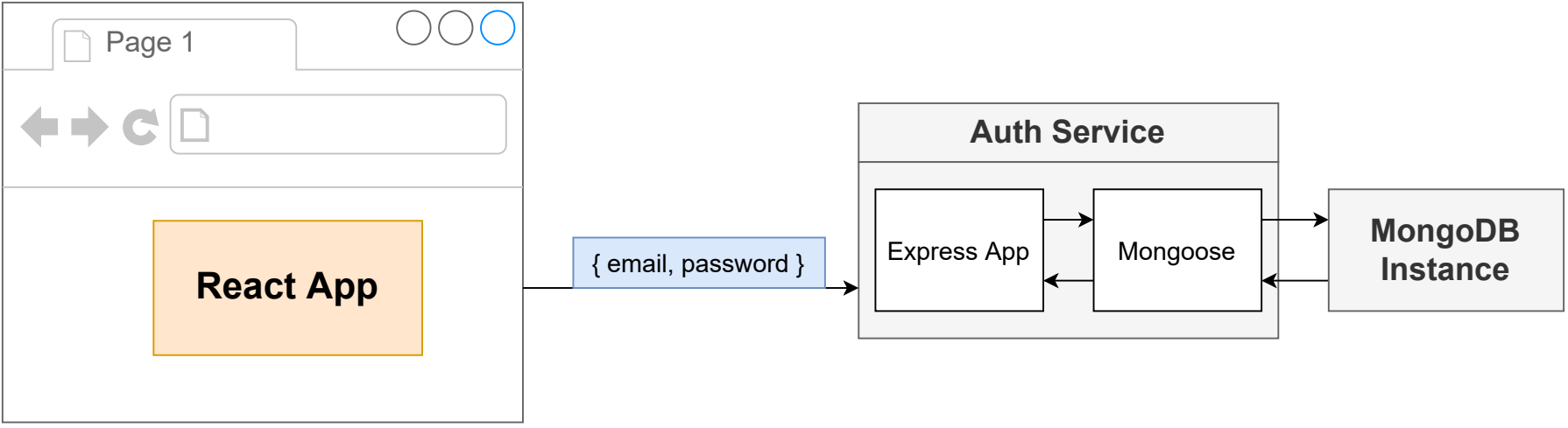
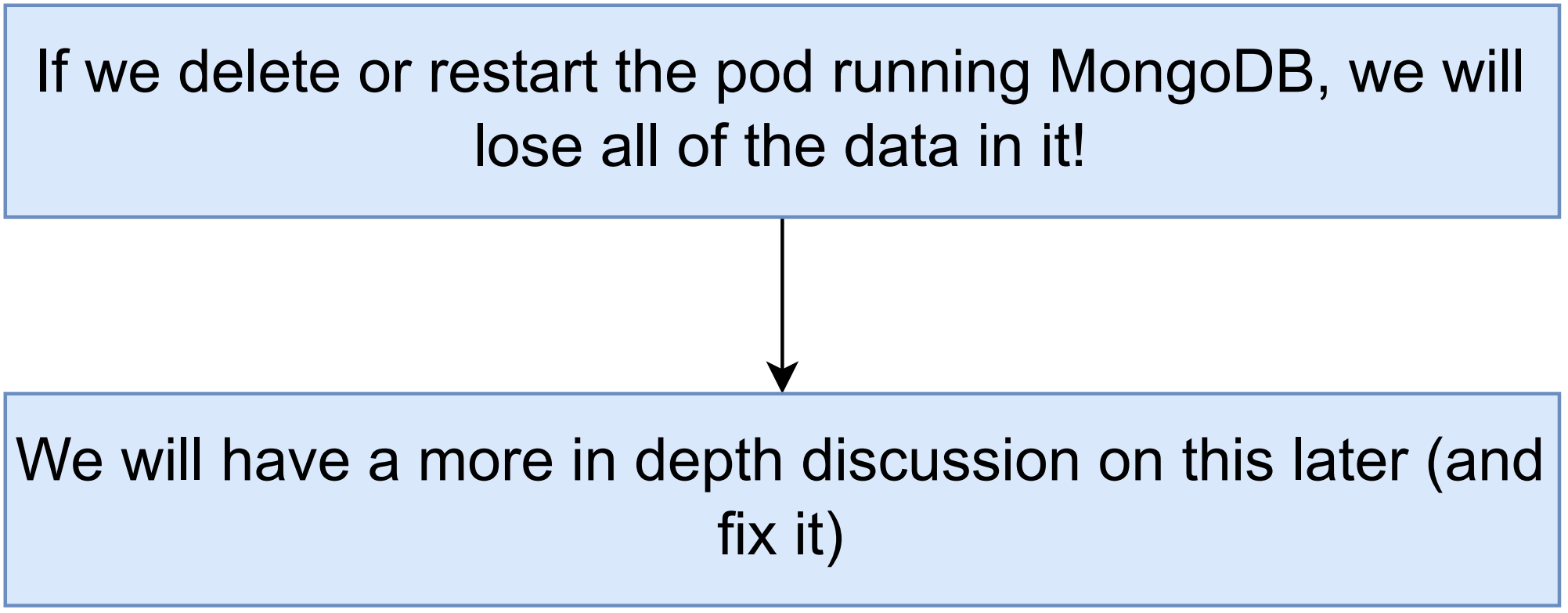
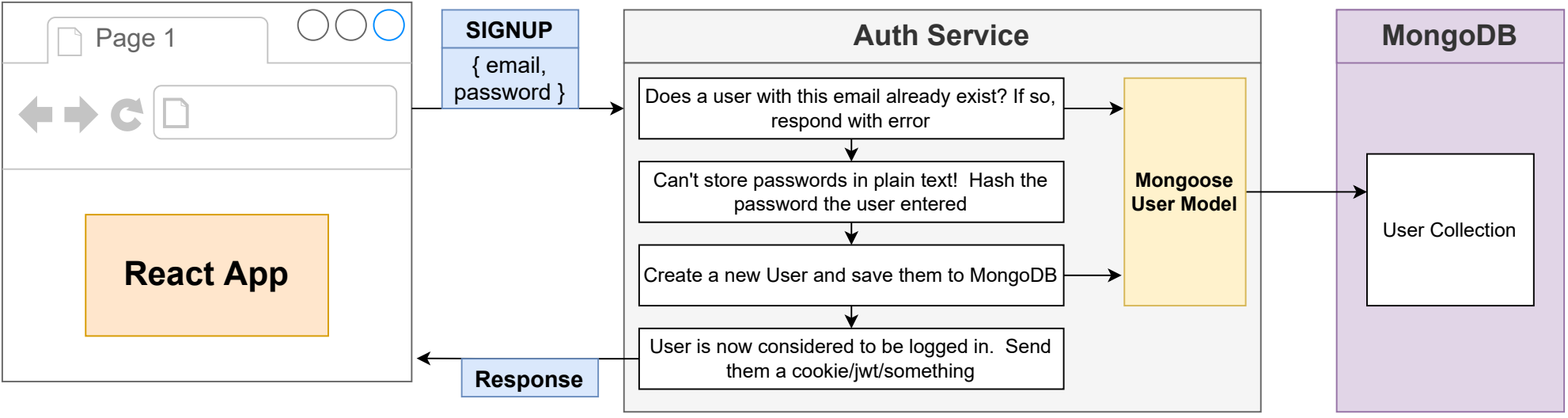
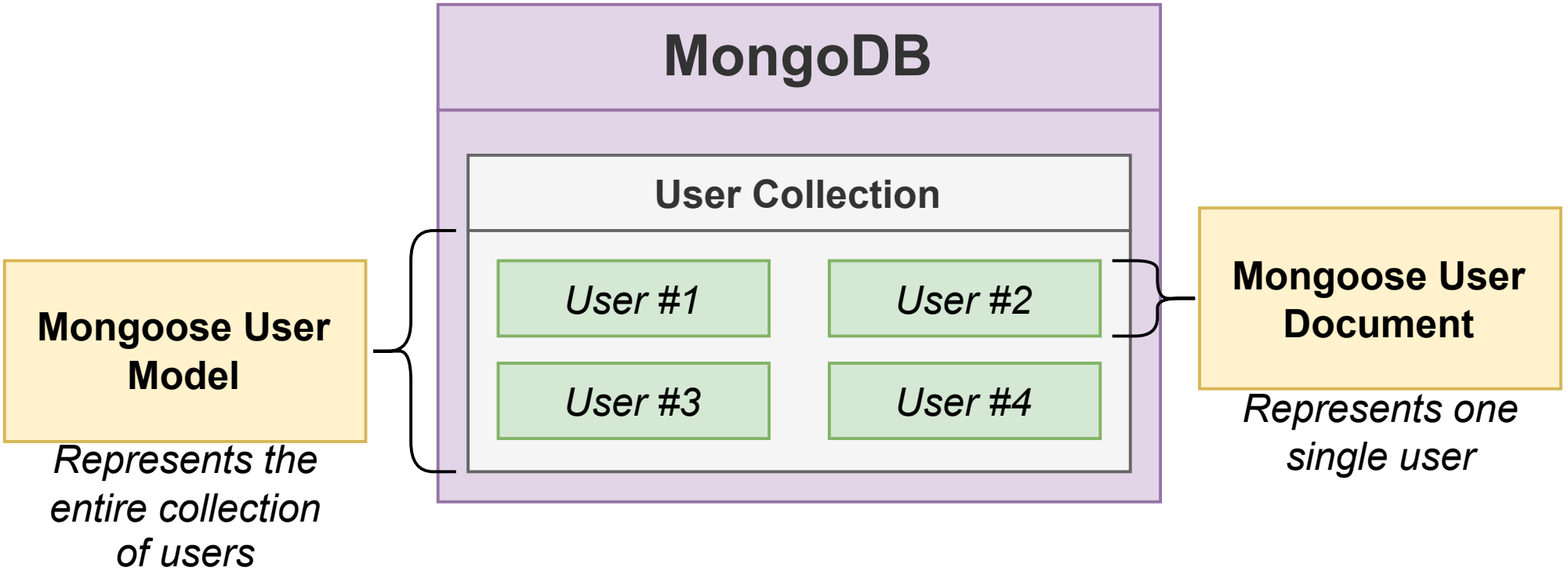

Page 1
React App
{ email, password }
Auth Service
Express App
Mongoose
MongoDB Instance


If we delete or restart the pod running MongoDB, we will lose all of the data in it!
We will have a more in depth discussion on this later (and fix it)


Page 1
React App
SIGNUP
{ email, password }
Response
Auth Service
Does a user with this email already exist? If so, respond with error
Can't store passwords in plain text! Hash the password the user entered
Create a new User and save them to MongoDB
User is now considered to be logged in. Send them a cookie/jwt/something
Mongoose User Model
MongoDB
User Collection


MongoDB
User Collection
User #1
User #2
User #3
User #4
Mongoose User Model
Represents the entire collection of users
Mongoose User Document
Represents one single user

Issue #1 with TS + Mongoose

Creating a new User Document

```
new User({ email: 'test@test.com', password: 'lk325kj2' })
```

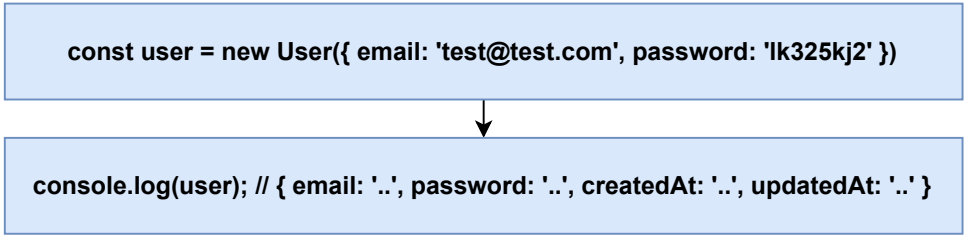
Issue #1 with TS + Mongoose

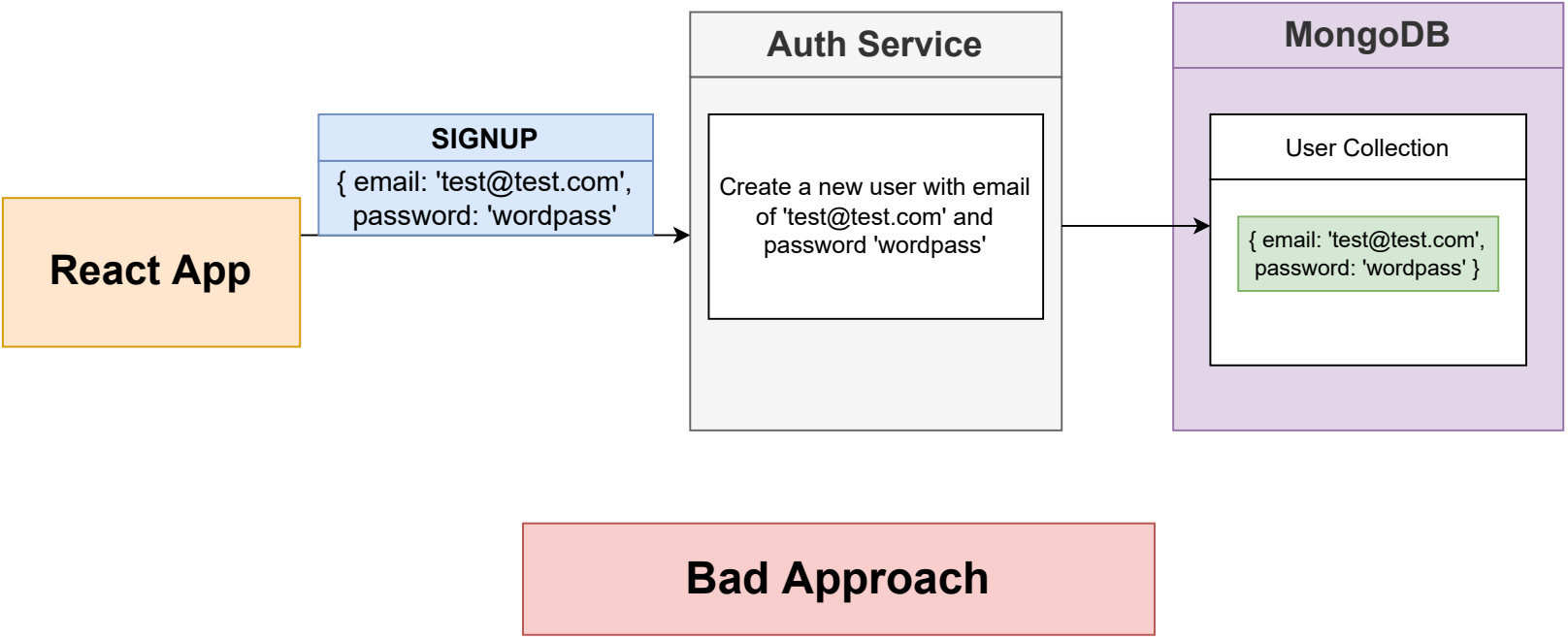
Creating a new User Document

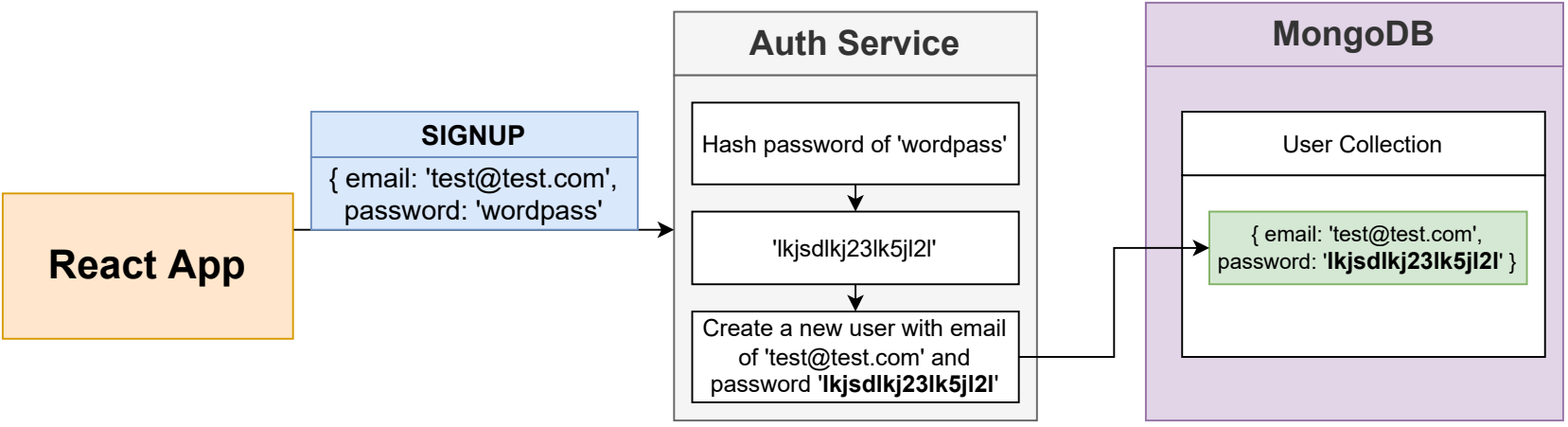
```
new User({ email: 'test@test.com', password: 'lk325kj2' })
```

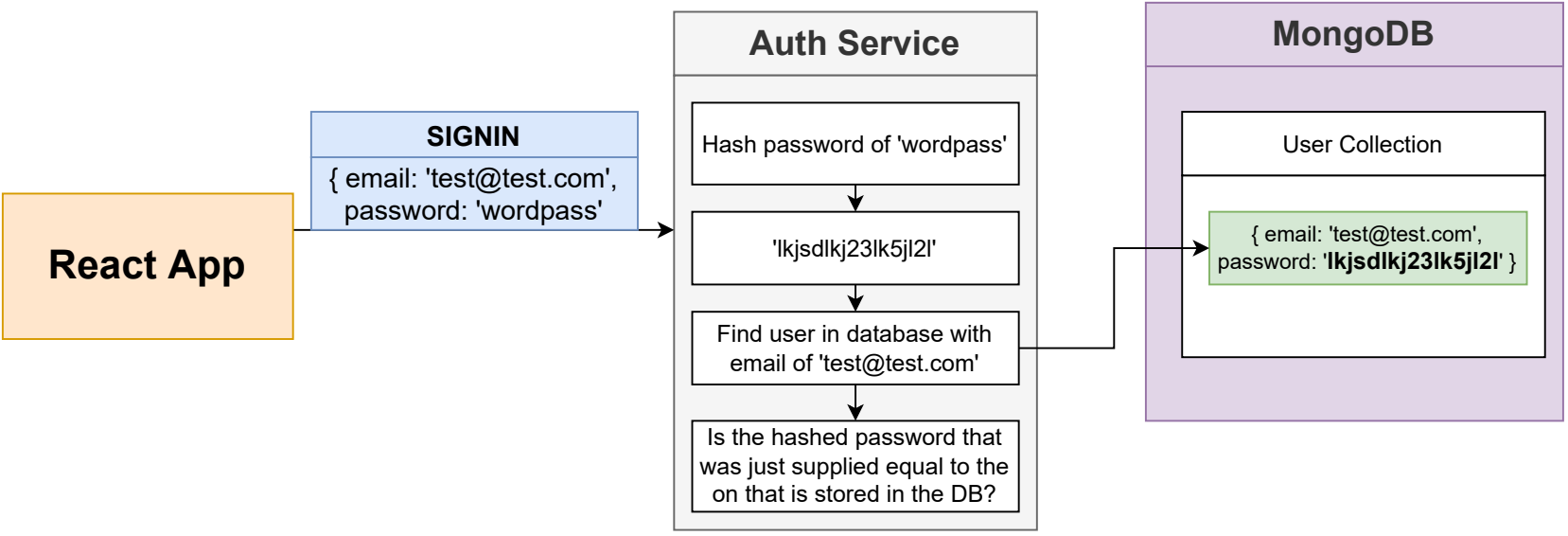
Typescript wants to make sure we are providing the correct properties - Mongoose does not make this easy!

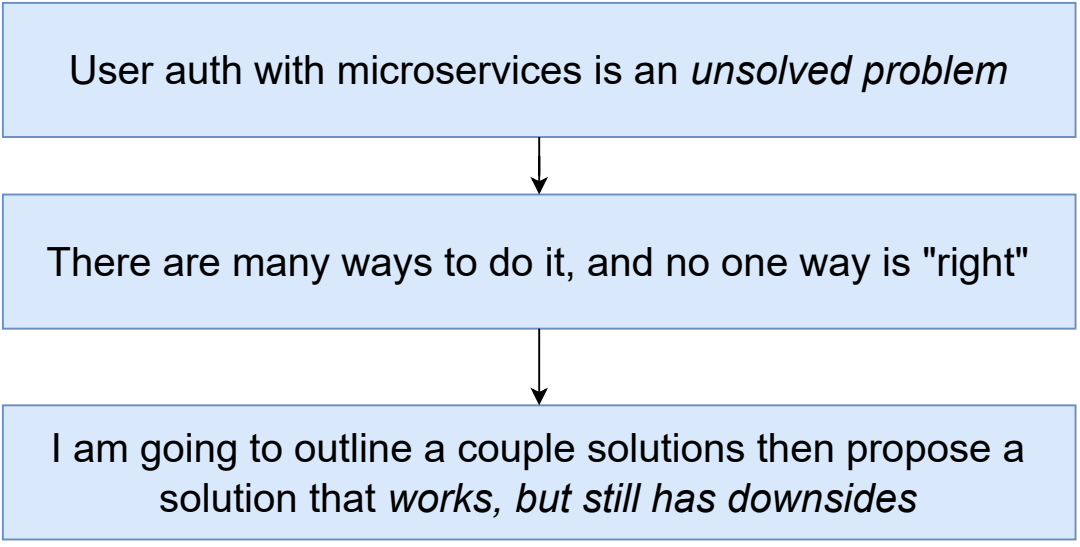
Issue #2 with TS + Mongoose

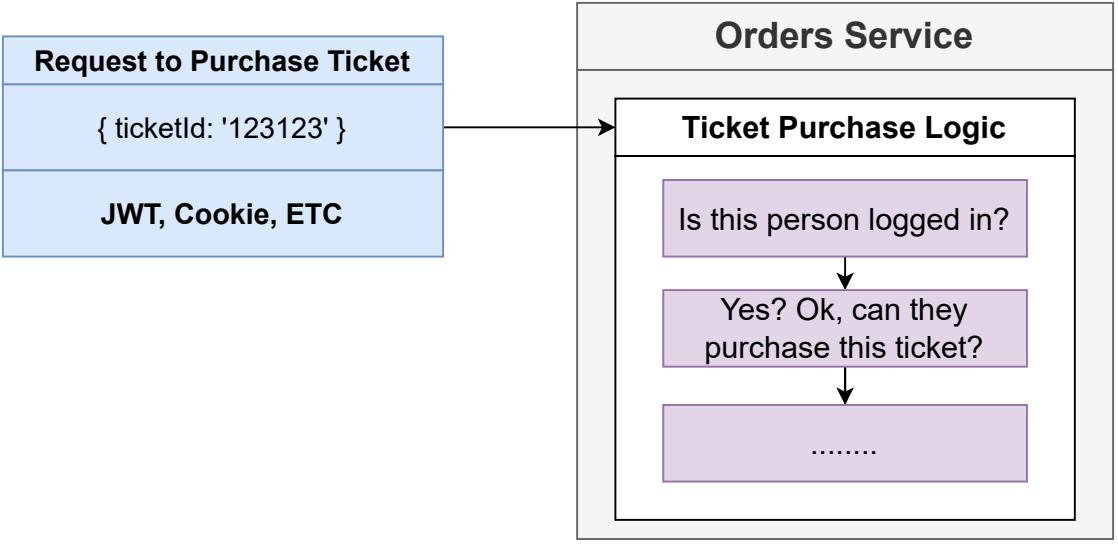



Auth Service
MongoDB
SIGNUP
{ email: 'test@test.com',
password: 'wordpass'
React App
Create a new user with email of 'test@test.com' and password 'wordpass'
User Collection
{ email: 'test@test.com',
password: 'wordpass' }
Bad Approach


React App
SIGNUP
{ email: 'test@test.com', password: 'wordpass'
Auth Service
Hash password of 'wordpass'
'lkjsdlkj23lk5jl2l'
Create a new user with email of 'test@test.com' and password 'lkjsdlkj23lk5jl2l'
MongoDB
User Collection
{ email: 'test@test.com', password: 'lkjsdlkj23lk5jl2l' }

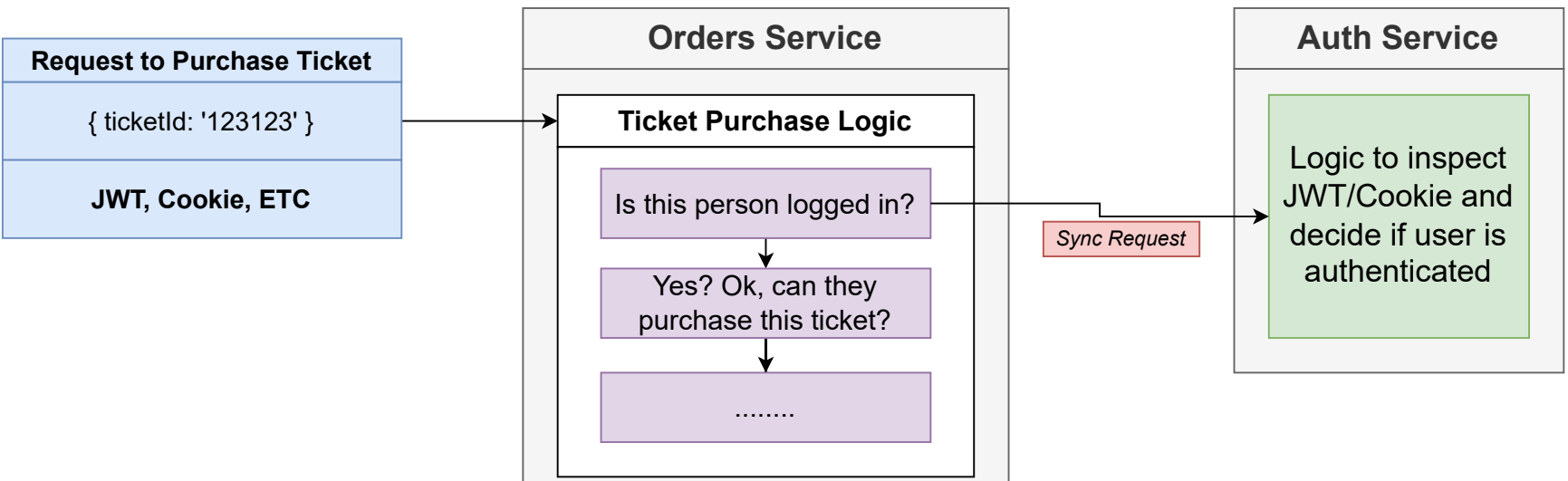

React App
SIGNIN
{ email: 'test@test.com', password: 'wordpass'
Auth Service
Hash password of 'wordpass'
'lkjsdlkj23lk5jl2l'
Find user in database with email of 'test@test.com'
Is the hashed password that was just supplied equal to the on that is stored in the DB?
MongoDB
User Collection
{ email: 'test@test.com', password: 'lkjsdlkj23lk5jl2l' }


User auth with microservices is an *unsolved problem*
There are many ways to do it, and no one way is "right"
I am going to outline a couple solutions then propose a solution that *works, but still has downsides*


Request to Purchase Ticket
{ ticketId: '123123' }
JWT, Cookie, ETC
Orders Service
Ticket Purchase Logic
Is this person logged in?
Yes? Ok, can they purchase this ticket?
.....

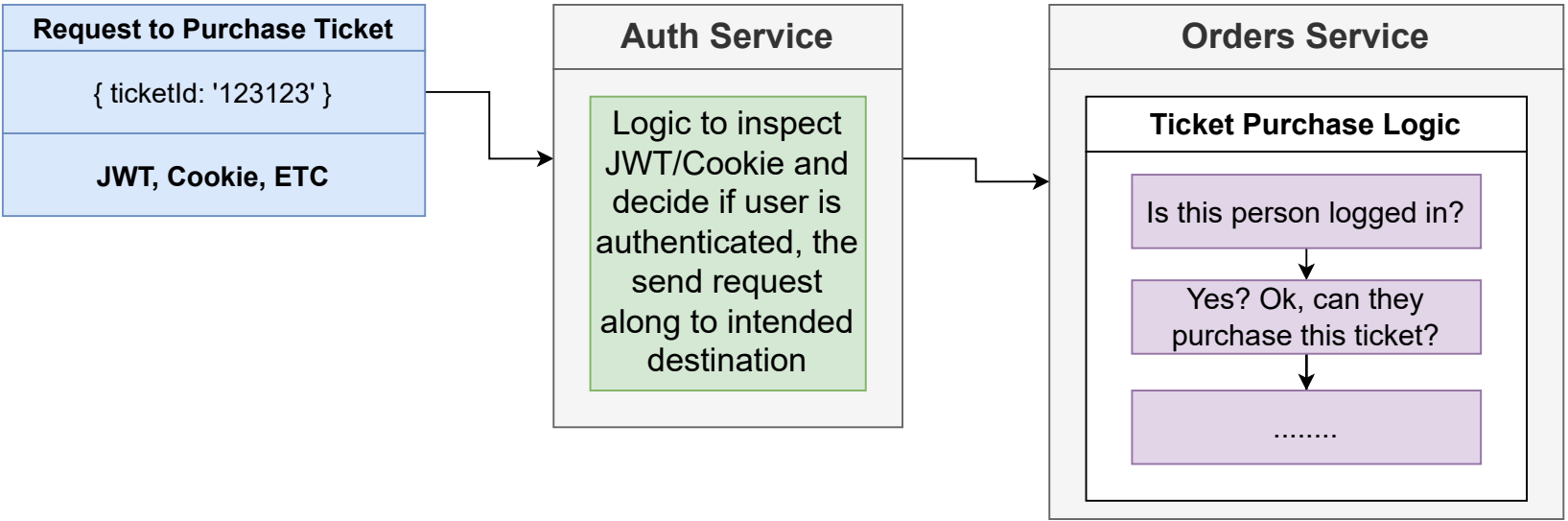
Fundamental Option #1

Individual services rely on the auth service



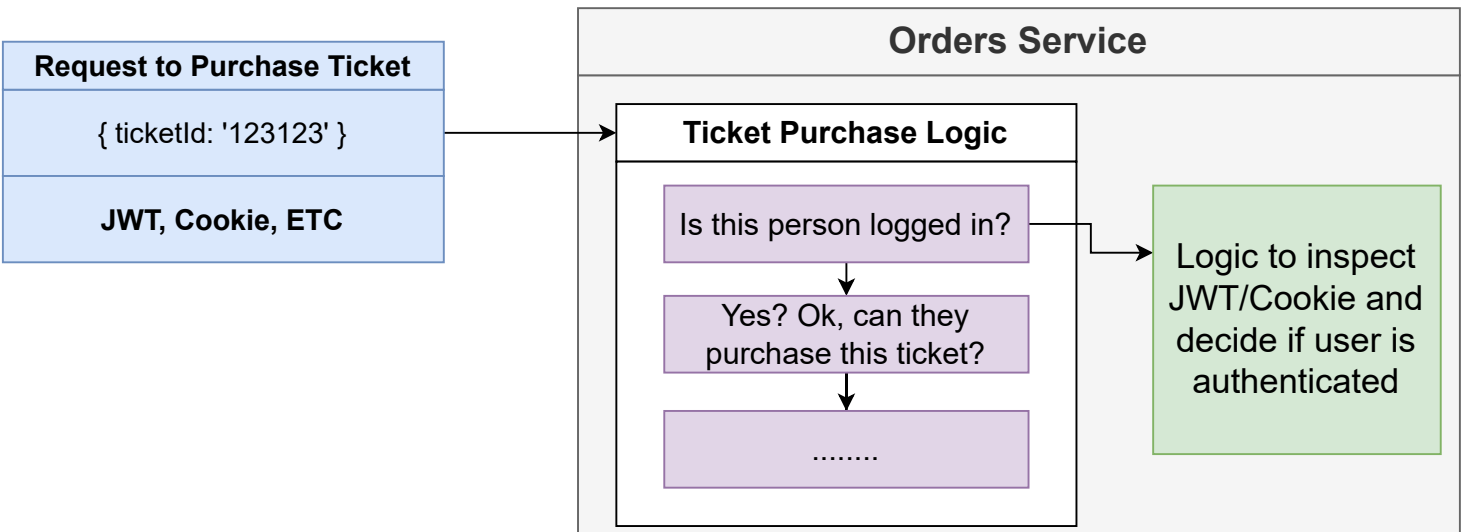
Fundamental Option #1.1

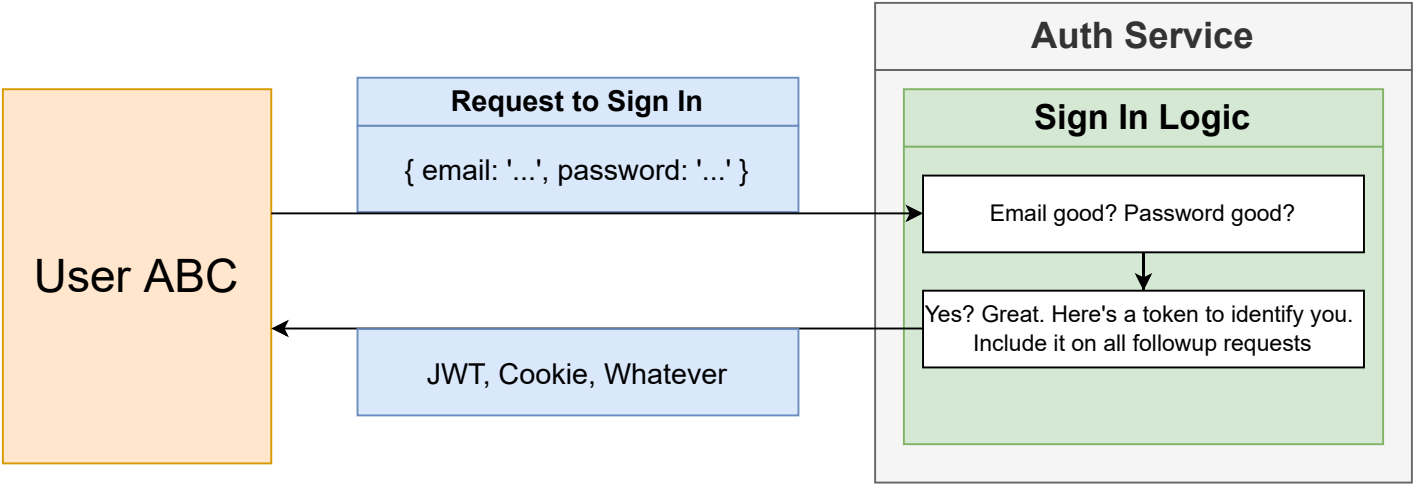
Individual services rely on the auth service as a gateway

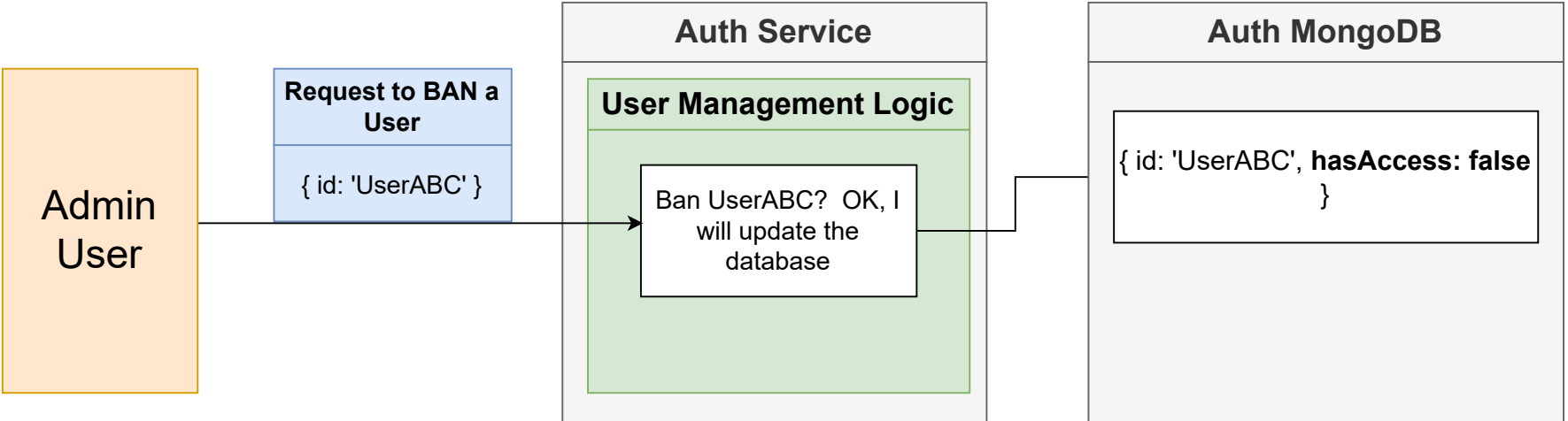


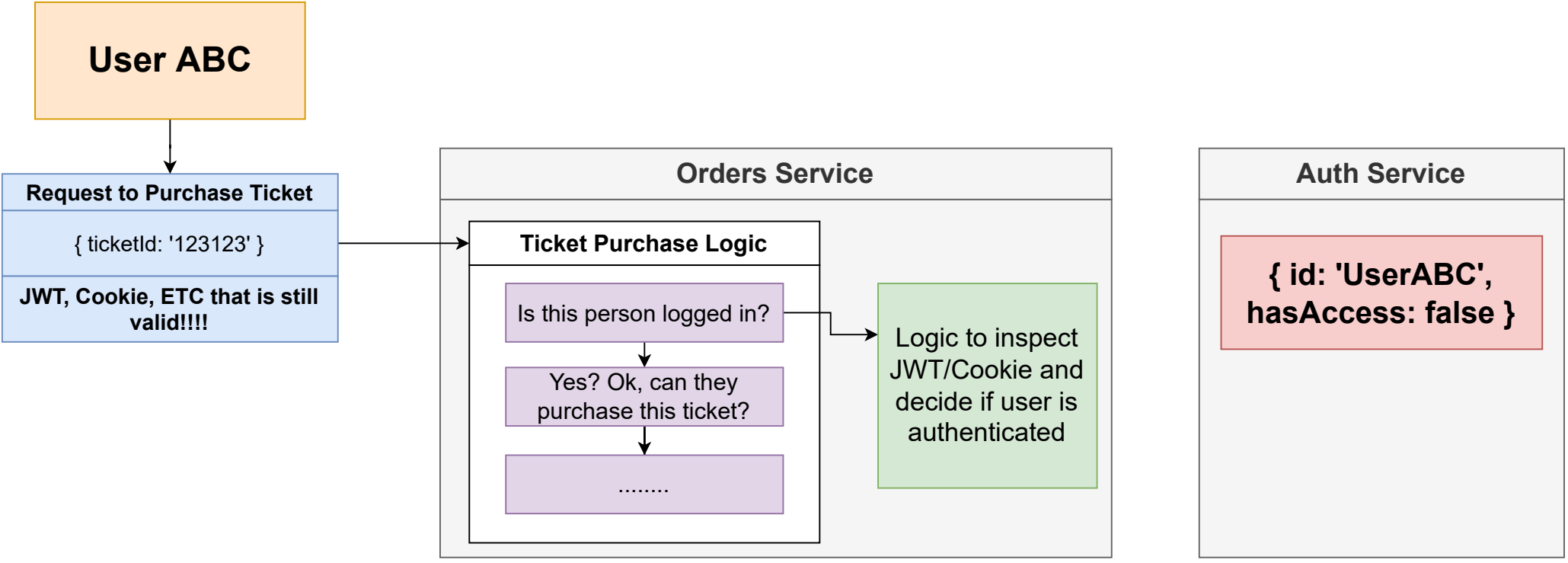
Fundamental Option #2

Individual services know how to authenticate a user




Auth Service
Sign In Logic
Email good? Password good?
Yes? Great. Here's a token to identify you.
Include it on all followup requests
Request to Sign In
{ email: '...', password: '...' }
JWT, Cookie, Whatever
User ABC


Admin User
Request to BAN a User
{ id: 'UserABC' }
Auth Service
User Management Logic
Ban UserABC? OK, I will update the database
Auth MongoDB
{ id: 'UserABC', hasAccess: false }


User ABC
Request to Purchase Ticket
{ ticketId: '123123' }
JWT, Cookie, ETC that is still valid!!!!
Orders Service
Ticket Purchase Logic
Is this person logged in?
Yes? Ok, can they purchase this ticket?
.....
Logic to inspect JWT/Cookie and decide if user is authenticated
Auth Service
{ id: 'UserABC', hasAccess: false }

Fundamental Option #1

Individual services rely on the auth service

Changes to auth state are immediately reflected

Auth service goes down? Entire app is broken

Fundamental Option #2

Individual services know how to authenticate a user

Auth service is down? Who cares!

Some user got banned? Darn, I just gave them the keys to my car 5 minutes ago...

Fundamental Option #1

Individual services rely on the auth service

Changes to auth state are immediately reflected

Auth service goes down? Entire app is broken

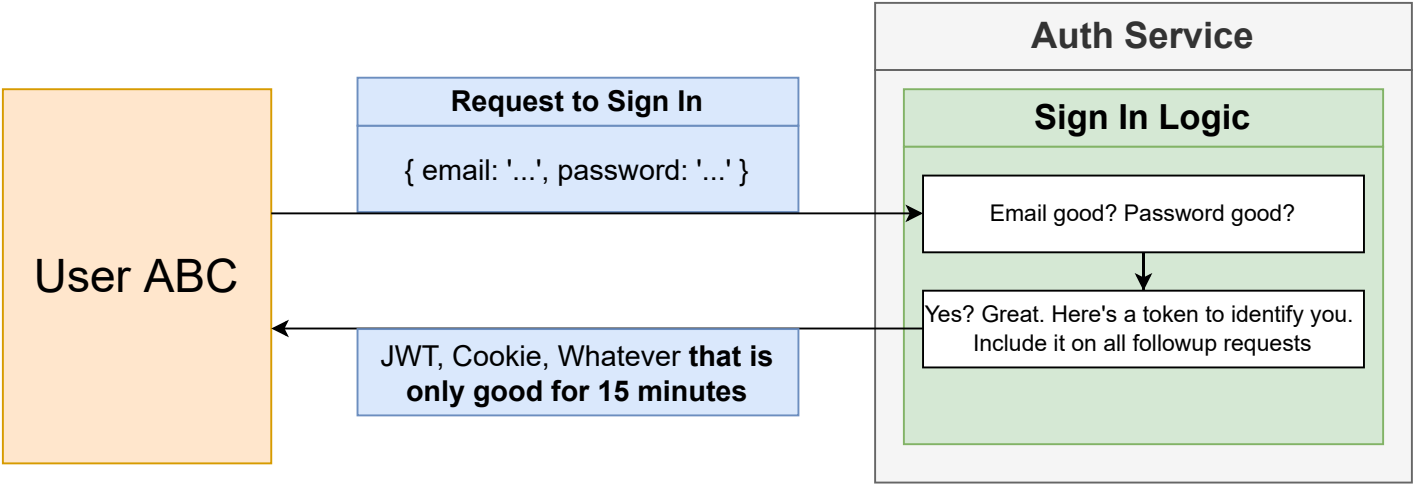
Fundamental Option #2

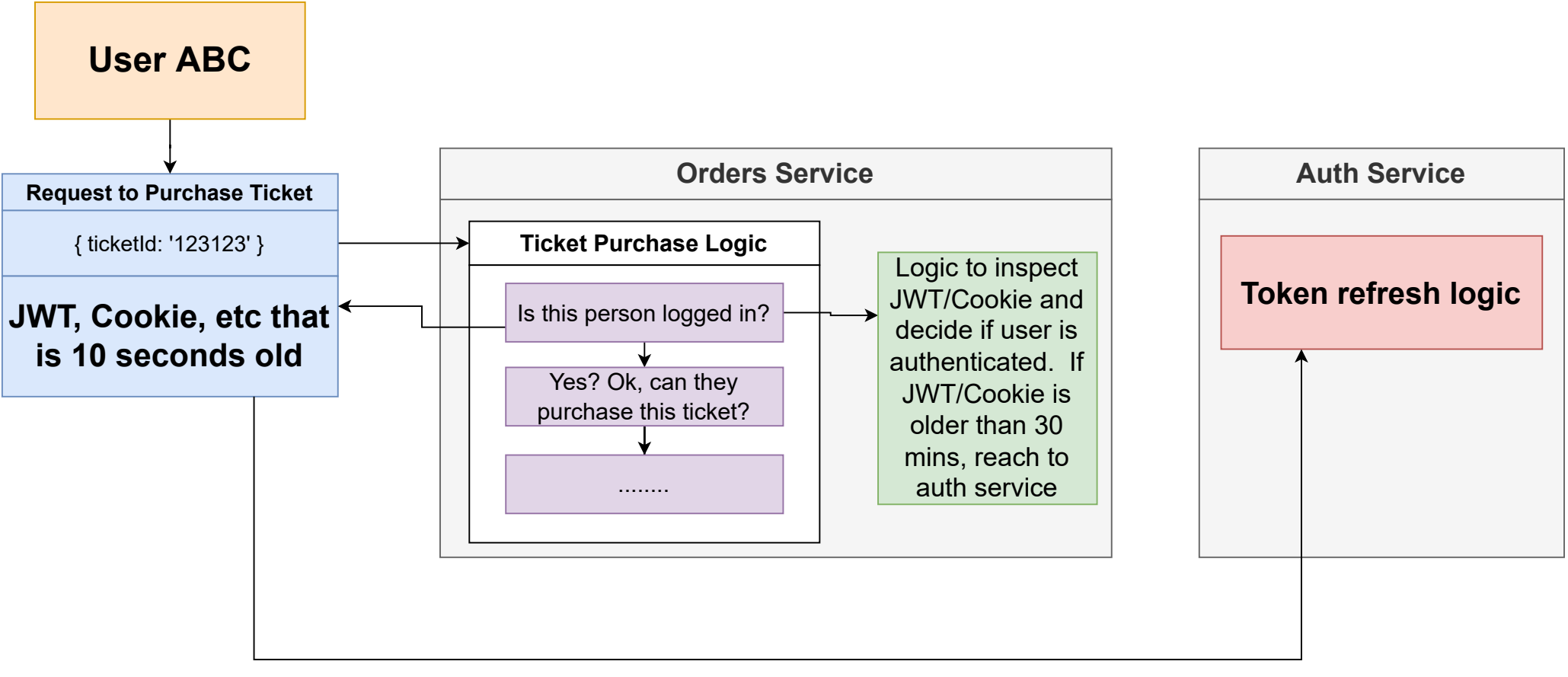
Individual services know how to authenticate a user

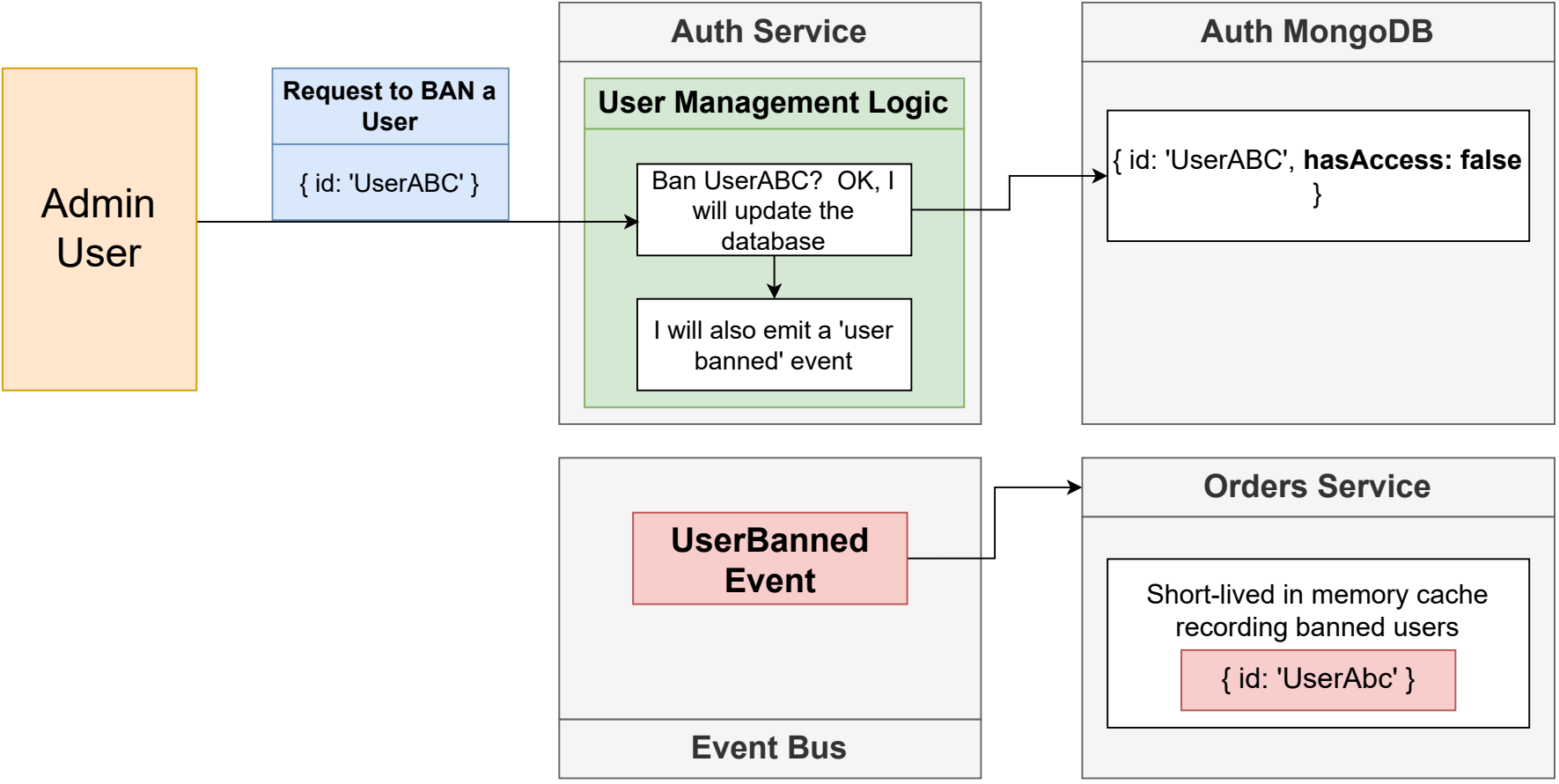
Auth service is down? Who cares!

Some user got banned? Darn, I just gave them the keys to my car 5 minutes ago...

We are going with Option #2 to stick with the idea of independent services


Auth Service
Sign In Logic
Email good? Password good?
Yes? Great. Here's a token to identify you.
Include it on all followup requests
User ABC
Request to Sign In
{ email: '...', password: '...' }
JWT, Cookie, Whatever that is only good for 15 minutes


User ABC
Request to Purchase Ticket
{ ticketId: '123123' }
JWT, Cookie, etc that is 10 seconds old
Orders Service
Ticket Purchase Logic
Is this person logged in?
Yes? Ok, can they purchase this ticket?
.....
Logic to inspect JWT/Cookie and decide if user is authenticated. If JWT/Cookie is older than 30 mins, reach to auth service
Auth Service
Token refresh logic


Admin User
Request to BAN a User
{ id: 'UserABC' }
Auth Service
User Management Logic
Ban UserABC? OK, I will update the database
I will also emit a 'user banned' event
Auth MongoDB
{ id: 'UserABC', hasAccess: false }
UserBanned Event
Event Bus
Orders Service
Short-lived in memory cache recording banned users
{ id: 'UserAbc' }