

Backend Developer Intern Assessment

ASSESSMENT INSTRUCTIONS

Position: Backend Developer Intern

- ❖ **Start Time:** 29th December 2025, 11:00 AM (IST)
- ❖ **Deadline:** 31st December 2025, 11:00 AM (IST)

STRICT 48-HOUR DEADLINE - NO EXTENSIONS GRANTED

Submission Requirements:

- **Format:** Word document (.docx) , Submission must be done via GitHub commit timestamp + shared deployment links before the deadline.
 - **File Name:** Backend Intern _ Assessment _[Your _ Name] _ December 2025
 - **Email to:** [career@purplemerit.com] with subject "Backend Intern Assessment Submission - [Your Name]"
- ❖ LATE SUBMISSIONS ARE DISQUALIFIED.**

Project Title

Mini User Management System

Assessment Brief

Build a **User Management System** a web application that manages user accounts with different roles and permissions. The system must support user authentication, role-based authorization, and basic user lifecycle management. This assessment evaluates your understanding of authentication flows, API security, role-based access control (RBAC), and clean architectural patterns.

Assessment Overview

This is a **48-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)

Authentication: JWT or session-based

Password Hash: bcrypt or argon2

Deployment: Render/Railway (Backend), Vercel/Netlify (Frontend)

Your Task

Build a **full-stack web application** with the following features:

1. Backend Requirements

- **Stack:** Node.js + Express OR Python (Flask/Django)
- **Database:** MongoDB or PostgreSQL

Authentication

- User signup with email, password, full name
- Email format validation
- Password strength validation
- Authentication token on signup
- User login with email and password
- Credentials verification
- Authentication token on login
- Endpoint to get current user information
- User logout functionality

User Management - Admin Functions

- View all users with pagination
- Activate user accounts
- Deactivate user accounts

User Management - User Functions

- View own profile information
- Update full name and email
- Change password

Security Requirements

- Password hashing with bcrypt or argon2
- Protected routes with authentication verification
- Role-based access control (admin/user)
- Input validation on all endpoints
- Consistent error response format
- Proper HTTP status codes
- Environment variables for sensitive data (JWT secret)

2. Frontend Requirements

Login Page

- Email and password input fields

- Client-side form validation
- Redirect to dashboard on success
- Error message display
- Link to signup page

Signup Page

- Full name, email, password, confirm password inputs
- Required field validation
- Email format validation
- Password strength validation
- Password confirmation matching
- Server-side error display
- Redirect to login on success

Admin Dashboard

- Table displaying all users
- Columns: email, full name, role, status, actions
- Pagination (10 users per page)
- Activate user button
- Deactivate user button
- Confirmation dialog before actions
- Success/error notifications

User Profile Page

- Display user information
- Edit full name and email
- Change password section
- Save and cancel buttons
- Success/error messages after updates

Navigation Bar

- Display logged-in user name
- Display user role
- Role-based navigation links
- Logout button
- Redirect to login after logout

Protected Routes

- Prevent unauthenticated access
- Admin-only pages restricted to admins
- Redirect to login for unauthorized users

User Interface Components

- Input fields with validation messages
- Primary action buttons
- Secondary action buttons
- Destructive action buttons
- Loading spinners during API calls
- Toast notifications (success/error/info)
- Modal dialogs for confirmations
- Pagination for tables
- Clear error messages
- Responsive design (desktop & mobile)

3. Database Requirements

User Collection

- Unique email address (no duplicates)
- Securely hashed password
- Full name
- Role (admin or user)
- Status (active or inactive)
- Created date (auto-managed)
- Updated date (auto-managed)
- Last login timestamp

4. 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.
- **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)

5. Documentation Requirements

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

6. 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)**
 - A **comprehensive README** file in Markdown format.
 - 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 (3 – 5 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

7. Time Constraint & Submission

- **Time Limit:** 48 hours from the time you receive dataset & credentials.
- Submit:
 1. GitHub Repo Link
 2. Live Deployment Links
 3. Walkthrough Video Link
- Late submissions will be disqualified unless pre-approved.

We look forward to reviewing your comprehensive approach. Best of luck!