Backend Framework Plan (A–Z)

A. Goal

Build a backend that is multi-tenant, RBAC-driven, consistent, and scalable.

B. Folder Structure

src/

app.js

server.js

config/, constants/, models/, routes/, controllers/, services/,

validations/, middlewares/, utils/, jobs/, tests/, docs/

C. .env Policy

No hardcoding. All keys, secrets, DB URIs, cron schedules via .env.

D. .env Template

PORT, API_PREFIX, MONGO_URI, JWT secrets, CORS origins, rate limit, storage, etc.

E. Config Loader

config/env.js validates required .env keys and exports typed config.

F. Server Bootstrap

server.js: connects DB, registers jobs, starts express.

G. Database Config

config/db.js uses env.mongoUri, env.mongoDbName, env.poolSize.

H. Express Core

Middleware chain: CORS → rateLimit → JSON parser → auth → tenant → routes → error.

I. Auth Middleware

Verifies JWT, attaches req.user.

J. Tenant Middleware

Injects req.schoolId from JWT.

K. RBAC + Permissions

roles.js, permissions.js, rolePermissions.js, permit.middleware.js.

L. Validation

validate.middleware.js uses Zod/Joi schemas.

M. Routes Layer

Routes define path + permit + validate + controller.

N. Controllers Layer

Thin. Calls services and returns ApiResponse.

O. Services Layer

All business logic: quota, team creation, reporting, promotion.

P. Models Layer

Schema + indexes + timestamps. No mixed logic.

Q. Base Model

Shared fields: schoolId, createdById, updatedById.

R. Error Strategy

ApiError + global error.middleware.js.

S. Response Strategy

apiResponse.ok(), .created(), .paginated().

T. Audit Logging

Middleware logs admin write actions.

U. Pagination

Shared util for all list endpoints.

V. Rate Limiting

Env-driven express-rate-limit.

W. Logging

pino/winston configured by LOG_LEVEL.

X. Jobs / Cron

Promotion, team pipeline, sync tasks scheduled via env cron strings.

Y. Tests

Unit tests for services; integration tests for RBAC/tenant enforcement.

Z. Rules

No queries without schoolId; no business logic in controllers; env only in config/env.js.