B2B Platform MVP

#### Problem Statement

B2B is planning to work with businesses by registering them and creating their accounts on the B2B platform.In turn, a business owner whose business is registered on the B2B platform, can go ahead and create user accounts for their businesses’ department heads(for example soft drinks department, toiletries department, etc), who are the actual active users of the platform.

In each business, the department heads are responsible for monitoring the status of the inventory, and are required by the business owner to make orders using the B2B platform each time before the inventory in their department runs out.

On a typical day from its many businesses, the B2B platform sometimes gets up to 1,000 requests per minute from different department heads of multiple businesses, all makingorders through the platform. B2B stores this order data in a Postgres database.

However, in its future product plan, B2B plans to start offering a buy now pay later service to its registered businesses, where businesses will be able to make orders for inventory, and can pay for it later.

To achieve this, the B2B platform has started logging transaction data into MongoDB. B2B logs the following data of a transaction/order: businessID, amount, date, status The platform intends to later use this data to calculate the credit score of a business, to determine how much credit a business can access, using the following formulae:

Total transactions or order amount / ( Number of transactions \* 100)

Also, the government tax authority has introduced a new law, which requires the B2B platform to make POST requests to an API it has provided for each order the B2B platform receives, so that it can log this data for tax purposes.The government tax authority STRICTLY dictates that the details of each order must be passed to its API.

#### Functional Requirement

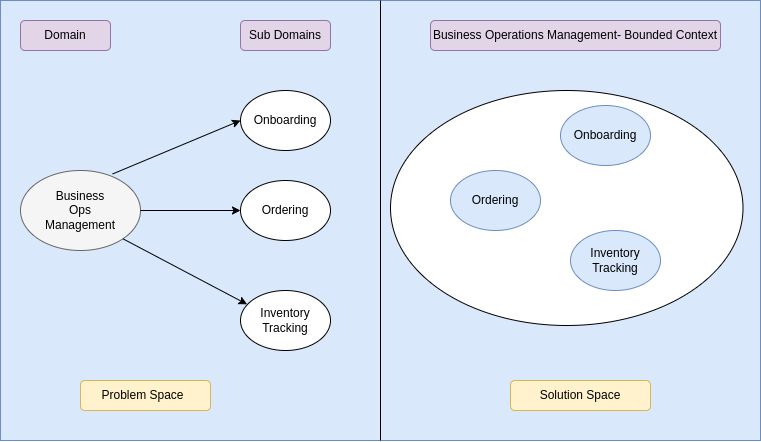
1. Curate orders from each businesses’ department lead
2. Process and store the order details as described above
3. Provide an API endpoint for a business to get their credit score
4. Provide an API endpoint for a business to fetch the order details (total number of orders, total amount of orders, total number of orders today, total amount of orders today)
5. Create a React App to display the orders of each business in a table. When one record is clicked a modal should pop up to show the order details
6. Deploy the backend solution as a docker image
7. Bonus points if you can setup a simple CI/CD pipeline to deploy the working frontend app on netlify and provide a preview link

#### Analysis

The **Business Domain** is a B2B business operations management platform that provides business workflow automation, Onboarding, Order management, Inventory management services from businesses to businesses. From this understanding from the problem statement, the problem to solution space description will be described as a business Ops Manager.

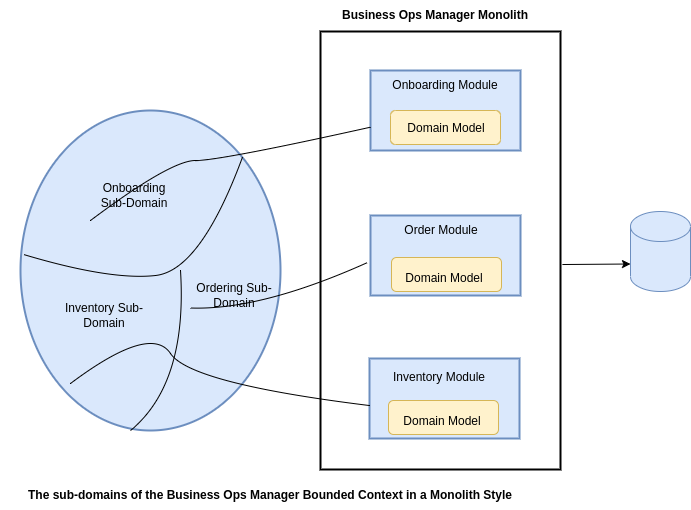
The **Sub Domains** are identified to be the following; Onboarding, Purchase Ordering, Inventory handling.

The **Bounded Context** is the carving out of the solution space for the business domain and its subdomains representing the capture of the core business problem and the direction of what their solution looks like. We have separate domain spaces for each of our subdomains and so our bounded context will be mapped to a specific bounded context.



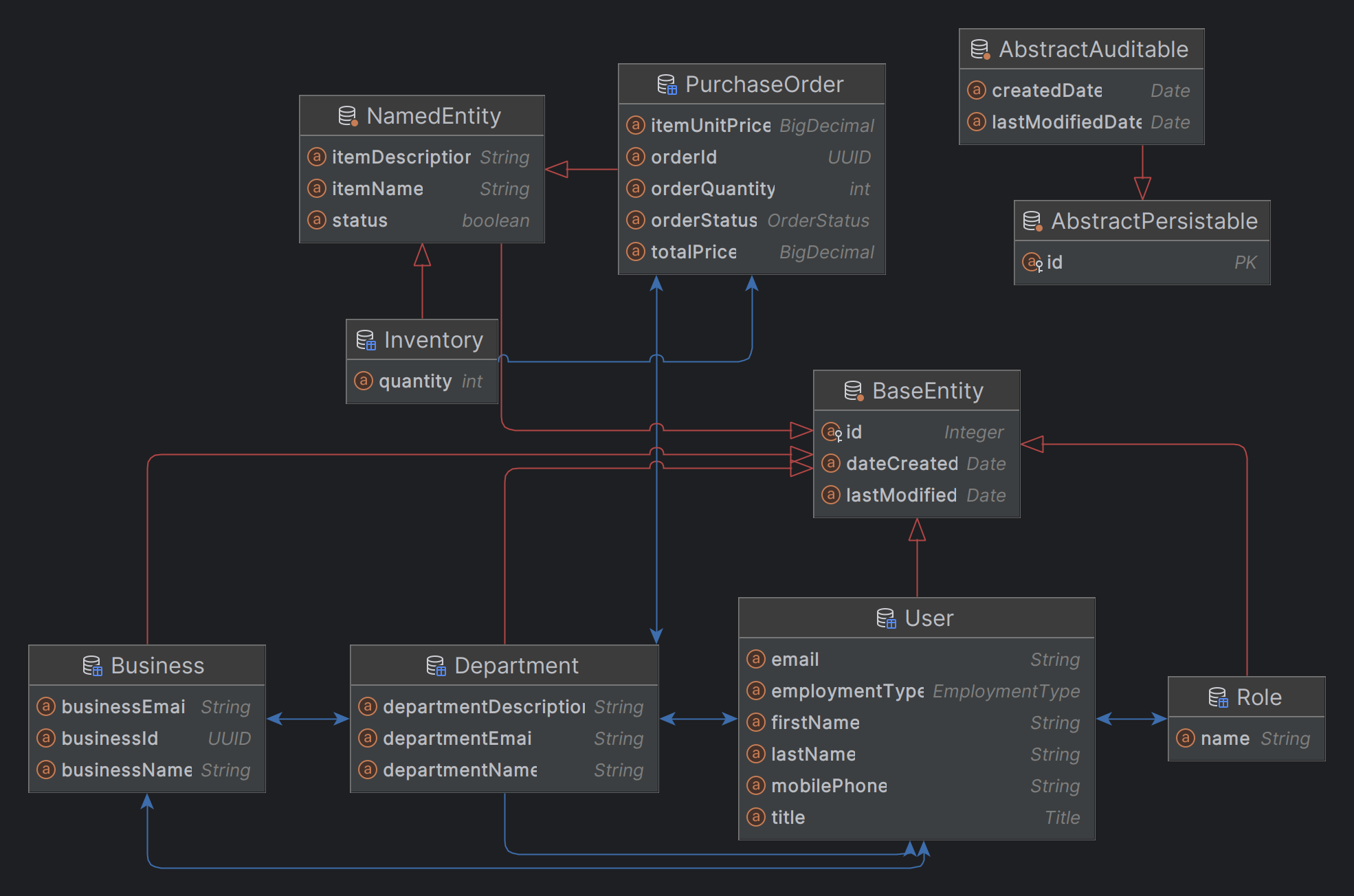
Within the scope of our Sub-Domain/Bounded Contexts, we have identified the following business areas:

* Onboarding:
  + Open Business Account
  + Users Setup
  + Department Setup
* Ordering: To maintain service delivery to its customers, a business orders goods and services from suppliers.
  + Raise purchase request
  + Process purchase request approval
  + Compute credit score
  + Generate order report
* Inventory Tracking: As stocks are getting sold, the system tracks threshold and triggers signals when breached.
  + Trigger signal when quantity is less than threshold



#### Design

**Domain Model**



**Data Access | Repository**

Repository pattern is used to implement data access for the entities.

| **s/n** | **Entity** | **Repository** |
| --- | --- | --- |
| 1 | Business | BusinessRepository |
| 2 | User | UserRepository |
| 3 | Department | DepartmentRepository |
| 4 | Purchase Order | PurchaseOrderRepository |
| 5 | Inventory | InventoryRepository |

**Service Layer**

Service (facade pattern) interface is used to expose all operations that are to be implemented in service implementation class.

| **s/n** | **B2BService** | **B2BService Implementation** |
| --- | --- | --- |
| 1 | Business Operations | 1. GenericResponse<BusinessRecord> openBusinessAccount(BusinessRecord businessRecord); 2. GenericResponse<UserRecord> addStaffToABusiness(Integer id, UserRecord userRecord); 3. GenericResponse<DepartmentRecord> createDepartmentForABusiness(Integer id, DepartmentRecord departmentRecord); 4. Optional<Business> getByEntityId(Integer id); 5. Optional<Business> getByBusinessId(String businessId); |
| 2 | Department Operations | 1. GenericResponse<DepartmentRecord> addStaffToABusinessDepartment(Integer departmentId, Integer userId); 2. GenericResponse<String> calculateCreditScore(Integer businessId); 3. GenericResponse<DepartmentRecord> addStaffToABusinessDepartment(Integer departmentId, UserRecord userRecord); |
| 3 | Order Operations | 1. GenericResponse<PurchaseOrderRecord> createPurchaseOrder(Integer departmentId, PurchaseOrderRecord purchaseOrderRecord); 2. GenericResponse<PurchaseOrderSummary> getDepartmentPurchaseOrderReport(Integer departmentId); |

**Web API**

The web layer is implemented in a RESTful style architectural pattern relieving it of any UI responsibility. The api documentation is a postman collection and will be added as a link.

#### Testing

#### Installation Requirement

The following softwares is required to be available on the host machine in order to run this software.

1. Java >= JDK 17
2. Postgres SQl Database latest

Technology Stack include:

1. Spring Framework
2. Spring Boot
3. Spring MVC
4. Spring DataJPA

Installation Steps:

Database Script

B2BPlatform Jar