Car Dealership Management System

BY: AMR ASHRAF MAHMOUD ABDELRAZEK

Full Functionality Report

This document explains the entire functionality and behavior of the Car Dealership Management System developed in JavaFX with a MySQL backend. All actions, checks, flows, and structural decisions are described in a detailed and functional format.

1. Authentication and Session Management

- Admin login is credential-locked with hardcoded values. If the credentials are correct, admin data is loaded from the database (name, balance, history, etc.). If not, login is denied.
- User signup includes input validation. If any field is missing or the username already exists, the operation fails with an appropriate message.
- User login verifies credentials against the database. A match grants access and loads the user's balance, name, and cars. A mismatch rejects the login.
- Session state is stored in memory controllers. All logged-in roles maintain an active session until explicitly logged out or the app is closed.

2. Admin Functionalities

- Add Car: Requires all fields (name, model, price, type, image). If any are missing, it shows an error. If filled, it checks balance. If sufficient, the car is added and the price is deducted from balance and logged in 'moneySpent'.
- Edit Car: Only available for cars that haven't been sold. Any car marked as sold is locked from editing to preserve historical accuracy.
- Delete Car: Allowed only for available (unsold) cars. Deleting removes the car from the DB and the live UI. Sold cars remain for record integrity.
- Wallet - Deposit: Accepts any positive amount. On success, the balance is updated and deposit history is logged. Invalid or empty input is ignored.
- Wallet - Withdraw: Allowed if amount is positive and less than or equal to current balance. The balance is reduced and withdrawal info is saved. Else, it’s rejected.
- Sold Cars Page: Admin can view all sold cars, along with usernames of buyers and timestamps of sale. This table updates automatically.
- Balance Tracking: Every deposit, withdrawal, or car addition immediately reflects in the balance and gets saved to the DB.

3. User Functionalities

- Browse Cars: All cars not marked as sold are shown. If a car is bought or deleted, it's removed live from the list. The UI uses threads to auto-refresh.
- Deposit to Wallet: Accepts positive numeric input only. If valid, updates the user's balance in real-time. Invalid input is ignored.
- Buy Car: Before completing purchase, system checks if the user's balance is sufficient and if the car is still available. If both pass, the transaction goes through — user balance drops, car is moved to 'Owned Cars', and admin gets the money.
- Owned Cars Page: Instantly shows cars that the user has purchased. Pulled directly from DB and reflects every successful purchase.

4. Real-Time Synchronization

- Admin and User dashboards each have background threads that poll the database at intervals.
- UI updates occur live — car lists, wallet balances, and sold/bought status update without needing refresh.
- Admin dashboard auto-refreshes balance, available cars, and sold car logs.
- User dashboard auto-refreshes available cars and wallet status to prevent outdated or duplicate views.

5. Data Architecture and Consistency

- Admin object is singleton. There's always one active admin session, fetched using getInstance().
- User objects are passed between screens after login. Controllers hold the active user instance.
- All core operations (car transactions, wallet edits, login data) are stored in a MySQL database.
- No values exist only in memory. All persistent states are fetched from DB and updated on every change.
- If the app is closed and reopened, the system fetches the most recent values — there is no session loss.

This system is designed to act like a real-world management solution. Every user action is validated, no state changes without confirmation, and the backend and UI are fully connected. Edge cases (like insufficient balance or already sold cars) are handled cleanly to prevent errors or false data.