# **Technical Documentation: PDF Form Manager with Google Login**

## **1. Overview**

The PDF Form Manager is a PHP-based system that extracts variables from a predefined PDF template, dynamically generates a form for user input, validates data, and stores it in a database. Users authenticate via Google Login to view, edit, delete, and download documents with version control.

## **2. System Architecture**

* **Frontend**: HTML, CSS, JavaScript (jQuery/React for dynamic UI)
* **Backend**: PHP (Laravel or Core PHP)
* **Database**: MySQL
* **Authentication**: Google OAuth 2.0
* **PDF Processing**: TCPDF / FPDF
* **Server**: Apache/Nginx
* **Storage**: Local server storage for PDFs

## **3. Functional Modules**

### **3.1 User Authentication (Google OAuth 2.0)**

* Uses Google OAuth for secure authentication.
* Upon login, stores user details in the database.
* Manages session handling.

### **3.2 Form Generation from PDF Template**

* Extracts variables (images, headings, paragraphs) from a predefined template.
* Generates dynamic input fields based on extracted content.
* Ensures input validation and error handling.

### **3.3 PDF Generation & Version Control**

* Stores user inputs in the database.
* Generates a new PDF version upon form submission.
* Stores multiple versions with timestamps.
* Provides download links for each version.

### **3.4 User Dashboard**

* Displays uploaded documents with:
  + **Document Title**
  + **PDF Download Button**
  + **View/Edit/Delete options**
* Allows users to edit fields and generate a new PDF version.
* Allows deletion of documents along with all associated versions.

## **4. Database Schema**

### **4.1 Users Table**

| **Column** | **Type** | **Description** |
| --- | --- | --- |
| id | INT (PK) | Unique user ID |
| google\_id | VARCHAR | Google OAuth ID |
| name | VARCHAR | User's name |
| email | VARCHAR | User's email |

### **4.2 Documents Table**

| **Column** | **Type** | **Description** |
| --- | --- | --- |
| id | INT (PK) | Unique document ID |
| user\_id | INT (FK) | User who uploaded |
| title | VARCHAR | Document title |
| created\_at | TIMESTAMP | Upload timestamp |

### **4.3 Document Versions Table**

| **Column** | **Type** | **Description** |
| --- | --- | --- |
| id | INT (PK) | Unique version ID |
| document\_id | INT (FK) | Related document ID |
| version\_no | INT | Version number |
| pdf\_path | TEXT | Filepath to stored PDF |
| updated\_at | TIMESTAMP | Timestamp of version creation |

## **5. API Endpoints**

### **5.1 User Authentication**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /auth/google | Initiate Google Login |
| GET | /auth/callback | Handle OAuth Callback |
| GET | /logout | Log out user |

### **5.2 Document Handling**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /documents | List user documents |
| POST | /documents/create | Upload new document |
| GET | /documents/{id} | View document details |
| POST | /documents/{id}/edit | Edit document and create new version |
| DELETE | /documents/{id} | Delete document and versions |

### **5.3 PDF Handling**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /pdf/download/{id}/{version} | Download PDF version |

## **6. Error Handling & Security Measures**

* **Form validation** to prevent incorrect inputs.
* **SQL Injection Protection** via prepared statements.
* **File upload security** (only PDF, size limit, MIME-type check).
* **Session Handling** using secure Google OAuth.
* **Version control** to prevent accidental data loss.

## **7. User Flow**

1. **User logs in** via Google OAuth.
2. **Dashboard loads** with a list of uploaded documents.
3. User **adds a new document** by filling a dynamically generated form.
4. On **successful validation**, a **PDF is generated and saved**.
5. User can **view, edit, or delete** documents.
6. If **edited**, a **new PDF version** appears.
7. If **deleted**, all associated PDFs are removed.

## **8. Future Enhancements**

* Add support for more **social logins** (Microsoft, LinkedIn, etc.).
* Implement **role-based access control**.
* Enable **bulk document uploads & exports**.
* Introduce **document sharing & collaboration features**.

### **📌 Next Steps:**

* **Confirm feature priorities** and finalize UI design.
* **Set up the database & backend structure**.
* **Develop & test the form builder and PDF generator**.
* **Integrate Google Login & error handling**.

Let me know if any modifications are needed! 🚀