

Car Booking System - Task Summary and Status Update

Updated: 2025-06-22

* Overview

โครงการพัฒนาระบบ "Car Booking System" มีการออกแบบและพัฒนา API สำหรับการจองรถยนต์ การจัดการอุปกรณ์ประกอบ การจอง (booking equipments) และส่วนอื่น ๆ โดยพัฒนาเป็นระบบ Backend ด้วย Node.js + MySQL มีการใช้ Postman ทดสอบ และเชื่อมต่อฐานข้อมูลผ่าน phpMyAdmin

✓ Module หลักและ Task ที่ดำเนินการ

① Vehicle Module

- [x] CRUD Vehicle Types (Model, Controller, Route, SQL Table)
- [x] CRUD Vehicle Brands (Model, Controller, Route, SQL Table)
- [x] CRUD Vehicles (พร้อม upload รูปภาพด้วย multer)

② Booking Module

- [x] Booking CRUD (Model, Controller, Route, SQL Table)
- [x] Booking Filter, Search, Pagination
- [x] Booking Detail Page + Approval Flow (เริ่มต้น 5 ระดับ)
- [x] Booking Report + Export

③ Booking Equipments Module

- [x] Table: booking equipments สร้างแล้ว (เพิ่ม quantity)
- [x] Model: bookingEquipmentModel.js (ครบ CRUD)
- [x] Controller: bookingEquipmentController.js (ดำเนินการแล้วบางส่วน)
- [x] Route: bookingEquipmentRoutes.js (เชื่อม API ครบทั้งหมด)
- [] Postman Test: ยังไม่เริ่มทดสอบ API booking equipments

④ Auth Module

- [x] Auth Register/Login/Logout
- [x] JWT Access Token / Refresh Token
- [x] JWT Blacklist / Revoke
- [x] LDAP Login Integration (พร้อมฟังก์ชัน sync user)

⑤ Setting Module



- [x] CRUD: vehicle_types, vehicle_brands
- [x] CRUD: equipments (อุปกรณ์เสริม)

- [x] CRUD: approval_levels (ลำดับการอนุมัติ)
- [] CRUD: settings system-wide (ยังไม่เริ่ม)


⑥ Dashboard / Log / User

- [x] Dashboard ข้อมูลสรุปการจอง/จำนวนรถ/จำนวนรถอนุมัติ
- [x] CRUD: users + roles + permissions + departments
- [x] CRUD: login_logs (เก็บประวัติการเข้าใช้)
- [] Export report/dashboard (CSV/Excel) ยังไม่ทำ

Task ที่ยังรอดำเนินการ

1. ☒ ทดสอบ API booking equipments:
2. GET /bookings/:id/equipments
3. GET /bookings/:id/equipments/:eid
4. POST /bookings/:id/equipments
5. PUT /bookings/:id/equipments/:eid
6. DELETE /bookings/:id/equipments/:eid
7. ☒ ตรวจสอบการใช้งาน UUID ให้เป็น 64 หลักในทุก Model/Table
8. ☒ จัดทำ Postman Collection แบบ RAW ให้ครบทุก API
9. ☒ เตรียม mock data (vehicles, users, bookings, equipments) ให้ครบ
10. ☒ เอกสาร API/Deployment + Swagger (อยู่ระหว่างเริ่มต้น)
11.  ตั้งค่าระบบทั่วไป เช่น การอนุมัติแบบขึ้นบันได, การเปิด/ปิดฟีเจอร์
12.  สร้าง Report + Export ข้อมูลการจองเป็น PDF/Excel

สถานะรวม

Module	สถานะ	หมายเหตุ
Vehicle Management	<input checked="" type="checkbox"/> เสร็จแล้ว	CRUD ครบ ทดสอบแล้ว
Booking Workflow	<input checked="" type="checkbox"/> เสร็จแล้ว	CRUD + อนุมัติ + รายงานพื้นฐาน
Booking Equipments	 โครงสร้างเสร็จ	รอทดสอบ API
Auth & LDAP	<input checked="" type="checkbox"/> เสร็จแล้ว	JWT/Refresh/LDAP ครบ

Module	สถานะ	หมายเหตุ
User/Role Management	✓ เสร็จแล้ว	แก้ไขได้, มีสิทธิ์แยกตาม module
Dashboard/Log/Stats	✓ เสร็จแล้วบางส่วน	Dashboard ข้อมูลออมนิตี + จอจรวม
Settings + Config	☀ บางส่วน	อุปกรณ์, ยี่ห้อ, ประเภท ทำแล้ว
Export Reports	☀ ยังไม่เริ่ม	PDF/Excel ยังไม่มี

หากต้องการ Export เอกสารนี้เป็น PDF/Word หรือ Markdown สำหรับรายงานหรือส่งต่อ ให้แจ้งได้เลยครับ

1. ภาพรวม Task งานและสถานะ (จากแรกถึงปัจจุบัน)

ลำดับ	Task	สถานะ	หมายเหตุ / ปัญหา & การแก้ไข
1	Project Setup: สร้างโครงสร้าง Node.js + Express + db.js + .env	เสร็จสิ้น	—
2	ออกแบบ Database Schema (users, departments, roles, permissions, bookings, vehicles, etc.)	เสร็จสิ้น	—
3	สร้างตาราง Users + seed, เปลี่ยน UUID format, คอมเมนต์ภาษาไทย	เสร็จสิ้น	—
4	สร้างตาราง departments, roles, permissions, role_permissions, user_roles	เสร็จสิ้น	ปัญหา Duplicate key (roles.name) → แก้ไขด้วย INSERT IGNORE
5	ขยายตาราง bookings: origin/destination, odometer, status enum	เสร็จสิ้น	—
6	Seed ข้อมูลเริ่มต้น (departments, roles, permissions, role_permissions)	เสร็จสิ้น	ปัญหา duplicate → ใช้ INSERT IGNORE
7	ติดตั้ง mysql2, dotenv, uuid และแก้ไขการเชื่อมต่อ DB	เสร็จสิ้น	แก้ error MODULE_NOT_FOUND
8	พัฒนา UserModel + ทดสอบ createUser/ getUserByUsername	เสร็จสิ้น	ปัญหา Data too long for department_id → ใช้ค่า department_id ที่ถูกต้อง
9	สร้าง AuthController: register, login, JWT, bcrypt	เสร็จสิ้น	—
10	สร้าง authRoutes, authMiddleware, ทดสอบ API register/login	เสร็จสิ้น	ปัญหา router handler must be function → แก้ import ข้างใน authRoutes
11	เพิ่ม Refresh Token & Logout Token: ตาราง refresh_tokens, jwt_blacklist, AuthController.refresh/logout	เสร็จสิ้น	ปรับ refresh ให้ revoke เก่า + สร้างใหม่, ปรับ logout ให้ blacklist jti
12	ทดสอบ POST /auth/token/refresh และ POST /auth/logout	เสร็จสิ้น	ทดสอบผ่านใน Postman, blacklist ทำงานถูกต้อง

ลำดับ	Task	สถานะ	หมายเหตุ / ปัญหา & การแก้ไข
13	Integration LDAP Authentication (Active Directory)	เสร็จสิ้น	ปัญหา syntax error, duplicate import → แก้ authRoutes และ authController
14	พัฒนา JWT middleware ตรวจสอบ jti จาก jwt_blacklist	เสร็จสิ้น	ทดสอบ protected route (/vehicles/ping) ผ่านทั้งก่อนและหลัง logout
15	Vehicle Management: CRUD vehicle_types, vehicle_brands, vehicles	รอเริ่ม	กำลังเตรียม SQL DDL และโครงสร้างไฟล์ Model/Controller/Route

2. สรุปงานที่ทำไปแล้ว

1. **Project Setup:** สร้าง boilerplate Express, ตั้งค่า .env, db.js (MySQL Pool)
2. **Database Design:** ออกแบบ Entity-Relationship ครอบคลุมผู้ใช้, ยานพาหนะ, การจอง, อนุมัติ, อุปกรณ์, audit logs
3. **User Module:**
4. Model: createUser, getUserByUsername
5. Controller: register, login (bcrypt & JWT)
6. Route & Middleware: /api/auth/register, /api/auth/login, JWT verify
7. **Refresh & Logout:**
8. สร้างตาราง refresh_tokens + jwt_blacklist
9. Controller.refresh: revoke เก่า, สร้างคู่ใหม่
10. Controller.logout: revoke refresh + insert jti blacklist
11. **JWT Blacklist:** ดัก middleware ปฏิเสธ token ที่ jti อยู่ใน blacklist
12. **LDAP Login:**
13. ติดตั้ง ldapauth-fork, สร้าง config/ldap.js
14. Controller.ldapLogin: authenticate → createLocalUser ถ้าไม่พบ → ออก JWT/Refresh
15. Route: POST /api/auth/login/ldap
16. ทดสอบผ่าน Postman เก็บ tokens เข้า environment
17. **Protected Route:** ทดลอง GET /vehicles/ping ด้วย JWT ก่อน-หลัง logout

3. ปัญหาที่พบ & การแก้ไข

ปัญหา	สาเหตุ	การแก้ไข
Duplicate entry 'admin' for key 'roles.name'	seed ซ้ำค่าเดียวกัน	ใช้ INSERT IGNORE หรือ CHECK EXISTS ก่อน insert
MODULE_NOT_FOUND: mysql2 / dotenv	ลืมติดตั้ง dependencies	ติดตั้ง <code>npm install mysql2 dotenv uuid</code>
Data too long for column 'department_id'	ส่ง UUID ไม่ตรงกับความยาว CHAR(32)	ส่งค่า <code>UUID.replace('-', '')</code> (32 chars)

สรุปผลการดำเนินงานระบบ Car Booking System

1. ภาพรวมโครงการ

ระบบ Car Booking System คือระบบจองยานพาหนะภายในองค์กร ที่รองรับผู้ใช้งานทั่วไป เจ้าหน้าที่ และผู้ดูแลระบบ โดยมีโมดูลหลักดังนี้: - ระบบลงทะเบียนและเข้าสู่ระบบ (Register/Login) พร้อมระบบ Token - การยืนยันตัวตนด้วย LDAP (Active Directory) - ระบบจัดการยานพาหนะ (Vehicle Management) - ระบบจองรถ (Booking Workflow) - การแนบอุปกรณ์เสริมในการจอง (Booking Equipments)

2. Task และสถานะ (เรียงตามลำดับที่ดำเนินการ)

ลำดับ	Task	สถานะ	หมายเหตุ
1	สร้างโครงสร้าง Express + ตั้งค่า .env, db.js, middleware	✓ เสร็จสมบูรณ์	Setup ระบบพื้นฐาน
2	ออกแบบฐานข้อมูล users, roles, bookings, vehicles	✓ เสร็จสมบูรณ์	พร้อม seed ข้อมูลเริ่มต้น
3	พัฒนา AuthController (Register, Login) + JWT	✓ เสร็จสมบูรณ์	ใช้ bcrypt และ jwt
4	เพิ่ม refresh token + logout token + blacklist	✓ เสร็จสมบูรณ์	รองรับ security สูงสุด
5	เชื่อมต่อ LDAP Authentication	✓ เสร็จสมบูรณ์	สร้าง local user อัตโนมัติ
6	พัฒนา Vehicle CRUD + Types + Brands	✓ เสร็จสมบูรณ์	มีระบบแบ่งประเภท/ยี่ห้อ
7	พัฒนา Booking Workflow CRUD	✓ เสร็จสมบูรณ์	พร้อมคำนวณระยะทาง total_distance
8	เพิ่มโมดูล Booking-Equipment Workflow	✓ เสร็จสมบูรณ์	Many-to-Many พร้อม quantity
9	เพิ่ม controller/routes ครอบคลุม action ในอนาคต	✓ เสร็จสมบูรณ์	getPaged, getById, countAll, update, delete

3. ปัญหาที่พบและแนวทางแก้ไข

ปัญหา	สาเหตุ	แนวทางแก้ไข
Duplicate Entry	seed ซ้ำ	ใช้ <code>INSERT IGNORE</code> หรือ check ก่อน insert
MODULE_NOT_FOUND	ลืมติดตั้ง dependencies	ติดตั้ง <code>mysql2</code> , <code>uuid</code> , <code>dotenv</code> เพิ่ม
column length mismatch	UUID ไม่ตรงกับ CHAR(32)	ใช้ <code>.replace('-', '')</code> ก่อน insert
route handler not function	import ฟังก์ชันซ้ำ	รวม destructure ให้ชัดเจน
Token ยังใช้ได้หลัง logout	ไม่เช็ค blacklist	เพิ่ม middleware ตรวจสอบ jti ใน blacklist

4. สรุปโมดูลที่พัฒนาแล้ว

✓ Auth & User

- Register / Login พร้อม JWT, Refresh Token
- รองรับ LDAP Login (สร้างผู้ใช้จาก AD)
- Logout ไล่ token ลง blacklist
- ตรวจสอบสิทธิ์ด้วย middleware `authMiddleware.js`

✓ Vehicle Management

- ตาราง `vehicle_types`, `vehicle_brands`, `vehicles`
- API CRUD ครบทุกตัว
- ระบบ pagination + search ในอนาคต

✓ Booking Workflow

- CRUD: สร้าง แก้ไข ลบ รายการการจองรถ
- รองรับข้อมูลต้นทาง/ปลายทาง, ไมล์, ผู้ขับ, เหตุผล
- คำนวณ `total_distance` อัตโนมัติ

✓ Booking Equipment Workflow

- ตาราง `booking equipments` (Many-to-Many)
- quantity สำหรับแต่ละอุปกรณ์ต่อการจอง
- route ครบทุกตัว (getAll, getById, add, update, remove, paged)
- รองรับแผนการขยายฟೀเจอร์อนาคต

5. แผนงานถัดไป

- Dashboard สรุปจำนวนจอง, รถ, สถานะการอนุมัติ, รายงาน

- ระบบอนุมัติการจอง (multi-level approval)
 - Audit Logs + Notifications
 - การอัปโหลดไฟล์เอกสารแนบ (ใบขออนุญาต, รูป)
 - เตรียม Deploy ด้วย Docker + Document Swagger + CI/CD
-

เอกสารนี้สรุปจากข้อมูลจริงของไฟล์ระบบที่พัฒนาแล้วใน Node.js + Express โดยเชื่อมต่อ MySQL และพร้อมใช้งานจริงในระดับองค์กร

1. ภาพรวมโครงการ (Project Overview)

โครงการ **ระบบจองรถ (CarBooking)** สำหรับองค์กรภาครัฐ (อบจ.) โดยให้เจ้าหน้าที่ระดับ sw.สต. จองรถเพื่อปฏิบัติงานในพื้นที่ต่างๆ พร้อมระบบอนุมัติลำดับขั้นสูงสุด 5 ระดับ, รายงานสถานะ, ปฏิทินการจอง, ระบบสมาชิก และสิทธิ์การเข้าถึงตามบทบาท

วัตถุประสงค์

- จัดการคำขอจองรถและอนุมัติตามลำดับขั้น
- แสดง Dashboard สรุปสถานะการจอง
- รองรับการเลือกอุปกรณ์เสริมและคนขับ
- เก็บประวัติการใช้งานและ Export รายงาน
- รองรับ 2 ภาษา (ไทย-อังกฤษ)

2. เทคโนโลยีหลัก (Tech Stack)

- **Backend:** Node.js + Express.js
- **Frontend:** Nuxt 3 + Nuxt UI Pro
- **Database:** MySQL (UTF8MB4)
- **ORM/DB Layer:** mysql2 + Promise
- **Authentication:** JWT, OAuth, LDAP
- **Upload:** Multer
- **Logging:** Morgan
- **Security:** bcryptjs, .env

3. โครงสร้างโฟลเดอร์โปรเจกต์ (Folder Structure)

```
car-booking-backend/
├── server.js           # entry point
├── .env                # ค่าคอนฟิกความลับ
├── package.json
├── config/
│   └── db.js          # ตั้งค่าการเชื่อมต่อ MySQL
├── routes/             # กำหนดเส้นทาง API
├── controllers/        # จัดการ logic ของแต่ละ route
├── models/             # ฟังก์ชันติดต่อฐานข้อมูล (userModel, authModel, ...)
├── middleware/         # JWT check, error handler, logging
```

```
└─ uploads/          # เก็บไฟล์อัปโหลด (รูปภาพ)
└─ testUserModel.js  # สคริปต์ทดสอบ Model
```

4. โครงสร้างฐานข้อมูล (Database Schema)

4.1 ตารางหลัก (Entities)

ตาราง	PK	ฟิลด์สำคัญอื่น ๆ
users	user_id CHAR32	username, email, password, first_name, last_name, gender, citizen_id, phone, address, country, province, postal_code, avatar_path, department_id, status, created_at, updated_at
departments	department_id	name, created_at
roles	role_id	name, created_at
permissions	permission_id	name, description, created_at
vehicles	vehicle_id	license_plate, type_id, brand_id, capacity, color, description, image_path, is_public, created_at
vehicle_types	type_id	name, created_at
vehicle_brands	brand_id	name, created_at
equipments	equipment_id	name, created_at
vehicle_equipments	vehicle_equipment_id	vehicle_id, equipment_id, created_at
drivers	driver_id	name, phone, created_at
bookings	booking_id	user_id, vehicle_id, driver_id, num_passengers, reason, phone, start_date, start_time, end_date, end_time, origin_location, destination_location, start_odometer, end_odometer, total_distance, status, created_at
booking_equipments	booking_equipment_id	booking_id, equipment_id, created_at
approvals	approval_id	booking_id, approver_id, step, status, comment, approved_at, created_at
audit_logs	log_id	user_id, module, action, ip_address, created_at

4.2 ตารางเชื่อมโยง (Relationships)

- **user_roles** (user_id ↔ role_id)
- **role_permissions** (role_id ↔ permission_id)

- **vehicle equipments** (vehicle_id ↔ equipment_id)
- **booking equipments** (booking_id ↔ equipment_id)

5. SQL Scripts

5.1 สร้างฐานข้อมูลและตาราง users

```
CREATE DATABASE IF NOT EXISTS car_booking
  CHARACTER SET utf8mb4
  COLLATE utf8mb4_unicode_ci;
USE car_booking;

CREATE TABLE IF NOT EXISTS users (
  user_id CHAR(32) PRIMARY KEY COMMENT 'รหัสผู้ใช้งานแบบ UUID v4 ไม่มีขีด',
  username VARCHAR(50) NOT NULL UNIQUE COMMENT 'ชื่อบัญชี',
  email VARCHAR(100) NOT NULL UNIQUE COMMENT 'อีเมล',
  password VARCHAR(255) NOT NULL COMMENT 'hash รหัสผ่าน',
  first_name VARCHAR(100) COMMENT 'ชื่อจริง',
  last_name VARCHAR(100) COMMENT 'นามสกุล',
  gender ENUM('male','female','other') COMMENT 'เพศ',
  citizen_id VARCHAR(13) COMMENT 'เลขบัตร ปชช.',
  phone VARCHAR(20) COMMENT 'เบอร์โทร',
  address TEXT COMMENT 'ที่อยู่',
  country VARCHAR(100) COMMENT 'ประเทศ',
  province VARCHAR(100) COMMENT 'จังหวัด',
  postal_code VARCHAR(10) COMMENT 'รหัสไปรษณีย์',
  avatar_path VARCHAR(255) COMMENT 'พารูป Avatar',
  department_id CHAR(32) COMMENT 'FK → departments',
  status ENUM('active','inactive') DEFAULT 'active' COMMENT 'สถานะ',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP COMMENT 'สร้างเมื่อ',
  updated_at TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT 'แก้ไขล่าสุด'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

5.2 สร้างตารางอื่น ๆ

- departments, roles, permissions, user_roles, role_permissions
- vehicle_types, vehicle_brands, vehicles, equipments, vehicle equipments
- drivers, bookings (+ เพิ่ม origin/destination/odometer/distance/status), booking equipments
- approvals, audit_logs

(รายละเอียด SQL สามารถดูในส่วนก่อนหน้า)

6. Seed Data

```
-- 1) departments
INSERT IGNORE INTO departments (department_id, name) VALUES
  (REPLACE(UUID(), '-', ''), 'ฝ่ายบุคคล'), ...;
-- 2) roles
INSERT IGNORE INTO roles (...) VALUES (...);
-- 3) permissions
INSERT IGNORE INTO permissions (...) VALUES (...);
-- 4) role_permissions
INSERT IGNORE ... SELECT ... FROM roles CROSS JOIN permissions WHERE ...;
```

7. ตัวอย่าง Code Model & Test

7.1 models/userModel.js

```
const db = require('../config/db');
const { v4: uuidv4 } = require('uuid');
async function createUser(data) { ... }
async function getUserByUsername(username) { ... }
async function getUserById(id) { ... }
module.exports = { createUser, getUserByUsername, getUserById };
```

7.2 testUserModel.js

```
const bcrypt = require('bcryptjs');
const { createUser, getUserByUsername } = require('../models/userModel');
(async () => { ... console.log(...) })();
```

8. ขั้นตอนถัดไป (Next Steps)

1. สร้าง Auth Controller (register/login) + Route
2. เขียน Middleware ตรวจสอบ JWT
3. พัฒนา Features อื่น ๆ ตามลำดับ (Booking, Approval, Dashboard)
4. เขียน Unit Test และ Load Test

เอกสารฉบับนี้สรุปภาพรวมและขั้นตอนสำคัญสำหรับการศึกษการพัฒนา *CarBooking System*

ปัญหา	สาเหตุ	การแก้ไข
argument handler must be a function (authRoutes)	import ฟังก์ชันซ้ำสองบรรทัด	ลบ import ซ้ำ, รวม destructure ให้อยู่บรรทัดเดียว
Cannot GET /api/auth/register	ใช้ method GET แทน POST	ส่งเป็น POST /api/auth/register
เส้นทาง Logout ไม่ตอบ message	route ยังไม่เปิดใช้งาน หรือ header ผิด	เปิด router.post('/logout', auth, logout), ตรวจสอบ header
Cannot read properties of undefined (pm.response.code)	ใช้ Pre-request แทน Post-response script	ย้ายโค้ดไปเก็บ Tests (post-response)

4. แผนการทำงานถัดไป

1. **Vehicle Types:**
2. สร้างตาราง `vehicle_types`
3. Model/Controller/Route CRUD
4. ทดสอบใน Postman
5. **Vehicle Brands & Vehicle Equipments:**
6. สร้างตาราง/โมเดล/คอนโทรลเลอร์/รุต
7. ทดสอบรวมกับ Equipment Selection
8. **Vehicles:**
9. สร้างตาราง `vehicles` + สัมพันธ์กับ `vehicle_types` / `vehicle_brands`
10. CRUD, อัปเดต (multer)
11. **Booking Workflow:**
12. พัฒนา booking routes/controllers
13. เชื่อมโยงกับ vehicles, equipments, approvals
14. **UI Dashboard & Calendar + Audit Logs**
15. **เอกสาร API (Swagger) และ Deployment (Docker Compose, CI/CD)**

เอกสารนี้สรุปความคืบหน้าจนถึงขั้นตอน *Authentication + LDAP* เสร็จสมบูรณ์ และพร้อมเข้าสู่ *Vehicle Management* ต่อไป

1. ภาพรวมโครงการ (Project Overview)

CarBooking เป็นระบบบริหารการจองรถยนต์สำหรับองค์กรภาครัฐ (อบจ.) เพื่อให้เจ้าหน้าที่ sw.สต. และบุคลากรภายในองค์กรสามารถ:

1. **สร้างคำขออนุญาต** โดยระบุวันที่ เวลา ตำแหน่งต้นทาง-ปลายทาง และอุปกรณ์เสริมที่ต้องการ
2. **ตรวจสอบสถานะคำขอ** (รออนุมัติ, อนุมัติ, ปฏิเสธ, ยกเลิก)
3. **อนุมัติคำขอ** ตามกระบวนการลำดับขั้น (สูงสุด 5 ระดับ)
4. **จัดการข้อมูลยานพาหนะ** (เพิ่ม แก้ไข ลบ), ประเภทยานพาหนะ, ยี่ห้อ, สี, รายละเอียด, รูปภาพ
5. **ดู Dashboard** สรุปสถิติการจอง และปฏิทินการจองรายเดือน
6. **เก็บประวัติการใช้งาน** (Audit Logs) และส่งออกเป็นรายงาน PDF/Excel
7. **ควบคุมสิทธิ์การเข้าถึง** ตามบทบาท (Roles & Permissions)
8. **รองรับสองภาษา** (ไทย/อังกฤษ)

1.1 วัตถุประสงค์หลัก

- เพิ่มประสิทธิภาพการจัดสรรและใช้งานยานพาหนะภายในองค์กร
- ลดปัญหาการซ้ำซ้อนในการจองและการจัดสรรทรัพยากร
- ยกระดับความโปร่งใสด้วยระบบอนุมัติและบันทึกประวัติการใช้งาน

2. สถาปัตยกรรมระบบ (System Architecture)

1. **Client (Frontend):** Nuxt 3 + Nuxt UI Pro
2. **Server (Backend):** Node.js + Express.js
3. **Database:** MySQL (utf8mb4)
4. **Authentication:** JWT + OAuth + LDAP
5. **File Upload:** Multer
6. **Logging & Monitoring:** Morgan, Winston (optional)
7. **Environment Configuration:** dotenv (.env)
8. **Deployment:** Docker + Docker Compose, CI/CD Pipeline

3. โครงสร้างโปรเจกต์ (Folder Structure)

```
car-booking-backend/  
├── server.js           # Entry point ของแอป  
└── .env               # ตัวแปรสภาพแวดล้อม (DB, JWT_SECRET, PORT)
```

```

├─ package.json          # Dependencies & Scripts
├─ config/
│   └─ db.js             # ตั้งค่าการเชื่อมต่อ DB (mysql2)
├─ models/              # ฟังก์ชันติดต่อกับ DB (ORM-lite)
│   ├── userModel.js
│   ├── authModel.js
│   └─ bookingModel.js
├─ controllers/         # จัดการ Logic ของแต่ละ Route
│   ├── authController.js
│   ├── userController.js
│   ├── bookingController.js
│   └─ vehicleController.js
├─ routes/              # กำหนด API Endpoints
│   ├── authRoutes.js
│   ├── userRoutes.js
│   ├── bookingRoutes.js
│   └─ vehicleRoutes.js
├─ middleware/          # Middleware (JWT, Error Handler)
│   ├── authMiddleware.js
│   └─ errorHandler.js
├─ services/            # Business Logic แยกเป็นบริการ (optional)
├─ uploads/             # เก็บไฟล์อัปโหลด (รูป, Avatar)
├─ utils/               # ฟังก์ชันช่วยเหลือทั่วไป
└─ tests/               # Unit & Integration Tests
    ├── models/
    ├── controllers/
    └─ routes/

```

4. โครงสร้างฐานข้อมูล (Database Schema)

4.1 ตารางหลัก (Entities)

ตาราง	PK (Type)	ฟิลด์สำคัญ
users	user_id CHAR(32)	username, email, password, first_name, last_name, gender, citizen_id, phone, address, country, province, postal_code, avatar_path, department_id, status, created_at, updated_at
departments	department_id CHAR(32)	name, created_at
roles	role_id CHAR(32)	name, created_at
permissions	permission_id CHAR(32)	name, description, created_at

ตาราง	PK (Type)	ฟิลด์สำคัญ
user_roles	user_role_id CHAR(32)	user_id (FK), role_id (FK), created_at
role_permissions	role_permission_id CHAR(32)	role_id (FK), permission_id (FK), created_at
vehicle_types	type_id CHAR(32)	name, created_at
vehicle_brands	brand_id CHAR(32)	name, created_at
vehicles	vehicle_id CHAR(32)	license_plate, type_id (FK), brand_id (FK), capacity, color, description, image_path, is_public, created_at
equipments	equipment_id CHAR(32)	name, created_at
vehicle_equipments	vehicle_equipment_id CHAR(32)	vehicle_id (FK), equipment_id (FK), created_at
drivers	driver_id CHAR(32)	name, phone, created_at
bookings	booking_id CHAR(32)	user_id (FK), vehicle_id (FK), driver_id (FK null), num_passengers, reason, phone, start_date, start_time, end_date, end_time, origin_location, destination_location, start_odometer, end_odometer, total_distance, status, created_at
booking_equipments	booking_equipment_id CHAR(32)	booking_id (FK), equipment_id (FK), created_at
approvals	approval_id CHAR(32)	booking_id (FK), approver_id (FK), step, status, comment, approved_at, created_at
audit_logs	log_id CHAR(32)	user_id (FK), module, action, ip_address, created_at

4.2 ความสัมพันธ์ (Relationships)

```

users—<user_roles>—roles—<role_permissions>—permissions
|
└─<bookings>—vehicles—<vehicle_equipments>—equipments
    |
    └─<booking_equipments>—equipments
        |
        └─<approvals>
            └─drivers
departments—users

```

- **1:N**: users-bookings, departments-users, roles-role_permissions

- **N:M**: users-roles (user_roles), roles-permissions (role_permissions), vehicles-equipments (vehicle_equipments), bookings-equipments (booking_equipments)

5. SQL Scripts (ตัวอย่างโค้ด SQL แบบละเอียด)

5.1 สร้างฐานข้อมูลและตาราง users

```
CREATE DATABASE IF NOT EXISTS car_booking
  CHARACTER SET utf8mb4
  COLLATE utf8mb4_unicode_ci;
USE car_booking;

CREATE TABLE IF NOT EXISTS users (
  user_id CHAR(32) PRIMARY KEY COMMENT 'UUID v4 ไม่มีซ้ำ',
  username VARCHAR(50) NOT NULL UNIQUE COMMENT 'บัญชีล็อกอิน',
  email VARCHAR(100) NOT NULL UNIQUE COMMENT 'อีเมล',
  password VARCHAR(255) NOT NULL COMMENT 'hash ด้วย bcryptjs',
  first_name VARCHAR(100) COMMENT 'ชื่อจริง',
  last_name VARCHAR(100) COMMENT 'นามสกุล',
  gender ENUM('male','female','other') COMMENT 'เพศ',
  citizen_id VARCHAR(13) COMMENT 'เลขบัตร ปชช.',
  phone VARCHAR(20) COMMENT 'เบอร์โทร',
  address TEXT COMMENT 'ที่อยู่',
  country VARCHAR(100) COMMENT 'ประเทศ',
  province VARCHAR(100) COMMENT 'จังหวัด',
  postal_code VARCHAR(10) COMMENT 'รหัสไปรษณีย์',
  avatar_path VARCHAR(255) COMMENT 'รูปโปรไฟล์',
  department_id CHAR(32) COMMENT 'FK->departments.department_id',
  status ENUM('active','inactive') DEFAULT 'active' COMMENT 'สถานะสมาชิก',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP COMMENT 'สร้างเมื่อ',
  updated_at TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT 'แก้ไขล่าสุด'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

5.2 ตาราง departments

```
CREATE TABLE IF NOT EXISTS departments (
  department_id CHAR(32) PRIMARY KEY COMMENT 'UUID v4 ไม่มีซ้ำ',
  name VARCHAR(100) NOT NULL COMMENT 'ชื่อแผนก',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP COMMENT 'สร้างเมื่อ'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

(ต่อในลักษณะเดียวกันกับ roles, permissions, user_roles ... audit_logs)

6. Seed Data (ตัวอย่างการเติมข้อมูลเริ่มต้น)

```
INSERT IGNORE INTO departments (department_id,name) VALUES
  (REPLACE(UUID(),'-',''),'ฝ่ายบุคคล'),
  (REPLACE(UUID(),'-',''),'ฝ่ายวิชาการ'),
  (REPLACE(UUID(),'-',''),'สารสนเทศ'),
  (REPLACE(UUID(),'-',''),'พัสดุ'),
  (REPLACE(UUID(),'-',''),'IT Support');

INSERT IGNORE INTO roles (role_id,name) VALUES
  (REPLACE(UUID(),'-',''),'admin'),
  (REPLACE(UUID(),'-',''),'manager'),
  (REPLACE(UUID(),'-',''),'staff');

INSERT IGNORE INTO permissions (permission_id,name,description) VALUES
  (REPLACE(UUID(),'-',''),'create_booking','สร้างการจองรถ'),
  (REPLACE(UUID(),'-',''),'view_booking','ดูรายการจอง'),
  ...;

-- ผูกบทบาทกับสิทธิ์
INSERT IGNORE INTO role_permissions (role_permission_id,role_id,permission_id)
  SELECT REPLACE(UUID(),'-',''),r.role_id,p.permission_id
  FROM roles r CROSS JOIN permissions p WHERE r.name='admin';
```

7. ตัวอย่าง Code (Model, Controller, Middleware)

7.1 models/userModel.js

```
const db = require('../config/db');
const { v4: uuidv4 } = require('uuid');
async function createUser(data){ /* insert */ }
async function getUserByUsername(u) { /* select */ }
module.exports = {createUser,getUserByUsername};
```

7.2 controllers/authController.js

```
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const { createUser, getUserByUsername } = require('../models/userModel');
exports.register = async (req,res) => { /* hash → save → return token */ };
exports.login = async (req,res) => { /* verify → jwt.sign → return token */ };
```

7.3 middleware/authMiddleware.js

```
const jwt = require('jsonwebtoken');
module.exports = (req, res, next) =>
{ /* ตรวจสอบ token จาก header → jwt.verify → attach req.user */ };
```

8. ขั้นตอนถัดไป (Next Steps)

1. สร้าง Auth Routes (routes/authRoutes.js)
2. พัฒนา Booking Controller & Routes
3. สร้าง Dashboard API & Frontend
4. เขียน Unit Tests (Jest) และ Load Tests (Artillery)
5. จัดทำเอกสาร API (Swagger/OpenAPI)

เอกสารฉบับสมบูรณ์ (Detailed) สำหรับการศึกษาและอ้างอิงในการพัฒนา *CarBooking System*

สรุปการพัฒนาระบบจองรถยนต์ (Car Booking System)

บทบทโครงการพัฒนา

- เป็น Backend: Node.js (Express.js)
 - ภาษเฟ้นข้อมูล: MySQL
 - เป้าหมาย: JWT Token Authentication
 - เครื่องโบนแบบ MVC
 - ตรวจโฟลเดอร์:
 - controllers/
 - models/
 - routes/
 - middleware/
 - config/
-

ตารางข้อมูล MySQL ที่สร้าง

- users
 - roles
 - permissions
 - role_permissions
 - user_roles
 - bookings
 - booking equipments
 - vehicles
 - vehicle_types
 - vehicle_brands
 - approvals
 - approval_flows
 - approval_steps
 - booking_approval_logs
 - booking_approval_status
 - refresh_tokens
-

เพิ่มเติม API ส้นหลัก

ส่วน Auth:

- POST /api/login
- POST /api/register
- POST /api/logout
- POST /api/refresh-token

Vehicles

- CRUD `/api/vehicles`, `/api/vehicles/types`, `/api/vehicles/brands`

Booking

- GET `/api/bookings`, `/api/bookings/:id`
- POST `/api/bookings`
- PUT `/api/bookings/:id`
- DELETE `/api/bookings/:id`
- POST `/api/bookings/:id/approve`
- POST `/api/bookings/:id/reject`
- GET `/api/bookings/:id/logs`

Approval Flow

- POST `/api/approval-flows`
- POST `/api/approval-flows/:id/steps`
- PUT `/api/approval-flows/:id/deactivate`
- GET `/api/approval-flows/:id/steps`

Role & Permission

- GET `/api/roles`, POST `/api/roles`
- GET `/api/permissions`, POST `/api/permissions`
- POST `/api/role-permissions/assign`
- POST `/api/role-permissions/remove`

การกำหนดและ Middleware

- `authMiddleware.js` เพื่อตรวจสอบ JWT ใน Header
- `checkPermission.js` เพื่อตรวจสอบ permission ตาม role ของ user

เพิ่ม Test Case สำคัญ (Postman)

Booking Approval Flow

1. Login และ Get Token
2. Create Booking
3. Create Approval Flow
4. Add Step to Flow
5. Assign Flow to Booking
6. Approve Step 1 → Step 2 → Step 3
7. Reject Booking
8. Check Logs

Role & Permission Test

- Create Role
 - Create Permission
 - Assign Permission to Role
 - Remove Permission from Role
-

ปัญหาที่พบบพบ

- SyntaxError: db already declared ✓ แก้ไข export/require
 - API 404: เป็น path ไม่ตรง server.js
 - Postman: Invalid Authorization Header ใช้ Token ไม่ตรง Bearer
 - Column name cannot be null ✓ ตรวจสอบ req.body.name
 - Argument handler must be a function ✓ แก้ไข router.post('/path', controller.function)
-

บันทึก

- ตรวจสอบ Routing API ใต้ทุก Module
 - เพิ่มเตมใส่ POSTMAN และ Collection ตรวจสอบ Testing
 - โครงสร้างข้เบียบ approval แบบ dynamic
 - เสร็จโครงสร้าง Role & Permission เต็มเก็บ Booking Workflow
-

หากต้องระเบิด บอกผู้มให้ช่วย update code ขย้างแลว ได้ทันที