**CHAPTER-9**

**APPENDICES**

**9.1 APPENDIX-A: SAMPLE SOURCE CODE**

**Main.py**

from django.http import HttpResponse

from django.shortcuts import get\_object\_or\_404, render, redirect

from django.contrib.auth.models import User

from django.contrib.auth import login,authenticate,logout

from django.contrib.auth.decorators import login\_required

from django.contrib import messages

from .models import EvaluationResult, ExamSubmission,Exam

from .evaluation.ocr import generate\_ocr

from .evaluation.extract\_question\_answerkey import question\_answer\_content

from .evaluation.preprocess\_ocr import preprocess\_ocr\_question\_wise

from .evaluation.evalution import evaluate\_exam\_with\_ocr\_to\_json

from .evaluation.report import generate\_report

from.evaluation.proper\_json import parse\_json\_string

import json

def home(request):

    return render(request, 'home.html')

def signup\_view(request):

    if request.method == "POST":

        username = request.POST['username']

        email = request.POST['email']

        password1 = request.POST['password1']

        password2 = request.POST['password2']

        if password1 != password2:

            messages.error(request, "Passwords do not match!")

            return redirect('signup')

        if User.objects.filter(username=username).exists():

            messages.error(request, "Username already taken!")

            return redirect('signup')

        if User.objects.filter(email=email).exists():

            messages.error(request, "Email is already in use!")

            return redirect('signup')

        user = User.objects.create\_user(username=username, email=email, password=password1)

        login(request, user)

        messages.success(request, "Account created successfully!")

        return redirect('login')

    return render(request, 'authentication/signup.html')

def login\_view(request):

    if request.method == "POST":

        username = request.POST['username']

        password = request.POST['password']

        user = authenticate(request, username=username, password=password)

        if user is not None:

            login(request, user)

            messages.success(request, "Login successful!")

            return redirect('student\_dashboard')

        else:

            messages.error(request, "Invalid username or password!")

    return render(request, 'authentication/login.html')

def student\_dashboard(request):

    exams = ExamSubmission.objects.filter(student=request.user)  # Fetch exams created by logged-in student

    return render(request, 'dashboard/student/student\_dashboard.html', {'exams': exams})

def logout\_view(request):

    logout(request)

    messages.success(request, "Logged out successfully!")

    return redirect('login')

def student\_exam\_fill(request):

    if request.method == "POST":

        subject = request.POST.get("subject")

        exam\_type = request.POST.get("exam\_type")

        year = request.POST.get("year")

        staff\_name = request.POST.get("staff\_name")

        # Check if an exam exists with these details

        exam = Exam.objects.filter(year=year).first()

        if not exam:

            messages.error(request, "❌ No matching exam found. Please check the details.")

            return redirect("student\_exam\_fill")  # Prevent saving if exam doesn't exist

        # Create a new submission linked to this exam

        submission = ExamSubmission.objects.create(

            exam=exam,  # Assigning the required exam field

            student=request.user,

            subject=subject,

    if request.method == "POST":

        subject = request.POST.get("subject")

        exam\_type = request.POST.get("exam\_type")

        year = request.POST.get("year")

        staff\_name = request.POST.get("staff\_name")             exam\_type=exam\_type,

            year=year,

            staff\_name=staff\_name,

        )

        messages.success(request, "✅ Exam submission successful!")

        return redirect("student\_dashboard")

    return render(request, "dashboard/student/exam\_fill.html")

def teacher\_login(request):

    if request.method == "POST":

        username = request.POST["username"]

        password = request.POST["password"]

        user = authenticate(request, username=username, password=password)

        if user is not None:

            if user.is\_superuser:  # Allow only superusers

                login(request, user)

                messages.success(request, "Welcome, Teacher!")

                return redirect("teacher\_dashboard")  # Redirect to teacher dashboard

            else:

                messages.error(request, "Access Denied! Only teachers (superusers) can log in.")

        else:

            messages.error(request, "Invalid Username or Password!")

    return render(request, "dashboard/teacher/teacher\_login.html")

@login\_required

def teacher\_dashboard(request):

    if not request.user.is\_superuser:

        return redirect("home")  # Redirect unauthorized users

    exams = Exam.objects.all().order\_by("-id")  # Fetch all exams

    return render(request, "dashboard/teacher/teacher\_dashboard.html", {"exams": exams})

@login\_required

def create\_exam(request):

    if not request.user.is\_superuser:

        messages.error(request, "❌ Unauthorized access!")

        return redirect("home")

        question\_paper = request.FILES.get("question\_paper")

        answer\_key = request.FILES.get("answer\_key")

        if not all([subject, exam\_type, year, staff\_name, question\_paper, answer\_key]):

            messages.error(request, "⚠️ All fields are required!")

            return redirect("create\_exam")

        Exam.objects.create(

            subject=subject,

            exam\_type=exam\_type,

            year=year,

            staff\_name=staff\_name,

            question\_paper=question\_paper,

            answer\_key=answer\_key

        )

        messages.success(request, "✅ Exam successfully created!")

        return redirect("teacher\_dashboard")

    return render(request, "dashboard/teacher/create\_exam.html")

@login\_required

def view\_submissions(request, exam\_id):

    exam = get\_object\_or\_404(Exam, id=exam\_id)

    submissions = ExamSubmission.objects.filter(year=exam.year)

    if request.method == "POST":

        for submission in submissions:

            file\_field\_name = f"answer\_sheet\_{submission.id}"

            if file\_field\_name in request.FILES:

                if submission.answer\_sheet:

                    messages.warning(request, f"⚠️ Answer sheet for {submission.student.username} already uploaded.")

                else:

                    submission.answer\_sheet = request.FILES[file\_field\_name]

                    submission.save()

                    messages.success(request, f"✅ Answer sheet uploaded for {submission.student.username}.")

        return redirect('view\_submissions', exam\_id=exam.id)

    return render(request, "dashboard/teacher/view\_submissions.html", {"exam": exam, "submissions": submissions})

def evaluate\_submission\_view(request, submission\_id):

    submission = get\_object\_or\_404(ExamSubmission, id=submission\_id)

    # 🔍 Check if the submission is already evaluated

    # Render the evaluation results page

    return render(request, 'dashboard/teacher/evaluate\_submission.html', {

        'submission': submission,

        'formatted\_report': formatted\_report,

        'total\_score': total\_score,

        'max\_score': max\_score

    })     evaluation = EvaluationResult.objects.filter(submission=submission).first()

    if evaluation:

        messages.info(request, "This submission has already been evaluated.")

        formatted\_report = parse\_json\_string(evaluation.formatted\_report)

        total\_score = evaluation.total\_score

        max\_score = evaluation.max\_score

    else:

        #OCR text from uploaded answer sheet

        ocr\_text = generate\_ocr(submission.answer\_sheet.path)

        # Extract question paper and answer key

        question\_paper\_text = question\_answer\_content(submission.exam.question\_paper.path)

        answer\_key\_text = question\_answer\_content(submission.exam.answer\_key.path)

        # Preprocess OCR text to align with question numbers

        structured\_ocr\_text = preprocess\_ocr\_question\_wise(ocr\_text, question\_paper\_text)

        # Evaluate answers using Gemini API

        evaluation\_result\_json = evaluate\_exam\_with\_ocr\_to\_json(structured\_ocr\_text, answer\_key\_text)

        formatted\_report = generate\_report(evaluation\_result\_json)

        formatted\_report = parse\_json\_string(formatted\_report)

        print(formatted\_report)

        total\_score = formatted\_report["summary"]["user\_total\_score"]

        max\_score =  formatted\_report["summary"]["total\_possible\_score"]

        # Save the evaluation result in the database

        evaluation = EvaluationResult.objects.create(

            submission=submission,

            evaluated\_by=request.user,

            formatted\_report=json.dumps(formatted\_report),

            total\_score=total\_score,

            max\_score=max\_score,

        )

        submission.is\_graded = True

        submission.save()

        messages.success(request, "Evaluation completed successfully!")

def view\_results(request,exam\_id):

    submission = get\_object\_or\_404(ExamSubmission, id=exam\_id)

    # Check if the submission is already evaluated

    evaluation = EvaluationResult.objects.filter(submission=submission).first()

    if evaluation:

        messages.info(request, "This submission has already been evaluated.")

        formatted\_report = parse\_json\_string(evaluation.formatted\_report)

        total\_score = evaluation.total\_score

        max\_score = evaluation.max\_score

    return render(request, 'dashboard/teacher/evaluate\_submission.html', {

        'submission': submission,

        'formatted\_report': formatted\_report,

        'total\_score': total\_score,

        'max\_score': max\_score

    })

**Urls.py**

from django.contrib import admin

from django.urls import path

from app import views

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

    path("admin/", admin.site.urls),

    path('', views.home, name='home'),

    path('signup/', views.signup\_view, name='signup'),

    path('login/', views.login\_view, name='login'),

    path('logout/', views.logout\_view, name='logout'),

    path('student\_dashboard/', views.student\_dashboard, name='student\_dashboard'),

    path('view-results/<int:exam\_id>/', views.view\_results, name='view\_results'),

    path('student\_exam\_fill/', views.student\_exam\_fill, name='student\_exam\_fill'),

    path('teacher-login/', views.teacher\_login, name='teacher\_login'),

    path('teacher-dashboard/', views.teacher\_dashboard, name='teacher\_dashboard'),

    path('create-exam/', views.create\_exam, name='create\_exam'),

    path('view-submissions/<int:exam\_id>/', views.view\_submissions, name='view\_submissions'),

    path('evaluate/<int:submission\_id>/', views.evaluate\_submission\_view, name='evaluate\_submission'),

]+ static(settings.MEDIA\_URL,document\_root=settings.MEDIA\_ROOT)

urlpatterns+= static(settings.STATIC\_URL,document\_root=settings.STATIC\_ROOT)

**Models.py**

from django.db import models

from django.contrib.auth.models import User

class Exam(models.Model):

    YEAR\_CHOICES = [

        (1, "First Year"),

        (2, "Second Year"),

        (3, "Third Year"),

        (4, "Fourth Year"),

    ]

    EXAM\_TYPE\_CHOICES = [

        ("CAT1", "CAT 1"),

        ("CAT2", "CAT 2"),

    ]

    subject = models.CharField(max\_length=255)

    exam\_type = models.CharField(max\_length=4, choices=EXAM\_TYPE\_CHOICES, default="CAT1")

    year = models.IntegerField(choices=YEAR\_CHOICES)

    staff\_name = models.CharField(max\_length=255)

    question\_paper = models.FileField(upload\_to='question\_papers/')

    answer\_key = models.FileField(upload\_to='answer\_keys/')

    created\_at = models.DateTimeField(auto\_now\_add=True)

    def \_\_str\_\_(self):

        return f"{self.subject} - {dict(self.YEAR\_CHOICES).get(self.year, 'Unknown')} - {self.get\_exam\_type\_display()}"

class ExamSubmission(models.Model):

    EXAM\_TYPES = [

        ('CAT1', 'CAT 1'),

        ('CAT2', 'CAT 2'),

    ]

    YEARS = [

        (1, "First Year"),

        (2, "Second Year"),

        (3, "Third Year"),

        (4, "Fourth Year"),

    ]

    exam = models.ForeignKey(Exam, on\_delete=models.CASCADE)  # Remove null=True, blank=True

    student = models.ForeignKey(User, on\_delete=models.CASCADE)

    subject = models.CharField(max\_length=100)

    exam\_type = models.CharField(max\_length=10, choices=EXAM\_TYPES)

    year = models.CharField(max\_length=1, choices=YEARS)

    staff\_name = models.CharField(max\_length=100)

    answer\_sheet = models.FileField(upload\_to='answer\_sheets/', null=True, blank=True)

    is\_graded = models.BooleanField(default=False)

    def \_\_str\_\_(self):

        return f"{self.subject} - {self.exam\_type} ({self.get\_year\_display()})"

class EvaluationResult(models.Model):

    submission = models.OneToOneField(

        ExamSubmission,

        on\_delete=models.CASCADE,

        related\_name="evaluation"

    )

    evaluated\_by = models.ForeignKey(

        User,

        on\_delete=models.SET\_NULL,

        null=True,

        blank=True,

        related\_name="evaluations"

    )

    formatted\_report = models.TextField()  # Stores only the human-readable report

    total\_score = models.FloatField(default=0.0)

    max\_score = models.FloatField(default=0.0)

    created\_at = models.DateTimeField(auto\_now\_add=True)

    def \_\_str\_\_(self):

        exam\_subject = self.submission.exam.subject if self.submission.exam else "Unknown Exam"

        return f"Evaluation for {self.submission.student.username} {exam\_subject}"

**Admin.py**

from django.contrib import admin

from .models import EvaluationResult,Exam

admin.site.register(EvaluationResult)

admin.site.register(Exam)

**Student-dashboard.html**

{% extends 'base.html' %}

{% block content %}

<div class="container mt-5">

    <div class="card shadow-lg p-4">

        <h2 class="text-center text-primary fw-bold">Welcome, {{ request.user.username }}!</h2>

        <hr>

        <div class="d-flex justify-content-between align-items-center mb-4">

            <h3 class="text-secondary fw-semibold">📄 Your Submitted Exams</h3>

            <a href="{% url 'student\_exam\_fill' %}" class="btn btn-success btn-lg shadow-sm">

                + Fill Exam Details

            </a>

        </div>

        {% if exams %}

        <div class="table-responsive">

            <table class="table table-hover table-bordered text-center align-middle">

                <thead class="table-dark">

                    <tr>

                        <th>📚 Subject</th>

                        <th>📝 Exam Type</th>

                        <th>🎓 Year</th>

                        <th>👨‍🏫 Staff Name</th>

                        <th>📊 Status</th>

                        <th>🔍 Actions</th>

                    </tr>

                </thead>

                <tbody>

                    {% for exam in exams %}

                    <tr>

                        <td class="fw-bold">{{ exam.subject }}</td>

                        <td>{{ exam.get\_exam\_type\_display }}</td>

                        <td>{{ exam.get\_year\_display }}</td>

                        <td>{{ exam.staff\_name }}</td>

                        <td>

                            {% if exam.is\_graded %}

                                <span class="badge bg-success px-3 py-2">Graded</span>

                            {% else %}

                                <span class="badge bg-warning text-dark px-3 py-2">Pending</span>

                            {% endif %}

                        </td>

                        <td>

                            {% if exam.is\_graded %}

                                <a href="{% url 'view\_results' exam.id %}" class="btn btn-primary btn-sm shadow-sm">

                                    View Results

                                </a>

                            {% else %}

                                <button class="btn btn-secondary btn-sm shadow-sm" disabled>Awaiting Grading</button>

                            {% endif %}

                        </td>

                    </tr>

                    {% endfor %}

                </tbody>

            </table>

        </div>

        {% else %}

        <div class="alert alert-info text-center">

            <p class="mb-0">🚀 No exams submitted yet. Start by filling out your first exam!</p>

        </div>

        {% endif %}

        <div class="text-center mt-4">

            <a href="{% url 'logout' %}" class="btn btn-danger btn-lg px-4 shadow-sm">🚪 Logout</a>

        </div>

    </div>

</div>

<style>

    body {

        background-color: #f8f9fa;

    }

    .card {

        border-radius: 12px;

        border: none;

        box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);

    }

    .table th {

        background-color: #212529;

        color: white;

    }

    .table td {

        vertical-align: middle;

    }

    .btn {

        border-radius: 8px;

    }

    .btn-success {

        background-color: #28a745;

    }

</style>

{% endblock %}

**Teacher-dashboard.html**

{% extends 'base.html' %}

{% block content %}

<div class="container-fluid">

    <div class="row">

        <!-- Sidebar -->

        <nav class="col-md-3 col-lg-2 d-md-block bg-dark sidebar vh-100 p-3">

            <h4 class="text-white text-center">📚 Teacher Panel</h4>

            <hr class="text-white">

            <ul class="nav flex-column">

                <li class="nav-item">

                    <a class="nav-link text-white" href="{% url 'teacher\_dashboard' %}"> Dashboard</a>

                </li>

                <li class="nav-item">

                    <a class="nav-link text-white" href="{% url 'create\_exam' %}">✏️ Create Exam</a>

                </li>

                <li class="nav-item">

                    <a class="nav-link text-white" href="{% url 'logout' %}"> Logout</a>

                </li>

            </ul>

        </nav>

        <!-- Main Content -->

        <main class="col-md-9 ms-sm-auto col-lg-10 px-md-4 mt-4">

            <div class="d-flex justify-content-between align-items-center">

                <h2 class="text-primary">👨‍🏫 Welcome, {{ request.user.username }}</h2>

                <a href="{% url 'create\_exam' %}" class="btn btn-success btn-lg shadow-sm">

                    + Create Exam

                </a>

            </div>

            <hr>

            <h3 class="text-secondary">📄 Created Exams</h3>

            {% if exams %}

                <div class="table-responsive">

                    <table class="table table-hover table-bordered text-center">

                        <thead class="table-dark">

                            <tr>

                                <th>Subject</th>

                                <th>Exam Type</th>

                                <th>Year</th>

                                <th>Actions</th>

                            </tr>

                        </thead>

                        <tbody>

                            {% for exam in exams %}

                                <tr>

                                    <td class="fw-bold">{{ exam.subject }}</td>

                                    <td>{{ exam.get\_exam\_type\_display }}</td>

                                    <td>{{ exam.get\_year\_display }}</td>

                                    <td>

  <a href="{% url 'view\_submissions' exam.id %}" class="btn btn-primary btn-sm">

                                            📂 View Submissions

                                        </a>

                                    </td>

                                </tr>

                            {% endfor %}

                        </tbody>

                    </table>

                </div>

            {% else %}

                <div class="alert alert-info text-center">

                    <p class="mb-0">No exams created yet.</p>

                </div>

            {% endif %}

        </main>

    </div>

</div>

<style>

    /\* Sidebar Styling \*/

    .sidebar {

        height: 100vh;

        position: fixed;

        left: 0;

        top: 0;

        width: 250px;

    }

    /\* Adjust main content \*/

    main {

        margin-left: 260px;

    }

    /\* Button Styling \*/

    .btn-sm {

        font-size: 0.9rem;

    }

    /\* Responsive Design \*/

    @media (max-width: 768px) {

        .sidebar {

            position: relative;

            height: auto;

            width: 100%;

        }

        main {

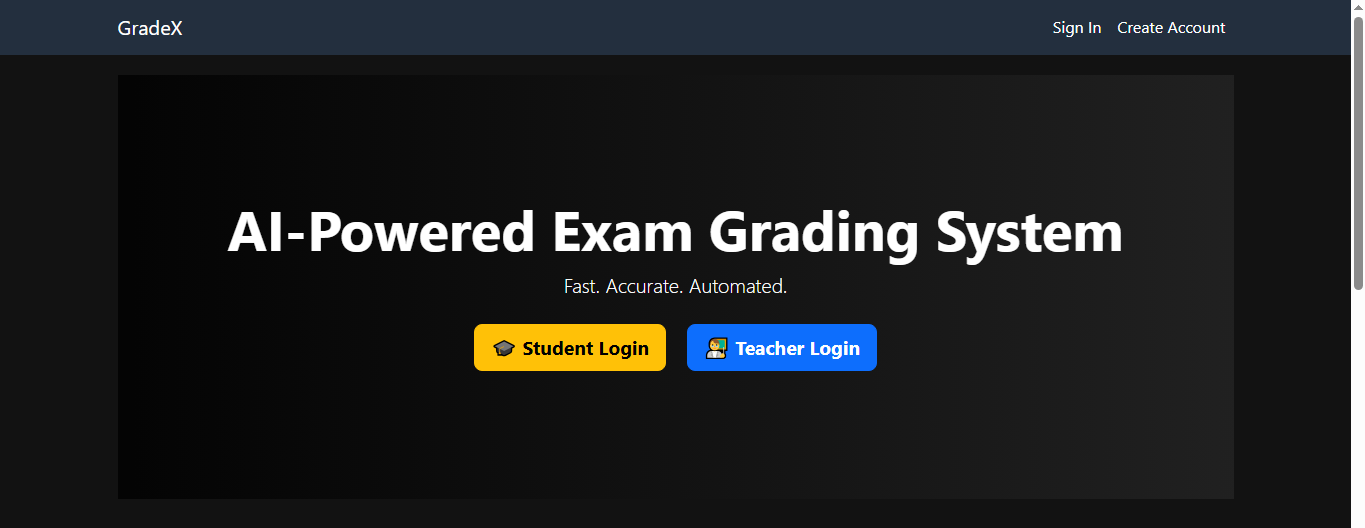
            margin-left: 0;

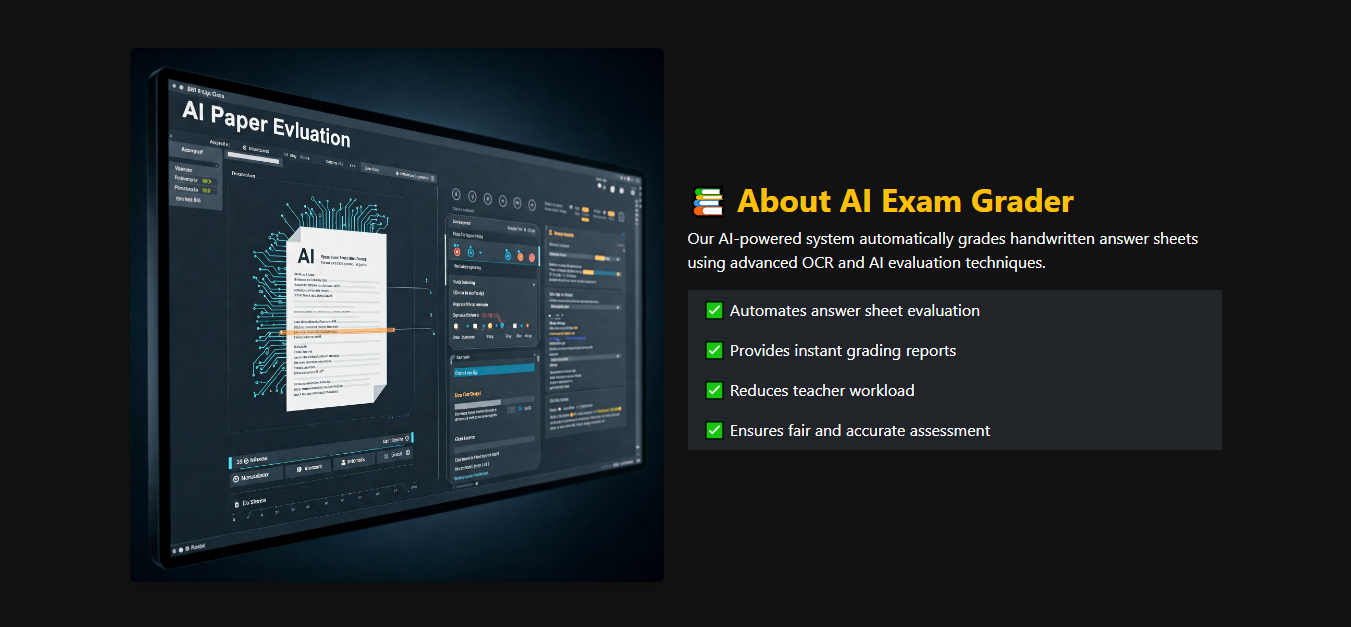
        }

    }

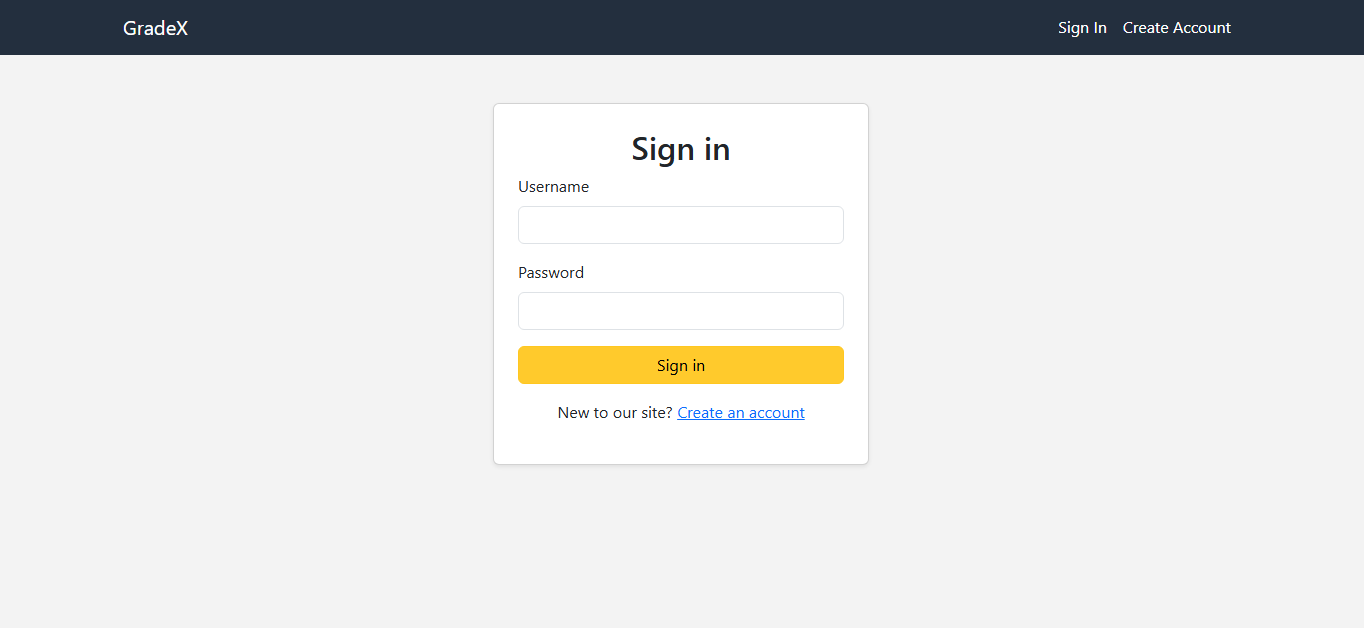
</style>

{% endblock %}

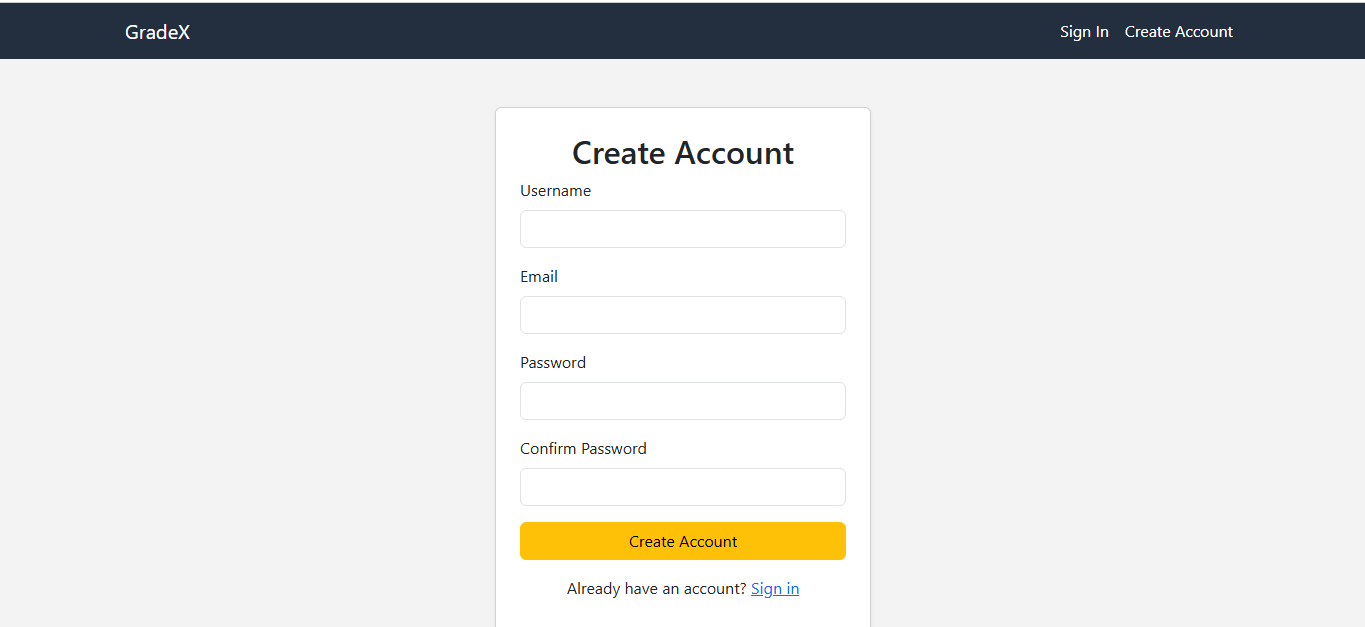
**9.2 APPENDIX-B: DEMO SCREENSHOTS**

**Fig: 9.1 GradeX Website**

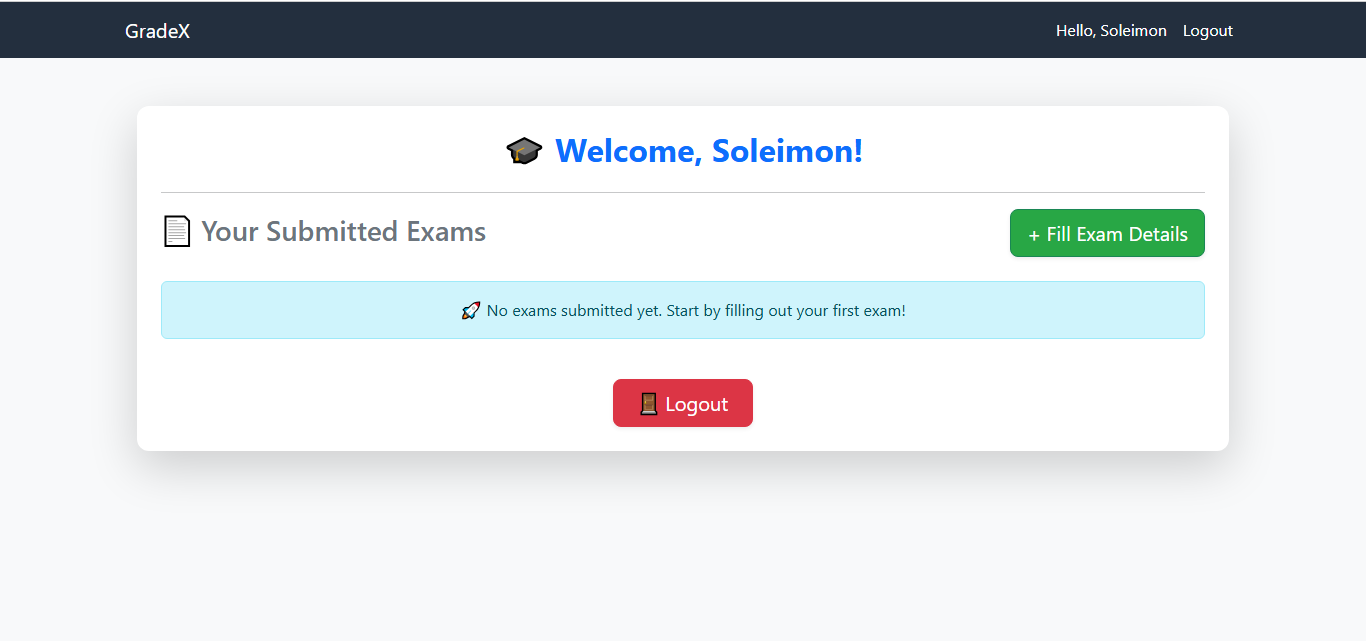
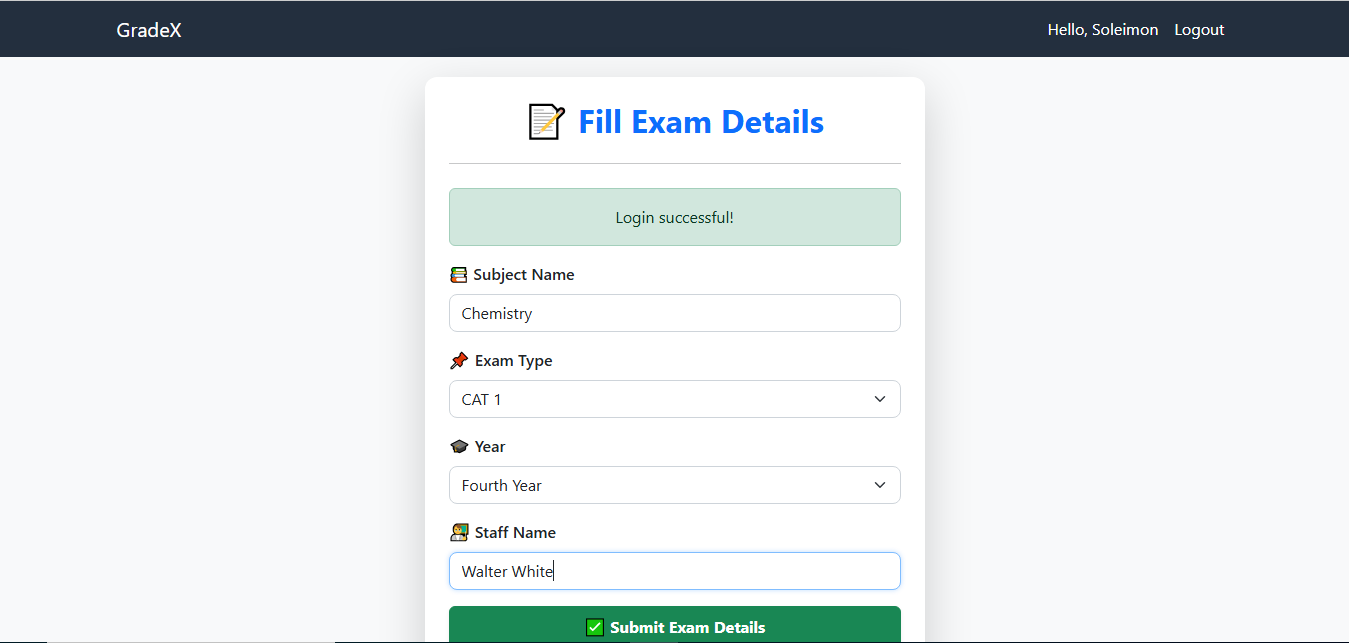
**Fig: 9.2 GradeX Information**



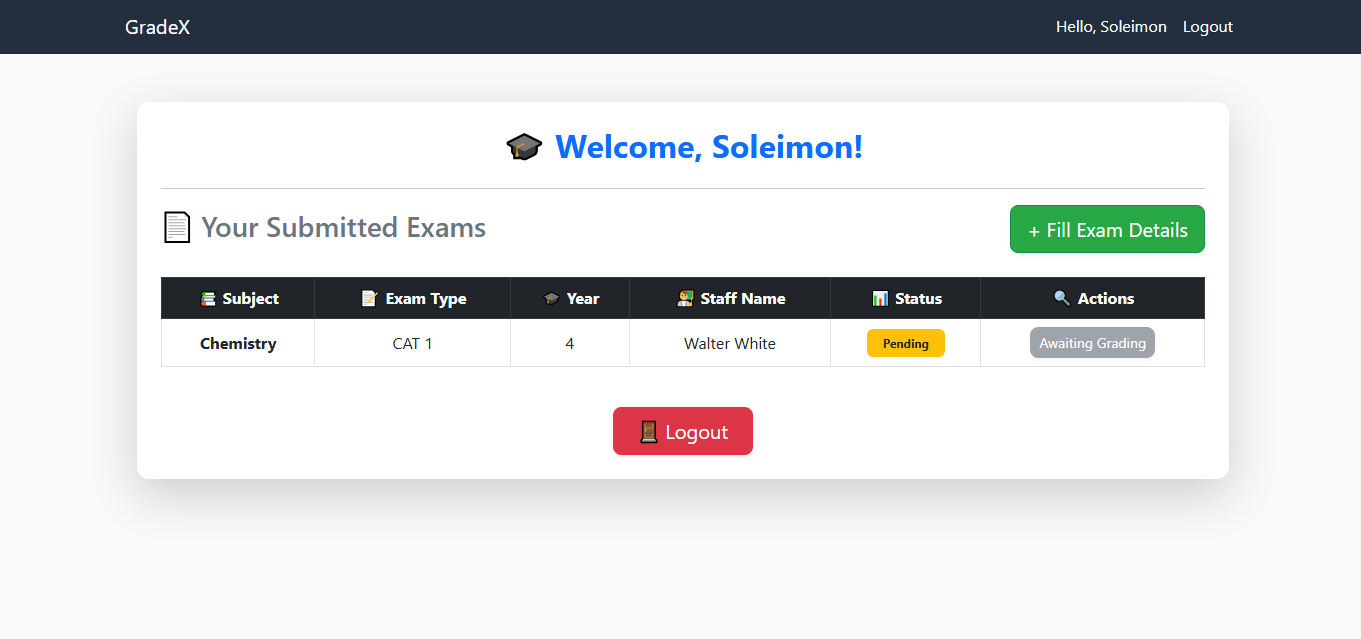
**Fig: 9.3 Gradex Student Signin**

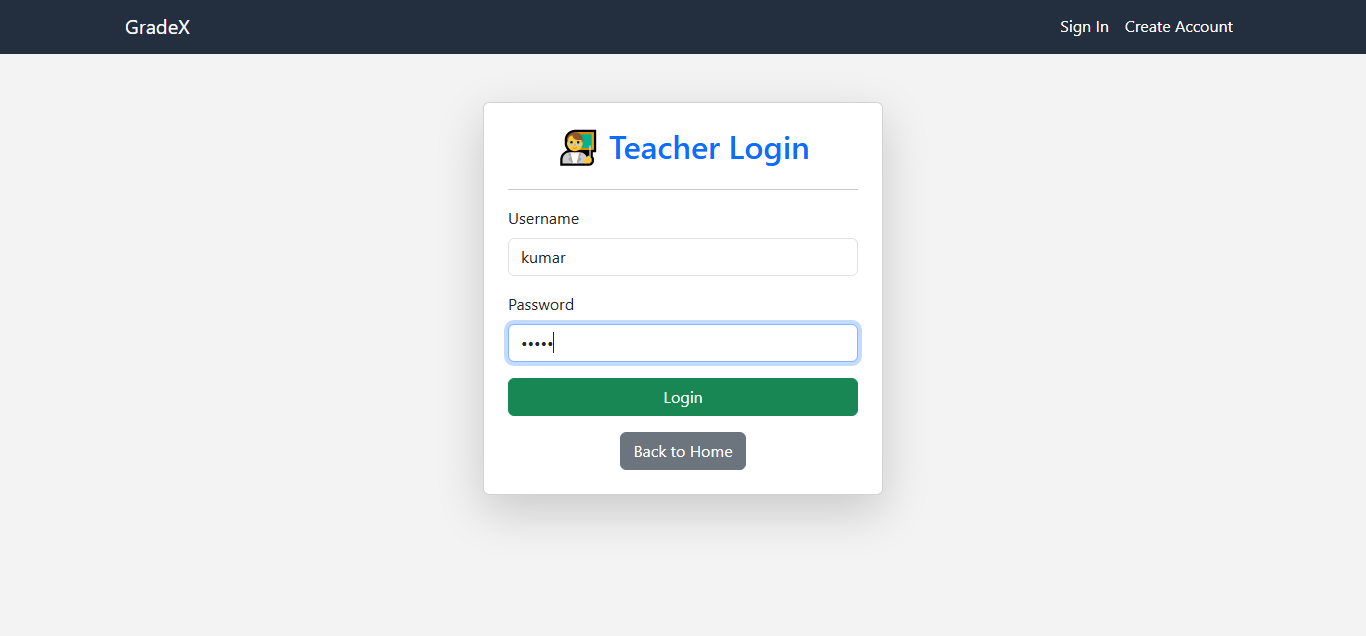


**Fig: 9.4 Gradex Student Sign up**

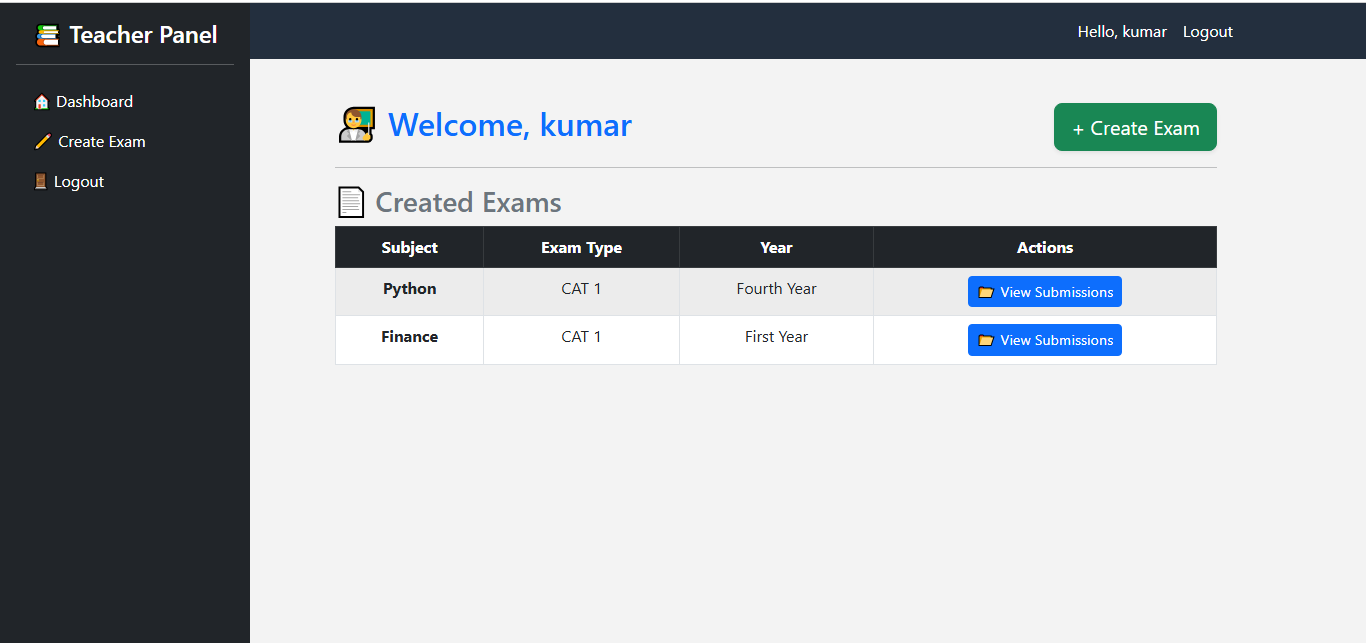
**Fig: 9.5 Gradex Student Dashboard**

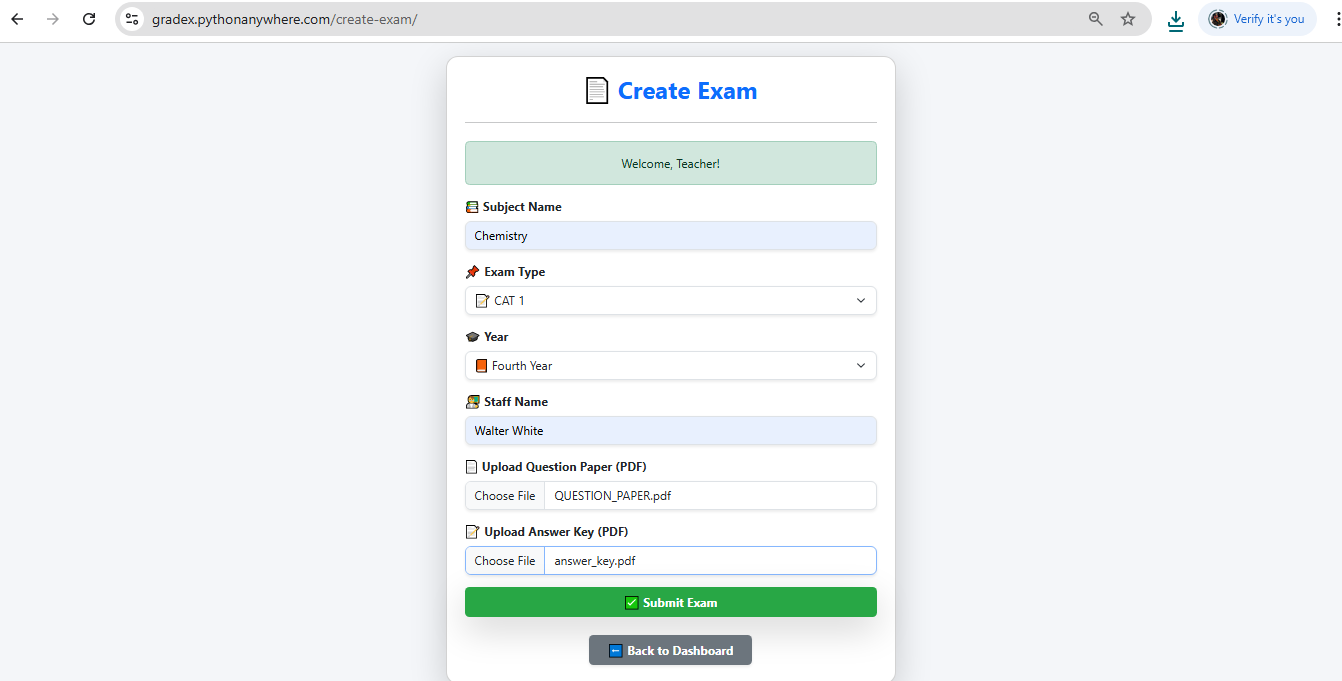
**Fig: 9.6 Gradex Student Exam Fill**



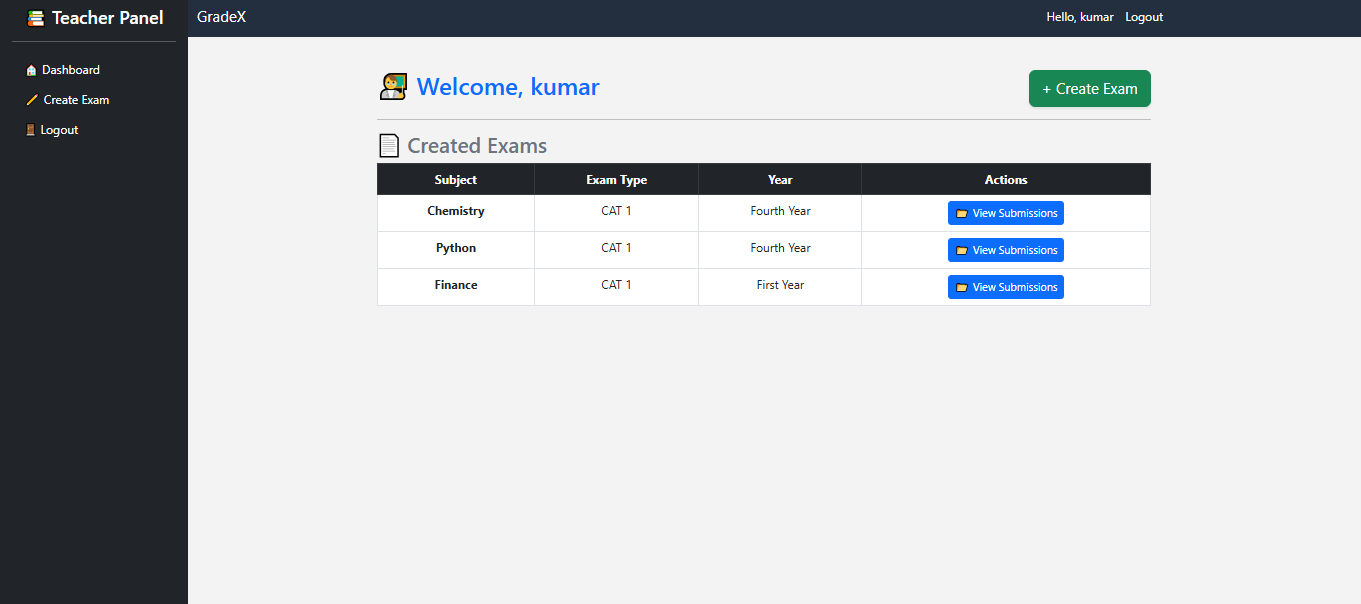
**Fig: 9.7 Gradex Student Awaiting Exam Status**

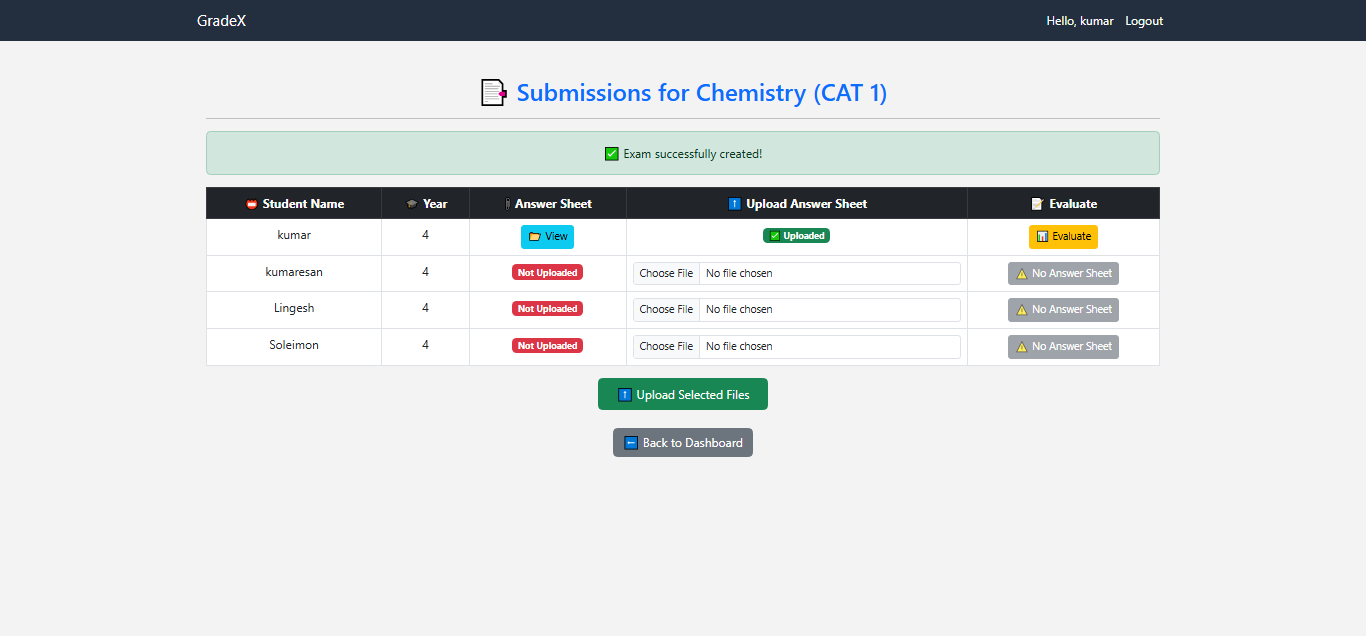
**Fig: 9.8 Gradex Teacher Login**



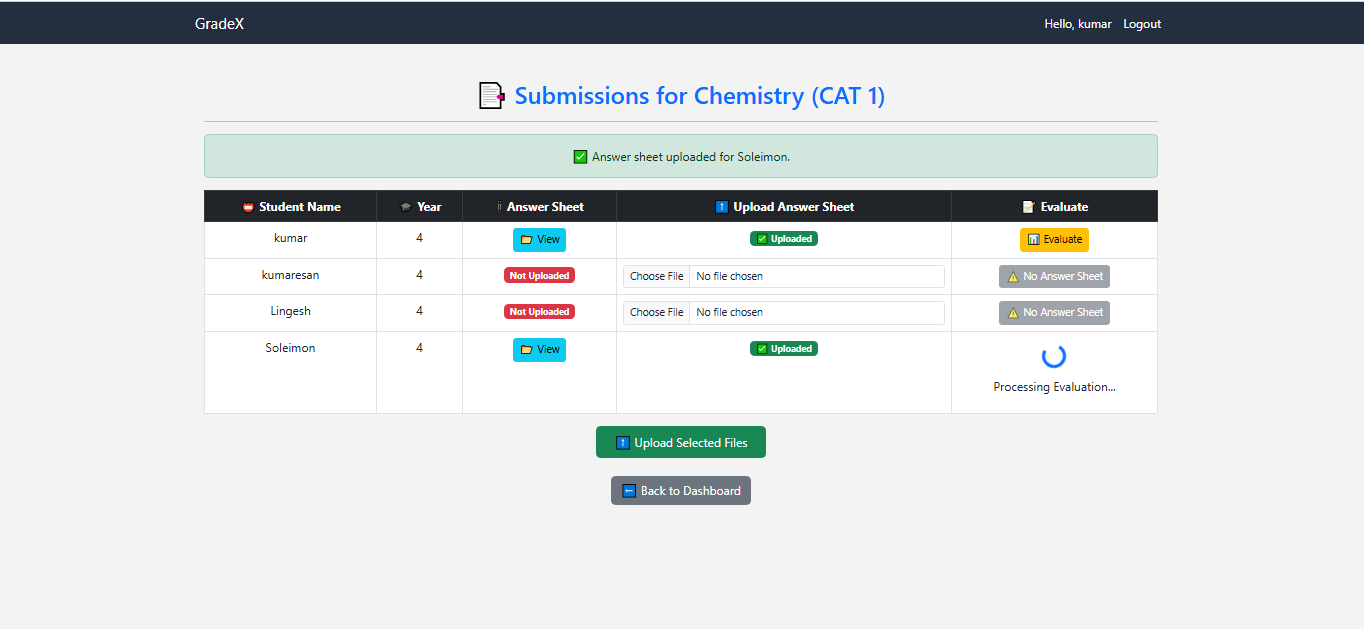
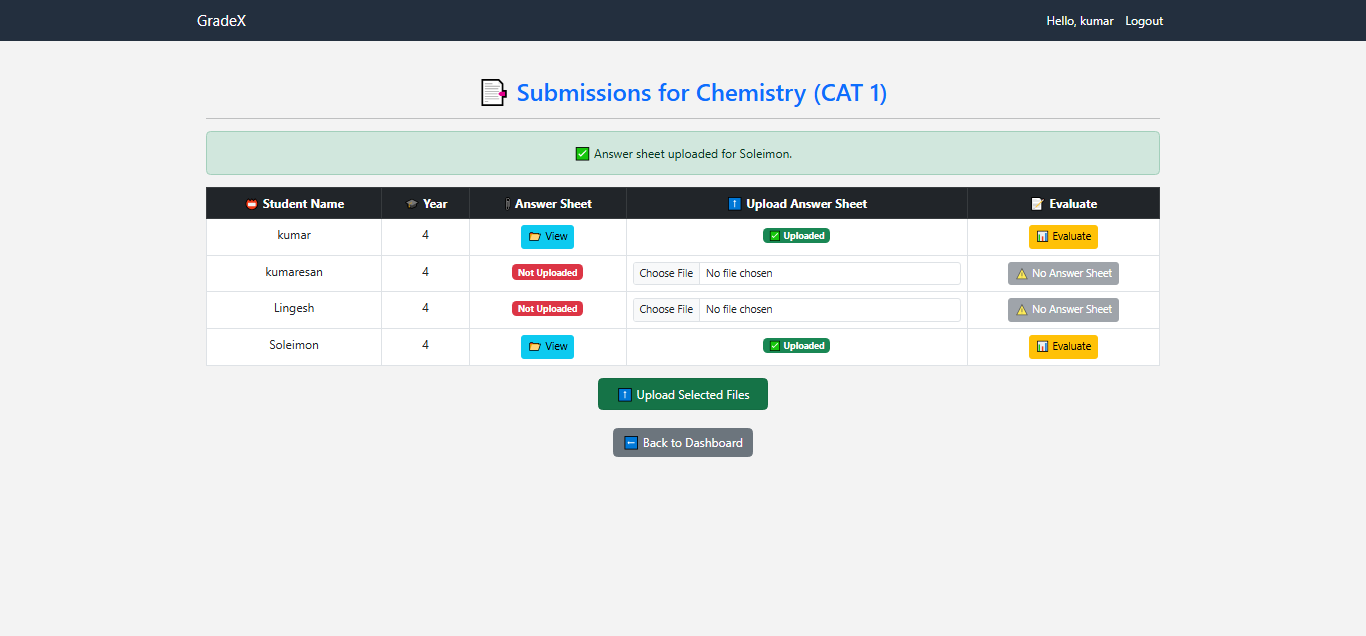
**Fig: 9.9 Gradex Teacher Dashboard**

**Fig: 9.10 Gradex Teacher Exam Creation**

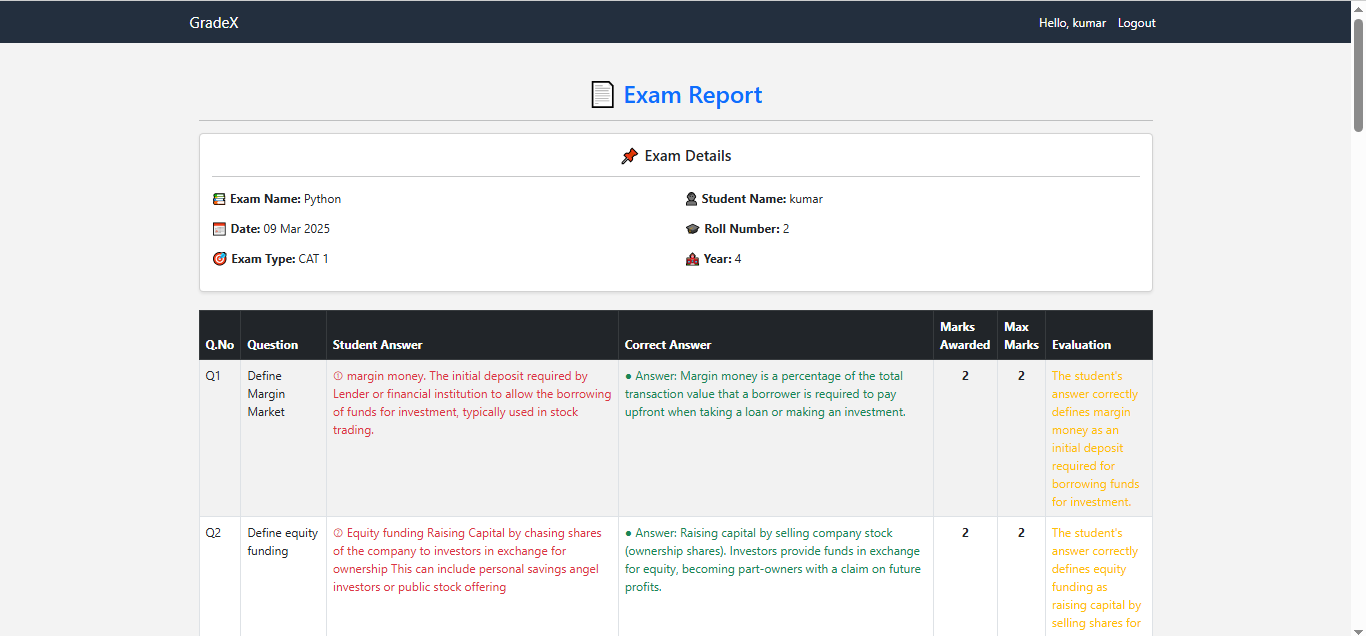


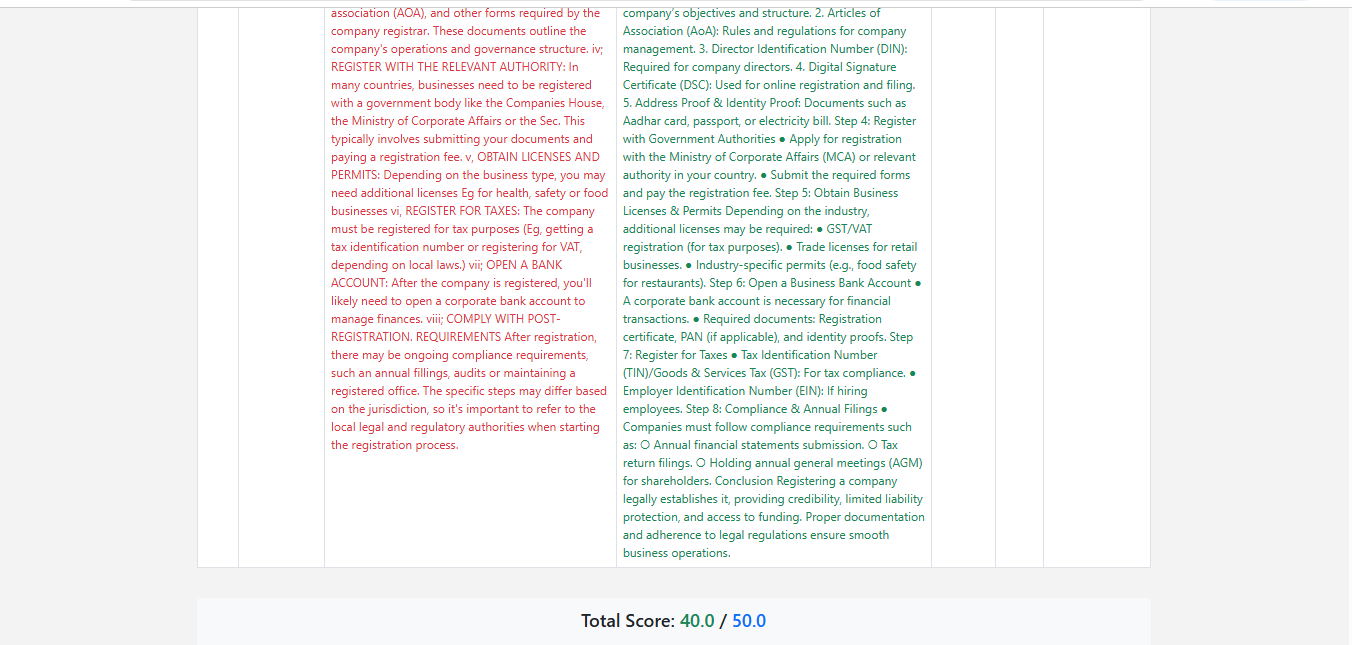
**Fig: 9.11 Gradex Teacher Dashboard-Created Exams**

**Fig: 9.12 Gradex Teacher Dashboard-Exam Submissin List**

**Fig: 9.13 Gradex Teacher Dashboard-Answer Sheet Upload** 

**Fig: 9.14 Gradex Teacher Dashboard-Answer Sheet Evaluating**



**Fig: 9.15 Gradex Teacher Dashboard-Results-1**

**Fig: 9.16 Gradex Teacher Dashboard-Results-2**

**Chapter 10**

**Future Enhancement**

While the Gradex system has successfully streamlined the process of evaluating handwritten student answer sheets using OCR and AI, the current workflow still requires manual scanning or photographing of the answer sheets before processing. A major area for improvement lies in automating this input step to make the system more seamless and scalable.

The primary future enhancement will focus on digitizing the answer sheet collection process. Instead of manually scanning or converting student-written sheets into PDFs or image formats, the system can be integrated with school digital infrastructure to automatically ingest answer sheets directly from:

* **Smart exam papers** written on digital pads or tablets with stylus input
* **Mobile app-based capture systems** where teachers simply click pictures, and the app auto-converts and uploads them to the backend
* **Scanner integration APIs** that trigger evaluation as soon as papers are scanned

This would eliminate delays, reduce human effort, and improve the overall efficiency of the system from input to evaluation.

In addition to this, several other enhancements are planned for the Gradex platform:

* **Multilingual Answer Sheet Support**  
  Expanding OCR and NLP capabilities to evaluate responses written in regional languages.
* **Real-time Evaluation via Digital Input Devices**  
  Supporting direct writing on tablets to allow instant feedback and auto-evaluation.
* **Learning Feedback Loop for Scoring Adjustment**  
  Using machine learning models to learn from teacher corrections and adapt future scoring.
* **Advanced Student Performance Analytics**  
  Generating detailed reports with topic-wise analytics, progress tracking, and feedback suggestions.
* **Plagiarism Detection**  
  Adding modules to detect similar or copied content between students' answers.
* **LMS and Mobile Integration**  
  Integrating with Learning Management Systems (LMS) and offering mobile apps for easy access by both students and teachers.

By focusing on automating the initial input step, Gradex will not only become more efficient but also truly scalable for large-scale educational deployments.

**REFERENCES**

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] A. Graves, S. Fernández, M. Liwicki, H. Bunke, and J. Schmidhuber, "Unconstrained online handwriting recognition with recurrent neural networks," in Proc. 20th Int. Conf. Neural Inf. Process. Syst. (NIPS), 2008, pp. 577–584.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in Proc. European Conf. Computer Vision (ECCV), 2014, pp. 818–833.

[5] R. Smith, "An overview of the Tesseract OCR engine," in Proc. Int. Conf. Document Anal. Recognit. (ICDAR), 2007, pp. 629–633.

[6] OpenAI, "GPT-4 technical report," arXiv preprint arXiv:2303.08774, 2023.

[7] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR), 2005, vol. 1, pp. 886–893.

[9] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in Proc. Int. Conf. Learn. Represent. (ICLR), 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 770–778.

**CONFERENCE CERTIFICATES**







**COURSE CERTIFICATES**

