# Logic Camp – Backend Documentation

## 1. Overview

The Logic Camp backend provides RESTful APIs to manage:  
- Users (registration, authentication, roles, approval, notifications)  
- Teams (creation, membership, leadership)  
- Projects (ownership by Admin/Team Lead, file uploads, status tracking)  
- Goals (linked to projects, milestones with deadlines)  
- Tasks (linked to goals, assigned to users, smallest unit of work)  
- Task Comments (discussion, file attachments on tasks)  
  
It enforces role-based access (Admin, Team Lead, Employee) and ownership rules as defined in the database documentation.

## 2. Authentication & Authorization

- JWT-based authentication  
 - On login, a JWT is issued.  
 - Token must be passed in Authorization: Bearer <token> header.  
  
- Authorization Rules:  
 - Admin: Full access (manage users, teams, projects, goals, tasks).  
 - Team Lead: Manage their team, create/manage projects/goals/tasks under their team.  
 - Employee: Can view/join assigned teams, work on assigned tasks, comment.

## 3. API Endpoints

Users (Base URL: /api/users)  
- POST /register → Register new user (default role: employee, is\_approved=false).  
- POST /login → Authenticate user, return JWT.  
- GET / (Admin only) → List all users.  
- PATCH /approve/:id (Admin) → Approve/reject user.  
- GET /:id → Fetch user details.  
- PATCH /:id → Update user profile.  
- DELETE /:id (Admin) → Soft delete user.  
  
Teams (Base URL: /api/teams)  
- POST / (Admin/Team Lead) → Create team, assign team\_lead\_id.  
- GET / → List all teams.  
- GET /:id → Get team details + members.  
- PATCH /:id (Admin/Team Lead) → Update team info.  
- DELETE /:id (Admin) → Deactivate team (is\_active=false).  
  
Projects (Base URL: /api/projects)  
- POST / (Admin/Team Lead) → Create project, set owner\_id and team\_id, allow file uploads.  
- GET / → List all projects with filters (status, team\_id, owner\_id).  
- GET /:id → Get project details, goals, tasks.  
- PATCH /:id (Owner/Admin) → Update project info or status.  
- DELETE /:id (Admin) → Archive project (status=archived).  
  
Ownership Rules:  
- Admin can create/manage all projects.  
- Team Lead can only create/manage projects for their team.  
- Employees cannot create projects.  
  
Goals (Base URL: /api/goals)  
- POST / (Admin/Team Lead) → Create goal under a project.  
- GET /project/:projectId → List all goals for a project.  
- GET /:id → Get goal details.  
- PATCH /:id (Owner/Admin) → Update goal status/details.  
- DELETE /:id (Admin/Owner) → Delete goal.  
  
Tasks (Base URL: /api/tasks)  
- POST / (Admin/Team Lead) → Create task under a goal, assign to user.  
- GET /goal/:goalId → List tasks for a goal.  
- GET /assigned/:userId → List tasks assigned to a user.  
- GET /:id → Get task details.  
- PATCH /:id (Assigned user/Admin/Owner) → Update task status/details.  
- DELETE /:id (Admin/Owner) → Delete task.  
  
Task Comments (Base URL: /api/task-comments)  
- POST / (Any authenticated user) → Add comment or file to task.  
- GET /task/:taskId → List comments for a task.  
- PATCH /:id (Owner/Admin) → Edit comment.  
- DELETE /:id (Owner/Admin) → Delete comment.

## 4. Business Logic Validations

- User Registration: Email must be unique. Default role is employee. Needs admin approval before login success.  
- Teams: team\_lead\_id must be a user with role team\_lead or admin.  
- Projects:  
 - owner\_id must be Admin/Team Lead.  
 - end\_date >= start\_date.  
- Goals/Tasks: Deadlines must be within project date range.  
- Tasks: assigned\_to must belong to the project’s team.  
- Comments: Must contain at least text or file.

## 5. File Uploads & Notifications

- Files: Stored in JSON fields (projects.files, task\_comments.files). Recommend using cloud storage (e.g., AWS S3, Cloudinary).  
- Notifications: Stored in users.notifications. Events: new task, comment, approval, project update.

## 6. Error Handling

- 401 Unauthorized: Invalid/missing token.  
- 403 Forbidden: User not allowed to perform action.  
- 404 Not Found: Entity doesn’t exist.  
- 422 Validation Error: Business rules violated.  
- 500 Internal Server Error: Unexpected errors.

## 7. Recommended Tech Stack

- Backend Framework: Next.js   
- Database: PostgreSQL / MySQL (fits well with your relational design)

- Authentication: JWT + bcrypt for password hashing  
- File Handling: Multer + cloud storage (AWS S3, Cloudinary)