**Nghiên cứu: Thiết kế & Triển khai Event Form**

**Tên Sinh Viên:** [Tạ Văn Thắng]

**Mục tiêu:** Báo cáo nghiên cứu chi tiết về yêu cầu, thiết kế, cài đặt và kiểm thử một **Event Form** cho hệ thống quản lý sự kiện — kèm mã nguồn demo (React), kiến trúc backend mẫu, đánh giá và hướng mở rộng.

**Tóm tắt**

Báo cáo này trình bày chi tiết khái niệm, yêu cầu, thiết kế giao diện và kỹ thuật, triển khai một form tạo sự kiện (Event Form) bằng ReactJS, kèm backend mẫu (Express + Node.js), cơ sở dữ liệu đơn giản, chiến lược kiểm thử và các lưu ý về bảo mật, khả năng truy cập (accessibility) và tối ưu hiệu năng. Mục tiêu là cung cấp tài liệu đủ chi tiết để sinh viên hoặc lập trình viên tự triển khai một tính năng tạo/sửa sự kiện hoàn chỉnh.

**Mục lục**

1. Giới thiệu
2. Cơ sở lý thuyết
3. Phân tích yêu cầu
4. Thiết kế chi tiết
   * Mô tả trường dữ liệu
   * Sơ đồ luồng dữ liệu (DFD)
   * Wireframe giao diện
5. Cài đặt (Implementation)
   * Frontend (React)
   * Backend (Express) — mẫu API
   * Database schema
6. Kiểm thử
   * Kế hoạch kiểm thử
   * Test case mẫu
7. Bảo mật & Khả năng truy cập
8. Mở rộng và tối ưu
9. Kết luận
10. Tài liệu tham khảo  
    Appendix: Mã nguồn chính

**1. Giới thiệu**

Sự kiện (Event) là một phần quan trọng trong nhiều hệ thống: hội thảo, lớp học, buổi họp, meetup, hội chợ... Một **Event Form** là giao diện để người quản trị hoặc người dùng tạo mới / chỉnh sửa thông tin sự kiện. Form này phải đảm bảo thu thập đủ thông tin cần thiết, dễ dùng, an toàn và tương thích với nhiều thiết bị.

Mục tiêu báo cáo: mô tả chi tiết cách thiết kế một Event Form thân thiện và robust, từ phân tích yêu cầu đến cài đặt, kiểm thử và triển khai.

**2. Cơ sở lý thuyết**

* **Form trong web**: Là giao diện thu thập dữ liệu (input) từ người dùng. HTML cung cấp các control: input, textarea, select, checkbox, file.
* **Event handling**: Các sự kiện (event) như onChange, onSubmit, onClick dùng để tương tác với dữ liệu. Trong React, các control thường là *controlled components* (nội dung điều khiển bởi state).
* **Validation**: Kiểm tra tính hợp lệ của dữ liệu (bắt buộc, định dạng email, giới hạn chuỗi, số nguyên dương...). Có thể thực hiện client-side (tăng UX) và server-side (bảo mật).
* **Accessibility (A11Y)**: Form phải tương thích với người dùng dùng bàn phím, screen reader: sử dụng label liên kết for, aria attributes, tabindex hợp lý.

**3. Phân tích yêu cầu**

**3.1 Yêu cầu chức năng (Functional)**

* Tạo mới sự kiện với các trường cơ bản: tiêu đề, ngày, giờ, địa điểm, mô tả.
* Thêm tags, upload ảnh minh họa (preview trước khi lưu).
* Validation client-side và server-side.
* Lưu sự kiện vào backend (API) hoặc localStorage.
* Hiển thị danh sách sự kiện đã tạo và xem chi tiết.

**3.2 Yêu cầu phi chức năng (Non-functional)**

* Thân thiện di động (responsive).
* Hiệu suất chấp nhận được (render nhanh, tải ảnh nhẹ).
* Bảo mật: ngăn XSS, CSRF, validate file upload.
* Khả năng mở rộng: dễ thêm trường mới, tích hợp API.

**4. Thiết kế chi tiết**

**4.1 Mô tả trường dữ liệu**

Bảng dưới đây liệt kê các trường chính cho Event:

| **Tên trường** | **Kiểu** | **Bắt buộc** | **Mô tả** |
| --- | --- | --- | --- |
| title | string | Có | Tên sự kiện, tối đa 150 ký tự |
| date | date | Có | Ngày tổ chức |
| time | time | Có | Giờ bắt đầu |
| location | string | Không | Địa điểm |
| description | text | Không | Mô tả chi tiết |
| tags | array[string] | Không | Danh sách tag (kỹ năng, chủ đề) |
| image | file(url) | Không | Ảnh minh họa |
| capacity | number | Không | Số lượng tối đa |
| organizerName | string | Không | Tên người tổ chức |
| organizerEmail | string | Có | Email người tổ chức |
| isPublic | boolean | Có | Công khai hay riêng tư |

**4.2 Sơ đồ luồng dữ liệu (DFD) — Level 0 và Level 1**

**DFD Level 0 (tổng quan)**

[User] --> (Event Form UI) --> (Frontend) --> (API Server) --> (Database)

**DFD Level 1 (chi tiết hơn)**

User -> (Input data into form) -> Frontend (validate client-side) -> If ok -> POST /api/events -> Server validates -> store to DB -> return created event

-> If invalid -> 400 + error

Frontend can also -> GET /api/events to fetch list

**4.3 Wireframe (mô tả giao diện)**

Mô tả giao diện form theo thứ tự vertical: Title (input), Date & Time (inline), Location (input), Category (select), Tags (input + chips), Image (file + preview), Description (textarea), Organizer (name, email), Capacity (number), Toggle public/private, Buttons (Save, Cancel).

Gợi ý layout: responsive, trên mobile hiển thị full-width; trên desktop chia thành 2 cột: cột trái form, cột phải preview.

**5. Cài đặt (Implementation)**

Phần này bao gồm mã nguồn frontend (React) chi tiết đã comment, mẫu backend API (Express) và schema DB.

**5.1 Frontend — React (kiến trúc)**

* Component chính: EventForm.jsx (controlled form)
* State lưu form: object formData + tags + imagePreview + errors
* Handlers: handleChange, handleTagAdd, handleImageChange, handleSubmit
* Gửi dữ liệu: fetch('/api/events', { method: 'POST', body: formDataFormData }) hoặc lưu localStorage.

**5.2 Mã nguồn (EventForm.jsx) — tóm tắt**

Lưu ý: toàn bộ mã đầy đủ nằm trong phần Appendix.

Các điểm kỹ thuật cần chú ý trong mã:

* Sử dụng FormData khi gửi file image tới backend.
* Client-side validation trước khi gửi.
* Xử lý trạng thái loading và hiển thị thông báo lỗi.

**5.3 Backend mẫu (Express + Node.js)**

Mô tả API:

* POST /api/events — tạo event mới. Nhận multipart/form-data (khi có ảnh) hoặc application/json.
* GET /api/events — lấy danh sách events.
* GET /api/events/:id — lấy chi tiết event.

Mã ví dụ (Express):

// index.js (Express - minimal)

const express = require('express');

const multer = require('multer');

const app = express();

const upload = multer({ dest: 'uploads/' });

app.use(express.json());

app.post('/api/events', upload.single('image'), (req, res) => {

const { title, date, time, location, description, organizerEmail } = req.body;

if (!title || !date || !time || !organizerEmail) return res.status(400).json({ error: 'Thiếu trường bắt buộc' });

// TODO: Lưu vào DB. Giả sử lưu thành object

const newEvent = { id: Date.now(), title, date, time, location, description, image: req.file ? req.file.path : null };

// respond

res.status(201).json(newEvent);

});

app.listen(3001, () => console.log('API running on 3001'));

**5.4 Database schema (MySQL / PostgreSQL simple)**

CREATE TABLE events (

id BIGINT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

date DATE NOT NULL,

time TIME NOT NULL,

location VARCHAR(255),

description TEXT,

image\_url VARCHAR(500),

capacity INT,

organizer\_name VARCHAR(255),

organizer\_email VARCHAR(255) NOT NULL,

is\_public BOOLEAN DEFAULT TRUE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE event\_tags (

event\_id BIGINT REFERENCES events(id),

tag VARCHAR(100),

PRIMARY KEY (event\_id, tag)

);

**6. Kiểm thử**

**6.1 Kế hoạch kiểm thử**

* **Unit tests**: test các hàm validation, xử lý tags.
* **Integration tests**: test luồng tạo event (frontend -> backend -> DB). Có thể dùng Jest + Supertest.
* **Manual tests (UAT)**: nhập form với dữ liệu hợp lệ, thiếu dữ liệu, email sai định dạng, upload file lớn, thử trên mobile.

**6.2 Test case mẫu**

| **ID** | **Mô tả** | **Bước thực hiện** | **Kết quả mong đợi** |
| --- | --- | --- | --- |
| TC01 | Tạo event hợp lệ | Điền title, date, time, email; bấm Lưu | Event được tạo, hiển thị trong danh sách |
| TC02 | Thiếu trường bắt buộc | Bỏ trống title; bấm Lưu | Hiển thị thông báo lỗi client và không gửi request |
| TC03 | Email sai định dạng | Điền email abc@ | Hiển thị lỗi validation |
| TC04 | Upload file không phải ảnh | Chọn file .exe | Hiển thị lỗi hoặc chặn upload |
| TC05 | Tags | Thêm tag bằng Enter | Tag hiển thị dưới dạng chip |

**6.3 Unit test ví dụ (Jest)**

// validation.test.js

const { validateEvent } = require('./validation');

test('validateEvent returns error when title missing', () => {

const data = { title: '', date: '2025-01-01', time: '10:00', organizerEmail: 'a@b.com' };

expect(validateEvent(data).title).toBeTruthy();

});

**7. Bảo mật & Khả năng truy cập**

**7.1 Bảo mật**

* **Server-side validation**: không tin tưởng client.
* **XSS**: Escaping dữ liệu khi hiển thị mô tả (ví dụ: dùng innerText, không dùng dangerouslySetInnerHTML nếu không lọc).
* **CSRF**: Nếu dùng cookie-based auth, bảo vệ bằng CSRF token.
* **File upload**: kiểm tra MIME type, giới hạn kích thước, lưu file ra vùng an toàn.
* **Rate limiting / authentication**: tránh spam.

**7.2 Accessibility (A11Y)**

* Sử dụng <label for="..."> hoặc label bọc input.
* Thêm aria-invalid, aria-describedby khi có lỗi.
* Bảo đảm keyboard accessible: tab order hợp lý, các button có type rõ ràng.
* Đảm bảo contrast màu, kích thước font.

**8. Mở rộng và tối ưu**

* **Lưu offline**: dùng localStorage hoặc IndexedDB để tạm lưu form khi mất mạng.
* **Chỉnh sửa & xóa**: bổ sung endpoints PATCH/DELETE.
* **Quản lý ảnh**: upload lên cloud storage (S3), trả về URL.
* **Tối ưu performance**: debounce input tìm kiếm, lazy-load ảnh, pagination cho danh sách events.
* **i18n**: hỗ trợ đa ngôn ngữ.

**9. Kết luận**

Bài báo cáo trình bày từ lý thuyết đến thực hành thiết kế một Event Form đầy đủ: phân tích yêu cầu, thiết kế giao diện, cài đặt frontend bằng React, backend mẫu, database schema, kiểm thử và các lưu ý quan trọng về bảo mật và truy cập. Với hướng dẫn này, bạn có thể triển khai tính năng tạo sự kiện trong dự án thực tế và mở rộng theo nhu cầu.

**10. Tài liệu tham khảo**

* React official docs — Forms and controlled components
* MDN Web Docs — HTML forms, accessibility
* Express.js docs — file upload (multer)
* OWASP — các hướng dẫn bảo mật web cơ bản

**Appendix: Mã nguồn (chi tiết)**

**A. EventForm.jsx (React) — ví dụ hoàn chỉnh**

import React, { useState, useRef } from 'react';

export default function EventForm() {

const [formData, setFormData] = useState({

title: '', date: '', time: '', location: '', description: '', organizerName: '', organizerEmail: '', capacity: '', isPublic: true

});

const [tags, setTags] = useState([]);

const [tagInput, setTagInput] = useState('');

const [imagePreview, setImagePreview] = useState(null);

const [errors, setErrors] = useState({});

const fileRef = useRef(null);

const handleChange = (e) => {

const { name, value, type, checked } = e.target;

setFormData(prev => ({ ...prev, [name]: type === 'checkbox' ? checked : value }));

};

const handleTagKeyDown = (e) => {

if (e.key === 'Enter') { e.preventDefault(); addTag(); }

};

const addTag = () => {

const t = tagInput.trim();

if (t && !tags.includes(t)) { setTags(prev => [...prev, t]); setTagInput(''); }

};

const removeTag = (t) => setTags(prev => prev.filter(x => x !== t));

const handleImageChange = (e) => {

const file = e.target.files && e.target.files[0];

if (file) { setImagePreview(URL.createObjectURL(file)); }

else setImagePreview(null);

};

const validate = () => {

const err = {};

if (!formData.title.trim()) err.title = 'Tiêu đề là bắt buộc';

if (!formData.date) err.date = 'Chọn ngày';

if (!formData.time) err.time = 'Chọn giờ';

if (!formData.organizerEmail) err.organizerEmail = 'Email bắt buộc';

else if (!/^\S+@\S+\.\S+$/.test(formData.organizerEmail)) err.organizerEmail = 'Email không hợp lệ';

setErrors(err);

return Object.keys(err).length === 0;

};

const handleSubmit = async (e) => {

e.preventDefault();

if (!validate()) return;

// Prepare data to send

const fd = new FormData();

Object.keys(formData).forEach(k => fd.append(k, formData[k]));

fd.append('tags', JSON.stringify(tags));

if (fileRef.current && fileRef.current.files[0]) fd.append('image', fileRef.current.files[0]);

try {

// Example: POST to /api/events

const res = await fetch('/api/events', { method: 'POST', body: fd });

if (!res.ok) throw new Error('Server error');

const created = await res.json();

alert('Tạo sự kiện thành công!');

// reset

setFormData({ title: '', date: '', time: '', location: '', description: '', organizerName: '', organizerEmail: '', capacity: '', isPublic: true });

setTags([]); setImagePreview(null); if (fileRef.current) fileRef.current.value = '';

} catch (err) {

alert('Lỗi khi tạo sự kiện: ' + err.message);

}

};

return (

<div style={{ maxWidth: 900, margin: 'auto' }}>

<h1>Tạo sự kiện</h1>

<form onSubmit={handleSubmit} noValidate>

<div>

<label htmlFor="title">Tiêu đề\*</label>

<input id="title" name="title" value={formData.title} onChange={handleChange} />

{errors.title && <div style={{ color: 'red' }}>{errors.title}</div>}

</div>

<div style={{ display: 'flex', gap: 8 }}>

<div>

<label>Ngày\*</label>

<input type="date" name="date" value={formData.date} onChange={handleChange} />

{errors.date && <div style={{ color: 'red' }}>{errors.date}</div>}

</div>

<div>

<label>Giờ\*</label>

<input type="time" name="time" value={formData.time} onChange={handleChange} />

{errors.time && <div style={{ color: 'red' }}>{errors.time}</div>}

</div>

</div>

<div>

<label>Địa điểm</label>

<input name="location" value={formData.location} onChange={handleChange} />

</div>

<div>

<label>Mô tả</label>

<textarea name="description" rows={4} value={formData.description} onChange={handleChange} />

</div>

<div>

<label>Tags</label>

<div>

{tags.map(t => (

<span key={t} style={{ padding: '4px 8px', background: '#eee', marginRight: 6 }}>{t} <button type="button" onClick={() => removeTag(t)}>x</button></span>

))}

<input value={tagInput} onChange={e => setTagInput(e.target.value)} onKeyDown={handleTagKeyDown} placeholder="Gõ tag rồi Enter" />

<button type="button" onClick={addTag}>Thêm</button>

</div>

</div>

<div>

<label>Ảnh</label>

<input type="file" accept="image/\*" ref={fileRef} onChange={handleImageChange} />

{imagePreview && <img src={imagePreview} alt="preview" style={{ width: 160, marginTop: 8 }} />}

</div>

<div>

<label>Email người tổ chức\*</label>

<input name="organizerEmail" value={formData.organizerEmail} onChange={handleChange} />

{errors.organizerEmail && <div style={{ color: 'red' }}>{errors.organizerEmail}</div>}

</div>

<div style={{ marginTop: 12 }}>

<button type="submit">Tạo sự kiện</button>

</div>

</form>

</div>

);

}

**B. Backend sample (Express + multer)**

const express = require('express');

const multer = require('multer');

const upload = multer({ dest: './uploads', limits: { fileSize: 5 \* 1024 \* 1024 } });

const app = express();

app.use(express.json());

app.post('/api/events', upload.single('image'), (req, res) => {

const body = req.body;

// server-side validation

if (!body.title || !body.date || !body.time || !body.organizerEmail) return res.status(400).json({ error: 'Thiếu trường' });

const event = { id: Date.now(), ...body, image: req.file ? req.file.filename : null };

// Save to DB...

res.status(201).json(event);

});

app.listen(3001, () => console.log('Server started'));

**Ghi chú kết thúc**

Báo cáo này thiết kế để dễ mở rộng: bạn có thể chuyển đổi giao diện sang Tailwind CSS, Material UI, hoặc kết nối authentication để chỉ người dùng được phép mới tạo sự kiện. Nếu bạn muốn mình soạn **file .docx** hoàn chỉnh (định dạng Word) sẵn để tải xuống, mình có thể tạo file đó và cung cấp link tải.