Course File Manager System (CFMS)

Functional Requirement Document (FRD)

Version: 1.0
Project Type: Industrial Level Web Application
System Name: Course File Manager System (CFMS)

---

1. Introduction

The **Course File Manager System (CFMS)** is a web-based platform developed for academic institutes to digitally manage **course files**. A course file contains structured documents related to a course such as syllabus, lesson plan, attendance, assignments, question banks, exam papers, result analysis, CO-PO mapping, etc.

The system ensures:

- Standard template-based course file structure
- Controlled approval workflow
- Comment-based communication (like chat)
- Notification and progress tracking
- Document versioning
- Department-level monitoring and reporting

---

2. System Users & Dashboards

The system contains **four user dashboards**:

1. **Admin**
2. **HOD (Head of Department)**
3. **Subject Head (Course Coordinator / Incharge)**
4. **Teacher**

Each dashboard provides features based on role permissions.

---

3. User Roles & Responsibilities

3.1 Admin (Super Administrator)

Objective: Institute-level system configuration and master management.

Admin Responsibilities:

- Create and manage Institute records

- Create departments
- Create teacher profiles and user accounts
- Assign roles (Admin, HOD, Subject Head, Teacher)
- Setup academic structure:
    - Program
    - Branch
    - Semester
    - Courses (optional centralized approach)
- Monitor overall system usage and logs

Admin Key Modules:

- Institute Management
- Department Management
- Teacher & User Management
- Program/Branch/Semester Setup
- Course Setup (if allowed at Admin level)
- Reports & Audit logs

3.2 HOD Dashboard (Department Authority)

Objective: Department governance, template standardization, course assignment, final approval.

✅ **HOD has major control inside the department.**

HOD Responsibilities:

A) Department Template Management

- Create and manage department templates
- Define course file structure (headings)
- Define checklist requirements (mandatory documents)
- Update templates for each academic year if needed

B) Course Management

- Add and manage department-level courses (if HOD is permitted)
- Map courses to programs/branch/semester

C) Assign Subject Head

- Assign subject head (course coordinator) for each course

D) Assign Course Teachers

- Assign teachers for specific course / academic year / section (stored in **Course_Teacher** table)

E) Final Approval

- Review course files after Subject Head approval
- Approve / Return / Reject with official remarks
- Ensure all requirements and checklist are completed

HOD Dashboard Features

- Template creation & update
- Course assignment module
- Teacher assignment module
- Final approvals module
- Department progress monitoring
- Reporting module

3.3 Subject Head Dashboard (Course Authority)

Objective: Course-level verification and first-stage approval.

Subject Head Responsibilities:

- Receive course files submitted by teachers
- Review course file completeness and document quality
- Add comments at multiple levels (course/heading/document)
- Return file for correction OR approve and forward to HOD

Subject Head Dashboard Features

- Course file review panel
- Commenting module
- Document validation module
- Approval (Stage 1) module
- Forward-to-HOD module

3.4 Teacher Dashboard (Course File Creator)

Objective: Create and maintain course files with all required documents.

Teacher Responsibilities:

- Create course file for assigned courses
- Upload documents under headings
- Maintain file versions (updated document uploads)
- Respond to comments from Subject Head and HOD
- Submit for approval

Teacher Dashboard Features

- My courses (assigned courses list)
- Create course file (academic year)
- Upload documents under headings
- Comment replies and discussion thread
- Submit for approval
- Status tracking

---

4. Core Workflow (End-to-End) (Rpeat hai Rohit deeply understood karney key liy)

4.1 Master Setup Flow

Actor: Admin + HOD

1. Admin creates institute, departments, teacher profiles
2. Admin creates users and assigns roles
3. Admin/HOD sets academic structure (Program, Branch, Semester, Course)
4. HOD creates department template and checklist

✅ System becomes ready for usage.

4.2 Course Assignment Flow

Actor: HOD

1. HOD selects course
2. Assigns Subject Head for that course
3. Assigns course teacher(s) with academic year and section
4. Mapping stored in **Course_Teacher table**

✅ Teachers get assigned course visible in dashboard.

4.3 Course File Creation Flow

Actor: Teacher

1. Teacher selects assigned course + academic year
2. Teacher creates course file
3. Record added in **Course_File**
4. Default status: Draft

✅ Course File instanceF created.

4.4 Template → Headings Generation Flow

Actor: System

When a course file is created:

1. System loads HOD template
2. Template headings copied into Heading table
3. Nested structure is supported (parent_heading_id)

✅ Teacher now sees structured headings ready for uploads.

4.5 Document Upload Flow

Actor: Teacher

Teacher uploads files heading-wise:

1. Document stored in server/storage
2. DB record inserted in **Document**
3. Version is tracked using version_no

✅ Documents become part of course file.

4.6 Submission Flow

Actor: Teacher

1. Teacher clicks **Submit**
2. Status updates: Submitted
3. Notification generated for Subject Head

✅ Course file moves to review stage.

4.7 Subject Head Approval Flow (Stage 1)

Actor: Subject Head

Subject Head reviews and decides:

- ✅ Approve → Forward to HOD
- ❌ Return/Reject → Teacher must fix issues

Approval record inserted in **Approval table** with:

- stage = SubjectHead
- status = Approved/Returned/Rejected

✅ File proceeds or returns.

4.8 HOD Approval Flow (Final Approval)

Actor: HOD

HOD reviews and decides:

- ✅ Final Approved → status = Approved
- ❌ Returned → Teacher must correct

Approval record inserted in **Approval table** with:

- stage = HOD
- status = Approved/Returned/Rejected

✅ Course file becomes approved.

---

5. Course File Status Lifecycle

Recommended industry-grade status lifecycle:

1. Draft
2. Submitted
3. Under_Review_SubjectHead
4. Returned_By_SubjectHead
5. Under_Review_HOD

6. Returned_By_HOD
7. Approved
8. Archived (optional)

Comment Process

---

6. Comment System Specification (Chat + Threads)

This is the most critical feature for your project.

The **Comment table** supports **multi-level commenting** and **threaded replies**, which means the system will behave like **a structured chat**.

6.1 Comment Levels Supported

A) Course File Level Comment (Overall Comment)

Used for general feedback on full course file.

✅ Examples:

- “Course file is correct.”
- “Course file is incomplete.”

DB Pattern

- course_file_id ✅
- heading_id = NULL

- document_id = NULL

B) Heading/Subheading Level Comment

Used for comments on a section.

✅ Examples:

- “Unit 3 notes missing.”
- “Lesson plan heading is incomplete.”

DB Pattern

- course_file_id ✅
- heading_id ✅
- document_id = NULL

C) Document Level Comment (Specific File Comment)

Used for specific document corrections.

✅ Examples:

- “Wrong academic year.”
- “Upload signed PDF.”
- “Replace file format.”

DB Pattern

- course_file_id ✅
- heading_id ✅
- document_id ✅

6.2 Threaded Replies (Chat System)

You have this in DB:

- parent_comment_id

This enables chat-like replies:

Example Thread:

1. Subject Head: “Wrong template used.”
2. Teacher Reply: “Okay, I will re-upload today.”

3. Subject Head Reply: “Please upload signed copy also.”

All replies stored in **same table** with parent linking.

✅ This becomes a complete discussion log.

6.3 Teacher Reply System

Teacher can reply to:

- Course file level comments
- Heading level comments
- Document level comments

Teacher replies are also stored in Comment table with:

- parent_comment_id = original comment id

6.4 Comment Visibility Rules

- Teacher sees all comments related to their course files
- Subject Head sees all comments on courses they supervise
- HOD sees all comments inside their department

6.5 Comment “Received” Feature

Comment table has:

- is_received

Meaning:

- When teacher views a comment thread → system marks it received
- Reviewer can confirm teacher has seen remarks

---

7. Approval System Specification

Approval and Comment are separated intentionally.

Approval Table = Official decision (stage wise)

Comment Table = Discussion & review communication

7.1 Approval Stages

Approval table stages:

- SubjectHead approval stage (Stage 1)
- HOD approval stage (Final)

Approval table stores:

- course_file_id
- approver_id
- stage
- status
- official remark/comment
- acted_at timestamp

✅ Provides approval audit trail.

---

8. Comment + Approval Combined Workflow

Case Example:

Teacher submits course file → goes to Subject Head

Subject Head can:

1. Add course file comment: "File is incomplete"
2. Add heading comment: "Unit 2 heading missing notes"
3. Add document comment: "Wrong PDF uploaded"

Then Subject Head selects decision:

- Return file → creates approval record (Returned)
  or
- Approve file → creates approval record (Approved)

Same pattern for HOD.

---

9. Notification System Specification (All Dashboards)

Notification table:

- user_id
- type

- payload(JSON)
- is_read

9.1 Notification Triggers

A) Course File Events

- Teacher submits → notify Subject Head
- Subject Head approves → notify HOD & Teacher
- HOD approves → notify Teacher & Subject Head
- Returned/Rejection → notify Teacher

B) Comment Events

- Subject Head comment → notify Teacher
- HOD comment → notify Teacher
- Teacher reply → notify Subject Head or HOD

C) Document Events

- Teacher uploads new version after return → notify reviewer

9.2 Notification Payload (Recommended)

```
{
  "course_file_id": 101,
  "heading_id": 22,
  "document_id": 12,
  "message": "New comment on Unit 2 Notes",
  "redirect_url": "/course-file/101/heading/22/document/12"
}
```

✅ Notification click takes user to exact page.

9.3 Notification Display Per Dashboard

Teacher Dashboard

- New comments
- Returned/Rejected updates
- Approval results

- Submission updates

Subject Head Dashboard

- Teacher submission alerts
- Teacher reply alerts
- Upload update alerts after return

HOD Dashboard

- Course forwarded by subject head
- Pending final approvals
- Teacher correction update alerts

---

10. Important Implementation Notes (For Coding)

✅ Comment module must support:

- filtering by course_file_id
- optionally filtering by heading_id/document_id
- threaded UI using parent_comment_id
- marking received

✅ Approval module must:

- record actions in Approval table
- update Course_File.status accordingly

✅ Notification module must:

- create notification entry
- update is_read on viewing
- support redirect URL

✅ Document module must:

- store version history
- allow multiple uploads under same heading
- show latest version by default