ApnaResume - Database Schema Documentation

Table of Contents

Database Overview

Database Type

- **Database**: MongoDB (NoSQL Document Database)
- **ODM**: Mongoose v8.8.3
- **Driver**: MongoDB Node.js Driver v6.8.0

Collections

Collection	Model Name	Description
`resumes`	Resume	Main resume documents with metadata
`experiences`	Experience	Work experience entries
`educations`	Education	Educational background entries
`skills`	Skill	Skill categories and lists

Database Statistics

Metric	Typical Value
Avg Resume Document Size	~2-5 KB
Avg Experience Document Size	~500 bytes
Avg Education Document Size	~400 bytes
Avg Skill Document Size	~200 bytes
Max Experiences per Resume	Unlimited (typically 3-10)
Max Education per Resume	Unlimited (typically 1-3)
Max Skills per Resume	Unlimited (typically 4-8 categories)

Connection Configuration

Environment Variable

```
NEXT_MONGODB_URL=mongodb://username:password@host:port/database?options
```

Connection Code

File: `lib/mongoose.ts`

```
import mongoose from "mongoose";

let isConnected = false;

export const connectToDB = async () => {
  mongoose.set("strictQuery", true);

  if (!process.env.NEXT_MONGODB_URL) {
    return console.error("MongoDB URL not found");
  }

  if (isConnected) {
    return;
  }

  try {
    await mongoose.connect(process.env.NEXT_MONGODB_URL);
    isConnected = true;
    console.log("✓ MongoDB connected successfully");
  } catch (error) {
    console.error("X MongoDB connection failed:", error);
  }
};
```

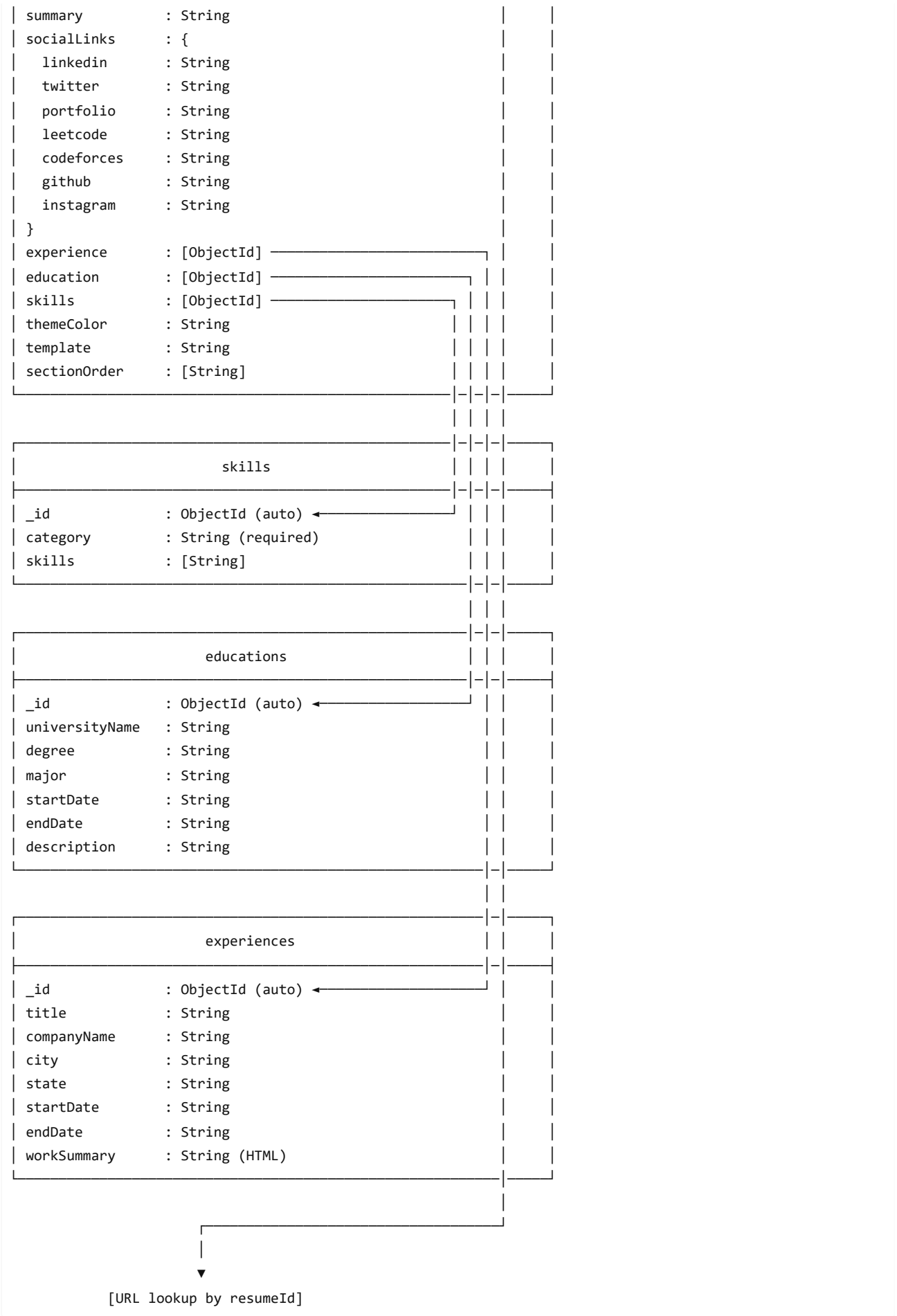
Connection Options

- **strictQuery**: `true` - Strict mode for queries
- **Connection Pool**: Default Mongoose pooling
- **Retry**: Default MongoDB driver retry logic

Collection Schemas

Visual Schema Overview

```
┌─────────────────────────────────────────────────────────┐
│                        resumes                          │
├─────────────────────────────────────────────────────────┤
│ _id              : ObjectId (auto)                      │
│ resumeId         : String (unique, required) ──────────┐ │
│ userId           : String (required)                   │ │
│ title            : String (required)                   │ │
│ updatedAt        : Date                                │ │
│ firstName        : String                              │ │
│ lastName         : String                              │ │
│ jobTitle         : String                              │ │
│ address          : String                              │ │
│ phone            : String                              │ │
│ email            : String                              │ │
```


summary : String
socialLinks : {
linkedin : String
twitter : String
portfolio : String
leetcode : String
codeforces : String
github : String
instagram : String
}
experience : [ObjectId]
education : [ObjectId]
skills : [ObjectId]
themeColor : String
template : String
sectionOrder : [String]
skills
_id : ObjectId (auto)
category : String (required)
skills : [String]
educations
_id : ObjectId (auto)
universityName : String
degree : String
major : String
startDate : String
endDate : String
description : String
experiences
_id : ObjectId (auto)
title : String
companyName : String
city : String
state : String
startDate : String
endDate : String
workSummary : String (HTML)
[URL lookup by resumeId]

Schema Definitions

Resume Schema

File: `lib/models/resume.model.ts`

```
import mongoose from "mongoose";
import { themeColors } from "../utils";

const resumeSchema = new mongoose.Schema({
  // Primary identifier
  resumeId: {
    type: String,
    required: true,
    unique: true
  },

  // Owner reference (Clerk user ID)
  userId: {
    type: String,
    required: true
  },

  // Resume metadata
  title: {
    type: String,
    required: true
  },
  updatedAt: {
    type: Date,
    default: Date.now
  },

  // Personal information
  firstName: { type: String },
  lastName: { type: String },
  jobTitle: { type: String },
  address: { type: String },
  phone: { type: String },
  email: { type: String },
  summary: { type: String },

  // Social links (embedded document)
  socialLinks: {
    linkedin: { type: String },
    twitter: { type: String },
    portfolio: { type: String },
    leetcode: { type: String },
    codeforces: { type: String },
    github: { type: String },
    instagram: { type: String },
  },

  // References to related collections
  experience: [{
    type: mongoose.Schema.Types.ObjectId,
```

```
    ref: "Experience"
  }],
  education: [{
    type: mongoose.Schema.Types.ObjectId,
    ref: "Education"
  }],
  skills: [{
    type: mongoose.Schema.Types.ObjectId,
    ref: "Skill"
  }],

  // Appearance settings
  themeColor: {
    type: String,
    default: themeColors[0]  // "#90A4AE"
  },
  template: {
    type: String,
    default: "classic"
  },
  sectionOrder: {
    type: [String],
    default: ["summary", "experience", "education", "skills"]
  },
});

const Resume = mongoose.models.Resume || mongoose.model("Resume", resumeSchema);

export default Resume;
```

Experience Schema

File: `lib/models/experience.model.ts`

```
import mongoose from "mongoose";

const experienceSchema = new mongoose.Schema({
  title: { type: String },
  companyName: { type: String },
  city: { type: String },
  state: { type: String },
  startDate: { type: String },
  endDate: { type: String },
  workSummary: { type: String },
});

const Experience =
  mongoose.models.Experience || mongoose.model("Experience", experienceSchema);

export default Experience;
```

Education Schema

File: `lib/models/education.model.ts`

```
import mongoose from "mongoose";

const educationSchema = new mongoose.Schema({
  universityName: { type: String },
  degree: { type: String },
  major: { type: String },
  startDate: { type: String },
  endDate: { type: String },
  description: { type: String },
});

const Education =
  mongoose.models.Education || mongoose.model("Education", educationSchema);

export default Education;
```

Skill Schema

File: `lib/models/skill.model.ts`

```
import mongoose from "mongoose";

const skillSchema = new mongoose.Schema({
  category: { type: String, required: true },
  skills: [{ type: String }],
});

const Skill = mongoose.models.Skill || mongoose.model("Skill", skillSchema);

export default Skill;
```

Field Specifications

Resume Collection

Field	Type	Required	Unique	Default	Description
`_id`	ObjectId	Auto	Yes	Auto	MongoDB document ID
`resumeId`	String	Yes	Yes	-	UUID v4 identifier
`userId`	String	Yes	No	-	Clerk user ID
`title`	String	Yes	No	-	Resume name
`updatedAt`	Date	No	No	`Date.now`	Last update
`firstName`	String	No	No	-	User first name
`lastName`	String	No	No	-	User last name
`jobTitle`	String	No	No	-	Target job
`address`	String	No	No	-	Location
`phone`	String	No	No	-	Phone number

`email`	String	No	No	-	Email address
`summary`	String	No	No	-	Professional summary
`socialLinks`	Object	No	No	`{}`	Social media URLs
`experience`	[ObjectId]	No	No	`[]`	Experience refs
`education`	[ObjectId]	No	No	`[]`	Education refs
`skills`	[ObjectId]	No	No	`[]`	Skill refs
`themeColor`	String	No	No	`#90A4AE`	Hex color
`template`	String	No	No	`classic`	Template type
`sectionOrder`	[String]	No	No	See below	Section order

Default sectionOrder: `["summary", "experience", "education", "skills"]`

Experience Collection

Field	Type	Required	Default	Description
`_id`	ObjectId	Auto	Auto	Document ID
`title`	String	No	-	Job title
`companyName`	String	No	-	Company name
`city`	String	No	-	Work city
`state`	String	No	-	Work state
`startDate`	String	No	-	Start date
`endDate`	String	No	-	End date (empty = current)
`workSummary`	String	No	-	HTML-formatted description

Education Collection

Field	Type	Required	Default	Description
`_id`	ObjectId	Auto	Auto	Document ID
`universityName`	String	No	-	Institution name
`degree`	String	No	-	Degree type
`major`	String	No	-	Field of study
`startDate`	String	No	-	Start date
`endDate`	String	No	-	End date
`description`	String	No	-	Additional info

Skill Collection

Field	Type	Required	Default	Description

`_id`	ObjectId	Auto	Auto	Document ID
`category`	String	Yes	-	Skill category
`skills`	[String]	No	`[]`	Skills list

Relationships & References

Reference Diagram

```
resumes.experience[0] ──────► experiences._id
resumes.experience[1] ──────► experiences._id
resumes.experience[N] ──────► experiences._id

resumes.education[0] ──────► educations._id
resumes.education[N] ──────► educations._id

resumes.skills[0] ──────────► skills._id
resumes.skills[N] ──────────► skills._id
```

Population Query

```javascript
const resume = await Resume.findOne({ resumeId: resumeId })
  .populate({
    path: "experience",
    model: Experience,
  })
  .populate({
    path: "education",
    model: Education,
  })
  .populate({
    path: "skills",
    model: Skill,
  });
```

Populated Result Structure

```
{
  "_id": "ObjectId",
  "resumeId": "uuid-string",
  "userId": "clerk-user-id",
  "experience": [
    {
      "_id": "ObjectId",
      "title": "Software Engineer",
      "companyName": "TechCorp",
      ...
    }
  ],
  "education": [
    {
      "_id": "ObjectId",
```

```
      "universityName": "MIT",
      ...
    }
  ],
  "skills": [
    {
      "_id": "ObjectId",
      "category": "Languages",
      "skills": ["JavaScript", "Python"]
    }
  ]
}
```

Sample Documents

Resume Document

```
{
  "_id": {"$oid": "6789abcdef123456789abcde"},
  "resumeId": "550e8400-e29b-41d4-a716-446655440000",
  "userId": "user_2abc123def456",
  "title": "Full Stack Developer Resume",
  "updatedAt": {"$date": "2025-01-10T12:00:00.000Z"},
  "firstName": "Alex",
  "lastName": "Johnson",
  "jobTitle": "Senior Full Stack Developer",
  "address": "San Francisco, CA 94102",
  "phone": "+1 (555) 123-4567",
  "email": "alex.johnson@email.com",
  "summary": "Passionate Full Stack Developer with 6+ years of experience...",
  "socialLinks": {
    "linkedin": "https://linkedin.com/in/alexjohnson",
    "github": "https://github.com/alexjohnson",
    "portfolio": "https://alexjohnson.dev",
    "twitter": "https://twitter.com/alexjohnsondev",
    "leetcode": "https://leetcode.com/alexjohnson"
  },
  "experience": [
    {"$oid": "6789abcdef123456789abcd1"},
    {"$oid": "6789abcdef123456789abcd2"},
    {"$oid": "6789abcdef123456789abcd3"}
  ],
  "education": [
    {"$oid": "6789abcdef123456789abce1"}
  ],
  "skills": [
    {"$oid": "6789abcdef123456789abcf1"},
    {"$oid": "6789abcdef123456789abcf2"},
    {"$oid": "6789abcdef123456789abcf3"}
  ],
  "themeColor": "#2563eb",
  "template": "classic",
  "sectionOrder": ["summary", "experience", "education", "skills"]
}
```

Experience Document


```
{
  "_id": {"$oid": "6789abcdef123456789abcd1"},
  "title": "Senior Full Stack Developer",
  "companyName": "TechCorp Inc.",
  "city": "San Francisco",
  "state": "CA",
  "startDate": "Jan 2022",
  "endDate": "",
  "workSummary": "<ul><li>Led a team of 5 developers to build a real-time analytics dashboard serving 100K+ daily users</li><li>Architected microservices infrastructure using Node.js and Kubernetes</li></ul>"
}
```


Education Document


```
{
  "_id": {"$oid": "6789abcdef123456789abce1"},
  "universityName": "University of California, Berkeley",
  "degree": "Bachelor of Science",
  "major": "Computer Science",
  "startDate": "Aug 2013",
  "endDate": "May 2017",
  "description": "Graduated with honors. Relevant coursework: Data Structures, Algorithms, Database Systems."
}
```


Skill Document


```
{
  "_id": {"$oid": "6789abcdef123456789abcf1"},
  "category": "Languages",
  "skills": ["JavaScript", "TypeScript", "Python", "Java", "SQL", "HTML/CSS"]
}
```

Indexes

Recommended Indexes

```
// resumes collection
db.resumes.createIndex(
  { resumeId: 1 },
  { unique: true, name: "idx_resumeId_unique" }
);

db.resumes.createIndex(
  { userId: 1 },
  { name: "idx_userId" }
);
```

```
db.resumes.createIndex(
  { userId: 1, updatedAt: -1 },
  { name: "idx_userId_updatedAt" }
);

// No additional indexes typically needed for experiences/educations/skills
// as they are accessed via ObjectId references
```

Index Usage

Query Pattern	Index Used
`findOne({ resumeId })`	`idx_resumeId_unique`
`find({ userId })`	`idx_userId`
`find({ userId }).sort({ updatedAt: -1 })`	`idx_userId_updatedAt`
`findById(objectId)`	`_id_` (default)

Validation Rules

Application-Level Validation (Zod)

File: `lib/validations/resume.ts`

```
import * as z from "zod";

export const ResumeNameValidationSchema = z.object({
  name: z
    .string()
    .min(3, { message: "Name must be at least 3 characters long" }),
});
```

Template Validation

```
type TemplateType = "classic" | "modern" | "minimal";

// Valid templates
const validTemplates = ["classic", "modern", "minimal"];
```

Theme Color Validation

```
const themeColors = [
  "#90A4AE", "#E57373", "#64B5F6", "#81C784", "#FFF176",
  "#FFB74D", "#BA68C8", "#4DB6AC", "#AED581", "#FF8A65",
  "#A1887F", "#F06292", "#7986CB", "#FFD54F", "#FFEB3B"
];
```

Section Order Validation

```
const validSections = ["summary", "experience", "education", "skills"];

// Default order
const defaultSectionOrder = ["summary", "experience", "education", "skills"];
```

Migration Guide

Adding a New Field to Resume

1. Update `lib/models/resume.model.ts`:

```
const resumeSchema = new mongoose.Schema({
  // ... existing fields
  newField: { type: String, default: "" },
});
```

2. Update TypeScript types if applicable

3. Update relevant forms and previews

Adding a New Related Collection

1. Create new model in `lib/models/`:

```
import mongoose from "mongoose";

const newSchema = new mongoose.Schema({
  field1: { type: String },
  field2: { type: Number },
});

const NewModel =
  mongoose.models.NewModel || mongoose.model("NewModel", newSchema);

export default NewModel;
```

2. Add reference in Resume schema:

```
newCollection: [{
  type: mongoose.Schema.Types.ObjectId,
  ref: "NewModel"
}],
```

3. Update population in `fetchResume`:

```
.populate({
  path: "newCollection",
  model: NewModel,
})
```

Backfilling Existing Documents

```
// MongoDB shell or migration script
db.resumes.updateMany(
  { newField: { $exists: false } },
  { $set: { newField: "default value" } }
);
```

Backup & Restore

Backup Command

```
mongodump --uri="mongodb://host:port/apnaresume" --out=/backup/path
```

Restore Command

```
mongorestore --uri="mongodb://host:port/apnaresume" /backup/path/apnaresume
```

Export Single Collection

```
mongoexport --uri="mongodb://host:port/apnaresume" \
  --collection=resumes \
  --out=resumes.json
```

Performance Considerations

Query Optimization

1. **Always use resumeId for single lookups** - indexed and unique
2. **Use projection** when only specific fields needed:

   ```
   Resume.findOne({ resumeId }, { title: 1, updatedAt: 1 })
   ```

3. **Limit population** when not needed:

   ```
   // For dashboard listing, don't populate
   Resume.find({ userId })

   // For edit/view, populate all
   Resume.findOne({ resumeId }).populate(...)
   ```

Document Size

- Keep HTML in `workSummary` concise
- Limit number of experience/education entries
- Consider pagination for users with many resumes

This schema documentation provides a complete reference for understanding and working with the ApnaResume database structure.