




Trust Indicators Project: Backup & Recovery Plan

Version: 1.0 Last Updated: May 21, 2025

1. Introduction











This Backup and Recovery Plan aims to ensure that the Trust Indicators project can effectively protect critical data and applications and can quickly restore services in the event of unforeseen incidents such as data loss, hardware failure, security breaches, or service interruptions. Data is at the core of the project, making a comprehensive plan crucial.


Objectives:

-  **Minimize Data Loss (RPO - Recovery Point Objective):** Ensure that the amount of data lost is minimal in the event of a disaster.
-  **Rapid Service Recovery (RTO - Recovery Time Objective):** Restore normal application operation as quickly as possible.
-  **Ensure Data Integrity & Security:** Protect backup data from unauthorized access or corruption.

2. Key Assets to Back Up

Key assets requiring backup include:

Icon	Asset Category	Specific Content	Backup Importance
	Source Code	Complete Flask application code, including Python scripts, HTML templates, CSS, JavaScript files, etc.	 Critical
	Database	SQLite database file (<code>MyDatabase.db</code>), containing all user information, image information, metadata, trust profiles, etc.	 Critical
	Configuration Files	Deployment scripts (configurations mentioned in <code>deploy.md</code>), environment variables (<code>.env</code> file template or instructions), Caddy configuration files, etc.	 High
	Project Documentation	All technical documents, user manuals, handover documents, API documentation, etc.	 High
	API Keys & Credentials	API keys for Alibaba Cloud AIGC, ImgBB, email services, etc. (Note: Actual keys should not be backed	 Critical (Secure


Icon	Asset Category	Specific Content	Backup Importance
		up directly in easily compromised locations; backup instructions for their retrieval or secure storage location instead).	Storage)
	(Optional) Original User-Uploaded Images (if stored separately)	If original images are stored in a form other than database binary storage. <i>Currently, the project stores images as binary data in the database.</i>	● High (If Applicable)



3. Backup Strategy




3.1. Source Code

- **Backup Method:**
 -  **Version Control System (Git):** Primary backup method. All code changes should be committed to the GitHub repository (<https://github.com/bridgeL/Trust-indicator>).
 - [Icon: Git/GitHub Logo]
 - Periodically clone the GitHub repository to multiple secure offline locations.
- **Backup Frequency:**
 - **Git:** Immediately after every significant code change (Continuous).
 - **Offline Clones:** At least once a week, or after major version releases/iterations.
- **Backup Location:**
 - Primary: GitHub private/public repository.
 - Secondary: Secure external hard drives of core project members, encrypted cloud storage (e.g., MEGA, Tresorit, ensuring compliance).
- **Retention Policy:**
 - **Git:** Retain all commit history indefinitely.
 - **Offline Clones:** Retain the last 3-5 major version clones.




3.2. Database (SQLite)


- **Backup Method:**
 -  **File Copy:** Since SQLite is used, the database is a single file (`instance/MyDatabase.db`).
 - Write a script to periodically copy this database file to a designated backup directory.
 - [Icon: Script file]
 - Consider stopping the application service or briefly disconnecting database connections before copying to ensure file consistency (more critical for high-concurrency scenarios, may have less impact on the current project but still good practice).
- **Backup Frequency:**
 - At least **once daily**. During high-usage periods, consider increasing backup frequency (e.g., every 12 or 6 hours).
- **Backup Location:**
 - A non-application directory on the server (e.g., `/var/backups/trust_indicator_db/`).

- Periodically synchronize backup files to a secure remote location (e.g., using `rsync` or secure cloud storage services like AWS S3 Glacier, Backblaze B2, with encryption enabled).
 - [Icon: Cloud storage]
- **Retention Policy:**
 - **Daily Backups:** Retain for the last 7-14 days.
 - **Weekly Backups:** Retain for the last 4-8 weeks.
 - **Monthly Backups:** Retain for the last 6-12 months.
 - Consider retaining at least one annual backup.

3.3. 🔑 Configuration Files

- **Backup Method:**
 -  **Version Control (Git):** For non-sensitive configurations (like Caddyfile templates, deployment script templates), include them in Git version control.
 - **Secure Storage:** For `.env` files or specific server configuration files containing sensitive information (like actual API keys), store them in a secure password manager (e.g., Bitwarden, 1Password) or an encrypted archive.
 - [Icon: Password manager/Safe]
- **Backup Frequency:**
 - After every change to configuration files.
- **Backup Location:**
 - Git repository (non-sensitive configurations).
 - Secure password manager or encrypted storage (sensitive configurations).
- **Retention Policy:**
 - **Git:** Indefinitely.
 - **Password Manager:** Latest version and necessary historical versions.

3.4. 📄 Project Documentation

- **Backup Method:**
 -  **Version Control (Git):** Include Markdown documents, etc., in the project's Git repository.
 - **Cloud Storage/Shared Drive:** For binary documents like PDFs, use Google Drive, OneDrive, Dropbox, etc., with appropriate permissions set.
 - [Icon: Google Drive/Dropbox]
- **Backup Frequency:**
 - Whenever there is a significant update to the documentation.
- **Backup Location:**
 - Git repository.
 - Team-shared cloud storage.
- **Retention Policy:**
 - **Git:** Indefinitely.
 - **Cloud Storage:** Latest version and necessary historical versions.

4. Recovery Procedures

In the event of a failure, follow these steps for recovery:

4.1. 🚨 Assess the Situation

1. **Identify the Problem:** Determine the type and scope of the failure (e.g., database corruption, code deployment error, server hardware failure, or security incident).
2. **Notify Relevant Personnel:** Inform the project team and (if applicable) users.

4.2. 💻 Source Code Recovery

1. Restore from Git:

- On a new or repaired server, clone the latest stable branch/tag from GitHub:

```
➤ git clone [https://github.com/bridgeL/Trust-indicator.git](https://github.com/bridgeL/Trust-indicator.git)
cd Trust-indicator
# git checkout <stable-branch-or-tag> # (if a specific version is needed)
```

- If GitHub is unavailable, clone from the latest offline backup.
 - [Icon: Command line]

4.3. 🗄️ Database Recovery (SQLite)

1. **Select Backup:** Based on the time of failure and RPO, choose the most recent, consistent database backup file.
2. **Stop Application:** Ensure the Flask application (or any process accessing the database) is stopped to avoid conflicts during recovery.

```
➤ # (Example: if using systemd)
# sudo systemctl stop trust_indicator_app
```

3. Restore File:

- Copy the selected backup database file (`MyDatabase.db.backup_YYYYMMDD_HHMMSS`) back to the application instance directory (e.g., `instance/`) and rename it to `MyDatabase.db` .
- Ensure correct file permissions and ownership.

```
➤ cp /var/backups/trust_indicator_db/MyDatabase.db.backup_YYYYMMDD_HHMMSS /path/to/project/instance/MyDatabase.db
chown <your_app_user>:<your_app_group> /path/to/project/instance/MyDatabase.db
chmod 600 /path/to/project/instance/MyDatabase.db # (or appropriate permissions for the app)
```

- [Icon: File copy]

4. (Optional) Verify Database:

Use the SQLite command-line tool to check the integrity of the database file.

```
➤ sqlite3 /path/to/project/instance/MyDatabase.db "PRAGMA integrity_check;"
```

5. Restart Application:

Start the Flask application.

```
▶ # (Example: if using systemd)
# sudo systemctl start trust_indicator_app
```

4.4. 🔑 Configuration File Recovery

1. **Restore from Git:** Restore non-sensitive configuration files.
2. **Restore from Secure Storage:** Retrieve sensitive configurations (like `.env` file content) from the password manager or encrypted archive and place them in the correct location on the server. Ensure file permissions are secure.
 - [Icon: Key]

4.5. 🧪 Test the Recovery

1. **Functional Testing:** Comprehensively test all core website functionalities (user login, image upload, image Browse, AIGC detection, trust profiles, etc.).
2. **Data Validation:** Sample check the restored data for accuracy and completeness.
3. **Performance Monitoring:** Monitor the application's performance after recovery.

🤖 5. Backup & Recovery Testing

- **Test Frequency:** At least **once per quarter**, or after significant system changes.
- **Test Scope:**
 - Simulate different types of failures (e.g., database file deletion, code rollback requirements).
 - Execute the full recovery procedure.
 - Verify if RPO and RTO are within acceptable limits.
- **Record Results:** Document the testing process, encountered issues, and solutions in detail. Update this plan based on test results.
 - [Icon: Checklist/Clipboard]

🛡️ 6. Security Considerations

- **Encrypt Backups:** Encrypt database backups and sensitive configuration file backups stored in remote locations or on external media.
- **Access Control:** Strictly control access permissions to backup files and recovery tools.
- **Backup Integrity:** Regularly check backup files for corruption.
- **Off-site Storage:** Store at least one copy of critical backups in a physical location different from the primary server.
 - [Icon: Shield]

📞 7. Responsible Parties & Contacts

Role/Responsibility	Responsible Person/Team	Contact (Placeholder)
Backup Strategy Design & Maintenance	Project Technical Lead	techlead@example.com
Daily Backup Execution & Monitoring	Server Administrator/Team	admin@example.com
Recovery Procedure Execution	Project Technical Lead	techlead@example.com
Backup & Recovery Testing	Project Team	team@example.com

(Please have the next team update the actual responsible parties and contact information.)

This plan will be reviewed and updated periodically to ensure its continued effectiveness.