FoodieGram
Chloe Le

Run mongod
Run node app.js
Go to http://localhost:3000

Express (app.js)
(cookie session, body parser, fileUpload, require mongoose models, EJS)
Ajax (/public/js)
Database Mongoose (/db/user.js)

- userSchema

```
// define schemas
var userSchema = new Schema({
  username: { type: String, lowercase: true, required: true, unique: true },
  password: { type: String, required: true },
  fullname: { type: String, required: true },
  email: {type: String, lowercase: true, required: true, unique: true, match: [/\S+@\S+\.\S+/, 'Invalid Ema
  bio: { type: String, default: ''},
  posts: [{type: Schema.ObjectId, ref: 'Post'}],
  following: [{ type: Schema.ObjectId, ref: 'User' }],
  followers: [{ type: Schema.ObjectId, ref: 'User' }],
  favorites: [{type: Schema.ObjectId, ref: 'Post'}],
  profilePic: {data: Buffer, contentType: String}
});
```

- postSchema

```
var postSchema = new Schema({
  img: {data: Buffer, contentType: String},
  _creater: { type: Schema.ObjectId, ref: 'User' },
  likes: [{ type: Schema.ObjectId, ref: 'User' }],
  rating: {type: Number, max: 5, min: 0},
  caption: { type: String },
  created_at: Date,
  place: {type: Schema.ObjectId, ref: 'Place'},
  comments: [{ type: Schema.ObjectId, ref: 'Comment' }]
});
```

- commentSchema

```
var commentSchema = new Schema({
  content: {type: String},
  author : { type: Schema.ObjectId, ref: 'User' },
  username: {type: String, lowercase: true},
  created_at: Date
});

// user schema functions
```

-
```
// save models
var User = mongoose.model('User', userSchema);
var Place = mongoose.model('Place', placeSchema);
var Post = mongoose.model('Post', postSchema);
var Comment = mongoose.model('Comment', commentSchema);


module.exports = {user: User, post: Post, comment: Comment};
```

All views use /public/style.css, for all views, we have a check for (req.session exists)

Login: (/views/login.html, /routes/login.js)

- redirects to profile if req.session exists
- unique username
- sends error if username does not exists
- sends error if wrong password
- Interacts with User.checkIfLegit(username, password)
- If ok→ redirect('feed')
- Sign Up

Signup: (/views/signup.html, /routes/signup.js)

- checks for valid email format
- checks for unique username
- checks for matching password
- if ok→post request:
    - calls User.addUser: saves a newUser with username, fullname, password, email, and a default grey profile pic (can later be changed)
    - redirects to Login

Feed: (/views/feed.html, /routes/feed.js)

- get request /feed
- template with req.session.username
- can view all posts of yourself and the people you follow with the most recent at the top of the feed
- We find the user
    - users User.getFollowing(returns user.following, an array of object id's of users you follow)

    - goes through all posts, and for each, we check if the post's creater is someone you follow or yourself, if so add the post to an array
    - post images are stored using /post/:postID
    - posts are clickable with get
        - /postPic/:postID or /mypostpic/:postID
        - which is based on if the req.session.username matches post creater's username

mypostpic and postPic: (/views/mypost.html, /views/otherspost.html, /routes/postpic.js, /public/js/postpage.js)

- EJS template
- Pass in the username, date, number of likes, whether the current user liked it, rate, whether the current user saved to favorites, comments, and comment input
- Ajax for:
    - Likes, favorites, comments
    - Post requests, sends necessary data (postID), which can be accessed using req.body.postID in the router module, which interacts with the database, and then res.send(necessary data)
    - On success: update the document using jQuery
- Can see most recent 10 comments
    - A button exists for viewing all comments

My Profile: (/views/myprofile.html, /routes/myprofile.js)

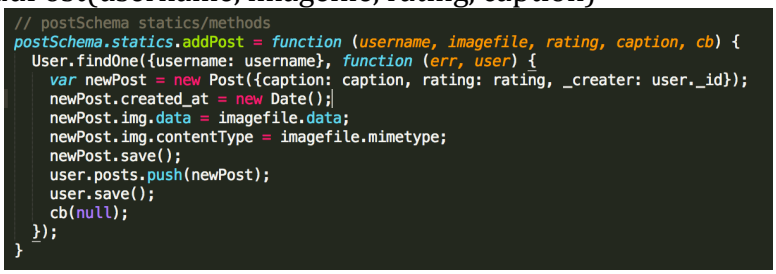
- posts displayed with the most recent post being the first one
- EJS
- Can Edit Profile
- Can upload New Post

Edit profile: (/views/editprofile.html, /routes/editprofile.js)

- templated with name and bio from the database as texts in the input boxs
- can update:
    - fullname
    - bio
    - profile picture
- Update profile button:
    - Post request to interact with the database, checks whether profile photo is null, if not, update, if so, only save name and bio
    - Redirects to profile

Upload New Post: (/views/newpost.html, /routes/newpost.js)

- post request through 'share post' button, which is accessible if no image file has been uploaded
- Can rate the food

- Can have a caption
- Post request /newpost
    - Imagefile = req.files.imagefile
    - Post.addPost(username, imagefile, rating, caption)
        - 

```
// postSchema statics/methods
postSchema.statics.addPost = function (username, imagefile, rating, caption, cb) {
  User.findOne({username: username}, function (err, user) {
    var newPost = new Post({caption: caption, rating: rating, _creater: user._id});
    newPost.created_at = new Date();
    newPost.img.data = imagefile.data;
    newPost.img.contentType = imagefile.mimetype;
    newPost.save();
    user.posts.push(newPost);
    user.save();
    cb(null);
  });
}
```

    - redirects to profile
- Can click on followers/ following→ get request /followers/:username /following/:username

Follower/Following List (views/listusers.html, get request from routes/otherprofiles.js)

- User.getFollowing and User.getFollowers to get array of userID's
- Find an array of users based on our array of ids
- Render with ejs templates
- Can click on the users to view their profiles

Others profile(/views/othersprofile.html, /routes/othersprofile.js, /public/js/othersprofile.js)

- posts displayed with the most recent post being the first one
- EJS
- Ajax for follow/unfollow (similar fashion to post-pages above)
- Can view followers and following as above
- Can click on posts as usual, and can click to commenter's profile

Discover Foodies: (/views/search.html, /public/js/serachuser.js, /routes/serachuser.js)

- Ajax request on keyup event
    - Post request passing in the search string
    - On Success, append returned data array of users on screen
- Post /searchuser
    - First, display the user with the exact user name on the very top by pushing it an array (findone(username))
    - We also display users that have fullname that matches the search term (User.find(fullname)
    - Res.send(arrayOfUsers)

My Favorites: (/views/favorites.html, /routes/myprofile.js)

- get request 'favorites'
- displays all posts you have added to your favorites' list

Logout:

- empties out req.session
- redirects to login