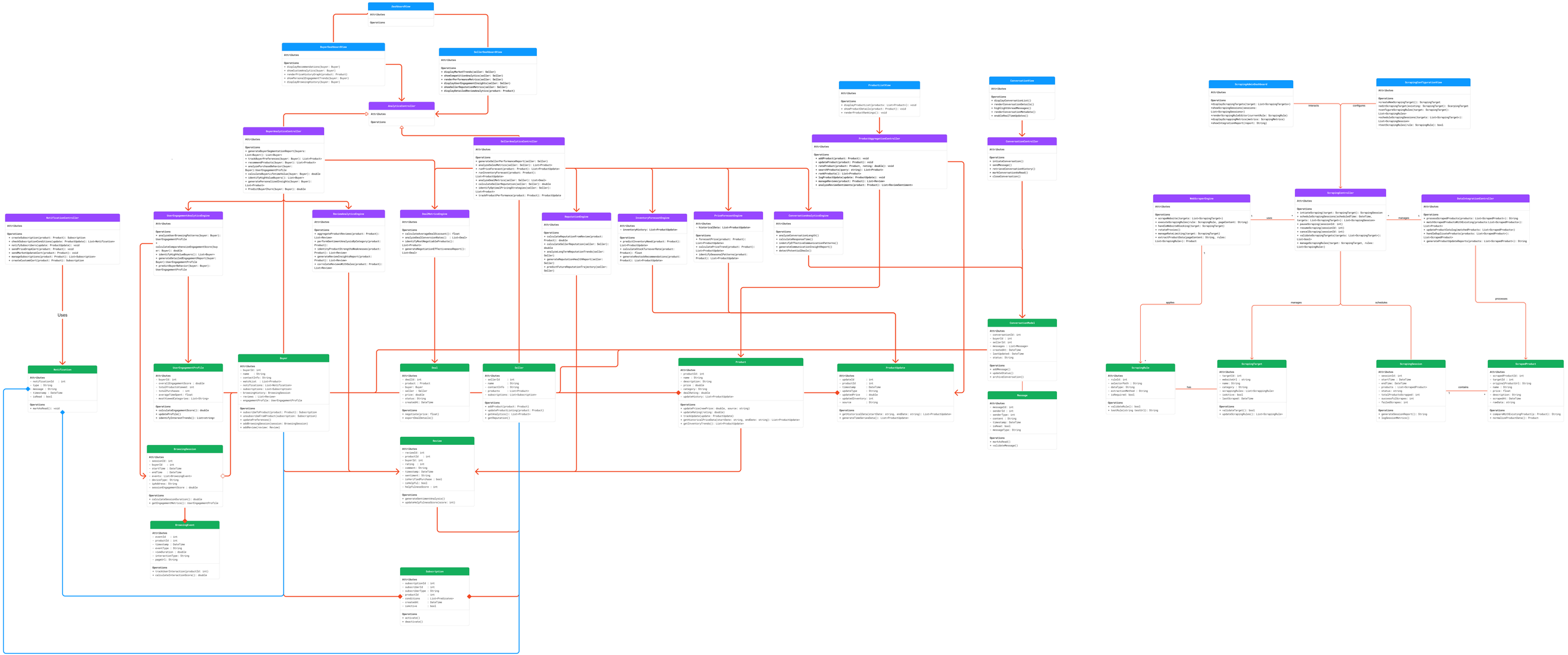
DashboardView
Attributes
Operations
BuyerDashboardView
Attributes
Operations
+ displayPersonalRecommendations(buyer: Buyer)
+ showBuyerPurchaseAnalytics(buyer: Buyer)
+ renderPriceHistoryGraph(product: Product)
+ showPersonalEngagementTrends(buyer: Buyer)
+ displayPromotingHistory(buyer: Buyer)
SellerDashboardView
Attributes
Operations
+ displayMarketTrends(seller: Seller)
+ showCompetitiveAnalytics(seller: Seller)
+ displayInventoryOptimization(seller: Seller)
+ displayUserEngagementInsights(seller: Seller)
+ showSellerReputationMetrics(seller: Seller)
+ displayDetailedRecommendations(product: Product)
AnalyticsController
Attributes
Operations
BuyerAnalyticsController
Attributes
Operations
+ generateBuyerSegmentationReport(buyers: List<Buyer>): List<Buyer>
+ trackBuyerPreferences(buyer: Buyer): List<Product>
+ recommendProducts(buyer: Buyer): List<Product>
+ analyzeBrowsingBehavior(buyer: BrowsingSession): BrowsingPattern
+ calculateBuyerLifetimeValue(buyer: Buyer): double
+ identifyHighValueBuyers(): List<Buyer>
+ generatePersonalizedInsights(buyer: Buyer): List<Product>
+ predictBuyerChurn(buyer: Buyer): double
SellerAnalyticsController
Attributes
Operations
+ generateSellerPerformanceReport(seller: Seller)
+ analyzeSalesMetrics(seller: Seller): List<Product>
+ runPriceForecast(product: Product): List<ProductUpdate>
+ runInventoryForecast(product: Product): List<ProductUpdate>
+ analyzeSellMetrics(seller: Seller): List<Deal>
+ calculateSellerReputation(seller: Seller): double
+ identifyOptimalPricingStrategies(seller: Seller): List<Product>
+ trackProductPerformance(product: Product): ProductUpdate
ProductListView
Attributes
Operations
+ displayProductList(products: List<Products>) :void
+ showProductDetails(product: Product): void
+ renderProductRankings() :void
ProductAggregationController
Attributes
Operations
+ addProduct(product: Product): void
+ updateProduct(product: Product): void
+ deleteProduct(product: Product): void
+ searchProducts(query: string): List<Product>
+ rankProducts(): List<Product>
+ logProductUpdate(update: ProductUpdate): void
+ manageReviews(product: Product): List<Review>
+ analyzeReviewSentiments(product: Product): List<ReviewSentiment>
ConversationView
Attributes
Operations
+ displayConversationList()
+ renderConversationDetails()
+ showNewMessages()
+ renderConversationMetadata()
+ enableRealTimeUpdates()
ConversationController
Attributes
Operations
+ initiateConversation()
+ sendMessage()
+ retrieveConversationHistory()
+ markConversationsAsRead()
+ closeConversation()
ScrapingWebDashboard
Attributes
Operations
+displayScrapingTargets(target: List<ScrapingTarget>)
+showScrapingSessions(sessions: List<ScrapingSession>)
+renderScrapingRuleEditor(currentRule: ScrapingRule)
+displayScrapingMetrics(metrics: ScrapingMetrics)
+showIntegrationReport(report: String)
ScrapingConfigurationView
Attributes
Operations
+createNewScrapingTarget():: ScrapingTarget
+editScrapingTarget(existing: ScrapingTarget): ScrapingTarget
+configureScrapingRules(target: ScrapingTarget):: ScrapingRule[]
+scheduleScrapingSessions(targets: List<ScrapingTarget>): List<ScrapingSession>
+testScrapingRules(rule: ScrapingRule): bool
interacts
configures
NotificationController
Attributes
Operations
+ createSubscription(product: Product): Subscription
+ checkSubscriptionConditions(update: ProductUpdate): List<Notification>
+ notifySubscribers(update: ProductUpdate): void
+ sendPriceDropAlert(product: Product): void
+ sendMarketUpdateAlert(product: Product): void
+ manageSubscriptions(product: Product): List<Subscription>
+ createCustomAlert(product: Product): Subscription
Uses
UserEngagementAnalyticsEngine
Attributes
Operations
+ analyzeUserBrowsingPatterns(buyer: Buyer): UserEngagementProfile
+ calculateEngagementSentimentScore(buyer: Buyer): double
+ identifyHighValueBuyers(): List<Buyer>
+ generateDetailedEngagementReport(buyer: Buyer):UserEngagementProfile
+ predictBuyerBehavior(buyer: Buyer): UserEngagementProfile
RecommendationEngine
Attributes
Operations
+ aggregateProductReviews(product: Product): List<Review>
+ performSentimentAnalysisByCategory(product: Product):
+ identifyProductStrengthsWeaknesses(product: Product): List<Review>
+ generateReviewInsightReport(product: Product): List<Review>
+ correlateUserJourneyWithSales(product: Product): List<Review>
DealDetectionEngine
Attributes
Operations
+ calculateAverageDealDiscount():: float
+ analyzeDealClosureRate() : List<Deal>
+ identifyMostNegotiableProducts():: List<Product>
+ generateNegotiationEffectivenessReport():: List<Deal>
ReputationEngine
Attributes
Operations
+ calculateReputationFromReview(product: Product): double
+ calculateSellerReputation(seller: Seller): double
+ analyzeLongTermReputationTrends(seller: Seller)
+ generateReputationHealthReport(seller: Seller)
+ predictFutureReputationTrajectory(seller: Seller)
InventoryForecastEngine
Attributes
- inventoryHistory: List<ProductUpdate>
Operations
+ predictInventoryNeed(product: Product): List<ProductUpdate>
+ calculateStockTurnoverRate(product: Product): float
+ generateRestockRecommendations(product: Product): List<ProductUpdate>
PriceForecastEngine
Attributes
- historicalData: List<ProductUpdate>
Operations
+ forecastPrice(product: Product): List<ProductUpdate>
+ calculatePriceTrend(product: Product): List<ProductUpdate>
+ identifySeasonalPatterns(product: Product): List<ProductUpdate>
ConversationAnalyticsEngine
Attributes
Operations
+ analyzeConversationLength()
+ calculateResponseTime()
+ identifyEffectiveCommunicationPatterns()
+ generateCommunicationInsightReport()
+ detectPotentialDeals()
ScrapingEngine
Attributes
Operations
+ scrapeWebsite(targets: List<ScrapingTarget>)
+ executeScrapingRule(rule: ScrapingRule, pageContent: String)
+ manageRateLimiting(target: ScrapingTarget)
+ extractProductData(pageContent: String, rules: List<ScrapingRule>): Product
DataIntegrationController
Attributes
Operations
+ processScrapedProducts(products: List<ScrapedProduct>): String
+ matchOrCreateProduct(existingProducts:List<ScrapedProduct>): ScrapedProduct
+ updateProductCatalog(matchedProducts: List<ScrapedProduct>)
+ handleDuplicateProducts(products: List<ScrapedProduct>): List<ScrapedProduct>
+ generateProductUpdateReports(products: List<ScrapedProduct>): String
1
uses
*
manages
1
1
applies
manages
schedules
processes
Notification
Attributes
- notificationId : int
- type : String
- message : String
- timestamp : DateTime
- isRead : bool
Operations
+ markAsRead() :void
UserEngagementProfile
Attributes
- buyerId: int
- overallEngagementScore : double
- totalProductsViewed: int
- totalPurchases : int
- averageTimeSpent: float
- mostViewedCategories: List<String>
Operations
+ calculateEngagementScore():: double
+ updateProfile()
+ identifyInterestTrends():: List<string>
Buyer
Attributes
- buyerId: int
- name : String
- contactInfo: String
- viewHistory: List<Product>
- notifications: List<Notification>
- subscriptions: List<Subscription>
- browsingHistory: BrowsingSession
- reviews : List<Review>
- engagementProfile: UserEngagementProfile
Operations
+ subscribeToProduct(product: Product): Subscription
+ unsubscribeFromProduct(subscription: Subscription)
+ updatePreferences()
+ addBrowsingSession(session: BrowsingSession)
+ addReview(review: Review)
Deal
Attributes
- dealId: int
- product : Product
- buyer: Buyer
- seller : Seller
- price: double
- status: String
- created: DateTime
Operations
+ negotiate(price: float)
+ exportDealDetails()
Seller
Attributes
- sellerId : int
- name : String
- contactInfo : String
- products : List<Product>
- subscriptions: List<Subscription>
Operations
+ addProduct(product: Product)
+ updateProductListing(product: Product)
+ getAnalytics() : List<Product>
+ getReputation()
Product
Attributes
- productID: int
- name : String
- productURL: String
- price : double
- seller : Seller
- category: String
- dealRating: double
- updateHistory: List<ProductUpdate>
Operations
+ updatePrice(newPrice: double, source: string)
+ updateRating(rating: double)
+ trackUpdate(update: ProductUpdate)
+ getHistoricalPrices(startDate: string, endDate: string): List<ProductUpdate>
+ getInventoryTrends() : List<ProductUpdate>
ProductUpdate
Attributes
- updateId : int
- productID : int
- timestamp : DateTime
- updateType : String
- updatePrice : double
- updateInventory : int
- source : String
Operations
+ getHistoricalData(startDate: string, endDate: string): List<ProductUpdate>
+ generateTimeSeriesData() : List<ProductUpdate>
ConversationModel
Attributes
- conversationID: int
- buyerID: int
- sellerID: int
- messages : List<Message>
- createdAt: DateTime
- lastUpdated: DateTime
- status: String
Operations
+ addMessage()
+ updateStatus()
+ archiveConversation()
Message
Attributes
- messageID: int
- senderID : int
- senderType: int
- content : String
- timestamp: DateTime
- isRead: bool
- messageType: String
Operations
+ markAsRead()
+ validateMessage()
ScrapingRule
Attributes
- ruleId: int
- selectorPath : String
- dataType: String
- extractionMethod : String
- isRequired: bool
Operations
+ validateRule():: bool
+ testRule(testing testUrl): String
has
ScrapingTarget
Attributes
- targetId: int
- websiteUrl : string
- name: String
- category : String
- scrapingRules: List<ScrapingRule>
- isActive: bool
- lastScraped: DateTime
Operations
+ validateTarget():: bool
+ updateScrapingRules() : List<ScrapingRule>
ScrapingSession
Attributes
- sessionId: int
- startTime: DateTime
- endTime: DateTime
- products : List<ScrapedProduct>
- status: string
- totalProductsScraped: int
- successfulScrapes: int
- failedScrapes: int
Operations
+ generateSessionReport():: String
+ logSessionMetrics()
contains
1
*
ScrapedProduct
Attributes
- scrapedProductId: int
- targetId : int
- originalProductUrl: String
- name : String
- price: float
- description: String
- scrapedAt: DateTime
- rawData: string
Operations
+ compareWithExistingProduct(p: Product): String
+ normalizeProductData(): Product
Review
Attributes
- reviewId: int
- productId : int
- buyerId: int
- rating : int
- reviewDate: DateTime
- timestamp: DateTime
- sentiment : String
- isVerifiedPurchase : bool
- isHelpful: bool
- helpfulness: int
Operations
+ generateSentimentAnalysis()
+ updateHelpfulness(helpfulness: int)
BrowsingSession
Attributes
- sessionId: int
- buyerId : int
- startTime : DateTime
- endTime : DateTime
- events : List<BrowsingEvent>
- deviceType: String
- ipAddress: String
- sessionEngagementScore : double
Operations
+ calculateSessionDuration():: double
+ getEngagementMetrics() :UserEngagementProfile
BrowsingEvent
Attributes
- productID : int
- productID : int
- timestamp : DateTime
- eventType : String
- viewDuration : double
- interactionType: String
- pageUrl: String
Operations
+ trackUserInteraction(productId: int)
+ calculateInteractionScore():: double
Subscription
Attributes
- subscriptionId : int
- subscriberId : int
- subscriberType : String
- productId : int
- conditions : List<Predicates>
- createAt: DateTime
- isActive : bool
Operations
+ activate()
+ deactivate()