Distributed LMS with Raft Consensus - Project Structure

```
distributed-lms/
│
├── README.md                     # Project documentation
├── requirements.txt              # Python dependencies
├── config.json                   # System configuration
├── setup.py                      # Installation script
│
├── protos/                       # Protocol Buffer definitions
│   ├── lms.proto                # LMS service definitions
│   ├── raft.proto               # Raft protocol definitions
│   └── tutoring.proto           # Tutoring service definitions
│
├── raft/                        # Raft consensus implementation
│   ├── __init__.py
│   ├── node.py                  # Raft node implementation
│   ├── state.py                 # Node states (Follower, Candidate, Leader)
│   ├── log.py                   # Raft log implementation
│   ├── storage.py               # Persistent storage for Raft
│   ├── rpc.py                   # Raft RPC handlers
│   ├── config.py                # Raft configuration
│   └── snapshot.py              # Log compaction and snapshots
│
├── lms/                         # LMS server implementation
│   ├── __init__.py
│   ├── server.py                # Main LMS server
│   ├── auth.py                  # Authentication and token management
│   ├── database.py              # Data models and storage
│   ├── course_manager.py        # Course content management
│   ├── assignment_manager.py    # Assignment handling
│   ├── grade_manager.py         # Grading system
│   └── query_manager.py         # Student queries handling
│
├── tutoring/                    # Tutoring server implementation
│   ├── __init__.py
│   ├── server.py                # Tutoring server with Ollama
│   ├── llm_client.py            # Ollama integration
│   ├── context_builder.py       # Context preparation for LLM
│   └── adaptive_tutor.py        # Adaptive difficulty based on student level
│
├── client/                      # Client implementations
│   ├── __init__.py
│   ├── student_client.py        # Student client application
│   ├── instructor_client.py     # Instructor client application
│   └── base_client.py           # Base client functionality
│
├── common/                      # Shared utilities
```

```
│   ├── __init__.py
│   ├── logger.py                # Logging configuration
│   ├── utils.py                 # Common utilities
│   └── constants.py             # System constants
│
├── data/                        # Data storage directory
│   ├── raft_logs/              # Raft log storage
│   ├── raft_snapshots/         # Raft snapshots
│   ├── course_materials/       # PDF and TXT files
│   ├── assignments/            # Student submissions
│   └── database/               # Application data
│
├── scripts/                     # Utility scripts
│   ├── start_cluster.py        # Start all nodes
│   ├── stop_cluster.py         # Stop all nodes
│   ├── test_raft.py           # Raft testing utilities
│   └── generate_protos.sh      # Generate Python from protos
│
└── tests/                      # Test suite
    ├── __init__.py
    ├── test_raft.py           # Raft consensus tests
    ├── test_lms.py            # LMS functionality tests
    └── test_integration.py     # Full system integration tests
```

Key Components:

1. **Raft Implementation** - Full consensus protocol with:
    - Leader election
    - Log replication
    - Log compaction and snapshots
    - Network partition handling
    - Dynamic membership based on online nodes
2. **LMS Server** - Manages all course operations with Raft consistency
3. **Tutoring Server** - Ollama integration with adaptive difficulty
4. **Clients** - Separate implementations for students and instructors
5. **Storage** - JSON-based persistent storage for Raft logs and application data