## Kế hoạch Phát triển Giao diện Người dùng ITAPIA với Vue.js (4 Tuần)

**Mục tiêu cuối cùng:** Một ứng dụng web (Single Page Application - SPA) hoạt động, cho phép người dùng đăng nhập, xem phân tích chi tiết, nhận khuyến nghị, và thiết lập hồ sơ cá nhân hóa.

**Công nghệ chủ đạo:**

- **Framework:** Vue 3 (sử dụng Composition API với `<script setup>`)
- **Build Tool:** Vite
- **Routing:** Vue Router
- **State Management:** Pinia
- **HTTP Client:** Axios
- **UI Component Library: Vuetify 3** (Gợi ý: Rất mạnh mẽ, đầy đủ component, và có thiết kế Material Design đẹp mắt)

---

## Tuần 1: Nền tảng Vue.js và Hiển thị Dữ liệu Thô

*Mục tiêu tuần: Hiểu các khái niệm cốt lõi của Vue và có thể gọi API để hiển thị dữ liệu JSON lên màn hình.*

- **Ngày 1-2: Cài đặt và Khám phá Khái niệm Cơ bản**
    - **Học:** Đọc phần "Essentials" của tài liệu Vue 3. Tập trung vào: Reactivity (`ref`, `reactive`), Component Basics (`props`, `events`).
    - **Làm:**
        1. Cài đặt Node.js và npm/yarn.
        2. Sử dụng Vite để tạo dự án Vue 3 mới: `npm create vue@latest`.
        3. Tạo và thử nghiệm các component đơn giản để hiểu cách truyền dữ liệu từ cha xuống con.
- **Ngày 3: Vòng đời Component và Gọi API**
    - **Học:** Lifecycle Hooks (đặc biệt là `onMounted`).
    - **Làm:**
        1. Cài đặt Axios: `npm install axios`.
        2. Tạo một component `AnalysisView.vue`.
        3. Trong `onMounted`, viết hàm gọi API `GET /analysis/quick/{ticker}` với một ticker cứng (ví dụ: 'AAPL').
        4. Lưu kết quả JSON vào một biến `ref()` và `console.log()` để xác nhận.
- **Ngày 4-5: Hiển thị Dữ liệu Động và Xử lý Trạng thái**
    - **Học:** Template Syntax của Vue (`v-if`, `v-else`, `v-for`, `{{ mustache }}`).
    - **Làm:**
        1. Trong `AnalysisView.vue`, sử dụng `v-if` để hiển thị "Loading..." khi đang gọi API và "Error" nếu có lỗi.
        2. Khi có dữ liệu, dùng `v-for` để lặp và hiển thị một phần dữ liệu đơn giản, ví dụ: danh sách `triggered_rules`.
        3. Sử dụng `{{ result.ticker }}` để hiển thị tên mã cổ phiếu.
- **Ngày 6-7: Tích hợp Thư viện Component & Xây dựng Bố cục**
    - **Học:** Cách tích hợp một thư viện component như Vuetify 3 vào dự án Vite.
    - **Làm:**
        1. Cài đặt Vuetify vào dự án.
        2. Xây dựng bố cục (layout) chính cho ứng dụng: một thanh điều hướng (Navbar) ở trên, một chân trang (Footer) ở dưới, và vùng nội dung chính ở giữa.
        3. Bắt đầu thay thế các thẻ HTML thô bằng các component của Vuetify (ví dụ: `v-card`, `v-list`, `v-chip`) để làm cho trang `AnalysisView` trông chuyên nghiệp hơn.

---

## Tuần 2: Tương tác Người dùng và Cấu trúc Ứng dụng

*Mục tiêu tuần: Biến ứng dụng từ một trang tĩnh thành một ứng dụng đa trang, có tương tác.*

- **Ngày 8-9: Định tuyến (Routing)**
    - **Học:** Cài đặt và cấu hình Vue Router.
    - **Làm:**
        1. Tạo các "Views" (các component trang) chính: `HomeView.vue`, `AnalysisView.vue`, `LoginView.vue` (trống), `ProfileView.vue` (trống), `NotFoundView.vue`.
        2. Cấu hình router để liên kết các path (`/`, `/analysis/:ticker`, `/login`...) với các View tương ứng.
        3. Sử dụng `<router-link>` trong Navbar và `<router-view>` trong layout chính.
- **Ngày 10-11: Trang chủ và Tương tác**
    - **Học:** Xử lý sự kiện (`@click`) và `v-model` để binding dữ liệu hai chiều.
    - **Làm:**
        1. Trong `HomeView.vue`, tạo một ô input (`v-text-field` của Vuetify) và bind nó vào một biến `ref()` tên là `tickerInput`.
        2. Tạo một nút "Analyze". Khi click, lấy giá trị từ `tickerInput` và điều hướng người dùng đến trang `/analysis/{tickerInput}` bằng `router.push()`.
- **Ngày 12-14: Quản lý Trạng thái Toàn cục với Pinia**
    - **Học:** Các khái niệm cơ bản của Pinia: `defineStore`, `state`, `getters`, `actions`.
    - **Làm:**
        1. Cài đặt Pinia.

2. Tạo một store đầu tiên: `analysisStore.js`.
3. Di chuyển toàn bộ logic gọi API và lưu trữ kết quả từ component `AnalysisView.vue` vào bên trong `analysisStore`.
4. Trong `AnalysisView.vue`, gọi `action` từ store và sử dụng `state` hoặc `getters` để hiển thị dữ liệu. Điều này giúp tách biệt logic và giao diện.

---

## Tuần 3: Authentication và Personalization

*Mục tiêu tuần: Hoàn thiện luồng người dùng cốt lõi, từ đăng nhập đến xem kết quả được cá nhân hóa (ở mức độ MVP).*

- **Ngày 15-17: Luồng Đăng nhập (Authentication)**
  - **Học:** Luồng OAuth2/PKCE cho ứng dụng SPA.
  - **Làm:**
    1. Tạo một `authStore.js` trong Pinia để quản lý trạng thái đăng nhập, thông tin người dùng và token JWT.
    2. Trong `LoginView.vue`, tạo nút "Login with Google". Khi click, điều hướng người dùng đến URL đăng nhập của Google mà backend cung cấp.
    3. Tạo một trang callback (ví dụ `/auth/callback`) để nhận token từ backend sau khi đăng nhập thành công. Lưu token vào `authStore` và `localStorage`.
    4. Cập nhật Navbar để hiển thị tên người dùng và nút "Logout" nếu đã đăng nhập.
- **Ngày 18-19: Bảo vệ Route và Trang Hồ sơ**
  - **Học:** Navigation Guards của Vue Router.
  - **Làm:**
    1. Sử dụng `beforeEach` trong router để kiểm tra xem người dùng đã đăng nhập chưa (dựa vào `authStore`). Nếu chưa, chuyển hướng họ về trang `/login` khi họ cố gắng truy cập các trang được bảo vệ.
    2. Xây dựng **Trang Profile (`ProfileView.vue`)**. Gọi API để lấy hồ sơ người dùng hiện tại và hiển thị nó.
- **Ngày 20-21: Form Cá nhân hóa**
  - **Học:** Cách làm việc với form và validation trong Vuetify.
  - **Làm:**
    1. Trên `ProfileView.vue`, tạo một form đơn giản với các lựa chọn (ví dụ: `v-radio-group`) cho khẩu vị rủi ro.
    2. Khi người dùng submit form, gọi API `POST /profile` để cập nhật lựa chọn của họ. Hiển thị thông báo thành công.

---

## Tuần 4: Hoàn thiện và Tinh chỉnh (Polish)

*Mục tiêu tuần: Nâng cấp trải nghiệm người dùng và làm cho sản phẩm trông hoàn chỉnh.*

- **Ngày 22-24: Hiển thị Dữ liệu Nâng cao**
  - **Làm:**
    1. Trong `AnalysisView.vue`, sử dụng các component biểu đồ (ví dụ: `chart.js` với wrapper cho Vue) để vẽ một vài biểu đồ đơn giản từ dữ liệu kỹ thuật.
    2. Thiết kế lại cách hiển thị `triggered_rules`. Có thể dùng `v-expansion-panels` của Vuetify để người dùng có thể click và xem chi tiết bằng chứng.
    3. Tích hợp endpoint `/explain` để hiển thị bản tóm tắt văn bản một cách đẹp mắt.
- **Ngày 25-26: Cải thiện Trải nghiệm Người dùng (UX)**
  - **Làm:**
    1. Thêm các "skeleton loader" (hiệu ứng chờ tải) trong khi API đang được gọi, thay vì chỉ hiển thị chữ "Loading...".
    2. Cải thiện việc xử lý lỗi. Hiển thị các thông báo lỗi (`v-alert`) thân thiện hơn cho người dùng.
    3. Đảm bảo ứng dụng có giao diện đáp ứng (responsive) trên các kích thước màn hình khác nhau (sử dụng hệ thống grid của Vuetify).
- **Ngày 27-28: Dọn dẹp, Tái cấu trúc và Ngày đệm**
  - **Làm:**
    1. Xem lại toàn bộ code. Chia nhỏ các component lớn và phức tạp thành các component con, dễ quản lý hơn.
    2. Kiểm tra lại tất cả các luồng hoạt động.
    3. Sử dụng thời gian còn lại để sửa các lỗi nhỏ hoặc tinh chỉnh bất kỳ phần nào bạn cảm thấy chưa hài lòng.