`

L1 Software Engineer Technical Design Challenge

Company: Orbtronics Ltd
Prepared by: Shergaun Roserie, Nysa Pierre

INNOVATION BEGINS HERE

Challenge Details

Challenge: “Mini Tasks” — Build & Deploy in 5 Days

Brief

We’re looking for engineers who can balance practical execution with good engineering design. In this challenge, you’ll build and deploy a lightweight Task Manager web app that allows users to manage their tasks, with a few optional bonus features that test your creativity and technical range.

This is a timeboxed challenge (5 days) designed to simulate real-world work: shipping quickly, documenting decisions, and deploying to production.

Build a **task manager** (web) where users can:

- Sign up / sign in
- Create/update/delete tasks (title, description, priority, due date, status)
- See tasks in a list with simple filters (status, due this week)
- (Bonus) AI: “Suggest due date” button that proposes a due date from the description

Hard Requirements

- **Frontend:** Next.js + React
- **Backend:** Python
- **DB:** MongoDB
- **Auth:** Email/password (session/JWT) no social logins
- **Deployment:** Live URL(s) required
- **API:** REST JSON

- **Testing:** At least 3 minimal tests (unit or API)
- **Docs:** README.md with setup, env vars, deployment notes, and architecture diagram (can be a PNG or Mermaid)

Nice to Have (Bonus)

- AI: "Suggest due date" button powered by OpenAI API
- Realtime task updates (WebSocket/SSE) or optimistic UI
- Basic responsive design & accessibility checks
- Dockerfiles for both services

Deliverables

- Public GitHub repo with:
  - /frontend and /backend folders
  - README.md (with how-to-run instructions, env vars, and URLs to deployed app + API)
  - Architecture diagram (Mermaid or image)
  - Short note on tradeoffs & future improvements
- Live URLs:
  - Web app
  - API base
- Demo credentials (test user)

Timebox

- **Take home window:** 5 days from receipt of challenge

Evaluation Rubric (100 pts)

- **Correctness & Requirements (25)** — Auth works, CRUD works, filters work, deployed
- **Code Quality (20)** — Clear structure, readable, types where sensible, linting
- **API & Data Design (15)** — Sensible models, status codes, validation
- **Security & Secrets (10)** — Hashing (bcrypt/argon2), JWT/session safety, CORS, env vars
- **DX & Docs (10)** — Clean README, run scripts, .env.example, diagram
- **Testing (10)** — Basic tests pass locally/CI
- **Deployment (5)** — Stable DO deployment, health checks
- **Bonus Features (up to 5)** — OpenAI due date, real time, Docker

Guardrails

- May use UI libs (e.g., Tailwind, shadcn/ui) but **no full CRUD templates/starters**.
- Must write your own auth flow; ok to use a small JWT lib.
- If using the OpenAI bonus, **explain prompt + safeguards** and keep the key server side.

At Orbtronics, we value practical execution, thoughtful design choices, and innovation. This challenge is your chance to show how you approach a real-world build under constraints.