เอกสารข้อกำหนดความต้องการซอฟต์แวร์

Software Requirements Specification (SRS)

ระบบจัดการโครงสร้างองค์กร (Organization Structure Management System)

ข้อมูลเอกสาร

- **รหัสเอกสาร**: SRS-ORGSTRUCT-001
- **เวอร์ชัน**: 1.0
- **วันที่จัดทำ**: [วันที่ปัจจุบัน]
- **สถานะ**: ฉบับร่าง (Draft)
- **ผู้จัดทำ**: [ชื่อผู้จัดทำ]
- **ผู้ทบทวน**: [ชื่อผู้ทบทวน]
- **ผู้อนุมัติ**: [ชื่อผู้อนุมัติ]

การควบคุมเวอร์ชัน

เวอร์ชัน	วันที่	รายละเอียดการแก้ไข	ผู้แก้ไข
1.0	[วันที่]	สร้างเอกสารฉบับแรก	[ชื่อ]

สารบัญ

---

1. บทนำ

1.1 วัตถุประสงค์ (Purpose)

เอกสารฉบับนี้จัดทำขึ้นเพื่อกำหนดความต้องการทั้งหมดสำหรับการพัฒนา "ระบบจัดการโครงสร้างองค์กร" ซึ่งเป็นระบบกลางสำหรับจัดการข้อมูลโครงสร้างองค์กรแบบลำดับชั้น ประกอบด้วย บริษัท (Company), สาขา (Branch), ฝ่าย (Division) และแผนก (Department) โดยให้บริการผ่าน API เพื่อให้ระบบอื่นๆ ในองค์กรสามารถเรียกใช้ข้อมูลได้จากแหล่งเดียว

1.2 ขอบเขต (Scope)

ระบบนี้ครอบคลุม:

- การจัดการข้อมูลโครงสร้างองค์กร 4 ระดับ
- การให้บริการ RESTful API สำหรับระบบอื่น
- หน้าเว็บสำหรับจัดการข้อมูล
- ระบบจัดการสิทธิ์การเข้าถึง API

ระบบนี้ไม่ครอบคลุม:

- การจัดการข้อมูลพนักงาน
- การจัดการตำแหน่งงาน
- การเชื่อมต่อกับระบบ HR โดยตรง

1.3 คำนิยาม คำย่อ และตัวย่อ (Definitions, Acronyms, and Abbreviations)

คำศัพท์	คำอธิบาย
API	Application Programming Interface - ช่องทางการเชื่อมต่อระหว่างระบบ
REST	Representational State Transfer - รูปแบบสถาปัตยกรรมของ Web Service
MVC	Model-View-Controller - รูปแบบการออกแบบซอฟต์แวร์
CRUD	Create, Read, Update, Delete - การดำเนินการพื้นฐานกับข้อมูล
JSON	JavaScript Object Notation - รูปแบบข้อมูลสำหรับแลกเปลี่ยนข้อมูล
SQL	Structured Query Language - ภาษาสำหรับจัดการฐานข้อมูล

1.4 เอกสารอ้างอิง (References)

- มาตรฐาน IEEE 830-1998 สำหรับ Software Requirements Specifications
- RESTful API Design Guidelines
- Microsoft SQL Server Documentation
- Node.js Best Practices
- Tailwind CSS Documentation

1.5 ภาพรวม (Overview)

เอกสารนี้ประกอบด้วย 6 ส่วนหลัก:

- ส่วนที่ 1: บทนำ - อธิบายวัตถุประสงค์และขอบเขต
- ส่วนที่ 2: คำอธิบายโดยรวม - อธิบายภาพรวมของระบบ
- ส่วนที่ 3: ความต้องการเฉพาะ - รายละเอียดความต้องการด้านฟังก์ชัน
- ส่วนที่ 4: ความต้องการที่ไม่ใช่ด้านฟังก์ชัน - ประสิทธิภาพ ความปลอดภัย

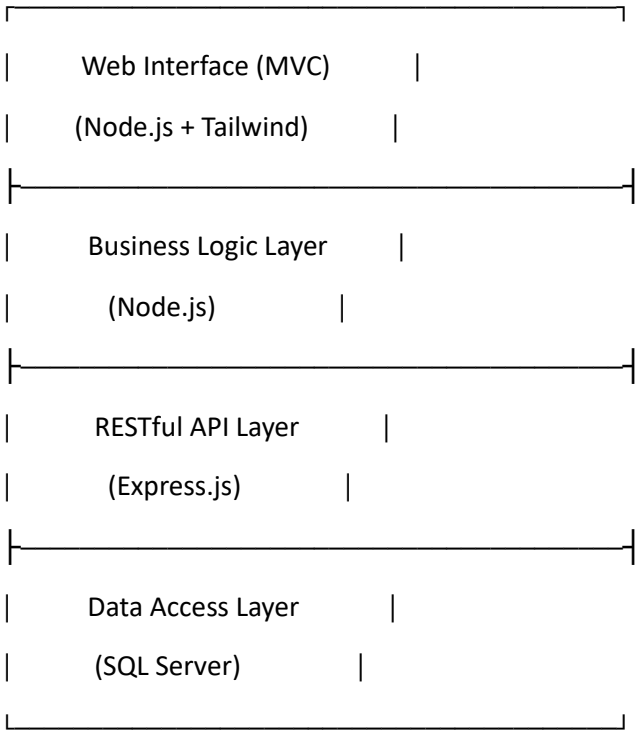
- ส่วนที่ 5: ความต้องการด้านระบบ - สภาพแวดล้อมและเทคโนโลยี
- ส่วนที่ 6: ภาคผนวก - ข้อมูลเพิ่มเติม

2. คำอธิบายโดยรวม

2.1 มุมมองของผลิตภัณฑ์ (Product Perspective)

ระบบจัดการโครงสร้างองค์กรเป็นระบบใหม่ที่จะมาแทนที่การจัดเก็บข้อมูลโครงสร้างองค์กรแบบกระจายในแต่ละระบบ โดยจะทำหน้าที่เป็นแหล่งข้อมูลกลาง (Single Source of Truth) สำหรับข้อมูลโครงสร้างองค์กรทั้งหมด

2.1.1 ส่วนประกอบของระบบ



2.1.2 ความสัมพันธ์กับระบบอื่น

- **ระบบ ERP**: ดึงข้อมูลโครงสร้างเพื่อใช้ในการบันทึกบัญชี
- **ระบบ Payroll**: ใช้ข้อมูลสังกัดสำหรับการจ่ายเงินเดือน
- **ระบบ HR**: ใช้ข้อมูลโครงสร้างในการบริหารพนักงาน
- **ระบบอื่น ๆ**: ระบบใดก็ตามที่ต้องการข้อมูลโครงสร้างองค์กร

2.2 หน้าที่ของผลิตภัณฑ์ (Product Functions)

1. **การจัดการข้อมูลโครงสร้างองค์กร**
    - จัดการข้อมูลบริษัท
    - จัดการข้อมูลสาขา (ถ้ามี)
    - จัดการข้อมูลฝ่าย
    - จัดการข้อมูลแผนก

2. **การให้บริการ API**
    - ให้บริการข้อมูลผ่าน RESTful API
    - รองรับการค้นหาและกรองข้อมูล
    - รองรับการดึงข้อมูลแบบมีความสัมพันธ์
3. **การจัดการสิทธิ์**
    - จัดการ API Key
    - กำหนดสิทธิ์การเข้าถึง
    - บันทึกประวัติการใช้งาน

2.3 คุณลักษณะของผู้ใช้ (User Characteristics)

ประเภทผู้ใช้	คำอธิบาย	ทักษะที่ต้องการ
ผู้ดูแลข้อมูล	บุคลากรที่รับผิดชอบการจัดการข้อมูลโครงสร้างองค์กร	ความรู้พื้นฐานคอมพิวเตอร์ การใช้งานเว็บไซต์
นักพัฒนาระบบ	ผู้พัฒนาที่ต้องการเชื่อมต่อกับ API	ความรู้ด้าน REST API, JSON, HTTP
ผู้ดูแลระบบ	ผู้ดูแลระบบ IT	ความรู้ด้าน Database, Server, Network

2.4 ข้อจำกัด (Constraints)

1. **ข้อจำกัดด้านเทคโนโลยี**
    - ต้องพัฒนาด้วย Node.js
    - ต้องใช้ Microsoft SQL Server เป็นฐานข้อมูล
    - Frontend ต้องใช้ Tailwind CSS
    - ต้องออกแบบตาม MVC Pattern
2. **ข้อจำกัดด้านธุรกิจ**
    - ข้อมูลต้องเก็บอยู่ภายในประเทศเท่านั้น
    - ต้องรองรับทั้งองค์กรที่มีและไม่มีสาขา
    - การเปลี่ยนแปลงข้อมูลต้องมีผลทันที

2.5 สมมติฐานและการพึ่งพา (Assumptions and Dependencies)

1. **สมมติฐาน**
    - องค์กรมีโครงสร้างแบบลำดับชั้นไม่เกิน 4 ระดับ
    - ผู้ใช้งานมีความรู้พื้นฐานการใช้งานคอมพิวเตอร์
    - มีโครงสร้างพื้นฐาน Network ที่พร้อมใช้งาน

2. **การพึ่งพา**
    - Microsoft SQL Server ต้องติดตั้งและพร้อมใช้งาน
    - Node.js Runtime Environment
    - Web Server สำหรับ Deploy ระบบ

3. ความต้องการเฉพาะ (Specific Requirements)

3.1 ความต้องการด้านฟังก์ชัน (Functional Requirements)

3.1.1 การจัดการข้อมูลบริษัท (Company Management)

FR-COM-001: ระบบต้องเก็บข้อมูลบริษัท

คำอธิบาย: ระบบต้องสามารถจัดเก็บข้อมูลพื้นฐานของบริษัท

ข้อมูลที่ต้องเก็บ:

ชื่อฟิลด์	ประเภทข้อมูล	ขนาด	คำอธิบาย	ข้อกำหนด
company_code	VARCHAR	20	รหัสบริษัท	บังคับ, ไม่ซ้ำ
company_name_th	NVARCHAR	200	ชื่อบริษัทภาษาไทย	บังคับ
company_name_en	VARCHAR	200	ชื่อบริษัทภาษาอังกฤษ	ไม่บังคับ
tax_id	VARCHAR	13	เลขประจำตัวผู้เสียภาษี	ไม่บังคับ
is_active	BIT	-	สถานะการใช้งาน	ค่าเริ่มต้น = 1
created_date	DATETIME	-	วันที่สร้าง	ระบบสร้างอัตโนมัติ
created_by	VARCHAR	50	ผู้สร้าง	ระบบบันทึกอัตโนมัติ
updated_date	DATETIME	-	วันที่แก้ไข	ระบบอัพเดทอัตโนมัติ
updated_by	VARCHAR	50	ผู้แก้ไข	ระบบบันทึกอัตโนมัติ

เงื่อนไข:

- รหัสบริษัทต้องไม่ซ้ำกันในระบบ
- ชื่อบริษัทภาษาไทยต้องกรอก
- เลขประจำตัวผู้เสียภาษีต้องเป็นตัวเลข 13 หลัก (ถ้ากรอก)

FR-COM-002: การดำเนินการกับข้อมูลบริษัท

ระบบต้องรองรับการดำเนินการดังนี้:

- สร้างข้อมูลบริษัทใหม่

- แก้ไขข้อมูลบริษัท (ยกเว้นรหัสบริษัท)
- เปลี่ยนสถานะการใช้งาน (Active/Inactive)
- ค้นหาบริษัทด้วยรหัสหรือชื่อ
- แสดงรายการบริษัททั้งหมด

3.1.2 การจัดการข้อมูลสาขา (Branch Management)

FR-BRN-001: ระบบต้องเก็บข้อมูลสาขา

คำอธิบาย: ระบบต้องรองรับการจัดเก็บข้อมูลสาขา โดยบางบริษัทอาจไม่มีสาขา

ข้อมูลที่ต้องเก็บ:

ชื่อฟิลด์	ประเภทข้อมูล	ขนาด	คำอธิบาย	ข้อกำหนด
branch_code	VARCHAR	20	รหัสสาขา	บังคับ, ไม่ซ้ำ
branch_name	NVARCHAR	200	ชื่อสาขา	บังคับ
company_code	VARCHAR	20	รหัสบริษัทที่สังกัด	บังคับ, FK
is_headquarters	BIT	-	เป็นสำนักงานใหญ่	ค่าเริ่มต้น = 0
is_active	BIT	-	สถานะการใช้งาน	ค่าเริ่มต้น = 1

เงื่อนไข:

- รหัสสาขาต้องไม่ซ้ำกันในระบบ
- สาขาต้องสังกัดบริษัทที่มีอยู่จริง
- แต่ละบริษัทมีสำนักงานใหญ่ได้แห่งเดียว

FR-BRN-002: การดำเนินการกับข้อมูลสาขา

- สร้างสาขาใหม่
- แก้ไขข้อมูลสาขา
- เปลี่ยนสถานะการใช้งาน
- ค้นหาสาขาในบริษัท

3.1.3 การจัดการข้อมูลฝ่าย (Division Management)

FR-DIV-001: ระบบต้องเก็บข้อมูลฝ่าย

คำอธิบาย: ระบบต้องจัดเก็บข้อมูลฝ่าย โดยฝ่ายอาจสังกัดบริษัทโดยตรง (กรณีไม่มีสาขา) หรือสังกัดสาขา

ข้อมูลที่ต้องเก็บ:

ชื่อฟิลด์	ประเภทข้อมูล	ขนาด	คำอธิบาย	ข้อกำหนด
division_code	VARCHAR	20	รหัสฝ่าย	บังคับ, ไม่ซ้ำ
division_name	NVARCHAR	200	ชื่อฝ่าย	บังคับ
company_code	VARCHAR	20	รหัสบริษัท	บังคับ, FK
branch_code	VARCHAR	20	รหัสสาขา	ไม่บังคับ, FK
is_active	BIT	-	สถานะการใช้งาน ค่าเริ่มต้น = 1	

เงื่อนไข:

- รหัสฝ่ายต้องไม่ซ้ำกันในระบบ
- ฝ่ายต้องสังกัดบริษัทเสมอ
- ถ้ามีสาขา ฝ่ายอาจสังกัดสาขา
- ถ้าไม่มีสาขา ฝ่ายสังกัดบริษัทโดยตรง

FR-DIV-002: การดำเนินการกับข้อมูลฝ่าย

- สร้างฝ่ายใหม่
- แก้ไขข้อมูลฝ่าย
- ย้ายฝ่ายไปสังกัดสาขาอื่น (ถ้ามี)
- เปลี่ยนสถานะการใช้งาน

3.1.4 การจัดการข้อมูลแผนก (Department Management)

FR-DEP-001: ระบบต้องเก็บข้อมูลแผนก

คำอธิบาย: ระบบต้องจัดเก็บข้อมูลแผนก โดยแผนกต้องสังกัดฝ่ายเสมอ

ข้อมูลที่ต้องเก็บ:

ชื่อฟิลด์	ประเภทข้อมูล	ขนาด	คำอธิบาย	ข้อกำหนด
department_code	VARCHAR	20	รหัสแผนก	บังคับ, ไม่ซ้ำ
department_name	NVARCHAR	200	ชื่อแผนก	บังคับ
division_code	VARCHAR	20	รหัสฝ่ายที่สังกัด	บังคับ, FK
is_active	BIT	-	สถานะการใช้งาน ค่าเริ่มต้น = 1	

เงื่อนไข:

- รหัสแผนกต้องไม่ซ้ำกันในระบบ

- แผนกต้องสังกัดฝ่ายที่มีอยู่จริงและใช้งานอยู่

FR-DEP-002: การดำเนินการกับข้อมูลแผนก

- สร้างแผนกใหม่
- แก้ไขข้อมูลแผนก
- ย้ายแผนกไปสังกัดฝ่ายอื่น
- เปลี่ยนสถานะการใช้งาน

3.2 ความต้องการด้าน API (API Requirements)

3.2.1 รูปแบบ API ทั่วไป

FR-API-001: โครงสร้าง URL

https://[domain]/api/v1/[resource]

- ใช้ HTTPS เท่านั้น
- มีเวอร์ชัน API ใน URL
- ใช้ชื่อ Resource เป็นพหูพจน์

FR-API-002: รูปแบบ Request/Response

- ใช้ JSON เป็นรูปแบบข้อมูล
- ใช้ UTF-8 Encoding
- Content-Type: application/json

FR-API-003: รูปแบบ Response มาตรฐาน

Success Response:

```
{
  "success": true,
  "data": { },
  "message": "ดำเนินการสำเร็จ"
}
```

Error Response:

```
{
  "success": false,
  "error": {
    "code": "ERROR_CODE",
    "message": "คำอธิบายข้อผิดพลาด",
    "field": "ฟิลด์ที่ผิดพลาด (ถ้ามี)"
```

}

}

3.2.2 API Endpoints

บริษัท (Companies)

Method	Endpoint	คำอธิบาย
GET	/api/v1/companies	ดึงรายการบริษัททั้งหมด
GET	/api/v1/companies/{code}	ดึงข้อมูลบริษัทตามรหัส
POST	/api/v1/companies	สร้างบริษัทใหม่
PUT	/api/v1/companies/{code}	แก้ไขข้อมูลบริษัท
PATCH	/api/v1/companies/{code}/status	เปลี่ยนสถานะบริษัท

สาขา (Branches)

Method	Endpoint	คำอธิบาย
GET	/api/v1/branches	ดึงรายการสาขาทั้งหมด
GET	/api/v1/branches/{code}	ดึงข้อมูลสาขาตามรหัส
GET	/api/v1/companies/{code}/branches	ดึงสาขาทั้งหมดของบริษัท
POST	/api/v1/branches	สร้างสาขาใหม่
PUT	/api/v1/branches/{code}	แก้ไขข้อมูลสาขา

ฝ่าย (Divisions)

Method	Endpoint	คำอธิบาย
GET	/api/v1/divisions	ดึงรายการฝ่ายทั้งหมด
GET	/api/v1/divisions/{code}	ดึงข้อมูลฝ่ายตามรหัส
GET	/api/v1/companies/{code}/divisions	ดึงฝ่ายทั้งหมดของบริษัท
GET	/api/v1/branches/{code}/divisions	ดึงฝ่ายทั้งหมดของสาขา
POST	/api/v1/divisions	สร้างฝ่ายใหม่
PUT	/api/v1/divisions/{code}	แก้ไขข้อมูลฝ่าย

แผนก (Departments)

Method	Endpoint	คำอธิบาย
GET	/api/v1/departments	ดึงรายการแผนกทั้งหมด
GET	/api/v1/departments/{code}	ดึงข้อมูลแผนกตามรหัส
GET	/api/v1/divisions/{code}/departments	ดึงแผนกทั้งหมดของฝ่าย
POST	/api/v1/departments	สร้างแผนกใหม่
PUT	/api/v1/departments/{code}	แก้ไขข้อมูลแผนก

อื่น ๆ

Method	Endpoint	คำอธิบาย
GET	/api/v1/organization-tree	ดึงโครงสร้างองค์กรทั้งหมด
GET	/api/v1/organization-tree/{company_code}	ดึงโครงสร้างองค์กรของบริษัทที่ระบุ
GET	/api/v1/search	ค้นหาข้ามทุกระดับ

Flexible Query (ดึงข้อมูลแบบยืดหยุ่น)

Method	Endpoint	คำอธิบาย
GET	/api/v1/flexible/company-departments	ดึงบริษัทพร้อมแผนกทั้งหมด (ข้ามสาขาและฝ่าย)
GET	/api/v1/flexible/company-full	ดึงบริษัทพร้อมสาขา ฝ่าย แผนก ในครั้งเดียว
GET	/api/v1/flexible/custom	ดึงข้อมูลตามที่กำหนด (ใช้ Query Parameters)

3.2.3 Query Parameters

- page - หน้าที่ต้องการ (เริ่มที่ 1)
- limit - จำนวนรายการต่อหน้า (ค่าเริ่มต้น 20, สูงสุด 100)
- sort - ฟิลด์ที่ใช้เรียงลำดับ
- order - ลำดับการเรียง (asc/desc)
- active - กรองเฉพาะที่ใช้งานอยู่ (true/false)
- q - คำค้นหา
- include - ระบุข้อมูลที่ต้องการรวม (branches,divisions,departments)
- company - กรองตามรหัสบริษัท
- branch - กรองตามรหัสสาขา

3.2.4 Flexible Data Retrieval (การดึงข้อมูลแบบยืดหยุ่น)

FR-API-004: ดึงข้อมูลแบบกำหนดเอง

คำอธิบาย: ระบบต้องรองรับการดึงข้อมูลในรูปแบบต่างๆ ตามความต้องการของแต่ละ Application

ตัวอย่างการใช้งาน:

1. **ดึงบริษัทพร้อมแผนกทั้งหมด (ข้ามสาขาและฝ่าย)**

GET /api/v1/flexible/company-departments?company=ABC

Response: {

"company": { ... },

"departments": [...]

}

1. **ดึงข้อมูลทั้งหมดในครั้งเดียว**

GET /api/v1/flexible/company-full?company=ABC

Response: {

"company": { ... },

"branches": [...],

"divisions": [...],

"departments": [...]

}

1. **ดึงข้อมูลแบบกำหนดเอง**

GET /api/v1/flexible/custom?company=ABC&include=divisions,departments&skip=branches

FR-API-005: Response Format สำหรับ Flexible Query

{

"success": true,

"data": {

"company": { },

"branches": [ ], // ถ้า include

"divisions": [ ], // ถ้า include

"departments": [ ] // ถ้า include

},

"meta": {

"included": ["company", "departments"],

"total_departments": 25

}

}

3.3 ความต้องการด้านส่วนติดต่อผู้ใช้ (User Interface Requirements)

3.3.1 หน้าจอหลัก (Main Pages)

FR-UI-001: หน้า Dashboard

- แสดงจำนวนบริษัท สาขา ฝ่าย แผนก
- แสดงกิจกรรมล่าสุด
- แสดงสถิติการใช้งาน API

FR-UI-002: หน้าจัดการบริษัท

- แสดงรายการบริษัทในรูปแบบตาราง
- มีฟังก์ชันค้นหา
- มีปุ่มเพิ่ม แก้ไข เปลี่ยนสถานะ
- แสดง Pagination เมื่อมีข้อมูลมาก

FR-UI-003: หน้าจัดการสาขา

- แสดงรายการสาขาแยกตามบริษัท
- มีตัวกรองตามบริษัท
- แสดงสถานะสำนักงานใหญ่

FR-UI-004: หน้าจัดการฝ่าย

- แสดงรายการฝ่าย
- มีตัวกรองตามบริษัทและสาขา
- รองรับการแสดงฝ่ายที่ไม่มีสาขา

FR-UI-005: หน้าจัดการแผนก

- แสดงรายการแผนกแยกตามฝ่าย
- มีตัวกรองตามฝ่าย

FR-UI-006: หน้าจัดการ API Keys

- แสดงรายการ API Key
- สร้าง API Key ใหม่
- กำหนดสิทธิ์การเข้าถึง
- ดูสถิติการใช้งาน

3.3.2 องค์ประกอบ UI

FR-UI-007: การออกแบบ

- ใช้ Tailwind CSS แบบ Offline (Compiled/Standalone)
- ใช้สีตาม Brand Guidelines ของ Ruxchai Cold Storage
- จัดการสีแบบ Centralized ผ่าน CSS Variables
- Responsive Design รองรับทุกขนาดหน้าจอ
- มี Loading State ขณะประมวลผล
- ไม่ต้องใช้ Internet สำหรับ CSS และ JavaScript Libraries

FR-UI-010: มาตรฐานการใช้สี

สีหลัก (**Primary)**:

- Ruxchai Blue (#0090D3) - ใช้กับ Header, ปุ่มหลัก, Link
- พื้นหลังที่ต้องการเน้น

สีรอง (**Secondary)**:

- Green (#3AAA35) - ใช้กับสถานะ Active, Success Message
- Grey (#555452) - ใช้กับข้อความทั่วไป, Border, Background รอง

การใช้งาน:

- ห้ามใส่สีตรงๆ ใน HTML/View
- ใช้ CSS Class หรือ Variables เท่านั้น
- แก้สีที่ variables.css ที่เดียว

FR-UI-008: การนำทาง

- มี Sidebar Menu สำหรับเมนูหลัก
- มี Breadcrumb แสดงตำแหน่งปัจจุบัน
- มีปุ่ม Back สำหรับย้อนกลับ

FR-UI-009: Form และ Validation

- แสดง Error Message ที่ชัดเจน
- มี Tooltip อธิบายฟิลด์ที่สำคัญ
- Validate ข้อมูลทั้งฝั่ง Client และ Server

4. ความต้องการที่ไม่ใช่ด้านฟังก์ชัน (**Non-functional Requirements)**

4.1 ความต้องการด้านประสิทธิภาพ (Performance Requirements)

NFR-PER-001: Response Time

- API Response Time ต้องไม่เกิน 500ms สำหรับ 95% ของ requests
- หน้าเว็บต้องโหลดเสร็จภายใน 3 วินาที
- Database Query ต้องไม่เกิน 100ms

NFR-PER-002: Throughput

- รองรับ API Request 1,000 requests/minute
- รองรับผู้ใช้งานพร้อมกัน 50 users
- รองรับข้อมูลรวม 100,000 records

NFR-PER-003: Resource Usage

- CPU Usage ไม่เกิน 70% ในภาวะปกติ
- Memory Usage ไม่เกิน 4GB
- Database Size ไม่เกิน 10GB ใน 5 ปี

4.2 ความต้องการด้านความปลอดภัย (Security Requirements)

NFR-SEC-001: Authentication

- ใช้ API Key สำหรับการเข้าถึง API
- API Key ต้องมีความยาวอย่างน้อย 32 ตัวอักษร
- เก็บ API Key แบบ Hash ในฐานข้อมูล

NFR-SEC-002: Authorization

- กำหนดสิทธิ์การเข้าถึงตาม API Key
- แยกสิทธิ์ Read และ Write
- Log ทุกการเข้าถึง

NFR-SEC-003: Data Protection

- ใช้ HTTPS สำหรับการสื่อสารทั้งหมด
- Encrypt ข้อมูลสำคัญในฐานข้อมูล
- ป้องกัน SQL Injection
- ป้องกัน XSS Attack

4.3 ความต้องการด้านความน่าเชื่อถือ (Reliability Requirements)

NFR-REL-001: Availability

- ระบบต้องมี Uptime 99.5% (Downtime ไม่เกิน 3.5 ชั่วโมง/เดือน)
- มี Graceful Degradation เมื่อเกิดปัญหา
- มี Error Handling ที่เหมาะสม

NFR-REL-002: Backup and Recovery

- Backup ข้อมูลทุกวัน
- เก็บ Backup 30 วัน
- สามารถ Restore ได้ภายใน 2 ชั่วโมง
- ทดสอบ Restore ทุกเดือน

4.4 ความต้องการด้านการใช้งาน (Usability Requirements)

NFR-USB-001: ความง่ายในการเรียนรู้

- ผู้ใช้ใหม่สามารถใช้งานได้ภายใน 30 นาที
- มีคู่มือการใช้งานภาษาไทย
- UI เป็นภาษาไทย

NFR-USB-002: ความง่ายในการใช้งาน

- ใช้งานได้ด้วย Mouse หรือ Keyboard
- มี Shortcut Keys สำหรับฟังก์ชันที่ใช้บ่อย
- มี Search และ Filter ที่ใช้งานง่าย

NFR-USB-003: Error Prevention

- มี Confirmation Dialog สำหรับการลบ/เปลี่ยนแปลงสำคัญ
- มี Default Value ที่เหมาะสม
- Validate ข้อมูลก่อน Submit

4.5 ความต้องการด้านการบำรุงรักษา (Maintainability Requirements)

NFR-MNT-001: Code Quality

- ใช้ MVC Pattern อย่างเคร่งครัด
- มี Code Comments อธิบายส่วนสำคัญ
- ตั้งชื่อตัวแปรและฟังก์ชันที่สื่อความหมาย
- แยก Business Logic ออกจาก Presentation
- จัดการ CSS แบบ Centralized ไม่ใส่ style ใน View

NFR-MNT-004: CSS Management

- ใช้ CSS Variables สำหรับค่าที่ใช้ซ้ำ (สี, spacing, font)
- รวม Global Styles ไว้ที่ variables.css
- ห้ามใส่ inline styles ใน HTML
- ใช้ Utility Classes ของ Tailwind เป็นหลัก
- Custom CSS เฉพาะกรณีจำเป็น

NFR-MNT-002: Documentation

- มี Technical Documentation
- มี API Documentation
- มี Database Schema Documentation
- Update Documentation ทุกครั้งที่มีการเปลี่ยนแปลง

NFR-MNT-003: Testing

- มี Unit Test สำหรับ Business Logic
- มี Integration Test สำหรับ API
- Test Coverage อย่างน้อย 70%

4.6 ความต้องการด้านความเข้ากันได้ (Compatibility Requirements)

NFR-COM-001: Browser Support

- Google Chrome (เวอร์ชัน 90+)
- Mozilla Firefox (เวอร์ชัน 88+)
- Microsoft Edge (เวอร์ชัน 90+)
- Safari (เวอร์ชัน 14+)

NFR-COM-002: API Compatibility

- รองรับ JSON Format
- รองรับ UTF-8 Encoding
- HTTP/1.1 และ HTTP/2

5. ความต้องการด้านระบบ (System Requirements)

5.1 สภาพแวดล้อมการพัฒนา (Development Environment)

SR-DEV-001: Software Requirements

- Node.js version 18 LTS หรือสูงกว่า
- npm หรือ yarn สำหรับจัดการ packages

- Visual Studio Code หรือ IDE อื่น
- Git สำหรับ Version Control
- Postman สำหรับทดสอบ API

SR-DEV-002: Development Database

- Microsoft SQL Server 2019 Developer Edition
- SQL Server Management Studio (SSMS)
- Sample Data สำหรับทดสอบ

5.2 สภาพแวดล้อมการใช้งานจริง (Production Environment)

SR-PRD-001: Server Requirements

- Operating System: Windows Server 2019 หรือ Linux
- CPU: 4 Cores ขึ้นไป
- RAM: 8 GB ขึ้นไป
- Storage: 100 GB ขึ้นไป
- Network: 100 Mbps ขึ้นไป

SR-PRD-002: Software Requirements

- Node.js version 18 LTS
- PM2 สำหรับ Process Management
- Nginx สำหรับ Reverse Proxy (Optional)
- Microsoft SQL Server 2019 Standard Edition
- ไม่ต้องใช้ Internet สำหรับ Frontend Libraries

SR-PRD-003: Client Requirements

- Modern Web Browser
- ไม่ต้องใช้ Internet สำหรับ CSS และ JavaScript Libraries
- Screen Resolution 1366x768 ขึ้นไป

5.3 การจัดการ Dependencies

SR-DEP-001: Frontend Dependencies (Offline)

คำอธิบาย: ระบบต้องรวม Library ทั้งหมดไว้ใน Application เพื่อใช้งานแบบ Offline

Libraries ที่ต้องรวม:

- Tailwind CSS (แบบ Compiled หรือ Standalone)
- Alpine.js หรือ Vanilla JavaScript (สำหรับ Interactivity)

- Font Files (ถ้าใช้)
- Icon Files (แทนการใช้ Icon Font จาก CDN)

วิธีการ Bundle:

1. ใช้ Tailwind CLI สร้าง CSS File
2. Download และ Include JavaScript Libraries
3. Bundle ทุกอย่างไว้ใน public/assets folder
4. ใช้ Local Path แทน CDN

SR-DEP-002: โครงสร้าง Assets Folder

```
/public
  /assets
    /css
      - tailwind.min.css (compiled)
      - variables.css (ตัวแปรสีและ theme)
      - custom.css
    /js
      - alpine.min.js (ถ้าใช้)
      - app.js
    /fonts
      - (font files)
    /images
      - (icons, logos)
```

SR-DEP-003: การจัดการสี Theme (Ruxchai Cold Storage)

คำอธิบาย: ระบบต้องใช้สีตาม Brand Guidelines ของ Ruxchai Cold Storage และจัดการแบบ Centralized

Color Palette:

```
/* ไฟล์: /assets/css/variables.css */

:root {
  /* Primary Colors */
  --color-primary: #0090D3;      /* Ruxchai Blue */
  --color-primary-rgb: 0, 144, 211;

  /* Secondary Colors */
  --color-secondary-green: #3AAA35;  /* Green */
```

```
--color-secondary-green-rgb: 58, 170, 53;

--color-secondary-grey: #555452;   /* Grey */
--color-secondary-grey-rgb: 85, 84, 82;

/* Blue Shades */
--color-blue-dark: #047685;    /* 100% */
--color-blue-medium: #00B1EF;  /* 70% */
--color-blue-light: #4DD1FF;   /* 50% */

/* Green Shades */
--color-green-dark: #258C1F;   /* 100% */
--color-green-medium: #24C617; /* 70% */
--color-green-light: #56E04F;  /* 50% */

/* Grey Shades */
--color-grey-dark: #161616;    /* 100% */
--color-grey-medium: #6B6B6B;  /* 70% */
--color-grey-light: #999999;   /* 50% */

/* Semantic Colors */
--color-success: var(--color-secondary-green);
--color-info: var(--color-primary);
--color-warning: #FFC107;
--color-danger: #DC3545;

/* Background Colors */
--bg-primary: var(--color-primary);
--bg-secondary: #F8F9FA;
--bg-dark: var(--color-grey-dark);

/* Text Colors */
--text-primary: var(--color-grey-dark);
--text-secondary: var(--color-grey-medium);
--text-light: var(--color-grey-light);
```

```
  --text-white: #FFFFFF;
}
```

การใช้งานใน Tailwind Config:

```
// tailwind.config.js
module.exports = {
  theme: {
    extend: {
      colors: {
        'ruxchai-blue': '#0090D3',
        'ruxchai-green': '#3AAA35',
        'ruxchai-grey': '#555452',
        // เพิ่มสีอื่นๆ
      }
    }
  }
}
```

หลักการใช้งาน:

1. ห้ามใส่สีตรงๆ ใน View files
2. ใช้ CSS Variables หรือ Tailwind Classes เท่านั้น
3. ถ้าต้องการเปลี่ยนสี แก้ที่ variables.css ที่เดียว
4. ใช้ Semantic Colors สำหรับ UI Elements

5.3 โครงสร้างฐานข้อมูล (Database Schema)

SR-DB-001: ตารางหลัก

```
Companies
├── Branches (Optional)
├── Divisions
|   └── Departments
└── API_Keys
    └── API_Logs
```

SR-DB-002: Indexes และ Constraints

- Primary Key ทุกตาราง
- Foreign Key สำหรับความสัมพันธ์

- Unique Index สำหรับ Code ทุกตาราง
- Index สำหรับฟิลด์ที่ใช้ค้นหาบ่อย

5.4 การติดตั้งและการ Deploy

SR-DPL-001: ขั้นตอนการติดตั้ง

1. ติดตั้ง Node.js และ Dependencies
2. สร้าง Database และ Tables
3. กำหนดค่า Environment Variables
4. Build และ Deploy Application
5. กำหนดค่า Web Server
6. ทดสอบระบบ

SR-DPL-002: Environment Variables

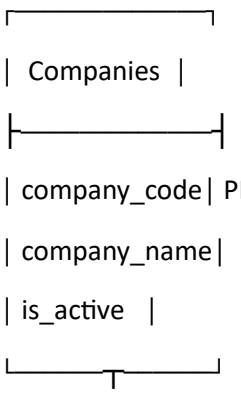
- DATABASE_CONNECTION - Connection String
- API_PORT - Port สำหรับ API
- JWT_SECRET - Secret Key
- NODE_ENV - Environment (development/production)

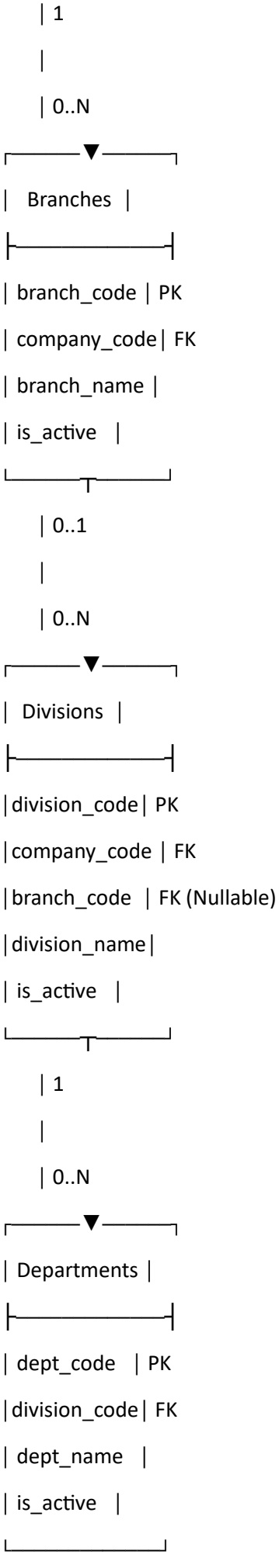
6. ภาคผนวก (Appendices)

6.1 รายการคำศัพท์ (Glossary)

- **Single Source of Truth**: แหล่งข้อมูลเดียวที่เชื่อถือได้
- **RESTful**: รูปแบบการออกแบบ Web Service ที่เป็นมาตรฐาน
- **MVC**: Model-View-Controller การแยกส่วนการทำงานของโปรแกรม
- **API Key**: รหัสสำหรับยืนยันตัวตนในการเข้าถึง API
- **Soft Delete**: การลบแบบเปลี่ยนสถานะ ไม่ลบข้อมูลจริง

6.2 แผนภาพ Entity Relationship





6.3 ตัวอย่าง API Response

สำเร็จ - รายการบริษัท


```
{
  "success": true,
  "data": [
    {
      "company_code": "ABC",
      "company_name_th": "บริษัท เอบีซี จำกัด",
      "company_name_en": "ABC Company Limited",
      "tax_id": "0105558123456",
      "is_active": true,
      "created_date": "2024-01-15T10:30:00Z"
    }
  ],
  "message": "ดึงข้อมูลสำเร็จ",
  "meta": {
    "page": 1,
    "limit": 20,
    "total": 1
  }
}
```


สำเร็จ - ดึงบริษัทพร้อมแผนกทั้งหมด


```
{
  "success": true,
  "data": {
    "company": {
      "company_code": "ABC",
      "company_name_th": "บริษัท เอบีซี จำกัด",
      "company_name_en": "ABC Company Limited"
    },
    "departments": [
      {
        "department_code": "ACC001",
        "department_name": "แผนกบัญชี",
        "division_code": "FIN001",
```

```
      "division_name": "ฝ่ายการเงิน",
      "branch_code": "HQ",
      "branch_name": "สำนักงานใหญ่"
    },
    {
      "department_code": "HR001",
      "department_name": "แผนกทรัพยากรบุคคล",
      "division_code": "ADM001",
      "division_name": "ฝ่ายบริหาร",
      "branch_code": null,
      "branch_name": null
    }
  ]
 },
 "meta": {
  "included": ["company", "departments"],
  "total_departments": 2
 }
}
```


สำเร็จ - ดึงข้อมูลทั้งหมดของบริษัท


```
{
 "success": true,
 "data": {
  "company": {
   "company_code": "ABC",
   "company_name_th": "บริษัท เอบีซี จำกัด"
  },
  "branches": [
   {
    "branch_code": "HQ",
    "branch_name": "สำนักงานใหญ่",
    "is_headquarters": true
   }
  ],
```

```
    "divisions": [
      {
        "division_code": "FIN001",
        "division_name": "ฝ่ายการเงิน",
        "branch_code": "HQ"
      },
      {
        "division_code": "ADM001",
        "division_name": "ฝ่ายบริหาร",
        "branch_code": null
      }
    ],
    "departments": [
      {
        "department_code": "ACC001",
        "department_name": "แผนกบัญชี",
        "division_code": "FIN001"
      }
    ]
  }
}
```


ผิดพลาด - ข้อมูลซ้ำ


```
{
  "success": false,
  "error": {
    "code": "DUPLICATE_CODE",
    "message": "รหัสบริษัท ABC มีอยู่แล้วในระบบ",
    "field": "company_code"
  }
}
```


6.4 การเปลี่ยนแปลงในอนาคต (Future Enhancements)

1. เพิ่มระบบ Workflow สำหรับการอนุมัติการเปลี่ยนแปลง
2. เพิ่มการ Export ข้อมูลเป็น Excel

3. เพิ่มระบบแจ้งเตือนเมื่อมีการเปลี่ยนแปลงข้อมูล
4. รองรับหลายภาษา (Multi-language)
5. เพิ่ม Mobile Application

***** สิ้นสุดเอกสาร *****