INDIAN INSTITUTE OF TECHNOLOGY MADRAS

COURSE PROJECT GUIDELINES

Z5007: Programming and Data Structures

M.Tech Data Science & Artificial Intelligence

Project Weight: 25% of Course Grade

Total Project Score: 100 Points

Odd Semester 2025 - 2026

Project Overview

The course project is a comprehensive assignment designed to assess your ability to apply data structures and algorithms to solve real-world problems in Data Science and Artificial Intelligence. This project accounts for 25% of your final course grade and is scored out of 100 points (which will be scaled to 25% in final grading).

Key Information

- **Team Size:** Individual or pairs (maximum 2 students)
- **Total Points:** 100 (scales to 25% of course grade)
- **Duration:** 9 weeks (Week 6 to Week 14)
- **Programming Language:** Python only
- **Submission Mode:** Through course portal

Learning Objectives

By completing this project, you will:

1. Design and implement complex data structures from scratch
2. Analyze time and space complexity of algorithms
3. Apply data structures to solve practical AI/DS problems
4. Write clean, well-documented, and efficient code
5. Present technical work clearly and professionally
6. Compare custom implementations with industry-standard solutions

Project Options

Choose **ONE** of the following project options:

Option 1: Machine Learning Algorithm from Scratch

Overview: Implement a complete machine learning algorithm without using scikit-learn or similar ML libraries. You must build all components using fundamental data structures.

Approved Algorithms:

1. **Decision Tree Classifier**
   - Use tree data structure
   - Implement ID3 or CART algorithm
   - Support pruning
   - Handle categorical and numerical features
2. **K-Nearest Neighbors (KNN)**
   - Implement KD-Tree for efficient neighbor search
   - Support multiple distance metrics
   - Handle weighted voting
   - Optimize for large datasets
3. **K-Means Clustering**
   - Use efficient distance calculations
   - Implement K-Means++ initialization
   - Support elbow method for optimal K
   - Handle outliers
4. **Naive Bayes Classifier**
   - Use hash tables for probability storage
   - Support Gaussian, Multinomial, and Bernoulli variants
   - Implement Laplace smoothing
   - Handle missing values

Minimum Requirements:

- No ML libraries (numpy and pandas allowed for data manipulation only)
- Must implement core data structures (trees, heaps, hash tables as needed)
- Train and test on real dataset (minimum 1000 samples)
- Achieve reasonable accuracy (within 5% of sklearn on same dataset)
- Provide performance comparison with sklearn
- Include complexity analysis

Option 2: Recommendation System

Overview: Build a complete recommendation engine that uses multiple data structures and algorithms to provide personalized recommendations.

Approved System Types:

1. **Collaborative Filtering Recommender**
   - User-based or item-based filtering
   - Use graphs for user/item relationships
   - Use hash tables for fast lookups
   - Support similarity metrics (cosine, Pearson)
2. **Content-Based Recommender**
   - Use TF-IDF with hash tables
   - Implement inverted index
   - Support text and numerical features
   - Use priority queues for ranking
3. **Hybrid Recommender**
   - Combine collaborative and content-based
   - Use weighted scoring
   - Implement cold-start handling
   - Support real-time updates

Minimum Requirements:

- Support minimum 1000 items and 500 users
- Use at least 3 different data structures (graph, hash table, heap/priority queue)
- Provide recommendations in under 1 second
- Include evaluation metrics (precision, recall, NDCG)
- Support new user/item addition
- Include complexity analysis

Option 3: Graph-Based System

Overview: Implement a system that heavily uses graph algorithms and data structures.

Approved Systems:

1. **Social Network Analyzer**
   - Find communities (clustering algorithms)
   - Identify influencers (centrality measures)
   - Suggest connections (link prediction)
   - Analyze network properties
2. **Route Optimization System**

- Implement Dijkstra and A* algorithms
- Support multiple constraints (time, distance, cost)
- Handle real-time updates
- Visualize routes

3. **Knowledge Graph Query System**
    - Build graph from structured data
    - Support complex queries
    - Implement graph traversal algorithms
    - Find patterns and relationships

Minimum Requirements:

- Graph with minimum 500 nodes and 2000 edges
- Implement at least 2 graph algorithms from scratch
- Use adjacency list/matrix appropriately
- Provide visualization of results
- Include performance benchmarks
- Include complexity analysis

Project Timeline and Milestones

The project spans 9 weeks with three major milestones:

Milestone	Due Date	Points	Deliverable
Proposal	Nov 25, 2025	10	Project proposal document
Progress Report	Dec 23, 2025	30	Working code + progress doc
Final Submission	Jan 20, 2026	60	Complete code + doc + presentation
		Total:	**100 Points**

Milestone 1: Project Proposal (10 Points)

Due: November 25, 2025, End of Lab Session (Week 6)

What to Submit:

- Proposal document (3-4 pages) using provided template
- Must include IIT Madras title page
- Submit as PDF through course Google Drive

Content Requirements:

1. **Title Page** (use provided template with IIT Madras logo)
2. **Team Information**
    - Team member names and roll numbers

- Email addresses
- Whether working individually or in pair

3. **Project Selection**
   - Which option chosen (1, 2, or 3)
   - Specific algorithm/system you will build
   - Why you chose this project
4. **Technical Approach**
   - Data structures you will use (with justification)
   - Algorithms you will implement
   - Expected time and space complexity
   - High-level system architecture diagram
5. **Dataset**
   - Dataset you will use (name, source, size)
   - Description of data (features, samples)
   - Link to dataset or how to obtain it
   - Data preprocessing requirements
6. **Implementation Plan**
   - Week-by-week breakdown of tasks
   - Which team member will do what (for pairs)
   - Testing strategy
7. **Expected Challenges**
   - Technical difficulties you anticipate
   - How you plan to address them
   - Backup plans if needed
8. **Success Criteria**
   - How will you measure success?
   - Benchmark metrics
   - Comparison baseline

Evaluation Criteria:

- Clarity and completeness: 4 points
- Technical feasibility: 3 points
- Appropriate scope for timeframe: 2 points
- Proper formatting and presentation: 1 point

Milestone 2: Progress Report (30 Points)

Due: December 23, 2025, End of Lab Session (Week 10)

What to Submit:

- Working code (Python files in organized folder)
- Progress report document (4-5 pages)
- Sample outputs/screenshots
- Submit as ZIP file through course portal

Code Requirements (20 points):

1. **Core Data Structures Implemented** (8 points)
    - At least 2 major data structures complete
    - Proper class structure
    - Key operations working
2. **Basic Functionality Working** (8 points)
    - Can load and process data
    - Core algorithm partially working
    - Produces some output
3. **Code Quality** (4 points)
    - Clean code structure
    - Meaningful variable names
    - Basic comments included
    - No critical bugs

Document Requirements (10 points):

1. **Progress Summary** (3 points)
    - What has been completed
    - What percentage of project is done
    - Current status of each component
2. **Technical Details** (4 points)
    - Data structures implemented
    - Algorithms implemented
    - Time complexity analysis of implemented parts
    - Design decisions and rationale
3. **Challenges and Solutions** (2 points)
    - Problems encountered
    - How you solved them
    - Current blockers

4. **Remaining Work** (1 point)
    - What still needs to be done
    - Updated timeline
    - Confidence in meeting deadline

Important Notes:

- Code must run without errors
- Include README with instructions to run
- If using external data, include sample data file
- Late submission: NO late submission allowed

Milestone 3: Final Submission (60 Points)

Due: January 20, 2026, End of Lab Session (Week 14)
What to Submit:

1. Complete Python code (organized in folders)
2. README file with setup and usage instructions
3. Technical report (8-10 pages)
4. Presentation slides (PDF or PPTX)
5. Sample dataset or link to dataset
6. Demo video (3-5 minutes, optional but recommended)

Submit as single ZIP file: `ProjectName_RollNumber.zip`
Evaluation Breakdown:

A. Code Implementation (30 points)

1. **Correctness and Completeness** (12 points)
    - All required features implemented
    - Algorithm works correctly
    - Handles edge cases
    - Produces accurate results
2. **Data Structure Implementation** (10 points)
    - Custom data structures built from scratch
    - Appropriate choice of structures
    - Efficient implementation
    - Proper integration
3. **Code Quality** (5 points)
    - Clean, readable code
    - Proper organization (classes, functions)

- Meaningful names
- Consistent style

4. **Testing** (3 points)
    - Comprehensive test cases
    - Edge case handling
    - Test coverage

B. Documentation (15 points)

1. **Code Comments** (3 points)
    - Clear function/class documentation
    - Complex logic explained
    - Inline comments where needed
2. **README File** (3 points)
    - Clear setup instructions
    - Dependencies listed
    - Usage examples
    - Troubleshooting tips
3. **Technical Report** (9 points)
    - Problem description and motivation (1 point)
    - System architecture and design (2 points)
    - Data structures and algorithms explained (2 points)
    - Complexity analysis (2 points)
    - Experimental results and analysis (2 points)

C. Performance and Analysis (10 points)

1. **Efficiency** (4 points)
    - Runs in reasonable time
    - Handles large datasets
    - Memory efficient
2. **Complexity Analysis** (3 points)
    - Correct time complexity analysis
    - Space complexity analysis
    - Big-O notation properly used
3. **Benchmarking** (3 points)
    - Performance metrics provided
    - Comparison with baseline
    - Results properly visualized

D. Presentation (5 points)

Format: 10-minute demo + 5-minute Q&A

1. **Demo Quality** (2 points)
    - System works smoothly
    - Shows key features
    - Well-prepared
2. **Technical Explanation** (2 points)
    - Clear explanation of approach
    - Design decisions justified
    - Complexity discussed
3. **Q&A** (1 point)
    - Answers questions confidently
    - Shows deep understanding
    - Addresses concerns

Technical Requirements

Must Have:

- Python 3.8 or higher
- Code runs without errors
- All dependencies listed in requirements.txt
- Works on standard Python installation

Allowed Libraries:

- **Data manipulation:** numpy, pandas
- **Visualization:** matplotlib, seaborn
- **Basic utilities:** math, random, time, collections

NOT Allowed:

- scikit-learn (for ML projects)
- networkx (for graph projects - except for validation)
- Pre-built data structure libraries that do the work for you
- Any library that implements your core algorithm

Academic Integrity

What is Allowed:

- Discussing general approaches with classmates
- Using online resources for understanding concepts
- Consulting documentation for allowed libraries
- Getting help during office hours

What is NOT Allowed:

- Copying code from online sources
- Sharing code with other teams
- Using GitHub/Stack Overflow solutions directly
- Submitting code you did not write

Consequences:

- First offense: Zero on project (25% of course grade lost)
- Second offense: Fail the course
- Reported to institute administration

When in doubt, ask! Cite any external code you reference.

Grading Scale

Your project score (out of 100) will be scaled to 25% of your final course grade:

Project Score	Percentage	Course Grade Impact
90-100	Excellent	22.5-25%
80-89	Very Good	20-22.4%
70-79	Good	17.5-19.9%
60-69	Satisfactory	15-17.4%
50-59	Pass	12.5-14.9%
Below 50	Fail	Less than 12.5%

Resources

Datasets:

- **Kaggle:** `https://www.kaggle.com/datasets`
- **UCI ML Repository:** `https://archive.ics.uci.edu/ml`
- **MovieLens:** `https://grouplens.org/datasets/movielens/`
- **SNAP:** `https://snap.stanford.edu/data/` (for graphs)

Learning Resources:

- Course lecture notes and slides
- VisuAlgo.net for algorithm visualization
- Python documentation
- Course discussion forum

Getting Help:

- **Office Hours:** Tue 2-4 PM
- **Email:** innocent@iitmz.ac.in
- **Lab Sessions:** Use lab time for project work
- **Discussion Forum:** Post general questions

Submission Checklist

Before submitting, ensure you have:

For Proposal (Week 6):

☐ Title page with IIT Madras logo

☐ All team member information

☐ Project option clearly stated

☐ Technical approach described

☐ Dataset identified

☐ Implementation plan included

☐ Submitted as PDF

For Progress Report (Week 10):

☐ Code runs without errors

☐ README included

☐ Core data structures implemented

☐ Progress document (4-5 pages)

☐ Sample outputs included

☐ Submitted as ZIP file

For Final Submission (Week 14):

- ☐ Complete, working code
- ☐ README with full instructions
- ☐ requirements.txt file
- ☐ Technical report (8-10 pages)
- ☐ Presentation slides
- ☐ Sample dataset or link
- ☐ All files in organized folders
- ☐ Submitted as ZIP file
- ☐ File name: ProjectName_RollNumber.zip

Important Dates Summary

Milestone	Date	Week
Proposal Due	November 25, 2025	Week 6
Progress Report Due	December 23, 2025	Week 10
Final Submission	January 20, 2026	Week 14
Presentations	January 20-22, 2026	Week 14

Late Submission Policy:

- Up to 24 hours late: 10% penalty
- 24-48 hours late: 20% penalty
- More than 48 hours: Not accepted (0 points)

Questions?
Contact your instructor.

Good luck with your projects!