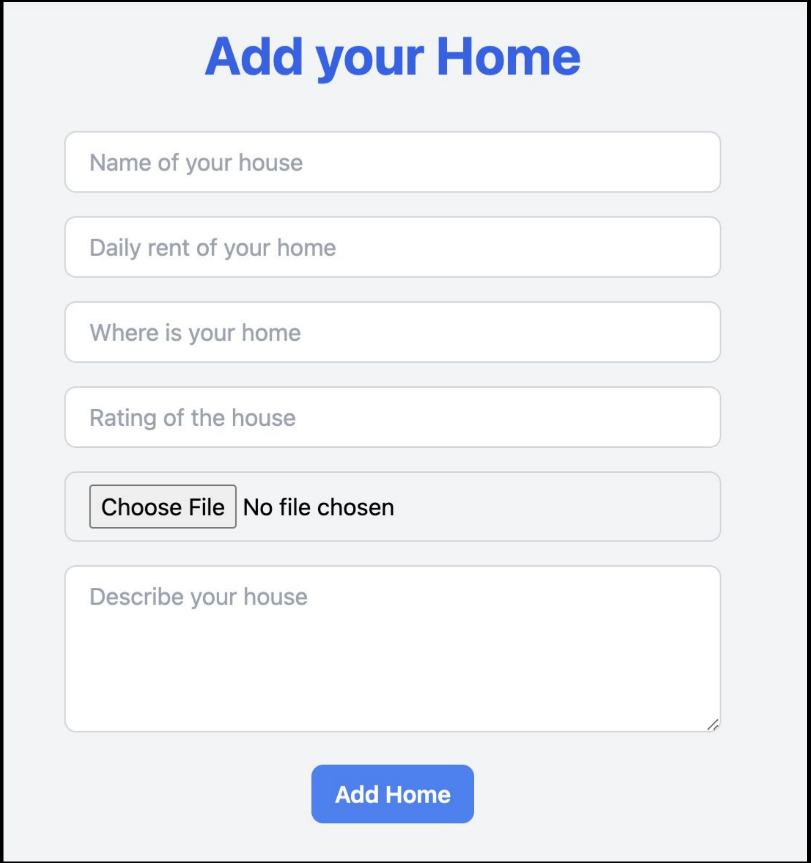
21. File Upload & Download

1. Adding a File Picker
2. Creating Multipart Form
3. Handling Multipart Form Data
4. Saving Image Files
5. Custom File Names
6. Restricting Upload File Types
7. Handling Edits
8. Serving Saved Data
9. Serving Files after Auth
10. Deleting Files

21.1 Adding a File Picker

- **Input Element:** Use <input type="file"> to create a file picker.
- **Multiple Files:** Add multiple attribute to allow selecting multiple files.
- **File Types:** Use accept attribute to restrict file types (e.g., accept=".jpg, .png").

```
<input type="file"
  name="photo"
  class="w-full px-4 py-2 mb-4 border ⬜border-gray-300
  rounded-lg focus:outline-none 🟦focus:border-blue-500"
/>
```



Add your Home

Name of your house

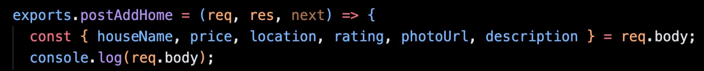
Daily rent of your home

Where is your home

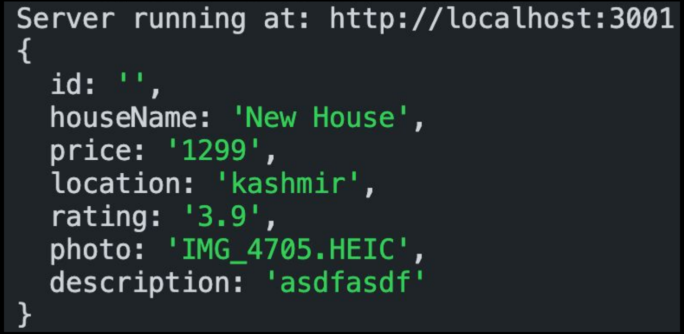
Rating of the house

Describe your house

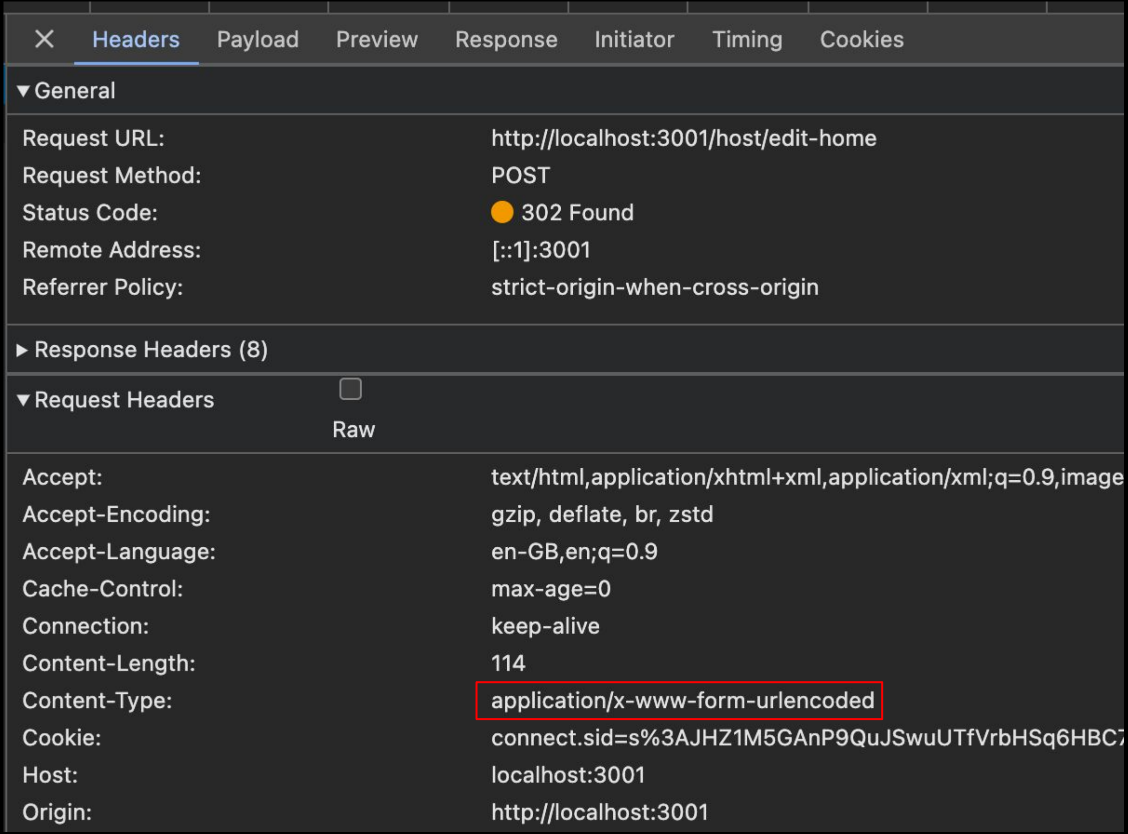
21.2 Creating Multipart Form



```
exports.postAddHome = (req, res, next) => {
  const { houseName, price, location, rating, photoUrl, description } = req.body;
  console.log(req.body);
```



```
Server running at: http://localhost:3001
{
  id: '',
  houseName: 'New House',
  price: '1299',
  location: 'kashmir',
  rating: '3.9',
  photo: 'IMG_4705.HEIC',
  description: 'asdfasdf'
}
```



▼ General

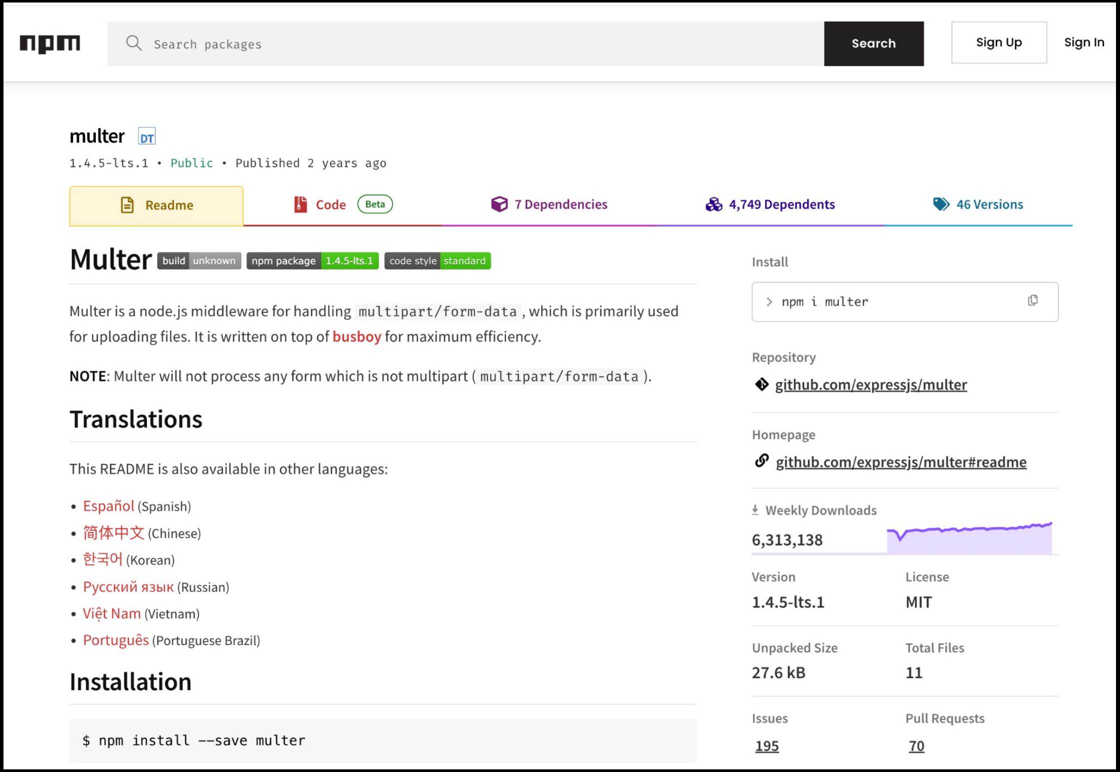
Request URL:	http://localhost:3001/host/edit-home
Request Method:	POST
Status Code:	● 302 Found
Remote Address:	[::1]:3001
Referrer Policy:	strict-origin-when-cross-origin

▶ Response Headers (8)

▼ Request Headers

Accept:	text/html,application/xhtml+xml,application/xml;q=0.9,image
Accept-Encoding:	gzip, deflate, br, zstd
Accept-Language:	en-GB,en;q=0.9
Cache-Control:	max-age=0
Connection:	keep-alive
Content-Length:	114
Content-Type:	$\boxed{\text{application/x-www-form-urlencoded}}$
Cookie:	connect.sid=s%3AJHZ1M5GAnP9QuJSwuUTfVrbHSq6HBC7
Host:	localhost:3001
Origin:	http://localhost:3001

21.3 Handling Multipart Form Data



multer DT

1.4.5-lts.1 • Public • Published 2 years ago

Multer

Multer is a node.js middleware for handling `multipart/form-data`, which is primarily used for uploading files. It is written on top of busboy for maximum efficiency.

NOTE: Multer will not process any form which is not multipart (`multipart/form-data`).

Translations

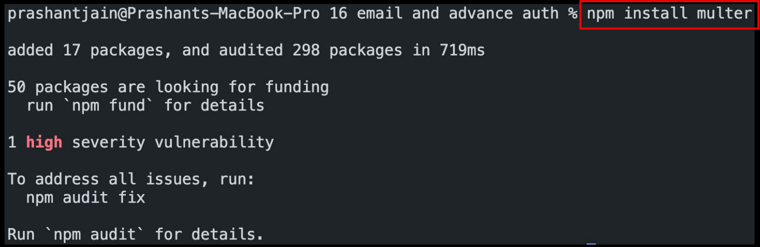
This README is also available in other languages:

- Español (Spanish)
- 简体中文 (Chinese)
- 한국어 (Korean)
- Русский язык (Russian)
- Việt Nam (Vietnam)
- Português (Portuguese Brazil)

Installation

```
$ npm install --save multer
```

Install	`npm i multer`
Repository	github.com/expressjs/multer
Homepage	github.com/expressjs/multer#readme
Weekly Downloads	6,313,138
Version	1.4.5-lts.1
License	MIT
Unpacked Size	27.6 kB
Total Files	11
Issues	195
Pull Requests	70



```
prashantjain@Prashants-MacBook-Pro 16 email and advance auth % npm install multer

added 17 packages, and audited 298 packages in 719ms

50 packages are looking for funding
  run `npm fund` for details

1 high severity vulnerability

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
```

21.3 Handling Multipart Form Data

```
<form action="/host/<%= editing ? "edit-home" : "add-home"%>" method="POST" enctype="multipart/form-data"
class="w-full max-w-md">
```

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

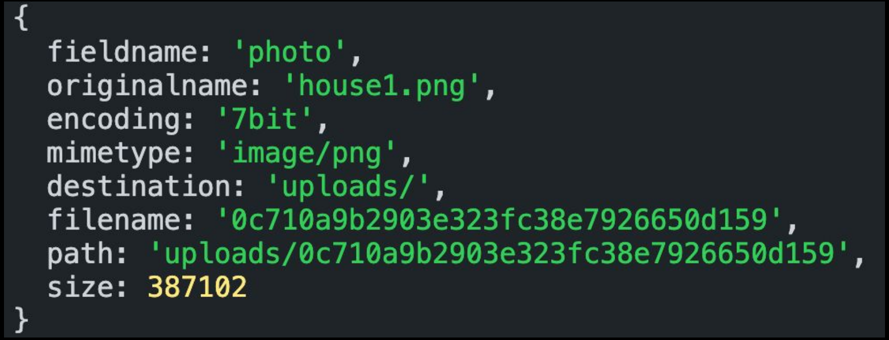
```
app.use(express.static(path.join(rootDir, "public")));
app.use(bodyParser.urlencoded({ extended: true }));
```

```
console.log(req.file);
```

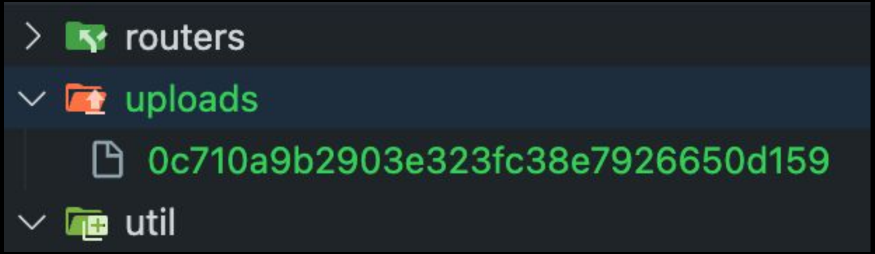
```
{
  fieldname: 'photo',
  originalname: 'IMG_4705.HEIC',
  encoding: '7bit',
  mimetype: 'image/heic',
  buffer: <Buffer 00 00 00 1c 66 74 79 70 68 65 69 63 00 00 00 00 74 6d
00 00 21 68 64 6c 72 00 00 ... 2246669 more bytes>,
  size: 2246719
}
```

21.4 Saving Image Files

```
app.use(multer({ dest: 'uploads/'}).single('photo'));
```



```
{
  fieldname: 'photo',
  originalname: 'house1.png',
  encoding: '7bit',
  mimetype: 'image/png',
  destination: 'uploads/',
  filename: '0c710a9b2903e323fc38e7926650d159',
  path: 'uploads/0c710a9b2903e323fc38e7926650d159',
  size: 387102
}
```

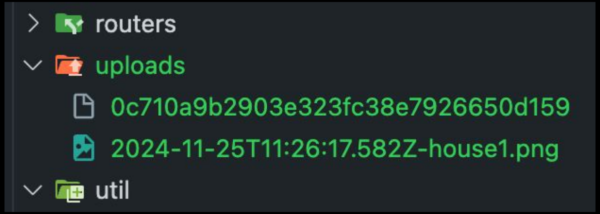


```
> routers
v uploads
    0c710a9b2903e323fc38e7926650d159
v util
```

21.5 Custom File Names

```javascript
// This defines where uploaded files will be stored and how they'll be named
const storage = multer.diskStorage({
  // Set the destination folder for uploaded files
  destination: (req, file, cb) => {
    cb(null, 'uploads/'); // Files will be saved in the 'uploads' directory
  },
  // Set the filename for uploaded files
  filename: (req, file, cb) => {
    cb(null, new Date().toISOString() + '-' + file.originalname);
  }
});
```

```javascript
app.use(multer({ storage }).single('photo'));
```



```
> routers
v uploads
    0c710a9b2903e323fc38e7926650d159
    2024-11-25T11:26:17.582Z-house1.png
v util
```

21.6 Restricting Upload File Types

```
const fileFilter = (req, file, cb) => {
  if ([ 'image/jpeg', 'image/png', 'image/jpg' ].includes(file.mimetype)) {
    cb(null, true);
  } else {
    cb(null, false);
  }
};
```

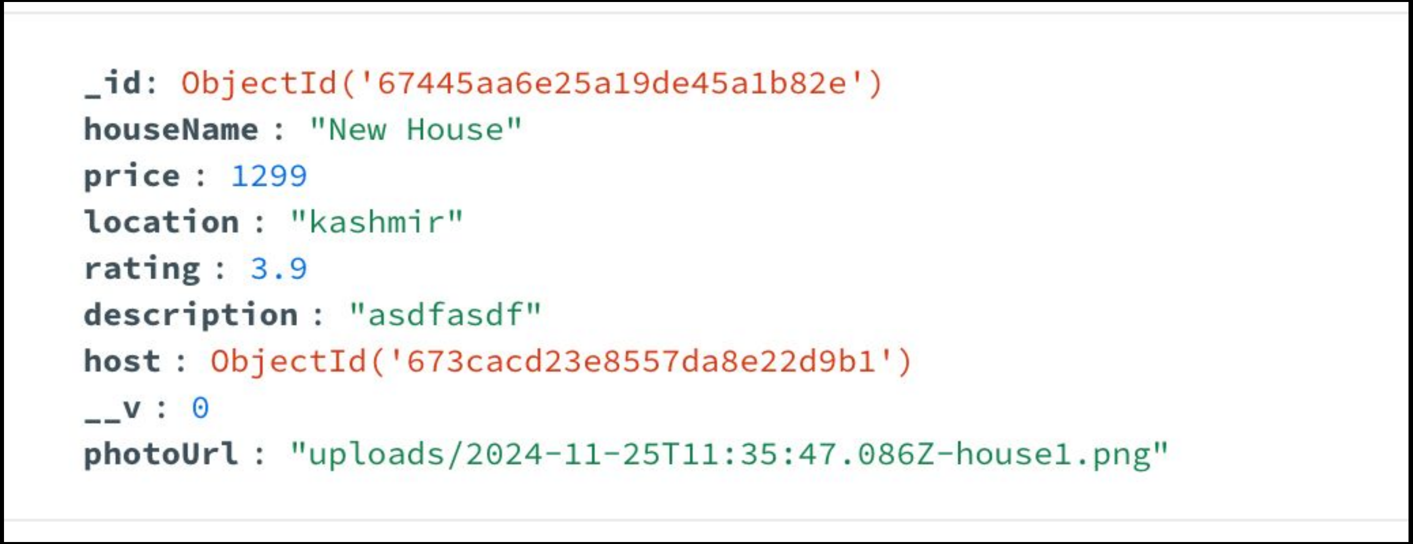
```
app.use(multer({ storage, fileFilter }).single('photo'));
```

Adding this filter, now multer would not have the file if it does not match the type and req.file will just be undefined.

```
exports.postEditHome = (req, res, next) => {
  const { id, houseName, price, location, rating, photoUrl, description } =
    req.body;

  if (!req.file) {
    return res.status(400).send('No image provided');
  }
```

21.6 Restricting Upload File Types

```
if (!req.file) {
  return res.status(400).send('No image provided');
}

const photoUrl = req.file.path;
```



```
_id: ObjectId('67445aa6e25a19de45a1b82e')
houseName : "New House"
price : 1299
location : "kashmir"
rating : 3.9
description : "asdfasdf"
host : ObjectId('673cacd23e8557da8e22d9b1')
__v : 0
photoUrl : "uploads/2024-11-25T11:35:47.086Z-house1.png"
```

21.7 Handling Edits

```
// business logic outside model
Home.findById(id)
  .then((existingHome) => {
    if (!existingHome) {
      console.log("Home not found for editing");
      return res.redirect("/host/host-homes");
    }
    existingHome.houseName = houseName;
    existingHome.price = price;
    existingHome.location = location;
    existingHome.rating = rating;
    if (req.file) {
      existingHome.photoUrl = req.file.path;
    }
    existingHome.description = description;
    return existingHome.save();
```

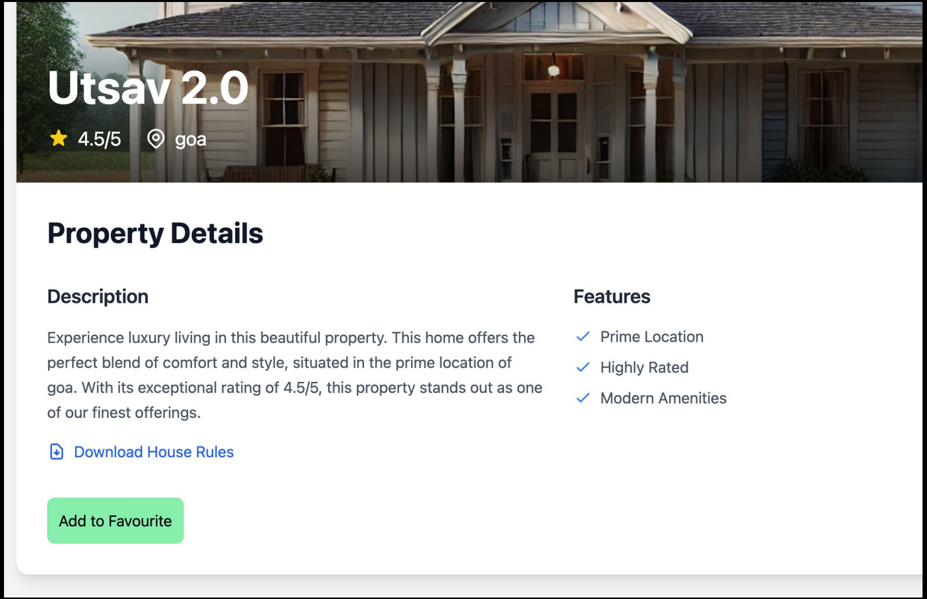
While editing we can make sure that we only overwrite the photoUrl in case a new valid image was uploaded. And don't force the user to upload images each time they edit.

21.8 Serving Saved Data

```
app.use(express.static(path.join(rootDir, "public")));
app.use('/uploads', express.static(path.join(rootDir, "uploads")));
app.use(bodyParser.urlencoded({ extended: true }));
```

This is done as like in case of public, things inside public will be served like they are in root folder and the url does not have public in it. Here also either remove uploads/ from the image path, or add the qualifier.

21.9 Serving Files after Auth



Utsav 2.0

4.5/5 goa

Property Details

Description

Experience luxury living in this beautiful property. This home offers the perfect blend of comfort and style, situated in the prime location of goa. With its exceptional rating of 4.5/5, this property stands out as one of our finest offerings.

Download House Rules

Features

- ✓ Prime Location
- ✓ Highly Rated
- ✓ Modern Amenities

```
<div class="mt-4">
  <a href="/rules/<%= home._id %>" class="inline-flex items-center text-blue-600 hover:text-blue-800">
    <svg class="w-5 h-5 mr-2" fill="none" stroke="currentColor" viewBox="0 0 24 24">
      <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 10v6m0 0l-3-3m3 3l3-3m2
      8H7a2 2 0 01-2-2V5a2 2 0 012-2h5.586a1 1 0 01.707.293l5.414 5.414a1 1 0 01.293.707V19a2 2 0 01-2 2z"/>
    </svg>
    Download House Rules
  </a>
</div>
```

21.9 Serving Files after Auth

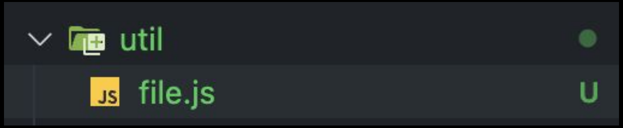
```
storeRouter.get("/rules/:homeId", storeController.getHouseRules);
```

```
const path = require('path');
const rootDir = require('../util/path-util');
```

```
exports.getHouseRules = [(req, res, next) => {
  if (!req.session.isLoggedIn) {
    return res.redirect("/login");
  }
  next();
},

(req, res, next) => {
  const homeId = req.params.homeId;
  // We can make it house specific after have homeId in file name like this: `House Rules-${homeId}.pdf`
  const rulesFileName = `House Rules.pdf`;
  const filePath = path.join(rootDir, 'rules', rulesFileName);
  // res.sendFile(filePath);
    res.download(filePath, 'Rules.pdf');
  }
];
```

21.10 Deleting Files

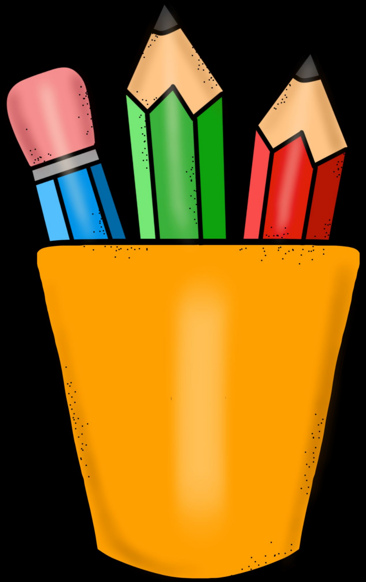
```javascript
const fs = require('fs');

exports.deleteFile = (filePath) => {
  fs.unlink(filePath, (err) => {
    if (err) throw err;
  });
};
```



```javascript
existingHome.location = location;
existingHome.rating = rating;
if (req.file) {
  deleteFile(existingHome.photoUrl);
  existingHome.photoUrl = req.file.path;
}
```

Revision

1. Adding a File Picker
2. Creating Multipart Form
3. Handling Multipart Form Data
4. Saving Image Files
5. Custom File Names
6. Restricting Upload File Types
7. Handling Edits
8. Serving Saved Data
9. Serving Files after Auth
10. Deleting Files



Practise Milestone

Take your airbnb forward:

1. Add a new button for rules upload on the edit page.
2. Make it downloadable on the home detail page.