



**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**B. Tech., Fall Semester 2021-2022**

**Academics Performance Tracker**

**Course Faculty**

**Batch Members**

**M. BRAVEEN**

1. MADHUMITHAA RP, 20BCE1648
2. DEEPAVARSHINI T, 20BCE1661
3. SUDARSUN S, 20BCE1699

## **Table of Contents**

<b>S. NO.</b>	<b>TITLE</b>	<b>PAGE NUMBER</b>
1	Abstract	3
2	Introduction	4-5
3	Requirement Gathering	6-8
4	Structure of the Project	9
5	Planning and Scheduling	10-11
6	UML Diagrams	12-14
7	Architecture	15
8	User Interface	16-40
9	Verification and Validation  1. Test Case Document  2. Automated Testing	41-42
10	Conclusion	43
11	References	44

# **Abstract**

Many students are highly irregular with their deadlines and due dates and fail to plan and prepare accordingly for their exams and hence they receive poor scores. Though one can track everything in the age-old paper-pen method, it is becoming uncommon and hard to maintain it for many subjects. This is very common among the students' community and when a survey was conducted to understand the problems that students face which keeps them away from staying organised, we discovered it was the absence of a planner. Hence the solution for the same was to build an academic performance tracker for the students which offers functionalities more than just a planner. It is a comprehensive tool containing to-do lists, and a performance summary page which gives detailed analysis of the student's performance in a comprehensive and target oriented manner with appropriate conclusion messages.

# Introduction

Inconsistency is a very prominent behaviour shown by students of all ages due to various reasons which tend to be all the way from insignificant to prominent causes. Students are irregular with their studies and we conducted a survey to find out the reason and formulated the solution. The survey was circulated among fellow students who shared their valuable feedback for the successful understanding of the reason. They were asked questions related to consistency, target setting, to-do lists, goal setting and actively achieving them. The responses helped us to derive the conclusion that the age-old method of pen and paper to keep track of all the events and scores is highly inconvenient for a large scale of subjects and exams. Also the internet tools available online are so parched and unorganised and are available individually making it harder to access them all.

Hence, the solution to this is to build an academic performance tracker website that does more than just track performance, it also gives detailed analysis of your performance with data representation in an infographic way to motivate us to perform better.

## 2.1 Purpose

The purpose of this document is to define the requirements for the Academic Performance Tracker app. The intended audience of this document includes the admin of the webpage, the content creators, and the end users of the Webapp. The first release of the Accademic Performance Tracker Webapp is projected to have the version number 1.0. This document is to describe its functionality. It is also intended to provide guidance to the requirements team, requirements analyst, design team, and other members of the developing organization.

## **2.2 Product Scope**

This webapp is being developed for students to track their academic performances. This web app helps students to analyze their level of performance in academics by comparing the present scores with the past achievements. It allows them to predict how hard they should focus on academics so that they can accomplish their targets effectively. It also allows them to track their performance across various activities like CAT exams, Digital Assignments, semester GPAs, etc. They can view their previous scores and search the particular content they want. This helps them to boast their average scores and maintain consistency in the academics so that they can achieve their goals in more effective manner.

## **2.3 Product Perspective**

Our Performance Tracker is a web-based system. This web-app performs basic operations like calculating average marks, storing each input for later use, comparing with previous history, providing inference based on the comparisons, etc. The system provides a secure environment for uploading and storing the academic information and privacy will be maintained for each student

# **Software Requirements Specification**

## **3.1. System Features**

### **3.1.1 Analysis report of scores:**

#### **3.1.1.1 Description and priority**

The user will enter the marks of a particular exam. This score is related to the targets set for the exam and a detailed analysis report of the same is shown to user.

#### **3.1.1.2 Stimulus/ Response Sequences**

The user enters the marks of an exam. This score is compared with the targets set for the particular subject and exam, compares with the previous exam, etc to provide a detailed report on the progress made by the user in terms of marks change and percentage change. The user is made to set new targets for the upcoming exams and provides the average change in performance in comparison to the previous exams.

#### **3.1.1.3 Functional Requirements:**

REQ-1: The data entered by the user will be collected in the database.

REQ-2: The marks are compared with the previous exam to find out the change in marks either increase or decrease.

REQ-3: The marks change is shown to the user.

REQ-4: If there is no marks change, then it shows the user a message “A little progress each day adds up to big results”

REQ-5: If there is no previous exam marks or target marks to compare with, then calculate the average to show the user.

### **1.1.2 To do list with reminder**

#### **1.1.2.1 Description and priority**

The user can add to do activities to the list with due date mentioned and a can setup an reminder to do the activity.

#### **1.1.2.2 Stimulus/ Response sequence**

The user has to add an activity to the to-do list and mention the due date. The user can then choose to set up a reminder to do the activity. The user can also prioritise the to do list.

#### **1.1.2.3 Functional Requirements:**

REQ-1: The user has to create a to do list.

REQ-2: The user has to enter the due date for the activity.

REQ-3: The user can choose to set up a reminder for activity.

## **3.2. Other Nonfunctional Requirements**

### **3.2.1 Performance Requirements**

- There should not have any delays to make the system interactive.
- There should be maximum of 2 secs delay in data storage and verification process.
- Uploading and opening of files, the performance of the application should be less than 2 seconds.
- The dis-connectivity or problem in connecting to the server should be resolved in less than 20 seconds, or the error notice should pop up to notify the user.
- Conversion of file types or uploading of larger files/documents can take up to 10seconds maximum.

### **3.2.2 Safety Requirements**

Each given by the student must be encrypted so that only if valid login credentials are entered, the data will be decrypted back and displayed normally.

### **3.2.3 Security Requirements**

Since the data provided by each student is confidential, login is the most necessary requirement. All the data stored in this tracker is allowed to be used by the respective student only. Also, no one is allowed to duplicate our project using the source code.. Since the information in the database is private, strict security implementations must be followed.

### **3.2.4 Software Quality Attributes**

#### **3.2.4.1 Reliability**

As the system provides the right tools for discussion, problem solving it must be made sure that the system is reliable in its operations and for securing the sensitive details.

#### **3.2.4.2 Availability**

If the internet service gets disrupted while sending information to the server, the information can be sent again for verification.

#### **3.2.4.3 Security**

The main security concern is for users account hence proper login mechanism should be used to avoid hacking. The student id registration is way to spam check for increasing the security. Hence, security is provided from unwanted use of recognition software.

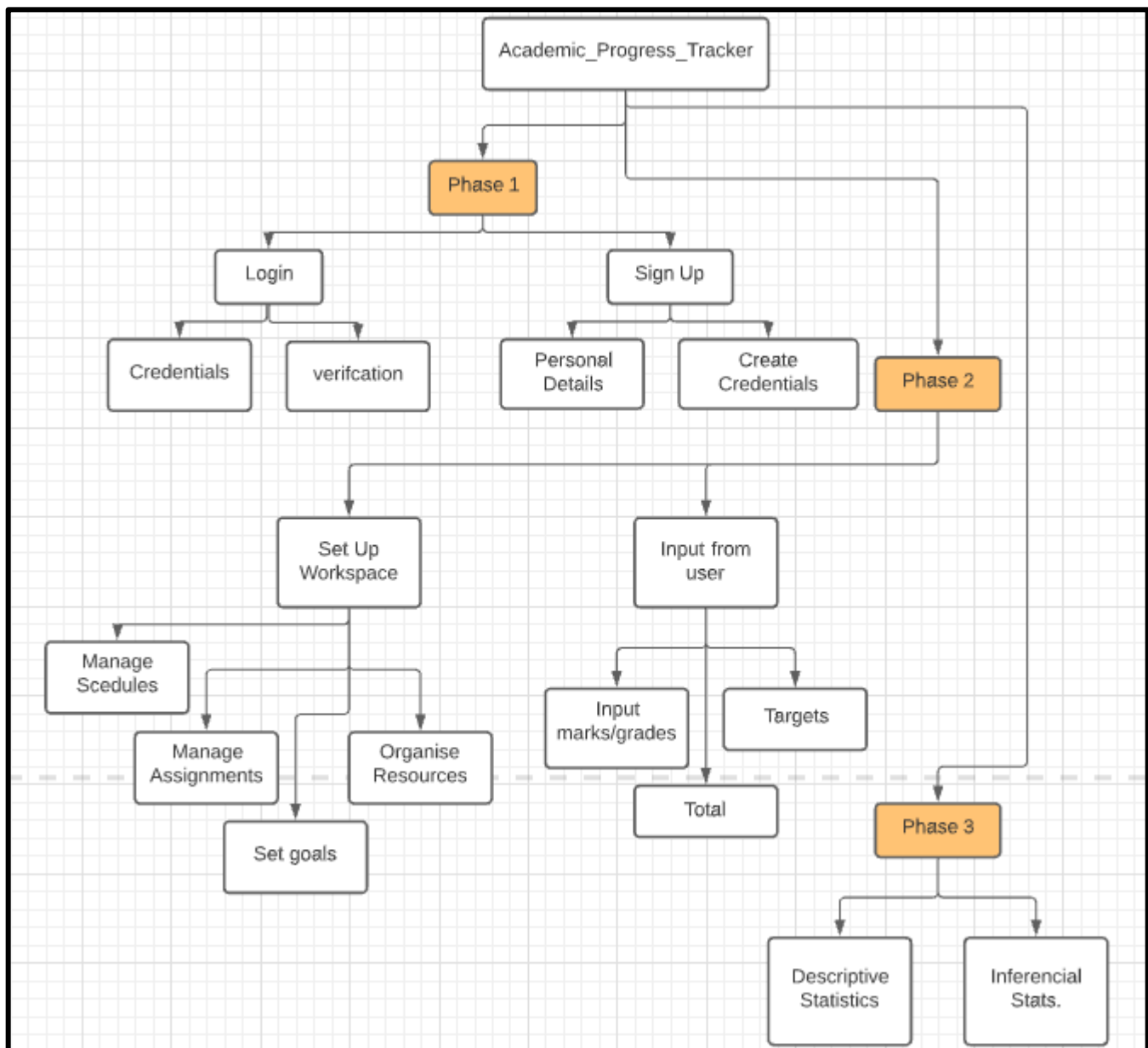
#### **3.2.4.4 Usability**

As the system is easy to handle and navigates in the most expected way with no delays. In that case the system program reacts accordingly and transverses quickly between its states.



# Structure of the Project

## WORK BREAKDOWN STRUCTURE:



# Planning and Scheduling

**XP** has separate iterations, even though both suggest small frequent releases and a high level of flexibility and adaptiveness to the changing requirements.

**Split Planning:** The planning and requirements analysis are driven by user stories. XP assumes that the customer does not have a clear picture of the finished product at the beginning of the project. The customer must participate from the initial stage of the project and provide regular feedback.

**Pair Programming:** Paired-Programming is one of the main distinguishing characteristics of XP. This practice requires two programmers to work jointly on the same code.

While the first developer focuses on writing, the other one reviews code, suggests improvements, and fixes mistakes along the way. Such teamwork results in high-quality software and faster knowledge sharing. This technique reduces unnecessary errors and programming work.

**TDD:** By providing a Test-driven development method, each phase of the development can be divided into several units and tested separately. Changes if any can be made at short notice, new user stories can be implemented, even during the later stage of the development. This results in a clear and comprehensible code, thus improving the Quality and Stability of the software.

**Planning:** Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

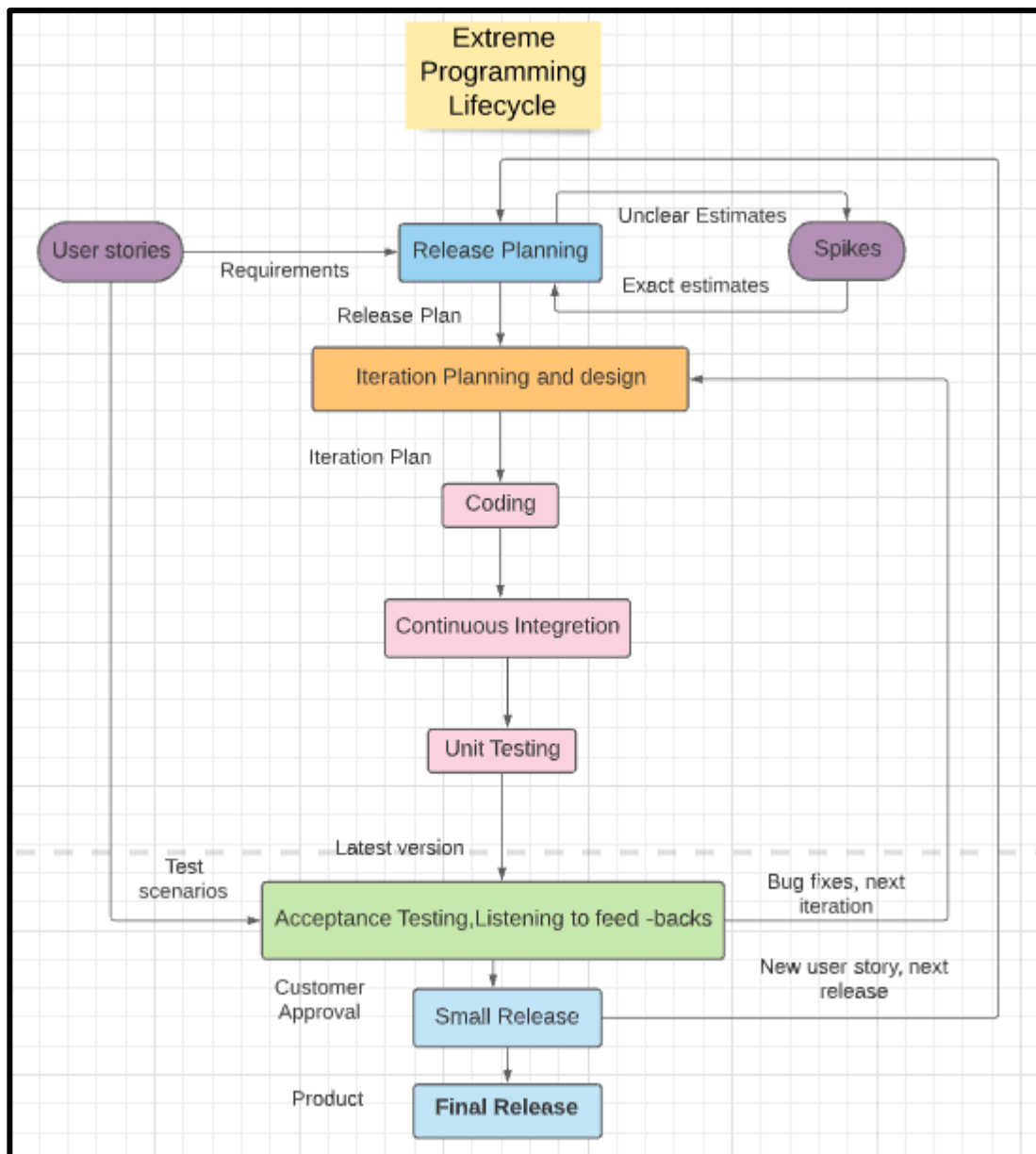
Django-excel is based on pyexcel and makes it easy to consume/produce information stored in excel files over HTTP protocol as well as on file system. This library can turn the excel data into a list of lists, a list of records(dictionaries), dictionaries of lists. And vice versa. Hence it lets you focus on data in Django based web development, instead of file formats.

matplotlib.pyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.

What is Matplotlib used for?

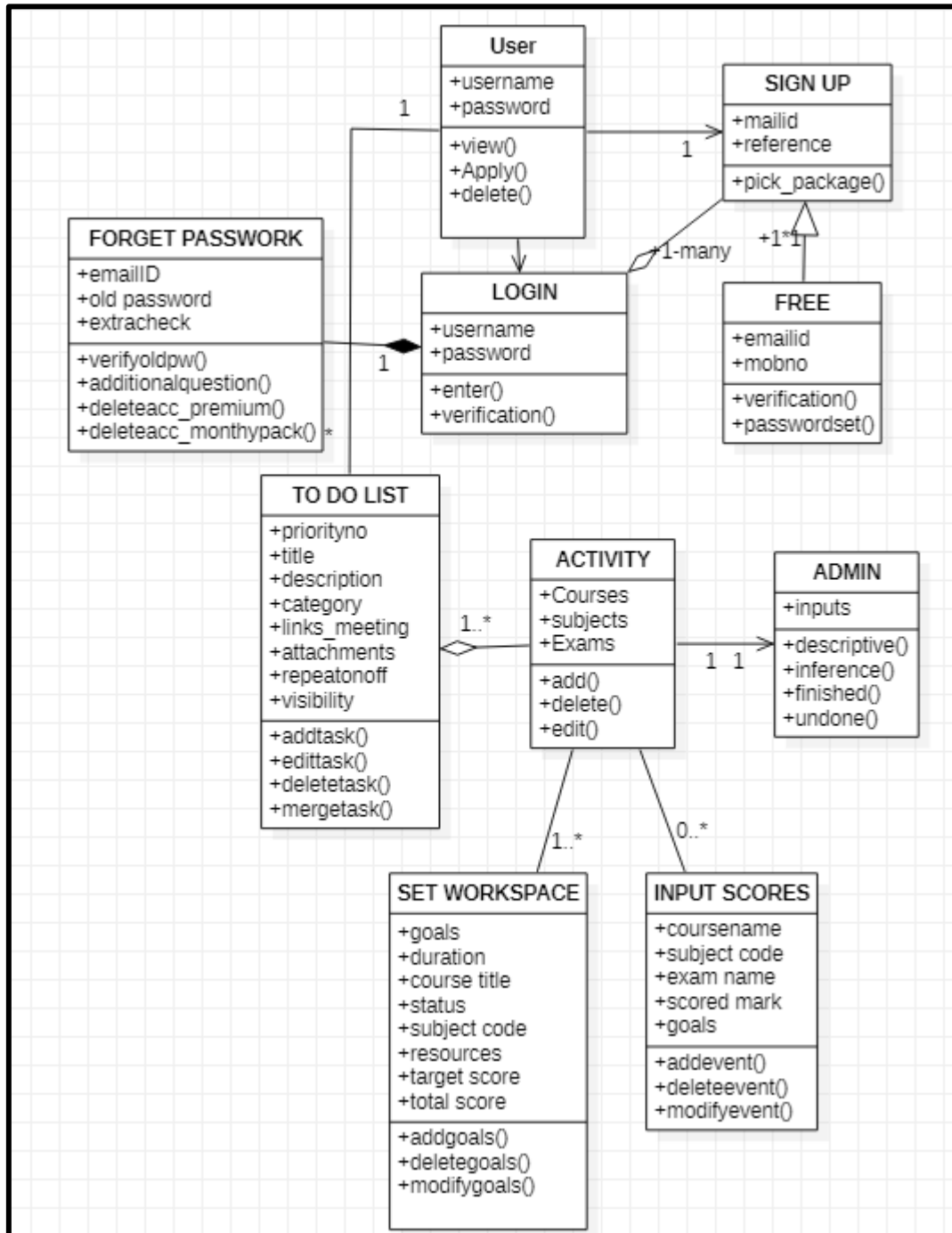
Matplotlib is a Python Library used for plotting, this python library provides and objected-oriented APIs for integrating plots into applications.

### Scheduling:

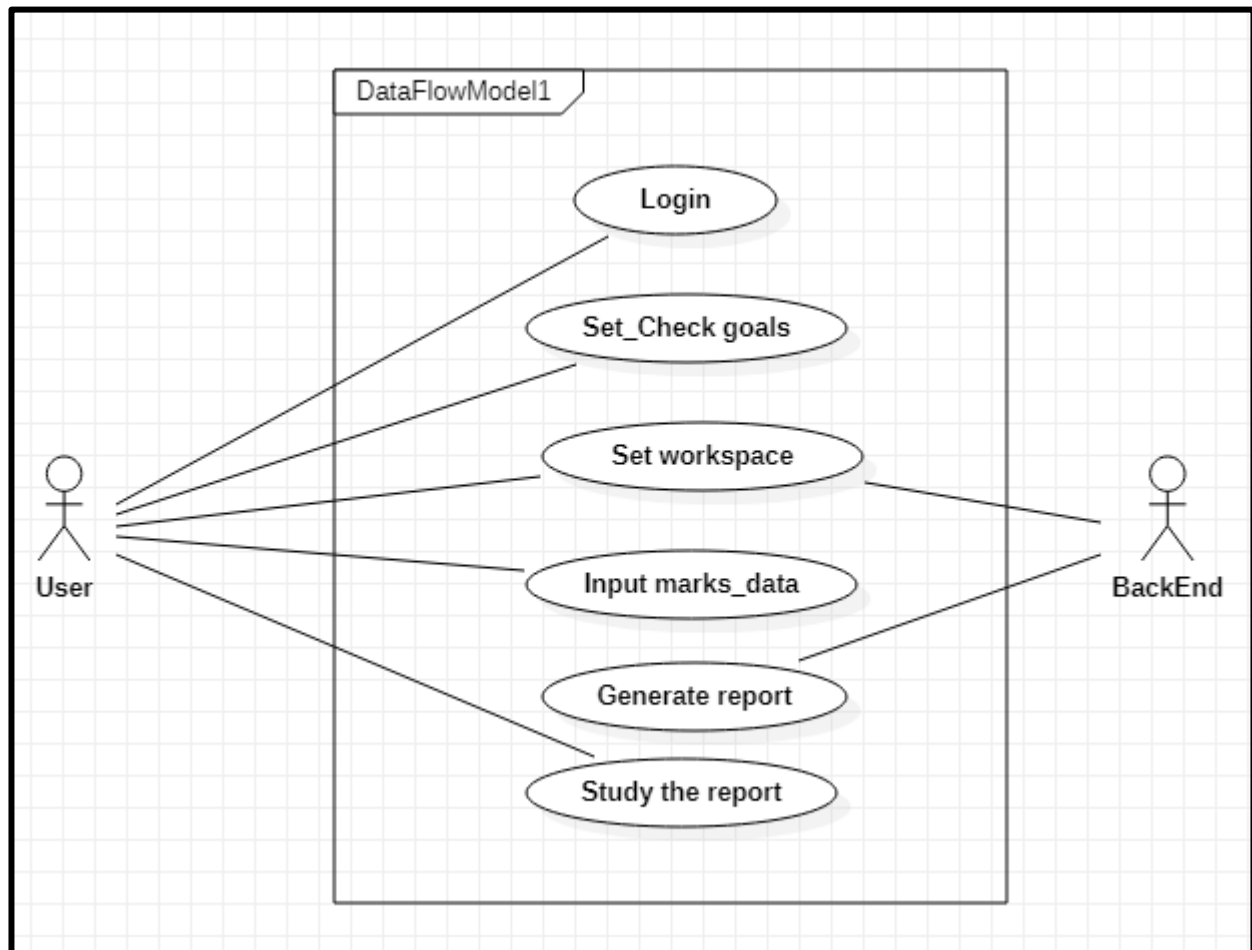


# UML Diagrams

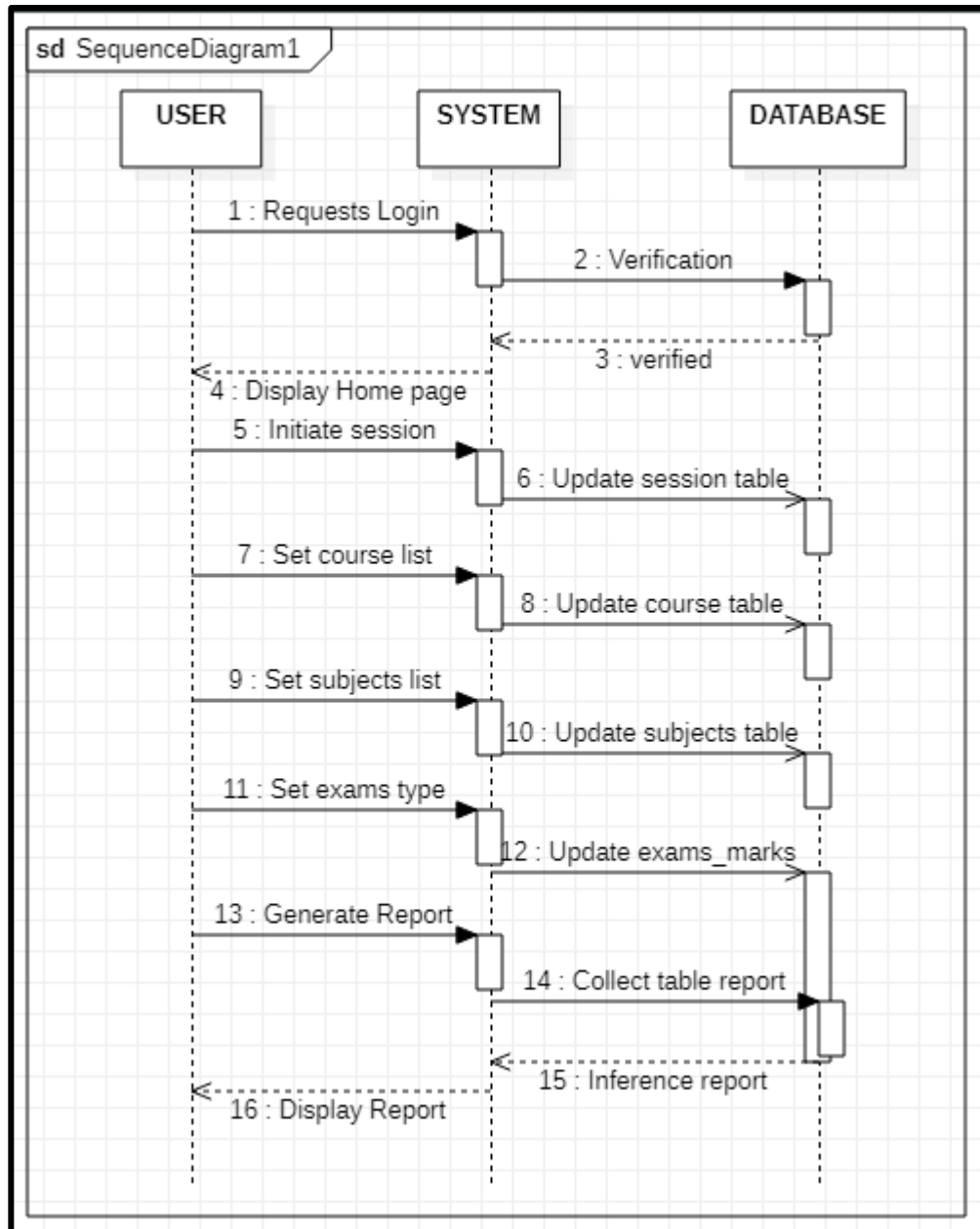
## CLASS DIAGRAM:



## USE CASE DIAGRAM:

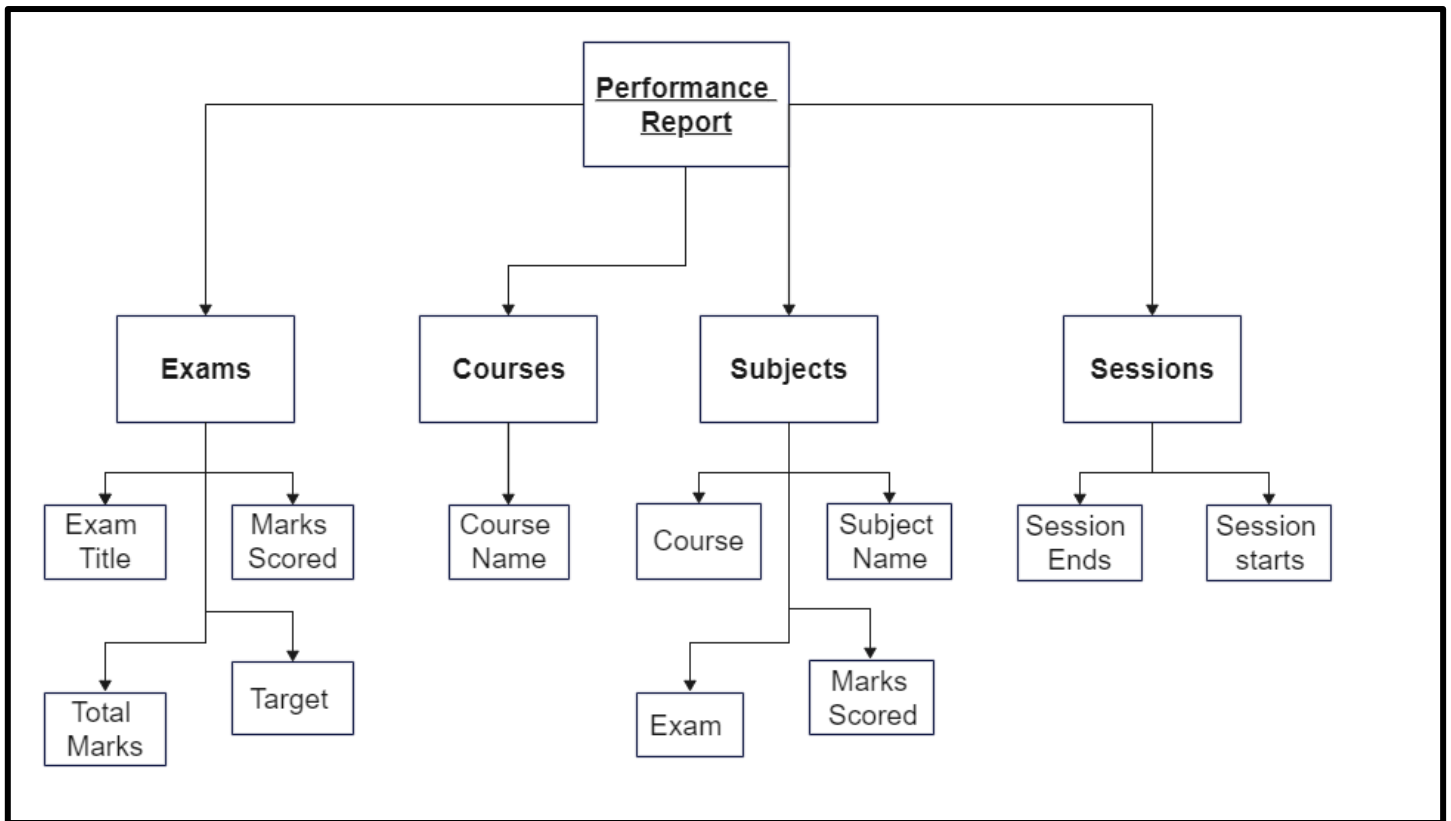


## SEQUENCE DIAGRAM:



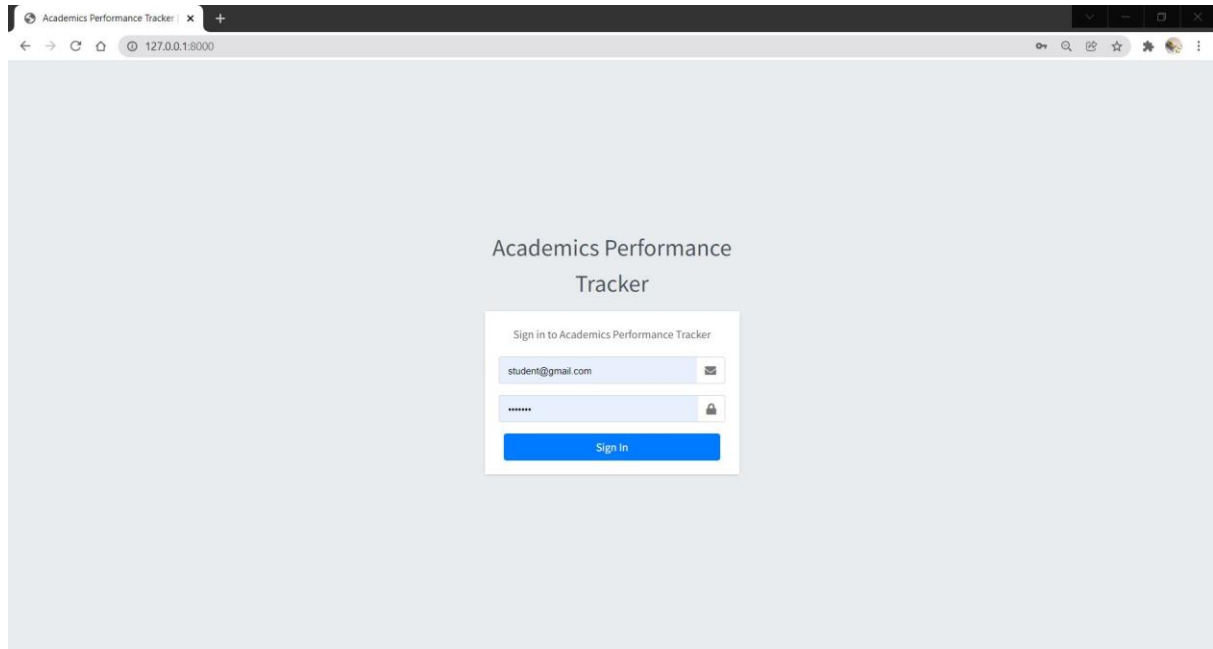
# Architecture

## Call and return method

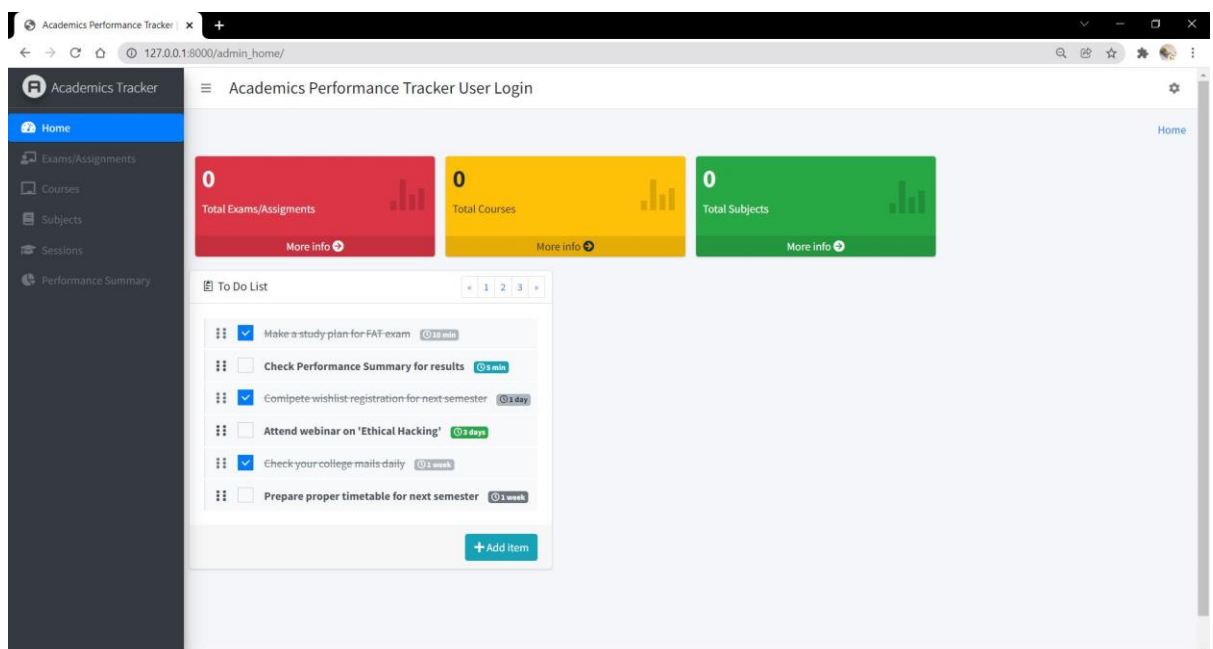


# Graphical User Interface

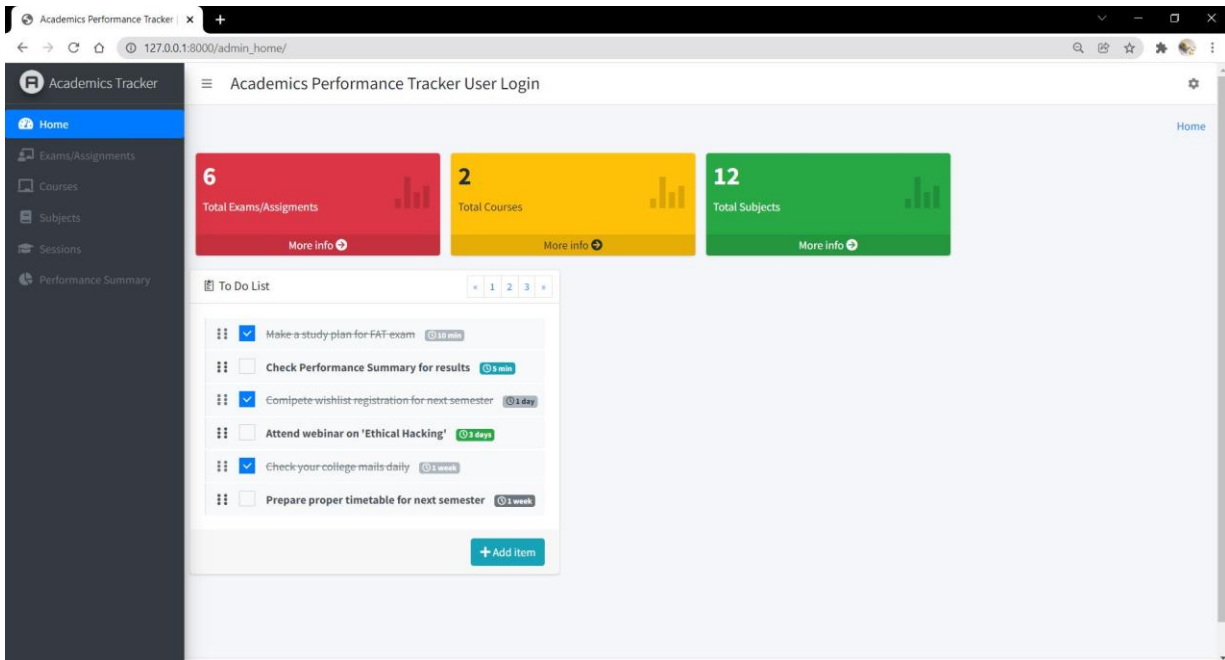
## LOGIN:



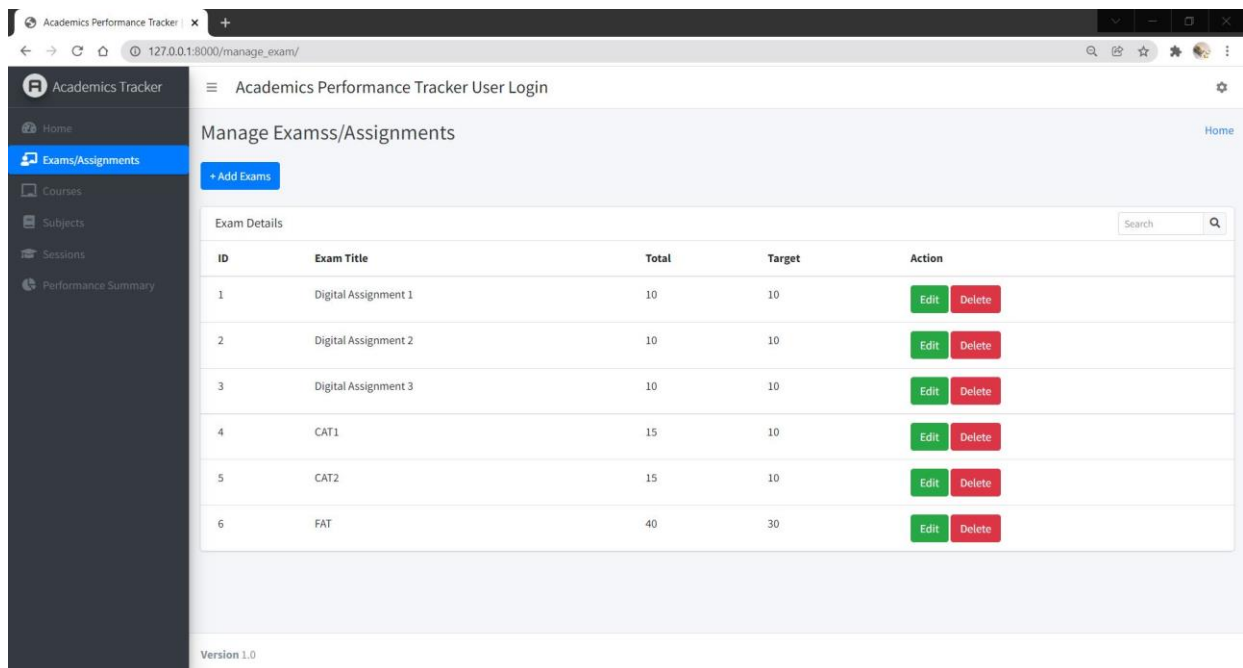
## HOME PAGE:







## EXAMS:



## COURSE:

Academics Performance Tracker | x

127.0.0.1:8000/manage\_course/

Academics Performance Tracker User Login

Manage Course

+ Add Course

Course Details

ID	Course Name	Created At	Updated At	Action
1	Semester 1	Dec. 3, 2021, noon	Dec. 3, 2021, noon	<a href="#">Edit</a> <a href="#">Delete</a>
2	Semester 2	Dec. 3, 2021, noon	Dec. 3, 2021, noon	<a href="#">Edit</a> <a href="#">Delete</a>

Version 1.0

## SUBJECTS:

Academics Performance Tracker | x

127.0.0.1:8000/manage\_subject/

Academics Performance Tracker User Login

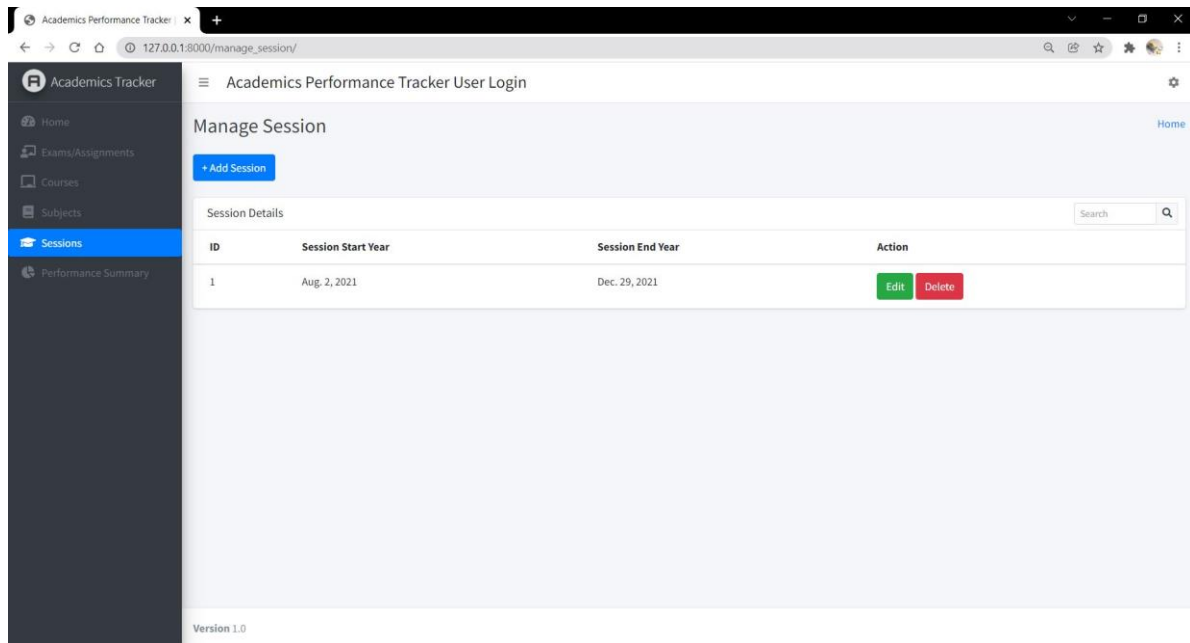
Manage Subjects

+ Add Subject

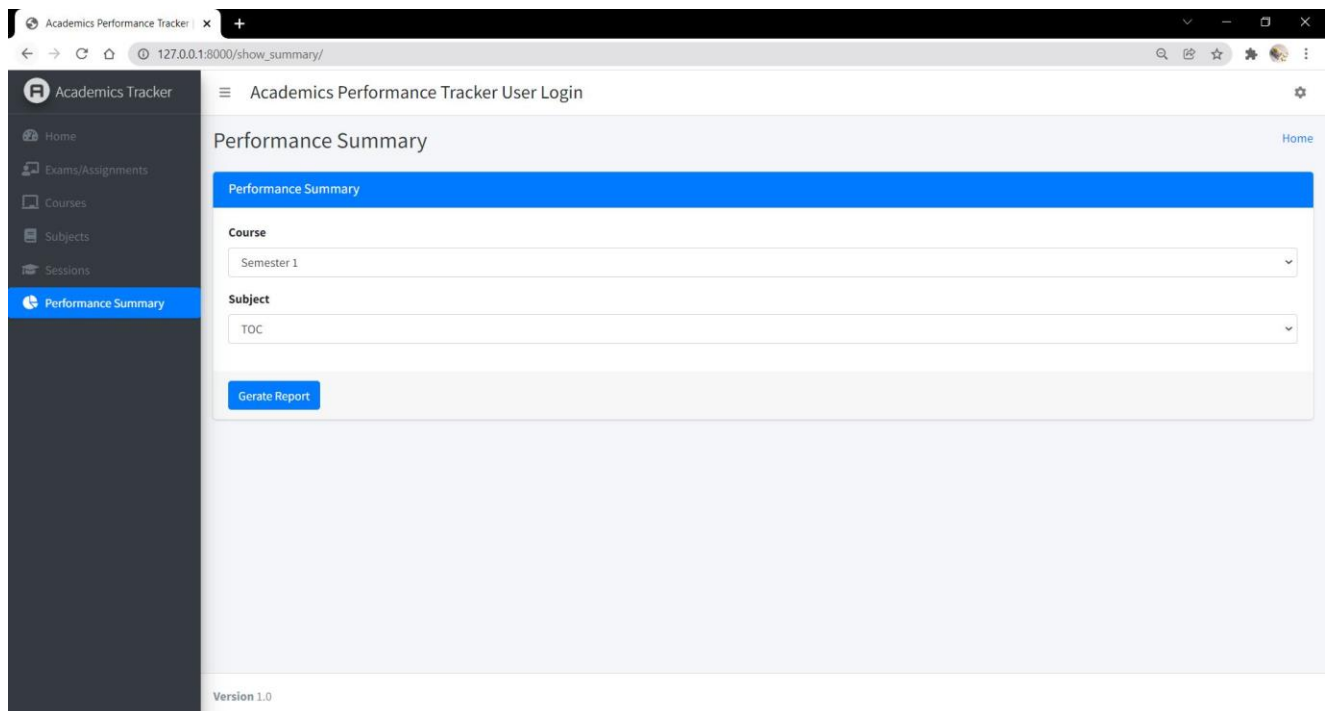
Subject Details

ID	Subject Name	Course	Exam	Score	Created At	Updated At	Action
1	TOC	Semester 1	Digital Assignment 1	10.0	Dec. 3, 2021, noon	Dec. 3, 2021, noon	<a href="#">Edit</a> <a href="#">Delete</a>
2	TOC	Semester 1	Digital Assignment 2	10.0	Dec. 3, 2021, 12:01 p.m.	Dec. 3, 2021, 12:01 p.m.	<a href="#">Edit</a> <a href="#">Delete</a>
3	TOC	Semester 1	Digital Assignment 3	10.0	Dec. 3, 2021, 12:01 p.m.	Dec. 3, 2021, 12:01 p.m.	<a href="#">Edit</a> <a href="#">Delete</a>
4	TOC	Semester 1	CAT1	5.0	Dec. 3, 2021, 12:01 p.m.	Dec. 3, 2021, 12:01 p.m.	<a href="#">Edit</a> <a href="#">Delete</a>
5	TOC	Semester 1	CAT2	8.0	Dec. 3, 2021, 12:01 p.m.	Dec. 3, 2021, 12:01 p.m.	<a href="#">Edit</a> <a href="#">Delete</a>
6	TOC	Semester 1	FAT	27.0	Dec. 3, 2021, 12:02 p.m.	Dec. 3, 2021, 12:02 p.m.	<a href="#">Edit</a> <a href="#">Delete</a>
7	SWE	Semester 2	Digital Assignment 1	10.0	Dec. 3, 2021, 12:02 p.m.	Dec. 3, 2021, 12:03 p.m.	<a href="#">Edit</a> <a href="#">Delete</a>
8	SWE	Semester 2	Digital Assignment 2	10.0	Dec. 3, 2021, 12:03 p.m.	Dec. 3, 2021, 12:03 p.m.	<a href="#">Edit</a> <a href="#">Delete</a>
9	SWE	Semester 2	Digital Assignment 3	10.0	Dec. 3, 2021, 12:04 p.m.	Dec. 3, 2021, 12:04 p.m.	<a href="#">Edit</a> <a href="#">Delete</a>

## SESSION:



## REPORT:



## GENERATED REPORT:

Academics Performance Tracker User Login

### Generated Report For TOC

Exam	Score Obtained	Total Score	Target Score
Digital Assignment 1	10.0	10	10
Digital Assignment 2	10.0	10	10
Digital Assignment 3	10.0	10	10
CAT1	5.0	15	10
CAT2	8.0	15	10
FAT	27.0	40	30

Expected Percentage	Scored Percentage
80.0	70.0

It's not the time to look for excuses, Keep trying hard.

[Back](#)

Academics Performance Tracker User Login

### Generated Report For SWE

Exam	Score Obtained	Total Score	Target Score
Digital Assignment 1	10.0	10	10
Digital Assignment 2	10.0	10	10
Digital Assignment 3	10.0	10	10
CAT1	14.0	15	10
CAT2	15.0	15	10
FAT	38.0	40	30

Expected Percentage	Scored Percentage
80.0	97.0

The tassel was worth the hassle. Congratulations.

[Back](#)

# Source Code






















## FILES:

Project APT				
Name	Date modified	Type	Size	
academics_performance_tracker	26-12-2020 01:14	File folder		
academics_tracker_app	23-11-2021 00:33	File folder		
static	26-12-2020 01:14	File folder		
.DS_Store	26-12-2020 01:14	DS_STORE File	11 KB	
db.sqlite3	26-12-2020 01:14	SQLITE3 File	284 KB	
manage	16-11-2021 13:40	Python File	1 KB	
README.md	26-12-2020 01:14	MD File	4 KB	
requirements	26-12-2020 01:14	Text Document	1 KB	

Project APT > academics_performance_tracker				
Name	Date modified	Type	Size	
__pycache__	23-11-2021 08:38	File folder		
__init__	26-12-2020 01:14	Python File	0 KB	
asgi	16-11-2021 11:33	Python File	1 KB	
settings	23-11-2021 08:38	Python File	4 KB	
urls	16-11-2021 11:29	Python File	1 KB	
wsgi	16-11-2021 11:30	Python File	1 KB	

Project APT > academics\_tracker\_app

Name	Date modified	Type	Size
__pycache__	03-12-2021 13:36	File folder	
migrations	03-12-2021 11:49	File folder	
templates	23-11-2021 17:49	File folder	
.DS_Store	26-12-2020 01:14	DS_STORE File	9 KB
__init__	18-11-2021 13:43	Python File	0 KB
admin	01-12-2021 17:58	Python File	1 KB
apps	03-12-2021 13:33	Python File	1 KB
EmailBackEnd	26-12-2020 01:14	Python File	1 KB
forms	16-11-2021 11:37	Python File	5 KB
HodViews	03-12-2021 09:46	Python File	16 KB
LoginCheckMiddleWare	22-11-2021 23:38	Python File	2 KB
models	03-12-2021 11:46	Python File	4 KB
tests	26-12-2020 01:14	Python File	1 KB
urls	03-12-2021 09:39	Python File	3 KB
views	23-11-2021 09:30	Python File	2 KB

Project APT > academics_tracker_app > templates > hod_template				
Name	Date modified	Type	Size	
 .DS_Store	26-12-2020 01:14	DS_STORE File	7 KB	
 add_course_template	23-11-2021 17:20	Chrome HTML Do...	4 KB	
 add_exam_template	23-11-2021 16:57	Chrome HTML Do...	4 KB	
 add_session_template	23-11-2021 17:28	Chrome HTML Do...	4 KB	
 add_subject_template	23-11-2021 17:25	Chrome HTML Do...	5 KB	
 admin_profile	26-12-2020 01:14	Chrome HTML Do...	5 KB	
 base_template	30-11-2021 22:26	Chrome HTML Do...	11 KB	
 edit_course_template	03-12-2021 12:16	Chrome HTML Do...	4 KB	
 edit_exam_template	23-11-2021 17:18	Chrome HTML Do...	4 KB	
 edit_session_template	23-11-2021 17:29	Chrome HTML Do...	4 KB	
 edit_subject_template	03-12-2021 12:29	Chrome HTML Do...	5 KB	
 footer	17-11-2021 12:25	Chrome HTML Do...	1 KB	
 form_template	26-12-2020 01:14	Chrome HTML Do...	3 KB	
 home_content	03-12-2021 10:52	Chrome HTML Do...	9 KB	
 manage_course_template	26-12-2020 01:14	Chrome HTML Do...	5 KB	
 manage_exam_template	23-11-2021 01:25	Chrome HTML Do...	5 KB	
 manage_session_template	26-12-2020 01:14	Chrome HTML Do...	5 KB	
 manage_subject_template	30-11-2021 18:22	Chrome HTML Do...	5 KB	
 report	03-12-2021 09:45	Chrome HTML Do...	5 KB	
 sidebar_template	30-11-2021 18:01	Chrome HTML Do...	4 KB	
 summary_template	02-12-2021 12:56	Chrome HTML Do...	5 KB	



## HTML & PYTHON CODES:

```
HodViews.py - C:\Users\Sudarsun\Desktop\Project APT\academics_tracker_app\HodViews.py (3.9.7)
File Edit Format Run Options Window Help
from django.shortcuts import render, redirect
from django.http import HttpResponse, HttpResponseRedirect, JsonResponse
from django.contrib import messages
from django.core.files.storage import FileSystemStorage #To upload Profile Picture
from django.urls import reverse
from django.views.decorators.csrf import csrf_exempt
from django.core import serializers
from django.db.models import Sum
import json
from academics_tracker app.models import CustomUser, exams, Courses, Subjects, SessionYearModel, Summary, Report
from .forms import AddStudentForm, EditStudentForm

def admin_home(request):
    subject_count = Subjects.objects.all().count()
    course_count = Courses.objects.all().count()
    exam_count = exams.objects.all().count()
    course_all = Courses.objects.all()
    course_name_list = []
    subject_count_list = []
    for course in course_all:
        subjects = Subjects.objects.filter(course_id=course.id).count()
        course_name_list.append(course.course_name)
        subject_count_list.append(subjects)
    subject_all = Subjects.objects.all()
    subject_list = []
    for subject in subject_all:
        course = Courses.objects.get(id=subject.course_id.id)
        subject_list.append(subject.subject_name)

    # For Exams
    exam_name_list=[]
    Exams = exams.objects.all()
    for exam in Exams:
        subject_ids = Subjects.objects.filter(exam_id=exam.id)
        exam_name_list.append(exam.exam_name)

    context={
        "subject_count": subject_count,
        "course_count": course_count,
        "exam_count": exam_count,
        "course_name_list": course_name_list,
        "subject_count_list": subject_count_list,
        "subject_list": subject_list,
        "exam_name_list": exam_name_list
    }
    return render(request, "hod_template/home_content.html", context)
```

Ln: 180 Col: 0



```
HodViews.py - C:\Users\Sudarsun\Desktop\Project APT\academics_tracker_app\HodViews.py (3.9.7)
File Edit Format Run Options Window Help
def add_exam(request):
    return render(request, "hod_template/add_exam_template.html")

def add_exam_save(request):
    if request.method != "POST":
        messages.error(request, "Invalid Method ")
        return redirect('add_exam')
    else:
        exam_name = request.POST.get('exam_name')
        total_score = request.POST.get('total_score')
        target_score = request.POST.get('target_score')
        exams_model = exams(exam_name=exam_name, total_score=total_score, target_score = target_score)
        exams_model.save()
        messages.success(request, "Exam Added Successfully!")
        return redirect('add_exam')

def manage_exam(request):
    Exams = exams.objects.all()
    context = {
        "exams": Exams
    }
    return render(request, "hod_template/manage_exam_template.html", context)

def edit_exam(request, exam_id):
    exam = exams.objects.get(id=exam_id)

    context = {
        "exam": exam,
        "id": exam_id
    }
    return render(request, "hod_template/edit_exam_template.html", context)

def edit_exam_save(request):
    if request.method != "POST":
        return HttpResponse("<h2>Method Not Allowed</h2>")
    else:
        exam_id=request.POST.get('exam_id')
        exam_name = request.POST.get('exam_name')
        total_score = request.POST.get('total_score')
        target_score = request.POST.get('target_score')
        try:
            exam_model = exams.objects.get(id=exam_id)
            exam_model.exam_name = exam_name
            exam_model.total_score = total_score
            exam_model.target_score = target_score
            exam_model.save()
            messages.success(request, "Exam Updated Successfully.")
```

Ln: 95 Col: 50

```
HodViews.py - C:\Users\Sudarsun\Desktop\Project APT\academics_tracker_app\HodViews.py (3.9.7)
File Edit Format Run Options Window Help

    messages.success(request, "Exam Updated Successfully.")
    return redirect('/edit_exam/'+exam_id)
except:
    messages.error(request, "Failed to Update Exam.")
    return redirect('/edit_exam/'+exam_id)

def delete_exam(request, exam_id):
    exam = exams.objects.get(id=exam_id)
    try:
        exam.delete()
        messages.success(request, "Exam Deleted Successfully.")
        return redirect('manage_exam')
    except:
        messages.error(request, "Failed to Delete Exam.")
        return redirect('manage_exam')

def add_course(request):
    return render(request, "hod_template/add_course_template.html")

def add_course_save(request):
    if request.method != "POST":
        messages.error(request, "Invalid Method!")
        return redirect('add_course')
    else:
        course = request.POST.get('course')
        try:
            course_model = Courses(course_name=course)
            course_model.save()
            messages.success(request, "Course Added Successfully!")
            return redirect('add_course')
        except:
            messages.error(request, "Failed to Add Course!")
            return redirect('add_course')

def manage_course(request):
    courses = Courses.objects.all()
    context = {
        "courses": courses
    }
    return render(request, 'hod_template/manage_course_template.html', context)

def edit_course(request, course_id):
    course = Courses.objects.get(id=course_id)
    context = {
        "course": course,
        "id": course_id
    }
```

Ln: 141 Col: 5

```
HodViews.py - C:\Users\Sudarsun\Desktop\Project APT\academics_tracker_app\HodViews.py (3.9.7)
File Edit Format Run Options Window Help

return render(request, 'hod_template/edit_course_template.html', context)

def edit_course_save(request):
    if request.method != "POST":
        HttpResponse("Invalid Method")
    else:
        course_id = request.POST.get('course_id')
        course_name = request.POST.get('course')

        try:
            course = Courses.objects.get(id=course_id)
            course.course_name = course_name
            course.save()
            messages.success(request, "Course Updated Successfully.")
            return redirect('/edit_course/'+course_id)

        except:
            messages.error(request, "Failed to Update Course.")
            return redirect('/edit_course/'+course_id)

def delete_course(request, course_id):
    course = Courses.objects.get(id=course_id)
    try:
        course.delete()
        messages.success(request, "Course Deleted Successfully.")
        return redirect('manage_course')
    except:
        messages.error(request, "Failed to Delete Course.")
        return redirect('manage_course')

def manage_session(request):
    session_years = SessionYearModel.objects.all()
    context = {
        "session_years": session_years
    }
    return render(request, "hod_template/manage_session_template.html", context)

def add_session(request):
    return render(request, "hod_template/add_session_template.html")

def add_session_save(request):
    if request.method != "POST":
        messages.error(request, "Invalid Method")
        return redirect('add_course')
    else:
        session_start_year = request.POST.get('session_start_year')
        session_end_year = request.POST.get('session_end_year')
```

Ln: 188 Col: 12

```
HodViews.py - C:\Users\Sudarsun\Desktop\Project APT\academics_tracker_app\HodViews.py (3.9.7)
File Edit Format Run Options Window Help
    session_end_year = request.POST.get('session_end_year')
    try:
        sessionyear = SessionYearModel(session_start_year=session_start_year, session_end_year=session_end_year)
        sessionyear.save()
        messages.success(request, "Session Year added Successfully!")
        return redirect("add_session")
    except:
        messages.error(request, "Failed to Add Session Year")
        return redirect("add_session")

def edit_session(request, session_id):
    session_year = SessionYearModel.objects.get(id=session_id)
    context = {
        "session_year": session_year
    }
    return render(request, "hod_template/edit_session_template.html", context)

def edit_session_save(request):
    if request.method != "POST":
        messages.error(request, "Invalid Method!")
        return redirect('manage_session')
    else:
        session_id = request.POST.get('session_id')
        session_start_year = request.POST.get('session_start_year')
        session_end_year = request.POST.get('session_end_year')

        try:
            session_year = SessionYearModel.objects.get(id=session_id)
            session_year.session_start_year = session_start_year
            session_year.session_end_year = session_end_year
            session_year.save()
            messages.success(request, "Session Year Updated Successfully.")
            return redirect('/edit_session/'+session_id)
        except:
            messages.error(request, "Failed to Update Session Year.")
            return redirect('/edit_session/'+session_id)

def delete_session(request, session_id):
    session = SessionYearModel.objects.get(id=session_id)
    try:
        session.delete()
        messages.success(request, "Session Deleted Successfully.")
        return redirect('manage_session')
    except:
        messages.error(request, "Failed to Delete Session.")
        return redirect('manage_session')

Ln: 217 Col: 31
```

```
HodViews.py - C:\Users\Sudarsun\Desktop\Project APT\academics_tracker_app\HodViews.py (3.9.7)
File Edit Format Run Options Window Help
def add_subject(request):
    courses = Courses.objects.all()
    Exams = exams.objects.all()
    context = {
        "courses": courses,
        "exams": Exams
    }
    return render(request, 'hod_template/add_subject_template.html', context)

def add_subject_save(request):
    if request.method != "POST":
        messages.error(request, "Method Not Allowed!")
        return redirect('add_subject')
    else:
        course_id = request.POST.get('course')
        course = Courses.objects.get(id=course_id)
        exam_id = request.POST.get('exam')
        exam = exams.objects.get(id=exam_id)
        subject_name = request.POST.get('subject')
        marks_scored = request.POST.get('marks')
        try:
            subject = Subjects(subject_name=subject_name, course_id=course, exam_id=exam, marks_scored=marks_scored)
            subject.save()
            messages.success(request, "Subject Added Successfully!")
            return redirect('add_subject')
        except:
            messages.error(request, "Failed to Add Subject")
            return redirect("add_subject")

def manage_subject(request):
    subjects = Subjects.objects.all()
    context = {
        "subjects": subjects
    }
    return render(request, 'hod_template/manage_subject_template.html', context)

def edit_subject(request, subject_id):
    subject = Subjects.objects.get(id=subject_id)
    courses = Courses.objects.all()
    Exams = exams.objects.all()
    context = {
        "subject": subject,
        "courses": courses,
        "exams": Exams,
        "id": subject_id,
    }
    return render(request, 'hod_template/edit_subject_template.html', context)
```

Ln: 281 Col: 0



```
HodViews.py - C:\Users\Sudarsun\Desktop\Project APT\academics_tracker_app\HodViews.py (3.9.7)
File Edit Format Run Options Window Help
def edit_subject_save(request):
    if request.method != "POST":
        HttpResponse("Invalid Method.")
    else:
        subject_id = request.POST.get('subject_id')
        subject_name = request.POST.get('subject')
        course_id = request.POST.get('course')
        exam_id = request.POST.get('exam')
        marks_scored=request.POST.get('marks')
        try:
            subject = Subjects.objects.get(id=subject_id)
            subject.subject_name = subject_name
            subject.marks_scored = marks_scored
            course = Courses.objects.get(id=course_id)
            subject.course_id = course
            exam = exams.objects.get(id=exam_id)
            subject.exam_id = exam
            subject.save()
            messages.success(request, "Subject Updated Successfully.")
            # return redirect('/edit_subject/'+subject_id)
            return HttpResponseRedirect(reverse("edit_subject", kwargs={"subject_id":subject_id}))
        except:
            messages.error(request, "Failed to Update Subject.")
            return HttpResponseRedirect(reverse("edit_subject", kwargs={"subject_id":subject_id}))
            # return redirect('/edit_subject/'+subject_id)

def delete_subject(request, subject_id):
    subject = Subjects.objects.get(id=subject_id)
    try:
        subject.delete()
        messages.success(request, "Subject Deleted Successfully.")
        return redirect('manage_subject')
    except:
        messages.error(request, "Failed to Delete Subject.")
        return redirect('manage_subject')

def show_summary(request):
    Exams = exams.objects.all()
    courses = Courses.objects.all()
    subjects = Subjects.objects.all()
    summary = Summary.objects.all()
    report=Report.objects.all()
    context = {
        "exams": Exams,
        "courses": courses,
        "subjects": subjects,
        "summary": summary,
```

Ln: 329 Col: 24

```
HodViews.py - C:\Users\Sudarsun\Desktop\Project APT\academics_tracker_app\HodViews.py (3.9.7)
File Edit Format Run Options Window Help
    "subjects": subjects,
    "summary": summary,
    "report": report
}
return render(request, 'hod_template/summary_template.html', context)

def summary_content(request):
    if request.method != "POST":
        return redirect('show_summary')
    else:
        Summary.objects.all().delete()
        Report.objects.all().delete()
        course_id = request.POST.get('course')
        course = Courses.objects.get(id=course_id)
        subject_id = request.POST.get('subject')
        subject = Subjects.objects.get(id=subject_id)

        total = exams.objects.aggregate(Sum('total_score'))
        target = exams.objects.aggregate(Sum('target_score'))
        exp=(list(target.values())[0])/(list(total.values())[0])*100
        scored = Subjects.objects.filter(subject_name=subject.subject_name).aggregate(marks_scored=Sum('marks_scored'))
        obt=(list(scored.values())[0])/(list(total.values())[0])*100
        diff=obt-exp
        if diff<0:
            out='Improvement needed'
        elif diff==0:
            out='Target Achieved'
        else:
            out='Did Better'

        try:
            summary = Summary(course_id=course, subject_id=subject)
            summary.save()
            report = Report(expected_percent=exp,scored_percent=obt,difference=diff,outcome=out)
            report.save()
            return redirect('show_report')

        except:
            messages.error(request, "Failed to Generate Report")
            return redirect("show_summary")

def show_report(request):
    Exams = exams.objects.all()
    courses = Courses.objects.all()
    subjects = Subjects.objects.all()
    summary = Summary.objects.all()
    report=Report.objects.all()
```

Ln: 374 Col: 15

```
HodViews.py - C:\Users\Sudarsun\Desktop\Project APT\academics_tracker_app\HodViews.py (3.9.7)
File Edit Format Run Options Window Help

def show_report(request):
    Exams = exams.objects.all()
    courses = Courses.objects.all()
    subjects = Subjects.objects.all()
    summary = Summary.objects.all()
    report=Report.objects.all()
    context = {
        "exams": Exams,
        "courses": courses,
        "subjects": subjects,
        "summary": summary,
        "report": report
    }
    return render(request, 'hod_template/report.html',context)

@csrf_exempt
def check_email_exist(request):
    email = request.POST.get("email")
    user_obj = CustomUser.objects.filter(email=email).exists()
    if user_obj:
        return HttpResponse(True)
    else:
        return HttpResponse(False)

@csrf_exempt
def check_username_exist(request):
    username = request.POST.get("username")
    user_obj = CustomUser.objects.filter(username=username).exists()
    if user_obj:
        return HttpResponse(True)
    else:
        return HttpResponse(False)

def admin_profile(request):
    user = CustomUser.objects.get(id=request.user.id)

    context={
        "user": user
    }
    return render(request, 'hod_template/admin_profile.html', context)

def admin_profile_update(request):
    if request.method != "POST":
        messages.error(request, "Invalid Method!")
        return redirect('admin_profile')
    else:
```

Ln: 414 Col: 9



```
HodViews.py - C:\Users\Sudarsun\Desktop\Project APT\academics_tracker_app\HodViews.py (3.9.7)
File Edit Format Run Options Window Help
def check_email_exist(request):
    email = request.POST.get("email")
    user_obj = CustomUser.objects.filter(email=email).exists()
    if user_obj:
        return HttpResponse(True)
    else:
        return HttpResponse(False)

@csrf_exempt
def check_username_exist(request):
    username = request.POST.get("username")
    user_obj = CustomUser.objects.filter(username=username).exists()
    if user_obj:
        return HttpResponse(True)
    else:
        return HttpResponse(False)

def admin_profile(request):
    user = CustomUser.objects.get(id=request.user.id)

    context={
        "user": user
    }
    return render(request, 'hod_template/admin_profile.html', context)

def admin_profile_update(request):
    if request.method != "POST":
        messages.error(request, "Invalid Method!")
        return redirect('admin_profile')
    else:
        first_name = request.POST.get('first_name')
        last_name = request.POST.get('last_name')
        password = request.POST.get('password')

        try:
            customuser = CustomUser.objects.get(id=request.user.id)
            customuser.first_name = first_name
            customuser.last_name = last_name
            if password != None and password != "":
                customuser.set_password(password)
            customuser.save()
            messages.success(request, "Profile Updated Successfully")
            return redirect('admin_profile')
        except:
            messages.error(request, "Failed to Update Profile")
            return redirect('admin_profile')
```

Ln: 430 Col: 0

## Report Generation Code:

```
report - Notepad
File Edit Format View Help
{% extends 'hod_template/base_template.html' %}
{% block page_title %}
{% for summ in summary %}
    Generated Report For {{summ.subject_id.subject_name}}
{% endfor %}
{% endblock page_title %}
{% block main_content %}
{% load static %}
<section class="content">
    <div class="container-fluid">
        <div class="row">
            <div class="col-md-12">
                <div class="card">
                    <div class="card-header">
<!-- /.card-header -->
                <div class="card-body table-responsive p-0">
                    <table class="table table-hover text-nowrap">
                        <thead>
                            <tr>
                                <th>Exam</th>
                                <th>Score Obtained</th>
                                <th>Total Score</th>
                                <th>Target Score</th>
                            </tr>
                        </thead>
                        <tbody>
                            {% for summ in summary %}
                                {% for subject in subjects %}
                                    {% if subject.subject_name == summ.subject_id.subject_name %}
                                        <tr>
                                            <td>{{ subject.exam_id.exam_name }}</td>
                                            <td>{{ subject.marks_scored }}</td>
                                            <td>{{ subject.exam_id.total_score }}</td>
                                            <td>{{ subject.exam_id.target_score }}</td>
                                        </tr>
                                    {% endif %}
                                {% endfor %}
                            {% endfor %}
                        </tbody>
                    </table>
                </div>
                <!-- /.card-body -->
            </div>
        </div>
    </div>
</div>
</section>
```

```
report - Notepad
File Edit Format View Help

        </td>
        </tr>
        {% endif %}
        {% endfor %}
        {% endfor %}
    </tbody>
</table>
</div>
<!-- /.card-body -->
</div>
<!-- /.card -->
</div>
</div><!-- /.container-fluid -->
</section>

<section class="content">
<div class="container-fluid">
<div class="row">
<div class="col-md-12">
<div class="card">
<div class="card-header">

<!-- /.card-header -->
<div class="card-body table-responsive p-0">
<table class="table table-hover text-nowrap">
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>Expected Percentage</th>
<th>Scored Percentage</th>
</tr>
</thead>
<tbody>
{% for rep in report %}
<tr>
<td></td>
<td></td>
<td></td>
<td>{{ rep.expected_percent }}</td>
<td>{{ rep.scored_percent }}</td>
</tr>
{% endfor %}
</tbody>
</table>
</div>
Ln 83, Col 7    90%    Windows (CRLF)    UTF-8
```

```
report - Notepad
File Edit Format View Help
<!-- /.card-header -->
<div class="card-body table-responsive p-0">
  <table class="table table-hover text-nowrap">
    <thead>
      <tr>
        <th></th>
        <th></th>
        <th></th>
        <th>Expected Percentage</th>
        <th>Scored Percentage</th>
      </tr>
    </thead>
    <tbody>
      {% for rep in report %}
      <tr>
        <td></td>
        <td></td>
        <td></td>
        <td>{{ rep.expected_percent }}</td>
        <td>{{ rep.scored_percent }}</td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
</div>
<br>
{% for rep in report %}
{% if rep.outcome == "Improvement needed" %}
<h3 style="background-color:LightCoral; font-family:Baskerville Old Face; text-align:center; font-size:200%;">
It's not the time to look for excuses, Keep trying hard. </h3>
{% elif rep.outcome == "Did Better" %}
<h3 style="background-color:#90ee90; font-family:Baskerville Old Face; text-align:center; font-size:200%;">
The tassel was worth the hassle. Congratulations. </h3>
{% endif %}
{% endfor %}
<br>
<a class="btn btn-primary" href="{% url 'summary_content' %}" role="button">Back</a> <br>&nbsp;
<!-- /.card-body -->
</div>
<!-- /.card -->
</div>
</div><!-- /.container-fluid -->
</section>
{% endblock main_content %}
```

## DATABASE:

### Set of tables created:

```
mysql> use academics_performance_tracker;
Database changed
mysql> show tables;
+-----+
| Tables_in_academics_performance_tracker |
+-----+
| academics_tracker_app_adminhod          |
| academics_tracker_app_courses           |
| academics_tracker_app_customuser        |
| academics_tracker_app_customuser_groups |
| academics_tracker_app_customuser_user_permissions |
| academics_tracker_app_exams             |
| academics_tracker_app_report            |
| academics_tracker_app_sessionyearmodel  |
| academics_tracker_app_subjects          |
| academics_tracker_app_summary           |
| auth_group                              |
| auth_group_permissions                  |
| auth_permission                         |
| django_admin_log                       |
| django_content_type                    |
| django_migrations                      |
| django_session                         |
+-----+
17 rows in set (0.00 sec)
```

```
mysql> select * from academics_tracker_app_exams;
+-----+-----+-----+-----+-----+-----+
| id | created_at | updated_at | exam_name | target_score | total_score |
+-----+-----+-----+-----+-----+-----+
| 1 | 2021-12-03 06:27:30.993995 | 2021-12-03 06:41:16.929267 | Digital Assignment 1 | 10 | 10 |
| 2 | 2021-12-03 06:28:37.062510 | 2021-12-03 06:28:37.062510 | Digital Assignment 2 | 10 | 10 |
| 3 | 2021-12-03 06:28:46.466006 | 2021-12-03 06:28:46.466006 | Digital Assignment 3 | 10 | 10 |
| 4 | 2021-12-03 06:28:55.763097 | 2021-12-03 06:28:55.763097 | CAT1 | 10 | 15 |
| 5 | 2021-12-03 06:29:04.958766 | 2021-12-03 06:29:04.958766 | CAT2 | 10 | 15 |
| 6 | 2021-12-03 06:29:24.120932 | 2021-12-03 06:29:24.120932 | FAT | 30 | 40 |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```



```
mysql> select * from academics_tracker_app_exams;
```

id	created_at	updated_at	exam_name	target_score	total_score
1	2021-12-03 06:27:30.993995	2021-12-03 06:41:16.929267	Digital Assignment 1	10	10
2	2021-12-03 06:28:37.062510	2021-12-03 06:28:37.062510	Digital Assignment 2	10	10
3	2021-12-03 06:28:46.466006	2021-12-03 06:28:46.466006	Digital Assignment 3	10	10
4	2021-12-03 06:28:55.763097	2021-12-03 06:28:55.763097	CAT1	10	15
5	2021-12-03 06:29:04.958766	2021-12-03 06:29:04.958766	CAT2	10	15
6	2021-12-03 06:29:24.120932	2021-12-03 06:29:24.120932	FAT	30	40

```
6 rows in set (0.00 sec)
```

```
mysql> select * from academics_tracker_app_courses;
```

id	course_name	created_at	updated_at
1	Semester 1	2021-12-03 06:30:15.098110	2021-12-03 06:46:20.242192
2	Semester 2	2021-12-03 06:30:19.541727	2021-12-03 06:30:19.541727

```
2 rows in set (0.00 sec)
```

```
mysql> select * from academics_tracker_app_sessionyearmodel;
```

id	session_start_year	session_end_year
1	2021-08-02	2021-12-29

```
1 row in set (0.00 sec)
```

```
mysql> select * from academics_tracker_app_subjects;
```

id	subject_name	created_at	updated_at	course_id_id	exam_id_id	marks_scored
1	TOC	2021-12-03 06:30:51.004929	2021-12-03 06:59:20.819276	1	1	10
2	TOC	2021-12-03 06:31:02.311143	2021-12-03 06:31:02.311143	1	2	10
3	TOC	2021-12-03 06:31:14.541063	2021-12-03 06:31:14.541063	1	3	10
4	TOC	2021-12-03 06:31:35.505141	2021-12-03 06:31:35.505141	1	4	5
5	TOC	2021-12-03 06:31:54.931109	2021-12-03 06:31:54.931109	1	5	8
6	TOC	2021-12-03 06:32:19.677056	2021-12-03 06:32:19.677056	1	6	27
7	SWE	2021-12-03 06:32:59.939590	2021-12-03 06:33:19.087161	2	1	10
8	SWE	2021-12-03 06:33:34.978830	2021-12-03 06:33:34.978904	2	2	10
9	SWE	2021-12-03 06:34:31.907892	2021-12-03 06:34:31.907892	2	3	10
10	SWE	2021-12-03 06:35:22.382827	2021-12-03 06:35:22.385346	2	4	14
11	SWE	2021-12-03 06:36:09.510248	2021-12-03 06:36:09.510248	2	5	15
12	SWE	2021-12-03 06:36:33.332229	2021-12-03 06:36:33.332229	2	6	38

```
12 rows in set (0.00 sec)
```

```
mysql> select * from academics_tracker_app_summary;
```

id	course_id_id	subject_id_id
2	2	7

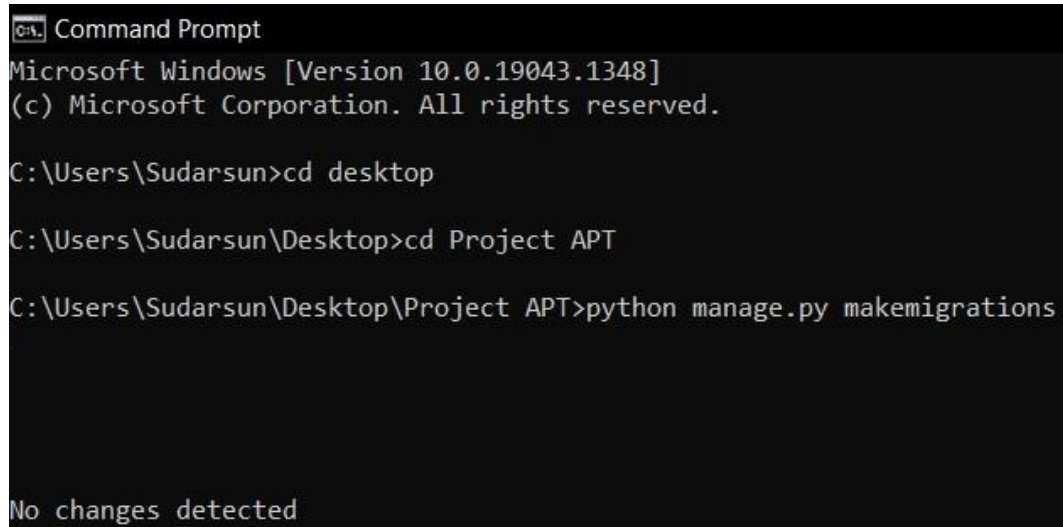
```
1 row in set (0.00 sec)
```

```
mysql> select * from academics_tracker_app_report;
```

id	expected_percent	scored_percent	difference	outcome
2	80	97	17	Did Better

```
1 row in set (0.00 sec)
```

## Command Prompt:



```
C:\> Command Prompt
Microsoft Windows [Version 10.0.19043.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sudarsun>cd desktop

C:\Users\Sudarsun\Desktop>cd Project APT

C:\Users\Sudarsun\Desktop\Project APT>python manage.py makemigrations

No changes detected
```

Command Prompt

```
C:\Users\Sudarsun\Desktop\Project APT>python manage.py migrate
System check identified some issues:

Operations to perform:
  Apply all migrations: academics_tracker_app, admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0001_initial... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying academics_tracker_app.0001_initial... OK
  Applying academics_tracker_app.0002_auto_20200626_1629... OK
  Applying academics_tracker_app.0003_auto_20200627_0534... OK
  Applying academics_tracker_app.0004_auto_20200703_0740... OK
  Applying academics_tracker_app.0005_studentresult... OK
  Applying academics_tracker_app.0006_alter_customuser_first_name... OK
  Applying academics_tracker_app.0007_auto_20211117_1238... OK
  Applying academics_tracker_app.0008_auto_20211118_1329... OK
  Applying academics_tracker_app.0009_auto_20211122_1705... OK
  Applying academics_tracker_app.0010_auto_20211122_2018... OK
  Applying academics_tracker_app.0011_auto_20211122_2349... OK
  Applying academics_tracker_app.0012_alter_customuser_user_type... OK
  Applying academics_tracker_app.0013_alter_exams_id... OK
  Applying academics_tracker_app.0014_alter_subjects_exam_id... OK
  Applying academics_tracker_app.0015_subjects_marks... OK
  Applying academics_tracker_app.0016_auto_20211123_1521... OK
  Applying academics_tracker_app.0017_summary... OK
  Applying academics_tracker_app.0018_report... OK
  Applying academics_tracker_app.0019_delete_studentresult... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

Command Prompt - python manage.py runserver

```
C:\Users\Sudarsun\Desktop\Project APT>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified some issues:
December 03, 2021 - 13:51:14
Django version 3.2.7, using settings 'academics_performance_tracker.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```



# Testing Results

## GENERATED REPORT:

Academics Tracker

HomeExams/AssignmentsCoursesSubjectsSessionsPerformance Summary

Academics Performance Tracker User Login

Generated Report For TOC

Exam	Score Obtained	Total Score	Target Score
Digital Assignment 1	10.0	10	10
Digital Assignment 2	10.0	10	10
Digital Assignment 3	10.0	10	10
CAT1	5.0	15	10
CAT2	8.0	15	10
FAT	27.0	40	30

Expected Percentage	Scored Percentage
80.0	70.0

It's not the time to look for excuses, Keep trying hard.

Back

Academics Tracker

HomeExams/AssignmentsCoursesSubjectsSessionsPerformance Summary

Academics Performance Tracker User Login

Generated Report For SWE

Exam	Score Obtained	Total Score	Target Score
Digital Assignment 1	10.0	10	10
Digital Assignment 2	10.0	10	10
Digital Assignment 3	10.0	10	10
CAT1	14.0	15	10
CAT2	15.0	15	10
FAT	38.0	40	30

Expected Percentage	Scored Percentage
80.0	97.0

The tassel was worth the hassle. Congratulations.

Back

# TESTCASE DOCUMENT:

A	B	C	D	E	F	G	H	I	J	K
	TestCase_ID	Objectives	Precondition	Steps	Test Data	Expected Result	Postcondition	Actual Result	Status	
	PH1_1	Sign Up	Should be a new user	Open the system Click on sign up option Enter personal details Update auto-login details with your choice Cereentials sent to email.	Username password name email institution name	1.1 Success  1.2 Error	Link sent to mail  invalid detials	success  error	verified  verified	
	PH1_2	Sign In	Already registered user	Open the system Click on sign in option Enter the username Enter the password Click on login	username password	2.1 Success  2.2 Error	Moves to home page  Incorrect Uname/Pw	success  error	verified  verified	
	PH2_1	Manage Session	Outdated sessions Current sessions	add start session: 1. select the current year 2. pick a month ahead 3. pick the first day of college  add end session: 1. enter end semester year 2. pick your exam month 3. select the last day	start time: year month date  end time: year month date	1.1 Succes 1.2 Error  1.3 Success 1.3 Error	session created session clash-clear data  valid duration ends before start of the session	success error  Success Error	verified verified  verified verified	
	PH2_2	Manage Course	Pre-existing titles	add/edit course: 1.Pick a session 2.Enter the course name	duration text	2.1 Success 2.2 Error	create new/update data course already exists	success error	verified verified	
	PH2_3	Manage Exams	nil	add/edit exam details 1.Enter exam name 2.Enter exam weightage	text numerical	3.1 Success 3.2 Error	update data clear psst data	success error	verified verified	
	PH2_4	Manage subjects	session; course; exams table should not be empty	add/edit subjects tab 1. pick a session 2. enter the course 3. enter the exam name 4. type the subject name 5. enter scored marks	duration course title exam type subject major numerical	4.1 Success 4.2 Error	create database invalid options	success error	verified verified	
	PH3_1	Positive report	success in all phase 2 conditions	pick a session; course; subject	data from existing database	scored >= target	display comparison congrats message	success	verified	
	PH3_2	Negative report	success in all phase 2 conditions	pick a session; course; subject	data from existing database	scored < target	display comparison do better message	success	verified	

## **Conclusion**

The academic performance tracker has been built using Python,HTML and MySql which is more than just a tracker. It has inbuilt todo list and performance summary page which displays the report using the input marks and targets set for each exam in an innovative manner along with appropriate feedback to motivate the students depending on whether they achieve the target or not. This software can help students to stay on top of their academic activities and exams, thereby helping them to overcome the difficulties of the age-old pen and paper method of tracking along with having the high end helpful tools which they can utilize to score awesome grades in their academics.

# References

- <https://techvidvan.com/tutorials/python-to-do-list/>
- <https://itsourcecode.com/free-projects/python-projects/to-do-list-project-in-python-with-source-code/>
- <https://stackoverflow.com/questions/63153577/pandas-python-creating-an-indexed-performance-tracker>
- <https://www.geeksforgeeks.org/create-mysql-database-login-page-in-python-using-tkinter/>
- <https://www.codegrepper.com/code-examples/python/how+to+create+a+login+page+in+python>
- <https://www.section.io/engineering-education/user-login-web-system/>