USER ACCESS MANAGEMENT SYSTEM

Group 25

INTRODUCTION

- The User Access Management System (UAM) is a time-based access control tool that manages user logins on shared computers.
- It allows administrators to assign specific time slots to users and automatically checks whether access is permitted or denied.
- Developed using Python, Flask, and SQLite, it combines a simple web dashboard with an automated background script for real-time access verification and logging.

OBJECTIVE

1. Build a tool that lets admins control user logins based on time slots.
2. Allow access through local or remote methods (SSH/RDP).
3. Store user data and schedules securely in a database.
4. Automate login permission checks using Task Scheduler.
5. Display access history and login attempts on a dashboard.
6. Improve overall security and management of shared devices.

SYSTEM OVERVIEW

Frontend (Flask Dashboard):

- Admin adds users, defines schedules, and monitors logs.

Backend (SQLite Database):

- Stores user credentials, time slots, and logs persistently.

Automation Script (access_checker.py):

- Checks access permission every minute and updates logs.

IMPLEMENTATION

01

Database Setup:

- Creates the SQLite database and defines tables for users, schedules, and access logs.
- Ensures proper structure for storing user data and tracking access history.

02

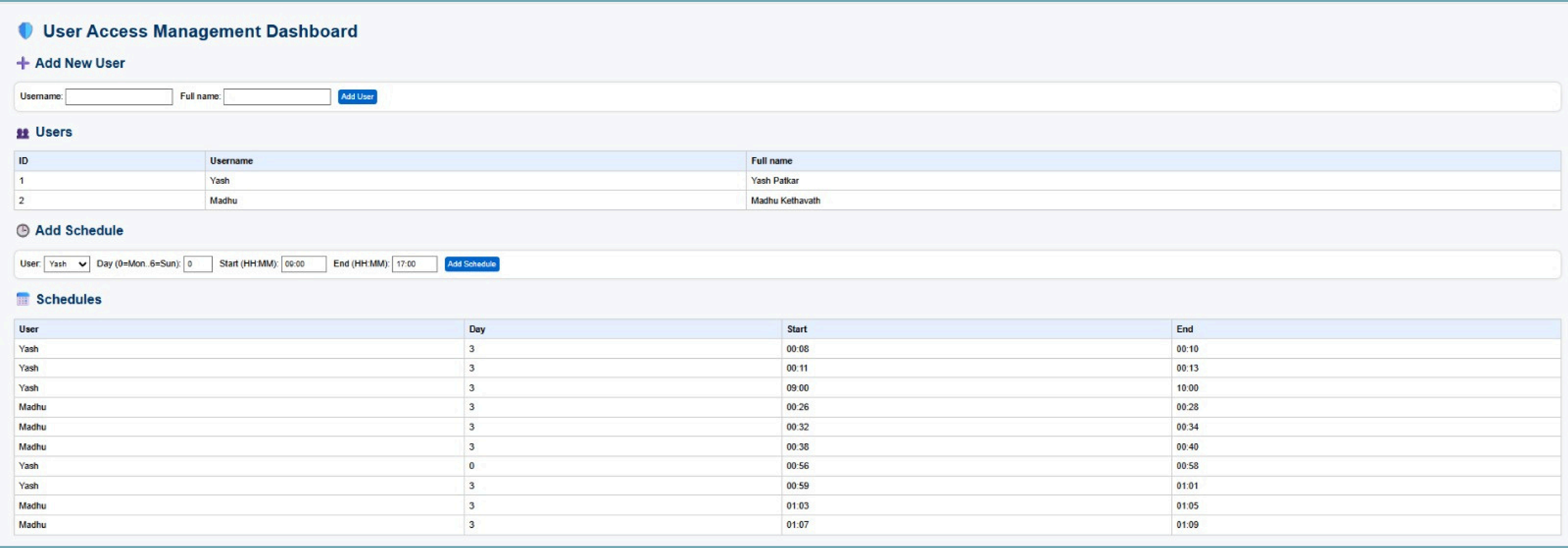
Flask Dashboard:

- Provides a web interface for administrators to add users, set time slots, and view access logs.
- Manages interaction between the user and the database.

03

Access Validator:

- Runs automatically at scheduled intervals to compare system time with user schedules.
- Logs whether each user is ALLOWED or DENIED and updates the dashboard.



User Access Management Dashboard

Add New User

Username: [ ] Full name: [ ] Add User

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 Add Schedule

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

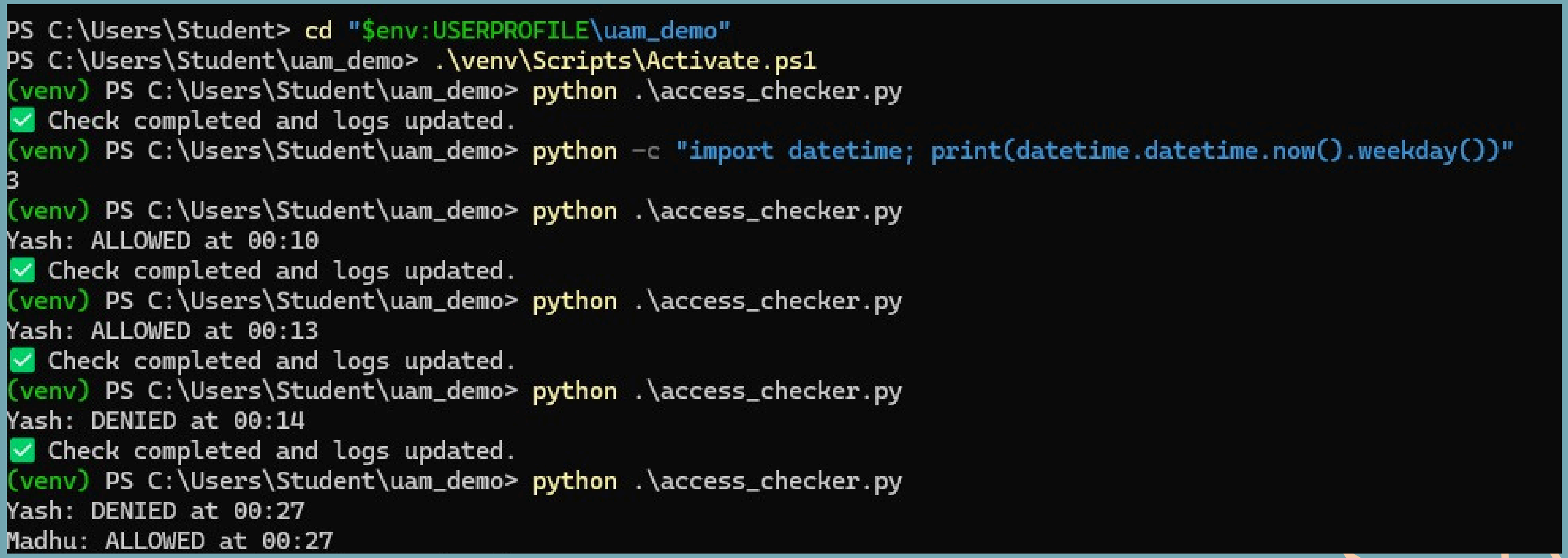
SYSTEM WORKFLOW & RESULTS

Process Flow:

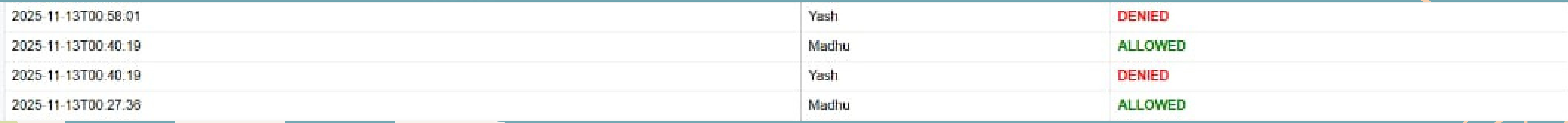
1. Admin adds users and defines allowed time slots.
2. Scheduler runs every minute to check current time.
3. Logs user’s status as ALLOWED or DENIED in database.
4. Dashboard updates automatically.

Testing Users:

- Yash → DENIED during allowed slot.
- Madhu → ALLOWED outside schedule.



```
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
```

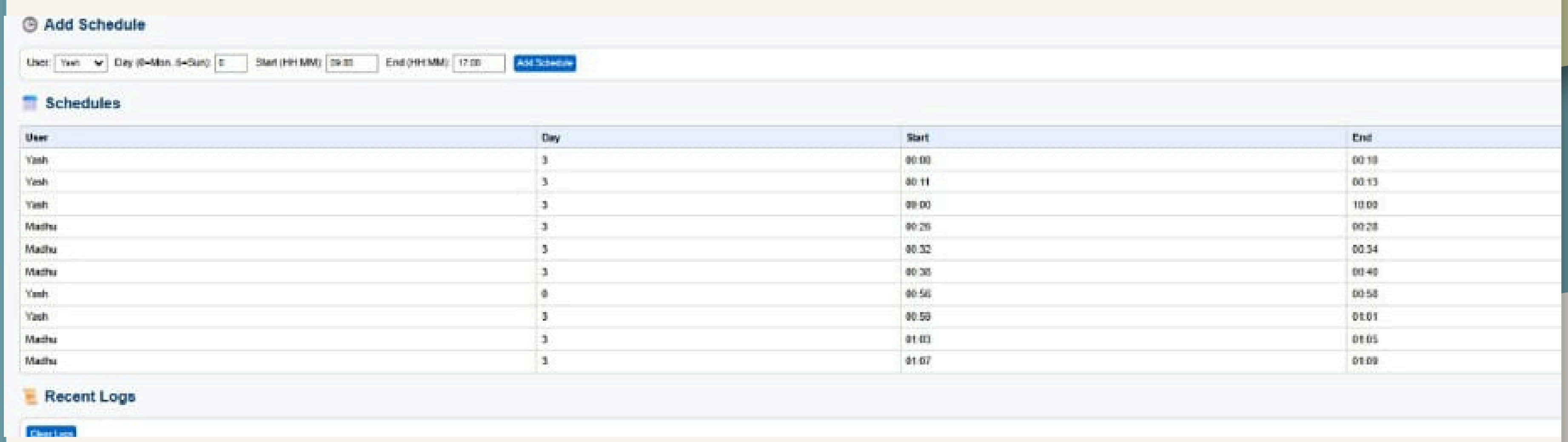


2025-11-13T00:58:01	Yash	DENIED
2025-11-13T00:40:19	Madhu	ALLOWED
2025-11-13T00:40:19	Yash	DENIED
2025-11-13T00:27:36	Madhu	ALLOWED

RESULTS & DASHBOARD DEMONSTRATION

Demonstration Summary:

- Dashboard displays all user activities with timestamps.
- Multiple users tested under different time slots.
- Logs update dynamically with every access check.

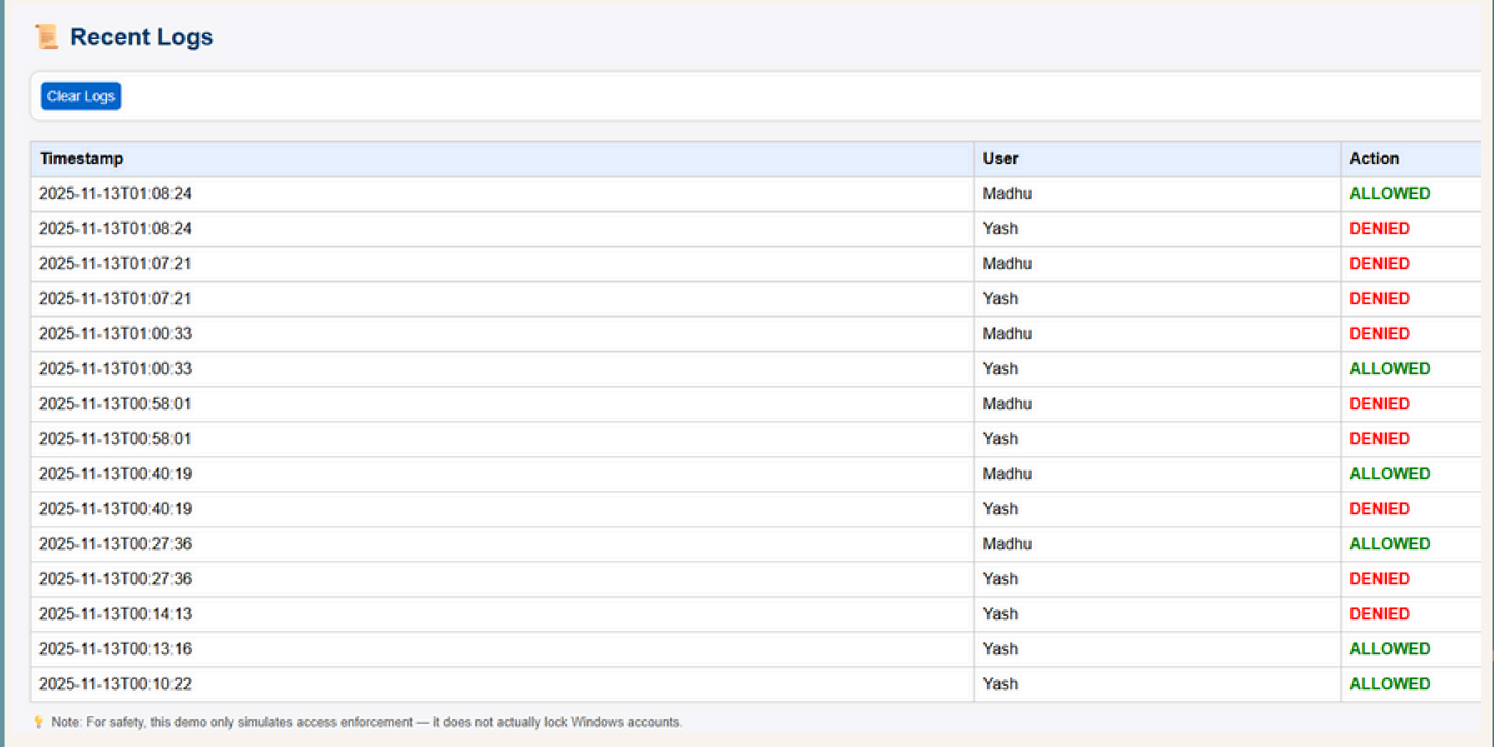


Add Schedule

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



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.

CONCLUSIONS

1. Successfully developed a working prototype for time-based access control using Python, Flask, and SQLite.
2. Implemented automated permission checks and real-time logging through a simple web dashboard.
3. Validated system accuracy with multiple users — correctly distinguishing ALLOWED and DENIED access.
4. Designed a scalable and modular system ready for integration with real SSH/RDP enforcement and multi-admin roles.

THANK YOU

REFERENCES

1. Microsoft Learn – Task Scheduler Overview
2. https://learn.microsoft.com/en-us/windows/win32/taskschd/task-scheduler-start-page
3. MongoDB Documentation – Managing Time-Based Records https://www.mongodb.com/docs/manual/tutorial/manage-timestamps/
4. Linux Manual Pages – cron(8) and systemd.timer(5)https://man7.org/linux/man-pages/man8/cron.8.html
5. OpenSSH Project – Secure Shell (SSH) Manualhttps://www.openssh.com/manual.html
6. Flask Framework Documentation – Pallets Projects (2024)https://flask.palletsprojects.com/
7. SQLite Documentation – SQLite Consortium (2023)https://www.sqlite.org/docs.html
8. Python Documentation – Python Software Foundation (2024)