FULL PROJECT REPORT

VedDrive

A personal cloud drive for organizing and accessing files across devices.

PROJECT	VedDrive
CREATOR	Vedhant Khajuria
ROLE	Creator and Full-Stack Developer
REPORT DATE	29 September 2026
PROJECT SITE	https://veddrive.vedhant.in

A technical and product overview of the application, its architecture, data flows, operations, deployment, security boundaries, and future development opportunities.

Prepared by Vedhant Khajuria

This report describes the project based on its application source and deployment guide. Private server credentials are intentionally excluded.

01 / OVERVIEW

Executive summary

VedDrive is a web-based personal file drive. It gives a signed-in member a central place to upload files, arrange them in folders, find them with search and filters, mark favorites, and manage deleted items through Trash. An administrator can manage member access and registration; upload quotas are enforced by the API, though the current admin screen has no quota editor.

The application is built as a lightweight PHP site with a JavaScript interface and MySQL metadata store. File bytes are divided into configured chunks and sent to a Telegram channel through the Telegram Bot API. VedDrive keeps the mapping between each file and its remote chunks in MySQL. When a member downloads a file, the PHP endpoint retrieves those parts and streams the requested byte range.

The design is suited to a small, self-hosted project where a conventional object-storage service is not being used. It keeps file metadata and user identity under the application database while using Telegram as the remote byte store. That choice also creates operational dependencies on Telegram limits, network availability, and correct private configuration.

Project goals

- Keep a member's files organized in a single, responsive interface.
- Support large-file transfer by uploading and retrieving files in parts.
- Provide account, folder, favorite, Trash, and administrator workflows.
- Offer a deployment path for a PHP/MySQL cPanel host and a sanitized source package for GitHub.

At a glance

Area	Implementation
User interface	HTML, CSS, and browser JavaScript
Application API	PHP endpoints for session, authentication, library, uploads, and management
Metadata database	MySQL with users, folders, files, chunks, settings, and login attempt tables
File byte storage	Telegram Bot API document messages, referenced by chunk metadata
Hosting target	PHP/MySQL hosting such as cPanel, with cURL and PDO MySQL enabled

02 / EXPERIENCE

Product capabilities

Public landing and access

Visitors see a product landing page with calls to sign in or create an account. Email and password registration and login are implemented in the application. The authentication dialog also contains a Google sign-in slot. Google sign-in requires a configured OAuth web client ID in the private server config and an allowed site origin in Google Cloud; the checked-in example leaves that value blank.

Drive workspace

- Overview cards summarize stored bytes, ready files, folders, and favorites.
- The file library supports grid and list layouts, sort order, filename search, and file-type filters.
- Folders can be nested. Opening a folder filters its files and shows the current location.
- Members can upload by selecting files or dragging files into the drop area.
- File actions include download, favorite toggle, move to Trash, restore, and permanent deletion from Trash.

Account and administrator tools

- Account settings allow members to update their display name and password after confirming their current password.
- The administrator console lists accounts and can block or unblock member accounts.
- The administrator can pause or reopen new registrations.
- The user schema includes per-account quota bytes; the upload API checks that quota before accepting a new file. The current admin screen does not expose a quota editor.

Responsive layout

The interface uses a sidebar workspace on larger screens and a mobile navigation pattern at smaller widths. Controls wrap on narrow screens, action buttons use readable contrast, and dialogs group form labels and submission controls for sign-in, account creation, and folder creation.

Member journey

Step	What happens
1. Sign in	The browser obtains a session and CSRF token from the PHP API.
2. Browse	The library API returns the member's files and folders.
3. Organize	Search, filter, sort, open folders, mark favorites, or move items to Trash.
4. Transfer	Upload chunks to Telegram; later stream them through VedDrive for download.
5. Manage	Update account details or use administrator tools when authorized.

03 / SYSTEM DESIGN

Architecture and request flow

VedDrive separates the browser interface, PHP application, relational metadata, and remote file transport. The browser does not connect directly to MySQL or Telegram. All application actions pass through the PHP layer, which checks the session and ownership before operating on a member's data.

BROWSER	PHP APPLICATION	MYSQL	TELEGRAM
HTML / CSS / JS Forms, views, uploads	api.php / download.php Sessions, validation, access checks	Users, folders, files Chunk refs, settings	Remote document parts Bot API send/get file

Main components

Component	Responsibility
index.php	Serves the application shell and sets browser security headers, including a Content Security Policy.
app.js	Bootstraps the session, renders views, submits forms, handles upload chunks, and calls the API.
api.php	Implements session and library reads plus authentication, profile, folder, upload, file, and admin actions.
download.php	Checks file ownership and ready status, retrieves remote chunks, and streams downloads with byte-range support.
private/bootstrap.php	Loads private config, starts secure session handling, creates the PDO connection, and provides shared validation helpers.
schema.sql	Defines the relational tables and foreign-key relationships.
setup.php	Installs schema tables and creates the initial administrator account.

Typical API request

- The browser sends a request to api.php with an action name; protected POST requests include the session CSRF token.
- The PHP layer authenticates the session, validates input, and scopes file and folder queries to the current user.
- Database changes update metadata or chunk references; file content transfer calls the Telegram API from the server.
- JSON responses update the interface, which refreshes the library after successful changes.

04 / DATA LIFECYCLE

File transfer and metadata

Upload lifecycle

Stage	Application behavior
Start	The browser sends filename, MIME type, size, and target folder. The server checks configured maximum size, folder ownership, and available account quota, then creates a file row in uploading state.
Split	The browser slices the file into parts using the configured chunk_bytes value. The example deployment config uses 8 MiB.
Transfer	Each part is POSTed to api.php. The server validates part index and exact expected size, then sends the temporary upload to Telegram as a document.
Index	After Telegram accepts a part, VedDrive stores the remote file_id, Telegram message ID, and part size in the chunks table.
Finish	The API checks that all expected parts and bytes exist before marking the file ready.
Download	download.php verifies the signed-in owner and ready status, retrieves remote chunks, and streams bytes in order. HTTP range requests are supported.

Data model

Table	Purpose and relationships
users	Identity, password hash, role, blocked state, quota, and session version. Email is unique.
folders	Member-owned folder hierarchy through parent_id. Folder ownership references users.
files	Member-owned file metadata, optional folder, MIME type, byte size, status, favorite flag, and created time.
chunks	Remote Telegram file reference, message ID, part order, and byte size for each file part.
settings	Application-level values such as whether registration is open.
login_attempts	Hashed rate-limit bucket, attempt count, and time window for authentication throttling.

Lifecycle states

A file begins as uploading, becomes ready after all parts are indexed, and can be moved to trash. Restoring moves it back to ready. Permanent deletion removes the VedDrive metadata and chunk references, which revokes access through VedDrive; it does not erase the already-sent Telegram messages from the remote channel.

05 / SAFEGUARDS

Security, privacy, and trust boundaries

Implemented controls

- Passwords are stored as hashes using PHP password_hash and checked with password_verify.
- Login and registration regenerate the session ID after successful authentication.
- Session cookies are HttpOnly, SameSite=Lax, and marked Secure when HTTPS is detected.
- Protected POST operations check a session CSRF token.
- Authentication attempts are rate-limited by hashed IP and email buckets.
- PDO prepared statements are used for database queries. File and folder lookups are scoped to the authenticated user.
- Blocked users and session-version changes invalidate access to protected requests.
- The private directory has an .htaccess rule denying direct web access, and configuration files are excluded from the GitHub package.

Important privacy boundary

The current transfer code sends file parts to Telegram as document uploads. The inspected implementation does not encrypt file content before upload. Do not describe the current system as end-to-end encrypted or assume that Telegram is a private, encrypted object store. Anyone with sufficient access to the configured Telegram channel or bot may be able to access stored parts. Protect the bot token, restrict channel membership, and evaluate whether the data is appropriate for this storage design.

Operational security checklist

- Use HTTPS on the site and keep PHP, database, and server software patched.
- Keep private/config.php out of public repositories, screenshots, and public backups.
- Protect the Telegram bot token and restrict the storage channel to trusted administrators.
- Delete setup.php from the public web root immediately after initial installation.
- Back up the MySQL database and test recovery; database loss breaks the mapping from files to remote Telegram parts.
- Set PHP upload, post, execution-time, and memory limits to values compatible with configured chunk_bytes.
- Use Google sign-in only with the intended OAuth client ID and exact authorized origin.

Known technical constraints

The application relies on external Telegram API availability and hosting outbound cURL access. Telegram Bot API file-size and download constraints may limit practical transfer sizes even when the VedDrive max_file_bytes setting is higher. Chunk size, PHP limits, Telegram limits, and hosting quotas must be checked together before accepting large files.

06 / OPERATIONS

Deployment and configuration

cPanel installation

Order	Deployment action
1	Create a MySQL database and database user in cPanel, then grant the required privileges.
2	Import schema.sql through phpMyAdmin.
3	Upload and extract the cPanel package into the site document root. Keep private/ under the root with its access-denial file in place.
4	Edit private/config.php with the database host, database name, database user, password, Telegram bot token and channel ID. Add google_client_id if Google login should be enabled.
5	Visit setup.php, provide setup_key, and choose a strong administrator password.
6	Remove setup.php from the live web root immediately after setup.
7	Verify sign-in, uploads, downloads, file management, and admin controls over HTTPS.

Private configuration

The server configuration contains database credentials, the Telegram bot token and channel ID, administrator email, setup key, upload-size limits, chunk size, OAuth client ID, and profile links. The GitHub package should contain only private/config.example.php with placeholders. The cPanel package may contain the private deployment config and must be handled as a secret-bearing archive.

Google OAuth setup

A Google OAuth web client ID must be configured in private/config.php. The site origin must also be added as an authorized JavaScript origin in Google Cloud Console. The browser receives the configured client ID, renders the Google Identity Services button, and posts the returned identity credential to the PHP API. The server validates the credential against Google, checks the expected audience and verified email, then signs in or creates a member account if registration is open.

Package and repository hygiene

- Keep config.php out of Git. Commit only config.example.php with safe placeholders.
- Keep cPanel archives private because they may include deploy-only secrets.
- After publishing a new package, inspect the archive entry list and scan for credentials before upload.

Deployment parameters are read from the private config file. Blank Google client ID leaves Google sign-in disabled. The included sample uses an 8 MiB chunk and a 2 GiB maximum file size; host, network, and Telegram constraints must be validated separately.

07 / ROADMAP

Quality, limitations, and future work

Project strengths

- Clear separation between browser UI, PHP endpoints, metadata database, and remote file transport.
- Chunked uploads and range-aware downloads allow transfer in manageable pieces.
- Member-scoped data access, CSRF protection, password hashing, and admin controls are part of the application flow.
- cPanel setup instructions and a sanitized configuration template support repeatable deployment.

Current limitations to communicate

- No end-to-end or application-side file encryption is implemented in the inspected upload/download code.
- Permanent deletion removes the app index but leaves remote Telegram messages in place.
- Google OAuth depends on a client ID and origin configuration that are deployment-specific.
- The present environment had no PHP CLI, so the recent source edits could not be PHP-linted or integration-tested in that workspace.
- The 8 MiB sample chunk size and 2 GiB sample max file size are configuration values, not a guarantee that the hosting provider or Telegram will accept every transfer.

Recommended next steps

Priority	Improvement
High	Add file encryption before remote transfer, with a documented key-management and recovery model, if confidential storage is a requirement.
High	Add automated PHP linting and integration checks for authentication, ownership boundaries, quotas, upload completion, restore, and download ranges.
High	Handle partial upload cleanup and remote orphan cleanup; provide a reliable retry/resume strategy.
Medium	Add folder rename, move, and delete workflows to the interface, plus quota editing in the admin console.
Medium	Add database backup and restore guidance and a storage integrity audit.
Medium	Add audit logging for administrator actions and important file lifecycle operations.
Later	Consider replacing Telegram storage with a dedicated object-storage service if scale, privacy, or operational guarantees require it.

Verification note

This report explains the application design from the project files and setup instructions. It does not claim a production security audit, load test, or successful deployment verification. The site owner should test the deployed configuration and provider limits before relying on VedDrive for important data.

08 / AUTHOR

Credits and project links

VedDrive was created by

Vedhant Khajuria

Creator and Full-Stack Developer

Project

https://veddrive.vedhant.in

GitHub profile

github.com/vedhant-khajuria

LinkedIn profile

linkedin.com/in/vedhantkhajuriaofficial

Project summary

VedDrive is a self-hosted personal cloud drive combining PHP, MySQL, browser JavaScript, and Telegram-based chunk storage. It provides account access, a responsive file library, folders, search, favorites, Trash, administrator controls, and deployment instructions for cPanel.

Technology and implementation summary

Layer	Technologies / responsibilities
Frontend	HTML, CSS, JavaScript; responsive layouts and client-side upload slicing.
Backend	PHP; session and CSRF handling, validation, user-scoped API operations, file streaming.
Database	MySQL with PDO and InnoDB tables for users, folders, files, chunks, settings, and login attempts.
Remote storage	Telegram Bot API document upload and file retrieval for chunk payloads.
Hosting	cPanel-compatible PHP/MySQL setup with cURL and PDO MySQL extensions.

Prepared on 29 September 2026. This report intentionally omits passwords, bot tokens, OAuth secrets, setup keys, and other private deployment values.