**MongoDB Data Modelling: Airline Management System**

**1. What Is Data Modelling?**

The Airline Management System is designed to manage airline operations, including flights, passengers, bookings, and crew assignments.  
The system uses MongoDB to store and manage semi-structured data, providing flexibility for operational queries and analytics such as flight occupancy, revenue reports, and passenger booking history.

**2. Identify Collections**

| **Collection** | **Purpose** |
| --- | --- |
| flights | Stores details about flights, aircraft, timings, seat availability, and pricing. |
| passengers | Stores passenger profiles, passport details, loyalty points, and booking history. |
| bookings | Stores all reservations, seat allocation, payment, and check-in status. |
| crew | Stores crew information, their roles, assigned flights, and availability. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Relationship | Type | Description | Fi  eld Reference | Modeling Approach | Justification |
| 1 | Flight ⟷ Booking | One-to-Many | A flight can have many bookings | bookings.flight\_id → flights.\_id | Reference | Bookings are transactional and should remain independent of flights. |
| 2 | Passenger ⟷ Booking | One-to-Many | A passenger can book multiple flights | bookings.passenger\_id → passengers.\_id | Reference | Allows fetching all bookings for a passenger easily. |
| 3 | Passenger ⟷ Loyalty Points | One-to-One | Each passenger has a single loyalty point record | passengers.loyalty\_points | Embed | Frequent access with passenger profile, embedding is optimal. |
| 4 | Flight ⟷ Crew | Many-to-Many | Multiple crew members serve multiple flights | crew.assigned\_flights[] → flights.\_id | Reference | Crew assignment is dynamic; referencing allows easy updates. |
| 5 | Booking ⟷ Check-in Status | One-to-One | Each booking has a single check-in status | bookings.check\_in\_status | Embed | Always accessed with booking document. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Relationship | Type | Description | Field Reference | Modeling Approach | Justification |
| 1 | Flight ⟷ Booking | One-to-Many | A flight can have many bookings | bookings.flight\_id → flights.\_id | Reference | Bookings are transactional and should remain independent of flights. |
| 2 | Passenger ⟷ Booking | One-to-Many | A passenger can book multiple flights | bookings.passenger\_id → passengers.\_id | Reference | Allows fetching all bookings for a passenger easily. |
| 3 | Passenger ⟷ Loyalty Points | One-to-One | Each passenger has a single loyalty point record | passengers.loyalty\_points | Embed | Frequent access with passenger profile, embedding is optimal. |
| 4 | Flight ⟷ Crew | Many-to-Many | Multiple crew members serve multiple flights | crew.assigned\_flights[] → flights.\_id | Reference | Crew assignment is dynamic; referencing allows easy updates. |
| 5 | Booking ⟷ Check-in Status | One-to-One | Each booking has a single check-in status | bookings.check\_in\_status | Embed | Always accessed with booking document. |

| **Use Case** | **Recommended Modeling** |
| --- | --- |
| Store aircraft details in flights | Embed inside flights (fast access) |
| Link bookings to passengers | Reference (separate collection) |
| Store crew assignments to flights | Reference via assigned\_flights[] |
| Store loyalty points with passenger | Embed inside passenger document |

FLIGHTS

{

"\_id": "FL100",

"airline": "Air India",

"source": "DEL",

"destination": "BOM",

"departure\_time": "2025-08-01T09:30:00",

"arrival\_time": "2025-08-01T11:45:00",

"status": "Scheduled",

"aircraft": {

"aircraft\_id": "A320-IND100",

"type": "Airbus A320",

"total\_seats": 180

},

"available\_seats": 120,

"price": 4500

}

PASSENGERS

{

"\_id": "PAX001",

"name": "John Doe",

"email": "john@example.com",

"phone": "+91-9876543210",

"passport\_number": "M1234567",

"loyalty\_points": 1200,

"booking\_history": ["BKG1001","BKG1005"]

}

BOOKINGS

{

"\_id": "BKG1001",

"flight\_id": "FL100",

"passenger\_id": "PAX001",

"booking\_date": "2025-07-30T14:20:00",

"seat\_number": "12A",

"price\_paid": 4500,

"payment\_status": "Paid",

"check\_in\_status": "Not Checked-In"

}

CREW

**6. Justification of Design**

1. Separation of concerns: Bookings are separate to avoid embedding large arrays in flights or passengers.
2. Efficient queries: References + $lookup enable joining passengers with flights when needed.
3. Fast access to frequently used data: Embedding aircraft info and loyalty points reduces joins.
4. Scalable: MongoDB’s flexible schema supports adding more fields like meal preferences, baggage info, or crew certifications without migrations.