

# Svelte 5 + Hono

Part 4: layout sidebar + fetch wrapper



[www.youtube.com/@GetterSethya](https://www.youtube.com/@GetterSethya)

## Shadcn sidebar

```
npx shadcn-svelte@next add sidebar-07
```

Copy content from sidebar-page.svelte to protected/layout

Setup dark mode

```
npm i mode-watcher
```

[Dark mode - shadcn-svelte](#)

Adjust the sidebar component

Move logout mutation to sidebar

```
<DropdownMenu.Item onclick={() =>$logoutMutation.mutate()}>  
  <LogOut />  
  Log out  
</DropdownMenu.Item>
```

## Update class JWT on client

```
public validateAccess(): boolean {
    const access = localStorage.getItem(JWT.ACCESS_TOKEN);
    if (!access) return false;

    const decodedAccess = Jose.decodeJwt(access);
    if (!decodedAccess.exp) return false;

    return decodedAccess.exp > new Date().getTime() / 1000;
}

public validateRefresh(): boolean {
    const refresh = localStorage.getItem(JWT.REFRESH_TOKEN);
    if (!refresh) return false;

    const decodedRefresh = Jose.decodeJwt(refresh);
    if (!decodedRefresh.exp) return false;

    return decodedRefresh.exp > new Date().getTime() / 1000;
}

type GetNewJWTArgs = {
    onFailed: () => void;
};
```

## Update class JWT on client

```
public async getNewJWT(args?: Partial<GetNewJWTArgs>) {
  const refreshToken = localStorage.getItem(JWT.REFRESH_TOKEN);
  if (!refreshToken) throw new AuthenticationError('invalid/missing refresh token');

  const response = await Client.value.authentication.refresh.$post({
    form: { refresh_token: refreshToken }
  });
  const { result } = await response.json();

  if (!result || response.status !== 200) {
    localStorage.removeItem(JWT.ACCESS_TOKEN);
    localStorage.removeItem(JWT.REFRESH_TOKEN);
    if (args && args.onFailed) {
      args.onFailed();
      goto('/login');
    }

    throw new AuthenticationError('invalid token');
  }

  localStorage.setItem(JWT.ACCESS_TOKEN, result.access_token);
  localStorage.setItem(JWT.REFRESH_TOKEN, result.refresh_token);
}
```

# AppFetch

```
export async function appFetch(input: URL | RequestInfo, init?: RequestInit | undefined, fetcher?: typeof fetch) {  
  const jwt = new JWT();  
  
  if (!jwt.validateAccess()) {  
    console.log('token tidak valid, mencoba refresh jwt');  
    if (!jwt.validateRefresh()) {  
      localStorage.removeItem(JWT.ACCESS_TOKEN);  
      localStorage.removeItem(JWT.REFRESH_TOKEN);  
      goto('/login');  
  
      throw new AuthenticationError('invalid token');  
    }  
    await jwt.getNewJWT();  
  
    if (!init) init = {};  
    if (!init.headers) init.headers = new Headers();  
  
    const accessToken = localStorage.getItem(JWT.ACCESS_TOKEN);  
    if (!accessToken) throw new AuthenticationError('invalid token');  
    const newHeader = new Headers(init.headers);  
    newHeader.set('Authorization', accessToken);  
    init.headers = newHeader;  
    return fetcher ? await fetcher(input, init) : await fetch(input, init);  
  }  
  return fetcher ? await fetcher(input, init) : await fetch(input, init);  
}
```

## Update root layout load func

```
fetch: (input: URL | RequestInfo, init?: RequestInit | undefined) => appFetch(input,  
init, fetch)
```