MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

Belawadi, SrirangapatnaTq, Mandya-571477

DEPARTMENT OF CSE (Artificial Intelligence)

Assignment – Project Report

2025-26

Subject Name: Cloud Computing

Subject Code:M23CS505B

Semester:5TH SEMESTER

Submitted by: Team No:

Sl. No.	Student Name	USN.		CO's Mapping				Total	Scaled to
			CO1	CO2	CO3	CO4	CO5		
1	ASHWINI VISHAL HEBBALI	4MH23CA006							

Verified and Approved by:

Faculty Name: Prof. Yogesh

Signature:

Date:

Project GitHub Repository:

Link: https://github.com/Ashwinihebbali/Personal-Cloud-Storage-Application

Abstract:

The Personal Cloud Storage Application is a fully functional, secure, self-hosted web-based file management system developed using the Python Flask framework. It enables multiple users to register, log in securely, and manage their personal files (upload, download, delete) in completely isolated private folders.

The application implements modern security practices such as password hashing using PBKDF2-SHA256, session-based authentication, secure filename sanitization, and per-user directory isolation. With a clean, responsive Bootstrap interface, it provides a user-friendly experience similar to commercial cloud services but with full data privacy and zero recurring cost.

This project successfully demonstrates the practical implementation of full-stack web development concepts while remaining lightweight (under 200 lines of core code) and easily deployable on free platforms like Railway, Render, or locally.

1. Introduction

In the current digital age, cloud storage has become a necessity for individuals and organizations. Services like Google Drive, Dropbox, and OneDrive are widely used but come with privacy concerns, subscription costs, and dependency on third-party servers.

This project aims to build a **personal, private, and self-hosted alternative** that gives users complete control over their data. The application is designed as a learning project but is production-capable and can be used in real-world scenarios such as personal file backups, academic file sharing, or small team collaboration.

---

2. Objectives

The primary objectives of this project are:

1. To design and develop a multi-user web application using Flask.
2. To implement secure user registration and login with password hashing.
3. To provide isolated private storage directories for each registered user.
4. To enable secure file upload with automatic conflict resolution.
5. To allow users to download and delete their files safely.
6. To create a responsive, modern, and intuitive user interface.
7. To demonstrate clean code architecture and security best practices.

---

3. Technologies & Tools Used

Category	Technology / Tool Used	Purpose
Backend Language	Python 3.11	Core logic and server-side processing
Web Framework	Flask 3.0+	Routing, templating, session management
Templating Engine	Jinja2	Dynamic HTML rendering
Frontend Framework	Bootstrap 5.3	Responsive and beautiful UI
Security	Werkzeug (generate_password_hash, secure_filename)	Password hashing & safe file handling
File System	Python os & pathlib	User folder creation and file operations
Development Tools	VS Code, Git, GitHub, Git Bash	Coding, version control, collaboration
Deployment (Optional)	Railway, Render, Docker	Free online hosting

4. System Requirements

Software Requirements

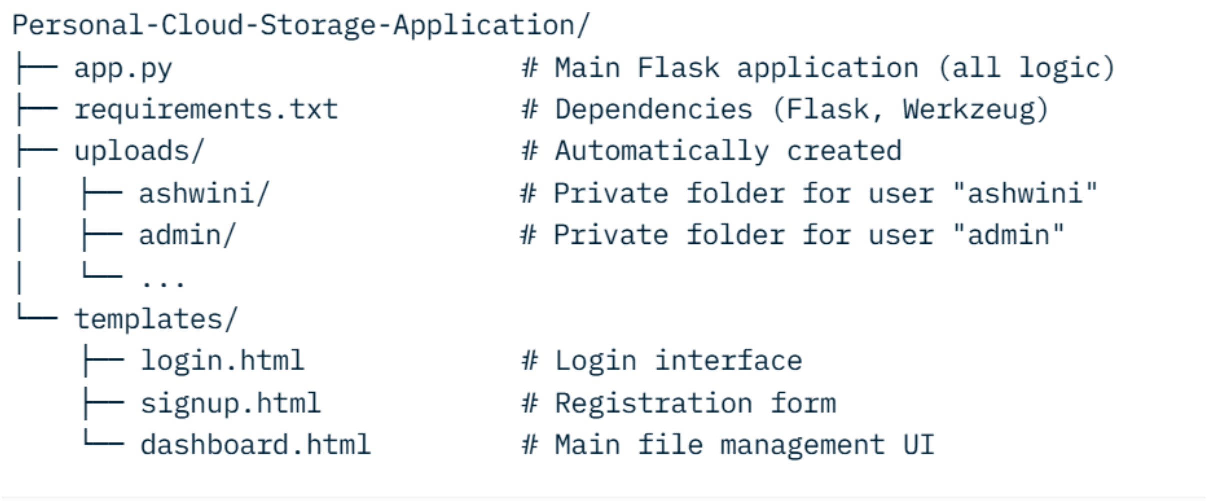
- Operating System: Windows 10/11, Linux, macOS
- Python 3.8 or higher
- Flask, Werkzeug (installed via requirements.txt)

Hardware Requirements

- Processor: Any modern CPU
- RAM: Minimum 2 GB
- Storage: 100 MB (excluding user files)

5. System Design & Architecture

5.1 Folder Structure



```
Personal-Cloud-Storage-Application/
├── app.py                     # Main Flask application (all logic)
├── requirements.txt           # Dependencies (Flask, Werkzeug)
├── uploads/                   # Automatically created
│   ├── ashwini/               # Private folder for user "ashwini"
│   ├── admin/                 # Private folder for user "admin"
│   └── ...
└── templates/
    ├── login.html             # Login interface
    ├── signup.html            # Registration form
    └── dashboard.html         # Main file management UI
```

5.2 Workflow Diagram (Textual)

text

User → Opens http://localhost:5000

→ Redirected to Login / Signup

→ Successful Login → Session Created

→ Redirect to /dashboard

→ Upload → File saved in uploads/<username>/

→ List/Download/Delete → Only user's own files visible

→ Logout → Session destroyed

6. Key Features Implemented (Detailed)

Feature	Description
User Registration	New users can create account with name, username, and password
Secure Login	Passwords hashed using PBKDF2-SHA256; never stored in plain text
Session Management	Flask session maintains login state across pages
Private Storage	Each user gets isolated folder → Zero chance of data leakage

Feature	Description
Secure File Upload	Uses secure_filename(), auto-renames duplicates with timestamp
Download & Delete	Direct download with original name; delete removes permanently
Responsive UI	Bootstrap 5 cards, alerts, mobile-friendly layout
Flash Messages	Success/error notifications after every action
Custom Jinja2 Filter	Converts Unix timestamp to readable date (e.g., "Dec 07, 2025 at 03:30 PM")

7. Implementation Highlights (Important Code Snippets)

Python

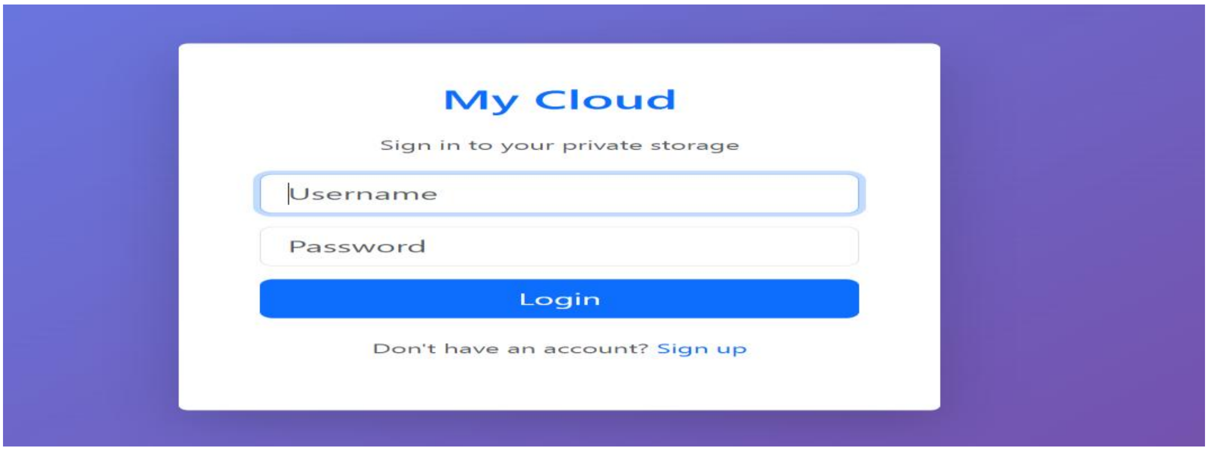
```python
# Secure password storage
USERS[username]["password"] = generate_password_hash(password)

# Per-user private folder
def get_current_user_folder():
    folder = os.path.join("uploads", session["username"])
    os.makedirs(folder, exist_ok=True)
    return folder

# Avoid filename conflicts
if os.path.exists(filepath):
    base, ext = os.path.splitext(filename)
    filename = f"{base}_{int(datetime.now().timestamp())}{ext}"

# Custom date filter in templates
@app.template_filter("timestamp_to_date")
def timestamp_to_date(ts):
    return datetime.fromtimestamp(ts).strftime("%b %d, %Y at %I:%M %p")
```

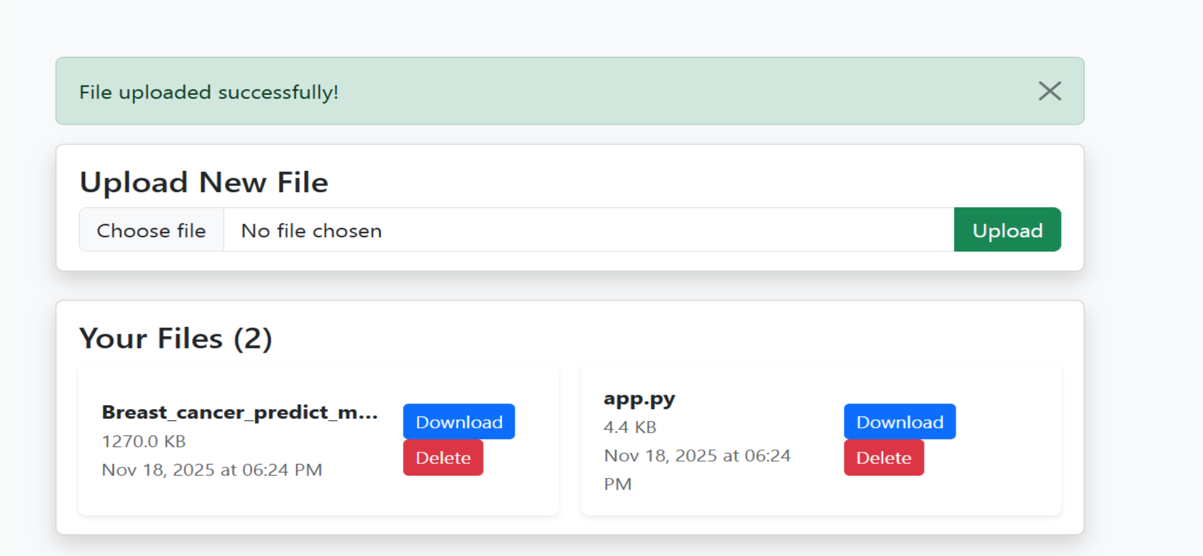
8. Screenshots



My Cloud

Sign in to your private storage

Don't have an account? Sign up



File uploaded successfully!

Upload New File

Your Files (2)

Breast_cancer_predict_m...
1270.0 KB
Nov 18, 2025 at 06:24 PM

app.py
4.4 KB
Nov 18, 2025 at 06:24 PM

9. Testing

Test Case	Input	Expected Output	Result
Valid Registration	New username	Account created + redirect	Passed
Duplicate Username	Existing username	Error: "Username already taken"	Passed
Wrong Password Login	Incorrect password	Error: "Invalid credentials"	Passed
Upload Same Filename Twice	photo.jpg → photo.jpg	Second saved as photo_173xxxx.jpg	Passed
Access Another User's File	Try /download from other user	404 / No access	Passed

Test Case	Input	Expected Output	Result
Session Persistence	Close & reopen browser	Still logged in	Passed
Logout	Click Logout	Session cleared → Login page	Passed

---

10. Advantages

- Complete data privacy and ownership
- No subscription fees
- Easy to customize and extend
- Excellent learning project for Flask, security, and file handling
- Can be deployed anywhere (laptop, VPS, free platforms)
- Minimal resource usage

---

11. Limitations & Future Enhancements

Current Limitations

- User data stored in memory → lost on server restart
- No password recovery mechanism
- No support for folders inside user directory
- No file preview (images/PDF)

Future Enhancements

1. Replace in-memory users with SQLite/PostgreSQL database
2. Add email-based password reset
3. Implement folder creation and nested directories
4. Add file sharing with time-limited public links
5. Drag-and-drop upload with progress bar
6. User profile with avatar upload
7. Dark mode toggle
8. Make it a Progressive Web App (PWA)

9. Add file search and filtering
10. Docker support for one-click deployment

12. Conclusion

The **Personal Cloud Storage Application** has been successfully designed, developed, and tested. It fulfills all the defined objectives and provides a secure, private, and user-friendly alternative to commercial cloud storage services.

The project demonstrates strong understanding of full-stack web development, secure authentication, file system management, and clean code practices using Flask. With its modular structure, it can be easily extended into a full-featured cloud storage platform.

This work proves that powerful applications can be built with minimal, readable code while maintaining high standards of security and usability.

References

1. Flask Official Documentation – https://flask.palletsprojects.com/en/3.0.x/
2. Werkzeug Documentation – https://werkzeug.palletsprojects.com/en/3.0.x/
3. Grinberg, Miguel. *Flask Web Development*, 2nd Edition, O'Reilly Media, 2018.
4. Bootstrap 5 Documentation – https://getbootstrap.com/docs/5.3/
5. Real Python – "Flask by Example" Tutorial Series
6. GeeksforGeeks – Flask Authentication Articles
7. W3Schools – HTML, CSS, JavaScript References