

CHAPTER-9

APPENDICES

9.1 APPENDIX-A: SAMPLE SOURCE CODE

Main.py

```
from django.http import HttpResponseRedirect
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.evaluation 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=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

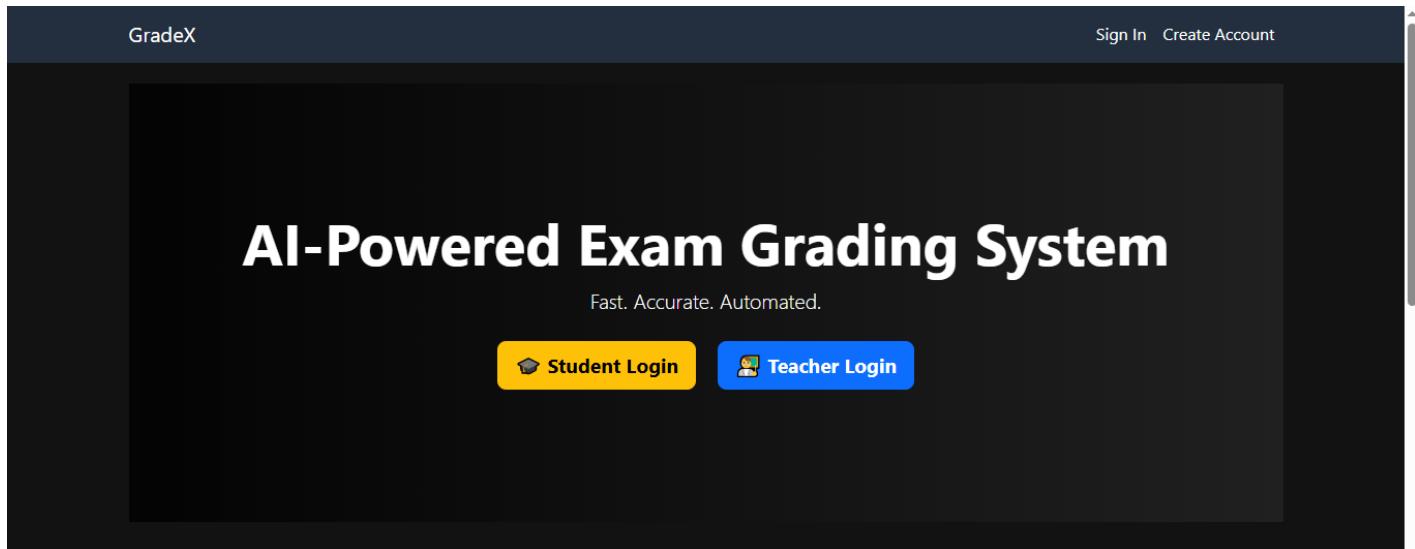
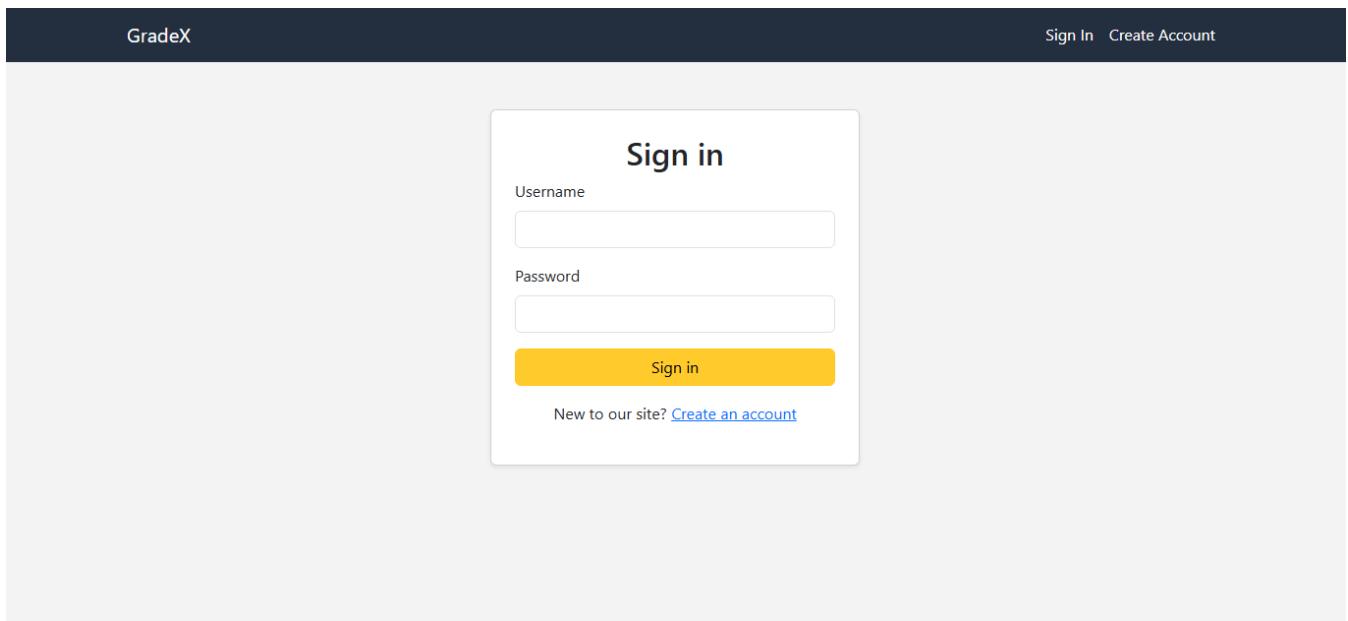


Fig: 9.1 GradeX Website

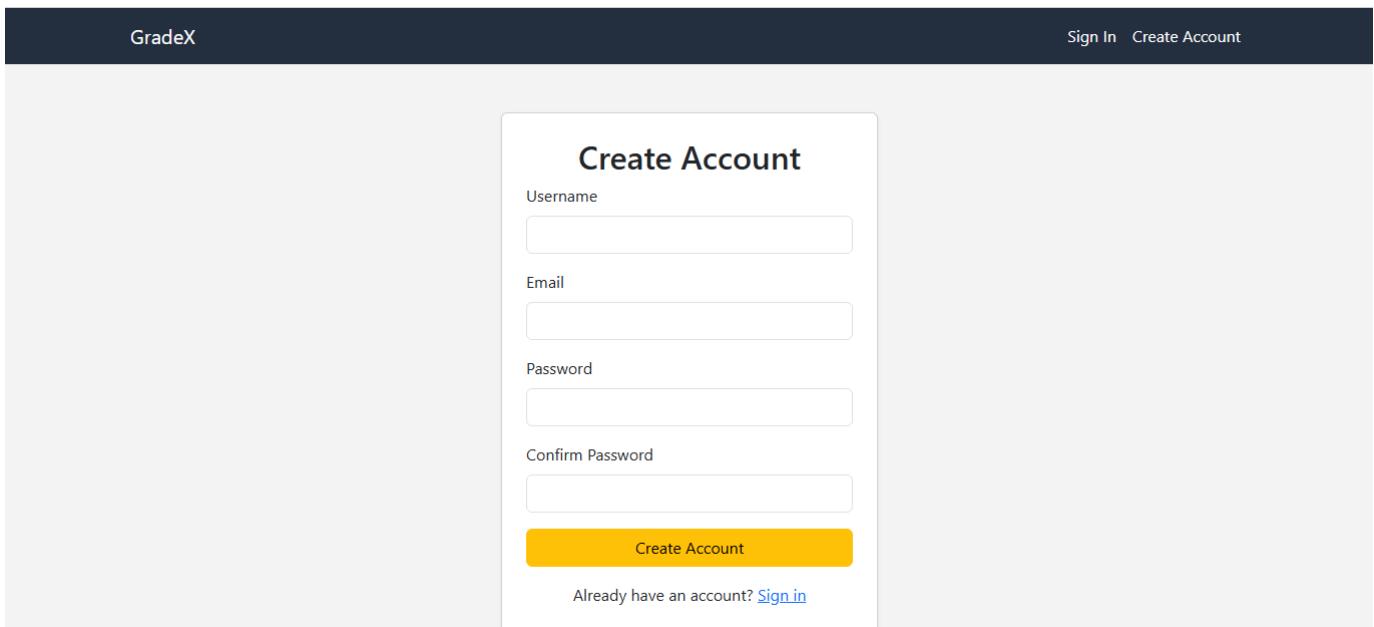
A screenshot of the AI Paper Evaluation software interface. On the left, a large window shows a document titled "AI Paper Evaluation" with a circuit board graphic overlaid. To the right, there are several smaller windows showing various data processing and analysis tools. On the far right, a sidebar contains sections like "About AI Exam Grader", "Our AI-powered system automatically grades handwritten answer sheets using advanced OCR and AI evaluation techniques.", and a list of four benefits with checkmarks: "Automates answer sheet evaluation", "Provides instant grading reports", "Reduces teacher workload", and "Ensures fair and accurate assessment".

Fig: 9.2 GradeX Information



The screenshot shows the GradeX sign-in page. At the top left is the GradeX logo. At the top right are links for "Sign In" and "Create Account". The main content area is a white box titled "Sign in". It contains two input fields: "Username" and "Password", both with placeholder text. Below these is a yellow "Sign in" button. At the bottom of the box is a link "New to our site? [Create an account](#)".

Fig: 9.3 Gradex Student Signin



The screenshot shows the GradeX "Create Account" page. At the top left is the GradeX logo. At the top right are links for "Sign In" and "Create Account". The main content area is a white box titled "Create Account". It contains four input fields: "Username", "Email", "Password", and "Confirm Password", each with a placeholder. Below these is a yellow "Create Account" button. At the bottom of the box is a link "Already have an account? [Sign in](#)".

Fig: 9.4 Gradex Student Sign up

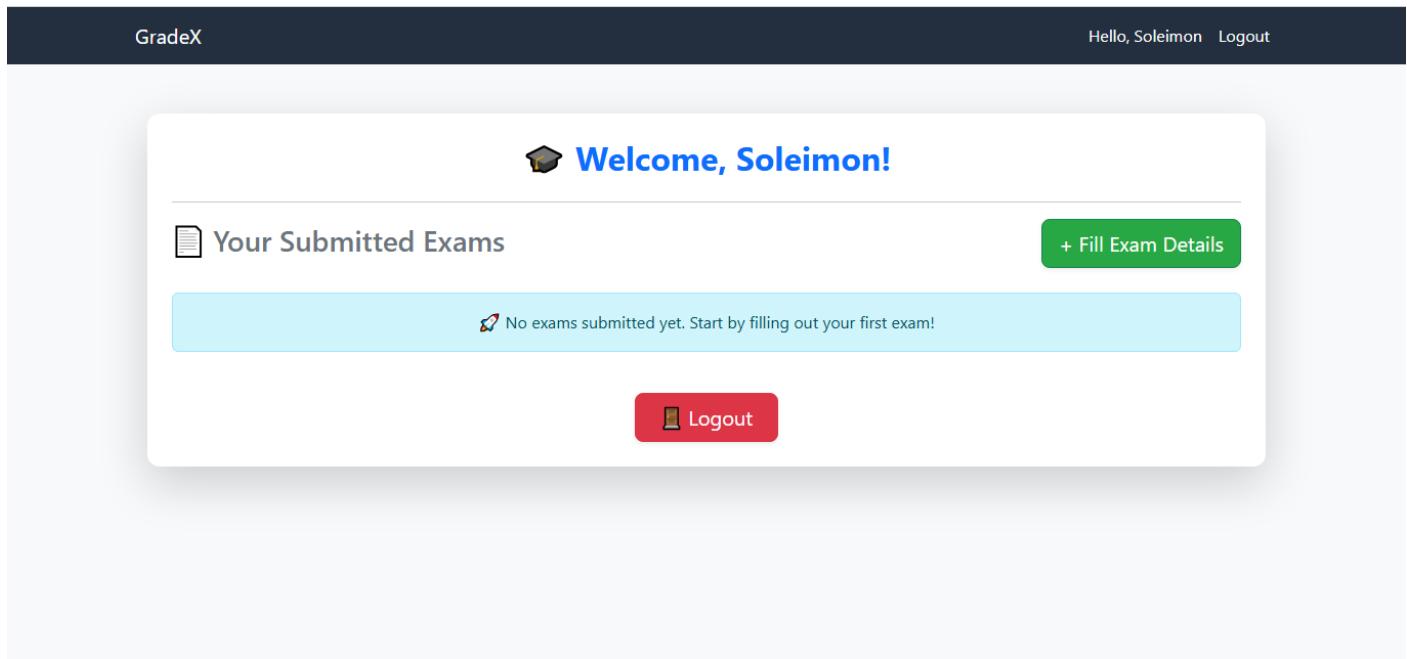


Fig: 9.5 Gradex Student Dashboard

The screenshot shows the "Fill Exam Details" page. At the top, there is a dark header bar with the text "GradeX" on the left and "Hello, Soleimon Logout" on the right. Below the header is a light gray content area. In the center, there is a blue pencil and paper icon followed by the text "Fill Exam Details". Below the title is a green rectangular box with the text "Login successful!". There are four input fields: "Subject Name" (with the value "Chemistry"), "Exam Type" (with the value "CAT 1" in a dropdown menu), "Year" (with the value "Fourth Year" in a dropdown menu), and "Staff Name" (with the value "Walter White"). At the bottom is a green button with the text "Submit Exam Details" and a checkmark icon.

Fig: 9.6 Gradex Student Exam Fill

The screenshot shows the GradeX student dashboard. At the top, it says "GradeX" on the left and "Hello, Soleimon Logout" on the right. Below that is a large white box with a header "Welcome, Soleimon!" featuring a graduation cap icon. On the left of the box is a "Your Submitted Exams" section with a document icon. On the right is a green button labeled "+ Fill Exam Details". Below this is a table with columns: Subject, Exam Type, Year, Staff Name, Status, and Actions. One row is visible: Chemistry, CAT 1, 4, Walter White, Pending (yellow button), and Awaiting Grading (grey button). At the bottom of the white box is a red "Logout" button with a bell icon.

Fig: 9.7 Gradex Student Awaiting Exam Status

The screenshot shows the GradeX teacher login page. At the top, it says "GradeX" on the left and "Sign In Create Account" on the right. Below that is a white login form titled "Teacher Login" with a teacher icon. It has two input fields: "Username" containing "kumar" and "Password" containing ".....". Below the password field is a blue "Login" button. At the bottom of the form is a grey "Back to Home" button.

Fig: 9.8 Gradex Teacher Login

The screenshot shows the 'Teacher Panel' interface. At the top right, there is a greeting 'Hello, kumar' and a 'Logout' link. On the left, a sidebar menu includes 'Dashboard', 'Create Exam', and 'Logout'. The main content area features a 'Welcome, kumar' message with a user icon. Below it is a section titled 'Created Exams' with a table:

Subject	Exam Type	Year	Actions
Python	CAT 1	Fourth Year	View Submissions
Finance	CAT 1	First Year	View Submissions

Fig: 9.9 Gradex Teacher Dashboard

The screenshot shows a 'Create Exam' form. At the top, it says 'Welcome, Teacher!'. The form fields are as follows:

- Subject Name:** Chemistry
- Exam Type:** CAT 1
- Year:** Fourth Year
- Staff Name:** Walter White
- Upload Question Paper (PDF):** Choose File: QUESTION_PAPER.pdf
- Upload Answer Key (PDF):** Choose File: answer_key.pdf

At the bottom are two buttons: a green 'Submit Exam' button with a checked checkbox and a grey 'Back to Dashboard' button.

Fig: 9.10 Gradex Teacher Exam Creation

The screenshot shows the GradeX Teacher Dashboard. On the left, there's a sidebar with icons for Dashboard, Create Exam, and Logout. The main area has a header with "GradeX", "Hello, kumar", and "Logout". Below the header, it says "Welcome, kumar" with a user icon. A green button on the right says "+ Create Exam". Underneath, a section titled "Created Exams" lists three exams in a table:

Subject	Exam Type	Year	Actions
Chemistry	CAT 1	Fourth Year	<button>View Submissions</button>
Python	CAT 1	Fourth Year	<button>View Submissions</button>
Finance	CAT 1	First Year	<button>View Submissions</button>

Fig: 9.11 Gradex Teacher Dashboard-Created Exams

The screenshot shows the GradeX Teacher Dashboard. The main area has a header with "GradeX", "Hello, kumar", and "Logout". Below the header, it says "Submissions for Chemistry (CAT 1)". A green box at the top indicates "Exam successfully created!". Below is a table of student submissions:

Student Name	Year	Answer Sheet	Upload Answer Sheet	Evaluate
kumar	4	<button>View</button>	<button>Uploaded</button>	<button>Evaluate</button>
kumaresan	4	<button>Not Uploaded</button>	Choose File No file chosen	<button>No Answer Sheet</button>
Lingesh	4	<button>Not Uploaded</button>	Choose File No file chosen	<button>No Answer Sheet</button>
Soleimon	4	<button>Not Uploaded</button>	Choose File No file chosen	<button>No Answer Sheet</button>

At the bottom, there are two buttons: "Upload Selected Files" and "Back to Dashboard".

Fig: 9.12 Gradex Teacher Dashboard-Exam Submissin List

GradeX

Hello, kumar Logout

Submissions for Chemistry (CAT 1)

✓ Answer sheet uploaded for Soleimon.

Student Name	Year	Answer Sheet	Upload Answer Sheet	Evaluate
kumar	4	View	Uploaded	Evaluate
kumaresan	4	Not Uploaded	Choose File No file chosen	No Answer Sheet
Lingesh	4	Not Uploaded	Choose File No file chosen	No Answer Sheet
Soleimon	4	View	Uploaded	Evaluate

[Upload Selected Files](#)

[Back to Dashboard](#)

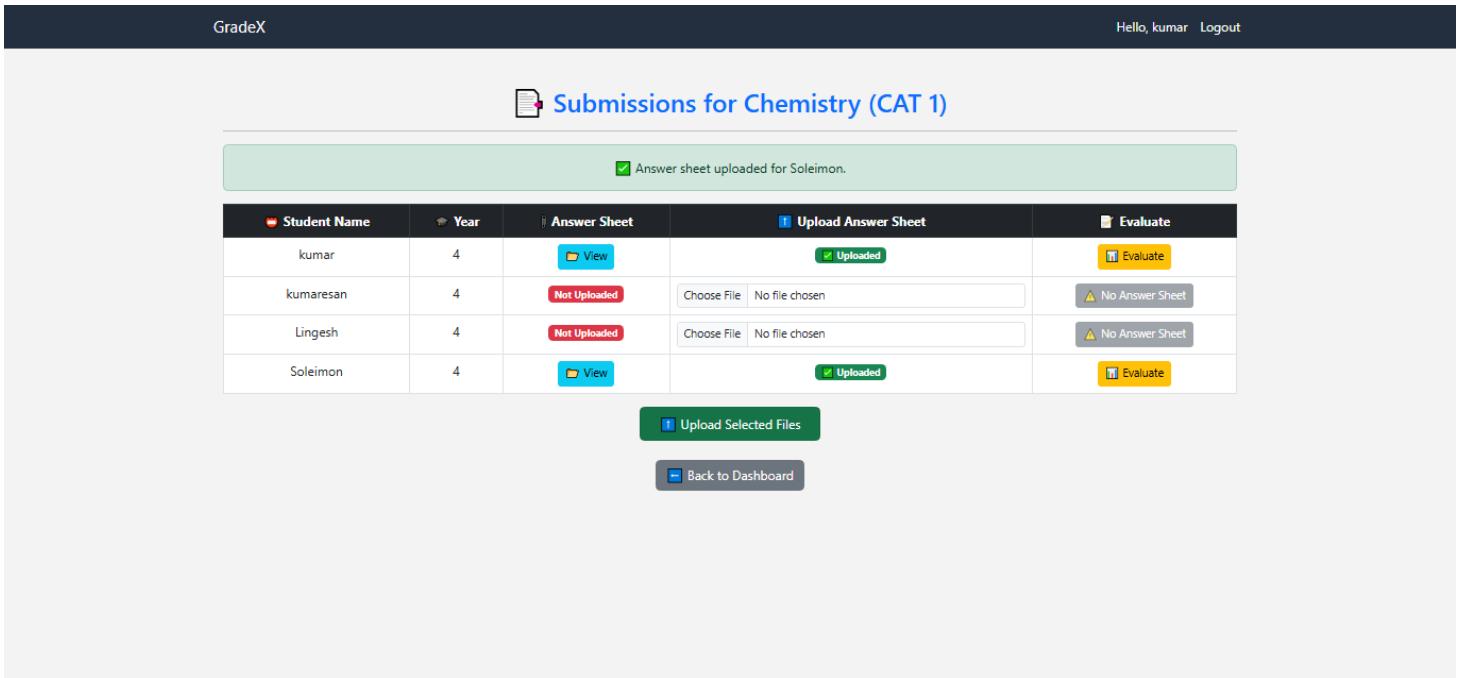


Fig: 9.13 Gradex Teacher Dashboard-Answer Sheet Upload

GradeX

Hello, kumar Logout

Submissions for Chemistry (CAT 1)

✓ Answer sheet uploaded for Soleimon.

Student Name	Year	Answer Sheet	Upload Answer Sheet	Evaluate
kumar	4	View	Uploaded	Evaluate
kumaresan	4	Not Uploaded	Choose File No file chosen	No Answer Sheet
Lingesh	4	Not Uploaded	Choose File No file chosen	No Answer Sheet
Soleimon	4	View	Uploaded	 Processing Evaluation...

[Upload Selected Files](#)

[Back to Dashboard](#)

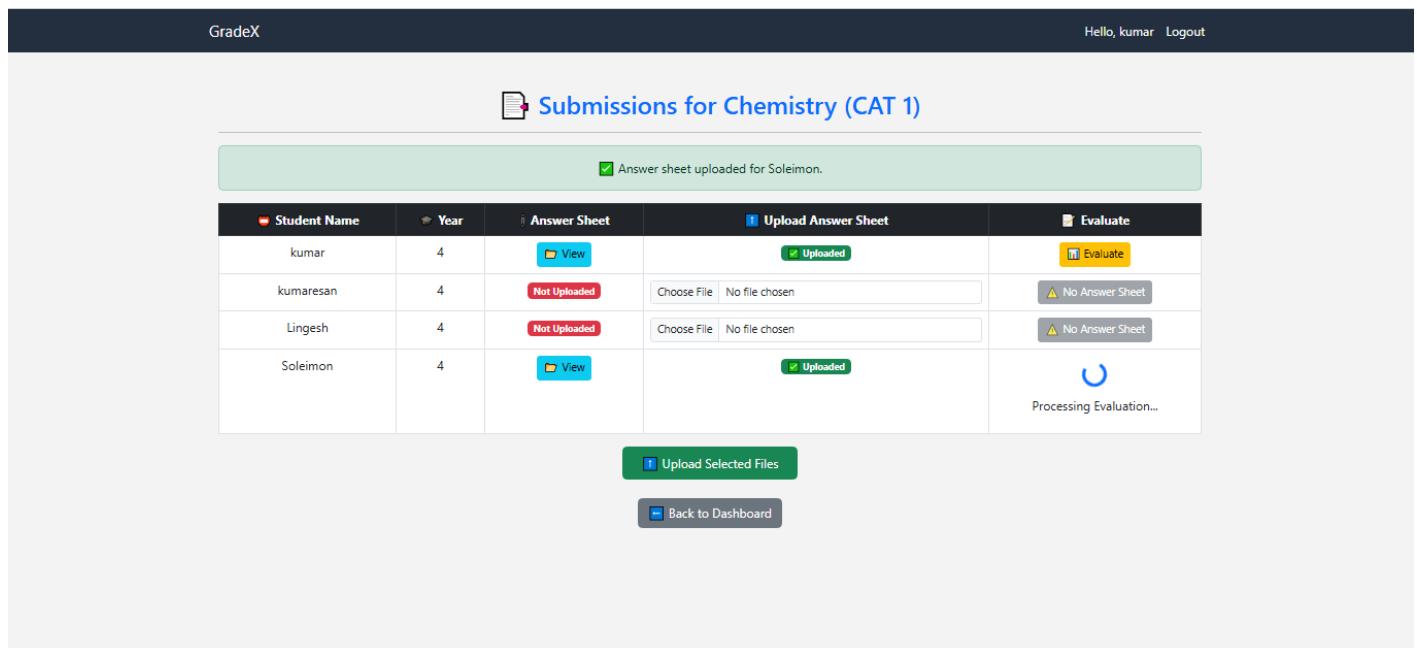


Fig: 9.14 Gradex Teacher Dashboard-Answer Sheet Evaluating

Exam Report

Exam Details

Exam Name: Python	Student Name: kumar
Date: 09 Mar 2025	Roll Number: 2
Exam Type: CAT 1	Year: 4

Q.No	Question	Student Answer	Correct Answer	Marks Awarded	Max Marks	Evaluation
Q1	Define Margin Market	<input checked="" type="radio"/> margin money. The initial deposit required by Lender or financial institution to allow the borrowing of funds for investment, typically used in stock trading.	<ul style="list-style-type: none"> Answer: Margin money is a percentage of the total transaction value that a borrower is required to pay upfront when taking a loan or making an investment. 	2	2	The student's answer correctly defines margin money as an initial deposit required for borrowing funds for investment.
Q2	Define equity funding	<input checked="" type="radio"/> Equity funding Raising Capital by chasing shares of the company to investors in exchange for ownership This can include personal savings angel investors or public stock offering	<ul style="list-style-type: none"> Answer: Raising capital by selling company stock (ownership shares). Investors provide funds in exchange for equity, becoming part-owners with a claim on future profits. 	2	2	The student's answer correctly defines equity funding as raising capital by selling shares for

Fig: 9.15 Gradex Teacher Dashboard-Results-1

		<p>association (AOA) and other forms required by the company registrar. These documents outline the company's operations and governance structure; iv; REGISTER WITH THE RELEVANT AUTHORITY: In many countries, businesses need to be registered with a government body like the Companies House, the Ministry of Corporate Affairs or the SEC. This typically involves submitting your documents and paying a registration fee; v; OBTAIN LICENSES AND PERMITS: Depending on the business type, you may need additional licenses (e.g., for health, safety, or food businesses); vi; REGISTER FOR TAXES: The company must be registered for tax purposes (e.g., getting a tax identification number or registering for VAT, depending on local laws); vii; OPEN A BANK ACCOUNT: After the company is registered, you'll likely need to open a corporate bank account to manage finances; viii; COMPLY WITH POST-REGISTRATION REQUIREMENTS: After registration, there may be ongoing compliance requirements, such as annual filings, audits, or maintaining a registered office. The specific steps may differ based on the jurisdiction, so it's important to refer to the local legal and regulatory authorities when starting the registration process.</p>	<p>company's objectives and structure. 2. Articles of Association (AoA): Rules and regulations for company management. 3. Director Identification Number (DIN): Required for company directors. 4. Digital Signature Certificate (DSC): Used for online registration and filing. 5. Address Proof & Identity Proof: Documents such as Aadhar card, passport, or electricity bill. Step 4: Register with Government Authorities • Apply for registration with the Ministry of Corporate Affairs (MCA) or relevant authority in your country. • Submit the required forms and pay the registration fee. Step 5: Obtain Business Licenses & Permits: Depending on the industry, additional licenses may be required: • GST/VAT registration (for tax purposes). • Trade licenses for retail businesses. • Industry-specific permits (e.g., food safety for restaurants). Step 6: Open a Business Bank Account • A corporate bank account is necessary for financial transactions. • Required documents: Registration certificate, PAN (if applicable), and identity proofs. Step 7: Register for Taxes • Tax Identification Number (TIN)/Goods & Services Tax (GST): For tax compliance. • Employer Identification Number (EIN): If hiring employees. Step 8: Compliance & Annual Filings • Companies must follow compliance requirements such as: • Annual financial statements submission. • Tax return filings. • Holding annual general meetings (AGM) for shareholders. Conclusion: Registering a company legally establishes it, providing credibility, limited liability protection, and access to funding. Proper documentation and adherence to legal regulations ensure smooth business operations.</p>		
Total Score: 40.0 / 50.0					

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





SEGi
UNIVERSITY

CERTIFICATE OF APPRECIATION

This is to certify that Mr.A.Mohamad Asick, Kings College of Engineering,
has presented a paper entitled on AI-Powered Answer Sheet Evaluation Using OCR and Large Language Models for Automated Grading in International Conference on “Sustainable Innovations in Management, Technology & Advanced Computing-SIMTAC'25” in association with SEGi University, KL, Malaysia organized by School of Management Studies & School of Computer Applications, on **4th April 2025** at Karpagam College of Engineering, Coimbatore.

Convenor
Director- MBA / MCA
KCE Coimbatore

Dr. Ratneswary Rasiah
Head - Learning & Quality assurance
SEGi University, Malaysia

Dr.V.Kumar Chinnaiyan
Principal
KCE Coimbatore

COURSE CERTIFICATES



Certificate no: UC-3308c41b-93cb-485a-9ee3-755110f8ef37
Certificate url: ude.my/UC-3308c41b-93cb-485a-9ee3-755110f8ef37
Reference Number: 0008

CERTIFICATE OF COMPLETION

PyTorch Ultimate 2024: From Basics to Cutting-Edge

Instructors **Bert Gollnick**

Mohamed Asick A

Date 12 Feb 2025



Certificate no: UC-3308c41b-93cb-485a-9ee3-481000f8ef106
Certificate url: ude.my/UC-3308c41b-93cb-485a-9ee3-481000f8ef106
Reference Number: 0003

CERTIFICATE OF COMPLETION

PyTorch Ultimate 2024: From Basics to Cutting-Edge

Instructors **Bert Gollnick**

Lingesh R S

Date 12 Feb 2025



Certificate no: UC-9ae414c0-8f47-4026-Bedf-e1Qc6fc4b725
Certificate url: ude.my/UC-9ae414c0-8f47-4026-Bedf-e1Qc6fc4b725
Reference Number: 0004

CERTIFICATE OF COMPLETION

PyTorch Ultimate 2024: From Basics to Cutting-Edge

Instructors **Bert Gollnick**

Kumaresan

Date **12 Feb 2025**
Length **19 total hours**