Kiến trúc Frontend cho Hệ thống Chấm công

1. Tech Stack Đề xuất

Core Framework

- **React 18** với TypeScript
- **Next.js 14** (App Router) - cho SSR và routing
- **Tailwind CSS** - cho styling
- **Zustand** hoặc **Redux Toolkit** - cho state management

UI Components

- **Ant Design** hoặc **Material-UI** - cho component library
- **React Hook Form** - cho form handling
- **Zod** - cho validation
- **Framer Motion** - cho animations

Real-time & API

- **Socket.io Client** - cho real-time MQTT notifications
- **Axios** hoặc **React Query (TanStack Query)** - cho API calls
- **SWR** - cho data fetching và caching

2. Cấu trúc thư mục Frontend

```
frontend/
├── public/
├── src/
│   ├── components/           # Reusable components
│   │   ├── ui/              # Basic UI components
│   │   ├── forms/           # Form components
│   │   ├── layout/          # Layout components
│   │   └── common/          # Common components
│   ├── pages/               # Page components (Next.js pages)
│   │   ├── dashboard/
│   │   ├── devices/
│   │   ├── users/
│   │   ├── history/
│   │   └── settings/
│   ├── hooks/               # Custom hooks
│   ├── services/            # API services
│   ├── store/               # State management
│   ├── types/               # TypeScript types
│   ├── utils/               # Utility functions
│   ├── constants/           # Constants
│   └── styles/              # Global styles
├── package.json
├── tailwind.config.js
├── next.config.js
└── tsconfig.json
```

3. Các Module/Pages chính

3.1 Dashboard Module

```
pages/dashboard/
├── index.tsx                # Dashboard overview
├── components/
│   ├── StatsCard.tsx
│   ├── AttendanceChart.tsx
│   ├── RecentActivity.tsx
│   └── QuickActions.tsx
```

Chức năng:

- Hiển thị thống kê tổng quan (số người dùng, thiết bị, lượt chấm công)
- Biểu đồ chấm công theo thời gian
- Hoạt động gần đây
- Notifications real-time

3.2 Users Module

```
pages/users/
├── index.tsx                # User list
├── create.tsx               # Create user
├── [id]/
│   ├── index.tsx           # User detail
│   ├── edit.tsx            # Edit user
│   └── fingerprint.tsx     # Manage fingerprint
├── components/
│   ├── UserList.tsx
│   ├── UserForm.tsx
│   ├── UserCard.tsx
│   ├── FingerprintModal.tsx
│   └── CardNumberModal.tsx
```

Chức năng:

- CRUD operations cho users
- Quản lý fingerprint và card number
- Gán user vào devices
- Xem lịch sử chấm công của user

3.3 Devices Module

```
pages/devices/
├── index.tsx                # Device list
├── create.tsx               # Create device
├── [id]/
│   ├── index.tsx           # Device detail
│   ├── edit.tsx            # Edit device
│   └── users.tsx           # Manage device users
├── components/
│   ├── DeviceList.tsx
│   ├── DeviceForm.tsx
│   ├── DeviceCard.tsx
│   ├── DeviceStatus.tsx
│   └── UserDeviceManager.tsx
```

Chức năng:

- CRUD operations cho devices
- Device verification process
- Quản lý users trên device

- Sync/delete all users
- Monitor device status

3.4 History Module

```
pages/history/
├── index.tsx                # Attendance history
├── reports.tsx              # Reports
├── components/
│   ├── HistoryTable.tsx
│   ├── HistoryFilter.tsx
│   ├── AttendanceReport.tsx
│   └── ExportButtons.tsx
```

Chức năng:

- Xem lịch sử chấm công
- Lọc theo user, device, ngày
- Export reports
- Thống kê attendance

4. Services Layer

4.1 API Services

typescript

```typescript
// services/api.ts
export class ApiService {
  private baseURL = process.env.NEXT_PUBLIC_API_URL;

  // Users
  async createUser(data: CreateUserDto) { }
  async getUsers() { }
  async updateUser(id: string, data: UpdateUserDto) { }
  async deleteUser(id: string) { }
  async addFingerprint(data: AddFingerprintDto) { }
  async addCardNumber(data: AddCardNumberDto) { }

  // Devices
  async createDevice(data: CreateDeviceDto) { }
  async getDevices() { }
  async addUserToDevice(deviceId: string, userId: string) { }
  async removeUserFromDevice(deviceId: string, userId: string) { }
  async syncAllUsers(deviceId: string) { }
  async deleteAllUsers(deviceId: string) { }

  // History
  async getAttendanceHistory(filters?: HistoryFilters) { }
  async exportHistory(format: 'csv' | 'excel') { }
}
```

4.2 WebSocket Service

typescript

```typescript
// services/websocket.ts
export class WebSocketService {
  private socket: Socket;

  connect() {
    this.socket = io(process.env.NEXT_PUBLIC_WS_URL);
    this.setupEventListeners();
  }

  private setupEventListeners() {
    this.socket.on('attendance-noti', this.handleAttendanceNotification);
    this.socket.on('device-status', this.handleDeviceStatus);
  }

  private handleAttendanceNotification(data: any) {
    // Show notification to user
  }
}
```

5. State Management

5.1 Zustand Store Structure

typescript

```typescript
// store/useUserStore.ts
interface UserStore {
  users: User[];
  selectedUser: User | null;
  loading: boolean;
  error: string | null;

  fetchUsers: () => Promise<void>;
  createUser: (data: CreateUserDto) => Promise<void>;
  updateUser: (id: string, data: UpdateUserDto) => Promise<void>;
  deleteUser: (id: string) => Promise<void>;
  setSelectedUser: (user: User) => void;
}

// store/useDeviceStore.ts
interface DeviceStore {
  devices: Device[];
  selectedDevice: Device | null;
  loading: boolean;

  fetchDevices: () => Promise<void>;
  createDevice: (data: CreateDeviceDto) => Promise<void>;
  verifyDevice: (deviceMac: string) => Promise<boolean>;
}

// store/useHistoryStore.ts
interface HistoryStore {
  history: AttendanceRecord[];
  filters: HistoryFilters;
  loading: boolean;

  fetchHistory: () => Promise<void>;
  setFilters: (filters: HistoryFilters) => void;
  exportHistory: (format: string) => Promise<void>;
}
```

6. TypeScript Types

typescript

```typescript
// types/user.ts
export interface User {
  _id: string;
  userId: string;
  name: string;
  email?: string;
  fingerTemplate?: string;
  cardNumber?: string;
  createdAt: Date;
  updatedAt?: Date;
  devices: string[];
  history: string[];
}

export interface CreateUserDto {
  userId: string;
  name: string;
  createdAt: Date;
}

// types/device.ts
export interface Device {
  _id: string;
  deviceMac: string;
  description: string;
  users: string[];
  status?: 'online' | 'offline';
}

// types/history.ts
export interface AttendanceRecord {
  _id: string;
  user: User;
  date: Date;
  time_in: string;
  time_out?: string;
}
```

7. Key Features Implementation

7.1 Real-time Notifications

- Sử dụng Socket.io để nhận notifications từ MQTT
- Toast notifications khi có người chấm công

- Real-time update attendance list

7.2 Device Management

- Device verification flow với loading states
- Real-time device status monitoring
- Batch operations (sync all, delete all users)

7.3 Fingerprint Management

- Modal flows cho việc thêm fingerprint
- Progress tracking cho fingerprint enrollment
- Error handling cho device communication

7.4 Reports & Analytics

- Interactive charts với Chart.js hoặc Recharts
- Export functionality
- Date range filtering
- User/Device specific reports

8. Security & Performance

8.1 Security

- JWT token management
- API request interceptors
- Input validation với Zod
- CSRF protection

8.2 Performance

- Code splitting với Next.js
- Image optimization
- API response caching với React Query
- Lazy loading cho large lists
- Debounced search inputs

9. Mobile Responsiveness

- Mobile-first design approach
- Touch-friendly UI components

- Responsive tables với horizontal scroll
- Mobile navigation patterns

10. Development Tools

- **ESLint + Prettier** - code formatting
- **Husky** - git hooks
- **Jest + Testing Library** - testing
- **Storybook** - component documentation
- **Cypress** - E2E testing

Này là kiến trúc tổng quan cho frontend. Bạn có muốn tôi detail hóa thêm phần nào cụ thể không?