Student Admission Funnel & Reporting System

Objective

Build a backend system to manage student leads, counselor interactions, and admission tracking. Your task is to design and implement APIs to log data and generate meaningful reports that help analyze lead conversion and counselor performance.

Data Models

`leads`

```
{
  "_id": ObjectId,
  "name": String,
  "email": String,
  "phone": String,
  "source": "organic" | "ads" | "referral",
  "createdAt": ISODate,
  "counselorId": ObjectId,
  "status": "new" | "contacted" | "demoed" | "admitted" | "rejected"
}
```

`counselors`

```
{
  "_id": ObjectId,
  "name": String,
  "region": String
}
```

`interactions`

```
{
  "_id": ObjectId,
  "leadId": ObjectId,
  "counselorId": ObjectId,
  "interactionType": "call" | "demo" | "followup" | "email",
  "timestamp": ISODate,
  "notes": String,
  "duration": Number
}
```

Tasks

1. Funnel Report API

`GET /report/funnel?from=&to=&source=`

Generate a stage-wise summary of leads: - Number of leads in each stage (new, contacted, demoed, admitted) - Show how many leads moved from one stage to the next - Include percentage conversion between stages

---

2. Counselor Performance API

`GET /report/counselor-performance?region=&from=&to=`

For each counselor: - Total leads handled - Number of demos or key interactions - Average time spent with leads - Number of leads converted to admitted

---

3. Drop-off Report

`GET /report/dropoffs`

- List rejected leads with their last known interaction
- Group or categorize based on inactivity, lack of demo, or other possible reasons

---

4. Time-Based Lead Report

`GET /report/lead-buckets`

- Break down leads created into weekly or monthly intervals
- For each interval, show how many were eventually admitted

⚙ Technical Expectations

- Use Node.js with Express
- MongoDB for database
- Design clear, reusable APIs
- Structure your code for maintainability
- Think about performance, especially with filtering and grouping large datasets

---

Bonus (Optional)

- Export reports to CSV

- Add pagination where applicable
- Use Redis to cache heavy report responses

What to Submit

- Full Node.js project (GitHub or ZIP)
- Sample dataset (JSON or seed script)
- README explaining:
  - Project structure
  - Design decisions
  - Example API usages (Postman/Curl)

Evaluation Criteria

Area	Weight
Data Modeling	20%
Problem-Solving Approach	30%
API Design & Code Clarity	20%
Report Accuracy	20%
Documentation	10%