**public abstract class BaseEntity** {

#Long id;

#LocalDateTime createdAt;

#LocalDateTime updatedAt;

+Long getId();

+void setId(Long id);

+LocalDateTime getCreatedAt();

+void setCreatedAt(LocalDateTime createdAt);

+LocalDateTime getUpdatedAt();

+void setUpdatedAt(LocalDateTime updatedAt);

}

**public enum Role** {

CUSTOMER, // End user

STAFF, // Receptionist, service advisor

TECHNICIAN, // Mechanic performing services

ADMIN // System administrator

}

@Entity

@Table(name = "users")

**public class User extends BaseEntity** {

-String username;

-String password;

-String email;

-String phone;

-String fullName;

-String address;

-Boolean isActive;

-Role role;

+String login(String username, String password);

+void logout();

+void changePassword(String oldPassword, String newPassword);

+boolean hasRole(Role role);

}

@Entity

@Table(name = "customers")

**public class Customer extends User** {

// ============================================

// ATTRIBUTES

// ==========================================

-String customerNumber;

-String company;

-String taxCode;

-CustomerType type;

-CustomerTier tier;

-Integer loyaltyPoints;

-BigDecimal totalSpent;

-Date registrationDate;

-Date lastVisitDate;

-Boolean isVIP;

-Boolean receivePromotions;

-String preferredContactMethod;

// Relationships

-List<Vehicle> vehicles;

-List<Appointment> appointments;

-List<CustomerPackage> packages;

-List<Invoice> invoices;

-List<ChatRoom> chatRooms;

-List<ServiceFeedback> feedbacks;

-NotificationPreference notificationPreference;

// ============================================

// GETTERS & SETTERS

// ============================================

+String getCustomerNumber();

+void setCustomerNumber(String number);

+String getCompany();

+void setCompany(String company);

+String getTaxCode();

+void setTaxCode(String taxCode);

+CustomerType getType();

+void setType(CustomerType type);

+CustomerTier getTier();

+void setTier(CustomerTier tier);

+Integer getLoyaltyPoints();

+void setLoyaltyPoints(Integer points);

+BigDecimal getTotalSpent();

+void setTotalSpent(BigDecimal spent);

+Date getRegistrationDate();

+void setRegistrationDate(Date date);

+Date getLastVisitDate();

+void setLastVisitDate(Date date);

+Boolean getIsVIP();

+void setIsVIP(Boolean vip);

+Boolean getReceivePromotions();

+void setReceivePromotions(Boolean receive);

+String getPreferredContactMethod();

+void setPreferredContactMethod(String method);

+void setFeedbacks(List<ServiceFeedback> feedbacks);

+List<Vehicle> getVehicles();

+List<Appointment> getAppointments();

+List<CustomerPackage> getPackages();

+List<Invoice> getInvoices();

+List<ChatRoom> getChatRooms();

+List<ServiceFeedback> getFeedbacks();

+NotificationPreference getNotificationPreference();

+void setNotificationPreference(NotificationPreference pref);

+Double getAverageFeedbackRating();

+Integer getTotalFeedbacksSubmitted();

+List<ServiceFeedback> getPendingFeedbacks();

// ============================================

// VEHICLE MANAGEMENT

// ============================================

+Vehicle addVehicle(VehicleDTO dto);

+Vehicle updateVehicle(Long vehicleId, VehicleDTO dto);

+void removeVehicle(Long vehicleId);

+Vehicle getVehicleByVIN(String vin);

+Vehicle getVehicleByLicensePlate(String licensePlate);

+List<Vehicle> getActiveVehicles();

+List<Vehicle> getVehiclesNeedingMaintenance();

+Integer getVehicleCount();

// ============================================

// APPOINTMENT MANAGEMENT

// ============================================

+Appointment bookAppointment(AppointmentDTO dto);

+void cancelAppointment(Long appointmentId, String reason);

+void rescheduleAppointment(Long appointmentId, LocalDateTime newDate);

+List<Appointment> getUpcomingAppointments();

+List<Appointment> getPastAppointments();

+List<Appointment> getAppointmentsByStatus(AppointmentStatus status);

+List<Appointment> getAppointmentsByVehicle(Long vehicleId);

+Appointment getNextAppointment();

+boolean hasAppointmentOn(LocalDateTime datetime);

// ============================================

// SERVICE HISTORY

// ============================================

+List<ServiceRecord> getServiceHistory();

+List<ServiceRecord> getServiceHistoryByVehicle(Long vehicleId);

+List<ServiceRecord> getServiceHistoryByDateRange(Date from, Date to);

+List<ServiceRecord> getServiceHistoryByType(ServiceType serviceType);

+ServiceRecord getLastService();

+ServiceRecord getLastServiceByVehicle(Long vehicleId);

+Integer getTotalServicesCount();

+Map<ServiceType, Integer> getServiceCountByType();

// ============================================

// MAINTENANCE PACKAGE MANAGEMENT

// ============================================

+CustomerPackage subscribeToPakage(Long packageId);

+CustomerPackage renewPackage(Long customerPackageId);

+void cancelPackage(Long customerPackageId, String reason);

+List<CustomerPackage> getActivePackages();

+List<CustomerPackage> getExpiredPackages();

+List<CustomerPackage> getPackagesExpiringSoon(Integer days);

+CustomerPackage getPackageById(Long packageId);

+boolean hasActivePackage();

+Integer getTotalRemainingServices();

// ============================================

// INVOICE & PAYMENT MANAGEMENT

// ============================================

+List<Invoice> getAllInvoices();

+List<Invoice> getPaidInvoices();

+List<Invoice> getUnpaidInvoices();

+List<Invoice> getOverdueInvoices();

+Invoice getInvoiceById(Long invoiceId);

+BigDecimal getTotalUnpaidAmount();

+BigDecimal getTotalSpent();

+BigDecimal getAverageServiceCost();

+List<Payment> getPaymentHistory();

+Payment getLastPayment();

// ============================================

// LOYALTY & REWARDS

// ============================================

+void addLoyaltyPoints(Integer points);

+void deductLoyaltyPoints(Integer points);

+Integer getLoyaltyPoints();

+String redeemPoints(Integer points);

+List<String> getAvailableRewards();

+BigDecimal getDiscountFromLoyalty();

+boolean isEligibleForDiscount();

+void updateTier();

+CustomerTier calculateTier();

// ============================================

// NOTIFICATION & COMMUNICATION

// ============================================

+List<Notification> getNotifications();

+List<Notification> getUnreadNotifications();

+void markNotificationAsRead(Long notificationId);

+void markAllNotificationsAsRead();

+Integer getUnreadNotificationCount();

+void updateNotificationPreferences(NotificationPreferenceDTO dto);

+ChatRoom startChat(String initialMessage);

+List<ChatRoom> getActiveChatRooms();

// ============================================

// REMINDERS

// ============================================

+List<MaintenanceReminder> getActiveReminders();

+List<MaintenanceReminder> getOverdueReminders();

+void dismissReminder(Long reminderId);

+void snoozeReminder(Long reminderId, Integer days);

// ============================================

// REPORTS & ANALYTICS

// ============================================

+CustomerDashboardDTO getDashboard();

+Map<String, Object> getSpendingAnalytics(Integer months);

+List<ServiceRecord> getFrequentServices(Integer limit);

+Map<Vehicle, BigDecimal> getSpendingByVehicle();

+BigDecimal getYearlySpending(Integer year);

+BigDecimal getMonthlyAverageSpending();

// ============================================

// FEEDBACK & RATING

// ============================================

+ServiceFeedback submitFeedback(Long serviceRecordId, FeedbackDTO dto);

+List<ServiceFeedback> getMyFeedbacks();

+ServiceFeedback getFeedbackByService(Long serviceRecordId);

+void updateFeedback(Long feedbackId, FeedbackDTO dto);

// ============================================

// PREFERENCES

// ============================================

+void updateProfile(CustomerProfileDTO dto);

+void updateContactPreferences(ContactPreferenceDTO dto);

+void setPreferredServiceCenter(Long serviceCenterId);

+ServiceCenter getPreferredServiceCenter();

+void setPreferredTechnician(Long technicianId);

+Technician getPreferredTechnician();

// ============================================

// VALIDATION & HELPERS

// ============================================

+boolean canBookAppointment();

+boolean hasUnpaidInvoices();

+boolean needsProfileUpdate();

+String getDisplayName();

+String getFormattedCustomerNumber();

// ============================================

// INTERNAL METHODS

// ============================================

-void calculateTotalSpent();

-void updateLastVisitDate(Date date);

-void checkAndUpgradeTier();

-void sendWelcomeEmail();

-void sendBirthdayGreeting();

-boolean validateVehicleOwnership(Long vehicleId);

}

@Entity

@Table(name = "staff")

**public class Staff extends User** {

// ============================================

// ATTRIBUTES

// ============================================

-String employeeId;

-String position;

-String department;

-ServiceCenter serviceCenter;

-Date hireDate;

-BigDecimal salary;

-Boolean isActive;

// Relationships

-List<Appointment> managedAppointments;

-List<ChatRoom> chatRooms;

// ============================================

// GETTERS & SETTERS

// ============================================

+String getEmployeeId();

+void setEmployeeId(String employeeId);

+String getPosition();

+void setPosition(String position);

+String getDepartment();

+void setDepartment(String department);

+ServiceCenter getServiceCenter();

+void setServiceCenter(ServiceCenter center);

+Date getHireDate();

+void setHireDate(Date date);

+BigDecimal getSalary();

+void setSalary(BigDecimal salary);

+Boolean getIsActive();

+void setIsActive(Boolean active);

+List<Appointment> getManagedAppointments();

+List<ChatRoom> getChatRooms();

// ============================================

// BUSINESS METHODS

// ============================================

// Appointment Management

+Appointment receiveAppointment(AppointmentDTO dto);

+void confirmAppointment(Long appointmentId);

+void assignToTechnician(Long appointmentId, Long technicianId);

+void cancelAppointment(Long appointmentId, String reason);

+List<Appointment> getTodayAppointments();

+List<Appointment> getPendingAppointments();

// Service Record Management

+ServiceRecord createServiceRecord(ServiceRecordDTO dto);

+ServiceRecord updateServiceRecord(Long recordId, ServiceRecordDTO dto);

+void addPartsToService(Long recordId, List<PartDTO> parts);

// Customer Management

+Customer registerCustomer(CustomerDTO dto);

+Customer updateCustomerInfo(Long customerId, CustomerDTO dto);

+Vehicle registerVehicle(Long customerId, VehicleDTO dto);

+List<ServiceRecord> getCustomerHistory(Long customerId);

// Invoice & Payment

+Invoice generateInvoice(Long serviceRecordId);

+Invoice updateInvoice(Long invoiceId, InvoiceDTO dto);

+Payment processPayment(PaymentDTO dto);

+Payment recordCashPayment(Long invoiceId, BigDecimal amount);

// Communication

+ChatRoom createChatRoom(Long customerId);

+ChatMessage sendMessage(Long roomId, String message);

+void closeChat(Long roomId);

// Notification

+void sendNotificationToCustomer(Long customerId, String message, NotificationType type);

// Reporting (Basic)

+List<Appointment> getAppointmentsByDateRange(Date from, Date to);

+BigDecimal getDailyRevenue(Date date);

// Internal Methods

-boolean validateAppointmentTime(LocalDateTime datetime);

-void notifyTechnician(Technician tech, Appointment appointment);

-void logActivity(String action, String details);

}

@Entity

@Table(name = "technicians")

**public class Technician extends User** {

// ============================================

// ATTRIBUTES

// ============================================

-String employeeId;

-String specialization;

-Set<String> certifications;

-Integer experienceYears;

-ServiceCenter serviceCenter;

-Date hireDate;

-BigDecimal hourlyRate;

-Boolean isActive;

-Boolean isAvailableNow;

// Relationships

-List<TechnicianSchedule> schedules;

-List<ServiceRecord> assignedServices;

-List<ServiceRecord> completedServices;

-List<ServiceFeedback> receivedFeedbacks;

// ============================================

// GETTERS & SETTERS

// ============================================

+String getEmployeeId();

+void setEmployeeId(String employeeId);

+String getSpecialization();

+void setSpecialization(String specialization);

+Set<String> getCertifications();

+void setCertifications(Set<String> certs);

+void addCertification(String cert);

+void removeCertification(String cert);

+Integer getExperienceYears();

+void setExperienceYears(Integer years);

+ServiceCenter getServiceCenter();

+void setServiceCenter(ServiceCenter center);

+Date getHireDate();

+void setHireDate(Date date);

+BigDecimal getHourlyRate();

+void setHourlyRate(BigDecimal rate);

+Boolean getIsActive();

+void setIsActive(Boolean active);

+Boolean getIsAvailableNow();

+void setIsAvailableNow(Boolean available);

+List<TechnicianSchedule> getSchedules();

+List<ServiceRecord> getAssignedServices();

+List<ServiceRecord> getCompletedServices();

+List<ServiceFeedback> getReceivedFeedbacks();

+void setReceivedFeedbacks(List<ServiceFeedback> feedbacks);

// ============================================

// BUSINESS METHODS

// ============================================

// Service Execution

+void startService(Long serviceRecordId);

+void pauseService(Long serviceRecordId, String reason);

+void resumeService(Long serviceRecordId);

+void completeService(Long serviceRecordId);

+void updateServiceStatus(Long recordId, ServiceStatus status);

// Service Documentation

+void addDiagnosis(Long recordId, String diagnosis);

+void addWorkPerformed(Long recordId, String workPerformed);

+void addServiceNote(Long recordId, String note);

+void updateMileage(Long recordId, Integer mileage);

+Double getAverageRating();

+Double getAverageServiceQualityRating();

+Integer getTotalFeedbacks();

+Integer getPositiveFeedbackCount();

+Integer getNegativeFeedbackCount();

+Double getCustomerSatisfactionRate();

// Parts Management

+void requestParts(Long recordId, List<PartRequestDTO> parts);

+void addPartToService(Long recordId, Long partId, Integer quantity);

+void removePartFromService(Long recordId, Long servicePartId);

// Checklist Management

+void checkItem(Long checklistId, String notes);

+void uncheckItem(Long checklistId, String reason);

+void addChecklistItem(Long recordId, String itemName, ChecklistCategory category);

+List<Checklist> getServiceChecklist(Long recordId);

// Schedule Management

+List<ServiceRecord> getTodayTasks();

+List<ServiceRecord> getActiveTasks();

+ServiceRecord getCurrentTask();

+Integer getWorkload();

+boolean isAvailable(LocalDateTime datetime);

// Performance

+Integer getCompletedServicesCount(Date from, Date to);

+Double getAverageCompletionTime(Date from, Date to);

+BigDecimal getRevenueGenerated(Date from, Date to);

+TechnicianPerformanceDTO getPerformanceStats(Date from, Date to);

// Internal Methods

-void notifyStaffOnCompletion(ServiceRecord record);

-void deductPartsFromInventory(ServiceRecord record);

-Integer calculateLaborHours(ServiceRecord record);

-void validateCertificationForService(ServiceType serviceType);

}

@Entity

@Table(name = "admins")

**public class Admin extends User** {

// ============================================

// ATTRIBUTES

// ============================================

-String employeeId;

-String department;

-Set<String> permissions;

-Date hireDate;

-Boolean isSuperAdmin;

// ============================================

// GETTERS & SETTERS

// ============================================

+String getEmployeeId();

+void setEmployeeId(String employeeId);

+String getDepartment();

+void setDepartment(String department);

+Set<String> getPermissions();

+void setPermissions(Set<String> permissions);

+void addPermission(String permission);

+void removePermission(String permission);

+Date getHireDate();

+void setHireDate(Date date);

+Boolean getIsSuperAdmin();

+void setIsSuperAdmin(Boolean superAdmin);

// ============================================

// BUSINESS METHODS

// ============================================

// User Management

+List<User> getAllUsers(UserFilterDTO filter);

+Customer createCustomer(CustomerDTO dto);

+Staff createStaff(StaffDTO dto);

+Technician createTechnician(TechnicianDTO dto);

+User updateUser(Long userId, UserDTO dto);

+void deleteUser(Long userId);

+void activateUser(Long userId);

+void deactivateUser(Long userId);

+String resetUserPassword(Long userId);

+void changeUserRole(Long userId, Role newRole);

// Service Center Management

+List<ServiceCenter> getAllServiceCenters();

+ServiceCenter createServiceCenter(ServiceCenterDTO dto);

+ServiceCenter updateServiceCenter(Long centerId, ServiceCenterDTO dto);

+void deleteServiceCenter(Long centerId);

+void assignStaffToCenter(Long staffId, Long centerId);

+void assignTechnicianToCenter(Long techId, Long centerId);

// Inventory Management

+List<Inventory> getAllInventory(InventoryFilterDTO filter);

+Inventory createInventory(Long centerId, Long partId, InventoryDTO dto);

+void updateInventoryLevel(Long inventoryId, Integer quantity);

+void setMinStockLevel(Long inventoryId, Integer minQuantity);

+List<Inventory> getLowStockAlerts();

+void updatePartPrediction(Long inventoryId, Integer predictedDemand);

+Map<Part, Integer> predictPartsDemand(Long centerId, Integer months);

// Parts Management

+List<Part> getAllParts(PartFilterDTO filter);

+Part createPart(PartDTO dto);

+Part updatePart(Long partId, PartDTO dto);

+void deletePart(Long partId);

+void updatePartPrice(Long partId, BigDecimal newPrice);

+void activatePart(Long partId);

+void deactivatePart(Long partId);

// Service Type Management

+List<ServiceType> getAllServiceTypes();

+ServiceType createServiceType(ServiceTypeDTO dto);

+ServiceType updateServiceType(Long typeId, ServiceTypeDTO dto);

+void deleteServiceType(Long typeId);

+void updateServicePrice(Long typeId, BigDecimal newPrice);

// Maintenance Package Management

+List<MaintenancePackage> getAllPackages();

+MaintenancePackage createPackage(PackageDTO dto);

+MaintenancePackage updatePackage(Long packageId, PackageDTO dto);

+void deletePackage(Long packageId);

+List<CustomerPackage> getActiveSubscriptions();

// Staff & Technician Management

+List<Staff> getAllStaff(StaffFilterDTO filter);

+List<Technician> getAllTechnicians(TechnicianFilterDTO filter);

+void updateTechnicianCertifications(Long techId, Set<String> certifications);

+void updateTechnicianSchedule(Long techId, List<ScheduleDTO> schedules);

+void approveCertification(Long techId, String certification);

+TechnicianPerformanceDTO getTechnicianPerformance(Long techId, Date from, Date to);

// Financial Management

+List<Invoice> getAllInvoices(InvoiceFilterDTO filter);

+List<Payment> getAllPayments(PaymentFilterDTO filter);

+BigDecimal getTotalRevenue(Date from, Date to);

+BigDecimal getTotalCost(Date from, Date to);

+BigDecimal getProfit(Date from, Date to);

+Map<ServiceCenter, BigDecimal> getRevenueByCenter(Date from, Date to);

+Map<ServiceType, BigDecimal> getRevenueByServiceType(Date from, Date to);

// Reports & Analytics

+byte[] generateRevenueReport(Date from, Date to, ReportFormat format);

+byte[] generateServiceReport(Date from, Date to, ReportFormat format);

+byte[] generateTechnicianPerformanceReport(Long techId, Date from, Date to, ReportFormat format);

+Map<String, Object> getDashboardAnalytics();

+List<ServiceType> getTopServices(Integer limit);

+List<String> getCommonIssues(Integer limit);

+Map<String, Long> getServiceTrends(Integer months);

// System Configuration

+void updateSystemConfig(ConfigDTO config);

+ConfigDTO getSystemConfig();

+void updateTaxRate(BigDecimal taxRate);

+void updateWorkingHours(String workingHours);

+void configureNotifications(NotificationConfigDTO config);

// Data Management

+byte[] exportData(ExportType type, ExportFormat format);

+void importData(MultipartFile file, ImportType type);

+boolean backupDatabase();

+boolean restoreDatabase(String backupFile);

// Audit & Logging

+List<AuditLog> getAuditLogs(AuditFilterDTO filter);

+List<ActivityLog> getUserActivity(Long userId, Date from, Date to);

+void logAdminAction(String action, String details);

// Permissions

+boolean hasPermission(String permission);

-void validatePermission(String permission);

}

@Entity

@Table(name = "vehicle\_models")

**public class VehicleModel extends BaseEntity** {

-String brand;

-String modelName;

-String batteryType;

-Integer recommendedMaintenanceKm;

-Integer recommendedMaintenanceMonths;

+Date calculateNextMaintenance(Date lastMaintenanceDate);

+boolean isEligibleForService(Integer mileage, Date lastServiceDate);

}

@Entity

@Table(name = "vehicles")

**public class Vehicle extends BaseEntity** {

-String vin;

-String licensePlate;

-Integer year;

-Integer currentMileage;

-Date lastMaintenanceDate;

-Date nextMaintenanceDue;

-Customer owner;

-VehicleModel model;

+boolean needsMaintenance();

+void updateMileage(Integer newMileage);

+void calculateNextMaintenance();

}

**public enum AppointmentStatus** {

PENDING, // Waiting for confirmation

CONFIRMED, // Confirmed by staff

IN\_PROGRESS, // Service is being performed

COMPLETED, // Service completed

CANCELLED // Cancelled by customer or staff

}

**public enum ServiceStatus** {

WAITING, // Waiting to start

IN\_PROGRESS, // Currently being serviced

COMPLETED, // Service completed

ON\_HOLD, // Paused (waiting for parts, approval)

CANCELLED // Service cancelled

}

@Entity

@Table(name = "appointments")

**public class Appointment extends BaseEntity** {

-String appointmentNumber;

-Customer customer;

-Vehicle vehicle;

-ServiceCenter serviceCenter;

-ServiceType serviceType;

-LocalDateTime appointmentDate;

-AppointmentStatus status;

-Staff assignedStaff;

-String customerNotes;

-String staffNotes;

-Boolean isEmergency;

+void confirm();

+void cancel();

+void assignStaff(Staff staff);

+void reschedule(LocalDateTime newDate);

}

@Entity

@Table(name = "service\_types")

**public class ServiceType extends BaseEntity** {

-String code;

-String name;

-String description;

-Integer estimatedDuration;

-BigDecimal basePrice;

-Boolean isActive;

+BigDecimal calculatePrice(Vehicle vehicle);

+boolean isAvailable();

}

@Entity

@Table(name = "service\_records")

**public class ServiceRecord extends BaseEntity** {

-String recordNumber;

-Appointment appointment;

-Vehicle vehicle;

-Technician technician;

-ServiceType serviceType;

-LocalDateTime serviceDate;

-Integer mileageAtService;

-ServiceStatus status;

-LocalDateTime startTime;

-LocalDateTime endTime;

-String diagnosis;

-String workPerformed;

-String notes;

-Integer laborHours;

-BigDecimal laborCost;

-BigDecimal partsCost;

-BigDecimal totalCost;

-List<ServiceRecordPart> usedParts;

-List<Checklist> checklists;

-ServiceFeedback feedback;

+void start();

+void complete();

+void addPart(Long partId, Integer quantity, BigDecimal price);

+BigDecimal calculateTotalCost();

+void updateStatus(ServiceStatus newStatus);

+ServiceFeedback getFeedback();

+void setFeedback(ServiceFeedback feedback);

+boolean hasFeedback();

}

@Entity

@Table(name = "service\_record\_parts")

**public class ServiceRecordPart extends BaseEntity** {

-ServiceRecord serviceRecord;

-Part part;

-Integer quantity;

-BigDecimal unitPrice;

-BigDecimal subtotal;

+BigDecimal calculateSubtotal();

}

**public enum ChecklistCategory** {

SAFETY, // Safety checks

BATTERY, // Battery system

MOTOR, // Electric motor

BRAKES, // Brake system

ELECTRICAL, // Electrical system

CHARGING // Charging system

}

**public enum ExportFormat** {

PDF, // PDF document

EXCEL, // Excel spreadsheet (.xlsx)

CSV, // Comma-separated values

JSON // JSON format

}

**public class PartUsageHistory extends BaseEntity** {

-Part part;

-ServiceCenter serviceCenter;

-ServiceRecord serviceRecord;

@Column(nullable = false)

private Date usageDate;

@Column(nullable = false)

private Integer quantityUsed;

@Column(length = 2)

private String seasonality; // Q1, Q2, Q3, Q4

private Integer vehicleCountInMonth; // Số xe được phục vụ trong tháng đó

@Column(length = 100)

private String vehicleModelType; // Loại xe (để predict theo model)

private BigDecimal totalServiceRevenue; // Doanh thu dịch vụ tháng đó

@Column(length = 50)

private String usageType; // NORMAL\_SERVICE, EMERGENCY, RECALL, etc.

// ============================================

// GETTERS & SETTERS

// ============================================

public Part getPart()

public void setPart(Part part)

public ServiceCenter getServiceCenter()

public void setServiceCenter(ServiceCenter center)

public ServiceRecord getServiceRecord()

public void setServiceRecord(ServiceRecord record)

public Date getUsageDate()

public void setUsageDate(Date date)

public Integer getQuantityUsed()

public void setQuantityUsed(Integer quantity)

public String getSeasonality()

public void setSeasonality(String seasonality)

public Integer getVehicleCountInMonth()

public void setVehicleCountInMonth(Integer count)

public String getVehicleModelType()

public void setVehicleModelType(String modelType)

public BigDecimal getTotalServiceRevenue()

public void setTotalServiceRevenue(BigDecimal revenue)

public String getUsageType()

public void setUsageType(String type)

// ============================================

// BUSINESS METHODS

// ============================================

public String getQuarter()

public boolean isHighDemandSeason()

@Entity

@Table(name = "checklists")

**public class Checklist extends BaseEntity** {

-ServiceRecord serviceRecord;

-String itemName;

-ChecklistCategory category;

-Boolean isChecked;

-String notes;

-LocalDateTime checkedAt;

+void check();

+void uncheck();

}

@Entity

@Table(name = "parts")

**public class Part extends BaseEntity** {

-String partName;

-String partNumber;

-String category;

-BigDecimal unitPrice;

-String manufacturer;

-Boolean isActive;

-List<PartUsageHistory> usageHistory;

+boolean isInStock(Long serviceCenterId);

+Integer getStockLevel(Long serviceCenterId);

+List<PartUsageHistory> getUsageHistory()

+void setUsageHistory(List<PartUsageHistory> history)

+Integer getTotalUsageInPeriod(ServiceCenter center, Date from, Date to);

+ Double getAverageMonthlyUsage(ServiceCenter center, Integer months)

}

@Entity

@Table(name = "inventory")

**public class Inventory extends BaseEntity** {

-ServiceCenter serviceCenter;

-Part part;

-Integer quantity;

-Integer minQuantity;

-Integer predictedMonthlyDemand;

+void addStock(Integer quantity);

+void removeStock(Integer quantity);

+boolean isLowStock();

+boolean needsReorder();

+void updatePrediction(Integer demand);

}

**public enum InvoiceStatus** {

DRAFT, // Draft invoice

ISSUED, // Issued to customer

PAID, // Fully paid

PARTIALLY\_PAID, // Partially paid

OVERDUE, // Payment overdue

CANCELLED // Invoice cancelled

}

**public enum PaymentMethod** {

CASH, // Cash payment

BANK\_TRANSFER, // Bank transfer

VNPAY, // VNPay gateway

MOMO, // MoMo wallet

ZALOPAY // ZaloPay wallet

}

**public class ZaloPayGateway implements PaymentGateway** {

-String appId;

-String key1;

-String key2;

-String endpoint;

+PaymentResult processPayment(Payment payment);

+boolean verifyTransaction(String transactionId);

+RefundResult refund(Payment payment, BigDecimal amount);

+String getPaymentUrl(Payment payment);

}

**public enum PaymentStatus** {

PENDING, // Payment pending

COMPLETED, // Payment successful

FAILED, // Payment failed

REFUNDED // Payment refunded

}

@Entity

@Table(name = "invoices")

**public class Invoice extends BaseEntity** {

-String invoiceNumber;

-ServiceRecord serviceRecord;

-Customer customer;

-LocalDateTime issueDate;

-LocalDateTime dueDate;

-BigDecimal subtotal;

-BigDecimal tax;

-BigDecimal discount;

-BigDecimal totalAmount;

-InvoiceStatus status;

-List<Payment> payments;

+BigDecimal calculateTotal();

+void applyDiscount(BigDecimal amount);

+void markAsPaid();

+boolean isOverdue();

}

@Entity

@Table(name = "payments")

**public class Payment extends BaseEntity** {

-String paymentNumber;

-Invoice invoice;

-PaymentMethod method;

-BigDecimal amount;

-String transactionId;

-PaymentStatus status;

-LocalDateTime paymentDate;

+boolean process();

+boolean verify();

}

**public enum PackageStatus** {

ACTIVE, // Package is active

EXPIRED, // Package has expired

CANCELLED // Package was cancelled

}

@Entity

@Table(name = "maintenance\_packages")

**public class MaintenancePackage extends BaseEntity** {

-String code;

-String name;

-String description;

-Integer durationMonths;

-BigDecimal price;

-Integer maxServices;

-Boolean isActive;

+CustomerPackage subscribe(Customer customer);

+BigDecimal calculateDiscount();

}

@Entity

@Table(name = "customer\_packages")

**public class CustomerPackage extends BaseEntity** {

-String packageNumber;

-Customer customer;

-MaintenancePackage package;

-LocalDateTime startDate;

-LocalDateTime endDate;

-PackageStatus status;

-Integer usedServices;

+boolean isActive();

+boolean isExpired();

+void useService(ServiceRecord record);

+CustomerPackage renew();

+Integer getRemainingServices();

}

**public enum NotificationType** {

APPOINTMENT\_CONFIRMATION,

APPOINTMENT\_REMINDER,

SERVICE\_COMPLETED,

PAYMENT\_DUE,

MAINTENANCE\_DUE,

PACKAGE\_EXPIRING,

LOW\_STOCK\_ALERT

}

@Entity

@Table(name = "notifications")

**public class Notification extends BaseEntity** {

-User recipient;

-NotificationType type;

-String title;

-String message;

-Boolean isRead;

-LocalDateTime sentAt;

+void send();

+void markAsRead();

}

@Entity

@Table(name = "chat\_rooms")

**public class ChatRoom extends BaseEntity** {

-String roomId;

-Customer customer;

-Staff staff;

-Boolean isActive;

-LocalDateTime startedAt;

-LocalDateTime lastMessageAt;

-List<ChatMessage> messages;

+ChatMessage sendMessage(User sender, String content);

+void closeRoom();

}

@Entity

@Table(name = "chat\_messages")

**public class ChatMessage extends BaseEntity** {

-ChatRoom room;

-User sender;

-String message;

-LocalDateTime sentAt;

-Boolean isRead;

+void markAsRead();

}

@Entity

@Table(name = "service\_centers")

**public class ServiceCenter extends BaseEntity** {

-String name;

-String address;

-String phone;

-String workingHours;

-BigDecimal hourlyLaborRate;

-Boolean isActive;

-List<PartUsageHistory> partUsageHistory;

+boolean hasAvailableSlot(LocalDateTime datetime);

+List<Staff> getAvailableStaff(LocalDateTime datetime);

+ List<PartUsageHistory> getPartUsageHistory();

+void setPartUsageHistory(List<PartUsageHistory> history);

+Map<Part, Integer> getPartUsageSummary(Date from, Date to)

+List<Part> getTopUsedParts(Integer limit)

+Integer checkInventory(Part part);

}

**public enum Shift** {

MORNING, // 8:00-12:00

AFTERNOON, // 13:00-17:00

EVENING, // 17:00-21:00

FULL\_DAY // 8:00-17:00

}

**public enum ScheduleStatus** {

SCHEDULED, // Scheduled for work

WORKING, // Currently working

COMPLETED, // Shift completed

CANCELLED // Shift cancelled

}

@Entity

@Table(name = "technician\_schedules")

**public class TechnicianSchedule extends BaseEntity** {

-Staff technician;

-Date workDate;

-Shift shift;

-ScheduleStatus status;

-LocalDateTime startTime;

-LocalDateTime endTime;

+boolean isAvailable();

+void markAsWorking();

+void markAsCompleted();

}

**public interface PaymentGateway** {

+PaymentResult processPayment(Payment payment);

+boolean verifyTransaction(String transactionId);

+RefundResult refund(Payment payment, BigDecimal amount);

+String getPaymentUrl(Payment payment);

}

@Service

**public class VNPayGateway implements PaymentGateway** {

-String merchantId;

-String secretKey;

-String apiUrl;

+PaymentResult processPayment(Payment payment);

+boolean verifyTransaction(String transactionId);

+RefundResult refund(Payment payment, BigDecimal amount);

+String getPaymentUrl(Payment payment);

}

@Service

**public class MoMoGateway implements PaymentGateway** {

-String partnerCode;

-String accessKey;

-String secretKey;

+PaymentResult processPayment(Payment payment);

+boolean verifyTransaction(String transactionId);

+RefundResult refund(Payment payment, BigDecimal amount);

+String getPaymentUrl(Payment payment);

}

**public interface NotificationChannel** {

+boolean send(Notification notification);

+boolean supports(NotificationType type);

}

@Service

**public class EmailNotificationChannel implements NotificationChannel** {

-String smtpHost;

-Integer smtpPort;

-String username;

+boolean send(Notification notification);

+boolean supports(NotificationType type);

}

@Service

**public class SMSNotificationChannel implements NotificationChannel** {

-String apiKey;

-String apiUrl;

+boolean send(Notification notification);

+boolean supports(NotificationType type);

}

**public interface PriceCalculator** {

+BigDecimal calculatePrice(Vehicle vehicle, ServiceType service);

+BigDecimal applyDiscount(BigDecimal price, Customer customer);

}

@Service

@Primary

**public class StandardPriceCalculator implements PriceCalculator** {

+BigDecimal calculatePrice(Vehicle vehicle, ServiceType service);

+BigDecimal applyDiscount(BigDecimal price, Customer customer);

}

@Service

**public class LoyaltyPriceCalculator implements PriceCalculator** {

-Integer pointsThreshold;

-BigDecimal discountPercent;

+BigDecimal calculatePrice(Vehicle vehicle, ServiceType service);

+BigDecimal applyDiscount(BigDecimal price, Customer customer);

}

**public interface PartsDemandPredictor** {

+Integer predictDemand(ServiceCenter center, Part part, Integer months);

+Double getConfidenceScore();

}

@Service

@Primary

**public class ManualPredictor implements PartsDemandPredictor** {

+Integer predictDemand(ServiceCenter center, Part part, Integer months);

+Double getConfidenceScore();

+void setManualPrediction(Inventory inventory, Integer demand);

}

@Service

**public class AIPartsDemandPredictor implements PartsDemandPredictor** {

- usageHistoryRepository: PartUsageHistoryRepository |

| - serviceRecordRepository: ServiceRecordRepository |

| - inventoryRepository: InventoryRepository |

| - modelPath: String |

| - confidenceThreshold: Double |

| - lastConfidenceScore: Double |

|---------------------------+-------------------|

| + predictDemand(center: ServiceCenter, part: Part, months: Integer): Integer |

| + getConfidenceScore(): Double |

| + predictAllParts(center: ServiceCenter, months: Integer): Map<Part, Integer> |

| + trainModel(trainingData: List<PartUsageHistory>): void |

| + getPartsNeedingReorder(center: ServiceCenter, months: Integer): List<Inventory> |

| - extractFeatures(history: List<PartUsageHistory>, center: ServiceCenter, part: Part, months: Integer): Map<String, Double> |

| - calculateTrend(history: List<PartUsageHistory>): double |

| - calculateSeasonalityFactor(history: List<PartUsageHistory>, currentQuarter: String): double |

| - calculateVehicleCorrelation(history: List<PartUsageHistory>): double |

| - calculateRecentSpike(history: List<PartUsageHistory>): double |

| - getCategoryFactor(category: String): double |

| - calculateRecencyScore(history: List<PartUsageHistory>): double |

| - getCurrentQuarter(): String |

| - runPredictionModel(features: Map<String, Double>): Integer |

| - calculateConfidence(history: List<PartUsageHistory>, features: Map<String, Double>): Double |

| - calculateVariance(history: List<PartUsageHistory>): double |

| + getModelPath(): String |

| + setModelPath(modelPath: String): void |

| + getConfidenceThreshold(): Double |

| + setConfidenceThreshold(confidenceThreshold: Double): void }

**public interface ReportGenerator** {

+byte[] generate(ReportParams params);

+ReportFormat getFormat();

}

**public enum ReportFormat** {

PDF, // PDF document

EXCEL, // Excel spreadsheet (.xlsx)

CSV // Comma-separated values

}

@Service

**public class PDFReportGenerator implements ReportGenerator** {

+byte[] generate(ReportParams params);

+ReportFormat getFormat();

}

**public class PaymentResult** {

-boolean success;

-String transactionId;

-String message;

-String gatewayResponse;

+boolean isSuccess();

+String getTransactionId();

+String getMessage();

}

**public class RefundResult** {

-boolean success;

-String refundId;

-BigDecimal refundedAmount;

-String message;

+boolean isSuccess();

+String getRefundId();

}

**public class ReportParams** {

-ReportType type;

-LocalDate startDate;

-LocalDate endDate;

-Long serviceCenterId;

-Map<String, Object> additionalParams;

+ReportType getType();

+LocalDate getStartDate();

}

@Entity

@Table(name = "maintenance\_reminders")

**public class MaintenanceReminder extends BaseEntity** {

// ============================================

// ATTRIBUTES

// ============================================

-Customer customer;

-Vehicle vehicle;

-LocalDateTime reminderDate;

-String message;

-Boolean isSent;

-Boolean isRead;

-Boolean isOverdue;

// Loại nhắc nhở: theo thời gian hoặc theo số km

-ReminderType type;

-Integer dueMileage; // nếu là nhắc theo km

-LocalDateTime dueDate; // nếu là nhắc theo thời gian

// ============================================

// GETTERS & SETTERS

// ============================================

+Customer getCustomer();

+void setCustomer(Customer customer);

+Vehicle getVehicle();

+void setVehicle(Vehicle vehicle);

+LocalDateTime getReminderDate();

+void setReminderDate(LocalDateTime date);

+String getMessage();

+void setMessage(String message);

+Boolean getIsSent();

+void setIsSent(Boolean sent);

+Boolean getIsRead();

+void setIsRead(Boolean read);

+Boolean getIsOverdue();

+void setIsOverdue(Boolean overdue);

+ReminderType getType();

+void setType(ReminderType type);

+Integer getDueMileage();

+void setDueMileage(Integer mileage);

+LocalDateTime getDueDate();

+void setDueDate(LocalDateTime dueDate);

// ============================================

// BUSINESS LOGIC

// ============================================

+boolean isDue();

+void markAsSent();

+void markAsRead();

+boolean isMileageBased();

+boolean isTimeBased();

}

**public enum ReminderType** {

MILEAGE\_BASED, // Nhắc theo km

TIME\_BASED // Nhắc theo ngày/tháng

}

@Entity

@Table(name = "notification\_preferences")

**public class NotificationPreference extends BaseEntity** {

-Customer customer;

-Boolean receiveEmail;

-Boolean receiveSMS;

-Boolean receiveAppPush;

-Boolean maintenanceReminders;

-Boolean paymentReminders;

-Boolean promotions;

-String preferredChannel; // "EMAIL", "SMS", "PUSH"

+Customer getCustomer();

+void setCustomer(Customer customer);

+Boolean getReceiveEmail();

+void setReceiveEmail(Boolean receiveEmail);

+Boolean getReceiveSMS();

+void setReceiveSMS(Boolean receiveSMS);

+Boolean getReceiveAppPush();

+void setReceiveAppPush(Boolean receiveAppPush);

+Boolean getMaintenanceReminders();

+void setMaintenanceReminders(Boolean maintenanceReminders);

+Boolean getPaymentReminders();

+void setPaymentReminders(Boolean paymentReminders);

+Boolean getPromotions();

+void setPromotions(Boolean promotions);

+String getPreferredChannel();

+void setPreferredChannel(String preferredChannel);

+boolean allows(NotificationType type);

}

@Entity

@Table(name = "system\_config")

**public class SystemConfig extends BaseEntity** {

-BigDecimal taxRate;

-String workingHours;

-String currency;

-String companyName;

-String companyAddress;

-String supportEmail;

-String supportPhone;

-Integer reminderDaysBeforeDue; // số ngày trước hạn gửi nhắc bảo dưỡng

-Boolean autoGenerateReports;

+BigDecimal getTaxRate();

+void setTaxRate(BigDecimal taxRate);

+String getWorkingHours();

+void setWorkingHours(String workingHours);

+String getCurrency();

+void setCurrency(String currency);

+String getCompanyName();

+void setCompanyName(String companyName);

+String getCompanyAddress();

+void setCompanyAddress(String companyAddress);

+String getSupportEmail();

+void setSupportEmail(String email);

+String getSupportPhone();

+void setSupportPhone(String phone);

+Integer getReminderDaysBeforeDue();

+void setReminderDaysBeforeDue(Integer days);

+Boolean getAutoGenerateReports();

+void setAutoGenerateReports(Boolean autoGenerateReports);

// BUSINESS LOGIC

+void updateFromDTO(ConfigDTO dto);

+void validateConfig();

}

**public class AIPartsDemandPredictor implements PartsDemandPredictor** {

- usageHistoryRepository: PartUsageHistoryRepository

- serviceRecordRepository: ServiceRecordRepository

- modelPath: String

- confidenceThreshold: Double

- lastConfidenceScore: Double

+-----------------------------------------------------------+

+ predictDemand(center: ServiceCenter, part: Part, months: Integer): Integer

+ getConfidenceScore(): Double

+ predictAllParts(center: ServiceCenter, months: Integer) : Map<Part, Integer>

- extractFeatures(history: List<PartUsageHistory>,center: ServiceCenter, part: Part, months: Integer): Map<String, Double>

- calculateTrend(history: List<PartUsageHistory>): double

- calculateSeasonalityFactor(history: List<PartUsageHistory>, |quarter: String): double

- calculateVehicleCorrelation(history: List<PartUsageHistory>): double

- calculateRecentSpike(history: List<PartUsageHistory>): double

- getCategoryFactor(category: String): double

- runPredictionModel(features: Map<String, Double>): Integer

- calculateConfidence(history: List<PartUsageHistory>, features: Map<String, Double>): Double

- calculateVariance(history: List<PartUsageHistory>): double

- getCurrentQuarter(): String

@Entity

@Table(name = "audit\_logs")

**public class AuditLog extends BaseEntity** {

-User actor;

-String action;

-String entityName;

-Long entityId;

-String details;

-LocalDateTime actionTime;

-String ipAddress;

+User getActor();

+void setActor(User actor);

+String getAction();

+void setAction(String action);

+String getEntityName();

+void setEntityName(String entityName);

+Long getEntityId();

+void setEntityId(Long entityId);

+String getDetails();

+void setDetails(String details);

+LocalDateTime getActionTime();

+void setActionTime(LocalDateTime actionTime);

+String getIpAddress();

+void setIpAddress(String ipAddress);

+String formatLogEntry();

}

@Entity

@Table(name = "activity\_logs")

**public class ActivityLog extends BaseEntity** {

-User user;

-String activityType; **// LOGIN, LOGOUT, BOOK\_APPOINTMENT, MAKE\_PAYMENT, etc.**

-String description;

-LocalDateTime activityTime;

-String ipAddress;

+User getUser();

+void setUser(User user);

+String getActivityType();

+void setActivityType(String type);

+String getDescription();

+void setDescription(String desc);

+LocalDateTime getActivityTime();

+void setActivityTime(LocalDateTime time);

+String getIpAddress();

+void setIpAddress(String ip);

+String summarize();

}

**public enum CustomerType** {

INDIVIDUAL, // Khách hàng cá nhân

BUSINESS // Khách hàng doanh nghiệp (có công ty, mã số thuế)

}

***public enum CustomerTier*** *{*

*BRONZE, // Mới đăng ký hoặc chi tiêu dưới 5 triệu*

*SILVER, // Chi tiêu 5-15 triệu*

*GOLD, // Chi tiêu 15-30 triệu*

*PLATINUM, // Chi tiêu trên 30 triệu*

*VIP // Khách hàng đặc biệt*

*}*

**public enum ReportType** {

REVENUE, // Báo cáo doanh thu

SERVICE, // Báo cáo dịch vụ

TECHNICIAN\_PERFORMANCE, // Hiệu suất kỹ thuật viên

INVENTORY, // Báo cáo tồn kho

CUSTOMER\_ANALYTICS, // Phân tích khách hàng

PARTS\_USAGE // Báo cáo sử dụng phụ tùng

}

**public enum ExportType** {

CUSTOMERS,

VEHICLES,

APPOINTMENTS,

SERVICE\_RECORDS,

INVOICES,

INVENTORY,

ALL\_DATA

}

**public enum ImportType** {

CUSTOMERS,

VEHICLES,

PARTS,

INVENTORY,

SERVICE\_TYPES

}

**public class ServiceFeedback extends BaseEntity** {

// ============================================

// ATTRIBUTES

// ============================================

- serviceRecord : ServiceRecord

- customer : Customer

- technician : Technician

- rating : Integer

- serviceQualityRating : Integer

- technicianRating : Integer

- facilitiesRating : Integer

- priceRating : Integer

- comment : String

- strengths : String

- improvements : String

- submittedAt : LocalDateTime

- isPublished : Boolean

- isVerified : Boolean

*// ============================================*

*// METHOD*

*// ============================================*

+ getServiceRecord() : ServiceRecord

+ setServiceRecord(serviceRecord : ServiceRecord) : void

+ getCustomer() : Customer

+ setCustomer(customer : Customer) : void

+ getTechnician() : Technician

+ setTechnician(technician : Technician) : void

+ getRating() : Integer

+ setRating(rating : Integer) : void

+ getServiceQualityRating() : Integer

+ setServiceQualityRating(rating : Integer) : void

+ getTechnicianRating() : Integer

+ setTechnicianRating(rating : Integer) : void

+ getFacilitiesRating() : Integer

+ setFacilitiesRating(rating : Integer) : void

+ getPriceRating() : Integer

+ setPriceRating(rating : Integer) : void

+ getComment() : String

+ setComment(comment : String) : void

+ getStrengths() : String

+ setStrengths(strengths : String) : void

+ getImprovements() : String

+ setImprovements(improvements : String) : void

+ getSubmittedAt() : LocalDateTime

+ setSubmittedAt(submittedAt : LocalDateTime) : void

+ getIsPublished() : Boolean

+ setIsPublished(isPublished : Boolean) : void

+ getIsVerified() : Boolean

+ setIsVerified(isVerified : Boolean) : void

+ getAverageRating() : Double

+ isPositive() : boolean

+ isNegative() : boolean

+ publish() : void

+ unpublish() : void

+ verify() : void

**public class CustomerDashboardDTO** {

- customerName : String

- customerNumber : String

- tier : CustomerTier

- loyaltyPoints : Integer

- totalVehicles : Integer

- vehiclesNeedingMaintenance : Integer

- upcomingAppointments : Integer

- nextAppointment : Appointment

- totalSpent : BigDecimal

- unpaidAmount : BigDecimal

- totalInvoices : Integer

- activePackages : Integer

- remainingServices : Integer

- activeReminders : Integer

- urgentReminders : List<MaintenanceReminder>

- recentServices : List<ServiceRecord>

- unreadNotifications : Integer

- pendingFeedbacks : Integer

+ CustomerDashboardDTO()

+ getCustomerName() : String

+ setCustomerName(name : String) : void

+ getCustomerNumber() : String

+ setCustomerNumber(number : String) : void

+ getTier() : CustomerTier

+ setTier(tier : CustomerTier) : void

+ getLoyaltyPoints() : Integer

+ setLoyaltyPoints(points : Integer) : void

+ getTotalVehicles() : Integer

+ setTotalVehicles(total : Integer) : void

+ getVehiclesNeedingMaintenance() : Integer

+ setVehiclesNeedingMaintenance(count : Integer) : void

+ getUpcomingAppointments() : Integer

+ setUpcomingAppointments(count : Integer) : void

+ getNextAppointment() : Appointment

+ setNextAppointment(appointment : Appointment) : void

+ getTotalSpent() : BigDecimal

+ setTotalSpent(spent : BigDecimal) : void

+ getUnpaidAmount() : BigDecimal

+ setUnpaidAmount(amount : BigDecimal) : void

+ getTotalInvoices() : Integer

+ setTotalInvoices(total : Integer) : void

+ getActivePackages() : Integer

+ setActivePackages(active : Integer) : void

+ getRemainingServices() : Integer

+ setRemainingServices(remaining : Integer) : void

+ getActiveReminders() : Integer

+ setActiveReminders(active : Integer) : void

+ getUrgentReminders() : List<MaintenanceReminder>

+ setUrgentReminders(reminders : List<MaintenanceReminder>) : void

+ getRecentServices() : List<ServiceRecord>

+ setRecentServices(services : List<ServiceRecord>) : void

+ getUnreadNotifications() : Integer

+ setUnreadNotifications(count : Integer) : void

+ getPendingFeedbacks() : Integer

+ setPendingFeedbacks(count : Integer) : void

**public class FeedbackDTO** {

- serviceRecordId : Long

- rating : Integer

- serviceQualityRating : Integer

- technicianRating : Integer

- facilitiesRating : Integer

- priceRating : Integer

- comment : String

- strengths : String

- improvements : String

+ FeedbackDTO()

+ getServiceRecordId() : Long

+ setServiceRecordId(id : Long) : void

+ getRating() : Integer

+ setRating(rating : Integer) : void

+ getServiceQualityRating() : Integer

+ setServiceQualityRating(rating : Integer) : void

+ getTechnicianRating() : Integer

+ setTechnicianRating(rating : Integer) : void

+ getFacilitiesRating() : Integer

+ setFacilitiesRating(rating : Integer) : void

+ getPriceRating() : Integer

+ setPriceRating(rating : Integer) : void

+ getComment() : String

+ setComment(comment : String) : void

+ getStrengths() : String

+ setStrengths(strengths : String) : void

+ getImprovements() : String

+ setImprovements(improvements : String) : void

**public class ServiceTypeDTO** {

- id : Long

- code : String

- name : String

- description : String

- estimatedDuration : Integer

- basePrice : BigDecimal

- isActive : Boolean

- category : String

+ ServiceTypeDTO()

+ getId() : Long

+ setId(id : Long) : void

+ getCode() : String

+ setCode(code : String) : void

+ getName() : String

+ setName(name : String) : void

+ getDescription() : String

+ setDescription(description : String) : void

+ getEstimatedDuration() : Integer

+ setEstimatedDuration(duration : Integer) : void

+ getBasePrice() : BigDecimal

+ setBasePrice(price : BigDecimal) : void

+ getIsActive() : Boolean

+ setIsActive(active : Boolean) : void

+ getCategory() : String

+ setCategory(category : String) : void

**public class TechnicianPerformanceDTO** {

- technicianId : Long

- technicianName : String

- employeeId : String

- totalServicesCompleted : Integer

- servicesInProgress : Integer

- totalWorkDays : Integer

- averageCompletionTime : Double

- totalWorkHours : Double

- efficiency : Double

- revenueGenerated : BigDecimal

- averageServiceValue : BigDecimal

- averageRating : Double

- totalFeedbacks : Integer

- positiveFeedbacks : Integer

- negativeFeedbacks : Integer

- certifications : Set<String>

- servicesByType : Map<String, Integer>

- performanceLevel : String

- rankInCenter : Integer

+ TechnicianPerformanceDTO()

+ getTechnicianId() : Long

+ setTechnicianId(id : Long) : void

+ getTechnicianName() : String

+ setTechnicianName(name : String) : void

+ getEmployeeId() : String

+ setEmployeeId(id : String) : void

+ getTotalServicesCompleted() : Integer

+ setTotalServicesCompleted(total : Integer) : void

+ getServicesInProgress() : Integer

+ setServicesInProgress(inProgress : Integer) : void

+ getTotalWorkDays() : Integer

+ setTotalWorkDays(days : Integer) : void

+ getAverageCompletionTime() : Double

+ setAverageCompletionTime(time : Double) : void

+ getTotalWorkHours() : Double

+ setTotalWorkHours(hours : Double) : void

+ getEfficiency() : Double

+ setEfficiency(efficiency : Double) : void

+ getRevenueGenerated() : BigDecimal

+ setRevenueGenerated(revenue : BigDecimal) : void

+ getAverageServiceValue() : BigDecimal

+ setAverageServiceValue(value : BigDecimal) : void

+ getAverageRating() : Double

+ setAverageRating(rating : Double) : void

+ getTotalFeedbacks() : Integer

+ setTotalFeedbacks(total : Integer) : void

+ getPositiveFeedbacks() : Integer

+ setPositiveFeedbacks(count : Integer) : void

+ getNegativeFeedbacks() : Integer

+ setNegativeFeedbacks(count : Integer) : void

+ getCertifications() : Set<String>

+ setCertifications(certifications : Set<String>) : void

+ getServicesByType() : Map<String, Integer>

+ setServicesByType(services : Map<String, Integer>) : void

+ getPerformanceLevel() : String

+ setPerformanceLevel(level : String) : void

+ getRankInCenter() : Integer

+ setRankInCenter(rank : Integer) : void

+ isTopPerformer() : boolean

+ getCustomerSatisfactionRate() : Double

**public class ConfigDTO {**

- taxRate : BigDecimal

- workingHours : String

- currency : String

- timezone : String

- companyName : String

- companyAddress : String

- companyPhone : String

- companyEmail : String

- companyWebsite : String

- companyLogo : String

- smtpHost : String

- smtpPort : Integer

- smtpUsername : String

- smtpPassword : String

- smtpUseTLS : Boolean

- smsProvider : String

- smsApiKey : String

- smsApiUrl : String

- reminderDaysBeforeDue : Integer

- reminderHoursBeforeAppointment : Integer

- autoSendReminders : Boolean

- enableOnlinePayment : Boolean

- enabledPaymentMethods : Set<String>

- pointsPerVND : Integer

- tierDiscounts : Map<CustomerTier, BigDecimal>

- enableAutoReorder : Boolean

- lowStockThreshold : Integer

- enableAIPrediction : Boolean

- autoGenerateReports : Boolean

- reportSchedule : String

- reportRecipients : List<String>

+ ConfigDTO()

+ getTaxRate() : BigDecimal

+ setTaxRate(rate : BigDecimal) : void

+ getWorkingHours() : String

+ setWorkingHours(hours : String) : void

+ getCurrency() : String

+ setCurrency(currency : String) : void

+ getTimezone() : String

+ setTimezone(zone : String) : void

+ getCompanyName() : String

+ setCompanyName(name : String) : void

+ getCompanyAddress() : String

+ setCompanyAddress(address : String) : void

+ getCompanyPhone() : String

+ setCompanyPhone(phone : String) : void

+ getCompanyEmail() : String

+ setCompanyEmail(email : String) : void

+ getCompanyWebsite() : String

+ setCompanyWebsite(website : String) : void

+ getCompanyLogo() : String

+ setCompanyLogo(logo : String) : void

+ getSmtpHost() : String

+ setSmtpHost(host : String) : void

+ getSmtpPort() : Integer

+ setSmtpPort(port : Integer) : void

+ getSmtpUsername() : String

+ setSmtpUsername(username : String) : void

+ getSmtpPassword() : String

+ setSmtpPassword(password : String) : void

+ getSmtpUseTLS() : Boolean

+ setSmtpUseTLS(useTLS : Boolean) : void

+ getSmsProvider() : String

+ setSmsProvider(provider : String) : void

+ getSmsApiKey() : String

+ setSmsApiKey(apiKey : String) : void

+ getSmsApiUrl() : String

+ setSmsApiUrl(url : String) : void

+ getReminderDaysBeforeDue() : Integer

+ setReminderDaysBeforeDue(days : Integer) : void

+ getReminderHoursBeforeAppointment() : Integer

+ setReminderHoursBeforeAppointment(hours : Integer) : void

+ getAutoSendReminders() : Boolean

+ setAutoSendReminders(auto : Boolean) : void

+ getEnableOnlinePayment() : Boolean

+ setEnableOnlinePayment(enable : Boolean) : void

+ getEnabledPaymentMethods() : Set<String>

+ setEnabledPaymentMethods(methods : Set<String>) : void

+ getPointsPerVND() : Integer

+ setPointsPerVND(points : Integer) : void

+ getTierDiscounts() : Map<CustomerTier, BigDecimal>

+ setTierDiscounts(discounts : Map<CustomerTier, BigDecimal>) : void

+ getEnableAutoReorder() : Boolean

+ setEnableAutoReorder(enable : Boolean) : void

+ getLowStockThreshold() : Integer

+ setLowStockThreshold(threshold : Integer) : void

+ getEnableAIPrediction() : Boolean

+ setEnableAIPrediction(enable : Boolean) : void

+ getAutoGenerateReports() : Boolean

+ setAutoGenerateReports(auto : Boolean) : void

+ getReportSchedule() : String

+ setReportSchedule(schedule : String) : void

+ getReportRecipients() : List<String>

+ setReportRecipients(recipients : List<String>) : void

**public class VehicleDTO** {

- id: Long

- vin: String

- licensePlate: String

- modelId: Long

- year: Integer

- currentMileage: Integer

- lastMaintenanceDate: Date

- color: String

- purchaseDate: Date

+ VehicleDTO()

+ getVin(): String

+ setVin(vin: String): void

+ getLicensePlate(): String

+ setLicensePlate(plate: String): void

**public class AppointmentDTO** {

- id: Long

- customerId: Long

- vehicleId: Long

- serviceCenterId: Long

- serviceTypeId: Long

- appointmentDate: LocalDateTime

- customerNotes: String

- isEmergency: Boolean

+ AppointmentDTO()

+ getVehicleId(): Long

+ setVehicleId(id: Long): void

+ getAppointmentDate(): LocalDateTime

+ setAppointmentDate(date: LocalDateTime): void

// ... other getters/setters ...

**public class ServiceRecordDTO** {

- id: Long

- appointmentId: Long

- vehicleId: Long

- technicianId: Long

- serviceTypeId: Long

- mileageAtService: Integer

- diagnosis: String

- workPerformed: String

- notes: String

+------------------------------------------------+

+ ServiceRecordDTO()

+ getId(): Long

+ setId(id: Long): void

+ getAppointmentId(): Long

+ setAppointmentId(id: Long): void

+ getVehicleId(): Long

+ setVehicleId(id: Long): void

+ getTechnicianId(): Long

+ setTechnicianId(id: Long): void

+ getServiceTypeId(): Long

+ setServiceTypeId(id: Long): void

+ getMileageAtService(): Integer

+ setMileageAtService(mileage: Integer): void

+ getDiagnosis(): String

+ setDiagnosis(diagnosis: String): void

+ getWorkPerformed(): String

+ setWorkPerformed(work: String): void

+ getNotes(): String

+ setNotes(notes: String): void

**public class PartDTO** {

- partId: Long

- quantity: Integer

- unitPrice: BigDecimal

- notes: String

+------------------------------------------+

+ PartDTO()

+ getPartId(): Long

+ setPartId(id: Long): void

+ getQuantity(): Integer

+ setQuantity(qty: Integer): void

+ getUnitPrice(): BigDecimal

+ setUnitPrice(price: BigDecimal): void

+ getNotes(): String

+ setNotes(notes: String): void

**public class CustomerProfileDTO** {

- fullName: String

- email: String

- phone: String

- address: String

- company: String

- taxCode: String

- receivePromotions: Boolean

+---------------------------------------------------+

+ CustomerProfileDTO()

+ getFullName(): String

+ setFullName(name: String): void

+ getEmail(): String

+ setEmail(email: String): void

+ getPhone(): String

+ setPhone(phone: String): void

+ getAddress(): String

+ setAddress(address: String): void

+ getCompany(): String

+ setCompany(company: String): void

+ getTaxCode(): String

+ setTaxCode(code: String): void

+ getReceivePromotions(): Boolean

+ setReceivePromotions(flag: Boolean): void

**public class CustomerDTO {**

- username: String

- password: String

- email: String

- phone: String

- fullName: String

- address: String

- company: String

- taxCode: String

- type: CustomerType

- receivePromotions: Boolean

+---------------------------------------------------+

+ CustomerDTO()

+ getUsername(): String

+ setUsername(username: String): void

+ getPassword(): String

+ setPassword(password: String): void

+ getEmail(): String

+ setEmail(email: String): void

+ getPhone(): String

+ setPhone(phone: String): void

+ getFullName(): String

+ setFullName(fullName: String): void

+ getAddress(): String

+ setAddress(address: String): void

+ getCompany(): String

+ setCompany(company: String): void

+ getTaxCode(): String

+ setTaxCode(taxCode: String): void

+ getType(): CustomerType

+ setType(type: CustomerType): void

+ getReceivePromotions(): Boolean

+ setReceivePromotions(receive: Boolean): void

**public class StaffDTO** {

- username: String

- password: String

- email: String

- phone: String

- fullName: String

- employeeId: String

- position: String

- department: String

- serviceCenterId: Long

- hireDate: Date

- salary: BigDecimal

+---------------------------------------------------+

| + StaffDTO()

+ getUsername(): String

+ setUsername(username: String): void

+ getPassword(): String

+ setPassword(password: String): void

+ getEmail(): String

+ setEmail(email: String): void

+ getPhone(): String

+ setPhone(phone: String): void

+ getFullName(): String

+ setFullName(fullName: String): void

+ getEmployeeId(): String

+ setEmployeeId(employeeId: String): void

+ getPosition(): String

+ setPosition(position: String): void

+ getDepartment(): String

+ setDepartment(department: String): void

+ getServiceCenterId(): Long

+ setServiceCenterId(serviceCenterId: Long): void

+ getHireDate(): Date

+ setHireDate(hireDate: Date): void

+ getSalary(): BigDecimal

+ setSalary(salary: BigDecimal): void

**public class TechnicianDTO** {

- username: String |

| - password: String |

| - email: String |

| - phone: String |

| - fullName: String |

| - employeeId: String |

| - specialization: String |

| - certifications: Set<String> |

| - experienceYears: Integer |

| - serviceCenterId: Long |

| - hireDate: Date |

| - hourlyRate: BigDecimal |

+-----------------------------------------------------------+

| + TechnicianDTO() |

| + getUsername(): String |

| + setUsername(username: String): void |

| + getPassword(): String |

| + setPassword(password: String): void |

| + getEmail(): String |

| + setEmail(email: String): void |

| + getPhone(): String |

| + setPhone(phone: String): void |

| + getFullName(): String |

| + setFullName(fullName: String): void |

| + getEmployeeId(): String |

| + setEmployeeId(employeeId: String): void |

| + getSpecialization(): String |

| + setSpecialization(specialization: String): void |

| + getCertifications(): Set<String> |

| + setCertifications(certifications: Set<String>): void |

| + getExperienceYears(): Integer |

| + setExperienceYears(experienceYears: Integer): void |

| + getServiceCenterId(): Long |

| + setServiceCenterId(serviceCenterId: Long): void |

| + getHireDate(): Date |

| + setHireDate(hireDate: Date): void |

| + getHourlyRate(): BigDecimal |

| + setHourlyRate(hourlyRate: BigDecimal): void

**public class UserDTO** {

- id: Long |

| - username: String |

| - email: String |

| - phone: String |

| - fullName: String |

| - address: String |

| - isActive: Boolean |

| - role: Role |

+-----------------------------------------------------+

| + UserDTO() |

| + getId(): Long |

| + setId(id: Long): void |

| + getUsername(): String |

| + setUsername(username: String): void |

| + getEmail(): String |

| + setEmail(email: String): void |

| + getPhone(): String |

| + setPhone(phone: String): void |

| + getFullName(): String |

| + setFullName(fullName: String): void |

| + getAddress(): String |

| + setAddress(address: String): void |

| + getIsActive(): Boolean |

| + setIsActive(isActive: Boolean): void |

| + getRole(): Role |

| + setRole(role: Role): void

**public class ServiceCenterDTO** {

- id: Long |

| - name: String |

| - address: String |

| - phone: String |

| - workingHours: String |

| - hourlyLaborRate: BigDecimal |

| - isActive: Boolean |

+------------------------------------------------------+

| + ServiceCenterDTO() |

| + getId(): Long |

| + setId(id: Long): void |

| + getName(): String |

| + setName(name: String): void |

| + getAddress(): String |

| + setAddress(address: String): void |

| + getPhone(): String |

| + setPhone(phone: String): void |

| + getWorkingHours(): String |

| + setWorkingHours(workingHours: String): void |

| + getHourlyLaborRate(): BigDecimal |

| + setHourlyLaborRate(rate: BigDecimal): void |

| + getIsActive(): Boolean |

| + setIsActive(isActive: Boolean): void

**public class InventoryDTO** {

- serviceCenterId: Long |

| - partId: Long |

| - quantity: Integer |

| - minQuantity: Integer |

| - predictedMonthlyDemand: Integer |

+------------------------------------------------------+

| + InventoryDTO() |

| + getServiceCenterId(): Long |

| + setServiceCenterId(serviceCenterId: Long): void |

| + getPartId(): Long |

| + setPartId(partId: Long): void |

| + getQuantity(): Integer |

| + setQuantity(quantity: Integer): void |

| + getMinQuantity(): Integer |

| + setMinQuantity(minQuantity: Integer): void |

| + getPredictedMonthlyDemand(): Integer |

| + setPredictedMonthlyDemand(demand: Integer): void

**public class PackageDTO** {

- id: Long |

| - code: String |

| - name: String |

| - description: String |

| - durationMonths: Integer |

| - price: BigDecimal |

| - maxServices: Integer |

| - isActive: Boolean |

+------------------------------------------------------+

| + PackageDTO() |

| + getId(): Long |

| + setId(id: Long): void |

| + getCode(): String |

| + setCode(code: String): void |

| + getName(): String |

| + setName(name: String): void |

| + getDescription(): String |

| + setDescription(description: String): void |

| + getDurationMonths(): Integer |

| + setDurationMonths(durationMonths: Integer): void |

| + getPrice(): BigDecimal |

| + setPrice(price: BigDecimal): void |

| + getMaxServices(): Integer |

| + setMaxServices(maxServices: Integer): void |

| + getIsActive(): Boolean |

| + setIsActive(isActive: Boolean): void

**public class InvoiceDTO** {

- id: Long |

| - serviceRecordId: Long |

| - customerId: Long |

| - issueDate: LocalDateTime |

| - dueDate: LocalDateTime |

| - subtotal: BigDecimal |

| - tax: BigDecimal |

| - discount: BigDecimal |

| - totalAmount: BigDecimal |

| - status: InvoiceStatus |

+--------------------------------------------------------+

| + InvoiceDTO() |

| + getId(): Long |

| + setId(id: Long): void |

| + getServiceRecordId(): Long |

| + setServiceRecordId(serviceRecordId: Long): void |

| + getCustomerId(): Long |

| + setCustomerId(customerId: Long): void |

| + getIssueDate(): LocalDateTime |

| + setIssueDate(issueDate: LocalDateTime): void |

| + getDueDate(): LocalDateTime |

| + setDueDate(dueDate: LocalDateTime): void |

| + getSubtotal(): BigDecimal |

| + setSubtotal(subtotal: BigDecimal): void |

| + getTax(): BigDecimal |

| + setTax(tax: BigDecimal): void |

| + getDiscount(): BigDecimal |

| + setDiscount(discount: BigDecimal): void |

| + getTotalAmount(): BigDecimal |

| + setTotalAmount(totalAmount: BigDecimal): void |

| + getStatus(): InvoiceStatus |

| + setStatus(status: InvoiceStatus): void

**public class PaymentDTO** {

- invoiceId: Long |

| - method: PaymentMethod |

| - amount: BigDecimal |

| - transactionId: String |

| - paymentDate: LocalDateTime |

+----------------------------------------------------+

| + PaymentDTO() |

| + getInvoiceId(): Long |

| + setInvoiceId(invoiceId: Long): void |

| + getMethod(): PaymentMethod |

| + setMethod(method: PaymentMethod): void |

| + getAmount(): BigDecimal |

| + setAmount(amount: BigDecimal): void |

| + getTransactionId(): String |

| + setTransactionId(transactionId: String): void |

| + getPaymentDate(): LocalDateTime |

| + setPaymentDate(paymentDate: LocalDateTime): void

**public class NotificationPreferenceDTO** {

| - receiveEmail: Boolean |

| - receiveSMS: Boolean |

| - receiveAppPush: Boolean |

| - maintenanceReminders: Boolean |

| - paymentReminders: Boolean |

| - promotions: Boolean |

| - preferredChannel: String |

+------------------------------------------------------------------+

| + NotificationPreferenceDTO() |

| + getReceiveEmail(): Boolean |

| + setReceiveEmail(receiveEmail: Boolean): void |

| + getReceiveSMS(): Boolean |

| + setReceiveSMS(receiveSMS: Boolean): void |

| + getReceiveAppPush(): Boolean |

| + setReceiveAppPush(receiveAppPush: Boolean): void |

| + getMaintenanceReminders(): Boolean |

| + setMaintenanceReminders(maintenanceReminders: Boolean): void |

| + getPaymentReminders(): Boolean |

| + setPaymentReminders(paymentReminders: Boolean): void |

| + getPromotions(): Boolean |

| + setPromotions(promotions: Boolean): void |

| + getPreferredChannel(): String |

| + setPreferredChannel(preferredChannel: String): void

**public class ContactPreferenceDTO {**

- preferredContactMethod: String |

| - receivePromotions: Boolean |

| - preferredLanguage: String |

+-------------------------------------------------------------+

| + ContactPreferenceDTO() |

| + getPreferredContactMethod(): String |

| + setPreferredContactMethod(method: String): void |

| + getReceivePromotions(): Boolean |

| + setReceivePromotions(receive: Boolean): void |

| + getPreferredLanguage(): String |

| + setPreferredLanguage(language: String): void

**public class ScheduleDTO** {

| - technicianId: Long |

| - workDate: Date |

| - shift: Shift |

| - startTime: LocalDateTime |

| - endTime: LocalDateTime |

+-------------------------------------------------------------+

| + ScheduleDTO() |

| + getTechnicianId(): Long |

| + setTechnicianId(technicianId: Long): void |

| + getWorkDate(): Date |

| + setWorkDate(workDate: Date): void |

| + getShift(): Shift |

| + setShift(shift: Shift): void |

| + getStartTime(): LocalDateTime |

| + setStartTime(startTime: LocalDateTime): void |

| + getEndTime(): LocalDateTime |

| + setEndTime(endTime: LocalDateTime): void

**public class PartRequestDTO** {

- partId: Long |

| - quantity: Integer |

| - reason: String |

| - isUrgent: Boolean |

+------------------------------------------------------+

| + PartRequestDTO() |

| + getPartId(): Long |

| + setPartId(partId: Long): void |

| + getQuantity(): Integer |

| + setQuantity(quantity: Integer): void |

| + getReason(): String |

| + setReason(reason: String): void |

| + getIsUrgent(): Boolean |

| + setIsUrgent(isUrgent: Boolean): void

**public class NotificationConfigDTO** {

- enableEmailNotifications: Boolean |

| - enableSMSNotifications: Boolean |

| - enablePushNotifications: Boolean |

| - reminderDaysBeforeDue: Integer |

| - reminderHoursBeforeAppointment: Integer |

| - autoSendReminders: Boolean |

+-----------------------------------------------------------+

| + NotificationConfigDTO() |

| + getEnableEmailNotifications(): Boolean |

| + setEnableEmailNotifications(enable: Boolean): void |

| + getEnableSMSNotifications(): Boolean |

| + setEnableSMSNotifications(enable: Boolean): void |

| + getEnablePushNotifications(): Boolean |

| + setEnablePushNotifications(enable: Boolean): void |

| + getReminderDaysBeforeDue(): Integer |

| + setReminderDaysBeforeDue(days: Integer): void |

| + getReminderHoursBeforeAppointment(): Integer |

| + setReminderHoursBeforeAppointment(hours: Integer): void |

| + getAutoSendReminders(): Boolean |

| + setAutoSendReminders(auto: Boolean): void

// ============================================

// FILTER DTOs (Data Transfer Objects)

// ============================================

**public class UserFilterDTO** {

- role: Role |

| - isActive: Boolean |

| - searchKeyword: String |

-----------------------------------------

| + UserFilterDTO() |

| + getRole(): Role |

| + setRole(role: Role): void |

| + getIsActive(): Boolean |

| + setIsActive(isActive: Boolean): void|

| + getSearchKeyword(): String |

| + setSearchKeyword(keyword: String): void

**public class StaffFilterDTO** {

- serviceCenterId: Long |

| - position: String |

| - isActive: Boolean |

--------------------------------------------

| + StaffFilterDTO() |

| + getServiceCenterId(): Long |

| + setServiceCenterId(id: Long): void |

| + getPosition(): String |

| + setPosition(position: String): void |

| + getIsActive(): Boolean |

| + setIsActive(isActive: Boolean): void

**public class TechnicianFilterDTO** {

- serviceCenterId: Long |

| - specialization: String |

| - isActive: Boolean |

| - isAvailable: Boolean |

--------------------------------------------------------

| + TechnicianFilterDTO() |

| + getServiceCenterId(): Long |

| + setServiceCenterId(id: Long): void |

| + getSpecialization(): String |

| + setSpecialization(specialization: String): void |

| + getIsActive(): Boolean |

| + setIsActive(isActive: Boolean): void |

| + getIsAvailable(): Boolean |

| + setIsAvailable(isAvailable: Boolean): void

**public class InvoiceFilterDTO** {

- customerId: Long |

| - status: InvoiceStatus |

| - fromDate: LocalDateTime |

| - toDate: LocalDateTime |

--------------------------------------------------------

| + InvoiceFilterDTO() |

| + getCustomerId(): Long |

| + setCustomerId(id: Long): void |

| + getStatus(): InvoiceStatus |

| + setStatus(status: InvoiceStatus): void |

| + getFromDate(): LocalDateTime |

| + setFromDate(fromDate: LocalDateTime): void |

| + getToDate(): LocalDateTime |

| + setToDate(toDate: LocalDateTime): void

**public class PaymentFilterDTO** {

- customerId: Long |

| - method: PaymentMethod |

| - status: PaymentStatus |

| - fromDate: LocalDateTime |

| - toDate: LocalDateTime |

--------------------------------------------------------

| + PaymentFilterDTO() |

| + getCustomerId(): Long |

| + setCustomerId(id: Long): void |

| + getMethod(): PaymentMethod |

| + setMethod(method: PaymentMethod): void |

| + getStatus(): PaymentStatus |

| + setStatus(status: PaymentStatus): void |

| + getFromDate(): LocalDateTime |

| + setFromDate(fromDate: LocalDateTime): void |

| + getToDate(): LocalDateTime |

| + setToDate(toDate: LocalDateTime): void

**public class InventoryFilterDTO** {

- serviceCenterId: Long |

| - partCategory: String |

| - isLowStock: Boolean |

--------------------------------------------------------

| + InventoryFilterDTO() |

| + getServiceCenterId(): Long |

| + setServiceCenterId(id: Long): void |

| + getPartCategory(): String |

| + setPartCategory(category: String): void |

| + getIsLowStock(): Boolean |

| + setIsLowStock(isLowStock: Boolean): void

**public class PartFilterDTO** {

- category: String |

| - isActive: Boolean |

| - searchKeyword: String |

--------------------------------------------------------

| + PartFilterDTO() |

| + getCategory(): String |

| + setCategory(category: String): void |

| + getIsActive(): Boolean |

| + setIsActive(isActive: Boolean): void |

| + getSearchKeyword(): String |

| + setSearchKeyword(keyword: String): void

**public class AuditFilterDTO** {

- userId: Long |

| - action: String |

| - entityName: String |

| - fromDate: LocalDateTime |

| - toDate: LocalDateTime |

-------------------------------------------------------------

| + AuditFilterDTO() |

| + getUserId(): Long |

| + setUserId(userId: Long): void |

| + getAction(): String |

| + setAction(action: String): void |

| + getEntityName(): String |

| + setEntityName(entityName: String): void |

| + getFromDate(): LocalDateTime |

| + setFromDate(fromDate: LocalDateTime): void |

| + getToDate(): LocalDateTime |

| + setToDate(toDate: LocalDateTime): void T

**Customer 1 →\* Vehicle (vehicles)**

@OneToMany(mappedBy="owner", cascade=CascadeType.ALL, orphanRemoval=true)

Inverse: Vehicle.owner

Description: Customer owns multiple vehicles

**Customer 1 →\* Appointment (appointments)**

@OneToMany(mappedBy="customer")

Inverse: Appointment.customer

Description: Customer books multiple appointments

**Customer 1 →\* CustomerPackage (packages)**

@OneToMany(mappedBy="customer")

Inverse: CustomerPackage.customer

Description: Customer subscribes to multiple maintenance packages

**Customer 1 →\* Invoice (invoices)**

@OneToMany(mappedBy="customer")

Inverse: Invoice.customer

Description: Customer has multiple invoices

**Customer 1 →\* ChatRoom (chatRooms)**

@OneToMany(mappedBy="customer")

Inverse: ChatRoom.customer

Description: Customer can have multiple chat sessions

**Customer 1 → 1 NotificationPreference (notificationPreference)**

@OneToOne(cascade=CascadeType.ALL)

@JoinColumn(name="notification\_preference\_id")

Description: Customer has one notification preference setting

**Customer 1 →\* MaintenanceReminder (reminders)**

@OneToMany(mappedBy="customer")

Inverse: MaintenanceReminder.customer

Description: Customer receives multiple reminders

**Staff N → 1 ServiceCenter (serviceCenter)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="service\_center\_id")

Inverse: ServiceCenter.staff (implicit)

Description: Staff works at one service center

**Staff 1 →\* Appointment (managedAppointments)**

@OneToMany(mappedBy="assignedStaff")

Inverse: Appointment.assignedStaff

Description: Staff manages multiple appointments

**Staff 1 →\* ChatRoom (chatRooms)**

@OneToMany(mappedBy="staff")

Inverse: ChatRoom.staff

Description: Staff handles multiple chat rooms

**Technician N → 1 ServiceCenter (serviceCenter)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="service\_center\_id")

Description: Technician works at one service center

**Technician 1 →\* TechnicianSchedule (schedules)**

@OneToMany(mappedBy="technician", cascade=CascadeType.ALL)

Inverse: TechnicianSchedule.technician

Description: Technician has multiple work schedules

**Technician 1 →\* ServiceRecord (assignedServices)**

@OneToMany(mappedBy="technician")

Inverse: ServiceRecord.technician

Description: Technician performs multiple services

**Vehicle N → 1 Customer (owner)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="customer\_id", nullable=false)

Inverse: Customer.vehicles

Description: Vehicle belongs to one customer

**Vehicle N → 1 VehicleModel (model)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="vehicle\_model\_id", nullable=false)

Description: Vehicle is of one model

**Vehicle 1 →\* ServiceRecord (serviceRecords)**

@OneToMany(mappedBy="vehicle")

Inverse: ServiceRecord.vehicle

Description: Vehicle has multiple service records

**Vehicle 1 →\* Appointment (appointments)**

@OneToMany(mappedBy="vehicle")

Inverse: Appointment.vehicle

Description: Vehicle has multiple appointments

**Vehicle 1 →\* MaintenanceReminder (reminders)**

@OneToMany(mappedBy="vehicle")

Inverse: MaintenanceReminder.vehicle

Description: Vehicle has multiple maintenance reminders

**VehicleModel 1 →\* Vehicle (vehicles)**

Implicit inverse of Vehicle.model

Description: One model has many vehicles

**Appointment N → 1 Customer (customer)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="customer\_id", nullable=false)

Inverse: Customer.appointments

Description: Appointment belongs to one customer

**Appointment N → 1 Vehicle (vehicle)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="vehicle\_id", nullable=false)

Inverse: Vehicle.appointments

Description: Appointment is for one vehicle

**Appointment N → 1 ServiceCenter (serviceCenter)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="service\_center\_id", nullable=false)

Description: Appointment at one service center

**Appointment N → 1 ServiceType (serviceType)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="service\_type\_id", nullable=false)

Description: Appointment requests one service type

**Appointment N → 1 Staff (assignedStaff)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="assigned\_staff\_id")

Inverse: Staff.managedAppointments

Description: Appointment assigned to one staff member (nullable)

**Appointment \*→ 1 AppointmentStatus (status)**

@Enumerated(EnumType.STRING)

Description: Appointment has one status

**Appointment 1 → 0..1 ServiceRecord (serviceRecord)**

Implicit inverse of ServiceRecord.appointment

Description: Appointment may result in one service record

**ServiceRecord N → 0..1 Appointment (appointment)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="appointment\_id")

Description: Service record may come from appointment (nullable for walk-ins)

**ServiceRecord N → 1 Vehicle (vehicle)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="vehicle\_id", nullable=false)

Inverse: Vehicle.serviceRecords

Description: Service record is for one vehicle

**ServiceRecord N → 1 Technician (technician)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="technician\_id", nullable=false)

Inverse: Technician.assignedServices

Description: Service record performed by one technician

**ServiceRecord N → 1 ServiceType (serviceType)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="service\_type\_id", nullable=false)

Description: Service record is of one service type

**ServiceRecord \*→ 1 ServiceStatus (status)**

@Enumerated(EnumType.STRING)

Description: Service record has one status

**ServiceRecord 1 →\* ServiceRecordPart (usedParts)**

@OneToMany(mappedBy="serviceRecord", cascade=CascadeType.ALL, orphanRemoval=true)

Inverse: ServiceRecordPart.serviceRecord

Description: Service record uses multiple parts

**ServiceRecord 1 →\* Checklist (checklists)**

@OneToMany(mappedBy="serviceRecord", cascade=CascadeType.ALL, orphanRemoval=true)

Inverse: Checklist.serviceRecord

Description: Service record has multiple checklist items

**ServiceRecord 1 → 0..1 Invoice (invoice)**

Implicit inverse of Invoice.serviceRecord

Description: Service record generates one invoice

**ServiceRecordPart N → 1 ServiceRecord (serviceRecord)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="service\_record\_id", nullable=false)

Inverse: ServiceRecord.usedParts

Description: Part usage belongs to one service record

**ServiceRecordPart N → 1 Part (part)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="part\_id", nullable=false)

Description: Part usage references one part

**Checklist N → 1 ServiceRecord (serviceRecord)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="service\_record\_id", nullable=false)

Inverse: ServiceRecord.checklists

Description: Checklist item belongs to one service record

**Checklist \*→ 1 ChecklistCategory (category)**

@Enumerated(EnumType.STRING)

Description: Checklist item has one category

**ServiceType 1 →\* Appointment (appointments)**

Implicit inverse of Appointment.serviceType

Description: Service type used in multiple appointments

**ServiceType 1 →\* ServiceRecord (serviceRecords)**

Implicit inverse of ServiceRecord.serviceType

Description: Service type used in multiple service records

**Part 1 →\* Inventory (inventoryRecords)**

@OneToMany(mappedBy="part")

Inverse: Inventory.part

Description: Part stocked in multiple service centers

**Part 1 →\* ServiceRecordPart (usages)**

@OneToMany(mappedBy="part")

Inverse: ServiceRecordPart.part

Description: Part used in multiple service records

**Inventory N → 1 ServiceCenter (serviceCenter)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="service\_center\_id", nullable=false)

Inverse: ServiceCenter.inventory

Description: Inventory at one service center

**Inventory N → 1 Part (part)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="part\_id", nullable=false)

Inverse: Part.inventoryRecords

Description: Inventory tracks one part

Constraint: Unique(serviceCenter, part) - one inventory record per part per center

**Invoice N → 1 ServiceRecord (serviceRecord)**

@OneToOne(fetch=FetchType.LAZY)

@JoinColumn(name="service\_record\_id", unique=true, nullable=false)

Inverse: ServiceRecord.invoice

Description: Invoice generated from one service record

**Invoice N → 1 Customer (customer)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="customer\_id", nullable=false)

Inverse: Customer.invoices

Description: Invoice belongs to one customer

**Invoice \*→ 1 InvoiceStatus (status)**

@Enumerated(EnumType.STRING)

Description: Invoice has one status

**Invoice 1 →\* Payment (payments)**

@OneToMany(mappedBy="invoice", cascade=CascadeType.ALL)

Inverse: Payment.invoice

Description: Invoice can have multiple payments (partial payments)

**Payment N → 1 Invoice (invoice)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="invoice\_id", nullable=false)

Inverse: Invoice.payments

Description: Payment pays one invoice

**Payment \*→ 1 PaymentMethod (method)**

@Enumerated(EnumType.STRING)

Description: Payment uses one method

**Payment \*→ 1 PaymentStatus (status)**

@Enumerated(EnumType.STRING)

Description: Payment has one status

**MaintenancePackage 1 →\* CustomerPackage (subscriptions)**

@OneToMany(mappedBy="package")

Inverse: CustomerPackage.package

Description: Package has multiple customer subscriptions

**MaintenancePackage \*→ 1 PackageStatus (implicit through CustomerPackage)**

Description: Package availability status

**CustomerPackage N → 1 Customer (customer)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="customer\_id", nullable=false)

Inverse: Customer.packages

Description: Package subscription belongs to one customer

**CustomerPackage N → 1 MaintenancePackage (package)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="package\_id", nullable=false)

Inverse: MaintenancePackage.subscriptions

Description: Subscription is for one maintenance package

**CustomerPackage \*→ 1 PackageStatus (status)**

@Enumerated(EnumType.STRING)

Description: Subscription has one status

**Notification N → 1 User (recipient)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="recipient\_id", nullable=false)

Description: Notification sent to one user (Customer, Staff, Technician, Admin)

**Notification \*→ 1 NotificationType (type)**

@Enumerated(EnumType.STRING)

Description: Notification has one type

**ChatRoom N → 1 Customer (customer)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="customer\_id", nullable=false)

Inverse: Customer.chatRooms

Description: Chat room with one customer

**ChatRoom N → 1 Staff (staff)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="staff\_id", nullable=false)

Inverse: Staff.chatRooms

Description: Chat room handled by one staff member

**ChatRoom 1 →\* ChatMessage (messages)**

@OneToMany(mappedBy="room", cascade=CascadeType.ALL, orphanRemoval=true)

Inverse: ChatMessage.room

Description: Chat room contains multiple messages

**ChatMessage N → 1 ChatRoom (room)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="room\_id", nullable=false)

Inverse: ChatRoom.messages

Description: Message belongs to one chat room

**ChatMessage N → 1 User (sender)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="sender\_id", nullable=false)

Description: Message sent by one user (Customer or Staff)

**MaintenanceReminder N → 1 Customer (customer)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="customer\_id", nullable=false)

Inverse: Customer.reminders (implicit)

Description: Reminder sent to one customer

**MaintenanceReminder N → 1 Vehicle (vehicle)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="vehicle\_id", nullable=false)

Inverse: Vehicle.reminders

Description: Reminder is for one vehicle

**MaintenanceReminder \*→ 1 ReminderType (type)**

@Enumerated(EnumType.STRING)

Description: Reminder has one type (MILEAGE\_BASED or TIME\_BASED)

**NotificationPreference 1 → 1 Customer (customer)**

@OneToOne(fetch=FetchType.LAZY)

@JoinColumn(name="customer\_id", unique=true, nullable=false)

Inverse: Customer.notificationPreference

Description: Preference belongs to one customer

**ServiceCenter 1 →\* Staff (staff)**

Implicit inverse of Staff.serviceCenter

Description: Service center employs multiple staff

**ServiceCenter 1 →\* Technician (technicians)**

Implicit inverse of Technician.serviceCenter

Description: Service center employs multiple technicians

**ServiceCenter 1 →\* Inventory (inventory)**

@OneToMany(mappedBy="serviceCenter", cascade=CascadeType.ALL)

Inverse: Inventory.serviceCenter

Description: Service center maintains multiple inventory records

**ServiceCenter 1 →\* Appointment (appointments)**

Implicit inverse of Appointment.serviceCenter

Description: Service center handles multiple appointments

**TechnicianSchedule N → 1 Technician (technician)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="technician\_id", nullable=false)

Inverse: Technician.schedules

Description: Schedule belongs to one technician

**TechnicianSchedule \*→ 1 Shift (shift)**

@Enumerated(EnumType.STRING)

Description: Schedule has one shift

**TechnicianSchedule \*→ 1 ScheduleStatus (status)**

@Enumerated(EnumType.STRING)

Description: Schedule has one status

**AuditLog N → 1 User (actor)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="actor\_id")

Description: Audit log records action by one user

**ActivityLog N → 1 User (user)**

@ManyToOne(fetch=FetchType.LAZY)

@JoinColumn(name="user\_id", nullable=false)

Description: Activity log tracks one user's activity

**Customer \* → 1 CustomerType (type)**

**Customer \* → 1 CustomerTier (tier)**

**Technician 1 → \* ServiceFeedback (receivedFeedbacks)**

**Customer 1 → \* ServiceFeedback (feedbacks)**

**ServiceRecord 1 → 0..1 ServiceFeedback (feedback)**

**ServiceFeedback N → 1 Technician (technician)**

**ServiceFeedback N → 1 Customer (customer)**

**ServiceFeedback N → 1 ServiceRecord (serviceRecord)**

**PartUsageHistory N → 1 Part (part)**

**PartUsageHistory N → 1 ServiceCenter (serviceCenter)**

**PartUsageHistory N → 0..1 ServiceRecord (serviceRecord)**

**Part 1 → \* PartUsageHistory (usageHistory)**

**ServiceCenter 1 → \* PartUsageHistory (partUsageHistory)**