



## Title Of The Project:

# **PARKING MANAGEMENT SYSTEM**

Course Code:CSE310

Submitted To:Amarinder Kaur mam

Submitted By:

Student Name	Reg.no.
A.Sairam	12112076
K.Rushyendra	12103028
Y.Sivindra	12107557

# Abstract

This report introduces the process of creating part of a Parking management system which is a data-driven website used by the instructors and students. This website has four major components: registration, login, parking policy, and parking booking. The said 4-parts of the group components and minor parts are implemented by Sairam-12112076, Rushyendra-12103028 and Sivindra-12107557 and the implementation details will be introduced in the report. These implementation are done using Java which is excellent for complex data-driven development. Part of this report will introduce how to use sql to create a database table, user interface and inner logic to handle user request by going through the group component implementation process.

# Objectives

- We can park our vehicle in our own slot by paying
- Because of that there is no towing problems.
- And our vehicle  
has been parked as a secure condition.
- There is no risk for vehicle owner for parking the car
- In case of any damages and problem of vehicle that  
will claim by parking management.
- As the world is facing many threads daily,  
robberies are done easily with no track to trace, bomb  
blasts occur with the use of vehicle, so if a proper  
system is adopted each and every record can be saved  
and anyone can be track easily therefore mainly is to  
make a better and fast software, most important user-  
friendly
- Maintain records in short time of period.
- Determines the parking area is full or not.
- Enhances the visitor's experience.

# Introduction

Parking management system for managing the records of the incoming and outgoing vehicles in an parking houseIt's an easy for Admin to retrieve the data if the vehicle has been visited through number he can get that data.Now days in many public places such as malls, multiplex system, hospitals, offices, market areas there is a crucial problem of vehicle parking. The vehicle parking area has many lanes/slots for car parking. So to park a vehicle one has to look for all the lanes.Moreover this involves a lot of manual labour and investment. Instead of vehicle caught in towing the vehicle can park on safe and security with low cost.Parking control system has been generated in such a way that it is filled with many secure devices such as, parking control gates, toll gates, time and attendance machine, car counting system etc. These features are hereby very necessary nowadays to secure your car and also to evaluate the fee structure for every vehicles entry and exitThe objective of this project is to build a Vehicle Parking management system that enables the time management and control of vehicles using number plate recognition.The system that will track the entry and exit of cars, maintain a listing of cars within the parking lot, and determine if the parking lot is full or not. It will determine the cost of per vehicle according to their time consumption.

# GANTT CHART

Task Description	Duration	Start Date	End Date	Responsible
Phase 1: Project Planning	1 week	22 Mar 2023	28 Mar 2023	All
Determine project requirements	3 days	22 Mar 2023	24 Mar 2023	All
Assign tasks and deadlines	1 day	25 Mar 2023	25 Mar 2023	All
Develop Gantt chart	1 day	26 Mar 2023	26 Mar 2023	All
Review and finalize plan	2 days	27 Mar 2023	28 Mar 2023	All
Phase 2: Report Writing	2 weeks	29 Mar 2023	11 Apr 2023	A.sairam
Research and gather information	5 days	29 Mar 2023	04 Apr 2023	A.sairam
Write introduction and scope	2 days	05 Apr 2023	06 Apr 2023	A.sairam
Write requirements and design	3 days	07 Apr 2023	09 Apr 2023	K.rushyendra
Write implementation details	4 days	10 Apr 2023	13 Apr 2023	Y.sivindra
Write conclusion and review	2 days	14 Apr 2023	15 Apr 2023	A.sairam
Phase 3: Coding and Testing	1 week	12 Apr 2023	16 Apr 2023	All
Develop <code>add_vehicle</code> method	1 day	12 Apr 2023	12 Apr 2023	A.sairam
Develop <code>remove_vehicle</code> method	1 day	13 Apr 2023	13 Apr 2023	K.rushyendra
Develop <code>get_available_slots</code> method	1 day	14 Apr 2023	14 Apr 2023	Y.sivindra
Develop <code>get_parked_vehicles</code> method	1 day	15 Apr 2023	15 Apr 2023	A.sairam
Test and debug code	2 days	16 Apr 2023	17 Apr 2023	All
Prepare final presentation	1 day	18 Apr 2023	18 Apr 2023	All

# Modules

1. `java.util.ArrayList`: This module is used to create an ArrayList that can hold a dynamic number of elements of a specific data type. In this code, two ArrayLists are created: `parkedCars` and `parkedBikes` to store the license plate numbers of the parked vehicles.
2. `java.util.Scanner`: This module is used to create a Scanner object that can read user input from the console. In this code, Scanner is used to ask the user to select a block, select an action, and input the license plate number and duration of parking.
3. `java.lang.System`: This module provides access to the standard input, output, and error streams. In this code, it's used to print messages to the console.
4. `java.lang.Math`: This module provides a set of methods that perform common mathematical operations such as rounding, trigonometric functions, and exponential functions. In this code, Math is not used.
5. `java.lang.String`: This module is used to represent a sequence of characters. In this code, String is used to store the license plate numbers of the parked vehicles.

6. `java.lang.Integer`: This module is used to perform operations on integers. In this code, Integer is used to validate the block choice entered by the user.
7. `java.lang.Double`: This module is used to perform operations on double-precision floating-point numbers. In this code, Double is used to calculate the cost of parking based on the duration of parking and the rate per hour.

The ParkingSystem class has the following attributes:

- `totalSlots`: an `int` representing the total number of parking slots available in both blocks combined.
- `availableSlots`: an `int` representing the current number of available parking slots in both blocks combined.
- `parkedCars`: an `ArrayList` of `String` representing the license plate numbers of all cars currently parked in the system.
- `parkedBikes`: an `ArrayList` of `String` representing the license plate numbers of all bikes currently parked in the system.
- `availableSlotsBlock29`: an `int` representing the current number of available parking slots in block 29.

- `availableSlotsBlock34`: an `int` representing the current number of available parking slots in block 34.

### The ParkingSystem class also has the following methods:

1. `main()`:
2. The main method is the entry point of the program, where the execution starts. It displays a welcome message and asks the user to select a block. After validating the block choice, it shows the available parking slots for the selected block. Then, it presents a menu of options for the user to choose from. Based on the user's input, it calls other methods.
3. `parkCar(int blockChoice)`:
4. This method is called when the user chooses to park a car. It takes the block choice as a parameter to determine which block the car is parked in. First, it checks if there are any available parking slots in the selected block. If there are none, it displays a message and returns. Otherwise, it prompts the user to enter the license plate number and the duration of parking in hours. Then, it calculates the cost of

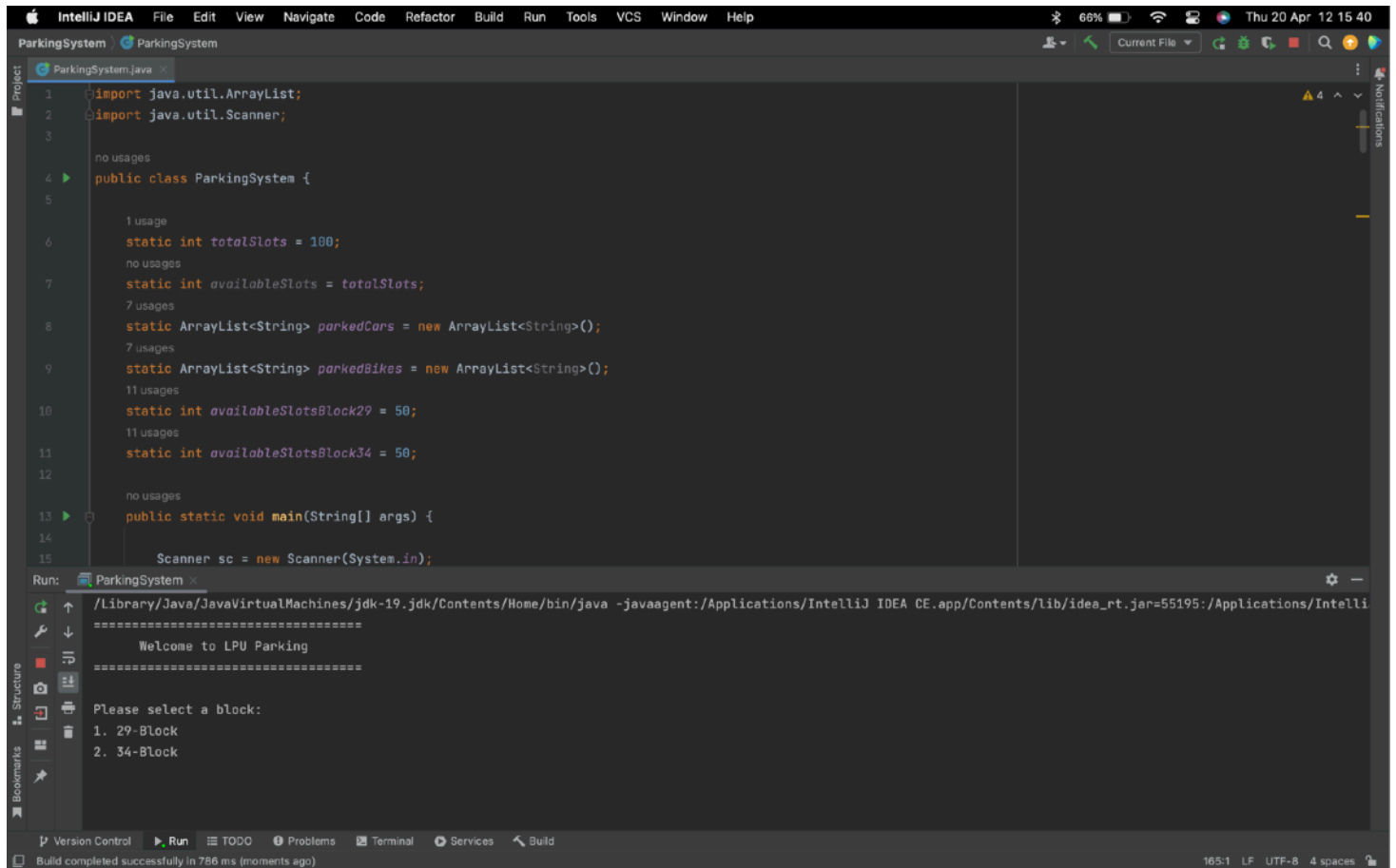


parking based on a rate per hour and adds the car's information to the parkedCars array list. Finally, it reduces the number of available slots in the selected block and displays a success message with the cost of parking.

5. `parkBike(int blockChoice):`
6. This method is called when the user chooses to park a bike. It takes the block choice as a parameter to determine which block the bike is parked in. The logic is similar to the `parkCar` method, except it adds the bike's information to the parkedBikes array list instead of the parkedCars array list.
7. `removeVehicle():`
8. This method is called when the user chooses to remove a vehicle. It prompts the user to enter the license plate number of the vehicle to be removed. If the vehicle is found in either the parkedCars or parkedBikes array list, it removes it and increases the number of available slots in the corresponding block. Otherwise, it displays a message that the vehicle was not found.
9. `viewParkedVehicles():`
10. This method is called when the user chooses to view the parked vehicles. It simply displays the contents of the parkedCars and parkedBikes array lists.

Overall, these methods work together to provide a simple parking system that allows users to park and remove vehicles, view parked vehicles, and calculate the cost of parking.

# Code OutPut:



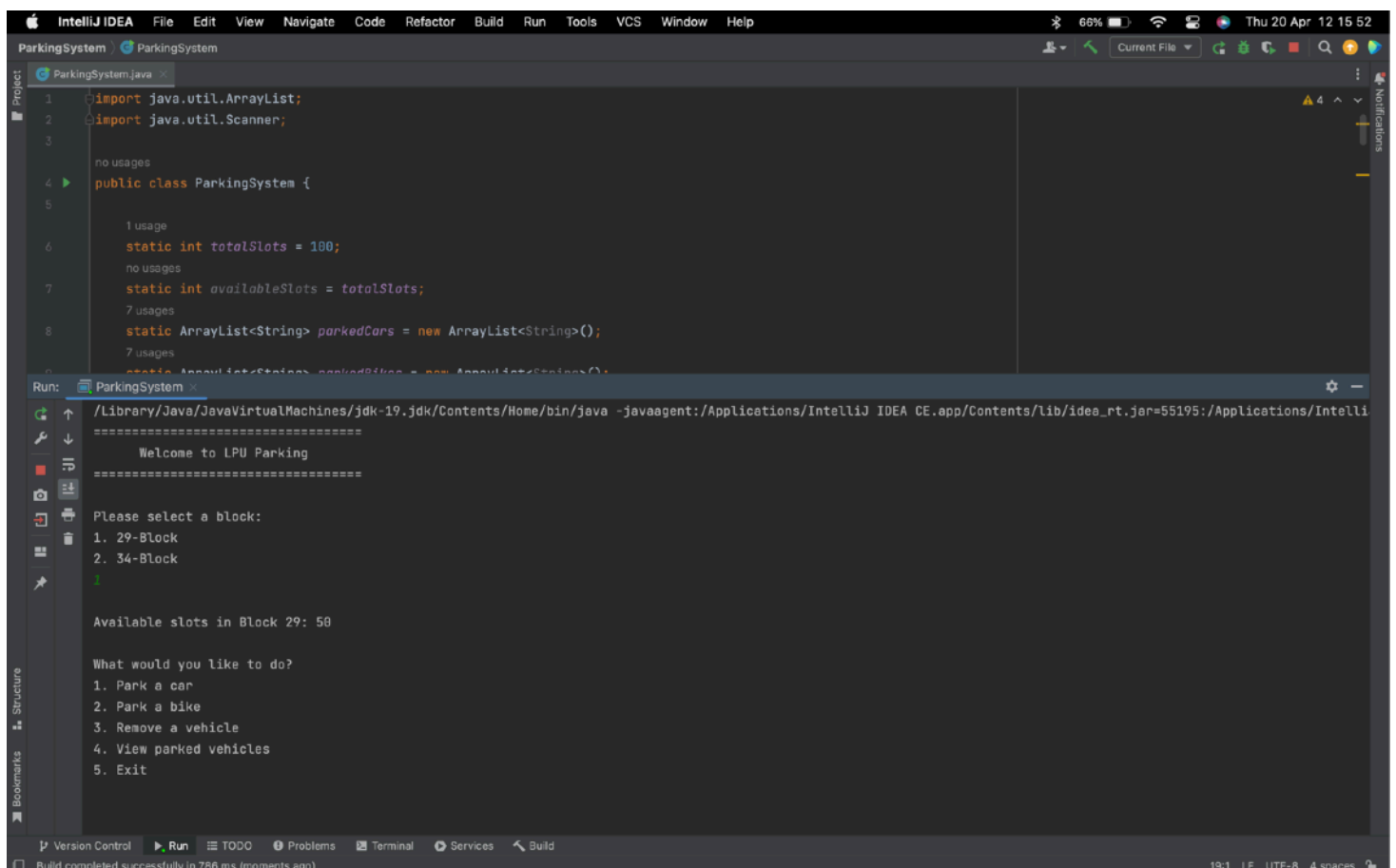
The screenshot shows the IntelliJ IDEA IDE with the file `ParkingSystem.java` open. The code defines a `ParkingSystem` class with static variables for total slots, available slots, and parked vehicles, along with a `main` method that prompts the user to select a block.

```
1 import java.util.ArrayList;
2 import java.util.Scanner;
3
4 public class ParkingSystem {
5
6     static int totalSlots = 100;
7     static int availableSlots = totalSlots;
8     static ArrayList<String> parkedCars = new ArrayList<String>();
9     static ArrayList<String> parkedBikes = new ArrayList<String>();
10    static int availableSlotsBlock29 = 50;
11    static int availableSlotsBlock34 = 50;
12
13    public static void main(String[] args) {
14
15        Scanner sc = new Scanner(System.in);
```

The Run window shows the output of the program:

```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=55195:/Applications/IntelliJ
=====
Welcome to LPU Parking
=====
Please select a block:
1. 29-Block
2. 34-Block
```

The status bar at the bottom indicates the build completed successfully in 766 ms.



The screenshot shows the IntelliJ IDEA IDE with the file `ParkingSystem.java` open. The code is the same as in the previous screenshot.

```
1 import java.util.ArrayList;
2 import java.util.Scanner;
3
4 public class ParkingSystem {
5
6     static int totalSlots = 100;
7     static int availableSlots = totalSlots;
8     static ArrayList<String> parkedCars = new ArrayList<String>();
9     static ArrayList<String> parkedBikes = new ArrayList<String>();
10
11    public static void main(String[] args) {
12
13        Scanner sc = new Scanner(System.in);
```

The Run window shows the output of the program after the user has entered '1' to select Block 29:

```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=55195:/Applications/IntelliJ
=====
Welcome to LPU Parking
=====
Please select a block:
1. 29-Block
2. 34-Block
1
Available slots in Block 29: 50

What would you like to do?
1. Park a car
2. Park a bike
3. Remove a vehicle
4. View parked vehicles
5. Exit
```

The status bar at the bottom indicates the build completed successfully in 766 ms.

The screenshot shows the IntelliJ IDEA IDE with the file `ParkingSystem.java` open. The code defines a `ParkingSystem` class with a `Scanner` for user input. It prompts the user to enter the license plate number of a bike and the duration of parking in hours. It then calculates the cost based on a rate of 15 per hour. The code also includes a menu for parking a car, parking a bike, removing a vehicle, viewing parked vehicles, and exiting. The `Run` output shows the program execution: "Car parked successfully. Available slots: 49", "Cost for parking: Rs.50.00", and the menu options. The status bar at the bottom indicates the build completed successfully in 715 ms.

```
103
104 Scanner sc = new Scanner(System.in);
105 System.out.println("Enter the license plate number of the bike:");
106 String licensePlate = sc.next();
107 System.out.println("Enter the duration of parking in hours:");
108 int duration = sc.nextInt();
109
110 // Calculate cost based on a rate per hour
111 double ratePerHour = 15; // example rate
112 double cost = duration * ratePerHour;
113
114 if (blockChoice == 1) {
115     parkedBikes.add("Block 29 - " + licensePlate);
116     availableSlotsBlock29--;
```

Run: ParkingSystem x

Car parked successfully. Available slots: 49  
Cost for parking: Rs.50.00

What would you like to do?  
1. Park a car  
2. Park a bike  
3. Remove a vehicle  
4. View parked vehicles  
5. Exit

Parked cars:  
Block 29 - TS03ed1234  
Parked bikes:

What would you like to do?  
1. Park a car  
2. Park a bike  
3. Remove a vehicle  
4. View parked vehicles  
5. Exit

Build completed successfully in 715 ms (a minute ago)

The screenshot shows the IntelliJ IDEA IDE with the file `ParkingSystem.java` open. The code is the same as in the first screenshot. The `Run` output shows the program execution: "Block 29 - TS03ed1234", "Parked bikes:", and the menu options. The user enters "1" to park a car, and the program prompts for the license plate number of the vehicle to remove. The user enters "TS03ed1234", and the program outputs "Car removed successfully. Available slots: 50". The status bar at the bottom indicates the build completed successfully in 715 ms.

```
103
104 Scanner sc = new Scanner(System.in);
105 System.out.println("Enter the license plate number of the bike:");
106 String licensePlate = sc.next();
107 System.out.println("Enter the duration of parking in hours:");
108 int duration = sc.nextInt();
109
110 // Calculate cost based on a rate per hour
111 double ratePerHour = 15; // example rate
112 double cost = duration * ratePerHour;
113
114 if (blockChoice == 1) {
115     parkedBikes.add("Block 29 - " + licensePlate);
116     availableSlotsBlock29--;
```

Run: ParkingSystem x

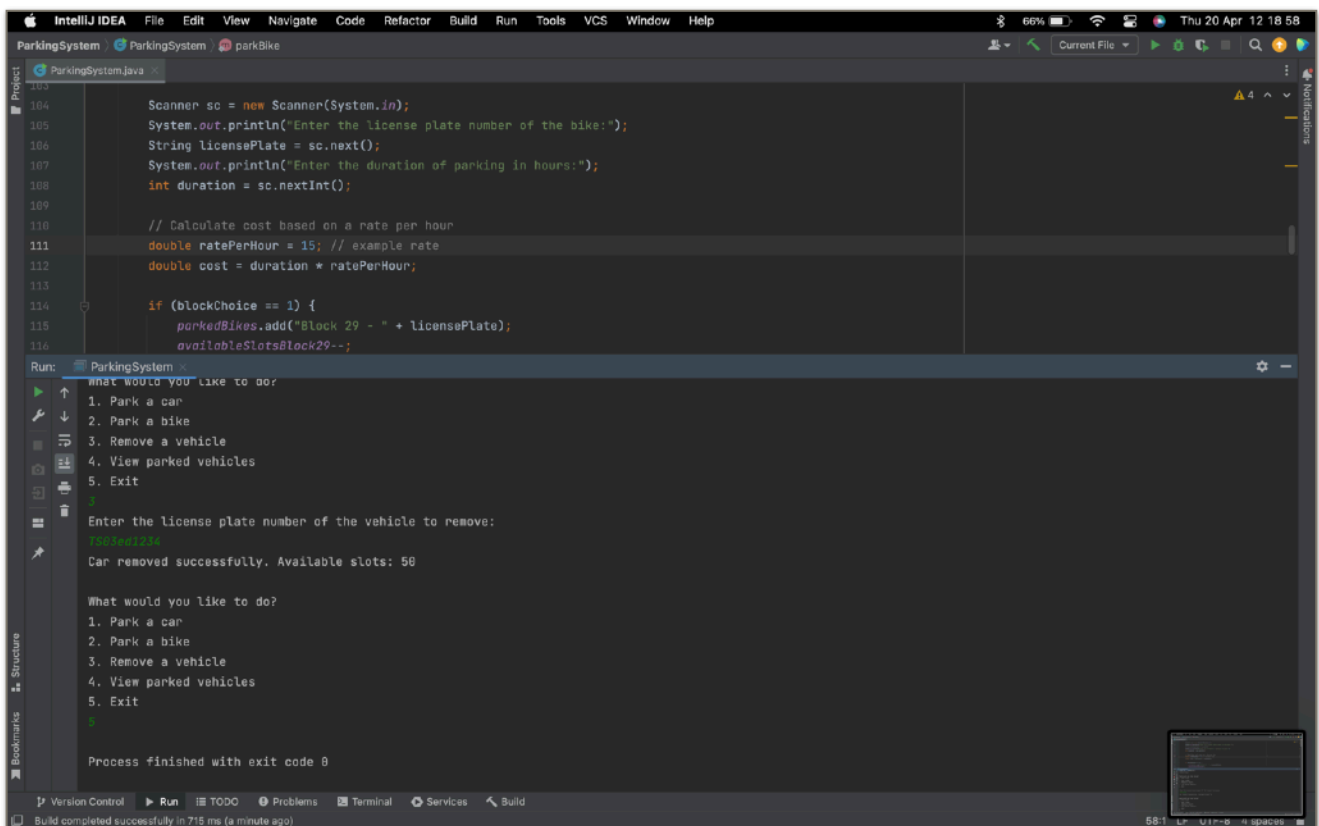
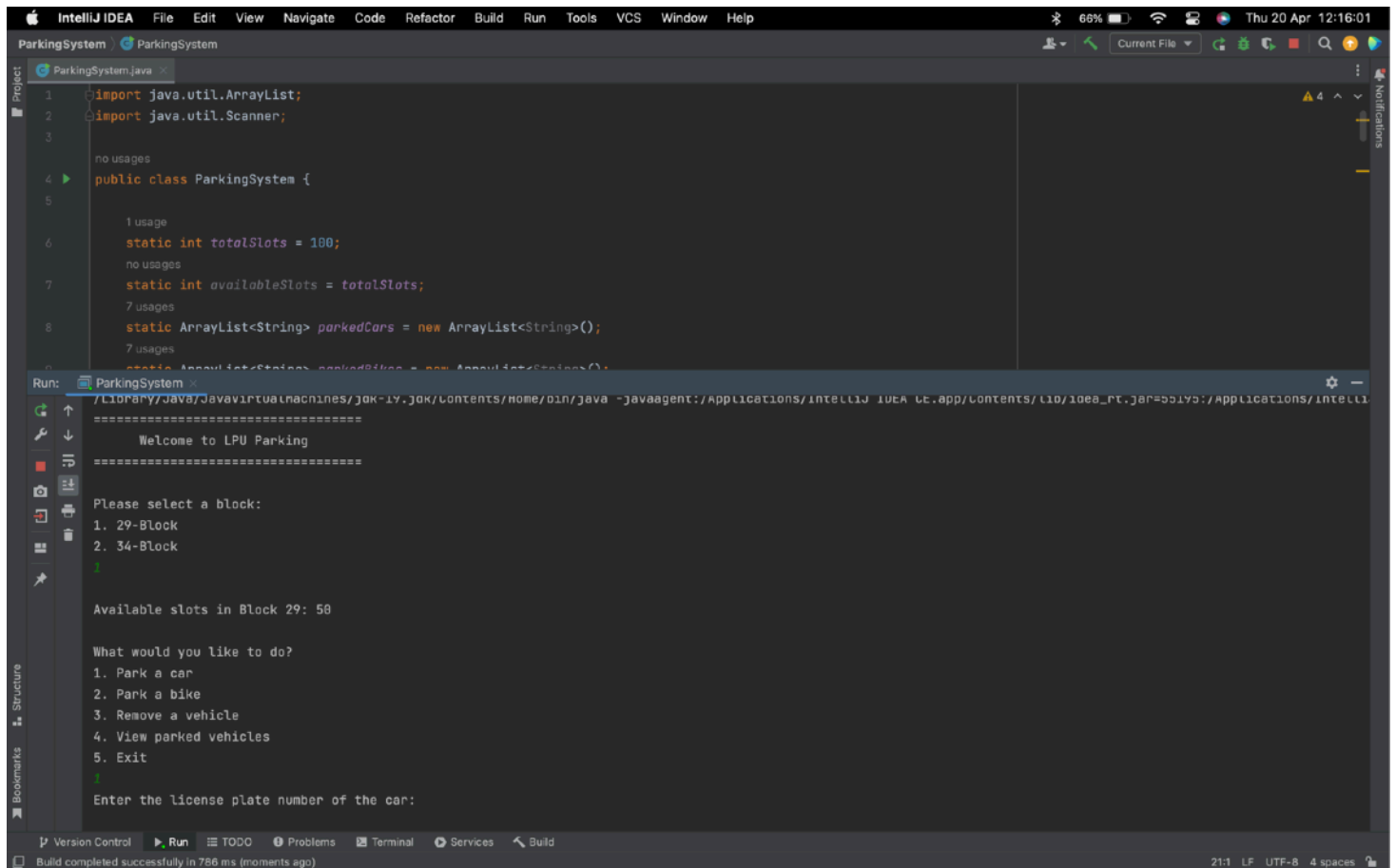
Block 29 - TS03ed1234  
Parked bikes:

What would you like to do?  
1. Park a car  
2. Park a bike  
3. Remove a vehicle  
4. View parked vehicles  
5. Exit

Enter the license plate number of the vehicle to remove:  
TS03ed1234  
Car removed successfully. Available slots: 50

What would you like to do?  
1. Park a car  
2. Park a bike  
3. Remove a vehicle  
4. View parked vehicles  
5. Exit

Build completed successfully in 715 ms (a minute ago)



# Conclusion

This Project is minimizing the task of parking a vehicle by paying and saying some details about customer and vehicle to save data .In this the vehicle is parked as a safe and secure.

The developed application is tested with sample inputs and outputs obtained in according to the requirement. Even though I have tried our level best to make it a dream project. Due to time constraints I could not add more facilities to it.The efficiency of the developed system can be enhanced with some minor modifications. Future development can be made in proposed system by integration more services like:

It can be implemented through web pages.

New effectives modules can be added time to time

# **Bibliography**

[www.google.co.in](http://www.google.co.in)

[www.w3schools.com](http://www.w3schools.com)

[www.youtube.com](http://www.youtube.com)

[www.DocFoc.com](http://www.DocFoc.com)

[www.slideshare.com](http://www.slideshare.com)

[www.codeproject.com](http://www.codeproject.com)

Chatgpt.