  
        </thead>  
        <tbody>  
          {% for score in scores %}  
            <tr>  
              <td>{{ score.subject }}</td>  
              <td>{{ score.marks_obtained }}</td>  
              <td>{{ score.total_marks }}</td>  
              <td>{{ score.exam_date }}</td>  
            </tr>  
          {% endfor %}  
        </tbody>  
      </table>
```

```

<div class="mt-4">
    <a href="/scores/report/{{ student._id }}" class="btn btn-outline-primary"
target="_blank">Download PDF Report</a>
</div>
</div>
</body>
</html>

```

13. auth_utils.py

```

# app/utils/auth_utils.py

from functools import wraps
from flask import request, jsonify
import jwt
import os
from dotenv import load_dotenv

load_dotenv()
SECRET_KEY = os.getenv("SECRET_KEY")

def token_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        token = None
        if 'Authorization' in request.headers:
            bearer = request.headers['Authorization']
            token = bearer.split(" ")[1] if " " in bearer else bearer

        if not token:
            return jsonify({'error': 'Token is missing!'}), 401

        try:
            data = jwt.decode(token, SECRET_KEY, algorithms=["HS256"])
            request.user_id = data['user_id']
        except jwt.ExpiredSignatureError:
            return jsonify({'error': 'Token expired!'}), 401
        except jwt.InvalidTokenError:
            return jsonify({'error': 'Invalid token!'}), 401

        return f(*args, **kwargs)
    return decorated

```

14. imge_handler.py

```
# app/utils/image_handler.py

from gridfs import GridFS
from bson.objectid import ObjectId
from flask import current_app

def save_image(file):
    fs = GridFS(current_app.mongo.db)
    return fs.put(file, filename=file.filename)

def get_image(photo_id):
    fs = GridFS(current_app.mongo.db)
    try:
        image = fs.get(ObjectId(photo_id))
        return image.read(), image.content_type, image.filename
    except:
        return None, None, None

def delete_image(photo_id):
    fs = GridFS(current_app.mongo.db)
    try:
        fs.delete(ObjectId(photo_id))
        return True
    except:
        return False
```

15. pdf_generator.py

```
# app/utils/pdf_generator.py

from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas
from io import BytesIO

def generate_student_report(student, scores):
    buffer = BytesIO()
    c = canvas.Canvas(buffer, pagesize=A4)
    width, height = A4

    # Header
```

```

c.setFont("Helvetica-Bold", 16)
c.drawString(50, height - 50, f"Progress Report: {student['full_name']}") 
c.setFont("Helvetica", 12)
c.drawString(50, height - 80, f"Class: {student['class_name']} | Roll No: {student['roll_number']}") 
c.drawString(50, height - 100, f"Date of Birth: {student['date_of_birth']}")

# Table Header
y = height - 140
c.setFont("Helvetica-Bold", 12)
c.drawString(50, y, "Subject")
c.drawString(200, y, "Marks Obtained")
c.drawString(350, y, "Total Marks")
c.drawString(470, y, "Exam Date")

# Table Rows
c.setFont("Helvetica", 12)
for score in scores:
    y -= 20
    c.drawString(50, y, score["subject"])
    c.drawString(200, y, str(score["marks_obtained"]))
    c.drawString(350, y, str(score["total_marks"]))
    c.drawString(470, y, score["exam_date"])

c.showPage()
c.save()
buffer.seek(0)
return buffer

```

16. __init__.py

```

# app/__init__.py

from flask import Flask
from flask_pymongo import PyMongo
from dotenv import load_dotenv
import os

load_dotenv()
SECRET_KEY = os.getenv("SECRET_KEY")

mongo = PyMongo()

def create_app():
    load_dotenv()

```

```

app = Flask(__name__)
app.config["MONGO_URI"] = os.getenv("MONGO_URI")

mongo.init_app(app)

# Blueprint registration (move these inside the function)
from app.routes.students import student_bp
from app.routes.scores import score_bp
from app.routes.main import main_bp
from app.routes.dashboard import dashboard_bp # Add this
from app.routes.auth import auth_bp

# Then register it:
@app.register_blueprint(auth_bp, url_prefix='/auth')
app.register_blueprint(auth_bp) # NO prefix
app.register_blueprint(dashboard_bp) # This line tells Flask about your dashboard
routes
app.register_blueprint(main_bp)
app.register_blueprint(score_bp, url_prefix='/scores')
app.register_blueprint(student_bp, url_prefix='/students')
return app

```

17. authGuard.js

```

// inside static/js/authGuard.js
document.addEventListener('DOMContentLoaded', function() {
    const authToken = localStorage.getItem('authToken');
    const loginButton = document.getElementById('loginButton');
    const logoutButton = document.getElementById('logoutButton');

    if (authToken) {
        // User is authenticated
        loginButton.style.display = 'none';
        logoutButton.style.display = 'block';
    } else {
        // User is not authenticated
        loginButton.style.display = 'block';
        logoutButton.style.display = 'none';
    }

    logoutButton.addEventListener('click', function() {
        localStorage.removeItem('authToken');
        window.location.href = '/login'; // Redirect to login page
    });
});

<script src="{{ url_for('static', filename='js/authGuard.js') }}></script>

```

18. subjectChart.js

```
//inside static/js/subjectChart.js

const token = localStorage.getItem('token');
fetch('/scores/analytics/subject-averages', {
  headers: { 'Authorization': 'Bearer ' + token }
})
.then(res => res.json())
.then(data => {
  const subjects = data.map(x => x.subject);
  const averages = data.map(x => x.average);
  new Chart(document.getElementById('subjectChart'), {
    type: 'bar',
    data: {
      labels: subjects,
      datasets: [{ label: 'Average %', data: averages, backgroundColor: 'steelblue' }]
    },
    options: { scales: { y: { beginAtZero: true, max: 100 } } }
  });
});
```

19. .env

```
MONGO_URI=mongodb://localhost:27017/spt_db
SECRET_KEY=d10952582a68ffc37c05a283d1715aedd5d1890256f73475d8b5972bd6d10608
```

20. config.py

```
SECRET_KEY="d10952582a68ffc37c05a283d1715aedd5d1890256f73475d8b5972bd6d10608"
```

21. requirements.txt

```
Flask
Flask-PyMongo
pymongo
python-dotenv
gunicorn
```

22. run.py

```
from app import create_app

app = create_app()

if __name__ == "__main__":
    app.run(debug=True)
```

What I got today

127.0.0.1:5000/dashboard

Suggested Sites Java Platform Stand... Hortonworks Sandb... Excelerate EC2 Dashboard Train/Test Split and... Getting Started Gui... https://www.career... wordpress cc

SPT Dashboard Students Scores Analytics Logout

Dashboard

Welcome to the Student Progress Tracker Dashboard!

Use the navigation bar to access different sections.

Recent Student Scores

Student Name	Subject	Score (%)	Date
--------------	---------	-----------	------

Subject-wise Performance Overview