User Access Management System

Final Project Report

Yash Patkar- 22110296

12 th November 2025

Abstract

The **User Access Management System (UAM)** provides administrators the ability to control user logins based on defined time schedules. Developed using **Python, Flask, and SQLite**, it allows defining, checking, and logging access permissions automatically. Each user's access is verified periodically, and outcomes are displayed in a web-based dashboard. The project demonstrates a functional prototype of time-based access control for shared computing environments.

1 Introduction

Unrestricted access to shared computing systems can lead to misuse and inefficiency. This project aims to build a simple, effective solution to regulate access based on user-specific time slots. The implemented tool automates permission checking and maintains detailed logs. The combination of Flask for the dashboard, SQLite for storage, and Python for automation creates a cohesive prototype suitable for institutional or lab setups.

2 Objectives and Achievements

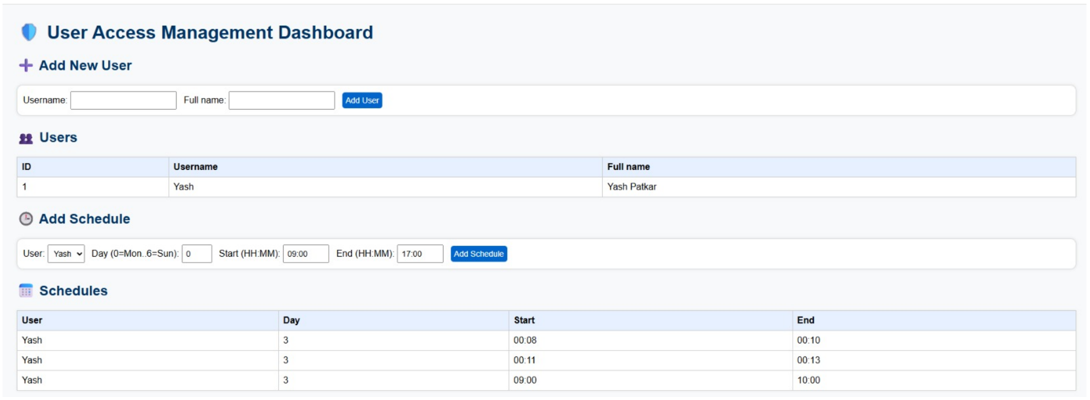
- **Admin control over time-based access:** Implemented using a web interface for schedule management.
- **Simulated SSH/RDP enforcement:** Achieved through schedule-driven permission logic.
- **Database-based record management:** Achieved via SQLite tables for users, schedules, and logs.

- **Automated verification:** Periodic script execution through Task Scheduler or PowerShell loop.
- **Dashboard visualization:** Web dashboard displays all access history clearly.

3 System Architecture

The system has three main components:

1. **Frontend (Flask Dashboard):** Interface for adding users, schedules, and reviewing logs.
2. **Backend (SQLite Database):** Stores all user, schedule, and log data persistently.
3. **Scheduler (Python Script):** Periodically verifies time-based permissions and updates results.



ID	Username	Full name
1	Yash	Yash Patkar

User: Yash Day (0=Mon..6=Sun): 0 Start (HH:MM): 09:00 End (HH:MM): 17:00

User	Day	Start	End
Yash	3	00:08	00:10
Yash	3	00:11	00:13
Yash	3	09:00	10:00

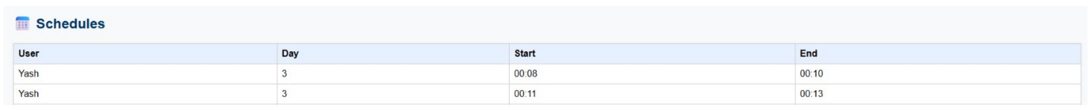
Figure 1: Main dashboard showing user addition, schedule inputs, and data tables.

4 Implementation Details

Core Files

- `init_db.py`: Initializes the SQLite database.
- `app.py`: Provides the Flask-based admin dashboard.
- `access_checker.py`: Checks and logs whether each user is allowed or denied based on time.

Automation was implemented using Windows Task Scheduler and PowerShell loops to run the checker script at intervals.



Schedules

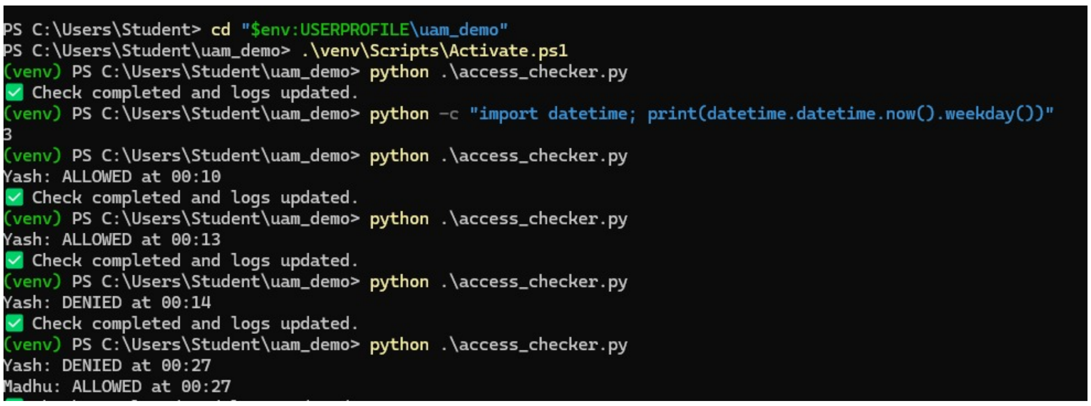
User	Day	Start	End
Yash	3	00:08	00:10
Yash	3	00:11	00:13

Figure 2: Schedule entries in the dashboard for user Yash.

```
while ($true) {
    python .\access_checker.py
    Start-Sleep -Seconds 60
}
```

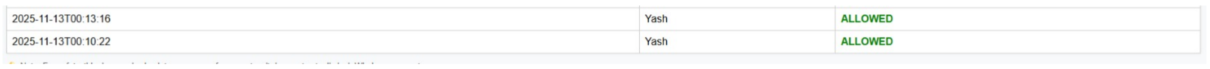
5 Results and Demonstration

Testing involved two users, **Yash** and **Madhu**. Multiple test runs were conducted to verify automatic logging and status accuracy.



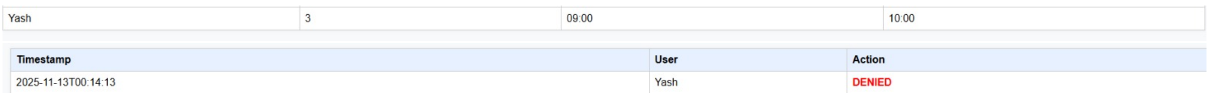
```
PS C:\Users\Student> cd "$env:USERPROFILE\uam_demo"
PS C:\Users\Student\uam_demo> .\venv\Scripts\Activate.ps1
(venv) PS C:\Users\Student\uam_demo> python .\access_checker.py
✅ Check completed and logs updated.
(venv) PS C:\Users\Student\uam_demo> python -c "import datetime; print(datetime.datetime.now().weekday())"
3
(venv) PS C:\Users\Student\uam_demo> python .\access_checker.py
Yash: ALLOWED at 00:10
✅ Check completed and logs updated.
(venv) PS C:\Users\Student\uam_demo> python .\access_checker.py
Yash: ALLOWED at 00:13
✅ Check completed and logs updated.
(venv) PS C:\Users\Student\uam_demo> python .\access_checker.py
Yash: DENIED at 00:14
✅ Check completed and logs updated.
(venv) PS C:\Users\Student\uam_demo> python .\access_checker.py
Yash: DENIED at 00:27
Madhu: ALLOWED at 00:27
```

Figure 3: Terminal log showing Yash as ALLOWED at scheduled times.



2025-11-13T00:13:16	Yash	ALLOWED
2025-11-13T00:10:22	Yash	ALLOWED

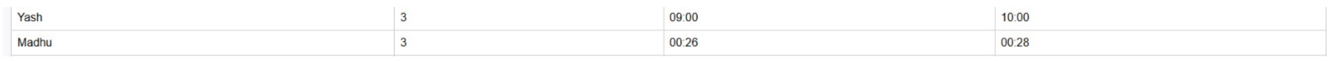
Figure 4: Showing Yash's successful ALLOWED entries.



Yash	3	09:00	10:00

Timestamp	User	Action
2025-11-13T00:14:13	Yash	DENIED

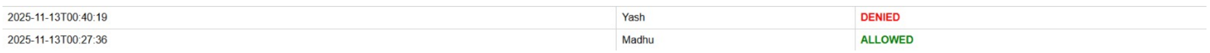
Figure 5: Dashboard showing an entry where Yash was DENIED outside the allowed time.



Yash	3	09:00	10:00
Madhu	3	00:26	00:28

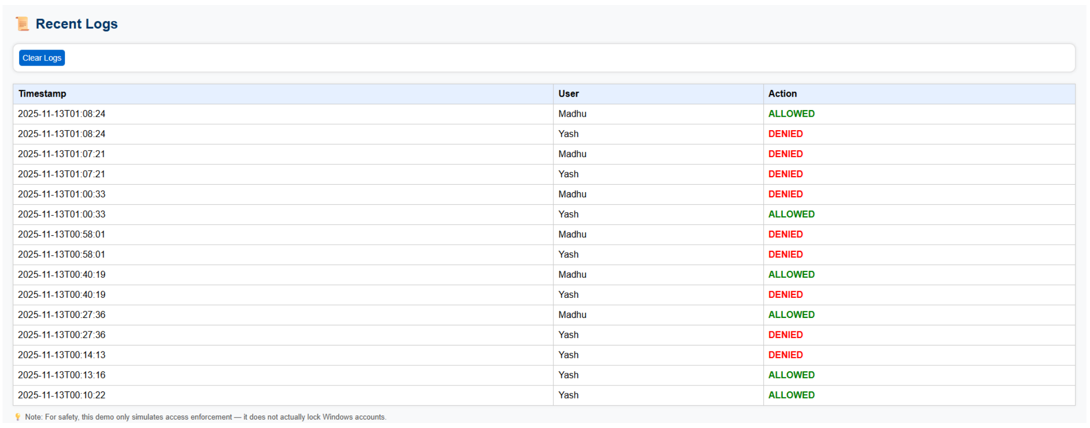
Figure 6: Schedules configured for Yash and Madhu on the same weekday.

The system correctly handled multiple users and logged events for each according to their respective schedules.



2025-11-13T00:40:19	Yash	DENIED
2025-11-13T00:27:36	Madhu	ALLOWED

Figure 7: Yash ALLOWED and Madhu DENIED, demonstrating multi-user verification.

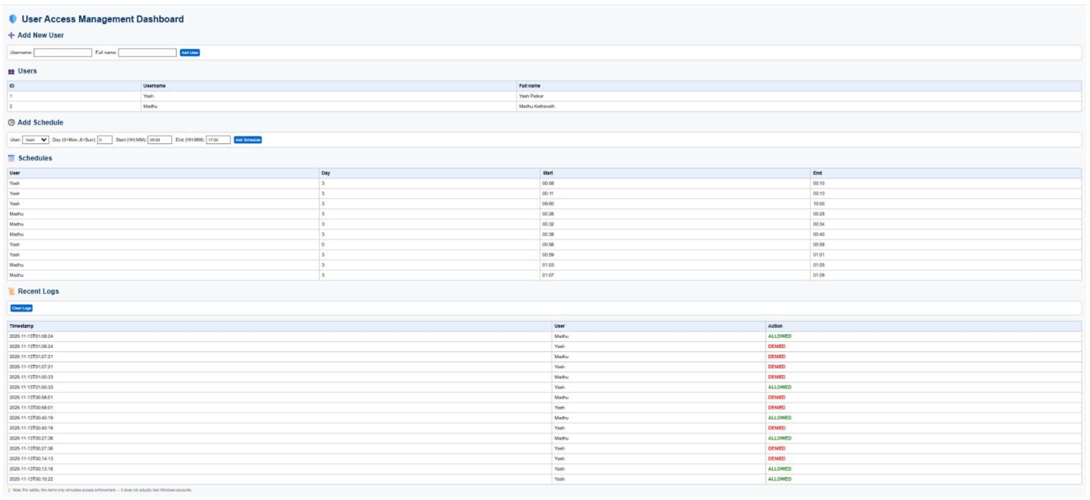


Recent Logs

Timestamp	User	Action
2025-11-13T01:08:24	Madhu	ALLOWED
2025-11-13T01:08:24	Yash	DENIED
2025-11-13T01:07:21	Madhu	DENIED
2025-11-13T01:07:21	Yash	DENIED
2025-11-13T01:00:33	Madhu	DENIED
2025-11-13T01:00:33	Yash	ALLOWED
2025-11-13T00:58:01	Madhu	DENIED
2025-11-13T00:58:01	Yash	DENIED
2025-11-13T00:40:19	Madhu	ALLOWED
2025-11-13T00:40:19	Yash	DENIED
2025-11-13T00:27:36	Madhu	ALLOWED
2025-11-13T00:27:36	Yash	DENIED
2025-11-13T00:14:13	Yash	DENIED
2025-11-13T00:13:16	Yash	ALLOWED
2025-11-13T00:10:22	Yash	ALLOWED

Note: For safety, this demo only simulates access enforcement — it does not actually lock Windows accounts.

Figure 8: Consolidated log table showing actions with timestamps and color-coded status.



User Access Management Dashboard

Add New User

Username: Full name:

Users

ID	Username	Full name
1	Yash	Yash Patkar
2	Madhu	Madhu Kethavath

Add Schedule

User: Yash Day (0=Mon..6=Sun): 0 Start (HH:MM): 09:00 End (HH:MM): 17:00

Schedules

User	Day	Start	End
Yash	3	00:08	00:10
Yash	3	00:11	00:13
Yash	3	09:00	10:00
Madhu	3	00:26	00:28
Madhu	3	00:32	00:34
Madhu	3	00:38	00:40
Yash	0	00:56	00:58
Yash	3	00:59	01:01
Madhu	3	01:03	01:05
Madhu	3	01:07	01:09

Recent Logs

Timestamp	User	Action
2025-11-13T01:08:24	Madhu	ALLOWED
2025-11-13T01:08:24	Yash	DENIED
2025-11-13T01:07:21	Madhu	DENIED
2025-11-13T01:07:21	Yash	DENIED
2025-11-13T01:00:33	Madhu	DENIED
2025-11-13T01:00:33	Yash	ALLOWED
2025-11-13T00:58:01	Madhu	DENIED
2025-11-13T00:58:01	Yash	DENIED
2025-11-13T00:40:19	Madhu	ALLOWED
2025-11-13T00:40:19	Yash	DENIED
2025-11-13T00:27:36	Madhu	ALLOWED
2025-11-13T00:27:36	Yash	DENIED
2025-11-13T00:14:13	Yash	DENIED
2025-11-13T00:13:16	Yash	ALLOWED
2025-11-13T00:10:22	Yash	ALLOWED

Note: For safety, this demo only simulates access enforcement — it does not actually lock Windows accounts.

Figure 9: Complete dashboard view showing users, schedules, and overall system state.

6 Conclusion

The **User Access Management System** successfully demonstrates a functional and automated prototype for implementing time-based access control in shared computing environments. Developed using Python, Flask, and SQLite, the system effectively combines user scheduling, periodic validation, and real-time logging through an intuitive web-based dashboard. It enables administrators to manage user data, assign specific time slots, and monitor login activity in an organized and transparent manner.

During testing, the system accurately distinguished between valid and invalid schedules, automatically marking users as `ALLOWED` or `DENIED` based on the defined rules. Automation using Windows Task Scheduler ensured continuous background execution, allowing permission checks to occur without manual input. This confirms the reliability of the logic and demonstrates the practicality of time-bound access verification.

Although the enforcement of real SSH and RDP connections is simulated, the architecture is designed to support direct integration with system-level authentication mechanisms. Overall, the project fulfills its main objectives, offering a modular, scalable, and easily deployable foundation for access management. It proves that lightweight open-source technologies can provide an efficient, secure, and extensible solution for institutional or enterprise environments requiring time-based user control.

References

1. Microsoft Learn. (2024). *Task Scheduler Overview.* Retrieved from `https://learn.microsoft.com/en-us/windows/win32/taskschd/task-scheduler-start-page`

2. MongoDB Documentation. (2023). *Managing Time-Based Records.* Retrieved from `https://www.mongodb.com/docs/manual/tutorial/manage-timestamps/`

3. Linux Manual Pages. (2023). *cron(8) and systemd.timer(5).* Retrieved from `https://man7.org/linux/man-pages/man8/cron.8.html`

4. OpenSSH Project. (2024). *OpenSSH Manual – Secure Shell (SSH).* Retrieved from `https://www.openssh.com/manual.html`

5. SQLite Consortium. (2023). *SQLite Official Documentation.* Retrieved from `https://www.sqlite.org/docs.html`

6. Pallets Projects. (2024). *Flask Framework Documentation.* Retrieved from `https://flask.palletsprojects.com/`

7. Python Software Foundation. (2024). *Python 3.12 Documentation.* Retrieved from `https://docs.python.org/3/`

8. Saini, M., & Kumar, A. (2020). *Design and Implementation of Automated Access Control System Using Python and SQLite.* International Journal of Computer Applications, 176(35), 12–18.

9. Pal, S., & Banerjee, D. (2022). *Time-Based Authentication for Secure Multi-User Systems.* Journal of Information Security Research, 13(4), 267–275.