Task Management System — Project Design & Client Agreement

1. Project Summary

Project name: Company Task & Project Management System (CTPMS)

Purpose: Build a web-based system to allow hierarchical task and project management (MD → Managers → Employees), supporting task/project assignment, team or single assignments, status & progress tracking, timelines, due-date alerts, meeting links (Google Meet / Zoom), WhatsApp shareable URLs, and annual performance summaries based on manager ratings.

Primary goals: - Centralize task & project assignments. - Provide progress visibility and reporting for MD and Managers. - Automate reminders, meeting links, and WhatsApp sharing. - Provide annual performance reports and role-based dashboards.

---

2. Stakeholders

- **Client (Company / MD)** — project sponsor and admin.
- **Managers** — create/assign projects & tasks, rate employees.
- **Employees** — receive and update tasks, join meetings, log progress.
- **Project Team** — developers, QA, DevOps, UI/UX, PM.

---

3. Scope (In-Scope / Out-of-Scope)

In-scope: - User roles & permissions: MD (Admin), Manager, Employee. - Projects and Tasks: create, assign (single/team), status, priority, start/end dates, estimated days. - Progress tracking: % complete, subtasks, attachments, comments. - Notifications: in-app, email, optional WhatsApp URL share, calendar invites for Google Meet/Zoom. - Performance reporting: per-task ratings, aggregated quarterly/annual performance. - Dashboards: MD view (company-level), Manager view (team-level), Employee view (personal tasks). - Audit logs, basic analytics, CSV/PDF exports.

Out-of-scope (initial release): - Payroll or HR integrations (unless requested). - Advanced AI-assisted task suggestions (can be future phase). - Deep mobile app (responsive web only; native apps later).

---

4. Functional Requirements

4.1. Authentication & Authorization

- Role-based access control (RBAC). MD = super-admin.
- Single Sign-On (optional, later): Google Workspace / Microsoft.

4.2. Projects

- Create/edit/delete projects (Manager/MD).
- Project metadata: name, description, owner, start & end dates, status, total estimated days, stakeholders.
- Project-level progress (auto from tasks).

4.3. Tasks

- Create single/team tasks with: title, description, assignees, due date, estimated days, priority, attachments, subtasks.
- Status lifecycle: To Do → In Progress → Blocked → In Review → Done.
- Time tracking (optional): manual log or integration with time tracker.
- Reassign, comment, tag, link to project.

4.4. Teams

- Managers create teams, add employees.
- Team tasks and multi-assignee tasks support percentage split or shared responsibility.

4.5. Notifications & Integrations

- Email and in-app notifications for assignment, due soon, overdue, comments, status changes.
- Calendar invite generation with Google Meet or Zoom link when creating a meeting event.
- Generate WhatsApp message link (https://wa.me/?text=...) prefilled with task summary for quick sharing.
- Optional webhooks for external systems.

4.6. Reporting & Performance

- Task-level ratings by manager on completion (1–5 scale) with optional comments.
- Aggregated performance score per employee: weighted average across tasks, quarterly and annual rollups.
- Reports: tasks by status, overdue tasks, resource utilization, performance summary.

4.7. Admin Features

- Role & user management, audit logs, system settings, export/import of tasks and users.

---

5. Non-Functional Requirements

- **Security:** TLS 1.2+, secure password policies, RBAC, data encryption at rest for sensitive fields.
- **Availability:** 99.5% SLA for production.
- **Scalability:** Support for 500–5,000 users (design for horizontal scaling).
- **Performance:** Page load <2s for dashboards under typical load.
- **Compliance:** GDPR-ready export/delete functionality (if required).

---

6. High-Level Architecture

- **Frontend:** React (or Next.js) SPA, responsive design.
- **Backend:** RESTful API (Node.js/Express or Python FastAPI) with JWT-auth and RBAC.
- **Database:** PostgreSQL (relational data) + Redis (caching, rate-limiting, pub/sub for notifications).
- **Storage:** S3-compatible object storage for attachments.
- **Integrations:** Zoom API / Google Calendar & Meet, WhatsApp share links (simple URL), Email service (SendGrid/Mailgun).
- **Hosting:** Cloud (AWS/GCP/Azure) — Kubernetes or managed services; CI/CD pipeline.

Diagram: (to be added in design workshop — recommended: draw sequence flows for task assignment, notification flow, and performance aggregation)

---

7. Data Model (Core Entities)

- **User**: id, name, email, role, manager_id, team_id, metadata
- **Project**: id, title, owner_id, description, start_date, end_date, status, estimated_days
- **Task**: id, project_id, title, description, assignee_ids, status, priority, start_date, due_date, estimated_days, actual_days, percent_complete
- **Subtask**: id, task_id, title, status
- **Comment**: id, parent_id(task/comment), author_id, content, timestamp
- **Rating**: id, task_id, rater_id, ratee_id, score, comment, timestamp
- **AuditLog**: action, user_id, target_id, timestamp

(Provide DB schema in next steps.)

---

8. API Endpoints (sample)

- `POST /api/login` — authenticate
- `GET /api/projects` — list projects (RBAC filtered)
- `POST /api/projects` — create project
- `GET /api/projects/{id}` — project details
- `POST /api/projects/{id}/tasks` — create task
- `PATCH /api/tasks/{id}` — update task (status, assignee)
- `POST /api/tasks/{id}/comments` — add comment

- `POST /api/tasks/{id}/rating` — add completion rating
- `GET /api/reports/performance?year=YYYY` — performance report

---

9. UI / UX Outline

Key screens: 1. Login / SSO 2. Dashboard (role-specific) 3. Projects list & project detail 4. Task list (My Tasks / Team Tasks) with filters (status, due, priority) 5. Task detail modal (comments, attachments, subtasks, activity) 6. Create task/project wizard 7. Calendar & Timeline (Gantt-lite) 8. Reports & Export 9. Admin panel

UX notes: - Bulk actions (assign, change status) for managers. - Mobile-first responsive layout; keep primary actions reachable.

---

10. Notifications & Meeting Flow

1. Manager creates a meeting task → choose meeting provider (Google Meet / Zoom) → system creates calendar invite with provider link (via API) and attaches to task.
2. System sends email + in-app notification to assignees.
3. Provide a "Share via WhatsApp" button that builds a `wa.me` link prefilled with the meeting/task summary for quick sharing.

Privacy note: For Google/Zoom API use, the system will request OAuth consent from the company workspace account. For WhatsApp, only use share links — avoid automated messaging without user consent.

---

11. Performance & Rating Calculation (Example)

- Per-task score = manager rating (1–5) * task weight.
- Task weight = estimated_days / sum(estimated_days for all tasks in period) or manager-assigned weight.
- Quarterly score = sum(per-task score) / sum(weights).
- Annual score = weighted average of quarterly scores.

(Provide spreadsheet or calculation module in implementation.)

---

12. Acceptance Criteria (for each deliverable)

- Feature works end-to-end: create → assign → update → complete → rating.
- Notifications trigger and are received by intended recipients.
- Performance report matches test-case calculations.
- Security tests passed (OWASP Top 10 basic checks).
- Cross-browser compatibility for latest Chrome/Edge/Firefox.

13. Milestones & Timeline (Suggested)

- **Discovery & Requirements Finalization** — 1 week
- **Design & Prototyping** — 2 weeks
- **MVP Development (Core features)** — 6–8 weeks
- **Testing & UAT** — 2 weeks
- **Deployment & Handover** — 1 week

Total: ~12–14 weeks (adjustable based on scope)

14. Cost Estimate (Placeholder)

Provide a placeholder section here; fill with estimates after scoping session. Example structure: - Discovery & Design: X - Development (MVP): Y - Testing & Deployment: Z - Maintenance (monthly): M

15. Change Request & Support

- All scope changes require written Change Request with impact on timeline and cost.
- Support SLA (example): 30 business days bug fixes included; expedited support & feature development billed separately.

16. Risk & Mitigation

- **Integration delays (Zoom/Google)** — mitigation: queue optional manual link input as fallback.
- **Over-scoping** — mitigation: define strict MVP and phased roadmap.
- **Adoption resistance** — mitigation: training session + user guides.

17. Security & Data Privacy

- Enforce least-privilege access.
- Regular backups and retention policy.
- Optional 2FA for sensitive roles.

18. Deliverables (to sign off)

- Functional web application (MVP) with listed features.
- Admin guide and user manual.
- Deployment scripts & CI/CD pipeline docs.

• Training session for Managers & MD.
• Source code handover (or hosted service as agreed).

---

19. Client Sign-off (Template)

Company: ________

Project: Company Task & Project Management System

Scope agreed: (attach scope page)

Start date: //____

Acceptance sign-off date: //____

Payment terms: ________

Authorized signature (Client): ___ **Date:** /__/____

Authorized signature (Vendor): ___ **Date:** /__/____

---

20. Next steps (recommended)

1. Book a 60–90 minute discovery workshop with key stakeholders (MD, 2–3 Managers, IT lead) to finalize scope and priorities.
2. Prepare detailed estimates and a Statement of Work (SoW) based on discovered scope.
3. Sign the agreement and start the Design sprint.

Prepared by: Dev Team *Date:* (to be filled)

If you want, I can convert this into a PDF or a more formal SoW and include a cost estimate and Gantt timeline. Tell me which parts you'd like expanded.