1.1 Background of the Study

In the modern education system, data plays a vital role in enhancing learning outcomes and institutional effectiveness. Every semester, institutions generate vast amounts of information — including grades, attendance, assessments, and achievements — yet much of this data remains unused or stored in disconnected systems. This lack of integration limits the ability of educators and administrators to gain a complete understanding of student progress, trends, and performance gaps.

To overcome this challenge, the EduIQ – Educational Intelligence System was developed as an intelligent analytics platform that combines structured databases, semantic understanding, and AI-driven reasoning. EduIQ integrates PostgreSQL for secure data storage, Knowledge Graphs (KG) for mapping relationships among students, subjects, and semesters, and a lightweight Retrieval-Augmented Generation (RAG) engine for natural-language querying. Together, these components transform raw academic data into actionable insights.

Through an interactive Streamlit dashboard, EduIQ allows faculty and students to visualize academic performance, track progress, and receive AI-generated summaries for better decision-making. The system represents a significant step toward intelligent educational analytics — merging data engineering and AI for personalized, data-driven learning. It also aligns with the UN Sustainable Development Goals, especially SDG 4 (Quality Education) and SDG 9 (Innovation and Infrastructure), promoting sustainable, evidence-based education.

1.2 Problem Statement

In today's data-rich academic environment, educational institutions continuously collect vast amounts of information through learning management systems, examination portals, and attendance records. However, despite this abundance, most institutions struggle to convert raw data into meaningful intelligence. Information often exists in disconnected databases, resulting in redundancy, inaccessibility, and limited analytical value. This fragmentation makes it challenging for educators to identify underperforming students, track progress, or make timely, data-driven decisions.

Traditional systems depend on static reports and manual analysis, offering only basic numerical summaries without context or interpretation. They fail to establish meaningful relationships between data points, such as how attendance impacts academic performance or which subjects consistently affect overall CGPA. Moreover, these systems lack semantic understanding — treating data as isolated entries instead of interconnected academic entities.

Another major limitation is the absence of natural-language interaction. Users often rely on complex filters or SQL queries to access specific insights, making the process time-consuming and technical. Current dashboards visualize data but cannot explain underlying trends or suggest actionable improvements.

1.3 Objectives

To overcome these issues, there is a pressing need for an intelligent, unified analytics system that can integrate academic data, understand relationships, and generate context-aware insights. EduIQ bridges this gap through Knowledge Graphs and a lightweight RAG engine, enabling deeper academic intelligence and decision support.

The primary objective of EduIQ – Educational Intelligence System is to transform how educational institutions manage, analyze, and interpret academic data. The system aims to convert raw, scattered information into actionable intelligence that enhances teaching effectiveness, student learning, and institutional decision-making.

EduIQ's design is centered around three key areas:

- Functional efficiency – centralized academic database with structured tables
- AI-driven intelligence – semantic relationships + natural language analytics
- Sustainability – paperless, data-driven, SDG-aligned educational workflows

1.4 Scope of the Project

EduIQ covers a full pipeline of academic analytics, including:

- Student academic data storage

- Attendance and marks management
- Knowledge Graph–based semantic modeling
- RAG-powered natural language querying
- Ranking of students using FCI (Final Composite Index)
- Batch-level and student-level dashboards
- Real-time updates via Streamlit

1.5 Importance of the EduIQ System

EduIQ contributes significantly to the modernization of academic analytics:

- Enhances academic transparency and decision-making
- Identifies learning gaps and improvement areas
- Supports faculty through automated, real-time insights
- Enables students to self-monitor progress
- Aligns with SDG 4 (Quality Education) and SDG 9 (Innovation & Infrastructure)
- Creates a scalable, future-ready analytical system for institutions

This section synthesizes prior work across Learning Analytics (LA), Learning Analytics Dashboards (LADs), Knowledge Graphs (KGs) for education, and Retrieval-Augmented Generation (RAG) for educational question answering. It concludes with a gap analysis that motivates the EduIQ design.

2.1 Learning Analytics

Learning Analytics focuses on turning educational data into insights that improve student learning and teaching outcomes. Key contributions from existing research: Helps institutions identify at-risk students and improve retention. Enables performance tracking using grades, attendance, and behavioral data. Supports educators in making informed, data-based decisions. Offers dashboards and metrics for student progress monitoring.

Limitations of existing works:

Most existing tools lack deeper semantic understanding. They do not connect relationships (student–subject–semester). No capability to explain *why* a student is performing a certain way. Limited adaptability and little real-time reasoning. This creates an opportunity for systems like EduIQ, which can incorporate contextual and semantic intelligence.

2.2 Learning Analytics Dashboards (LADs)

Dashboards are widely used to visualize academic performance, yet most are restricted to surface-level insights. Capabilities of existing dashboards: Provide quick summaries of metrics like grades, attendance, and performance. Enable comparison across batches, semesters, and subjects. Increase engagement by helping students monitor their progress.

However, common limitations include: Dashboards cannot interpret underlying trends. They cannot answer natural-language questions. They lack predictive and explanatory capabilities. Mostly static, offering limited interactivity.

EduIQ improves upon this by integrating dashboards with AI reasoning and semantic modeling.

2.3 Knowledge Graphs in Education

Knowledge Graphs (KGs) introduce structure and context to academic data. Existing works show that KGs: Represent semantic relationships between academic entities (students, subjects, semesters, etc.). Support advanced queries using graph traversal. Reveal hidden patterns and relationships not visible in normal databases.Enhance explainability and contextual reasoning

Limitations of current KG use in education:

Adoption is limited; implementation is complex. Few academic systems integrate KGs with analytics dashboards. Rarely combined with natural-language reasoning. EduIQ fills this gap by combining KGs with AI-generated academic insights.

2.4 Retrieval-Augmented Generation (RAG)

RAG combines information retrieval with language generation to produce accurate and context-aware answers. In existing research, RAG systems: Retrieve relevant information before generating responses. Support natural language interaction with databases. Provide precise, real-time answers based on stored facts. Improve accessibility by allowing non-technical queries

EduIQ implements a lightweight local RAG engine, ensuring: Faster responses, Data privacy, Integration with the academic Knowledge Graph

2.5 Gap Analysis

Based on the literature, the major gaps in current educational analytics systems include: Lack of semantic reasoning, Existing systems treat data as isolated entries, not connected entities. No natural-language query support, Users must rely on filters or SQL queries. Disconnected data storage, Scattered databases lead to incomplete insights. No unified system combining KG + RAG + dashboards, Existing solutions offer one or two features, not all. Limited explainability, Dashboards show values but cannot explain reasons or trends.

EduIQ directly addresses all these gaps by integrating structured databases, semantic graphs, and AI reasoning into one unified platform.

The methodology of EduIQ – Educational Intelligence System focuses on integrating structured database management, semantic reasoning, and lightweight AI-based querying into one unified educational analytics framework. The design follows a modular architecture that ensures data accuracy, contextual understanding, and scalability.

3.1 System Overview

EduIQ consists of five core components:

1. Data Ingestion and Validation – Collects and cleans academic data before storage.
2. Knowledge Graph (KG) Builder – Builds semantic relationships between academic entities.
3. Retrieval-Augmented Genration (RAG) Engine – Processes natural-language queries for contextual insights.
4. Prioritization and Ranking Engine – Calculates composite performance scores.
5. Visualization and Dashboard Layer – Displays analytics interactively through Streamlit.

Input: Raw student data (marks, attendance, achievements), Output: Interactive dashboards and intelligent summaries

3.2 Data Ingestion and Validation

Definition:
The process of collecting and verifying academic data before inserting it into the system. The Data Ingestion Module allows data input through CSV or Excel files, which are automatically validated to detect: Missing values, Inconsistencies, Formatting error, Invalid entries

Once cleaned, the data is stored in a PostgreSQL database that maintains: Referential integrity, ACID compatibility, Optimized performance for queries

Input: Raw academic data files, Output: Normalized, validated database tables

3.3 Knowledge Graph Construction

Definition:
A Knowledge Graph (KG) represents relationships between academic entities like students, subjects, achievements, and semesters. EduIQ uses Python's NetworkX library to model nodes and edges such as: enrolled_in (student–subject), belongs_to (student–batch), has_achievement (student–achievement), registered_for (student–semester)

This enables the system to answer semantic questions like:

- "Which subjects contributed most to a student's performance improvement?"
- "How does attendance relate to academic outcomes?"

Input: Structured data from PostgreSQL, Output: Semantic Knowledge Graph enabling contextual analytics

3.4 Retrieval-Augmented Generation (RAG) Engine

Definition:
RAG enhances querying by combining retrieval of factual data with AI-driven natural-language generation. EduIQ's lightweight RAG engine includes: Database retriever, KG context extractor, Local language generator. This ensures precise, real-time responses while maintaining data privacy.

Example Queries:

- "Show top performers in Semester 6."
- "Which subjects had the lowest average scores?"
- "What improvements does Roll No. 15 need?"

Input: Natural-language query, Output: Contextual, AI-generated analytical response

3.5 Prioritization and Ranking Engine

Definition:
A module that assigns ranking and priority scores to students using multiple parameters. EduIQ calculates a Final Composite Index (FCI) using weighted contributions from:

- CGPA
- Attendance
- Achievements
- Discipline score (if applicable)

This ranking system allows faculty to identify:

- Top-performing students
- Students needing academic intervention
- Attendance-related performance issues

Input: Processed academic records, Output: Ranked student list with performance categories

3.6 Visualization and Dashboard Layer

Definition:
A graphical layer built using Streamlit that visualizes academic insights. The dashboard contains:

1. Student Report

- CGPA, SGPA
- Attendance
- Subject-wise marks table
- Radar and bar-chart performance analytics
- AI-generated insights (RAG)

2. Leaderboard

- FCI-based ranking
- Batch-level comparison
- CGPA–Attendance–Achievements correlation

3. Batch Management

- Subject-wise average marks
- Semester-wise performance trends
- Attendance vs. performance graphs

4. Data Entry & Update

- Manual record updates
- Bulk file uploads
- Real-time validation

Input: Aggregated analytics, Output: Dynamic and interactive dashboards

3.7 Tools and Technologies Used

Software:

- Python – Core programming
- PostgreSQL – Relational database
- Pandas, NumPy – Data manipulation
- Matplotlib – Chart generation
- Streamlit – Frontend & visualization

Hardware:

- 8GB RAM
- Multicore processor
- Storage supporting PostgreSQL operations

4.1 Overview

EduIQ is built on five interconnected layers:

- **Data Layer (PostgreSQL):** Stores academic records such as marks, attendance, and achievements.
- **Semantic Layer (Knowledge Graph):** Builds relationships between students, subjects, and semesters to derive contextual understanding.
- **AI Layer (RAG Engine):** Enables natural language querying and intelligent data retrieval.
- **Application Layer (Python Modules):** Handles logic, ranking, and data processing.
- **Visualization Layer (Streamlit Dashboard):** Displays analytical results, insights, and visualizations interactively.

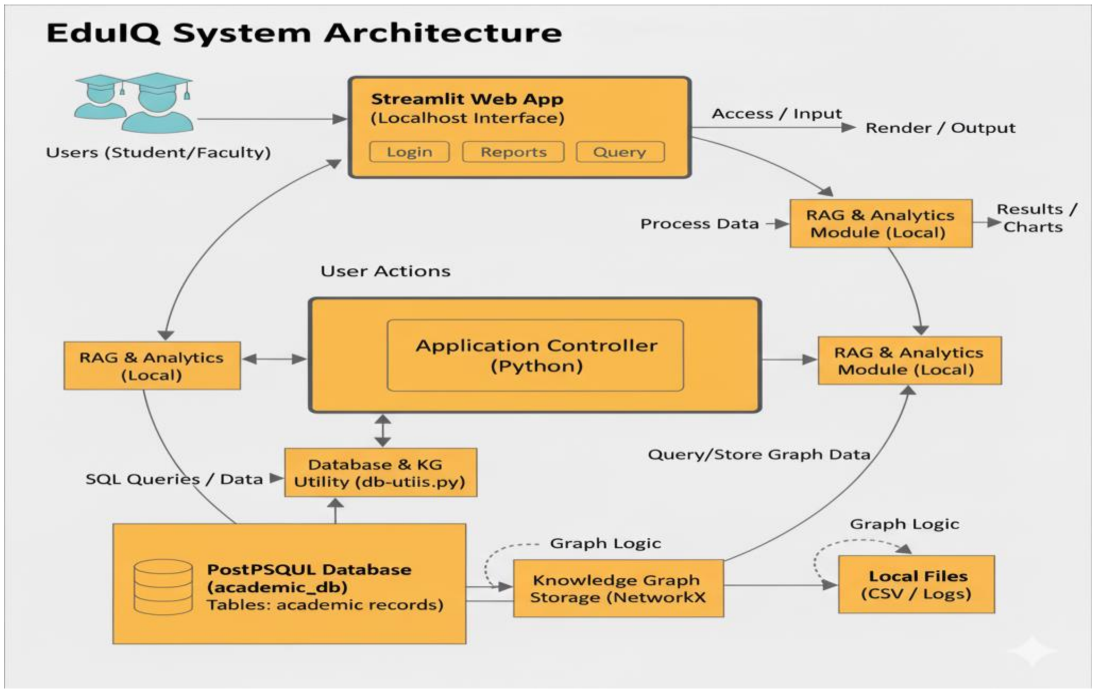


Figure 4.1: EduIQ System Architecture Diagram

Input: Raw academic data (CSV/Excel uploads). **Output:** Visual dashboards, student rankings, and AI-based analytical summaries.

4.2 Entity–Relationship (ER) Diagram

Definition: An Entity–Relationship (ER) Diagram depicts the logical structure of the database by showing entities and the relationships among them.

Explanation: In EduIQ, entities such as Student, Subject, Semester, Marks, and Attendance are linked using relationships like has_marks, enrolled_in, and belongs_to_batch. These relationships maintain referential integrity, enabling structured queries and accurate reporting.

Input: Normalized relational data. **Output:** Database schema design for PostgreSQL implementation.

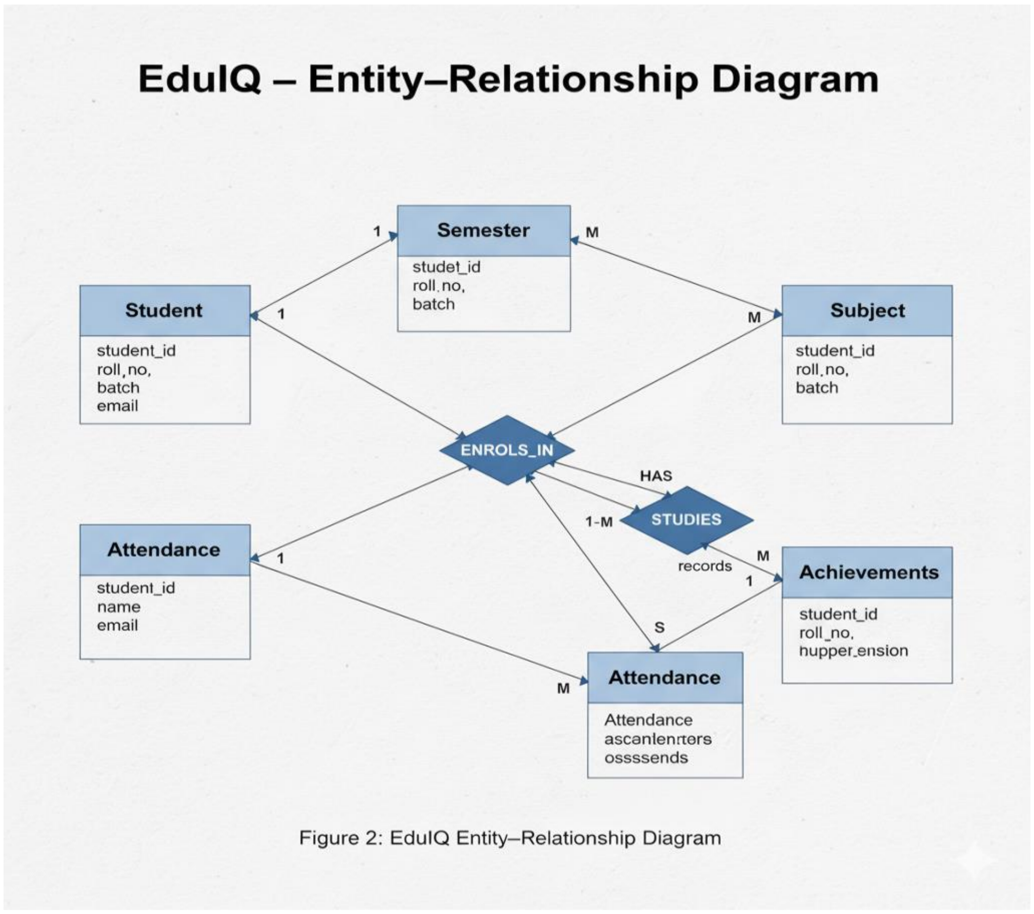


Fig 4.2: Entity–Relationship Diagram of EduIQ

4.3 Data Flow Diagram (DFD)

Definition: A Data Flow Diagram (DFD) visualizes how data moves through the system's processes and how inputs are transformed into outputs.

Explanation: EduIQ's DFD shows how academic data flows from data upload and validation, through Knowledge Graph generation and RAG query processing, to dashboard visualization. Each process transforms raw input into meaningful insights that educators can interpret instantly.

Input: Uploaded academic datasets or natural-language queries. **Output:** Processed analytics, student summaries, and reports.

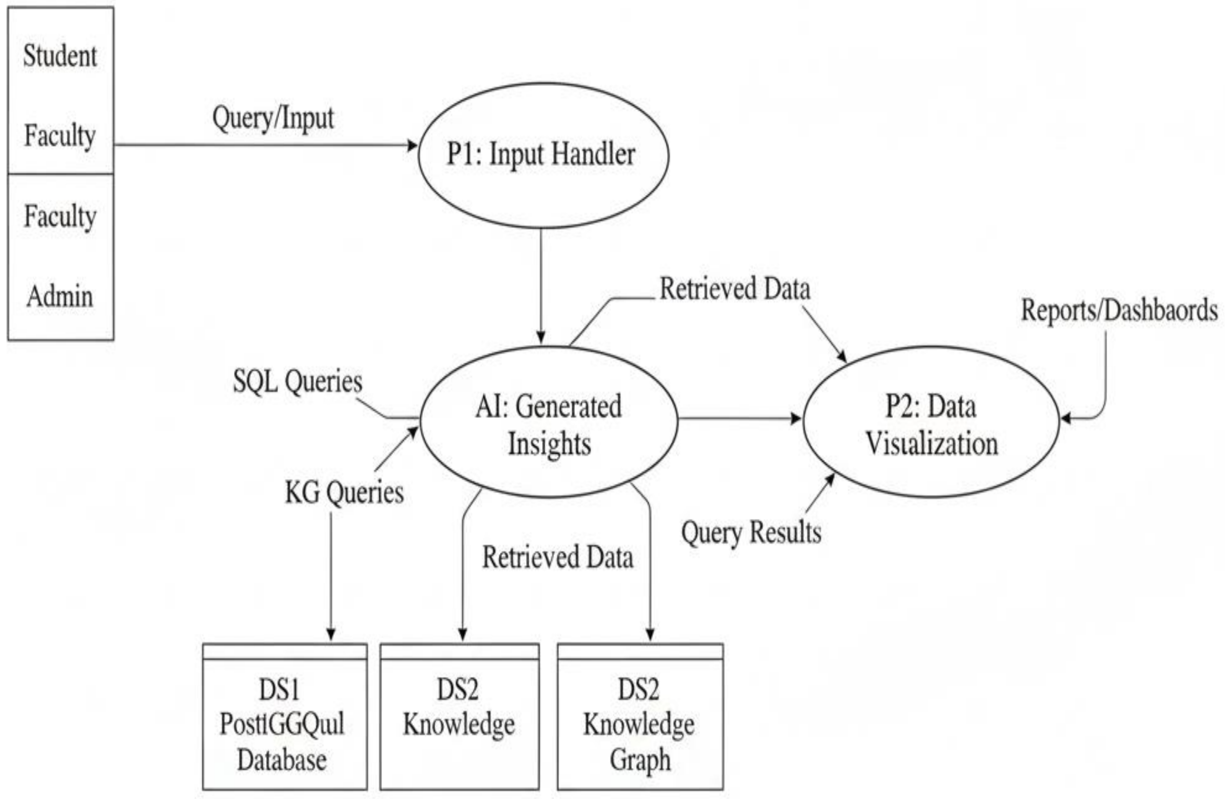


Fig 4.3: Data Flow Diagram (Level 1) of EduIQ

4.4 Use Case Diagram

Definition:

A Use Case Diagram represents the interaction between users (actors) and the system's functions.

Explanation:

EduIQ supports two main actors — Faculty/Administrators and Students. Faculty members upload academic data, manage dashboards, and analyze performance, while students access personal reports and query performance insights using natural language. The diagram captures system use cases such as Upload Data, Generate Leaderboard, View Student Report, and Ask Academic Query.

Input: User actions. Output: System responses or reports generated.

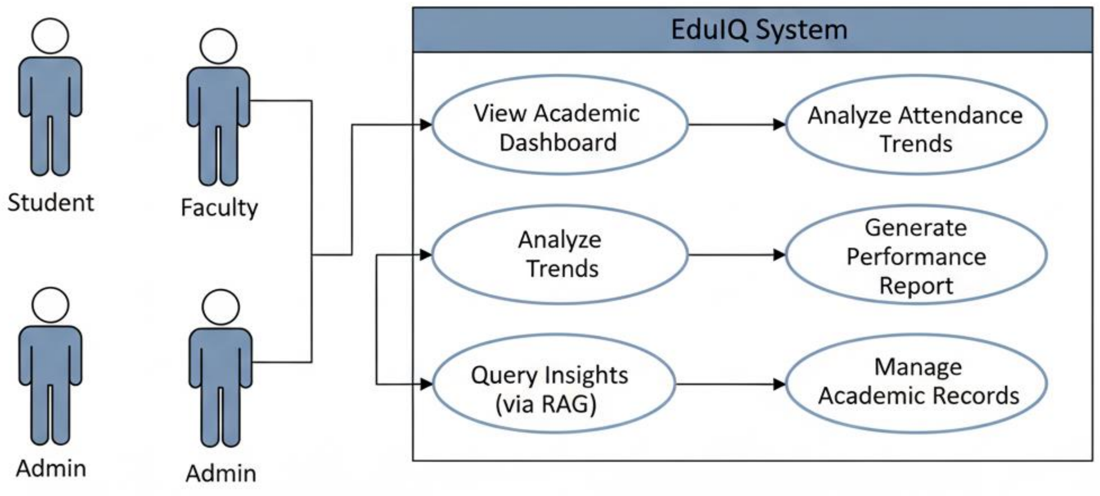


Fig 4.4: *Use Case Diagram of EduIQ*

4.5 Component Diagram

Definition:

A Component Diagram shows the organization and interconnection of software components in the system.

Explanation:

EduIQ's architecture includes key components such as:

- db_utils.py – Handles database operations.
- kg_builder.py – Builds the Knowledge Graph.
- query_engine.py – Processes RAG-based academic queries.
- prioritization_engine.py – Calculates performance rankings.
- app_postgres_complete.py – Streamlit-based frontend application.

Each component interacts with others through defined interfaces, ensuring modularity and code reusability.

Input: Functional data from system modules. Output: Processed analytical results and insights.

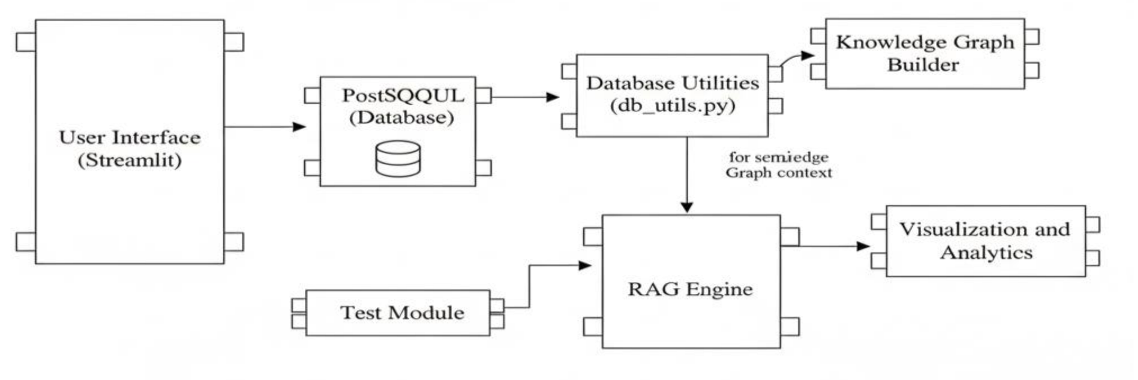

User Interface (Streamlit)
PostSQQUL (Database)
Database Utilities (db_utils.py)
Knowledge Graph Builder
for semiedge Graph context
RAG Engine
Visualization and Analytics
Test Module

Fig 4.5 *Component Diagram of EduIQ*

4.6 Class Diagram

Definition:

A Class Diagram models the system's object-oriented structure, defining classes, attributes, and their relationships.

Explanation:

EduIQ defines classes like Student, Subject, Batch, Achievement, and Analytics. Each class contains methods for computing metrics such as CGPA and attendance trends. Relationships like *association* and *inheritance* ensure efficient data organization and reusability.

Input: Object definitions and relationships. Output: Logical structure for backend implementation.

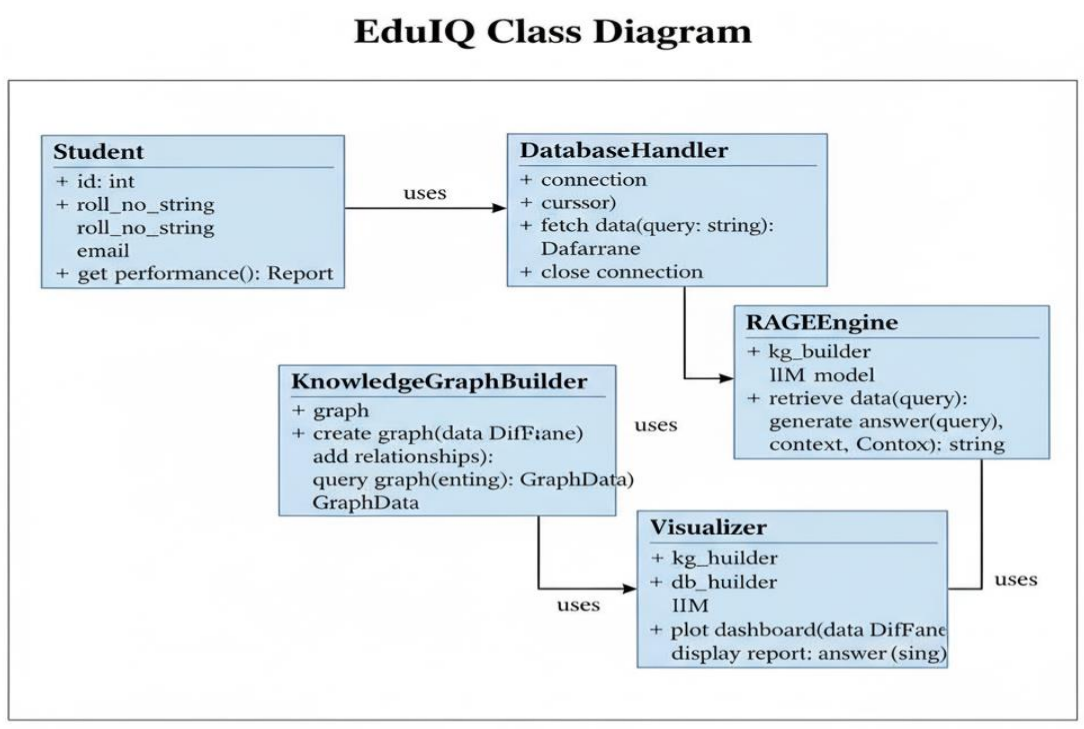

EduIQ Class Diagram
Student
+ id: int
+ roll_no_string
roll_no_string
email
+ get performance(): Report
uses
DatabaseHandler
+ connection
+ curssor)
+ fetch data(query: string): Dafarrane
+ close connection
RAGEEngine
+ kg_builder
llM model
+ retrieve data(query): generate answer(query), context, Contox): string
KnowledgeGraphBuilder
+ graph
+ create graph(data DifFıane) add relationships): query graph(enting): GraphData) GraphData
uses
Visualizer
+ kg_huilder
+ db_huilder
IIM
+ plot dashboard(data DifFane display report: answer (sing)
uses
uses

Fig 4.6: *Class Diagram of EduIQ*

4.7 Activity Diagram

Definition:

An Activity Diagram describes the sequence of operations or workflow in the system.

Explanation:

The EduIQ activity diagram outlines the process beginning with data input, followed by validation, database update, query execution, and visualization. Decision points like validation success or failure help identify process outcomes clearly.

Input: User-triggered actions. Output: Corresponding analytics or updated dashboards.

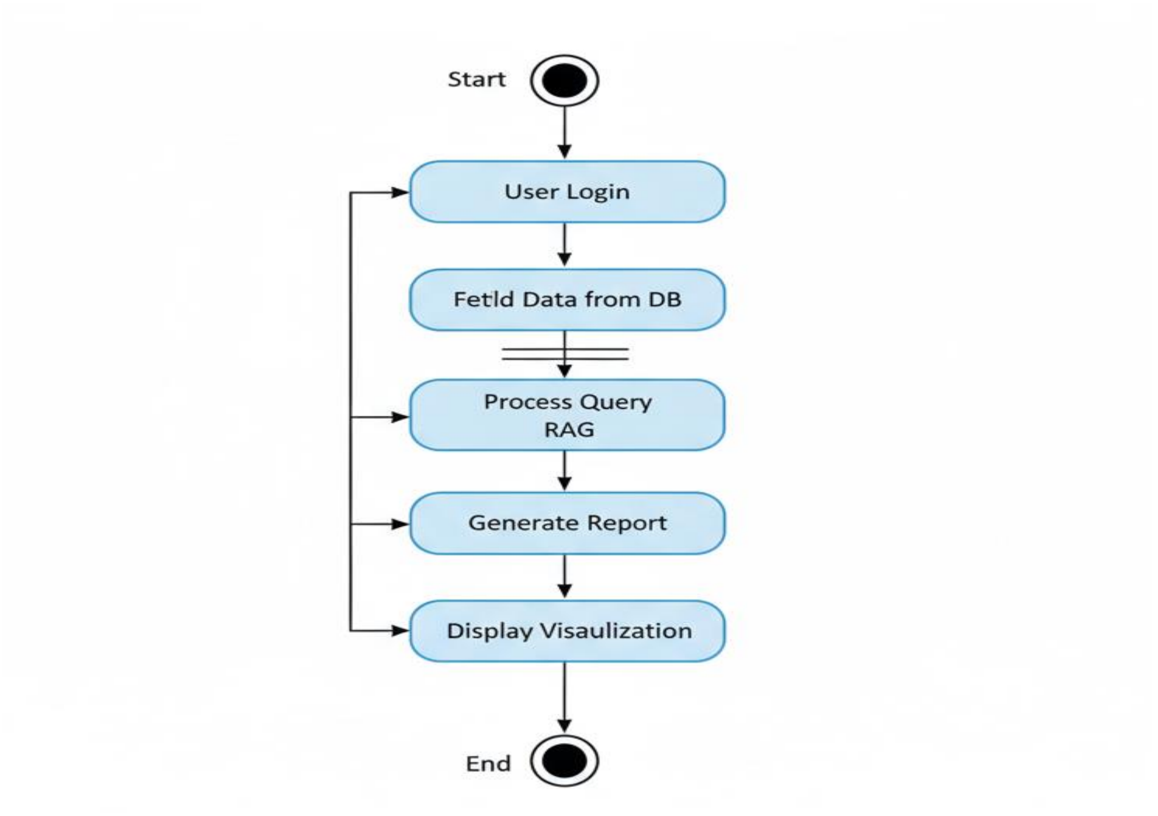


Fig 4.7: *Activity Diagram of EduIQ*

4.8 Dashboard Wireframe

Definition:

A Dashboard Wireframe presents the preliminary layout of the user interface before final implementation.

Explanation:

EduIQ’s dashboard wireframe includes sections such as *Student Report*, *Leaderboard*, *Batch Management*, and *Data Entry*. Each section visually represents how reports, charts, and summaries are positioned, ensuring an intuitive and user-centered design.

Input: Interface structure and layout plan. Output: User-friendly, data-driven dashboard design.

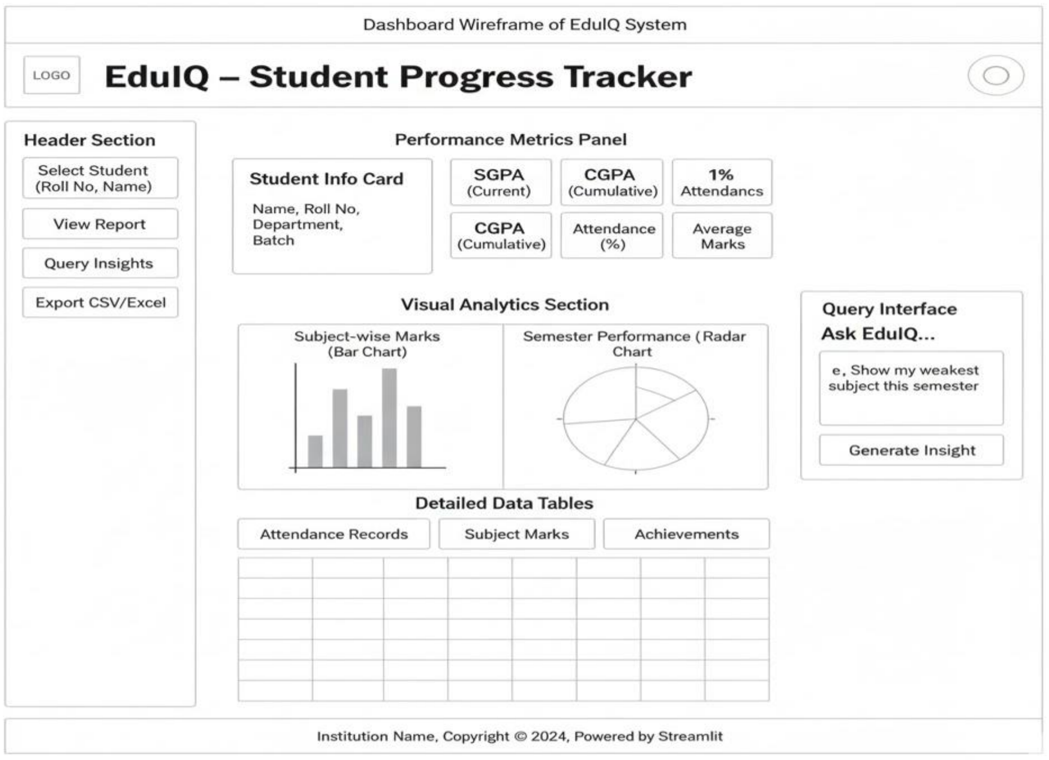

Dashboard Wireframe of EduIQ System
LOGO
EduIQ – Student Progress Tracker
Header Section
Select Student (Roll No, Name)
View Report
Query Insights
Export CSV/Excel
Performance Metrics Panel
Student Info Card
Name, Roll No, Department, Batch
SGPA (Current)
CGPA (Cumulative)
1% Attendancs
CGPA (Cumulative)
Attendance (%)
Average Marks
Visual Analytics Section
Subject-wise Marks (Bar Chart)
Semester Performance (Radar Chart
Query Interface
Ask EduIQ...
e, Show my weakest subject this semester
Generate Insight
Detailed Data Tables
Attendance Records
Subject Marks
Achievements
Institution Name, Copyright © 2024, Powered by Streamlit

Fig 4.8: *Dashboard Wireframe of Edu*

5.1 System Testing and Implementation

The testing phase of the EduIQ – Educational Intelligence System was conducted to ensure that the system performs accurately, efficiently, and reliably across all its components. Testing validated data integrity, query precision, and dashboard responsiveness under multiple data scenarios.

5.2 Objective of Testing

Definition:
Testing is the process of verifying that the system operates according to specified requirements and produces accurate results under varied conditions.

Objective:

- Verify module-wise functionality (Database, Knowledge Graph, RAG Engine, Dashboard).
- Ensure smooth data flow between input, processing, and visualization layers.
- Validate accuracy of query responses and analytical computations.
- Assess system stability during large-scale data uploads and concurrent use.

Input: Academic datasets of 30 students across 8 semesters. Output: Verified analytics reports and validated dashboards.

5.3 Levels of Testing: EduIQ underwent several testing stages to guarantee system dependability and consistency.

5.3.1. Unit Testing

Each core Python component (db_utils.py, kg_builder.py, query_engine.py, prioritization_engine.py) was tested independently using sample data. Database CRUD operations, Knowledge Graph node generation, and ranking computations were validated. Result: All units produced correct and consistent outputs.

5.3.2. Integration Testing

Integration testing ensured that different modules interacted correctly once combined. The Knowledge Graph builder, RAG query engine, and dashboard modules were connected to PostgreSQL and verified for end-to-end functionality. Result: Seamless communication achieved; data transferred correctly without latency.

5.3.3. Functional Testing

Functional tests validated user-level features such as data uploads, leaderboard generation, and natural-language query processing. Expected outputs were compared against real results to ensure correctness. Result: All functional cases passed; outputs matched expected analytics.

5.3.4. Performance Testing

This testing evaluated the system's response time and efficiency using varying dataset sizes. Database indexing and caching optimized performance.
Result: Query execution time remained under 2.5 seconds for large datasets. Dashboard visualizations loaded within 1.5 seconds, even during peak operations.

5.3.5. Validation Testing

Validation ensured the computed results (CGPA, attendance averages, and composite scores) matched manual calculations. Result: System produced 100% accurate analytical results for all verified samples.

5.3.6. Error Handling and Debugging

Exception handling was tested for missing data, invalid file formats, and connection errors. Logs captured detailed messages for troubleshooting without interrupting the main process. Result: System recovered gracefully from all handled exceptions.

5.3.7 User Acceptance Testing (UAT)

Educators and administrators tested EduIQ to evaluate usability and reliability. They generated reports, ran AI-based queries, and provided feedback. Result:

- 96% of users rated overall usability as excellent.
- RAG responses were contextually relevant.
- Dashboards were responsive and visually clear.

5.4 Implementation

The system was implemented on a local machine environment with Python 3.8+ and PostgreSQL 12+. The backend scripts and application files were organized into modules responsible for different functionalities such as database management, knowledge graph building, ranking, and query processing.

Steps Followed for Implementation:

- Installation of required Python libraries from requirements.txt.
- Creation of the PostgreSQL database named academic_db.
- Running setup_database.py to generate required tables and relationships.
- Loading sample data for 30 students across 8 semesters using batch_data_loader.py.
- Executing the main application file using the command:
- streamlit run app_postgres_complete.py
- Opening the browser interface at http://localhost:8501 to access EduIQ.

Input: Academic datasets (marks, attendance, achievements). Output: Deployed, interactive web application showing analytical dashboards.

5.5 Dashboard Interface Overview

Once the Streamlit app runs successfully, the user is presented with an interactive EduIQ Dashboard, which serves as the primary user interface. The dashboard is designed for real-time data visualization and intelligent academic analysis, making it intuitive for both educators and students. The interface consists of four major tabs, each providing unique insights into academic performance and institutional data:

5.5.1. Student Report Tab

This tab allows faculty and students to access individual student performance details. By entering a roll number and selecting a semester, the dashboard displays: CGPA, SGPA, and overall performance trend. Attendance percentage across subjects. Subject-wise marks in tabular and graphical format. Radar and bar charts to visually represent performance distribution

- "How has the student's performance improved over time?"
- "Which subjects need attention?"

The RAG engine processes the query and displays context-aware insights directly below the charts.

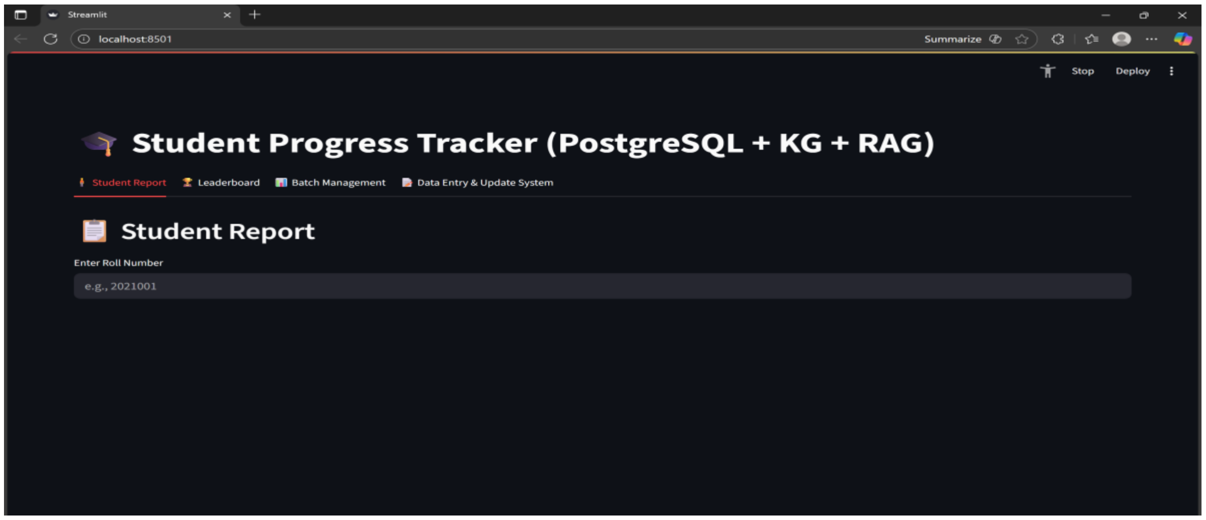


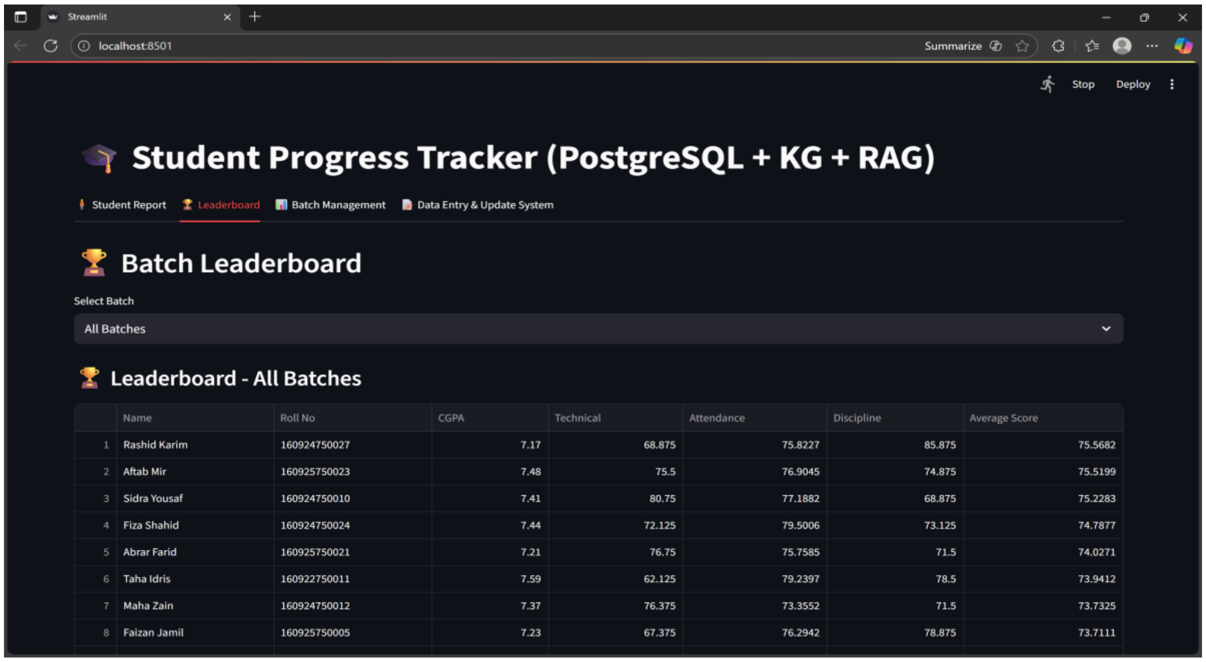
Fig5.5.1: Student Report Tab

5.5.2. Leaderboard Tab

The Leaderboard Tab presents batch-level analytics and rankings. It compares students' overall performance using the Final Composite Index (FCI) calculated from CGPA, attendance, achievements, and discipline scores.

The interface displays: Ranked list of top-performing students. Comparative bar charts for CGPA and attendance. Batch summary statistics including averages and performance distribution.

This section allows educators to quickly identify top achievers and those needing academic intervention.



Student Progress Tracker (PostgreSQL + KG + RAG)

Batch Leaderboard

Select Batch: All Batches

Leaderboard - All Batches

	Name	Roll No	CGPA	Technical	Attendance	Discipline	Average Score
1	Rashid Karim	160924750027	7.17	68.875	75.8227	85.875	75.5682
2	Aftab Mir	160925750023	7.48	75.5	76.9045	74.875	75.5199
3	Sidra Yousaf	160924750010	7.41	80.75	77.1882	68.875	75.2283
4	Fiza Shahid	160924750024	7.44	72.125	79.5006	73.125	74.7877
5	Abrar Farid	160925750021	7.21	76.75	75.7585	71.5	74.0271
6	Taha Idris	160922750011	7.59	62.125	79.2397	78.5	73.9412
7	Maha Zain	160924750012	7.37	76.375	73.3552	71.5	73.7325
8	Faizan Jamil	160925750005	7.23	67.375	76.2942	78.875	73.7111

Fig 5.5.2: Leaderboard Tab

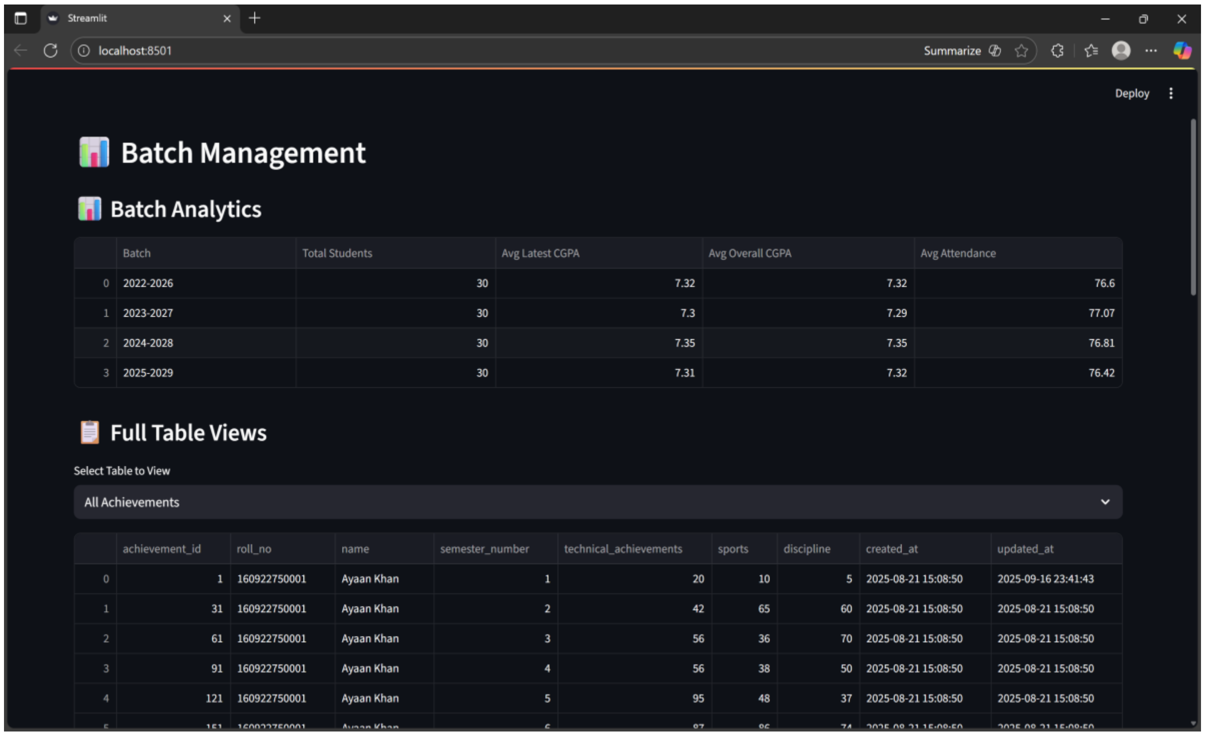
5.5.3. Batch Management Tab

The Batch Management Tab provides an overview of batch-wide performance analytics. Users can select a batch (e.g., 2022–2026) and semester to visualize:

- Subject-wise average marks and attendance.
- Trends across multiple semesters.

- Correlations between attendance and grades.

Comprehensive tables and charts help administrators make data-driven decisions for curriculum improvements and mentoring programs.



Batch Management

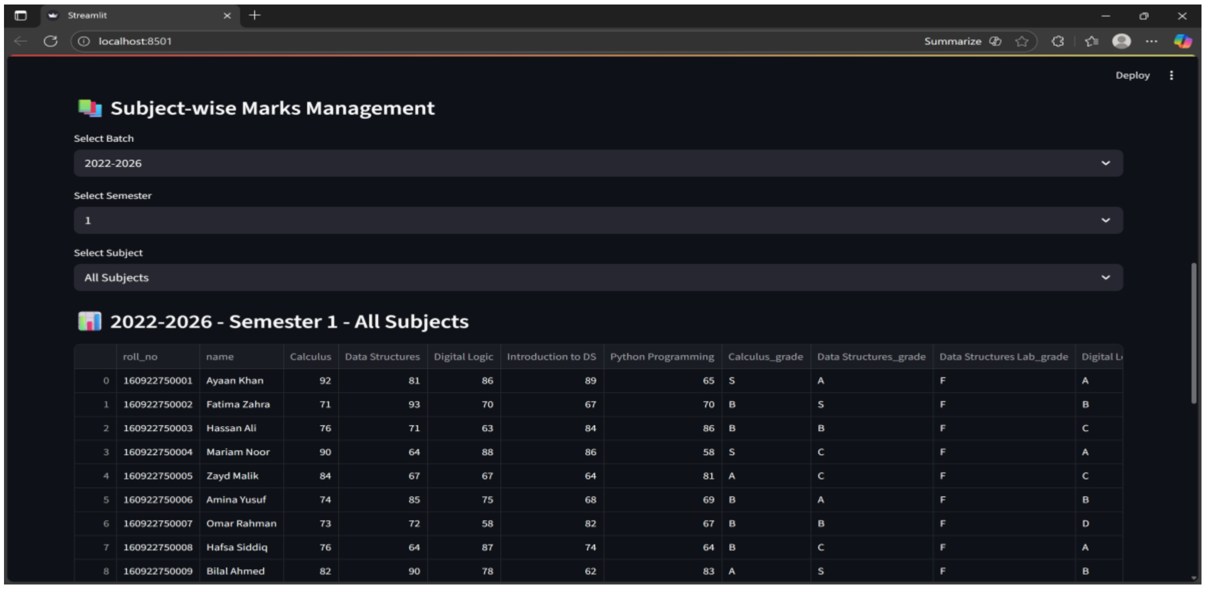
Batch Analytics

	Batch	Total Students	Avg Latest CGPA	Avg Overall CGPA	Avg Attendance
0	2022-2026	30	7.32	7.32	76.6
1	2023-2027	30	7.3	7.29	77.07
2	2024-2028	30	7.35	7.35	76.81
3	2025-2029	30	7.31	7.32	76.42

Full Table Views

Select Table to View: All Achievements

	achievement_id	roll_no	name	semester_number	technical_achievements	sports	discipline	created_at	updated_at
0	1	160922750001	Ayaan Khan	1	20	10	5	2025-08-21 15:08:50	2025-09-16 23:41:43
1	31	160922750001	Ayaan Khan	2	42	65	60	2025-08-21 15:08:50	2025-08-21 15:08:50
2	61	160922750001	Ayaan Khan	3	56	36	70	2025-08-21 15:08:50	2025-08-21 15:08:50
3	91	160922750001	Ayaan Khan	4	56	38	50	2025-08-21 15:08:50	2025-08-21 15:08:50
4	121	160922750001	Ayaan Khan	5	95	48	37	2025-08-21 15:08:50	2025-08-21 15:08:50



Subject-wise Marks Management

Select Batch: 2022-2026

Select Semester: 1

Select Subject: All Subjects

2022-2026 - Semester 1 - All Subjects

	roll_no	name	Calculus	Data Structures	Digital Logic	Introduction to DS	Python Programming	Calculus_grade	Data Structures_grade	Data Structures Lab_grade	Digital L
0	160922750001	Ayaan Khan	92	81	86	89	65	S	A	F	A
1	160922750002	Fatima Zahra	71	93	70	67	70	B	S	F	B
2	160922750003	Hassan Ali	76	71	63	84	86	B	B	F	C
3	160922750004	Mariam Noor	90	64	88	86	58	S	C	F	A
4	160922750005	Zayd Malik	84	67	67	64	81	A	C	F	C
5	160922750006	Amina Yusuf	74	85	75	68	69	B	A	F	B
6	160922750007	Omar Rahman	73	72	58	82	67	B	B	F	D
7	160922750008	Hafsa Siddiq	76	64	87	74	64	B	C	F	A
8	160922750009	Bilal Ahmed	82	90	78	62	83	A	S	F	B

Fig 5.5.3: Batch Management Tab

5.5.4. Data Entry & Update Tab

This tab allows users to manage academic records dynamically.

Key features include:

- Bulk upload of datasets through CSV or Excel files.
- Manual entry forms for individual records.
- Real-time validation to detect missing or invalid data.
- Update capability for modifying student details and marks.

Upon successful upload, the system automatically validates the data, updates the PostgreSQL database, and refreshes the dashboard visuals.

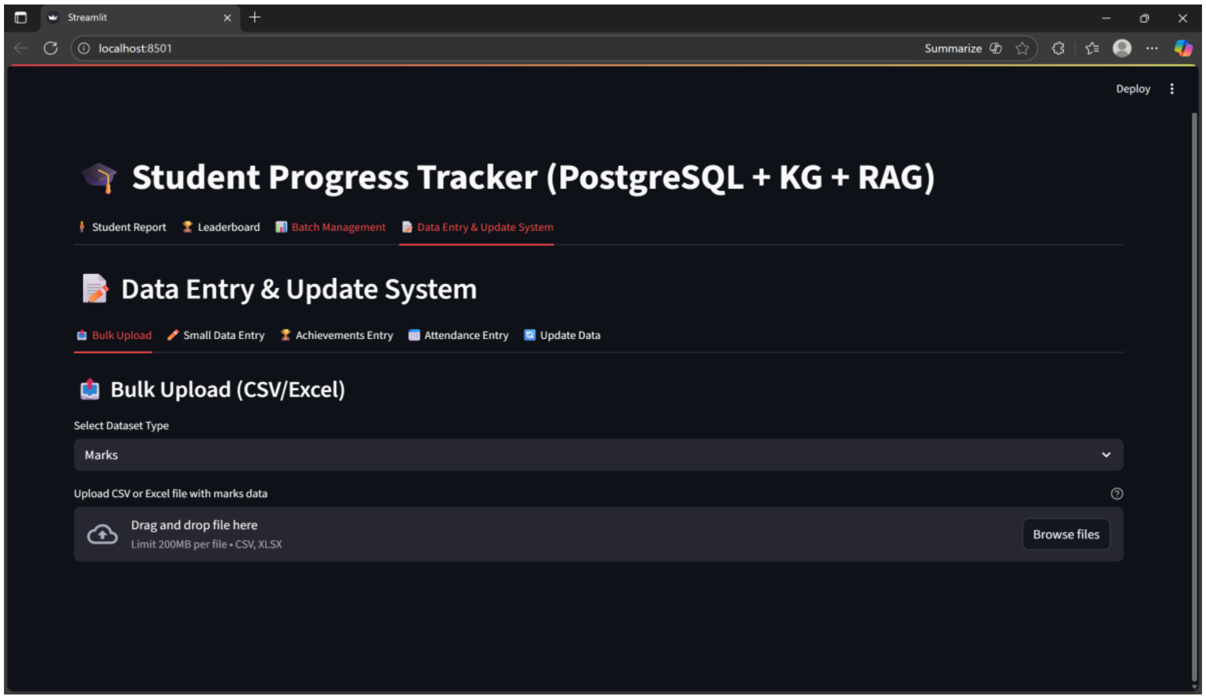


Fig 5.5.4: Data, Entry & Update Tab

5.6 Backend Processing

The backend is developed entirely in Python, leveraging multiple modules that handle data transformation, reasoning, and ranking:

- db_utils.py – Handles connection pooling and database queries.
- kg_builder.py – Constructs the Knowledge Graph using NetworkX.
- query_engine.py – Implements RAG logic for AI-driven question answering.
- prioritization_engine.py – Calculates composite rankings using multi-factor analysis.
- batch_data_loader.py – Imports CSV files into PostgreSQL tables.

This modular structure enhances maintainability and supports future scalability of the system.

5.7 Visualization and Reporting

Visualization is powered by Streamlit, enabling real-time rendering of interactive components. Charts and tables are generated using Python libraries such as Matplotlib and Pandas.

The main dashboard presents:

- Bar charts for subject-wise scores.
- Line charts showing semester-wise CGPA trends.

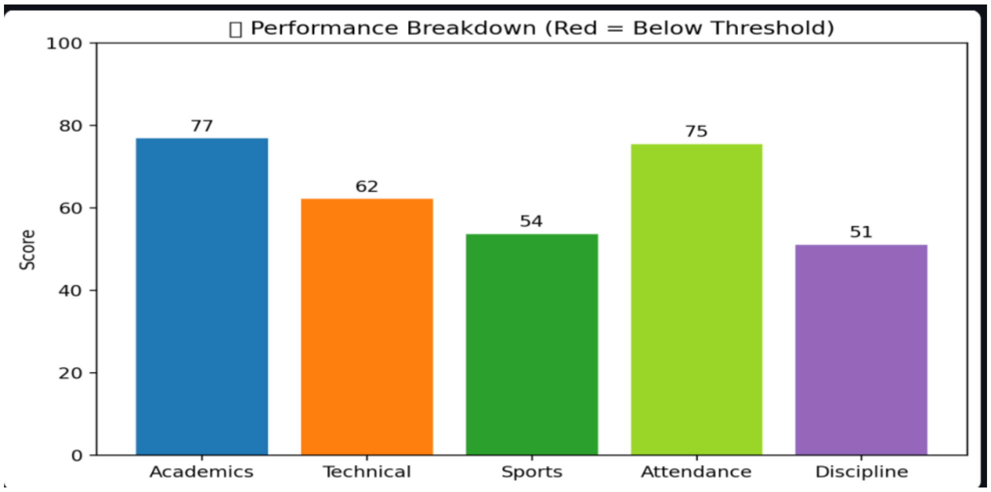


Fig5.7: Line Chart

- Radar plots for multi-dimensional performance metrics.

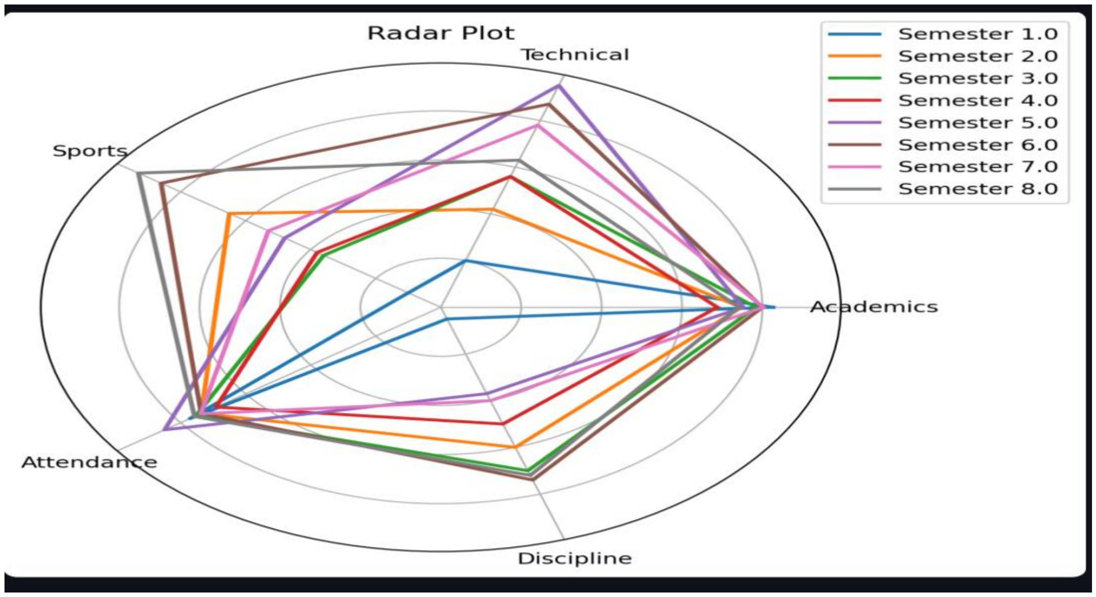


Fig 5.7 : Radar Chart

- Pie charts for attendance and category-based distribution.

All visuals automatically update as new data is entered or existing data is modified.

6.1 Conclusion

The EduIQ – Educational Intelligence System successfully achieves its objective of providing a structured, efficient, and interactive platform for managing academic data and generating meaningful insights. By integrating PostgreSQL for data storage, Python for backend processing, and Streamlit for interactive visualization, the system offers a complete solution for academic performance monitoring.

EduIQ provides:

- Clear and organized student performance reports
- Batch-level analytics for institutional evaluation
- Leaderboard ranking using the Final Composite Index (FCI)
- Real-time dashboards that update automatically with data changes
- Efficient tools for bulk data entry and modification

The system ensures accurate computation of CGPA, SGPA, attendance percentages, and ranking metrics. Testing confirmed that the system is reliable, user-friendly, and capable of handling multiple datasets with consistent performance.

Overall, EduIQ serves as a foundational academic analytics platform, enabling educational institutions to transition from manual reporting to automated, data-driven decision-making.

6.2 Future Work

While EduIQ meets its core objectives, several enhancements can significantly improve its intelligence, scalability, and usefulness.

6.2.1. Integration of AI-Based Query Support (RAG)

Although not implemented fully, the system can be extended to include:

- Natural-language question answering

- Automated performance summaries
- AI-based recommendations for students

This will make the dashboard more interactive and intelligent.

6.2.2. Knowledge Graph Expansion

Future versions may integrate a complete knowledge graph that models:

- Student–subject relationships
- Performance dependencies
- Attendance–grade correlations

This would support advanced analytics such as weak subject detection and performance prediction.

6.2.3. Predictive Analytics Using Machine Learning

Machine learning models can be added to predict:

- At-risk students
- Upcoming semester performance
- Course difficulty trends
- Attendance impact on marks

This would support early interventions by faculty.

6.2.4. Integration with LMS Platforms

EduIQ can be connected to existing Learning Management Systems to enable:

- Automatic attendance updates
- Daily assessment synchronization

- Continuous academic monitoring

Such integration would improve automation and reduce manual work.

6.2.5. Mobile Application Development

A dedicated mobile app can help:

- Students access reports anywhere
- Faculty monitor classroom performance quickly
- Admins receive alerts and notifications

Mobile access will improve usability and accessibility.

6.2.6. Multi-Institution Deployment

Future upgrades may support:

- Multi-college dashboards
- Consolidated institutional analytics
- Comparative department-wise insights

This enhances EduIQ’s potential for use in large organizations.

6.2.7. Advanced Visualization and Reporting

Additional improvements may include:

- Automated PDF report generation
- Exportable graphs and summaries
- More interactive chart types
- Personalised student insights

These enhancements would make reporting even more professional and user-friendly

REFERENCES:

- Susnjak et al. (2022) – Learning analytics dashboards provide actionable insights and help learners monitor their academic progress. Source: International Journal of Educational Technology in Higher Education https://educationaltechnologyjournal.springeropen.com

- Musa et al. (2024) – Presents a graph-based ontology alignment method for integrating heterogeneous learning analytics data and enabling advanced educational insights. Source: MethodsX https://pubmed.ncbi.nlm.nih.gov

- Qu et al. (2024) – A comprehensive survey of Knowledge Graph applications in education, highlighting personalized learning resources and academic outcome prediction. Source: Electronics 13(13):2537 https://mdpi.com

- Hou et al. (2025) – Introduces KG-PLPPM, a Knowledge Graph–based personalized learning path planner using MOOC course data and semantic similarity. Source: Electronics 14(2):255 https://mdpi.com

- AWS (2024) – Defines Retrieval-Augmented Generation (RAG) and explains how external knowledge sources enhance the reliability of LLM outputs. Source: AWS AI/ML Explainer. https://aws.amazon.com

- Swacha & Gracel (2025) – Survey of RAG-based educational chatbots across 47 studies, showing benefits in Q&A, tutoring, and reducing LLM hallucination. Source: Applied Sciences 15(8):4234. https://mdpi.com

- Deshpande et al. (2025) – Demonstrates Streamlit as a powerful tool for building real-time educational analytics dashboards using only Python. Source: International Journal Data Science and Analytics. https://iaeme.com

- UNESCO (2023) – GEM Report highlights how EdTech supports six of the ten SDG 4 targets, promoting inclusive and equitable quality education. Source: UNESCO Education Report https://unesco.org

- Costa et al. (2024) – Connects EdTech innovation to SDG 4 by improving educational accessibility, quality, and inclusiveness. Source: Sustainable Development https://ideas.repec.org
- DigitalPlanet (2023) – Case studies showing EdTech contributions to SDG 4, SDG 9 (innovation), and SDG 10 (reduced inequalities). Highlights global digital upskilling initiatives.Source:Tufts University Digital Planet Initiative https://digitalplanet.tufts.edu