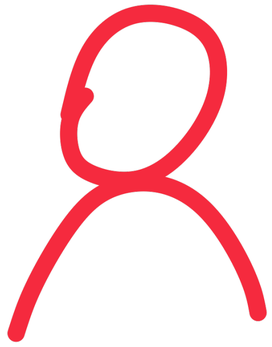
INTERLINK

JWT for auth and more

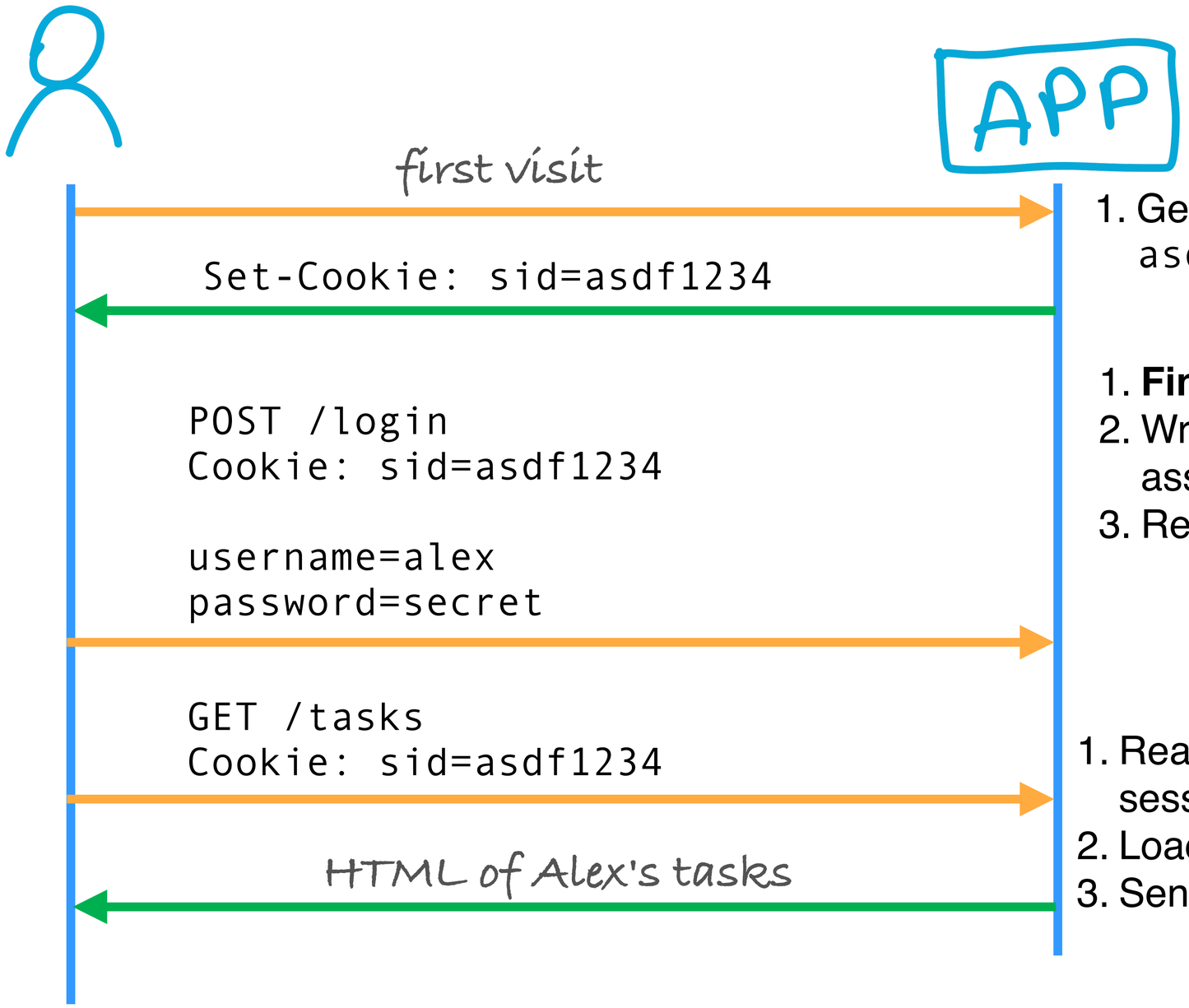
Old good stateful approach

Cookie: user_id=123



Cookie: user_id=123

BAD!!

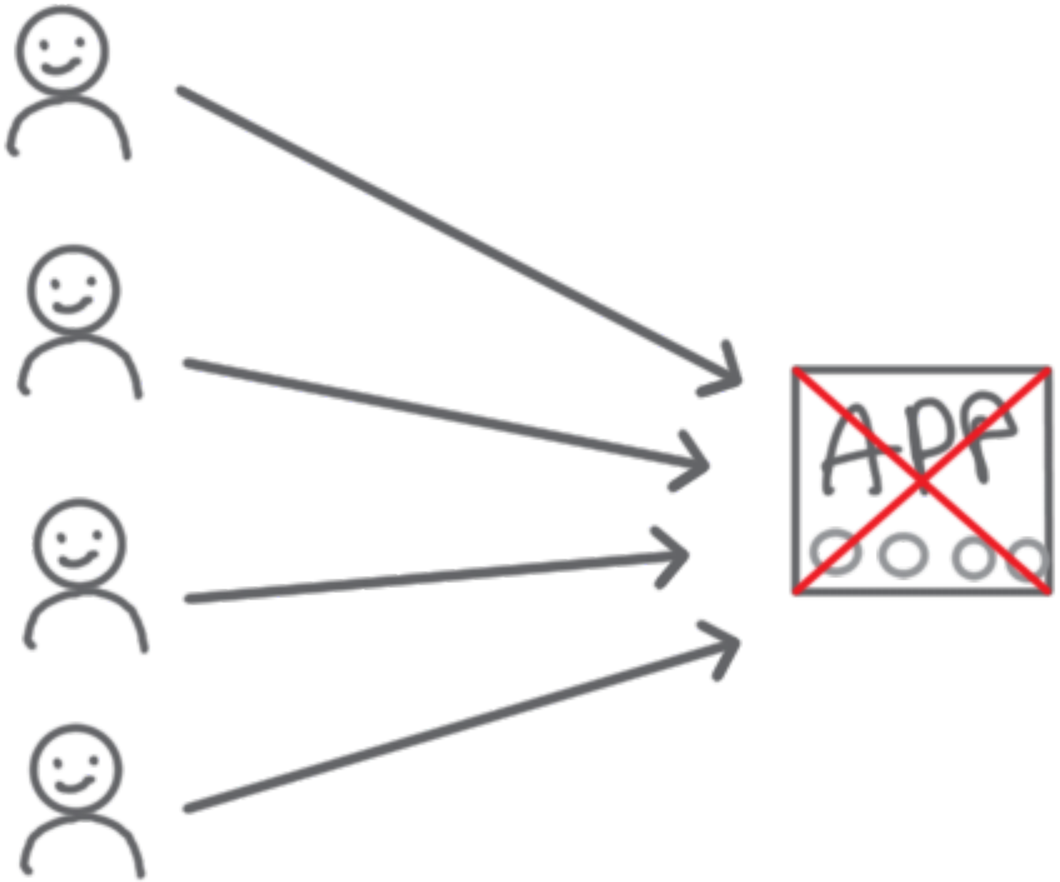


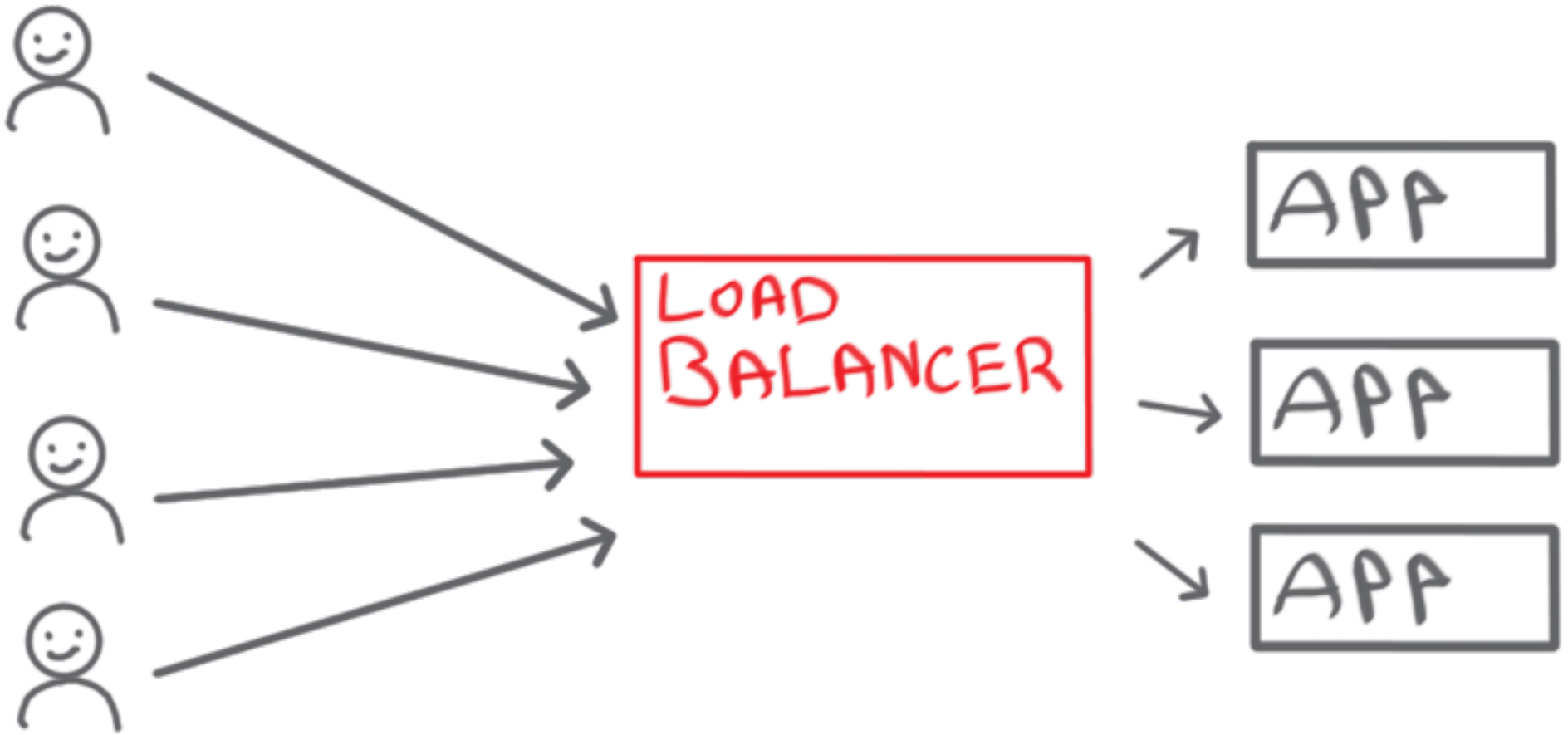
1. Generate random unique session ID `asdf1234`

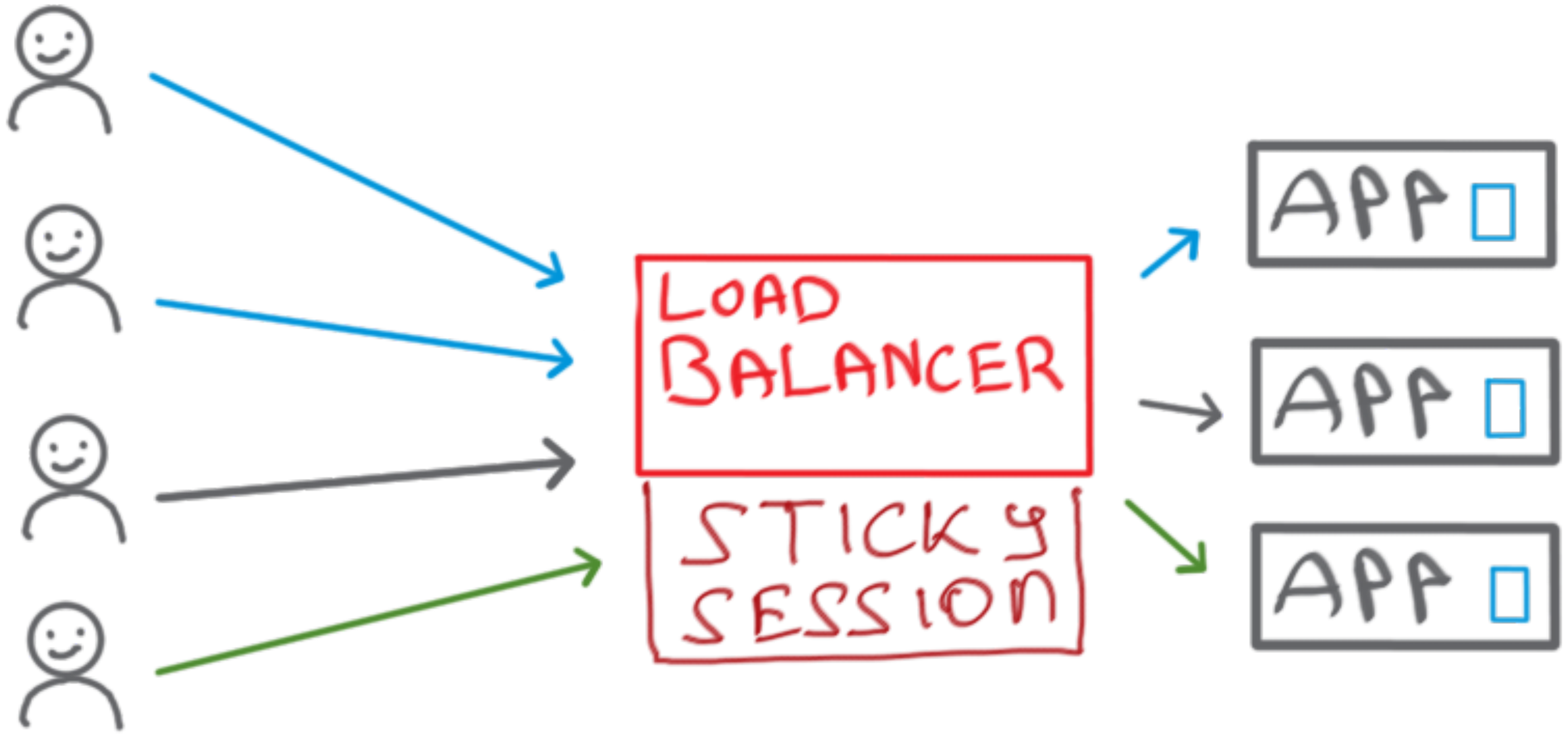
1. **Find user** and **verify password**
2. Write **user ID** into session bucket associated with **current user session ID**
3. Redirect to user home page

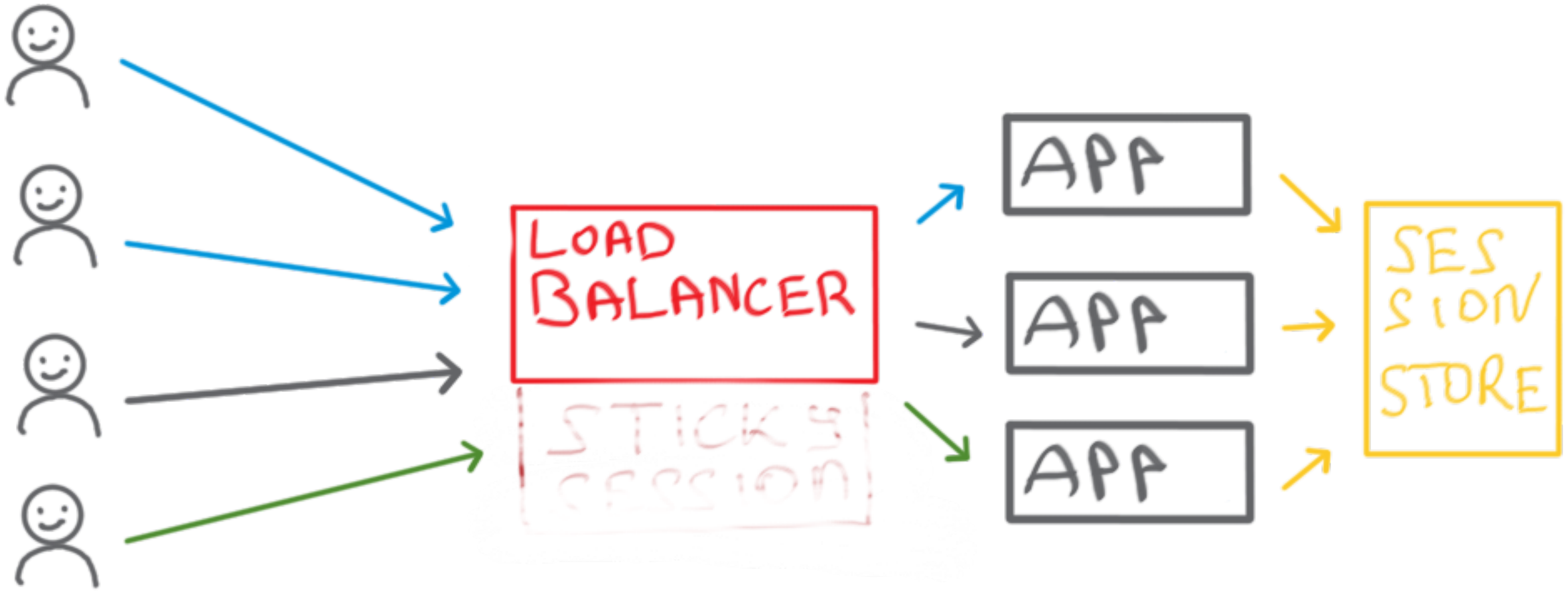
1. Read user ID from session storage by session ID `asdf1234`
2. Load tasks from DB by **user ID**
3. Send tasks data back

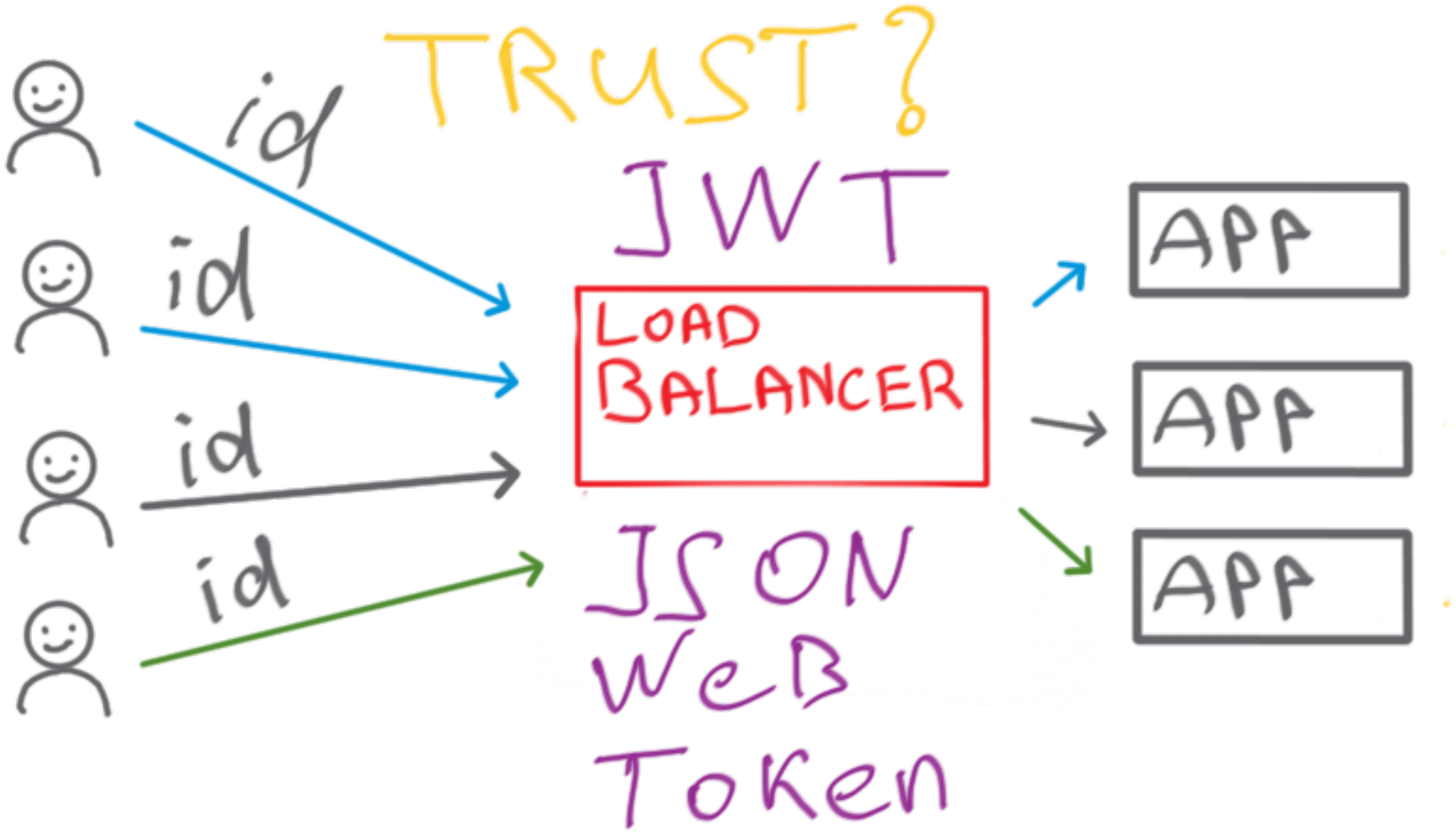
So what is a problem?











jwt.io

JWT Registered Claims (payload properties)

- **iss** (issuer): Issuer of the JWT
- **sub** (subject): Subject of the JWT (the user)
- **aud** (audience): Recipient for which the JWT is intended
- **exp** (expiration time): Time after which the JWT expires
- **iat** (issued at time): Time at which the JWT was issued; can be used to determine age of the JWT
- **jti** (JWT ID): Unique identifier; can be used to prevent the JWT from being replayed (allows a token to be used only once)

Public Claims

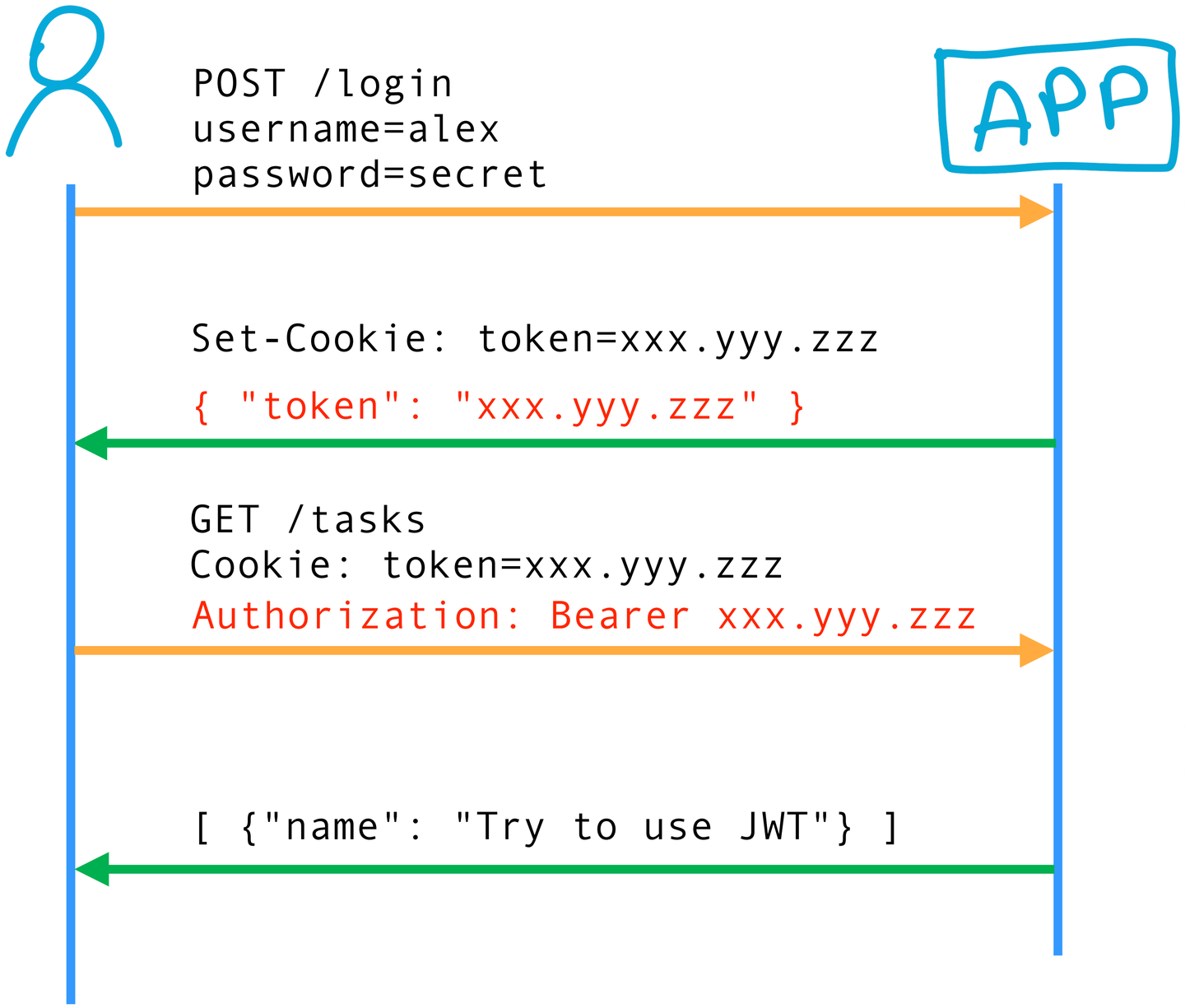
https://www.iana.org/assignments/jwt/jwt.xhtml

- **name**: Full name
- **nickname**: Casual name
- **profile**: Profile page URL
- **picture**: Profile picture URL
- **website**: Web page or blog URL
- **email**: Preferred e-mail address
- **scope**: Scope Values

Private Claims

- `"is_admin": true,`
- `"roles": "hr,manager"`

JWT usage flow



1. **Find user** and **verify password**
2. **Compose token** payload
   - user id (sub)
   - expiration time (exp)
   - issued at (iat)
   - issuer (iss)
3. **Sign token** with secret key

1. Check that the JWT is well formed
2. Check the signature
3. Check the standard claims (exp, iss)
4. Provide data for current user (sub)

JWT Tips and Tricks

Session expiration and sliding window

1. Chose desired max inactive duration
2. Add it to current time and set as token **exp**
3. For every request (or periodically) issue new token with updated expiration time

Single sign-on (SSO)

- Authorization server holds user accounts and issue access tokens
- Different services provide data if user has access token from trusted authorization server
- Microservices is a special case

Social Login

- Create auth0.com account
- Add up to 2 providers for free
- Validate tokens issues by Auth0 with their lib

Public and private claims

```
{
  "sub": "alexko",
  "email": "alexko@in6k.com",
  "nickname": "СашКо",
  "name": "Alexander Kotov",
  "is_admin": true,
  "scope": "meetup incamp",
  "timezone": "Europe/Kiev"
}
```

Confirmation of email address

1. Create a token with **email**, **limited scope** and desired **expiration time**
2. Send URL with token to user email address
   `/confirm?jwt=xxxxx.yyyyy.zzz`
3. When server receive a token
   1. Parse and validate token
   2. Mark email from token as confirmed

One time access – file download

- Authorization server creates token that
  - limited in scope and audiance
  - short-lived (about 2 minutes)
- Token passed to sateless file server in URL
  - `/documents/42/file?jwt=xxxxx.yyyyy.zzz`

But what can go wrong?

"Logout" from all devices

1. Hold issuing time in token (iat)
2. Give a user the "Logout me from everywhere" button
3. Store a time, when this button was clicked last time
4. Treat all tokens issued before this time as invalid

JWT limitations

- Secret key must be strongly protected
- Not suitable for stateful services as gets to big in size
- Requires white- or black-list to reject access with compromised (stolen) token
- Requires shared state for "log me out from all devices"

JWT usage tips

- Instead of longterm JWT use short-lived tokens and renew them when expiration time approaches
- You must be serious about allowed **alg**, **iss** and **aud**
- For web app store token in **http only, secure cookie**
  - limited by `/api` path, if applicable
- For better security consider using a pair of refresh/access tokens

INTERLINK

Do you want some code?

blog.interlink-ua.com

friends@interlink-ua.com

facebook.com/interlinkua