GenAI Curriculum Generator

AI-Powered Educational Curriculum Design with IBM Granite 3.3 2B

Project Description

GenAI Curriculum Generator is an intelligent curriculum design platform that leverages IBM's Granite 3.3 2B AI model (via Ollama) to provide comprehensive educational curriculum generation and planning recommendations. The platform addresses the challenge of accurate curriculum design by delivering AI-powered insights, detailed course structures, topic recommendations, and semester-wise syllabi.

Using Granite 3.3 2B's advanced language capabilities, the system analyzes educational parameters (skill, education level, number of semesters, weekly hours, industry focus), generates accurate course names, learning topics, course descriptions, and detailed curricula tailored to individual educational specifications. The system ensures fast response times through local Ollama deployment while maintaining high-quality curriculum advice through sophisticated AI analysis and educational algorithms.

GenAI Curriculum Generator transforms curriculum design into an intelligent, user-friendly experience through its modern interface, comprehensive feature set, and AI-powered analysis that provides personalized educational guidance considering skill complexity, learning level progression, and industry relevance.

Scenarios

Scenario 1 – Machine Learning Masters Program:
A university designs a 2-year Master's program in Machine Learning using the GenAI Curriculum Generator by entering parameters such as ML skill set, 4 semesters, 20–25 weekly hours, and AI industry focus. The Granite 3.3 2B model creates a logical progression of courses from basic machine learning to advanced topics like NLP and Reinforcement Learning. Each module contains structured topics such as CNNs, backpropagation, and optimization techniques. The system assigns 4 credits per course, builds semester-wise schedules, and defines capstone guidelines. Finally, a professional, publication-ready PDF syllabus is generated for admissions and student onboarding.

Scenario 2 – Full Stack Web Development Certification:
A bootcamp develops a 6-month Full Stack Web Development certification by specifying certification level, 2 semesters, 30 weekly hours, and web industry focus. The Granite 3.3 2B model produces a beginner-to-advanced curriculum covering HTML, CSS, JavaScript, React, Node.js, and SQL. Each course includes practical, industry-oriented topics such as REST APIs, state management, and deployment strategies. The system organizes the learning path into a concise semester-wise schedule with project-based outcomes. An exportable curriculum document is delivered for immediate marketing.

Scenario 3 – Python for Data Science Program:
An online platform creates a 1-year Python for Data Science program by providing inputs like Python skill, BTech level, 2 semesters, and data science industry focus. The AI model generates a balanced curriculum progressing from Python fundamentals to advanced data science applications. Courses include NumPy, Pandas, statistics, web scraping, and visualization with Matplotlib and Seaborn. Practical projects reinforce real-world data science skills. A professional PDF syllabus is produced to support student recruitment and promotions.

Scenario 4 – Artificial Intelligence Diploma Program:
A technical institute launches a 3-year full-time AI diploma by entering AI skill, diploma level, 6 semesters, and AI R&D focus. The Granite 3.3 2B model generates an 18-course roadmap progressing from mathematical foundations to Machine Learning, Deep Learning, NLP, Computer Vision, and Reinforcement Learning. Each semester contains well-paced courses with 5–7 relevant topics. Research-oriented learning outcomes and capstone preparation are included. The institute receives a complete, accreditation-ready syllabus for official approval.

Architecture Overview

GenAI Curriculum Generator is built as a modular platform combining Flask backend with IBM Granite 3.3 2B AI model for intelligent curriculum design. The architecture prioritizes accuracy, speed, and user experience by leveraging local AI inference and sophisticated curriculum generation algorithms.

Core Technologies

- **Flask**: Lightweight Python web framework for routing and request processing
- **IBM Granite 3.3 2B**: Local AI inference via Ollama for intelligent curriculum generation
- **Ollama**: Local LLM runtime for Granite model execution
- **Python Algorithms**: Curriculum structure and validation logic
- **HTML5/CSS3/JavaScript**: Modern, responsive frontend with smooth animations
- **ReportLab**: Professional PDF document generation
- **Local Deployment**: No cloud API keys required, completely self-contained

Pre-requisites

Software Requirements

- **Python 3.8+**: Download from python.org
- **Ollama**: Download from ollama.ai
- **Granite 3.3 2B Model**: Pre-downloaded via Ollama (ollama pull granite:3.3-2b)
- **Git**: Download from git-scm.com
- **Code Editor**: VS Code, PyCharm, or any preferred IDE

Knowledge Prerequisites

- **Python Basics**: Functions, classes, OOP, exception handling
- **Flask Framework**: Routing, request handling, JSON responses
- **HTML/CSS**: Responsive design, Flexbox/Grid layouts
- **JavaScript**: DOM manipulation, Fetch API, async/await
- **Educational Domain**: Basic understanding of curriculum design, learning outcomes
- **Mathematics**: Basic calculations, cost analysis

Hardware Requirements

- **Processor**: Intel i5/AMD Ryzen 5 or better
- **RAM**: Minimum 8GB (16GB recommended for Ollama)
- **Storage**: 10GB free space (for Granite 3.3 2B model)
- **Internet**: Required for initial setup only

Project Workflow

Phase 1: Environment Setup & Ollama Configuration

Establish local AI infrastructure and validate model connectivity

Activity 1.1: Set up Python Environment

- Create project directory structure
- Initialize virtual environment
- Create requirements.txt with dependencies (Flask, requests, reportlab)
- Install all Python packages

Activity 1.2: Configure Ollama & Granite Model

- Download and install Ollama
- Pull Granite 3.3 2B model: ollama pull granite:3.3-2b
- Verify model accessibility on port 11434
- Test local inference capabilities with sample prompts

Activity 1.3: Validate System Configuration

- Test Flask application startup
- Verify Ollama API connectivity
- Test model response times (target: < 60 seconds)
- Document configuration steps for reference

Phase 2: Core Backend Development

Build Flask infrastructure and curriculum generation logic

Activity 2.1: Set up Flask Application Structure

- Initialize Flask app with proper architecture
- Create route structure (/api/generate-curriculum, /api/download-pdf, /health)
- Implement CORS headers for frontend communication
- Set up request/response handling with JSON

Activity 2.2: Implement Ollama/Granite API Integration

- Create Ollama API communication module with error handling
- Implement prompt engineering for curriculum generation
- Add JSON response parsing and validation
- Handle API timeouts and implement fallback system

Activity 2.3: Build Curriculum Generation Engine

- Create single-batch curriculum generator function
- Implement course name generation logic using AI
- Build topic extraction and formatting system
- Develop curriculum structure builder (semesters, courses, credits)

Activity 2.4: Develop PDF Export Engine

- Create ReportLab PDF generation system
- Implement professional table formatting
- Add semester-wise curriculum layout
- Build topic wrapping and text formatting

Activity 2.5: Build JSON Export Functionality

- Create JSON serialization system
- Implement data validation
- Add file download handling
- Create backup data structure

Phase 3: Frontend Development

Create modern, responsive user interface

Activity 3.1: Design Responsive HTML Structure

- Create semantic HTML5 layout
- Design form sections for curriculum input
- Build results display templates
- Implement semester accordion structure

Activity 3.2: Develop CSS Styling & Animations

- Implement modern gradient color scheme
- Create responsive grid layouts
- Add smooth transitions and animations
- Build mobile-responsive design
- Create topic tag styling

Activity 3.3: Create Interactive JavaScript Logic

- Implement form validation
- Build real-time curriculum display
- Create semester accordion toggle
- Add PDF/JSON download functionality
- Implement loading and error states

Activity 3.4: Build Results Display Components

- Create dynamic course cards
- Implement semester containers
- Build topic tag visualization
- Create summary card display
- Implement capstone project display

Phase 4: Integration & Testing

Integrate all components and comprehensive testing

Activity 4.1: End-to-End Integration

- Connect frontend to backend API
- Test API communication flow
- Validate data flow between components
- Test error handling and fallback system

Activity 4.2: Functional Testing

- Test with various skills (ML, Web, Python, Data Science, Java)
- Validate different education levels (Diploma, BTech, Masters, Certification)
- Test all semester durations (2, 4, 6, 8 semesters)
- Verify PDF generation accuracy and formatting
- Test JSON export functionality

Activity 4.3: Performance Testing

- Test Ollama response times (target: 15-30 seconds)
- Profile API call performance
- Optimize batch curriculum generation
- Load test with multiple concurrent requests

Phase 5: Deployment & Documentation

Deploy and document the system

Activity 5.1: Local Deployment

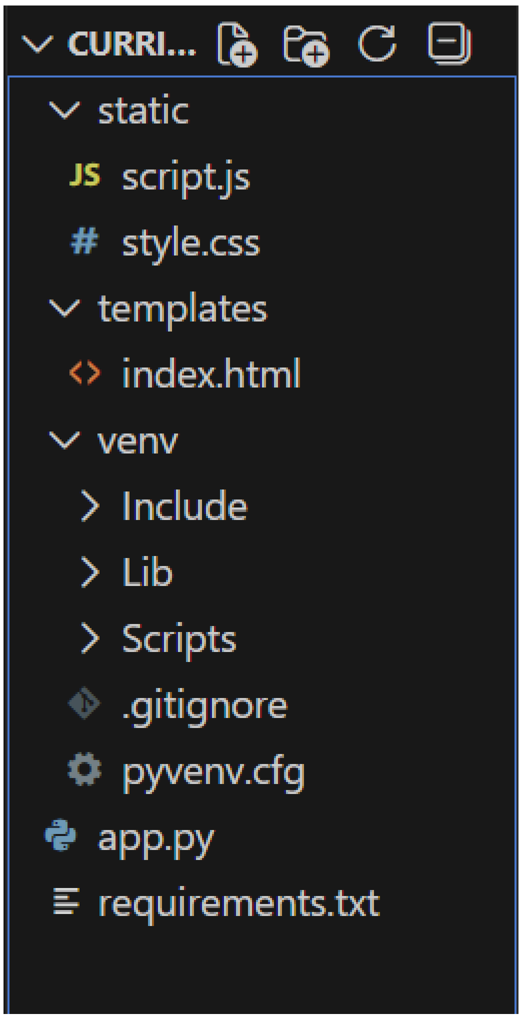
- Set up production environment
- Configure logging and error handling
- Document deployment steps
- Create user guide with screenshots

Activity 5.2: Create Comprehensive Documentation

- Write API documentation
- Create user manual with examples
- Document configuration steps
- Create troubleshooting guide

Technical Architecture

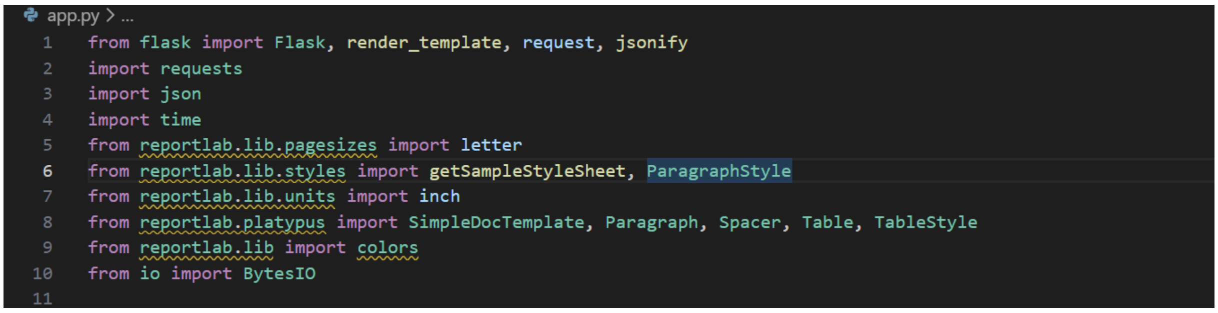
Project Structure


```
CURRI...
  static
    script.js
    style.css
  templates
    index.html
  venv
    Include
    Lib
    Scripts
    .gitignore
    pyvenv.cfg
  app.py
  requirements.txt
```

MILESTONE 1: Environment Setup & Ollama Configuration

This milestone establishes the complete local environment by installing Python, Flask, ReportLab, and Ollama with the IBM Granite 3.3 2B model. It verifies model connectivity, API response time, and system readiness for curriculum generation.

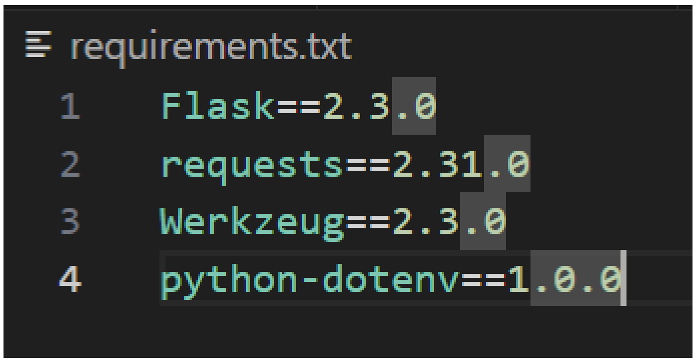
Activity 1.1: Install Dependencies & Setup Environment



app.py > ...

```python
from flask import Flask, render_template, request, jsonify
import requests
import json
import time
from reportlab.lib.pagesizes import letter
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.lib.units import inch
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Table, TableStyle
from reportlab.lib import colors
from io import BytesIO
```

Step 2: Create requirements.txt



requirements.txt

```
Flask==2.3.0
requests==2.31.0
Werkzeug==2.3.0
python-dotenv==1.0.0
```

Step 3: Create Virtual Environment

python -m venv venv

# Activate virtual environment
# Windows:
venv\Scripts\activate
# Mac/Linux:
source venv/bin/activate

Step 4: Install Dependencies

```
pip install -r requirements.txt
```

Activity 1.2: Configure & Verify Ollama Setup

Install Ollama:

1. Download Ollama from ollama.ai
2. Follow installation instructions for your OS
3. Start Ollama service

Pull Granite Model:

```
ollama pull granite:3.3-2b
```

Verify Installation:

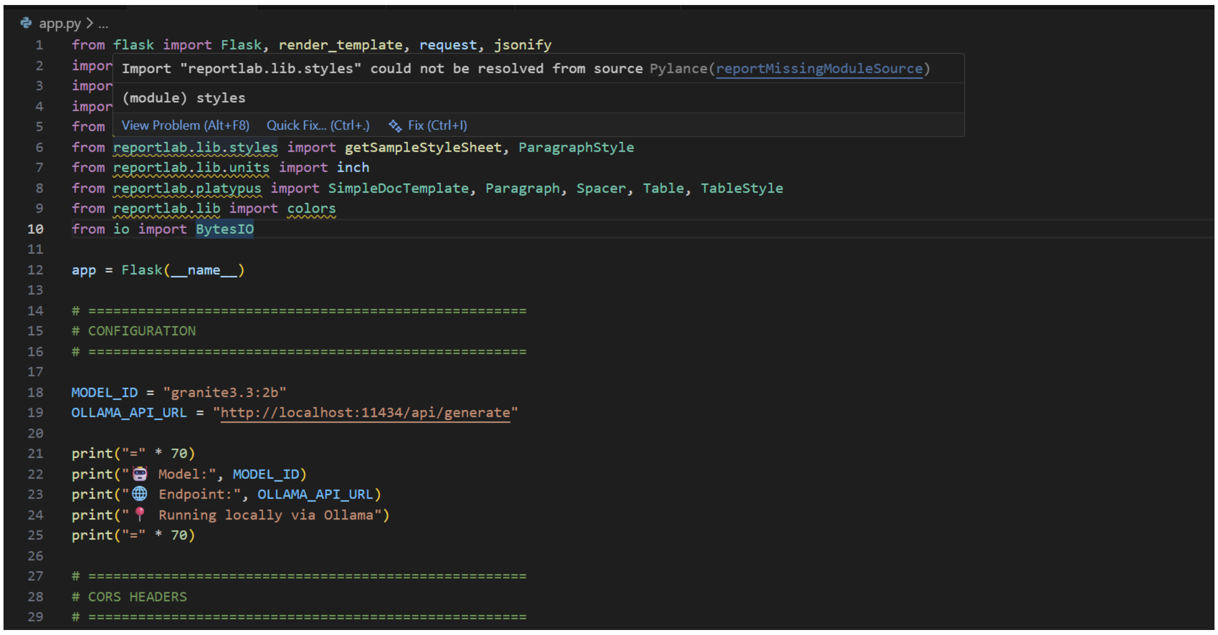
```
ollama list
# Output should show: granite:3.3-2b
```

MILESTONE 2: Core Backend Development

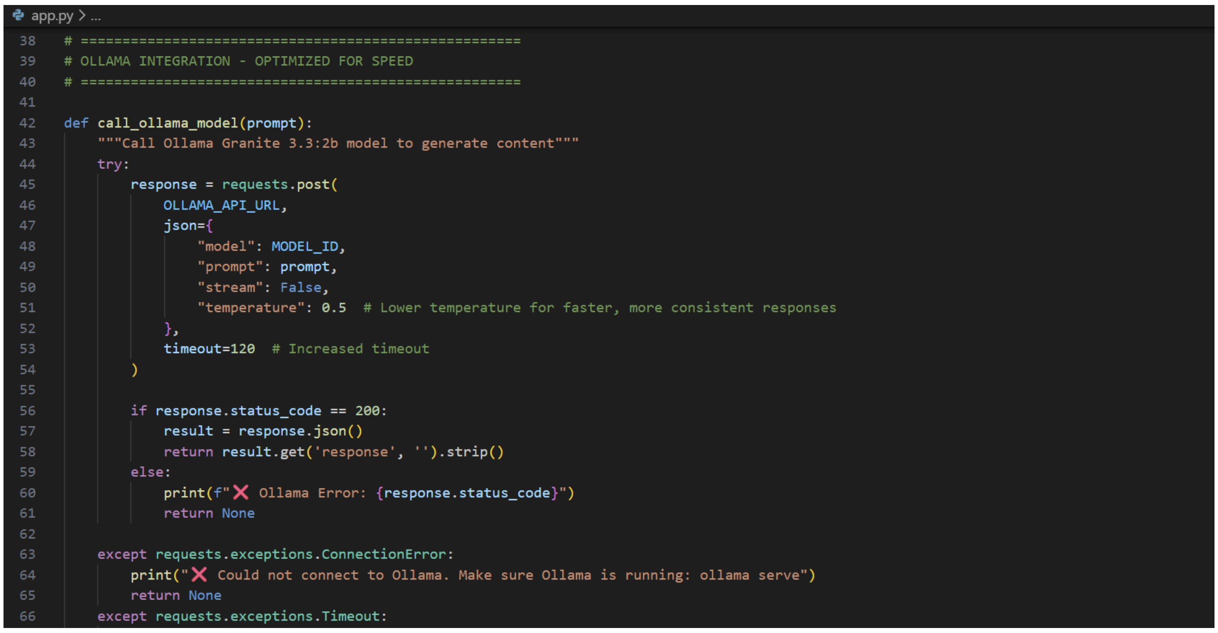
This phase builds the Flask backend, integrates the Ollama–Granite API, and implements the curriculum generation engine. It enables automated course creation, topic extraction, semester structuring, and PDF/JSON export functionality.

Activity 2.1: Flask Application Setup

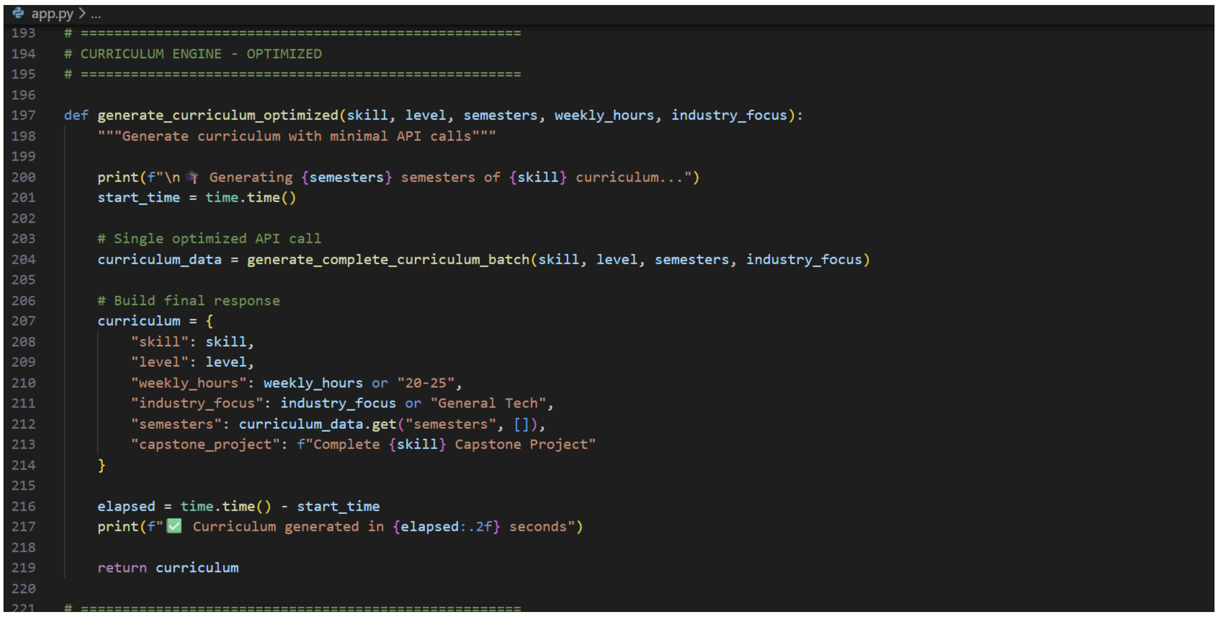
app.py:



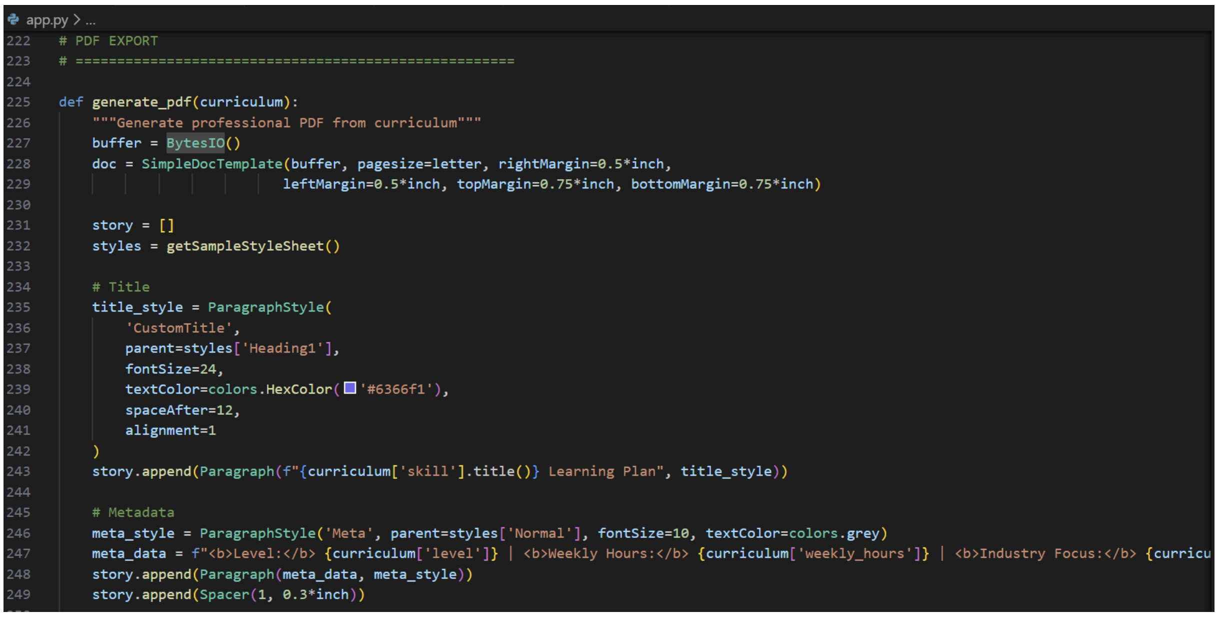
```
app.py > ...
from flask import Flask, render_template, request, jsonify
impor Import "reportlab.lib.styles" could not be resolved from source Pylance(reportMissingModuleSource)
impor
impor (module) styles
from  View Problem (Alt+F8)   Quick Fix... (Ctrl+.)   Fix (Ctrl+I)
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.lib.units import inch
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Table, TableStyle
from reportlab.lib import colors
from io import BytesIO

app = Flask(__name__)

# =====================================================
# CONFIGURATION
# =====================================================

MODEL_ID = "granite3.3:2b"
OLLAMA_API_URL = "http://localhost:11434/api/generate"

print("=" * 70)
print("🤖 Model:", MODEL_ID)
print("🌐 Endpoint:", OLLAMA_API_URL)
print("📍 Running locally via Ollama")
print("=" * 70)

# =====================================================
# CORS HEADERS
# =====================================================
```



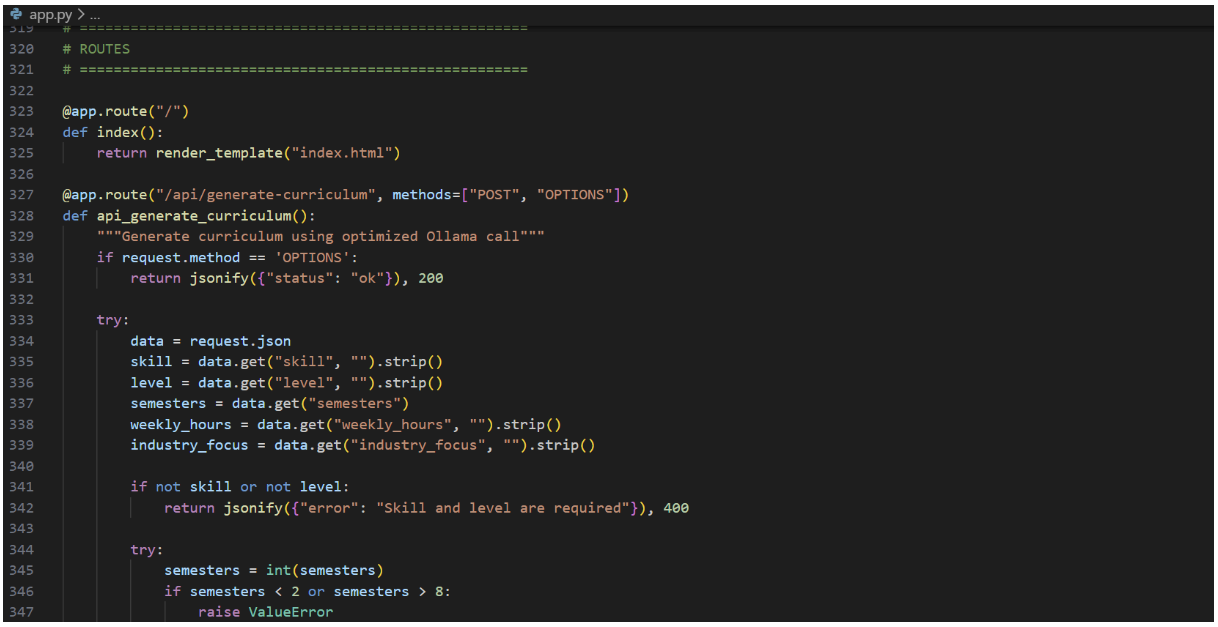
```
app.py > ...
# =====================================================
# OLLAMA INTEGRATION - OPTIMIZED FOR SPEED
# =====================================================

def call_ollama_model(prompt):
    """Call Ollama Granite 3.3:2b model to generate content"""
    try:
        response = requests.post(
            OLLAMA_API_URL,
            json={
                "model": MODEL_ID,
                "prompt": prompt,
                "stream": False,
                "temperature": 0.5  # Lower temperature for faster, more consistent responses
            },
            timeout=120  # Increased timeout
        )

        if response.status_code == 200:
            result = response.json()
            return result.get('response', '').strip()
        else:
            print(f"❌ Ollama Error: {response.status_code}")
            return None

    except requests.exceptions.ConnectionError:
        print("❌ Could not connect to Ollama. Make sure Ollama is running: ollama serve")
        return None
    except requests.exceptions.Timeout:
```



```python
# =====================================================
# CURRICULUM ENGINE - OPTIMIZED
# =====================================================

def generate_curriculum_optimized(skill, level, semesters, weekly_hours, industry_focus):
    """Generate curriculum with minimal API calls"""

    print(f"\n🎓 Generating {semesters} semesters of {skill} curriculum...")
    start_time = time.time()

    # Single optimized API call
    curriculum_data = generate_complete_curriculum_batch(skill, level, semesters, industry_focus)

    # Build final response
    curriculum = {
        "skill": skill,
        "level": level,
        "weekly_hours": weekly_hours or "20-25",
        "industry_focus": industry_focus or "General Tech",
        "semesters": curriculum_data.get("semesters", []),
        "capstone_project": f"Complete {skill} Capstone Project"
    }

    elapsed = time.time() - start_time
    print(f"✅ Curriculum generated in {elapsed:.2f} seconds")

    return curriculum

# =====================================================
```

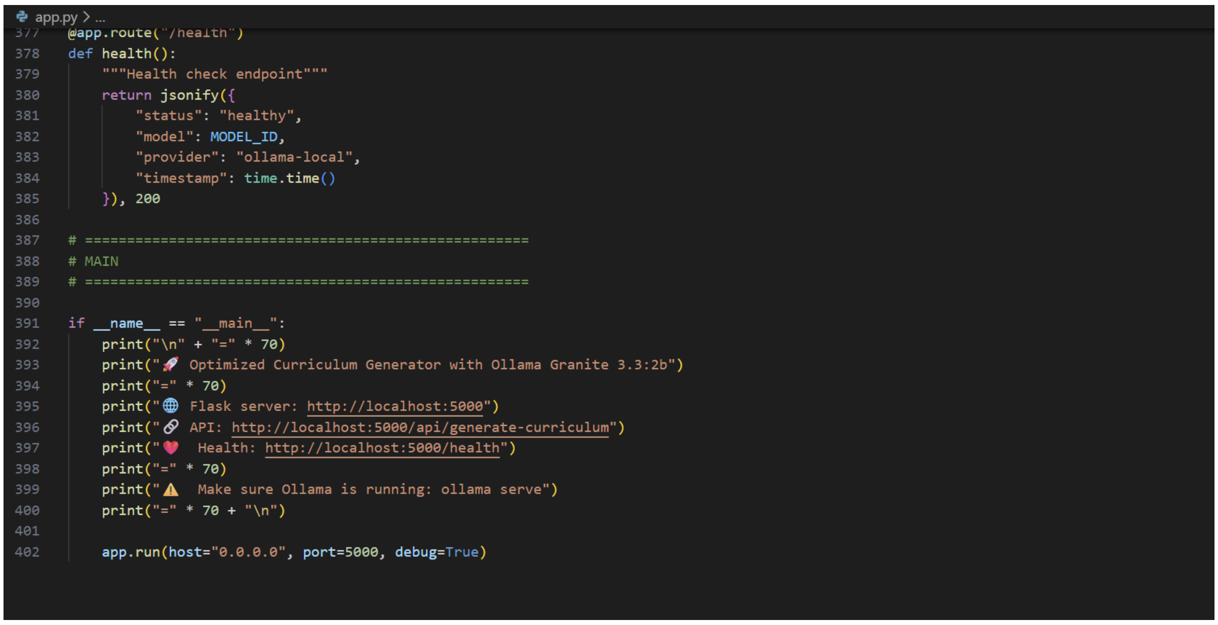


```python
# PDF EXPORT
# =====================================================

def generate_pdf(curriculum):
    """Generate professional PDF from curriculum"""
    buffer = BytesIO()
    doc = SimpleDocTemplate(buffer, pagesize=letter, rightMargin=0.5*inch,
                            leftMargin=0.5*inch, topMargin=0.75*inch, bottomMargin=0.75*inch)

    story = []
    styles = getSampleStyleSheet()

    # Title
    title_style = ParagraphStyle(
        'CustomTitle',
        parent=styles['Heading1'],
        fontSize=24,
        textColor=colors.HexColor('#6366f1'),
        spaceAfter=12,
        alignment=1
    )
    story.append(Paragraph(f"{curriculum['skill'].title()} Learning Plan", title_style))

    # Metadata
    meta_style = ParagraphStyle('Meta', parent=styles['Normal'], fontSize=10, textColor=colors.grey)
    meta_data = f"<b>Level:</b> {curriculum['level']} | <b>Weekly Hours:</b> {curriculum['weekly_hours']} | <b>Industry Focus:</b> {curricu
    story.append(Paragraph(meta_data, meta_style))
    story.append(Spacer(1, 0.3*inch))
```



app.py > ...

```python
# =====================================================
# ROUTES
# =====================================================

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/api/generate-curriculum", methods=["POST", "OPTIONS"])
def api_generate_curriculum():
    """Generate curriculum using optimized Ollama call"""
    if request.method == 'OPTIONS':
        return jsonify({"status": "ok"}), 200

    try:
        data = request.json
        skill = data.get("skill", "").strip()
        level = data.get("level", "").strip()
        semesters = data.get("semesters")
        weekly_hours = data.get("weekly_hours", "").strip()
        industry_focus = data.get("industry_focus", "").strip()

        if not skill or not level:
            return jsonify({"error": "Skill and level are required"}), 400

        try:
            semesters = int(semesters)
            if semesters < 2 or semesters > 8:
                raise ValueError
```



app.py > ...

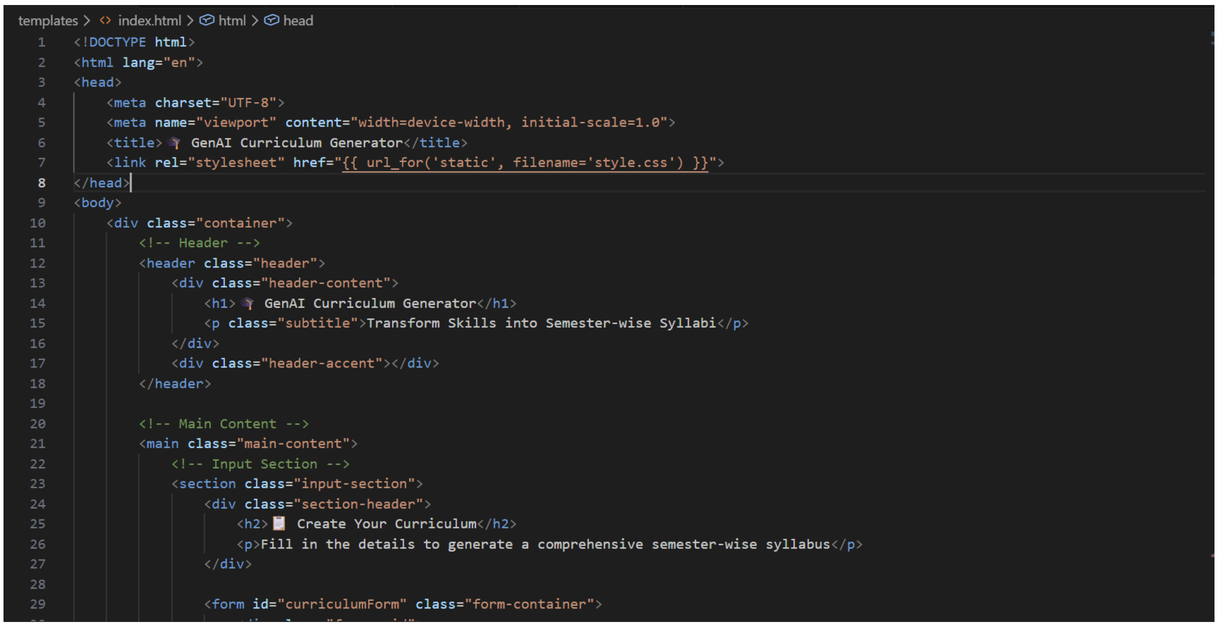
```python
@app.route("/health")
def health():
    """Health check endpoint"""
    return jsonify({
        "status": "healthy",
        "model": MODEL_ID,
        "provider": "ollama-local",
        "timestamp": time.time()
    }), 200

# =====================================================
# MAIN
# =====================================================

if __name__ == "__main__":
    print("\n" + "=" * 70)
    print("🚀 Optimized Curriculum Generator with Ollama Granite 3.3:2b")
    print("=" * 70)
    print("🌐 Flask server: http://localhost:5000")
    print("🔗 API: http://localhost:5000/api/generate-curriculum")
    print("❤️  Health: http://localhost:5000/health")
    print("=" * 70)
    print("⚠️  Make sure Ollama is running: ollama serve")
    print("=" * 70 + "\n")

    app.run(host="0.0.0.0", port=5000, debug=True)
```

MILESTONE 3: Frontend Development

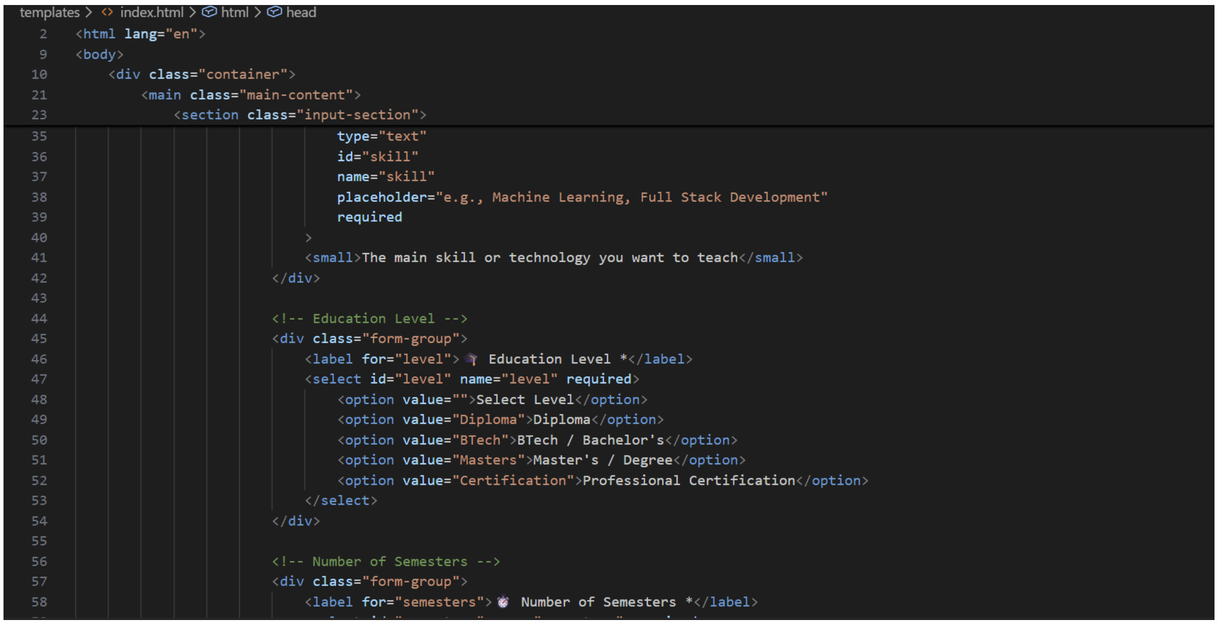
This milestone designs a modern, responsive interface for curriculum input and visualization using HTML, CSS, and JavaScript. It allows users to view generated courses, semesters, capstone details, and download PDF or JSON outputs.

Create modern, responsive user interface

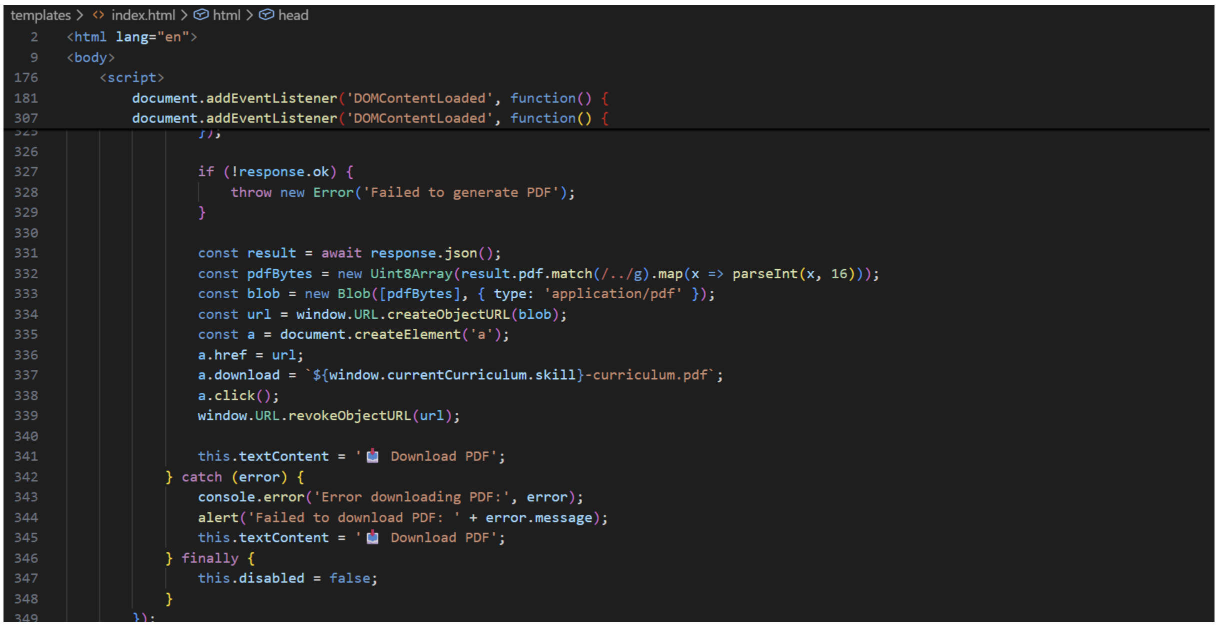
Create templates/index.html:



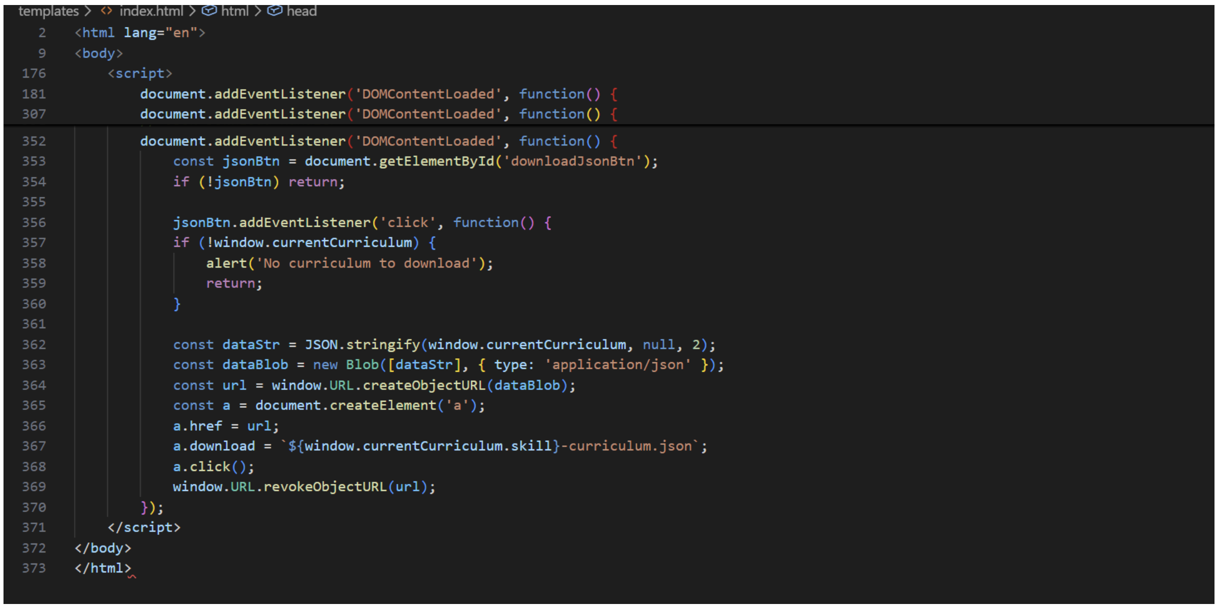
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>🎓 GenAI Curriculum Generator</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <!-- Header -->
        <header class="header">
            <div class="header-content">
                <h1>🎓 GenAI Curriculum Generator</h1>
                <p class="subtitle">Transform Skills into Semester-wise Syllabi</p>
            </div>
            <div class="header-accent"></div>
        </header>

        <!-- Main Content -->
        <main class="main-content">
            <!-- Input Section -->
            <section class="input-section">
                <div class="section-header">
                    <h2>📋 Create Your Curriculum</h2>
                    <p>Fill in the details to generate a comprehensive semester-wise syllabus</p>
                </div>

                <form id="curriculumForm" class="form-container">
```



```html
<html lang="en">
<body>
    <div class="container">
        <main class="main-content">
            <section class="input-section">
                                type="text"
                                id="skill"
                                name="skill"
                                placeholder="e.g., Machine Learning, Full Stack Development"
                                required
                            >
                            <small>The main skill or technology you want to teach</small>
                        </div>

                        <!-- Education Level -->
                        <div class="form-group">
                            <label for="level">🎓 Education Level *</label>
                            <select id="level" name="level" required>
                                <option value="">Select Level</option>
                                <option value="Diploma">Diploma</option>
                                <option value="BTech">BTech / Bachelor's</option>
                                <option value="Masters">Master's / Degree</option>
                                <option value="Certification">Professional Certification</option>
                            </select>
                        </div>

                        <!-- Number of Semesters -->
                        <div class="form-group">
                            <label for="semesters">⏰ Number of Semesters *</label>
```



```
<html lang="en">
<body>
    <script>
        document.addEventListener('DOMContentLoaded', function() {
        document.addEventListener('DOMContentLoaded', function() {
                });

                if (!response.ok) {
                    throw new Error('Failed to generate PDF');
                }

                const result = await response.json();
                const pdfBytes = new Uint8Array(result.pdf.match(/../g).map(x => parseInt(x, 16)));
                const blob = new Blob([pdfBytes], { type: 'application/pdf' });
                const url = window.URL.createObjectURL(blob);
                const a = document.createElement('a');
                a.href = url;
                a.download = `${window.currentCurriculum.skill}-curriculum.pdf`;
                a.click();
                window.URL.revokeObjectURL(url);

                this.textContent = '📥 Download PDF';
            } catch (error) {
                console.error('Error downloading PDF:', error);
                alert('Failed to download PDF: ' + error.message);
                this.textContent = '📥 Download PDF';
            } finally {
                this.disabled = false;
            }
        });
```



```
<html lang="en">
<body>
    <script>
        document.addEventListener('DOMContentLoaded', function() {
        document.addEventListener('DOMContentLoaded', function() {
        document.addEventListener('DOMContentLoaded', function() {
            const jsonBtn = document.getElementById('downloadJsonBtn');
            if (!jsonBtn) return;

            jsonBtn.addEventListener('click', function() {
            if (!window.currentCurriculum) {
                alert('No curriculum to download');
                return;
            }

            const dataStr = JSON.stringify(window.currentCurriculum, null, 2);
            const dataBlob = new Blob([dataStr], { type: 'application/json' });
            const url = window.URL.createObjectURL(dataBlob);
            const a = document.createElement('a');
            a.href = url;
            a.download = `${window.currentCurriculum.skill}-curriculum.json`;
            a.click();
            window.URL.revokeObjectURL(url);
        });
    </script>
</body>
</html>
```

MILESTONE 4: Deployment

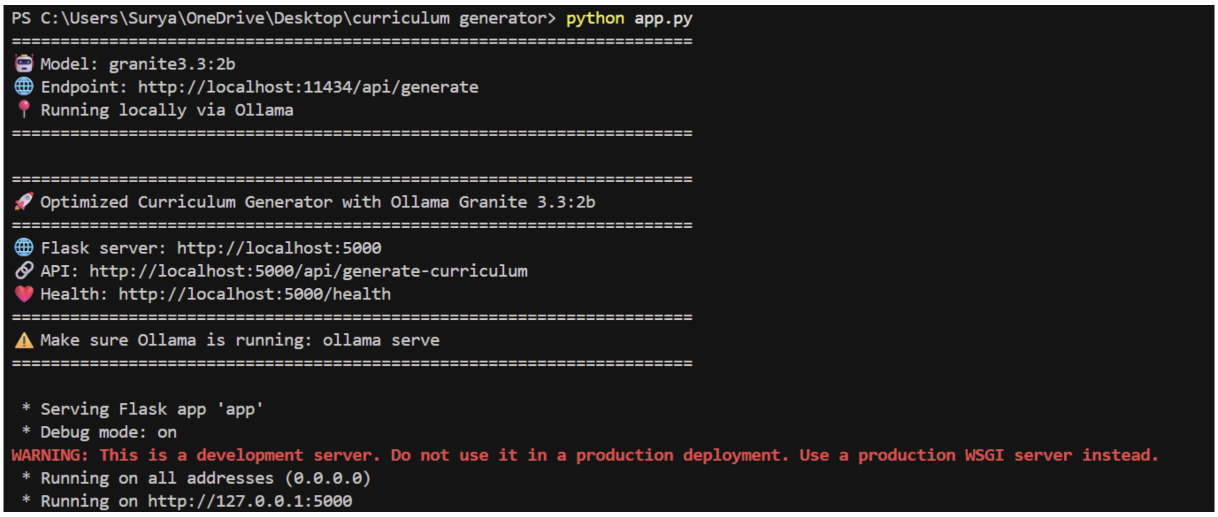
This phase focuses on running the complete application locally by starting the Ollama service and Flask server. It ensures the system is accessible through a browser with stable performance.

Activity 4.1: Local Deployment

Step 1: Start Ollama Service

ollama serve

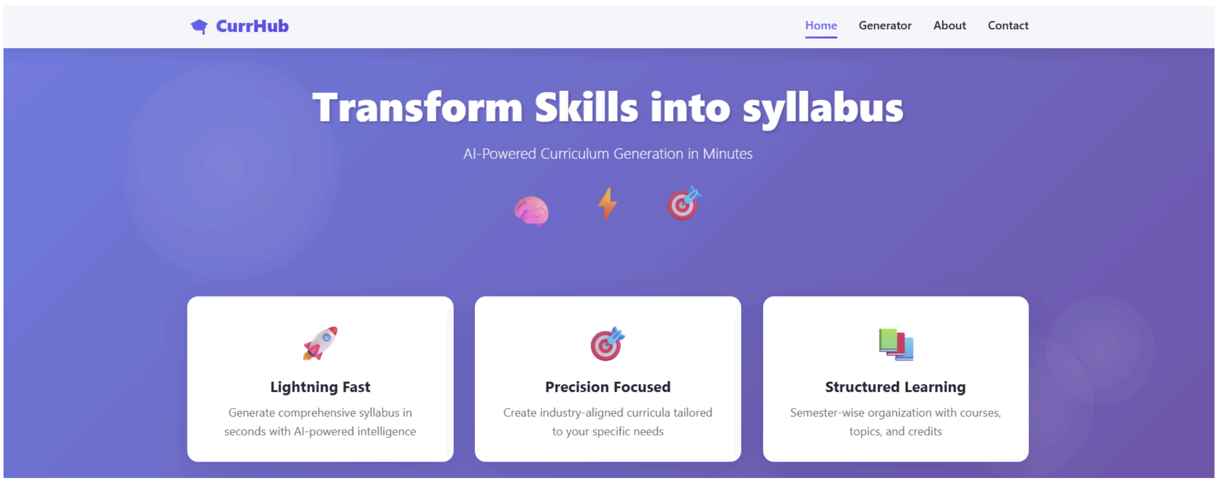
Step 2: Run Flask Application



```
PS C:\Users\Surya\OneDrive\Desktop\curriculum generator> python app.py
======================================================================
🤖 Model: granite3.3:2b
🌐 Endpoint: http://localhost:11434/api/generate
📍 Running locally via Ollama
======================================================================

======================================================================
🚀 Optimized Curriculum Generator with Ollama Granite 3.3:2b
======================================================================
🌐 Flask server: http://localhost:5000
🔗 API: http://localhost:5000/api/generate-curriculum
❤️ Health: http://localhost:5000/health
======================================================================
⚠️ Make sure Ollama is running: ollama serve
======================================================================

 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
```

Step 3: Access Application

http://localhost:5000



CurrHub

Home Generator About Contact

Transform Skills into syllabus

AI-Powered Curriculum Generation in Minutes

Lightning Fast

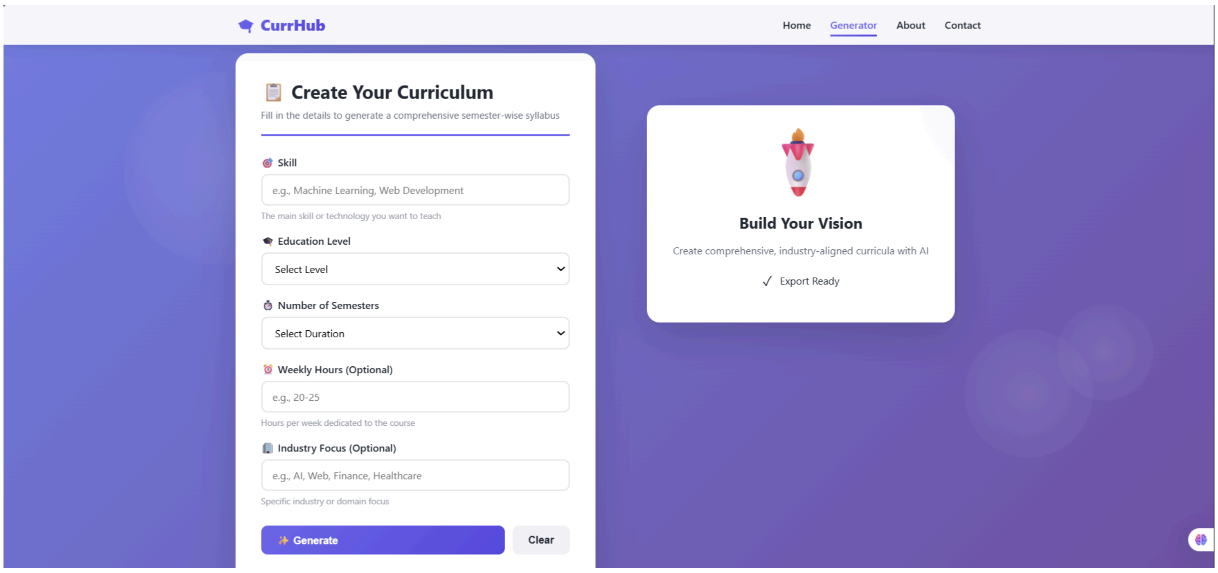
Generate comprehensive syllabus in seconds with AI-powered intelligence

Precision Focused

Create industry-aligned curricula tailored to your specific needs

Structured Learning

Semester-wise organization with courses, topics, and credits



CurrHub

Home Generator About Contact

Create Your Curriculum

Fill in the details to generate a comprehensive semester-wise syllabus

Skill

e.g., Machine Learning, Web Development

The main skill or technology you want to teach

Education Level

Select Level

Number of Semesters

Select Duration

Weekly Hours (Optional)

e.g., 20-25

Hours per week dedicated to the course

Industry Focus (Optional)

e.g., AI, Web, Finance, Healthcare

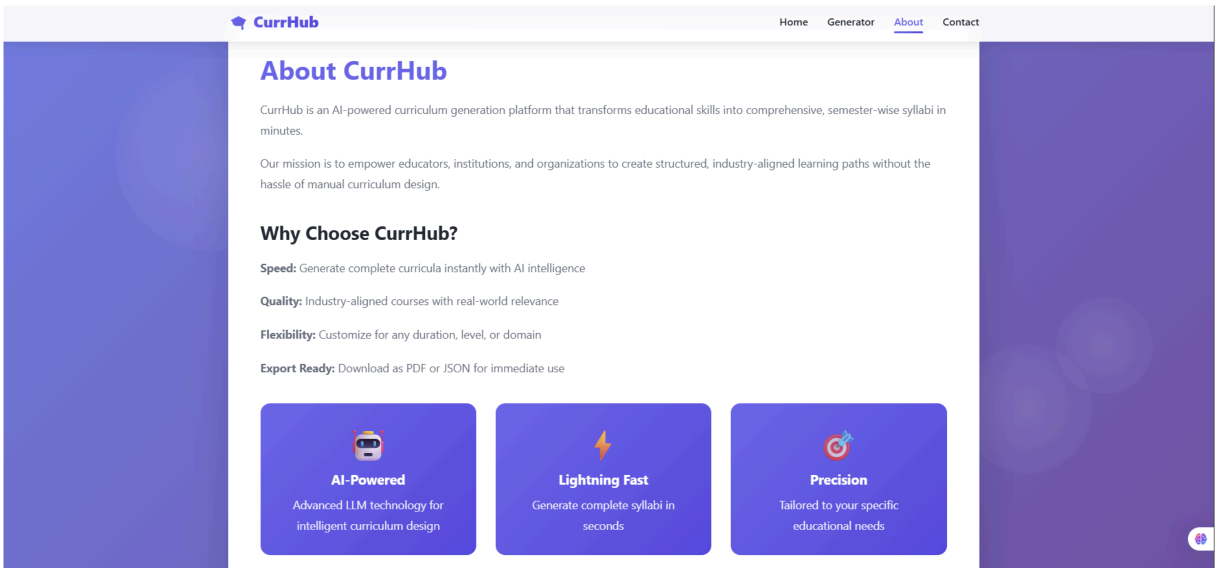
Specific industry or domain focus

Generate Clear

Build Your Vision

Create comprehensive, industry-aligned curricula with AI

✓ Export Ready



CurrHub

Home Generator About Contact

About CurrHub

CurrHub is an AI-powered curriculum generation platform that transforms educational skills into comprehensive, semester-wise syllabi in minutes.

Our mission is to empower educators, institutions, and organizations to create structured, industry-aligned learning paths without the hassle of manual curriculum design.

Why Choose CurrHub?

Speed: Generate complete curricula instantly with AI intelligence

Quality: Industry-aligned courses with real-world relevance

Flexibility: Customize for any duration, level, or domain

Export Ready: Download as PDF or JSON for immediate use

AI-Powered

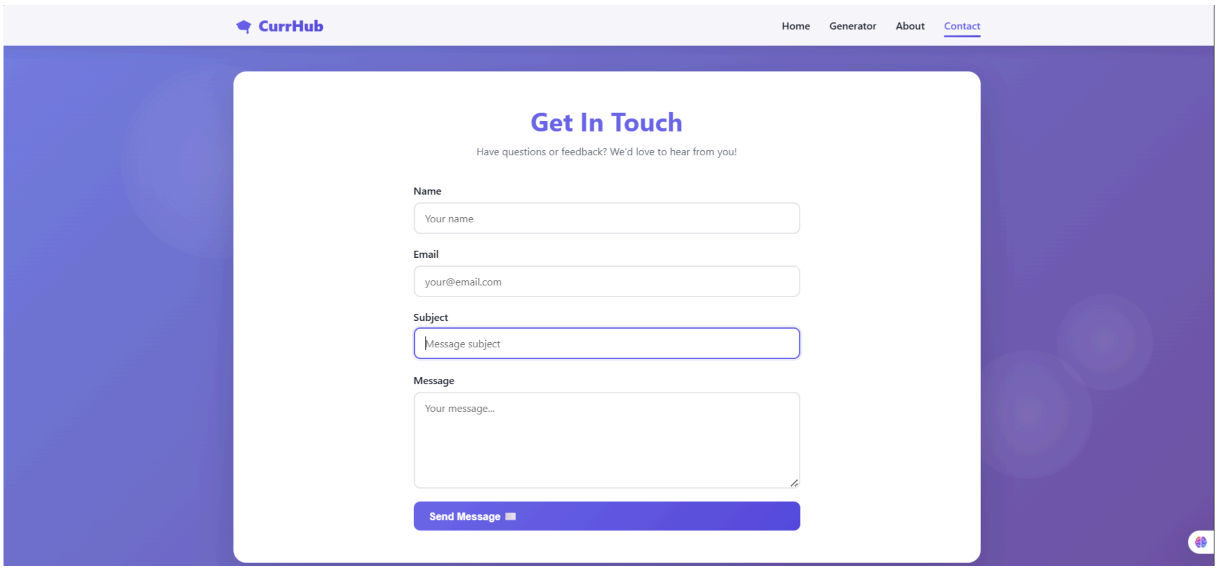
Advanced LLM technology for intelligent curriculum design

Lightning Fast

Generate complete syllabi in seconds

Precision

Tailored to your specific educational needs


CurrHub

Home Generator About Contact

Get In Touch

Have questions or feedback? We'd love to hear from you!

Name

Your name

Email

your@email.com

Subject

Message subject

Message

Your message...

Send Message

MILESTONE 5: Integration & Testing

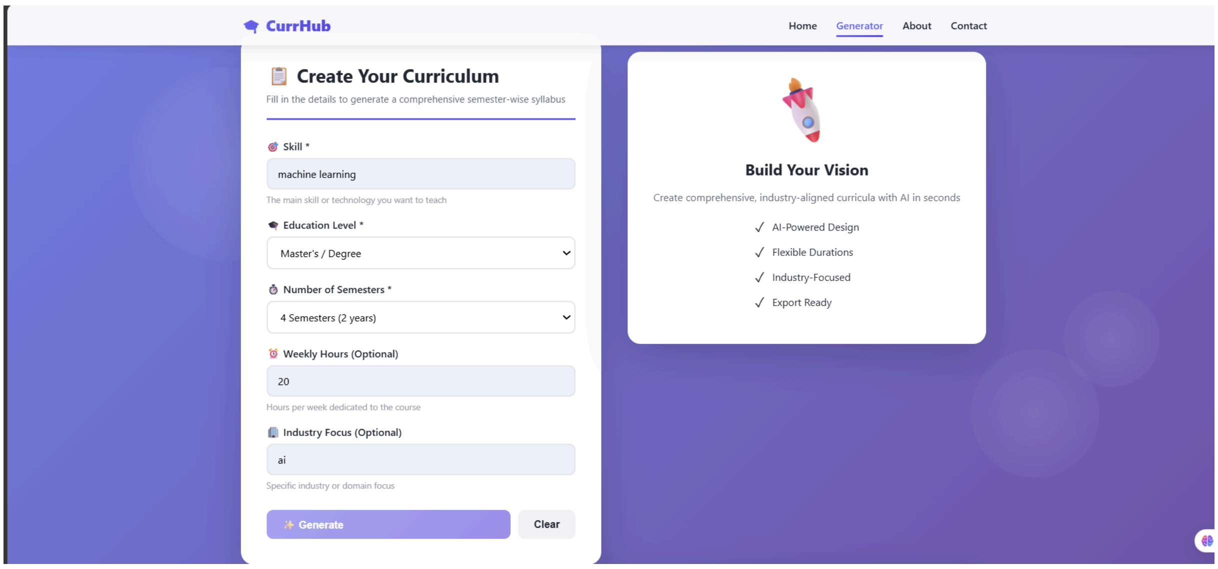
This milestone integrates all modules and performs functional, performance, and validation testing across different skills and education levels. It verifies AI response accuracy, PDF formatting quality, and end-to-end system reliability.

Activity 5.1: Functional Testing

Test Case : Machine Learning Masters

Input:

- Skill: Machine Learning
- Level: Masters
- Semesters: 4
- Weekly Hours: 20-25
- Industry Focus: AI



Create Your Curriculum

Fill in the details to generate a comprehensive semester-wise syllabus

Skill *: machine learning

The main skill or technology you want to teach

Education Level *: Master's / Degree

Number of Semesters *: 4 Semesters (2 years)

Weekly Hours (Optional): 20

Hours per week dedicated to the course

Industry Focus (Optional): ai

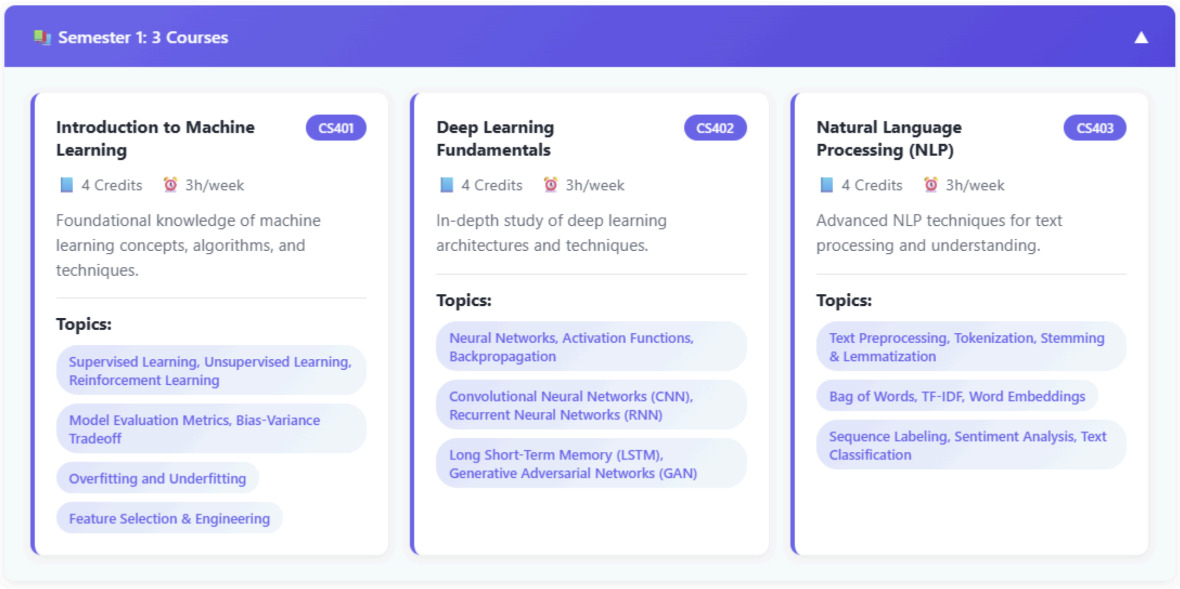
Specific industry or domain focus

Build Your Vision

Create comprehensive, industry-aligned curricula with AI in seconds

- ✓ AI-Powered Design
- ✓ Flexible Durations
- ✓ Industry-Focused
- ✓ Export Ready

Out put :



Semester 1: 3 Courses

Introduction to Machine Learning CS401

4 Credits 3h/week

Foundational knowledge of machine learning concepts, algorithms, and techniques.

Topics:
- Supervised Learning, Unsupervised Learning, Reinforcement Learning
- Model Evaluation Metrics, Bias-Variance Tradeoff
- Overfitting and Underfitting
- Feature Selection & Engineering

Deep Learning Fundamentals CS402

4 Credits 3h/week

In-depth study of deep learning architectures and techniques.

Topics:
- Neural Networks, Activation Functions, Backpropagation
- Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN)
- Long Short-Term Memory (LSTM), Generative Adversarial Networks (GAN)

Natural Language Processing (NLP) CS403

4 Credits 3h/week

Advanced NLP techniques for text processing and understanding.

Topics:
- Text Preprocessing, Tokenization, Stemming & Lemmatization
- Bag of Words, TF-IDF, Word Embeddings
- Sequence Labeling, Sentiment Analysis, Text Classification

📚 Semester 2: 3 Courses ▲

Machine Learning Systems Design CS404

📘 4 Credits ⏰ 3h/week

Designing, deploying, and scaling machine learning systems.

Topics:

- Distributed Computing, MapReduce, Spark
- Cloud Platforms (AWS, Google Cloud, Azure)
- Data Pipelines and ETL Tools

Advanced Deep Learning CS405

📘 4 Credits ⏰ 3h/week

Exploring advanced deep learning architectures and optimization techniques.

Topics:

- Deep Convolutional Networks (DCN), RNN Architectures (LSTM, GRU)
- Transformers, BERT, GPT-2
- Neural Architecture Search (NAS)

AI for Computer Vision CS406

📘 4 Credits ⏰ 3h/week

Applying machine learning techniques to computer vision tasks.

Topics:

- Image Classification, Object Detection (YOLO, Faster R-CNN)
- Semantic Segmentation, 3D Object Detection
- Convolutional Neural Networks (CNN) for Computer Vision

📚 Semester 3: 3 Courses ▲

Reinforcement Learning CS407

📘 4 Credits ⏰ 3h/week

Understanding and implementing reinforcement learning algorithms.

Topics:

- Markov Decision Processes (MDP), Q-Learning, SARSA
- Deep Reinforcement Learning (DQR)
- Policy Gradients, Actor-Critic Methods

Ethics and AI CS408

📘 4 Credits ⏰ 3h/week

Exploring the societal and ethical implications of AI.

Topics:

- Bias in Machine Learning, Fairness, Accountability
- Privacy Preservation Techniques (Differential Privacy)
- AI Governance, Ethical Frameworks

AI for Time Series Analysis CS409

📘 4 Credits ⏰ 3h/week

Applying machine learning techniques to time series data.

Topics:

- Time Series Forecasting (ARIMA, SARIMA)
- Prophet, LSTM for Time Series
- Anomaly Detection in Time Series

📚 Semester 4: 3 Courses ▲

Natural Language Processing Advanced Topics CS410

📘 4 Credits ⏰ 3h/week

Advanced NLP techniques and deep learning applications.

Topics:

- Sentiment Analysis, Topic Modeling (LDA)
- Named Entity Recognition (NER), Part-of-Speech Tagging
- Deep Learning for NLP (Transformers)

AI for Healthcare CS411

📘 4 Credits ⏰ 3h/week

Applying machine learning to healthcare data and challenges.

Topics:

- Medical Image Analysis, Disease Diagnosis
- Predictive Modeling for Patient Outcomes
- Natural Language Processing in Electronic Health Records

AI Project Workshop & Capstone CS412

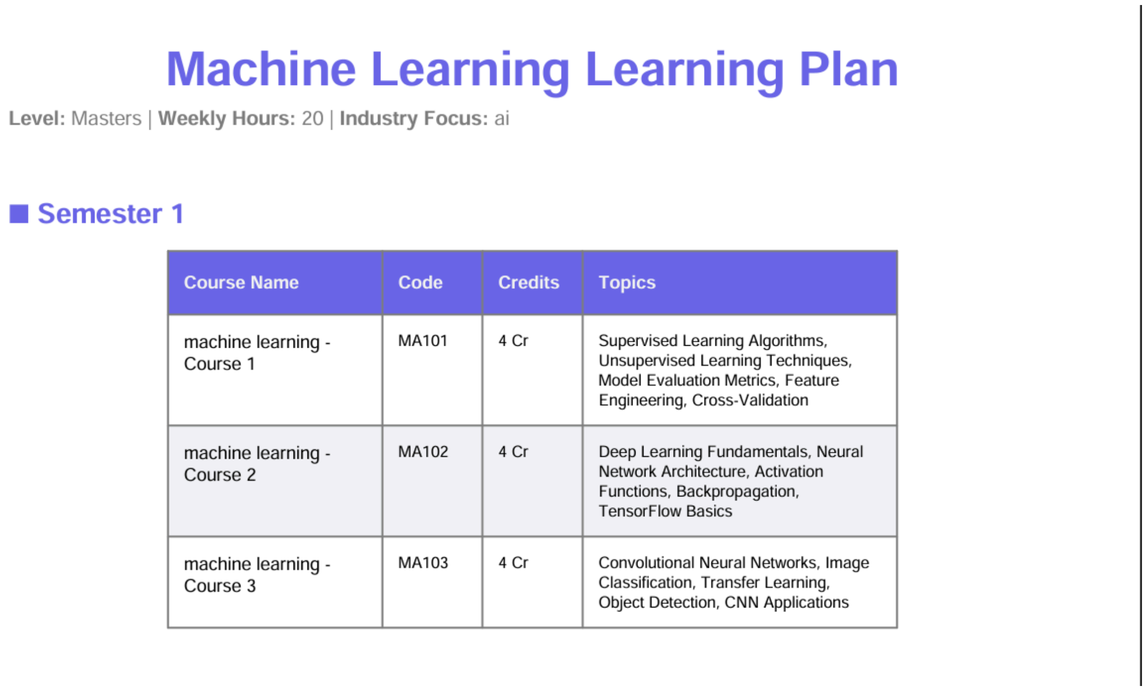
📘 4 Credits ⏰ 3h/week

Hands-on experience with designing, implementing, and presenting a machine learning project.

Topics:

- Project Proposal, Development, and Presentation
- Peer Review and Feedback
- Capstone Project Presentations

PDF download :

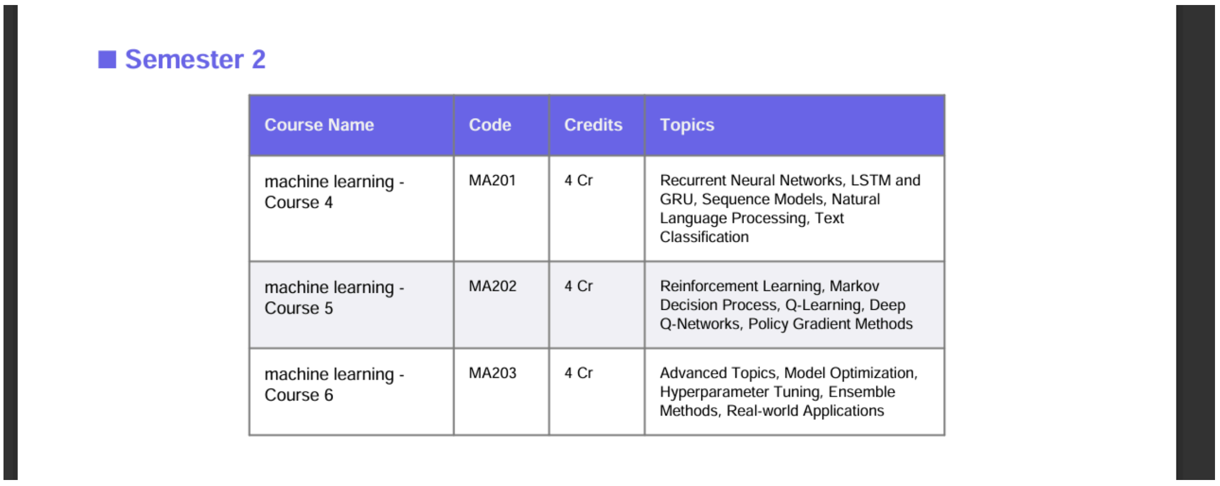


Machine Learning Learning Plan

Level: Masters | **Weekly Hours:** 20 | **Industry Focus:** ai

■ Semester 1

Course Name	Code	Credits	Topics
machine learning - Course 1	MA101	4 Cr	Supervised Learning Algorithms, Unsupervised Learning Techniques, Model Evaluation Metrics, Feature Engineering, Cross-Validation
machine learning - Course 2	MA102	4 Cr	Deep Learning Fundamentals, Neural Network Architecture, Activation Functions, Backpropagation, TensorFlow Basics
machine learning - Course 3	MA103	4 Cr	Convolutional Neural Networks, Image Classification, Transfer Learning, Object Detection, CNN Applications



■ Semester 2

Course Name	Code	Credits	Topics
machine learning - Course 4	MA201	4 Cr	Recurrent Neural Networks, LSTM and GRU, Sequence Models, Natural Language Processing, Text Classification
machine learning - Course 5	MA202	4 Cr	Reinforcement Learning, Markov Decision Process, Q-Learning, Deep Q-Networks, Policy Gradient Methods
machine learning - Course 6	MA203	4 Cr	Advanced Topics, Model Optimization, Hyperparameter Tuning, Ensemble Methods, Real-world Applications

Conclusion

GenAI Curriculum Generator successfully delivers an intelligent curriculum design assistant that combines advanced AI with educational expertise. By leveraging local Ollama deployment with Granite 3.3 2B, the system provides fast, accurate, and personalized curriculum recommendations without requiring cloud API keys. The platform demonstrates the viability of AI-powered curriculum design, making professional educational planning accessible to educators, institutions, and course designers while maintaining high performance, accuracy, and data privacy through local processing.