International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Student Leave Application Portal

Mrs. N. Gayathri [1], Jaya Sree J [2], Manimaran M [2], Mithrajith K S [2], Nisanth V [2], Pandiyan S [2]

[1]Department of Artificial Intelligence and Machine Learning, Sri Shakthi Institute of Engineering and Technology, Coimbatore – 641062, India
[2]Bachelor of Technology, Artificial Intelligence and Machine Learning, Second Year, Sri Shakthi Institute of Engineering and Technology, Coimbatore – 641062, India

A B S T R A C T

This paper presents the design and implementation of SLAP – Student Leave Application Portal, a full-stack web application built using Python (Flask) for managing student leave workflows in educational institutions. The system enables students to submit leave requests online, which are reviewed by staff and administrators through a multi-tiered approval process. SLAP incorporates secure authentication mechanisms, session management, PDF generation with institutional seals, and dynamic QR code embedding to ensure document integrity. It is scalable, role-based, and designed with modular REST APIs. The application addresses existing paper-based inefficiencies and demonstrates practical use of Flask, Report Lab, SQLite3, and PIL.

Keywords: Leave Management, Flask, Web Application, Authentication, PDF Generation, Role-based Access, QR Code, SQLite3

1. Main text

The Student Leave Application Portal (SLAP) is designed to digitize and streamline the leave approval process in educational institutions. Traditionally, student leave requests involve physical forms and manual approvals, which are time-consuming, prone to loss, and lack digital verification. SLAP replaces this with a secure, modular, and maintainable full-stack solution.

Nomenclature

Symbol	Definition
A	Approval status (Approved/Rejected)
B	Leave balance of student
C	Seal image generated with institution details
D	PDF document with watermark and QR code

1.1 Structure

SLAP is built with modular design principles:

- Frontend: HTML, CSS, JavaScript

- Backend: Python (Flask)

- Database: SQLite3

- Libraries: ReportLab for PDF, PIL for seal image, Flask-Session for session handling

- Authentication: Salted password hashing using PBKDF2-HMAC-SHA256

- Security: Role-based access control with decorators, secure file uploads with Werkzeug

SLAP follows RESTful practices. Its routing includes endpoints for logging in, submitting leave, approving/rejecting applications, and downloading documents.

Table 1 - Key API Endpoints in SLAP

Functionality	**Endpoint**	**Method**
User login	/api/login	POST
Leave submission	/api/submit_leave	POST
Leave approval/rejection	/api/update_leave_status	POST
Fetch applications	/api/get_leave_applications	GET

1.2 Construction of references

The paper uses references to standard Flask documentation and Python cryptography libraries, with academic sources on secure web design and full-stack development best practices.

1.3 Section headings

The paper is divided into structured sections: Introduction, System Design, Implementation, Results, and Conclusion. Each part explains a component of SLAP in detail.

1.4 General guidelines for the preparation of your text

Symbols used in equations are clearly defined. Variables such as L (leave days), U (user role), and P (PDF document path) are used consistently. The system uses SI units wherever applicable in timestamps and data size (KB/MB).

1.5 File naming and delivery

Files are structured within a Flask project format:

- app.py – Main application
- /static – CSS, JS, uploads
- /templates – HTML templates
- /seal – Seal generation scripts
- utils.py – Helper functions (hashing, QR code, etc.)

1.6 Footnotes

All explanatory notes about security methods, such as PBKDF2 iteration count and salt usage, are discussed inline to maintain clarity. Footnotes are minimized for readability.

2. Illustrations

Fig. 1 - Workflow of SLAP from student submission to admin approval.

Fig. 1 - (a) first picture;

3. Equations

A sample formula for generating the hashed password is:

$$\text{Hashed}=\text{PBKDF2-HMAC}(\text{password},\text{salt},\text{iterations}=100000,\text{sha256})$$

4. Online license transfer

The authors agree to license the work for publication under Procedia's standard agreement. All code is developed originally by the author and open-sourced on request.

Acknowledgements

The author sincerely thanks the faculty of Sri Shakthi Institute of Engineering and Technology for their guidance, and the student testers who provided feedback during the deployment of SLAP.

An example appendix

A.1. Leave Submission Example:

```
{
  "reg_no": "21AIML001",
  "name": "John Doe",
  "dept": "AIML",
  "leave_date": "2025-03-15",
  "reason": "Medical Leave"
}
```

References

Van der Geer, J., Hanraads, J. A. J., & Lupton, R. A. (2000). The art of writing a scientific article. *Journal of Science Communication*, 163, 51–59.

Mettam, G. R., & Adams, L. B. (1999). How to prepare an electronic version of your article. In B. S. Jones & R. Z. Smith (Eds.), *Introduction to the electronic age* (pp. 281–304). New York: E-Publishing Inc.

Flask Documentation: https://flask.palletsprojects.com/

Python ReportLab User Guide: https://www.reportlab.com/docs/reportlab-userguide.pdf

OWASP Foundation. (2022). Web Application Security Cheat Sheet.