University of Information Technology & Sciences

Department of
Computer Science and Engineering

Project Report Submission

Course Title: Software Project Design and Development
Course Code: CSE-416

Submitted To

AL Imtiaz
Department Head of CSE Department,
UITS

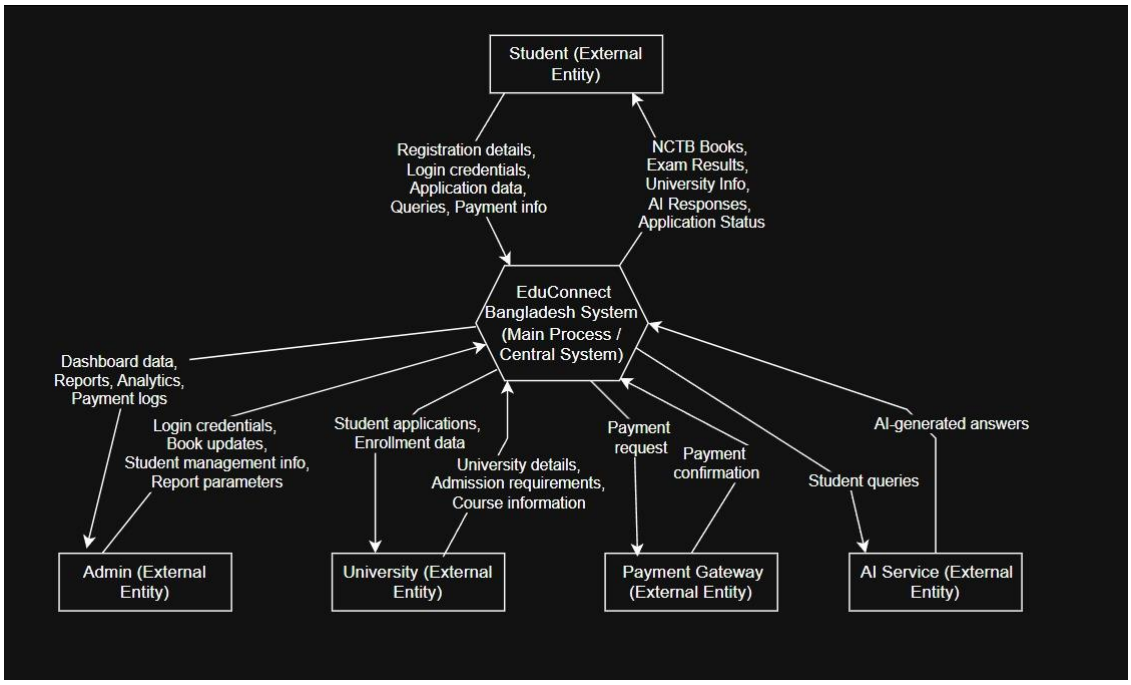
Submitted By

Name : Mobarok Hossain Zobaer
Id : 0432220005101029
Name : Sadia Akter Tuly
Id : 0432220005101030
Batch : 52
Semester: Autumn 25
Section : 7A

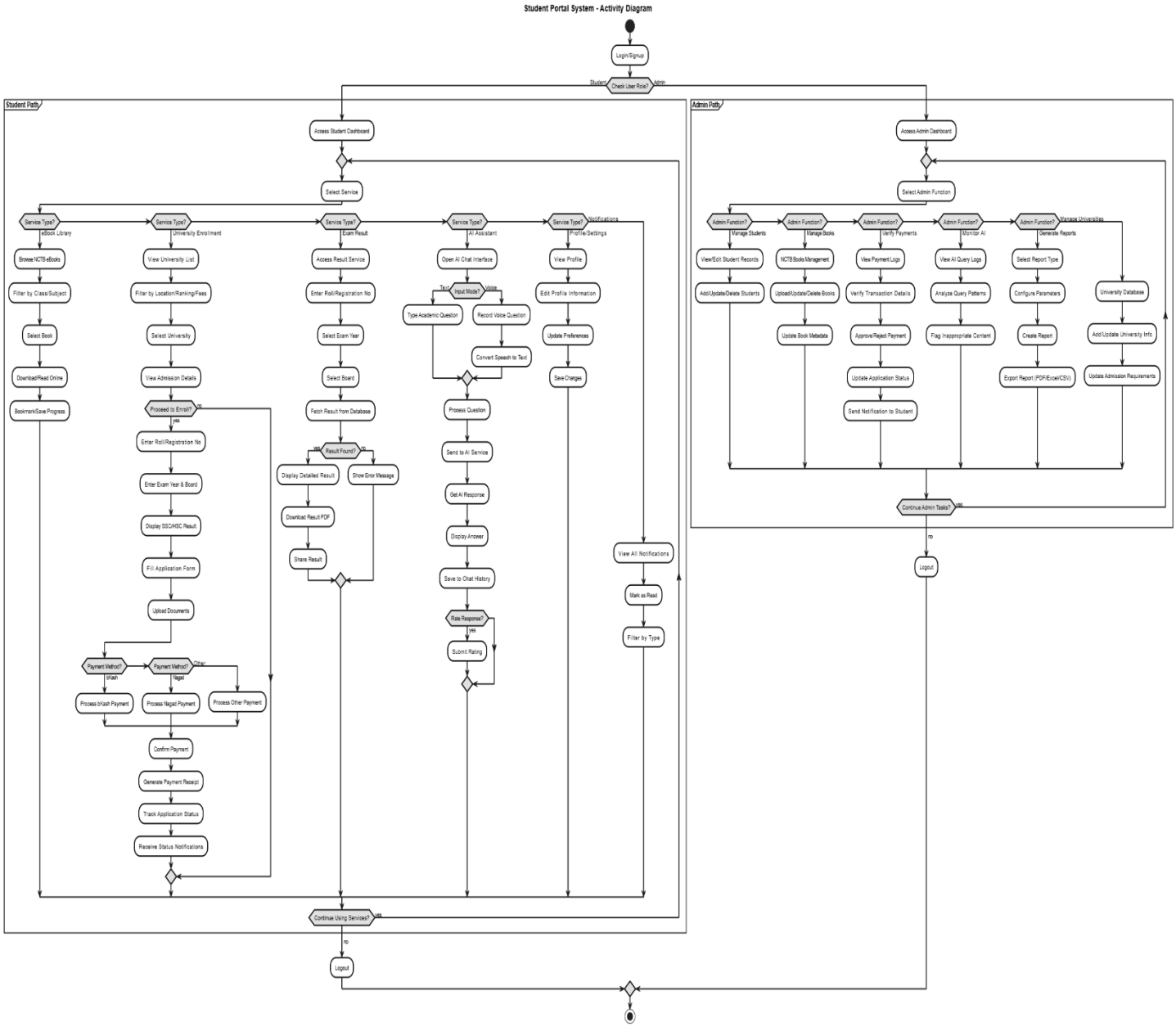
EduConnect -A Smart Education Service Platform

1. Updated DFD, Activity, ER Diagram

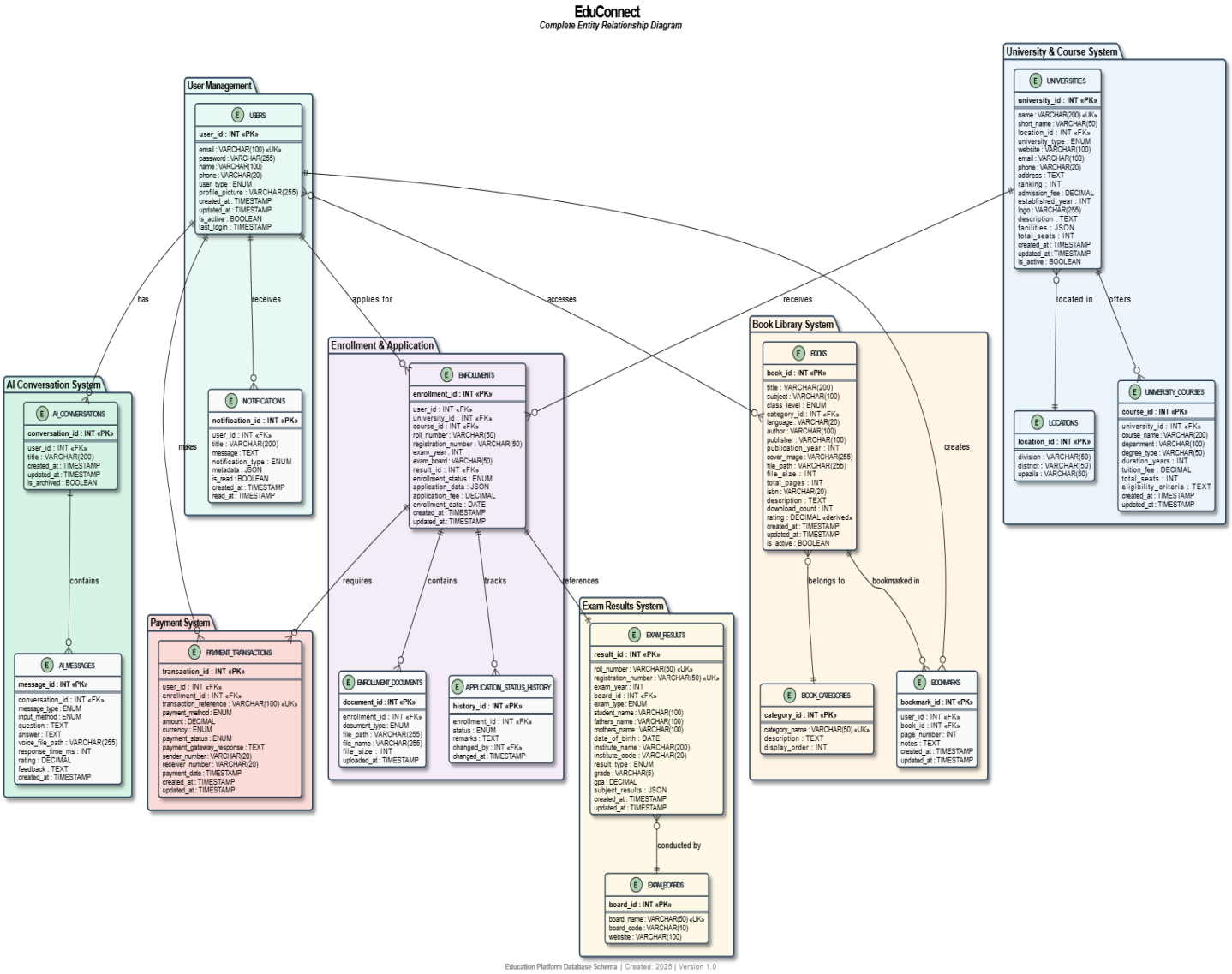
- **DFD**



- **Activity Diagram**



- **ER Diagram**


EduConnect
Complete Entity Relationship Diagram
User Management
USERS
user_id : INT «PK»
email : VARCHAR(100) «UK»
password : VARCHAR(255)
name : VARCHAR(100)
phone : VARCHAR(20)
user_type : ENUM
profile_picture : VARCHAR(255)
created_at : TIMESTAMP
updated_at : TIMESTAMP
is_active : BOOLEAN
last_login : TIMESTAMP
NOTIFICATIONS
notification_id : INT «PK»
user_id : INT «FK»
title : VARCHAR(200)
message : TEXT
notification_type : ENUM
metadata : JSON
is_read : BOOLEAN
created_at : TIMESTAMP
read_at : TIMESTAMP
has
receives
applies for
accesses
receives
makes
AI Conversation System
AI_CONVERSATIONS
conversation_id : INT «PK»
user_id : INT «FK»
title : VARCHAR(200)
created_at : TIMESTAMP
updated_at : TIMESTAMP
is_archived : BOOLEAN
AI_MESSAGES
message_id : INT «PK»
conversation_id : INT «FK»
message_type : ENUM
input_method : ENUM
question : TEXT
answer : TEXT
voice_file_path : VARCHAR(255)
response_time_ms : INT
rating : DECIMAL
feedback : TEXT
created_at : TIMESTAMP
contains
Payment System
PAYMENT_TRANSACTIONS
transaction_id : INT «PK»
user_id : INT «FK»
enrollment_id : INT «FK»
transaction_reference : VARCHAR(100) «UK»
payment_method : ENUM
amount : DECIMAL
currency : ENUM
payment_status : ENUM
payment_gateway_response : TEXT
sender_number : VARCHAR(20)
receiver_number : VARCHAR(20)
payment_date : TIMESTAMP
created_at : TIMESTAMP
updated_at : TIMESTAMP
Enrollment & Application
ENROLLMENTS
enrollment_id : INT «PK»
user_id : INT «FK»
university_id : INT «FK»
course_id : INT «FK»
roll_number : VARCHAR(50)
registration_number : VARCHAR(50)
exam_year : INT
exam_board : VARCHAR(50)
result_id : INT «FK»
enrollment_status : ENUM
application_data : JSON
application_fee : DECIMAL
enrollment_date : DATE
created_at : TIMESTAMP
updated_at : TIMESTAMP
ENROLLMENT_DOCUMENTS
document_id : INT «PK»
enrollment_id : INT «FK»
document_type : ENUM
file_path : VARCHAR(255)
file_name : VARCHAR(255)
file_size : INT
uploaded_at : TIMESTAMP
APPLICATION_STATUS_HISTORY
history_id : INT «PK»
enrollment_id : INT «FK»
status : ENUM
remarks : TEXT
changed_by : INT «FK»
changed_at : TIMESTAMP
requires
contains
tracks
references
Exam Results System
EXAM_RESULTS
result_id : INT «PK»
roll_number : VARCHAR(50) «UK»
registration_number : VARCHAR(50) «UK»
exam_year : INT
board_id : INT «FK»
exam_type : ENUM
student_name : VARCHAR(100)
fathers_name : VARCHAR(100)
mothers_name : VARCHAR(100)
date_of_birth : DATE
institute_name : VARCHAR(200)
institute_code : VARCHAR(20)
result_type : ENUM
grade : VARCHAR(5)
gpa : DECIMAL
subject_results : JSON
created_at : TIMESTAMP
updated_at : TIMESTAMP
EXAM_BOARDS
board_id : INT «PK»
board_name : VARCHAR(50) «UK»
board_code : VARCHAR(10)
website : VARCHAR(100)
conducted by
Book Library System
BOOKS
book_id : INT «PK»
title : VARCHAR(200)
subject : VARCHAR(100)
class_level : ENUM
category_id : INT «FK»
language : VARCHAR(20)
author : VARCHAR(100)
publisher : VARCHAR(100)
publication_year : INT
cover_image : VARCHAR(255)
file_path : VARCHAR(255)
file_size : INT
total_pages : INT
isbn : VARCHAR(20)
description : TEXT
download_count : INT
rating : DECIMAL «derived»
created_at : TIMESTAMP
updated_at : TIMESTAMP
is_active : BOOLEAN
BOOK_CATEGORIES
category_id : INT «PK»
category_name : VARCHAR(50) «UK»
description : TEXT
display_order : INT
BOOKMARKS
bookmark_id : INT «PK»
user_id : INT «FK»
book_id : INT «FK»
page_number : INT
notes : TEXT
created_at : TIMESTAMP
updated_at : TIMESTAMP
belongs to
bookmarked in
creates
University & Course System
UNIVERSITIES
university_id : INT «PK»
name : VARCHAR(200) «UK»
short_name : VARCHAR(50)
location_id : INT «FK»
university_type : ENUM
website : VARCHAR(100)
email : VARCHAR(100)
phone : VARCHAR(20)
address : TEXT
ranking : INT
admission_fee : DECIMAL
established_year : INT
logo : VARCHAR(255)
description : TEXT
facilities : JSON
total_seats : INT
created_at : TIMESTAMP
updated_at : TIMESTAMP
is_active : BOOLEAN
LOCATIONS
location_id : INT «PK»
division : VARCHAR(50)
district : VARCHAR(50)
upazila : VARCHAR(50)
UNIVERSITY_COURSES
course_id : INT «PK»
university_id : INT «FK»
course_name : VARCHAR(200)
department : VARCHAR(100)
degree_type : VARCHAR(50)
duration_years : INT
tuition_fee : DECIMAL
total_seats : INT
eligibility_criteria : TEXT
created_at : TIMESTAMP
updated_at : TIMESTAMP
located in
offers
Education Platform Database Schema | Created: 2025 | Version 1.0

2. Software Requirement analysis report

Project: **EduConnect**
Prepared for: EduConnect stakeholders
Prepared by: (Mobarok Hossain Zobaer and Sadia Akter Tuly)

Executive Summary
EduConnect is a web-based educational platform (hosted at https://educonnectservice.netlify.app/) that connects students, instructors, and administrators. Based on the supplied DFD (Level-0), Activity Diagram, and ER Diagram, this Software Requirements Analysis (SRA) documents the functional and non-functional requirements, data structures, process flows, actors, acceptance criteria and recommended next steps to guide design, development, testing and deployment.

Purpose and Scope
Purpose
Capture and formalize requirements for EduConnect so that developers, QA, and stakeholders have a single source of truth prior to implementation or further design.

Scope (in-scope)
- User registration and authentication (students, instructors, admins)
- Course creation, browsing, enrollment and management
- Content management (lectures, files, links)
- Messaging / notifications between users
- Basic analytics / reporting for instructors and admins
- Role-based access control and simple admin functions (user management)
- Data storage modeled per provided ER diagram (Users, Courses, Enrollments, Content, Messages, Roles)

Out of scope (for this phase)
- Payment / e-commerce (unless you specify)
- Advanced learning analytics / AI recommendations
- Mobile native apps (web responsive only)
- Third-party SSO integrations (can be future enhancement)
- Stakeholders & Actors
- Primary stakeholders
- Students: browse courses, enroll, access course content, message instructors.
- Instructors: create/manage courses, add content, interact with students, view enrollments and basic analytics.
- Administrators: manage users, moderate content, view system reports, assign roles.
- System/DevOps: maintain hosting, backups, deployments.
- QA/Testers: validate requirements and acceptance criteria.

Actors (system use-case perspective)
i. Anonymous visitor
ii. Registered Student
iii. Registered Instructor
iv. System Administrator
v. Automated system (email/notification sender)

vi. System Overview (mapped to DFD Level-0)
vii. Top-level processes (from your Level-0 DFD)
viii. P1 — User Management: Registration, login, profile management, role assignment.
ix. P2 — Course Management: Course creation, update, delete, publish/unpublish.
x. P3 — Content Delivery: Upload lecture material, attachments, streaming links, content retrieval.
xi. P4 — Enrollment & Interaction: Enroll/unenroll students, messaging, discussions, notifications.
xii. Data stores (from ER & DFD)
xiii. D1: User repository (Users, Roles, Profiles)
xiv. D2: Course repository (Courses, Categories, Metadata)
xv. D3: Content repository (Lectures, Files, Attachments)
xvi. D4: Enrollment and Activity logs
xvii. External entities
xviii. Email/Notification service (for password resets, enrollment confirmations)
xix. File storage (cloud or local for attachments)
xx. Functional Requirements
xxi. All functional requirements are written in the form: "The system shall…"

Authentication & Authorization

- FR1.0: The system allows users to register with name, email, password, role (student/instructor) and profile details.
- FR1.1: The system shall validate email uniqueness and enforce password strength rules.
- FR1.2: The system shall support secure login with session management (expiry and logout).
- FR1.3: The system shall support password reset flows via email.
- FR1.4: The system shall enforce role-based access control (students vs instructors vs admins).

User Profiles

- FR2.0: The system shall allow users to view and edit their profile information (name, bio, contact, photo).
- FR2.1: The system shall allow instructors to add a short bio and teaching areas.

Course Lifecycle

- FR3.0: The system shall allow instructors to create a course with title, description, category, thumbnail, syllabus, visibility (public/private/draft).
- FR3.1: The system shall allow instructors to update and delete courses they own.
- FR3.2: The system shall allow admins to manage (update/delete) any course.
- FR3.3: The system shall display a list of published courses to authenticated and anonymous users with search and basic filters (category, instructor).

Course Content

- FR4.0: The system shall allow instructors to add content items to a course—lectures, documents (PDF, PPT), links, videos, and quizzes (if planned).
- FR4.1: The system shall store metadata for each content item (title, type, upload date, uploader, associated course).
- FR4.2: The system shall support downloading attachments and viewing online resources via link.

Enrollment & Access

- FR5.0: The system shall allow students to enroll in published courses.
- FR5.1: The system shall record enrollments in the Enrollment data store and provide students access to course content once enrolled.
- FR5.2: The system shall allow instructors to view the list of enrolled students for their courses.

Communication & Notifications

- FR6.0: The system shall provide a messaging feature for student↔instructor interactions (simple messages, attachments optional).
- FR6.1: The system shall send email notifications for key events: registration confirmation, password reset, enrollment confirmation, important system announcements.
- FR6.2: The system shall show in-app notifications (toasts or notification center) for new messages, enrollment events, or admin announcements.

Administration

- FR7.0: The system shall provide admin UI for user management: view users, change roles, deactivate users.
- FR7.1: The system provides basic content moderation: report content, admin remove/restore content.

Reporting & Analytics (basic)

- FR8.0: The system shall provide instructors with a dashboard showing course stats: number of enrollments, recent activity (e.g., last 30 days).
- FR8.1: The system shall provide admins with system-level metrics (total users, active users, courses count).

Search & Navigation

- FR9.0: The system shall provide course search by title, instructor, and category.
- FR9.1: The system shall provide breadcrumbs and simple navigation between course pages, profile and dashboard.
- Non-Functional Requirements (NFR)
- 6.1 Performance
- NFR1: The system shall respond to 95% of user interactions (page loads, search) within 2 seconds under normal load (up to X concurrent users — define during sizing).
- NFR2: File upload should support resumable uploads for files > 10 MB (if large content is expected).

Reliability & Availability

- NFR3: The system shall be available 99.5% monthly (depending on hosting SLA).
- NFR4: Backups of database and file storage shall occur daily with retention of at least 14 days.

Security

- NFR5: Passwords must be stored hashed (bcrypt/argon2) and never in plaintext.
- NFR6: All user-facing traffic shall be served over HTTPS.
- NFR7: Implement basic protections against XSS, CSRF, SQL Injection, and enforce input validation.
- NFR8: Sensitive actions (role changes, deletions) shall be audited in an admin log.

Usability

- NFR9: The UI shall be responsive and usable on desktop and mobile viewports.
- NFR10: Onboarding flows (first login, profile setup) shall guide new users.

Maintainability & Extensibility

- NFR11: Codebase should follow modular design, documented APIs and be ready for integration of future features (payments, certificates).
- NFR12: Provide API endpoints for core functionality (user, courses, content) for future mobile apps / integrations.

Privacy and Compliance

- NFR13: System must offer a privacy policy and a way for users to request data export or account deletion (GDPR-style workflows if relevant).
- Data Model Summary (inferred from ER diagram)
- Primary entities and key attributes:
  - User (user_id PK, name, email unique, password_hash, role_id FK, profile fields, status, created_at, updated_at)
  - Role (role_id PK, name e.g., Student/Instructor/Admin)
  - Course (course_id PK, title, description, category_id FK, instructor_id FK, status (draft/published), thumbnail, created_at)
  - Content (content_id PK, course_id FK, title, type (video, pdf, link), file_path/URL, uploader_id FK, created_at)
  - Enrollment (enrollment_id PK, course_id FK, user_id FK, enrolled_on, status)
  - Message (message_id PK, sender_id FK, recipient_id FK, course_id FK optional, content, sent_at)
  - Category (category_id PK, name)
  - ActivityLog (activity_id PK, user_id FK, action, target_id, timestamp)

Relationships

- User (1) — (M) Course (as Instructor)
- Course (1) — (M) Content
- Course (1) — (M) Enrollment (by Student)
- User (1) — (M) Message (sent)
- Role (1) — (M) User

Data integrity constraints

- Enrollments must reference existing course and user.
- Deleting an instructor should reassign or warn if courses exist (business rule).
- Unique constraints on email and course slug (if using slugs).

Process & Flow Mapping (from DFD / Activity diagram)

- Key flows (high level)
- Registration flow: Anonymous → Register → Email verification → Login → Profile setup.
- Course authoring flow: Instructor logs-in → Create course metadata → Add content → Publish course → Students browse & enroll.
- Enrollment flow: Student views course → clicks enroll → system validates prerequisites and enrollment rules → creates Enrollment record → send confirmation notification.
- Messaging flow: Sender composes message → system stores Message entity → notify recipient by in-app notification & email (optional).
- Admin moderation flow: Admin receives report → views reported content → performs moderation action (remove/restore) → records audit.

Security & Privacy Considerations (detailed)

- Use HTTPS everywhere and set secure flags on cookies (HttpOnly, Secure, SameSite).
- Implement rate limiting for login and sensitive endpoints.
- Use parameterized queries / ORM to prevent SQL injection.
- Sanitize user-provided HTML (if allowing rich text) and apply Content Security Policy (CSP).
- Audit trail for role changes and deletions.
- Encryption at rest for backups or sensitive information if required.
- Data retention policy for messages and logs—specify retention period.

Acceptance Criteria (sample)

- AC1: A registered student can log in and enroll in a published course; enrollment appears in their dashboard and in the instructor's enrolled students list.
- AC2: An instructor can create a course, add content items, and publish it; the course is discoverable via search.
- AC3: The system sends a password reset email that allows the user to create a new password (token expires within configurable TTL).
- AC4: Admin can deactivate a user account and deactivated user cannot log in.
- AC5: All core flows (registration, create course, add content, enroll, message) function end-to-end with appropriate role permissions enforced.

Constraints & Assumptions

<u>Assumptions</u>

- Hosting on Netlify (frontend) with a separate backend/API and database (e.g., Node/Express + PostgreSQL or Firebase) — the site on Netlify is the frontend only.
- File storage will be provided by a cloud object store (S3, DigitalOcean Spaces, Firebase Storage) or a server endpoint.
- Email service (SMTP, SendGrid or similar) will be available for notifications.

<u>Constraints</u>

- Netlify static hosting limits server-side functionality; dynamic features require a separate API / serverless functions (Netlify Functions, Vercel, AWS Lambda) or backend hosting.
- If real-time messaging is required, additional services (WebSocket server or pub/sub) will be needed.

Traceability Matrix (short)

- Map a few source artifacts to requirements:
- DFD P1 (User Management) → FR1.0–1.4, FR2.0
- DFD P2 (Course Management) → FR3.0–3.3
- DFD P3 (Content Delivery) → FR4.0–4.2
- Activity Diagram (Enrollment flow) → FR5.0–5.2
- ER Diagram Entities → Data Model (Section 7)

Recommended Next Steps

- Finalize scope: confirm payments/certificates/quizzes as in-scope or postponed.
- Choose tech stack for backend & storage (recommend Node.js + PostgreSQL or Firebase for faster prototyping).

- Create API specification (OpenAPI) for the backend that maps to FRs.
- Create wireframes for all major pages (landing, course page, instructor dashboard, admin panel).
- Prepare security & deployment checklist (HTTPS, CSP, backups, secrets management).
- Create a prioritized backlog / sprint plan: MVP features — auth, course CRUD, content upload, enrollment, messaging.
- Prepare test plans and acceptance test cases mapped to ACs above.

Appendix

- Sample User Stories (for Agile backlog)
- As a Student, I want to register with my email so I can enroll in courses.
- As an Instructor, I want to create a course and add lecture files so my students can study.
- As an Admin, I want to deactivate spam accounts so the platform remains safe.

Sample API Endpoints (suggested)

- POST /api/auth/register
- POST /api/auth/login
- POST /api/auth/forgot-password
- GET /api/courses
- POST /api/courses
- GET /api/courses/
- POST /api/courses//content
- POST /api/courses//enroll
- GET /api/users//enrollments
- POST /api/messages
-

Example Data Retention & Backup Policy

- Daily DB snapshot + 14 days retention.
- File backups weekly, retention 30 days (depending on storage costs).

3. Software Development Plan(Time + resource+Man power, GANTT Chart)

A. High-level timeline (phases & weeks)

Phase	Description	Duration	Updated Dates
Phase 0	Kickoff & Planning	Week 1	**Aug 1 – Aug 7, 2025**
Phase 1	Requirements & SRS Finalization	Week 2	**Aug 8 – Aug 14, 2025**
Phase 2	UX/UI Design	Weeks 3–4	**Aug 15 – Aug 28, 2025**
Phase 3	Frontend & Backend Core Development (parallel)	Weeks 4–7	**Aug 22 – Sep 18, 2025**
Phase 4	Content Integration & Messaging	Weeks 6–8	**Sep 5 – Sep 25, 2025**
Phase 5	Integration & E2E Testing	Weeks 8–10	**Sep 19 – Oct 9, 2025**
Phase 6	Performance/Security Tuning + CI/CD Deployment	Weeks 10–11	**Oct 3 – Oct 16, 2025**
Phase 7	UAT & Fixes	Weeks 11–13	**Oct 10 – Oct 30, 2025**
Phase 8	Go-live & Hypercare	Weeks 13–15	**Oct 31 – Nov 20, 2025**

B. Roles & Suggested Manpower (summary)

- Project Manager (PM): 1 (0.5 FTE during dev; 0.25–0.5 during planning & UAT)
- Business Analyst (BA): 1 (1.0 FTE during requirements; 0.5 during UAT)
- UI/UX Designer: 1 (0.8 FTE for design weeks + handoff)
- Frontend Developers: 2 (1.0 FTE each through implementation)
- Backend Developers: 2 (1.0 FTE each through implementation)
- QA Engineers: 1 (0.8 FTE for testing and regression)
- DevOps / Security: 1 (0.5 FTE for CI/CD, infra & tuning)
- Content Specialist: 1 (0.5 FTE to prepare/upload course content)

C. Task-level step-by-step (what to do during each task)

Project Kickoff & Planning (Week 1)

- Kickoff meeting with stakeholders, validate scope, define MVP.
- Agree on tech stack (recommended: React + Node/Express + PostgreSQL or Firebase).
- Set up repo, branching strategy, basic CI pipeline.
- Deliverables: Project charter, initial backlog, sprint plan.

Requirements Refinement & SRS (Week 2)

- Expand the SRS from your earlier SRA: detailed FRs, NFRs, API list.
- Create acceptance criteria for each user story.
- Deliverables: Final SRS doc, prioritized backlog.

UX/UI Design (Weeks 3–4)

- Create low-fi wireframes for main pages (landing, auth, course, dashboard, admin).
- Produce hi-fi mockups and interactive prototype.
- Deliverables: Design system (colors, typography), component list, assets.

Frontend Setup & Core Pages (Weeks 4–6)

- Scaffold project, implement routing, auth flows, landing & course listing pages.
- Integrate with mock API endpoints (or real backend if ready).
- Deliverables: Frontend skeleton, components library, responsive layouts.

Backend API & DB (Weeks 3–6)

- Design DB schema (use the ER diagram), implement core APIs: auth, courses, enrollments, content metadata.
- Implement secure auth (bcrypt/argon2), token/session management.
- Deliverables: API endpoints, database migrations, Swagger/OpenAPI draft.

Content Upload & Storage Integration (Weeks 6–8)

- Integrate file storage (S3 / Firebase storage / server filesystem), implement resumable uploads if needed.
- Deliverables: Upload API, storage lifecycle rules.

Messaging & Notifications (Weeks 6–8)

- Implement simple messaging DB model and API.
- Integrate email service (SMTP / SendGrid) for transactional emails.
- Deliverables: Messaging endpoints, email templates.

Integration & End-to-End Testing (Weeks 8–10)

- Integrate frontend and backend, run E2E test suites (Cypress / Playwright).
- Fix integration bugs and edge cases.
- Deliverables: E2E test reports, regression test plan.

Performance Tuning & Security Hardening (Week 10)

- Add rate limiting, input validation, CSP, HTTPS enforcement, backups.
- Run penetration checks, load tests (locust/jmeter small runs).
- Deliverables: Security checklist, tuned DB indices.

Deployment, CI/CD & Staging (Week 11)

- Configure CI (build, test) and CD to staging (Netlify/Vercel for frontend; serverless or container infra for backend).
- Deliverables: CI/CD pipelines, staging environment.

UAT & Fixes (Weeks 11–13)

- Stakeholders test flows, capture bugs, prioritize fixes.
- Deliverables: UAT signoff, resolved critical/major bugs.

Go-Live & Post-Launch Support (Weeks 13–15)

- Deploy to production, monitor logs, error tracking, hotfix if needed.
- Deliverables: Production deployment, monitoring dashboards, handover docs.

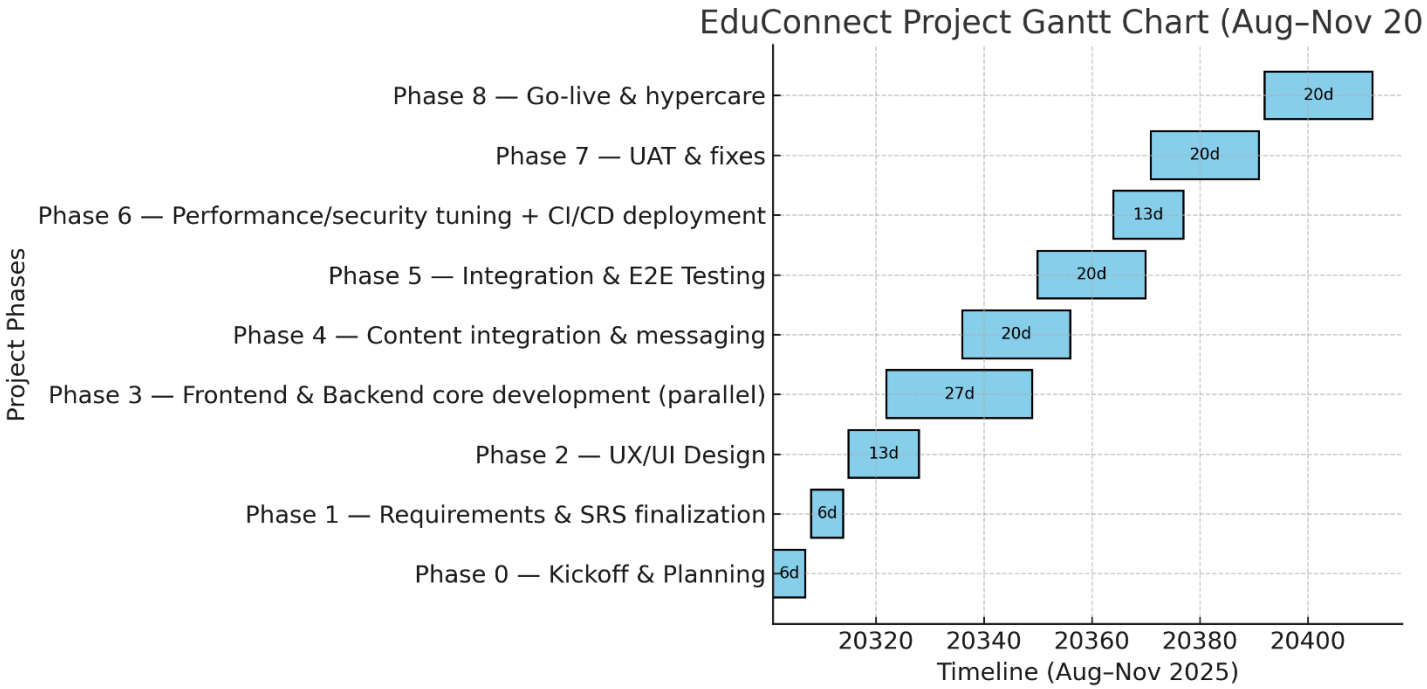
D. Resource allocation by week (how many people on each sprint)

- The CSV/XLSX includes Estimated_FTE per task. Quick weekly snapshot:
- Weeks 1–2: PM, BA, light DevOps (3 people active ~2.0 FTE total)
- Weeks 3–6: Peak dev phase — Frontend (2), Backend (2), UI/UX (1), PM (1), BA (1) => ~7 people active (approx 6–7 FTE)
- Weeks 7–10: Integration & testing — QA joins (1), Devs continue, DevOps increases for infra (~6 people)
- Weeks 11–13: UAT + fixes — PM, BA, Devs, QA (~5–6 people)
- Weeks 13–15: Hypercare — smaller crew: DevOps, 1 dev, PM, QA (~3 people)

E. Milestones (suggested)

- M1: SRS sign-off (end of Week 2)
- M2: Design approval (end of Week 4)
- M3: Core API & frontend skeleton complete (end of Week 7)
- M4: Integration & E2E pass (end of Week 10)
- M5: UAT signoff (end of Week 13)
- M6: Production Go-Live (end of Week 15)

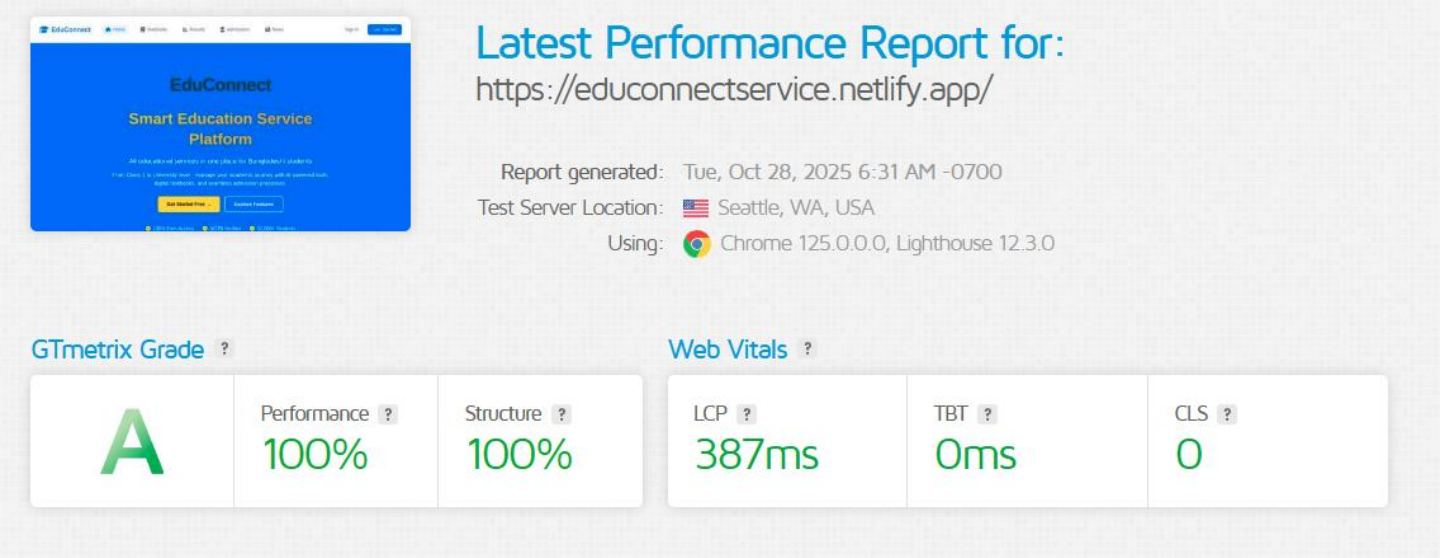
F. How to use the Gantt chart & files I generated



Link 01(xlsx):
https://github.com/mhzobaer26/CSE-Lab-Courses/blob/main/7th%20Semester/CSE-416%20%5BSoftware%20Project%20Design%20and%20Development%5D/Project/Assignment/Gant%20chart%2C%20CSV%2C%20XLSX/educonnect_schedule_updated.xlsx

Link 02(CSV):
https://github.com/mhzobaer26/CSE-Lab-Courses/blob/main/7th%20Semester/CSE-416%20%5BSoftware%20Project%20Design%20and%20Development%5D/Project/Assignment/Gant%20chart%2C%20CSV%2C%20XLSX/educonnect_schedule_updated.csv

4. Software Testing report



5. Business Model (with ROI calculation)

a) Business Overview

EduConnect is a digital education platform designed to connect students with qualified tutors, institutions, and learning resources. It bridges the gap between education seekers and providers using a user-friendly online interface.

b) Value Proposition

- **For Students:** Easy access to verified tutors, courses, and study materials.
- **For Tutors:** Opportunity to earn income and grow reputation.
- **For Institutions:** Increased visibility and student enrollment through digital presence.

c) Target Market

- **Primary:** High school and university students (ages 16–25) seeking tutoring or online courses.
- **Secondary:** Tutors, educational institutions, and parents of students.

Market Size:

- Estimated 25 million students enrolled in secondary to tertiary education.
- At least 30% use online learning platforms or private tutors.
  So, potential online user base ≈ **7.5 million**.

d) Revenue Streams

Source	Description	Expected Share
Commission on Tutor Fees	EduConnect takes 10–15% per completed tutoring session	45%
Subscription Plans	Monthly/annual premium plans for tutors/institutions to get higher visibility	30%
Advertisements	Ads from educational brands, tools, or books	15%
Affiliate Marketing	Referral commission from course platforms (Coursera, Udemy, etc.)	10%

e) Cost Structure

Category	Description	Estimated Cost (BDT)	Duration
Web Development & Hosting	Domain, hosting, backend/frontend setup	80,000	1-time
Design & Branding	Logo, UI/UX, promotional materials	20,000	1-time
Marketing & Advertising	Social media, campaigns, influencer ads	30,000/month	6 months
Team Salary	Developer, designer, marketing lead	1,20,000/month	6 months
Maintenance & Support	Bug fixing, updates, support staff	15,000/month	6 months

Total Initial 6-Month Cost: ≈ BDT 5,30,000

f) Resource & Manpower Allocation

Role	Responsibility	Count	Monthly Cost (BDT)
Project Manager	Supervise development and marketing	1	30,000
Web Developer	Build and maintain website	2	40,000
UI/UX Designer	User interface and experience design	1	20,000
Role	Responsibility	Count	Monthly Cost (BDT)
Project Manager	Supervise development and marketing	1	30,000

g) Revenue Forecast (First Year)

Month	Students	Tutors	Expected Monthly Revenue (BDT)
1–2	100 users	20 tutors	10,000
3–4	400 users	60 tutors	30,000
5–6	1,000 users	100 tutors	80,000
7–8	2,500 users	200 tutors	1,50,000
9–10	4,000 users	300 tutors	2,50,000
11–12	6,000 users	400 tutors	3,50,000

Total First-Year Revenue: ≈ BDT 13,20,000

h) ROI (Return on Investment) Calculation

$$ROI = \frac{(Total\ Revenue - Total\ Cost)}{Total\ Cost} \times 100$$

- **Total Revenue:** BDT 13,20,000
- **Total Cost:** BDT 5,30,000

$$ROI = \frac{(13{,}20{,}000 - 5{,}30{,}000)}{5{,}30{,}000} \times 100 = 149\%$$

ROI = 149% in the first year

That means for every 1 taka invested, EduConnect earns 1.49 taka profit within the first year.

i) Break-Even Point

$$Break - even = \frac{Fixed\ Costs}{(Revenue\ per\ User - Variable\ Cost\ per\ User)}$$

Assuming:

- Average revenue per user = 80 BDT
- Variable cost per user = 20 BDT
- Fixed cost = 5,30,000

$$Break - even = \frac{5{,}30{,}000}{(80 - 20)} = 8834\ users$$

EduConnect will break even after approximately **9,000 total users**.

j) Future Growth Plan

1. **Mobile App Launch** for Android/iOS.
2. **AI-based Tutor Recommendation System.**
3. **Institution Dashboard** for analytics and enrollment management.
4. **Partnerships** with universities and e-learning companies.
5. **Expansion** to other South Asian education markets.

6. User Manual / Documentation

I. Introduction

EduConnect is a web-based education service platform that connects students, tutors, and institutions. It allows users to search for courses, communicate with educators, and manage tutoring sessions efficiently.

- **Website:** https://educonnectservice.netlify.app
- **Platform Type:** Web Application
- **Developed Using:** HTML, CSS, JavaScript, React.js (Frontend) | Node.js (Backend) | SQL Database (Storage)
- **Target Users:** Students, Tutors, Institutions, and Admins

II. System Overview

EduConnect provides an online environment to simplify education networking and course management.

Key Features:

- Student and Tutor Registration/Login
- Profile and Course Management
- Tutor Search and Booking
- Institution Directory
- Admin Dashboard
- Feedback and Rating System

III. System Requirements

3.1 Hardware

Component	Minimum Requirement
Processor	Dual-core CPU
RAM	4 GB
Storage	500 MB free
Internet	Stable connection (1 Mbps+)

3.2 Software

Software	Version
OS	Windows / macOS / Linux
Browser	Chrome, Edge, or Firefox (latest)
Backend	Node.js v18+
Database	MySQL / PostgreSQL
Hosting	Netlify (Frontend), Render/Heroku (Backend)

IV. Installation & Access

For Users:

1. Open your browser and visit: **https://educonnectservice.netlify.app**
2. Create an account (Student/Tutor/Institution).
3. Verify your email and log in.
4. Explore available tutors and services.

For Admins:

1. Access admin dashboard with credentials.
2. Approve tutor profiles and manage listings.
3. Monitor usage statistics and system logs.

V. Functional Modules

Module	Description	User Type
Authentication Module	Handles registration, login, and password reset.	All
Profile Module	Manage personal info, experience, and courses.	Tutor, Student
Search & Booking	Allows students to find and book tutors.	Student
Institution Management	Add, view, and rate institutions.	All
Payment & Subscription	Secure payment for sessions and premium access.	Tutor, Student
Feedback System	Collect and display user ratings/reviews.	All
Admin Control Panel	Manage users, content, and reports.	Admin

VI. Step-by-Step User Guide

6.1 For Students

1. **Sign Up** → Choose "Student".
2. **Login** → Access personalized dashboard.
3. **Search Tutor** → Filter by subject, experience, or rating.
4. **Book Session** → Confirm time and payment method.
5. **Give Feedback** → Rate tutor after class.

6.2 For Tutors

1. **Sign Up** → Choose "Tutor" and add credentials.
2. **Add Courses** → Define subjects, price, and schedule.
3. **Accept Bookings** → Receive notifications from students.
4. **Withdraw Earnings** → Via integrated payment system.

6.3 For Admins

1. **Login** using admin credentials.
2. **Monitor Reports** (user count, revenue).
3. **Approve/Reject** new tutors or institutions.
4. **Manage Content** and handle complaints.

VII. Data Flow Diagram (DFD)

(Level 0 Overview)
[User] → (Login/Sign Up) → [EduConnect System] → (Access Dashboard) → [Database]

VIII. Entity–Relationship (ER) Overview

Entities:

- User (User_ID, Name, Role, Email, Password)
- Tutor (Tutor_ID, Subject, Rating)
- Student (Student_ID, Course, Feedback)
- Booking (Booking_ID, Student_ID, Tutor_ID, Date)
- Institution (Institution_ID, Name, Type, Rating)

IX. Activity Diagram Summary

- **Login Activity:** User enters credentials → System verifies → Redirects to dashboard.
- **Booking Activity:** Student selects tutor → Confirms schedule → Updates database → Notification sent.
- **Feedback Activity:** Session complete → Student submits review → Average rating recalculated.

X. Troubleshooting

Issue	Cause	Solution
Can’t log in	Wrong password	Use “Forgot Password” option
Slow load	Weak connection	Check network or try again later
Missing tutor	Not verified yet	Wait for admin approval

XI. Future Enhancements

- AI Tutor Matching System
- Mobile App (Android & iOS)
- Live Video Integration
- Smart Learning Analytics

XII. Developer Information

Role	Name	Responsibility
Project Lead	Mobarok Hossain Zobaer	Planning & Management
Web Developer	[Anyone]	Frontend + Backend
Designer	[Anyone]	UI/UX
Database Engineer	[Anyone]	Data Management
QA & Deployment	[Anyone]	Testing + Hosting

7. PO attainment report

https://github.com/mhzobaer26/CSE-Lab-Courses/blob/main/7th%20Semester/CSE-416%20%5BSoftware%20Project%20Design%20and%20Development%5D/Project/Assignment/Assignment%2003/Sample%20PO%20Justification%20report%20%20(0432220005101029).pdf