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.
- approval of request by admins.
- Securely store & serve files.
- Provide dashboards & notifications.

3. Scope

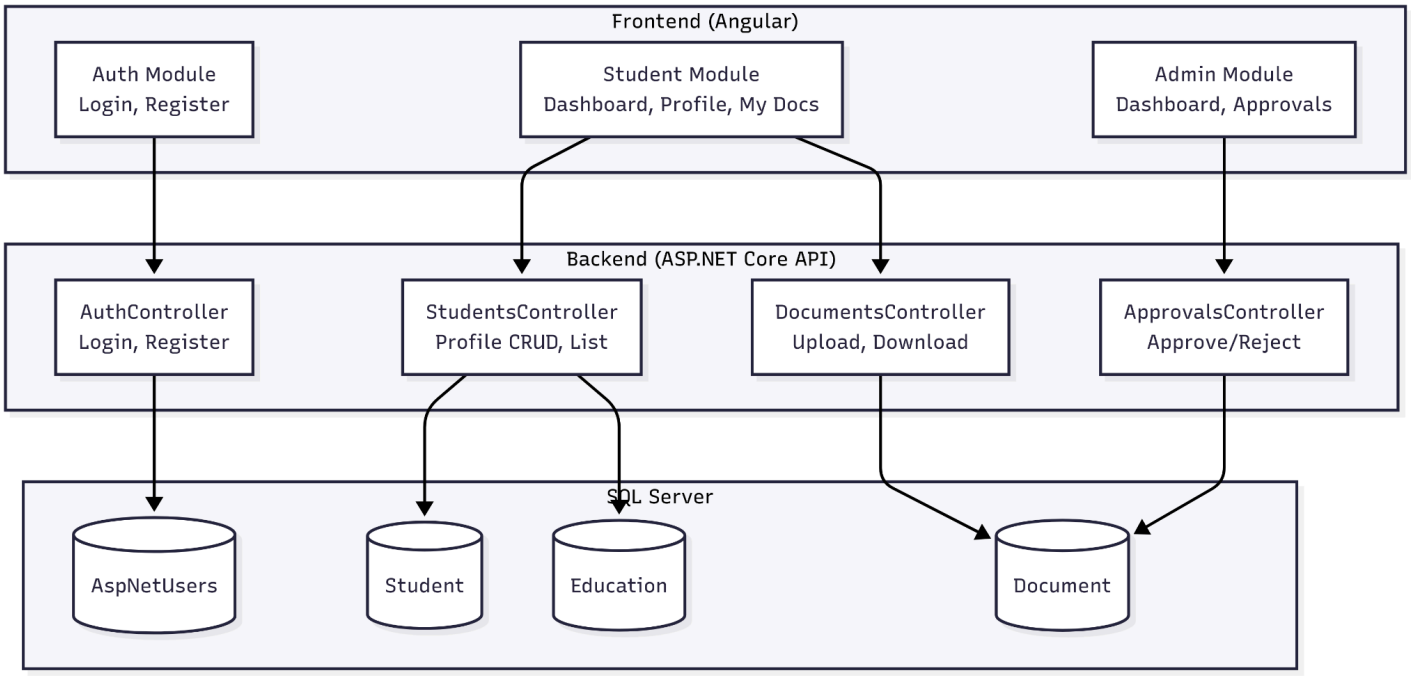
- Student registration and login
- Profile management
- Upload, resubmit, and download documents
- Admin approval/rejection workflows
- Role-based dashboards
- Secure file storage and access
- Search using filters

4. Technology Stack

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

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. **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.

3. **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
`AuthController`	Login, Registration, Role Creation
`StudentsController`	Profile CRUD, list/search students for admin
`DocumentsController`	Upload, list, download, resubmit
`ApprovalsController`	Approve/reject students & documents, provide remarks
DashboardController	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, generate 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 authentication and authorization checks
- **CORS and HTTPS** enforced

7. Backend APIs

Module	Endpoint	Method	Description
Auth	/auth/login	POST	Login, return JWT, role and status
Student	/students/register	POST	Student registration
	/students/me	GET	Get own profile
	/students/me/profile	POST	Add 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

7.1 Sample Request and Response

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

	/students/getallstudents	GET	Admin: list/search students	Header: `Authorization: Bearer <admin_token>`	`[ { "studentId":1,"name":"John Doe","email":"john@example.com","status":"Pending","courseApplied":"BSc CS" } ]`
	/students/{id}/approve	PATCH	Admin: approve student	`{ "Email": "abc@email.com", “Fullname” : “John Thomas”}`	`{ "message": "Student approved" }`
	/students/{id}/reject	PATCH	Admin: reject student	`{ "Email": "abc@email.com", “Fullname” : “John Thomas”}`	`{ "message": "Student rejected" }`
Documents	/documents/upload	POST	Student uploads document	Multipart/form-data: file + `documentTypeId`	`{ "documentId": 1, "status": "Pending" }`
	/documents/mydocuments	GET	Student lists own documents	Header: `Authorization: Bearer <token>`	`[ { "documentId":1,"fileName":"marksheet.pdf","status":"Pending","uploadedOn":"2025-08-18T10:00:00Z" } ]`
	/documents/{id}/resubmit	POST	Student resubmits document	Multipart/form-data: file	`{ "message": "Document resubmitted successfully", "status": "Pending" }`
	/documents/{id}/download	GET	Download document	Header: `Authorization: Bearer <token>`	Streams PDF file with `Content-Type: application/pdf` `Content-Disposition: attachment; filename="marksheet.pdf"`

	/documents/student/{id}	GET	Admin: list all documents of a student	Header: `Authorization: Bearer <admin_token>`	`[ { "documentId":1,"fileName":"marksheet.pdf","status":"Pending","uploadedOn":"2025-08-18T10:00:00Z" } ]`
	/documents/{id}/approve	PATCH	Admin approves document	`{ "remarks": "Verified" }`	`{ "message": "Document approved" }`
	/documents/{id}/reject	PATCH	Admin rejects document	`{ "remarks": "Blurry scan" }`	`{ "message": "Document rejected" }`
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 }`

8. Frontend Design (Angular)

8.1 Modules & Components

Module	Components
AdminModule	Dashboard, Approve Student Request, Approve Documents, 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 the Backend API: POST /StudentAccount/register.
- Backend validates input, creates AspNetUsers record with Status = Pending.
- Frontend shows message: "Registration submitted. Await admin approval."
- Admin reviews pending registrations:
    - Approve: Students 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.
- Students upload 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

- The student sees rejected documents with remarks.
- Student re-uploads corrected documents.
- Frontend calls API: POST /Student/profile-upload
- Backend updates Document table with new file path and Status = Pending.
- Frontend updates document list to Pending.
- Admins views and approves or rejects newly uploaded documents.

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

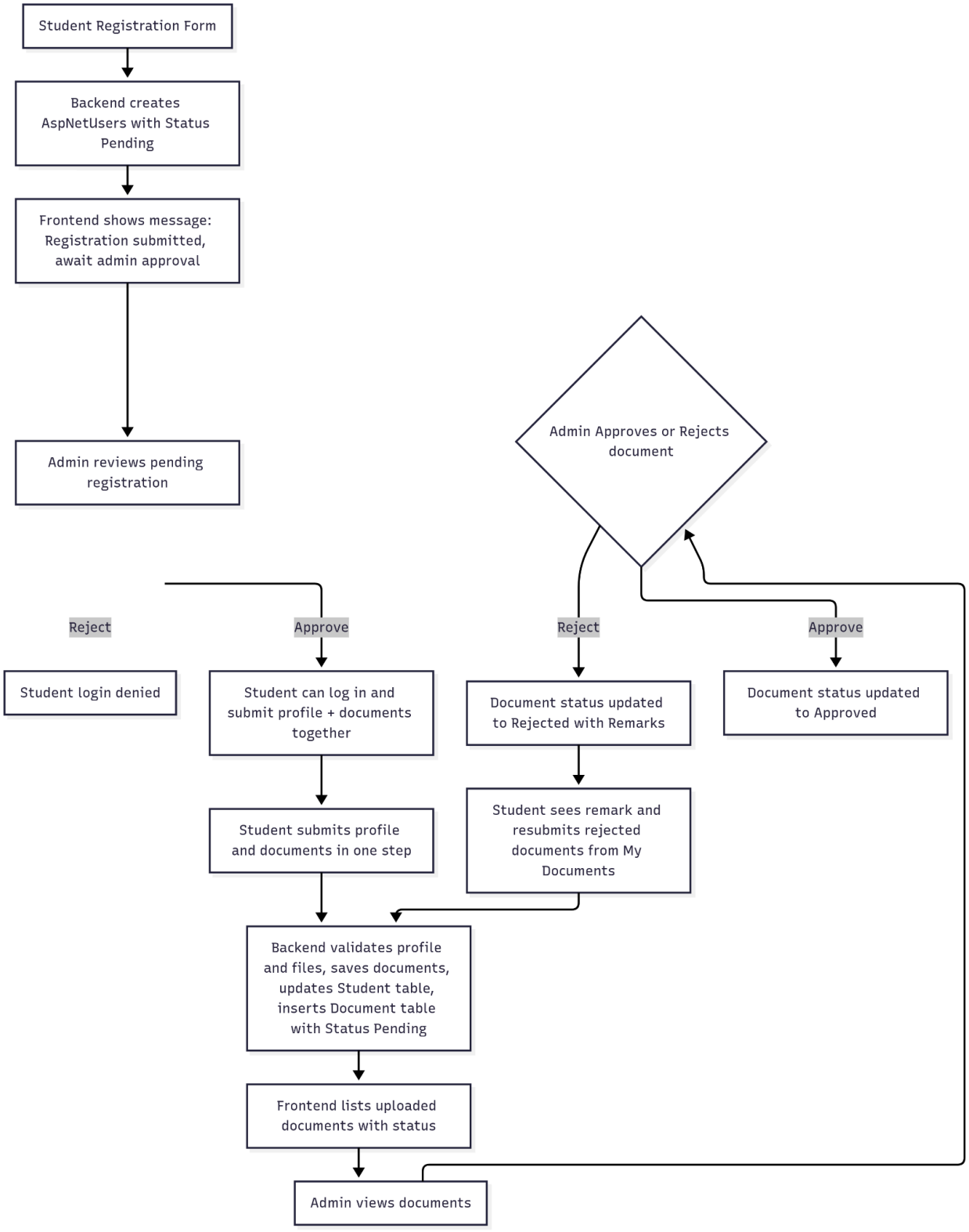
Student Frontend:

- 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)

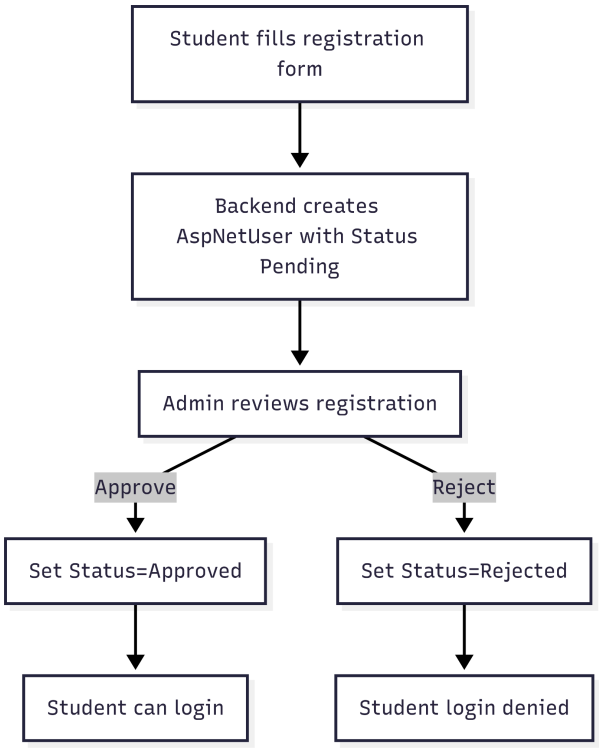
Admin Frontend:

- 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

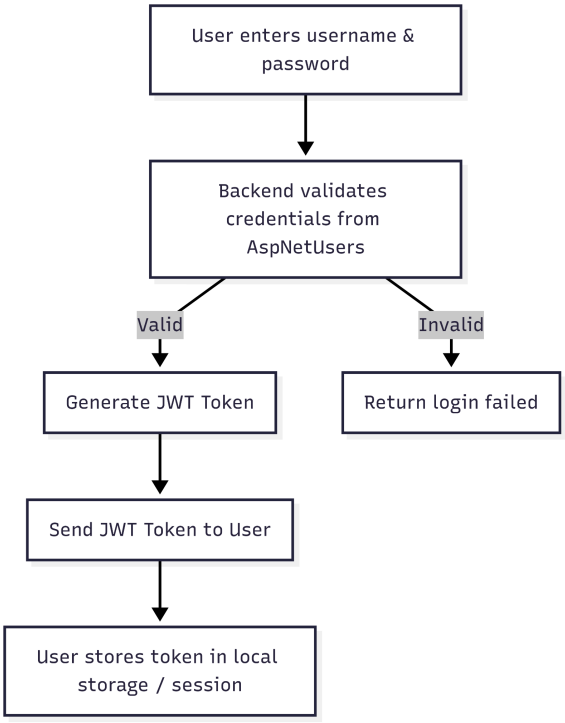
DATA FLOW DIAGRAM



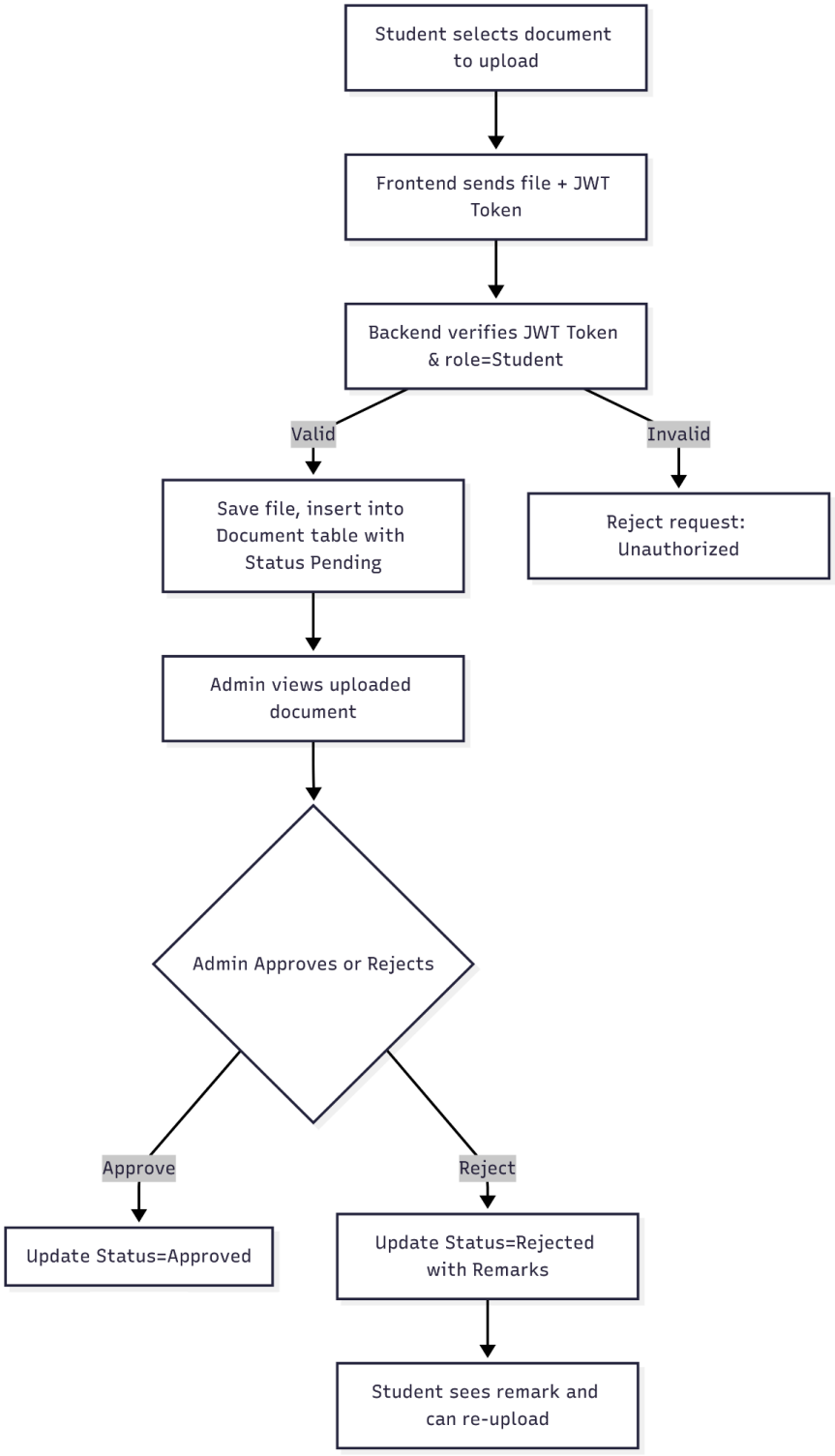
Registration



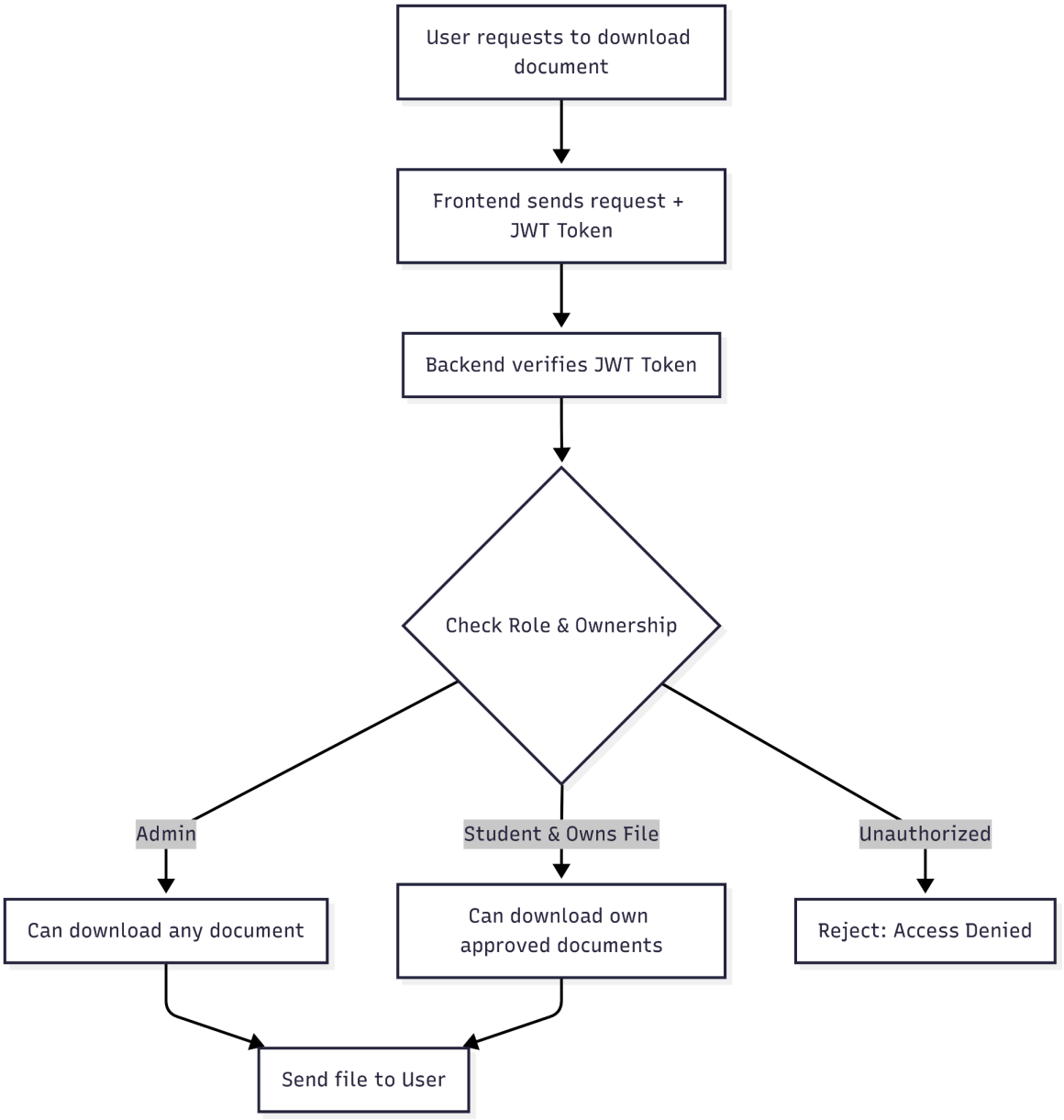
Login



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.) 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 (e.g., B.Sc., B.Com., MBA). Linked with the 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, Profiles, Documents). Contains categories like Pending, Approved, Rejected, Changes Needed.

9. **DocumentType**

Stores the predefined categories of documents (e.g., Marksheet, ID Proof, Transfer Certificate) that students can upload for admission.

10. **AspNetRoles**

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

11. **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
int	DocumentTypeId	FK
string	FileName	
string	FilePath	
int	StatusId	FK
string	Remarks	
datetime	UploadedOn	
datetime	ApprovedOn	

Course		
int	CourseId	PK
string	CourseName	

IdProofType		
int	IdProofTypeId	PK
string	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

DocumentType		
int	DocumentTypeId	PK
string	TypeName	

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

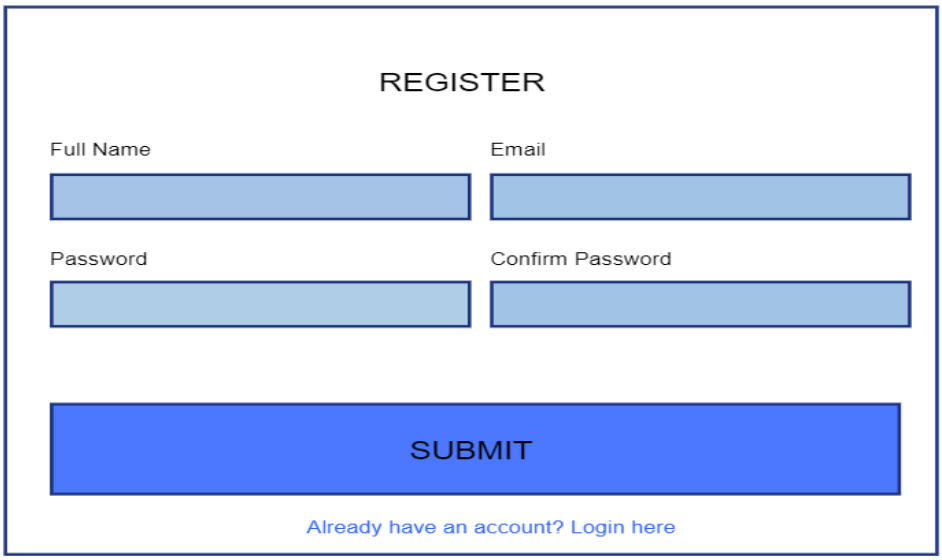
- One Student → Many Notifications (1:M)
    - A student can receive multiple system notifications.
    - Notification.StudentId (FK) → Student.StudentId.

12. Student → StatusMaster

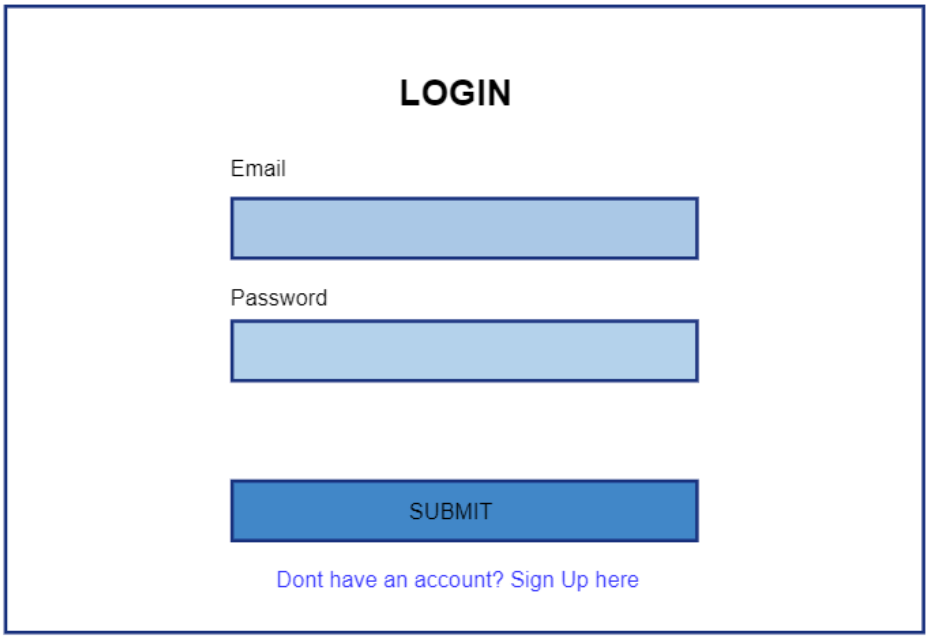
- Many Students → One Application Status (M:1)
    - Each student application has a status (Pending, Approved, Changes Needed).
    - Student.ApplicationStatusId (FK) → StatusMaster.StatusId.

UI Prototypes

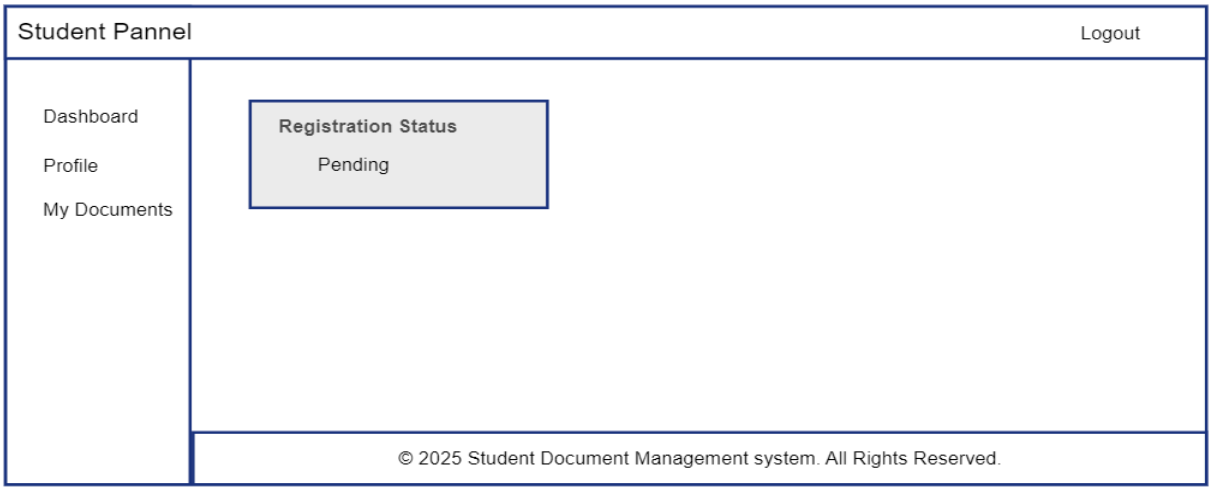
REGISTER PAGE

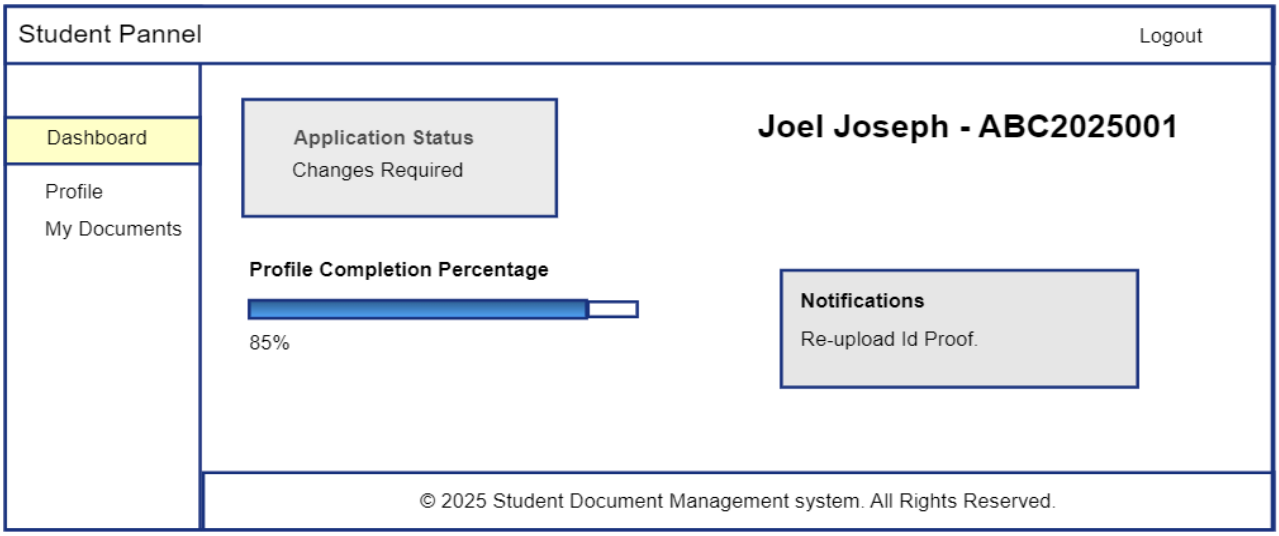


LOGIN PAGE

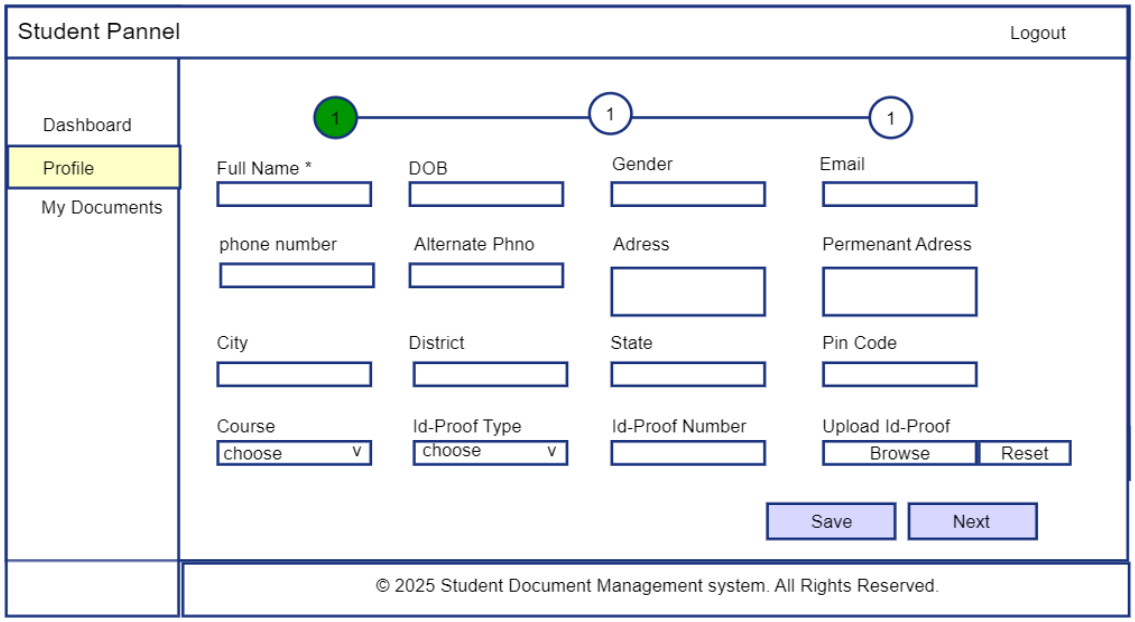


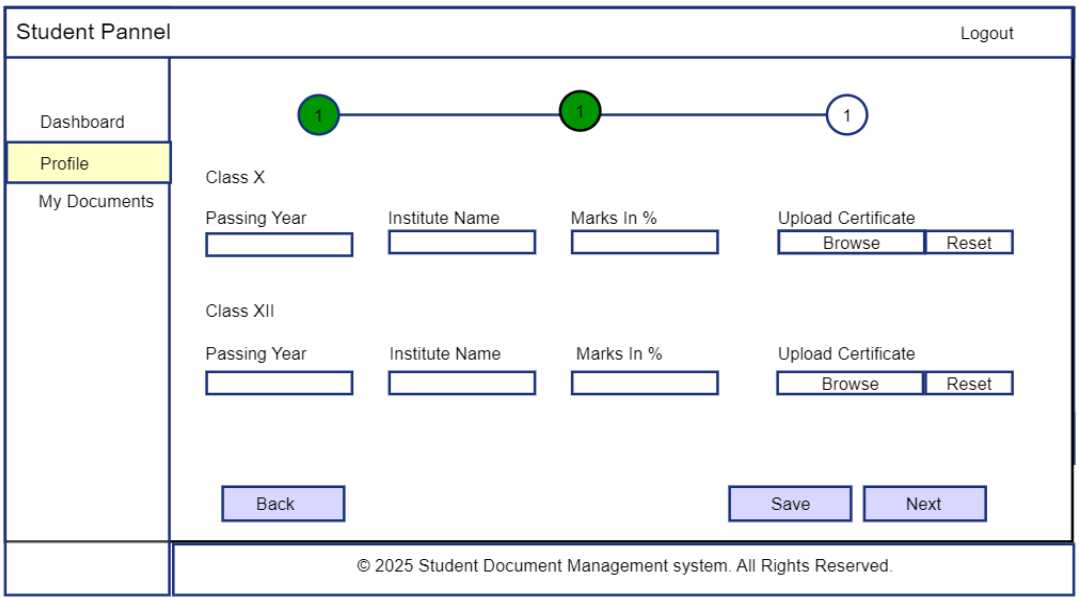
STUDENT DASHBOARD

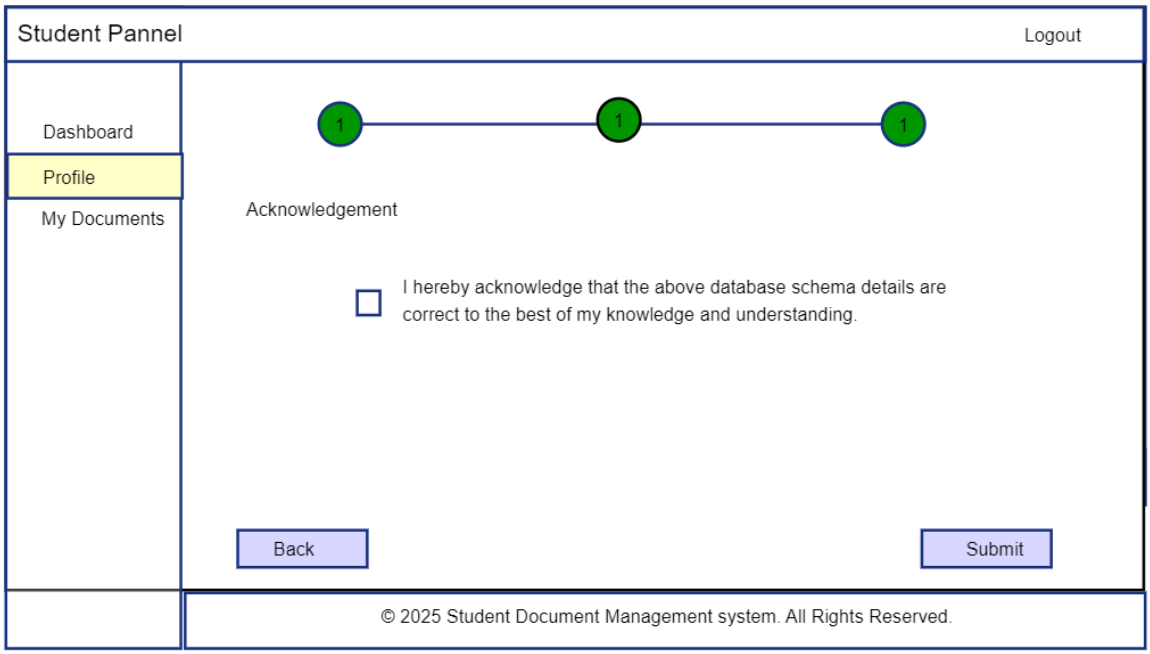




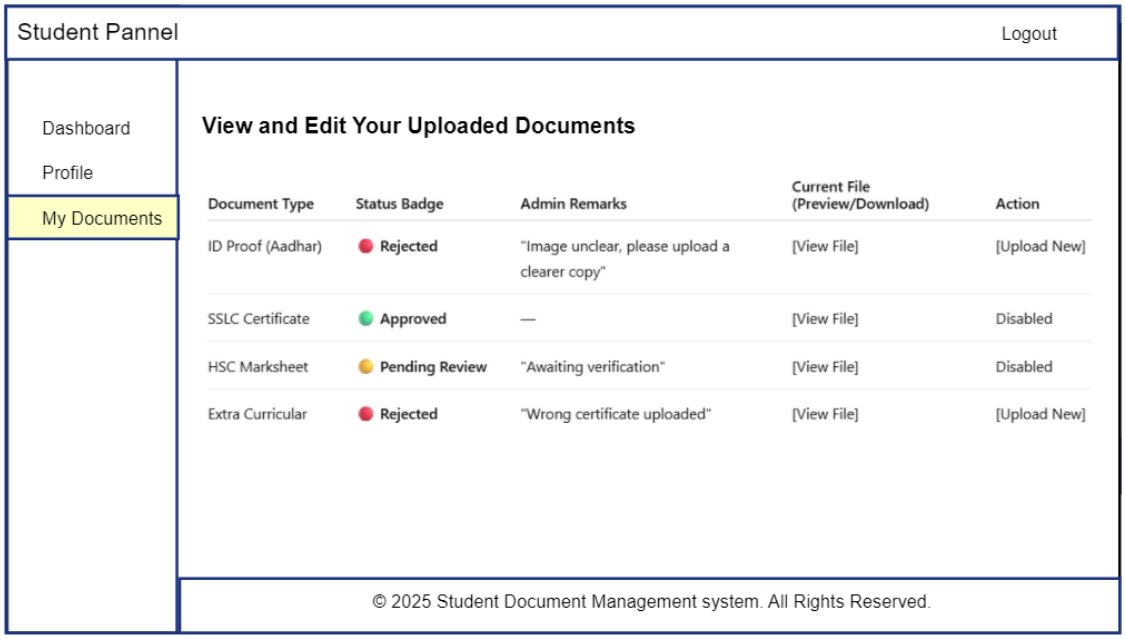
PROFILE


Student Pannel
Logout
Dashboard
Profile
My Documents
1
1
1
Full Name *
DOB
Gender
Email
phone number
Alternate Phno
Adress
Permenant Adress
City
District
State
Pin Code
Course
choose
Id-Proof Type
choose
Id-Proof Number
Upload Id-Proof
Browse
Reset
Save
Next
© 2025 Student Document Management system. All Rights Reserved.

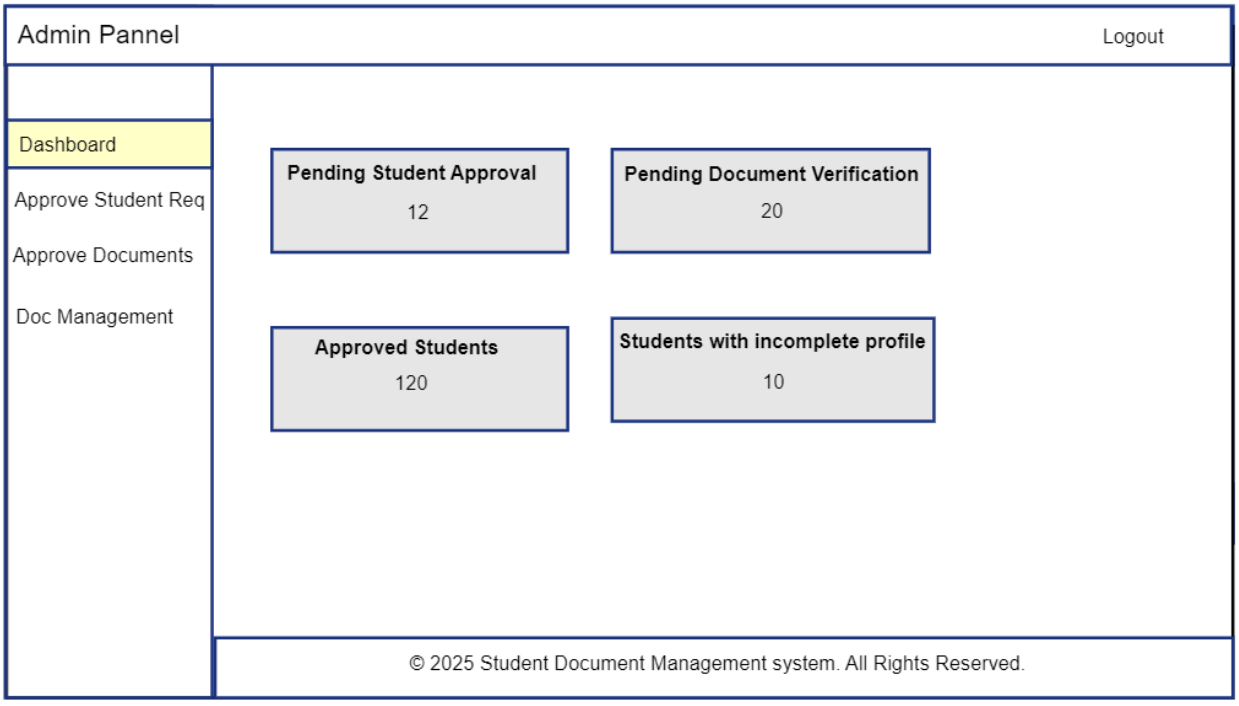

Student Pannel
Logout
Dashboard
Profile
My Documents
1
1
1
Class X
Passing Year
Institute Name
Marks In %
Upload Certificate
Browse
Reset
Class XII
Passing Year
Institute Name
Marks In %
Upload Certificate
Browse
Reset
Back
Save
Next
© 2025 Student Document Management system. All Rights Reserved.


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.

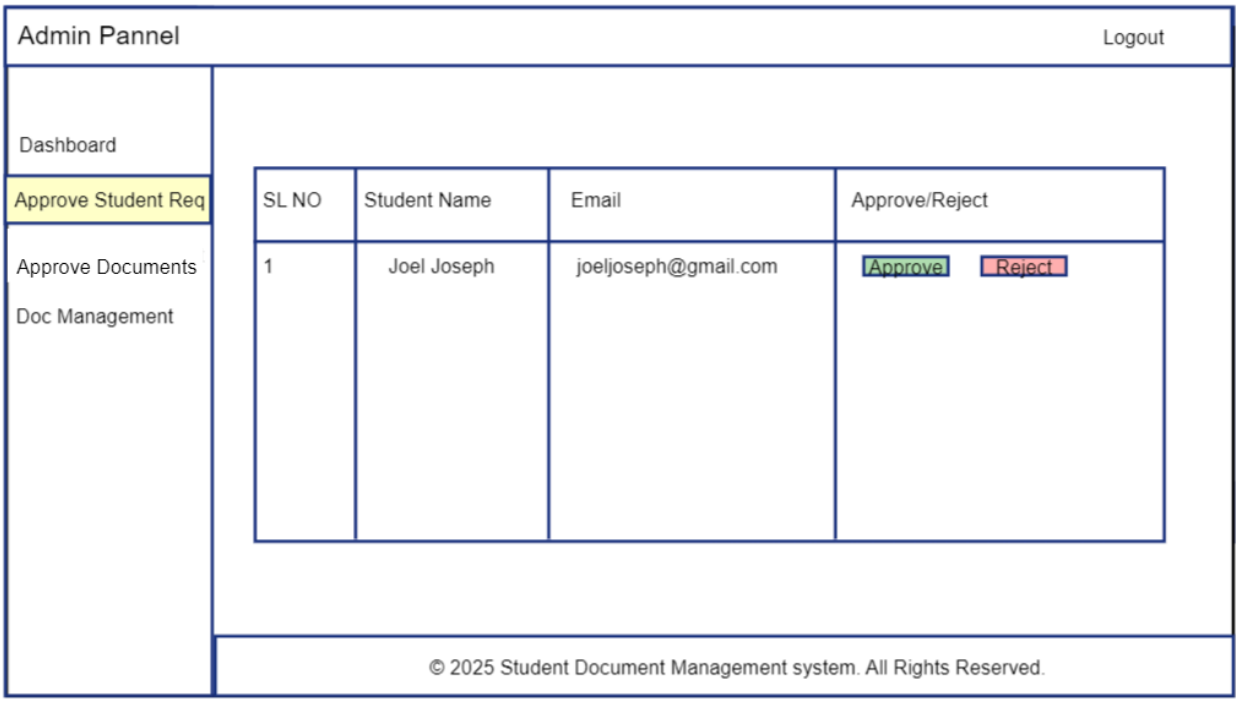
MY DOCUMENTS



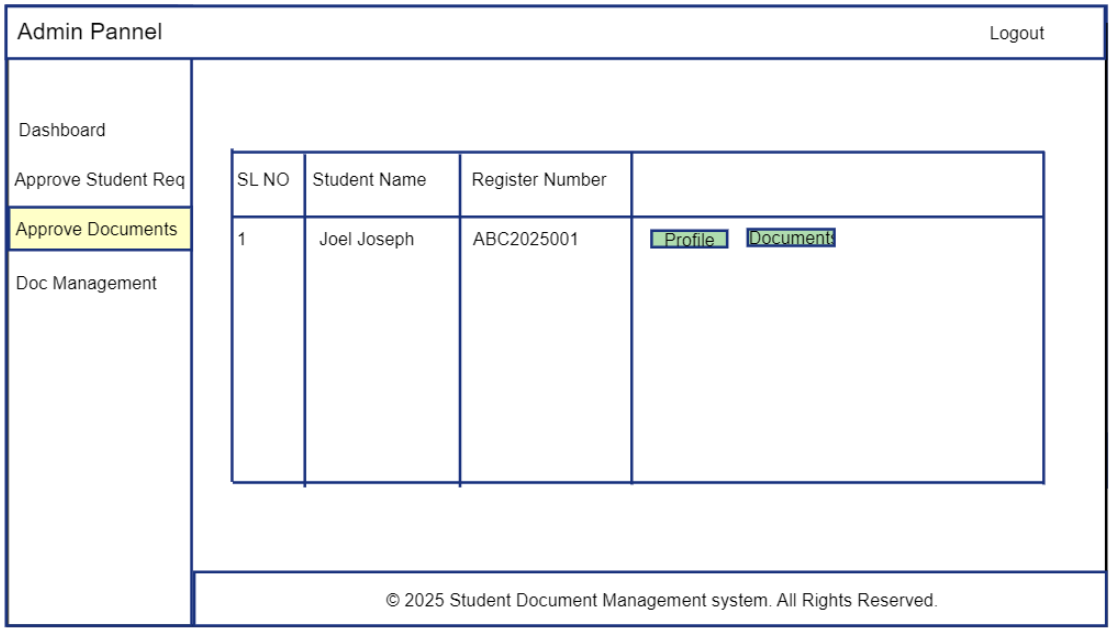
ADMIN DASHBOARD

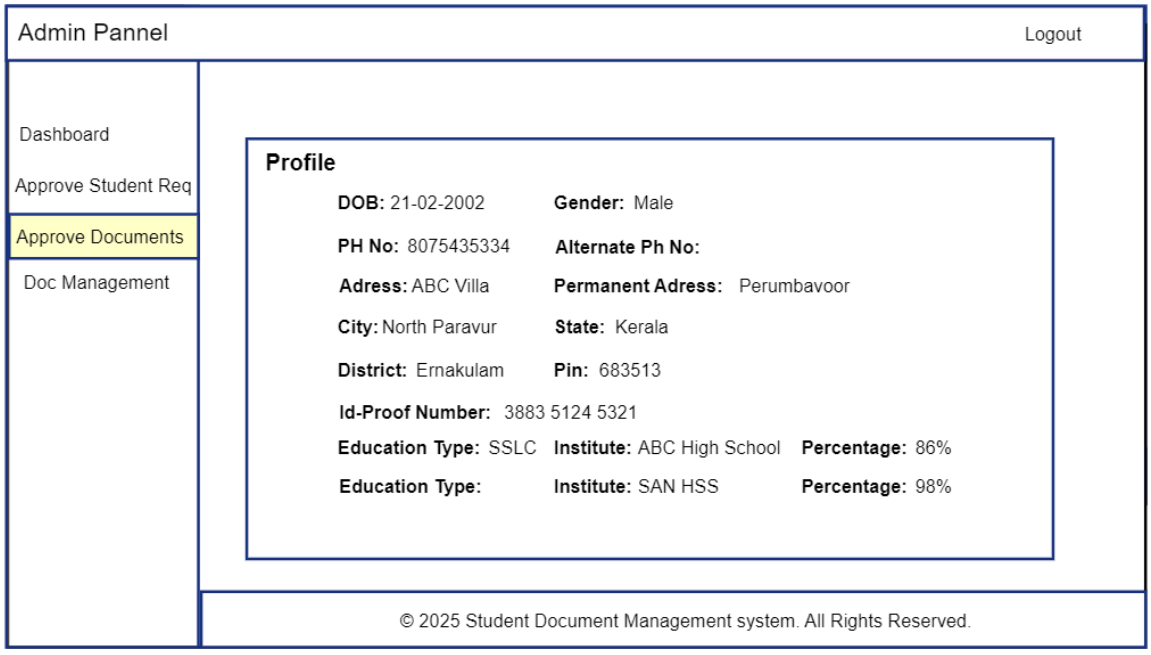


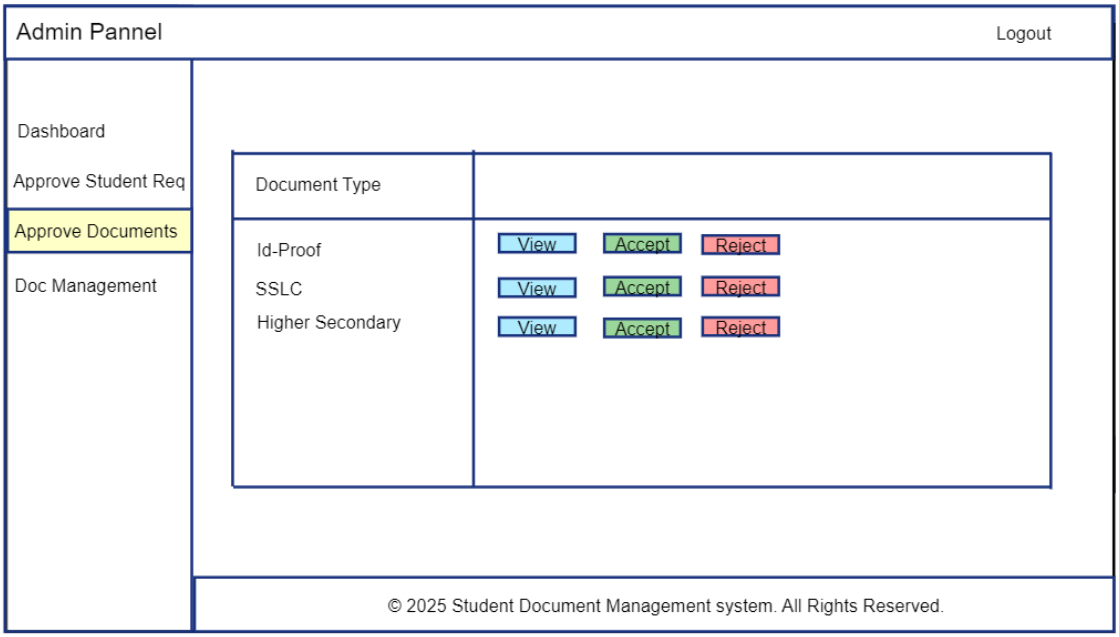
APPROVE STUDENT REQUEST



APPROVE DOCUMENTS




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
SSLC
Higher Secondary
View
Accept
Reject
View
Accept
Reject
View
Accept
Reject
© 2025 Student Document Management system. All Rights Reserved.

DOCUMENT MANAGEMENT