Storing & Using Config Keys in MongoDB (App Settings)

Step 1: Create Config Model (models/AppConfig.js)

```
const mongoose = require('mongoose');

const configSchema = new mongoose.Schema({
  key: { type: String, required: true, unique: true },
  value: mongoose.Schema.Types.Mixed
});

module.exports = mongoose.model('AppConfig', configSchema);
```

Step 2: Insert Initial Keys (e.g. from Postman)

```
POST http://localhost:5000/api/config/add
Body:
{
  "key": "unitRatePerKW",
  "value": 7
}
```

More examples:

```
[
  { "key": "unitRatePerKW", "value": 7 },
  { "key": "roiPercent", "value": 3.5 },
  { "key": "co2SavedPerKW", "value": 25 }
]
```

Step 3: Fetch a Key from DB (in controller)

```
const AppConfig = require('../models/AppConfig');
const rate = await AppConfig.findOne({ key: "unitRatePerKW" });
console.log("Unit rate is:", rate.value);
```

Step 4: Update a Key in DB

```
await AppConfig.findOneAndUpdate(
  { key: "unitRatePerKW" },
  { value: 6.5 },
  { new: true }
);
```

Step 5: Create Public API to Fetch Key (optional)

----

routes/configRoutes.js:

```
const express = require('express');
const router = express.Router();
const AppConfig = require('../models/AppConfig');

router.get('/config/:key', async (req, res) => {
  const key = req.params.key;
  const config = await AppConfig.findOne({ key });
  if (config) {
    res.json({ value: config.value });
  } else {
    res.status(404).json({ message: "Key not found" });
  }
});
```

```
module.exports = router;
```

Frontend Example Usage

GET http://localhost:5000/api/config/unitRatePerKW

Response:

```
{ "value": 7 }
```

Best Practice Summary

- Use 'key' and 'value' pair to manage settings centrally
- Fetch config wherever needed instead of hardcoding
- Easy to update via GUI or admin route
- Store values like rates, constants, thresholds etc.