🎓

Smart Campus Management System — Technical Documentation

Purpose:

This document serves as the technical reference for submission, covering:

- System Architecture
- Application Flow
- API Specification
- File-Level Responsibilities

For setup and run instructions, refer to `README.md`.

1️⃣ System Overview

1.1 Architecture

Layer	Description
Frontend	Flutter application (Web & macOS) consuming REST APIs. Single-page flow with role-based routes.
Backend	Node.js + Express REST API, MongoDB (Mongoose), JWT Authentication, Role-based Authorization Middleware
Database	MongoDB
Authentication	JWT (Bearer Token)

1.2 User Roles

👩‍🎓 Student

- Login via `/login`
- Can register
- Access student dashboard

👨‍🏫 Faculty

- Login via `/login`
- Can register
- Access faculty dashboard

👨‍💼 Admin

- Login via `/admin-login`
- No public registration
- Only existing admin can add new admin
- Default admin auto-created if none exists

2️⃣ Technology Stack & Libraries

2.1 Backend (Node.js + Express)

Core Packages

Package	Purpose
express	HTTP server, routing
mongoose	MongoDB ODM
jsonwebtoken	JWT generation & verification
bcryptjs	Password hashing
cors	Cross-origin requests
dotenv	Environment variable loading
express-validator	Request validation
multer	File upload handling
cloudinary	Cloud file storage

Entry Point

`backend/server.js`

Responsibilities:

- Load environment variables
- Connect to MongoDB
- Run `ensureDefaultAdmin()`
- Mount route modules under `/api/*`
- Start server on `PORT` (default: 5001)

2.2 Frontend (Flutter + Dart)

Built using:

- **Flutter Framework**
- **Dart Language**
- Provider for state management
- GoRouter for routing

Main Packages

Package	Purpose
provider	Global state (AuthProvider)
go_router	Declarative routing
http	API requests
intl	Date & number formatting
file_picker	File selection
url_launcher	Open file URLs

Entry File

`frontend/lib/main.dart`

Responsibilities:

- Wrap app with `ChangeNotifierProvider`
- Configure router
- Set theme
- Initial route: `/login`

3️⃣ Application Flow

3.1 Backend Startup Flow

```
server.js
  →connectDB()
  →ensureDefaultAdmin()
  → mount routes
  → app.listen(PORT)
```

3.2 Frontend Startup Flow

```
main()
  → SmartCampusApp
  → Provider initialized
  → Router created
  → Redirectto /login
```

4️⃣ Authentication Flow

4.1 Student / Faculty

1. Login → `POST /api/auth/login`
2. Store JWT token
3. Store profile
4. Redirect to role dashboard

If email not found:

→ Redirect to `/register`

→ `POST /api/auth/register`

→ Redirect after success

4.2 Admin

1. Login → `POST /api/admin/login`
2. Store token
3. Redirect to `/admin`
4. No public registration

5️⃣ Route Guards

Each screen checks:

```
auth.isLoggedIn && auth.user.role =='role'
```

If not:

- Student/Faculty → Redirect `/login`
- Admin → Redirect `/admin-login`

6️⃣ Backend Structure

6.1 Configuration

File	Responsibility
server.js	Bootstrap application
config/db.js	MongoDB connection
config/cloudinary.js	Cloudinary setup
scripts/ensureDefaultAdmin.js	Create default admin
scripts/seed.js	Seed test data

6.2 Middleware

File	Purpose
middleware/auth.js	JWT verification + role protection
middleware/upload.js	Multer config for uploads

6.3 Models

Core Identity

- User
- Student
- Faculty
- Admin

Domain Models

- Leave
- Notice
- Timetable
- Attendance
- Complaint
- Notification
- Expense
- Budget
- Mark
- FeePayment
- FeeReceipt
- SalarySlip
- Certificate
- Meeting
- IDCardRequest
- AttendanceGrievance

7️⃣ API Overview

Base URL:

```
http://localhost:5001/api
```

Authenticated routes require:

```
Authorization: Bearer <token>
```

7.1 Auth APIs

Method	Endpoint	Description
GET	/auth/check-email	Check email existence
POST	/auth/register	Register student/faculty
POST	/auth/login	Login
GET	/auth/me	Get current user

7.2 Admin APIs

Method	Endpoint	Description
POST	/admin/login	Admin login
GET	/admin/me	Get admin
CRUD	/admin/admins	Manage admins

7.3 Users

Admin can:

- Add / Edit / Delete Students
- Add / Edit / Delete Faculty

Students & Faculty:

- Update own profile

7.4 Major Feature APIs

- Attendance

- Leaves
- Notices
- Notifications
- Complaints
- Expenses
- Timetable
- Fee Payments
- Marks
- Budget Requests
- ID Card Requests
- Meetings
- Stats Dashboard

(Each role restricted via middleware)

8️⃣ Frontend Architecture

8.1 Core Layer

Path	Responsibility
core/config/api_config.dart	Base API URL
core/theme/app_theme.dart	App theme
core/models/user_model.dart	User + Profiles
core/providers/auth_provider.dart	Auth state management
core/services/api_service.dart	API calls
core/router/app_router.dart	Routing config

8.2 Feature Modules

Auth

- login_screen
- register_screen
- admin_login_screen

Admin

- Manage Students
- Manage Faculty
- Manage Admins
- Manage Fees
- Dashboard Stats
- Send Notifications

Student

- Dashboard
- Attendance
- Leaves
- Marks
- Fees
- Certificates
- Notifications

Faculty

- Dashboard
- Mark Attendance
- Review Leaves
- Upload Marks
- Budget Requests

9️⃣ Data Flow Summary

1. User logs in

2. Token stored in AuthProvider
3. ApiService attaches Bearer token
4. Backend validates JWT
5. Controller processes request
6. JSON returned
7. UI updates

Logout:

- Clear token
- Redirect to login

🔟 Environment Configuration

Backend `.env`

Must include:

```
MONGO_URI=
JWT_SECRET=
PORT=5001
CLOUDINARY_*
DEFAULT_ADMIN_*
```

Frontend

Update:

```
lib/core/config/api_config.dart
```

Set `baseUrl` to production server when deployed.

1️⃣1️⃣ Health Check

```
GET /api/health
```

Response:

```
{ok:true }
```

📌 Conclusion

This documentation provides a complete architectural and functional overview of the Smart Campus Management System.

It is intended for:

- Academic Submission
- Technical Evaluation
- Developer Reference

🎓