System documentation

SYSA21: Introduktion till informationssystem | 2024

Group: **25**
GitHub Team: **HT23-SYSA21-Grupp25**
GitHub Repository:
https://github.com/lu-informatics/ht23-sysa21-isp-ht23-sysa21-grupp25.git

021001-6580	Sänna Janfada Balu	Sanna274
971121-6730	Mirnes Dizdar	MirnesDizdar
940818-0314	Simon Baas	Kamiyumi

System documentation for Viking Express – logistics management system

System Objective

VikingExpress, a logistics company, has deployed a comprehensive system designed to centralize and streamline the management of its vehicular assets. This system consolidates various facets of logistics management including vehicle, cargo, and employee coordination, maintenance planning, service activity logging, and workshop interactions.
In alignment with VikingExpress's established protocols, the system ensures compliance with company standards and operational mandates. It showcases functionalities such as vehicle identification number (VIN) display, vehicle type classification, location tracking, service history access, and maintenance scheduling. It also includes detailed workshop affiliations, all of which are accessible through intuitively labeled tabs and buttons within the primary user interface. Each segment is dedicated to handling specific operational aspects:

Vehicle Management: Offers comprehensive tools to oversee the vehicle fleet, providing capabilities to add, modify, or delete vehicle records, while tracking intricate details like cargo specifics, service history, and cumulative service expenses.

Cargo Management: Facilitates cargo-related processes with features to insert, adjust, or discard cargo entries, alongside efficient vehicle allocation systems.

Employee Management: Serves as a repository for employee data, assigns personnel to vehicles, and monitors their associated vehicular activities.

Maintenance Scheduling: Provides a structured approach to maintaining vehicles, selecting service venues, and documenting upkeep timelines.

Service Activity Monitoring: Captures and administers details of vehicle service interventions, inclusive of comprehensive historical records and workshop visitations.

Workshop Oversight: Manages workshop affiliations, enabling the addition, alteration, or elimination of workshop particulars.

Calculations: Delivers analytical insights into vehicle-related finances, such as average service costs and the identification of the most expensive service provider.

The system also incorporates VikingExpress' specific restrictions, for instance, preventing the input of negative values for vehicle capacity by alerting users with error notifications when such inputs are attempted.

System Architecture

The application is engineered using Java within Visual Studio Code and integrates Scene Builder with JavaFX for crafting the user interface. Data storage is file-based, leveraging the existing package from ics.lu.se. Organized under the "models" package, the system architecture includes classes for Vehicle, Employee, Cargo, Maintenance, Factory, ServiceActivity, ServiceHistory, and Workshop. The Factory class contains various constructors for these model entities. The "TestDataPopulator" within this package is tasked with managing test data.
"SharedVehicleDataModel" is a central data model that manages observable lists from the classes to facilitate smoother associations. Additionally, CSS from a third-party library enhances the user interface button styling. Canva.com were used to create the logos for the application.
Moreover, an alternative testing entry point using Azure SQL Server is established to expand the application's scalability. Test data is categorized under relevant table names accessible in the "TestDataPopulatorSql" class, complete with corresponding SQL queries. The Shuffle-sort insertion method randomizes and integrates strings into the data model.

Before starting, confirm the installation of all software dependencies, such as:

- *Javafx.fxml*
- *Javafx.controls*
- *Java.sql*
- *Javafx.graphics*

Usage Instructions

- Start the Application
- Interface Navigation
- The main interface is segmented into distinct areas for
    - Vehicles
    - Cargo
    - Employees
    - Maintenance,
    - Services
    - Workshops.
    - Calculations
    - About me
- Navigate through these sections using clearly marked tabs and buttons.

Known Issues:

- A CSS parsing error occurs on startup due to an unexpected 'scale' function encountered while parsing the transform property at [31,15] in the /css/button.css directory.
- If the SQL server is inactive, activating the SQL button in the MainView scene will cause the application to terminate abruptly.

Recommendations for future versions

- Incorporate graphical representations or models for a more dynamic financial overview.
- Establish a binary connection to the SQL server, enabling data transmission from the front end to the backend.
- Automate data initialization from the SQL server during application startup to ensure a consistent and uninterrupted user experience with a stable dataset.
- Allow resizable window (full screen view).
- Expanded error handling, to give even clearer instructions on errors committed by the user.
- More detailed user profiles with pictures of the employees and vehicles.

Not explicitly defined in the system. All methods are used in the respective classes

<<Interface>>
Utility

+ setSharedDataModel(sharedDataModel: SharedVehicleDataModel): void
- applyDynamicTooltip(Control control, String tooltipText): void
- createDelayedTooltip(String text): Tooltip
- initializeSearchFunctionality(): void
+ initialize(): void
- setupTableColumns(): void

Omitted all @FXML handlers and attributes. Check FXML file for reference. This applies to all controllers

«artifact»
VehicleView.fxml

«artifact»
MainView.fxml

VehicleController

- loadDataIntoUI(): void
- setupComboBoxConverters(): void
- setupTableColumns(): void
- filterTable(searchQuery: String): void
- generateUniqueVin(): int
- vinExists(vin: int): boolean
- formatVisitedWorkshops(vehicle: Vehicle): String

MainViewController

- filterTable(searchQuery: String): void
- updateClock(): void
- addWindowFocusListener(): void
- setupTableColumns(): void
+ refreshMainViewTable(): void
- initializeSharedDataModel(): void

EmployeeController

- loadDataIntoUI(): void
- filterEmployeeTable(searchQuery: String): void
- setupComboBoxConverter(): void
- setupTableColumns(): void

«artifact»
EmployeeView.fxml

SharedVehicleDataModel

- vehicles: ObservableList<Vehicle>
- vehiclesCargo : ObservableList<Cargo>
- maintanenceSchedules: ObservableList <MaintanenceSchedule>
- workshops: ObservableList<Workshop>
- employees: ObservableList <Employee>
- serviceActivities : ObservableList <ServiceActivity>

+ addServiceActivity(serviceActivity: ServiceActivity): void
+ removeServiceActivity(serviceActivity: ServiceActivity): void
+ addAllServiceActivities(serviceActivityList: List<ServiceActivity>): void
+ addEmployee(employee: Employee): void
+ addAllEmployees(employeeList: List<Employee>): void
+ addCargoToVehicle(cargo: Cargo, vehicle: Vehicle): void
+ addAllCargoFromVehicles(): void
+ addVehicle(vehicle: Vehicle): void
+ addAllVehicles(vehicleList: List<Vehicle>): void
+ addMaintenanceSchedule(schedule: MaintenanceSchedule): void
+ removeMaintenanceSchedule(schedule: MaintenanceSchedule): void
+ addWorkshop(workshop: Workshop): void
+ addAllWorkshops(workshopList: List<Workshop>): void
+ removeWorkshop(workshop: Workshop): void
+ getMostExpensiveServiceActivity(): String
+ getAverageServiceCost(): double
+ getMostExpensiveWorkshopDetails(): String
+ getTotalServiceCost(): double

MaintanenceController

- updateComboBoxItems(): void
- initializeTableColumns(): void
+ populateMaintenanceTableView(): void
- filterMaintenanceTable(searchQuery: String): void

«artifact»
MaintanenceView.fxml

CargoController

- updateComboBoxItems(): void
- initializeTableColumns(): void
+ populateCargoTableView(): void
- filterCargoTable(searchQuery: String): void
+ addCargo(cargo: Cargo): void
+ removeCargo(cargo: Cargo): void
- clearCargoInputFields(): void

«artifact»
CargoView.fxml

ServiceController

- loadDataIntoUI(): void
- setupComboBoxConverters(): void
- calculateTotalPartsReplaced(vehicle: Vehicle): int
- showDecommissionAlert(vehicle: Vehicle): void
- showDecommissionAlert2(vehicle: Vehicle): void
- showDecommissionAlert3(vehicle: Vehicle): void
- filterServiceTable(String searchQuery): void

«artifact»
ServiceView.fxml

WorkshopController

- loadDataIntoUI(): void
- setupTableColumns(): void
- clearInputFields(): void
- filterTable(searchQuery: String): void

«artifact»
WorkshopView.fxml

Omitted 0:Many associatations for: Cargo, MaintanenceSchedule, Workshop, ServiceActivity for abundancy in model.

Employee

- name: String
- ssNo : int {readOnly, unique}
- address: String
- phoneNumber: String
- employeeType: String

+ Employee (String: name, int: SsNo, String: address, String: phoneNumber, String: employeeType)
+ toString(): String

Vehicle

- vin: int {unique}
- vehicleType : String
- name : String
- location : String
- totalServiceCost: double
- weightCapacity: double
- cargoWeight: double
- partsReplaced: int
- cargos: ObservableList <Cargo>

+ Vehicle()
+ Vehicle(int : vin, String : vehicleType, String : name, String : location, Cargo: cargo, double: weightCapacity, int : partsReplaced, ServiceHistory: serviceHistory, MaintanenceSchedule : vehicleMaintanenceSchedule, ArrayList<Workshop> : visistedWorkshops)
+ addTotalServiceCost(double: totalServiceCost
+ addCargoWeight(double: cargoWeight)
+ removeCargoWeight(double: cargoWeight)
+ assignDriver(Employee driver): void
+ displayServiceHistory(): serviceHistory.toString()
+ displayVisitedWorkshops() : visistedWorkshops.toString()
+ addResponsibleDriver(Employee driver): void
+ removeResponsibleDriver(Employee driver): void
+ addVisitedWorkshop(Workshop workshop): void
+ removVisitedWorkshop(Workshop workshop): void
+ recalculateTotalServiceCost(): void
+ recalculateTotalCargoWeight(): void
+ addCargo(Cargo cargo): void
+ removeCargo(Cargo cargo): void
+ getCargoDetails(): String
+ toString(): String
+ toStringTwo(): String
+ getDriverNames () String
+ getServiceHistoryDetails(): String
+ getMaintanenceDetails(): String
+ getVisitedWorkshopsDetails(): String
+ calculateCargoWeight () : double

MaintanaceSchedule

- scheduledMaintenanceDate : LocalDate
- maintenanceDescription: String

+ MaintenanceSchedule(Vehicle: scheduleForVehicle, localDate: scheduledMaintenanceDate, String: ´maintenanceDescription)
+ toString(): String
+ isEmpty() : boolean

Cargo

- cargoName : String
- cargoWeight: double
- cargoTotalWeight: double
- cargoAmount: int
- ean: int

+ Cargo(String : cargoName, double : cargoWeight, int : cargoAmount, int : ean)
+ toString(): String

has 2 Lists of cargo

getDetailsFunctions uses stream().map().collect() to append into one string
---
toStringTwo omits ServiceHistory, MaintainenceSchedule, VisitedWorkshops

Workshop

- name: String
- address: String
- isInternal: boolean
- totalCost : double
- averageCost : double

+ addVehicle(): void
+ removeVehicle(): void
+ addServiceActivity(): void
+ removeServiceActivity(): void
+ calculateCosts() void
+ toString(): String

ServiceHistory

+ addServiceActivityToHistory(ServiceActivity: serviceActivity) : void
+ removeServiceActivityFromHistory(ServiceActivity: serviceActivity): void
+ toString(): String
+ isEmpty(): boolean

ServiceActivity

- dateOfService : localDate
- serviceDescription: String
- serviceStatus: String
- serviceCost: double
- serviceStatus : String
- partsReplaced: String

+ ServiceActivity(Workshop workshop, Vehicle serviceForVehicle, ServiceHistory partOfServiceHistory, LocalDate dateOfService, String serviceDescription, double serviceCost, String serviceStatus, String partsReplaced)
+toString(): String

Factory

+ createVehicle(): Vehicle
+ createEmployee(): Employee
+ createCargo(): Cargo
+ createMaintenanceSchedule(): MaintenanceHistory
+ createServiceActivity(): ServiceActivity
+ createServiceHistory(): ServiceHistory
+ createWorshop(): Workshop

<<uses>>
hasListOf
responsibleDrivers
stores
has
responsibleWorkshop
serviced
visitedWorkshops
partOf
responsibleFor
<<creates>>