# Flight Booking Engine - System Documentation

## 1. Architectural Overview

This system is designed as a **BFF (Backend for Frontend) architecture**. It acts as a middleware layer that simplifies the complex, standardized interactions required by Global Distribution Systems (GDS) and aggregators like Duffel, exposing a clean RESTful API to the frontend client.

### Core Components

#### Frontend Client (Streamlit)

- **Role:** User Interface for search, selection, and data entry
- **Responsibility:**
  - State management (session_state)
  - Input validation
  - Rendering progress steps
  - Holds no business logic

#### Backend API (Middleware)

- **Role:** The "Brain" of the operation
- **Responsibility:**
  - **Orchestration:** Chaining multiple upstream API calls (e.g., getting an order, then paying, then fetching documents)
  - **Sanitization:** Stripping sensitive upstream data before sending it to the client
  - **Session Caching:** Temporarily storing offer_request_id or offer_id mappings if a database is connected

#### Upstream Provider (Duffel/GDS)

- **Role:** The inventory source
- **Responsibility:** Real-time communication with airlines (British Airways, Lufthansa, etc.) for availability and ticketing

---

# 2. API Reference & Endpoints

**Base URL:** `http://localhost:8000/api`

**Authentication:** Bearer Token (Recommended for production)

## Group A: Search & Discovery

#### 1. Search Flights (Create Offer Request)

Initializes a search session. This is an asynchronous-like operation; it returns an ID immediately, but offers are populated in the background.

**Endpoint:** `POST /offer-requests`

**Status:** `201 Created`

**Request Body Schema:**

| Field | Type | Required | Description |
|---|---|---|---|
| slices | Array | Yes | List of journey segments (Origin/Dest/Date) |
| passengers | Array | Yes | List of passenger types (adult, child, infant) |
| cabin_class | String | No | economy, premium_economy, business, first |
| max_connections | Int | No | Filter limit (e.g., 0 for direct flights) |

**Response Example:**


```
{
  "data": {
    "id": "orq_0000abc...",
    "live_mode": false,
    "slices": [
      {
        "origin": {
          "name": "John F. Kennedy",
          "iata_code": "JFK"
        },
        "destination": {
          "name": "Heathrow",
          "iata_code": "LHR"
        }
```

```
        }
      ],
      "created_at": "2025-08-19T10:00:00Z"
    }
  }
```

### 2. List Offers

Retrieves the flight options found for a specific request.

**Endpoint:** `GET /offers`

**Query Parameters:**

- `offer_request_id` (Required): The ID from step 1
- `sort` : `total_amount` (default) or `total_duration`
- `max_price` : Filter results by budget

**Detailed Response Item:**

```
{
  "id": "off_0000xyz...",
  "owner": {
    "iata_code": "BA",
    "name": "British Airways"
  },
  "total_amount": "450.00",
  "total_currency": "USD",
  "payment_requirements": {
    "requires_instant_payment": false,
    "payment_required_by": "2025-08-20T12:00:00Z"
  },
  "slices": [
    {
      "duration": "P0Y0M0DT7H0M0S",
      "segments": [
        {
          "operating_carrier": {
            "iata_code": "BA",
            "name": "British Airways"
          },
          "marketing_carrier": {
            "iata_code": "AA",
            "name": "American Airlines"
```

```
            },
            "aircraft": {
              "name": "Boeing 777-200"
            }
          }
        ]
      }
    ]
  }
```

## Group B: Booking & Orders

### 3. Create Order

The critical transactional step. Reserves inventory.

**Endpoint:** `POST /orders`

**Status:** `201 Created`

**Key Logic:**

- **Hold:** If `type=hold`, the seat is reserved but not ticketed. PNR is generated.
- **Instant:** If `type=instant`, payments array MUST be present. PNR is generated and ticketed immediately.

**Request Body (Instant Scenario):**

```
{
  "type": "instant",
  "selected_offers": ["off_0000xyz..."],
  "passengers": [
    {
      "type": "adult",
      "given_name": "Amelia",
      "family_name": "Earhart",
      "born_on": "1987-07-24",
      "title": "ms",
      "gender": "f",
      "email": "amelia@test.com",
      "phone_number": "+15550109999"
    }
  ],
```

```
  "payments": [
    {
      "type": "balance",
      "amount": "450.00",
      "currency": "USD"
    }
  ]
}
```

#### 4. Retrieve Order (Polling)

Fetches the latest status of an order. Used to check if documents (tickets) have been issued after payment.

**Endpoint:** `GET /orders/{id}`

**Response Fields of Note:**

- `synced_at` : Timestamp of last sync with airline
- `documents` : Array of ticket objects

## Group C: Payments & Fulfillment

#### 5. Create Payment

Captures funds for a "Hold" order.

**Endpoint:** `POST /payments`

**Status:** `201 Created`

**Error Handling:**

- `409 Conflict` : Order is already paid
- `422 Unprocessable Entity` : Payment amount does not match order total

## Group D: Expanded Functionality (Production Ready)

These endpoints are not in basic scripts but are standard for a complete system.

#### 6. Cancel Order

Releases the inventory and attempts a refund if applicable.

**Endpoint:** `POST /orders/{id}/actions/cancel`

**Request Body:**

```
{ "reason": "requested_by_customer" }
```

**Response:** Returns a refund quote or confirmation of cancellation.

#### 7. Seat Maps

Fetches the cabin layout for seat selection.

**Endpoint:** `GET /offers/{id}/seat-maps`

**Response:**

```
{
  "data": [
    {
      "cabin_class": "economy",
      "rows": [
        {
          "sections": [
            {
              "elements": [
                {
                  "type": "seat",
                  "designator": "12A",
                  "available_services": [...]
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}
```

---

## 3. Data Models & Schemas

### The Passenger Object

Used in both requests and responses.

| Field | Type | Description |
|---|---|---|
| id | String | Unique ID used to link a passenger to a specific seat/service |
| type | String | adult, child, or infant_without_seat |
| given_name | String | First name as it appears on passport |
| family_name | String | Last name |
| born_on | String | ISO Date YYYY-MM-DD |
| title | String | mr, ms, mrs, miss, dr |
| gender | String | m or f |

### The Document Object

Represents the actual travel authority (Ticket).

| Field | Type | Description |
|---|---|---|
| unique_identifier | String | The generic airline ticket number (e.g., 125-99887766) |
| type | String | Usually electronic_ticket |
| passenger_id | String | Links this ticket to a specific passenger in the order |

---

## 4. Error Handling Standards

The API uses standard HTTP status codes to indicate success or failure.

| Code | Meaning | Logic / User Action |
|---|---|---|
| 200 | OK | Request succeeded |
| 201 | Created | Resource (Order/Payment) created successfully |
| 400 | Bad Request | Malformed JSON or invalid parameters (e.g., past date) |
| 402 | Payment Required | The card failed or insufficient balance |
| 404 | Not Found | The offer_id or order_id is invalid or expired |
| 409 | Conflict | The flight sold out between Step 2 and Step 3 (Race Condition) |

| Code | Meaning | Logic / User Action |
|---|---|---|
| 500 | Internal Error | Connection to GDS/Duffel failed. Retry recommended |

---

# 5. Workflow Sequence

The complete flight booking workflow follows these sequential steps:

1. **User initiates search** → API returns `request_id`
2. **API polls airlines** → Caches results
3. **User requests offers** → API returns list from cache
4. **User selects flight** → API validates availability
5. **User submits Passenger Info** → API creates Order (Status: hold)
6. **User submits Payment** → API validates amount → Calls Airline → Updates Order to paid
7. **Airline issues Ticket** → API receives webhook/poll → User sees Ticket Number

---

# Document Information

- **Generated:** December 18, 2025
- **System:** Flight Booking Engine (BFF Architecture)
- **API Version:** RESTful with Bearer Token Authentication
- **Base URL:** http://localhost:8000/api