**Patient Collection:**

{

“\_id”: ObjectId,

“patient\_id”: integer,

“first\_name”: string,

“last\_name”: string,

“dob”: date,

“gender”: string,

}

**Visit Collection:**

{

“\_id”: ObjectId,

“visit\_id”: integer,

“patient\_id”: ObjectId,

“provider\_id”: ObjectId,

“visit\_date”: date,

“symptom”: [

{ “symptom”: integer,

“note”: string }

],

“diagnosis”: [

{ “diagnosis\_id”: integer,

“name”: string,

“icd10\_code”: string}

],

“lab”: [

{ “Lab\_id”: integer,

“cpt\_code”: string,

“lab\_name”: string}

],

“clinical\_procedures”: [

{ “procedure\_id”: integer,

“idc10\_code”: string,

“proc\_name”: string,

“description”: string}

]

}

**Provider Collection:**

{

“\_id”: ObjectId,

“provider\_id”: integer,

“first\_name”: string,

“last\_name”: string,

“specialty”: string

}

Embedding vs Referencing Justification:

**Patient Collection:**

The patient collection stores information about unique patients and is referenced by the visit collection. Referencing keeps patient documents lightweight and makes updates more manageable, especially for patients who visit frequently.

**Visit Collection:**

Reference patient: I decided to reference the patient collection because it would help avoid repeating patient details for every visit. It would also help in cases where a frequent patient updates their information.

Embed symptoms, labs, diagnosis, and clinical procedures: I decided to embed these four tables under the visit collection because all tables were tied to each specific visit in the original database.

**Provider Collection:**

The provider stays separate because they can be linked to many visits and patients. For example, if a provider treated a hundred patients and later updated their last name, we would then have to update a hundred different records.