

Task 1: Implementation guide

1. Scenario analysis – entities, attributes, relationships

Entities:

- **ParkingSite:** SiteID (PK), SiteName, Address, Postcode, Capacity Relationship: One ParkingSite has many ParkingSlots, many Bookings, many Payments.
- **ParkingSlot:** SlotID (PK), SiteID (FK), SlotNumber, Level, SlotType, Status Relationship: Each ParkingSlot belongs to one ParkingSite; one ParkingSlot can have many Bookings over time.
- **Customer:** CustomerID (PK), FirstName, LastName, Phone, Email Relationship: One Customer owns many Vehicles and makes many Bookings and Payments.
- **Vehicle:** VehicleID (PK), CustomerID (FK), RegistrationNumber, Make, Model, Colour Relationship: Each Vehicle belongs to one Customer; one Vehicle can appear in many Bookings.
- **Booking:** BookingID (PK), VehicleID (FK), SlotID (FK), SiteID (FK), StartDateTime, EndDateTime, Status Relationship: Each Booking links one Vehicle to one ParkingSlot (and implicitly one ParkingSite).
- **Payment:** PaymentID (PK), BookingID (FK), PaymentDateTime, Amount, Method, Status Relationship: Each Payment is associated with one Booking; one Booking can have one or more Payments (e.g., partial, failed, retried).

2. Entity-relationship diagram (ERD) – textual description

- ParkingSite (1) — (M) ParkingSlot ParkingSite.SiteID→ParkingSlot.SiteID
- Customer (1) — (M) Vehicle Customer.CustomerID→Vehicle.CustomerID
- Vehicle (1) — (M) Booking Vehicle.VehicleID→Booking.VehicleID
- ParkingSlot (1) — (M) Booking ParkingSlot.SlotID→Booking.SlotID
- ParkingSite (1) — (M) Booking ParkingSite.SiteID→Booking.SiteID
- Booking (1) — (M) Payment Booking.BookingID→Payment.BookingID

Cardinalities: mostly 1–M; no direct M–M because they are resolved via Booking and Payment.

3. Data structure development – table designs and constraints

Parking Site

- Attributes: SiteID (PK, AutoNumber), SiteName (Text), Address (Text), Postcode (Text), Capacity (Number).
- Constraints: SiteName required; Capacity ≥ 0 .

Parking Slot

- Attributes: SlotID (PK, AutoNumber), SiteID (FK), SlotNumber (Text), Level (Text), SlotType (Text: Standard/Disabled/EV), Status (Text: Free/Occupied/OutOfService).
- Constraints: SiteID required; SlotNumber unique per SiteID.

Customer

- Attributes: CustomerID (PK, AutoNumber), FirstName (Text), LastName (Text), Phone (Text), Email (Text).
- Constraints: Email format validation; FirstName, LastName required.

Vehicle

- Attributes: VehicleID (PK, AutoNumber), CustomerID (FK), RegistrationNumber (Text), Make (Text), Model (Text), Colour (Text).
- Constraints: RegistrationNumber unique; CustomerID required.

Booking

- Attributes: BookingID (PK, AutoNumber), VehicleID (FK), SlotID (FK), SiteID (FK), StartDateTime (Date/Time), EndDateTime (Date/Time), Status (Text: Active/Completed/Cancelled).
- Constraints: StartDateTime < EndDateTime; SlotID and VehicleID required; referential integrity enforced.

Payment

- Attributes: PaymentID (PK, AutoNumber), BookingID (FK), PaymentDateTime (Date/Time), Amount (Currency), Method (Text: Card/Cash/App), Status (Text: Paid/Pending/Failed).
- Constraints: Amount > 0; BookingID required.

4. Normalization compliance (to 3NF)

- 1NF: All tables have atomic attributes, no repeating groups (e.g., one slot per row, one vehicle per row).
- 2NF: No partial dependencies—each non-key attribute depends on the full primary key (PKs are single-column AutoNumbers).
- 3NF: No transitive dependencies—e.g., Customer contact details are stored only in Customer, not in Vehicle or Booking; Site details only in ParkingSite; payment details only in Payment.

5. Sample data population (example rows)

Parking Site

Site ID	Site Name	Address	Postcode	Capacity
1	City Centre	10 High St, Birmingham	B1 1AA	200
2	Station Car Park	5 Railway, Birmingham	B2 2BB	150

Parking Slot

SlotID	SiteID	SlotNumber	Level	SlotType	Status
1	1	A01	L1	Standard	Free
2	1	A02	L1	EV	Occupied
3	1	B10	L2	Disabled	Free
4	2	S05	G	Standard	Occupied
5	2	S06	G	Standard	Free

Customer

CustomerID	FirstName	LastName	Phone	Email
1	Sarah	Jones	07123456789	sarah.jones@example.com
2	Ahmed	Khan	07987654321	ahmed.khan@example.com
3	Emily	Brown	07700111222	emily.brown@example.com
4	David	Smith	07555123456	david.smith@example.com
5	Priya	Patel	07444123456	priya.patel@example.com

Vehicle

VehicleID	CustomerID	RegistrationNumber	Make	Model	Colour
1	1	BX12 ABC	Ford	Focus	Blue
2	2	KY65 XYZ	BMW	3 Series	Black
3	3	LM19 PQR	Tesla	Model 3	White
4	4	HG10 TUV	Honda	Civic	Silver
5	5	ZN22 LMN	VW	Golf	Red

Booking

BookingID	VehicleID	SlotID	SiteID	StartDateTime	EndDateTime	Status
1	1	2	1	2026-07-09 08:00	2026-07-09 10:00	Completed
2	2	4	2	2026-07-09 09:30	2026-07-09 12:00	Active
3	3	1	1	2026-07-09 07:45	2026-07-09 09:00	Completed
4	4	3	1	2026-07-09 11:00	2026-07-09 13:00	Active
5	5	5	2	2026-07-09 10:15	2026-07-09 11:15	Completed

Payment

Payment ID	Booking ID	Payment Date Time	Amount	Method	Status
1	1	2026-07-09 10:05	6.00	Card	Paid
2	2	2026-07-09 09:35	9.00	App	Paid
3	3	2026-07-09 09:05	5.00	Cash	Paid
4	4	2026-07-09 11:05	7.50	Card	Pending
5	5	2026-07-09 11:20	4.50	App	Paid

Query development – example queries

Parking slots currently in use

```
SELECT ParkingSlot.SlotNumber, ParkingSite.SiteName, Booking.StartDateTime,  
Booking.EndDateTime  
FROM Booking
```

INNER JOIN ParkingSlot ON Booking.SlotID = ParkingSlot.SlotID

INNER JOIN ParkingSite ON Booking.SiteID = ParkingSite.SiteID

WHERE Booking.Status = 'Active';

Vehicles parked at a particular time

SELECT Vehicle.RegistrationNumber, Customer.FirstName, Customer.LastName,
ParkingSlot.SlotNumber, ParkingSite.SiteName

FROM Booking

INNER JOIN Vehicle ON Booking.VehicleID = Vehicle.VehicleID

INNER JOIN Customer ON Vehicle.CustomerID = Customer.CustomerID

INNER JOIN ParkingSlot ON Booking.SlotID = ParkingSlot.SlotID

INNER JOIN ParkingSite ON Booking.SiteID = ParkingSite.SiteID

WHERE #2026-07-09 09:30# BETWEEN Booking.StartDateTime AND Booking.EndDateTime;

Details of the vehicle parked in a particular slot

SELECT ParkingSlot.SlotNumber, Vehicle.RegistrationNumber, Vehicle.Make, Vehicle.Model,
Customer.FirstName, Customer.LastName

FROM Booking

INNER JOIN ParkingSlot ON Booking.SlotID = ParkingSlot.SlotID

INNER JOIN Vehicle ON Booking.VehicleID = Vehicle.VehicleID

INNER JOIN Customer ON Vehicle.CustomerID = Customer.CustomerID

WHERE ParkingSlot.SlotNumber = 'A02' AND Booking.Status = 'Active';

7. Validation and testing – brief outline

- **Functional checks:**

- **Slot allocation:** Create bookings and confirm slot status changes from Free to Occupied.
- **Referential integrity:** Attempt to delete a Customer with existing Vehicles/Bookings—ensure Access prevents it.
- **Time logic:** Test invalid bookings (EndDateTime before StartDateTime) and confirm validation rules block them.
- **Reports/queries:** Run queries to verify correct counts, occupancy, and revenue totals.

- **Outcomes:**

- Data entry errors reduced via required fields and validation rules.
- Slot usage visible in real time through queries and forms.
- Payments correctly linked to bookings, enabling revenue reporting.

Table of contents:

1. Introduction
 - 1.1 Scenario overview
 - 1.2 Purpose and scope
2. System analysis and requirements
 - 2.1 Business context
 - 2.2 Functional requirements
3. Database design
 - 3.1 Entity identification
 - 3.2 ERD description
 - 3.3 Normalization to 3NF
4. Implementation in Microsoft Access
 - 4.1 Table creation and relationships
 - 4.2 Constraints and validation rules
 - 4.3 Sample data population
5. Queries and reporting
 - 5.1 Operational queries
 - 5.2 Management reports (occupancy, revenue)
6. Testing and evaluation
 - 6.1 Test plan
 - 6.2 Test results and discussion
7. Reflection and future improvements
 - 7.1 Design decisions
 - 7.2 Challenges faced
 - 7.3 Potential enhancements
8. Conclusion
9. References (IEEE format)

Scenario explanation (short):

- **Purpose:** To replace manual recording of vehicle entries, slot usage, customer details, and payments with a centralized Access database that supports real-time monitoring and reporting.

- **Context:** AutoPark operates multiple sites in a city; manual processes cause errors, lost data, and poor utilization. The database focuses on core operational entities (sites, slots, customers, vehicles, bookings, payments) and supports queries for occupancy and revenue.

Screenshots (to include in your actual document):

- **Tables:** Design view and datasheet view for each table.
- **Relationships:** Access Relationships window showing PK–FK links.
- **Queries:** Query design and results for key queries.
- **Forms/Reports (if built):** Main booking form, occupancy report.

Short design report and reflection (key points):

- **Design decisions:**
 - Chose Booking as the central transactional entity to link vehicles, slots, and sites.
 - Kept Payment separate to allow multiple payments per booking and flexible methods.
 - Normalized customer and vehicle data to avoid duplication.
- **Challenges:**
 - Ensuring time-based queries correctly handle overlapping bookings.
 - Designing validation rules that are strict but not overly restrictive for real-world data.
- **Future improvements:**
 - Add tariff tables for dynamic pricing by time of day or site.
 - Implement automated alerts for over-capacity or long-stay vehicles.
 - Add user roles and audit logging.

References (IEEE style – examples):

- **[1]** R. Elmasri and S. Navathe, *Fundamentals of Database Systems*, 7th ed. Pearson, 2016.
- **[2]** C. J. Date, *An Introduction to Database Systems*, 8th ed. Addison-Wesley, 2003.
- **[3]** Microsoft, “Designing a desktop database,” Access documentation, 2024.
- **[4]** A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*, 7th ed. McGraw-Hill, 2019.