Workshop No. 1

Project Definition and Database Modeling

Kevin Santiago Avella Torres – 2021202096

Juan Esteban Oviedo Sandoval – 20192020064

Cristian Camilo Tuso Mozo – 20201020053

Databases II

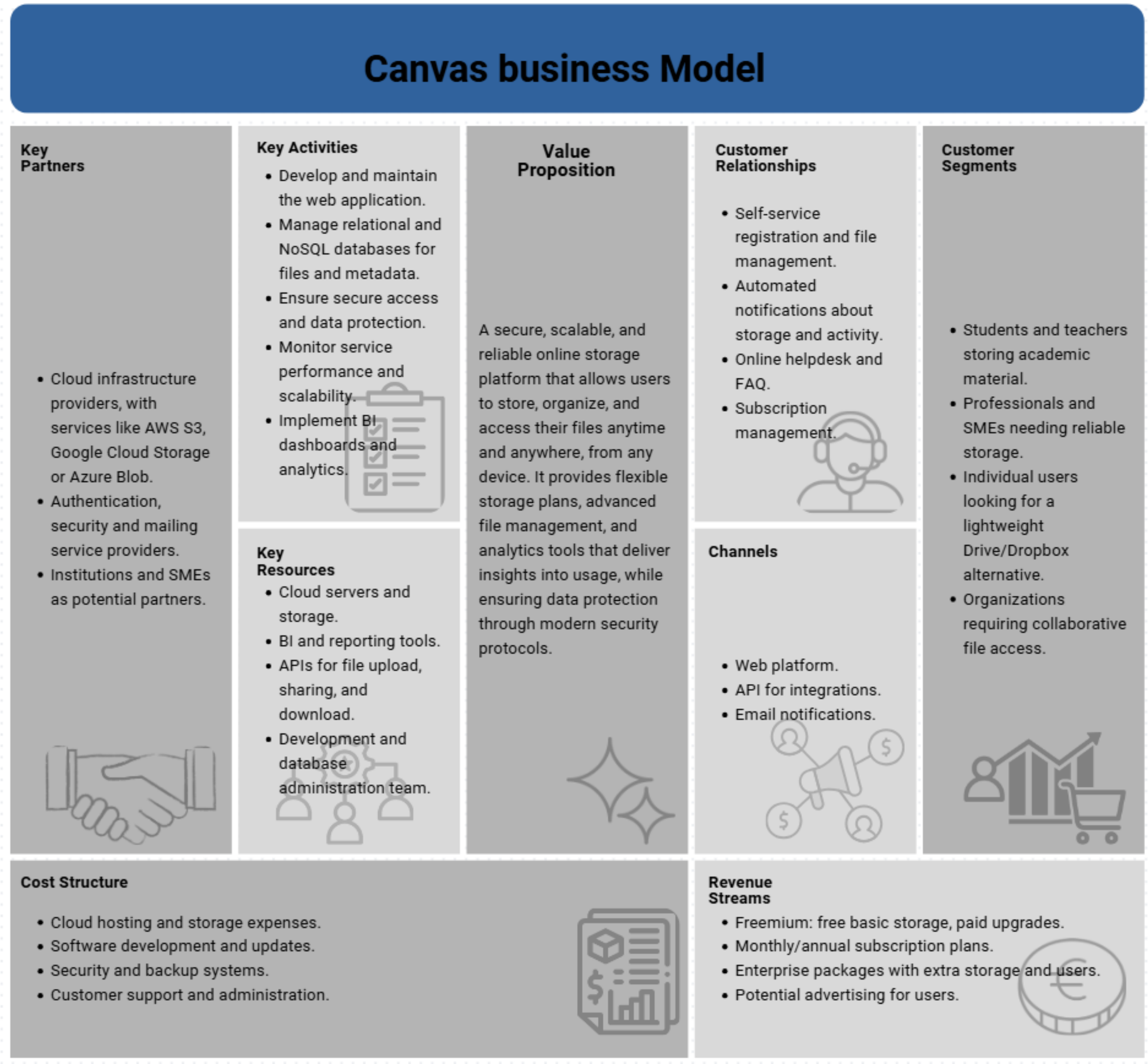
Computer Engineering Program

Universidad Distrital Francisco José de Caldas

1 Business Model Canvas

1.1 Desing Business Model

The following model provides a structured framework to define the key aspects of the proposed file storage platform. Identifies the value proposition, customer segments, partners, resources, and revenue streams, providing a clear overview of how the system creates and delivers value to its users.



Canvas business Model

Key Partners

- Cloud infrastructure providers, with services like AWS S3, Google Cloud Storage or Azure Blob.
- Authentication, security and mailing service providers.
- Institutions and SMEs as potential partners.

Key Activities

- Develop and maintain the web application.
- Manage relational and NoSQL databases for files and metadata.
- Ensure secure access and data protection.
- Monitor service performance and scalability.
- Implement BI dashboards and analytics.

Key Resources

- Cloud servers and storage.
- BI and reporting tools.
- APIs for file upload, sharing, and download.
- Development and database administration team.

Value Proposition

A secure, scalable, and reliable online storage platform that allows users to store, organize, and access their files anytime and anywhere, from any device. It provides flexible storage plans, advanced file management, and analytics tools that deliver insights into usage, while ensuring data protection through modern security protocols.

Customer Relationships

- Self-service registration and file management.
- Automated notifications about storage and activity.
- Online helpdesk and FAQ.
- Subscription management.

Channels

- Web platform.
- API for integrations.
- Email notifications.

Customer Segments

- Students and teachers storing academic material.
- Professionals and SMEs needing reliable storage.
- Individual users looking for a lightweight Drive/Dropbox alternative.
- Organizations requiring collaborative file access.

Cost Structure

- Cloud hosting and storage expenses.
- Software development and updates.
- Security and backup systems.
- Customer support and administration.

Revenue Streams

- Freemium: free basic storage, paid upgrades.
- Monthly/annual subscription plans.
- Enterprise packages with extra storage and users.
- Potential advertising for users.

Figure 1: Business Model Canvas for the File Storage Platform.

2 Requirements Documentation

2.1 Functional Requirements

- FR1. Allow users to register with their details and relevant information (email address, personal details, and other details to be agreed upon).
- FR2. Allow already registered users to log in, taking into account their log-in credentials (email and password).
- FR3. Allow users to manage their storage space using folders, which must have specific options (view, organize, delete, move). Similarly, it must be possible to navigate between folders by hierarchical level according to the user's organization, listing the items available in each of the folder layers.
- FR4. Users must be able to upload files to their workspace, organizing, and separating items as they wish into folders. Possible events during file upload must be guaranteed, indicating whether the files have been uploaded correctly or whether errors have occurred during upload. The valid file types for user upload (.docx, .pdf, .xls, .mp4, .mp3, .png, .jpg, .gif, .pptx) and the maximum upload size per file is 100MB.
- FR5. Users should be able to view the current capacity of their storage space. Highlight how much space is currently occupied and how much space is still available.
- FR6. Allow the user to view important data about the files they have in their storage space, including file name, size, root or file type, and upload date. In addition, users should also be allowed to download files available in their storage space. However, Allow users to manage their files (delete, move, hide, and organize).
- FR7. The system must maintain a traceability history of uploaded, deleted, and moved files. More importantly, it must sign each uploaded file as the property of an available and active account. In this way, priority must be given to ensuring that each account only has access to the files it owns; under no circumstances should unauthorized access to files from other accounts be permitted.
- FR8. The system must implement secure password recovery (email with temporary token).
- FR9. The system must validate all file accesses through session control and, optionally, generate temporary signed URLs for downloads.
- FR10. The system must allow two plan options: The basic plan, which will be free for all accounts created and is acquired by default upon registration. This free plan will have limited trial storage space. The second plan will be the premium plan, which differs in that it increases storage capacity. Users can access the free plan by making a set payment to purchase premium membership, which will have a predetermined value and be valid for six months only. Users can only have one active premium membership. The system will also validate membership status and purchased storage capacity, ensuring that the limits set in either plan are not exceeded.

2.2 Non-Functional Requirements

Non-functional requirements establish the quality attributes and operational constraints of the file storage platform. They define essential aspects such as security, scalability, performance, and usability, ensuring that the system is not only functional but also reliable, efficient, and adaptable to future needs.

- NFR1. Security. The system must guarantee the confidentiality, integrity, and availability of user information. All passwords must be stored using encryption algorithms, and sensitive operations like login, upload, and download must be performed over secure protocols (HTTPS). Unauthorized access to user files is strictly prohibited.
- NFR2. Scalability. The platform must be able to support a growing number of users, files, and transactions without performance degradation. Horizontal and vertical scaling strategies must be considered to ensure continuous expansion of service.

- NFR3. Availability. The system must ensure 24/7 availability with minimal downtime. Redundancy and failover mechanisms should be implemented to guarantee continuous operation.

- NFR4. Performance. The system must provide fast response times for core operations such as logging in, uploading / downloading files, and browsing. Queries and file retrieval should be optimized to handle concurrent requests efficiently.

- NFR5. Usability. The user interface must be intuitive, clear, and accessible for different user profiles. Navigation and file management should require minimal training.

- NFR6. Maintainability. The system must be easy to maintain, update, and extend. Clear modular design and proper documentation should allow developers to implement changes or fix issues quickly.

- NFR7. Portability / compatibility. The platform must run correctly across different browsers like Chrome, Firefox, Edge, Safari, and devices. Otherwise, mobile responsiveness must be guaranteed.

- NFR8. Reliability / Fault Tolerance. The system must tolerate failures without compromising user data. Mechanisms such as replication and distributed storage must ensure that files are never lost due to single-point failures.

- NFR9. Backup and recovery. The platform must include regular automatic backups and allow recovery procedures in case of data corruption, accidental deletion, or system failures.

- NFR10. Auditability / Logging. The system must log all relevant operations (login attempts, file uploads, deletions, downloads) to support traceability and security auditing.

- NFR11. Interoperability. The platform must expose APIs that allow integration with third-party applications, enabling users to connect their storage service with external tools, e.g., learning management systems or enterprise platforms.

3 User Stories

AS is common in Agile methodologies the estimates are given in Story Points (SP) using a simplified Fibonacci-like scale (1, 2, 3, 5, 8).

<table>
<tr><th>Title</th><th>Priority</th><th>Estimate</th></tr>
<tr><td>UH-01: User Registration</td><td>Must-Have</td><td>8 SP</td></tr>
<tr><td colspan="3">As a new user, I want to register for an account by providing my email, password, and personal details, so that I can have my own private storage space on the platform.</td></tr>
<tr><td colspan="3">Given I am a new user on the registration page, when I enter a valid email, a secure password, and my required personal details and submit the form, then my account is created, and I receive an email confirmation.</td></tr>
</table>

<table>
<tr><th>Title</th><th>Priority</th><th>Estimate</th></tr>
<tr><td>UH-02: User Login</td><td>Must-Have</td><td>3 SP</td></tr>
<tr><td colspan="3">As a registered user, I want to log in using my email and password, so that I can securely access my files and storage space.</td></tr>
<tr><td colspan="3">Given I am a registered user on the login page, when I enter my correct email and password and click ”Login”, then I am authenticated and redirected to my personal dashboard.</td></tr>
</table>

<table>
<tr><th>Title</th><th>Priority</th><th>Estimate</th></tr>
<tr><td>UH-03: Secure Password Recovery</td><td>Must-Have</td><td>5 SP</td></tr>
<tr><td colspan="3">As a user who forgot my password, I want to request a password reset via email with a temporary token, so that I can regain access to my account without compromising security.</td></tr>
<tr><td colspan="3">Given I am on the login page and click ”Forgot Password”, when I enter my email address and submit the request, then I receive an email with a secure, temporary link to create a new password.</td></tr>
</table>

<table>
<tr><th>Title</th><th>Priority</th><th>Estimate</th></tr>
<tr><td>UH-04: Upload a File</td><td>Must-Have</td><td>8 SP</td></tr>
<tr><td colspan="3">As a logged-in user, I want to upload a file (e.g., .pdf, .jpg, .mp3) up to 100MB to a specific folder, so that I can store and organize my content in the cloud.</td></tr>
<tr><td colspan="3">Given I am in my workspace and have navigated to the desired folder, when I drag-and-drop or select a valid file for upload, then the system uploads the file and displays a success message; if there is an error, it clearly informs me.</td></tr>
</table>

<table>
<tr><th>Title</th><th>Priority</th><th>Estimate</th></tr>
<tr><td>UH-05: Create and Navigate Folders</td><td>Must-Have</td><td>5 SP</td></tr>
<tr><td colspan="3">As a user, I want to create new folders and navigate through my folder hierarchy, so that I can organize my files logically and find them easily.</td></tr>
<tr><td colspan="3">Given I am viewing the contents of my current folder, when I click ”New Folder”, provide a name, and confirm, then the new folder appears in my current view, and I can click on it to navigate inside.</td></tr>
</table>

<table>
<tr><th>Title</th><th>Priority</th><th>Estimate</th></tr>
<tr><td>UH-06: View File List and Details</td><td>Must-Have</td><td>5 SP</td></tr>
<tr><td colspan="3">As a user, I want to see a list of my files with their name, size, type, and upload date, so that I can get an overview of my stored content and its properties.</td></tr>
<tr><td colspan="3">Given I am logged in and viewing a folder, when the page loads, then I see a list of all items in that folder, displayed in a table or grid with the relevant columns.</td></tr>
</table>

Title	**Priority**	**Estimate**
UH-07: Download a File	Must-Have	3 SP
As a user, I want to download a file from my storage space, so that I can access a local copy on my device.		
Given I am viewing the list of my files, when I click the ”Download” button next to a file, then the file download begins immediately via a secure, temporary URL.		

Title	**Priority**	**Estimate**
UH-08: Delete a File or Folder	Must-Have	5 SP
As a user, I want to delete files or folders I no longer need, so that I can free up storage space and keep my workspace tidy.		
Given I have selected one or more files/folders in my workspace, when I click the ”Delete” button and confirm the action, then the items are moved to a ”Trash” (or permanently deleted with a warning), and the storage space is updated.		

Title	**Priority**	**Estimate**
UH-09: View Storage Capacity	Must-Have	8 SP
As a user, I want to see a visual indicator of my used and available storage space, so that I can manage my uploads and avoid running out of space.		
Given I am logged into my dashboard, when the page loads, then I see a clear display (e.g., a bar chart or text) showing ”X MB used of Y MB available”.		

Title	**Priority**	**Estimate**
UH-10: Move Files/Folders	Should-Have	8 SP
As a user, I want to move files and folders from one location to another within my storage, so that I can reorganize my content without having to re-upload it.		
Given I have selected a file or folder, when I select the ”Move” action and choose a destination folder from my hierarchy, then the item is moved to the new location and removed from the original one.		

Title	**Priority**	**Estimate**
UH-11: Session Control and Security	Should-Have	5 SP
As a system security manager, I want all file accesses to be validated through session control, so that unauthorized users cannot access my files.		
Given a user is not logged in or their session has expired, when they try to access a direct file URL or an API endpoint, then the system denies the request and redirects them to the login page.		

Title	**Priority**	**Estimate**
UH-12: File Activity History	Could-Have	13 SP
As a user, I want to view a history of my recent actions (uploads, deletions, moves), so that I can track changes and recover from mistakes.		
Given I am on my dashboard, when I navigate to an ”Activity” or ”History” section, then I see a chronological list of my file operations with timestamps.		

Title	**Priority**	**Estimate**
UH-13: API for Third-Party Integration	Could-Have	13 SP
As a developer or power user, I want to use a well-documented API to interact with my files, so that I can integrate the storage service with other applications (e.g., an LMS).		
Given I have a valid API token, when I send a GET request to the ‘/api/files‘ endpoint, then I receive a JSON list of the files in my root directory.		

4 Initial Database Architecture

4.1 High-Level Architecture Proposal

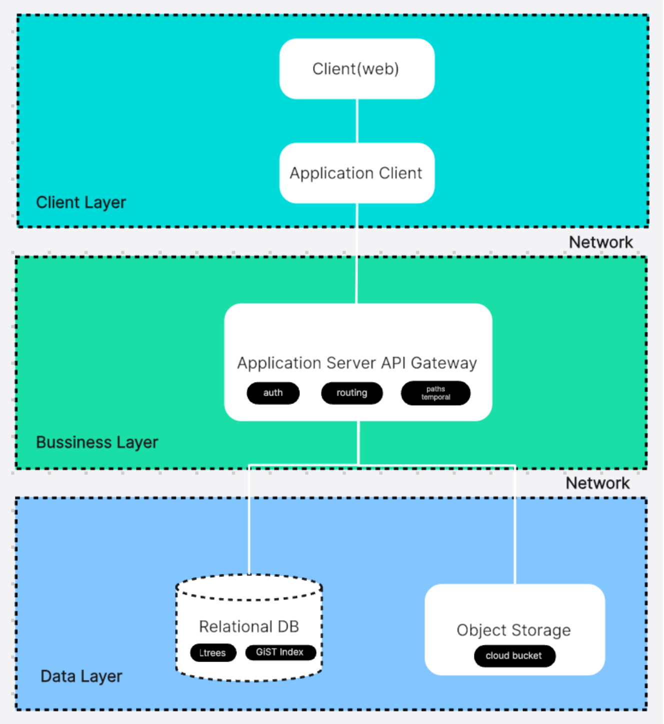
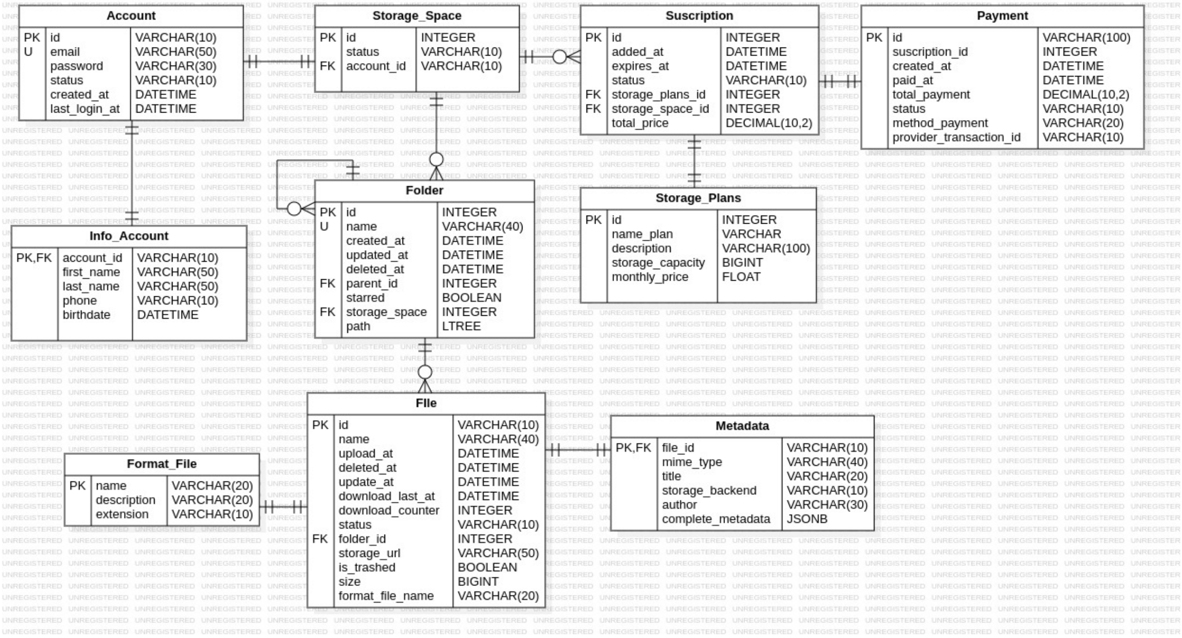


Figure 2: Architecture High Level Model for the File Storage Platform.

4.2 Entity Relationship Diagram - First Version



Account

PK	id	VARCHAR(10)
U	email	VARCHAR(50)
	password	VARCHAR(30)
	status	VARCHAR(10)
	created_at	DATETIME
	last_login_at	DATETIME

Storage_Space

PK	id	INTEGER
	status	VARCHAR(10)
FK	account_id	VARCHAR(10)

Suscription

PK	id	INTEGER
	added_at	DATETIME
	expires_at	DATETIME
	status	VARCHAR(10)
FK	storage_plans_id	INTEGER
FK	storage_space_id	INTEGER
	total_price	DECIMAL(10,2)

Payment

PK	id	VARCHAR(100)
	suscription_id	INTEGER
	created_at	DATETIME
	paid_at	DATETIME
	total_payment	DECIMAL(10,2)
	status	VARCHAR(10)
	method_payment	VARCHAR(20)
	provider_transaction_id	VARCHAR(10)

Folder

PK	id	INTEGER
U	name	VARCHAR(40)
	created_at	DATETIME
	updated_at	DATETIME
	deleted_at	DATETIME
FK	parent_id	INTEGER
	starred	BOOLEAN
FK	storage_space	INTEGER
	path	LTREE

Info_Account

PK,FK	account_id	VARCHAR(10)
	first_name	VARCHAR(50)
	last_name	VARCHAR(50)
	phone	VARCHAR(10)
	birthdate	DATETIME

Storage_Plans

PK	id	INTEGER
	name_plan	VARCHAR
	description	VARCHAR(100)
	storage_capacity	BIGINT
	monthly_price	FLOAT

FIle

PK	id	VARCHAR(10)
	name	VARCHAR(40)
	upload_at	DATETIME
	deleted_at	DATETIME
	update_at	DATETIME
	download_last_at	DATETIME
	download_counter	INTEGER
	status	VARCHAR(10)
FK	folder_id	INTEGER
	storage_url	VARCHAR(50)
	is_trashed	BOOLEAN
	size	BIGINT
	format_file_name	VARCHAR(20)

Metadata

PK,FK	file_id	VARCHAR(10)
	mime_type	VARCHAR(40)
	title	VARCHAR(20)
	storage_backend	VARCHAR(10)
	author	VARCHAR(30)
	complete_metadata	JSONB

Format_File

PK	name	VARCHAR(20)
	description	VARCHAR(20)
	extension	VARCHAR(10)

Figure 3: First version of the entity relationship diagram for the file storage platform.

4.2.1 Description of entities

- The **Account** entity represents the user account within the system. It contains main fields such as *id*, *email*, *password*, *status*, *created_at*, and *last_access*. Its role is to be the starting point for all user information, since both personal settings and storage space are derived from it.

- **Info_Account** complements the account with personal data: *first_name*, *last_name*, *phone*, and *birthdate*. It is directly associated with Account and serves to store user identification information.

- The **Storage_Space** entity manages the storage space allocated to each account. It includes *id*, *status*, and a foreign key *account_id*. Its role is to serve as a link between the user and their storage subscriptions.

- **Subscription** stores the data for an active subscription: *id*, *added_at*, *expires_at*, *status*, *total_price*, along with references to the contracted plan and storage space. Its function is to record the conditions of use of the service for a given period.

- The **Payment** entity represents the payments made for a subscription. It contains *id*, *created_at*, *paid_at*, *total_payment*, and *method_payment*. It allows you to keep track of the charges associated with each service contract.

- The **Folder** entity organizes files into hierarchies. Its most important fields are *id*, *name*, *created_at*, *updated_at*, *deleted_at*, *parent_id*, and *starred*. It represents directories that can contain files and subfolders.

- **File** stores user file information. It includes *id*, *name*, *created_at*, *deleted_at*, *download_counter*, *size*, and *folder_id*. It is responsible for representing the digital content within the folders.

- The **Format_File** entity describes the file format. It contains *name*, *description*, and *extension*. Its function is to identify the file type and its basic characteristics.

- Finally, **Metadata** complements the file with specific information such as *mime_type*, *title*, *author*, *storage_backend*, and *complete_metadata*. It serves to expand the technical and descriptive details of each file.

4.2.2 Description of Relationships between Entities

- Account is related to Info_Account, as each account has associated personal information.

- Account is linked to Storage_Space, indicating the storage space allocated to each user.

- Storage_Space is connected to Subscription, which defines the terms of the contracted service.

- Subscription is associated with Payment, which allows the payments for each subscription to be recorded.

- Folder is organized hierarchically by its parent_id, and in turn contains multiple Files.

- File is related to Format_File, to define the file type, and to Metadata, which expands its descriptive information.

4.3 Data Flow and Storage Solutions

Here describes the overall data flow of the platform and the adopted storage solutions. It explains how information moves through the system and how files and metadata are managed across different storage layers.

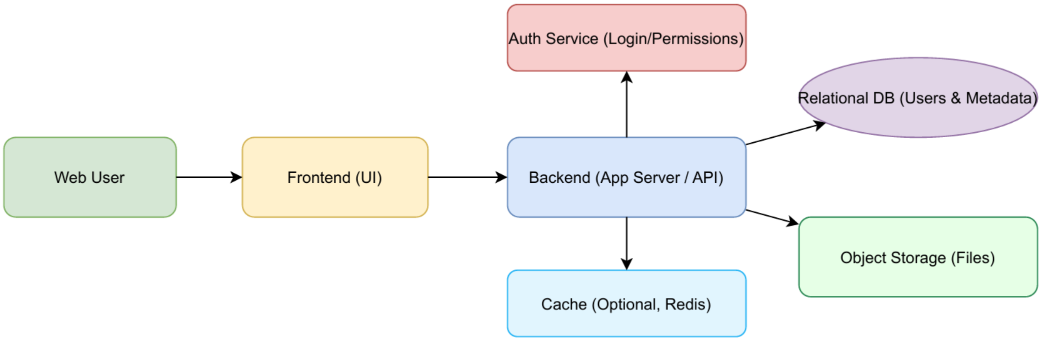

Auth Service (Login/Permissions)
Relational DB (Users & Metadata)
Web User
Frontend (UI)
Backend (App Server / API)
Object Storage (Files)
Cache (Optional, Redis)

Figure 4: Data Flow and Storage Solutions for the file storage platform.

For the dataflow, it proposed system manages two primary data types: user information and files. The data flow begins when a user interacts with the platform through the web or mobile interface. Requests are sent to the backend application server, which validates user credentials and permissions through the authentication service. Once authenticated, the backend processes the request according to its type:

- For file uploads, the binary object is stored in the cloud object storage, while metadata, e.g., file name, size, type, upload date, and owner is recorded in the relational database.
- For file downloads, the backend verifies user access rights and generates a temporary signed URL, allowing secure file retrieval from the object storage.
- For queries such as listing files or browsing folders, the backend retrieves metadata from the database and may leverage a cache layer to accelerate frequent lookups.

This flow ensures that every operation is validated, traceable, and optimized for performance while maintaining strict access control.

Regarding storage solutions, its design adopts a hybrid model:

- Relational Database: like PostgreSQL or MySQL is used for structured information such as user accounts, roles, permissions, and file metadata. This allows enforcing relationships and maintaining consistency.
- Object Storage: e.g., AWS S3, Azure Blob is used for binary files, ensuring scalability, durability, and redundancy. This approach allows efficient handling of large files with virtually unlimited capacity.
- Cache Layer: for supports faster access to frequently requested metadata, reducing database load.
- Backup and Replication strategies: ensure high availability and disaster recovery, minimizing the risk of data loss.

This architecture separates file storage from metadata management, providing scalability and flexibility. It also ensures that the platform can handle a growing number of users and files without compromising security, reliability, or performance.