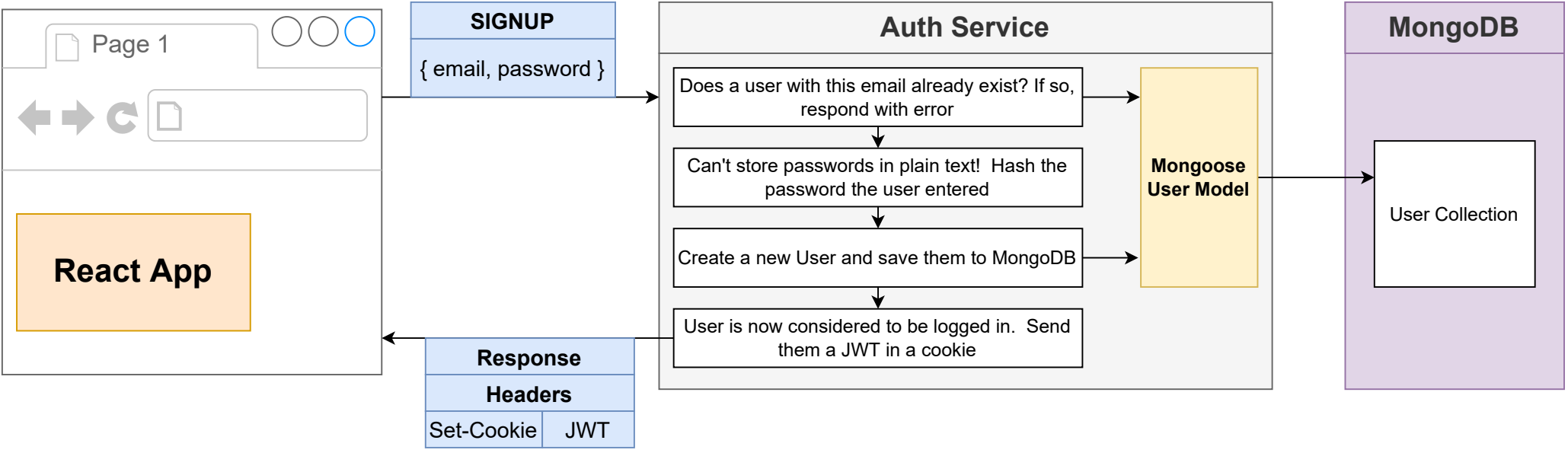

Page 1
React App
SIGNUP
{ email, password }
Auth Service
Does a user with this email already exist? If so, respond with error
Can't store passwords in plain text! Hash the password the user entered
Create a new User and save them to MongoDB
User is now considered to be logged in. Send them a JWT in a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

Requirements for Our Auth Mechanism

Must be able to tell us details about a user

Must be able to handle authorization info

Must have a built-in, tamper-resistant way to expire or invalidate itself

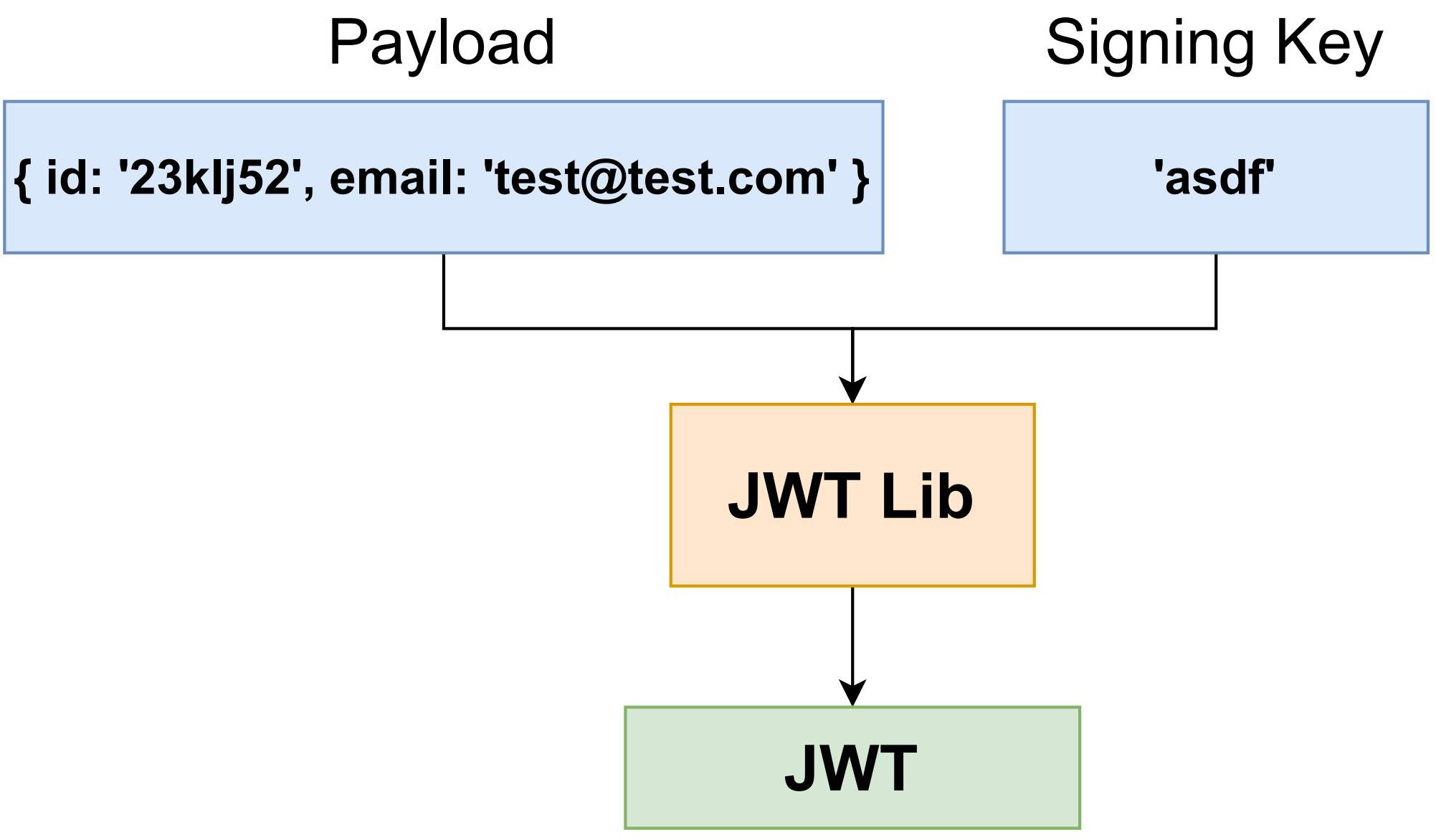
Must be easily understood between different languages → **Cookie handling across languages is usually an issue when we *encrypt* the data in the cookie**

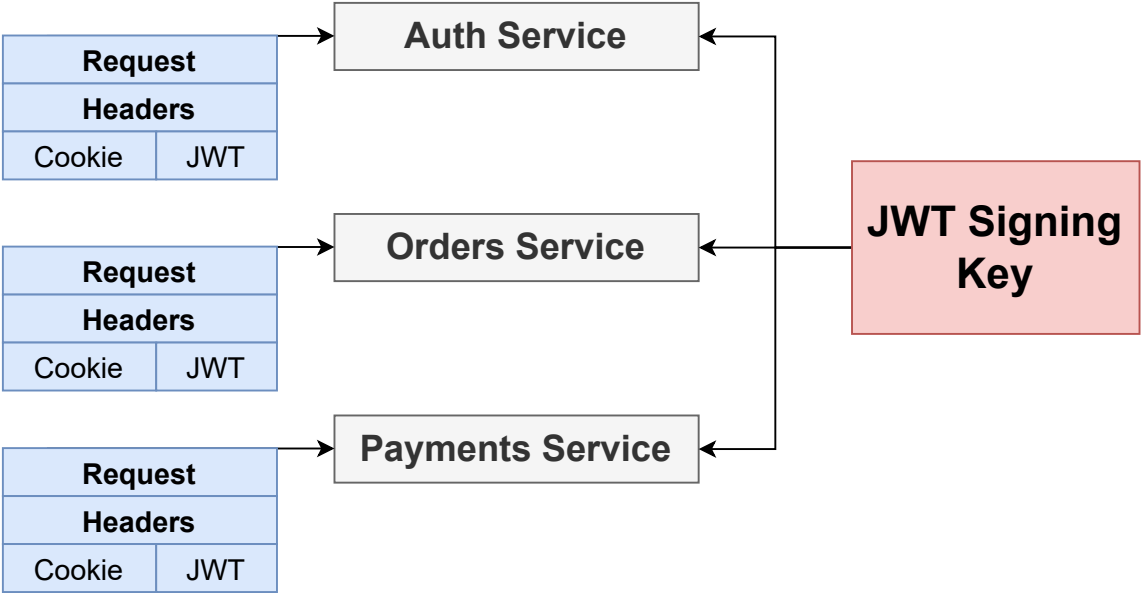
Must not require some kind of backing data store on the server

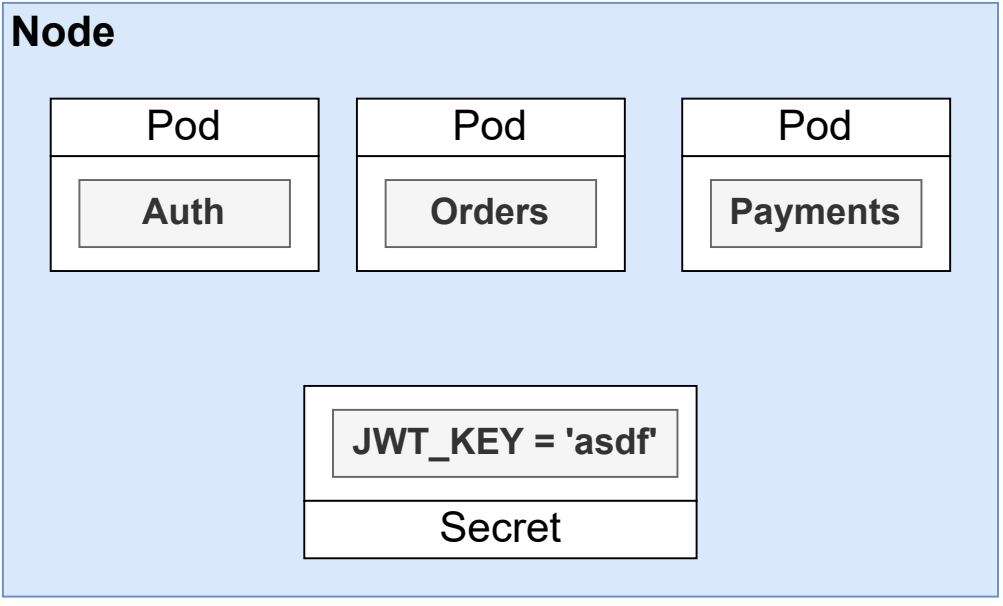
We will not encrypt the cookie contents.

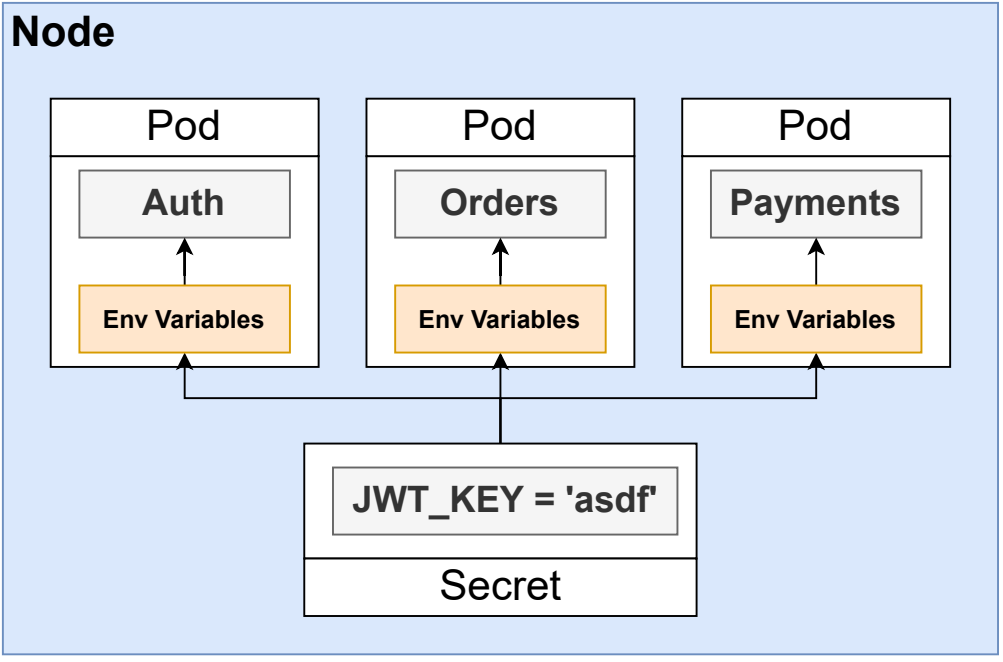
Remember, JWT's are tamper resistant

You can encrypt the cookie contents if this is a big deal to you


Payload
Signing Key
{ id: '23klj52', email: 'test@test.com' }
'asdf'
JWT Lib
JWT

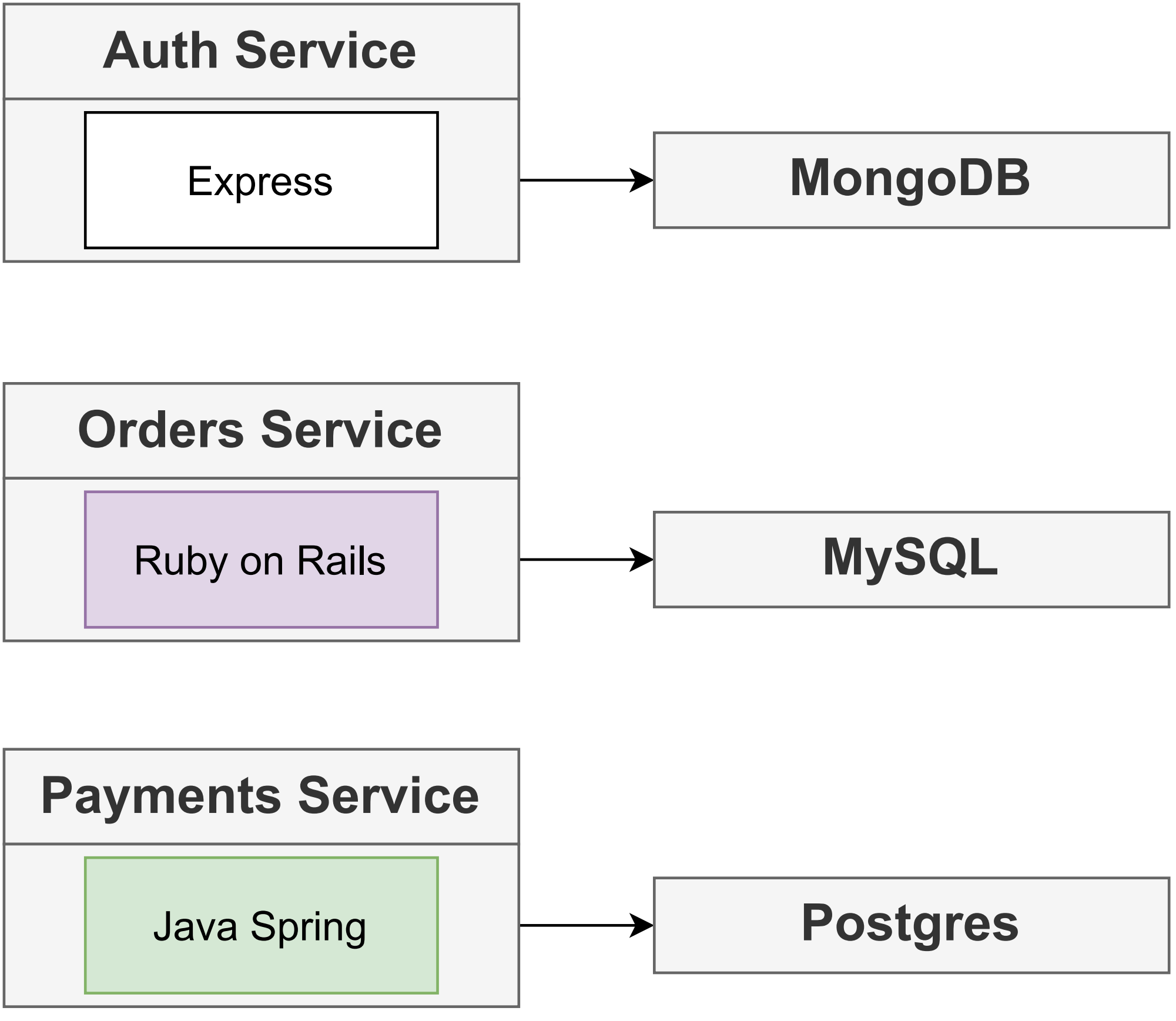

Request
Headers
Cookie
JWT
Auth Service
Request
Headers
Cookie
JWT
Orders Service
Request
Headers
Cookie
JWT
Payments Service
JWT Signing Key

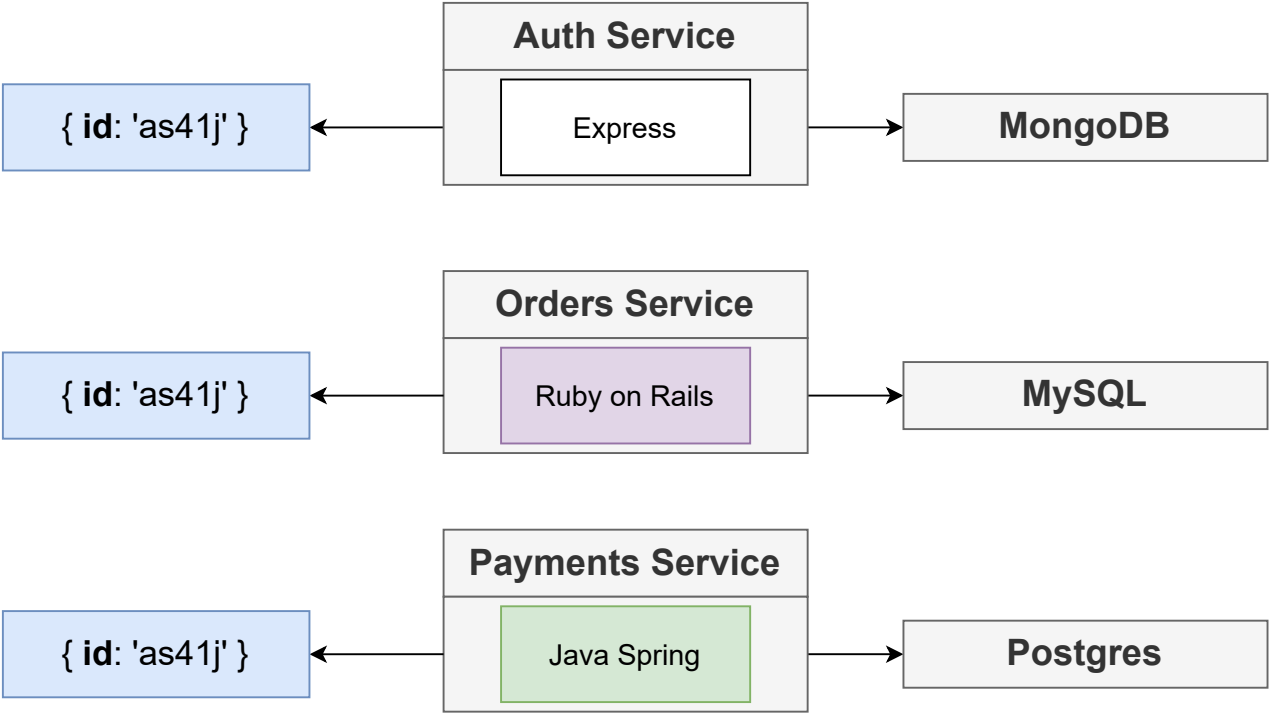

Node
Pod
Auth
Pod
Orders
Pod
Payments
JWT_KEY = 'asdf'
Secret

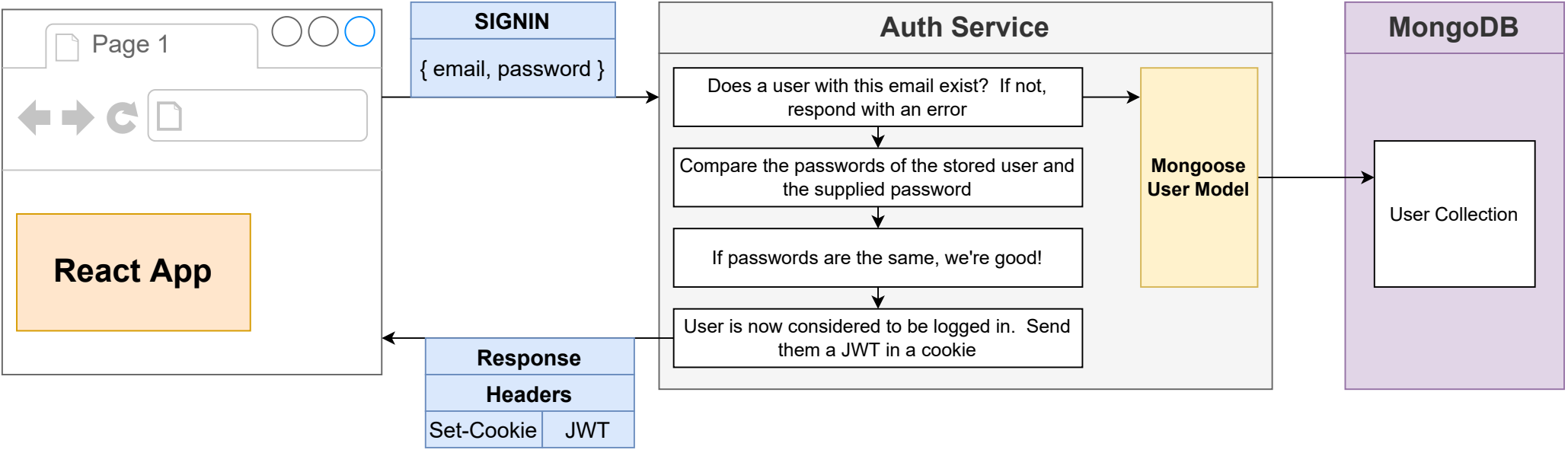

Node
Pod
Auth
Env Variables
Pod
Orders
Env Variables
Pod
Payments
Env Variables
JWT_KEY = 'asdf'
Secret

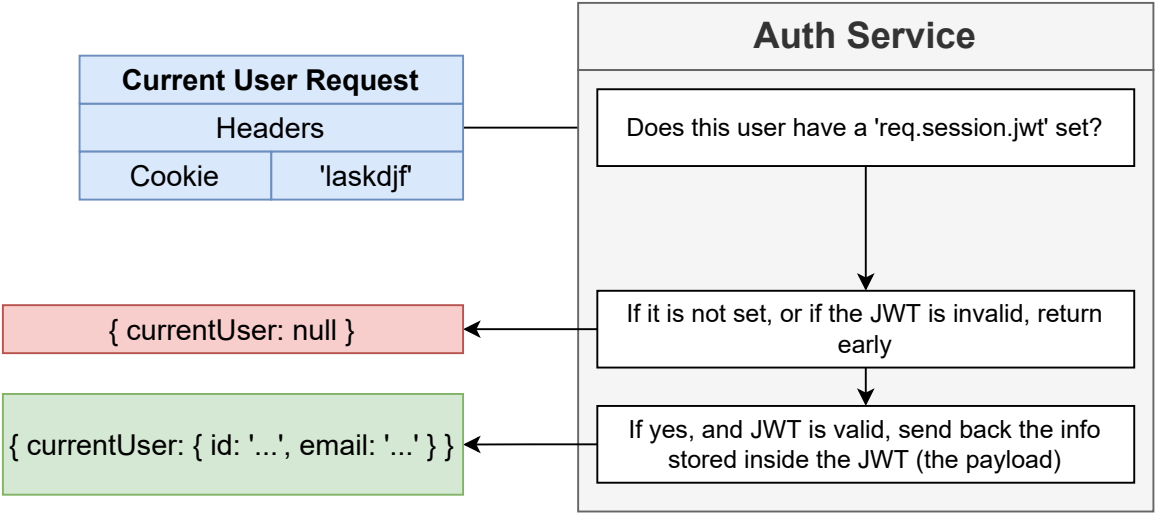
Creating a Secret

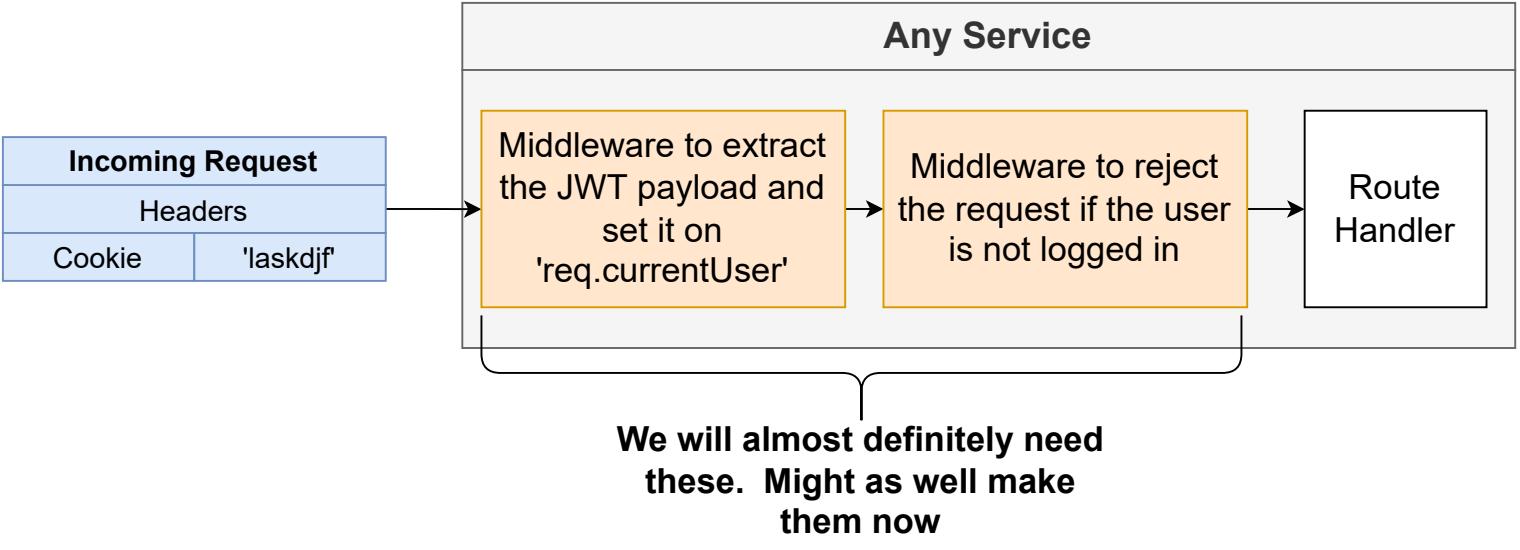
kubectl create secret generic **jwt-secret** --from-literal=**JWT_KEY=asdf**


Auth Service
Express
MongoDB
Orders Service
Ruby on Rails
MySQL
Payments Service
Java Spring
Postgres


Auth Service
{ id: 'as41j' }
Express
MongoDB
Orders Service
{ id: 'as41j' }
Ruby on Rails
MySQL
Payments Service
{ id: 'as41j' }
Java Spring
Postgres


Page 1
React App
SIGNIN
{ email, password }
Auth Service
Does a user with this email exist? If not, respond with an error
Compare the passwords of the stored user and the supplied password
If passwords are the same, we're good!
User is now considered to be logged in. Send them a JWT in a cookie
Mongoose User Model
MongoDB
User Collection
Response
Headers
Set-Cookie
JWT


Auth Service
Current User Request
Headers
Cookie
'laskdjf'
Does this user have a 'req.session.jwt' set?
If it is not set, or if the JWT is invalid, return early
{ currentUser: null }
If yes, and JWT is valid, send back the info stored inside the JWT (the payload)
{ currentUser: { id: '...', email: '...' } }


Any Service
Incoming Request
Headers
Cookie
'laskdjf'
Middleware to extract the JWT payload and set it on 'req.currentUser'
Middleware to reject the request if the user is not logged in
Route Handler
We will almost definitely need these. Might as well make them now