

Student Document Management System

System Design

1. Objective

To create a secure, scalable, and maintainable system that allows students to submit their personal and academic documents, enables admins to manage registrations and document approvals, and provides dashboards for all users.

2. Problem Statement

Manual document handling is error-prone, insecure, and inefficient.
This system aims to:

- Digitally manage student registrations & documents.
- Automate approval workflows for admins.
- Securely store & serve files.
- Provide dashboards & notifications.

3. Scope

- Student registration and login
- Profile management
- Upload, resubmit, and download documents
- Admin approval/rejection workflows
- Role-based dashboards
- Secure file storage and access
- Search using filters

4. Technology Stack

Layer	Technology
Frontend	Angular 20, TypeScript
Backend	ASP.NET Core Web API 8, C#
Database	SQL Server, v19.0
File Storage	Local file system

4. Architecture

1. Layered Architecture

- Clear separation of concerns:
Controller → Service → Repository → Storage
- Controllers handle HTTP requests and responses.
- Services contain business logic.
- Repositories manage database operations.
- Storage layer handles file storage (local).

2. Service-Oriented Web API

- Modular, reusable services for Students, Documents, Approvals, and Dashboard.
- Each service encapsulates a specific domain of functionality.
- Enables scalability and easier maintenance.

3. Secure File Serving

- Files are never served via public URLs.
- Download requests pass through API for **authentication and authorization checks**.
- Only owners (students) or authorized admins can access files.

4. Role-Based Authorization

- Roles: **Student**, **Admin**.
- Students can CRUD their profile and own documents.
- Admins can manage students, view and approve/reject documents.
- Enforced via JWT tokens and role guards both in backend and frontend.

6. Backend Design

6.1 Controllers

Controller	Responsibilities
<code>AuthController</code>	Login, Registration, Role Creation
<code>StudentsController</code>	Profile CRUD, list/search students for admin
<code>DocumentsController</code>	Upload, list, download, resubmit
<code>ApprovalsController</code>	Approve/reject students & documents, provide remarks
<code>DashboardController</code>	Admin/student dashboard metrics

6.2 Services

`StudentService` – profile & student listing logic

`DocumentService` – upload, resubmit, download and list documents

`ApprovalService` – approve/ reject students and documents

`DashboardService` – metrics

6.3 Infrastructure

IStorageProvider (Local)

- **Purpose:** Abstracts file storage for flexibility.
- **Implementation:** Local disk
- **Responsibilities:** Save files, generates unique paths, retrieve files, optional versioning/soft-delete.

IFileTypeValidator (Security Hook)

- **Purpose:** Ensures only valid files (e.g., PDFs) are uploaded.
- **Security Role:** Prevents malicious uploads, supports MIME checks, PDF inspection, antivirus hooks.

EF Core Repositories

- **Purpose:** Encapsulate database operations.
- **Responsibilities:** CRUD for Students, Documents, Admins, filtering, sorting, paging, soft delete/versioning.
- **Benefit:** Separates DB access from business logic; easier testing.

JWT Authentication & Role-Based Authorization

- **Authentication:** Users receive JWT with ID and role.
- **Authorization:** Backend checks token and role for each request.
- **Roles:**
 - Students: manage own profile/documents
 - Admins: manage all students/documents

6.4 Security

- **Authentication:** JWT access
- **Role-based Authorization:** Student/Admin
- **File validation:** Client & server-side PDF only, max size enforced

- **Secure download:** No direct URLs, streamed after authZ checks
- **CORS and HTTPS** enforced

7. Backend APIs

Module	Endpoint	Method	Description
Auth	/auth/login	POST	Login, return JWT & role
Student	/students/register	POST	Student registration
	/students/me	GET	Get own profile
	/students/me/updateprofile	PUT	Update profile
	/students/getallstudents	GET	Admin: list/search students
	/students/{id}/approve	PATCH	Admin: approve student
	/students/{id}/reject	PATCH	Admin: reject student
Documents	/documents/upload	POST	Student uploads document
	/documents/mydocuments	GET	Student lists own docs
	/documents/{id}/resubmit	POST	Student resubmits document
	/documents/{id}/download	GET	Download document (student/admin)
	/documents/student/{id}	GET	Admin: list all documents of a student
	/documents/{id}/approve	PATCH	Admin approves document
	/documents/{id}/reject	PATCH	Admin rejects document
Dashboard	/dashboard/admin	GET	Admin metrics
	/dashboard/student	GET	Student metrics
Admin Ops	/admins	POST	Add new admin

7.1 Sample Request and Response

Module	Endpoint	Method	Description	Request Example	Response Example
Auth	/auth/login	POST	Login, return JWT & role	<pre>{ "email": "student@example.com" , "password": "Password123" }</pre>	<pre>{ "token": "jwt_token_here", "role": "Student", "expiresIn": 3600 }</pre>
Student	/students/register	POST	Student registration	<pre>{ "name":"John Doe","email":"john@example.com", "password": "Password123", "dob": "2000-01-01", "courseApplied":"BSc CS" }</pre>	<pre>{ "studentId": 1, "status": "Pending" }</pre>
	/students/me	GET	Get own profile	Header: Authorization: Bearer <token>	<pre>{ "studentId":1,"name":"John Doe","email":"john@example.com", "dob":"2000-01-01" , "courseApplied":"BSc CS", "status":"Pending" }</pre>
	/students/me/updateprofile	PUT	Update profile	<pre>{ "name":"John Doe", "dob":"2000-01-01", "courseApplied":"B Sc CS", "address":"123 Street" }</pre>	<pre>{ "message": "Profile updated successfully" }</pre>
	/students/getallstudents	GET	Admin: list/search students	Header: Authorization: Bearer <admin_token>	<pre>[{ "studentId":1,"name":"John Doe", "email":"john@example.com", "status":"Pending"</pre>

					<code>, "courseApplied": "BSc CS" }]</code>
	<code>/students/{id}/approve</code>	PATCH	Admin: approve student	<code>{ "registerNumber": "REG12345" }</code>	<code>{ "message": "Student approved" }</code>
	<code>/students/{id}/reject</code>	PATCH	Admin: reject student	<code>{ "reason": "Incomplete documents" }</code>	<code>{ "message": "Student rejected" }</code>
Documents	<code>/documents/upload</code>	POST	Student uploads document	Multipart/form-data: file + <code>documentTypeId</code>	<code>{ "documentId": 1, "status": "Pending" }</code>
	<code>/documents/mydocuments</code>	GET	Student lists own documents	Header: <code>Authorization: Bearer <token></code>	<code>[{ "documentId": 1, "fileName": "marksheet.pdf", "status": "Pending", "uploadedOn": "2025-08-18T10:00:00Z" }]</code>
	<code>/documents/{id}/resubmit</code>	POST	Student resubmits document	Multipart/form-data: file	<code>{ "message": "Document resubmitted successfully", "status": "Pending" }</code>
	<code>/documents/{id}/download</code>	GET	Download document	Header: <code>Authorization: Bearer <token></code>	Streams PDF file with <code>Content-Disposition: attachment</code>
	<code>/documents/student/{id}</code>	GET	Admin: list all documents of a student	Header: <code>Authorization: Bearer <admin_token></code>	<code>[{ "documentId": 1, "fileName": "marksheet.pdf", "status": "Pending", "uploadedOn": "2025-08-18T10:00:00Z" }]</code>
	<code>/documents/{id}/approve</code>	PATCH	Admin approves document	<code>{ "remarks": "Verified" }</code>	<code>{ "message": "Document approved" }</code>
	<code>/documents/{id}/reject</code>	PATCH	Admin rejects document	<code>{ "remarks": "Blurry scan" }</code>	<code>{ "message": "Document rejected" }</code>

Dashboard	/dashboard/admin	GET	Admin metrics	Header: Authorization: Bearer <admin_token>	{ "totalStudents":50,"pendingApprovals":5,"uploadedDocuments":120,"incompleteProfiles":8 }
	/dashboard/student	GET	Student metrics	Header: Authorization: Bearer <token>	{ "registrationStatus":"Pending","profileCompletion":80,"pendingDocuments":2,"rejectedDocuments":1,"notifications":3 }
Admin Ops	/admins	POST	Add new admin	{ "name":"Admin One", "email":"admin@example.com", "password":"Password123" }	{ "adminId":1,"message":"Admin added successfully" }

8. Frontend Design (Angular)

8.1 Modules & Components

Module	Components
AdminModule	Dashboard, Student Management, Document Management
StudentModule	Dashboard, Profile Management, My Documents
AuthModule	Login, Register
Shared	Layout, Header, Footer, Guards, Pipes, Directives, Toasts

8.2 Angular Services

- **AuthService**: login, logout, authentication
- **StudentService**: profile CRUD, list/search
- **DocumentService**: upload, download, resubmit, approve/reject
- **DashboardService**: metrics

8.3 Guards

- **AuthGuard**: Protect routes for authenticated users
- **RoleGuard**: Restrict routes by role (Student/Admin)

9. Future Enhancements

- **Audit Log / Activity Tracking** – Track all user and admin actions (login, updates, approvals, rejections).
- **Notification System** – Email/SMS/WhatsApp alerts for application updates, approvals, or missing documents.
- **Cloud Storage Integration** – Store files in Azure/AWS/GCP instead of local server.

Data Flow

Overview

1. Student Registration & Admin Approval

- Student submits registration form via frontend.
- Frontend sends data to Backend API: POST /StudentAccount/register.
- Backend validates input, createsAspNetUsers record with Status = Pending.
- Frontend shows message: "Registration submitted. Await admin approval."
- Admin reviews pending registrations:
 - Approve: Student can log in and upload documents.
 - Reject: Student denied from login.

2. Student Login

- Student can log in even if Status = Pending.
- Backend returns user info and login token.
- Frontend behavior:
 - Approved: full access (profile + document upload).
 - Pending: login allowed but upload blocked; message shown: "Admin approval required."
 - Rejected: login denied.

3. Student Profile & Document Upload (Post-Approval)

- Student fills profile details: FullName, Phone, Address, CourseApplied, etc.
- Student uploads relevant documents (multiple files with DocumentType).
- Frontend calls API: POST /Student/profile-upload (multipart/form-data).
- Backend validates profile info and files.
- Saves files in /Uploads/{studentId}/{yyyy}/{MM}/{GUID}.pdf
- Updates Student table.
- Inserts into Document table (Status = Pending).
- Frontend lists uploaded documents with status (Pending / Approved / Rejected).

4. Admin Document Approval / Rejection

- Admin views pending documents.
- Approves or rejects each document:
 - Approve: Status = Approved.
 - Reject: Status = Rejected + Remarks.
- Frontend displays updated status to student.
- Rejected documents can be resubmitted.

5. Student Resubmission of Rejected Documents

- Student sees rejected documents with remarks.
- Student re-uploads corrected documents.
- Frontend calls API: POST /Student/profile-upload or PUT /Document/resubmit.
- Backend updates Document table with new file path and Status = Pending.
- Frontend updates document list to Pending.

6. Profile Completion Calculation

- Formula: (Number of required documents uploaded & approved / Total required documents) * 100%
- Only approved documents count.
- Frontend shows progress bar or percentage.

7. Summary

Action	Allowed?	Notes
Student login before approval	Yes	Upload blocked; can view registration status
Upload documents before approval	No	Message shown: "Await admin approval"
Upload documents after approval	Yes	Status = Pending initially
Admin approves registration	Yes	Enables upload & profile edit
Admin approves document	Yes	Status = Approved
Admin rejects document	Yes	Status = Rejected + Remarks; student can resubmit
Resubmit rejected document	Yes	Status = Pending

8. Full Data Flow

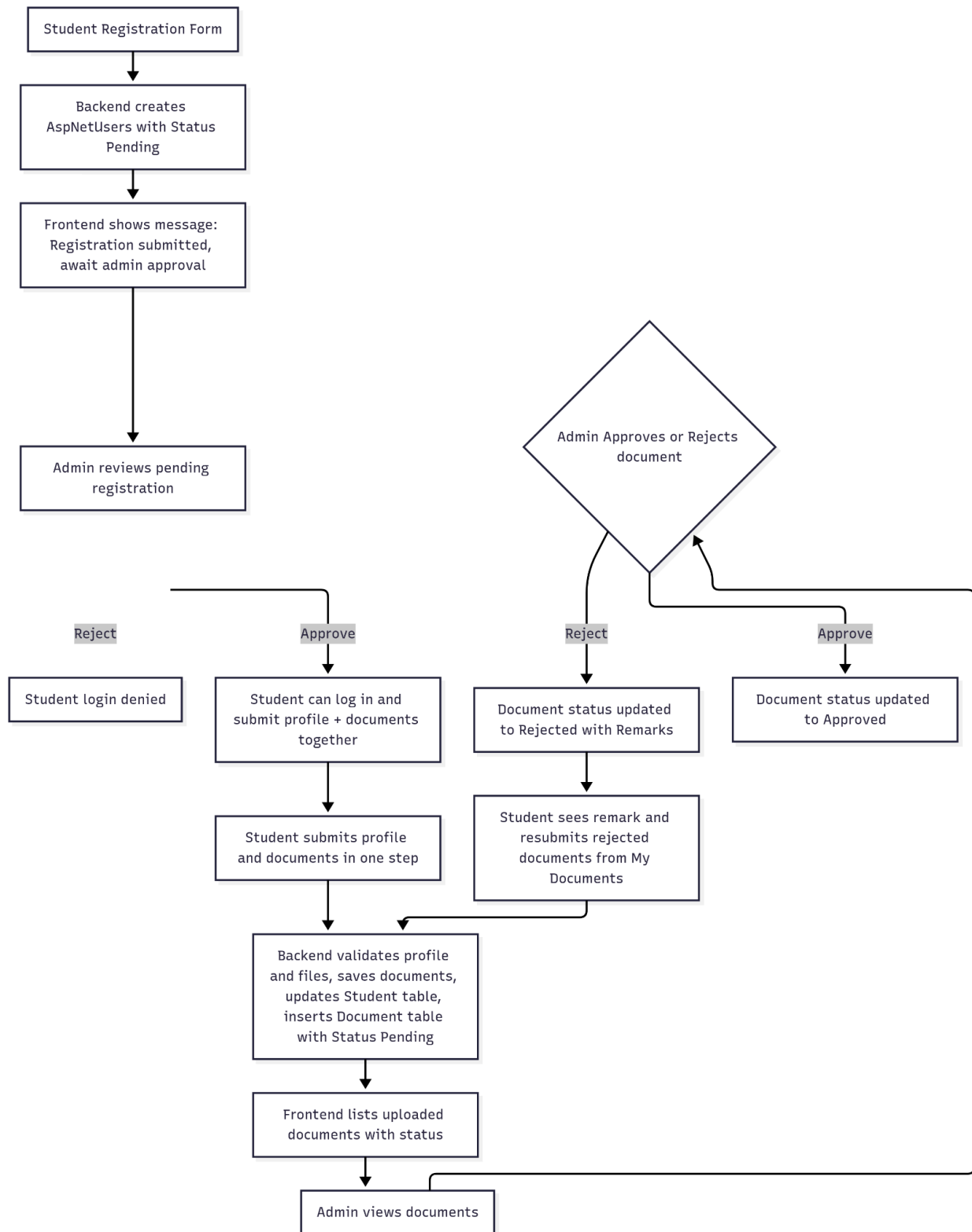
Student Frontend:

- Register → Backend → AspNetUsers (Status = Pending) → Show: "Await Admin Approval"
- Login → Backend (returns status + token)
 - If Approved → Access Profile & Upload
 - If Pending → Show message; upload blocked
 - If Rejected → Show rejection remarks
- Upload Profile + Documents → Backend → Update Student & Document Tables
- List documents (Pending/Approved/Rejected + Remarks)
- Resubmit Rejected Documents → Backend → Document Table (Status = Pending)

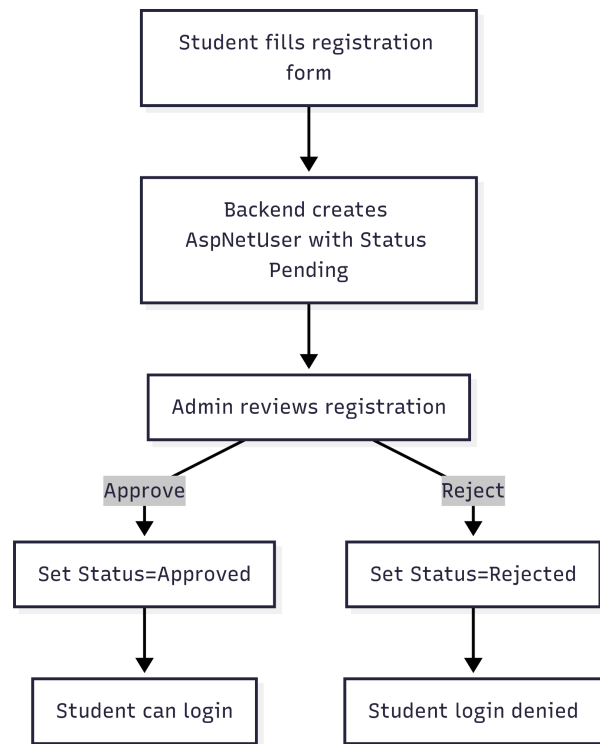
Admin Frontend:

- View Pending Registrations → Backend → AspNetUsers Table
- Approve/Reject Registration → Backend → Update Status + Remarks
- View Pending Documents → Backend → Document Table
- Approve/Reject Documents → Backend → Update Status + Remarks

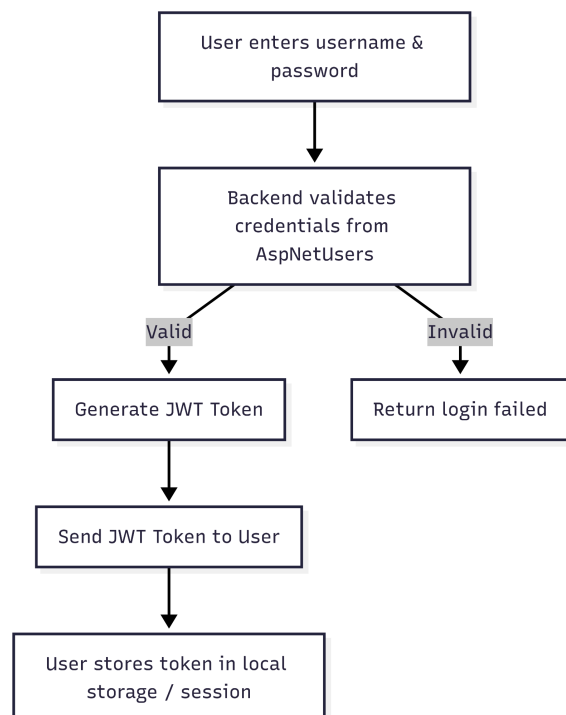
DATA FLOW DIAGRAM



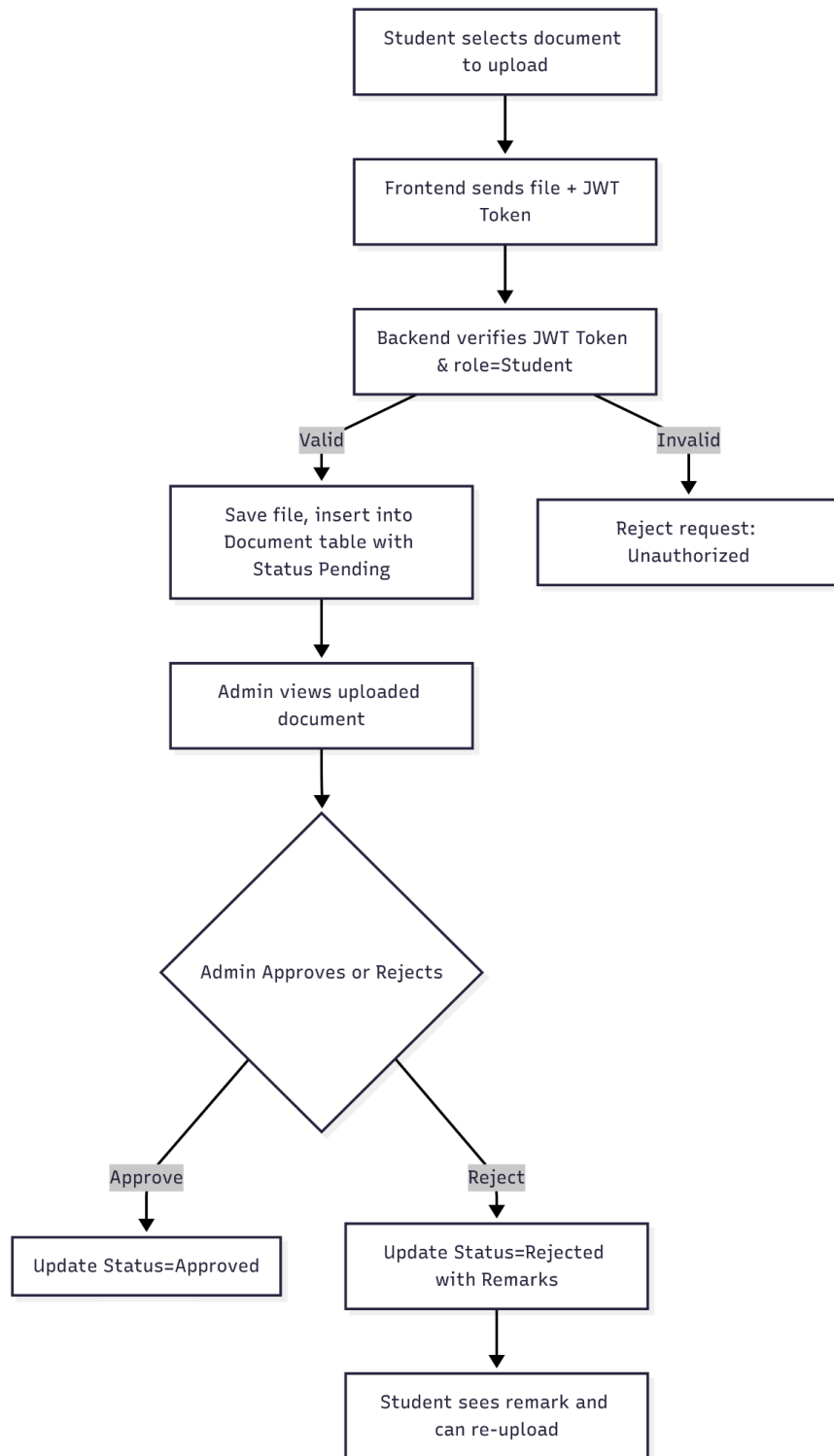
Registration



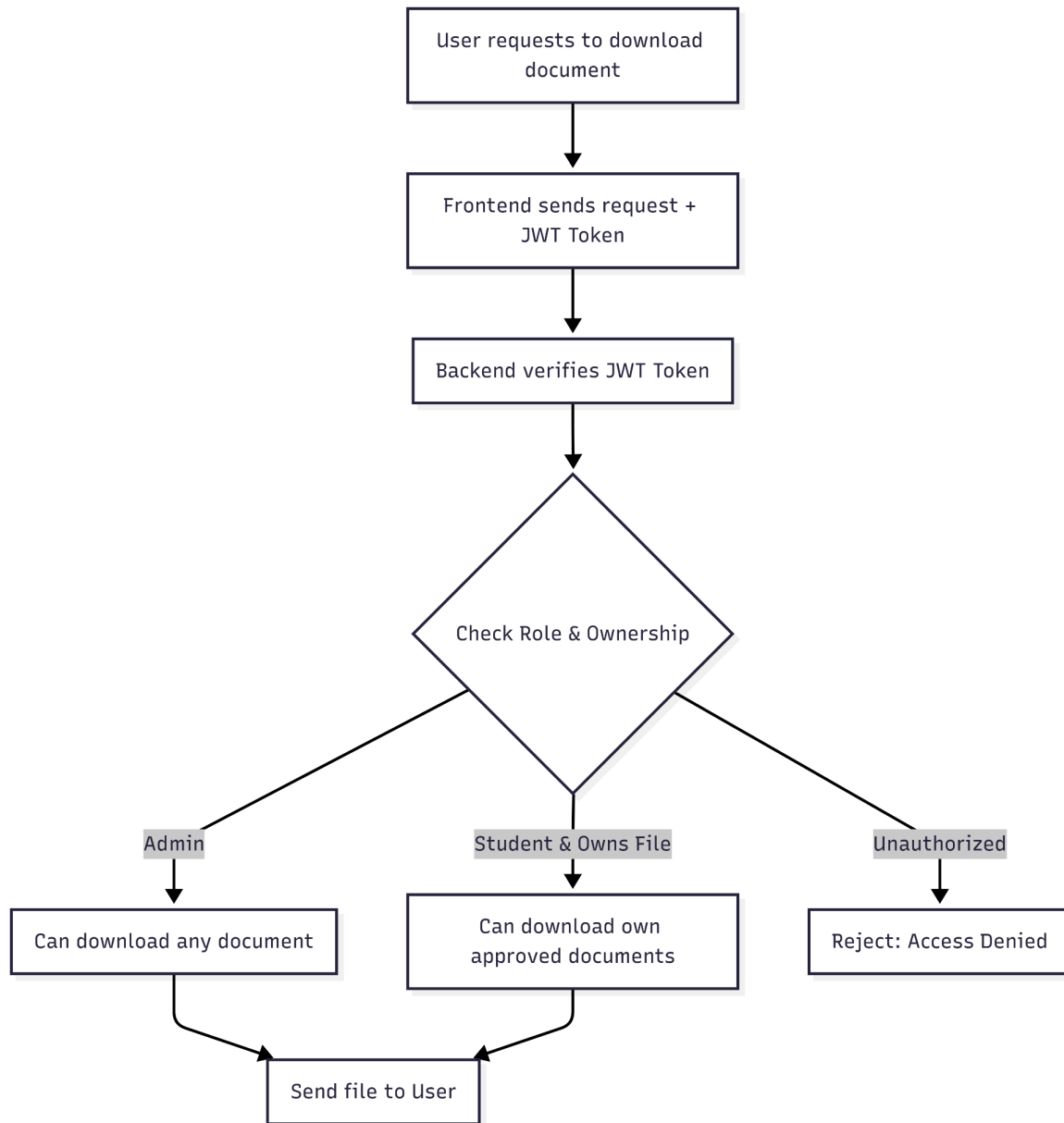
Login



Uploading Documents



Downloading Documents



Database Design

1.Tables

1. AspNetUsers

Stores user account details (from ASP.NET Identity). Extended with FullName and AccountStatusId to track admission account approval status.

2. Student

Holds student-specific profile details (linked to AspNetUsers), including personal info, course selection, application status, and identity proof.

3. StudentEducation

Stores academic history of each student (Class X, XII, UG, PG, etc.) with institute details, marks, and linked documents (marksheets).

4. Document

Contains uploaded student documents (ID proof, certificates, marksheets, etc.), along with file metadata and approval status.

5. Course

Defines available courses for admission (e.g., B.Sc., B.Com., MBA). Linked with Student table for course selection.

6. IdProofType

Reference table listing valid identity proof types (Aadhar, Passport, Driving License, etc.), used in the Student table.

7. Notification

Stores messages/alerts sent to students (application updates, document verification, approvals, etc.) with read/unread status.

8. StatusMaster

Centralized status lookup table (for User Accounts, Applications, Documents). Contains categories like Pending, Approved, Rejected, Changes Needed.

9. AspNetRoles

Stores application roles (e.g., Admin, Student) as part of ASP.NET Identity.

10. AspNetUserRoles

Mapping table linking users to their roles. Composite key: (UserId, RoleId).

2. Columns

AspNetUsers		
string	Id	PK
nvarchar	UserName	
nvarchar	Email	
nvarchar	FullName	
nvarchar	PasswordHash	
int	AccountStatusId	FK
datetime	CreatedOn	

Student		
int	StudentId	PK
string	UserId	FK
nvarchar	RegisterNo	
date	DOB	
nvarchar	Gender	
nvarchar	PhoneNumber	
nvarchar	AlternatePhoneNumber	
nvarchar	Address	
nvarchar	PermanentAddress	
nvarchar	City	
nvarchar	District	
nvarchar	State	
nvarchar	Pincode	
int	IdProofTypeId	FK
nvarchar	IdProofNumber	
int	CourseId	FK
int	ApplicationStatusId	FK
datetime	CreatedOn	
datetime	UpdatedOn	

StudentEducation		
int	EducationId	PK
int	StudentId	FK
nvarchar	EducationLevel	
nvarchar	InstituteName	
int	PassingYear	
decimal	MarksPercentage	
int	DocumentId	FK
datetime	CreatedOn	
datetime	UpdatedOn	

Document		
int	DocumentId	PK
int	StudentId	FK
nvarchar	DocumentType	
nvarchar	FileName	
nvarchar	FilePath	
nvarchar	FileType	
int	FileSizeKB	
int	StatusId	FK
nvarchar	Remarks	
datetime	UploadedOn	
datetime	ApprovedOn	

Course		
int	CourseId	PK
nvarchar	CourseName	

IdProofType		
int	IdProofTypeId	PK
nvarchar	TypeName	

Notification		
int	NotificationId	PK
int	StudentId	FK
nvarchar	Message	
bit	IsRead	
datetime	CreatedOn	

StatusMaster		
int	StatusId	PK
nvarchar	StatusType	
nvarchar	StatusName	

AspNetRoles		
string	Id	PK
nvarchar	Name	

AspNetUserRoles		
string	UserId	FK
string	RoleId	FK

3. Relations

1. AspNetUsers → Student

- One User → One Student (1:1)
 - Each user account (login identity) corresponds to exactly one student profile.
 - `Student.UserId (FK) → AspNetUsers.Id.`

2. AspNetUsers → AspNetUserRoles

- One User → Many UserRoles (1:M)
 - A user can have multiple roles.
 - `AspNetUserRoles.UserId (FK) → AspNetUsers.Id.`

3. AspNetRoles → AspNetUserRoles

- One Role → Many UserRoles (1:M)
 - A role can be assigned to many users.
 - `AspNetUserRoles.RoleId (FK) → AspNetRoles.Id.`
- Meaning: This creates a many-to-many between Users and Roles.

4. AspNetUsers → StatusMaster

- Many Users → One Status (M:1)
 - Each user account has a status (Pending, Approved, Rejected).
 - `AspNetUsers.AccountStatusId (FK) → StatusMaster.StatusId.`

5. Student → StudentEducation

- One Student → Many Educations (1:M)
 - Each student can upload multiple education records (X, XII, UG, PG).
 - `StudentEducation.StudentId (FK) → Student.StudentId`.

6. Student → Document

- One Student → Many Documents (1:M)
 - A student can upload multiple supporting documents.
 - `Document.StudentId (FK) → Student.StudentId`.

7. StudentEducation → Document

- One Education → One Document (1:1)
 - Each education record can be linked to one uploaded document (marksheet).
 - `StudentEducation.DocumentId (FK) → Document.DocumentId`.

8. Document → StatusMaster

- Many Documents → One Status (M:1)
 - Each document has a verification status (Pending, Approved, Rejected).
 - `Document.StatusId (FK) → StatusMaster.StatusId`.

9. Student → Course

- Many Students → One Course (M:1)
 - Each student applies for exactly one course.
 - `Student.CourseId (FK) → Course.CourseId`.

10. Student → IdProofType

- Many Students → One IdProofType (M:1)
 - Each student has one type of ID proof (Aadhar, Passport, etc.).
 - `Student.IdProofTypeId (FK) → IdProofType.IdProofTypeId`.

11. Student → Notification

- One Student → Many Notifications (1:M)
 - A student can receive multiple system notifications.
 - `Notification.StudentId (FK) → Student.StudentId`.

12. Student → StatusMaster

- Many Students → One Application Status (M:1)
 - Each student application has a status (Pending, Approved, Changes Needed).
 - `Student.ApplicationStatusId (FK) → StatusMaster.StatusId`.

UI Prototypes

REGISTER PAGE

REGISTER

Full Name	Email
<input type="text"/>	<input type="text"/>
Password	Confirm Password
<input type="password"/>	<input type="password"/>

SUBMIT

[Already have an account? Login here](#)

LOGIN PAGE

LOGIN

Email

Password

SUBMIT

[Dont have an account? Sign Up here](#)

STUDENT DASHBOARD

Student Pannel		Logout
Dashboard Profile My Documents	<div>Registration Status Pending</div>	
© 2025 Student Document Management system. All Rights Reserved.		

Student Pannel		Logout
Dashboard Profile My Documents	<div>Application Status Changes Required</div> <div>Joel Joseph - ABC2025001</div> <div>Profile Completion Percentage <div>85%</div></div> <div>Notifications Re-upload Id Proof.</div>	
© 2025 Student Document Management system. All Rights Reserved.		

PROFILE

Student Pannel

Logout

Dashboard

Profile

My Documents

1

1

1

Full Name *

DOB

Gender

Email

phone number

Alternate Phno

Adress

Permenant Address

City

District

State

Pin Code

Course

Id-Proof Type

Id-Proof Number

Upload Id-Proof

choose

v

choose

v

Browse

Reset

Save

Next

© 2025 Student Document Management system. All Rights Reserved.

Student Pannel

Logout

Dashboard

Profile

My Documents

1

1

1

Class X

Class XII

Passing Year

Institute Name

Marks In %

Upload Certificate

Browse

Reset

Passing Year

Institute Name

Marks In %

Upload Certificate

Browse

Reset

Back

Save

Next

© 2025 Student Document Management system. All Rights Reserved.

Student Pannel

Logout

Dashboard

Profile

My Documents

1

1

1

Acknowledgement

☐

I hereby acknowledge that the above database schema details are correct to the best of my knowledge and understanding.

Back

Submit

© 2025 Student Document Management system. All Rights Reserved.

MY DOCUMENTS

Student Pannel					Logout
<div>Dashboard</div> <div>Profile</div> <div>My Documents</div>	View and Edit Your Uploaded Documents				
	Document Type	Status Badge	Admin Remarks	Current File (Preview/Download)	Action
	ID Proof (Aadhar)	<div>Rejected</div>	"Image unclear, please upload a clearer copy"	[View File]	[Upload New]
	SSLC Certificate	<div>Approved</div>	—	[View File]	Disabled
	HSC Marksheet	<div>Pending Review</div>	"Awaiting verification"	[View File]	Disabled
	Extra Curricular	<div>Rejected</div>	"Wrong certificate uploaded"	[View File]	[Upload New]
	© 2025 Student Document Management system. All Rights Reserved.				

ADMIN DASHBOARD

Admin Pannel

Logout

Dashboard

Approve Student Req

Approve Documents

Doc Management

Pending Student Approval

12

Pending Document Verification

20

Approved Students

120

Students with incomplete profile

10

© 2025 Student Document Management system. All Rights Reserved.

APPROVE STUDENT REQUEST

Admin Pannel

Logout

Dashboard

Approve Student Req

Approve Documents

Doc Management

SL NO	Student Name	Email	Approve/Reject
1	Joel Joseph	joeljoseph@gmail.com	<div>Approve</div> <div>Reject</div>

© 2025 Student Document Management system. All Rights Reserved.

APPROVE DOCUMENTS

Admin Pannel

Logout

Dashboard

Approve Student Req

Approve Documents

Doc Management

SL NO	Student Name	Register Number	
1	Joel Joseph	ABC2025001	<div>Profile</div> <div>Document</div>

© 2025 Student Document Management system. All Rights Reserved.

Admin Pannel

Logout

Dashboard

Approve Student Req

Approve Documents

Doc Management

Profile

DOB: 21-02-2002

Gender: Male

PH No: 8075435334

Alternate Ph No:

Adress: ABC Villa

Permanent Adress: Perumbavoor

City: North Paravur

State: Kerala

District: Ernakulam

Pin: 683513

Id-Proof Number: 3883 5124 5321

Education Type: SSLC

Institute: ABC High School

Percentage: 86%

Education Type:

Institute: SAN HSS

Percentage: 98%

© 2025 Student Document Management system. All Rights Reserved.

Admin Pannel

Logout

Dashboard

Approve Student Req

Approve Documents

Doc Management

Document Type	
Id-Proof	<div>View</div> <div>Accept</div> <div>Reject</div>
SSLC	<div>View</div> <div>Accept</div> <div>Reject</div>
Higher Secondary	<div>View</div> <div>Accept</div> <div>Reject</div>

© 2025 Student Document Management system. All Rights Reserved.

DOCUMENT MANAGEMENT

Admin Pannel

Logout

Dashboard

Approve Student Req

Approve Documents

Doc Management

Student Name

Email

Search

SL NO	Student Name	Email	View/Download
1	Joel Joseph	joeljoseph@gmail.com	<div>View Profile</div> <div>View Documents</div> <div>Download Documents</div>

© 2025 Student Document Management system. All Rights Reserved.

