NOT@MRP Backend Developer Intern Assignment

Duration: 3 Days or till the end of 07/09/25

Choose ONE task from the options below

Tech Stack: Node.js, Express.js, MongoDB, Basic Authentication

---

📋 TASK OPTION 1: Backend for Inventory & Billing Management System

Build a simple backend system for small businesses to manage products, customers, vendors, and basic transactions.

Core Requirements

1. User Authentication

- username or email /password login (JWT)
- Basic session management
- Each business user manages their own data

2. Product Management

Product Schema:

```
{
  name: String,
  description: String,
  price: Number,
  stock: Number,
  category: String,
  businessId: String
}
```

- Add, edit, delete, and list products
- Simple stock tracking (increase/decrease)
- Basic search by name or category

3. Customer & Vendor Management

Customer/Vendor Schema:

```
{
  name: String,
  phone: String,
  email: String,
  address: String,
  type: 'customer' or 'vendor',
  businessId: String
}
```

- Add, edit, delete customers and vendors
- Simple list and search functionality

4. Transaction Management

Transaction Schema:

```
{
  type: 'sale' or 'purchase',
  customerId: String, // for sales
  vendorId: String,   // for purchases
  products: [{
    productId: String,
    quantity: Number,
    price: Number
  }],
  totalAmount: Number,
  date: Date,
  businessId: String
}
```

- Record sales (to customers) and purchases (from vendors)
- Automatically update product stock
- Calculate totals

5. Simple Reports

- List all transactions with filters (date, type)
- Current inventory with stock levels
- Customer/vendor transaction history

API Endpoints Required

Authentication:

```
POST /login
POST /register
GET /logout
```

Products:

```
GET /products
POST /products
PUT /products/:id
DELETE /products/:id
```

Customers/Vendors:

```
GET /contacts
POST /contacts
PUT /contacts/:id
DELETE /contacts/:id
```

Transactions:

```
GET /transactions
POST /transactions
```

Reports:
GET /reports/inventory
GET /reports/transactions

🚚 TASK OPTION 2: Delivery Partner System

Build a backend system for managing delivery partners with simple order assignment and basic tracking.

Core Requirements

1. Delivery Partner Management

Partner Schema:
```
{
  name: String,
  phone: String,
  email: String,
  vehicleType: String,
  status: 'available' or 'busy' or 'offline',
  currentLocation: {
    latitude: Number,
    longitude: Number
  }
}
```

- Partner registration and login
- Update availability status
- Update current location

2. Order Management

Order Schema:
```
{
  customerName: String,
  customerPhone: String,
  pickupAddress: String,
  deliveryAddress: String,
  orderValue: Number,
  status: 'pending' or 'assigned' or 'picked' or 'delivered',
  assignedPartnerId: String,
  createdAt: Date,
  deliveredAt: Date
}
```

- Create new orders (simulate random orders)
- Assign orders to available partners
- Track order status updates

3. Simple Order Generation
- Create a function that generates random orders every 2-3 minutes
- Include random customer details and addresses
- Vary order values between ₹100-₹500

4. Basic Payment System

Payment Schema:
```
{
  partnerId: String,
  orderId: String,
  baseAmount: 50, // ₹50 per delivery
  bonusAmount: 0,  // ₹10 bonus if delivered in < 30 minutes
  totalAmount: Number,
  deliveryTime: Number, // in minutes
  date: Date
}
```

- ₹50 for each completed delivery
- ₹10 bonus if delivered within 30 minutes of pickup
- Simple earnings calculation

5. Partner Dashboard Data
- Today's completed deliveries
- Total earnings for the day
- Current assigned orders
- Available orders for pickup

API Endpoints Required

Partner Auth:
POST /partner/login
POST /partner/register

Partner Management:
GET /partner/profile
PUT /partner/status
PUT /partner/location

Orders:
GET /orders/available
POST /orders/:id/accept
PUT /orders/:id/status
GET /orders/my-orders

Earnings:
GET /earnings/today
GET /earnings/history

System:
POST /orders (for creating random orders)

Technical Requirements (Both Tasks)

1. Basic Architecture

```
src/
├── app.js            # Main application file
├── routes/           # API route files
├── models/           # MongoDB schemas
├── controllers/      # Route handlers
├── middleware/       # Basic auth middleware
└── utils/            # Helper functions
```

2**. Database (MongoDB)**
- Use Mongoose for database operations
- schema design with relationships
- No complex aggregations required

3. **Authentication**
- session-based auth (JWT required)
- Basic password hashing using bcrypt
- Simple middleware to protect routes

4. **Error Handling**
- Basic try-catch blocks
- error responses
- Console logging for debugging

5. **API Documentation**
- Create a simple README with API endpoints
- Include example requests/responses
- Postman collection (optional)

Deliverables

1. GitHub Repository with complete source code (must be public), extra point if you deploy it on a free service like render, .etc,
2. API Documentation
3. Demo Data (extra point if you add an explanation video of the project [up-to 2 min])

Optional: Postman collection, validation, seeding script, logging

Evaluation Process

We will test your app by cloning repo, running it, and verifying endpoints, code quality, and documentation.

Submission Guidelines

Timeline: Duration: 3 Days or till the end of 07/09/25

Submission Link: https://forms.gle/yJx6DjbzjzSsfR5y5

Good Luck! We're excited to see your problem-solving approach and coding style. Focus on building something that works well rather than trying to implement every possible feature. Quality over complexity!

- NOT@MRP Team