PROGRAMMING PROJECT

COURSES CODE & NAME	PROGRAMMING (K_BCS_004)
FACULTY	SCHOOL OF TECHNOLOGY AND ARC
STUDENTS NAME & MATRIC NUMBER	1.NUR ALEESA RIZAL (100007381) 2.CHARVI SHARMA (100006734) 3.DEV CHAUDHARY (100006200) 4.PALAKSHA BORA (100007676) 5.DAKSH PATEL (100007931)
LECTURERS' NAME	PROFESSOR PAUL TANZER PROFESSOR PETER DILLINGER

Parking Management System

Desktop Client + REST Backend

Java Swing • Spring Boot • MySQL • JPA/Hibernate

Project Overview

Search Feature

Admin searches for specific car plates using the search bar. Table filters to show only matching results after clicking Apply.

Filter & Sort

Filter by vehicle type (CAR, BIKE, TRUCK, OTHER). Sort by Exit Time, Entry Time newest/oldest, or Car Plate A-Z / Z-A.

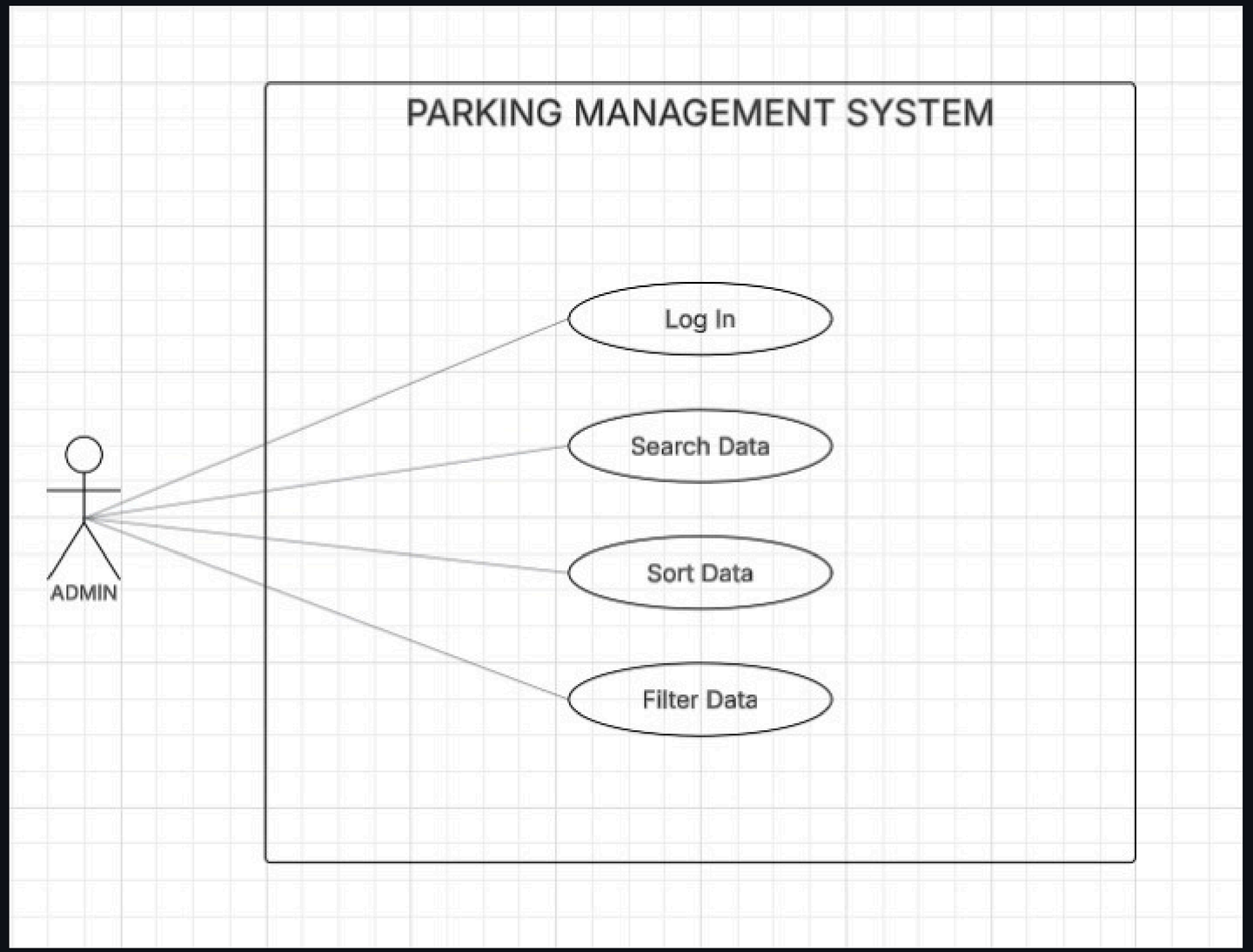
History Data View

Full table of all CLOSED tickets — ID, Plate, Type, Status, Entry Time, Exit Time, Amount. Shows the complete parking record.

Persistent MySQL Storage

All ticket data stored in MySQL database — survives restarts. Amount auto-calculated via a GENERATED column formula.

System Architecture — Use Case Diagram



Log In
Admin must authenticate before accessing the system.

Search Data
Admin searches for specific plate numbers in the table.

Sort Data
Admin sorts by Exit/Entry time or Car Plate A-Z.

Filter Data
Admin filters by vehicle type: CAR, BIKE, TRUCK, OTHER.

Tech Stack

Java + Swing

Programming Language & Desktop UI

Core language. Swing provides JFrame, JTable, JButton, ActionListeners, HttpClient, SwingWorker.

Spring Boot

REST Backend Framework

@RestController, @Service, @Transactional, Maven build tool.

IntelliJ IDEA

Development IDE

Main IDE used to develop the system. Used as backbone — allows development in Java with Spring Boot plugin.

MySQL Workbench

Database

Stores all ticket data. Allows adding, removing, editing and reading parking records.

GitHub — Version control. Holds all members' code contributions in one place.

MySQL — Tickets Table

Column	Type	Key Feature
id	BIGINT	PRIMARY KEY · AUTO_INCREMENT starts at 10001
plate	VARCHAR(20)	License plate — NOT NULL. Stored in uppercase.
vehicleType	ENUM	CAR / BIKE / TRUCK / OTHER — stored as String via @Enumerated
status	ENUM('OPEN','CLOSED'	OPEN when parked · CLOSED when exited
entryTime	DATETIME	DEFAULT CURRENT_TIMESTAMP — auto set on insert
exitTime	DATETIME	DEFAULT NULL · Set when car exits via manualExit()
amount	DECIMAL(10,2)	GENERATED: ceiling(minutes ÷ 60) — read-only, insertable=false
ticketID	VARCHAR(225)	Customer-facing ID e.g. TKT-10001, set by @PrePersist

Spring Boot Backend — Layer Architecture

Controller Layer

`@RestController`

TicketApiController — Receives HTTP GET requests to /api/tickets, builds dynamic Specification, returns JSON list of tickets

1

Service Layer

`@Service`

TicketService — manualEntry() checks if plate already has OPEN ticket. manualExit() finds OPEN ticket and closes it. Both use @Transactional

2

Repository Layer

`@Repository`

TicketRepository extends JpaRepository + JpaSpecificationExecutor. Spring auto-generates SQL — no manual SQL written

3

Entity Layer

`@Entity`

Ticket.java maps to MySQL 'tickets' table. @PrePersist lifecycle hook auto-sets ticketID and default status = OPEN before INSERT

4

Spring Boot — REST API Endpoints

POST `/api/tickets/entry`

Create new ticket — car enters parking

```
{ "plate": "ABC123", "vehicleType": "CAR" }
```

POST `/api/tickets/exit`

Close ticket — car exits, fee auto-calculated by MySQL

```
{ "plate": "ABC123" }
```

GET `/api/tickets?status=OPEN`

Get all currently parked vehicles (Live tab)

```
Returns: [ { id, plate, status,
entryTime... } ]
```

GET `/api/tickets?status=CLOSED&vehicleType=CAR&sort=exitTime,desc`

History with filters + sort (used by ParkingHistoryUI)

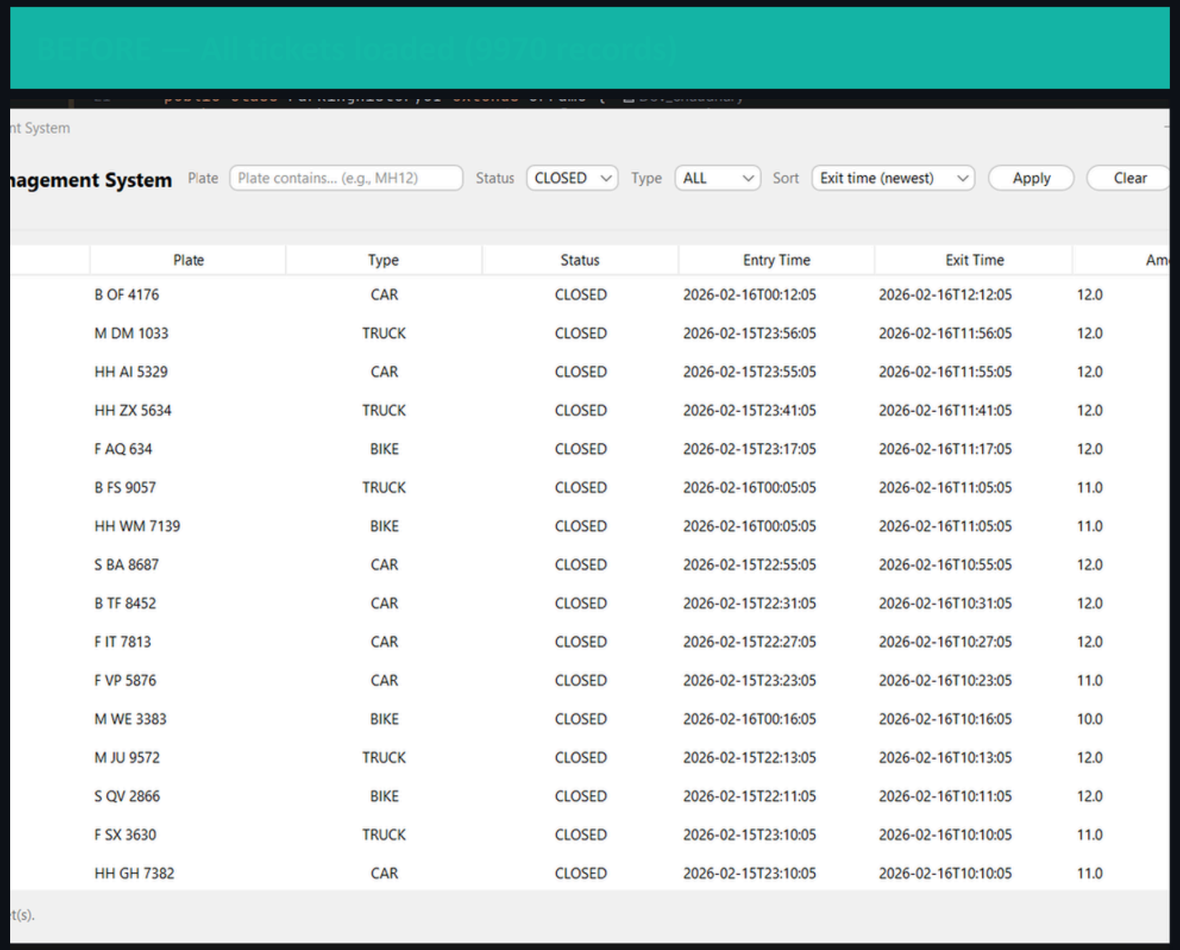
```
Returns: [ { id, plate, exitTime, amount...
} ]
```

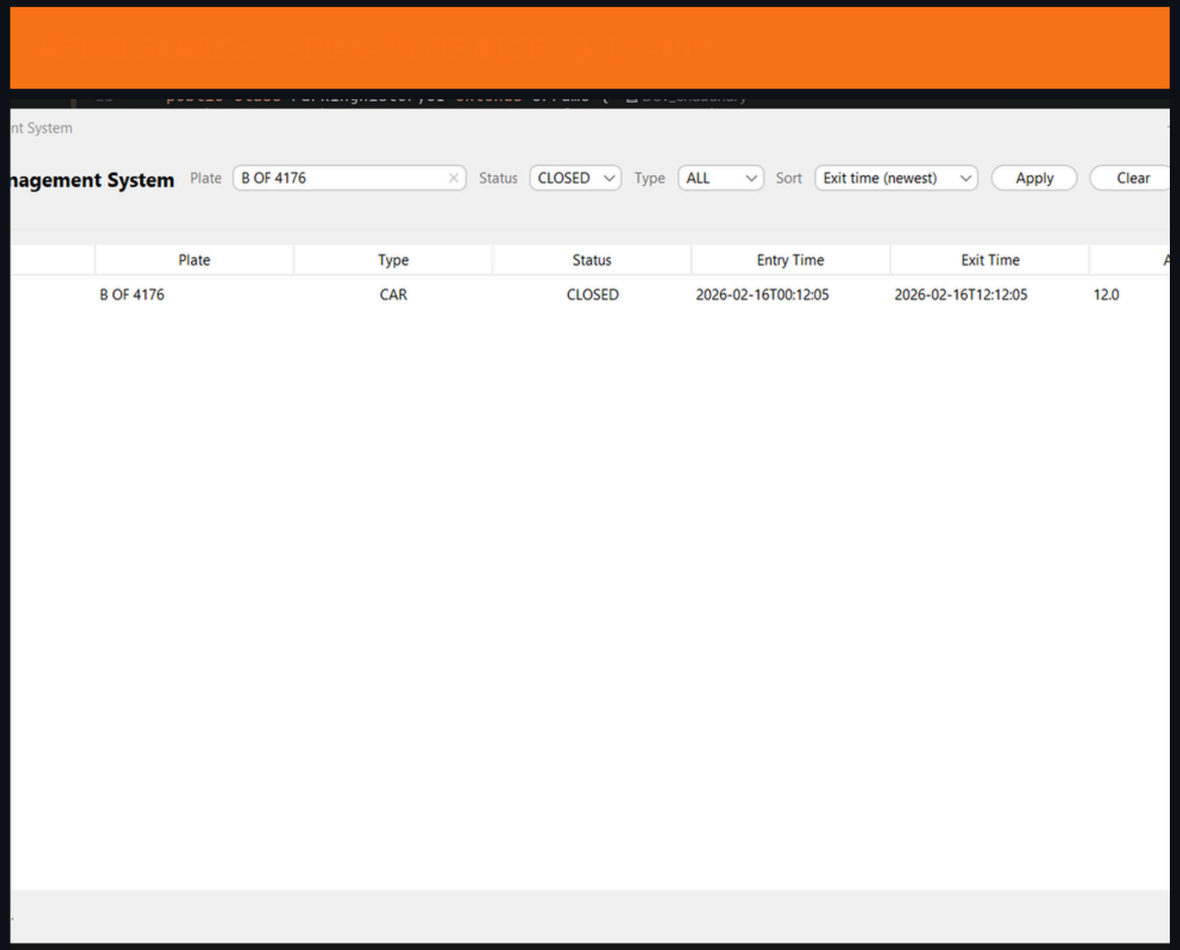
GET `/health`

Server health check — used by Ping button in UI

```
Returns: { "status": "UP" }
```

GUI Highlights — Actual System Screenshots





History section — columns: ID · Plate · Type · Status · Entry Time · Exit Time · Amount | Filter: Status + Type dropdowns | Sort: dropdown

End-to-End Flow — Login to History View

LOGIN FLOW (AdminLoginUI.java)

1	2	3	4	5	6
Admin opens app	Login screen shown	Enter Username and Password	doLogin() runs	Credentials match	Opens ParkingSwingUI

SEARCH / FILTER / SORT FLOW (ParkingHistoryUI.java)

1	2	3	4	5	6
Admin sets filters	refresh() builds URL	SwingWorker starts	GET /api/tickets	JSON returned	Table updates (EDT)

OOP Concepts in This Project

Encapsulation

All fields in Ticket.java are private — accessed only through getters/setters. The backend hides the MySQL database entirely from the Swing frontend.

Inheritance

AdminLoginUI extends JFrame (login window). ParkingSwingUI extends JFrame (dashboard). TicketRepository extends JpaRepository — inherits all CRUD methods.

Polymorphism

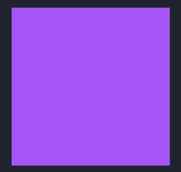
VehicleType enum (CAR, BIKE, TRUCK, OTHER) — each is a distinct polymorphic value. ThrowingSupplier functional interface overrides the standard Supplier behavior.

Abstraction

JpaRepository hides all SQL. HttpClient hides HTTP socket details. runAsync() hides threading complexity. Each layer exposes only what is needed.

Thank You

Parking Management System



Swing Client

Java 11 Desktop UI

Spring Boot

REST API Backend

MySQL + JPA

Persistent Storage

REST / JSON

Client-Server Protocol

Java • Spring Boot • MySQL • JPA/Hibernate • Maven • Git