Svelte 5 + Hono 🔥

Part 8: User controller, Profile page

www.youtube.com/@GetterSethya

PATCH method

```
export const updateUserValidator = z.object({
	name: z.string({ message: 'Name cant be blank' }).min(1, { message: 'Name cant be blank' }),
})
```

```
	.patch('/', appValidator('form', updateUserValidator), async (c) => {
		const userRepo = c.get('userRepo')
		const user = c.get('user')
		const form = c.req.valid('form')
		try {
			const updatedUser = await userRepo.update({ id: user.id, item: { name: form.name } })
			return appResponse(c, 'success', 200, updatedUser)
		} catch (error) {
			console.error(error)
			return appResponse(c, 'something went wrong', 500, null)
		}
	})
```

```
profile
  authentication
     +page.svelte
    +layout.svelte
    +page.ts
    +page.svelte
```

```
//Profile Layout

type ProfileItems = {
	href: string;
	title: string;
	icon?: any;
}[];

const items: ProfileItems = [
	{ href: '/profile', title: 'Profile', icon: UserIcon },
	{ href: '/profile/authentication', title: 'Authentication', icon: LockIcon }
];

let { children } = $props();

const [send, receive] = crossfade({
	duration: 250,
	easing: cubicInOut
});
```

```
<div class="flex w-full flex-col gap-2.5">
	<div class="flex flex-col lg:flex-row gap-2.5">
		<aside class="w-1/5">
			<nav class={'flex flex-row lg:flex-col gap-2.5'}>
				{#each items as item}
					{@const isActive = page.url.pathname === item.href}
					<Button
						href={item.href}
						variant="ghost"
						class={cn(
							!isActive && 'hover:underline',
							'relative justify-start hover:bg-transparent'
						)}
						data-sveltekit-noscroll
					>
						{#if isActive}
							<div
								class="absolute inset-0 rounded-md bg-muted"
								in:send={{ key: 'active-sidebar-tab' }}
								out:receive={{ key: 'active-sidebar-tab' }}
							></div>
						{/if}
						<div class="relative flex flex-row items-center gap-1">
							<item.icon class="hidden lg:block"/>
							{item.title}
						</div>
					</Button>
				{/each}
			</nav>
		</aside>
		<div class="flex-1 lg:max-w-2xl">
			{@render children()}
		</div>
	</div>
</div>
```

```
//nav-user
<DropdownMenu.Item class="cursor-pointer" onclick={() => goto('/profile')}>
	<UserIcon size={ICON_SIZE} />
	Profile
</DropdownMenu.Item>
```

```
//profile page.ts
export const load: PageLoad = async () => {
	return {
		form: await superValidate(zod(updateUserValidator))
	};
};
```

```svelte
//update-form.svelte
type UpdateFormProps = {
	data: SuperValidated<Infer<typeof updateUserValidator>>;
};
let { data }: UpdateFormProps = $props();
const user = LocalUser.getCtx();
const form = superForm(data, {
	validationMethod: 'auto',
	resetForm: false,
	validators: zodClient(updateUserValidator),
	SPA: true,
	onUpdate: async ({ form: fd }) => {
		if (fd.valid) {
			// do mutation
		}
	}
});
const { form: formData, enhance, reset } = form;
onMount(() => reset({ data: { name: $user.name } }));
```

```svelte
//update-form.svelte
<div class="flex flex-col gap-2.5">
	<Label>Email</Label>
	<Input disabled readonly bind:value={$user.email} placeholder="email" />
</div>
<form method="POST" use:enhance class="flex flex-col">
	<Form.Field {form} name="name">
		<Form.Control>
			{#snippet children({ props })}
				<div class="flex flex-col gap-2.5">
					<Form.Label>Name</Form.Label>
					<Input {...props} bind:value={$formData.name} placeholder="Profile name" />
				</div>
			{/snippet}
		</Form.Control>
		<Form.FieldErrors class="text-xs font-normal" />
	</Form.Field>
	<Form.Button disabled={$updateMutation.isPending} class="my-2.5 ms-auto">
		<span class="flex flex-row items-center gap-2.5">
			<span>Save</span>
			<SaveIcon size={ICON_SIZE} />
		</span>
	</Form.Button>
</form>
```

```
//update-form.svelte
	const updateMutation = createMutation({
		mutationFn: async (data: z.infer<typeof updateUserValidator>) => {
			const response = await $client.user.index.$patch(
				{form: data},
				{init: { headers: { Authorization: localStorage.getItem(JWT.ACCESS_TOKEN)! } },
				fetch: appFetch}
			);

			const resData = await response.json();

			if (response.status !== 200)
				throw new ResponseError(
					response.status,null,
					resData.message ? resData.message : 'Update user failed'
				);
			if (!resData.result)
				throw new ResponseError(
					response.status,resData.result,
					resData.message ? resData.message : 'Update user failed'
				);

			return resData.result;
		},
		onSuccess: (data) => {
			toast.success('Update user success');
			user.set(data);
		},

		onError: (error) => {
			if (error instanceof ResponseError) {
				toast.error(error.message);
				return;
			}

			toast.error('Something went wrong');
		}
	});
```

```
//update-form.svelte
const form = superForm(data, {
	validationMethod: 'auto',
	resetForm: false,
	validators: zodClient(updateUserValidator),
	SPA: true,
	onUpdate: async ({ form: fd }) => {
		if (fd.valid) {
			$updateMutation.mutate(fd.data);
		}
	}
});
```

```
//profile +page.svelte
<div class="flex w-full flex-col gap-2.5">
	<div class="flex flex-col">
		<span class="text-xl font-medium">Profile</span>
		<span class="text-foreground/60">Manage your profile information</span>
	</div>
	<UpdateForm data={data.form} />
</div>
```

```ts
// authValidator.ts
export const updatePasswordValidator = z
	.object({
		newPassword: z
			//
			.string({ message: 'New password cant be blank' })
			.min(8, { message: 'New password is too short' }),
		currentPassword: z
			.string({ message: 'Current password cant be blank' })
			.min(8, { message: 'Current password is too short' })
			.max(255, { message: 'Current password is too long' }),
		passwordConfirm: z
			.string({ message: 'Confirm password cant be blank' })
			.min(8, { message: 'Confirm password is too short' })
			.max(255, { message: 'Confirm password is too long' }),
	})
	.refine((ctx) => ctx.newPassword !== ctx.currentPassword, {
		message: 'New password must be different from the current password',
		path: ['newPassword'],
	})
	.refine((ctx) => ctx.newPassword === ctx.passwordConfirm, {
		message: 'Confirm password must match the new password',
		path: ['passwordConfirm'],
	})
```

```ts
// auth-controller.ts
	.patch('/password', jwtMiddleware, appValidator('form', updatePasswordValidator), async (c) => {
		const authRepo = c.get('authRepo')
		const user = c.get('user')
		const form = c.req.valid('form')
		try {
			const currentAuth = await authRepo.findByUser({ id: user.id })
			if (!currentAuth) return appResponse(c, 'user password is wrong/invalid', 400, null)

			const validCurrentPassword = await verify(currentAuth.hash_password, form.currentPassword)
			if (!validCurrentPassword) return appResponse(c, 'verify password is wrong/invalid', 400, null)

			const newPasswordHash = await hash(form.newPassword)
			await authRepo.update({ id: currentAuth.id, item: { hash_password: newPasswordHash } })
			return appResponse(c, 'success', 200, true)
		} catch (error) {
			console.error(error)
			return appResponse(c, 'something went wrong', 500, null)
		}
	})
```

```ts
// authentication +page.ts
export const load: PageLoad = async () => {
	return {
		form: await superValidate(zod(updatePasswordValidator))
	};
};
```

```svelte
// authentication update-form.svelte
type UpdateFormProps = {data: SuperValidated<Infer<typeof updatePasswordValidator>>};

let { data }: UpdateFormProps = $props();

const form = superForm(data, {
	validationMethod: 'auto',
	validators: zodClient(updatePasswordValidator),
	SPA: true,
	onUpdate: async ({ form: fd }) => {
		if (fd.valid) {
			$updateMutation.mutate(fd.data);
		}
	}
});

const { form: formData, enhance } = form;
```

```
// authentication update-form.svelte
<form method="POST" use:enhance class="flex flex-col">
	<Form.Field {form} name="currentPassword">
		<Form.Control>
			{#snippet children({ props })}
				<div class="flex flex-col gap-2.5">
					<Form.Label>Current Password</Form.Label>
					<Input
						{...props}
						bind:value={$formData.currentPassword}
						placeholder="Your current password"
						type="password"
					/>
				</div>
			{/snippet}
		</Form.Control>
		<Form.FieldErrors class="text-xs font-normal" />
	</Form.Field>
	<Form.Field {form} name="newPassword">
		<Form.Control>
			{#snippet children({ props })}
				<div class="flex flex-col gap-2.5">
					<Form.Label>New Password</Form.Label>
					<Input
						{...props}
						bind:value={$formData.newPassword}
						placeholder="Your new password"
						type="password"
					/>
				</div>
			{/snippet}
		</Form.Control>
		<Form.FieldErrors class="text-xs font-normal" />
	</Form.Field>
```

```
// authentication update-form.svelte
	<Form.Field {form} name="passwordConfirm">
		<Form.Control>
			{#snippet children({ props })}
				<div class="flex flex-col gap-2.5">
					<Form.Label>Confirm New Password</Form.Label>
					<Input
						{...props}
						bind:value={$formData.passwordConfirm}
						placeholder="Confirm your new password"
						type="password"
					/>
				</div>
			{/snippet}
		</Form.Control>
		<Form.FieldErrors class="text-xs font-normal" />
	</Form.Field>
	<Form.Button disabled={$updateMutation.isPending} class="my-2.5 ms-auto">
		<span class="flex flex-row items-center gap-2.5">
			<span>Save</span>
			<SaveIcon size={ICON_SIZE} />
		</span>
	</Form.Button>
</form>
```

```
// authentication +page.svelte
<script>
	import UpdateForm from './update-form.svelte';

	let { data } = $props();
</script>

<div class="flex w-full flex-col gap-2.5">
	<div class="flex flex-col">
		<span class="text-xl font-medium">Authentication</span>
		<span class="text-foreground/60">Manage your password</span>
	</div>
	<UpdateForm data={data.form} />
</div>
```