Duty Roster & Event Scheduling System (Full-stack Next.js)

Project Plan + Database Design (PostgreSQL)

Date: 29 Dec 2025

Platform: Web Responsive (Browser) • Deploy: Windows / Linux Server

1. Overview

ระบบสำหรับจัดตารางเวร/กิจกรรม (Event) ผ่านปฏิทินแบบ Responsive รองรับการกำหนดประเภทงาน, ช่วงเวลาเวร, วันหยุดองค์กร, วันลารายบุคคล (พร้อมอนุมัติ), ระบบตรวจสอบความซ้ำซ้อน และรายงานสรุปชั่วโมงทำงาน/จำนวนเวร/จำนวนลา รวมถึงกำหนดสิทธิ์การใช้งานและการแสดงเมนูรายผู้ใช้

1.1 Key Goals

- ลดความผิดพลาดจากการจัดเวร (ชนเวลา, ชนวันลา, เกินจำนวนคนต่อ Slot)
- ผู้ใช้ Staff เห็นปฏิทินและรายการของตนเองเท่านั้น (Data Scoping ที่ฝั่ง Server)
- ทำงานในวันหยุดแล้วสะสม Holiday Credit เพื่อใช้ลาภายหลังได้
- รองรับการ Deploy ได้ทั้ง Windows และ Linux (Node.js / Docker)

1.2 Roles & Visibility

การมองเห็น (Visibility) ของปฏิทินถูกควบคุมที่ฝั่ง Server ตามบทบาทและการผูกผู้ใช้กับพนักงาน (users.employee_id):

Role	Visibility / Capability
Admin	เห็นทั้งหมด จัดการข้อมูลหลัก/สิทธิ์/Policy ได้
Supervisor/Planner	เห็นของทีมที่รับผิดชอบ จัดเวร + อนุมัติลา + ดูรายงานทีม
Staff	เห็นเฉพาะรายการที่ตนเองถูก Assign และวันลาของตนเอง

2. Technology Stack (Next.js ทั้งระบบ)

2.1 Frontend

- Next.js (App Router) + React + TypeScript
- Tailwind CSS + shadcn/ui
- Calendar: FullCalendar (Month/Week/Day)
- Forms: React Hook Form + Zod
- Client caching (optional): TanStack Query

2.2 Backend (ภายใน Next.js)

- Route Handlers: app/api/**/route.ts (REST API)
- Server Actions (เหมาะกับฟอร์ม CRUD)
- Validation: Zod (body/query)
- Auth: Auth.js (NextAuth) + RBAC + User Menu Permission
- Transaction & Concurrency: ใช้ DB transaction ในส่วน Assign/Timeblock

2.3 Database (Development)

PostgreSQL (ฐานข้อมูลสำหรับพัฒนาระบบ):

```
POSTGRES_CONNECTION=Host=localhost;Port=5432;Database=PRD_MSCRM_SSO;Username=postgres;Pas
sword=Admin@9999;Pooling=true;
```

หมายเหตุ: สำหรับ Prisma/pg (Node) แนะนำใช้ DATABASE_URL แบบ URI และต้อง encode เครื่องหมาย '@' ในรหัสผ่าน:

```
DATABASE_URL="postgresql://postgres:Admin%409999@localhost:5432/PRD_MSCRM_SSO"
```

2.4 Deployment

- Deploy แบบ Docker Compose (แนะนำ): app + postgres
- Deploy แบบ Node service: pm2/systemd (Linux) หรือ Windows service
- Reverse proxy: Nginx (Linux) / IIS reverse proxy (Windows)

3. Functional Requirements

1. จัดการพนักงาน: เพิ่ม/แก้ไข/ปิดใช้งาน, ชื่อเล่น, บทบาท, ข้อมูลติดต่อ
2. จัดการประเภท Event: โค้ด, ชื่อ, สี, เป็นงานหรือวันหยุด, ระยะเวลาเริ่มต้น
3. จัดการ Shift Slot: เวลาเริ่ม-สิ้นสุด, Min/Max คน, ลำดับการแสดงผล
4. จัดตารางเวร: เพิ่มรายการเวร/กิจกรรมในวันนั้น, Assign พนักงาน 1-N คนต่อรายการ
5. บันทึกวันหยุดประจำปี: วันหยุดนักขัตฤกษ์/วันหยุดองค์กร
6. บันทึกวันหยุด/ลารายบุคคล: ช่วงวันที่, ประเภทลา, เหตุผล, สถานะอนุมัติ
7. ระบบตรวจสอบ: ชนเวลาซ้ำไม่ได้, ชนวันลาไม่ได้, ทำงานวันหยุดสะสม Holiday Credit, จำกัดจำนวน event ต่อคนตาม Policy
8. รายงาน: สรุปชั่วโมงทำงานรายคน, จำนวนเวรตามประเภท, จำนวนวันหยุด/ลา, Holiday Credit
9. กำหนดสิทธิ์การเข้าใช้ระบบ: Role + User Menu Permission (เปิด/ปิดเมนูราย user)

4. API Specification (Next.js Route Handlers)

หมายเหตุ: ทุก endpoint ต้อง enforce สิทธิ์และ Data Scoping ที่ฝั่ง Server (ห้ามส่งข้อมูลทั้งหมดให้ client แล้วค่อยกรอง)

4.1 Auth / Session

- POST /api/auth/login
- POST /api/auth/logout
- POST /api/auth/refresh (ถ้าใช้ refresh token)
- GET /api/auth/me

4.2 Users / Roles / Menus

- GET /api/users
- POST /api/users
- PUT /api/users/{id}
- PATCH /api/users/{id}/disable
- GET /api/roles
- PUT /api/users/{id}/roles
- GET /api/menus
- GET /api/permissions/users/{id}
- PUT /api/permissions/users/{id}

4.3 Employees

- GET /api/employees
- GET /api/employees/{id}
- POST /api/employees
- PUT /api/employees/{id}
- PATCH /api/employees/{id}/deactivate

4.4 Event Types

- GET /api/event-types
- POST /api/event-types
- PUT /api/event-types/{id}
- DELETE /api/event-types/{id}

4.5 Shift Slots

- GET /api/shift-slots
- POST /api/shift-slots
- PUT /api/shift-slots/{id}
- DELETE /api/shift-slots/{id}

4.6 Company Holidays

- GET /api/company-holidays?year=YYYY
- POST /api/company-holidays
- PUT /api/company-holidays/{id}
- DELETE /api/company-holidays/{id}

4.7 Leave Requests

- GET /api/leaves?mine=1|0&teamId=&status=&from=&to=
- GET /api/leaves/{id}
- POST /api/leaves
- PUT /api/leaves/{id}
- POST /api/leaves/{id}/approve
- POST /api/leaves/{id}/reject
- POST /api/leaves/{id}/cancel

4.8 Calendar / Roster

- GET /api/calendar/month?month=YYYY-MM (Staff คืนเฉพาะรายการของตนเอง)
- GET /api/calendar/day?date=YYYY-MM-DD
- POST /api/roster/entries
- PUT /api/roster/entries/{id}
- DELETE /api/roster/entries/{id}
- GET /api/roster/entries/{id} (เช็คสิทธิ์ก่อน)
- POST /api/roster/entries/{id}/assign (employeeIds[])
- DELETE /api/roster/entries/{id}/assign/{employeeId}

4.9 Validation / Availability

- POST /api/validate/roster-entry (ตรวจชนเวลา/ชนลา/min-max/policy)
- GET /api/availability?date=&start=&end=&slotId=&eventTypeId=

4.10 Holiday Credit

- GET /api/holiday-credits/mine
- GET /api/holiday-credits?employeeId=... (Supervisor/Admin)
- POST /api/holiday-credits/use

4.11 Reports

- GET /api/reports/work-hours?from=&to=&employeeId=
- GET /api/reports/shift-count?from=&to=&groupBy=eventType
- GET /api/reports/leave-summary?from=&to=
- GET /api/reports/holiday-credit-summary?from=&to=

4.12 Audit (แนะนำ)

- GET /api/audit?from=&to=&actorId=&entity=

5. Screen List (Web Responsive)

5.1 Auth

- Login
- Profile / My Account

5.2 Staff (เห็นเฉพาะของตนเอง)

- My Calendar (Month/Week/Day)
- My Day Detail (ดูรายการของวันนั้น)
- My Leave (ขอลา/ดูสถานะ/ยกเลิก)
- My Summary (ชั่วโมงทำงาน/จำนวนเวร/เครดิตวันหยุด)

5.3 Supervisor / Planner

- Team Calendar (มี filter ทีม/พนักงาน/ประเภท event)
- Day Planner (Day Editor) สร้างรายการ + Assign + เตือน conflict
- Leave Approvals (อนุมัติ/ปฏิเสธ)
- Team Reports

5.4 Admin / Settings

- Employees Management
- Teams/Org Units (ถ้ามี)
- Event Types Management
- Shift Slots Management
- Company Holidays Management
- Users Management
- Roles & Permissions
- User Menu Permission (เปิด/ปิดเมนูราย user)
- Policies/Settings (กติกา 1 event ต่อคน, วิธีคิดเครดิตวันหยุด, override ได้ไหม)
- Audit Log Viewer

6. Database Design (PostgreSQL)

ฐานข้อมูลออกแบบให้รองรับ: 1-N assignment ต่อรายการ, กติกาชนเวลาแบบ hard rule, วันลา/อนุมัติ, เครดิตวันหยุดแบบ ledger, และสิทธิ์เมนูรายuser

6.1 Schema & Extensions

```
CREATE SCHEMA IF NOT EXISTS duty;
CREATE EXTENSION IF NOT EXISTS pgcrypto;
CREATE EXTENSION IF NOT EXISTS btree_gist;
```

6.2 Table Overview

Table	Purpose
users	ผู้ใช้ระบบ (ผูก employee_id เพื่อ data scoping)
roles, user_roles	บทบาทและความสัมพันธ์ผู้ใช้-บทบาท
menus, role_menus, user_menu_overrides	สิทธิ์เมนูตาม role และ override ราย user
employees, teams	ข้อมูลพนักงาน และทีม/แผนก
event_types	ประเภท event (สี, is_work, is_holiday, default duration)
shift_slots	ช่วงเวลาเวร (start/end, min/max, sort)
company_holidays	วันหยุดองค์กร/นักขัตฤกษ์รายปี
leave_types, leave_requests	ประเภทลา + คำขอลา (approve/reject)
roster_entries	รายการเวร/กิจกรรม (ช่วงเวลา start/end)
roster_assignments	assignment 1-N ต่อรายการเวร
employee_timeblocks	timeblock ต่อพนักงานเพื่อ enforce กันชนเวลา (EXCLUDE constraint)
holiday_credit_ledger	บัญชีเครดิตวันหยุดแบบ ledger (+ได้, -ใช้)
settings	Policy/Setting (เช่น จำกัด 1 event ต่อวัน, วิธีคิดเครดิต)

6.3 Core DDL (CREATE TABLE)

สคริปต์ด้านล่างเป็น DDL หลักสำหรับสร้างตารางทั้งหมด (สามารถนำไปแยกเป็น migration ได้)

```
-- ========================
-- Users / Roles / Menus
-- ========================
CREATE TABLE duty.users (
  user_id        uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  username       text NOT NULL UNIQUE,
  password_hash  text NOT NULL,
  display_name   text NOT NULL,
  employee_id    uuid NULL,
  is_active      boolean NOT NULL DEFAULT true,
  created_at     timestamptz NOT NULL DEFAULT now(),
  updated_at     timestamptz NOT NULL DEFAULT now()
);

CREATE TABLE duty.roles (
```

```
  role_id    uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  role_code  text NOT NULL UNIQUE,
  role_name  text NOT NULL
);

CREATE TABLE duty.user_roles (
  user_id uuid NOT NULL REFERENCES duty.users(user_id) ON DELETE CASCADE,
  role_id uuid NOT NULL REFERENCES duty.roles(role_id) ON DELETE CASCADE,
  PRIMARY KEY (user_id, role_id)
);

CREATE TABLE duty.menus (
  menu_id     uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  menu_code   text NOT NULL UNIQUE,
  menu_name   text NOT NULL,
  path        text NULL,
  sort_order  int  NOT NULL DEFAULT 0,


  is_active   boolean NOT NULL DEFAULT true
);

CREATE TABLE duty.role_menus (
  role_id   uuid NOT NULL REFERENCES duty.roles(role_id) ON DELETE CASCADE,
  menu_id   uuid NOT NULL REFERENCES duty.menus(menu_id) ON DELETE CASCADE,
  can_view  boolean NOT NULL DEFAULT true,
  can_edit  boolean NOT NULL DEFAULT false,
  PRIMARY KEY (role_id, menu_id)
);

CREATE TABLE duty.user_menu_overrides (
  user_id     uuid NOT NULL REFERENCES duty.users(user_id) ON DELETE CASCADE,
  menu_id     uuid NOT NULL REFERENCES duty.menus(menu_id) ON DELETE CASCADE,
  allow_view  boolean NULL,
  allow_edit  boolean NULL,
  PRIMARY KEY (user_id, menu_id)
);

-- ========================
-- Master Data
-- ========================
CREATE TABLE duty.teams (
  team_id    uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  team_code  text UNIQUE,
  team_name  text NOT NULL
);

CREATE TABLE duty.employees (
  employee_id   uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  emp_code      text UNIQUE,
```

```
  first_name    text NOT NULL,


  last_name     text NOT NULL,
  nick_name     text NULL,
  phone         text NULL,
  email         text NULL,
  role_title    text NULL,
  team_id       uuid NULL REFERENCES duty.teams(team_id),
  is_active     boolean NOT NULL DEFAULT true,
  created_at    timestamptz NOT NULL DEFAULT now(),
  updated_at    timestamptz NOT NULL DEFAULT now()
);

ALTER TABLE duty.users
  ADD CONSTRAINT fk_users_employee
  FOREIGN KEY (employee_id) REFERENCES duty.employees(employee_id)
  ON DELETE SET NULL;

CREATE TABLE duty.event_types (
  event_type_id              uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  event_code                 text NOT NULL UNIQUE,
  event_name                 text NOT NULL,
  color_hex                  text NOT NULL DEFAULT '#3b82f6',
  is_work                    boolean NOT NULL DEFAULT true,
  is_holiday                 boolean NOT NULL DEFAULT false,
  default_duration_minutes   int NOT NULL DEFAULT 0,
  is_active                  boolean NOT NULL DEFAULT true,
  sort_order                 int NOT NULL DEFAULT 0
);

CREATE TABLE duty.shift_slots (
  shift_slot_id  uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  slot_code      text UNIQUE,
  slot_name      text NOT NULL,


  start_time     time NOT NULL,
  end_time       time NOT NULL,
  min_staff      int NOT NULL DEFAULT 0,
  max_staff      int NOT NULL DEFAULT 0,
  sort_order     int NOT NULL DEFAULT 0,
  is_active      boolean NOT NULL DEFAULT true,
  CHECK (end_time <> start_time),
  CHECK (min_staff >= 0),
  CHECK (max_staff >= 0),
  CHECK (max_staff = 0 OR max_staff >= min_staff)
);

-- ========================
-- Holidays / Leaves
```

```
-- ========================
CREATE TABLE duty.company_holidays (
  holiday_id    uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  holiday_date  date NOT NULL,
  holiday_name  text NOT NULL,
  holiday_type  text NOT NULL DEFAULT 'ORG',
  is_active     boolean NOT NULL DEFAULT true,
  UNIQUE (holiday_date, holiday_name)
);

CREATE TABLE duty.leave_types (
  leave_type_id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  leave_code    text NOT NULL UNIQUE,
  leave_name    text NOT NULL,
  is_active     boolean NOT NULL DEFAULT true
);

CREATE TABLE duty.leave_requests (


  leave_request_id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  employee_id      uuid NOT NULL REFERENCES duty.employees(employee_id),
  leave_type_id    uuid NOT NULL REFERENCES duty.leave_types(leave_type_id),
  date_from        date NOT NULL,
  date_to          date NOT NULL,
  reason           text NULL,
  status           text NOT NULL DEFAULT 'PENDING',
  requested_at     timestamptz NOT NULL DEFAULT now(),
  decided_at       timestamptz NULL,
  decided_by_user  uuid NULL REFERENCES duty.users(user_id),
  decision_note    text NULL,
  CHECK (date_to >= date_from),
  CHECK (status IN ('PENDING','APPROVED','REJECTED','CANCELED'))
);

CREATE INDEX idx_leave_employee_range ON duty.leave_requests (employee_id, date_from,
date_to);
CREATE INDEX idx_leave_status ON duty.leave_requests (status);

-- ========================
-- Roster / Assignments
-- ========================
CREATE TABLE duty.roster_entries (
  entry_id       uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  entry_date     date NOT NULL,
  event_type_id  uuid NOT NULL REFERENCES duty.event_types(event_type_id),
  shift_slot_id  uuid NULL REFERENCES duty.shift_slots(shift_slot_id),
  start_at       timestamptz NOT NULL,
  end_at         timestamptz NOT NULL,
  note           text NULL,
```

```
  created_by     uuid NULL REFERENCES duty.users(user_id),
  created_at     timestamptz NOT NULL DEFAULT now(),
  updated_at     timestamptz NOT NULL DEFAULT now(),


  CHECK (end_at > start_at)
);

CREATE INDEX idx_roster_entries_date ON duty.roster_entries (entry_date);
CREATE INDEX idx_roster_entries_time ON duty.roster_entries (start_at, end_at);

CREATE TABLE duty.roster_assignments (
  entry_id     uuid NOT NULL REFERENCES duty.roster_entries(entry_id) ON DELETE CASCADE,
  employee_id  uuid NOT NULL REFERENCES duty.employees(employee_id) ON DELETE RESTRICT,
  assigned_at  timestamptz NOT NULL DEFAULT now(),
  assigned_by  uuid NULL REFERENCES duty.users(user_id),
  PRIMARY KEY (entry_id, employee_id)
);

CREATE INDEX idx_assignments_employee ON duty.roster_assignments (employee_id);

-- =========================
-- Timeblocks (DB enforce: no overlap)
-- =========================
CREATE TABLE duty.employee_timeblocks (
  timeblock_id  uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  employee_id   uuid NOT NULL REFERENCES duty.employees(employee_id) ON DELETE CASCADE,
  entry_id      uuid NOT NULL REFERENCES duty.roster_entries(entry_id) ON DELETE CASCADE,
  period        tstzrange NOT NULL,
  entry_date    date NOT NULL,
  created_at    timestamptz NOT NULL DEFAULT now()
);

ALTER TABLE duty.employee_timeblocks
  ADD CONSTRAINT ex_employee_no_overlap
  EXCLUDE USING gist (
    employee_id WITH =,


    period WITH &&
  );

CREATE INDEX idx_timeblocks_employee_date ON duty.employee_timeblocks(employee_id,
entry_date);

-- (Optional) ถ้าตั้ง policy เป็น "1 event ต่อวัน"
-- ALTER TABLE duty.employee_timeblocks
--   ADD CONSTRAINT uq_employee_one_entry_per_day UNIQUE (employee_id, entry_date);

-- =========================
```

```
-- Holiday Credit Ledger
-- ========================
CREATE TABLE duty.holiday_credit_ledger (
  ledger_id        uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  employee_id      uuid NOT NULL REFERENCES duty.employees(employee_id) ON DELETE
CASCADE,
  entry_id         uuid NULL REFERENCES duty.roster_entries(entry_id) ON DELETE SET NULL,
  leave_request_id uuid NULL REFERENCES duty.leave_requests(leave_request_id) ON DELETE
SET NULL,
  minutes_delta    int NOT NULL,
  reason           text NULL,
  created_at       timestamptz NOT NULL DEFAULT now()
);

CREATE INDEX idx_holiday_ledger_employee ON duty.holiday_credit_ledger(employee_id,
created_at);

-- ========================
-- Settings (Policies)
-- ========================
CREATE TABLE duty.settings (
  setting_key   text PRIMARY KEY,
  setting_value text NOT NULL
);
```

6.4 Notes for Implementation (สำคัญ)

- เมื่อมีการ Assign พนักงานให้ roster_entries ต้องสร้าง employee_timeblocks ด้วย (period = [start_at, end_at)) ภายใน transaction เดียวกัน เพื่อให้ DB กันชนเวลาแบบ hard rule
- ก่อนสร้าง timeblock ให้เช็ค leave_requests ที่ APPROVED ว่าชนช่วงวันหรือไม่ (Policy: allow override หรือ block)
- Holiday Credit: เมื่อ entry อยู่ในวันหยุด (company_holidays) หรือ event_type.is_holiday=true และมีการ Assign ให้เพิ่ม holiday_credit_ledger (+minutes) อัตโนมัติ
- Calendar API ต้อง filter ตาม role: Staff คืนเฉพาะ entries ที่มี roster_assignments ของตนเอง