ISSN: 3139-0862

Design and Implementation of a Centralized Academic Project Archive System Using MySQL

Nilanjan Mane[1], Lokesh Patil[2], Varad Jamdar[3]

Department of Information Technology, AISSMS Institute of Information Technology, Pune, Maharashtra, India

[1]nilanjanmane13@gmail.com, [2]patillokesh2206@gmail.com, [3]varry1257@gmail.com

Abstract:
Academic project records in educational institutions are often scattered across spreadsheets, folders and individual memory, which makes earlier work hard to find, compare and reuse. This paper presents the design and implementation of the Academic Project Archive System, a centralized relational database built with MySQL Server and MySQL Workbench. The schema, named project archive, contains 14 tables that model projects, students, teams, guides, domains, technologies, competitions, documents, reviews and project outputs. Four junction tables resolve many-to-many relationships, and 13 foreign-key constraints enforce referential integrity. The implementation consists of SQL scripts for schema creation, sample-data loading, joins, filtering, grouping, nested subqueries, three views and reporting queries. Executing the scripts on a sample dataset of 50 projects (IDs 301 to 350) produced domain-wise, status-wise, year-wise, technology and competition summaries, for example 34 completed and 16 in-progress projects across 13 domains. The study also records design limitations, including the absence of uniqueness and check constraints, which allowed a duplicate enrolment number in the sample data. The contribution is not a new database technique but a documented, reproducible schema that links projects with competition outcomes and publication or patent records, together with an honest account of its limits. No user interface, performance benchmark or user study is reported.

Keywords—Database Management System; MySQL; Academic Project Archive; Relational Database; SQL

I. INTRODUCTION

A. Background

Every engineering institution produces a steady stream of academic projects: mini projects, capstone projects, hackathon prototypes, research-oriented work and competition entries. Each project involves several kinds of information at once. There are students organised into teams, a faculty member who supervises the work, a problem domain, a set of technologies, a source repository, reports and presentations, review comments and, sometimes, a competition result, a research paper or a patent application. Traditionally this information has been kept as printed reports in a department library, as files on a shared drive, or as rows in a spreadsheet maintained by a project coordinator. Institutional repositories for theses and dissertations have a longer history and are well established, but they usually concentrate on the final document and its bibliographic description rather than on the relationships around it [2].

The relational model proposed by Codd [9] gives a principled way to store such interrelated facts. Data is held in relations, identified by keys and linked by key references, so that each fact is stored once and queried declaratively. Chen's entity–relationship model [10] complements it by providing a way to reason about entities and relationships before tables are designed. Together, these two ideas form the standard route from a real-world description to a working database, and they are the route followed in this work.

B. Problem statement

When project information is held in separate files, several problems follow in practice. The same student may appear under slightly different names in different sheets. A team's membership may be recorded in one file and its project in another, with no reliable link between them. A coordinator who wants to know which projects used a particular technology, which ones reached a national competition, or which ones produced a paper has to open and cross-read many files by hand. Two copies of a record can be updated differently, so that the archive contradicts itself. Past projects become difficult to retrieve, and as a result students sometimes repeat problem statements that were already addressed in earlier batches, and faculty cannot easily judge how project activity is distributed across domains and years.

The prototype final-year-project systems reported in the literature show that institutions do recognise these difficulties [1], [3]. However, those systems are described mainly as workflow applications for proposals, supervision and grading. The database design that underlies them, and the way it relates projects to technologies, competitions and outputs, is seldom described in enough detail to be reproduced or critiqued.

C. Need for a centralized relational database

A centralized relational database addresses these problems in three ways. First, it stores each entity once and refers to it by key, which removes most duplication. Second, declared primary- and foreign-key constraints make it impossible to record, for example, a review for a project that does not exist, so the archive stays internally consistent. Third, SQL provides one uniform way to answer questions that cut across entities: joins combine students, teams and projects; grouping summarises domains, years or outcomes; and views package recurring reports. Relationships are central here. Students belong to teams, teams work on projects, projects belong to domains and are guided by faculty, use technologies, take part in competitions, and have documents, reviews and outputs. A flat file cannot express these links without repeating data, whereas a relational schema expresses them directly.

D. Proposed approach, objectives and scope

The proposed Academic Project Archive System is a MySQL database named project_archive. It was developed with MySQL Server and MySQL Workbench, using SQL scripts that create the schema, load sample records and run retrieval and reporting queries. The objectives are:

ISSN: 3139-0862

- to design a relational schema that represents academic projects together with their students, teams, guides, domains, technologies, competitions, documents, reviews and outputs;
- to enforce referential integrity through primary and foreign keys;
- to implement SQL queries for retrieval, filtering, searching, aggregation and reporting; and
- to evaluate the schema honestly, stating what has been implemented and what is missing.

The scope is deliberately limited to the database layer. The work does not include a web or mobile interface, user authentication, automated approval workflows, file upload, or integration with an institutional ERP. The DOCUMENT table stores a file name, type and path, not the file itself. Practical uses within this scope include a searchable record of earlier projects for new student teams, year-wise and domain-wise reports for coordinators, and a register of competition results and publication or patent records for accreditation and departmental reporting.

E. Research contribution

This paper does not claim a new database technique or a completely new kind of system. Project management systems and repositories already exist [1]–[4]. The contribution is narrower and can be checked against the supplied files. (i) A complete 14-table schema is documented, including the junction tables and a PROJECT_OUTPUT table for research papers and patents. (ii) The schema links projects to competition outcomes, which the final-year-project systems in the literature do not emphasise, even though hackathon research shows that such outcomes relate to what happens to a project afterwards [5]. (iii) The SQL scripts are analysed for what they demonstrate, and the actual results of running them on the sample dataset are reported. (iv) Weaknesses of the current design are stated explicitly, together with differences between the conceptual ER diagram and the implemented schema.

II. LITERATURE SURVEY

A search was made for work published between 2020 and 2026 on academic project repositories, student project management systems, institutional digital repositories, relational design for academic information systems, and hackathon project documentation and reuse. Each paper in Table I was opened at its publisher, repository or preprint page, and its authors, year, title, venue and (where shown) DOI were checked against that page. The method and limitations reported below are those stated by the papers' own abstracts and text; no paper is included that could not be opened.

TABLE I. LITERATURE REVIEW TABLE (VERIFIED PAPERS, 2020–2024)

Sr. No.	Author(s) and Year	Paper Title	Method / Technology	Key Findings	Limitations / Research Gap
1	Ramli, Hibadullah and Alwi (2021) [1]	SCIMPA FYP Management System	Web-based system built by prototyping; evaluated with 20 respondents on a 5-point scale.	Respondents mostly agreed the system was useful and easy to learn; archiving and retrieval of past projects is a stated aim.	Small sample (mostly students); database and technology stack not specified; reporting for department heads still planned.
2	Tsolakidis et al. (2023) [2]	Developing the infrastructure for managing institutional research archives	Ontology-based, linked-data repository built by adapting VIVO at the University of West Attica.	Harvesting and merging sources gave a consistent institutional research repository with better visibility.	Covers research outputs of staff, not student project teams; requires semantic-web infrastructure.
3	Isa et al. (2024) [3]	Prototype Development of Final Year Project Management System to Monitor Student's Performance	PHP/JavaScript web dashboard with a MySQL database; roles for student, lecturer and coordinator.	Prototype to replace manual logbook monitoring; benefits are expected rather than measured.	No testing or user-study results; one faculty; schema not described.
4	Salvador et al. (2024) [4]	Development of a Web-based Research Consortium Database Management System: Advancing Data-driven and Knowledge-based Project Management	Web-based database system for a 29-institution research consortium, with data entry and reporting parts.	Centralizes project data to support information sharing and near-real-time monitoring.	Research consortium, not student projects; evaluation details not given in the abstract.
5	Nolte, Chounta and Herbsleb (2020) [5]	What Happens to All These Hackathon Projects? Identifying Factors to Promote Hackathon Project Continuation	Large-scale quantitative study of hackathon projects.	Many projects continue; short- and long-term continuation relate to different factors (e.g. technologies used, winning, team skill diversity).	Studies continuation, not how project records should be stored or retrieved.
6	Imam et al. (2021) [6]	The Secret Life of Hackathon Code: Where does it come from and where does it go?	Archival analysis of 22,183 Devpost projects traced through World of Code.	About 28.8% of event-created code blobs were reused elsewhere; reuse rose with the number of technologies and participants.	Exact-match reuse only; assumed 72-hour events; preprint.
7	Ratsoga and Primus (2023) [7]	Factors influencing post-hackathon project continuation in an African corporate setting	Case study of one company's hackathons held in 2018–2020.	Funding, team skills and integration with operations influenced continuation.	Single corporate case; not an educational setting.

ISSN: 3139-0862

Sr. No.	Author(s) and Year	Paper Title	Method / Technology	Key Findings	Limitations / Research Gap
8	Suryawasnhi, Sonwale and Deshpande (2023) [8]	Analysis of the Smart India Hackathon (SIH): A Data-Driven Approach	Web-scraped SIH records, preprocessed in Python and visualised in Power BI.	Participation and winning patterns differ by technology category and location.	Analyses public event records; no persistent archive design; SSRN posting of a conference paper.

A. Indian hackathon and student-project repository research

The same search was extended to Indian hackathons, hackathon project documentation, digital repositories of student projects and reuse of earlier hackathon work. This part of the search is indicative only. It used a general web search engine and publisher pages on 31 October 2026, with terms such as "Indian hackathon", "Smart India Hackathon", "hackathon project continuation", "hackathon code reuse" and "student project repository". It was not a systematic review across IEEE Xplore, Scopus or the ACM Digital Library with a fixed inclusion protocol, so no publication count is claimed.

Within these limits the picture is as follows. International hackathon research is active, and covers continuation [5], code reuse [6] and corporate settings [7]. Work focused on Indian hackathons is much thinner: the only India-specific research item located and verified was the data analysis of the Smart India Hackathon [8], itself a conference paper posted on SSRN. A further item on SIH 2019 turned up in the search, but its bibliographic details could not be verified and it is therefore not cited. Many Indian institutional pages report hackathon participation, but these are event reports and not research. No paper was found that proposes a relational archive in which hackathon or competition entries are stored as first-class records linked to students, teams, guides, technologies and documents for later retrieval and reuse.

B. Research gap

Three gaps follow. The student-project systems [1], [3] describe workflows but not their schemas. The repository work [2], [4] addresses research outputs and consortia, not student teams. The hackathon studies [5]–[8] examine outcomes and reuse but do not describe how institutions should record projects so that earlier work can be found. The present paper addresses this narrower gap by documenting a concrete, inspectable relational schema.

III. METHODOLOGY

A. System overview

The system is a MySQL database, project_archive, driven entirely by SQL scripts. Four scripts make up the implementation: 01_schema.sql (database and table creation, with later ALTER statements), 02_sample_data.sql (sample records), 03_queries.sql (retrieval queries, relational-operator exercises, views, update transactions and an output-summary report) and 04_Project_Retrieval_and_Reports.sql (a fourteen-section retrieval and reporting dashboard). A user interacts with the system (SQL syntax per the MySQL reference manual [12]) by running these scripts or ad-hoc SQL in MySQL Workbench. There is no application layer. The working cycle is: record entities and relationships, then retrieve and summarise them with SQL.

B. Entity–relationship design

The conceptual design was drawn as an ER diagram (Fig. 1) with draw.io. Its entities are Student, Team, Project, Domain, Guide, Technology, Competition, Document and Review. Relationship types are Member_Of (Student–Team, with attribute Role), Works_On (Team–Project), Belongs_to (Domain–Project), Guided_By (Guide–Project), Uses (Project–Technology), Participates_IN (Project–Competition, with attributes result, rank and award) and two Has relationships (Project–Document, Project–Review). Key attributes are underlined in the diagram, and Email and Phone of Student are drawn as double ellipses.

 ISSN: 3139-0862

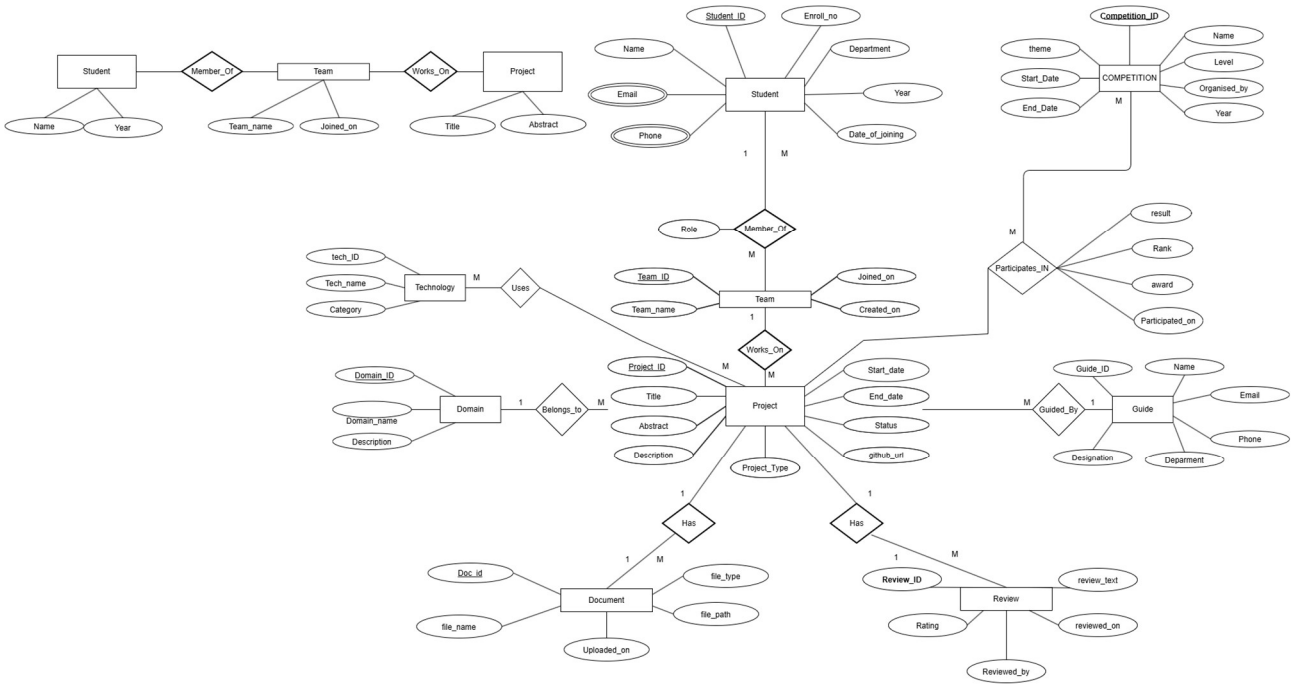


Fig. 1. Entity–relationship diagram of the Academic Project Archive System (draw.io diagram).

The cardinalities in the implemented schema are as follows. Student–Team is many-to-many, resolved by MEMBER_OF. Team–Project is many-to-many, resolved by WORKS_ON. Project–Technology is many-to-many, resolved by USES. Project–Competition is many-to-many, resolved by PARTICIPATES_IN. Domain–Project and Guide–Project are one-to-many, implemented as foreign keys in PROJECT. Project–Document, Project–Review and Project–Output are one-to-many, with a Project_Id foreign key in each of the three child tables. No one-to-one relationship is present.

The SQL files are treated as authoritative, and three differences were found. (1) The diagram labels Team–Project as one-to-many, whereas WORKS_ON allows a team to have several projects; in the sample data four teams work on two projects each. (2) The diagram draws Email and Phone as multi-valued, but STUDENT stores each as a single column. (3) The diagram has no PROJECT_OUTPUT entity, no Problem_Statement attribute and no Achievement_Reason, Judge_Feedback or Improvement_Recommendation attributes, all of which exist in the implemented schema, and it shows a Participated_on attribute that has no column. The supplied MySQL Workbench reverse-engineered diagram appears to come from an earlier version of the schema (it shows a team_id column in PROJECT and has no PROJECT_OUTPUT table). Neither diagram was altered. Both should be regenerated from the final database before submission.

C. *Relational database design*

Each entity becomes a table with a primary key, and each many-to-many relationship becomes a junction table whose composite primary key is made of the two foreign keys. Table II summarises the implemented structure; the design follows standard textbook practice [11]. In total there are 14 tables with 80 columns and 13 foreign-key constraints, all declared inline in CREATE TABLE statements; no ON DELETE or ON UPDATE rule is specified, so MySQL's default restrictive behaviour applies. Referential integrity therefore prevents deleting a project that still has documents, reviews, outputs or junction rows, and prevents inserting a child row whose parent does not exist.

Normalization. By inspecting the declared attributes, every table is in first normal form because attributes are atomic and each table has a primary key. In the junction tables, the non-key attributes (Role, Result, Competition_Rank, Award and the feedback fields) depend on the whole composite key. In the entity tables, the descriptive attributes appear to depend only on the surrogate key, so under the natural functional dependencies of this domain the schema appears to satisfy third normal form. This is an informal assessment and not a formal proof. Several points weaken it: Technology.Category and Student.Department are repeated text values that could become lookup tables; PROJECT.Project_type partly overlaps with the domain relationship (13 distinct Project_type values against 13 domains in use, with similar but not identical labels); and Review.Reviewed_By is free text and not a key. Consistency is supported by keys and foreign keys, but not by unique or check constraints (Section IV-F).

D. *SQL implementation*

The scripts use data-definition, data-manipulation and query statements. Selected examples follow, each taken from the supplied files.

Database and table creation (01_schema.sql). A junction table with a composite primary key and two foreign keys:

```
CREATE TABLE MEMBER_OF (
  Student_Id INT, Team_Id INT, Role VARCHAR(50),
  PRIMARY KEY (Student_Id, Team_Id),
  FOREIGN KEY (Student_Id) REFERENCES STUDENT(Student_Id),
  FOREIGN KEY (Team_Id) REFERENCES TEAM(Team_Id));
```

The schema then evolves through ALTER TABLE statements: columns are renamed to a consistent capitalised style, PROJECT gains Abstract, Github_url, Project_type and Problem_Statement columns, PARTICIPATES_IN gains Achievement_Reason and Judge_Feedback, and REVIEW gains Improvement_Recommendation. PROJECT_OUTPUT is added late, for research papers and patents.

 ISSN: 3139-0862

Sample data (02_sample_data.sql). The script inserts rows with INSERT statements. It first inserts five records per reference table and re-keys them with UPDATE statements (for example domain IDs 401–450, guide IDs 501 onwards, student IDs from 101, team IDs from 201), then adds the full set: 50 projects (301–350), 50 students, 50 teams and so on.

Joins and reporting (03 and 04). The project listing joins PROJECT with DOMAIN and GUIDE:

```
SELECT P.Project_Id, P.Title, D.Domain_Name, G.Name AS
Guide_Name
FROM PROJECT P JOIN DOMAIN D ON P.Domain_Id = D.Domain_Id
       JOIN GUIDE  G ON P.Guide_Id  = G.Guide_Id;
```

Filtering, sorting and aggregation. The scripts use AND, OR, NOT, LIKE, ORDER BY, COUNT, AVG, MAX, MIN, SUM, GROUP BY and HAVING. For example, the domain report counts projects per domain with an outer join so that empty domains would still appear:

```
SELECT D.Domain_Name, COUNT(P.Project_Id) AS
Total_Projects
FROM DOMAIN D LEFT JOIN PROJECT P ON D.Domain_Id =
P.Domain_Id
GROUP BY D.Domain_Id, D.Domain_Name
ORDER BY Total_Projects DESC, D.Domain_Name;
```

Subqueries. Nested queries find projects whose review rating exceeds the average rating:

```
SELECT Title FROM PROJECT WHERE Project_Id IN (
 SELECT Project_Id FROM REVIEW
 WHERE Rating > (SELECT AVG(Rating) FROM REVIEW));
```

Views and transactions. Three views are created: Hackathon_Achievements (a six-table join of competitions, results, projects, teams and domains), Project_Status_View and Student_Count_By_Department. The scripts also use a START TRANSACTION…COMMIT block with a CASE expression to reassign the domain of all 50 projects in one update, and a LEFT JOIN report with CASE to classify each project as having no outputs, a paper, a patent, or both.

IV. Experimentation and Results

All figures in this section come from executing the supplied scripts. Because a MySQL Workbench session or screenshots were not supplied, the scripts were re-executed for this paper on a MariaDB 10.11 server, which is MySQL-compatible for the statements used. Two adjustments were needed: the sample-data script was run after the column rename in 03_queries.sql, because 02 refers to a column that 01 names differently (Section IV-F), and Linux's case-sensitive table names caused harmless errors in two SELECT statements that write Domain and Guide in mixed case. These counts should be confirmed in MySQL Workbench before submission.

A. Database overview

The database project_archive contains 14 base tables and 3 views. Of the 14 tables, 12 hold 50 rows each, WORKS_ON holds 54 (four projects are linked to two teams each) and PROJECT_OUTPUT holds 35. The 50 projects have IDs 301–350, which matches the project context, and their status is Completed for 34 and In Progress for 16. The sample data appear to be illustrative: the PROJECT_OUTPUT rows are labelled in their own Details and identifier fields as illustrative or sample records, and some GitHub URLs do not correspond to the project title they are attached to (for example project 301 carries the URL of a repository named student-performance-prediction). The results therefore demonstrate the schema and queries and say nothing about real institutional activity.

B. Table description

TABLE II. Tables of the project_archive database (from 01_schema.sql and 03_queries.sql)

Table Name	Purpose	Important Attributes / Keys
PROJECT	Core record of each academic project	PK Project_Id; FK Domain_Id, Guide_Id; Title, Problem_Statement, Description, Abstract, Project_type, Start_Date, End_Date, Status, Github_url
STUDENT	Student details	PK Student_Id; Name, Email, Phone, Enroll_No, Year, Department, Date_Of_Joining
TEAM	Project teams	PK Team_Id; Team_Name, Joined_On, Created_On
DOMAIN	Domain classification	PK Domain_Id; Domain_Name, Description
GUIDE	Supervising faculty	PK Guide_Id; Name, Email, Phone, Designation, Department
TECHNOLOGY	Technologies and tools	PK Tech_id; Tech_name, Category
COMPETITION	Competitions and hackathons	PK Competition_id; Name, Theme, Start_date, End_date, Level, Organized_by, Year
DOCUMENT	Project file metadata (name, type, path)	PK Doc_Id; FK Project_Id; File_Name, File_Type, File_Path, Uploaded_On
REVIEW	Ratings and feedback	PK Review_Id; FK Project_Id; Rating, Review_Text, Reviewed_On, Reviewed_By, Improvement_Recommendation
MEMBER_OF	Student–Team membership (M:N)	PK (Student_Id, Team_Id); both FK; Role
WORKS_ON	Team–Project assignment (M:N)	PK (Team_Id, Project_Id); both FK
USES	Project–Technology link (M:N)	PK (Project_Id, Tech_Id); both FK
PARTICIPATES_IN	Project–Competition entry (M:N)	PK (Project_Id, Competition_id); both FK; Result, Competition_Rank, Award, Achievement_Reason, Judge_Feedback
PROJECT_OUTPUT	Research papers and patents of a project	PK Output_Id; FK Project_Id; Output_Type, Title, Status, Application_or_Paper_ID, Filed_Date, Published_Date, Details

ISSN: 3139-0862

C. SQL query execution

Table III lists representative queries from the supplied scripts and the output obtained on the sample data.

TABLE III. REPRESENTATIVE QUERIES AND OBSERVED OUTPUTS

Query (script)	Purpose	Observed output on sample data
Project–domain–guide join (03)	Retrieve each project with its domain and guide	50 rows. First: 301 Academic Project Archive System, Database Management, Dr. Mrunal Pathak
Status count (03, 04 §7)	GROUP BY Status	Completed 34; In Progress 16
Year-wise count (04 §9)	GROUP BY YEAR(Start_Date)	2023: 10; 2024: 10; 2025: 12; 2026: 18
Winning projects (04 §10)	Join with Result = 'Winner'	13 winners; first row: project 301, IEEE SYNAPSE 2026, rank 1
Above-average rating (03, nested)	Subquery on REVIEW	Mean rating 3.72 (50 reviews, range 1–5); 33 projects above the mean
Keyword search 'IoT' (04 §12.4)	LIKE on Title and Problem_Statement	2 projects (303 and 331)
Output summary (03, LEFT JOIN + CASE)	Papers and patents per project	35 output records on 34 projects: 20 papers (13 published, 7 submitted), 15 patents (10 filed, 5 granted); all labelled illustrative in the data
Hackathon_Achievements view	Six-table join	54 rows (50 entries, plus extra rows for the four two-team projects)
Duration report (04 §14)	DATEDIFF on completed projects	Completed projects: minimum 6, maximum 878, mean 418.6 days

No Workbench result-grid screenshots were supplied; the outputs above were obtained by re-running the scripts, as noted at the start of this section, and are not fabricated.

D. Charts from SQL-derived counts

Figures 2 and 3 plot counts obtained from queries on the loaded database. Thirteen of the 50 domain rows are used by at least one project (37 are unused), with Internet of Things (9), Artificial Intelligence (8) and Data Science (7) the largest. Fig. 2(b) shows that all 16 in-progress projects began in 2026, and that 2026 is the busiest start year (18 projects). In Fig. 3(a), technology usage is shown by category because the sample has exactly 50 project–technology links across 49 distinct technologies, so a per-technology chart would be nearly flat. Likewise each of the 50 competitions appears exactly once in PARTICIPATES_IN, so Fig. 3(b) groups entries by competition level and result instead: 13 winners, 12 runners-up and 25 participants.

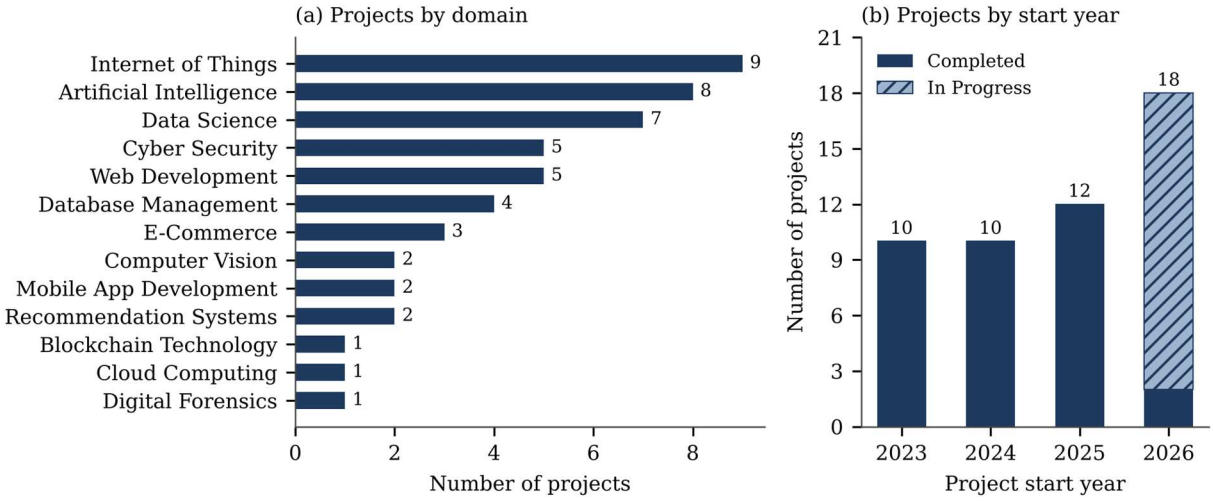


Fig. 2. (a) Projects by domain and (b) projects by start year and status, from the sample database.

 ISSN: 3139-0862

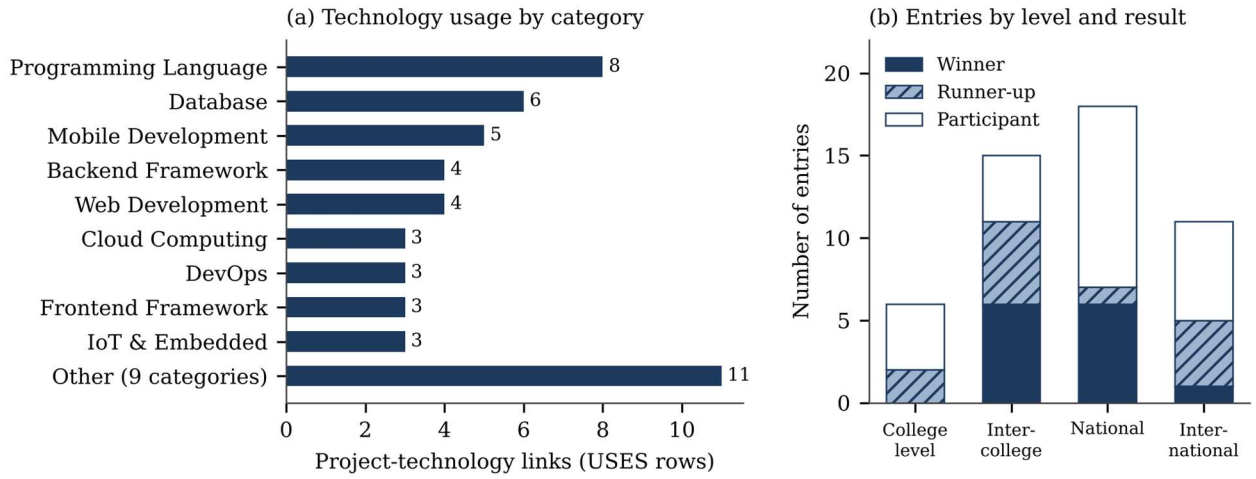


Fig. 3. (a) Technology usage by category and (b) competition entries by level and result, from the sample database.

Project distribution across teams is also simple in the sample: 46 teams work on one project and 4 teams on two. Every team has exactly one member recorded in MEMBER_OF (50 students, 50 teams), so the sample does not exercise multi-member teams.

E. Existing versus proposed system

TABLE IV. QUALITATIVE COMPARISON (NO MEASURED PERFORMANCE DIFFERENCE IS CLAIMED)

Parameter	Traditional approach	Proposed Academic Project Archive System
Data storage	Separate files, sheets or printed reports	One MySQL database with 14 related tables
Organization	Folder and file naming conventions	Entities with keys; domains, technologies and competitions as reference tables
Project retrieval	Manual opening and searching of files	SQL filters, LIKE search and joins across tables (no search interface)
Data consistency	Depends on manual discipline; duplicates likely	Keys and foreign keys enforce links; uniqueness and value checks not yet declared
Relationship management	Links recorded by hand, if at all	Junction tables for team membership, team–project, technology and competition links
Reporting	Compiled manually for each report	Reusable SQL reports and three views
Data maintenance	Edits repeated across files	A fact is updated once; changes made through SQL, as there is no form-based interface

F. Discussion

Observations. The schema loads and the reporting scripts run on the sample dataset. They answer questions that cut across entities (domain, year, status, technology, competition result, outputs) without any change to the schema. Fourteen sections of reports in 04_Project_Retrieval_and_Reports.sql show that routine coordinator questions can be answered with short SQL statements.

Limitations found in the files. (i) There are no UNIQUE constraints, and the sample data contain two students with the same Enroll_No (EN2025019). (ii) There are no CHECK or domain constraints: Status, Result and Output_Type are free text and Rating can take any integer. (iii) There are no ON DELETE or ON UPDATE rules. (iv) Script order matters: 02_sample_data.sql inserts into PROJECT_OUTPUT using the column names Application_or_Paper_ID and Output_Date, while 01_schema.sql creates Identifier and Output_Date and 03_queries.sql renames them afterwards, so running 01, 02, 03 in order fails at that insert. (v) Sample IDs are re-keyed by UPDATE statements on primary keys, which is fragile. (vi) 37 domain rows are unused, and the sample data is small and synthetic. (vii) Reviews are stored with a free-text reviewer name and no link to a user. (viii) No query plans, indexes beyond key indexes, or timing measurements are reported, so no performance claim is made.

Implemented versus expected. What is implemented and demonstrated is the schema, the sample data, the queries and the views. Expected, but not shown, benefits are less duplicated effort by student teams, easier accreditation reporting and use as a base for a web portal; these would need a pilot with real records and user feedback to be evidenced.

V. CONCLUSION AND FUTURE SCOPE

This paper described the design and implementation of the Academic Project Archive System, a 14-table MySQL database that organizes academic projects with their students, teams, guides, domains, technologies, competitions, documents, reviews and outputs. Relational concepts were applied directly: entities became tables, many-to-many relationships became junction tables with composite keys, and foreign keys enforce referential integrity. SQL scripts demonstrate retrieval, filtering, joins, aggregation, nested queries, views and a transactional update. Running them on a 50-project sample produced domain, year, status, technology and competition summaries. The contribution is a documented and reproducible schema that links project records to competition and publication or patent outcomes, which the reviewed literature does not describe. Such a database could serve institutions as a searchable project record and reporting base.

Current limitations are that the sample is small and appears illustrative, the design lacks uniqueness and check constraints, the ER diagram differs from the implemented schema, script ordering must be fixed, and no interface, performance test or user study exists. Planned future work, none of which is

ISSN: 3139-0862

implemented, includes: (i) a web-based interface; (ii) user authentication and role management; (iii) advanced project search, possibly with full-text indexes; (iv) project recommendation; (v) an analytics dashboard; (vi) integration with institutional ERP systems; and (vii) improved document management, including storing or linking real files. Immediately, the schema should be hardened with UNIQUE, CHECK and lookup-table constraints and validated with real, anonymised project records.

REFERENCES

[1] N. A. Ramli, C. F. Hibadullah and A. Alwi, "SCIMPA FYP Management System," Multidisciplinary Applied Research and Innovation, vol. 2, no. 3, pp. 90–94, 2021, doi: 10.30880/mari.2021.02.03.018.

[2] A. Tsolakidis, I. Triantafyllou, E. Triperina and C. Skourlas, "Developing the infrastructure for managing institutional research archives," Journal of Integrated Information Management, vol. 8, no. 1, 2023, doi: 10.26265/jiim.v8i1.4515.

[3] R. Isa, S. Othman, A. S. Ali, N. Azizan and J. Ferguson, "Prototype Development of Final Year Project Management System to Monitor Student's Performance," J. Adv. Res. Appl. Sci. Eng. Technol., vol. 40, no. 1, pp. 164–173, 2024, doi: 10.37934/araset.40.1.164173.

[4] M. A. Salvador, K. A. Botangen, M. C. Rabang, I. C. Salinas, M. Naagas and A. Balagot, "Development of a Web-based Research Consortium Database Management System: Advancing Data-driven and Knowledge-based Project Management," arXiv:2411.00483, Nov. 2024, doi: 10.1145/3670105.3670120.

[5] A. Nolte, I.-A. Chounta and J. D. Herbsleb, "What Happens to All These Hackathon Projects? Identifying Factors to Promote Hackathon Project Continuation," Proc. ACM Hum.-Comput. Interact., vol. 4, no. CSCW2, pp. 1–26, 2020.

[6] A. Imam, T. Dey, A. Nolte, A. Mockus and J. D. Herbsleb, "The Secret Life of Hackathon Code: Where does it come from and where does it go?," arXiv:2103.01145, 2021.

[7] Z. Ratsoga and M. Primus, "Factors influencing post-hackathon project continuation in an African corporate setting," The African Journal of Information and Communication, no. 31, Jun. 2023, doi: 10.23962/ajic.i31.15564.

[8] C. Suryawasnhi, S. Sonwale and P. Deshpande, "Analysis of the Smart India Hackathon (SIH): A Data-Driven Approach," in Proc. 5th Int. Conf. Communication and Information Processing (ICCIP), 2023; SSRN, Sep. 2024, doi: 10.2139/ssrn.4945744.

[9] E. F. Codd, "A relational model of data for large shared data banks," Commun. ACM, vol. 13, no. 6, pp. 377–387, 1970, doi: 10.1145/362384.362685.

[10] P. P.-S. Chen, "The entity-relationship model—toward a unified view of data," ACM Trans. Database Syst., vol. 1, no. 1, pp. 9–36, 1976, doi: 10.1145/320434.320440.

[11] R. Elmasri and S. B. Navathe, Fundamentals of Database Systems, 7th ed. Pearson, 2016.

[12] Oracle Corporation, "MySQL 8.0 Reference Manual," dev.mysql.com/doc/refman/8.0/en/.