

Assessment Report

Full-Stack Software Engineer Assessment

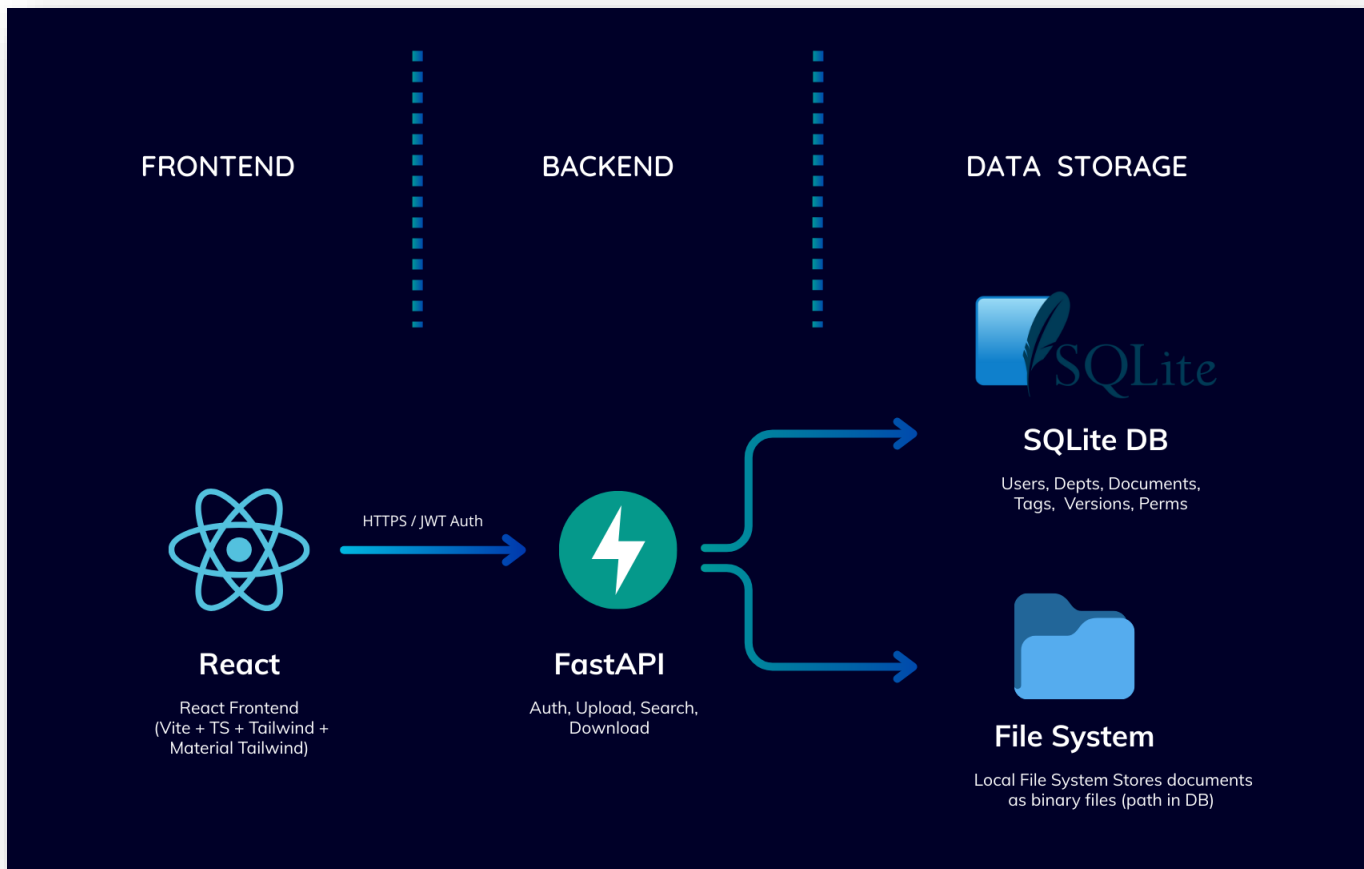
SIEMENS

Table of Contents

Deliverable 1:	3
System Architecture:	3
.....	3
Design Choices:	3
Deliverable 2:	4
Schema:	4
Backend overview and constraints:	4
SQL Queries	8
1. Get all documents accessible to a user in their department.	8
2. Get the 10 most recently uploaded documents tagged as Finance.	8
3. Find all versions of a given document (by document ID).	9
4. Get the number of documents uploaded by each department in the last 30 days.....	9
Proof of Concept (POC) Project.....	10
Backend (API) Endpoints:	10
API Documentation:	10
Frontend:	11
Figma UI Design	11
Pages:	12
Deliverable 3:	18
GitHub Repo Link:	18
Video Link:	18

Deliverable 1:

System Architecture:

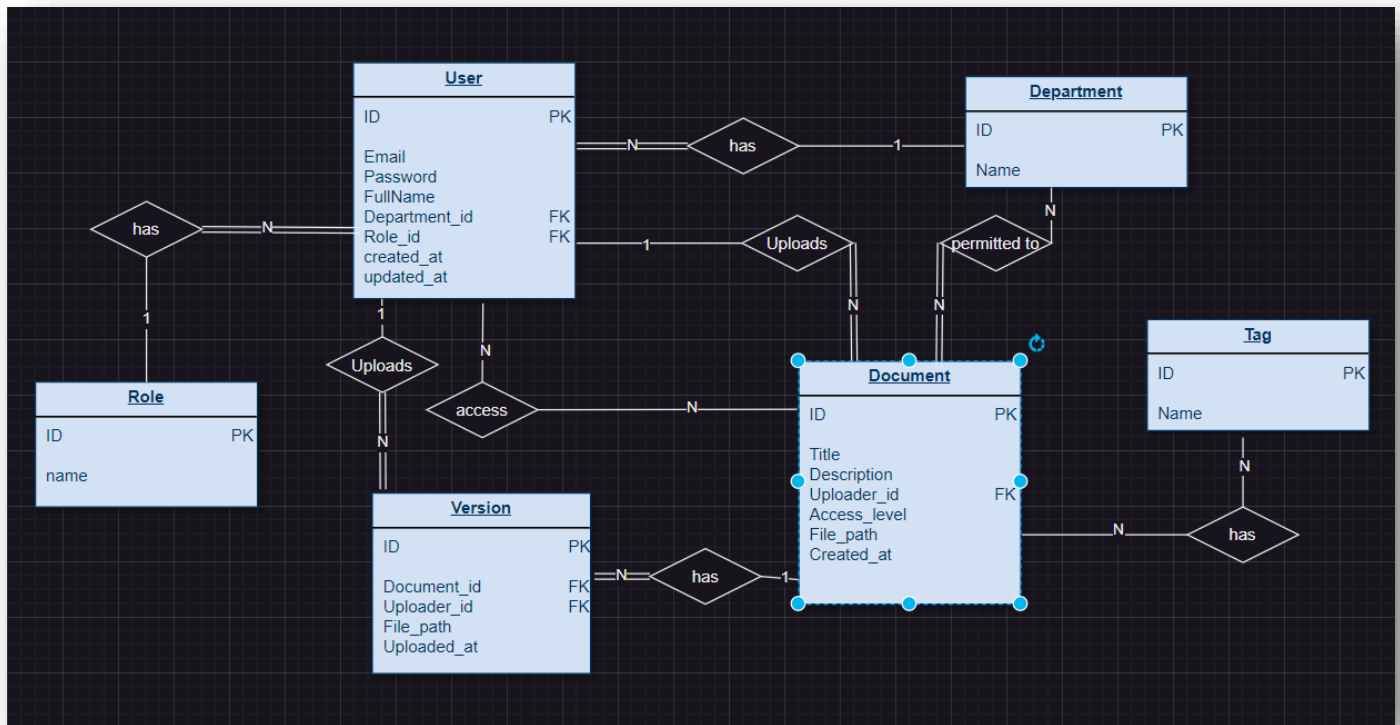


Design Choices:

- **React + TS + Tailwind** → modern, flexible, rapid prototyping ,clean UI and components based architecture.
- **JWT Auth** → stateless, secure, well-suited for SPAs.
- **FastAPI** → lightweight, async-first, automatic API docs.
- **SQLite** → minimal setup, good for development & testing (we can then migrate to **PostgreSQL** in production)
- **Local FS** → free, no extra dependencies (we can then migrate to cloud based storage for production (**e.g. Amazon S3**) to be more reliable and scalable)

Deliverable 2:

Schema:



Backend overview and constraints:

Access Control Model

The system enforces **multi-layered access control** to documents:

1. **User Roles (global) (admin & user)** (we can add super admin in the future)
2. **Department Membership**
3. **Document-Level Permissions**

Each layer refines who can view documents.

1. User Roles

- Users are assigned one role (users.role_id → roles table).
- **Default roles identified in the system:**

- **Admin:** Full access to all documents and operations (create/update/delete users, departments, roles).
- **User:** Can create and manage documents they own and documents accessible to their department or explicit permissions.
- **Role enforcement:**
 - Certain endpoints require `get_current_admin_user` to authorize admin-only operations.
 - Role-based restrictions are applied globally before checking document-specific permissions.

2. Department Membership

- Each user belongs to a department (`users.department_id` → `departments`).
- Documents can have an `access_level`:
 - **public:** Accessible to all authenticated users.
 - **department:** Only users in the same department as the uploader can access.
 - **private:** Restricted to the creator user or explicit users via document-level permissions.
- Departments can also be granted **document-level overrides** via the `DocumentDepartmentPermission` table.
- Example:
 - A document with `access_level='department'` uploaded by Finance is visible to Finance department users.
 - Legal department can be explicitly granted access via `DocumentDepartmentPermission`.

3. Document-Level Permissions

- **Explicit granted control** through two tables:
 1. **DocumentUserPermission:**
 - Maps `user_id` → `document_id` with permission.
 - Overrides general access rules for specific users.
 2. **DocumentDepartmentPermission:**
 - Maps `department_id` → `document_id` with permission.
 - Overrides general department-based access.
- Example:
 - Document marked as private:

- Only users in document_user_permissions or the uploader can access it.
- Document marked as department:
 - Users in the department can access by default.
 - Additional departments can be granted access through document_department_permissions.

4. Access Decision Flow

When a user requests a document (can_access_document helper):

1. **Check global role:**
 - If user.role.name == "admin" → grant full access.
2. **Check document access_level:**
 - public → allow.
 - department → check if user.department_id == document.uploader.department_id.
 - If not, check explicit document_department_permissions.
 - private → only uploader or explicit document_user_permissions can access.
3. **Explicit permissions overrides:**
 - Check DocumentUserPermission for user-specific access.
 - Check DocumentDepartmentPermission for department-specific access.
4. **Otherwise** → deny access.

5. Document Versions

- **Versioning implemented via DocumentVersion table:**
 - Each document can have multiple versions.
 - Versions are tied to the uploader (uploaded_by) and maintain version_number.
 - Download and inline preview respect the same access control rules (can_access_document is reused).

6. Tags

- Documents can be tagged (Tag and DocumentTagLink tables).
- Tag-based search is integrated with access control (only returns documents the user can access).

8. File Storage & Access

- Uploaded documents stored in Documents/ folder with unique UUID filenames.
- Versions have separate files with version UUID.
- Downloads use FileResponse and respect access control.
- Inline viewing is supported via /documents/{doc_id}/file or /versions/{version_id}/file.

9. Security Notes

- Authentication via JWT (OAuth2PasswordBearer).
- Passwords hashed using hash_password utility.
- Token expiration handled via settings.ACCESS_TOKEN_EXPIRE_MINUTES.
- Admin-only operations explicitly enforced via get_current_admin_user.
- All file access checks through can_access_document to prevent unauthorized downloads.

SQL Queries

1. Get all documents accessible to a user in their department.

```
1  -- 1. Get all documents accessible to a user in their department. -----
2
3  -- Only documents from the user's department.
4  -- Skip private documents as they are private to other users.
5  -- Include documents that are either:
6  --     uploaded by someone in the same department and marked as "department" or "public", OR
7  --     explicitly shared with the department in document_department_permissions.
8
9  SELECT DISTINCT d.*
10 FROM documents d
11 JOIN users u ON u.id = :user_id
12 LEFT JOIN users uploader ON uploader.id = d.uploader_id
13 LEFT JOIN document_department_permissions ddp ON ddp.document_id = d.id AND ddp.department_id = u.department_id
14 WHERE d.access_level != 'private'
15 AND (
16     (d.access_level = 'department' AND uploader.department_id = u.department_id)
17     OR ddp.id IS NOT NULL
18 );
```

2. Get the 10 most recently uploaded documents tagged as Finance.

```
1  -- 2. Get the 10 most recently uploaded documents tagged as Finance.-----
2
3  -- a) If we mean document creation order:
4  SELECT d.*
5  FROM documents d
6  JOIN document_tags dt ON dt.document_id = d.id
7  JOIN tags t ON t.id = dt.tag_id
8  WHERE t.name = 'Finance'
9  ORDER BY d.created_at DESC
10 LIMIT 10;
11
12 --b) If we instead mean last uploaded versions in general even if there are more than one version per document:
13 SELECT d.*
14 FROM documents d
15 JOIN document_tags dt ON dt.document_id = d.id
16 JOIN tags t ON t.id = dt.tag_id
17 JOIN document_versions v ON v.document_id = d.id
18 WHERE t.name = 'Finance'
19 ORDER BY v.uploaded_at DESC
20 LIMIT 10;
```


3. Find all versions of a given document (by document ID).

```
1  -- 3. Find all versions of a given document (by document ID). -----
2
3      SELECT v.*
4      FROM document_versions v
5      WHERE v.document_id = :doc_id
6      ORDER BY v.version_number ASC;
```

4. Get the number of documents uploaded by each department in the last 30 days.

```
1  -- 4. Get the number of documents uploaded by each department in the last 30 days. -----
2
3      SELECT d.department_id, COUNT(*) AS document_count
4      FROM documents d
5      WHERE d.created_at >= NOW() - INTERVAL '30 days'
6      GROUP BY d.department_id
7      ORDER BY document_count DESC;
```

Proof of Concept (POC) Project

Backend (API) Endpoints:

All the required endpoints [User registration & login, Upload document (store file + metadata), Search documents (by title, tags, uploader), View/download document, Fetch version history of a document.] are implemented and more endpoints.

All the required endpoints are integrated with the frontend and shown in the video.

API Documentation:

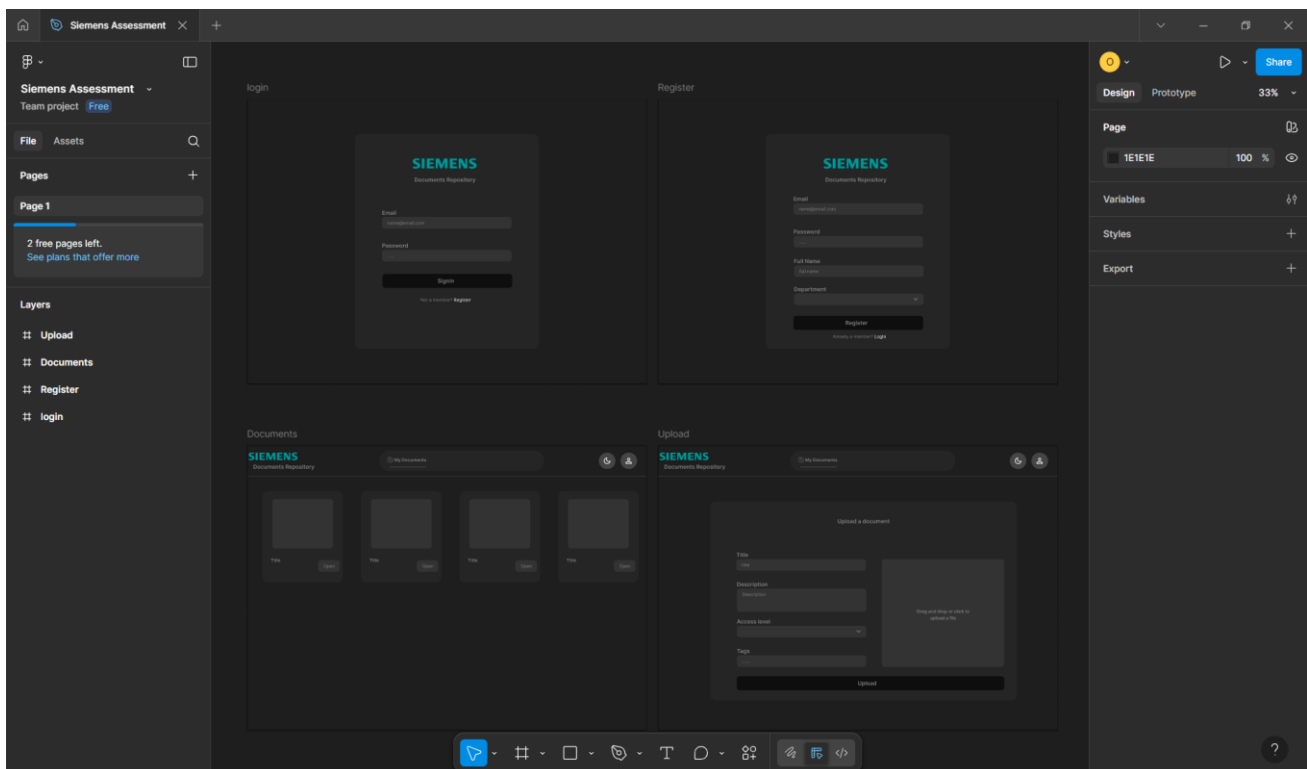
Authentication			^
POST	/auth/register	Register User	▼
POST	/auth/login	Login User	▼
GET	/auth/me	Read Users Me	🔒 ▼
Users			^
GET	/users/	List Users	🔒 ▼
POST	/users/	Create User	🔒 ▼
GET	/users/{user_id}	Get User	🔒 ▼
PATCH	/users/{user_id}	Update User	🔒 ▼
DELETE	/users/{user_id}	Delete User	🔒 ▼
Roles			^
GET	/roles/	List Roles	🔒 ▼
POST	/roles/	Create Role	🔒 ▼
DELETE	/roles/{role_id}	Delete Role	🔒 ▼
Documents			^
GET	/documents/	List Documents	🔒 ▼
POST	/documents/	Create Document	🔒 ▼
GET	/documents/my	List My Documents	🔒 ▼
GET	/documents/search	Search Documents	🔒 ▼
GET	/documents/{doc_id}/download	Download Document	🔒 ▼
GET	/documents/{doc_id}/file	Get Document File	🔒 ▼
GET	/documents/by-path/{path}	Get Document By Path	🔒 ▼
GET	/documents/file/{doc_id}	Get File	🔒 ▼
GET	/documents/{doc_id}	Get Document	🔒 ▼
PATCH	/documents/{doc_id}	Update Document	🔒 ▼
DELETE	/documents/{doc_id}	Delete Document	🔒 ▼
POST	/documents/{doc_id}/versions	Upload Version	🔒 ▼
GET	/documents/{doc_id}/versions	List Versions	🔒 ▼
GET	/documents/versions/{version_id}	Get Version	🔒 ▼

DELETE	/documents/versions/{version_id}	Delete Version	🔒
GET	/documents/{doc_id}/versions/{version_number}/download	Download Version	🔒
GET	/documents/versions/{version_id}/file	Get Version File	🔒
Departments			
GET	/departments/	List Departments	🔒
POST	/departments/	Create Department	🔒
DELETE	/departments/{department_id}	Delete Department	🔒
Permissions			
GET	/permissions/users	List User Permissions	🔒
POST	/permissions/users	Add User Permission	🔒
GET	/permissions/departments	List Department Permissions	🔒
POST	/permissions/departments	Add Department Permission	🔒
Tags			
GET	/tags/	List Tags	🔒
POST	/tags/	Create Tag	🔒
POST	/tags/attach/{document_id}	Attach Tag To Document	🔒
GET	/tags/document/{document_id}	Get Document Tags	🔒
DELETE	/tags/detach/{document_id}/{tag_id}	Detach Tag From Document	🔒

Frontend:

Figma UI Design

Started with Figma design for pages layout and choosing the color palette for themes.

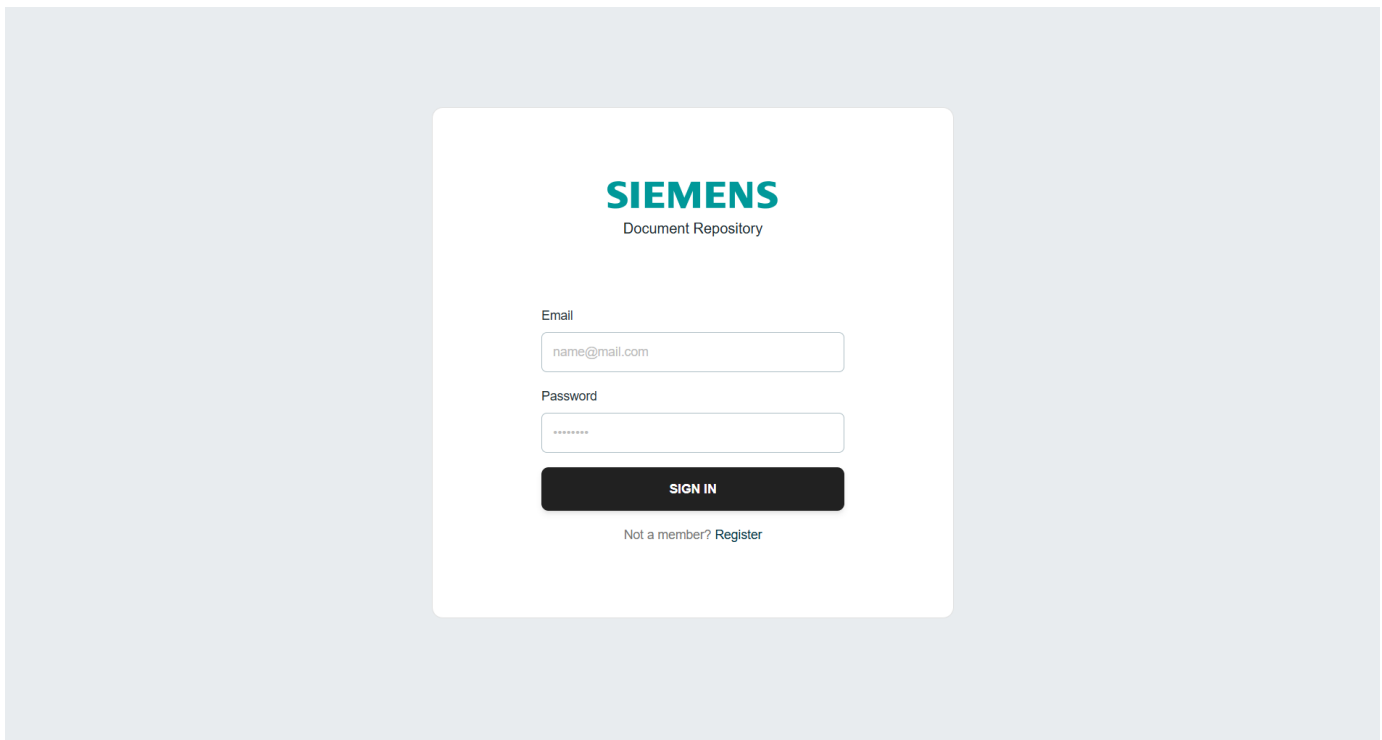


Note:

All the frontend components and pages are fully responsive to handle different screen sizes and have dark and light themes handled

Pages:

1. Login Page

The image shows a login page for the SIEMENS Document Repository. It features a white card centered on a light gray background. At the top of the card is the SIEMENS logo in teal, followed by the text "Document Repository". Below this are two input fields: "Email" with the placeholder "name@mail.com" and "Password" with a masked password "*****". A dark gray "SIGN IN" button is positioned below the password field. At the bottom of the card, there is a link that says "Not a member? Register".

SIEMENS
Document Repository

Email
name@mail.com

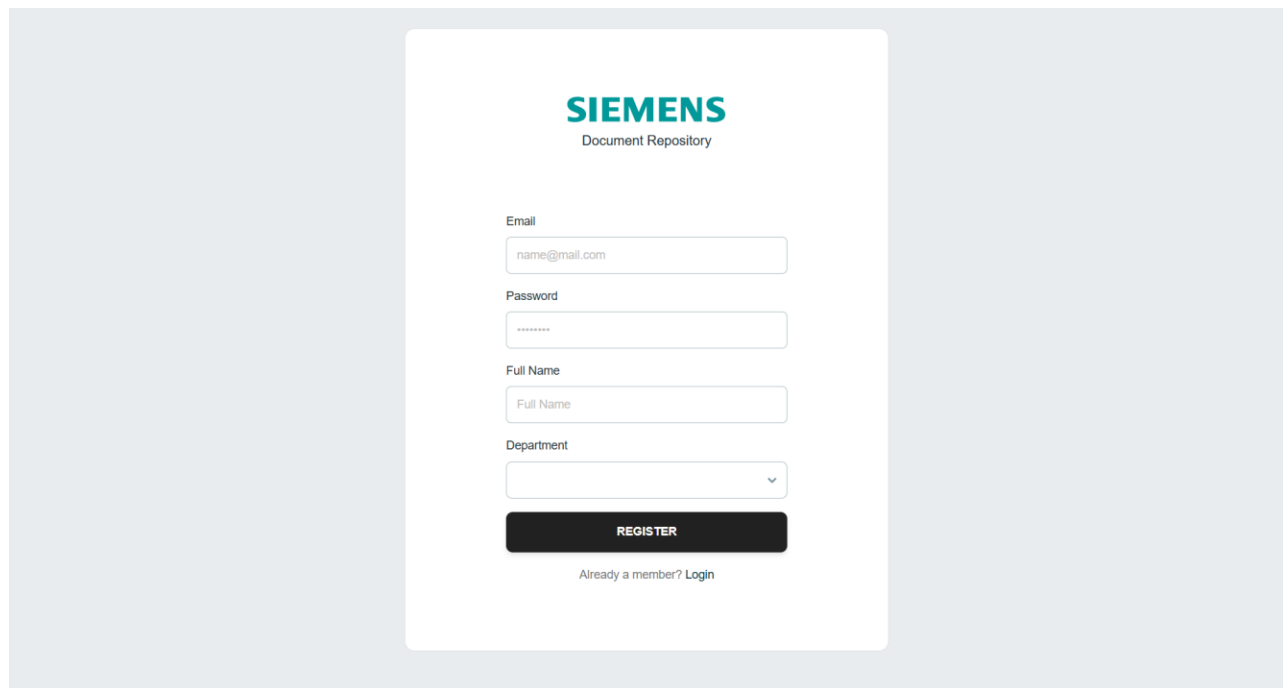
Password

SIGN IN

Not a member? [Register](#)

- Form with email + password.
- On submit: sends credentials → gets token → saves to localStorage.
- Handles loading + error (“invalid email/password”).
- Link to register page.

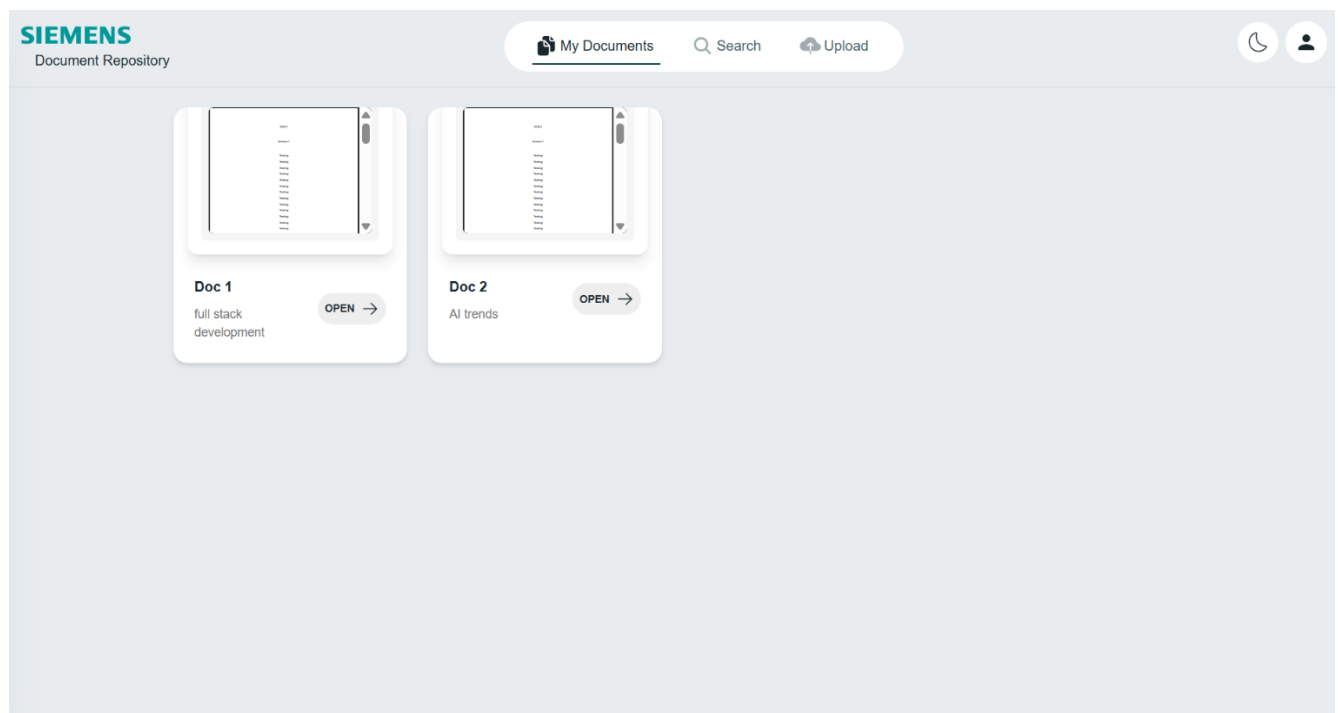
2. Register Page



The image shows a registration form for the SIEMENS Document Repository. The form is centered on a light gray background. It features the SIEMENS logo at the top, followed by the text "Document Repository". Below this, there are four input fields: "Email" (with the placeholder "name@mail.com"), "Password" (with a masked password "*****"), "Full Name" (with the placeholder "Full Name"), and "Department" (a dropdown menu). A black "REGISTER" button is positioned below the input fields. At the bottom, there is a link that says "Already a member? Login".

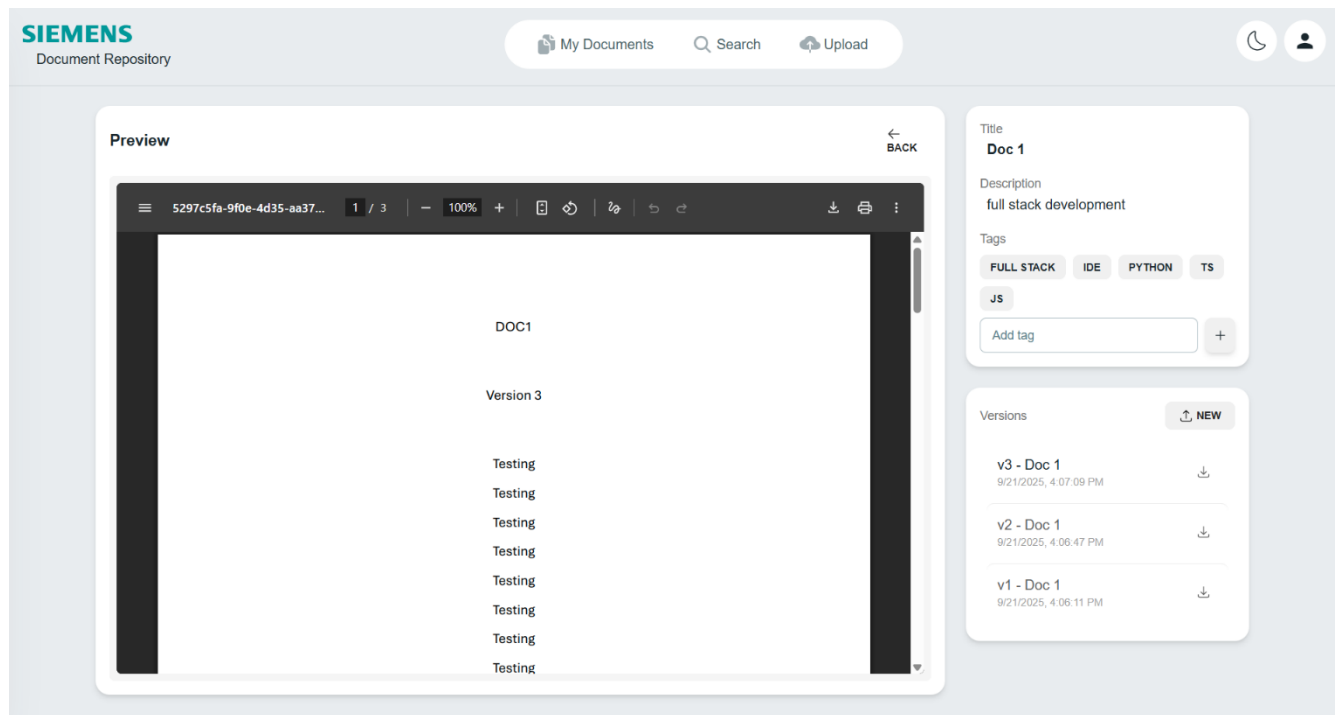
- Form with email, password, full name, and department selector.
- Fetches department list from backend.
- On submit: creates user → stores token → redirects to home.
- Handles loading + errors.
- Link to login page.

3. Documents Page



- Lists user's documents in a grid.
- Each card: preview (image/pdf/placeholder), title, description, and "Open" button.
- Handles loading, empty state, and failed previews.

4. Document Detail Page:



- Shows one document with metadata, tags, and versions.
- Features:
 - Preview latest version (PDF/image/none).
 - Switch between versions.
 - Download versions and upload new versions.
 - Add new tags.
 - Upload new version (modal).

5. Upload Document Page:

SIEMENS
Document Repository

My Documents Search Upload

Upload Document

Fill in the details on the left and upload your file on the right.

Title
Document Title

Description
Brief description of the document

Access Level
Select access level
Public

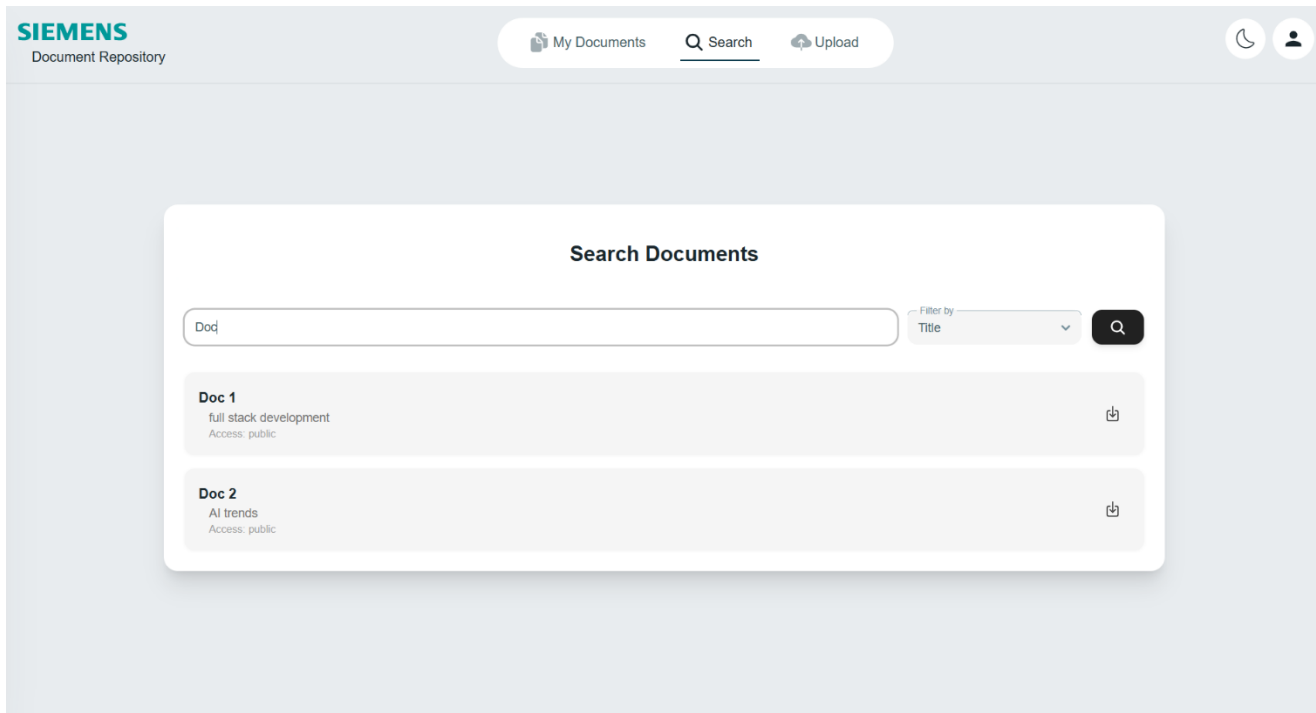
Tags
Type a tag and press Enter

Drag & Drop your file here
or click anywhere to browse

UPLOAD DOCUMENT

- Form with title, description, access level, tags, and file upload.
- Drag & drop or browse file.
- On submit: uploads doc + attaches tags.
- Redirects to detail page after success.

6. Search Page:



- Search bar + filter (title/tags/uploader).
- Paginated results.
- Each result: title, description, access level, open button, quick download latest version.
- Shows loading, empty, and results states.

Deliverable 3:

GitHub Repo Link:

<https://github.com/OsamaElHattab/Document-Repository>

Video Link:

https://drive.google.com/drive/folders/1rhynOEa1RElwSJ5TEJ_YL5ogLGW7vQ80?usp=sharing