4.NodeJS Assignment - Industry standard project

CENTRALOGIC

12/5/24

> 💡 **Github code reference using pool**
>
> https://github.com/gauravwani127/NODEJS_training/tree/main/postgresWithSequelize

Objective:

To design and implement an Organization Invoice Management System that allows organizations to register, login, invite users with different access levels, manage invoices, payments, and Statements of Work (SOWs) separately.

Omit Register, Login and access management . Start actual project flow directly on first priority. Registration flow is optional for the assignment.

Large Picture (Detailed info in Scenario)

1. An organization owns this platform for invoice generation
2. Organization has different customers (clients) . One of which signs a SOW with the organization
3. Some points are discussed in SOW contract → Like how invoice will be generated, when it will be generated, in how many installments it will be , All particulars and all.
4. When the invoice generation date approaches (as discussed in SOW) . , an invoice is genrated . Organization can decide which SOW items he can add in Invoice .(Totally configurable) . Once confirmed . He will make paymet to that invoice (Its upon him , how much payment he want to make) (Totally configurable)
5. User can see all invoices , all sows , all payments and reminders
6. FIN !

System Components:

1. **Organization Setup:**
   - Each organization can register itself with the following details:
     - ID
     - GST No.
     - PAN No.
     - Legal Organization Name
     - Invoice Template ID
     - Short Name
     - Contact Name
     - Display Name
     - Email
     - Address ID
     - Phone

Id	GST No.
	PAN No.
Legal OrgName	InvoiceTemplateId
ShortName	ContactName
DisplayName	Email
AddressId	Phone

2. **Customer Management:**
   - Organizations can manage their clients/customers with details such as:
     - ID
     - OrganizationId
     - MSA Valid From
     - MSA Valid Upto
     - Legal Name
     - NDA Signed On
     - Short Name
     - NDA Valid Upto
     - Address ID
     - NDA Valid From
     - Display Name

- NDA and MSA signing statuses

id	MSAValidFrom
	MSAValidUpto
LegalName	NDASignedOn
ShortName	NDAValidUpto
AdddressId	NDAValidFrom
DisplayName	IsNDASigned
IsMSASigned	
MSASignedOn	

3. **SOW (Statement of Work) Management:**

- Allows creation and management of SOWs with details including:
  - ID
  - Invoice Email Addresses
  - Customer ID
  - Customer PO Number
  - Title
  - Customer SO Number
  - Validity Period
  - Total Value
  - Currency

Sow is basically the deal information(Amount) that need to be approved (When approved – invoice and eventually payment is done against it)

4. **SOW Payment Plan:**

- Defines payment schedules for SOWs, including planned invoice dates and amounts.
- **This paymentPlan will be a list-like that will contain how and in how many dues SOW will be paid**

id	
sowId	
CustomerId	
PlannedInvoiceDate	
TotalActualAmount	

5. **SOW Payment Plan LineItem**

id	
SOWPaymentPlanId	
SOWId	
OrderId	
Particular	
Rate	
Unit	
Total	

6. **Invoice Management:**

- Automatically generates invoices based on the SOW payment plan. Details include:
- ID
- Total Invoice Value
- SOW ID
- Status (Drafted, Cancelled, Approved)
- Invoice Sent On
- Customer ID
- Payment Received On
- Invoice Version Number

Now when the date (PlannedInvoiceDate on the SOWPaymentPlan gets hit**) an invoice will be generated**

id	
	TotalInvoiceValue
SOWId	Status – Drafted , Cancelled , Approved
SOWPaymentPlanId	InvoiceSentOn
CustomerId	PaymentReceivedOn
InvoiceVersionNo	PayementId
InvoiceAmount	
InvoiceTaxAmount	

7. **Invoice Line Items:**

- Specifies individual line items on invoices, including particulars, rates, units, and totals.

id
InvoiceId
OrderNo
Particular
Rate
Unit
Total

8. **Payment Management:**
   - Tracks payments made against invoices, including details such as payment date, foreign exchange amount, currency, and payment status.

   Payment

id	
PaymentDate	
ForEx Amount	
Currency	
IndianAmount	
InvoiceId	
isFullPayment	
BankPayment	
Details	

Scenario:

- **Organization:** CentraAPIs - **CentraAPIs is an organization owning the rights for this system –Register, Login, Invite** . Organization can have multiple clients

```
{
  "id": "ORG_001",
  "gstNo": "GST123456789",
  "panNo": "ABCDE1234F",
  "legalOrganizationName": "CentraAPIs Pvt. Ltd.",
  "invoiceTemplateId": "TPL_001",
  "shortName": "CentraAPIs",
  "contactName": "John Doe",
  "displayName": "CentraAPIs Pvt. Ltd.",
  "email": "info@centraapis.com",
  "addressId": "ADDR_001",
  "phone": "+1234567890"
}
```

- **Client:** AmperX AmperX is a client to organization (CentraAPIs) but AmperX can be client to other organizations . When Client is created in your system. Client will have fields as mentioned .

```
{
  "id": "CLI_001",
  "orgId": "ORG_001", //Referenced to Organization table
  "MSAValidFrom": "2024-03-10",
  "MSAValidUpto": "2025-01-10",
  "LegalName": "AmperX Pvt. Ltd.",
  "NDASignedOn": "2024-03-10",
  "ShortName": "Amperx",
  "NDAValidFrom": "2024-01-10",
  "NDAValidUpto": "2025-01-10",
  "AddressId": "ADDRESS_ID_001", //Referenced to Address table
  "DisplayName": "Centra APIs Organization",
  "IsNDASigned": true,
  "IsMSASigned": true,
  "MSASignedOn": "2024-03-10"
}
```

- **MSA:** Valid from 10-03-2024 to 10-01-2025. Client field
- **SOW - Total Value:** 3,000

```
{
"id": "SOW_001",
"invoiceEmailAddresses": ["billing@customer.com", "accounts@customer.com"],
"customerId": "CUST_001",
"customerPONumber": "PO_123456",
"title": "Web Development Project",
"customerSONumber": "SO_654321",
```

```
"validityPeriod": {
"validFrom": "2024-03-10",
"validUpto": "2025-01-10"
},
"totalValue": 3000,
"currency": "USD"
}
```

- **SOW Payment Plan:**

```
[
        {
        "id": "PAY_PLAN_001",
        "sowId": "SOW_001",
        "customerId": "CUST_001",
        "plannedInvoiceDate": "2024-03-10",
        "totalActualAmount": 1000
        }
        {
        "id": "PAY_PLAN_002",
        "sowId": "SOW_001",
        "customerId": "CUST_001",
        "plannedInvoiceDate": "2024-10-10",
        "totalActualAmount": 1500
        }
        {
        "id": "PAY_PLAN_003",
        "sowId": "SOW_001",
        "customerId": "CUST_001",
        "plannedInvoiceDate": "2024-03-10",
        "totalActualAmount": 500
        }
]

//So these are installments in which the sow 'SOW_001' will be paid
```

  - Three installments: March 10, 2024 (abc), October 10, 2024 (xyz), January 10, 2025 (pqr) of 1000 1500 500 respectively
  - Each installment includes specific services with rates and quantities.
  - Payment Plan is kind of installments in which your SOW amount is divided.
- **SOW payment Plan Item (Each SOW PaymentPlan contains the breakage of how the installment is divided among entities , that is called Plan Item)**

  SOWPaymentPlanItem contains the particulars

```
[{
  "id": "LINE_ITEM_001",
  "sowPaymentPlanId": "PAY_PLAN_001",
  "sowId": "SOW_001",
  "orderId": "ORD_001",
  "particular": "App Development",
  "rate": 250,
  "unit": 2,
  "total": 500
},
{
    "id": "LINE_ITEM_002",
  "sowPaymentPlanId": "PAY_PLAN_001",
  "sowId": "SOW_001",
  "orderId": "ORD_001",
  "particular": "Website Maintenance",
  "rate": 500,
  "unit": 2,
  "total": 1000
}
] //SOW payment Plan Item = Breakage of SOW payment plan as we see in bill

//The above example was for SOWPaymentPlan -- PAY_PLAN_001.. There will be example similar for other SOWs and other
paymentPlans
```

- **Invoice**

  Invoice will be generated on the date as mentioned in sowpaymentPlan. (Hint. Create one api that will generate invoice from the SOW . Internally - API will fetch all the SOWs that have invoiceGenerationDate of today , Convert them into invoices with status = Drafted. So user will choose which line item he want to generate Invoice for.
- **Invoice Line Item , Similar flow like as that of SOW .**

  Similar to sow payment planLine Item , there are line items for Invoice as well
- **Payment**

  User can decide for which which invoice he want to make payment and that item will be saved in the sow.

- **Dashboard (Optional)**
- **Reminder (Optional)**

Assignment Tasks:

1. Design a user-friendly interface for:
    - Organization registration and login
    - Client management
    - SOW creation and management
    - Invoice generation and management
    - Payment tracking
2. Ensure data security and access control mechanisms to protect sensitive information.
3. Test the system thoroughly to ensure functionality and reliability.

.

Assignment

Below questions are not to be documented in assignment , but requested to search for answers to enhance knowledge and develop a sense of curiosity in the subject.

1. What are DTOs ? Can we filter out the response from database, tpo send only specific fields to the endUser?
2. How foreign keys are referenced in Sequelize? Can we fetch the same way using foreign key?
3. Try encapsulating the functions in class and try to build the application? What edge do they provide over the way we are using ?
4. How modularization in code help ur application to scale ?
5. When data increase in abundancy , it becomes complex by time to fetch all data, What techniques we can use to tackle , What is elastic search?
6. In above application we use referencing by key? Is this Normalization all about ? Or it is just part of Normalization? What are other techniques?
7. What are logs ?How can you maintain logs in your system?
8. What are has many , one to one , many to many. Can you implement it in Sequelize?
9. What do you mean by automapper ?
10. If in above application , something data mistakenly deletes or updates , How can we recover it back? What do you mean by versioning?

JavaScript basics

I suggest solving atleast one question every day-

https://exercism.org/tracks/javascript

https://javascript.info/

https://learning.postman.com/docs/sending-requests/requests/

How to submit assignment guidelines?

Theoretical questions are not to be submitted, Check documentations for finding answers to them

For 1. Coding question to be sent by pushing it into a public git respository . Exclude Node_Modules using gitignore

Maintain a diffeernt folder for Outputs , put here ss of the resposses of the apis given in the requirement

Submit a detailed :

- ***Screen capture (Record) the flow of the Project , using debugger***
- ***Put the code in a public git repository and share it***
- ***Attach screenshots of the outputs***
- ***Use best practices as they were followed in last session (Modular structure)***

Deadline: 10/6/2024

Note: Feel free to reach out for any clarifications or assistance during the assignment

+

Documentation is the key, Happy Learning---

CentraLogic

Gaurav Wani

Team Lead | CentraLogic Consultancy