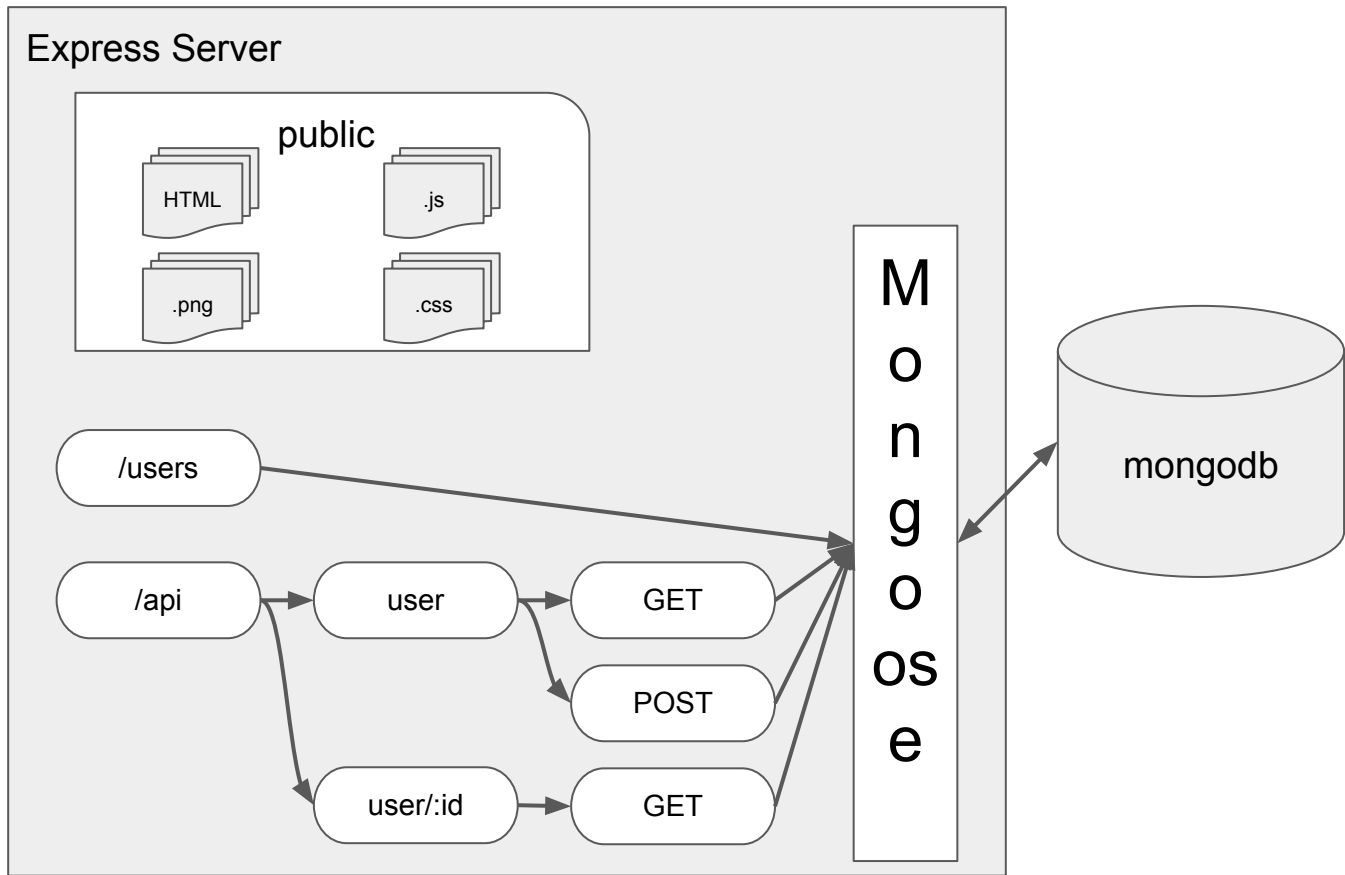
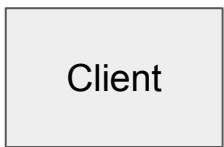
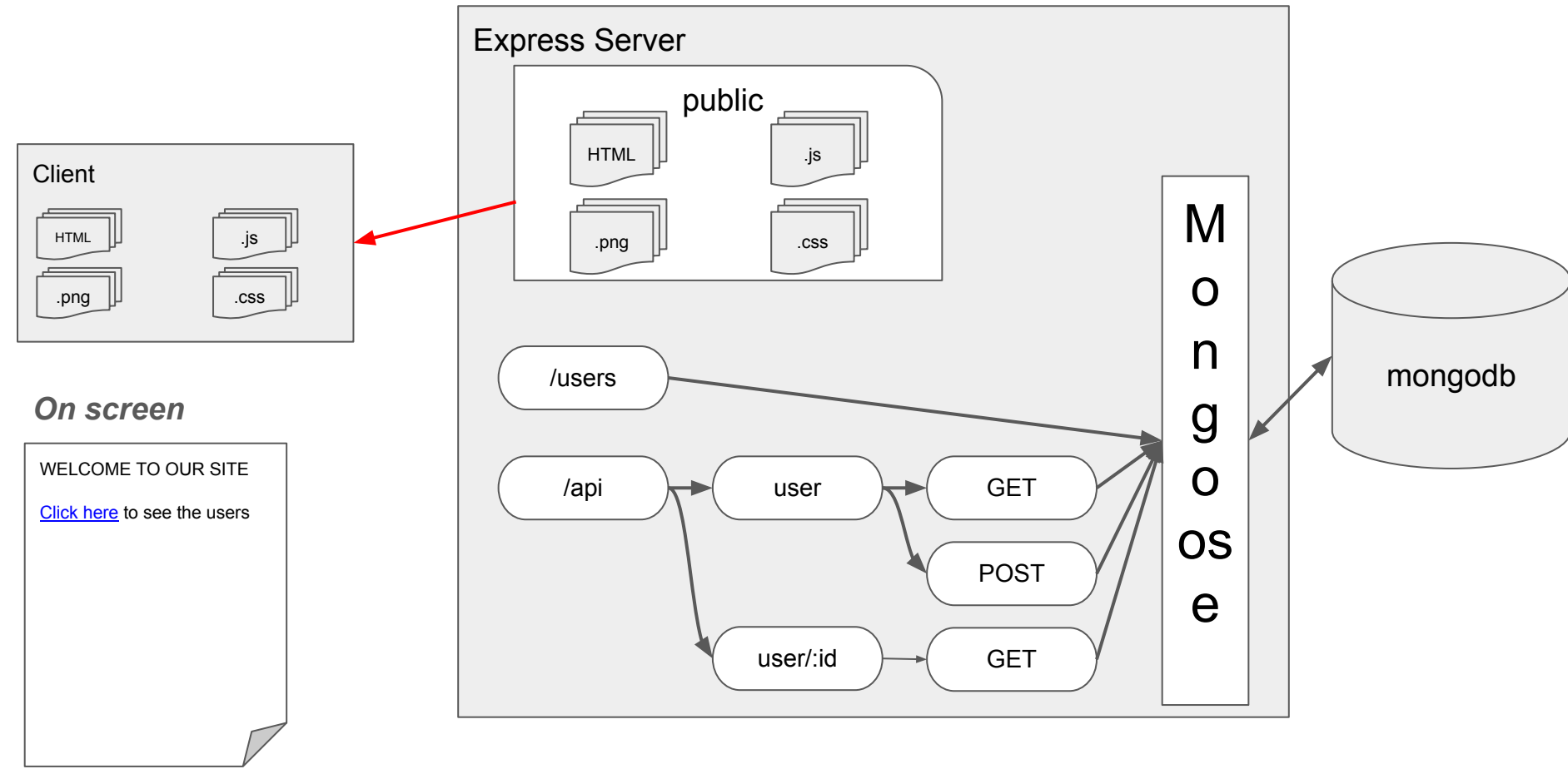


Server Side Rendered

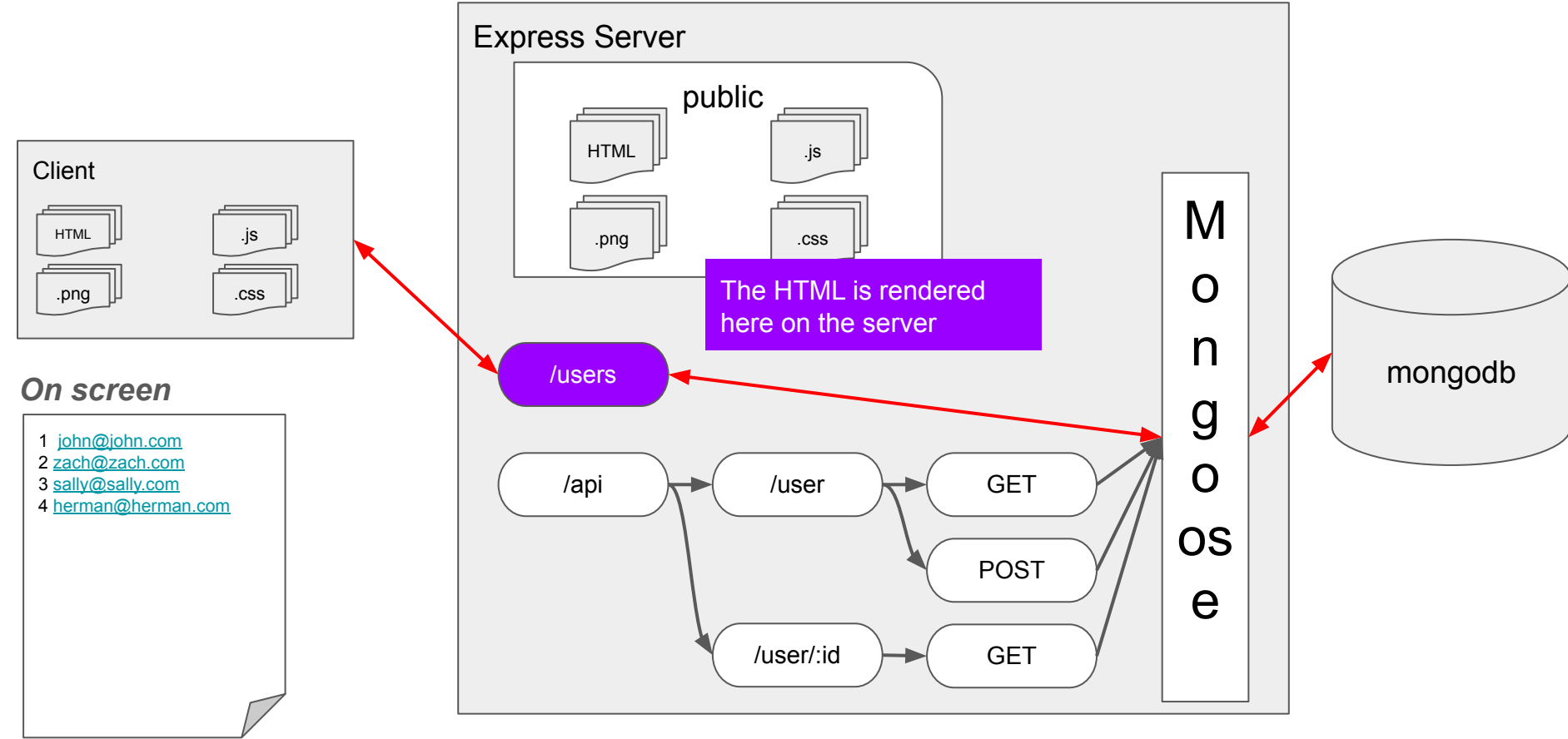
E.g., Handlebars



2. Client Requests `http://localhost:3000/`



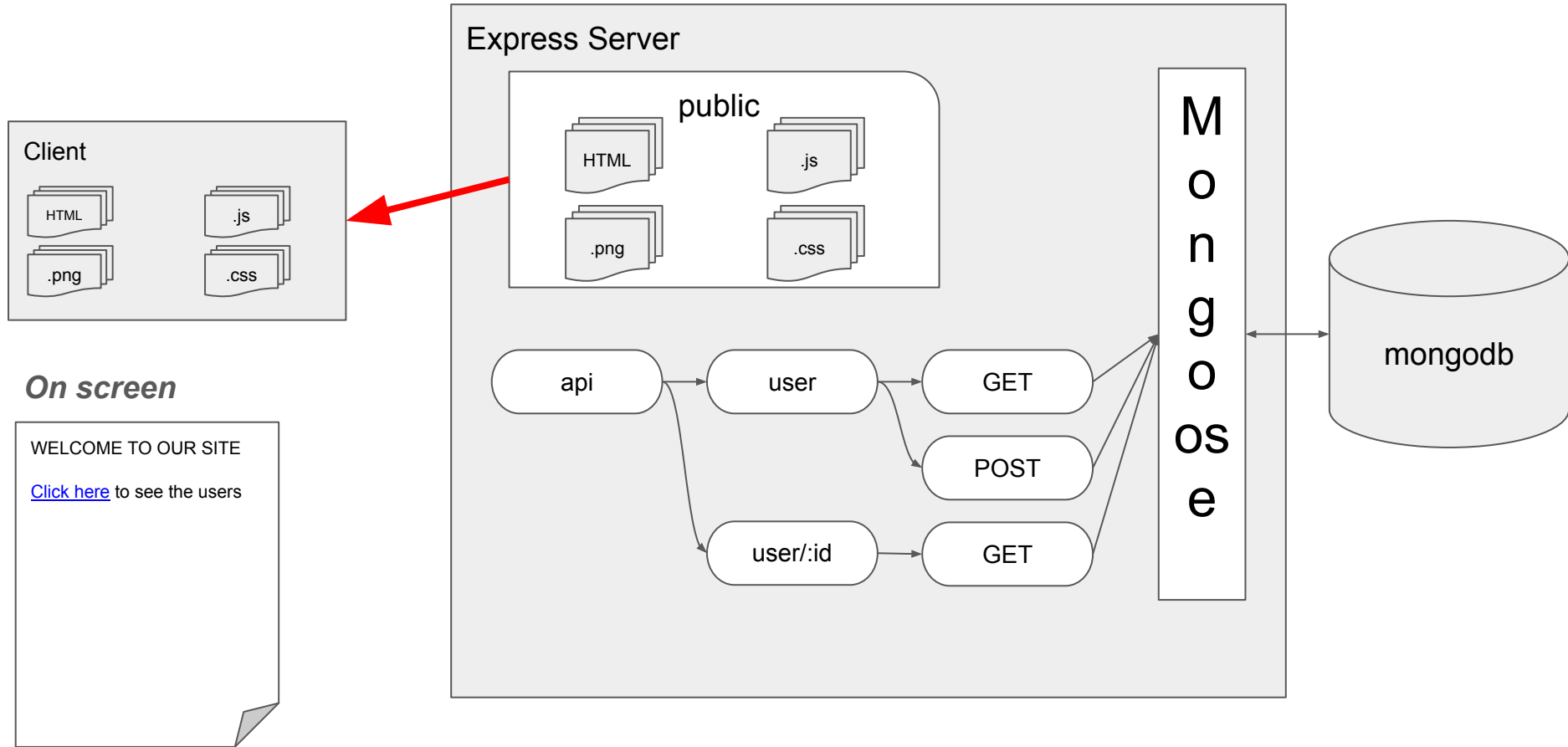
2. Client Requests `http://localhost:3000/users`



Single Page App

Client side rendered HTML using DOM manipulation

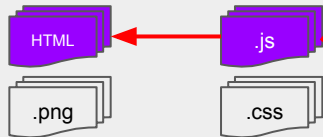
1. Client Loads Front End



2. Frontend JS requests list of users (still at http://localhost:3001)

Frontend JS updates the DOM with title and loading

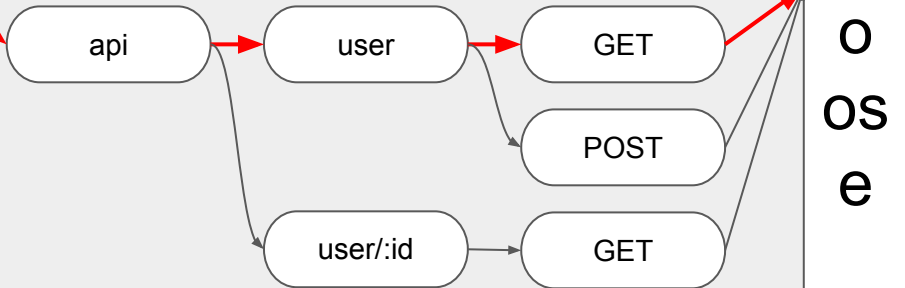
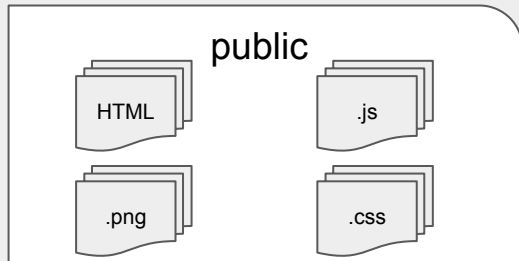
Client



On screen

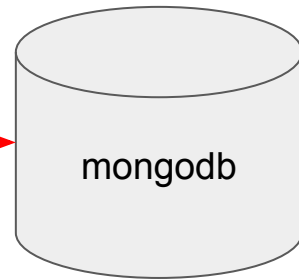
USER LIST
LOADING...

Express Server



M
o
n
g
o
o
s
e

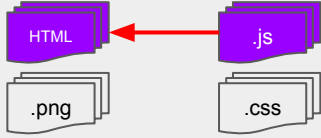
mongodb



3. Frontend JS updates the DOM, screen shows user list

User list HTML is rendered here on the client

Client

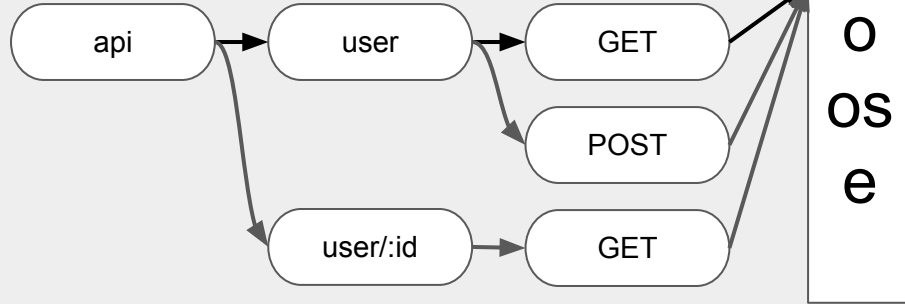
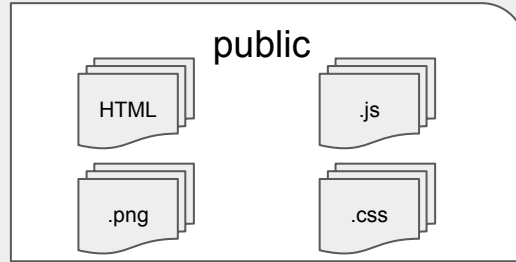


On screen

USER LIST

- 1 john@john.com
- 2 zach@zach.com
- 3 sally@sally.com
- 4 herman@herman.com

Express Server

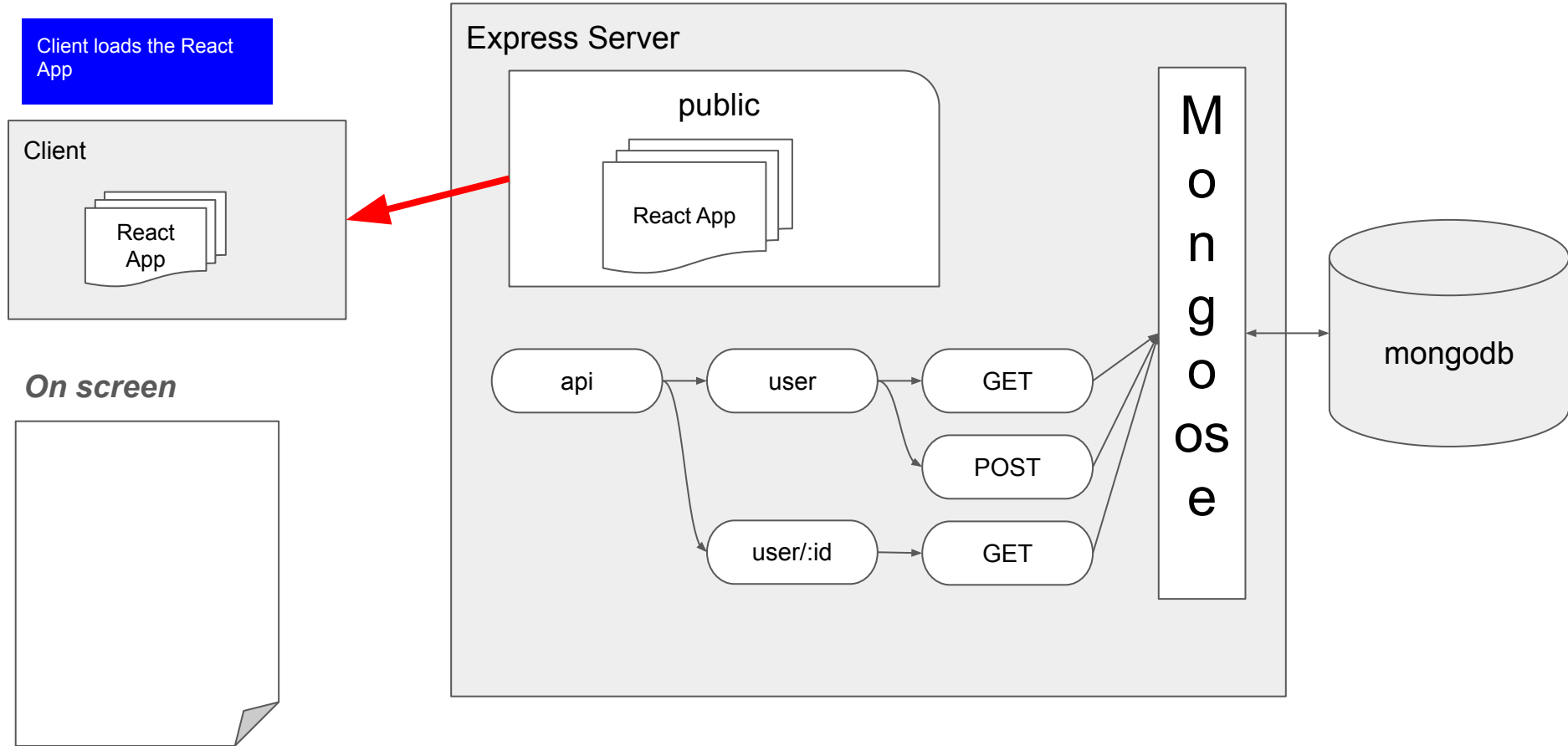


mongodb

Single Page App

Client side rendered React

1. Client Requests `http://localhost:3000/`



2. React renders Home page

ReactRouter renders the Home component because the path / points to Home

Client

React App

ReactRouter

Home

Home

Users

Profile

WELCOME TO OUR SITE

[Click here](#) to see the users

Express Server

public

React App

api

user

GET

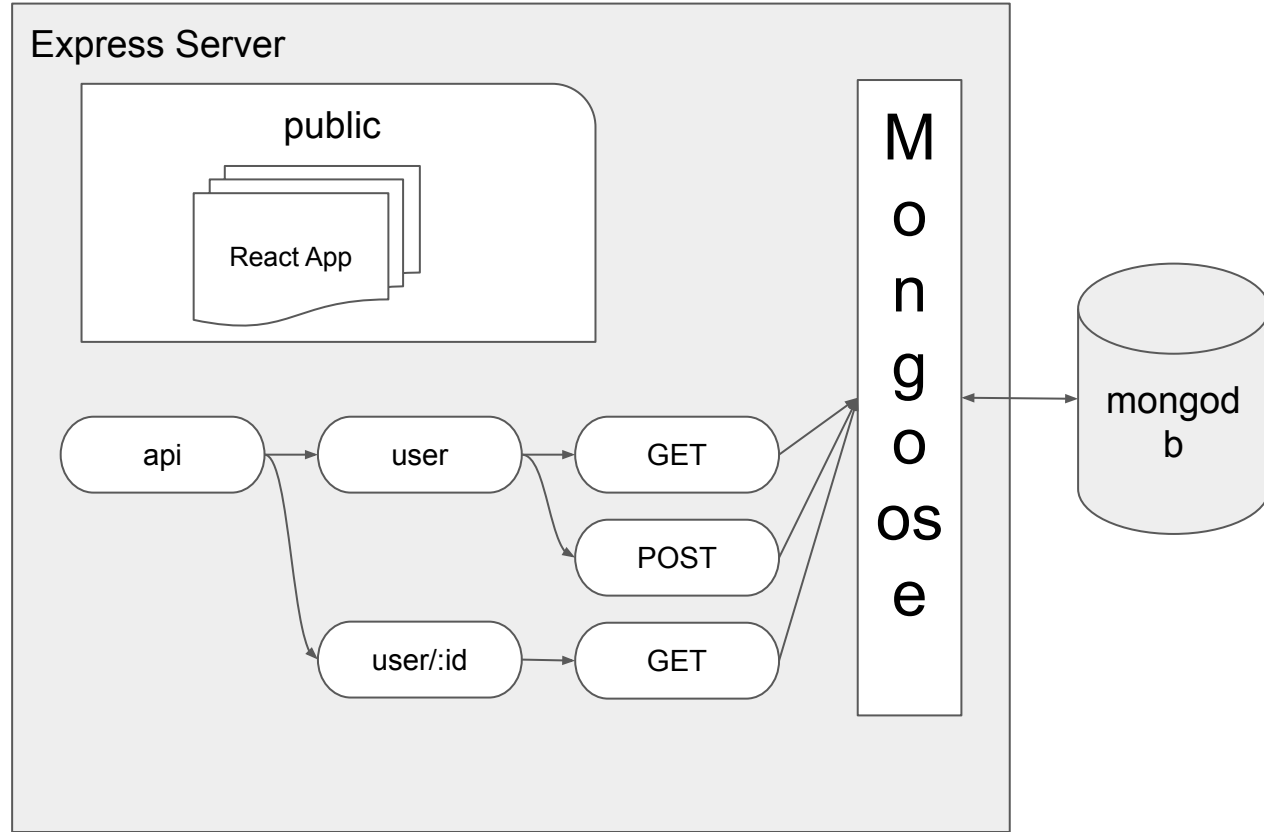
POST

user/:id

GET

M
o
n
g
o
o
s
e

mongod
b



2. Client Requests `http://localhost:3000/users`

ReactRouter renders the Users component because the path /users points to User

Client

React App

ReactRouter

Users

Home

Users

Profile

USER LIST

LOADING...

Express Server

public

React App

api

user

GET

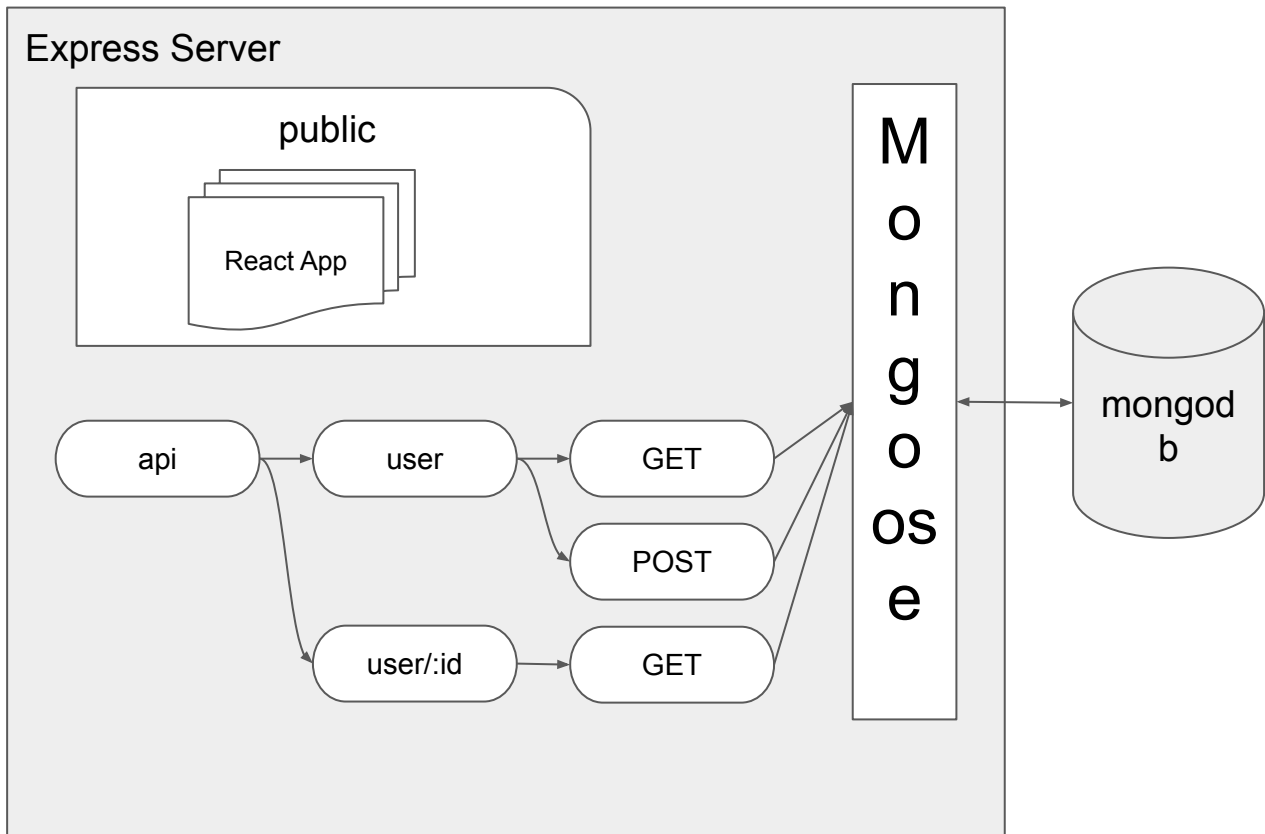
POST

user/:id

GET

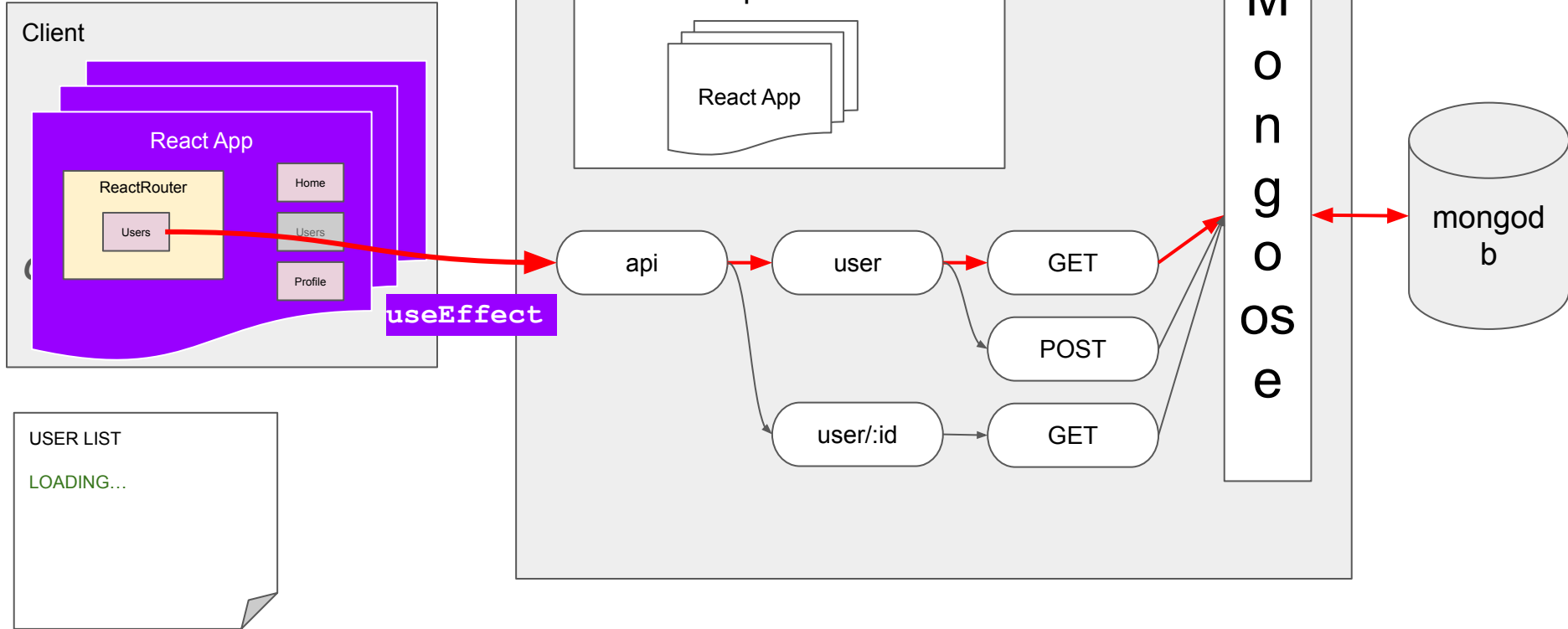
M
o
n
g
o
o
s
e

mongod
b



3. Users component useEffect requests the list of users and update state

ReactDOM renders the Users component because the path /users points to User



3. Users component re-renders because state has changed

ReactRouter renders the Users component because the path /users points to User

Client

React App

ReactRouter

Users

Home

Users

Profile

Express Server

public

React App

api

user

GET

POST

user/:id

GET

M
o
n
g
o
o
s
e

mongod
b

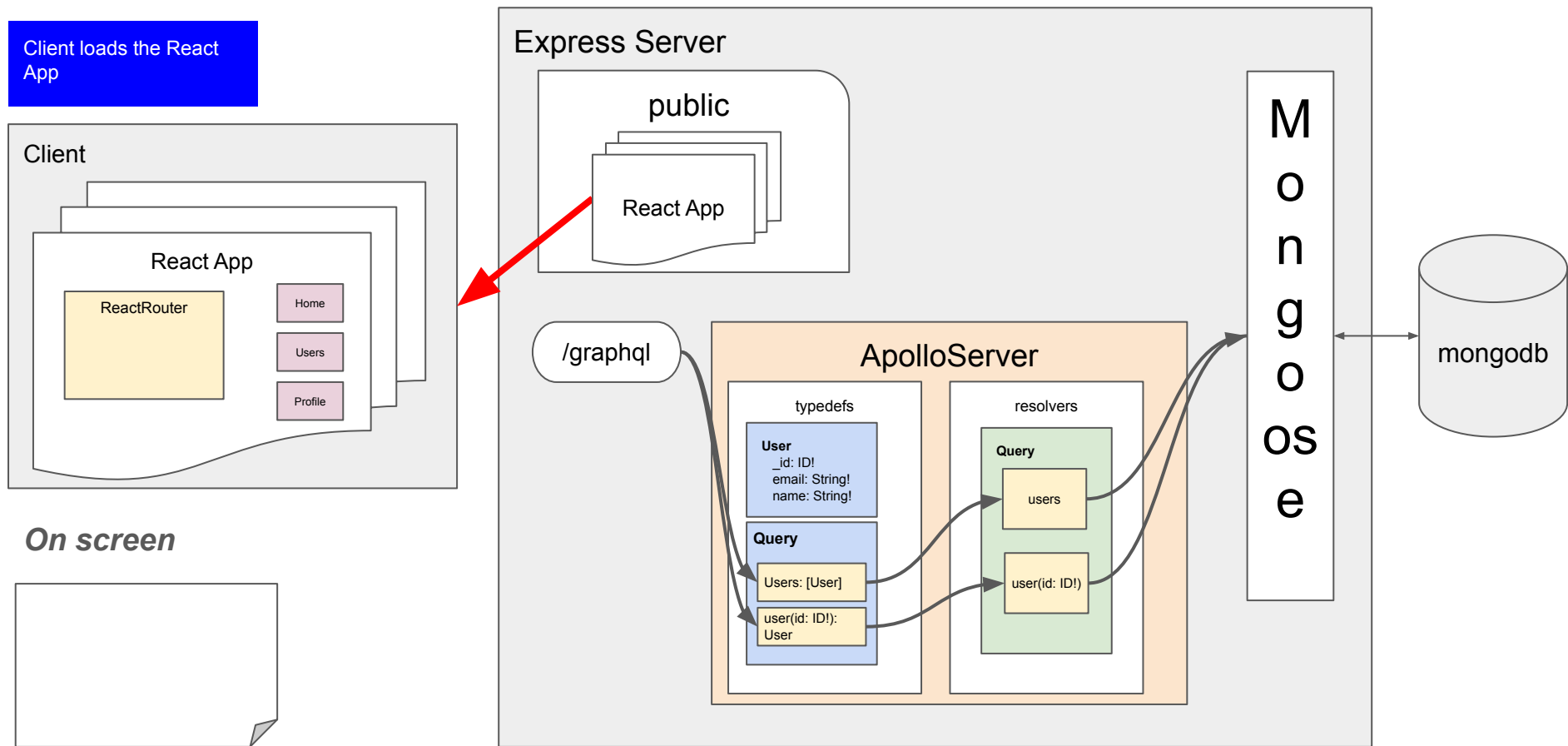
USER LIST

- 1 john@john.com
- 2 zach@zach.com
- 3 sally@sally.com
- 4 herman@herman.com

Single Page App

GraphQL

1. Client Requests `http://localhost:3000/`



2. React renders Home page

ReactRouter renders the Home component because the path / points to Home

Client

React App

ReactRouter

Home

Home

Users

Profile

On screen

WELCOME TO OUR SITE

[Click here](#) to see the users

Express Server

public

React App

/graphql

ApolloServer

typedefs

User
_id: ID!
email: String!
name: String!

Query

Users: [User]
user(id: ID!): User

resolvers

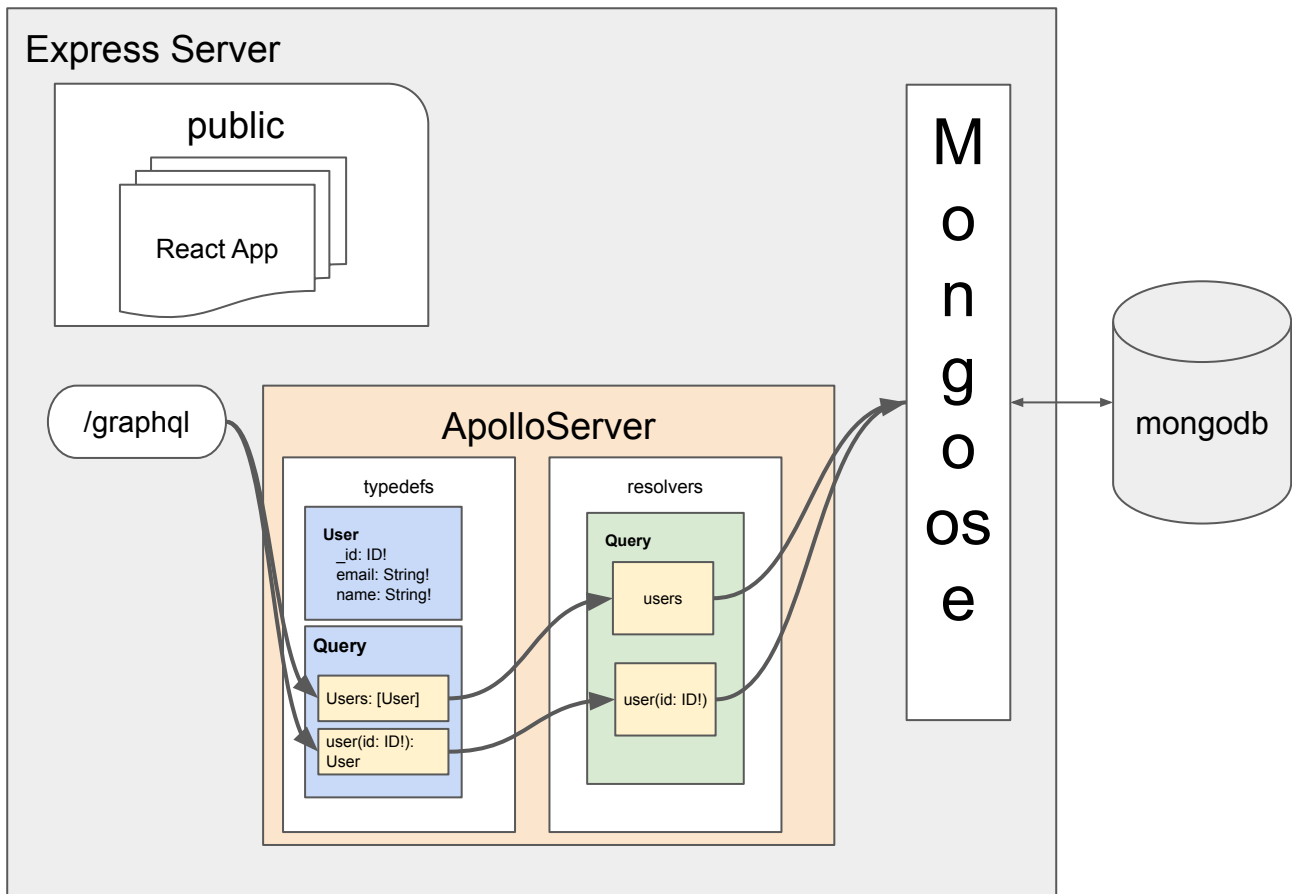
Query

users

user(id: ID!)

M
o
n
g
o
o
s
e

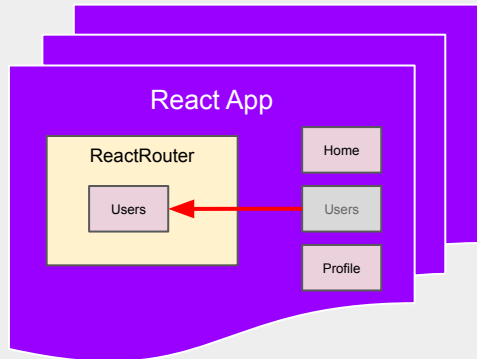
mongodb



3. Client Requests `http://localhost:3000/users`

ReactRouter renders the User component because the path /users points to User -

Client

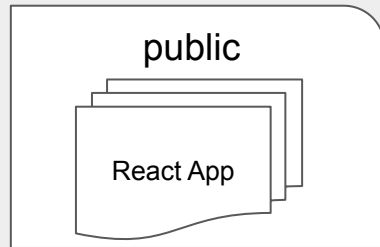


On screen

USER LIST

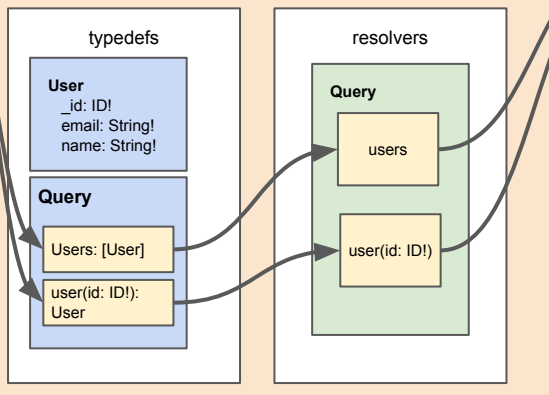
LOADING...

Express Server



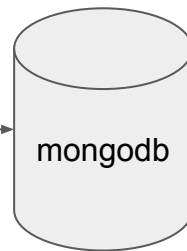
/graphql

ApolloServer

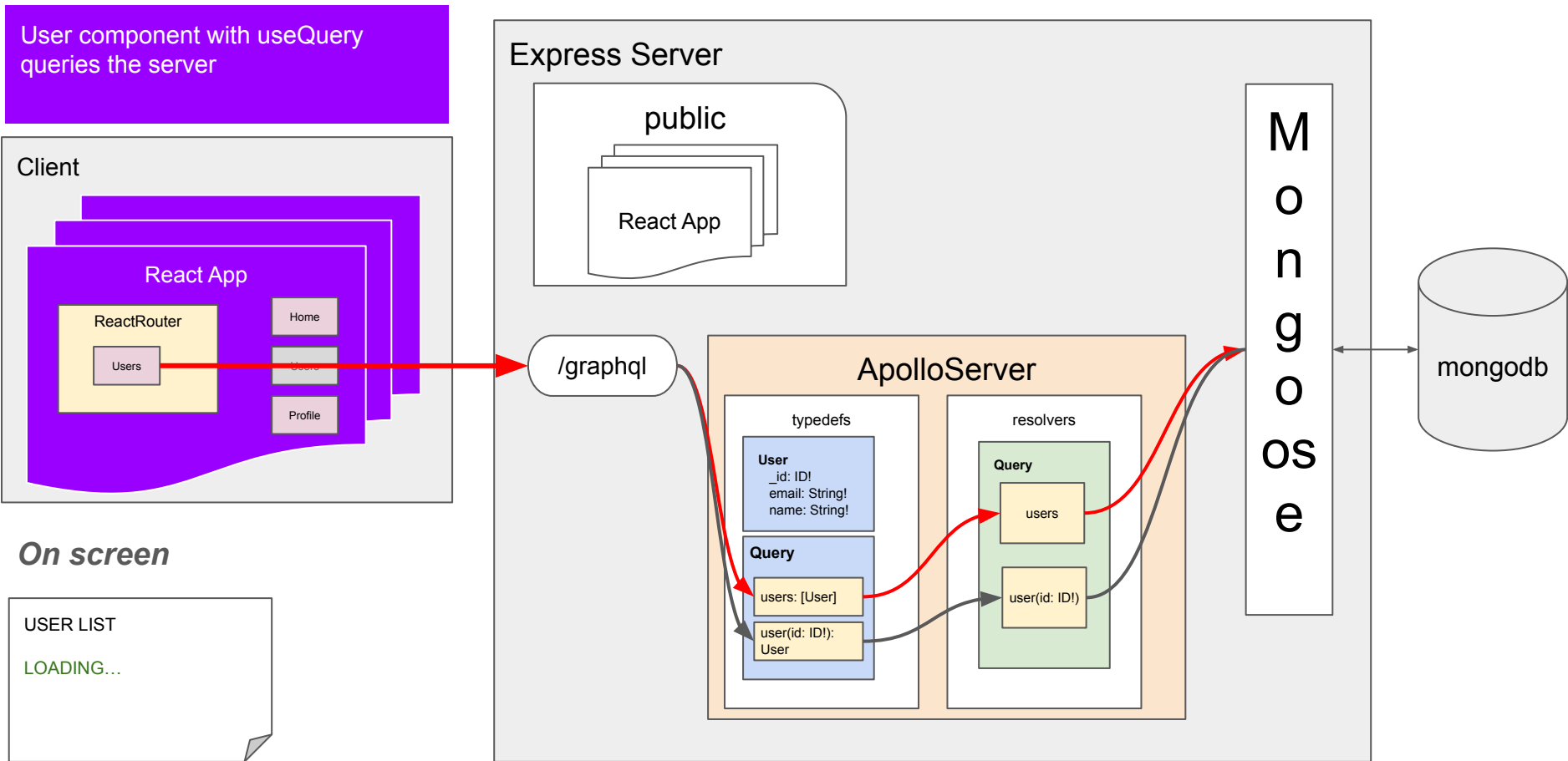


M
o
n
g
o
o
s
e

mongodb



4. Users component useQuery requests the list of users effectively updating state



5. User component re-renders

User component re-renders

Client

React App

ReactRouter

Users

Home

Users

Profile

On screen

USER LIST

- 1 john@john.com
- 2 zach@zach.com
- 3 sally@sally.com
- 4 herman@herman.com

Express Server

public

React App

/graphql

ApolloServer

typedefs

User
_id: ID!
email: String!
name: String!

Query

users: [User]
user(id: ID!):
User

resolvers

Query
users
user(id: ID!)

M
o
n
g
o
o
s
e

mongodb