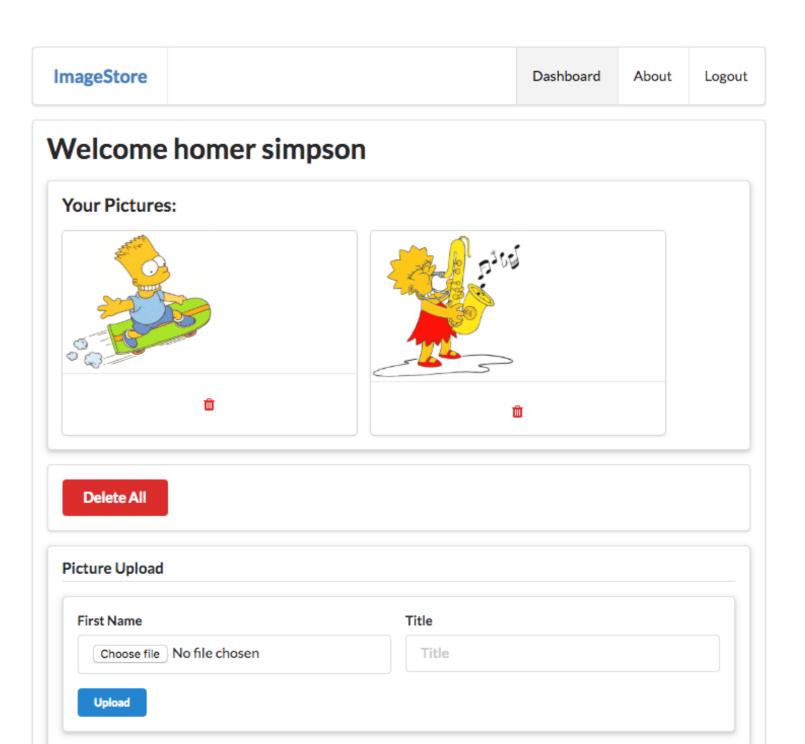
picture-store implementation

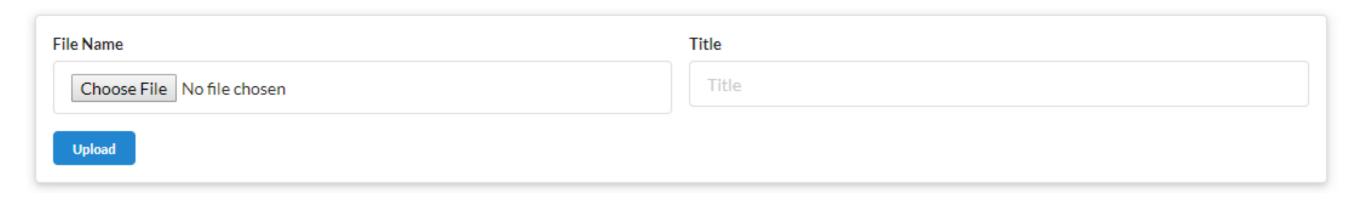
```
画 assets
controllers/about.js
controllers/accounts.js
controllers/dashboard.js
controllers/start.js
models/json-store.js
models/picture-store.js
models/picture-store.json
models/user-store.js
models/user-store.json
utils/logger.js
views/layouts/main.hbs
views/partials/card.hbs
views/partials/mainpanel.hbs
views/partials/menu.hbs
views/partials/welcomemenu.hbs
views/about.hbs
views/dashboard.hbs
views/login.hbs
views/signup.hbs
views/start.hbs
40 - .env
.gitignore
.jscsrc
README.md
package.json
routes.js
server.js
tempimage
```



```
লি assets
controllers/about.js
                                                           Dashboard Controller
controllers/accounts.js
controllers/dashboard.js
controllers/start.js
models/json-store.js
models/picture-store.js
models/picture-store.json
models/user-store.js
                                                         Wrapper for
models/user-store.json
utils/logger.js
                                                         cloudinary Service
views/layouts/main.hbs
views/partials/card.hbs
views/partials/mainpanel.hbs
views/partials/menu.hbs
views/partials/welcomemenu.hbs
views/about.hbs
views/dashboard.hbs
views/login.hbs
                                                          Display Individual
views/signup.hbs
views/start.hbs
                                                          Image
<del>d</del> .env.
.gitignore
.jscsrc
README.md
package.json
routes.js
                                                          Dashboard View
server.js
tempimage
```

# <u>Uploading Images</u>

### **Upload Picture Form**



```
<form action="/dashboard/uploadpicture" method="post" enctype="multipart/form-data">
        <div class="two fields">
            <div class="field">
            <label>File Name</label>
            <input type="file" name="picture" required> </input>
            </div>
            <div class="field">
                <label>Title</label>
            <input placeholder="Title" type="text" name="title">
                </div>
            </div>
            </div>
            <buttoon class="ui mini blue submit button"> Upload </button>
            </form>
```

## <u>Upload Picture Form + Route + Action</u>

router.post('/dashboard/uploadpicture', dashboard.uploadPicture);

```
const dashboard = {
...

uploadPicture(request, response) {
   const loggedInUser = accounts.getCurrentUser(request);
   pictureStore.addPicture(loggedInUser.id, request.body.title, request.files.picture, function () {
    response.redirect('/dashboard');
   });
},
...
};
```

#### picture-store module

```
const cloudinary = require('cloudinary');
                              const env = require('../.env.json');
                              console.log(env.cloudinary);
                              cloudinary.config(env.cloudinary);
                              const pictureStore = {
Local copy of image
                               store: new JsonStore('./models/picture-store.json', { pictures: [] }),
                               collection: 'pictures',
                       urls
   Get all image urls
                               getAlbum(userid) {
        for a give user
    Upload picture to
                               addPicture(userId, title, imageFile, response) {
              cloudinary
       Delete specific
                               deletePicture(userId, image) {
                   picture
   Delete all pictures
                               deleteAllPictures(userId) {
      for a given user
```

## picture-store.json

For each user, store the url + title of each picture uploaded

**Displaying Images** 

#### dashboard.index

logger.info('dashboard rendering');

const dashboard = {

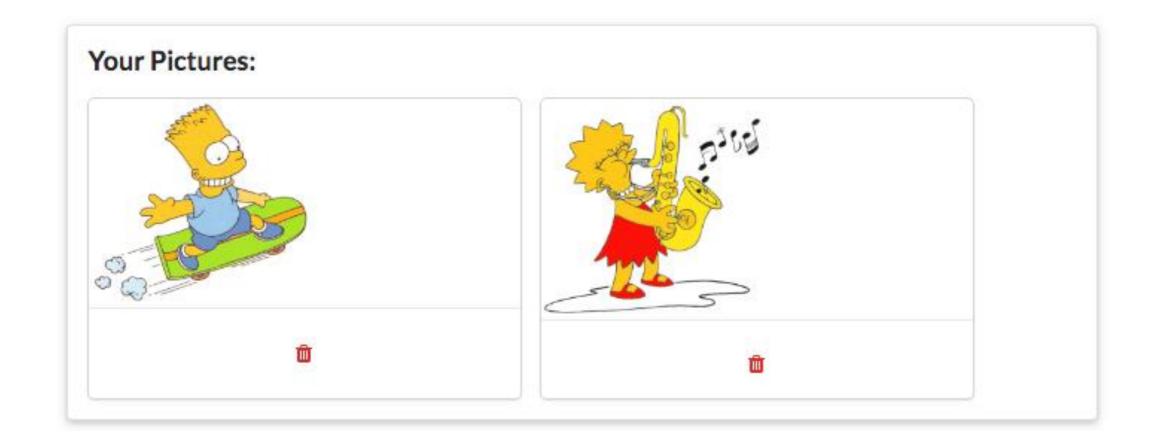
index(request, response) {

```
const loggedInUser = accounts.getCurrentUser(request);
const viewData = {
    title: 'Pictures',
    user: loggedInUser,
    album: pictureStore.getAlbum(loggedInUser.id),
};
response.render('dashboard', viewData);
},
```

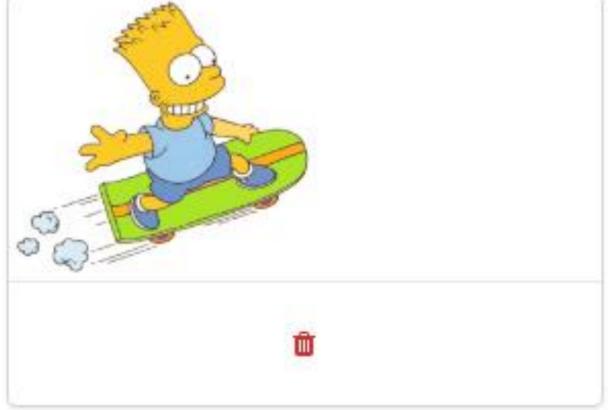
```
album
```

#### dashboard.hbs

```
<section class="ui raised segment">
  <h3 class="ui header">
    Your Pictures:
  </h3>
  <div class="ui cards">
    {#each album.photos}}
    {{> card }}
    {{/each}}
  </div>
  </section>
```



#### card.hbs





### delete picture

```
<a href="/dashboard/deletepicture?img={{img}}">
<i class="red icon trash"></i>
</a>
```

router.get('/dashboard/deletepicture', dashboard.deletePicture);

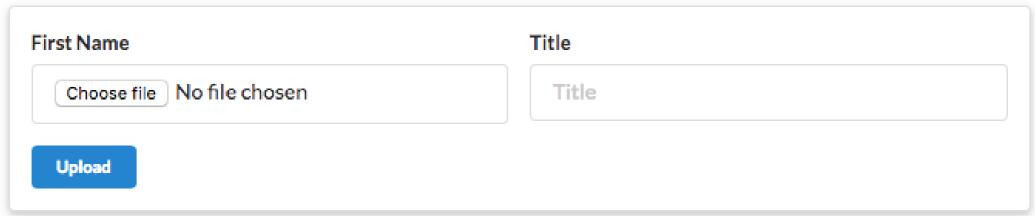
```
const dashboard = {
    ...

    deletePicture(request, response) {
    const loggedInUser = accounts.getCurrentUser(request);
    pictureStore.deletePicture(loggedInUser.id, request.query.img);
    response.redirect('/dashboard');
    },
};
```

Callbacks in picture-store

```
const dashboard = {
    ...
    uploadPicture(request, response) {
      const loggedInUser = accounts.getCurrentUser(request);
      pictureStore.addPicture(loggedInUser.id, request.body.title, request.files.picture, function () {
      response.redirect('/dashboard');
    });
    },
    ...
};
```

```
uploadPicture(request, response) {
 const loggedInUser = accounts.getCurrentUser(request);
 pictureStore.addPicture(loggedInUser.id,
                request.body.title,
                request.files.picture,
                function () {
                 response.redirect('/dashboard');
                );
```

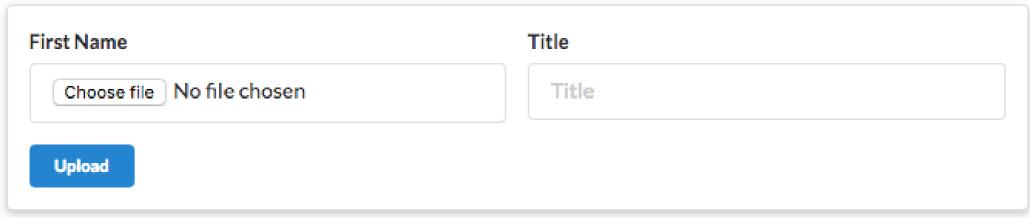


);



```
uploadPicture(request, response) {
 const loggedInUser = accounts.getCurrentUser(request);
 pictureStore.addPicture(loggedInUser.id,
                request.body.title,
                request.files.picture,
                function () {
                 response.redirect('/dashboard');
                );
```

Title entered in form



```
uploadPicture(request, response) {
 const loggedInUser = accounts.getCurrentUser(request);
 pictureStore.addPicture(loggedInUser.id,
                                                                                  The
               request.body.title,
                                                                                  picture
               request.files.picture,
                                                                                  itself
               function () {
                                                                                  (binary)
                response.redirect('/dashboard');
               );
```



```
uploadPicture(request, response) {
 const loggedInUser = accounts.getCurrentUser(request);
 pictureStore.addPicture(loggedInUser.id,
                request.body.title,
                request.files.picture,
                function () {
                 response.redirect('/dashboard');
                );
```

A 'Callback' function - involved when the picture has been successfully uploaded

```
function () {
  response.redirect('/dashboard');
}
```

Callback triggers refresh of dashboard

This will contain newly uploaded images

