Full-Stack Developer Assessment

ASSESSMENT INSTRUCTIONS

Position: Full Stack Developer

STRICT 24-HOUR DEADLINE - NO EXTENSIONS GRANTED

- **Start time begins when the candidate receives the dataset.**
- **Submission must be done via GitHub commit timestamp + shared deployment links before the deadline.**
- **LATE SUBMISSIONS ARE DISQUALIFIED.**
- **Do not forget to fill out the Google form sent to you in your email. Not submitting the form will disqualify you from the selection process. For your convenience, we are adding the form link below:**

➡ **FORM LINK: https://forms.gle/pKirxjECycijMc5N8.**

Submission Requirements:

- **Format: Word document (.docx)**
- **File Name: Full Stack Developer_Assessment_[Your_Name] _August 2025**
- **Email to: [career@purplemerit.com] with subject "PM Assessment Submission - [Your Name]"**

Assessment Overview

GreenCart Logistics – Delivery Simulation & KPI Dashboard

GreenCart Logistics is a fictional eco-friendly delivery company that operates in urban areas. You have been hired to **build an internal tool** that simulates delivery operations and calculates KPIs based on custom company rules. This tool will help managers experiment with staffing, delivery schedules, and route allocations to see their effect on profits and efficiency.

Assessment Overview

This is a **24-hour timed assessment** designed to evaluate your full-stack development skills across backend, frontend, database design, API development, business logic implementation, UI/UX, and deployment.

Tech Stack Requirements

Backend: Node.js + Express **OR** Python (Flask/Django)
Database: MongoDB or PostgreSQL (Cloud-hosted)
Frontend: React (Hooks) **OR** Vue.js
Authentication: JWT or session-based
Charts: Chart.js or Recharts

In addition to the suggested technology stack, you are free to use any other programming languages, frameworks, or tools of your choice to develop this project.

PART 1: Frontend

- **Stack:** React (Hooks) or Vue.js
- **Pages Required**:
    1. **Dashboard**: Displays
        - Total Profit
        - Efficiency Score
        - On-time vs Late Deliveries (chart)
        - Fuel Cost Breakdown (chart)
    2. **Simulation Page**:
        - Form to set simulation inputs (drivers, start time, max hours/day)
        - "Run Simulation" button → fetches backend results & displays new KPIs.
    3. **Management Pages**: CRUD interfaces for Drivers, Routes, Orders
- **UI Requirements**:
    - Responsive design (desktop & mobile)
    - At least 2 charts (Chart.js / Recharts)
    - Clear display of simulation results
    - Real-time updates after simulation

PART 2: Backend

1. Backend (Custom Logic Required)

- **Stack:** Node.js + Express OR Python (Flask/Django)
- **Database:** MongoDB or PostgreSQL
- Load initial data from a provided CSV/JSON file containing:
    - **Drivers** (name, current shift hours, past 7-day work hours)
    - **Routes** (route ID, distance in km, traffic level, base time)
    - **Orders** (order ID, value_rs, assigned route, delivery timestamp)
- Implement CRUD endpoints for:
    - Drivers
    - Routes
    - Orders
- Implement **Simulation Endpoint** that:

1. Accepts inputs:
    - Number of available drivers
    - Route start time (HH:MM)
    - Max hours per driver per day
2. Reallocates orders to drivers based on input.
3. Applies **company rules** (see below) to calculate KPIs.
4. Returns calculated KPI results to the frontend.

Must include **data validation & error handling**:

- Invalid/missing parameters should return a structured JSON error response with appropriate HTTP status codes.

- Backend must handle invalid/missing request parameters gracefully (with proper HTTP status codes & error messages).
- Example: If a simulation input is negative or exceeds driver count, return a structured error response.

2. Company Rules (Custom & Proprietary)

These rules are **only in this document** — not public knowledge.

1. **Late Delivery Penalty**: If delivery time > (base route time + 10 minutes), apply **₹50 penalty** to that order.
2. **Driver Fatigue Rule**: If a driver works **>8 hours in a day**, their delivery speed decreases by **30%** the next day.
3. **High-Value Bonus**: If order value > ₹1000 AND delivered on time → add **10% bonus** to order profit.
4. **Fuel Cost Calculation**:
    - Base cost: ₹5/km per route
    - If traffic level is "High" → +₹2/km fuel surcharge
5. **Overall Profit**:
    - Sum of (order value + bonus – penalties – fuel cost)
6. **Efficiency Score**:
    - Formula: Efficiency = $OnTime\frac{Deliveries}{Total\ Deliveries} \times 100$

PART 3: Requirements

- **Authentication**: Simple login system (JWT or session-based) so only managers can use the tool.
- Password hashing (bcrypt/argon2).
- JWT or session-based auth.
- Environment variables in. env (excluded via. gitignore).
- CORS configuration for secure API access.
- **Data Persistence**: Simulation results should be saved in DataBase with timestamp. Include ability to view past simulation history.
- **Testing**: At least 5-unit tests for backend logic. Bonus for integration tests or frontend unit tests.
- **Deployment:**
    - **Backend:** Render / Railway / Heroku / Vercel (public API endpoints)
    - **Frontend**: Vercel / Netlify (public URL)
    - **Database:** Cloud-hosted (MongoDB Atlas or Neon/PostgreSQL)

PART 4: Documentation Requirements

- **README.md** must include:
    1. Project Overview & Purpose.
    2. Setup steps
    3. Tech Stack Used
    4. Setup Instructions (Frontend & Backend)
    5. Environment Variables (list without actual values)
    6. Deployment Instructions
    7. API Documentation:
        - Swagger/OpenAPI spec **or** Postman Collection link.
        - Example requests & responses

PART 5: Deliverables

1. **GitHub Repository (Frontend & Backend)**
    - A single GitHub repository **containing both frontend and backend source code**, organized into **separate folders** (e.g., /frontend and /backend).
    - Must include **proper commit history** — bulk single commits (e.g., "final commit") will be considered poor practice.
    - All sensitive credentials must be stored in **.env files** and excluded from the repository using. gitignore.
    - The repository must be **public** so the review panel can access it without special permissions.
2. **Live Deployment Links**
    - **Frontend Deployment**: Must be hosted on **Vercel, Netlify, or equivalent**.
    - **Backend Deployment**: Must be hosted on **Render, Railway, AWS, or equivalent**, and accessible via public API endpoints.
    - **Database**: Must be cloud-hosted (e.g., MongoDB Atlas, Neon/PostgreSQL).
    - All URLs for deployed applications (frontend, backend, API docs) must be **clearly listed** in the README.
3. **README File (Setup Instructions & Tech Stack)**
    1. A **comprehensive README** file in Markdown format.
    2. Must include:
        - **Project Overview** — what the application does and its purpose.
        - **Tech Stack** — all major libraries, frameworks, and tools used.
        - **Setup Instructions** — step-by-step guide to run the project locally (both frontend & backend).
        - **Environment Variables** — list of variables required in. env (without revealing actual values).
        - **Deployment Instructions** — how the app was deployed (platform, steps taken).
        - **API Documentation** — list of available endpoints with request/response examples.

4. **Screen-Recorded Walkthrough Video (8–10 minutes)**
    - A clear, **narrated** screen recording showing:
        - User login & role-based access control in action.
        - Creating, editing, and deleting tasks.
        - AI simulation feature generating summaries and tags.
        - Filtering/searching tasks.
        - How the app looks and works on both desktop & mobile views.
        - Quick backend API demonstration (using Postman or browser).
        - Live deployment links in action.
    - Must be uploaded to **Google Drive (public link), YouTube (unlisted), or any shareable video link**.
5. **Bonus (Not Mandatory, but Adds Weight to Your Application)**
    - Unit & integration tests for backend APIs and/or frontend components.
    - Dockerized setup for both backend and frontend.
    - CI/CD pipeline configuration (GitHub Actions, GitLab CI, etc.).
    - Caching for simulation results

6. Time Constraint & Submission

- **Time Limit:** 24 hours from the time you receive the dataset & credentials.
- **Submit:**
    1. GitHub Repo Link
    2. Live Deployment Links
    3. Walkthrough Video Link
- Late submissions will be disqualified unless pre-approved.