



OOAD with Java
UE20CS352
PROJECT REPORT
“SMART CITY – PARKING SYSTEM”

SUBMITTED BY :

NAME :

SRN:

KANIKARAM ROOPA SREESAI

PES2UG20CS154

KOYYA DEEKSHITHA

PES2UG20CS167

MEKALA SANJANA

PES2UG20CS194

KUMARI SHIVANGI

PES2UG20CS173

PROBLEM STATEMENT / SYNOPSIS :

A car parking management system is a software solution that helps manage parking operations in a more efficient and organized way. The system can be used in various settings such as commercial parking lots, public parking spaces, private buildings, and institutions.

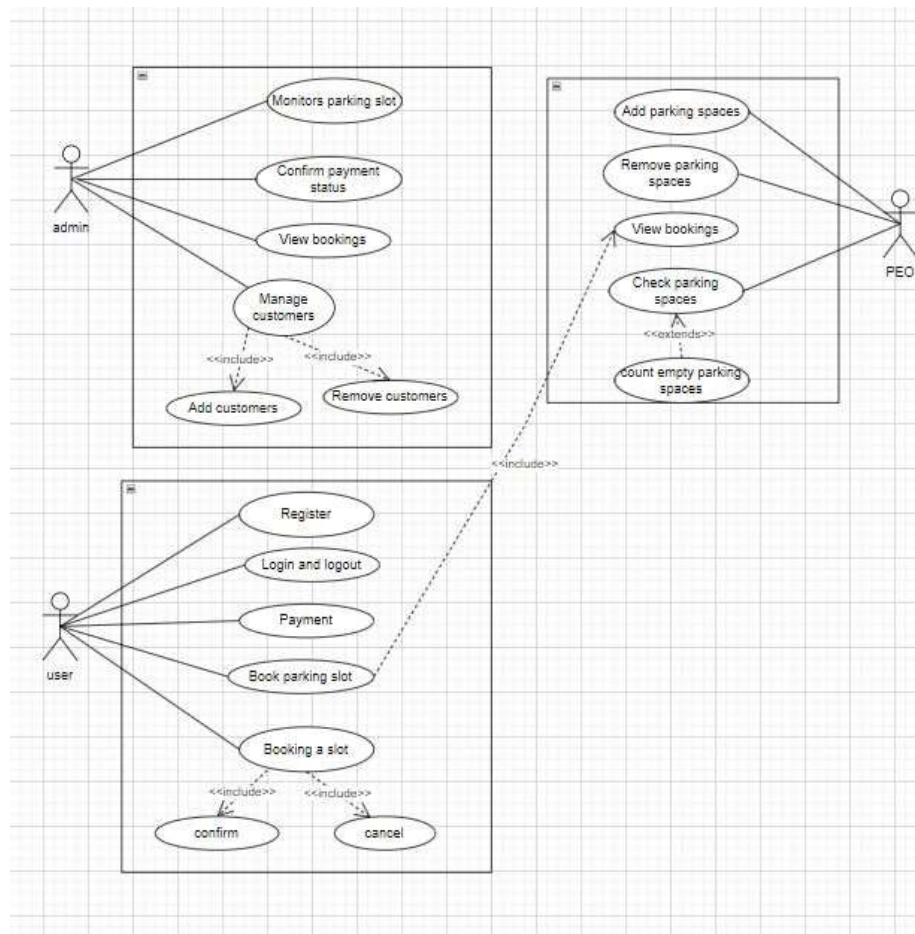
Here are some features that a car parking management system offers:

- registers / log-in
- book any parking space at any time, or for any time at the specified locations
- payment
- parking enforcement

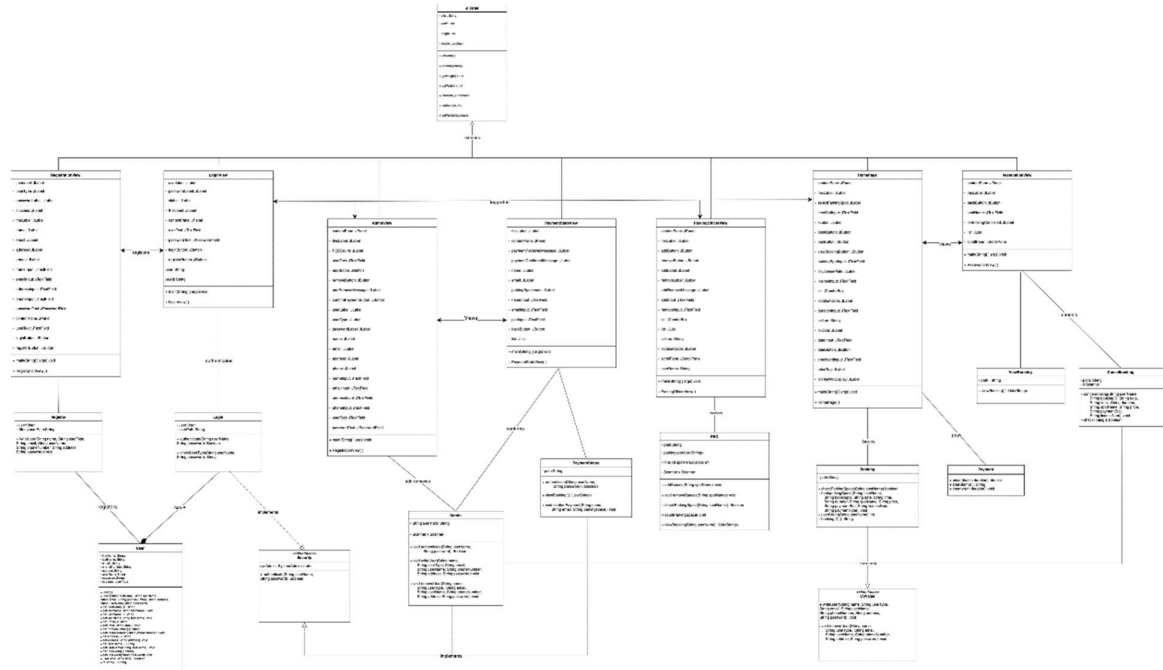
Overall, this project can help a lot with parking lot management in the age where more than 70% of population owns a vehicle or two. It is also useful for a city, where population bust happens.

MODELS :

1). USE CASE MODEL –



2). CLASS MODEL –



ARCHITECTURE PATTERNS :

In Java Swing, a common architecture pattern for a car parking management system is the Model-View-Controller (MVC) architecture pattern.

In this pattern, the system is divided into three main components:

1. **Model**: This component is responsible for managing the data and business logic of the system. It could contain classes to manage parking lot data, parking rates, and payment transactions.
2. **View**: This component is responsible for displaying the user interface to the user. In the context of a car parking management system using Java Swing, the View would be implemented as a set of UI components such as JPanels, JTextFields, JButtons, and JTables.
3. **Controller**: This component is responsible for managing the communication between the View and the Model. It receives input from the View, passes it to the Model for processing, and updates the View with the results. In the context of a car parking management system using Java Swing, the Controller would contain classes to manage user input, handle payment transactions, and update the View with the latest parking lot information.

The Java Swing MVC architecture pattern can be implemented using various approaches. One popular approach is to use an event-driven design, where user actions and system events trigger methods in the Controller, which in turn update the Model and View components.

Overall, the Java Swing MVC architecture pattern provides a scalable and maintainable solution for a car parking management system, allowing for separation of concerns between the data, presentation, and communication components while using the Swing UI toolkit to create a rich and interactive user interface.

DESIGN PATTERNS :

- **Singleton Design Pattern:** Singleton because, we are creating objects from classes which can be accessed directly without need to instantiate the object of the class. And the classes are required to control the execution of the program.

- **Builder Design pattern :** Here, we create a complex object that is built using simple objects in a step by step approach. Additionally, we are separating the implementation and representation of the object and hiding implementation details of the object from the user. The primary reason to use builder pattern is that it, separates the implementation of a complex object from its representation so that the same construction process can create different representations. In here, we have different representations and different functionalities for the representations. Builder pattern provides the ease of separating them and provides greater efficiency and control over the implementation process.

- **Data Access Object (DAO) Pattern :** This pattern is used to separate low level data accessing API or operations from high level services. The DAO interface, CSVOps, defines the standard operations to be performed on model. The DAO concrete class, Admin, Booking, and cancelBooking implements the interface, getting the data from the data source, and Model, payment, ViewBooking, stores the data retrieved by the concrete class.

Design Principles:

a) Single Responsibility Principle (SRP): This principle states that a class should have only one reason to change. This principle can be applied to parking management to ensure that each component of the application has a single, well-defined responsibility.

b)Open/closed Principle (OCP) :This principle promotes software entities (classes, modules, functions, etc.) open for extension but closed for modification. This principle can be used to make parking management more modular and testable.

c) Dependency Inversion (DI): This principle promotes loose coupling between components by allowing them to depend on abstractions rather than concrete implementations. This principle can be used to make parking management system development more modular and testable.

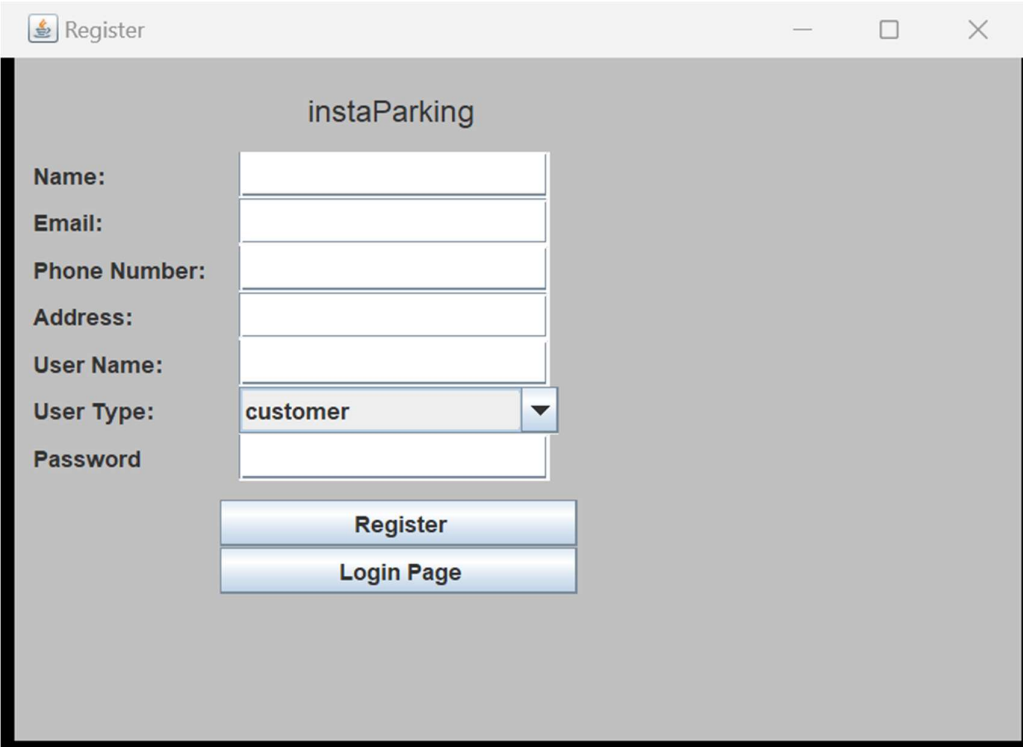
GITHUB LINK FOR THE CODE :

https://github.com/de-limiter/Smart_city_parking_management_system.git

Tools:

- Language: Java 8
- Framework: Java swing
- Build tool: Maven
- Testing Framework : JUnit

SCREENSHOTS WITH INPUT VALUES POPULATED AND OUTPUT SHOWN :



The screenshot shows a Java Swing window titled "Register" with a standard macOS-style title bar (minimize, maximize, close buttons). The window content has a light gray background. At the top center, the text "instaParking" is displayed. Below this, there are seven labels on the left side, each followed by a text input field: "Name:", "Email:", "Phone Number:", "Address:", "User Name:", "User Type:", and "Password". The "User Type" field is a dropdown menu with "customer" selected. Below the input fields, there are two buttons: "Register" and "Login Page".

Field Label	Value / Selection
Name:	
Email:	
Phone Number:	
Address:	
User Name:	
User Type:	customer
Password	

Buttons: Register, Login Page

Register

—

□

×

instaParking

Name:

ks

Email:

k@g.c

Phone Number:

1212121212

Address:

12A

User Name:

ks1

User Type:

customer

▼

Password

•

Register

Login Page

You are Successfully registered!!

Customer HomePage

—

□

×

instaParking

Logout

Choose Location:

Yorkdale

▼

Available Spots: 0

Check Spot:

ydl12

Check Sp...

(Choose any number between 1-50)

Date:

27/04/2023

D

Start Time:

13:50:14

T

Duration (Hrs):

1

...

License Plate#:

1231

Credit Card#:

1345

.....

▼

Price (\$ CAD):

Book Spot

View Bookings

Space Reserved, complete payment to book the spot

Reservation Page

—

□

×

instaParking

Back

ks1 , 13:50:14 , 1hrs , ydl12 , \$2.0

Parking Spot:

ydl12

Cancel Booking

Booking cancelled