**บทที่ 3**

**ขั้นตอนการดำเนินงาน**

ปริญญานิพนธ์ฉบับบนี้ได้ทำการพัฒนาซอฟต์แวร์ open-source ที่มีอยู่ในปัจจุบันนำมาประยุกต์ใช้ในการสร้างระบบในการวิเคราะห์ข้อมูลเพื่อสนับสนุนข้อมูลในการตัดสินใจซึ่งจะมีตั้งแต่ส่วนของการรับค่าจากอุปกรณ์ตามมาตรฐานอุตสาหกรรมแล้วนำข้อมูลที่ได้ไปเก็บยังฐานข้อมูลและนำข้อมูลที่ได้มาใช้งานต่าง ๆ ในการวิเคราะห์ระบบที่ทำการศึกษา

ในบทนี้จะกล่าวถึงขั้นตอนการดำเนินงานสร้างระบบวิเคราะห์ข้อมูลเพื่อสนับสนุนข้อมูลในการตัดสินใจจะประกอบด้วย 7 ขั้นตอนหลัก ๆ คือ

ขั้นตอนการศึกษารูปแบบของซอฟต์แวร์และเลือกสิ่งที่จะนำมาพัฒนา

ขั้นการพัฒนาซอฟต์แวร์ Open-Source ในการดึงข้อมูลที่ต้องการด้วยภาษา Visual C#

ขั้นตอนการพัฒนาตัวกลางในการเชื่อมต่อระหว่างส่วนต่าง ๆ เข้ากับฐานข้อมูล

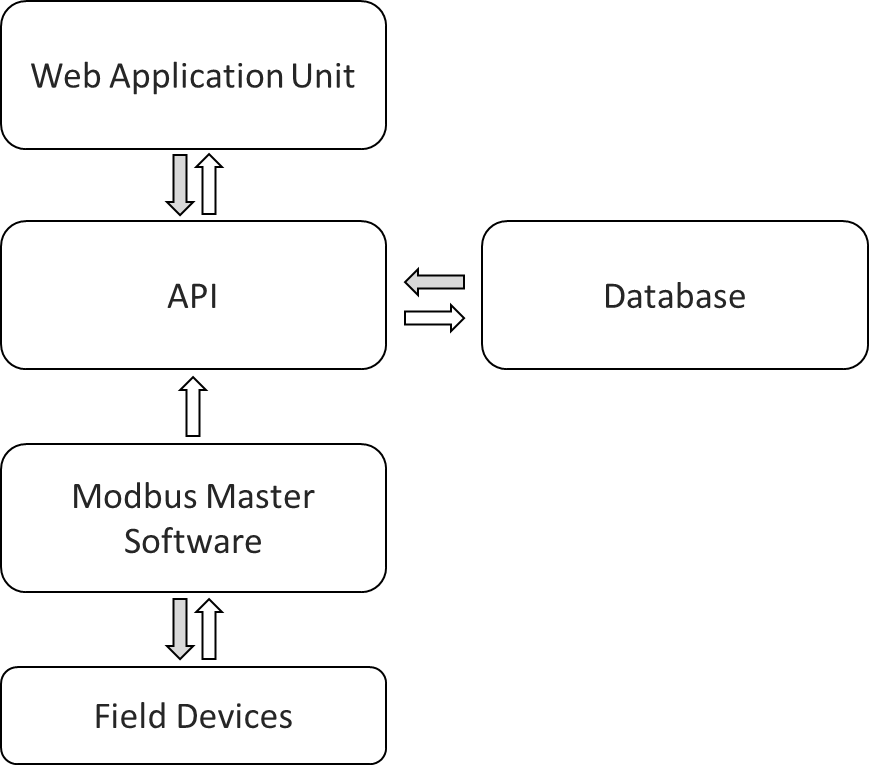
ขั้นตอนการพัฒนา Web Application สำหรับการแสดงผลข้อมูล

ขั้นตอนการเพิ่มความสามารถในการเข้ารหัสและการยืนยันตัวตน

3.1 )ขั้นตอนการศึกษารูปแบบของซอฟต์แวร์และเลือกสิ่งที่จะนำมาพัฒนา

เนื่องด้วยปริญญานิพนธ์ฉบับนี้ได้ตั้งเป้าหมายไว้ที่การรับค่าจากอุปกรณ์ในอุตสาหกรรมดังนั้น จากอุปกรณ์ power meter ที่ได้รับมาทำการทดสอบมีอยู่ 2 รุ่นด้วยกัน คือ ABB CMS-600 และ Schneider PM 3250 โดยทั้ง 2 รุ่นมีพอร์ตการสื่อสารแบบ RS-485 และรองรับโปรโตคอล Modbus RTU ทั้งคู่และมีความสามารถในการใช้งานที่แตกต่างกันคือ ABB CMS-600 จะเน้นที่การวัดกระแสเพียงอย่างเดียวและมีความสามารถในการวัดเฉพาะกระแสไฟฟ้าได้หลากหลายโดย ABB CMS-600 จะมีตัว current transformer เฉพาะรุ่นที่จะใช้ในการวัดเรียกว่า cms sensor และส่งข้อมูลไปให้ control unit ผ่าน cms bus ซึ่งสามารถรองรับ cms sensor ได้ถึง 32 ตัว ส่วน Schneider PM 3250 จะเป็น power meter ที่เน้นในเรื่องของการวัดแรงดัน,กระแส,และกำลังไฟฟ้าทั้ง 3 เฟส โดยต้องนำ current transformer และต้องต่อสายสำหรับแต่ละวัดแรงดันแต่ละเฟสเข้ากับตัว power meter ด้วยเช่นกันซึ่งโครงงานนี้เราต้องการที่จะศึกษาการใช้พลังงานของโหลดแต่ล่ะชนิดจึงได้เลือกใช้ตัว ABB CMS-600 ในโครงงานนี้ได้เลือกใช้ภาษา visual C# มาใช้ในการพัฒนาซอฟต์แวร์ในการดึงค่าจาก power meter เนื่องจาก visual C# เป็น 1 ในภาษาที่ใช้ในการพัฒนาบนแพลตฟอร์ม .NET Framework ซึ่งมี library รองรับหลากหลายแต่สามารถสร้างเป็นไบนารี่ไฟล์ที่มีนามสกุล.exe ได้สำหรับใช้บนปฏิบัติการวินโดวและมี open source library ที่ชื่อว่า EasyModBusTCP ที่รอบรับแพลตฟอร์ม .NET ทำให้สามารถส่งข้อมูลผ่านโปรโตคอล Modbus ได้โดยง่าย

เมื่อเราดึงข้อมูลมาได้สิ่งที่ต้องคำนึงถึงก็คือการเก็บข้อมูลซึ่งฐานที่เลือกใช้คือ MongoDB ซึ่งเป็นฐานข้อมูลแบบ NoSQL เนื่องจาก NoSQL database ถูกออกแบบมาให้มีความคล่องตัวและสามารถปรับเปลี่ยนขนาดเพื่อรองรับจำนวนผู้ใช้จำนวนมากสิ่งต่อมาคือการเชื่อมต่อข้อมูลเข้ากับส่วนต่าง ๆ เราเรียกว่า API โดย open source software ที่เรานำมาใช้ในการสร้าง API คือ NodeJS ใช้ร่วมกับ Express ซึ่งข้อดีของ NodeJS คือการรับมือกับการเชื่อมต่อพร้อม ๆ กันคราวละมาก ๆ โดยที่ยังสามารถส่งข้อมูลกลับให้ผู้ใช้หลาย ๆ คนพร้อมกันได้อย่างรวดเร็วทำให้เหมาะสมกับการทำงานหากมีอุปกรณ์หลาย ๆ ตัวพร้อม ๆ กันส่วนถัดไปจะเป็นส่วนที่ใช้ในการแสดงผลซึ่งสิ่งที่ต้องการสร้างคือWeb Applicationโดยการสร้างเว็บไซต์พื้นฐานก็จะมาจากภาษา HTML CSS และ Javascript อยู่แล้วซึ่ง Framework ที่เลือกนำมาพัฒนาคือ Angular framework โดย Angular จะรับผิดชอบในสิ่งที่ผู้ใช้จะเห็นทั้งหมดรวมทั้งข้อมูลที่เป็น Dynamic หน้าที่ของมันไม่ใช่แค่การใช้ Javascript สร้างหน้า HTML และ CSS แบบตายตัวแต่หน้าที่ของมันก็คือการอัพเดตหน้า UI เมื่อไหร่ก็ตามที่มีข้อมูลใหม่ๆเข้ามา



**ภาพที่ 3-1**

3.2) ขั้นการพัฒนาซอฟต์แวร์ Open-Source ในการดึงข้อมูลที่ต้องการด้วยภาษา Visual C#

จากโครงสร้างของระบบดังภาพที่ 3-1 สิ่งแรกที่ทำการพัฒนาขึ้นมาคือ Modbus Master Software โดยองค์ประกอบของโครงสร้างในการรับส่งข้อมูลระหว่างอุปกรณ์กับฐานข้อมูลมีลักษณะดังภาพที่ 3-2

**การเปิดพอร์ต**

Visual C# นั้นจะติดต่อกับพอร์ตที่ใช้โปรโตคอล Modbus โดยใช้คลาส ModbusClient โดยคลาสนี้จะอยู่ในเนมสเปซ EasyModbus ซึ่งพร็อปเพอร์ตี้ Connect() แต่ก่อนที่เราจะทำการเปิดพอร์ตเราจะต้องเซตค่าบางอย่างให้แก่พอร์ตซึ่งค่าพร็อปเพอร์ตี้สำคัญที่เราต้องมีกำหนดได้แก่

Com Port

Baud Rate

Parity Bit

Stop Bit

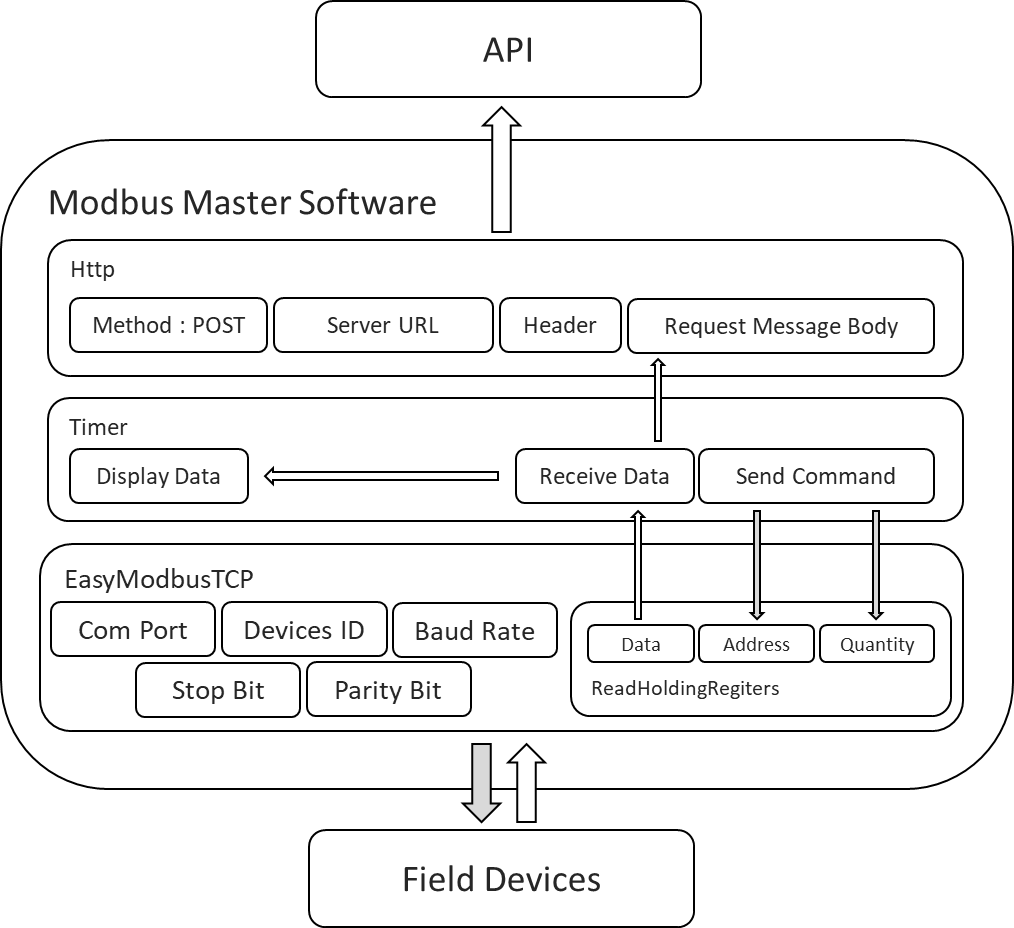
Devices ID

**การอ่านข้อมูลจากพอร์ต**

การอ่านค่า TrueRMS จากพอร์ตนั้นจะใช้เมธอด ReadHoldingRegisters( start address , quantity ) จะเป็นการอ่านข้อมูลโดยเริ่มจากตำแหน่งและจำนวณที่กำหนดและนำผลลัพธ์ไปเก็บไว้ในอาร์เรย์ที่ระบุโดยจะใช้ Timer ในการสั่งงานให้อ่านข้อมูลจากพอร์ตและนำค่าที่ได้ไปแสดงผลและส่งข้อมูลไปยัง Middleware ผ่าน Http Protocol

**การส่งข้อมูลผ่าน Http Protocol**

การส่งข้อมูลผ่าน Http Protocol นั้นจะต้องอาศัยคลาส HttpClient() ซึ่งอยู่ในเนมสเปซ System.Net.Http ซึ่งเป็นคลาสพื้นฐานในการส่ง Http request และรับ Http Response จาก URL ที่กำหนดไว้โดยมี HttpClient.PostAsync Method ในการส่ง POST Request ไปยัง URL ที่กำหนดแบบ Asynchronous และข้อมูลที่ส่งจะถูกกำหนดรูปแบบด้วยคลาส HttpContent ซึ่งจะรวมทั้งข้อมูลที่เป็นส่วนของ Header และ Body โดยข้อมูลที่เราได้จากการอ่านค่า TrueRMS จะถูกแนบไปกับส่วนของ message body



**ภาพที่ 3-2**

3.3) ขั้นตอนการพัฒนาตัวกลางในการเชื่อมต่อระหว่างส่วนต่าง ๆ เข้ากับฐานข้อมูล

โครงสร้างของตัวกลางในการเชื่อมต่อส่วนต่าง ๆ เข้าด้วยกันจะมีลักษณะดัง ภาพที่ 3-3 โดยจะใช้ package Node คือ Express และ Mongoose โดย package Mongoose จะทำหน้าที่ในการเชื่อมต่อและแปลงข้อมูลจาก MongoDB ให้อยู่ในรูปของ JavaScript Objects

**การเชื่อมต่อกับ MongoDB**

Mongoose สามารถเชื่อมต่อกับ MongoDB ได้โดยใช้พร็อปเพอร์ตี้ connect( URI ) โดยใช้ URI เป็นตัวกำหนดการเชื่อมต่อซึ่ง URI ของ MongoDB มีรูปแบบคือ mongodb://username:password@databaseURL

username - username ที่ใช้ในการยืนยันตัวตนกับ MongoDB

password - password ที่ใช้ในการยืนยันตัวตนกับ MongoDB

DatabaseURL – จะประกอบด้วย host, port ,และชื่อ database ที่จะเชื่อมต่อ

**การสร้างโครงสร้างข้อมูล**

โดยปกติแล้ว MongoDB จะไม่มีโครงสร้างของข้อมูลที่ตายตัวแต่ Mongoose จะมีโครงสร้างของข้อมูลแต่ไม่ได้หมายความว่าทุก ๆ ชุดข้อมูลจะมีโครงสร้างเหมือนกัน ที่มีโครงสร้างเพื่อให้ได้ข้อมูลที่ไปในทิศทางเดียวกันโดยเราสามารถสร้างโครงสร้างของข้อมูลได้ด้วยพร็อปเพอร์ตี้ Schema ( object ) ซึ่ง object คือ ข้อมูลที่นำมาสร้างโครงสร้าง และนำโครงสร้างของข้อมูลไปสร้าง model ด้วยพร็อปเพอร์ตี้ model ( name, Schema ) ซึ่ง name คือชื่อของ model ที่นำ Schema มาสร้าง

**Mongoose CRUD**

CRUD คือการจัดการข้อมูลต่าง ๆ ใน Mongoose จะนำ model ที่ได้จาก Schema มาใช้ในงานต่าง ๆ

การสร้าง Document จะเป็นการนำข้อมูลไปเก็บไว้ยังฐานข้อมูลโดยจะใช้ฟังก์ชัน save() ในพร็อปเพอร์ตี้ model

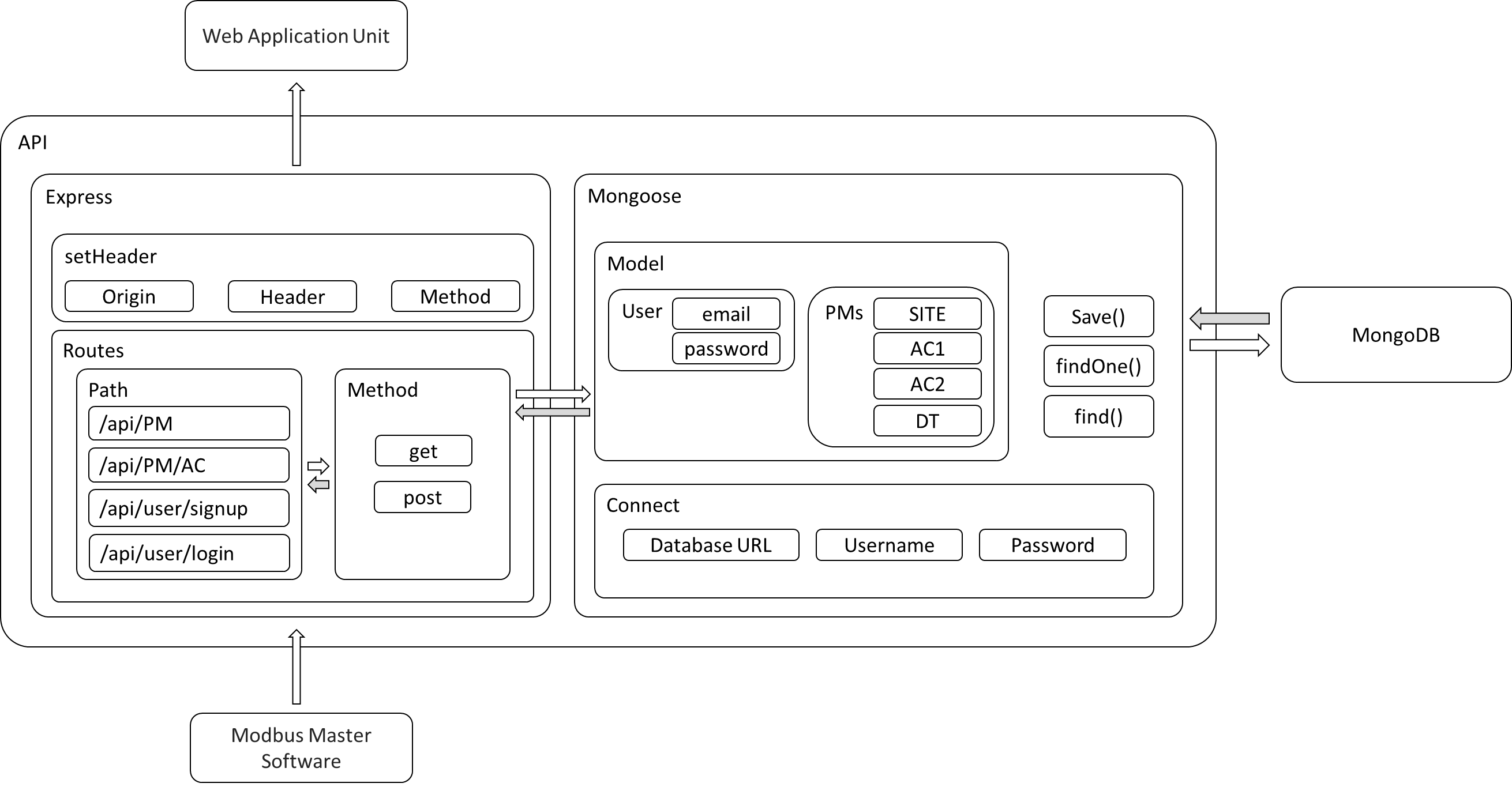
การอ่าน Document จะเป็นการดึงข้อมูลจากฐานข้อมูลมาใช้งานโดยจใช้ฟังก์ชัน find() ในพร็อปเพอร์ตี้ model ในการดึงข้อมูลทั้งหมด และฟังก์ชัน findOne() ในการค้นหาข้อมูลเพียง 1 ชุด

**สร้าง Middleware ด้วย Express**

Middleware จะเป็นการดักจับ request ที่เข้ามาใน server โดยฟังก์ชัน use() จะเป็นฟังก์ชันที่จะเรียกใช้ middleware ในทุก ๆ request ที่เข้ามาเช่นการกำหนด Header ไม่ว่าจะเป็นการกับหนด Origin , และ Method ที่อนุญาต

**API Routing ด้วย Express**

Routing จะเป็นการคัดกรอง URL ตามรูปแบบของ METHOD GET หรือ POST โดยฟังก์ชัน Router() ตามด้วย METHOD GET หรือ POST ในการเรียกใช้งานฟังก์ชันการทำงานต่าง ๆ ไม่ว่าจะเป็นการส่งหรืออ่านข้อมูลจากฐานข้อมูล

****

**ภาพที่ 3-3**

3.4) ขั้นตอนการพัฒนา Web Application สำหรับการแสดงผลข้อมูล

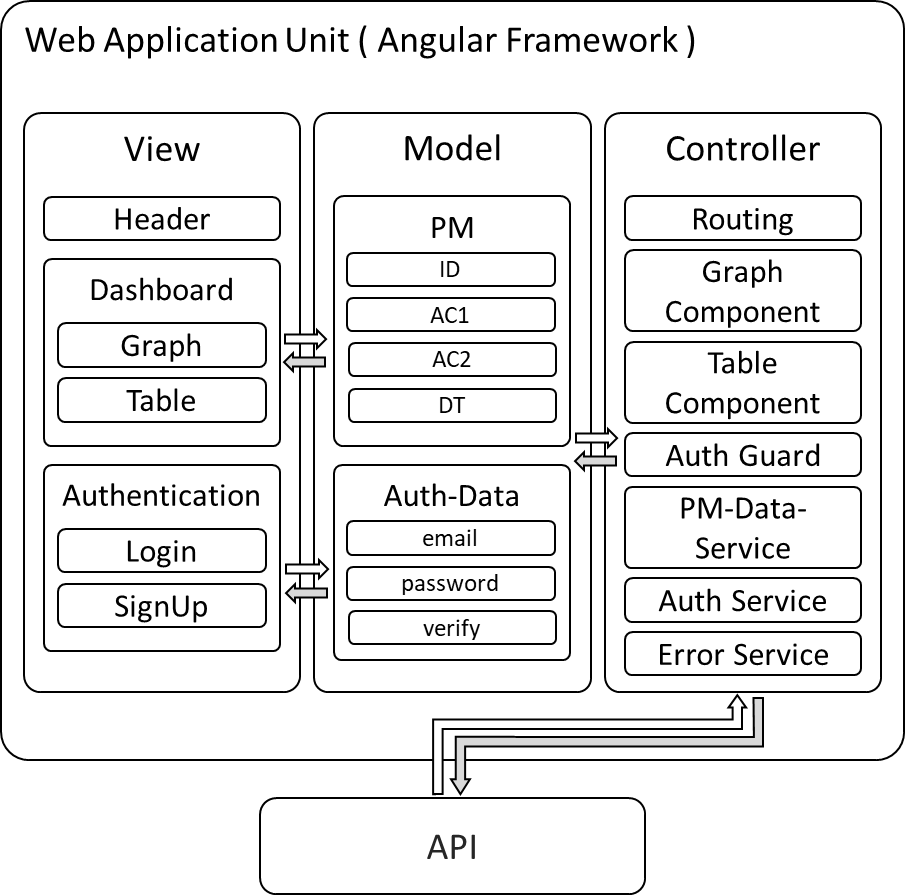
โครงสร้างของส่วนที่ใช้ในการแสดงผลมีลักษณะดังภาพที่ 3-4 โดยจะประอยู่ด้วยกัน 3 ส่วนหลักๆ คือ

ส่วนการแสดงผล Component ต่าง ๆ คือ ส่วนของ Header bar , หน้าต่าง login, signup, กราฟข้อมูล , ตารางข้อมูล

ส่วนของการจัดโครงสร้างของข้อมูลที่จะรับและส่งผ่าน API ต่าง ๆ ทั้งส่วนที่เป็นข้อมูลของค่าที่อ่านได้จากอุปกรณ์และส่วนของข้อมูลผู้ใช้

ส่วนควบคุมการทำงานต่าง ๆ ไม่ว่าจะเป็นการยืนยันตัวผู้ใช้( Auth Service ) , การดึงข้อมูลที่เก็บไว้( PM-Data Service ), การนำข้อมูลมาพล๊อตกราฟ( Graph Component ), การนำข้อมูลมาแสดงผลตาราง( Table Component ), การป้องกันการเข้าถึงข้อมูล( Auth Guard ), การจัดการการเข้าถึงหน้าเว็บไซต์( Routing ), การตรวจจับความผิดพลาด( Error Service )

โดยจะมีส่วนที่ต้องติดต่อกับ Sever อยู่ 3 ส่วนด้วยกันคือส่วนของ PM-Data Service , Auth Service และ Error Service ในการดึงข้อมูลมาประมวลผลในส่วนต่าง ๆ ต่อไป ส่วน Graph Component และ Table Component จะทำหน้าที่ในการดึงข้อมูลจาก PM-Data Service มา Render เป็นรูปกราฟและตารางออกมายังหน้า Dashboard ส่วน Auth Service จะทำหน้าที่ในการรอรับข้อมูลจากหน้า Login และ Sign Up และนำข้อมูลส่งไปให้ API ในการเปรียบเทียบข้อมูลผู้ใช้ในฐานข้อมูล

****

**ภาพที่ 3-4**

3.5) ขั้นตอนการเพิ่มความสามารถในการเข้ารหัสและการยืนยันตัวตน

ในส่วนนี้จะมุ่งเน้นในส่วนของการเพิ่มความสามารถของ Auth Service และ การทำ Server-Side Logic เช่นการยืนยันตัวตนไม่ให้ทำงานบน Browser เพื่อความปลอดภัย

**ส่วนของการลงชื่อเข้าใช้**



**ส่วนของการยืนยันตัวตน**

