Project Assignments

4th Semester: *Jan-May 2026*

Course: *Introduction to AI ML*

January 28, 2026

Index

1 Project 1: Intelligent Research Topic Analysis and Agentic AI Research Assistant

Project Objective

Design and implement an AI-based research automation system that progresses from traditional NLP-based topic analysis to a fully agentic AI research assistant capable of autonomous information retrieval, reasoning, and structured report generation.

Constraints and Requirements

- Team size: **3–4 students**
- Paid APIs are **not permitted**
- Only free-tier APIs or open-source models may be used
- A user interface is **mandatory**
- Final application must be **publicly hosted**
- Localhost-only demonstrations will not be accepted

Approved Technology Options

Traditional ML / NLP (Milestone 1)

- Python NLP libraries: NLTK, spaCy
- scikit-learn (TF-IDF, topic modeling, clustering)

LLMs (Milestone 2)

- Open-source models via Hugging Face
- Free-tier LLM APIs where available

Search and Retrieval

- Tavily (free tier)
- DuckDuckGo search wrappers
- Bing Web Search (free quota)

Agent Framework

- LangGraph (recommended for Milestone 2)

UI Framework

- Streamlit (recommended)
- Gradio

Hosting Platform

- Hugging Face Spaces
- Streamlit Community Cloud
- Render (free tier)

1.1 Dataset

- `https://www.kaggle.com/datasets/Cornell-University/arxiv`

Milestone 1: Research Topic Analysis System (Mid-Sem)

Duration: Jan 20 – Feb 15

Objective

Design and implement a traditional NLP-based research analysis system that processes documents related to a research topic and produces structured analytical summaries without using LLMs or agentic workflows.

Inputs

- Research topic keywords
- Uploaded research documents or articles

Functional Requirements

- Perform text preprocessing and feature extraction
- Identify key themes and keywords
- Apply topic modeling or clustering
- Generate extractive or statistical summaries
- Display analytical results through a basic UI

Technical Requirements

- Text preprocessing (tokenization, stop-word removal, lemmatization)
- Feature extraction using TF-IDF or Bag-of-Words
- Extractive summarization techniques
- Evaluation using coherence or interpretability measures

Outputs

- Key terms and themes
- Topic clusters or categories
- Extractive summary of content
- Analytical visualizations (optional)

Milestone 1 Deliverables

- Problem understanding and use-case description
- Input–output specification
- System architecture diagram (traditional NLP pipeline)
- ML/NLP implementation with explanations
- Working local application with basic UI
- Short report describing limitations of traditional approaches

Milestone 2: Agentic AI Research Assistant (End-Sem)

Duration: Feb 16 – Mar 12

Objective

Extend the research analysis system into a fully autonomous agentic AI research assistant that retrieves information, reasons across sources, manages state, and generates structured research reports.

Functional Requirements

- Accept open-ended research queries
- Perform web search and retrieval
- Aggregate information from multiple sources
- Maintain state across research steps
- Generate structured research reports

Technical Requirements

- Open-source or free-tier LLM integration
- Web search integration using approved tools
- Retrieval-augmented generation
- LangGraph-based agent workflow with:
  - Nodes for search, summarization, and validation
  - Edges for multi-step control flow
  - Explicit state management
- Prompt strategies to reduce hallucinations

Structured Output Requirements

The generated research report must include:

- Title
- Abstract
- Key Findings
- Sources (URLs)
- Conclusion

Robustness and Error Handling

- Handling missing or insufficient search results
- Graceful recovery from API or model failures
- Management of noisy or contradictory sources
- Clear user-facing error messages

Extension (Choose Any One)

- Follow-up question generation
- Topic expansion
- PDF or Markdown export
- Session-based memory

Final Deliverables

- Publicly accessible hosted application link
- GitHub repository with complete codebase
- Demo video (5–7 minutes)

2 Project 2: Intelligent Learning Analytics and Agentic Study Coach

Project Objective

Design and implement an AI-powered learning analytics system that analyzes student performance data and generates study recommendations. The system is progressively extended into an agentic AI study coach that reasons about learning gaps, retrieves resources, and produces personalized study plans.

Constraints and Requirements

- Team size: **3–4 students**
- Paid APIs are **not permitted**
- Only free-tier APIs or open-source models may be used
- A user interface is **mandatory**
- Final application must be **publicly hosted**
- Localhost-only demonstrations will not be accepted

2.1 Dataset

- `https://www.kaggle.com/datasets/spscientist/students-performance-in-exams`
- `https://archive.ics.uci.edu/ml/datasets/student+performance`

Recommended Tools (Not Mandatory)

- ML libraries: scikit-learn, NumPy, pandas
- NLP tools (optional): NLTK, spaCy
- LLMs (Milestone 2): Open-source models or free-tier APIs
- Agent workflow: LangGraph (recommended for Milestone 2)
- UI: Streamlit or Gradio
- Hosting: Hugging Face Spaces, Streamlit Community Cloud, Render (free tier)

2.2 Dataset

- `https://www.kaggle.com/datasets/Cornell-University/arxiv`

Milestone 1: ML-Based Learning Analytics System (Mid-Sem)

Objective

Build a data-driven learning analytics system using classical machine learning techniques to analyze student performance and generate basic study recommendations.

Inputs

- Student performance dataset (CSV) containing:
  - Quiz or test scores
  - Topic-wise accuracy
  - Time spent per topic

Functional Requirements

- Perform data preprocessing (handling missing values, scaling)
- Predict student performance or risk level
- Classify students into categories such as:
  - At-risk
  - Average
  - High-performing
- Generate rule-based or ML-based study recommendations

Technical Requirements

- Supervised learning models (e.g., Logistic Regression, Linear Regression)
- Optional clustering for learner grouping (e.g., K-Means)
- Model evaluation using appropriate metrics

User Interface Requirements

- Upload student data
- Display predictions and classifications
- Show generated study recommendations

Milestone 1 Deliverables

- Problem understanding and use-case description
- Input–output specification
- System architecture diagram (ML pipeline)
- Working application with a basic UI
- Brief analysis of model performance and limitations

Milestone 2: Agentic AI Study Coach (End-Sem)

Objective

Extend the analytics system into an autonomous AI study coach that reasons about student performance, plans learning strategies, and adapts recommendations.

Functional Requirements

- Accept student goals (e.g., exam preparation, skill improvement)
- Diagnose learning gaps using performance data
- Generate a multi-step personalized study plan
- Retrieve and summarize relevant learning resources
- Maintain session-based memory (optional)

Technical Requirements

- Integration of open-source or free-tier LLMs
- Retrieval-augmented generation (optional but recommended)
- Agentic workflow using LangGraph or equivalent
- Prompt strategies to ensure reliable recommendations

Structured Output Requirements

The generated study coach report must include:

- Learning diagnosis
- Personalized study plan
- Weekly or milestone-based goals
- Recommended learning resources (URLs)
- Progress feedback or next steps

Extension (Choose Any One)

- Quiz or practice question generation
- Adaptive difficulty adjustment
- PDF export of study plan
- Multi-student analytics dashboard

Final Deliverables

- Publicly accessible hosted application link
- GitHub repository with complete codebase
- Demo video (5–7 minutes)

3 Project 3: Intelligent Patient Risk Assessment and Agentic Health Support System

Project Objective

Design and implement an AI-based healthcare analytics system that predicts patient health risks using clinical data and later extends into an agentic AI assistant that generates structured health guidance and preventive recommendations.

Constraints and Requirements

- Team size: **3–4 students**
- Paid APIs are **not permitted**
- Only free-tier APIs or open-source models may be used
- A user interface is **mandatory**
- Final application must be **publicly hosted**
- Localhost-only demonstrations will not be accepted

Recommended Tools (Not Mandatory)

- ML libraries: scikit-learn, pandas, NumPy
- LLMs (Milestone 2): Open-source models or free-tier APIs
- Agent workflow: LangGraph (recommended)
- Vector stores (optional): Chroma, FAISS
- UI: Streamlit or Gradio
- Hosting: Hugging Face Spaces, Streamlit Community Cloud, Render (free tier)

3.1 Dataset

- `https://archive.ics.uci.edu/ml/index.php`
- `https://www.kaggle.com/datasets`

Milestone 1: ML-Based Patient Risk Assessment (Mid-Sem)

Objective

Build a machine learning system that predicts patient health risk based on structured clinical data without using large language models.

Inputs

- Patient health data (CSV), such as:
  - Age and demographics
  - Vital signs
  - Lab values
  - Medical history indicators

Functional Requirements

- Perform data preprocessing and feature engineering
- Predict patient risk category or risk score
- Display predictions and basic insights through a UI

Technical Requirements

- Supervised learning models (e.g., Logistic Regression, Linear Regression)
- Optional use of decision trees or neural networks
- Model evaluation using appropriate metrics

User Interface Requirements

- Upload patient data
- Display risk predictions and evaluation metrics
- Provide basic explanation of results

Milestone 1 Deliverables

- Problem understanding and healthcare use-case description
- Input–output specification
- System architecture diagram (ML pipeline)
- Working local application with basic UI
- Brief analysis of model performance

Milestone 2: Agentic AI Health Support Assistant (End-Sem)

Objective

Extend the risk assessment system into an agentic AI assistant that reasons over patient data and generates structured, evidence-based health guidance.

Functional Requirements

- Accept patient data and health-related queries
- Analyze risk predictions and contributing factors
- Retrieve relevant medical or educational information
- Generate structured health summaries and recommendations
- Handle missing or incomplete data gracefully

Technical Requirements

- Open-source or free-tier LLM integration
- Retrieval-augmented generation (optional but recommended)
- Agentic workflow with explicit state management
- Prompt strategies to reduce unsupported claims

Structured Output Requirements

The generated health report must include:

- Patient risk summary
- Key contributing factors
- Preventive or follow-up recommendations
- Sources or references used
- Medical disclaimer stating non-clinical use

Extension (Choose Any One)

- Follow-up question generation
- Trend analysis across patient records
- PDF export of health report
- Session-based memory for patient tracking

Final Deliverables

- Publicly accessible hosted application link
- GitHub repository with complete codebase
- Demo video (5–7 minutes)

4 Project 4: Clinical Appointment No-Show Prediction and Agentic Care Coordination System

Project Objective

Design and implement an AI-based healthcare operations system that predicts patient appointment no-shows and extends into an agentic AI assistant that generates actionable care coordination and intervention recommendations.

Constraints and Requirements

- Team size: **3–4 students**
- Paid APIs are **not permitted**
- Only free-tier APIs or open-source models may be used
- A user interface is **mandatory**
- Final application must be **publicly hosted**
- Localhost-only demonstrations will not be accepted

Recommended Tools (Not Mandatory)

- ML libraries: scikit-learn, pandas, NumPy
- LLMs (Milestone 2): Open-source models or free-tier APIs
- Agent workflow: LangGraph (recommended)
- Vector stores (optional): Chroma, FAISS
- UI: Streamlit or Gradio
- Hosting: Hugging Face Spaces, Streamlit Community Cloud, Render (free tier)

4.1 Dataset

- `https://www.kaggle.com/datasets/joniarroba/noshowappointments`

Milestone 1: ML-Based Appointment No-Show Prediction (Mid-Sem)

Objective

Build a machine learning system that predicts the likelihood of patient appointment no-shows using historical scheduling data.

Inputs

- Appointment data (CSV), such as:
  - Appointment lead time
  - Time and day of appointment
  - Patient attendance history
  - Department or appointment type

Functional Requirements

- Perform data preprocessing and feature engineering
- Predict no-show probability or classification
- Identify key contributing factors
- Display predictions through a basic UI

Technical Requirements

- Supervised learning models (e.g., Logistic Regression, Decision Trees)
- Optional use of ensemble methods
- Model evaluation using classification metrics

User Interface Requirements

- Upload appointment data
- Display no-show risk and predictions
- Show basic feature importance or explanations

Milestone 1 Deliverables

- Problem understanding and operational use-case description
- Input–output specification
- System architecture diagram (ML pipeline)
- Working local application with basic UI
- Brief evaluation and analysis of model performance

Milestone 2: Agentic AI Care Coordination Assistant (End-Sem)

Objective

Extend the prediction system into an agentic AI assistant that reasons about appointment risks and generates structured intervention strategies.

Functional Requirements

- Analyze predicted no-show risks
- Retrieve best-practice care coordination guidelines
- Generate actionable intervention recommendations
- Handle missing or noisy data gracefully

Technical Requirements

- Open-source or free-tier LLM integration
- Retrieval-augmented generation (optional but recommended)
- Agentic workflow with explicit state management
- Prompt strategies to reduce unsupported recommendations

Structured Output Requirements

The generated care coordination report must include:

- Appointment risk summary
- Key contributing factors
- Recommended intervention strategies
- Supporting sources or references
- Operational and ethical disclaimers

Extension (Choose Any One)

- Prioritization of high-risk appointments
- Batch-level operational insights
- PDF export of care coordination report
- Session-based tracking of interventions

Final Deliverables

- Publicly accessible hosted application link
- GitHub repository with complete codebase
- Demo video (5–7 minutes)

5 Project 5: Customer Churn Prediction and Agentic Retention Strategy System

Project Objective

Design and implement an AI-driven customer analytics system that predicts customer churn using historical behavior data and extends into an agentic AI assistant that generates structured retention strategies.

Constraints and Requirements

- Team size: **3–4 students**
- Paid APIs are **not permitted**
- Only free-tier APIs or open-source models may be used
- A user interface is **mandatory**
- Final application must be **publicly hosted**
- Localhost-only demonstrations will not be accepted

Recommended Tools (Not Mandatory)

- ML libraries: scikit-learn, pandas, NumPy
- LLMs (Milestone 2): Open-source models or free-tier APIs
- Agent workflow: LangGraph (recommended)
- Vector stores (optional): Chroma, FAISS
- UI: Streamlit or Gradio
- Hosting: Hugging Face Spaces, Streamlit Community Cloud, Render (free tier)

5.1 Dataset

- `https://www.kaggle.com/datasets/blastchar/telco-customer-churn`
- `https://www.kaggle.com/datasets`

Milestone 1: ML-Based Customer Churn Prediction (Mid-Sem)

Objective

Build a machine learning system that identifies customers at risk of churn using historical customer behavior and transactional data.

Inputs

- Customer data (CSV), such as:
  - Usage frequency and duration
  - Purchase or transaction history
  - Customer tenure
  - Support or interaction history

Functional Requirements

- Perform data preprocessing and feature engineering
- Predict churn probability or churn class
- Identify key drivers of churn
- Display predictions through a basic UI

Technical Requirements

- Supervised learning models (e.g., Logistic Regression, Decision Trees)
- Optional use of ensemble models or neural networks
- Model evaluation using classification metrics

User Interface Requirements

- Upload customer data
- Display churn predictions and risk levels
- Show important churn-driving features

Milestone 1 Deliverables

- Problem understanding and business use-case description
- Input–output specification
- System architecture diagram (ML pipeline)
- Working local application with basic UI
- Brief evaluation and analysis of model performance

Milestone 2: Agentic AI Retention Strategy Assistant (End-Sem)

Objective

Extend the churn prediction system into an agentic AI assistant that reasons about customer churn risk and generates personalized retention strategies.

Functional Requirements

- Analyze churn risk predictions and contributing factors
- Retrieve customer retention best practices
- Generate structured retention recommendations
- Handle missing or noisy customer data gracefully

Technical Requirements

- Open-source or free-tier LLM integration
- Retrieval-augmented generation (optional but recommended)
- Agentic workflow with explicit state management
- Prompt strategies to reduce unsupported or biased suggestions

Structured Output Requirements

The generated retention report must include:

- Customer churn risk summary
- Key contributing factors
- Recommended retention actions
- Supporting sources or references
- Business and ethical disclaimers

Extension (Choose Any One)

- Customer segmentation-based strategies
- Scenario-based retention planning
- PDF export of retention report
- Batch-level churn analytics dashboard

Final Deliverables

- Publicly accessible hosted application link
- GitHub repository with complete codebase
- Demo video (5-7 minutes)

6 Project 6: Vehicle Maintenance Prediction and Agentic Fleet Management System

Project Objective

Design and implement an AI-driven fleet analytics system that predicts vehicle maintenance needs using historical usage and sensor data, and extend it into an agentic AI assistant that generates structured fleet maintenance and servicing recommendations.

Constraints and Requirements

- Team size: **3–4 students**
- Paid APIs are **not permitted**
- Only free-tier APIs or open-source models may be used
- A user interface is **mandatory**
- Final application must be **publicly hosted**
- Localhost-only demonstrations will not be accepted

Recommended Tools (Not Mandatory)

- ML libraries: scikit-learn, pandas, NumPy
- LLMs (Milestone 2): Open-source models or free-tier APIs
- Agent workflow: LangGraph (recommended)
- Vector stores (optional): Chroma, FAISS
- UI: Streamlit or Gradio
- Hosting: Hugging Face Spaces, Streamlit Community Cloud, Render (free tier)

6.1 Dataset

- `https://www.kaggle.com/datasets/chavindudulaj/vehicle-maintenance-data`

Milestone 1: ML-Based Vehicle Maintenance Prediction (Mid-Sem)

Objective

Build a machine learning system that predicts vehicle maintenance risk or time-to-failure using historical vehicle usage and maintenance records.

Inputs

- Vehicle data (CSV), such as:
  - Mileage and engine hours
  - Fault or diagnostic codes
  - Service and repair history
  - Usage patterns

Functional Requirements

- Perform data preprocessing and feature engineering
- Predict maintenance risk or failure likelihood
- Identify key contributing factors
- Display predictions through a basic UI

Technical Requirements

- Supervised learning models (e.g., Logistic Regression, Decision Trees)
- Optional use of regression or neural network models
- Model evaluation using classification or regression metrics

User Interface Requirements

- Upload vehicle data
- Display maintenance risk predictions
- Show basic feature importance or explanations

Milestone 1 Deliverables

- Problem understanding and fleet management use-case description
- Input–output specification
- System architecture diagram (ML pipeline)
- Working local application with basic UI
- Brief analysis of model performance

Milestone 2: Agentic AI Fleet Management Assistant (End-Sem)

Objective

Extend the maintenance prediction system into an agentic AI assistant that reasons about vehicle health and generates actionable fleet maintenance and servicing plans.

Functional Requirements

- Analyze predicted maintenance risks
- Retrieve vehicle maintenance guidelines or best practices
- Generate structured servicing and scheduling recommendations
- Handle incomplete or noisy vehicle data gracefully

Technical Requirements

- Open-source or free-tier LLM integration
- Retrieval-augmented generation (optional but recommended)
- Agentic workflow with explicit state management
- Prompt strategies to reduce unsupported recommendations

Structured Output Requirements

The generated fleet management report must include:

- Vehicle health summary
- Maintenance risk assessment
- Recommended service actions and timelines
- Supporting sources or references
- Operational and safety disclaimers

Extension (Choose Any One)

- Predictive maintenance scheduling
- Fleet-level maintenance optimization
- PDF export of maintenance report
- Maintenance history tracking across sessions

Final Deliverables

- Publicly accessible hosted application link
- GitHub repository with complete codebase
- Demo video (5-7 minutes)

7 Project 7: Intelligent Contract Risk Analysis and Agentic Legal Assistance System

Project Objective

Design and implement an AI-driven legal document analysis system that identifies and classifies risky clauses in contracts, and extend it into an agentic AI assistant that generates structured contract risk assessments and explanations.

Constraints and Requirements

- Team size: **3–4 students**
- Paid APIs are **not permitted**
- Only free-tier APIs or open-source models may be used
- A user interface is **mandatory**
- Final application must be **publicly hosted**
- Localhost-only demonstrations will not be accepted

Recommended Tools (Not Mandatory)

- NLP / ML libraries: scikit-learn, NLTK, spaCy
- LLMs (Milestone 2): Open-source models or free-tier APIs
- Agent workflow: LangGraph (recommended)
- Vector stores (optional): Chroma, FAISS
- UI: Streamlit or Gradio
- Hosting: Hugging Face Spaces, Streamlit Community Cloud, Render (free tier)

7.1 Dataset

- `https://www.kaggle.com/datasets/mrjkanderson/legal-dataset`
- `https://indiankanoon.org/`

Milestone 1: ML-Based Contract Risk Classification (Mid-Sem)

Objective

Build a classical NLP and machine learning system that analyzes contract documents and identifies potentially risky clauses.

Inputs

- Contract documents in text or PDF format

Functional Requirements

- Perform text preprocessing and clause segmentation
- Classify clauses into risk categories
- Assign risk labels or scores to clauses
- Display flagged clauses through a basic UI

Technical Requirements

- Text feature extraction using TF-IDF or embeddings
- Supervised classification models (e.g., Logistic Regression, Decision Trees)
- Model evaluation using precision, recall, and F1 score

User Interface Requirements

- Upload contract documents
- Display clause-level risk labels
- Highlight risky clauses for inspection

Milestone 1 Deliverables

- Problem understanding and legal use-case description
- Input–output specification
- System architecture diagram (NLP pipeline)
- Working local application with basic UI
- Brief evaluation of model performance

Milestone 2: Agentic AI Legal Assistance System (End-Sem)

Objective

Extend the contract analysis system into an agentic AI assistant that reasons about legal risks and generates structured contract risk assessments.

Functional Requirements

- Analyze identified contract risks and patterns
- Retrieve relevant legal guidelines or best practices
- Generate structured contract risk reports
- Provide clause-level explanations
- Handle incomplete documents or retrieval failures gracefully

Technical Requirements

- Open-source or free-tier LLM integration
- Retrieval-augmented generation (optional but recommended)
- Agentic workflow with explicit state management
- Prompt strategies to reduce hallucinations

Structured Output Requirements

The generated legal assistance report must include:

- Contract summary
- Identified risks with clause references
- Risk severity assessment
- Recommended mitigation actions
- Legal and ethical disclaimers

Extension (Choose Any One)

- Clause comparison across multiple contracts
- Risk trend analysis
- PDF export of contract risk report
- Interactive clause-level explanations

Final Deliverables

- Publicly accessible hosted application link
- GitHub repository with complete codebase
- Demo video (5–7 minutes)

8 Project 8: Intelligent Crop Yield Prediction and Agentic Farm Advisory System

Project Objective

Design and implement an AI-driven agricultural analytics system that predicts crop yield using historical farm, soil, and weather data, and extend it into an agentic AI farm advisory assistant that generates structured crop management recommendations.

Constraints and Requirements

- Team size: **3–4 students**
- Paid APIs are **not permitted**
- Only free-tier APIs or open-source models may be used
- A user interface is **mandatory**
- Final application must be **publicly hosted**
- Localhost-only demonstrations will not be accepted

Recommended Tools (Not Mandatory)

- ML libraries: scikit-learn, pandas, NumPy
- LLMs (Milestone 2): Open-source models or free-tier APIs
- Agent workflow: LangGraph (recommended)
- Vector stores (optional): Chroma, FAISS
- UI: Streamlit or Gradio
- Hosting: Hugging Face Spaces, Streamlit Community Cloud, Render (free tier)

8.1 Dataset

- `https://www.kaggle.com/datasets/patelris/crop-yield-prediction-dataset`

Milestone 1: ML-Based Crop Yield Prediction (Mid-Sem)

Objective

Build a machine learning system that predicts crop yield or yield category using historical agricultural, soil, and weather data.

Inputs

- Farm and crop data (CSV), such as:
  - Rainfall and temperature
  - Soil type and nutrient indicators
  - Fertilizer and irrigation usage
  - Crop type and season

Functional Requirements

- Perform data preprocessing and feature engineering
- Predict crop yield or yield category
- Identify key factors affecting yield
- Display predictions through a basic UI

Technical Requirements

- Supervised learning models (e.g., Linear Regression, Decision Trees)
- Optional use of neural networks
- Model evaluation using regression metrics (MAE, RMSE, R^2)

User Interface Requirements

- Upload crop and farm data
- Display predicted yield values
- Show basic explanation of yield-driving features

Milestone 1 Deliverables

- Problem understanding and agricultural use-case description
- Input–output specification
- System architecture diagram (ML pipeline)
- Working local application with basic UI
- Brief evaluation of model performance

Milestone 2: Agentic AI Farm Advisory Assistant (End-Sem)

Objective

Extend the yield prediction system into an agentic AI assistant that reasons about farm conditions and generates actionable crop management advice.

Functional Requirements

- Analyze predicted crop yield and risk factors
- Retrieve agronomy and farming best practices
- Generate structured crop advisory recommendations
- Handle incomplete or noisy agricultural data gracefully

Technical Requirements

- Open-source or free-tier LLM integration
- Retrieval-augmented generation (optional but recommended)
- Agentic workflow with explicit state management
- Prompt strategies to reduce unsupported recommendations

Structured Output Requirements

The generated farm advisory report must include:

- Crop and field summary
- Yield prediction interpretation
- Identified risk factors (weather, soil, inputs)
- Recommended farming actions
- Supporting sources or references
- Agricultural and ethical disclaimers

Extension (Choose Any One)

- Seasonal planning recommendations
- Fertilizer or irrigation optimization
- PDF export of advisory report
- Multi-season yield comparison

Final Deliverables

- Publicly accessible hosted application link
- GitHub repository with complete codebase
- Demo video (5–7 minutes)

9 Project 9: Intelligent Property Price Prediction and Agentic Real Estate Advisory System

Project Objective

Design and implement an AI-driven real estate analytics system that predicts property prices using historical data and extends into an agentic AI assistant that generates structured buying or investment recommendations.

Constraints and Requirements

- Team size: **3–4 students**
- Paid APIs are **not permitted**
- Only free-tier APIs or open-source models may be used
- A user interface is **mandatory**
- Final application must be **publicly hosted**
- Localhost-only demonstrations will not be accepted

Recommended Tools (Not Mandatory)

- ML libraries: scikit-learn, pandas, NumPy
- LLMs (Milestone 2): Open-source models or free-tier APIs
- Agent workflow: LangGraph (recommended)
- Vector stores (optional): Chroma, FAISS
- UI: Streamlit or Gradio
- Hosting: Hugging Face Spaces, Streamlit Community Cloud, Render (free tier)

9.1 Dataset

- `https://www.kaggle.com/datasets/yasserh/housing-prices-dataset`
- `https://dataful.in/datasets/17611/`

Milestone 1: ML-Based Property Price Prediction (Mid-Sem)

Objective

Build a machine learning system that predicts property prices or price ranges using historical property and location data.

Inputs

- Property data (CSV), such as:
  - Location and neighborhood features
  - Property size and number of rooms
  - Property age and amenities
  - Market indicators (optional)

Functional Requirements

- Perform data preprocessing and feature engineering
- Predict property prices or price categories
- Identify key price-driving factors
- Display predictions through a basic UI

Technical Requirements

- Supervised learning models (e.g., Linear Regression, Decision Trees, Random Forests)
- Optional use of neural networks
- Model evaluation using regression metrics (MAE, RMSE, R^2)

User Interface Requirements

- Upload property data
- Display predicted prices or price ranges
- Show basic explanation of price-driving features

Milestone 1 Deliverables

- Problem understanding and real estate use-case description
- Input–output specification
- System architecture diagram (ML pipeline)
- Working local application with basic UI
- Brief evaluation of model performance

Milestone 2: Agentic AI Real Estate Advisory Assistant (End-Sem)

Objective

Extend the price prediction system into an agentic AI assistant that reasons about property value, market trends, and user preferences to generate structured advisory reports.

Functional Requirements

- Analyze predicted prices and market conditions
- Retrieve real estate trends and regulatory information
- Compare similar properties (optional)
- Generate structured buying or investment recommendations
- Handle incomplete or noisy property data gracefully

Technical Requirements

- Open-source or free-tier LLM integration
- Retrieval-augmented generation (optional but recommended)
- Agentic workflow with explicit state management
- Prompt strategies to reduce unsupported financial claims

Structured Output Requirements

The generated real estate advisory report must include:

- Property summary
- Price prediction interpretation
- Market trend insights
- Recommended actions for buyers or investors
- Supporting sources or references
- Legal and financial disclaimers

Extension (Choose Any One)

- Comparable property analysis
- Location-based price comparison
- PDF export of advisory report
- Multi-property portfolio analysis

Final Deliverables

- Publicly accessible hosted application link
- GitHub repository with complete codebase
- Demo video (5–7 minutes)

10 Project 10: Intelligent Credit Risk Scoring and Agentic Lending Decision Support System

Project Objective

Design and implement an AI-driven credit analytics system that evaluates borrower credit risk using historical loan and borrower data, and extend it into an agentic AI assistant that generates structured lending recommendations.

Constraints and Requirements

- Team size: **3–4 students**
- Paid APIs are **not permitted**
- Only free-tier APIs or open-source models may be used
- A user interface is **mandatory**
- Final application must be **publicly hosted**
- Localhost-only demonstrations will not be accepted

Recommended Tools (Not Mandatory)

- ML libraries: scikit-learn, pandas, NumPy
- LLMs (Milestone 2): Open-source models or free-tier APIs
- Agent workflow: LangGraph (recommended)
- Vector stores (optional): Chroma, FAISS
- UI: Streamlit or Gradio
- Hosting: Hugging Face Spaces, Streamlit Community Cloud, Render (free tier)

10.1 Dataset

- `https://www.kaggle.com/datasets/adilshamim8/credit-risk-benchmark-dataset`

Milestone 1: ML-Based Credit Risk Scoring (Mid-Sem)

Objective

Build a machine learning system that predicts borrower credit risk or probability of default using historical borrower and loan data.

Inputs

- Borrower and loan data (CSV), such as:
  - Income and employment status
  - Credit history and repayment behavior
  - Loan amount and tenure
  - Past defaults or delinquencies

Functional Requirements

- Perform data preprocessing and feature engineering
- Predict credit risk score or classification
- Identify key risk-driving factors
- Display predictions through a basic UI

Technical Requirements

- Supervised learning models (e.g., Logistic Regression, Decision Trees)
- Optional use of ensemble models or neural networks
- Model evaluation using classification metrics (Accuracy, Precision, Recall, ROC-AUC)

User Interface Requirements

- Upload borrower and loan data
- Display credit risk scores or categories
- Show basic explanation of risk factors

Milestone 1 Deliverables

- Problem understanding and lending use-case description
- Input–output specification
- System architecture diagram (ML pipeline)
- Working local application with basic UI
- Brief evaluation of model performance

Milestone 2: Agentic AI Lending Decision Support Assistant (End-Sem)

Objective

Extend the credit risk system into an agentic AI assistant that reasons about borrower profiles and supports responsible lending decisions.

Functional Requirements

- Analyze credit risk predictions and contributing factors
- Retrieve relevant financial regulations and lending guidelines
- Generate structured lending recommendations
- Handle incomplete or noisy borrower data gracefully

Technical Requirements

- Open-source or free-tier LLM integration
- Retrieval-augmented generation (optional but recommended)
- Agentic workflow with explicit state management
- Prompt strategies to reduce biased or unsupported decisions

Structured Output Requirements

The generated lending assessment report must include:

- Borrower profile summary
- Credit risk analysis
- Recommended lending decision
- Risk mitigation suggestions
- Supporting sources or references
- Legal and ethical disclaimers

Extension (Choose Any One)

- Scenario-based lending analysis
- Portfolio-level credit risk overview
- PDF export of lending assessment report
- Explainable AI visualizations

Final Deliverables

- Publicly accessible hosted application link
- GitHub repository with complete codebase
- Demo video (5–7 minutes)

11 Project 11: Intelligent News Credibility Analysis and Agentic Misinformation Monitoring System

Project Objective

Design and implement an AI-driven content analysis system that evaluates the credibility of news articles and extends into an agentic AI assistant that generates structured credibility assessments and supports misinformation monitoring.

Constraints and Requirements

- Team size: **3–4 students**
- Paid APIs are **not permitted**
- Only free-tier APIs or open-source models may be used
- A user interface is **mandatory**
- Final application must be **publicly hosted**
- Localhost-only demonstrations will not be accepted

Recommended Tools (Not Mandatory)

- NLP / ML libraries: scikit-learn, NLTK, spaCy
- LLMs (Milestone 2): Open-source models or free-tier APIs
- Agent workflow: LangGraph (recommended)
- Vector stores (optional): Chroma, FAISS
- UI: Streamlit or Gradio
- Hosting: Hugging Face Spaces, Streamlit Community Cloud, Render (free tier)

11.1 Dataset

- `https://www.kaggle.com/datasets/saurabhshahane/fake-news-classification`

Milestone 1: ML-Based News Credibility Classification (Mid-Sem)

Objective

Build a machine learning and NLP-based system that classifies news articles based on credibility or misinformation risk using textual features.

Inputs

- News articles in text format or via URLs

Functional Requirements

- Perform text preprocessing and feature extraction
- Classify articles by credibility level or misinformation risk
- Identify key textual patterns influencing predictions
- Display credibility scores through a basic UI

Technical Requirements

- Text feature extraction using TF-IDF or embeddings
- Supervised learning models (e.g., Logistic Regression, Decision Trees)
- Model evaluation using precision, recall, F1 score, and confusion matrix

User Interface Requirements

- Input or upload news articles or URLs
- Display credibility classification and confidence
- Show brief explanation of detected patterns

Milestone 1 Deliverables

- Problem understanding and media use-case description
- Input–output specification
- System architecture diagram (NLP pipeline)
- Working local application with basic UI
- Brief evaluation of model performance

Milestone 2: Agentic AI Misinformation Monitoring Assistant (End-Sem)

Objective

Extend the credibility classification system into an agentic AI assistant that reasons about content credibility and generates structured, evidence-based credibility assessments.

Functional Requirements

- Analyze credibility predictions and risk indicators
- Retrieve fact-checking and verification sources
- Generate structured credibility assessment reports
- Handle incomplete or unreliable content gracefully

Technical Requirements

- Open-source or free-tier LLM integration
- Retrieval-augmented generation (optional but recommended)
- Agentic workflow with explicit state management
- Prompt strategies to reduce unsupported claims

Structured Output Requirements

The generated credibility assessment report must include:

- Article summary
- Credibility indicators and risk factors
- Cross-source verification findings
- Confidence assessment
- Supporting sources or references
- Ethical and misinformation disclaimers

Extension (Choose Any One)

- Trend analysis across multiple articles
- Real-time news monitoring dashboard
- PDF export of credibility report
- Comparative analysis across sources

Final Deliverables

- Publicly accessible hosted application link
- GitHub repository with complete codebase
- Demo video (5–7 minutes)

12 Project 12: Intelligent Exam Question Analysis and Agentic Assessment Design System

Project Objective

Design and implement an AI-driven educational analytics system that analyzes exam questions and student responses to assess question quality and difficulty, and extend it into an agentic AI assistant that supports assessment design and improvement.

Constraints and Requirements

- Team size: **3–4 students**
- Paid APIs are **not permitted**
- Only free-tier APIs or open-source models may be used
- A user interface is **mandatory**
- Final application must be **publicly hosted**
- Localhost-only demonstrations will not be accepted

Recommended Tools (Not Mandatory)

- ML / NLP libraries: scikit-learn, NLTK, spaCy
- LLMs (Milestone 2): Open-source models or free-tier APIs
- Agent workflow: LangGraph (recommended)
- Vector stores (optional): Chroma, FAISS
- UI: Streamlit or Gradio
- Hosting: Hugging Face Spaces, Streamlit Community Cloud, Render (free tier)

12.1 Dataset

- `https://www.kaggle.com/datasets/stackoverflow/stackoverflow`

Milestone 1: ML-Based Exam Question Analytics (Mid-Sem)

Objective

Build a machine learning and NLP-based system that evaluates exam questions using historical student response data and textual features.

Inputs

- Exam questions in text format
- Student response data (correct/incorrect or scores)

Functional Requirements

- Perform text preprocessing and feature extraction
- Predict question difficulty or quality category
- Analyze student performance patterns per question
- Display analytical insights through a basic UI

Technical Requirements

- Text feature extraction using TF-IDF or embeddings
- Supervised learning models (e.g., Logistic Regression, Decision Trees)
- Model evaluation using accuracy, precision, recall, and confusion matrix

User Interface Requirements

- Upload exam questions and response data
- Display difficulty classifications and statistics
- Visualize question-wise performance trends (optional)

Milestone 1 Deliverables

- Problem understanding and education use-case description
- Input–output specification
- System architecture diagram (ML/NLP pipeline)
- Working local application with basic UI
- Brief evaluation of model performance

Milestone 2: Agentic AI Assessment Design Assistant (End-Sem)

Objective

Extend the exam analytics system into an agentic AI assistant that reasons about assessment quality and generates structured recommendations for improving exam design.

Functional Requirements

- Analyze question difficulty distribution and learning gaps
- Retrieve pedagogical and assessment design best practices
- Generate structured assessment improvement recommendations
- Handle incomplete or noisy assessment data gracefully

Technical Requirements

- Open-source or free-tier LLM integration
- Retrieval-augmented generation (optional but recommended)
- Agentic workflow with explicit state management
- Prompt strategies to reduce unsupported educational claims

Structured Output Requirements

The generated assessment design report must include:

- Assessment quality summary
- Question difficulty distribution
- Identified learning gaps
- Recommended assessment improvements
- Supporting pedagogical references
- Educational and ethical disclaimers

Extension (Choose Any One)

- Automated question generation
- Adaptive difficulty analysis
- PDF export of assessment report
- Multi-exam comparative analysis

Final Deliverables

- Publicly accessible hosted application link
- GitHub repository with complete codebase
- Demo video (5–7 minutes)

13 Project 13: Intelligent Solar Energy Generation Forecasting and Agentic Grid Optimization System

Project Objective

Design and implement an AI-driven analytics system that forecasts solar energy generation using historical and weather data, and extend it into an agentic AI assistant that generates structured grid optimization and energy utilization recommendations.

Constraints and Requirements

- Team size: **3–4 students**
- Paid APIs are **not permitted**
- Only free-tier APIs or open-source models may be used
- A user interface is **mandatory**
- Final application must be **publicly hosted**
- Localhost-only demonstrations will not be accepted

Recommended Tools (Not Mandatory)

- ML libraries: scikit-learn, pandas, NumPy
- Time-series tools (optional): statsmodels, Prophet
- LLMs (Milestone 2): Open-source models or free-tier APIs
- Agent workflow: LangGraph (recommended)
- Vector stores (optional): Chroma, FAISS
- UI: Streamlit or Gradio
- Hosting: Hugging Face Spaces, Streamlit Community Cloud, Render (free tier)

13.1 Dataset

- `https://www.kaggle.com/datasets/stucom/solar-energy-power-generation-dataset`

Milestone 1: ML-Based Solar Energy Forecasting (Mid-Sem)

Objective

Build a machine learning or time-series forecasting system that predicts short-term or long-term solar energy generation.

Inputs

- Solar power generation data
- Weather indicators such as irradiance, temperature, and cloud cover
- Time-based features (hour, day, season)

Functional Requirements

- Perform data preprocessing and feature engineering
- Forecast solar energy generation
- Analyze trends and seasonality
- Display predictions through a basic UI

Technical Requirements

- Regression or time-series models (e.g., Linear Regression, ARIMA)
- Optional use of advanced forecasting models
- Model evaluation using MAE and RMSE

User Interface Requirements

- Upload or select solar generation data
- Display forecasted energy values
- Visualize trends and prediction curves

Milestone 1 Deliverables

- Problem understanding and renewable energy use-case description
- Input–output specification
- System architecture diagram (forecasting pipeline)
- Working local application with basic UI
- Brief evaluation of forecasting accuracy

Milestone 2: Agentic AI Grid Optimization Assistant (End-Sem)

Objective

Extend the forecasting system into an agentic AI assistant that reasons about energy generation variability and generates grid optimization recommendations.

Functional Requirements

- Analyze solar generation forecasts and variability
- Retrieve grid management and renewable integration guidelines
- Generate structured energy optimization recommendations
- Handle incomplete data or forecast uncertainty gracefully

Technical Requirements

- Open-source or free-tier LLM integration
- Retrieval-augmented generation (optional but recommended)
- Agentic workflow with explicit state management
- Prompt strategies to reduce unsupported operational claims

Structured Output Requirements

The generated grid optimization report must include:

- Solar generation forecast summary
- Identified variability and risk periods
- Grid balancing and storage recommendations
- Energy utilization optimization strategies
- Supporting references

Extension (Choose Any One)

- Battery storage optimization analysis
- Multi-site solar forecasting
- PDF export of grid optimization report
- Scenario-based energy planning

Final Deliverables

- Publicly accessible hosted application link
- GitHub repository with complete codebase
- Demo video (5–7 minutes)

14 Project 14: Intelligent Player Churn Prediction and Agentic Game Engagement Optimization System

Project Objective

Design and implement an AI-driven gaming analytics system that predicts player churn using gameplay and engagement data, and extend it into an agentic AI assistant that generates structured engagement and retention strategies.

Constraints and Requirements

- Team size: **3–4 students**
- Paid APIs are **not permitted**
- Only free-tier APIs or open-source models may be used
- A user interface is **mandatory**
- Final application must be **publicly hosted**
- Localhost-only demonstrations will not be accepted

Recommended Tools (Not Mandatory)

- ML libraries: scikit-learn, pandas, NumPy
- LLMs (Milestone 2): Open-source models or free-tier APIs
- Agent workflow: LangGraph (recommended)
- Vector stores (optional): Chroma, FAISS
- UI: Streamlit or Gradio
- Hosting: Hugging Face Spaces, Streamlit Community Cloud, Render (free tier)

14.1 Dataset

- `https://www.kaggle.com/datasets/rabieelkharoua/predict-online-gaming-behavior-dataset`

Milestone 1: ML-Based Player Churn Prediction (Mid-Sem)

Objective

Build a machine learning system that predicts player churn using historical gameplay and engagement data.

Inputs

- Player behavior data (CSV), such as:
  - Session frequency and duration
  - Game progression levels
  - In-game actions or events
  - Engagement metrics

Functional Requirements

- Perform data preprocessing and feature engineering
- Predict player churn probability or classification
- Identify key engagement-related churn factors
- Display predictions through a basic UI

Technical Requirements

- Supervised learning models (e.g., Logistic Regression, Decision Trees)
- Optional use of ensemble models or neural networks
- Model evaluation using classification metrics

User Interface Requirements

- Upload player data
- Display churn predictions and risk levels
- Show basic feature importance or explanations

Milestone 1 Deliverables

- Problem understanding and gaming use-case description
- Input–output specification
- System architecture diagram (ML pipeline)
- Working local application with basic UI
- Brief evaluation of model performance

Milestone 2: Agentic AI Game Engagement Optimization Assistant (End-Sem)

Objective

Extend the churn prediction system into an agentic AI assistant that reasons about player behavior and generates personalized engagement strategies.

Functional Requirements

- Analyze churn risk predictions and engagement patterns
- Retrieve game engagement and retention strategies
- Generate structured engagement optimization recommendations
- Handle incomplete or noisy gameplay data gracefully

Technical Requirements

- Open-source or free-tier LLM integration
- Retrieval-augmented generation (optional but recommended)
- Agentic workflow with explicit state management
- Prompt strategies to reduce unsupported recommendations

Structured Output Requirements

The generated engagement optimization report must include:

- Player behavior summary
- Churn risk interpretation
- Engagement and retention recommendations
- Supporting references
- Ethical and user-experience disclaimers

Extension (Choose Any One)

- Personalized in-game reward strategies
- Adaptive difficulty or content recommendations
- PDF export of engagement report
- Multi-player segmentation analysis

Final Deliverables

- Publicly accessible hosted application link
- GitHub repository with complete codebase
- Demo video (5–7 minutes)

15 Project 15: Intelligent EV Charging Demand Prediction and Agentic Charging Infrastructure Planning System

Project Objective

Design and implement an AI-driven analytics system that predicts electric vehicle (EV) charging demand using historical charging data, and extend it into an agentic AI assistant that generates structured recommendations for charging infrastructure planning.

Constraints and Requirements

- Team size: **3–4 students**
- Paid APIs are **not permitted**
- Only free-tier APIs or open-source models may be used
- A user interface is **mandatory**
- Final application must be **publicly hosted**
- Localhost-only demonstrations will not be accepted

Recommended Tools (Not Mandatory)

- ML libraries: scikit-learn, pandas, NumPy
- Time-series tools (optional): statsmodels
- LLMs (Milestone 2): Open-source models or free-tier APIs
- Agent workflow: LangGraph (recommended)
- Vector stores (optional): Chroma, FAISS
- UI: Streamlit or Gradio
- Hosting: Hugging Face Spaces, Streamlit Community Cloud, Render (free tier)

15.1 Dataset

- `https://datadryad.org/dataset/doi:10.5061/dryad.np5hqc04z`

Milestone 1: ML-Based EV Charging Demand Prediction (Mid-Sem)

Objective

Build a machine learning or time-series forecasting system that predicts EV charging demand at charging stations using historical usage data.

Inputs

- Charging session data (CSV), such as:
  - Charging timestamps and duration
  - Location or station identifiers
  - Energy consumption per session
  - Time-based features (hour, day, season)

Functional Requirements

- Perform data preprocessing and feature engineering
- Predict EV charging demand or load levels
- Identify peak demand periods and trends
- Display predictions through a basic UI

Technical Requirements

- Regression or time-series models (e.g., Linear Regression, ARIMA)
- Optional use of advanced forecasting models
- Model evaluation using MAE and RMSE

User Interface Requirements

- Upload EV charging data
- Display demand predictions and trends
- Visualize peak usage periods

Milestone 1 Deliverables

- Problem understanding and EV infrastructure use-case description
- Input–output specification
- System architecture diagram (forecasting pipeline)
- Working local application with basic UI
- Brief evaluation of prediction accuracy

Milestone 2: Agentic AI EV Infrastructure Planning Assistant (End-Sem)

Objective

Extend the demand prediction system into an agentic AI assistant that reasons about charging demand patterns and generates infrastructure planning recommendations.

Functional Requirements

- Analyze predicted charging demand and peak usage patterns
- Retrieve EV infrastructure planning guidelines
- Generate structured infrastructure and scheduling recommendations
- Handle incomplete or noisy charging data gracefully

Technical Requirements

- Open-source or free-tier LLM integration
- Retrieval-augmented generation (optional but recommended)
- Agentic workflow with explicit state management
- Prompt strategies to reduce unsupported planning claims

Structured Output Requirements

The generated EV infrastructure planning report must include:

- Charging demand summary
- Identification of high-load locations
- Infrastructure expansion recommendations
- Scheduling and load-balancing insights
- Supporting references

Extension (Choose Any One)

- Station placement optimization
- Multi-location demand comparison
- PDF export of infrastructure planning report
- Scenario-based demand forecasting

Final Deliverables

- Publicly accessible hosted application link
- GitHub repository with complete codebase
- Demo video (5–7 minutes)