## 3. Form validation

### 3.1. Vấn đề

* Thông báo lỗi không rõ ràng:
  + Lỗi thiếu thông tin: Thông báo lỗi không chỉ ra cụ thể vấn đề gì sai với dữ liệu người dùng nhập vào, khiến họ khó sửa lỗi.
  + Ngôn ngữ khó hiểu: Thông báo lỗi sử dụng ngôn ngữ kỹ thuật hoặc không phù hợp với đối tượng người dùng, gây khó khăn trong việc hiểu và giải quyết lỗi.
  + Vị trí hiển thị không phù hợp: Thông báo lỗi xuất hiện ở vị trí không dễ nhìn thấy, khiến người dùng bỏ lỡ.
* Quy tắc validation quá phức tạp:
  + Yêu cầu quá nhiều thông tin không cần thiết từ người dùng, gây phiền hà và tốn thời gian.
  + Sử dụng các quy tắc validation không phù hợp với ngữ cảnh sử dụng, dẫn đến việc loại bỏ dữ liệu hợp lệ.
  + Quy tắc validation quá phức tạp và khó hiểu, gây khó khăn cho việc triển khai và bảo trì.
* Thiếu xử lý các trường hợp ngoại lệ:
  + Không xử lý các trường hợp dữ liệu đầu vào bất thường, dẫn đến lỗi và treo ứng dụng.
  + Không cung cấp hướng dẫn hoặc giải pháp cho người dùng khi gặp các trường hợp ngoại lệ.
* Khả năng sử dụng kém:
  + Giao diện form validation không thân thiện với người dùng, gây khó khăn trong việc sử dụng.
  + Thiếu các tính năng hỗ trợ như gợi ý, tự động hoàn thành, định dạng dữ liệu, v.v.
  + Phản hồi của form validation chậm hoặc không chính xác.

### 3.2. Cách giải quyết

* Thông báo lỗi không rõ ràng:
  + Giải pháp:
    - Cung cấp thông báo lỗi chi tiết, rõ ràng, giải thích cụ thể vấn đề và cách khắc phục.
    - Sử dụng ngôn ngữ đơn giản, dễ hiểu phù hợp với đối tượng người dùng.
    - Hiển thị thông báo lỗi gần với trường dữ liệu có lỗi để người dùng dễ dàng nhận biết.
* Quy tắc validation quá phức tạp:
  + Giải pháp:
    - Xác định rõ ràng mục đích sử dụng của form và chỉ yêu cầu những thông tin thực sự cần thiết.
    - Áp dụng các quy tắc validation phù hợp với loại dữ liệu và ngữ cảnh sử dụng.
    - Giữ cho quy tắc validation đơn giản, dễ hiểu và dễ dàng bảo trì.
* Thiếu xử lý các trường hợp ngoại lệ:
  + Giải pháp:
    - Lường trước và xử lý các trường hợp dữ liệu đầu vào bất thường.
    - Cung cấp thông báo lỗi và hướng dẫn cụ thể cho người dùng khi gặp các trường hợp ngoại lệ.
    - Cho phép người dùng bỏ qua hoặc nhập dữ liệu thay thế trong trường hợp ngoại lệ.
* Khả năng sử dụng kém:
  + Giải pháp:
    - Thiết kế giao diện form validation đơn giản, trực quan, dễ sử dụng.
    - Thêm các tính năng hỗ trợ như gợi ý, tự động hoàn thành, định dạng dữ liệu để giúp người dùng nhập dữ liệu nhanh chóng và chính xác.
    - Đảm bảo phản hồi của form validation nhanh chóng và chính xác.

### 3.3. Form Validation với React

* Một số phương pháp phổ biến để thực hiện form validation trong React:
* Sử dụng thư viện:
  + Nhiều thư viện React hỗ trợ form validation mạnh mẽ và dễ sử dụng, giúp bạn tiết kiệm thời gian và công sức phát triển. Một số thư viện phổ biến bao gồm:
  + Formik: Thư viện form validation phổ biến nhất cho React, cung cấp nhiều tính năng như xử lý lỗi, quản lý trạng thái, và định dạng dữ liệu.
  + React Hook Form: Thư viện nhẹ và linh hoạt, sử dụng Hooks để quản lý trạng thái và xử lý lỗi form validation.
  + Yup: Thư viện validation schema mạnh mẽ, cho phép bạn định nghĩa các quy tắc validation phức tạp một cách dễ dàng.
* Tự viết code validation:
  + Nếu bạn muốn kiểm soát hoàn toàn quy trình validation, bạn có thể tự viết code validation cho form của mình. Tuy nhiên, cách này đòi hỏi nhiều kiến thức về JavaScript và React, đồng thời tốn nhiều thời gian hơn so với sử dụng thư viện.
* Sử dụng các thành phần React tích hợp:
  + Một số thành phần React tích hợp sẵn chức năng validation cơ bản, như:
    - required prop: Yêu cầu người dùng phải nhập dữ liệu vào trường.
    - pattern prop: Xác định định dạng dữ liệu hợp lệ cho trường (ví dụ: email, số điện thoại).
    - min và max props: Xác định giá trị tối thiểu và tối đa cho trường số.

## Tạo frontend động (dynamic form)

### 4.1. Mô tả

* Dynamic Form hay còn gọi là Form Tự động, là một loại form có thể thay đổi cấu trúc và nội dung dựa trên các tương tác của người dùng hoặc dữ liệu được cung cấp. Nó cung cấp cho người dùng trải nghiệm linh hoạt và cá nhân hóa hơn so với các form truyền thống.
* Đặc điểm chính của Dynamic Form:
* Khả năng thay đổi: Cấu trúc và nội dung của form có thể thay đổi dựa trên các điều kiện nhất định, ví dụ như lựa chọn của người dùng hoặc dữ liệu nhập vào.
* Tính linh hoạt: Dynamic form có thể được sử dụng cho nhiều mục đích khác nhau, từ việc thu thập thông tin đơn giản đến việc tạo các quy trình phức tạp.
* Cá nhân hóa: Dynamic form có thể được cá nhân hóa cho từng người dùng dựa trên nhu cầu và thông tin của họ.
* Dễ sử dụng: Dynamic form thường có giao diện trực quan và dễ sử dụng, giúp người dùng dễ dàng cung cấp thông tin cần thiết.

### 4.2. Công nghệ BEEKAI

* BEEKAI là một trình tạo biểu mẫu trực tuyến miễn phí thế hệ mới giúp các doanh nghiệp và cá nhân tạo các ứng dụng biểu mẫu đẹp và chức năng mà không cần viết bất kỳ mã nào. Nó sử dụng giao diện biểu đồ quy trình công việc trực quan kéo thả để thiết kế biểu mẫu, giúp người dùng ở mọi cấp độ kỹ năng dễ dàng tạo các biểu mẫu phức tạp một cách dễ dàng.
* Tính năng chính của BEEKAI:
  + Trình tạo quy trình công việc trực quan: Tạo biểu mẫu bằng cách sử dụng biểu đồ quy trình công việc trực quan kéo thả, loại bỏ nhu cầu mã hóa.
  + Biểu mẫu có thể tùy chỉnh: Thiết kế biểu mẫu với nhiều yếu tố khác nhau, bao gồm trường văn bản, hộp kiểm, nút radio, menu thả xuống, v.v.
  + Logic điều kiện: Triển khai logic điều kiện để hiển thị hoặc ẩn các trường biểu mẫu dựa trên đầu vào của người dùng, tạo ra các biểu mẫu linh hoạt và được cá nhân hóa.
  + Xác thực dữ liệu: Đảm bảo độ chính xác của dữ liệu bằng cách áp dụng các quy tắc xác thực cho các trường biểu mẫu cụ thể.
  + Tích hợp: Tích hợp biểu mẫu BEEKAI với nhiều công cụ và nền tảng bên thứ ba khác nhau.
  + Thiết kế đáp ứng: Tạo biểu mẫu thích ứng liền mạch với các kích thước màn hình và thiết bị khác nhau.
  + Hợp tác: Hợp tác với các thành viên trong nhóm để thiết kế, xem xét và phê duyệt biểu mẫu.
* Lợi ích khi sử dụng BEEKAI:
  + Tiết kiệm thời gian và công sức: Loại bỏ nhu cầu mã hóa và đơn giản hóa quy trình tạo biểu mẫu.
  + Tạo biểu mẫu phức tạp dễ dàng: Xây dựng các biểu mẫu phức tạp với logic điều kiện và nhiều yếu tố biểu mẫu khác nhau mà không cần viết mã.
  + Nâng cao trải nghiệm người dùng: Thiết kế biểu mẫu thân thiện với người dùng và hấp dẫn phù hợp với các thiết bị khác nhau.
  + Thu thập dữ liệu chính xác: Đảm bảo tính toàn vẹn dữ liệu với các quy tắc xác thực dữ liệu và logic điều kiện.
  + Cải thiện hiệu quả quy trình làm việc: Tự động hóa việc thu thập dữ liệu và tích hợp biểu mẫu với các quy trình làm việc hiện có.
  + Giải pháp tiết kiệm chi phí: Sử dụng trình tạo biểu mẫu trực tuyến miễn phí mà không phát sinh thêm chi phí phần mềm.
* Trường hợp sử dụng cho BEEKAI:
  + Khảo sát khách hàng: Thu thập phản hồi và thông tin chi tiết từ khách hàng bằng cách sử dụng các cuộc khảo sát tương tác.
  + Đăng ký sự kiện: Quản lý đăng ký sự kiện và thu thập thông tin của người tham gia.
  + Tạo khách hàng tiềm năng: Thu thập khách hàng tiềm năng và chi tiết liên hệ cho khách hàng tiềm năng.
  + Ứng tuyển việc làm: Hợp lý hóa quy trình ứng tuyển việc làm và thu thập thông tin ứng viên.
  + Biểu mẫu nội bộ: Tạo biểu mẫu nội bộ cho việc tuyển dụng nhân viên, báo cáo chi phí và các quy trình kinh doanh khác.

### 4.3. Công nghệ React-hook

* React Hooks: Tăng cường sức mạnh cho các Component trong React
* React Hooks là một tính năng mới được giới thiệu trong React 16.8, mang đến cách tiếp cận mới để quản lý trạng thái và các hiệu ứng phụ (side effects) trong các component function.
* Lợi ích của React Hooks:
  + Sử dụng trạng thái và hiệu ứng phụ mà không cần lớp (class): React Hooks cho phép bạn sử dụng các tính năng cốt lõi của React như trạng thái và hiệu ứng phụ trong các component function thay vì phải viết các component class rườm rà.
  + Tái sử dụng logic: React Hooks khuyến khích việc viết logic tái sử dụng bằng cách cho phép bạn tách biệt logic trạng thái và hiệu ứng phụ khỏi component.
  + Dễ học và sử dụng: React Hooks sử dụng cú pháp JavaScript đơn giản và dễ hiểu, giúp bạn dễ dàng học hỏi và sử dụng.
  + Cải thiện hiệu suất: React Hooks có thể giúp cải thiện hiệu suất ứng dụng của bạn bằng cách tối ưu hóa cách quản lý trạng thái và hiệu ứng phụ.
* Các loại React Hooks phổ biến:
  + useState: Quản lý trạng thái cục bộ của một component.
  + useEffect: Thực hiện các hiệu ứng phụ, chẳng hạn như gọi API, đặt thời gian, hoặc đăng ký/hủy đăng ký sự kiện.
  + useContext: Chia sẻ trạng thái và logic giữa các component mà không cần truyền props qua từng cấp.
  + useReducer: Quản lý trạng thái phức tạp hơn với logic cập nhật trạng thái theo kiểu reducer.
  + useMemo: Ghi nhớ giá trị được tính toán để tránh tính toán lại không cần thiết.
  + useCallback: Ghi nhớ callback function để tránh tạo callback function mới mỗi lần render.