To create and manage the database, you need a MongoDB environment. Here's how to set it up and create the database step by step:

Step 1: Install MongoDB

1. **Option 1: Install MongoDB Locally**
    - Download and install MongoDB from the official MongoDB website.

After installation, start the MongoDB service:
mongod

    -
2. **Option 2: Use MongoDB Atlas (Cloud-Based Solution)**
    - Create a free account on MongoDB Atlas.
    - Set up a free cluster.

Obtain the connection string for your database (it will look like this):

mongodb+srv://<username>:<password>@cluster0.mongodb.net/<databaseName>?retryWrites=true&w=majority

    -

Step 2: Use a MongoDB Client

You can interact with MongoDB in several ways:

1. **Mongo Shell** (Command-line interface)
2. **MongoDB Compass** (GUI for MongoDB)
3. **Programming Language** (e.g., Node.js, Python)

We'll focus on using Node.js for building your backend.

Step 3: Set Up Your Project

1. **Install Node.js** (if not already installed): Download here.

Initialize a Node.js Project:

```
 mkdir candidate-management
cd candidate-management
npm init -y
```

2.

Install Required Packages:

```
 npm install mongodb
```

3.

Step 4: Create the Database

Here's how you can create the database and collections programmatically:

Example Code to Create Database and Collections

```
const { MongoClient } = require('mongodb');

// MongoDB connection string
const uri =
'mongodb+srv://<username>:<password>@cluster0.mongodb.net/candidateDB?retryWrites=tru
e&w=majority';

// Connect to MongoDB
async function main() {
  const client = new MongoClient(uri);

  try {
    // Connect to the database
    await client.connect();
    console.log('Connected to MongoDB!');

    // Create or select database
    const database = client.db('candidateDB');

    // Create collections (optional, MongoDB creates them automatically if they don't exist)
    await database.createCollection('candidates');
    await database.createCollection('documents');
    console.log('Collections created!');
```

```
    // Example of inserting a candidate document
    const candidatesCollection = database.collection('candidates');
    const result = await candidatesCollection.insertOne({
      personalInformation: {
        name: 'John Doe',
        email: 'john.doe@example.com',
        phone: '+1234567890',
      },
      education: {
        masters: {
          institution: 'XYZ University',
          program: 'M.Tech',
          branch: 'AI',
        },
      },
    });

    console.log(`Candidate inserted with ID: ${result.insertedId}`);
  } catch (err) {
    console.error('Error connecting to MongoDB:', err);
  } finally {
    // Close the connection
    await client.close();
  }
}

main();
```

Step 5: Verify the Database

1. **If Using MongoDB Compass**:
    - Open MongoDB Compass.
    - Connect using the same connection string.
    - Check for the `candidateDB` database and its `candidates` and `documents` collections.

If Using Mongo Shell:

```
 mongo
use candidateDB
show collections
```

db.candidates.find().pretty()

2.

Step 6: Manage Documents

- **Insert Candidates**: Use `insertOne` or `insertMany` methods.
- **Query Candidates**: Use `find` or `findOne`.
- **Store Documents**: Save file paths in the `documents` collection and store files in a file storage solution (e.g., AWS S3 or local server).

If you want code for specific operations (e.g., file uploads or advanced queries), let me know!