ระบบควบคุมการเข้าใช้งานแอพพลิเคชันจากภายนอก

CONTROL ACCESS SYSTEM FOR THE THIRD-PARTY APPLICATIONS

โดย

ฉัตรแก้ว วณิชเจริญการ

CHATKAEW WANITCHAROENKARN

อาจารย์ที่ปรึกษา

รศ.ดร. โชติพัชร์ ภรณวลัย

รายงานนี้เป็นส่วนหนึ่งของวิชาการศึกษาอิสระ 2

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคเรียนที่ 2 ปีการศึกษา 2555

CONTROL ACCESS SYSTEM FOR THE THIRD-PARTY APPLICATIONS

CHATKAEW WANITCHAROENKARN

A REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENT OF THE COURSE

INDEPENDENT STUDY 2

MASTER OF SCIENCE PROGRAM IN INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2/2012

หัวข้อ	ระบบควบคุมการเข้าใช้งานแอพพลิเคชันจากภายนอก
นักศึกษา	นางสาวฉัตรแก้ว วณิชเจริญการ
รหัสนักศึกษา	53660501
ปริญญา	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
แขนงวิชา	เทคโนโลยีระบบสารสนเทศ
ปีการศึกษา	2555
อาจารย์ที่ปรึกษา	รศ.ดร. โชติพัชร์ ภรณวลัย

บทคัดย่อ

เทคโนโลยี OAuth ได้ถูกนำมาใช้เพื่อจัดการระบบการพิสูจน์ตัวตนให้เป็นแบบ Single Sign-On ที่มีระบบตรวจสอบการพิสูจน์ตัวตนเพียงระบบเดียว สำหรับหลายแอพพลิเคชัน เพื่อลดความซ้ำซ้อนในการพัฒนาระบบ และความซ้ำซ้อนของข้อมูลที่เกิดจากแอพพลิเคชันที่มีอยู่มากมายในองค์กร และเพิ่มความปลอดภัยของข้อมูลในการเข้าใช้แอพพลิเคชันจากภายนอกที่เกี่ยวข้องกับองค์กรได้อย่างมีประสิทธิภาพ

การพัฒนาระบบควบคุมการเข้าใช้งานแอพพลิเคชันนี้ มีวัตถุประสงค์ในการพัฒนาระบบการพิสูจน์ตัวตน โดยการวิเคราะห์ออกแบบ และพัฒนาระบบ เพื่อมาประยุกต์ให้เข้ากับระบบขององค์กร ซึ่งนำระบบมาช่วยในการจัดการกระบวนการพิสูจน์ตัวตนให้มีประสิทธิภาพมากยิ่งขึ้น โดยมีเซิร์ฟเวอร์เป็นตัวจัดการระบบการพิสูจน์ตัวตนเพียงอย่างเดียว ส่วนแอพพลิเคชันไม่ต้องทำการพัฒนาระบบการพิสูจน์ตัวตนขึ้นมา เพียงแค่ใช้บริการผ่านทางเซิร์ฟเวอร์ก็จะสามารถระบุตัวตนของผู้เข้าใช้งานได้

Title	CONTROL ACCESS SYSTEM FOR THE THIRD-PARTY APPLICATIONS
Student	Ms. Chatkaew Wanitcharoenkarn
Student ID	53660501
Degree	Master of Science
Program	Information Technology
Major	Information System Technology
Academic Year	2012
Advisor	Assoc.Prof.Dr. Chotipat Pornavalai

ABSTRACT

O-Authentication technology (OAuth) is introduced in the organization to be used as a new system for a "Single Sign-On" which is to monitor and identify the identity of each employee in system. The purpose of this system is to reduce the duplicate information that is previously caused by various systems in the third-party. OAuth is expected to improve the security when we log on to the system via Internet.

In addition, this new system improvement is to develop the process of monitoring and identifying the employees' identity. The new system has been designed to match with the characteristic of the organization, which improves the current process. There will be only one server that is able to manage and control the system. Moreover, it is not necessary to improve the current and design a new application to support the system since it has been easily managed by using a single server.

กิตติกรรมประกาศ

โครงงานฉบับนี้ได้ประสบความสำเร็จเป็นอย่างดีด้วยความกรุณา และสนับสนุนจากท่านอาจารย์ที่ปรึกษา รศ.ดร. โชติพัชร์ ภรณวลัย ที่เสียสละเวลาอันมีค่าให้คำปรึกษาและคำแนะนำแก่ข้าพเจ้า ช่วยตรวจสอบแก้ไขข้อบกพร่อง ตลอดจนให้ความรู้และข้อคิดเห็นที่เป็นประโยชน์อย่างยิ่ง ข้าพเจ้ารู้สึกซาบซึ้งในความอนุเคราะห์จากท่านอาจารย์ และขอขอบพระคุณเป็นอย่างสูง

ขอขอบคุณเพื่อน ๆ พี่ ๆ น้อง ๆ ที่ช่วยสนับสนุนการทำโครงการ ช่วยให้คำแนะนำและข้อเสนอแนะที่เป็นประโยชน์ต่อการทำโครงการ อีกทั้งยังให้ข้อมูลสำหรับการจัดทำโครงการให้มีประสิทธิภาพมากยิ่งขึ้น

สุดท้ายขอกราบขอบพระคุณบิดามารดา และบุคคลในครอบครัวที่ให้ความสนับสนุน ความรัก ความห่วงใย และคอยเป็นกำลังใจที่สำคัญที่สุดอย่างดีเสมอมา

ฉัตรแก้ว วณิชเจริญการ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

การศึกษาและพัฒนาเทคโนโลยี Single Sign-On (SSO) ซึ่งเป็นเทคโนโลยีการเก็บข้อมูลสิทธิ์ของผู้ใช้ระบบที่ได้รับอนุญาตสำหรับเข้าใช้งานระบบสารสนเทศ และจะทำการตรวจสอบการพิสูจน์ตัวตนเพียงครั้งเดียวสามารถเข้าใช้บริการจากระบบงานที่ประกอบด้วยหลาย ๆ ระบบซึ่งมีความแตกต่างกันให้ทำงานสัมพันธ์กัน โดยมีความเหมาะสมกับแนวทางด้านการรักษาความปลอดภัยของแต่ละองค์การจึงเป็นสิ่งจำเป็นต่อองค์กรในปัจจุบัน

1.2 วัตถุประสงค์ของการศึกษา

- เพื่อศึกษาและพัฒนาระบบสำหรับพิสูจน์ตัวตนและกำหนดสิทธิ์ของผู้ใช้งาน
- เพื่อช่วยในการแก้ปัญหาด้านรหัสผ่านที่มีความยุ่งยากต่อการใช้งาน
- ช่วยลดความซ้ำซ้อนของการพัฒนาระบบ
- ช่วยลดความซ้ำซ้อนในการเก็บรักษาข้อมูลที่ใช้ในการพิสูจน์ตัวตนซึ่งมีความจำเป็นในการเข้าใช้บริการระบบที่แตกต่างกัน โดยส่วนมากข้อมูลเหล่านั้น คือ รหัสผู้ใช้งาน และรหัสผ่าน ที่แตกต่างกันไปในแต่ละระบบ
- ช่วยให้เกิดความสะดวกในการรักษาความถูกต้องของข้อมูล
- เพื่อเพิ่มความน่าเชื่อถือในการรักษาความปลอดภัยของข้อมูลในระบบสารสนเทศขององค์กรให้เหมาะสมกับนโยบายการรักษาความปลอดภัยขององค์กรนั้น ๆ

1.3 ประโยชน์ที่คาดว่าจะได้รับ

ได้ระบบการพิสูจน์ตัวตนและการกำหนดสิทธิ์ให้แก่ผู้ใช้งาน โดยผู้ใช้งานระบบมีรหัสผู้ใช้งานเพียงชุดเดียว สามารถใช้รหัสนั้นในการเข้าใช้งานระบบสารสนเทศต่าง ๆ ภายในองค์กรและเข้าถึงทรัพยากรของระบบได้ในระดับที่องค์กรกำหนดให้เข้าถึงได้ และองค์กรยังสามารถจัดการสิทธิ์การเข้าถึงทรัพยากรบนระบบฐานข้อมูลตัวเดียวกัน เพื่อให้มีความยืดหยุ่น เสถียรภาพ ความสะดวกในการใช้งานของผู้ใช้งานในระบบ และสามารถจัดการกับความปลอดภัยของข้อมูลทรัพยากรในระบบสารสนเทศต่าง ๆ ได้อย่างเหมาะสม

1.4 ขอบเขตของการพัฒนาระบบ

- หากลไกการพิสูจน์ตัวตนผู้ใช้สำหรับเว็บแอพพลิเคชันเพื่อที่จะระบุกลไกการพิสูจน์ตัวตนผู้ใช้ที่เหมาะสมสำหรับการประมวลผลในลักษณะกระจาย
- ออกแบบ พัฒนาระบบพิสูจน์ตัวตนที่มีความสามารถในการเข้าสู่ระบบด้วยรหัสผ่านชุดเดียวกัน สำหรับระบบสารสนเทศที่เป็นเว็บแอพพลิเคชัน
- สามารถตรวจสอบข้อมูลรายชื่อผู้ใช้งาน/รหัสผู้ใช้งานว่ามีความถูกต้องและทำการติดต่อกับฐานข้อมูลที่ใช้เก็บข้อมูลสิทธิ์ของผู้ใช้งาน เพื่อความปลอดภัยในการจัดเก็บข้อมูล โดยผ่านเว็บแอพพลิเคชัน
- ทดสอบระบบบนระบบปฏิบัติการ Windows

1.5 ขั้นตอนการดำเนินงาน

- ศึกษาหลักการทำงานเกี่ยวกับพิสูจน์ตัวตนผู้ใช้บนระบบปฏิบัติการ Windows
- ศึกษาหลักการทำงานของเทคโนโลยี Single Sign-On
- ศึกษาเทคโนโลยีที่เกี่ยวข้องกับระบบ Single Sign-On เช่น OAuth, Kerberos, OpenID เป็นต้น
- วิเคราะห์ ออกแบบ และ ทดสอบระบบการพิสูจน์ตัวตนผู้ใช้ที่มีความสามารถในการทำ Single Sign-On สำหรับระบบสารสนเทศที่เป็นเว็บแอพพลิเคชัน
- สรุปผล นำเสนอระบบการพิสูจน์ตัวตน
- จัดทำรายงานและเอกสารประกอบ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ทฤษฎีการพิสูจน์ตัวตน และการควบคุมสิทธิ์การใช้งาน

ในปัจจุบันระบบคอมพิวเตอร์ได้ถูกคุกคามมากขึ้นทั้งจากไวรัสคอมพิวเตอร์หรือจากผู้ไม่ประสงค์ดี ซึ่งความมั่นคงปลอดภัยคอมพิวเตอร์ (Computer Security) ช่วยปกป้องเครื่องคอมพิวเตอร์รวมไปถึงอุปกรณ์ต่างๆที่เกี่ยวข้อง และที่สำคัญยังสามารถช่วยปกป้องข้อมูลที่ได้จัดเก็บไว้ภายในระบบหรือใช้ในความหมายความปลอดภัยทางข้อมูลสารสนเทศ (Information Security) ก็ได้ จุดประสงค์หลักของความปลอดภัยทางข้อมูลคือ ความลับ (Confidentiality) ความสมบูรณ์ (Integrity) ความพร้อมใช้ (Availability) และการห้ามปฏิเสธความรับผิดชอบ (Non-Repudiation) ของข้อมูลต่างๆภายในองค์กร (CIA-N) โดยมีรายละเอียดดังนี้

- การรักษาความลับ (Confidentiality) คือการรับรองว่าจะมีการเก็บข้อมูลไว้เป็นความลับ และผู้มีสิทธิ์เท่านั้นจึงจะเข้าถึงข้อมูลนั้นได้

- การรักษาความสมบูรณ์ (Integrity) คือการรับรองว่าข้อมูลจะไม่ถูกเปลี่ยนแปลงหรือทำลายไม่ว่าจะเป็นโดย อุบัติเหตุหรือโดยเจตนา

- ความพร้อมใช้ (Availability) คือการรับรองว่าข้อมูลและบริการการสื่อสารต่าง ๆ พร้อมที่จะใช้ได้ในเวลาที่ต้องการใช้งาน

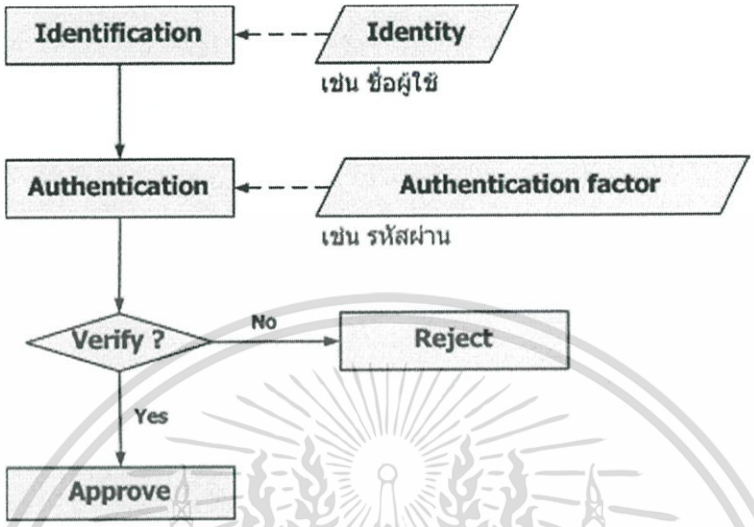
- การห้ามปฏิเสธความรับผิดชอบ (Non-Repudiation) คือวิธีการสื่อสารซึ่งผู้ส่งข้อมูลได้รับหลักฐานว่าได้มีการส่งข้อมูลแล้วและผู้รับก็ได้รับการยืนยันว่าผู้ส่งเป็นใคร ดังนั้นทั้งผู้ส่งและผู้รับจะไม่สามารถปฏิเสธได้ว่าไม่มีความเกี่ยวข้องกับข้อมูลดังกล่าวในภายหลัง

2.1.1 การพิสูจน์ตัวตน (Authentication)

การพิสูจน์ตัวตน คือขั้นตอนการยืนยันความถูกต้องของหลักฐาน (Identity) ที่แสดงว่าเป็นบุคคลที่กล่าวอ้างจริง ในทางปฏิบัติจะแบ่งออกเป็น 2 ขั้นตอน คือ

- การระบุตัวตน (Identification) คือขั้นตอนที่ผู้ใช้แสดงหลักฐานว่าตนเองคือใคร เช่น ชื่อผู้ใช้ (Username)

- การพิสูจน์ตัวตน (Authentication) คือขั้นตอนที่ตรวจสอบหลักฐานเพื่อแสดงว่าเป็นบุคคลที่กล่าวอ้างจริง



รูปที่ 2.1 แผนผังแสดงกระบวนการการพิสูจน์ตัวตน

จากรูปที่ 2.1 แสดงกระบวนการพิสูจน์ตัวตน ในขั้นแรกผู้ใช้จะทำการแสดงหลักฐานที่ใช้ในการพิสูจน์ตัวตนต่อระบบ ซึ่งในขั้นนี้คือการระบุตัวตน และในขั้นตอนต่อมาระบบจะทำการตรวจสอบหลักฐานที่ผู้ใช้นำมากล่าวอ้างซึ่งก็คือการพิสูจน์ตัวตน หลังจากระบบได้ทำการตรวจสอบหลักฐานเรียบร้อยแล้วถ้าหลักฐานที่นำมากล่าวอ้างถูกต้องจึงอนุญาตให้เข้าสู่ระบบได้ หากหลักฐานที่นำมากล่าวอ้างไม่ถูกต้องผู้ใช้จะถูกปฏิเสธจากระบบ

หลักฐานที่ผู้ใช้นำมากล่าวอ้างที่เกี่ยวกับเรื่องของความปลอดภัยนั้นสามารถจำแนกได้ 2 ชนิด

- Actual Identity คือหลักฐานที่สามารถบ่งบอกได้ว่า ในความเป็นจริงบุคคลที่กล่าวอ้างนั้นเป็นใคร

- Electronic Identity คือหลักฐานทางอิเล็กทรอนิกส์ซึ่งสามารถบ่งบอกข้อมูลของบุคคลนั้นได้ แต่ละบุคคลอาจมีหลักฐานทางอิเล็กทรอนิกส์ได้มากกว่า 1 หลักฐาน ตัวอย่างเช่น บัญชีชื่อผู้ใช้

กลไกของการพิสูจน์ตัวตน (Authentication Mechanisms) สามารถแบ่งออกได้เป็น 3 คุณลักษณะคือ

- สิ่งที่คุณมี (Possession Factor) เช่น กุญแจหรือเครดิตการ์ด เป็นต้น

- สิ่งที่คุณรู้ (Knowledge Factor) เช่น รหัสผ่าน (Passwords) หรือการใช้พิน (PINs) เป็นต้น
- สิ่งที่คุณเป็น (Biometric Factor) เช่น ลายนิ้วมือ รูปแบบเรตินา (Retinal Patterns) หรือใช้รูปแบบเสียง (Voice Patterns) เป็นต้น

กระบวนการพิสูจน์ตัวตนนั้นจะนำ 3 ลักษณะข้างต้นมาใช้ในการยืนยันหลักฐานที่นำมากล่าวอ้าง ทั้งนี้ขึ้นอยู่กับระบบ วิธีการที่นำมาใช้เพียงลักษณะอย่างใดอย่างหนึ่ง (Single-Factor Authentication) นั้นมีข้อจำกัดในการใช้ ตัวอย่างเช่น สิ่งที่คุณมี (Possession Factor) นั้นอาจจะสูญหายหรือถูกขโมยได้ สิ่งที่คุณรู้ (Knowledge Factor) อาจจะถูกดักฟัง เดา หรือขโมยจากเครื่องคอมพิวเตอร์ สิ่งที่คุณเป็น (Biometric Factor) จัดได้ว่าเป็นวิธีที่มีความปลอดภัยสูงอย่างไรก็ตามการที่จะใช้เทคโนโลยีนี้ได้นั้นจำเป็นต้องมีการลงทุนที่สูง เป็นต้น

ดังนั้นจึงได้มีการนำแต่ละคุณลักษณะมาใช้ร่วมกัน (Mult - Factor Authentication) ตัวอย่างเช่น ใช้สิ่งที่คุณมีกับสิ่งที่คุณรู้มาใช้ร่วมกัน เช่น การใช้ลายมือชื่อร่วมกับการใช้บัตรเครดิต หรือการใช้รหัสผ่านร่วมกับการใช้บัตร ATM เป็นต้น การนำแต่ละลักษณะของการพิสูจน์ตัวตนมาใช้ร่วมกันมากกว่า 1 ลักษณะ จะช่วยเพิ่มประสิทธิภาพในการรักษาความปลอดภัยของข้อมูล

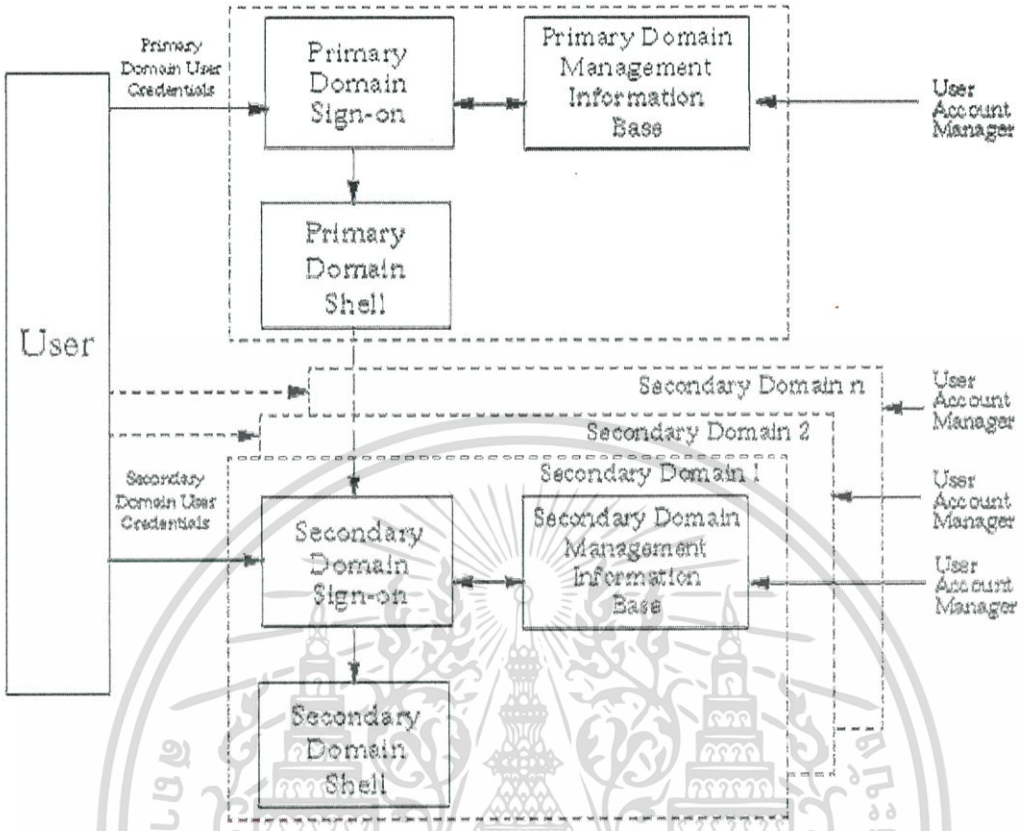
2.1.2 การควบคุมสิทธิ์การใช้งาน (Authorization)

ขั้นตอนในการอนุญาตให้แต่ละบุคคลสามารถเข้าถึงข้อมูลหรือระบบใดได้บ้าง ก่อนอื่นต้องทราบก่อนว่าบุคคลที่กล่าวอ้างนั้นคือใครตามขั้นตอนการพิสูจน์ตัวตนและต้องให้แน่ใจด้วยว่าการพิสูจน์ตัวตนนั้นถูกต้อง โดยตรวจสอบจาก Username และ Password ซึ่งจะมีการอนุญาติหรือกำหนดสิทธิ์ของการใช้งาน เช่น การเข้าถึงไดเรคทอรีของไฟล์ ชั่วโมงของการเข้าถึง จำนวนรวมของการจัดสรรพื้นที่จัดเก็บ และ อื่น ๆ

2.2 เทคโนโลยี Single Sign-On (SSO)

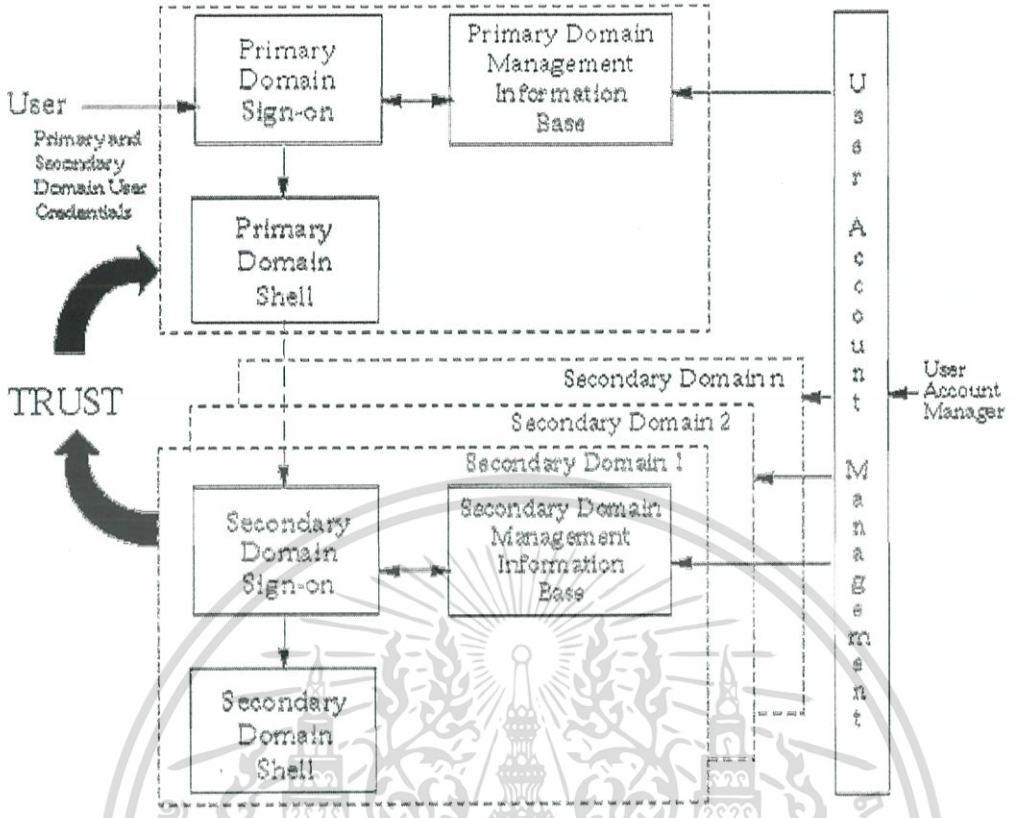
Single Sign-On เป็นกระบวนการซึ่งทำให้ผู้ใช้งานพิสูจน์ตัวตนเพียงครั้งเดียว สามารถเข้าใช้บริการจากระบบงานที่ประกอบด้วยหลาย ๆ แอพพลิเคชันซึ่งแตกต่างกันให้ทำงานสัมพันธ์กันโดยไม่จำเป็นต้องพิสูจน์ตัวตนอีกครั้งเมื่อเข้าใช้งานในแต่ละแอพพลิเคชัน

เมื่อมีการใช้งานระบบสารสนเทศที่มีหลาย ๆ แอพพลิเคชันอยู่ด้วยกัน ผู้ใช้งานจะต้องทำการพิสูจน์ตัวตน การเข้าใช้งานระบบหลาย ๆ ครั้ง ดังแสดงในรูปที่ 2.2



รูปที่ 2.2 การเข้าใช้งานระบบหลาย ๆ ระบบแบบดั้งเดิม

แต่เมื่อมีการพัฒนาเทคโนโลยี Single Sign-On ขึ้นมาจะช่วยให้ผู้ใช้งานไม่จำเป็นต้องทำการพิสูจน์ตัวตนหลาย ๆ ครั้งอีกต่อไป เนื่องจากระบบจะเก็บข้อมูลสิทธิ์ของผู้ใช้ที่ได้รับอนุญาตสำหรับผู้ใช้แต่ละคนและจะทำการตรวจสอบพิสูจน์ตัวตนเพียงครั้งเดียว ดังแสดงตามรูปที่ 2.3 ซึ่งผู้ใช้ระบบจะสามารถเข้าถึงทรัพยากรต่าง ๆ ได้ตามสิทธิ์ที่ได้รับอนุญาตจากระบบนั้น ๆ ทำให้ผู้ใช้งานสามารถเข้าใช้ระบบต่าง ๆ ได้ง่ายและรวดเร็วขึ้น นอกจากนี้ยังช่วยให้ผู้ดูแลระบบสามารถจัดการกับรหัสผู้ใช้งาน และรหัสผ่านได้อย่างเป็นระบบมากขึ้น



รูปที่ 2.3 การเข้าใช้งานระบบหลาย ๆ ระบบเพียงครั้งเดียว

2.3 เทคโนโลยี Open Authentication (OAuth)

OAuth เป็นมาตรฐานรองรับความถูกต้องและมีความปลอดภัย ช่วยให้ผู้ใช้งานสามารถแบ่งปันทรัพยากรต่าง ๆ เช่น ภาพถ่าย, วีดีโอ และ รายชื่อผู้ติดต่อ ที่อยู่บนเว็บไซต์ ได้โดยไม่ต้องเปิดเผยข้อมูลส่วนตัว (เช่น ชื่อผู้ใช้และรหัสผ่าน) นอกจากนั้น OAuth ยังสามารถกำหนดสิทธิ์ให้กับ Third-Party Application ให้สามารถเข้ามาใช้ข้อมูลได้ ซึ่ง OAuth ใช้ระบบ Token ในการรับรองความถูกต้องในการเข้าถึงข้อมูล

2.3.1 คุณสมบัติ OAuth

กระบวนการทำ Authentication แบบพื้นฐานของไคลเอนต์-เซิร์ฟเวอร์ในปัจจุบัน ผู้ใช้จะใช้ชื่อผู้ใช้และรหัสผ่านของตัวเองในการยืนยันเพื่อเข้าไปใช้ทรัพยากรบนเซิร์ฟเวอร์แต่สำหรับ OAuth นอกจากไคลเอนต์-เซิร์ฟเวอร์แล้วยังมีบทบาทใหม่คือ Resource Owner โดยไคลเอนต์ที่ต้องการเข้าถึงข้อมูลบนเซิร์ฟเวอร์ต้องได้รับอนุญาตจาก Resource Owner ไคลเอนต์ต้องมี Token

และ Matching Share-Secret โดยที่ Token จะเป็นสิ่งที่ใช้แสดงตัวแทน Resource Owner และเป็นสิ่งที่ใช้ระบุรายละเอียดของสิทธิ์ที่ได้รับเช่น ระดับการเข้าถึงข้อมูล ระยะเวลาในการใช้งาน

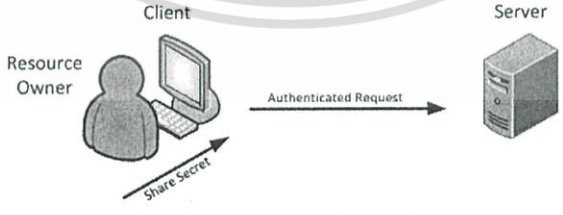
ปัจจุบัน OAuth ได้ออกมาตรฐานสำหรับรุ่น 1.0 เป็นที่เรียบร้อยโดยได้รวบรวมข้อดีของโพรโทคอลที่มีชื่อเสียง เช่น Google AuthSub, Yahoo BBAuth และ Flickr API เข้าด้วยกัน ส่วนที่ OAuth เหนือกว่าโพรโทคอลอื่นๆ คือมาตรฐานสำหรับการเชื่อมต่อที่นอกเหนือจากเว็บเซอร์วิส เช่น การเชื่อมต่อด้วยเดสก์ท็อปไคลเอนต์ (Desktop Client), Mobile Device, เซ็ท-ท็อปบ๊อกซ์ (Set-Top Box) และเว็บไซต์

2.3.1.1 กระบวนการทำงานของไคลเอนต์-เซิร์ฟเวอร์

ก่อนจะเข้าใจกลไกการทำงานของ OAuth เรามาศึกษากระบวนการทำงานของไคลเอนต์-เซิร์ฟเวอร์ก่อนเริ่มจาก

รูปที่ 2.4 ไคลเอนต์-เซิร์ฟเวอร์

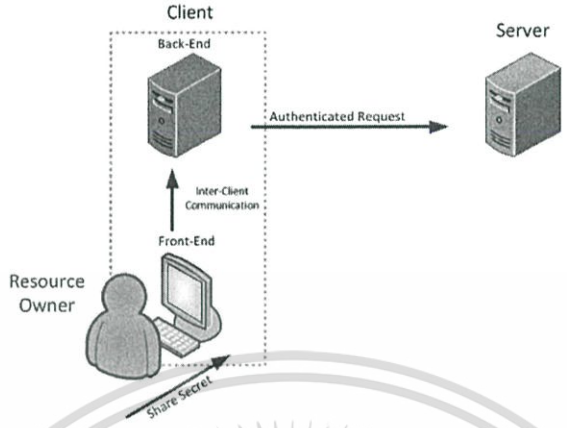
ไคลเอนต์ใช้ Credential ของตนเองเพื่อเชื่อมต่อกับเซิร์ฟเวอร์ โดยที่เซิร์ฟเวอร์ไม่สนใจในรายละเอียดว่า Credential ถูกส่งมาจากที่ไหน ต้องการเพียงแค่ Credential มีความถูกต้องตรงกับเซิร์ฟเวอร์



รูปที่ 2.5 ไคลเอนต์–เซิร์ฟเวอร์-ผู้ใช้

ไคลเอนต์สามารถทำหน้าที่เป็นตัวแทนของเอนทิตี (Entity) อื่นๆ โดยที่เอนทิตีเหล่านั้นสามารถเป็นได้ทั้งคนหรือระบบ ซึ่งไคลเอนต์จะต้องทำงานกับข้อมูลของ Resource Owner โดยใช้

Credential ของ Resource Owner เพื่อสร้างการร้องขอไปยังเซิร์ฟเวอร์ โดยข้อมูล Credential ที่ใช้ประกอบด้วยชื่อผู้ใช้และรหัสผ่าน



รูปที่ 2.6 เว็บ-เบส แอพพลิเคชัน

ไคลเอนต์ที่เป็นเว็บ-เบส แอพพลิเคชัน จะถูกแบ่งออกเป็น 2 ส่วน คือ ส่วนที่เป็น Front-end Component ที่ทำงานอยู่บนเบราว์เซอร์ของ Resource Owner และ Back-end Component ที่ทำงานอยู่บนเครื่องเซิร์ฟเวอร์ของไคลเอนต์ Resource Owner จะทำงานกับ Front-end ของไคลเอนต์ ขณะที่ส่วนที่เหลือจะคอยรับคำร้องขอเพื่อส่งต่อไปที่เซิร์ฟเวอร์ ถึงแม้ไคลเอนต์จะถูกแบ่งออกเป็นหลายส่วนแต่ก็ยังทำงานเป็นตัวแทนของ resource owner

2.3.1.2 2-Legged, 3-Legged, n-Legged

จำนวน Legged ที่ใช้หมายถึงจำนวน Party ที่เข้ามามีส่วนเกี่ยวข้องกับการทำงาน สำหรับการทำงานของ OAuthแบบธรรมดาที่ประกอบไปด้วยไคลเอนต์-เซิร์ฟเวอร์และ Resource Owner จะเรียกว่า 3-legged แต่สำหรับกรณีที่ไคลเอนต์กับ Resource Owner เป็นคนเดียวกันจะเรียกว่า 2-legged

2.3.1.3 องค์ประกอบของ OAuth

OAuth ประกอบด้วยผู้ใช้ 3 ส่วน คือ

- **ผู้ให้บริการ (Service Provider) :** Web Application ที่ช่วยให้เข้าใช้ผ่านทาง OAuth
- **ผู้ใช้ (Users) :** ผู้ที่มีแอคเคาท์ (Account) กับผู้ให้บริการ

- **ผู้ใช้บริการ (Consumer) :** เว็บไซต์หรือแอพพลิเคชันที่ใช้ OAuth เข้าถึงผู้ให้บริการในนามของผู้ใช้

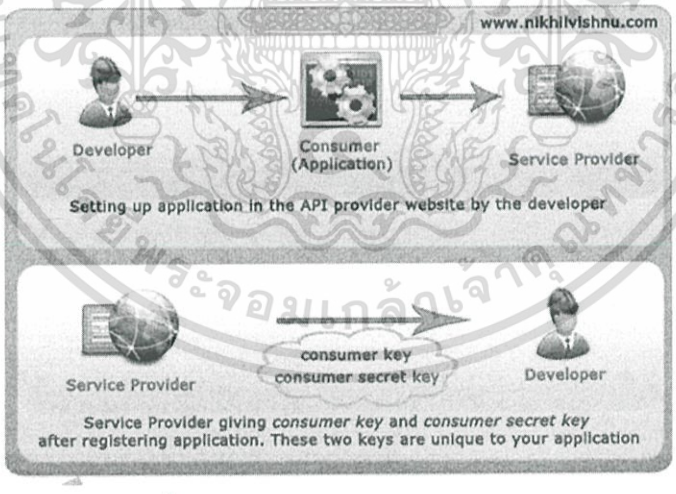
2.3.2. การนำOAuth มาใช้เพื่อแก้ปัญหาความปลอดภัย

Protected Resource คือข้อมูลใดๆ (รูปภาพ เอกสาร-รายชื่อ ข้อมูลทางการเงิน เป็นต้น) ก็ตามที่ถูกจัดเก็บโดยเซิร์ฟเวอร์และเมื่อใดก็ตามที่มีการเข้าถึงจะต้องผ่านกระบวนการ Authentication ก่อนเสมอข้อมูลนี้จะถูกควบคุมหรือเป็นเจ้าของโดย Resource Owner ดังนั้นใครก็ตามที่ต้องการเข้าถึงจำเป็นจะต้องได้รับ Authorize จาก Resource Owner ก่อน

2.3.2.1 ขั้นตอนการใช้งาน OAuth

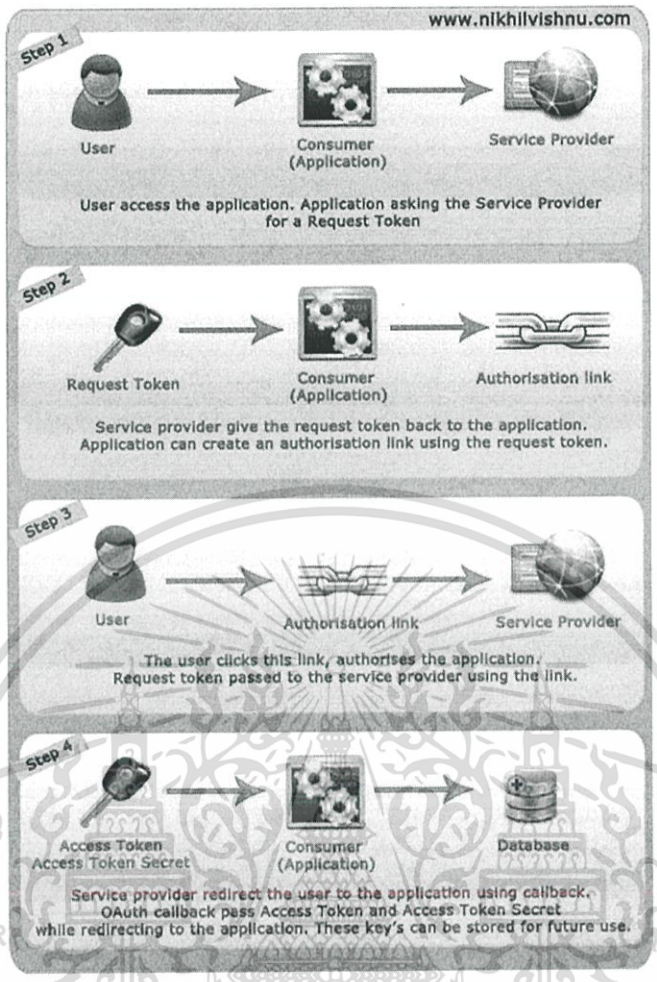
การนำ OAuth มาใช้งานสามารถแบ่งเป็น 3 ขั้นตอนดังนี้

ขั้นตอนที่ 1 ลงทะเบียนแอพพลิเคชัน ขั้นแรกต้องสร้างแอพพลิเคชันที่ใช้ OAuth Library และทำการติดตั้งแอพพลิเคชัน (ผู้ใช้งาน) กับเว็บไซต์ของผู้ให้บริการจะได้รับ Consumer Key และ Consumer Secret Key ซึ่งรหัสทั้งสองตัวจะต้องไม่ซ้ำกัน



รูปที่ 2.7 แผนภาพการตรวจสอบ OAuth

ขั้นตอนที่ 2 เมื่อรหัสได้รับการอนุมัติใช้ ก็สามารถเข้าสู่ระบบได้ แต่การขออนุมัติและการตอบรับจะแจ้งให้ผู้ใช้ทราบภายหลัง สำหรับกระบวนการขออนุมัตินี้แบ่งออกเป็น 4 ขั้นตอน ดังนี้



รูปที่ 2.8 แสดงขั้นตอนของการอนุมัติ

ขั้นที่ 1 เมื่อมีผู้ใช้งาน แอพพลิเคชันจะทำการร้องขอ HTTP ไปยังผู้ให้บริการเพื่อขอ Token

ขั้นที่ 2 ผู้ให้บริการอนุมัติคำขอ Token และจะร้องขอ Token Secret key ไปยังแอพพลิเค-ชัน แอพพลิเคชันจะสร้างการเชื่อมโยงการอนุมัติการขอ Token โดยผู้ให้บริการจะใช้ Token Secret key เพื่อระบุผู้ใช้งานแอพพลิเคชัน และจะปรากฏเว็บไซต์ให้กับผู้ใช้งาน

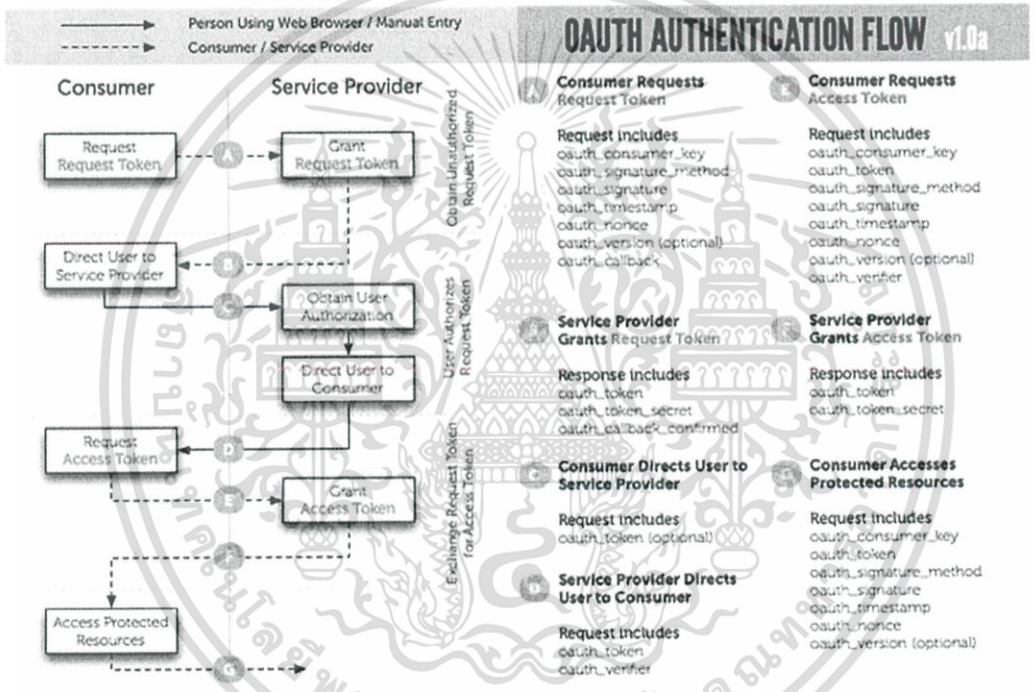
ขั้นที่ 3 ให้สิทธิ์แอพพลิเคชัน โดยคลิ๊กที่ลิงก์ที่ผู้ให้บริการแจ้งให้กับผู้ใช้ เพื่อช่วยให้ผู้ให้บริการสามารถเปลี่ยนเส้นทางไปยังแอพพลิเคชันที่เรียกใช้

ขั้นที่ 4 OAuth จะส่งผ่าน Access Token และ Access Token Secret ไปยังแอพพลิเคชัน ซึ่งรหัสทั้งสองตัวจะใช้สำหรับการเข้าถึงแอคเคาท์ของผู้ใช้ สามารถบันทึกรหัส ทั้งสองลงฐานข้อมูลเพื่อเก็บไว้ใช้งาน

ขั้นตอนที่ 3 การใช้งานผู้ใช้ทำการร้องขอไปยัง Application programming interface : API ก่อนที่จะร้องขอ OAuth ให้ทำการเพิ่ม Token ของผู้ใช้บริการและของผู้ใช้เข้าไป

2.3.2.2ขั้นตอนการทำ OAuth

เนื่องจากปัจจุบันเว็บไซต์ต่างๆ เช่น Twitter ได้ยกเลิก Basic Authentication ทุกๆ แอพพลิเคชันต้องใช้งานผ่านโพรโทคอล OAuth ซึ่งทุกๆ OAuth Library จะมีขั้นตอนการ Authentication ดังนี้



รูปที่ 2.9 แสดงขั้นตอน OAuth

ขั้นตอน A : ในขั้นตอนนี้เป็นการขอ “Request Token” จากผู้ใช้บริการโดยรายละเอียดของ Parameter ต่าง ๆ มีดังนี้

- OAuth_consumer_key คือ “key” ที่ผู้ให้บริการออกให้
- OAuth_signature_method คือ วิธีการเข้ารหัส Request เช่น HMAC-SHA1, RSA-SHA1

- OAuth_signature คือ ค่าที่ได้จากขั้นตอนการเข้ารหัสตาม OAuth_signature_method (โดยทั่วไป Parameter นี้จะถูกตั้งค่าให้อัตโนมัติในขั้นตอนสร้าง Request ของแต่ละ Library)
- OAuth_timestamp คือ เวลาที่ทำการ Request
- OAuth_nonce คือ ชุดของตัวหนังสือภาษาอังกฤษที่ถูกสุ่มขึ้นมาให้ไม่ซ้ำกันในแต่ละ Request เพื่อเอาไว้ตรวจสอบว่า Request นี้เป็น Request ที่ไม่เคยถูกใช้มาก่อน และป้องกันการโจมตีผ่าน HTTP
- OAuth_version คือ เวอร์ชั่นของ OAuth
- OAuth_callback คือ URL ที่จะให้ส่ง "Request Token" กลับไป

ขั้นตอน B : ในขั้นตอนนี้ผู้ให้บริการจะส่ง "Response Token" กลับไปให้รายละเอียดของ parameter ต่าง ๆ มีดังนี้

- OAuth_token คือ "Request Token" จากผู้ให้บริการ
- OAuth_token_secret คือ ค่าที่ผู้ให้บริการส่งมาพร้อมกับ "Request Token" เพื่อใช้ในการตรวจสอบ "Request Token" โดยค่านี้จะไม่ซ้ำกันในแต่ละ "Request Token"
- OAuth_callback_confirmed คือ เป็น TRUE/FALSE ขึ้นกับว่าได้รับการยืนยันจากผู้ให้บริการหรือไม่

ขั้นตอน C : ผู้ให้บริการจะส่งผู้ใช้ไปยังระบบที่ผู้ใช้บริการ เพื่อให้ผู้ใช้ทำการยืนยัน และให้ผู้ใช้ตัดสินใจว่าจะอนุญาตให้เข้าถึงข้อมูลของผู้ใช้ได้หรือไม่ Parameter ที่ใช้ในขั้นตอนนี้มีแค่ OAuth_token ซึ่งได้มาจากขั้นตอน B

ขั้นตอน D : ผู้ใช้บริการส่งผู้ใช้กลับไปยังผู้ให้บริการพร้อมทั้ง Parameter ดังนี้

- OAuth_token คือ "Request Token" จากขั้นตอน B (ในขั้นตอนนี้ "Request Token" ได้รับการอนุญาตให้ใช้งานได้จากผู้ใช้บริการแล้ว)
- OAuth_verifier คือ ค่าที่ผู้ใช้บริการส่งมาพร้อมกับ "Request Token" โดยค่านี้จะมีความเชื่อมโยงกับผู้ให้บริการ ค่านี้ถูกใช้ในขั้นตอน E เพื่อยืนยันว่าให้บริการ "Access Token" นั้นเป็นผู้ให้บริการเดียวกับที่ขอ "Request Token"

ขั้นตอน E : ผู้ให้บริการส่ง Request ไปยังผู้ใช้บริการเพื่อขอแลก "Request Token" เป็น "Access Token" รายละเอียดของ parameter ต่างๆที่เกี่ยวข้องมีดังนี้

- OAuth_consumer_key คือ "key" ที่ผู้ให้บริการส่งให้

- OAuth_token คือ “Request Token” ในขั้นตอน D
- OAuth_signature_method คือ วิธีการเข้ารหัส Request เช่น HMAC-SHA1, RSA-SHA1
- OAuth_signature คือ ค่าที่ได้จากขั้นตอนการเข้ารหัส
- OAuth_timestamp คือ เวลาที่ทำการ Request
- OAuth_nonce คือ ค่าชุดของตัวหนังสือภาษาอังกฤษที่ถูกสุ่มขึ้นมาให้ไม่ซ้ำกัน
- OAuth_version คือ เวอร์ชั่นของ OAuth
- OAuth_verifier คือ ค่าที่ได้จากขั้นตอน D

ขั้นตอน F : ผู้ใช้บริการส่ง “Access Token” ไปให้ผู้ให้บริการรายละเอียดของ Parameter ต่างๆที่เกี่ยวข้องมีดังนี้

- OAuth_token คือ “Access Token” ที่ได้รับจากผู้ใช้บริการ
- OAuth_token_secret คือ ค่าที่ผู้ใช้บริการส่งมาพร้อมกับ “Access Token” เพื่อใช้ในการตรวจสอบ “Access Token” โดยค่านี้จะไม่ซ้ำกันในแต่ละ “Access Token”

ขั้นตอน G : ผู้ให้บริการนำ “Access Token” ที่ได้เพื่อเข้าถึงข้อมูลของผู้ใช้ ซึ่ง parameter จะเหมือนขั้นตอนที่ผ่านมา

2.3.3. การเปรียบเทียบการใช้งานระหว่าง OAuth กับ Basic Authentication

ตารางที่ 2.1 แสดงการเปรียบเทียบความแตกต่างของ OAuth กับ Basic Authentication ในด้านต่างๆ

การเปรียบเทียบ	Basic Authentication	OAuth
การใช้งาน	ง่ายต่อการใช้งาน แค่มีชื่อผู้ใช้และรหัสผ่านก็สามารถใช้งานได้	ยากกว่า Basic Authentication เนื่องจากมีขั้นตอนที่ซับซ้อนและต้องได้รับสิทธิ์จากผู้ใช้ก่อนจึงจะสามารถใช้งานได้
การกำหนดสิทธิ์	ไม่มีการกำหนดสิทธิ์	มีการกำหนดสิทธิ์ในการเข้าถึงข้อมูลและระยะเวลาในการเข้าถึงข้อมูล

ตารางที่ 2.1 (ต่อ)

การนำไปใช้งานกับ Third-Party Application	ไม่สามารถกำหนดสิทธิ์ให้กับ Third-Party Application เข้ามาใช้งานได้ แต่ถ้าจะให้เข้ามาใช้งานต้องให้ข้อมูลส่วนตัว (ชื่อผู้ใช้และรหัสผ่าน) ทำให้เกิดความไม่ปลอดภัย	สามารถกำหนดสิทธิ์ให้กับ Third-Party Application ให้สามารถเข้ามาใช้ข้อมูลได้ โดยที่ไม่ต้องให้ข้อมูลส่วนตัว (ชื่อผู้ใช้และรหัสผ่าน)
ความปลอดภัย	มีความปลอดภัยต่ำกว่าOAuth เนื่องจากไม่มีการกำหนดสิทธิ์ในการเข้าถึงข้อมูล	มีความปลอดภัย เนื่องจากมีการกำหนดสิทธิ์ในการเข้าถึงข้อมูล

ถึงแม้ OAuth จะช่วยให้การเข้าถึงข้อมูลจากระบบหนึ่งไปสู่อีกระบบหนึ่งได้แต่ก็ยังไม่ใช่สิ่งที่ผู้ใช้ส่วนใหญ่ต้องการโดยผู้ใช้ส่วนใหญ่ต้องการความคล่องตัวในการแสดงตัวตนข้ามเว็บไซต์และการรักษาความปลอดภัยของข้อมูลเป็นหลัก แต่ขณะเดียวกันก็ยังคงต้องการได้รับความสะดวกในการได้รับสิทธิ์ให้เข้าถึงแหล่งข้อมูลได้โดยง่ายและสะดวก

บทที่ 3
การวิเคราะห์และออกแบบระบบ

3.1 ปัญหาที่พบในองค์กร

ระบบงานที่ใช้ในองค์กรแต่ละองค์กรมีอยู่หลายระบบงาน ผู้ใช้งานจำเป็นต้องเข้าใช้งานระบบเหล่านั้น ซึ่งแต่ละระบบจะมีลักษณะการยืนยันตัวตนที่แตกต่างกันออกไป โดยเฉพาะระบบที่พัฒนาจากองค์กรภายนอก (Third-party) ที่พนักงานบางแผนกมีความจำเป็นจะต้องใช้บริการ ในปัจจุบันองค์กรส่วนมากจะใช้ Active directory ในการควบคุมสิทธิการใช้งานระบบงานต่าง ๆ ในการเข้าใช้งานแอพพลิเคชันขององค์กร ซึ่งเราจะไม่สามารถใช้ Active directory ขององค์กรในการเข้าถึงระบบบางระบบที่พัฒนาจากองค์กรภายนอกที่เกี่ยวข้องกันได้ ทำให้ผู้ใช้งานมีรหัสการใช้งานเป็นจำนวนมาก ซึ่งอาจจะมีปัญหาในการจดจำรหัสเหล่านั้นในการเข้าใช้งานในแต่ละวัน นอกจากจะมีรหัสการใช้งานที่แตกต่างกันแล้ว รหัสแต่ละตัวจะมีระยะเวลาการใช้งานที่แตกต่างกันอีกด้วย เมื่อรหัสใดรหัสหนึ่งหมดอายุผู้ใช้จำเป็นจะต้องเปลี่ยนรหัสโดยที่ห้ามซ้ำกับของเดิม ซึ่งผู้ใช้อาจเกิดความสับสนกับรหัสผู้ใช้ที่มีอยู่มากมายนี้ ผู้ใช้งานบางคนอาจมีการจดรหัสต่าง ๆ ไว้บนโต๊ะเพื่อป้องกันความผิดพลาดในการเข้าใช้ระบบต่าง ๆ ทำให้มาตรฐานในความปลอดภัยของข้อมูลขององค์กรนั้น ๆ ต่ำลงได้

3.2 แนวคิดในการพัฒนาระบบ

สำหรับปัญหาที่พบในองค์กรดังกล่าวข้างต้นนั้นเราสามารถพัฒนาเทคโนโลยีที่ใช้ในการพัฒนาระบบขององค์กรและนำมาใช้ควบคุมการเข้าใช้งานแอพพลิเคชันต่าง ๆ ภายในองค์กรโดยใช้เทคโนโลยีที่มีชื่อว่า OAuth ที่มีหลักการการเชื่อมต่อแอพพลิเคชันต่าง ๆ เข้ากับ Service Provider เพื่อใช้บริการ Library ในการอนุมัติสิทธิ์ต่าง ๆในการเข้าถึงข้อมูลให้กับผู้ใช้งานที่ต้องการใช้แอพพลิเคชันหลาย ๆ ระบบ ซึ่งอาจจะเป็นระบบที่ไม่ได้อยู่ภายในองค์กรเดียวกัน (Third-party) โดยการนำเทคโนโลยี OAuth มาใช้ในการพัฒนาระบบดังกล่าวมีข้อดีตรงที่ ผู้ใช้จะมีรหัสผู้ใช้งานและรหัสผ่านในการเข้าสู่ระบบต่าง ๆ เพียงชุดเดียว ทำให้สะดวกในการจดจำรหัสและสามารถนำมาใช้งานได้อย่างมีประสิทธิภาพ นอกจากนี้ การกำหนดสิทธิ์ในการเข้าใช้จะสามารถทำได้บน Service Provider เท่านั้น โดยที่แอพพลิเคชันที่ใช้งานไม่จำเป็นต้องทราบข้อมูลของผู้ใช้ ซึ่ง OAuth จะทำ

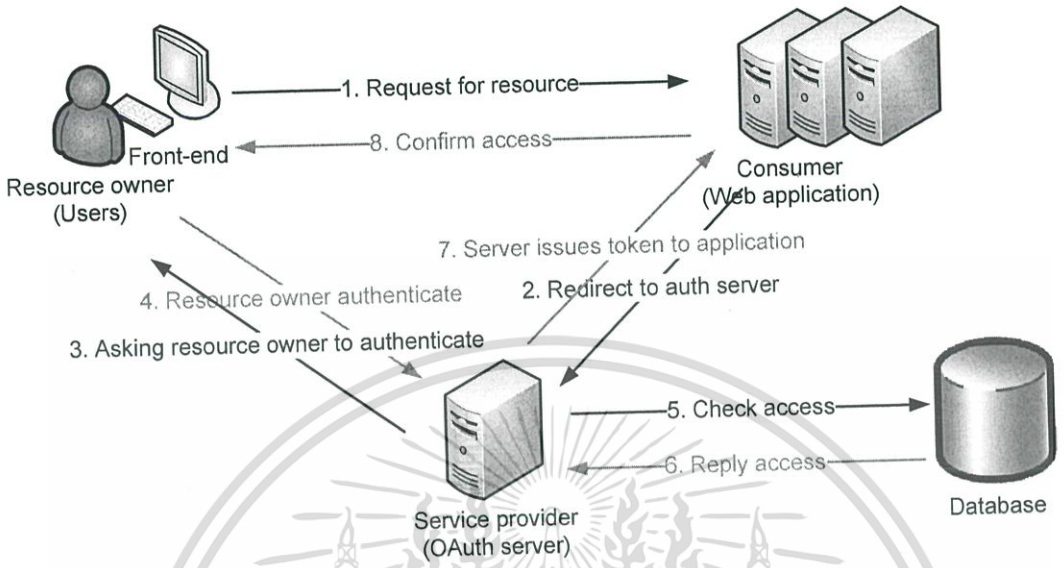
หน้าที่ตรวจสอบผู้ใช้งานโดยไม่ให้แอพพลิเคชันรู้ถึงวิธีที่ใช้ในการตรวจสอบรวมทั้งรหัสในการเข้าใช้ ทำให้ผู้ดูแลระบบมีความสะดวกในการจัดการดูแล และปรับปรุงระบบ เสมือนผู้ใช้เข้าใช้ แอพพลิเคชันผ่าน Service Provider ที่ให้บริการ OAuth Library นั่นเอง

การพัฒนาระบบในรูปแบบนี้ จะสามารถใช้เทคโนโลยีที่คล้ายคลึงกันได้อีกอย่างหนึ่ง ชื่อว่า OpenID แต่จะมีข้อแตกต่างอยู่ที่ เมื่อระบบได้ทำการเข้าสู่ระบบโดยใช้ OpenID แล้วระบบจะทำการติดต่อกับเซิร์ฟเวอร์ เพื่อขอข้อมูลผู้ใช้งานในการยืนยันตัวตน ซึ่งแอพพลิเคชันจะทำการบันทึก Cookie เอาไว้ เมื่อผู้ใช้คนเดิมเข้าใช้งานอีกครั้ง แอพพลิเคชันจะตรวจสอบกับ Cookie ที่มีอยู่ หากเคยยืนยันตัวตนแล้ว แอพพลิเคชันจะให้บริการกับผู้ใช้คนนั้นได้โดยไม่ต้องติดต่อกับเซิร์ฟเวอร์อีกครั้ง ซึ่งถ้าใช้ OAuth แอพพลิเคชันจะไม่สามารถเก็บประวัติการเข้าใช้งานของผู้ใช้ได้ ผู้ใช้จึงจำเป็นจะต้องทำการเข้าสู่ระบบใหม่หากมีการปิดเครื่อง หรือเริ่มทำงานในวันใหม่นั่นเอง ซึ่งการที่แอพพลิเคชันต้องทำการติดต่อกับเซิร์ฟเวอร์ใหม่ทุกครั้งที่ทำการรีบูทเครื่องใหม่นั้น จะช่วยให้ข้อมูลที่ได้รับมีความถูกต้องมากขึ้น สำหรับกรณีที่มีการเปลี่ยนแปลงแก้ไขข้อมูลบน OAuth Server หรือมีการเปลี่ยนแปลงสิทธิ์การเข้าถึงแอพพลิเคชันของผู้ใช้แต่ละคนบน OAuth Server นั้น

3.3 ความต้องการของระบบ

- สร้างระบบที่ให้บริการตรวจสอบข้อมูลและจำกัดสิทธิ์ในการเข้าใช้งานด้วยเทคโนโลยี OAuth เพื่อให้แอพพลิเคชันสามารถเรียกใช้บริการผ่านระบบนี้ สำหรับการยืนยันตัวตนของผู้ใช้ได้ ด้วยรหัสผ่านเพียงชุดเดียวที่ได้ทำการลงทะเบียนไว้กับระบบที่สร้างขึ้นมานี้
- ลดปัญหาการมีรหัสผู้ใช้งานที่มากเกินสำหรับระบบแต่ละระบบที่ต้องการการยืนยันตัวตนในลักษณะที่แตกต่างกัน
- เพิ่มประสิทธิภาพในการรักษาความปลอดภัยของข้อมูลสำคัญด้วยฐานข้อมูลที่ใช้เพียงฐานข้อมูลเดียวสำหรับหลาย ๆ แอพพลิเคชันที่ใช้ในองค์กร
- แอพพลิเคชันไม่สามารถรู้ได้ว่า ผู้ที่เข้าใช้ระบบเป็นใคร ทำให้แอพพลิเคชันเหล่านั้นไม่สามารถบันทึกประวัติการเข้าใช้จากผู้ใช้บริการ จะรู้แต่เพียงว่าสามารถเข้าถึงข้อมูลได้ในระดับไหนเท่านั้น เป็นการเพิ่มประสิทธิภาพในการบริหารจัดการสิทธิ์ต่าง ๆ ของผู้ใช้ทั้งหมด ที่ถูกเก็บอยู่บนฐานข้อมูลเดียว

3.4 หลักการทำงานโดยรวมของระบบ

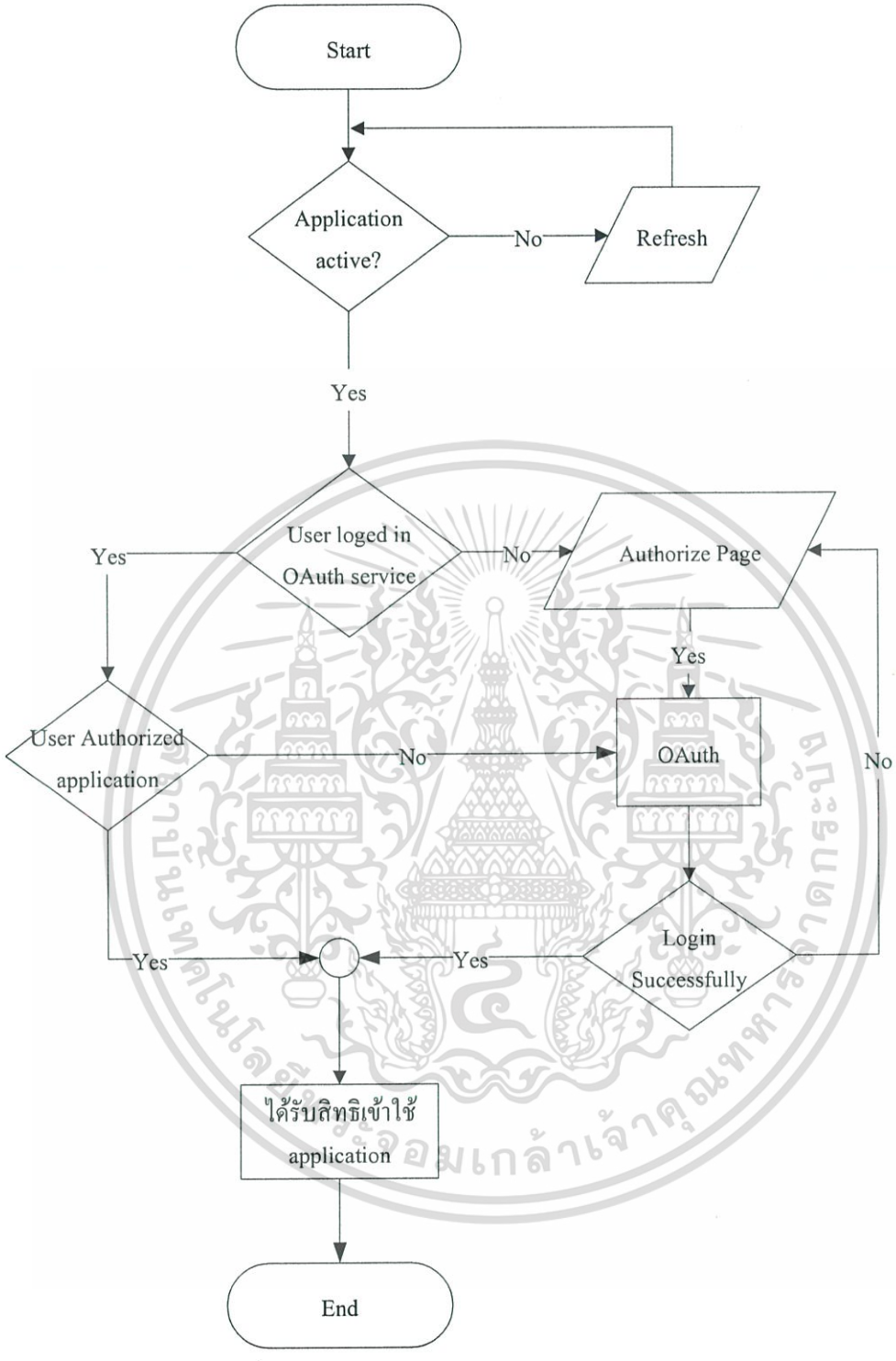


รูปที่ 3.1 ภาพรวมของระบบการพิสูจน์ตัวตนด้วยเทคโนโลยี OAuth

จากการศึกษาหลักการทำงานของ OAuth Protocol จะสามารถออกแบบโครงสร้างระบบโดยรวมที่พัฒนาด้วยเทคโนโลยี OAuth ได้ดังรูปที่ 3.1 มีคำอธิบายดังนี้ (1) เมื่อผู้ใช้ต้องการเข้าใช้งานแอพพลิเคชัน (2) แอพพลิเคชันจะทำการร้องขอไปที่ OAuth Server เพื่อทำการตรวจสอบผู้ใช้ดังกล่าว (3) เซิร์ฟเวอร์จะส่ง link สำหรับกรอกรหัสผ่านไปให้กับผู้ใช้โดยตรงเพื่อทำการยืนยันตัวตน (4) เมื่อผู้ใช้ทำการยืนยันรหัส (ID/Password) กลับมา (5,6) เซิร์ฟเวอร์จะทำการตรวจสอบความถูกต้องในการยืนยันตัวตน (7) เซิร์ฟเวอร์ส่ง Token กลับไปที่แอพพลิเคชัน เพื่อเป็นการยืนยันความถูกต้อง หลังจากนั้น (8) ผู้ใช้จะสามารถเข้าใช้งานแอพพลิเคชันได้ตามสิทธิ์ที่ได้รับอนุญาตจากเซิร์ฟเวอร์เท่านั้น

3.5 Flow Chart

Flow Chart แสดงลำดับการทำงานของระบบตั้งแต่ users เรียกใช้แอพพลิเคชันที่จำเป็นต้องมีการยืนยันตัวตนก่อนเข้าใช้งานแอพพลิเคชัน โดยแอพพลิเคชันจะทำการร้องขอไปที่ OAuth Server เพื่อให้ยืนยันสถานภาพของผู้ใช้กับแอพพลิเคชันให้สามารถเข้าใช้งานแอพพลิเคชันได้



รูปที่ 3.2 แสดงภาพรวมของลำดับการทำงานของระบบ

3.6 Use Case Diagram

ยูสเคสไดอะแกรมแสดงฟังก์ชันการทำงานของระบบที่จะพัฒนาขึ้น เป็นการอธิบายภาพรวมของระบบ สำหรับการทำงานของระบบควบคุมการเข้าใช้งานแอพพลิเคชัน โดยแสดงด้วยยูสเคสไดอะแกรมดังรูปที่ 3.3

รูปที่ 3.3 แสดง Use Case การทำงานของระบบ

คำอธิบาย Use Case Diagram

จากการออกแบบยูสเคสไดอะแกรมของระบบควบคุมการเข้าใช้งานแอพพลิเคชันขององค์กร โดยมี Actor ในระบบที่มีหน้าที่ความรับผิดชอบดังนี้

- User คือ ผู้ใช้ มีหน้าที่ในการเข้าใช้แอพพลิเคชันที่ใช้บริการระบบควบคุมการเข้าใช้งานแอพพลิเคชันขององค์กร

- Application คือ ผู้ขอใช้บริการระบบควบคุมการเข้าใช้งานแอพพลิเคชัน

- Developer application คือ ผู้พัฒนาระบบแอพพลิเคชันที่ต้องทำการลงทะเบียนกับเซิร์ฟเวอร์ก่อน จึงจะสามารถใช้บริการระบบควบคุมการเข้าใช้งานแอพพลิเคชันได้

รายละเอียดของแต่ละยูสเคสสามารถอธิบาย โดยเรียงตามลำดับการเข้าใช้งาน ตั้งแต่ผู้ขอใช้บริการร้องขอใช้งานจนถึงผู้ใช้ได้สิทธิ์เข้าใช้งาน ดังตารางต่อไปนี้

ตารางที่ 3.1 คำอธิบาย Use Case "Register consumer"

Use Case	**Register consumer**
วัตถุประสงค์	ผู้ขอใช้บริการทำการลงทะเบียนกับระบบ OAuth
เงื่อนไขเมื่อเริ่มต้น	-
แอคเตอร์ที่เกี่ยวข้อง	Developer application
สิ่งที่กระตุ้นการทำงาน	ผู้พัฒนาระบบเข้าไปทำการลงทะเบียนกับระบบ
Input	แอพพลิเคชันลงทะเบียนในหน้าลงทะเบียน
Output	เซิร์ฟเวอร์สร้าง Consumer key และ Consumer secret key ให้กับ Developer
รายละเอียด	เมื่อผู้พัฒนาระบบทำการลงทะเบียนกับเซิร์ฟเวอร์ เซิร์ฟเวอร์จะสร้าง Consumer key และ Consumer secret key ขึ้นมาให้กับผู้พัฒนาระบบ เพื่อใช้ในการยืนยันตัวตน ว่าแอพพลิเคชันได้ทำการลงทะเบียนกับเซิร์ฟเวอร์ไว้แล้ว

ตารางที่ 3.2 คำอธิบาย Use Case "Get key, secret key"

Use Case	**Get key, secret key**
วัตถุประสงค์	เป็นตัวยืนยันแอพพลิเคชันกับเซิร์ฟเวอร์
เงื่อนไขเมื่อเริ่มต้น	-
แอคเตอร์ที่เกี่ยวข้อง	Application
สิ่งที่กระตุ้นการทำงาน	-
Input	Developer ทำการลงทะเบียนกับทางเซิร์ฟเวอร์เพื่อขอใช้บริการ
Output	เซิร์ฟเวอร์สร้าง key และ secret key ให้กับ developer
รายละเอียด	หลังจากผู้ขอใช้บริการทำการลงทะเบียนกับทางเซิร์ฟเวอร์ เซิร์ฟเวอร์จะสร้าง key และ secret key ให้กับผู้ขอใช้ เอาไปใส่ในแอพพลิเคชัน เพื่อใช้บริการการพิสูจน์ตัวตนผ่านเซิร์ฟเวอร์

ตารางที่ 3.3 คำอธิบาย Use Case "Set target URL"

Use Case	**Set target URL**
วัตถุประสงค์	เรียกใช้งานเว็บแอพพลิเคชันที่ต้องการ
เงื่อนไขเมื่อเริ่มต้น	-
แอคเตอร์ที่เกี่ยวข้อง	User (ผู้ใช้)
สิ่งที่กระตุ้นการทำงาน	พิมพ์ URL ใน Address Bar ของ Browser ที่เลือกใช้บริการ เช่น Internet explorer
Input	URL ของ Web Application
Output	หน้า Web Application ทำงาน
รายละเอียด	เมื่อผู้ใช้งานเปิด Browser และทำการ input URL เข้าไปเพื่อขอใช้บริการแอพพลิเคชันที่ทำการลงทะเบียนไว้กับระบบ OAuth

ตารางที่ 3.4 คำอธิบาย Use Case "Request for request token, token secret"

Use Case	**Request for request token, token secret**
วัตถุประสงค์	เพื่อขอใช้บริการ authorization จาก OAuth Server
เงื่อนไขเมื่อเริ่มต้น	-
แอคเตอร์ที่เกี่ยวข้อง	Application
สิ่งที่กระตุ้นการทำงาน	แอพพลิเคชันติดต่อกับ OAuth Server เพื่อขอใช้บริการในการตรวจสอบสิทธิ์ของผู้ใช้
Input	-
Output	ขอ token ส่งไปที่ OAuth Server
รายละเอียด	- แอพพลิเคชันทำการร้องขอเข้าใช้บริการ OAuth Server โดยการร้องขอ token ไปที่ OAuth Server

ตารางที่ 3.5 คำอธิบาย Use Case "Authorization link"

Use Case	**Authorization link**
วัตถุประสงค์	เพื่อกำหนดสิทธิ์การใช้บริการให้กับแอพพลิเคชัน
เงื่อนไขเมื่อเริ่มต้น	Request for request token จากแอพพลิเคชัน
แอคเตอร์ที่เกี่ยวข้อง	User, Application
สิ่งที่กระตุ้นการทำงาน	แอพพลิเคชันทำการร้องขอ token ไปที่ OAuth Server
Input	-
Output	สิทธิ์ที่ผู้ใช้สามารถเข้าถึงแอพพลิเคชันได้
รายละเอียด	- แอพพลิเคชันขอใช้บริการไปที่ OAuth Server - OAuth Server ทำการส่ง token, token secret และ authorization link กลับไปที่แอพพลิเคชัน - แอพพลิเคชันส่ง authorization link ไปให้ผู้ใช้ทำการพิสูจน์ตัวตน - ผู้ใช้ทำการ authorize แอพพลิเคชันไปที่ OAuth Server เพื่อตรวจสอบสิทธิ์การเข้าใช้งานของผู้ใช้ - OAuth Server อนุญาติให้เข้าใช้แอพพลิเคชันได้

ตารางที่ 3.6 คำอธิบาย Use Case "User authentication"

Use Case	**User authentication**
วัตถุประสงค์	เป็นการส่ง Authorization link ให้กับผู้ใช้งานเพื่อยืนยันตัวตนโดยใส่รหัสผู้ใช้ กับรหัสผ่านเข้าไป
เงื่อนไขเมื่อเริ่มต้น	ผู้ใช้ขอใช้งานแอพพลิเคชัน
แอคเตอร์ที่เกี่ยวข้อง	User (ผู้ใช้)
สิ่งที่กระตุ้นการทำงาน	OAuth Server ส่ง Authorization link ไปให้ผู้ใช้งาน
Input	ผู้ใช้ทำการกรอกข้อมูล (User/Pass) เพื่อยืนยันตัวตนกับ OAuth Server
Output	ข้อมูลถูกส่งไปตรวจสอบกับฐานข้อมูล และตอบกลับไปที่แอพพลิเคชัน
รายละเอียด	- เมื่อมีการร้องขอเข้าใช้แอพพลิเคชันจะมีการติดต่อไปที่ OAuth Server เพื่อให้ตรวจสอบสิทธิ์ให้ โดย OAuth Server จะทำการส่ง link ไปให้ผู้ใช้ทำการยืนยันตัวตนโดยตรง เมื่อผู้ใช้ทำการกรอกข้อมูลเพื่อเป็นการยืนยันตัวตนให้กับ OAuth Server ทำการตรวจสอบและจะส่งสิทธิ์ที่สามารถเข้าใช้บริการได้ให้กับแอพพลิเคชัน

ตารางที่ 3.7 คำอธิบาย Use Case "Check database"

Use Case	**Check database**
วัตถุประสงค์	ตรวจสอบข้อมูลผู้ใช้บนฐานข้อมูล
เงื่อนไขเมื่อเริ่มต้น	-
แอคเตอร์ที่เกี่ยวข้อง	-
สิ่งที่กระตุ้นการทำงาน	ผู้ใช้ทำการกรอกข้อมูล (user/pass) และยืนยันตัวตนผ่าน authorization link ซึ่งถูกส่งมายัง User authentication โมดูล
Input	User/pass จากการยืนยันตัวตนของผู้ใช้บนหน้า login
Output	พบข้อมูลของผู้ใช้
รายละเอียด	- ทำการตรวจสอบข้อมูลที่ได้รับมาจากหน้า login บนฐานข้อมูล

ตารางที่ 3.8 คำอธิบาย Use Case "Access token, access token secret"

Use Case	**Access token, access token secret**
วัตถุประสงค์	OAuth Server ตอบกลับ แอพพลิเคชัน ด้วย access token และ access token secret
เงื่อนไขเมื่อเริ่มต้น	แอพพลิเคชันร้องขอ access token จาก OAuth Server และผู้ใช้ทำการยืนยันตัวตนในการเข้าใช้บริการกับ OAuth Server เรียบร้อยแล้ว
แอคเตอร์ที่เกี่ยวข้อง	-
สิ่งที่กระตุ้นการทำงาน	ผู้ใช้ทำการกำหนดสิทธิ์การเข้าใช้งาน แอพพลิเคชันผ่าน authorization link และส่งไปยัง OAuth Server
Input	-
Output	Access token, access token secret ซึ่งถูกสร้างโดย OAuth Server แล้วส่งกลับให้กับแอพพลิเคชัน
รายละเอียด	- ผู้ใช้ร้องขอการเข้าถึงแอพพลิเคชันไปที่ OAuth Server - OAuth Server ทำการกำหนดสิทธิ์ให้กับผู้ใช้งาน แล้วส่ง access token และ access token secret กลับไปยืนยันให้กับแอพพลิเคชัน

ตารางที่ 3.9 คำอธิบาย Use Case "Protected resource"

Use Case	**Protected resource**
วัตถุประสงค์	สิทธิ์การเข้าถึงข้อมูลของแอพพลิเคชันจาก OAuth Server
เงื่อนไขเมื่อเริ่มต้น	-
แอคเตอร์ที่เกี่ยวข้อง	Application
สิ่งที่กระตุ้นการทำงาน	ผู้ใช้ทำการยืนยันตัวตนกับ OAuth Server แล้ว
Input	-
Output	ข้อมูลผู้ใช้ที่แอพพลิเคชันร้องขอจาก OAuth Server
รายละเอียด	- เมื่อผู้ใช้งานทำการยืนยันตัวตนสมบูรณ์แล้ว - แอพพลิเคชันจะได้รับสิทธิ์การเข้าถึงข้อมูลของผู้ใช้จาก OAuth Server ตามที่ได้ลงทะเบียนไว้ในตอนแรก

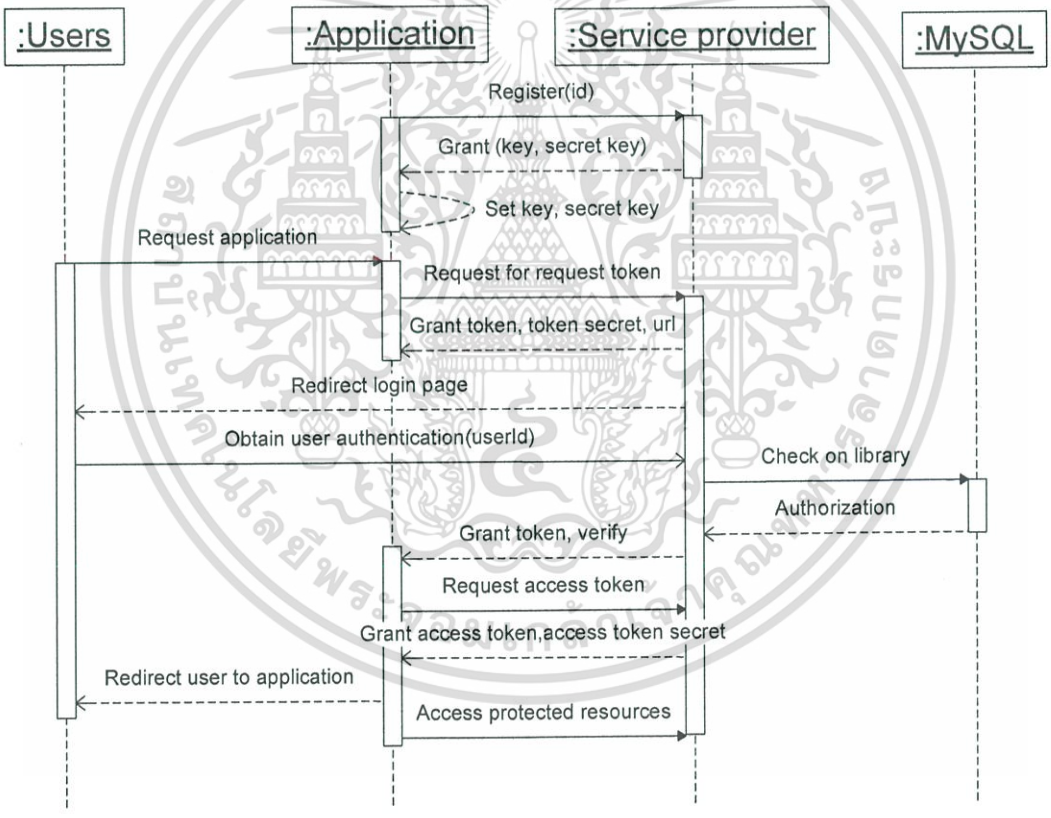
ตารางที่ 3.10 คำอธิบาย Use Case "Access application"

Use Case	**Access application**
วัตถุประสงค์	เว็บแอพพลิเคชันยอมให้ผู้ใช้เข้าใช้บริการได้
เงื่อนไขเมื่อเริ่มต้น	-
แอคเตอร์ที่เกี่ยวข้อง	User (ผู้ใช้)
สิ่งที่กระตุ้นการทำงาน	เว็บแอพพลิเคชันได้รับ Token การยืนยันตัวตนของผู้ใช้งานจาก OAuth Server เรียบร้อยแล้ว
Input	Token ที่ได้รับมาจาก OAuth Server เพื่อยืนยันตัวตนของผู้ใช้งาน
Output	ผู้ใช้สามารถเข้าใช้งานแอพพลิเคชันได้
รายละเอียด	- เมื่อแอพพลิเคชันได้รับ Token จาก OAuth Server เพื่อยืนยันสิทธิ์การเข้าใช้งานของผู้ใช้เรียบร้อยแล้ว - เว็บแอพพลิเคชัน ให้บริการกับผู้ใช้งานตามสิทธิ์ที่ได้รับมาจาก OAuth Server

3.7 Sequence Diagram

Sequence Diagram แสดงลำดับของกิจกรรมต่าง ๆ ที่เกิดขึ้นในระบบตั้งแต่ผู้ใช้งานทำการร้องขอใช้บริการแอพพลิเคชันจนได้รับสิทธิ์ในการเข้าถึงข้อมูลในแอพพลิเคชัน โดยผ่านระบบตรวจสอบข้อมูลด้วยเทคโนโลยี OAuth ซึ่งสามารถอธิบายหลักการทำงานได้ดังนี้

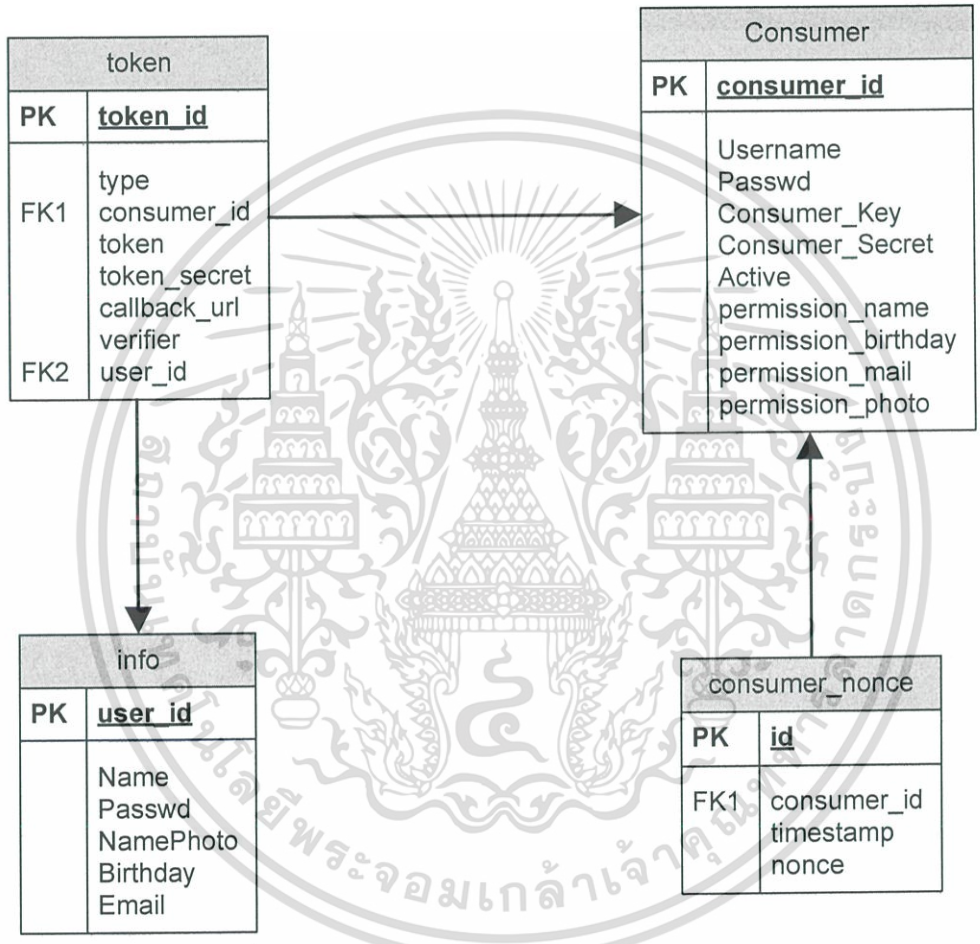
แอพพลิเคชันจะต้องทำการ register กับ OAuth Server ซึ่งจะได้รับ consumer key และ consumer secret key โดยแอพพลิเคชันจะนำ key ทั้ง 2 ตัวมาใช้ในการติดต่อกับ OAuth Server เพื่อยืนยันตัวตนของผู้เข้าใช้งานก่อน หลังจากนั้นผู้ใช้จะทำการยืนยันตัวตนกับ OAuth Server และขอสิทธิ์การเข้าถึงแอพพลิเคชัน ดังรูปที่ 3.4



รูปที่ 3.4 แสดง Sequence Diagram การทำงานของระบบการพิสูจน์ตัวตน

3.8 การออกแบบฐานข้อมูลของระบบ

จากการวิเคราะห์หลักการทำงานของเทคโนโลยี OAuth สามารถออกแบบฐานข้อมูลของระบบ OAuth library ที่ใช้ในการเก็บข้อมูลในส่วนของเซิร์ฟเวอร์ เช่น ข้อมูลผู้ใช้ (Users), ข้อมูลผู้ใช้บริการ (consumer), Key และ Token ต่าง ๆ ที่จำเป็นต่อการลงทะเบียนเพื่อใช้ยืนยันตัวตนได้ ดังรูปที่ 3.5



รูปที่ 3.5 แสดงระบบฐานข้อมูลของ OAuth Library

จากรูปที่ 3.5 ประกอบซึ่งแสดงตารางฐานข้อมูลของ OAuth provider จะสามารถอธิบายความหมายของตัวแปรต่าง ๆ ได้ดังนี้

ตารางที่ 3.11 แสดงโครงสร้างและรายละเอียดของตาราง consumer

Field	Type	Length	Key	Description
ID	int	11	PK	รหัสของผู้ใช้บริการ
Username	varchar	50		ชื่อของผู้ใช้บริการ
Passwd	varchar	50		รหัสผ่านของผู้ใช้บริการ
Consumer_Key	varchar	250		รหัสที่ใช้ในการยืนยันตัวตนกับ server
Consumer_Secret	varchar	250		รหัสที่ใช้ในการยืนยันตัวตนกับ server
Active	int	11		สถานะของผู้ใช้บริการ
permission_name	int	11		สิทธิ์ในการขอใช้ข้อมูลชื่อ
permission_birthday	int	11		สิทธิ์ในการขอใช้ข้อมูลวันเกิด
permission_mail	int	11		สิทธิ์ในการขอใช้ข้อมูลอีเมล
permission_photo	int	11		สิทธิ์ในการขอใช้ข้อมูลรูป

ตารางที่ 3.12 แสดงโครงสร้างและรายละเอียดของตาราง consumer_nonce

Field	Type	Length	Key	Description
id	int	11	PK	รหัสของข้อมูล
consumer_id	varchar	11	FK	รหัสของผู้ใช้บริการ
timestamp	bigint	20		เก็บค่าเวลาที่ใช้งาน
nonce	varchar	250		ข้อมูลของ OAuth provider

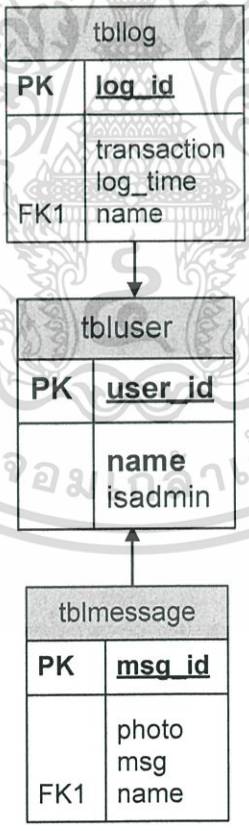
ตารางที่ 3.13 แสดงโครงสร้างและรายละเอียดของตาราง token

Field	Type	Length	Key	Description
id	int	11	PK	รหัสของข้อมูล
type	int	11		ประเภทของ token
consumer_id	int	11	FK	รหัสของผู้ใช้บริการ
token	varchar	250		รหัสที่ใช้ในการยืนยันตัวตนกับ server
token_secret	varchar	250		รหัสที่ใช้ในการยืนยันตัวตนกับ server
callback_url	varchar	250		url ที่ใช้ในการร้องขอใช้บริการ
verifier	varchar	250		รหัสที่ใช้ในการยืนยันตัวตนกับ server
user_id	int	11	FK	รหัสของผู้ใช้

ตารางที่ 3.14 แสดงโครงสร้างและรายละเอียดของตาราง info

Field	Type	Length	Key	Description
user_id	int	11	PK	รหัสของผู้ใช้
Name	varchar	40		ชื่อของผู้ใช้
Passwd	varchar	30		รหัสผ่านของผู้ใช้
Namephoto	varchar	30		ข้อมูลชื่อรูปภาพของผู้ใช้
Birthday	varchar	20		ข้อมูลวันเกิดของผู้ใช้
Email	varchar	100		ข้อมูล Email ของผู้ใช้

ฐานข้อมูลในส่วนของแอพพลิเคชันที่ใช้ในการเก็บสิทธิ์ของผู้ใช้งานและเก็บประวัติการเข้าใช้งานแอพพลิเคชัน เพื่อใช้ในดูแลรักษาระบบให้มีความปลอดภัยและมีความถูกต้องของข้อมูลเพิ่มมากขึ้น สามารถออกแบบฐานข้อมูลได้ดังนี้



รูปที่ 3.6 แสดงระบบฐานข้อมูลของเว็บแอพพลิเคชัน

จากรูปสามารถอธิบายความหมายของตัวแปรต่าง ๆ ได้ดังตารางต่อไปนี้

ตารางที่ 3.15 แสดงโครงสร้างและรายละเอียดของตาราง tblmessage

Field	Type	Length	Key	Description
msg_id	int	11	PK	รหัสของข้อความ
name	varchar	40	FK	ชื่อของผู้ใช้
photo	varchar	50		รูปของผู้ใช้
msg	varchar	200		ข้อความที่โพส

ตารางที่ 3.16 แสดงโครงสร้างและรายละเอียดของตาราง tbllog

Field	Type	Length	Key	Description
log_id	int	11	PK	รหัสของข้อมูลประวัติ
name	varchar	40	FK	ชื่อของผู้ใช้
transaction	varchar	100		สถานะการเข้าใช้งาน
log_time	timestamp	-		วัน-เวลาที่เข้าใช้งาน

ตารางที่ 3.17 แสดงโครงสร้างและรายละเอียดของตาราง tbluser

Field	Type	Length	Key	Description
user_id	int	11	PK	รหัสของผู้ใช้
name	varchar	40		ชื่อของผู้ใช้
isadmin	int	11		สิทธิ์การเป็นแอดมิน

บทที่ 4

การพัฒนาระบบ

4.1 เครื่องมือที่ใช้ในการพัฒนาระบบ

- ระบบปฏิบัติการ CentOS 5.9 สำหรับเครื่องเซิร์ฟเวอร์
- ภาษาที่ใช้ในการพัฒนาระบบใช้ PHP 5.3.3 ที่ลง OAuth extension
- ฐานข้อมูลของระบบใช้ MySQL
- เครื่องมือการจัดการระบบฐานข้อมูลแบบ GUI phpMyAdmin
- บราวเซอร์ที่ใช้ทดสอบระบบ Firefox, Google Chrome

4.2 โครงสร้างการทำงานของระบบ

การพัฒนาระบบจะแบ่งออกเป็น 2 ส่วนด้วยกัน คือ ส่วนของเซิร์ฟเวอร์ ที่ให้บริการสำหรับการพิสูจน์ตัวตนของผู้ใช้ ที่ถูกพัฒนาขึ้นมาโดยหลักการของเทคโนโลยี OAuth กับ ส่วนของผู้ใช้บริการหรือแอพพลิเคชันที่จำลองขึ้นมาร้องขอการใช้ระบบพิสูจน์ตัวตนจากเซิร์ฟเวอร์ ทั้งสองส่วนนี้มีการดำเนินการพัฒนาระบบโดยใช้ภาษา PHP ที่มีฟังก์ชันการทำงานของเทคโนโลยี OAuth ให้สามารถนำมาพัฒนาระบบขึ้นได้โดยอ้างอิงกับขั้นตอนการทำงานของเทคโนโลยีนี้ และมีการออกแบบฐานข้อมูลขึ้นเพื่อเก็บรายละเอียดต่าง ๆ ของผู้ใช้งาน ซึ่งดัดแปลงมาจากฐานข้อมูลของเทคโนโลยี OAuth บางส่วนเพื่อให้เข้ากับหลักการการทำงานของเทคโนโลยี OAuth คือ ตารางที่ใช้เก็บ key, token และ consumer nonce โดยจะเพิ่มตารางที่ใช้เก็บข้อมูลของผู้ใช้เข้าไปเพื่อเก็บรายละเอียดของผู้ใช้งาน และใช้เป็นข้อมูลอ้างอิงตัวบุคคลนั้น ๆ

ขั้นตอนการทำงานของระบบจะเริ่มจาก ผู้ใช้บริการหรือแอพพลิเคชันเข้าไปขอใช้บริการจากเซิร์ฟเวอร์ โดยการลงทะเบียนที่หน้าเว็บ ซึ่งจะได้ค่าที่ใช้ในการยืนยันตัวตนมา 2 ตัว เพื่อนำไปเก็บไว้พัฒนาแอพพลิเคชันขึ้นมาให้สามารถใช้งานการพิสูจน์ตัวตนจากเซิร์ฟเวอร์ได้ โดยที่แอพพลิเค-ชันจะต้องทำการขอข้อมูลที่ต้องการจากทางเซิร์ฟเวอร์เพื่อที่จะทราบว่าผู้ใช้คนไหนกำลังเข้าใช้งานแอพพลิเคชันอยู่ผ่านทางระบบลงทะเบียนนี้ด้วย หลังจากนั้น ผู้ใช้งาน (Users) จะต้องทำการลงทะเบียนกับเซิร์ฟเวอร์ก่อน จึงจะสามารถใช้งานแอพพลิเคชันได้ เมื่อผู้ใช้เข้ามาขอใช้แอพพลิเค-ชัน แอพพลิเคชันจะถามทางเซิร์ฟเวอร์ให้ทำการพิสูจน์ตัวตนของผู้ใช้ให้ โดยแอพพลิเคชันจะได้รับ

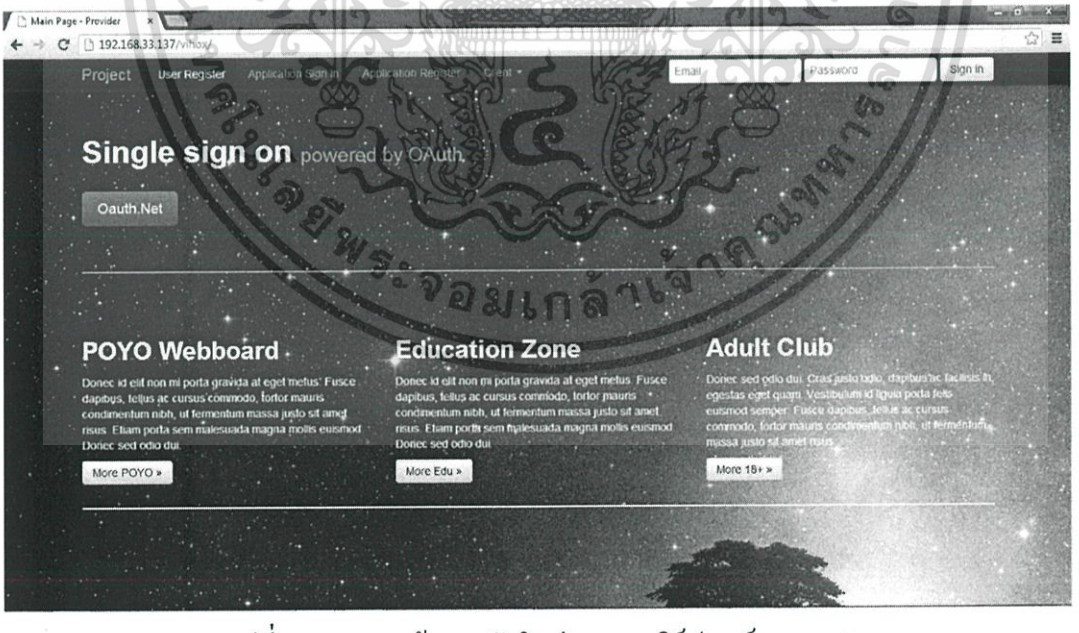
โทเคนและลิงค์ในการยืนยันตัวตนกลับมา และทำการส่งหน้าลิงค์ที่ใช้ในการยืนยันตัวตนไปให้กับผู้ใช้เข้าสู่ระบบ เมื่อผู้ใช้ทำการเข้าสู่ระบบเรียบร้อยแล้ว เซิร์ฟเวอร์จะส่งค่ากลับมาให้กับแอพพลิเคชันเพื่อยืนยันว่าผู้ใช้มีสิทธิ์เข้าใช้งานจริง หลังจากนั้นแอพพลิเคชันจะทำการขอข้อมูลที่จำเป็นกลับไปที่เซิร์ฟเวอร์และเมื่อเซิร์ฟเวอร์ตอบกลับข้อมูลกลับมา ผู้ใช้จะสามารถเข้าใช้งานแอพพลิเคชันได้อย่างสมบูรณ์

ขั้นตอนการทำงานของระบบสามารถอธิบายอย่างละเอียดได้ในหัวข้อต่อไป

4.3 การพัฒนาระบบการพิสูจน์ตัวตนสำหรับผู้ให้บริการ (OAuth)

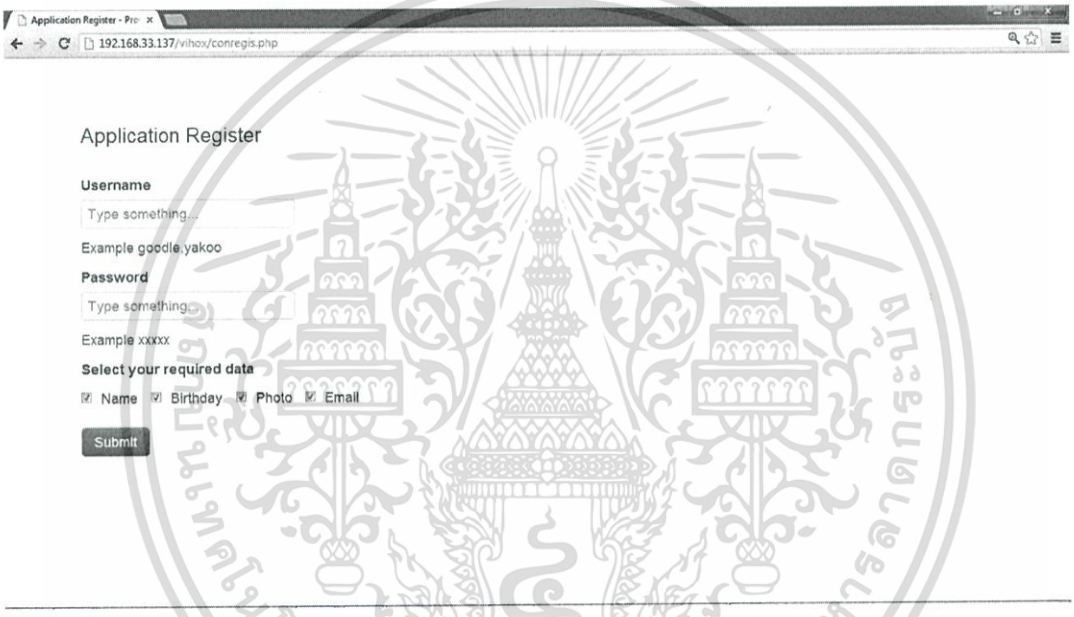
ในส่วนของผู้ให้บริการนี้จะนำเทคโนโลยี OAuth มาช่วยในการพัฒนา มีส่วนประกอบหลัก ๆ ดังนี้

- หน้าหลักของส่วนเซิร์ฟเวอร์ โดยจะมีฟังก์ชันการทำงานอยู่ 4 ฟังก์ชันด้วยกัน คือ 1. หน้า Application Register 2. หน้า Application Sign in 3. หน้า User Register และ 4. ส่วนสำหรับ User Sign in ซึ่งในหน้าหลักนี้ได้ใส่ URL สำหรับไปยังผู้ใช้บริการหลักที่ทำการลงทะเบียนกับ OAuth provider ไว้แล้วด้วย ดังรูปที่ 4.1



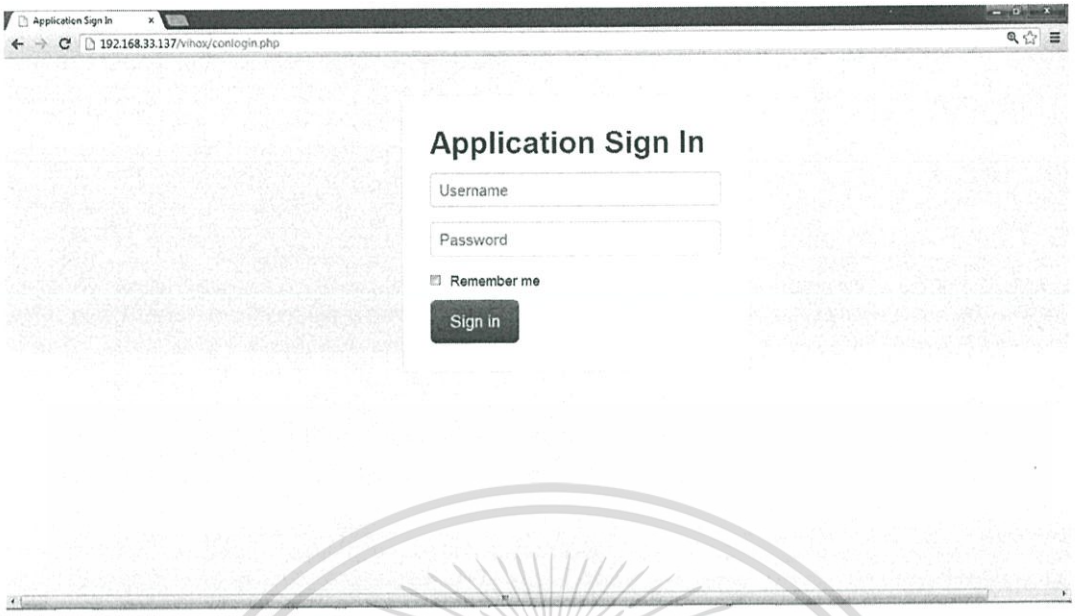
รูปที่ 4.1 แสดงหน้าจอหลักในส่วนของเซิร์ฟเวอร์ (OAuth)

- หน้า Application Register ใช้ในการลงทะเบียนผู้ขอใช้บริการ (Consumer) ซึ่งหลังจากลงทะเบียนเสร็จแล้วจะได้ Consumer Key และ Consumer Secret Key มา Key ทั้ง 2 ตัวนี้ใช้ในการยืนยันว่าเว็บไซต์ได้ทำการลงทะเบียนกับทาง OAuth provider เรียบร้อยแล้ว โดยผู้พัฒนาแอพพลิเคชันจะต้องนำ key ทั้ง 2 ตัวนี้ไปเก็บไว้ที่หน้าเว็บไซต์เพื่อขอใช้บริการในกรณีที่ผู้ใช้เข้าใช้แอพพลิเคชันนั้น ๆ ในการลงทะเบียนทางเซิร์ฟเวอร์จะให้ผู้พัฒนาแอพพลิเคชันทำการเลือกข้อมูลที่ต้องการให้ทางเซิร์ฟเวอร์ส่งมาให้กับแอพพลิเคชันทันที ซึ่งชื่อที่ให้เลือกนี้จะเป็นชื่อตัวแปรที่สามารถนำไปใช้ในการรับค่าเข้ามาใช้ในแอพพลิเคชันได้เลย



รูปที่ 4.2 แสดงหน้าจอที่ใช้ในการลงทะเบียนสำหรับผู้ขอใช้บริการ (Consumer)

- หน้า Application Sign in ใช้ในการเข้าสู่ระบบของผู้ขอใช้บริการเพื่อตรวจสอบ Consumer Key และ Consumer Secret Key มาใช้ในการพิสูจน์ตัวตนของผู้เข้าใช้เว็บไซต์ผ่านทาง OAuth provider



รูปที่ 4.3 แสดงหน้าจอที่ใช้ในการเข้าสู่ระบบสำหรับผู้ขอใช้บริการ (Consumer)

- หน้า User Register ใช้ในการลงทะเบียนของผู้ใช้ ที่ต้องการจะใช้งานเว็บที่ใช้บริการการพิสูจน์ตัวตนด้วยเซิร์ฟเวอร์

รูปที่ 4.4 แสดงหน้าจอลงทะเบียนสำหรับผู้ใช้ (users)

ส่วนที่ใช้ในการพิสูจน์ตัวตนของแอพพลิเคชันผ่านทางเซิร์ฟเวอร์นี้ไม่ได้แสดงออกมาทางหน้าเว็บเพจโดยตรง จะมีแต่ค่าที่ใช้รับ-ส่ง เพื่อยืนยันตัวตนระหว่างผู้ขอใช้บริการกับเซิร์ฟเวอร์

(Consumer key, Consumer secret key, Token) แต่ค่า Token secret, Verify, Access token, Access token secret ที่ใช้ในการพิสูจน์ตัวตนของผู้ใช้และดึงข้อมูลจากเซิร์ฟเวอร์มาใช้นั้น จะไม่ได้แสดงให้เห็น โดยจะเป็นส่วนที่ทำงานอยู่ภายในระบบ

4.4 การพัฒนาระบบการพิสูจน์ตัวตนสำหรับผู้ใช้บริการ (Consumer)

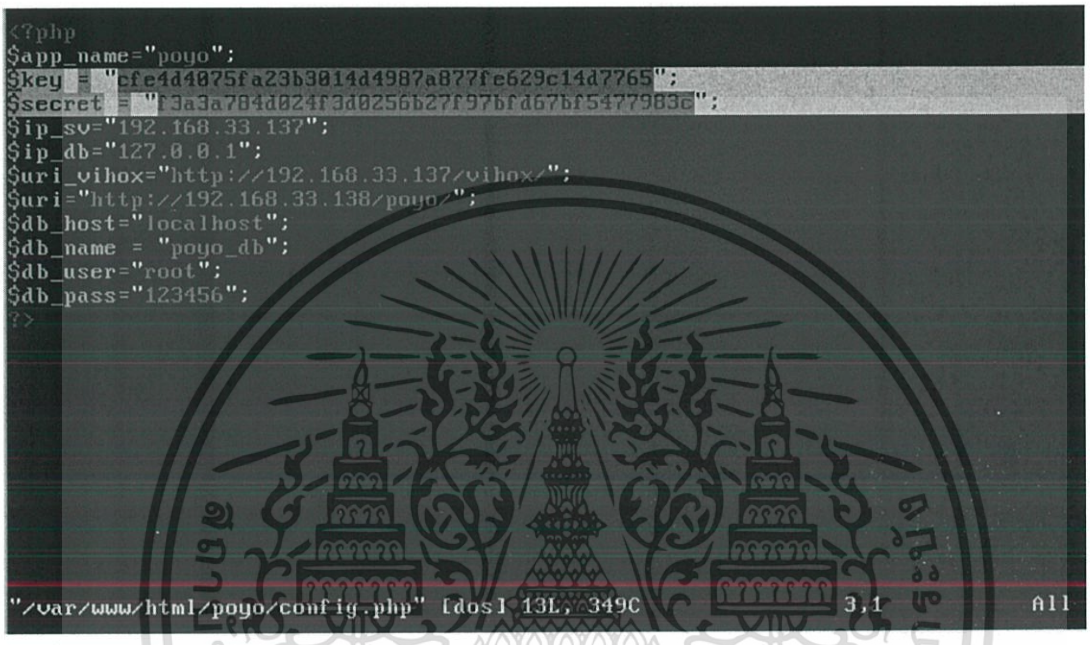
เมื่อผู้ใช้บริการต้องการใช้บริการของเซิร์ฟเวอร์ จะต้องเข้าไปลงทะเบียนในหน้า Application Register ในส่วนของเซิร์ฟเวอร์ก่อน หลังจากลงทะเบียนแล้วจะได้รับ Consumer Key และ Consumer Secret Key มาดังรูปที่ 4.5

รูปที่ 4.5 แสดงหน้าจอที่ผู้ขอใช้บริการจะได้รับหลังจากลงทะเบียนกับเซิร์ฟเวอร์แล้ว

เมื่อแอพพลิเคชันได้ Consumer Key และ Consumer Secret Key มาแล้ว ผู้พัฒนาแอพพลิเคชันจะต้องนำค่า 2 ค่านี้ไปเก็บไว้ส่วนใดส่วนหนึ่งในแอพพลิเคชันเพื่อใช้ในการขอใช้บริการการพิสูจน์ตัวตนจากทางเซิร์ฟเวอร์ ในกรณีที่มีผู้ขอเข้าใช้งานแอพพลิเคชันทุกครั้ง

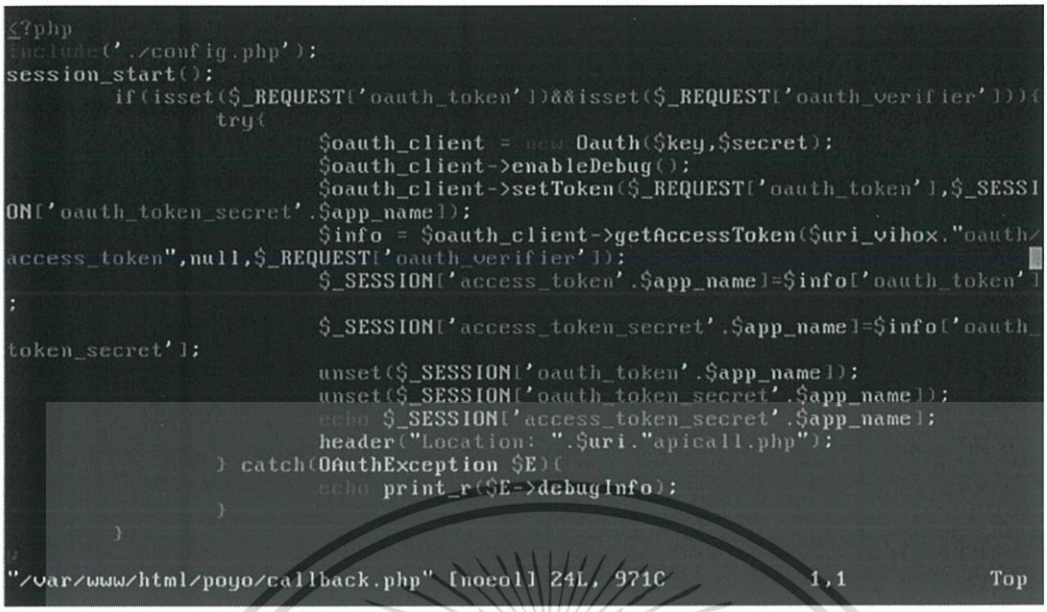
4.5 ตัวอย่างเว็บเพจที่ขอใช้บริการของเซิร์ฟเวอร์ (OAuth)

หลังจากลงทะเบียนกับทางเซิร์ฟเวอร์เรียบร้อยแล้ว ทางผู้พัฒนาแอพพลิเคชันจะต้องทำการเชื่อมต่อแอพพลเคชันกับทางเซิร์ฟเวอร์ เพื่อใช้บริการระบบพิสูจน์ตัวตนของทางเซิร์ฟเวอร์ โดยจะต้องทำการใส่ Consumer Key และ Consumer Secret Key เข้าไปในโปรแกรม ดังรูปที่ 4.6



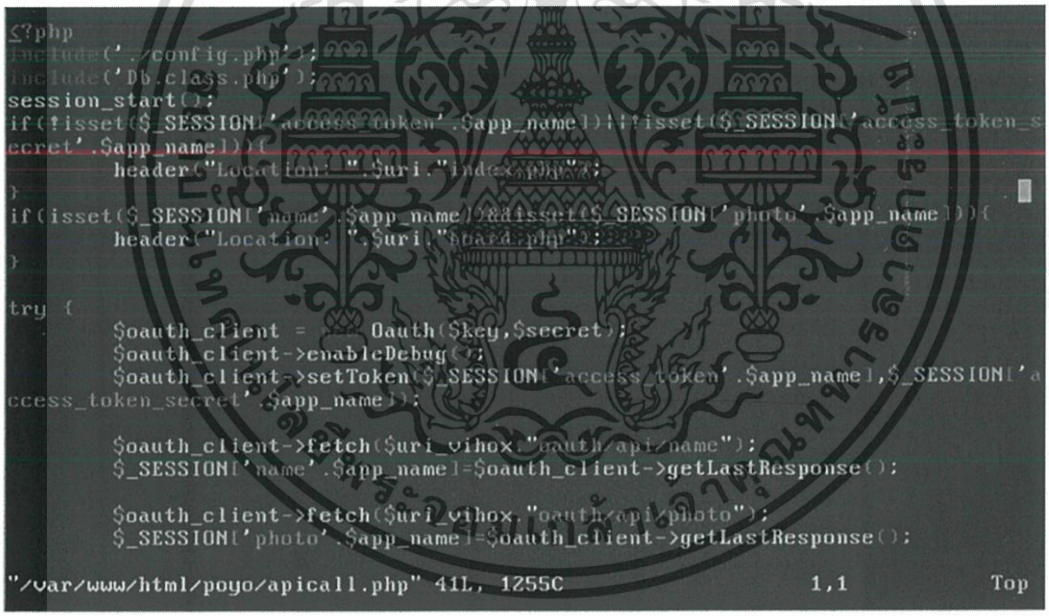
รูปที่ 4.6 แสดงการเพิ่ม Consumer Key และ Consumer Secret Key เข้าไปในส่วนของแอพพลิเคชัน

นอกจากนี้ผู้พัฒนาระบบจะต้องเขียนโปรแกรมเพื่อใช้ในการขอ Token ต่าง ๆ และรับค่าที่ทางเซิร์ฟเวอร์ส่งมาให้ขึ้นมา ซึ่งในโครงงานนี้ชื่อ callback.php กับ apicall.php โดยมีเนื้อหาคร่าว ๆ ดังนี้



```
<?php
include('./config.php');
session_start();
        if(isset($_REQUEST['oauth_token'])&&isset($_REQUEST['oauth_verifier'])){
                try{
                        $oauth_client = new Oauth($key,$secret);
                        $oauth_client->enableDebug();
                        $oauth_client->setToken($_REQUEST['oauth_token'],$_SESSI
ON['oauth_token_secret'.$app_name]);
                        $info = $oauth_client->getAccessToken($uri_vihox."oauth/
access_token",null,$_REQUEST['oauth_verifier']);
                        $_SESSION['access_token'.$app_name]=$info['oauth_token']
;
                        $_SESSION['access_token_secret'.$app_name]=$info['oauth_
token_secret'];
                        unset($_SESSION['oauth_token'.$app_name]);
                        unset($_SESSION['oauth_token_secret'.$app_name]);
                        echo $_SESSION['access_token_secret'.$app_name];
                        header("Location: ".$uri."apicall.php");
                } catch(OAuthException $E){
                        echo print_r($E->debugInfo);
                }
        }
?
"/var/www/html/poyo/callback.php" [nocol] 24L, 971C                1,1           Top
```

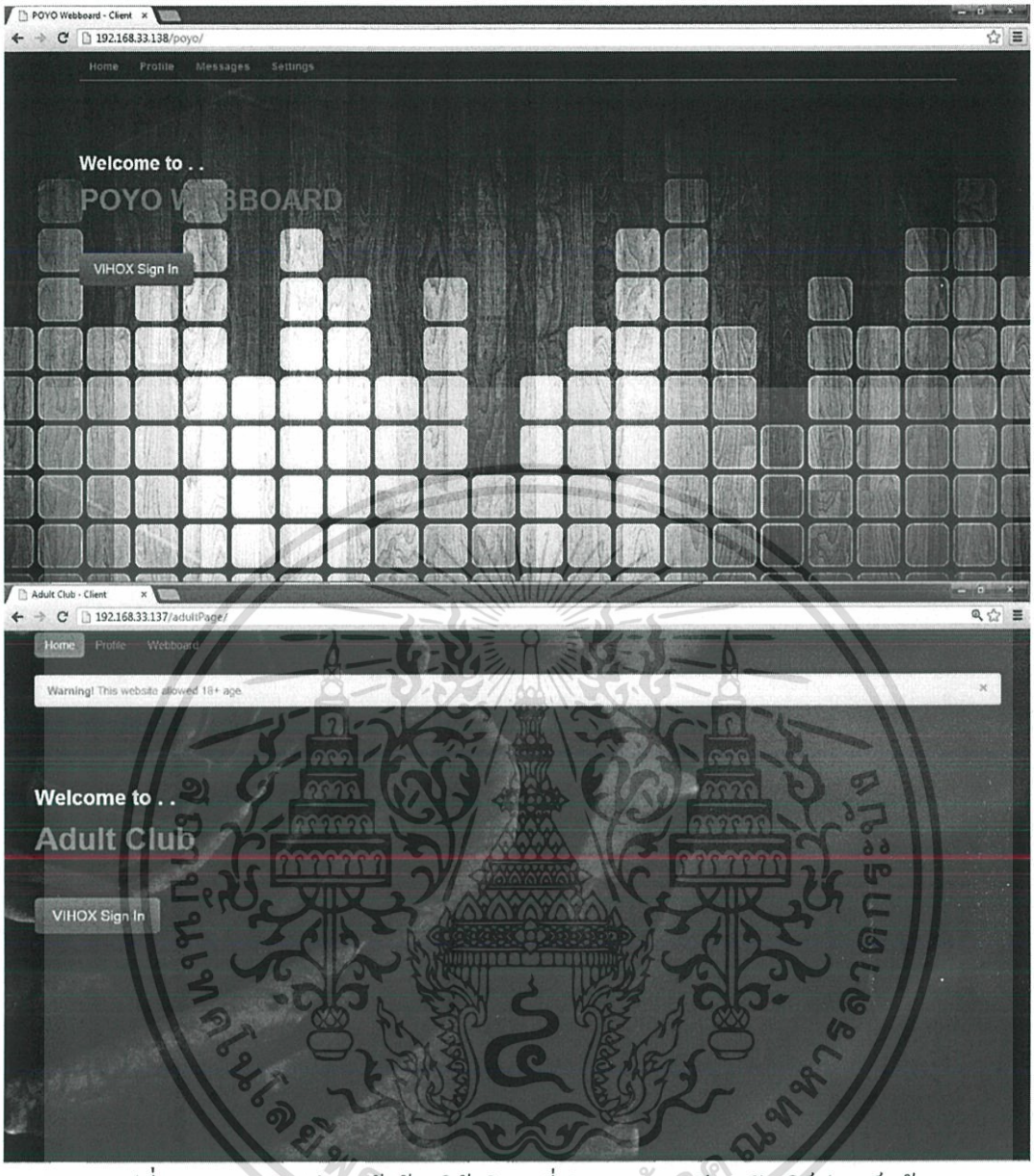
รูปที่ 4.7 แสดงส่วนของ callback.php ที่ใช้ในการขอ token กับทางเซิร์ฟเวอร์



```
<?php
include('./config.php');
include('Db.class.php');
session_start();
if(!isset($_SESSION['access_token'.$app_name])||!isset($_SESSION['access_token_s
ecret'.$app_name])){
        header("Location: ".$uri."index.php");
}
if(isset($_SESSION['name'.$app_name])&&isset($_SESSION['photo'.$app_name])){
        header("Location: ".$uri."board.php");
}

try {
        $oauth_client = new Oauth($key,$secret);
        $oauth_client->enableDebug();
        $oauth_client->setToken($_SESSION['access_token'.$app_name],$_SESSION['a
ccess_token_secret'.$app_name]);

        $oauth_client->fetch($uri_vihox."oauth/api/name");
        $_SESSION['name'.$app_name]=$oauth_client->getLastResponse();

        $oauth_client->fetch($uri_vihox."oauth/api/photo");
        $_SESSION['photo'.$app_name]=$oauth_client->getLastResponse();

"/var/www/html/poyo/apicall.php" 41L, 1255C                1,1           Top
```

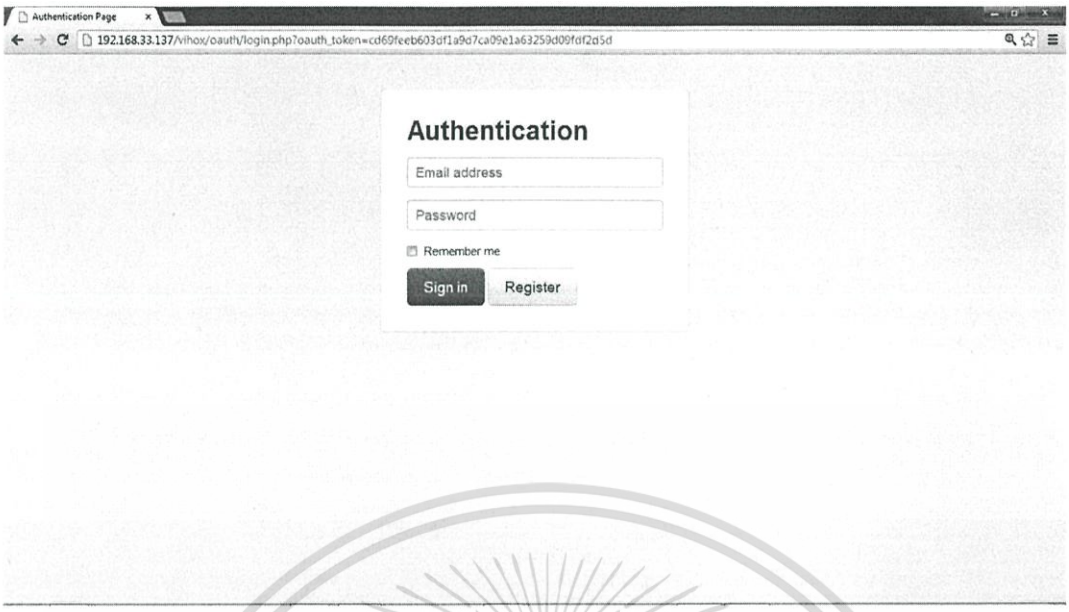
รูปที่ 4.8 แสดงส่วนของ apicall.php ที่ใช้ในการขอข้อมูลจากทางเซิร์ฟเวอร์

เมื่อผู้ใช้เข้าใช้บริการแอพพลิเคชัน จะมีปุ่มให้กด VIHOX Sign in เพื่อใช้ในการยืนยันตัวตนกับทางเว็บ VIHOX ก่อนเข้าใช้งานแอพพลิเคชัน ดังรูปที่ 4.9



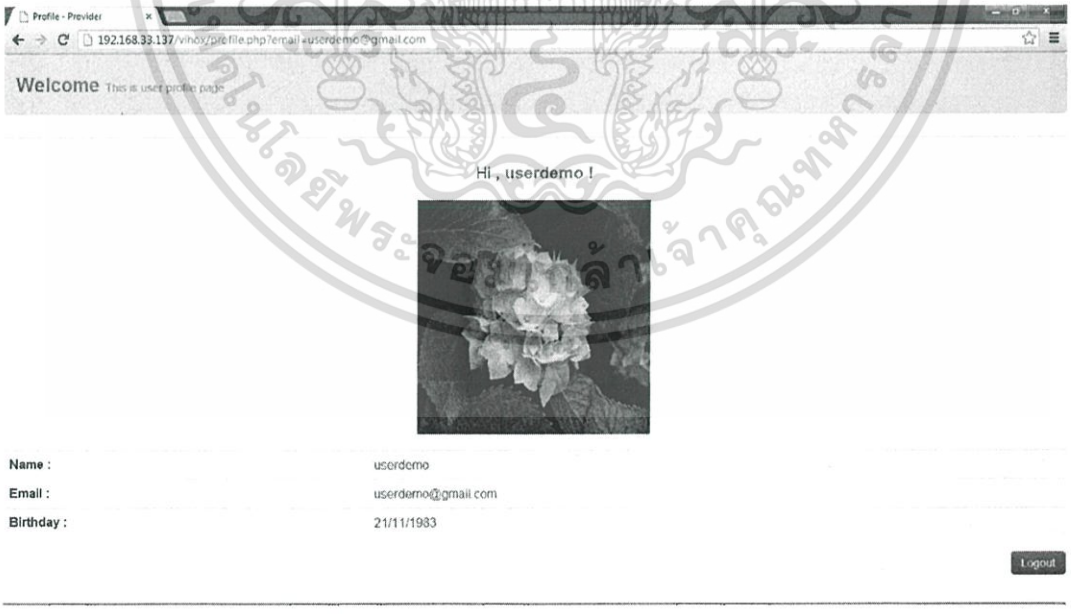
รูปที่ 4.9 แสดงตัวอย่างหน้าผู้ขอใช้บริการที่ทำการลงทะเบียนกับเซิร์ฟเวอร์แล้ว

หลังจากกดปุ่มแล้วแอพพลิเคชันจะทำการติดต่อขอ Token ไปที่เซิร์ฟเวอร์ ซึ่งจะได้ Token, Token secret และ Authentication link มาให้ผู้ใช้ทำการพิสูจน์ตัวตนกับเซิร์ฟเวอร์ โดยใช้ email กับรหัสผ่านที่ได้ทำการลงทะเบียนไว้ ดังรูปที่ 4.10 จะเห็นว่ามี Token ที่ได้รับมาแสดงอยู่ที่ในส่วนท้ายของ URL ของหน้าที่ทำการยืนยันตัวตน



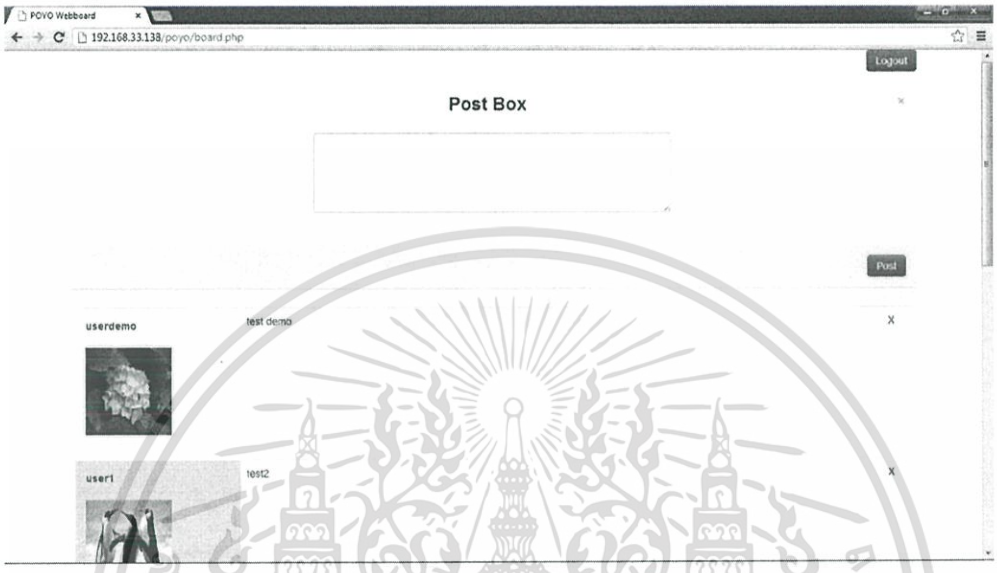
รูปที่ 4.10 แสดงหน้า Authentication สำหรับผู้ใช้ (Users)

หลังจากที่ผู้ใช้ทำการลงทะเบียนกับเซิร์ฟเวอร์แล้ว จะปรากฏหน้าต่างให้ทำการเข้าสู่ระบบ ในส่วนของเซิร์ฟเวอร์ เมื่อเข้าสู่ระบบเรียบร้อยแล้ว ทางเซิร์ฟเวอร์จะแสดงหน้ารายละเอียดของผู้ใช้ ดังรูปที่ 4.11



รูปที่ 4.11 แสดงหน้ารายละเอียดข้อมูลของผู้ใช้

ถ้าผู้ใช้งานทำการเข้าระบบผ่านทางแอพพลิเคชัน จะไม่แสดงหน้ารายละเอียดข้อมูลของผู้ใช้ แต่จะสามารถเข้าดูข้อมูลต่าง ๆ ของเว็บไซต์นั้น ๆ ได้ ดังตัวอย่างเว็บแอพพลิเคชันที่จำลองขึ้นมาเป็นเว็บบอร์ดสำหรับพูดคุยกัน



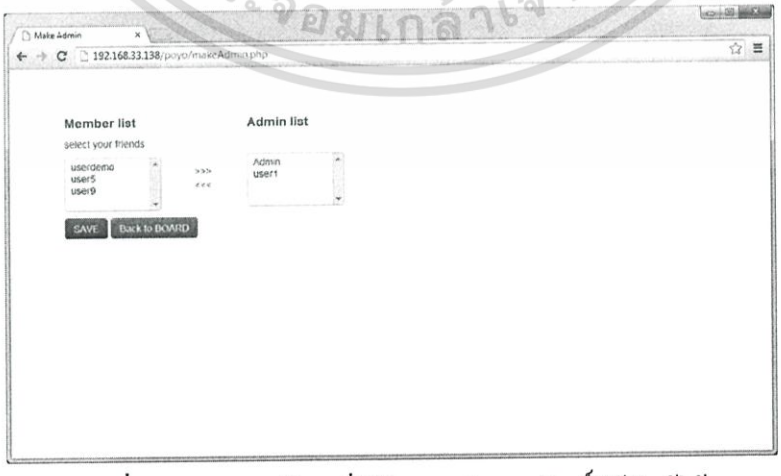
รูปที่ 4.12 แสดงตัวอย่างข้อมูลภายใน POYO เว็บบอร์ดที่ทางเว็บไซต์ได้กำหนดไว้

นอกจากนี้ทางแอพพลิเคชันเองจะมีส่วนที่ใช้ในการกำหนดสิทธิ์ให้กับผู้ใช้ที่จะเข้ามาใช้งานว่ามีสิทธิ์ที่จะสามารถทำอะไรได้บ้าง ในระบบที่จำลองขึ้นมานี้มีการกำหนดสิทธิ์แอดมินให้กับผู้ใช้ ซึ่งหากผู้ใช้ที่เข้ามาเป็นแอดมินจะสามารถจัดการกับหน้าเว็บบอร์ดได้ทั้งหมด แต่ถ้าไม่ใช่จะสามารถจัดการกับหน้าเว็บบอร์ดได้เฉพาะส่วนที่เป็นของตัวเองเท่านั้น ซึ่งการกำหนดสิทธิ์นี้จะสามารถทำได้เฉพาะผู้ใช้ที่มีสิทธิ์เป็นแอดมินอยู่แล้วเท่านั้น จึงจะสามารถเข้าถึงหน้านี้ เพื่อกำหนดสิทธิ์ให้กับผู้ใช้คนอื่นได้

รูปที่ 4.13 แสดงหน้าต่าง POYO เว็บบอร์ดที่เข้าสู่ระบบด้วยสิทธิ์แอดมิน

จะเห็นได้ว่า เมื่อเข้าสู่ระบบด้วยรหัสผู้ใช้ที่มีสิทธิ์เป็นแอดมินจะมีปุ่ม ADMIN MANAGEMENT ปรากฏขึ้นมาเพื่อใช้ในการกำหนดสิทธิ์ให้กับผู้ใช้คนอื่นให้มีสิทธิ์ในการดูแลเว็บแอพพลิเคชันนี้ด้วย

หน้าต่างที่ใช้ในการกำหนดสิทธิ์ให้กับผู้ใช้งาน จะแบ่งออกเป็น 2 ส่วน คือ ส่วนที่แสดงผู้ใช้ทั่วไปที่เข้ามาในเว็บแอพพลิเคชันนี้ และอีกส่วนคือ ผู้ใช้งานที่มีสิทธิ์เป็นแอดมิน เมื่อแอดมินของเว็บแอพพลิเคชันนี้ต้องการที่จะเพิ่มผู้ใช้งานให้มีสิทธิ์ในการแก้ไขข้อมูลของเว็บไซต์ได้ จะทำการเลือกจากลิสของผู้ใช้ทั่วไปและทำการเพิ่มเข้าไปในลิสของกลุ่มแอดมิน จากนั้นทำการยืนยันกับระบบอีกครั้ง ผู้ใช้คนที่ถูกเลือกก็จะมีสิทธิ์ในการจัดการกับเว็บแอพพลิเคชันได้ตามที่เว็บแอพพลิเคชันได้กำหนดไว้ ดังรูปที่ 4.14

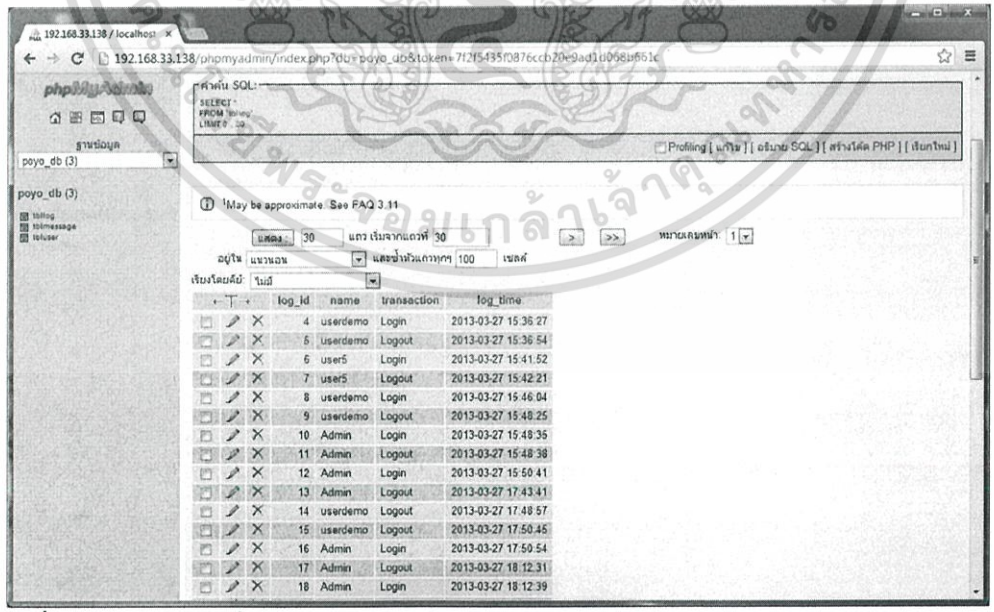


รูปที่ 4.14 แสดงหน้าจอที่ใช้ในการกำหนดสิทธิ์ให้กับผู้ใช้

ส่วนที่ใช้ในเก็บข้อมูลการเข้าใช้งานของผู้ใช้สำหรับแต่ละแอพพลิเคชันจะสามารถดูได้จากฐานข้อมูลในตาราง tbllog โดยจะเก็บชื่อผู้เข้าใช้ สถานะที่เข้าถึง (Login หรือ Logout) และวันที่-เวลาที่ได้เข้าถึงแอพพลิเคชันนั้น ในความเป็นจริงแล้วอาจมีข้อมูลที่จำเป็นสำหรับการเก็บข้อมูลการเข้าใช้อีกมากมาย เพื่อให้ผู้ดูแลระบบเข้ามาตรวจสอบความถูกต้องของการเข้าใช้งานแอพพลิเคชันได้ และยังใช้ในการเก็บสถิติในการเข้าใช้งาน เพื่อใช้เป็นส่วนหนึ่งในการพัฒนาระบบได้อีกด้วย โดยจะมีหลักในการเก็บข้อมูลการเข้าใช้งานดังนี้

เมื่อผู้ใช้ทำการเข้าสู่ระบบด้วยรหัสผ่านของตน แอพพลิเคชันจะทำการขอ token ไปที่เซิร์ฟเวอร์เพื่อให้ทางเซิร์ฟเวอร์ทำการตรวจสอบตัวตนของผู้ใช้ให้ หลังจากเซิร์ฟเวอร์ตรวจสอบพบและส่ง token กลับมาให้กับทางแอพพลิเคชัน แอพพลิเคชันจะขอข้อมูลของผู้ใช้บางส่วนมาใช้ในการแสดงตนบนหน้าเว็บแอพพลิเคชันจากทางเซิร์ฟเวอร์ เมื่อได้ข้อมูลมาแล้ว แอพพลิเคชันจะทำการเก็บข้อมูลเหล่านี้ลงในตาราง tbluser ในฐานข้อมูลของแอพพลิเคชันเองเพื่อใช้ในการจัดการสิทธิ์ต่าง ๆ ให้กับผู้ใช้งาน หลังจากนั้นจะทำการเก็บข้อมูลการใช้งานของผู้ใช้ที่ทำการเข้าสู่ระบบมาเข้าไปในตาราง tbllog โดยจะเก็บชื่อที่ผู้ใช้ได้ทำการลงทะเบียนไว้ เก็บสถานะการเข้าใช้งานเป็น Login และวัน-เวลาที่เข้ามาใช้งาน ลงไปในฐานข้อมูล ก็จะได้ข้อมูลที่ผู้ใช้ทำการขอเข้าใช้บริการ หลังจากนั้น เมื่อผู้ใช้ออกจากระบบด้วยการ Logout จากแอพพลิเคชัน ก็จะมีการเก็บข้อมูลการเข้าใช้ในสถานะ Logout เข้าสู่ฐานข้อมูลอีกครั้งหนึ่ง พร้อมวัน-เวลาที่ทำการออกจากระบบ

ตารางที่ใช้ในการเก็บข้อมูลการเข้าใช้งานของผู้ใช้ทั่วไปที่จำลองขึ้นมา สามารถแสดงได้ดังรูปที่ 4.15



รูปที่ 4.15 แสดงส่วนที่ใช้เก็บข้อมูลการเข้าใช้งานของผู้ใช้ทั่วไปในส่วนของเว็บแอพพลิเคชัน

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 บทสรุป

หลังจากการพัฒนาระบบงานนี้ สามารถสรุปได้ว่า การทำโครงงานนี้สามารถบรรลุตามวัตถุประสงค์ที่ได้นำเสนอไว้ ทำให้ได้ระบบที่ใช้ในการพิสูจน์ตัวตน ซึ่งสามารถยืนยันตัวตนของผู้ใช้งานจากหลาย ๆ แอพพลิเคชันผ่านระบบที่ใช้พิสูจน์ตัวตนเพียงระบบเดียว ช่วยให้ง่ายต่อการพัฒนาระบบ และลดความซ้ำซ้อนของการเก็บรักษาข้อมูลในองค์กรที่มีการใช้แอพพลิเคชันที่หลากหลาย

การทำงานของตัวระบบงานแบ่งออกเป็น 2 ส่วนหลัก ๆ คือ ส่วนของเซิร์ฟเวอร์ที่ใช้ในการพิสูจน์ตัวตนที่มีคุณสมบัติในการเข้าสู่ระบบหลาย ๆ ระบบด้วยรหัสผ่านเพียงชุดเดียวกันที่มีเซิร์ฟเวอร์ในการพิสูจน์ตัวตนเครื่องเดียวกัน โดยใช้หลักการของเทคโนโลยี OAuth ที่มีความปลอดภัยในการเก็บรักษาข้อมูลของผู้ใช้งานเป็นอย่างดี และส่วนของไคลเอนต์ที่จำลองขึ้นมาขอใช้บริการการพิสูจน์ตัวตนจากเซิร์ฟเวอร์ โดยซอฟต์แวร์นี้ สามารถนำไปดำเนินการพัฒนาต่อให้สอดคล้องกับการใช้งานในแต่ละองค์กรได้ตามต้องการอย่างเหมาะสม ซึ่งหากออกแบบฐานข้อมูลที่มีความซับซ้อนมากขึ้น จะทำให้ข้อมูลที่ได้มีความน่าเชื่อถือมากยิ่งขึ้นด้วย

5.2 ข้อเสนอแนะ

สำหรับผู้ที่สนใจจะนำระบบงานนี้ไปพัฒนาต่อไป ผู้พัฒนาเห็นว่าระบบที่ได้พัฒนาขึ้นมานี้ค่อนข้างสมบูรณ์แล้ว สามารถปรับแต่งข้อมูลที่ต้องการใช้จริง แต่ยังมีบางส่วนที่ยังต้องการการพัฒนาให้มีความสมบูรณ์มากยิ่งขึ้นอีก ดังต่อไปนี้

- ระบบนี้ใช้ http:// จึงยังไม่ปลอดภัยเพียงพอสำหรับบางเว็บแอพพลิเคชันที่ต้องการความปลอดภัยสูง ๆ ซึ่งเราสามารถพัฒนาระบบนี้ให้อยู่ในมาตรฐาน https:// ให้มีความปลอดภัยมากยิ่งขึ้นไปอีกได้

- สำหรับฐานข้อมูลที่โครงงานใช้ยังเป็น MySQL ที่เหมาะสำหรับระบบเล็ก ๆ ที่ต้องการการใช้งานและการพัฒนาระบบแบบง่าย แต่องค์กรส่วนมากมักจะใช้ Active Directory ช่วยในการจัดการกับข้อมูลผู้ใช้ เราควรพัฒนาระบบให้สามารถติดต่อกับฐานข้อมูลของ Active Directory

เพื่อให้สามารถพัฒนาระบบและง่ายต่อการจัดการระบบเพราะ Active Directory มีโครงสร้างที่เหมาะสมกับองค์กรขนาดใหญ่ ๆ สำหรับการกำหนดสิทธิ์ต่าง ๆ ที่จำเป็นต่อการควบคุมให้องค์กรขนาดใหญ่สามารถทำงานไปอย่างเป็นระบบได้

- ในการเข้าใช้งานแอพพลิเคชันแต่ละแอพพลิเคชัน ยังจำเป็นต้องทำการยืนยันตัวตนทุกครั้ง ซึ่งจริง ๆ แล้ว ควรจะเข้าแอพพลิเคชันอื่นได้โดยไม่ต้องยืนยันตัวตนซ้ำ แต่เนื่องด้วยการพัฒนา session ในโครงงานยังไม่สมบูรณ์พอที่จะทำให้เป็น Single Sign-On ได้

บรรณานุกรม

สิริพร จิตต์เจริญธรรม, เสาวภา ปานจันทร์ และ เลอศักดิ์ ลิ้มวิวัฒน์กุล. 2554. **ความรู้เบื้องต้นเกี่ยวกับการพิสูจน์ตัวตน**. [Online] Available : http://www.eco.ru.ac.th/MBE/boonkij/group3/Interconnection%20and%20Multicast%20Economics/security.htm

D. Hardt, Ed. 2012. **The OAuth 2.0 Authorization Framework**. [Online] Available : http://tools.ietf.org/html/rfc6749

E. Hammer-Lahav, Ed. 2010. **The Oauth 1.0 Protocol**. [Online] Available : http://tools.ietf.org/html/rfc5849

Kusuma Suthakum. 2554. **Chapter 1 การควบคุมการเข้าถึงข้อมูล**. [Online] Available : http://www.nanacm.com/bcom4102o/chapter1.pdf

Mather, T. et al. 2009. **Cloud Security and Privacy**. CA : O'Reilly Media

Nikhil Vishnu (นามแฝง). 2010. **Understanding OAuth (Open Authorization)**. [Online] Available : http://nikhilvishnu.com/2010/04/05/understanding-oauth-open-authorization/

Oauth (n.p.). [Online] Available : http://oauth.net/

Oauth (n.p.). [Online] Available : http://en.wikipedia.org/wiki/OAuth

Oauth: Valet Key for the Web (n.p.). [Online] Available : http://spring66.com/blogs/?p=723

PHP: Oauth (n.p.). [Online] Available : http://php.net/manual/en/book.oauth.php

Single Sign-On (n.p.). [Online] Available : http://www.opengroup.org/security/sso/