Domain Driven Design Document

[Training Management System]

I. Introduction

A. Purpose/Objective

(Define the goals and primary objective or mission of this DDD documentation for Training Management System)

B. Domain Scope

(Define the area/end users for the application)

C. Domain Description

(Describing the complete domain briefly including core business domains and functionalities)

D. Context

(Explain the boundaries of this analysis)

II. Strategic Design

A. Bounded Contexts

a. Define core bounded contexts within our training management system domain

Examples:

Training Context

User Context

Enrolment context

Assessment Context

Notification context

b. Every Contexts should be explained with these as follows:

a. Manages training sessions, materials, and trainers
b. Manages trainees, trainers, and their profiles
c. Manages enrollments, attendance, and completion status.
d. Manages quizzes, assessments, and scoring.
e. Manages alerts and communications (email/SMS)

B. **Context Map**

a. **Illustrate all relationships between the above defined bounded context.**

b. **Specify the proper type of relationships (Training session-Materials, trainers, Quizes)**

c. **Provide proper brief document for interactions using the relationships.**

Example: Training sessions are dependent on trainers and the trainers are dependent on materials.

C. Sub-Domains

d. **Identify the core, supporting and generic subdomains within the training management system**

Ex: Tabular or Document or visually represented

e. Give documents for identified domains that helps to prioritize development efforts and determine the proper appropriate strategies followed to be.

III. Tactical Design

A. **Product Context**

a. **Entities**

Training, Trainer, Session

b. **Value Objects**

Schedule, TrainingStatus

c. **Aggregates/ aggregate root**

Training (root)

d. **Domain Services**

TrainingSchedulerService

e. **Domain Events**

TrainingScheduled, TrainingCancelled

f. **Repositories**

TrainingRepository

g. **Factories**

TrainingFactory

h. **Application Services**

TrainingService.createTraining(), updateTraining()

B. **Customer Service**

a) **Entities**
b) **Value Objects**
c) **Aggregates/ aggregate root**
d) **Domain Services**
e) **Domain Events**
f) **Repositories**
g) **Factories**
h) **Application Services**

IV. **Implementation Considerations**

a. **Technology Considerations**

Technology Discussions (Spring Boot, Angular)

Frameworks (DDD)

Messaging System (RabbitMQ/Kafka)

Specific Data Persistence (ORM/OXM)

b. **Architectural Pattern**

Event Sourcing (Producer and Consumer)

SAGA (Transactions)

CQRS (Which Persistent Store)

Resilience Patterns: CircuitBreaker Pattern, RateLimiting (Controller Methods/Rest API)

Deployment: canary / BlueGreen patterns (Microservices)

c. **Testing**

Unit Testing (Junit5 & Jasmine)

Integration Testing (Mockito)

Domain Specific Testing (Stub)

Security Testing, Slice Testing, Repository Testing

d. **Security**

AuthN and AuthZ (LDAP, OAuth2.0, VMware Identity Manager/Workspace One)

Security using JWT, Web Security, Method Security

V. **Glossary**

a. Training: A program or session for skill development
b. Trainer: A user delivering the training
c. Trainee: A user receiving the training
d. Assessment: A quiz or assignment linked to training
e. Enrollment: A record of user participation in a training
f. Submission: A user's attempt at an assessment
g. Score: Evaluation metric for assessment

Example Illustrative Snippets:

Training Context – Entity

Entity: TrainingMast

Description: ---

Atttributes:

1. Training_ID
2. Training_Name
3. Training_Desc
4. Training_Price
5. Training_Category
6. Training_Image

UserContext :

1. Entities: User, Profile
2. Value Objects: Email, Role
3. Aggregates: User (root)
4. Domain Services: UserAccessService
5. Domain Events: UserRegistered, RoleChanged
6. Repositories: UserRepository
7. Factories: UserFactory
8. Application Services: UserService.createUser(), assignRole()