**Title: Automated File Backup Scheduler with Email Notification**

**Purpose of the Work and Expected Outcome**

The purpose of this project is to demonstrate and implement an automated backup system for user-uploaded files. This system compresses files into a ZIP archive and uploads them to Amazon S3 cloud storage. Additionally, an email notification is sent to the admin each time a backup is completed. This eliminates manual intervention and ensures regular, secure, and scalable data backup.

**Expected Outcomes:**

* A web-based file upload and backup system developed using Flask.
* ZIP compression of all uploaded files.
* Backup files stored securely on Amazon S3.
* Automatic email notification sent to the administrator upon successful backup.
* A professional web dashboard to manage uploads and view S3-stored backups.

**Literature Review**

Automated backups are essential for data integrity and disaster recovery. Cloud storage solutions like Amazon S3 offer high durability and scalability. Integration with Python libraries like boto3 and smtplib allows secure data transfers and automated notifications. Industry practices favor automation and cloud-native architectures for reliability and cost-effectiveness.

Key references:

* Amazon S3 Documentation: Secure and scalable object storage.
* Python Docs: shutil, zipfile, os, and smtplib modules.
* Flask Documentation: Lightweight backend web framework.

**Existing Process and Its Limitations**

Manual backups are time-consuming and prone to user error. Local backups are often overlooked and susceptible to data loss due to hardware failure. There is also no systematic tracking or notification of backup activity.

**Limitations:**

* Lack of automation.
* No offsite/cloud backup.
* No notifications of backup success/failure.
* High dependency on manual execution.

**Justification for Selecting a Particular Methodology**

The project uses the Agile development methodology for iterative development, enabling:

* Rapid prototyping of UI and backend logic.
* Continuous integration and testing of features.
* Feedback-based refinement of upload and backup processes.

**Dissertation Methodology**

1. **Requirement Gathering**: Define objectives and user flow for file upload, backup, and notification.
2. **System Design**:
   * Flask backend with routes for upload, backup, and view.
   * File handling and ZIP creation using shutil.
   * Integration with S3 using boto3.
   * Email integration using smtplib.
3. **Implementation**:
   * Build HTML templates for UI.
   * Implement file upload and validation.
   * Create ZIP archive of uploaded files.
   * Upload the archive to an S3 bucket.
   * Send email on successful upload.
4. **Testing**:
   * Unit and functional testing.
   * Email trigger validation.
   * S3 storage verification.
5. **Deployment**:
   * Local server (Flask run) or cloud platform (EC2).

**Hardware and Software Requirements**

**Hardware:**

* Processor: Intel i3 or higher
* RAM: 4GB minimum
* Disk: 50GB+ recommended

**Software:**

* Python 3.9+
* Flask
* boto3
* smtplib/email
* Amazon S3 (Free Tier)
* Bootstrap (UI styling)
* Git for version control

**Benefits Derivable from the Work**

* No manual effort required to manage backups.
* Reliable cloud storage via Amazon S3.
* Email alerts provide confirmation and transparency.
* Simple UI to manage uploads and backups.
* Can be extended to support scheduled backups or database backups.

**References**

* Amazon Web Services (AWS) Documentation
* Python Standard Library (shutil, zipfile, smtplib)
* Flask Documentation

**16-Week Plan Tasks and Deliverables**

|  |  |  |
| --- | --- | --- |
| **Week** | **Task** | **Deliverable** |
| 1 | Finalize requirements and technology stack | Requirement Document |
| 2 | Setup Flask project structure | Flask Skeleton |
| 3 | Design HTML templates using Bootstrap | UI Mockups |
| 4 | Implement file upload functionality | Upload Form |
| 5 | Create ZIP backup logic | ZIP Archiver Code |
| 6 | Configure S3 and test upload manually | S3 Test Report |
| 7 | Integrate boto3 to upload ZIP files | S3 Upload Module |
| 8 | Setup SMTP and email credentials | Email Config Setup |
| 9 | Implement email sending post-backup | Notification Email Feature |
| 10 | Error handling and validation | Logs and Edge Case Reports |
| 11 | Develop dashboard to view uploads/S3 files | View Page Code |
| 12 | UI enhancements (icons, layout, styling) | Finalized UI Screens |
| 13 | Conduct integration and functional testing | Bug Report & Fixes |
| 14 | Deployment to local or remote server | Deployment Instructions |
| 15 | Documentation and user manual | Project Report |
| 16 | Final Review and Submission | Final Project + Report |