**Non-Functional requirements For Zapsync**

* All files must be encrypted before uploading and decrypted only by authorized users.
* OAuth 2.0 must be used for user authentication, with role-based access control (RBAC)
* AI must flag sensitive data before sharing.
* Uploads and downloads must be optimized for minimal latency (<2s delay).
* The system should automatically scale using AWS S3 or similar cloud storage solutions.
* AI models should operate asynchronously to avoid slowing down file sharing.
* The system should remain available at all times with minimal downtime.
* Database and cloud storage should be backed up daily.
* The web and mobile app must work on all screen sizes.
* The system must ensure user data privacy per global regulations.
* All file-sharing activities should be logged for security auditing.un

**Functional User Requirements for ZapSync**

* The user must be able to “create an account” using a valid email and password.

* The user must be able to log in using their registered email and password.

* The user must be able to log out securely from their session.

* The user should be able to “reset their password” via a password recovery process.

* The user should be able to “view and edit their personal profile”, including full name and profile picture.

* The user must be able to \*upload files\* (e.g., documents, images) to the cloud-based system.

* The user must be able to \*view a list of uploaded files\* in their personal dashboard.

* The user must be able to \*download their own files\* from the system.

* The user must be able to \*delete their own files\* from the system.

* The user must be able to \*share files with other registered users\*.

* The user must be able to \*set sharing permissions\* (e.g., view-only or edit access) for each shared file.

* The user should be able to \*receive smart sharing recommendations\* (e.g., suggested permissions based on file type or past behavior).

* The user must be able to \*view a history of their actions\* (uploads, downloads, shares, deletions).

* The user should be able to \*filter activity logs\* by date, action type, or file name.

* The user must have access to a \*personalized dashboard\* to manage their files and activities.

* The user should be able to navigate the system easily via a \*simple and intuitive user interface\*.

* The user must be able to access Zap Sync from both \*desktop and mobile devices\*.

* The user must be able to create a \*strong password\* that meets minimum security criteria (e.g., length, special characters).

* The user should be automatically logged out after a \*period of inactivity\* for security reasons.

* The user must not be able to \*access files belonging to other users\* unless explicitly shared.

**Functional System Requirement For ZapSync**

* The system must allow users to register using a valid email and password.

* The system must securely hash and store user passwords in the MySQL database.

* The system must authenticate users using the provided login credentials and issue a session token (e.g., JWT).

* The system must validate and enforce password complexity rules (minimum length, alphanumeric, special characters).

* The system should provide a password reset mechanism (e.g., via email link).
* The system must store user profile data (e.g., name, email, profile image, registration date) in the database.

* The system should allow users to update their personal profile information.

* The system must log users out and invalidate session tokens when requested.

* The system must allow users to upload files via the frontend and store them securely in cloud or local storage.

* The system must store file metadata (file name, file type, upload timestamp, file size, owner ID) in MySQL.

* The system must allow users to download their own uploaded files securely via the API.

* The system must delete files and related metadata when a user requests deletion.

* The system must validate files during upload (e.g., max file size, allowed file types).

* The system must allow users to share files with other registered users by generating secure shareable links or direct sharing via email/username.

* The system must enforce sharing permissions (e.g., view-only, edit, download).

* The system should suggest permissions based on smart AI recommendations (e.g., based on file sensitivity or past sharing patterns).

* The system must prevent unauthorized users from accessing or modifying shared files.

* The system must log all user actions (file uploads, downloads, deletions, shares) to a secure activity log table.

* The system must provide an API to allow users to retrieve their activity history.

* The system should allow users to filter logs by file, action type, or date range.

* The system must implement \*JWT token-based authentication\* for securing API routes.

* The system must protect all API endpoints against unauthorized access.

* The system must ensure data encryption at rest (stored files) and in transit (HTTPS for APIs).

* The system should enforce rate limiting and input sanitization to prevent common attacks (e.g., SQL injection, XSS).

* The system should comply with basic data protection guidelines (e.g., GDPR principles).

* The system should handle \*concurrent file uploads\* without performance degradation.

* The system must provide fast file download speeds for users under normal load.

* The system must process anomaly detection tasks asynchronously to avoid slowing down the main application.

* The system must be compatible with modern browsers (Chrome, Firefox, Edge, Safari).

* The system should be mobile-responsive and work on smartphones and tablets.

* The system must be compatible with ZeroTier VPN networking.