🎟️ Ticket Management System API

📝 Objective

Create a comprehensive Ticket Management System API with the following functionalities:

- **Create a user**
- **Authentication**
- **Create a ticket**
- **Assign users to a ticket**
- **Get ticket details including assigned users and status**
- **Ticket history**
- **Ticket analytics**

📋 Requirements

1. Create a User

- **User Types:**
  - "customers"
  - "admin"
- **Endpoint:** `POST /users`
- **Request Body:**

```
{
  "name": "User Name",
  "email": "user@example.com",
  "type": "customer",  // type can only be "customer" or "admin"
  "password": "password123"  // Password should be hashe
```

```
d before storing
}
```

- **Constraints:**
  - The **email** must be unique.
  - The **password** should meet complexity requirements (e.g., minimum length, at least one special character).
  - An **admin** cannot be a **customer** and a **customer** cannot be an **admin**.
- **Response:**

```
{
  "id": "user_id",
  "name": "User Name",
  "email": "user@example.com"
}
```

---

2. Authentication

- **Endpoint:** `POST /auth/login`
- **Request Body:**

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

- **Response:**

```
{
  "token": "jwt_token"
}
```

- **Constraints:**
  - Use **JSON Web Tokens (JWT)** for authentication.
  - Validate user credentials and issue a **JWT token** on successful login.

---

3. Create a Ticket

- **Endpoint:** `POST /ticket`
- **Request Headers:**

```
Authorization: Bearer jwt_token
```

- **Request Body:**

```
{
  "title": "Ticket Title",
  "description": "Ticket Description",
  "type": "concert",  // type can be "concert", "confere
nce", "sports", etc.
  "venue": "Venue Name",
  "status": "open",  // status can be "open", "in-progre
ss", or "closed"
  "price": 223,
  "priority": "high",  // priority can be "low", "mediu
m", or "high"
  "dueDate": "2024-08-01T18:00:00Z",
  "createdBy": "user_id"
}
```

- **Constraints:**
  - The user creating the ticket must be authenticated.
  - `createdBy` must correspond to a valid user ID.
  - The `dueDate` must be a future date.
- **Response:**

```
{
  "id": "ticket_id",
  "title": "Ticket Title",
  "description": "Ticket Description",
  "type": "concert",
  "venue": "Venue Name",
  "status": "open",
```

```
  "priority": "high",
  "dueDate": "2024-08-01T18:00:00Z",
  "createdBy": "user_id",
  "assignedUsers": []
}
```

---

4. Assign a User to a Ticket

- **Endpoint:** `POST /tickets/:ticketId/assign`
- **Request Headers:**

```
Authorization: Bearer jwt_token
```

- **Request Body:**

```
{
  "userId": "user_id"
}
```

- **Constraints:**
  - A user cannot be assigned to the same ticket more than once.
  - A ticket should not be assignable if it is closed.
  - The `userId` must correspond to a valid user.
  - You can't assign a ticket to an **admin**.
  - The total number of assigned users should not exceed a defined limit (e.g., 5 users per ticket).
  - Only the user who created the ticket or an admin can assign users to the ticket.
- **Response (Success):**

```
{
  "message": "User assigned successfully"
}
```

- **Response (Failure):**

```
{
  "message": "User already assigned" // or "Cannot assig
n users to a closed ticket" or "User does not exist" or
"User assignment limit reached" or "Unauthorized"
}
```

---

5. Get Ticket Details

- **Endpoint:** `GET /tickets/:ticketId`
- **Request Headers:**

```
Authorization: Bearer jwt_token
```

- **Response:**

```
{
  "id": "ticket_id",
  "title": "Ticket Title",
  "description": "Ticket Description",
  "type": "concert",
  "venue": "Venue Name",
  "status": "open",
  "price": 223,
  "priority": "high",
  "dueDate": "2024-08-01T18:00:00Z",
  "createdBy": "user_id",
  "assignedUsers": [
    {
      "userId": "user_id_1",
      "name": "User Name 1",
      "email": "user1@example.com"
    },
    {
      "userId": "user_id_2",
      "name": "User Name 2",
      "email": "user2@example.com"
    }
```

```
  ],
  "statistics": {
    "totalAssigned": 2,
    "status": "open"
  }
}
```

- **Constraints:**
  - Only authenticated users can view ticket details.

---

6. Ticket History

- **Endpoint:** `GET /tickets/analytics`
- **Request Headers:**

```
Authorization: Bearer jwt_token
```

- **Query Parameters (optional):**
  - `startDate`: Filter tickets created after this date
  - `endDate`: Filter tickets created before this date
  - `status`: Filter tickets by status (e.g., "closed")
  - `priority`: Filter tickets by priority (e.g., "high")
  - `type`: Filter tickets by type (e.g., "concert")
  - `venue`: Filter tickets by venue
- **Response:**

```
{
  "totalTickets": 50,
  "closedTickets": 30,
  "openTickets": 15,
  "inProgressTickets": 5,
  "priorityDistribution": {
    "low": 10,
    "medium": 20,
    "high": 20
```

```
  },
  "typeDistribution": {
    "concert": 20,
    "conference": 15,
    "sports": 15
  },
  "tickets": [
    {
      "id": "ticket_id_1",
      "title": "Ticket Title 1",
      "status": "closed",
      "priority": "high",
      "type": "concert",
      "venue": "Venue 1",
      "createdDate": "2024-07-01T18:00:00Z",
      "createdBy": "user_id_1"
    },
    {
      "id": "ticket_id_2",
      "title": "Ticket Title 2",
      "status": "in-progress",
      "priority": "medium",
      "type": "conference",
      "venue": "Venue 2",
      "createdDate": "2024-07-05T18:00:00Z",
      "createdBy": "user_id_2"
    }
  ]
}
```

- **Constraints:**
  - Only authenticated users can view analytics.

---

7. Ticket Analytics

- **Endpoint:** `GET /dashboard/analytics`
- **Query Parameters (optional):**

- `startDate`: Filter tickets created after this date
- `endDate`: Filter tickets created before this date
- `status`: Filter tickets by status (e.g., "closed")
- `priority`: Filter tickets by priority (e.g., "high")
- `type`: Filter tickets by type (e.g., "concert")
- `venue`: Filter tickets by venue

- **Response:**

```
{
  "totalTickets": 50,
  "closedTickets": 30,
  "openTickets": 15,
  "averageCustomerSpending": 500, // The average of mone
y that a single customer has spent in the given timespan
  "AverageTicketsBookedPerDay": 100,
  "inProgressTickets": 5,
  "priorityDistribution": {
    "low": 10,
    "averageLowTicketsBookedPerDay": 1.2,
    "medium": 20,
    "averageMediumTicketsBookedPerDay": 2,
    "high": 20,
    "AverageHighTicketsBookedPerDay": 2
  },
  "typeDistribution": {
    "concert": 20,
    "conference": 15,
    "sports": 15
  }
}
```

🛠️ Instructions

- Implement the above endpoints using **Node.js, TypeScript, and Express**.
- Use **PostgreSQL** for data storage.

- Ensure the API endpoints handle **edge cases and validation**.
- Implement JWT-based authentication and authorization
- **Use Raw SQL Queries**