AcademiGit: A Git-Inspired Academic Repository System for Version-Controlled Educational Resource Management

Vinuta Belagali[1], Prof. Geetanjali Patrimath[2], Sanjay Lote[3]
[1] *PG Research Scholar, Department of Computer Science and Engineering, Secab Institute of Engineering & Technology, Visvesvaraya Technological University, Belagavi, Karnataka, India*
[2] *Assistant. Professor, Department of Computer Science and Engineering, Secab Institute of Engineering & Technology, Visvesvaraya Technological University, Belagavi, Karnataka, India*
[3] *Lecturer Selection Grade I, Department of computer science and Engineering, Government polytechnic Rabakavi - Banahatti, Karnataka, India*

***Abstract*—The rapid adoption of digital learning environments has significantly increased the demand for efficient academic content management systems. Traditional methods of sharing lecture notes, assignments, and announcements through messaging applications or email often led to version inconsistencies, poor organization, and limited accessibility. This paper presents AcademiGit, a Git-inspired academic repository platform that applies version-control concepts to educational resource management. The proposed system enables faculty members to upload, update, and maintain academic resources while preserving complete version history. Students can securely access the latest study materials, download previous versions when require, and receive institutional announcements through a centralized platform. The system follows the Django Model–View–Template (MVT) architecture with a Python backend, HTML/CSS/JavaScript frontend, and SQLite database. Role-based authentication ensures secure access for faculty and students. Experimental evaluation demonstrates that the proposed platform improves accessibility, resource organization, collaboration, and transparency while reducing manual communication overhead. AcademiGit provides a lightweight, scalable, and cost-effective solution suitable for educational institutions seeking an alternative to conventional Learning Management Systems.**

***Index Terms*—Academic Repository, Version Control, Git, Django, Learning Management System, Educational Technology, Repository Management.**

I. INTRODUCTION

The rapid advancement of Information and Communication Technology (ICT) has significantly transformed the education sector by enabling digital learning and efficient academic resource sharing. Many educational institutions still rely on emails, messaging applications, and cloud storage to distribute lecture notes, assignments, and announcements. These methods often result in scattered resources, duplicate files, outdated study materials, and inefficient communication between faculty and students.

To address these challenges, this paper proposes AcademiGit, a Git-inspired academic repository system that applies version-control concepts to educational resource management. The system provides a centralized platform where faculty members can upload, update, and organize academic materials while maintaining version history. Students can securely access the latest study resources, download previous versions when require, and receive academic announcements through a single web-based interface.

AcademiGit is developed using the Django Model–View–Template (MVT) architecture with Python as the backend programming language, HTML, CSS, and JavaScript for the frontend, and SQLite as the database. The system incorporates role-based authentication to ensure secure access for faculty and students while improving the organization and accessibility of educational resources.

The proposed platform reduces manual effort, minimizes file duplication, and enhances collaboration between faculty and students. Furthermore, its lightweight and scalable architecture makes it suitable for educational institutions seeking a cost-effective academic repository solution. The remaining sections present the related work, proposed methodology, system implementation, experimental results, and future enhancements.

II. LITERATURE SURVEY

[1] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," IEEE Computer, vol. 29, no. 2, pp. 38–47, Feb. 1996.

Sandhu et al. [1] proposed the Role-Based Access Control (RBAC) model to simplify user authorization by assigning permissions to roles instead of individual users. The model improves security, scalability, and administrative efficiency in multi-user systems. RBAC has become one of the most widely adopted access control mechanisms in enterprise and web applications because it enables secure management of users with different responsibilities. The proposed AcademiGit system adopts this concept by defining separate roles for faculty and students, ensuring secure access to notes, announcements, and student marks according to user privileges.

[2] S. Chacon and B. Straub, Pro Git, 2nd ed. New York, NY, USA: Apress, 2014.

Chacon and Straub [2] introduced Git as a distributed version control system that efficiently tracks modifications to digital content while maintaining complete version history. Git enables collaborative development, prevents data loss, and allows users to retrieve previous versions of files whenever required. Although Git was originally designed for software development, its version management principles can be effectively applied to academic document repositories. Inspired by this concept, the proposed AcademiGit system incorporates version control for study materials, allowing faculty members to maintain multiple versions of notes while enabling students to access both current and previous versions.

[3] Django Software Foundation, "Using the Django Authentication System," Django Documentation.

The Django Software Foundation [3] provides a comprehensive authentication and authorization framework for developing secure web applications. The framework includes user authentication, password management, session handling, and permission-based access control, thereby reducing development complexity while improving application security. Django's built-in authentication mechanisms are widely adopted in educational and enterprise applications due to their reliability and scalability. In the proposed AcademiGit system, Django authentication is utilized to implement secure login, institutional register number-based student registration, and role-based authorization for faculty and students.

[4] Thakur, Pooja, and Prashant Jadon. "Django: Developing web using python." 2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE). IEEE, 2023.

Thakur, Pooja, and Prashant Jadon (2023) presented a study titled "Django: Developing Web Using Python", which highlights the effectiveness of the Django framework in developing secure, scalable, and maintainable web applications. The authors discuss Django's Model–View–Template (MVT) architecture, built-in authentication system, Object Relational Mapping (ORM), URL routing, and security features such as protection against SQL injection and Cross-Site Scripting (XSS). The paper emphasizes that Django reduces development time through reusable components and simplifies database management, making it suitable for modern web applications. However, the study focuses on the capabilities of the Django framework rather than a specific academic management solution. Inspired by this work, the proposed AcademiGit system utilizes Django to implement secure user authentication, register number-based student registration, Git-inspired version control for academic notes, student marks management, announcement management, and role-based access control within a centralized academic repository.

[5] Lathkar, Malhar. Modern Django Web Development. 2025.

Lathkar, Malhar (2025), in the book Modern Django Web Development, presents modern techniques and best practices for building secure, scalable, and high-performance web applications using the Django framework. The book explains Django's Model–View–Template (MVT) architecture, Object Relational Mapping (ORM), authentication and authorization mechanisms, REST API development, database integration, security features, and deployment

strategies. It emphasizes writing modular, maintainable, and reusable code while leveraging Django's built-in components to accelerate web application development. The concepts discussed in the book provide a strong foundation for developing enterprise-grade web applications. Inspired by these practices, the proposed AcademiGit system utilizes Django to implement secure register number-based student registration, role-based access control, Git-inspired version control for academic notes, student marks management, announcements, and centralized academic resource management, resulting in a secure, scalable, and efficient academic repository system.

III. RELATED WORK

Several studies have focused on developing Learning Management Systems (LMSs) and academic repository platforms to improve the management and accessibility of educational resources. Modern LMS platforms such as Moodle and Google Classroom provide features including course management, assignment submission, online assessments, and communication between instructors and students. While these systems are comprehensive, they often include complex functionalities that may not be required by small educational institutions and can demand significant administrative effort.

Cloud-based file-sharing platforms, including Google Drive and Microsoft OneDrive, are widely used for storing and distributing academic materials. Although these platforms simplify file sharing, they lack structured academic organization, role-based academic workflows, and integrated version management tailored for educational environments. As a result, students may access outdated files, and faculty members often face challenges in maintaining multiple versions of study materials.

Version Control Systems (VCSs), particularly Git, have revolutionized collaborative software development by providing efficient version tracking, repository management, and change history. Git enables multiple contributors to work simultaneously while preserving previous versions of files. However, traditional Git platforms are designed primarily for software development and are not user-friendly for non-technical academic users.

Recent research has explored web-based academic management systems using Python and the Django framework due to their scalability, security, and rapid development capabilities. Django's Model–View–Template (MVT) architecture supports modular application design, secure authentication, and efficient database integration, making it suitable for educational applications.

The proposed **AcademiGit** system bridges the gap between conventional Learning Management Systems and software version-control platforms by introducing Git-inspired repository concepts into academic resource management. Unlike existing academic portals, AcademiGit combines centralized notes management, role-based authentication, announcement services, and simplified version tracking within a lightweight web application. This approach improves resource organization, minimizes file duplication, and enhances collaboration between faculty members and students while remaining simple enough for deployment in educational institutions with limited infrastructure.

Research Gap: Existing academic platforms either provide extensive LMS functionalities or basic file-sharing services without effective version management. There is a need for a lightweight academic repository that integrates centralized document management, controlled access, and version-aware resource organization. AcademiGit addresses this gap by applying Git-inspired concepts to educational content management through an intuitive and scalable web-based platform.

TABLE I
COMPARISON OF EXISTING SYSTEMS

Feature	Google Classroom	Moodle	GitHub	AcademiGit (Proposed)
Academic Notes Repository	✓	✓	×	✓
Version History	Limited	Limited	✓	✓
Faculty–Student Role Management	✓	✓	×	✓
Announcement Module	✓	✓	×	✓
Easy for Non-Technical Users	✓	Moderate	×	✓
Lightweight Deployment	✓	Moderate	×	✓
Academic Resource Organization	Moderate	High	Low	High

IV. PROPOSED WORK

The proposed AcademiGit system is a web-based academic repository designed to simplify the management and distribution of educational resources

within academic institutions. Inspired by the concepts of Git-based version control, the system provides a centralized platform where faculty members can upload, update, and organize study materials while maintaining a structured repository for each course. Students can securely access the latest learning resources, download previous versions when necessary, and receive academic announcements through a unified interface.

AcademiGit follows the Model–View–Template (MVT) architecture provided by the Django framework. This architecture separates the application into three independent components: the Model, which manages database operations and business logic; the View, which processes user requests and controls application workflow; and the Template, which provides the graphical user interface. Such separation improves modularity, maintainability, and scalability of the application.

The system is developed using Python and the Django framework for backend implementation, while HTML, CSS, and JavaScript are used to design a responsive and user-friendly frontend. SQLite serves as the database management system for storing user credentials, notes, announcements, course details, and profile information. Django's built-in authentication mechanism provides secure login and role-based authorization for faculty and students.

The overall workflow begins with user authentication. After successful login, the system verifies the user's role and redirects them to the corresponding dashboard. Faculty members are provided with facilities to upload notes, edit existing resources, organize files according to departments and courses, and publish announcements. Students can browse available study materials, download lecture notes, view announcements, and manage their personal profiles. All operations are validated before being stored in the centralized database.

The Notes Management Module functions as the primary repository of academic resources. Every uploaded document is stored with metadata including course, department, faculty information, upload date, and version details. The Announcement Module allows faculty to communicate important academic information to students without relying on third-party messaging applications. The Profile Management Module enables users to update personal information while maintaining secure access credentials.

The proposed methodology improves academic content organization by reducing duplicate files, simplifying document retrieval, and ensuring that users access the latest available study materials. The modular architecture also enables future enhancements such as cloud deployment, mobile application support, notification services, assignment submission, and analytics without major modifications to the existing system.

V. SYSTEM ARCHITECTURE

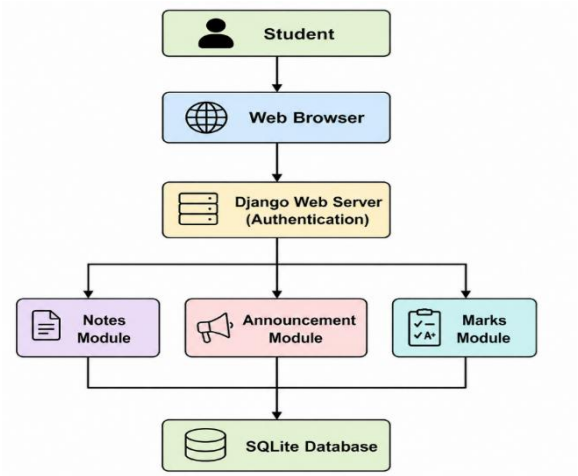

Fig. 1. Overall Architecture of AcademiGit

Figure 1 illustrates the overall architecture of the proposed AcademiGit system. Users access the application through a standard web browser. Authentication requests are processed by the Django web server, which verifies user credentials before granting access. Based on user roles, requests are directed to the Notes Module, Announcement Module, or Profile Module. These modules communicate with the SQLite database to retrieve and update academic information. This layered architecture ensures secure communication, modular implementation, and efficient management of academic resources.

VI. WORKING PROCEDURE

The operational workflow of AcademiGit consists of the following sequential steps:

1.User Registration and Login.

2.Authentication using Django authentication services.

3.Role verification (Faculty or Student).

4.Faculty upload notes and publishes announcements.

5.Student views and downloads study materials.
6.Database stores and retrieves academic resources securely.
7.Updated information is displayed through the user dashboard.
The proposed workflow ensures secure access, centralized academic resource management, and efficient communication between faculty members and students.

VII. SYSTEM IMPLEMENTATION AND RESULTS

7.1 System Implementation

The proposed AcademiGit system was implemented as a web-based application using the Django framework and the Python programming language. Django's Model–View–Template (MVT) architecture was adopted to separate business logic, user interface, and database operations, thereby improving maintainability and scalability. The frontend was developed using HTML, CSS, and JavaScript, providing a responsive and user-friendly interface. SQLite was employed as the backend database for storing user profiles, course information, academic notes, and announcements.

The application supports role-based authentication, allowing users to register and log in securely as either Faculty or Student. Django's built-in authentication mechanism validates user credentials and redirects users to their respective dashboards. Faculty members are authorized to upload study materials, edit or remove existing resources, and publish announcements, while students can browse, view, and download notes according to their department and course.

The Notes Management Module maintains a structured repository of academic resources. Each uploaded file is stored together with metadata such as title, department, course, faculty information, upload timestamp, and file location. This organized structure enables efficient retrieval and reduces duplication of learning materials. The Announcement Module provides a centralized communication channel through which faculty can share academic updates, examination schedules, and important notices. The Profile Management Module allows users to update personal information while ensuring secure access to the system.

The developed system was executed successfully in a local environment using the Django development server. The implementation demonstrates that the proposed architecture supports efficient academic resource management with minimal hardware and software requirements.

7.2. Experimental Results

The AcademiGit system was evaluated through functional testing of all major modules. The evaluation focused on verifying authentication, notes management, announcement services, and dashboard operations. Each module was tested independently before integration testing was conducted to ensure seamless interaction among system components.

The login module correctly authenticated authorized users and denied access to invalid credentials. Faculty members successfully uploaded lecture notes and announcements, which were immediately reflected in the student dashboard. Students were able to browse available resources, preview documents, and download study materials without errors. Profile updates and dashboard navigation also functioned as expected.

The experimental results indicate that the system provides reliable performance for academic environments. The centralized repository simplifies document management, minimizes redundant files, and enables efficient retrieval of educational resources. Compared with conventional file-sharing methods, AcademiGit significantly improves organization and accessibility of academic content.

TABLE III
PERFORMANCE COMPARISON

Parameter	Conventional Method	AcademiGit
Notes Availability	Distributed	Centralized
Version Tracking	No	Yes
File Duplication	High	Low
Resource Retrieval Time	Moderate	Fast
Faculty–Student Communication	Separate Platforms	Integrated
User Authentication	Basic	Role-Based
Repository Management	No	Yes

Fig 1: Home Page

The AcademiGit Home Page, which serves as the landing interface of the proposed system. The homepage provides a simple and user-friendly design with separate navigation options for Faculty and Students. Faculty members can access the system to upload and manage academic resources, while students can securely access and download study materials. This interface acts as the entry point to the platform, enabling users to navigate to their respective modules based on their roles.

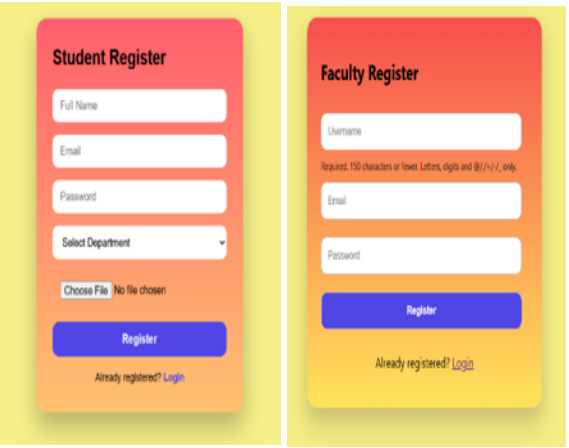

Fig 2 & 3: Student & Faculty Registration

The Student Registration and Faculty Registration interfaces of the proposed AcademiGit system. The registration forms allow users to create accounts by providing the required information, such as personal details, email address, password, and department. The system validates the entered information before completing the registration process, ensuring that only authorized users can access the platform. These interfaces provide a secure and user-friendly mechanism for enrolling faculty members and students into the AcademiGit system.

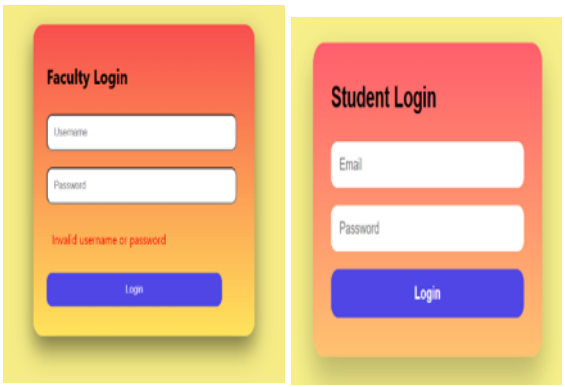

Fig 4 & 5: Faculty & Student Login

The Faculty Login and Student Login interfaces of the proposed AcademiGit system. These login pages enable registered users to securely access the application using their valid credentials. The system authenticates the entered username/email and password through Django's authentication mechanism before granting access to the respective dashboard. Invalid login attempts are rejected with appropriate error messages, ensuring secure and role-based access to academic resources.

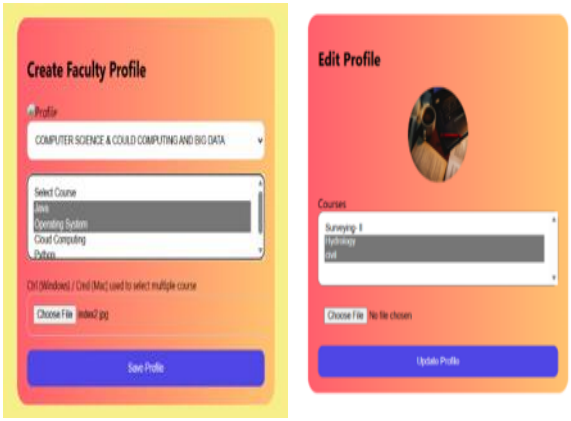

Fig 6 & 7: Faculty Profile Management

The Faculty Profile Management module of the proposed AcademiGit system. The interface enables faculty members to create and update their profiles by selecting the courses they teach, uploading a profile image, and modifying personal information. This module ensures that faculty profiles remain accurate and up to date, allowing efficient management of course-related responsibilities. It also provides a personalized user experience while maintaining secure storage of profile information within the system.

Fig 8: Faculty Dashboard

The Faculty Dashboard of the proposed AcademiGit system. The dashboard displays the faculty profile along with the courses handled and provides options to add new notes, edit the profile, and log out. It also lists the uploaded study materials with actions to preview, download, edit, or delete the notes. This interface enables faculty members to efficiently manage academic resources and maintain updated course materials through a centralized platform.

Fig 9 & 10: Add Notes & Edit Notes

The Add Notes and Edit Notes interfaces of the proposed AcademiGit system. The Add Notes page enables faculty members to upload study materials by selecting the department, course, title, and document file. The Edit Notes page allows faculty to modify the note title or replace the existing file while preserving the document's version history. These interfaces simplify the management of academic resources and ensure that students always have access to the most recent learning materials.

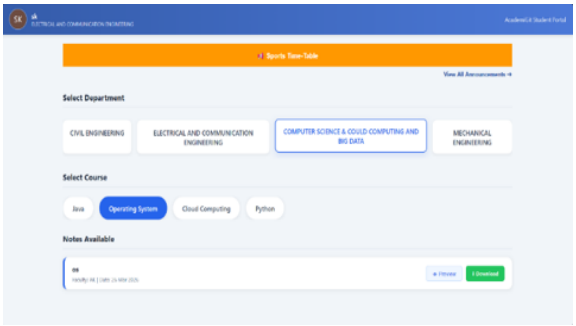

Fig 11: Student Dashboard

The Student Dashboard of the proposed AcademiGit system. The dashboard enables students to select their department and course to access relevant study materials uploaded by faculty members. It also displays important academic announcements and provides options to preview or download the latest course notes. This interface offers a centralized, user-friendly platform for accessing academic resources and staying updated with course-related information.

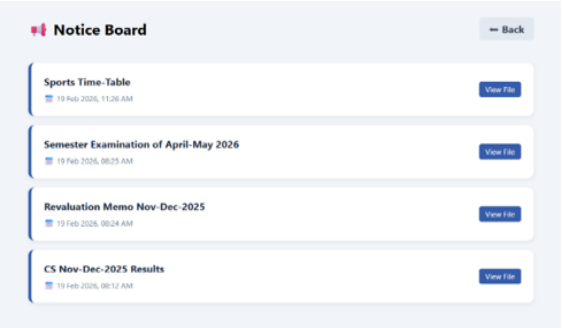

Fig 12: Notice Board

The Notice Board module of the proposed AcademiGit system. This interface displays important academic announcements such as examination schedules, sports timetables, revaluation notifications, and examination results published by faculty or administrators. Each notice includes the publication date and a View File option, allowing students to access detailed information. The centralized notice board ensures timely communication and keeps students informed about institutional updates through a single platform.

Fig 13: Add CIE Marks

The Add CIE Marks interface of the proposed AcademiGit system. This module enables faculty members to enter students' Continuous Internal Evaluation (CIE) marks by specifying the student register number, course, examination type, marks obtained, and maximum marks. It also provides an option to upload a remedial assignment file for

students with low scores. The entered marks are securely stored in the database, allowing students to view their academic performance through their personalized dashboard.

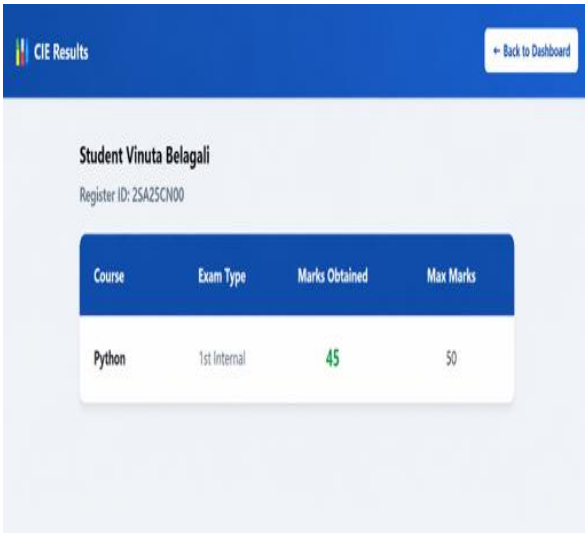

Fig 14: CIE Results

The CIE Results interface of the proposed AcademiGit system. This module enables students to view their Continuous Internal Evaluation (CIE) marks after they have been uploaded by the faculty. The interface displays essential information such as the course name, examination type, marks obtained, and maximum marks in a structured tabular format. This feature improves transparency by allowing students to securely monitor their academic performance through their personalized dashboard.

VIII. METHODOLOGY

The proposed AcademiGit system follows a modular web-based development approach using the Django framework and SQLite database. Initially, users register through their institutional register number and are authenticated using Django's built-in authentication system. Based on their role (Faculty or Student), users are granted appropriate access through Role-Based Access Control (RBAC). Faculty members can upload and manage study materials with Git-inspired version control, publish announcements, and update student marks, while students can securely access notes, view previous document versions, check academic marks, and receive announcements. All academic information is stored in a centralized database, ensuring secure data management, efficient retrieval, and improved collaboration between faculty and students.

8.1 Number of Modules

The proposed AcademiGit system consists of the following six major modules:

8.1.1 Authentication and Registration Module

This module enables secure user registration and login. Students are required to register using their institutional Register Number, ensuring that only authorized users can access the system. It also authenticates users and provides secure access based on their credentials.

8.1.2 Notes Management Module

This module allows faculty members to upload, update, organize, and manage academic notes. It supports Git-inspired version control, enabling multiple versions of study materials to be maintained. Students can view and download both the latest and previous versions of notes.

8.1.3 Student Marks Management Module

This module enables faculty members to upload, edit, and manage students' internal assessment and semester examination marks. Students can securely access their subject-wise academic performance through their personalized dashboard after successful authentication.

8.1.4 Announcement Management Module

This module allows faculty members to publish important academic announcements, notifications, and event updates. Students can access the latest announcements through the system, ensuring timely communication between faculty and students.

8.1.5 User Profile Management Module

This module maintains user information, including personal details, department, semester, and role. Users can view and update their profile information while ensuring that their academic records remain secure.

8.1.6 Database Management Module

This module is responsible for storing and managing all system data, including user accounts, institutional register numbers, notes, version history, student marks, announcements, and profile information. It ensures secure data storage, efficient retrieval, and maintains the integrity and consistency of academic records.

8.2 USE-CASE DIAGRAM:

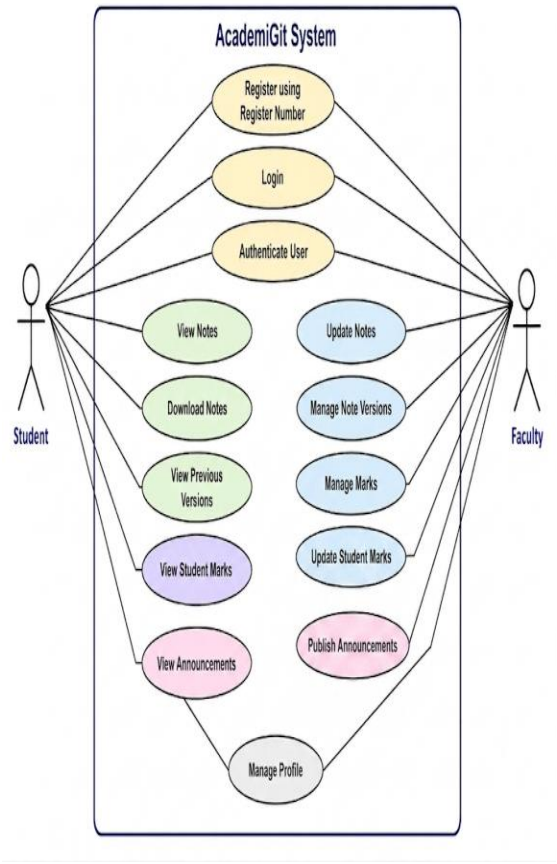

Fig : Use-Case Diagram

8.3 WORKFLOW DIAGRAM:

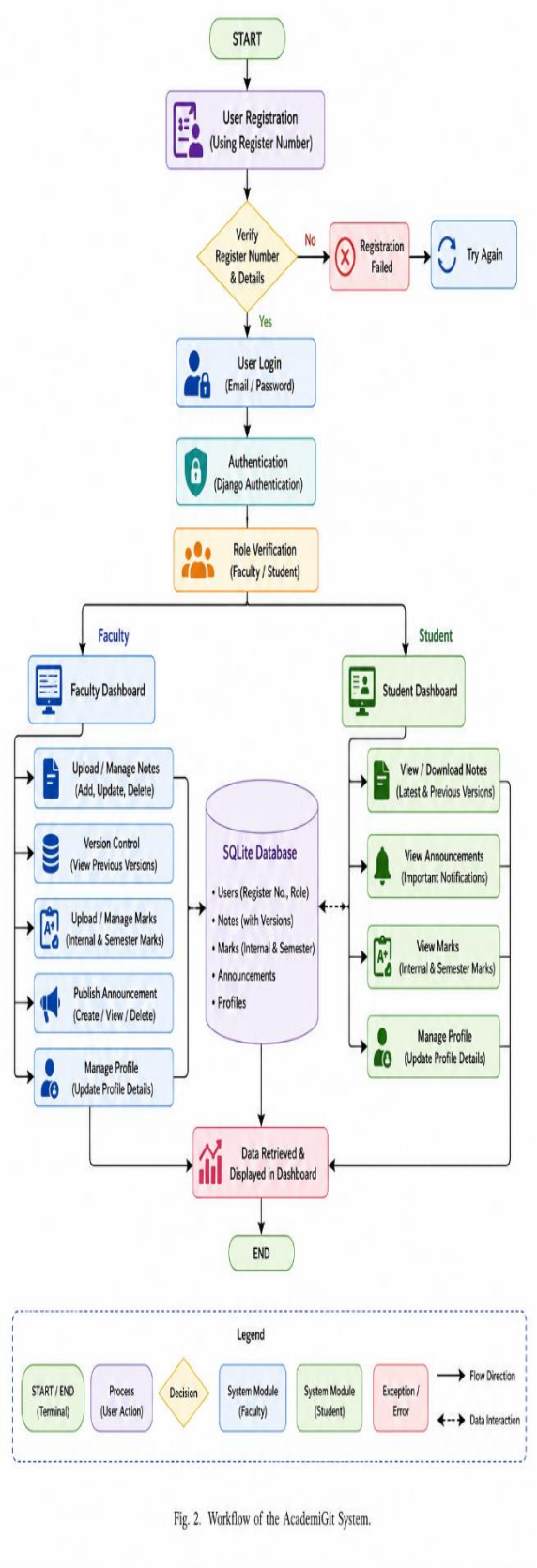

Fig. 2. Workflow of the AcademiGit System.

The workflow of the proposed AcademiGit system begins with student registration, where users create an account using their institutional Register Number. The system verifies the register number and user details before allowing registration. If the verification fails, the registration request is rejected, and the user is prompted to try again. After successful registration, users log in using their email and password. Django's built-in authentication mechanism validates the user credentials, and the system performs Role-Based Access Control (RBAC) to identify whether the user is a Faculty member or a Student. Based on the authenticated role, the user is redirected to the appropriate dashboard, ensuring secure and authorized access to system resources.

Once authenticated, faculty members can access the Faculty Dashboard to upload and manage study materials, maintain Git-inspired version control for notes, upload and update students' internal and semester examination marks, publish academic announcements, and manage their profiles. Students, through the Student Dashboard, can view and download the latest and previous versions of notes, access announcements, monitor their academic marks, and update their profile information. All academic data, including user accounts, register numbers, notes, version history, marks, announcements, and profile details, are stored securely in the SQLite database. The database processes all requests and returns the required information to the respective dashboards, ensuring secure data management, efficient retrieval, improved collaboration between faculty and students, and centralized academic information management.

VII. CONCLUSION AND FUTURE

7.1Conclusion

This paper presented AcademiGit, a Git-inspired academic repository system that facilitates efficient management and distribution of educational resources in academic institutions. The proposed system introduces version-control concepts into academic content management by providing centralized note repositories, structured version history, role-based access control, and integrated announcements.

The system was successfully implemented using the Django framework and tested for all major modules. Experimental results demonstrate that AcademiGit improves resource organization, reduces file duplication, and enhances accessibility of study materials. Faculty members can upload, update, and manage academic resources efficiently, while students can easily access the latest materials and previous versions when required. The platform also strengthens communication between faculty and students through the announcement module.

Overall, AcademiGit offers a lightweight, secure, and scalable solution that addresses the limitations of conventional file-sharing methods and complex Learning Management Systems (LMSs). The system is particularly suitable for educational institutions seeking a simple yet effective repository for academic content management.

7.2Future Work

The following enhancements can be incorporated into AcademiGit in future versions:

- Integration with cloud storage services for scalable and reliable file management.
- Development of a mobile application for improved accessibility.
- Addition of assignment submission and evaluation modules.
- Implementation of real-time notifications and email alerts.
- Analytics and reporting dashboard for tracking academic activities.
- Support for collaborative document editing and commenting.

REFERENCES

[1] R. M. Shafiq, N. Jamil, and M. Azeem, “A systematic review of learning management systems in higher education: Features, trends and challenges,” Education and Information Technologies, vol. 27, no. 2, pp. 2167–2192, Feb. 2022.

[2] A. Parmar and D. Patel, “Design and development of academic repository system for educational institutions,” in Proc. Int. Conf. Computer, Communication and Informatics (ICCCI), Greater Noida, India, Jan. 2021, pp. 1–6.

[3] S. M. Wani and N. A. Rather, “Cloud-based e-learning: A review,” International Journal of Engineering Research & Technology (IJERT), vol. 10, no. 5, pp. 41–46, May 2021.

[4] A. Kumar, S. Gupta, and P. Sharma, “Web-based academic portal for resource management and communication,” in Proc. Int. Conf. Smart Technologies in Computing, Electrical and Electronics (ICSTCEE), Bangalore, India, Aug. 2022, pp. 523–528.

[5] M. N. Huda and M. F. Abdullah, “A review on Git version control system and its applications,” Journal of Computing Research and Innovation, vol. 6, no. 2, pp. 89–97, Jun. 2021.

[6] P. López and J. A. Gómez, “Applying Git concepts to educational content management,” in Proc. IEEE Global Engineering Education Conference (EDUCON), Vienna, Austria, Apr. 2022, pp. 1–6.

[7] D. S. Bhadoriya and S. K. Soni, “Role-based access control in web applications: A survey,” International Journal of Computer Applications, vol. 183, no. 45, pp. 20–25, Jul. 2021.

[8] S. K. Panda and R. K. Jena, “Academic information management system using Django framework,” International Research Journal of Engineering and Technology (IRJET), vol. 9, no. 6, pp. 2412–2417, Jun. 2022.

[9] M. Khalid, I. Yaqoob, and S. A. Madani, “Secure authentication techniques in web applications: A review,” IEEE Access, vol. 9, pp. 114965–114986, Aug. 2021.

[10] V. Gupta and A. K. Sharma, “Version control systems: A review and comparison,” International Journal of Advanced Computer Science and Applications, vol. 12, no. 3, pp. 330–338, Mar. 2021.

[11] J. Singh and P. K. Singh, “Enhancing academic communication using web-based announcement systems,” in Proc. Int. Conf. Advances in Computing (ICAC), New Delhi, India, Dec. 2023, pp. 1–5.

[12] Y. Li, Z. Zheng, and X. Zhang, “Integration of version control concepts in educational platforms: A systematic study,” IEEE Transactions on Learning Technologies, vol. 16, no. 1, pp. 102–115, Jan.–Mar. 2023.