🔐 OAuth Module - Tài liệu chi tiết

> **Mô tả chi tiết module xác thực người dùng (Đăng nhập / Đăng ký / Quên mật khẩu)**
>
> Cập nhật: 2026-01-19

📋 Mục Lục

1. Tổng quan

1.1 Mục đích

Module OAuth xử lý toàn bộ luồng **xác thực người dùng** bao gồm:

- Đăng nhập bằng Email/Password
- Đăng nhập bằng Facebook OAuth
- Đăng ký tài khoản mới
- Xác thực email
- Quên mật khẩu & Đặt lại mật khẩu

1.2 Tech Stack

Công nghệ	Mục đích
Vue 3 Composition API	Framework
Vue Router	Navigation
Pinia	State management
Yup	Form validation
vue-i18n	Đa ngôn ngữ
tsyringe	Dependency Injection
Facebook SDK	OAuth đăng nhập FB

2. Cấu trúc thư mục

```
src/views/OAuth/
├── Login.vue              # Màn hình đăng nhập chính (email + FB)
├── LoginEmail.vue         # Đăng nhập với email + password
├── Register.vue           # Màn hình đăng ký (nhập email)
├── RegisterDetail.vue     # Form đăng ký đầy đủ
├── ForgotPassword.vue     # Quên mật khẩu
├── ResetPassword.vue      # Đặt lại mật khẩu
│
├── Alert.vue              # Component alert thông báo (success/info)
├── AlertError.vue         # Component alert lỗi
├── InputPassword.vue      # Input password với toggle show/hide
├── NewTo.vue              # Link "Chưa có tài khoản?"
├── GoLogin.vue            # Link "Đã có tài khoản?"
├── Or.vue                 # Divider "Hoặc"
│
├── service.ts             # Business logic & composables
├── store.ts               # Pinia store cho OAuth
├── validate.ts            # Validation schemas (Yup)
└── index.scss             # Styles chung
```

2.1 Layout Wrapper

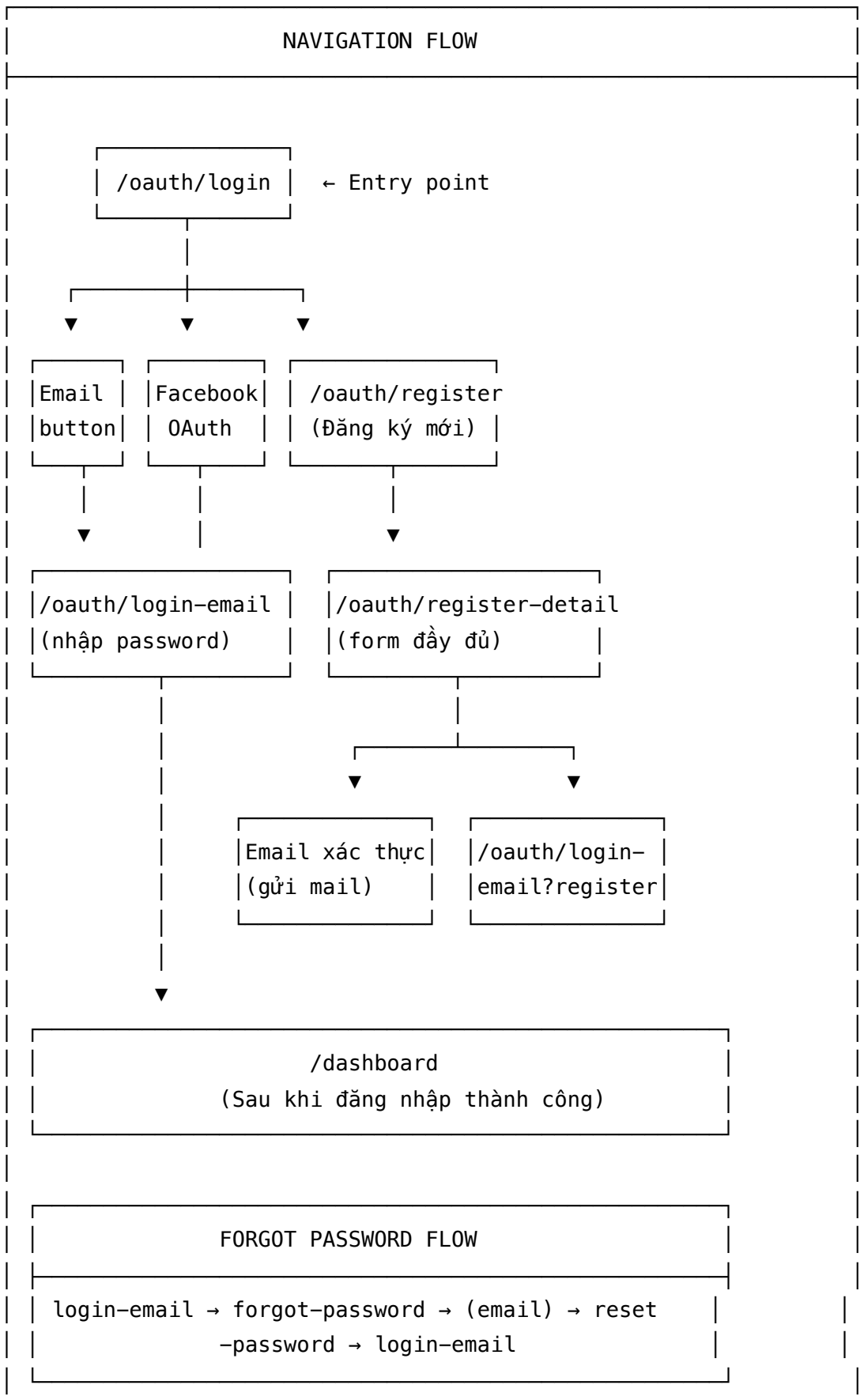
```
src/views/
├── OAuthV2.vue             # Layout wrapper cho toàn bộ OAuth
```

3. Routes & Navigation

3.1 Route Configuration

```
// src/router/routes.ts
{
  path: '/oauth',
  redirect: '/oauth/login',
  component: OAuthV2,
  children: [
    { path: 'login', component: Login },
    { path: 'login-email', component: LoginEmail },
    { path: 'register', component: Register },
    { path: 'register-detail', component: RegisterDetail },
    { path: 'forgot-password', component: ForgotPassword },
    { path: 'reset-password', component: ResetPassword },
  ],
},
{
  path: '/logout',
  redirect: '/oauth',
},
```

3.2 Navigation Flow



```
└──────────────────────────────────────────────────────────────────────┘
```

4. Components chi tiết

4.1 OAuthV2.vue (Layout Wrapper)

Mục đích: Layout bọc ngoài cho tất cả OAuth screens

Cấu trúc UI:

```
<section class="bg-gradient-secondary">
  <div class="bg-white max-w-[478px] shadow-xl rounded-xl">
    <!-- Logo partner -->
    <div :style="{ backgroundImage: partner.logo }" />

    <!-- Router View (children components) -->
    <RouterView />

    <!-- Footer: Chính sách & Điều khoản -->
    <div class="flex justify-between">
      <a :href="PRIVACY">Chính sách</a>
      <a :href="TERM">Điều khoản</a>
      <Language />
      <!-- Đổi ngôn ngữ -->
    </div>
  </div>
</section>
```

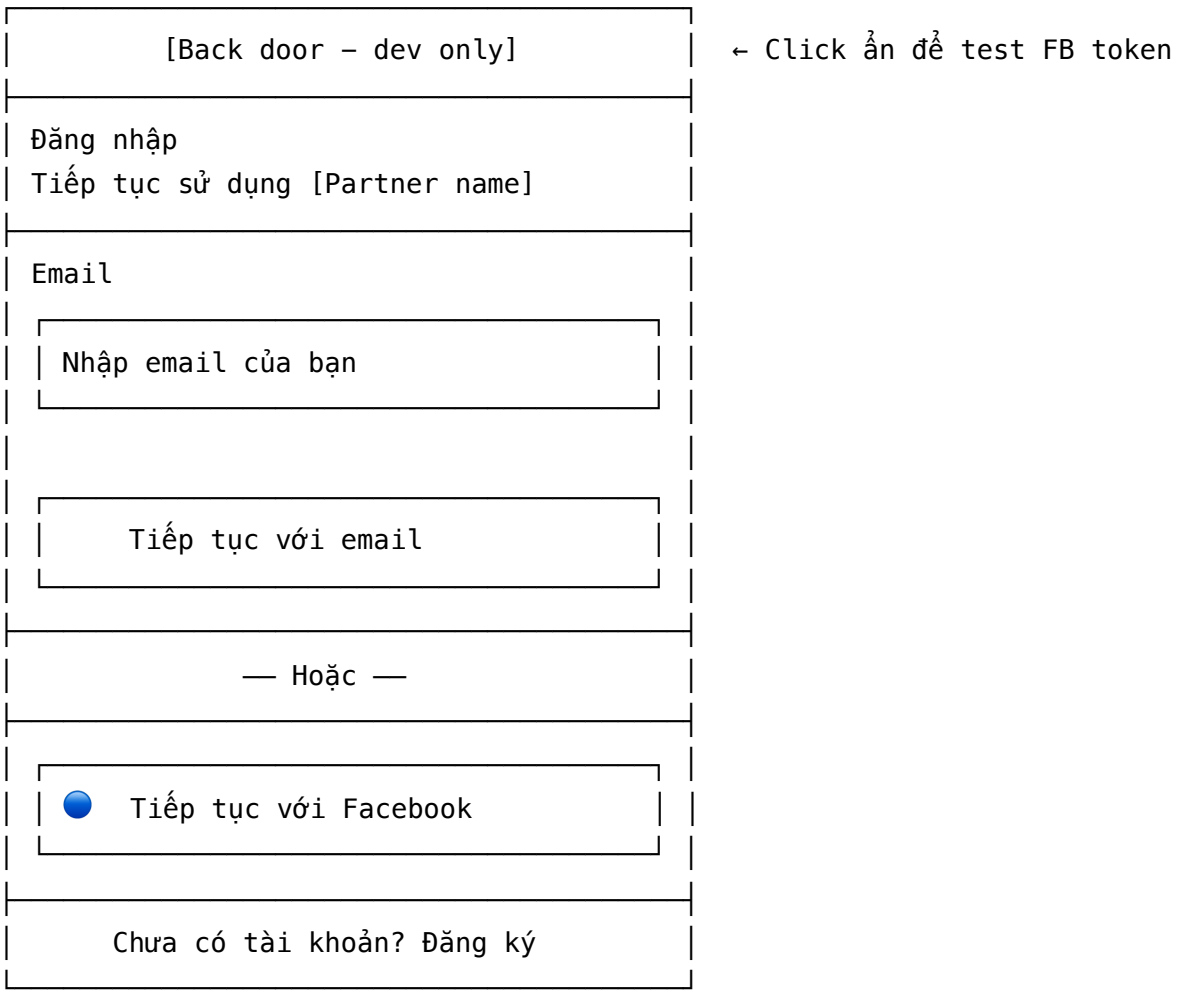
Background Gradient:

```
.bg-gradient-secondary {
  background: linear-gradient(180deg, hsla(0, 0%, 100%, 0) 50%, #fff, #fff),
    linear-gradient(90deg, #fdefe3, #f3f4f3 50%, #dce5ff);
}
```

4.2 Login.vue

Mục đích: Màn hình đăng nhập chính - chọn phương thức

UI Structure:



Logic chính:

```
class Main {
  /** Backdoor login bằng FB token (dev) */
  backDoorLoginFb() {
    modal_input('', '', (e, r) => {
      if (e || !r) return
      this.SERVICE_OAUTH.loginFb(r)
    })
  }

  /** Kiểm tra đã đăng nhập chưa → redirect */
  isAlreadyLogin() {
    if (!getItem('access_token')) return
    this.SERVICE_OAUTH.redirect('/chat')
  }

  /** Validate email → chuyển trang nhập password */
  async loginEmail() {
    await VLD_EMAIL.validate({ email: email.value })
    this.SERVICE_OAUTH.redirect({
      path: '/oauth/login-email',
      query: { email: email.value },
    })
  }
}

// Auto check khi mounted
onMounted(() => $main.isAlreadyLogin())
```

4.3 LoginEmail.vue

Mục đích: Form đăng nhập với email + password

UI Features:

- Input email (readonly từ query)
- Input password (toggle show/hide)
- Link quên mật khẩu
- Alert thông báo từ các flow khác:
  - Đăng ký thành công → cần xác thực email
  - Đặt lại mật khẩu thành công

- Email đã được xác nhận

State Management:

```
/** Form data */
const form = ref<{
  email: string
  password: string
}>({ email: '', password: '' })

/** Flags từ query params */
const is_redirect_from_register = ref(!!$route.query.register)
const verify_code = ref($route.query.verify_code as string)
const is_verify_email = ref(false)
const is_resend_verify_email = ref(false)
const is_redirect_from_forgot_password = ref(!!$route.query.forgot_password)
const is_redirect_from_reset_password = ref(!!$route.query.reset_password)
```

Methods:

```
class Main {
  /** Đăng nhập API */
  @handleLoadingOauth
  @handleErrorOauth(e => {
    // Nếu email chưa xác thực → bật cờ gửi lại
    if (e?.message === 'COMMON.EMAIL_NOT_VERIFY')
      is_redirect_from_register.value = true
  })
  async loginEmail() {
    await VLD_EMAIL_PASSWORD.validate(form.value)

    const { access_token: JWT } = await this.API_OAUTH_BASIC.login(
      form.value.email,
      form.value.password
    )

    setItem('access_token', JWT)
    this.SERVICE_OAUTH.redirect('/dashboard')
    this.TRIGGER_EVENT_REF.sendMessageLoginEmail(form.value.email)
  }

  /** Gửi lại email xác thực */
  async resendVerifyEmail() {
    if (is_resend_verify_email.value) return
    await this.API_OAUTH_BASIC.resendVerifyEmail(form.value?.email)
    is_resend_verify_email.value = true
  }

  /** Xác thực email từ link */
  async verifyEmail() {
    if (!verify_code.value) return
    await this.API_OAUTH_BASIC.verifyEmail(form.value.email, verify_code.value)
    is_verify_email.value = true
  }
}
```

4.4 Register.vue

Mục đích: Nhập email để đăng ký

Logic:

```
class Main {
  /** Validate email → check tồn tại → chuyển form chi tiết */
  @handleLoadingOauth
  @handleErrorOauth()
  async goRegisterDetail() {
    // Validate format email
    await VLD_EMAIL.validate({ email: email.value })

    // Check email đã tồn tại chưa
    await this.API_OAUTH_BASIC.checkEmail(email.value!)

    // Chuyển sang form đăng ký chi tiết
    this.SERVICE_OAUTH.redirect({
      path: '/oauth/register-detail',
      query: { email: email.value },
    })
  }

  /** Backdoor xóa tài khoản (dev) */
  async deleteAccount() {
    // Swal input → API delete → toast
  }
}
```

4.5 RegisterDetail.vue

Mục đích: Form đăng ký đầy đủ thông tin

Form Fields:

```
const form = ref<{
  email: string // Từ query
  first_name: string // Tên
  last_name: string // Họ
  password: string // Mật khẩu
  confirm_password: string // Xác nhận mật khẩu
}>({
  email: '',
  first_name: '',
  last_name: '',
  password: '',
  confirm_password: '',
})

/** Tên đầy đủ (xử lý VN vs thế giới) */
const full_name = computed(() => {
  if (locale.value === 'vn')
    return `${form.value.last_name} ${form.value.first_name}`
  return `${form.value.first_name} ${form.value.last_name}`
})
```

Register Flow:

```
async register() {
  // Trim whitespace
  form.value.email = form.value.email.trim()
  // ... other fields

  // Validate
  await VLD_EMAIL_REGISTER.validate(form.value)

  // Check password match
  if (form.value.password !== form.value.confirm_password)
    throw $t('Mật khẩu không khớp')

  // API Register
  await this.API_OAUTH_BASIC.register(
    form.value.email,
    form.value.password,
    full_name.value,
    form.value.first_name,
    form.value.last_name,
    this.SERVICE_LOCAL_STORAGE.getItem('ref')  // Referral code
  )

  // Success → redirect to login
  this.SERVICE_TOAST.success($t('Đăng ký tài khoản thành công'))
  this.SERVICE_OAUTH.redirect({
    path: '/oauth/login-email',
    query: { email: form.value.email, register: 'true' },
  })

  // Trigger analytics
  this.TRIGGER_EVENT_REF.sendMessageRegisterEmail(form.value.email)
}
```

4.6 Common Components

Alert.vue

```
<!-- Thông báo thành công/info -->
<div class="bg-green-100 border-l-4 border-green-500 p-3">
  <slot />
</div>
```

AlertError.vue

```
<!-- Thông báo lỗi từ store -->
<Alert
  v-if="oAuthStore.error_message"
  class="bg-red-100"
>
  {{ oAuthStore.error_message }}
</Alert>
```

InputPassword.vue

```
<!-- Input password với toggle show/hide -->
<div class="relative">
  <input
    :type="is_show ? 'text' : 'password'"
    v-model="model"
  />
  <button @click="is_show = !is_show">
    <EyeIcon v-if="is_show" />
    <EyeSlashIcon v-else />
  </button>
</div>
```

5. Services & Logic

5.1 service.ts

Composable chính:

```
export function composableService() {
  const commonStore = useCommonStore()
  const oAuthStore = useOAuthStore()
  const $router = useRouter()
  const { t: $t } = useI18n()

  /** Service xử lý đăng nhập */
  @singleton()
  class ServiceOAuth {
    constructor(
      private readonly API_OAUTH_FB = container.resolve(
        N4SerivcePublicOauthFacebok
      ),
      private readonly SERVICE_LOCAL_STORAGE = container.resolve(LocalStorage)
    ) {}

    /** Đăng nhập bằng FB token */
    @loadingV2(commonStore, 'is_loading_full_screen')
    @error()
    async loginFb(access_token: string) {
      const { access_token: JWT } = await this.API_OAUTH_FB.login(
        access_token,
        this.SERVICE_LOCAL_STORAGE.getItem('ref')
      )
      this.SERVICE_LOCAL_STORAGE.setItem('access_token', JWT)
      this.redirect('/dashboard')
    }

    /** Chuyển hướng trang */
    redirect(to: RouteLocationRaw) {
      oAuthStore.error_message = undefined
      $router.push(to)
    }
  }

  /** Toast thông báo custom */
  @singleton()
  class CustomToast extends Toast implements IAlert {
    public error(message: any): void {
      const CUSTOM_ERROR = customError(message?.message)
      if (CUSTOM_ERROR) {
        oAuthStore.error_message = CUSTOM_ERROR
        return
```

```
      }
      super.error(message?.message || message)
    }
  }

  /** Decorator loading cho OAuth */
  const handleLoadingOauth = loadingV2(commonStore, 'is_loading_full_screen')

  /** Decorator error cho OAuth */
  const handleErrorOauth = (callback?: Function) =>
    error(container.resolve(CustomToast), callback)

  /** Links external */
  const TERM = 'https://retion.ai/tos.html'
  const PRIVACY = 'https://retion.ai/privacy.html'

  return {
    ServiceOAuth,
    customError,
    handleLoadingOauth,
    handleErrorOauth,
    TERM,
    PRIVACY,
  }
}
```

5.2 store.ts

```
// Pinia store cho OAuth
export const useOAuthStore = defineStore('oauth', {
  state: () => ({
    error_message: undefined as string | undefined,
  }),
})
```

6. Validation

6.1 validate.ts

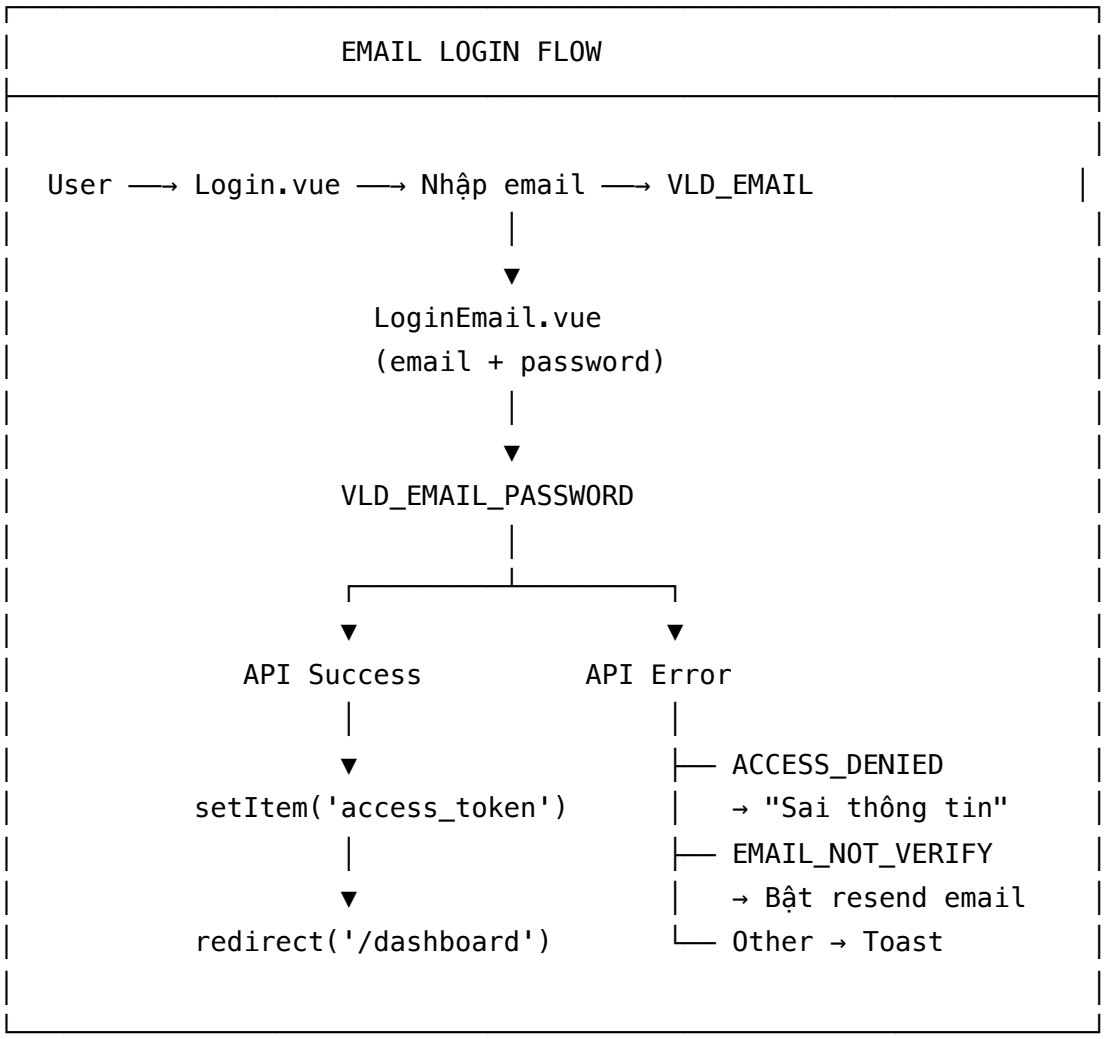
Sử dụng Yup schema:

```
export function composableValidate() {
  const { t: $t } = useI18n()

  // Labels
  const EMAIL = $t('Email')
  const FIRST_NAME = $t('Tên')
  const LAST_NAME = $t('Họ')
  const PASSWORD = $t('Mật khẩu')
  const CONFIRM_PASSWORD = $t('Xác nhận mật khẩu')

  /** Validate email */
  const VLD_EMAIL = object({
    email: string()
      .required($t('Bạn chưa nhập _', { name: EMAIL }))
      .noWhitespace($t('_ không hợp lệ', { name: EMAIL }))
      .email($t('_ không hợp lệ', { name: EMAIL })),
  })

  /** Validate email + password */
  const VLD_EMAIL_PASSWORD = object({
    password: string()
      .required($t('Bạn chưa nhập _', { name: PASSWORD }))
      .noWhitespace($t('_ không hợp lệ', { name: PASSWORD })),
    ...VLD_EMAIL.fields,
  })

  /** Validate đăng ký đầy đủ */
  const VLD_EMAIL_REGISTER = object({
    first_name: string()
      .required($t('Bạn chưa nhập _', { name: FIRST_NAME }))
      .noWhitespace($t('_ không hợp lệ', { name: FIRST_NAME })),
    last_name: string()
      .required($t('Bạn chưa nhập _', { name: LAST_NAME }))
      .noWhitespace($t('_ không hợp lệ', { name: LAST_NAME })),
    ...VLD_EMAIL_PASSWORD_AND_CONFIRM.fields,
  })

  return {
    VLD_EMAIL,
    VLD_EMAIL_PASSWORD,
    VLD_EMAIL_REGISTER,
    VLD_EMAIL_PASSWORD_AND_CONFIRM,
```

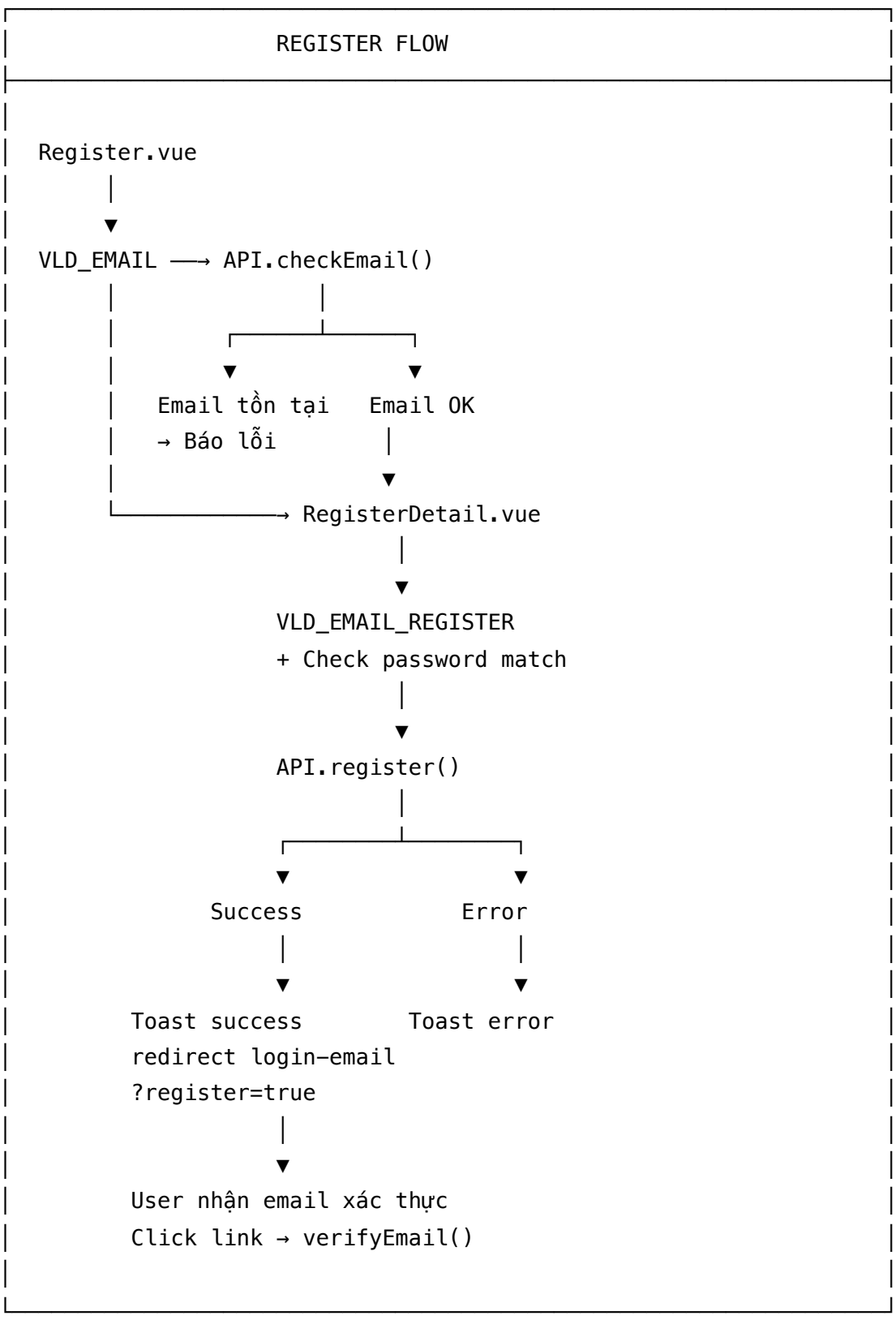
```
  }
}
```

7. Flow Diagrams

7.1 Luồng Đăng nhập Email



7.2 Luồng Đăng ký



8. API Integration

8.1 API Services

Service	File	Methods
`N4SerivcePublicOauthBasic`	`utils/api/N4Service/Oauth`	`login()`, `register()`, `checkEmail()`, `verifyEmail()`, `resendVerifyEmail()`, `forgotPassword()`, `resetPassword()`
`N4SerivcePublicOauthFacebok`	`utils/api/N4Service/Oauth`	`login(access_token, ref)`

8.2 API Endpoints

Method	Endpoint	Description
POST	`/oauth/basic/login`	Đăng nhập email/password
POST	`/oauth/basic/register`	Đăng ký tài khoản
POST	`/oauth/basic/check-email`	Kiểm tra email tồn tại
POST	`/oauth/basic/verify-email`	Xác thực email
POST	`/oauth/basic/resend-verify-email`	Gửi lại email xác thực
POST	`/oauth/basic/forgot-password`	Quên mật khẩu
POST	`/oauth/basic/reset-password`	Đặt lại mật khẩu
POST	`/oauth/facebook/login`	Đăng nhập Facebook

9. Error Handling

9.1 Custom Error Messages

```
function customError(message?: string): string | undefined {
  const errorMap: Record<string, string> = {
    'COMMON.ACCESS_DENIED': 'Tài khoản hoặc mật khẩu không đúng',
    'COMMON.EMAIL_NOT_VERIFY': 'Tài khoản chưa được xác thực',
    'COMMON.USER_NOT_FOUND': 'Tài khoản không tồn tại',
    'COMMON.USER_IS_VERIFY': 'Tài khoản đã xác thực',
    'COMMON.INVALID.VERIFY_CODE': 'Mã xác thực không đúng',
    'COMMON.EMAIL_EXISTED': 'Email đã tồn tại',
  }
  return errorMap[message || '']
}
```

9.2 Error Display

- **Field-level**: Hiển thị trong `AlertError.vue` (store error_message)
- **Toast**: Sử dụng `CustomToast.error()` cho lỗi không custom

10. LocalStorage

10.1 Keys được sử dụng

Key	Type	Description
`access_token`	string	JWT token sau đăng nhập
`ref`	string	Referral code (nếu có)

10.2 Flow

```
Đăng nhập success → setItem('access_token', JWT)
                  → redirect('/dashboard')

Truy cập /oauth   → getItem('access_token')
                  → Có token? → redirect('/chat')
                  → Không? → Hiện form login
```

📎 Related Files

- **Layout:** `src/views/OAuthV2.vue`
- **API:** `src/utils/api/N4Service/Oauth.ts`
- **Store:** `src/views/OAuth/store.ts`
- **Styles:** `src/views/OAuth/index.scss`

> **Note:** Module này sử dụng `tsyringe` cho Dependency Injection và `@singleton()` decorator để đảm bảo các service chỉ được khởi tạo một lần.