Success Planner Backend – Full & Extended Development Plan (2025)

# 🧾 Project Overview

Success Planner एक productivity app है जो personal users और admin दोनों के लिए है। इसमें authentication, user/admin roles, और planners (yearly, monthly, weekly, todo) के modules शामिल हैं। अब इसमें future-ready modules भी add किए गए हैं जैसे notifications, file uploads, and activity logs।

# 🛠️ Technologies Used

* Node.js – Backend runtime
* Express.js – Routing & server
* MongoDB + Mongoose – NoSQL Database
* JWT + cookie-parser – Authentication via tokens
* bcryptjs – Password encryption
* Zod – Input validation
* Nodemailer – OTP/Email for password reset
* Cloudinary – Image uploads
* Helmet, CORS, Rate Limiting – Security
* dotenv – Config management

# 📦 Modules Breakdown

* Auth: Register, Login, Logout, OTP-based Reset, Email Verification
* Role Management: User can switch to/from Admin
* Dashboard: Yearly, Monthly, Weekly planner + To-Do List (Doing/Done)
* Admin Panel: Total users, stats, role control (only admin access)
* User Profile: Avatar update, profile data, switch role
* Notifications: Task reminders via Email
* Task History & Logs: User actions log for review
* File Uploads: Upload personal notes or planner-related documents

# 🔁 Development Steps

1. 1. Project initialize (npm init)
2. 2. Folder structure setup (as per plan)
3. 3. MongoDB connect (config/db.js)
4. 4. user.model.js बनाएं (role, password, avatar)
5. 5. Auth: register/login/logout (JWT cookie-based)
6. 6. Middleware बनाएं: auth, role, error
7. 7. switchRole API बनाएं (PATCH /user/switch-role)
8. 8. Admin route (/admin/data) बनाएं
9. 9. Dashboard planner: model, controller, routes
10. 10. Notifications system: reminders via email
11. 11. File Uploads setup using Cloudinary
12. 12. Task Logs module for tracking
13. 13. Zod validation हर input पर
14. 14. Security: Helmet, Rate Limit, JWT secure
15. 15. Postman से test करें
16. 16. Deploy करें

# 🔐 Security Checklist

* JWT in secure, HTTPOnly cookies
* Zod validations for all user input
* Bcrypt hashed passwords
* Role switch only by authenticated users
* Role-based route protection (restrictTo("admin"))
* CORS + Helmet + Rate Limit enabled
* Email verification before role-switch
* Cloudinary upload security using signed URLs

# ✅ Final Notes

हर काम को services/controller में अलग-अलग रखो। User को manually admin बनाने का switch देना smart है, लेकिन protected रखना ज़रूरी है। UI में केवल authorized sections दिखाओ based on role। हर बार testing करते चलो: auth → middleware → role → dashboard। Notifications, file uploads, और history logs future SaaS feature की तरह काम आएंगे।

# 🧠 Module-Wise Planning and Action Points

## 1. Authentication (auth.controller.js, auth.service.js)

* ✔ Register user with Zod validation
* ✔ Hash password using bcryptjs
* ✔ Store JWT in HttpOnly cookie
* ✔ Forgot password via OTP (otp.model.js + sendEmail.js)
* ✔ Secure login and logout with cookie cleanup

## 2. Role Management (user.controller.js)

* ✔ Add role field to user model
* ✔ switchRole() function to toggle role from 'user' to 'admin' and vice versa
* ✔ Only allow switch if email is verified
* ✔ Route protected by JWT

## 3. Dashboard / Planner Modules (dashboard.controller.js)

* ✔ Create planner schema for yearly, monthly, weekly goals
* ✔ Add CRUD API for each planner
* ✔ Separate service logic in dashboard.service.js
* ✔ Use middleware to protect all planner routes

## 4. Admin Dashboard (admin.controller.js)

* ✔ Get total users, active/inactive counts
* ✔ Analytics data: tasks completed, planners used
* ✔ Protect route with restrictTo('admin')

## 5. Notifications (utils/sendEmail.js)

* ✔ Setup email reminder service using nodemailer
* ✔ Cron Job or manual trigger to send reminder emails
* ✔ Store reminder preferences in user model (optional)

## 6. File Uploads (config/cloudinary.js)

* ✔ Upload planner files, documents
* ✔ Store Cloudinary URLs in DB
* ✔ Add multer or base64 upload method

## 7. History & Logs

* ✔ Create model to log each planner update
* ✔ Store timestamp, action type, user ID
* ✔ Use in dashboard.service.js after task update

## 8. Security Enhancements

* ✔ Use helmet, cors, rate-limit in app.js
* ✔ Set cookie options: HttpOnly, Secure, SameSite
* ✔ Validate all inputs using Zod
* ✔ Restrict admin routes securely
* ✔ Store sensitive values in .env only