

**MAJOR PROJECT-1**

**Mid Term Report**  
For  
**Ransomware Resilient Backup System**

Submitted By

<b>Harsh Sharma</b>	<b>Satvik Verma</b>	<b>Swayam Prakash Mohanty</b>	<b>Abhay Pratap Topwal</b>
R2142211093	R2142211356	R2142211200	R2142211187



Cloud Software Operations Cluster  
School Of Computer Science  
UNIVERSITY OF PETROLEUM & ENERGY STUDIES,  
DEHRADUN- 248007. Uttarakhand

Under the guidance of-  
Dr. Narendra Dewangan

# Table of Contents

Topic	Page Number
Table of Content	
1. Introduction	1
1.1 Purpose of the Project	1
1.2 Target Beneficiary	1
1.3 Project Scope	1
1.4 References	1
2 Project Description	1-3
2.1 Data/ Data structure	1-2
2.2 SWOT Analysis	2
2.3 Project Features	2
2.4 Design and Implementation Constraints	2-3
3 System Requirements	3
3.1 User Interface	3
3.2 Technologies and Tools Used	3
4 Non-functional Requirements	3
4.1 Performance requirements	3
4.2 Security requirements	3
4.3 Software Quality Attributes	3
5 Code Implemented	4
5.1 Backend	4
6 Conclusion	5

# 1. Introduction

## 1.1 Purpose of the Project

The **Ransomware Resilient Backup System** is designed to provide robust backup and recovery solutions against ransomware attacks. It ensures the safety and integrity of critical data through advanced backup mechanisms, real-time monitoring, and secure storage methods. By using **MongoDB, Percona Backup for MongoDB, and Docker**, this system provides an effective defence strategy to recover from data breaches or encryptions caused by ransomware.

## 1.2 Target Beneficiary

The primary beneficiaries of this project include:

- Small to medium enterprises requiring secure data backup.
- IT administrators managing critical infrastructure.
- Educational institutions and individual developers seeking ransomware-proof solutions.

## 1.3 Project Scope

The project involves creating a distributed backup solution featuring:

- Real-time database backups using **Percona Backup for MongoDB**.
- User-friendly interface built using **React.js** and **Flask** for managing backups, restores, and timestamped recovery options.
- Secure and portable deployment through **Docker containers**.

## 1.4 References APA format

- [1] Docker, I. (2020). Docker. *linea*]. [Junio de 2017]. Disponible en: <https://www.docker.com/what-docker>.
- [2] Mongo, D. B. (2015). *Mongodb*.
- [3] Petrov, P., Kuyumdzhev, I., Dimitrov, G., & Kremenska, A. (2022). Relative Performance of Various Types of Repositories for MySQL Archive Backup and Restore Operations. *iJOE*, 18(13), 153.
- [4] Ghimire, D. (2020). Comparative study on Python web frameworks: Flask and Django.
- [5] Bui, M. (2020). Implementing cluster backup solution to build resilient cloud architecture.
- [6] Cybersecurity, C. I. (2018). Framework for improving critical infrastructure cybersecurity. URL: <https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.4162018.7>.
- [7] Gillies, A. (2011). Improving the quality of information security management systems with ISO27000. *The TQM Journal*, 23(4), 367-376.

# 2. PROJECT DESCRIPTION

## 2.1 Data structure

The system leverages the following data structures:

- JSON Objects:** To store backup metadata (e.g., timestamps, backup IDs).
- MongoDB Collections:** For database entries and recovery.
- Log Files:** To track backup and restore operations.

## 2.2 SWOT Analysis

<b>Strengths</b> <ol style="list-style-type: none"><li>1. Provides a reliable and resilient backup system against ransomware attacks.</li><li>2. User-friendly GUI for ease of use.</li><li>3. Open-source tools reduce project cost.</li></ol>	<b>Weaknesses</b> <ol style="list-style-type: none"><li>1. Relies heavily on Docker and MongoDB; may require compatibility checks.</li><li>2. Requires manual setup for Docker and MongoDB initially.</li><li>3. No integration with cloud platforms (e.g., AWS, Azure) in the current scope.</li></ol>
<b>Opportunities</b> <ol style="list-style-type: none"><li>1. Can be extended to support cloud backups.</li><li>2. Integration with AI/ML for predictive analytics on backup integrity.</li></ol>	<b>Threats</b> <ol style="list-style-type: none"><li>1. New ransomware methods may target backup systems.</li><li>2. Security vulnerabilities in underlying libraries (e.g., Docker, MongoDB).</li></ol>

## 2.3 Project Features

The features of this project are:

- i. **Backup Data:** Allows users to create backups of a MongoDB database.
- ii. **Restore Data:** Enables restoration from backup based on user-selected timestamps.
- iii. **Enter Data:** Provides a user-friendly GUI to add or modify database entries.
- iv. **Display Backups:** Displays a list of all available backup timestamps.
- v. **Dockerized Environment:** Ensures portability and reproducibility.

## 2.4 Design and Implementation Constraints

- i. **Hardware Constraints:** Requires Docker-compatible systems.
- ii. **Software Constraints:** Dependent on MongoDB and Percona Backup for MongoDB.
- iii. **Environmental Constraints:** Requires a secure and isolated environment for deployment.

## 2.5 Design diagram

Figure 1 represents diagram of the **Ransomware Resilient Backup System** illustrates the system's architecture and the flow of data between various components. The **User** interacts with the **React Frontend**, where they can perform actions, such as creating backups, restoring data, or entering new data. The **React Frontend** sends API requests to the **Flask API**, which processes these requests and manages communication with the **MongoDB Database** to store metadata like backup timestamps and data details. The **Flask API** also interfaces with the **File Storage** system, either locally or in the cloud, where the actual backup files are stored for retrieval during restoration. This flow ensures that the system is both user-friendly and resilient, offering seamless backup and restore operations while securely storing data.

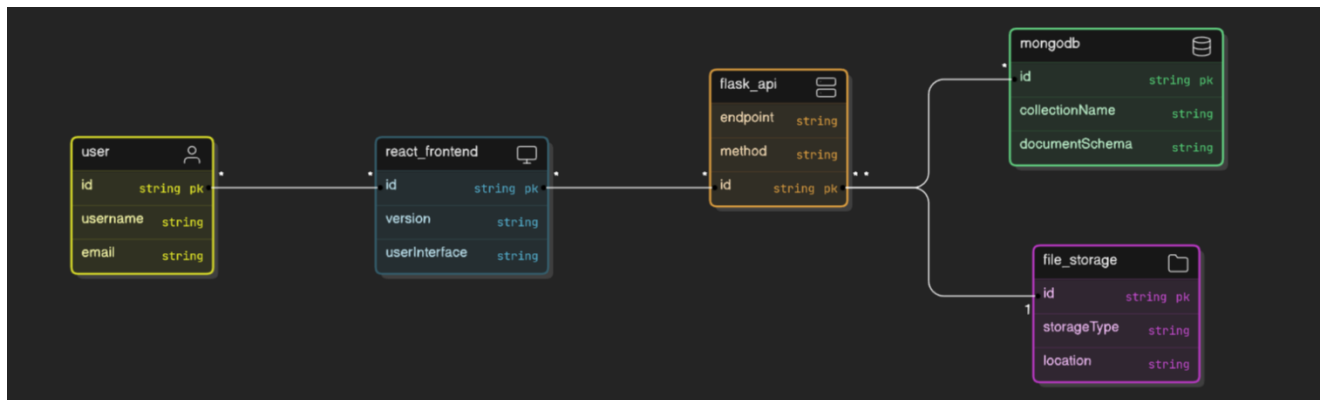


Fig 1: Design diagram

## 3. SYSTEM REQUIREMENTS

### 3.1 User Interface

- Interactive Dashboard:** Built with React.js, showing options for backup, restore, and data entry.
- Real-time Updates:** Displays live database and backup statuses.
- Error Alerts:** Notifies users of failed backups or restores.

### 3.2 Technologies and Tools Used

- Frontend:** React.js with Material-UI for styling.
- Backend:** Flask (Python).
- Database:** MongoDB.
- Backup Tool:** Percona Backup for MongoDB.
- Containerization:** Docker.

## 4. NON-FUNCTIONAL REQUIREMENTS

### 4.1 Performance Requirements

- i. **Real-time Backups:** Backup operations should execute within seconds for medium-sized datasets.
- ii. **Scalability:** The system should handle increasing data sizes effectively.

### 4.2 Security Requirements

- i. **Data Encryption:** Backup files are stored securely to prevent unauthorized access.
- ii. **Role-Based Access Control:** Prevents unauthorized users from modifying backups.

### 4.3 Software Quality Attributes

- i. **Usability:** GUI simplifies backup and restore operations.
- ii. **Portability:** Docker ensures cross-platform deployment.

## 5 CODE IMPLEMENTATION

### 5.1 Backend (Flask API implementation)

```
@app.route('/backup', methods=['POST'])
def create_backup():
    # Backup logic here
    return jsonify({"message": "Backup created successfully"})
```

Fig 2: Flask route for creating a backup

```
@app.route('/restore', methods=['POST'])
def restore_backup():
    # Restore logic here
    return jsonify({"message": "Backup restored successfully"})
```

Fig 3: Flask route for restoring data

## 6 Conclusion

The **Ransomware Resilient Backup System** successfully demonstrates a reliable and scalable solution for protecting critical data from ransomware attacks. With an intuitive user interface and secure backend, this project offers a practical approach to ensuring data safety. Future enhancements include cloud integration and predictive failure analytics.