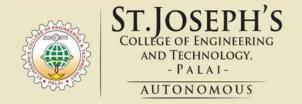
Dept. of **Computer Science and Engineering**



24SJPBCST304 OBJECT ORIENTED PROGRAMMING Project Abstract

| | | TEAM DETAILS | |
|-------|------------------|--------------------|---------------------|
| SI No | Class Roll No | University Reg. No | Name of the Student |
| 01 | 11 | SJC24CS036 | Alex Roy |
| 02 | 15 | SJC24CS044 | Alwin Jose George |
| 03 | 35 | SJC24CS113 | Febin Nobel |
| 04 | 41 | SJC24CS131 | Joe Martin Rince |

PROJECT DETAILS

Project Title:

Car Rental Management System (CRMS) with Integrated Approval Workflow and Local Persistence

Project Abstract:

Car Rental Management System (CRMS) - Comprehensive Project Abstract

The **Car Rental Management System (CRMS)** is a robust, multi-user desktop application developed as the capstone project for the **24SJPBCST304 – Object-Oriented Programming** course. This system provides a sophisticated platform for managing the entire lifecycle of a car rental operation, from vehicle listing and customer booking to final approval and status tracking.

m Architecture and Technology Stack

The system is engineered for **modularity, usability, and scalability**, strictly adhering to the **Model-View-Controller (MVC)** architectural pattern. This separation of concerns ensures a clear division between the data handling, business logic, and presentation layers, significantly enhancing **maintainability** and facilitating future feature **extensibility**.

- View Layer (Presentation): Utilizes Java Swing to deliver a responsive, native-feeling Graphical User Interface (GUI). This layer is designed with a professional UI/UX, featuring clean, minimalist layouts with Navy headers and Light backgrounds, alongside intuitive color-coded status indicators for immediate data recognition.
- Model Layer (Data & Logic): This layer encapsulates the core business logic and data access. It employs JDBC (Java Database Connectivity) to manage transactions.
- Data Persistence: The CRMS leverages SQLite for secure, persistent, file-based data storage (carrental.db). This crucial design choice eliminates the dependency on external network-based database servers, ensuring high portability and simplified deployment while maintaining data integrity.

Object-Oriented Programming (OOP) Implementation

The CRMS serves as a practical demonstration of advanced OOP principles:

| OOP Concept | Application within CRMS |
|---------------|--|
| Encapsulation | User and Car Classes: All sensitive attributes (like passwords, car status, booking IDs) are |

| | encapsulated within their respective classes (Admin, Seller, Customer, Car), with access controlled solely through public getter and setter methods. This ensures data security and integrity. |
|--------------|--|
| Inheritance | User Hierarchy: A base User class is extended by specialized subclasses like Admin, Seller , and Customer . This structure allows for code reuse while enabling each subclass to implement its unique functionalities and dashboard views. |
| Polymorphism | Role-Based Operations: The system uses polymorphism to process actions (e.g., login, view dashboard) differently based on the user object's actual type (Seller vs. Customer). |
| Abstraction | Service Layers: Abstract classes or Interfaces (e.g., BookingService or DBConnector) define contracts for core functionalities, hiding the underlying complexity of JDBC or business rule validation from the UI components. |

Core Features & Functional Highlights

The core functionality mirrors a real-world business process, ensuring a high level of operational realism:

- 1. Multi-Role Authentication: Supports distinct roles (Admin, Seller, Customer) with segregated access permissions and dedicated dashboard views.
- 2. Integrated Approval Workflow (Transaction Management): Each customer booking is initialized with a **Pending (Orange)** status. This mandates a **Seller** review, demonstrating a critical business control point.
- 3. Role-Based Notifications: Upon login, the Seller is instantly notified of all Pending booking requests specifically associated with their listed vehicles, facilitating timely action.
- 4. Transaction Finalization: Sellers can explicitly Confirm a booking (updating the booking status to Green and the car status to "Rented") or Reject it (reverting the car status to "Available"), ensuring accurate inventory tracking.
- 5. Robust Data Management (CRUD): Complete Create, Read, Update, and Delete (CRUD) operations are implemented for all entities (User Accounts, Car Listings, Booking Records), guaranteeing the reliability and integrity of the local database.

This project successfully demonstrates the ability to translate complex business requirements into a robust, object-oriented desktop software solution using industry-standard architectural patterns and modern Java technology.