Here's a more structured and detailed guide to building a URL shortener service using Node.js, Express, and MongoDB:

1. Project Setup

Install Node.js and npm:

Ensure Node.js and npm are installed. Download from nodejs.org.

Initialize your project:

sh

Copy code

mkdir url-shortener

cd url-shortener

npm init -y

Install dependencies:

sh

Copy code

npm install express mongoose shortid

- express: A web framework for Node.js.
- mongoose: An ODM (Object Data Modeling) library for MongoDB.
- shortid: Generates short, unique, non-sequential IDs.

2. Create the Server

Create server.js:

js

Copy code

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

const app = express();
const PORT = process.env.PORT || 5000;

// MongoDB connection
mongoose.connect('mongodb://localhost/urlshortener', {
```

```
  useNewUrlParser: true,
  useUnifiedTopology: true
});

// Define URL schema
const urlSchema = new mongoose.Schema({
  originalUrl: String,
  shortUrl: String,
  shortId: {
    type: String,
    default: shortid.generate
  }
});

const Url = mongoose.model('Url', urlSchema);

app.use(express.json());

// Routes
app.post('/shorten', async (req, res) => {
  const { originalUrl } = req.body;
  const url = new Url({ originalUrl });
  await url.save();
  res.json({ shortUrl: `http://localhost:${PORT}/${url.shortId}` });
});

app.get('/:shortId', async (req, res) => {
  const { shortId } = req.params;
  const url = await Url.findOne({ shortId });
  if (url) {
    res.redirect(url.originalUrl);
```

```
    } else {
        res.status(404).send('URL not found');
    }
});


app.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}`);
});
```

3. Run the Server

Start your server by running:

sh

Copy code

```
node server.js
```

4. Test the API

Shorten a URL:

Use a tool like curl or Postman to send a POST request:

sh

Copy code

```
curl -X POST -H "Content-Type: application/json" -d '{"originalUrl": "http://example.com"}' http://localhost:5000/shorten
```

Redirect using Short URL:

Open the short URL provided in the response in your browser to test the redirection.

5. Enhancements

- **Validation**: Add validation to ensure the input URL is a valid URL.
- **Analytics**: Track the number of times each short URL is accessed.
- **User Interface**: Build a simple front-end interface for users to input URLs and get short links.
- **Deployment**: Deploy your service on platforms like Heroku, AWS, or DigitalOcean for wider access.

Sample Enhancements Code

URL Validation:

You can use a package like valid-url to validate URLs:

sh

Copy code

npm install valid-url

Update server.js:

js

Copy code

```
const validUrl = require('valid-url');


// In the POST /shorten route
app.post('/shorten', async (req, res) => {
  const { originalUrl } = req.body;


  if (!validUrl.isUri(originalUrl)) {
    return res.status(400).json('Invalid URL');
  }


  const url = new Url({ originalUrl });
  await url.save();
  res.json({ shortUrl: `http://localhost:${PORT}/${url.shortId}` });
});
```

This structure provides a solid foundation for building and enhancing a URL shortener service. Let me know if you need further details on any specific part!