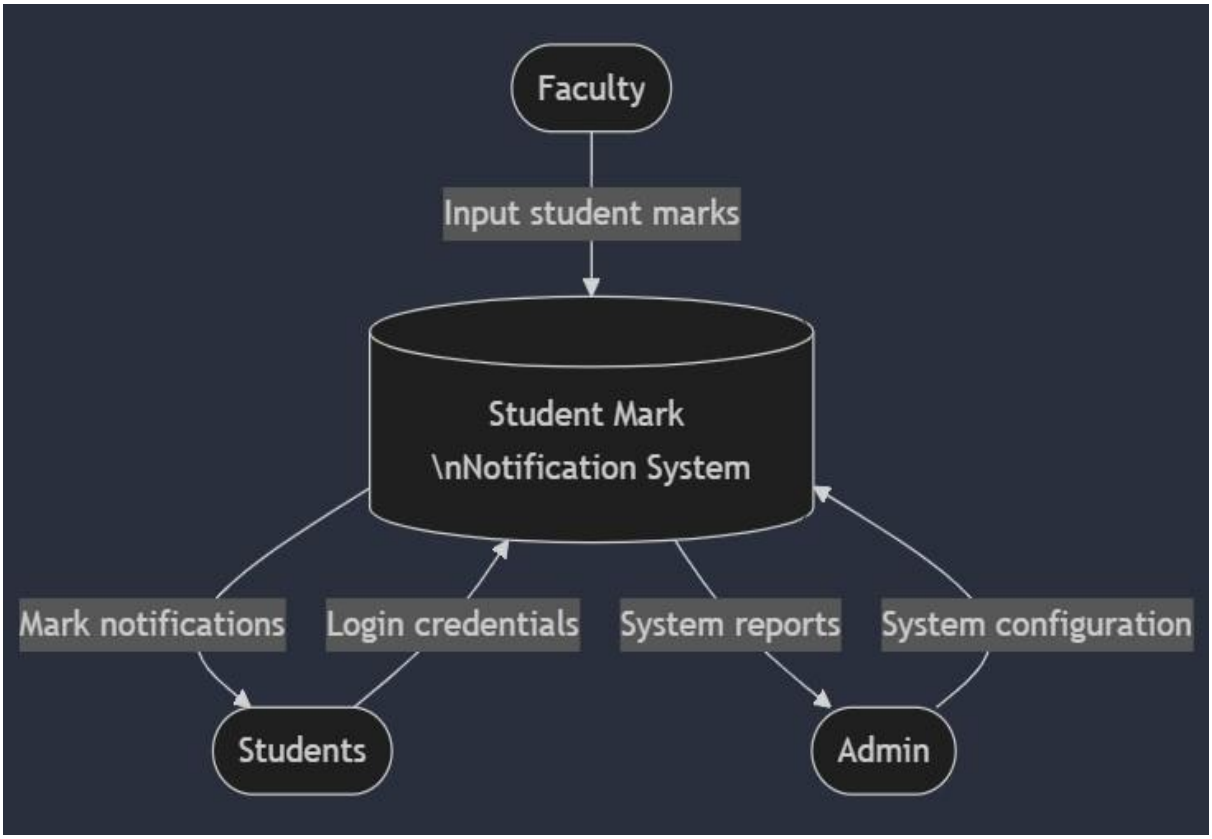
2.Requirement Analysis:

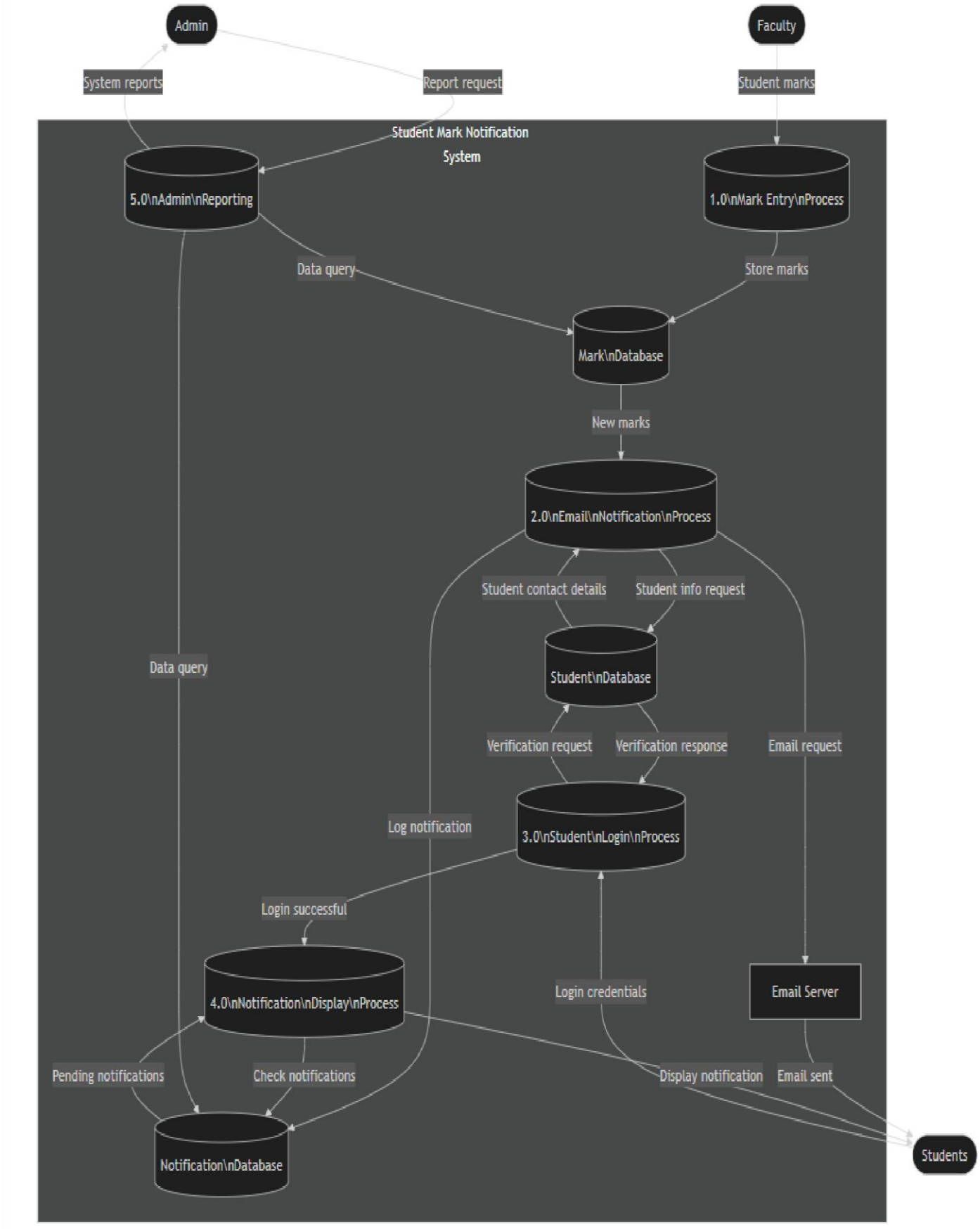
a) Flow models: **i. Level 0 DFD**



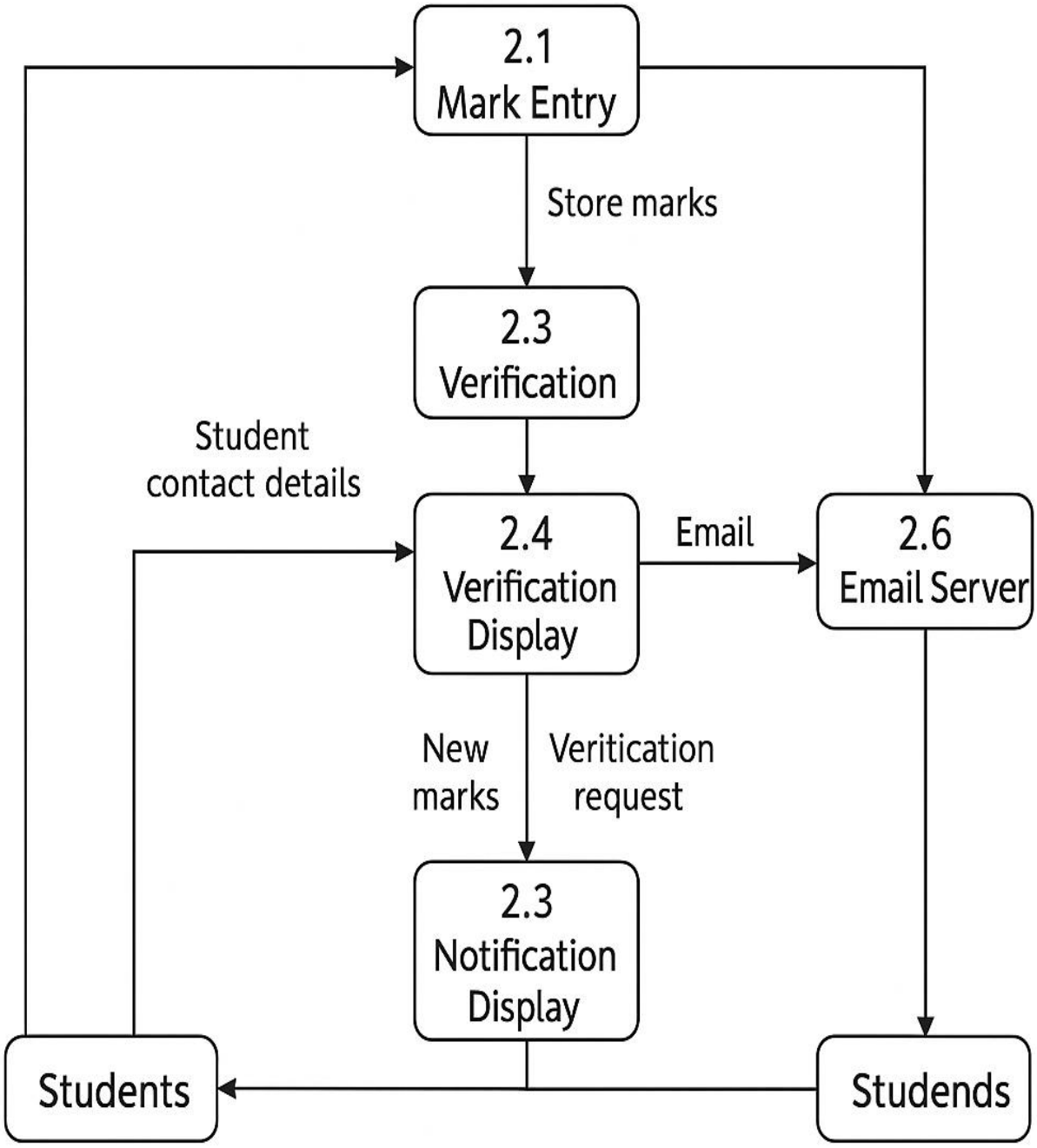
ii.Level 1 DFD

Breaks down the system into **detailed processes** like:

- **Faculty enters marks** → Stored in **database**.
- **Database triggers email system** → **Email sent to students**.
- **Students log in** → Retrieve **marks from database** → Show **pop-up notification**

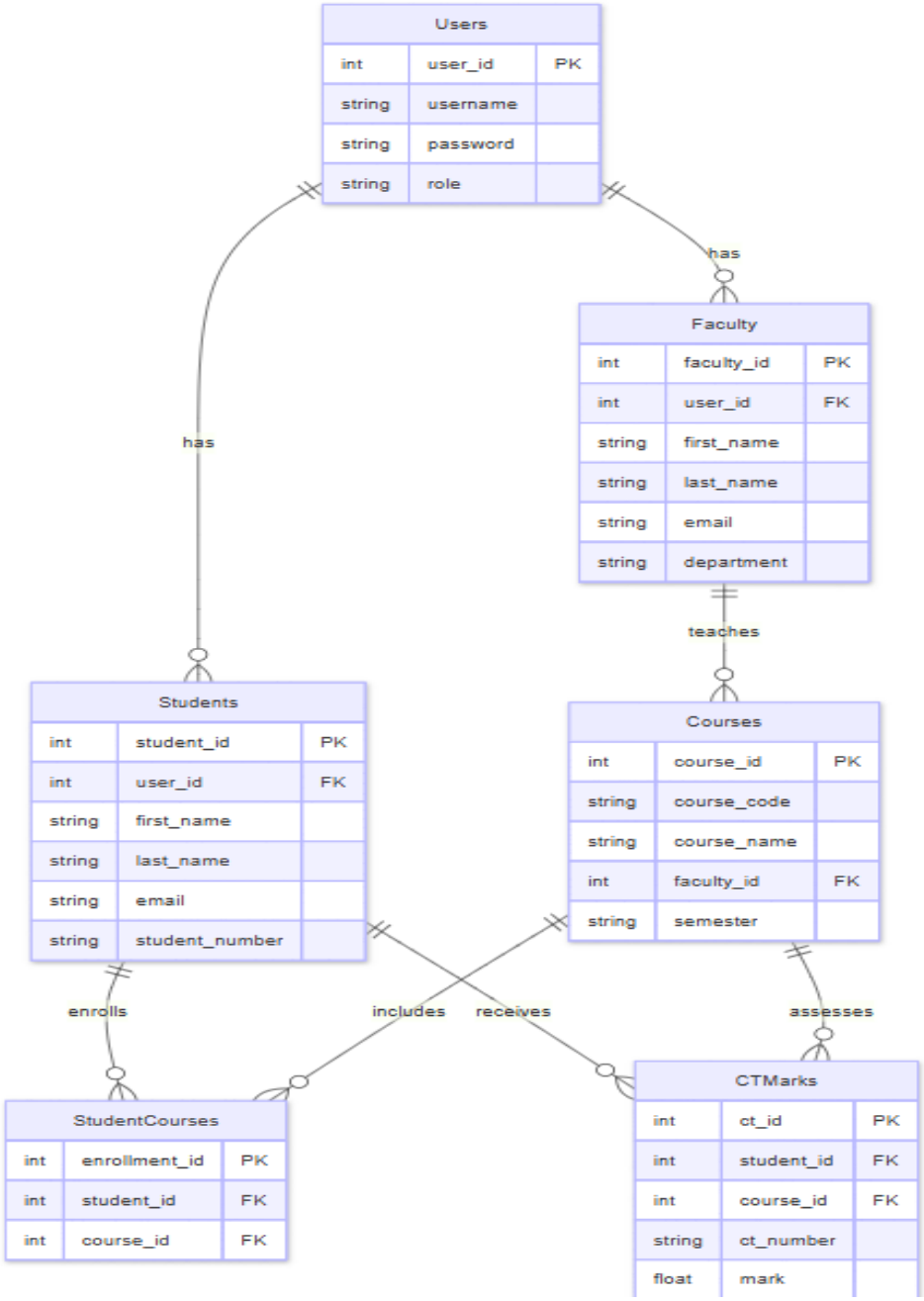

Admin
Faculty
System reports
Report request
Student marks
Student Mark Notification System
5.0\nAdmin\nReporting
1.0\nMark Entry\nProcess
Data query
Store marks
Mark\nDatabase
New marks
2.0\nEmail\nNotification\nProcess
Student contact details
Student info request
Student\nDatabase
Data query
Verification request
Verification response
Email request
Log notification
3.0\nStudent\nLogin\nProcess
Login successful
4.0\nNotification\nDisplay\nProcess
Login credentials
Email Server
Pending notifications
Check notifications
Display notification
Email sent
Notification\nDatabase
Students

iii.Level 2 DFD:



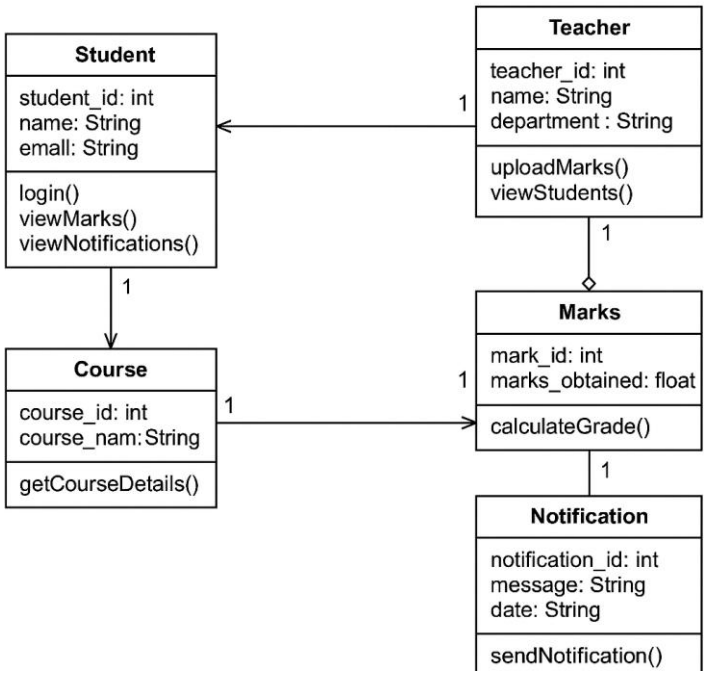
b) Class based models :

i.E-R diagram:



ii.class diagram:

- **Student**
    - Attributes: student_id, name, email, password
    - Methods: login(), viewMarks(), viewNotifications()
- **Teacher**
    - Attributes: teacher_id, name, department
    - Methods: uploadMarks(), viewStudents()
- **Course**
    - Attributes: course_id, course_name
    - Methods: getCourseDetails()
- **Marks**
    - Attributes: mark_id, marks_obtained
    - Methods: calculateGrade()
- **Notification**
    - Attributes: notification_id, message, date
    - Methods: sendNotification()



iii.CRC model:

Class-Responsibilitiey–Collaborator

Student	Responsibilities	Collaborators
• Register/Login • View marks • Receive notifications	• Upload marks • View assigned courses • Notify students	Marks Marks Notification
Course	**Marks**	**Notification**
• Store course details • Link teacher and students Teacher Student Marks	• Store and retrieve marks • Link student and course Student Course Teacher	Create and deliver messages Alert students of updates Student Teacher Admin

3.project planning:

i.WBS:

1. Planning
   - 1.1 Requirements gathering
   - 1.2 Feasibility study
   - 1.3 Technology selection

2. Design
   - 2.1 Database design (ERD)
   - 2.2 System architecture
   - 2.3 UI/UX design
   - 2.4 Class diagrams

3. Development
   - 3.1 Backend (authentication, APIs)
   - 3.2 Frontend (forms, dashboards)
   - 3.3 Notification system
   - 3.4 Integration with database

4. Testing
   - 4.1 Unit testing
   - 4.2 Integration testing
   - 4.3 User acceptance testing (UAT)

5. Deployment
   - 5.1 Hosting setup
   - 5.2 Final deployment

6. Documentation & Training

6.1 Technical documentation

6.2 User manual

6.3 Admin training

7. Maintenance

7.1 Bug fixes

7.2 Feature updates

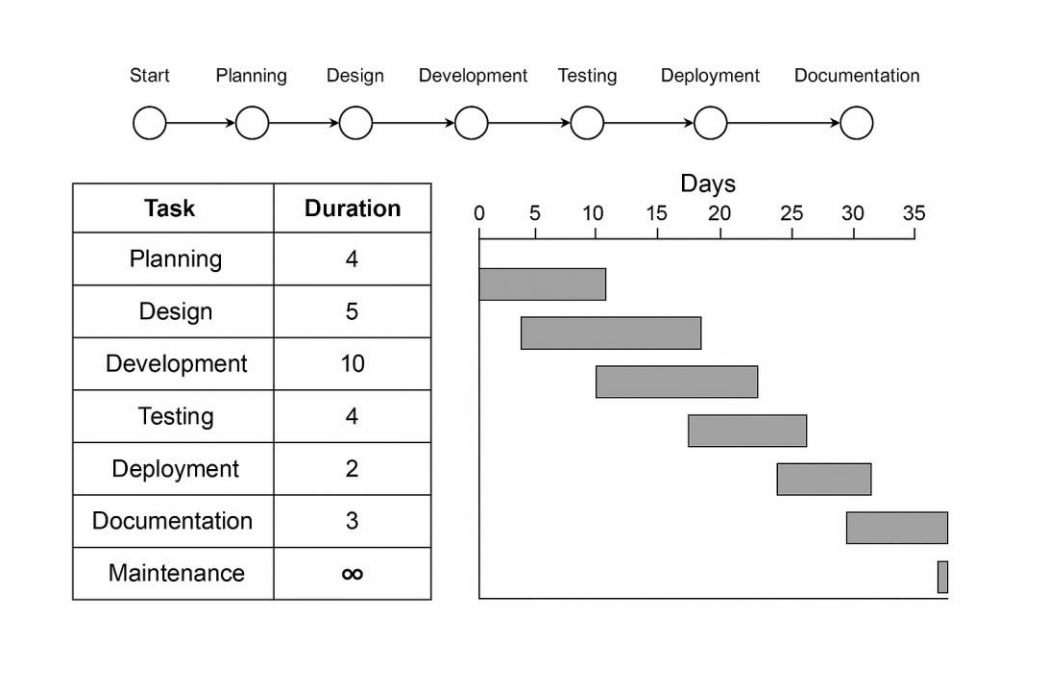
ii.project scheduling:cpm and ghantt chart

i.CPM:

Activity	Task Description	Duration (days)	Depends on
A	planning	4	-
B	design	5	A
C	development	10	B
D	testing	4	C
E	deployment	2	D
F	Documentation & Training	3	E

Ghantt chart:

Task	Duration (days)	Start	End
Planning	4	Day 1	Day 4
Design	5	Day 5	Day 9
Development	10	Day 10	Day 19
Testing	4	Day 20	Day 23
Deployment	2	Day 24	Day 25
Documentation	3	Day 26	Day 28
Maintenance	∞	Day 29	



iii.project estimation:

Module	Type	Complexity	Functionpoints
Student Registration	External Input	Medium	5
Login System	External Input	Medium	5
Marks Upload/View	External Output	High	10
Notifications	External Output	Medium	7
Course Management	Internal Logic	Medium	6
Report Generation	External Output	High	10
Admin Panel	External Interface	Medium	7

Total Function Points =50 fp

SLOC Estimation

- 1 Function Point ≈ 80 LOC (for medium-complexity web apps)
- **Estimated Code Size** = 50 FP × 80 = **4000 SLOC**

2. Effort Estimation

- **Using COCOMO Basic Model:**

For an **Organic** project type (simple, in-house, small team):

Effort (Effort (in person-months)=a×(KLOC)^b

Where:

- a = 2.4, b = 1.05
- KLOC = 4.0 (since 4000 SLOC)

Effort=2.4×(4.0)1.05≈10.2 person-months

Estimated Effort: ~10 person-month

3. Schedule Estimation

Duration (months)=2.5×(Effort)^0.38

Duration=2.5×(10.2)^0.38≈6.5 months

Estimated Schedule: **~6.5 months** (can be compressed with a larger team)

If working in a 3 member team:

Calendar Time=10.2 person-months/3 ≈3.4 months

4. Cost Estimation

Assuming:

Each team member's effort is valued at **৳15,000/month** (intern/student
Total Cost=10.2 person-months×৳15,000=৳153,000

Estimated Development Cost: **~৳153,000** (team of 3 over ~3 months)

5. Summary Table

Estimation Aspect	**Value**
Product Size	4000 SLOC (50 FP)
Effort	~10.2 person-months
Duration	~6.5 months (solo), 3.4 months (3 members)
Cost	~৳153,000

Risk Analysis:

i. Effort / Schedule / Cost Risks:

Risk	Category	Impact	Probability	Mitigation
Developer unavailable	Effort	High	Medium	Have backup developer or divide tasks smartly
Delay in integration	Schedule	High	Medium	Start integration testing early
Cost of hosting	Cost	Medium	Low	Use free/academic cloud plans

SWOT Analysis:

Strengths:

Simple, practical use-case

Clear module separation

Weaknesses:

Limited initial team

Learning curve for tech stack

Opportunities:

Can scale to other institutions

Useful for internship showcase

Threats:

Competition from similar apps

Tech bugs or data loss risks

RMMM Plan (Risk Mitigation, Monitoring, Management):

Risk	Mitigation	Monitoring	Management
Developer unavailability	Cross-training team members	Weekly standups to track presence	Reallocate tasks as needed
Schedule slippage	Buffer time in plan	Use tools like Trello, Gantt tracking	Adjust timelines early
Hosting failure	Use GitHub Pages or Render backups	Monitor site uptime	Set alerts for downtime
Data loss	Frequent backups	Log and verify backup tasks	Restore from latest backup
Tech stack complexity	Choose familiar frameworks	Review learning progress weekly	Provide extra learning resources