การสร้างและจัดการฟอร์มเอเอสพีดอตเน็ต รุ่น 2

ASP.NET FORMS GENERATOR VERSION 2

นายเจตน์สฤษฎิ์ พลเยี่ยม

นายวิสิฏฐ์ เพชรหนู

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี

พ.ศ. 2554

การสร้างและจัดการฟอร์มเอเอสพีดอตเน็ต รุ่น 2

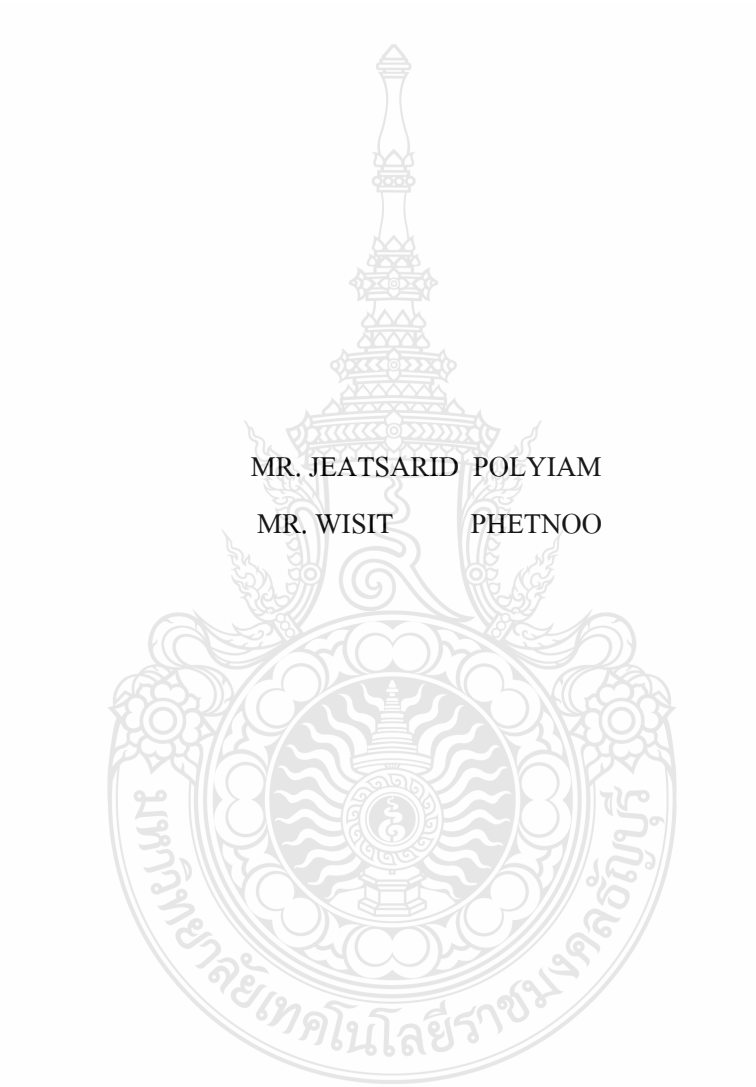
นายเจตน์สฤษฎ์ิ พลเยี่ยม
นายวิสิฎฐ์ เพชรหนู

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี
พ.ศ. 2554

ASP.NET FORMS GENERATOR VERSION 2

MR. JEATSARID POLYIAM

MR. WISIT PHETNOO



THIS PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE BACHELOR DEGREE OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

FACULTY OF ENGINEERING

RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI

YEAR 2011

หัวข้อปริญญานิพนธ์ การสร้างและการจัดการฟอร์มเอเอสพีดอตเน็ต รุ่น 2
นักศึกษา นายเจตน์สฤษฎิ์ พลเยี่ยม
นายวิสิฏฐ์ เพชรหนู
อาจารย์ที่ปรึกษา อาจารย์วีระ คมปรียารัตน์

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี อนุมัติให้ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

……...……หัวหน้าภาควิชาฯ
(อาจารย์มาโนช ประชา)

คณะกรรมการสอบปริญญานิพนธ์

……ประธานกรรมการ
(ดร.กิตติวัณณ์ นิ่มเกิดผล)

……..…….....กรรมการ
(อาจารย์มาโนช ประชา)

……..…….....…..กรรมการ
(อาจารย์วีระชัย แย้มวจี)

……....…กรรมการและอาจารย์ที่ปรึกษา
(อาจารย์วีระ คมปรียารัตน์)

ลิขสิทธิ์ของภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี

หัวข้อปริญญานิพนธ์ การสร้างและการจัดการฟอร์มเอเอสพีดอตเน็ต รุ่น 2
นักศึกษา นายเจตน์สฤษฎิ์ พลเยี่ยม รหัส 115140462028-8
นายวิสิฏฐ์ เพชรหนู รหัส 115140462014-8
อาจารย์ที่ปรึกษา อาจารย์วีระ คมปรียารัตน์
ปีการศึกษา 2554

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นการกล่าวถึง การสร้างและการจัดการฟอร์มเพื่อเชื่อมโยงกับระบบฐานข้อมูลจัดทำขึ้นในรูปแบบของโปรแกรมประเภท Open Source โดยการใช้ภาษา C# ในการสร้างและผลลัพธ์ที่ได้จะอยู่ในรูปแบบเว็บไซต์ ASP.NET การออกแบบโปรแกรมใช้ UML (Unified Modeling Language) ใช้ SMO (SQL Server Management Object) ในการเชื่อมต่อกับฐานข้อมูล และใช้ Regular Expression ในการ Generate ไฟล์

ASP.NET FORMS GENERATOR VERSION 2 จัดทำเพื่อเป็นตัวเลือกเพิ่มเติมอีกตัวเลือกหนึ่งของผู้ใช้ที่ต้องการที่จะสร้างฟอร์มจากฐานข้อมูลของ Microsoft SQL Server เพื่อนำไปใช้งานบนเว็บไซต์ได้ เพื่อก่อให้เกิดความสะดวกรวดเร็ว และประหยัดเวลาในการสร้างฟอร์มในการติดต่อกับฐานข้อมูล

จากการทดลองประสิทธิภาพของ ASP.NET FORMS GENERATOR VERSION 2 สามารถทำงานได้ตรงตามขอบเขต และช่วยอำนวยความสะดวกให้แก่ผู้ใช้ทั่วไปได้อย่างมีประสิทธิภาพ

คำสำคัญ ASP.NET Forms Generator, Microsoft SQL Server , SMO, UML

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้เป็นอย่างดี ด้วยความกรุณาจากอาจารย์ที่ปรึกษาคือ อาจารย์ วีระ คมปรียารัตน์ ที่คอยให้คำแนะนำและให้คำปรึกษาด้วยดีมาตลอดจนโครงงานนี้สำเร็จตามวัตถุประสงค์ได้ขอบพระคุณคณาจารย์ทุกท่านที่ได้กรุณาประสิทธิ์ประศาสน์วิชา ความรู้ในสาขาวิชา วิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยราชมงคลธัญบุรี ที่ได้ให้คำแนะนำและช่วยเหลือด้วยดีเสมอมา ขอบพระคุณสาขาวิชาวิศวกรรมคอมพิวเตอร์ ที่ได้จัดเตรียมสิ่งอำนวยความสะดวกในการพัฒนาโครงงานจนเสร็จสมบูรณ์

ขอบคุณเพื่อนๆ 51446 CPE สำหรับความช่วยเหลือและมิตรภาพที่มีให้แก่ผู้จัดทำเสมอมาตลอดจนบุคคลอื่นๆ ที่ได้ให้ความช่วยเหลือและเป็นกำลังใจสนับสนุนอยู่เบื้องหลังในความสำเร็จครั้งนี้

ท้ายสุดนี้ผู้จัดทำขอมอบคุณความดีของปริญญานิพนธ์ฉบับนี้ แด่ คุณพ่อ คุณแม่ ที่ให้การสนับสนุนและให้โอกาสทางการศึกษาแก่ผู้จัดทำ และหากว่ามีข้อผิดพลาดประการใดอันเกิดจากปริญญานิพนธ์ฉบับนี้ ผู้จัดทำขอน้อมรับและขออภัยเป็นอย่างสูงในความผิดพลาด ณ ที่นี้ด้วย

คณะผู้จัดทำ

สารบัญ

เนื้อหา	หน้า
บทคัดย่อ	ง
กิตติกรรมประกาศ	จ
สารบัญ	ฉ
สารบัญตาราง	ซ
สารบัญรูป	ฌ
บทที่ 1 บทนำ	1
1.1 ความเป็นมาและความสำคัญ	1
1.2 วัตถุประสงค์	1
1.3 ขอบเขต	2
1.4 ทฤษฎีที่จะนำมาใช้	2
1.5 ผลที่คาดว่าจะได้รับ	2
บทที่ 2 ทฤษฎีและหลักการ	3
2.1 งานวิจัยที่เกี่ยวข้อง	3
2.2 ทฤษฎี วิศวกรรมซอฟต์แวร์ (Software Engineering)	7
2.3 ทฤษฎีการพัฒนาระบบเชิงวัตถุด้วย UML (Unified Modeling Language)	21
2.4 ทฤษฎีเว็บแอปพลิเคชั่น (Web Application)	31
2.5 ทฤษฎีการใช้งาน Regular Expression	35
2.6 ทฤษฎีสคีมาฐานข้อมูล (Database schema)	37
2.7 เอแจ็กซ์ (AJAX)	38
บทที่ 3 วิธีการดำเนินงาน	41
3.1 วิเคราะห์การทำงานของ ASP.NET Form Generator Version 1	41
3.2 แผนการดำเนินงาน	58
3.3 การวิเคราะห์ระบบงาน (Use Case Diagrams)	59
3.4 แผนภาพที่ใช้แสดง Class และความสัมพันธ์ ระหว่าง Class (Class Diagrams)	61
3.5 วิเคราะห์พฤติกรรมของระบบ (Sequence Diagrams)	63

สารบัญ (ต่อ)

เนื้อหา	หน้า
3.6 ลำดับกิจกรรมของการทำงาน (Activity Diagrams)	65
3.7 ผังการทำงานของระบบ ASP.NET Forms Generator	68
3.8 ขั้นตอนการดำเนินงาน	68
3.9 การออกแบบหน้าจอระบบ ASP.NET Forms Generator 2	77
บทที่ 4 การทดสอบการใช้งาน	82
4.1 การทดสอบการเชื่อมต่อกับฐานข้อมูลของผู้ใช้	82
4.2 การทดสอบการตรวจสอบการ Update Database Schema	84
4.3 การทดสอบการเลือกตาราง และการเลือกคอลัมน์	85
4.4 การทดสอบการจัดรูปแบบฟอร์มข้อมูล และการเลือกธีม (Theme)	87
4.5 ทดสอบการ Generate ไฟล์ ASP.NET จากฐานข้อมูล	90
4.6 ทดสอบการ เพิ่ม ลบ แก้ไข และ ค้นหาข้อมูล	92
4.7 ทดสอบการทำงานของฟังก์ชั่น AJAX	95
บทที่ 5 สรุปผลของโครงงาน	98
5.1 สรุปผลที่ได้จากโครงงาน	98
5.2 ข้อเสนอแนะในการพัฒนาโครงงาน	98
5.3 อุปสรรคในการทำงาน	99
บรรณานุกรม	100
ภาคผนวก ก	101
วิธีการใช้งาน	101
ภาคผนวก ข	119
Source Code	119
ประวัติผู้จัดทำปริญญานิพนธ์	120

สารบัญตาราง

ตารางที่		หน้า
2.1	สัญลักษณ์ของ Regular Expression	36
3.1	แผนการดำเนินงานของโครงงาน	58
4.1	แสดงรายละเอียดการเชื่อมต่อกับฐานข้อมูลด้วย SQL Authentication Mode	83
4.2	แสดงรายละเอียดการเชื่อมต่อกับฐานข้อมูลด้วย Windows Authentication Mode	83
4.3	แสดงผลการทดสอบการเลือกตาราง	86
4.4	แสดงผลการทดสอบการเลือกคอลัมน์	87
4.5	แสดงรายละเอียดการทดสอบการเลือกธีม	89
4.6	ผลการทดสอบ Generate ไฟล์	90

สารบัญรูป

ภาพที่		หน้า
2.1	การนำ Analysis Model มาใช้ในการออกแบบ	7
2.2	วงจรการพัฒนาระบบ	14
2.3	Requirements Gathering	15
2.4	Application Analysis	16
2.5	การออกแบบโมเดล	17
2.6	Coding / Test and Implement	18
2.7	การออกแบบตามแนวทาง Data Oriented	19
2.8	การออกแบบตามแนวทาง Process Oriented	20
2.9	แสดงสัญลักษณ์ที่ใช้แทน Actor และ Use Case	22
2.10	แสดงลักษณะการเขียน Use Case Diagrams	22
2.11	แสดงขั้นตอนการทำงานของ Use Case Diagrams	23
2.12	แสดงการเกิด Use Case Diagrams	24
2.13	สัญลักษณ์การใช้ Inclusion และ Extension	25
2.14	แสดงสัญลักษณ์การกำหนด Class	26
2.15	การเขียน Attribute ใน Class	27
2.16	การระบุพารามิเตอร์และประเภทของข้อมูลให้ Operation	27
2.17	แสดงการสร้างข้อมูลระดับ Constrains เพิ่มเติมให้ Class	28
2.18	ตัวอย่างการเขียนหมายเหตุให้แก่ Class	29
2.19	แสดงโครงสร้างของ Sequence Diagrams	29
2.20	แสดงองค์ประกอบทั้งหมดเป็นสัญลักษณ์	30
2.21	สัญลักษณ์แสดงรูปการติดตั้งทั้ง 3 แบบของ Message	30
2.22	แสดงลักษณะของการแสดงเวลาของ Sequence Diagrams	31
2.23	โครงสร้างสกีมาฐานข้อมูล	38
2.24	เปรียบเทียบการทำงานแบบเดิม กับ AJAX	39
2.25	แสดงโครงสร้างของ AJAX	40
3.1	Use Case Diagrams ของระบบ ASP.NET Forms Generator	41

สารบัญรูป (ต่อ)

ภาพที่		หน้า
3.2	แสดงคลาสที่ใช้ในการเก็บข้อมูลของระบบ ASP.NET Forms Generator	43
3.3	แสดงคลาสที่ใช้ในการ Generate	44
3.4	Sequence Diagrams ของ Use Case รวมของระบบ ASP.NET Forms Generator	45
3.5	Activity Diagrams ของ Use Case รวมของระบบ ASP.NET Forms Generator	46
3.6	Activity Diagrams ของการแก้ไขโปรเจกต์เดิม	47
3.7	ผังการทำงานของระบบโดยรวม	48
3.8	วิธีการออกแบบหน้าจอการเชื่อมต่อฐานข้อมูลด้วย SMO	50
3.9	Code Program ในการเชื่อมต่อกับฐานข้อมูลด้วย SMO	50
3.10	หน้าจอและตัวอย่าง Code Program ในการเลือกตาราง	51
3.11	การเพิ่มข้อมูลลง Dataset แล้วแสดงในดาต้ากริดวิว	51
3.12	การ Replace สีพื้นหลังจากการเลือกของผู้ใช้งาน	52
3.13	Code ASP.NET ที่ใช้ในการออกแบบเทมเพลต	52
3.14	การออกแบบเทมเพลตจาก Code ASP.NET	53
3.15	วิธีการ Generate ไฟล์ และเขียนไฟล์ไปยังไดเร็กทอรี่	54
3.16	การออกแบบการเชื่อมต่อฐานข้อมูล และเลือกตาราง	55
3.17	การออกแบบหน้าจอแสดงคอลัมน์และการเลือกคอลัมน์ของตารางที่เลือกไว้	56
3.18	การออกแบบหน้าจอการแก้ไข Header, Footer และเลือกพื้นหลังของ Master -Page	56
3.19	การออกแบบหน้าตัวอย่างของเว็บไซต์ก่อนการ Generate	57
3.20	การออกแบบของการ Generate	57
3.21	Use Case Diagrams ของระบบ Forms Generator ASP.NET Version 2	59
3.22	Use Case Diagrams แสดงการตรวจสอบการ Update Database Schema	60
3.23	คลาสที่ใช้ในการเก็บข้อมูลของระบบ ASP.NET Forms Generator 2	61

สารบัญรูป (ต่อ)

ภาพที่		หน้า
3.24	คลาสที่ใช้ในการ Generate	62
3.25	Sequence Diagrams ของ Use Case รวมของระบบ ASP.NET Forms Generator 2	63
3.26	Sequence Diagrams ของ Use Case การตรวจสอบการ Update Database Schema	64
3.27	Activity Diagram ของ Use Case รวมของระบบ Forms Generator ASP.NET Version 2	65
3.28	Activity Diagrams ของ Use Case การตรวจสอบการ Update Database Schema	66
3.29	Activity Diagrams การตรวจสอบโครงสร้างฐานข้อมูล	67
3.30	ผังการทำงานของระบบโดยรวม	68
3.31	วิธีการออกแบบหน้าจอการเชื่อมต่อฐานข้อมูลด้วย SMO	70
3.32	ตัวอย่าง Code Program ในการเชื่อมต่อกับฐานข้อมูลด้วย SMO	70
3.33	ตัวอย่าง Code Program การตรวจสอบการ Update Schema	70
3.34	หน้าจอและตัวอย่าง Code Program ในการเลือกตาราง	72
3.35	การเพิ่มข้อมูลลง Dataset แล้วแสดงในดาต้ากริดวิว	72
3.36	ตัวอย่าง Code Program ในการจัดการรูปแบบฟอร์ม	73
3.37	แสดงการ Replace สีพื้นหลังจากการเลือกของผู้ใช้งาน	74
3.38	Code ASP.NET ที่ใช้ในการออกแบบเทมเพลต	74
3.39	การออกแบบเทมเพลตจาก Code ASP.NET	75
3.40	วิธีการ Generate ไฟล์ และเขียนไฟล์ไปยังไดเร็กทอรี่	76
3.41	การออกแบบการเชื่อมต่อฐานข้อมูล และเลือกตาราง	77
3.42	การออกแบบหน้าจอแสดงคอลัมน์และการเลือกคอลัมน์ของตารางที่เลือกไว้	78
3.43	การออกแบบหน้าจอการจัดรูปแบบฟอร์มของผู้ใช้	79
3.44	การออกแบบหน้าจอการแก้ไข Header, Footer และเลือกพื้นหลังของ Master Page	79

สารบัญรูป (ต่อ)

ภาพที่		หน้า
3.45	การออกแบบหน้าจอเพิ่มจากโครงงานเดิมสามารถใส่ Slide ,Background Image	80
3.46	การออกแบบหน้าตัวอย่างของเว็บไซต์ก่อนการ Generate	80
3.47	การออกแบบของการ Generate	81
4.1	แสดงผลการเชื่อมต่อเมื่อเชื่อมต่อสำเร็จ	83
4.2	แสดงผลการเชื่อมต่อ เมื่อเชื่อมไม่ต่อสำเร็จ	83
4.3	แสดงข้อความแจ้งเตือน เมื่อมีการเปลี่ยนฐานข้อมูล	84
4.4	แสดงข้อความแจ้งเตือน เมื่อมีตารางและคอลัมน์ใหม่เพิ่มขึ้นมา	85
4.5	แสดงข้อความแจ้งเตือน เมื่อมีคุณสมบัติของคอลัมน์มีการเปลี่ยนแปลง	85
4.6	แสดงข้อความแจ้งเตือน เมื่อมีตารางหรือคอลัมน์ถูกลบออกจากฐานข้อมูล	86
4.7	แสดงเว็บไซต์เมื่อ Generate จากฐานข้อมูล espcar และเลือก 4 ตาราง	87
4.8	แสดงการจัดรูปแบบฟอร์ม	88
4.9	แสดงผลการจัดรูปแบบฟอร์ม	88
4.10	เมื่อผู้ใช้เลือกพื้นหลังเป็นรูป ใช้ Image SlideShow และใช้รูปเป็น Header	89
4.11	แสดงข้อความเมื่อ Generate Source Code สำเร็จ	90
4.12	แสดงข้อความเมื่อ Generate DBML ไม่สำเร็จ	91
4.13	แสดงเว็บไซต์เมื่อผู้ใช้เลือก Generate ฐานข้อมูล espcar	91
4.14	แสดงการเพิ่มข้อมูล ลงฐานข้อมูล espcar	92
4.15	แสดงการแก้ไขข้อมูล ในฐานข้อมูล espcar	3
4.16	แสดงการลบข้อมูลในฐานข้อมูล espcar	93
4.17	แสดงข้อความแจ้งเตือนเมื่อลบข้อมูล เสร็จเรียบร้อย	94
4.18	แสดงข้อมูลทั้งหมดก่อนการค้นหา	94
4.19	แสดงข้อมูลเมื่อค้นพบข้อมูล	95
4.20	แสดงผลการทำงานของฟังก์ชั่น AJAX SlideShow	95

สารบัญรูป (ต่อ)

ภาพที่		หน้า
4.21	แสดงผลการทำงานของฟังก์ชั่น AJAX Always Visible Control	96
4.22	แสดงผลการทำงานของฟังก์ชั่น AJAX Animation	96
4.23	แสดงผลการทำงานของฟังก์ชั่น AJAX Color Picker	97

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

เนื่องจาก ASP.NET FORMS GENERATER ได้มาช่วยอำนวยความสะดวกแก่ผู้ที่จะสร้างเว็บฟอร์ม เพื่อเชื่อมโยงกับระบบฐานข้อมูลได้เป็นอย่างดีในระดับหนึ่ง แต่ถ้าหากผู้ใช้มีการปรับเปลี่ยนโครงสร้างฐานข้อมูล โปรแกรมจะไม่สามารถอัพเดทข้อมูลได้ ต้องทำการสร้างฟอร์มใหม่โดยเริ่มต้นตั้งแต่ต้น ทำให้เกิดการทำงานซ้ำซ้อน ซึ่งทำให้เสียเวลามาก การสร้างฟอร์มสามารถจัดรูปแบบฟอร์มให้มีความสวยงาม และยังไม่สามารถอำนวยความสะดวกได้เท่าที่ควร

1.2 วัตถุประสงค์

จากปัญหาของ ASP.NET FORMS GENERATER ที่ได้กล่าวมาข้างต้นจะเห็นได้ว่าปัญหาที่สำคัญ คือ เมื่อมีการปรับเปลี่ยนโครงสร้างฐานข้อมูล (Update Database Schema) จะต้องมาเริ่มทำการสร้างฟอร์มใหม่ทุกครั้ง ดังนั้นจึงมีแนวคิดในการสร้าง ASP.NET FORMS GENERATER VERSION 2 ที่จะนำเอาไฟล์โปรเจค ที่ได้ทำการบันทึกไว้มาใช้ประโยชน์โดยการให้โปรแกรมอ่านไฟล์ขึ้นมาพร้อมกับตรวจสอบว่าในฐานข้อมูลที่ใช้ มีการปรับเปลี่ยนโครงสร้างฐานข้อมูล หรือไม่ ถ้ามีก็จะแสดงโครงสร้างฐานข้อมูลอันใหม่ขึ้นมา และยังคงรูปแบบของรูปแบบฟอร์มเดิมไว้ ดังนั้นจึงสามารถแก้ไข้รูปแบบฟอร์มต่อจากเดิมได้ โดยไม่ต้องเริ่มต้นใหม่ นอกจากนั้น ผู้ใช้ยังสามารถจัดรูปแบบฟอร์มข้อมูลได้ โดยจะจัดตำแหน่งของ ข้อความและคอนโทรล ที่จะแสดงตอนการเพิ่มข้อมูล และการแก้ไขข้อมูล และยังได้นำเอาความสามารถของ AJAX (Asynchronous JavaScript and XML) มาช่วยทำให้หน้าเว็บฟอร์มมีความสวยงาม และอำนวยความสะดวกในการใช้งานมากกว่าเดิม

1.3 ขอบเขต

1.3.1 สามารถตรวจสอบการเปลี่ยนแปลง Database Schema และทำการสร้างฟอร์มเฉพาะส่วนที่มีการแก้ไข Database

1.3.2 สามารถสร้างฟอร์มจากฐานข้อมูลของ Microsoft SQL Server 2005 – 2008 ได้ แต่ไม่สามารถสร้างฟอร์มจากฐานข้อมูลตัวอื่นได้เช่น Access, MySQL, Oracle

1.3.3 สามารถจัดรูปแบบฟอร์มข้อมูลโดยผู้ใช้ได้

1.3.4 ผู้ใช้สามารถเลือกธีมโดยมีธีมอย่างน้อย 5 ธีมและสามารถเพิ่ม Header Footer ได้ และได้นำเอาฟังก์ชันของ AJAX เช่น Slide Show, Always Visible Control, Color Picker, Animation มาใส่ในธีมเพื่อเพิ่มความสวยงามและอำนวยความสะดวกแก่การใช้งานของผู้ใช้ด้วย

1.3.5 ผู้ใช้สามารถสร้างและจัดการแบบฟอร์มได้อัตโนมัติ

1.3.6 มีการเชื่อมต่อทุกจำนวนหน้าทั้งฐานข้อมูล

1.3.7 คอนโทรลที่สามารถทำได้คือ Label, TextBox, DropdownList, ListBox, RadioButton

1.3.8 สามารถเลือก Table View และ Column ในการ Generate ได้

1.3.9 สามารถเลือกคอลัมน์ในการค้นหาได้

1.3.10 การ Generate จะออกมาเป็น LINQ to SQL ASP.NET Framework Version 3.5 ซึ่งประกอบด้วย Insert, Update, Delete และ Views

1.3.11 สามารถทำการบันทึกเป็นไฟล์โปรเจคไว้กลับมาแก้ไขใหม่ได้

1.3.12 Code Behind File เป็นภาษา C#

1.4 ทฤษฎีที่จะนำมาใช้

1.4.1 ทฤษฎีวิศวกรรมซอฟแวร์ (Software Engineer)

1.4.2 ทฤษฎีการวิเคราะห์และออกแบบระบบ (System Analysis and Design)

1.4.3 ทฤษฎีการออกแบบฐานข้อมูล (Database Design)

1.4.4 ทฤษฎีการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming)

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 Forms Generator Version 2 สามารถสร้างเว็บไซต์ที่ติดต่อกับฐานข้อมูลได้โดยไม่ต้องทำการเขียนโปรแกรมด้วยตนเอง

1.5.2 สามารถสร้างเว็บไซต์ เพื่อนำมาประยุกต์ใช้งานในหน่วยงานของตนเองได้

บทที่ 2
ทฤษฎีและงานวิจัยเกี่ยวข้อง

ระบบงานในรูปแบบ Windows Application ซึ่งถูกพัฒนาโดยไมโครซอฟท์ วิชวลสตูดิโอ (Microsoft Visual Studio) ร่วมกับไมโครซอฟท์เอสคิวแอลเซิร์ฟเวอร์ (Microsoft SQL Server) ซึ่ง “Structured Query Language” จะถูกใช้เป็นตัวจัดการฐานข้อมูลทั้งหมดของการสร้างและจัดการฟอร์ม ASP.NET ทั้งหน้า Insert, Update, Delete, View โดยเนื้อหาที่เกี่ยวข้องกับโครงงานในเรื่องต่างๆมีรายละเอียดดังนี้ ทฤษฎีวิศวกรรมซอร์ฟแวร์ ทฤษฎีการพัฒนาระบบด้วย UML ทฤษฎีเว็บแอปพลิเคชั่น ทฤษฎีการใช้งาน Regular Expression ทฤษฎีสกีมาฐานข้อมูล (Database Schema) และหลักการทำงานเอแจ็กซ์ (AJAX : Asynchronous JavaScript and XML)

2.1 งานวิจัยที่เกี่ยวข้อง

2.1.1 การสร้างและจัดการฟอร์มเอเอสพีดอตเน็ต (ASP.NET FORMS GENERATOR) นายประกาสิทธิ์ ลาศรี นายพิเชษฐ สีสม 2553 [1]

โครงงานนี้มีแนวคิดที่จะจัดสร้างโปรแกรมที่มีความสามารถที่จะสร้างฟอร์ม โดยเมื่อมีการเจนเนอเรทออกมาแล้ว อยู่ในรูปแบบ ASP.NET ที่มี Behide Code เป็นภาษา C# เพื่อเชื่อมโยงกับระบบฐานข้อมูลในรูปแบบของโปรแกรมประเภท Open Source ขึ้นมา เพื่อเป็นตัวเลือกเพิ่มเติมอีกตัวเลือกหนึ่งของผู้ใช้ที่ต้องการที่จะสร้างฟอร์มเชื่อมโยงกับระบบฐานข้อมูลแล้วนำไปใช้งานบนเว็บไซต์ได้ สามารถใช้ ASP.NET Forms Generator ที่กลุ่มของผู้จัดทำได้สร้างขึ้นมาได้เป็นการอำนวยความสะดวกแก่ผู้ใช้ที่ต้องการนำไปใช้งานและผู้ที่มีความสนใจที่จะพัฒนาให้มีความสามารถมากกว่าเดิมได้

ขั้นตอนการทำงานของการสร้างและจัดการฟอร์มเอเอสพีดอตเน็ต (ASP.NET FORMS GENERATOR)

1) แบ่งระบบออกเป็นโมดูลย่อยๆ ดังนี้
   - ขั้นตอนการเชื่อมต่อฐานข้อมูล
   - ขั้นตอนการเลือกตาราง
   - ขั้นตอนการเลือกและแก้ไขรายละเอียดคอลัมน์ (Column)
   - ขั้นตอนการเลือกและแก้ไขธีม (Theme)
   - ขั้นตอนการ Generate

2) ศึกษาโปรแกรมและภาษาที่ต้องในการจัดทำระบบดังนี้

- โปรแกรม Microsoft Visual Studio ศึกษาพื้นฐานของการเขียนโปรแกรมด้วย Windows Form Application จากเครื่องมือต่างๆ และทดสอบการใช้เครื่องมือใน Visual Studio
- ภาษา C# การสร้าง Windows Form Applications การสร้าง Class ในการเก็บข้อมูลด้วย Dataset
- ภาษา ASP.NET 3.5 การสร้างเว็บไซต์ในการติดต่อกับฐานข้อมูล Insert Update Delete การ Validate Data Type การใช้งานเมนูใน Master Page
- การติดต่อฐานข้อมูลด้วย SMO (SQL Server Management Object) การออกแบบ การสร้างและจัดการฟอร์ม ASP.NET เมื่อเชื่อมต่อกับฐานข้อมูลแล้วจะต้องทำการ Query Schema ทั้งหมดของฐานข้อมูลเพื่อนำมาใช้ในการสร้างเว็บไซต์ ดังนั้นจึงต้องเชื่อมต่อกับฐานข้อมูลด้วย SMO
- ระบบฐานข้อมูล Microsoft SQL Server เนื่องจากการสร้างและจัดการฟอร์ม ASP.NET สร้างเว็บไซต์จาก Microsoft SQL Server ดังนั้นต้องเข้าใจการทำงานและคุณสมบัติของ Microsoft SQL Server
- การใช้งาน SQLMetal ในการสร้าง LINQ to SQL Classes จะต้องมีการสร้าง Class Database เพื่อเชื่อมต่อกับฐานข้อมูลเป็นตัวการจัดการ Insert Update Delete และ View โดยใช้ SQLMetal.exe ไฟล์ที่ได้ออกมาจะเป็นไฟล์นามสกุล .dbml และคลาส .designer.cs

3) เขียนโปรแกรม

- เขียนโปรแกรมในส่วนการเชื่อมต่อกับฐานข้อมูลด้วย SMO
- เขียนโปรแกรมในส่วนของการเลือกตารางโดยใช้ ListBox
- เขียนโปรแกรม Query ข้อมูลจากฐานข้อมูลลงใน Dataset แล้วทำการแสดงผลผ่านดาต้ากริดวิวโดย Query ข้อมูลจาก Class Dataset
- เขียนโปรแกรมในการเลือกธีมโดยสร้าง เทมเพลตไว้เป็น Default เมื่อผู้ใช้เลือกสีพื้นหลัง, รูปภาพ และลักษณะของกริดวิว ก็ทำการ Replace ค่าต่างลงใน Skin File
- ออกแบบเทมเพลตโดยสร้างเทมเพลตจาก Code ASP.NET

การออกแบบเทมเพลต [:Name:] คือตัวแปรที่จะทำการแทนที่โดยใช้ Regular Expression ค้นหาคำและ แทนที่คำที่ต้องการเพื่อให้ถูกต้องตามรูปแบบของเว็บไซต์ ASP.NET โดยสร้างเทมเพลตไว้ดังนี้

TemplateView.txt

TemplateInsert.txt

TemplateUpdate.txt

TemplateDelete.txt

TemplateMasterPage.txt

TemplateGridviews.txt

TemplateDetailViews.txt

TemplateSkinfile.txt

- เขียนโปรแกรมในการ Generate โดยการอ่านไฟล์จาก Template ที่จัดทำไว้โดยใช้ Regular Expression ในการค้นหาตัวแปรที่เรากำหนดไว้ เช่น [:label:] เมื่อเจอตัวแปรก็ทำการแทนค่าใหม่เข้าไปเพื่อให้ถูกต้องตามรูปแบบของ ASP.NET โดยจะสร้างไฟล์เทมเพลตไว้

ผลที่ได้จากโครงงาน

จากโครงงานที่ได้จัดการขึ้น เมื่อผู้ใช้มีฐานข้อมูลที่ถูกต้อง ระบบการสร้างและการจัดการฟอร์ม ASP.NET สามารถทำการสร้างและจัดการฟอร์มได้ดังนี้ สามารถสร้างเว็บไซต์ ASP.NET ติดต่อกับฐานข้อมูล Microsoft SQL Server 2005 ประกอบด้วยหน้า Insert Update Delete และ View ได้ สามารถค้นหาข้อมูลในหน้า View ได้ สามารถเลือกธีมได้หลากหลาย และเว็บไซต์ที่สร้างขึ้นมาสามารถใช้งานได้จริง

แนวทางการพัฒนาต่อ

การสร้างและจัดการฟอร์มควรมีความสามารถในการสร้างเว็บไซต์ได้มากขึ้นคือ

- สร้างเว็บไซต์จากฐานข้อมูล MSSQL Server, Microsoft Access, Oracle และ MySQL
- เลือกสร้างเว็บไซต์ได้ทั้ง ASP, PHP และ JAVA
- สร้าง Report จากฐานข้อมูลได้
- สร้าง Form จาก Store Procedure หรือ Function ของ Microsoft SQL Server มีการ Validate ข้อมูลจากฐานข้อมูล เช่น Compute Column เป็นต้น

2.1.2 การสร้างและการจัดการฟอร์ม (FORMS GENERATOR) นายพิทยา โพธิ์ชะคุ้ม นางสาวศิรินภา ระวิพานิช นางสาวสุภรางศุ์ จินะใจ 2551 [2]

แนวคิดของโครงงานคือจัดสร้างโปรแกรมที่มีความสามารถที่จะสร้างฟอร์ม โดยที่ไฟล์ที่เจนเนอเรทได้จะเป็นเป็นภาษา PHP เพื่อเชื่อมโยงกับระบบฐานข้อมูลในรูปแบบของโปรแกรมประเภท Open Source ขึ้นมา เพื่อเป็นตัวเลือกเพิ่มเติมอีกตัวเลือกหนึ่งของผู้ใช้ที่ต้องการที่จะสร้างฟอร์ม เพื่อเชื่อมโยงกับระบบฐานข้อมูลเพื่อนำไปใช้บนเว็บไซต์ได้ สามารถใช้งานโปรแกรมสร้างฟอร์มเพื่อเชื่อมโยงกับระบบฐานข้อมูลในรูปแบบประเภท Open Source ที่กลุ่มของผู้จัดทำได้สร้างขึ้นมาได้เป็นการอำนวยความสะดวกแก่ผู้ใช้ที่ต้องการนำไปใช้งานและผู้ที่มีความสนใจ

ขั้นตอนการดำเนินงานของการสร้างและการจัดการฟอร์ม

1) แบ่งระบบออกเป็นโมดูลย่อยๆ ดังนี้
   - ขั้นตอนการเชื่อมต่อฐานข้อมูล
   - ขั้นตอนการเลือกตาราง
   - ขั้นตอนการเลือกฟิลด์
   - ขั้นตอนการแก้ไขฟิลด์ เพิ่ม Header, Footer และเลือก Theme
   - ขั้นตอนการ Generate
2) ศึกษาโปรแกรมและภาษาที่ต้องในการจัดทำระบบดังนี้
   - โปรแกรม Microsoft Visual Studio 2008
   - ภาษา VC#
   - ภาษา PHP
3) ลงมือเขียนโปรแกรม

สรุปผลที่ได้จากโครงงาน

จากโครงงานที่ได้จัดการขึ้น เมื่อผู้ใช้มีฐานข้อมูลที่ถูกต้อง ระบบสร้างและการจัดการฟอร์มสามารถทำการสร้างและจัดการฟอร์มได้

แนวทางการพัฒนาต่อ

จากระบบได้ทำการกำหนดจำนวนของตารางไว้ 4 ตาราง ควรเพิ่มการเลือกจำนวนตารางการใช้งานขึ้น เนื่องจากกลุ่มคนหรือองค์กรต้องการใช้งานมากกว่า 4 ตาราง และการใช้งานของระบบยังมีความยุ่งยากอยู่ และควรพัฒนาในส่วนของ File ที่สร้างออกมาให้ผู้ใช้งาน ให้มีการใช้งานที่ง่ายขึ้นและมีการบอกรายละเอียดส่วนต่างๆ ของ Source Code สำหรับในการปรับแต่งแก้ไขของผู้ใช้งาน

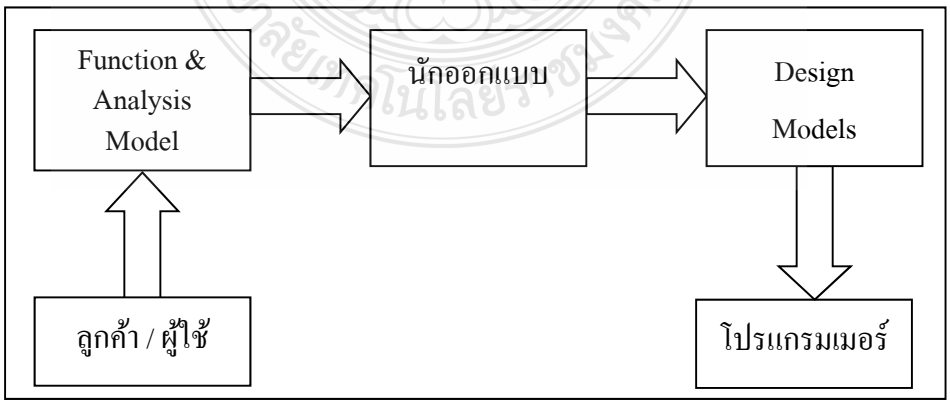
2.2 ทฤษฎีวิศวกรรมซอฟต์แวร์ (Software Engineering)

2.2.1 การออกแบบระบบ (System Design)

การออกแบบระบบ (System Design) หรือกล่าวในแง่ของวิศวกรรมซอฟต์แวร์นั้น ได้กำหนดถึงสิ่งที่ซอฟต์แวร์ต้องทำเพียงคร่าวๆ เท่านั้น ไม่ได้ระบุรายละเอียดการทำงานภายในและลักษณะอื่นๆ ของซอฟต์แวร์ไว้ในรูปแบบที่โปรแกรมเมอร์สามารถนำไปเขียนโปรแกรมได้ ดังนั้น ในขั้นตอนการออกแบบ ทีมงานจะต้องกำหนดรายละเอียดในแต่ละส่วนประกอบซอฟต์แวร์เพื่อเตรียมพร้อมไว้สำหรับการเขียนและทดสอบโปรแกรมในระยะการสร้างซอฟต์แวร์ต่อไป แต่ก่อนที่จะกล่าวถึงการออกแบบส่วนต่างๆ ของซอฟต์แวร์ในบทต่อไป เนื้อหาในบทนี้จะกล่าวถึงแนวคิดสำคัญของการออกแบบซอฟต์แวร์ในแง่ของวิศวกรรมซอฟต์แวร์ก่อน เพื่อความเข้าใจอันดีในแนวคิดดังกล่าว

2.2.2 ความหมายของการออกแบบซอฟต์แวร์

ก่อนที่จะกล่าวถึงความหมายของการออกแบบซอฟต์แวร์ ในที่นี้ขอกล่าวถึงความหมายโดยรวมของ “การออกแบบระบบ” เป็นการนำความต้องการของผู้ใช้มาแปลงให้อยู่ในรูปของแบบ (เปรียบได้กับพิมพ์เขียว)ก่อนนำไปสร้างเป็นผลิตภัณฑ์ที่แล้วเสร็จ สิ่งที่ได้จากการออกแบบคือ “ ข้อกำหนดเฉพาะของการออกแบบ” (Design Specification Document) ที่ประกอบไปด้วยแบบจำลองของส่วนประกอบที่จะรวมเข้าเป็นระบบบน สถาปัตยกรรมที่เหมาะสม ดังนั้นสิ่งจำเป็นที่สุดที่จะนำมาใช้ในการออกแบบ ก็คือข้อกำหนดความต้องการของผู้ใช้และแบบจำลองที่ได้จากการวิเคราะห์ เพื่อนำมาสร้างเป็นแบบจำลองของการออกแบบที่มีรายละเอียดทางเทคนิคมากพอที่จะเป็นประโยชน์ในการเขียนโปรแกรมได้ แสดงภาพแบบจำลองความสำคัญของข้อกำหนดความต้องการ ดังรูปที่ 2.1



รูปที่ 2.1 การนำ Analysis Model มาใช้ในการออกแบบ

แต่สำหรับงานด้านวิศวกรรมซอฟต์แวร์ที่มุ่งเน้นการผลิตซอฟต์แวร์เป็นหลักจะถือว่างานออกแบบซอฟต์แวร์เป็นหัวใจของกระบวนการผลิตโดย IEEE 610.12 ได้ให้คำจำกัดความของคำว่า “การออกแบบซอฟต์แวร์” ไว้ดังนี้

การออกแบบซอฟต์แวร์ (Software Design) คือ กระบวนการกำหนดสถาปัตยกรรม ส่วนประกอบ ส่วนประสานและลักษณะอื่นๆ ของระบบหรือส่วนประกอบของระบบ

2.2.3 การออกแบบซอฟต์แวร์ (Software Design)

คือ กระบวนการกำหนดสถาปัตยกรรม ส่วนประกอบ ส่วนประสานและลักษณะด้านอื่นๆ ของระบบหรือส่วนประกอบระบบ โดยการออกแบบซอฟต์แวร์ยังมีความหมายรวมถึงสิ่งที่ได้จากการออกแบบ ซึ่งก็คือ แบบจำลองการออกแบบ (Design Model) อีกด้วย

การออกแบบซอฟต์แวร์เป็นการนำข้อกำหนดความต้องการของผู้ใช้ (ลูกค้า) มากำหนดรายละเอียดโครงสร้างภายในของซอฟต์แวร์เพื่อนำไปใช้ในการเขียนและทดสอบโปรแกรม ในระยะการสร้างซอฟต์แวร์ในทางวิศวกรรมซอฟต์แวร์แล้วการนำความรู้ด้านวิศวกรรมมาประยุกต์ใช้กับการออกแบบ ก็คือ วิศวกรรมการออกแบบ (Design Engineering) ซึ่งมีเป้าหมาย คือ การสร้างแบบร่างของระบบหรือการนำเสนอระบบในแต่ละด้าน ให้มีคุณสมบัติที่ดี ได้แก่ “Firmness” (โปรแกรมที่ได้จากการออกแบบจะต้องไม่มีข้อผิดพลาด) “Commodity” (จะต้องตรงกับวัตถุประสงค์การใช้งาน) และ “Delight” (ต้องทำให้ผู้ใช้รู้สึกพึงพอใจ) ทั้งหมดคือ “คุณภาพ”ที่ได้จากงานออกแบบ แต่การที่จะได้งานออกแบบมีคุณภาพนั้น จะต้องอาศัยปัจจัยหลายอย่าง ไม่ว่าจะเป็นประสบการณ์ของบุคลากร หลักการ แนวทาง เครื่องมือ และระเบียบวิธีที่จะนำมาใช้ รวมถึงการกำหนดเงื่อนไขเพื่อวัดคุณภาพของงานก็เป็นสิ่งสำคัญเทียบเท่ากับการทำงานแบบวนซ้ำเพื่อปรับปรุงงานให้ดีที่สุด

2.2.4 กระบวนการออกแบบซอฟต์แวร์

จะมีลักษณะการทำงานแบบซ้ำๆ เนื่องจากต้องนำความต้องการของระบบที่ผ่านการวิเคราะห์แล้วในแต่ละด้าน ทั้งด้านข้อมูล ฟังก์ชันงาน และส่วนประสาน มาแปลงเป็นข้อกำหนดของการออกแบบระบบ ดังนั้น ข้อกำหนดการออกแบบจึงสอดคล้องกับข้อกำหนดความต้องการและสามารถใช้สื่อสารกับโปรแกรมเมอร์ได้

สำหรับกระบวนการออกแบบซอฟต์แวร์นั้น ในที่นี้ไม่ขอกล่าวถึงขั้นตอนอย่างละเอียด เนื่องจากขั้นตอนในการออกแบบดังกล่าว ขึ้นอยู่กับแนวทางและระเบียบวิธีที่ทีมงานเลือกใช้ ระดับบน กล่าวคือ เป็นการแสดงให้เห็นส่วนประกอบต่างๆ ของซอฟต์แวร์ภายใต้โครงสร้างสถาปัตยกรรมรูปแบบใดๆ

2.2.5 หลักการออกแบบซอฟต์แวร์

เนื่องจากการออกแบบซอฟต์แวร์ต้องคำนึงถึงคุณภาพของซอฟต์แวร์ที่จะผลิตด้วย ดังนั้น ทีมงานจึงควรใช้แนวทางการออกแบบบางประการ เพื่อนำไปสู่การออกแบบที่ดี ดังนี้

1) การออกแบบควรแสดงให้เห็นถึงรูปแบบสถาปัตยกรรมที่เลือกใช้อย่างชัดเจนและมีแบบแผนต้องเกิดจากการออกแบบคอมโพเน้นท์หรือส่วนประกอบอื่นๆที่ดี และต้องนำไปพัฒนาแบบ Evolutionary ได้ ดังนั้นจึงต้องออกแบบให้พัฒนาและทดสอบได้ง่าย

2) การออกแบบควรมีลักษณะเป็น Module (Modular) กล่าวคือ ควรมีการแบ่งระบบใหญ่ออกเป็นระบบย่อย

3) การออกแบบควรนำเสนอด้านข้อมูล สถาปัตยกรรม ส่วนประสาน และคอมโพเน้นท์ที่ชัดเจน ควรออกแบบคอมโพเน้นท์ให้มีอิสระต่อกันควรออกแบบให้ส่วนประสานระหว่างคอมโพเน้นท์กับสภาพแวดล้อมภายนอก มีความซับซ้อนน้อยที่สุด การออกแบบควรนำข้อมูลมาจากการวิเคราะห์ระบบและใช้ระเบียบปฏิบัติเดียวกันสัญลักษณ์ที่ใช้ในการออกแบบควรสื่อความหมายได้ชัดเจนและเป็นมาตรฐาน งานออกแบบควรมีโครงสร้างที่ดีเพื่อการแก้ไขที่ง่ายและใช้ต้นทุนน้อย การออกแบบในระดับคอมโพเน้นท์ที่มีลักษณะเป็นแบบ “Functional Independence” คือ ฟังก์ชันงานมีความเป็นอิสระไม่ขึ้นต่อกัน คอมโพเน้นท์ของซอฟต์แวร์จะต้องมีลักษณะการขึ้นต่อกันน้อยที่สุด (Loosely Coupled)

2.2.6 การเขียนโปรแกรม

เมื่อจบขั้นตอนการออกแบบ วิศวกรซอฟต์แวร์และทีมงานจะมีรายละเอียดในด้านต่างๆ ของซอฟต์แวร์ แสดงเป็นแผนภาพชนิดต่างๆ ประกอบกับคำอธิบายรวมอยู่ในเอกสารประกอบการออกแบบ หน้าที่ต่อไปของทีมงานคือ การนำรายละเอียดดังกล่าวมาสร้างเป็นผลิตภัณฑ์ซอฟต์แวร์ด้วย “การเขียนโปรแกรม” (Program Writing) ซึ่งเป็นขั้นตอนที่ค่อนข้างยาก เนื่องจากบางครั้งทีมงานออกแบบไม่ได้กำหนดลักษณะเฉพาะของแพลตฟอร์มและสภาพแวดล้อมต่างๆ ของการเขียนโปรแกรมมาให้ด้วย จึงทำให้โปรแกรมเมอร์ต้องกำหนดลักษณะเฉพาะดังกล่าวเพิ่มเติมนอกจากนี้ในการเขียนโปรแกรม นอกจากทีมโปรแกรมเมอร์จะต้องเขียนโปรแกรมที่อ่านเข้าใจง่ายเมื่อต้องกลับมาทดสอบและแก้ไขโปรแกรมแล้วทีมโปรแกรมเมอร์ยังต้องจัดทำเอกสารโปรแกรมที่อ่านเข้าใจง่ายสำหรับบุคคลอื่นที่เกี่ยวข้องอีกทั้งยังต้องคำนึงถึงโครงสร้างของโปรแกรมโครงสร้างข้อมูลและแนวทางการเขียนโปรแกรมแบบ Reuse อีกด้วย

2.2.7 หลักปฏิบัติในการเขียนโปรแกรม

สำหรับหัวข้อนี้จะกล่าวถึงหลักปฏิบัติในการเขียนโปรแกรมทั่วไป โดยไม่ขึ้นอยู่กับภาษาคอมพิวเตอร์ภาษาใดภาษาหนึ่งโดยเฉพาะ เนื่องจากทุกโปรแกรมที่ถูกสร้างขึ้น ไม่ว่าจะใช้

ภาษาคอมพิวเตอร์ใดก็ตาม จะต้องมีโครงสร้างหลักการเหมือนกัน 3 อย่าง ได้แก่ โครงสร้างควบคุมการทำงานของโปรแกรม (ControlStructure) อัลกอริทึม (Algorithm) และโครงสร้างข้อมูล (Data Structure) ดังนั้นในที่นี้จึงขอกล่าวถึงหลักการเขียนโปรแกรมในโครงสร้างแต่ละส่วน

2.2.8 โครงสร้างควบคุมการทำงานของโปรแกรม (Control Structure)

การควบคุมการทำงานของโปรแกรมย่อยทุกรูปแบบ ล้วนมีความสำคัญต่อการเขียนโปรแกรมทั้งสิ้น เนื่องจากการควบคุมการทำงานเป็นส่วนสะท้อนให้เห็นถึงโครงสร้างควบคุมการทำงานของโปรแกรม ทั้งนี้ก็เพื่อความสะดวกในการอ่านโค้ดให้เข้าใจการทำงานของโปรแกรมโดยไม่ต้องกังวลกับทิศทางการทำงานของโปรแกรม สำหรับแนวทางการเขียนโปรแกรมที่จะช่วยให้การอ่านโค้ดง่ายขึ้น

เขียนโค้ดจากบนลงล่าง การเขียนโค้ดให้อ่านจากบนลงล่าง คือ การเขียนในลักษณะที่เป็นไปตามคำสั่งควบคุมของโปรแกรม (Control Statement) กล่าวคือ จะไม่มีลักษณะการเขียนแบบกระโดดไปมา ซึ่งนอกจากจะทำให้การอ่านโค้ดยุ่งยากแล้ว ยังอาจทำให้การทำงานของโปรแกรมผิดพลาดได้

อัลกอริทึม (Algorithm) เป็นลำดับขั้นตอนการทำงานหรือการแก้ไขปัญหางานใดๆ เช่น อัลกอริทึม งานคำนวณภาษีเงินได้บุคคลธรรมดา และ งานเรียงลำดับแบบ QuickSort เป็นต้น อัลกอริทึมดังกล่าวจะแสดงให้เห็นการแก้ปัญหาแต่ละงานอย่างเป็นลำดับขั้นตอน โดยทีมงานออกแบบจะอธิบายอัลกอริทึมด้วยเครื่องมือหลายชนิด เช่น แผนผังโปรแกรม รหัสเทียม หรือภาษาอังกฤษเชิงโครงสร้างเป็นต้น

หน้าที่ของโปรแกรมเมอร์ในขั้นตอนนี้ก็คือ นำอัลกอริทึมที่ได้จากงานออกแบบมาเขียนเป็นโค้ดโปรแกรม ภายใต้ภาษาโปรแกรมมิ่ง แพลตฟอร์ม และฮาร์ดแวร์ที่กำหนด โดยหลักปฏิบัติที่ต้องคำนึงถึงเมื่อต้องแปลงอัลกอริทึมมาเป็นโค้ด คือ ต้องเขียนโค้ดให้มีประสิทธิภาพและประสิทธิผล ซึ่งหมายถึง คุณภาพของโค้ดนั่นเอง คุณภาพของโค้ดโปรแกรมไม่ได้หมายถึงการทำงานที่รวดเร็วเพียงอย่างเดียว แต่ยังหมายถึงผลลัพธ์ที่ได้จะต้องถูกต้องแม่นยำ ตรงตามมาตรฐานและความต้องการของผู้ใช้ด้วย

2.2.9 การจัดทำเอกสารโปรแกรม

สิ่งสำคัญอย่างหนึ่งในขั้นตอนการเขียนโปรแกรมก็คือ เอกสารประกอบโปรแกรม (Program Document) เป็นส่วนที่ช่วยให้บุคคลอื่นสามารถเข้าใจหน้าที่วัตถุประสงค์และการทำงานของโปรแกรมได้ง่ายขึ้น เนื่องจากความจำเป็นในการทดสอบ ประเมินและประกอบรวมโปรแกรมเข้ากับซอฟต์แวร์ โปรแกรมเมอร์จึงต้องจัดทำเอกสารประกอบโปรแกรมให้มีรายละเอียดที่ชัดเจน

สมบูรณ์ โดยสามารถแบ่งเอกสารออกเป็น 2 ประเภท ได้แก่ เอกสารภายใน (Internal Document) และเอกสารภายนอก (External Document)

1) การจัดทำเอกสารภายใน (Internal Documentation) คือ เอกสารที่แสดงรายละเอียดโค้ดทั้งหมดของซอฟต์แวร์ ซึ่งนอกจากส่วนที่เป็นโค้ดแล้ว ยังรวมถึงหมายเหตุโปรแกรมด้วย ในที่นี้กล่าวถึงหลักการเขียนหมายเหตุโปรแกรมและส่วนอื่นๆ ที่จำเป็น ดังนี้

Header Comment Block คือ การเขียนหมายเหตุไว้ที่ส่วนหัวของโปรแกรมหรือคอมโพเน้นท์ลงบนเอกสารประกอบโปรแกรม เป็นส่วนแสดงที่มา หน้าที่และการใช้งานโปรแกรม ซึ่งต้องประกอบไปด้วยรายละเอียดต่างๆ ดังนี้

- ชื่อโปรแกรม ควรสื่อความหมายชัดเจน
- ชื่อโปรแกรมเมอร์ ซึ่งรวมถึงเบอร์โทรศัพท์และ E-mail Address ด้วย
- หน้าที่ของโปรแกรม
- วันที่เขียนโปรแกรมเสร็จสิ้นและวันที่ปรับปรุง รวมถึงชื่อหรือหมายเลขรุ่นของโปรแกรมด้วย
- ความสำคัญของโปรแกรม
- วิธีใช้ข้อมูล อัลกอริทึมและวิธีควบคุมการทำงานนอกจากนี้ โปรแกรมเมอร์สามารถอ่านรายละเอียดเพิ่มเติมได้
- ชื่อและชนิดข้อมูลของตัวแปร
- อธิบายอัลกอริทึมและการตอบสนองต่อความผิดพลาดที่เกิดขึ้นพอสังเขป
- แสดงข้อมูลที่นำเข้าและผลลัพธ์ที่คาดหวัง
- วิธีทดสอบโปรแกรมและวิธีใช้งานโปรแกรม
- วิธีขยายขีดความสามารถของโปรแกรม
- การแก้ไขโปรแกรม

หมายเหตุส่วนอื่นๆ ของโปรแกรม คือ การเขียนหมายเหตุกำกับในแต่ละส่วนการทำงานของโปรแกรม โดยจะต้องเขียนรายละเอียดเพิ่มเติมที่จะช่วยให้บุคคลอื่นสามารถเข้าใจได้ง่ายขึ้น ไม่ควรเขียนซ้ำในสิ่งที่มีอยู่แล้ว

ตั้งชื่อตัวแปรให้สื่อความหมายชัดเจน ชื่อ Label และชื่อตัวแปรควรกำหนดให้สื่อความหมายได้อย่างชัดเจน สามารถจำแนกความแตกต่างได้ และไม่ควรตั้งชื่อซ้ำ

จัดรูปแบบโค้ดและหมายเหตุให้อ่านง่ายขึ้น การจัดรูปแบบหรือตกแต่งข้อความโค้ดจะช่วยให้อ่านหมายเหตุได้ง่ายขึ้น ไม่ปะปนประโยคคำสั่งของโค้ด โดยส่วนประโยคคำสั่ง

ควบคุมประเภทต่างๆ (เช่น การตัดสินใจหรือการทำซ้ำ) ควรมีการจัดย่อหน้าให้กับประโยคคำสั่งย่อยภายในประโยค จะช่วยให้จำแนกบล็อกคำสั่งได้ง่ายขึ้นด้วย

2) การทำเอกสารภายนอก (External Documentation) คือเอกสารที่แสดงรายละเอียดส่วนอื่นๆ นอกเหนือจากโค้ดโปรแกรมเป็นเอกสารที่โปรแกรมเมอร์สามารถอธิบายรายละเอียดอื่นเพิ่มเติมจากส่วนหมายเหตุโปรแกรมได้ เนื่องจากในส่วน Header Comment Block ของเอกสารภายในนั้น เป็นเพียงข้อมูลสรุป โดยโปรแกรมเมอร์ต้องเขียนขยายความหรือทำเป็นรายงานเพิ่มเติมด้วยเอกสารภายนอก โดยรายละเอียดส่วนใหญ่ของเอกสารภายนอก เป็นการอธิบายรายละเอียดของระบบมากกว่าจะอธิบายรายละเอียดของโปรแกรม

นอกจากนี้ ในเอกสารภายนอกควรแนบแผนภาพหรือแบบจำลองที่จะแสดงให้เห็นโครงสร้างของโปรแกรมหรือคอมโพเน้นท์ การรับส่งข้อมูลระหว่างคอมโพเน้นท์ ความสัมพันธ์ของคอมโพเน้นท์ทั้งหมด ตลอดจนสืบทอดคลาสของระบบด้วย สรุปส่วนประกอบของเอกสารภายนอก มีดังนี้

อธิบายปัญหา ส่วนแรกของเอกสาร ควรอธิบายปัญหา ซึ่งเป็นที่มาที่ทำให้ต้องเขียนโปรแกรมหรือคอมโพเน้นท์ขึ้นมาแก้ปัญหานั้น โดยการอธิบายปัญหาดังกล่าว ไม่ได้เป็นการกล่าวซ้ำกับเอกสารความต้องการของระบบ หากแต่เป็นการกล่าวถึงความสำคัญของคอมโพเน้นท์ที่สร้างขึ้นมา ว่าเพราะเหตุใดจึงจำเป็นต้องสร้างคอมโพเน้นท์ดังกล่าวขึ้น

อธิบายอัลกอริทึม เป็นการอธิบายอัลกอริทึมที่ใช้แก้ปัญหางานที่คอมโพเน้นท์รับผิดชอบ ซึ่งรวมถึงสูตรคำนวณที่ใช้ เงื่อนไขและขอบเขตในการแก้ไขปัญหางาน

อธิบายข้อมูล เป็นการอธิบายให้เห็นทิศทางการรับ-ส่งข้อมูลระหว่างคอมโพเน้นท์หรือระหว่างกระบวนการ ดังนั้นในเอกสารแนบแผนภาพกระแสข้อมูล (Data Flow Diagrams: DFD) และพจนานุกรมข้อมูล (Data Dictionary) ไว้ด้วย สำหรับระบบที่ใช้แนวทางเชิงวัตถุให้แนบแผนภาพ Class Diagrams ที่แสดงความสัมพันธ์ระหว่างคลาสด้วย

2.2.10 กระบวนการเขียนโปรแกรม

โปรแกรมที่มีคุณภาพ ไม่ได้มาจากการออกแบบที่มีคุณภาพเพียงปัจจัยเดียว หากแต่มีปัจจัยอื่นเข้ามาเกี่ยวข้องด้วย นั่นคือ ทักษะ ประสบการณ์ และความเฉลียวฉลาดที่เกิดจากการมีจินตนาการและกระบวนการแก้ปัญหาที่ดีของโปรแกรมเมอร์ ดังนั้นกระบวนการเขียนโปรแกรมจึงมีความเกี่ยวข้องกับกระบวนการแก้ปัญหาของโปรแกรมเมอร์ ในที่นี้จึงขอทำความเข้าใจกับกระบวนการแก้ไขปัญหา ที่เชื่อมโยงกับกระบวนการเขียนโปรแกรม โดยแบ่งเป็น 4 ขั้นตอน ดังนี้

1) ทำความเข้าใจปัญหา เป็นการแบ่งปัญหาออกเป็นส่วนย่อยแล้ววิเคราะห์ในแต่ละส่วนเพื่อให้เกิดความเข้าใจในสิ่งที่เกิดขึ้นอะไรคือสิ่งที่เกี่ยวข้องกับปัญหา อะไรไม่ควรเกี่ยวข้อง

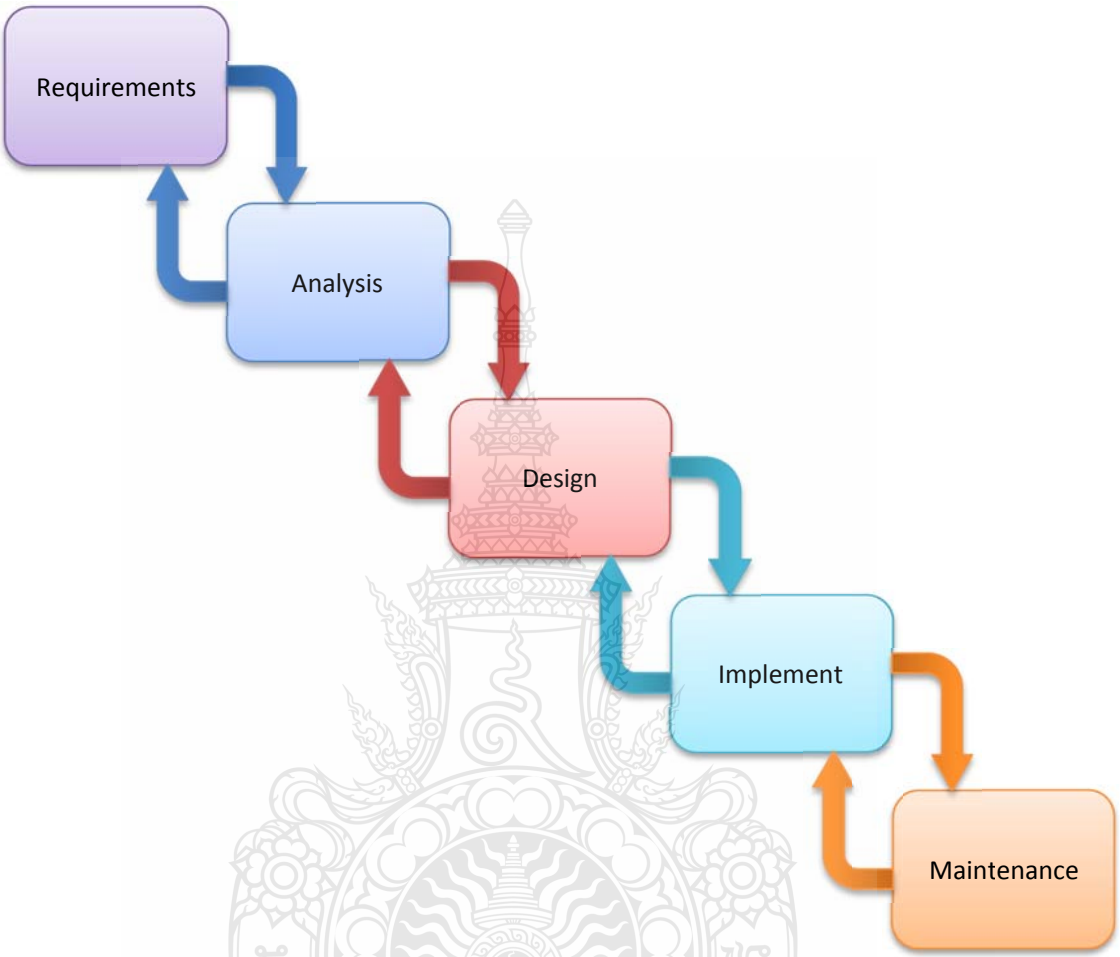
และควรตัดออก เช่นเดียวกับการวิเคราะห์ระบบเพื่อหาสิ่งแวดล้อมภายนอกที่เกี่ยวข้อง กับระบบ แต่สิ่งที่สำคัญในการวิเคราะห์ปัญหา คือ การทำความเข้าใจเงื่อนไขที่เป็นข้อจำกัดในการแก้ไขปัญหา

2) สำหรับโปรแกรมเมอร์แล้ว การทำความเข้าปัญหา ก็คือ การวิเคราะห์งาน นั่นเอง โดยส่วนใหญ่โปรแกรมเมอร์ จะใช้แผนผัง (Flowchart) หรือแผนภาพชนิดอื่นๆ เป็นเครื่องมือในการจำลองลำดับขั้นตอนของงาน ซึ่งนอกจากจะช่วยให้เข้าใจปัญหาได้ดีแล้ว แผนภาพยังช่วยให้โปรแกรมเมอร์ได้พิจารณาถึงเงื่อนไขในแต่ละส่วนอย่างถี่ถ้วน และช่วยแบ่งปัญหาออกเป็นส่วนย่อย เพื่อให้ทำงานได้ง่ายขึ้นด้วย

3) วางแผนแก้ไขปัญหา เป็นการออกแบบวิธีแก้ไขปัญหาที่ผ่านการวิเคราะห์มาแล้ว โดยอาจพิจารณาหาส่วนที่เกี่ยวข้องกับปัญหา เช่น ข้อมูล อัลกอริทึม ไลบรารี หรือระเบียบวิธีปฏิบัติ เพื่อนำมาปรับใช้หรืออาจใช้วิธีการค้นหาวิธีแก้ปัญหาชนิดเดียวกันจากแหล่งข้อมูลอื่นๆ ซึ่งในทางการเขียนโปรแกรมเชิงวัตถุ จะเรียกวิธีดังกล่าวว่า “DesignPattern”

4) ดำเนินตามแผน เมื่อวางแผนแก้ไขปัญหาเรียบร้อยแล้ว ขั้นต่อมาคือ การดำเนินงานตามแผนที่กำหนดไว้ ซึ่งในเนื้อหาของบทนี้ก็คือ “ลงมือเขียนโค้ดโปรแกรม” นั่นเองทบทวน เป็นการย้อนกลับไปพิจารณาถึงสิ่งที่ดำเนินการแก้ไขปัญหาไปในแต่ละส่วน ในขณะเดียวกันจะต้องประเมินวิธีแก้ไขปัญหานั้นด้วยว่าเหมาะสมหรือถูกต้องที่สุดหรือไม่หากเป็นวิธีที่ดีที่สุดแล้ว จะสามารถนำไปใช้ซ้ำได้หรือไม่ หรือสามารถใช้งานเป็น Pattern ได้หรือไม่

2.2.11 วงจรการพัฒนาระบบ (The System Development Life Cycle: SDLC)

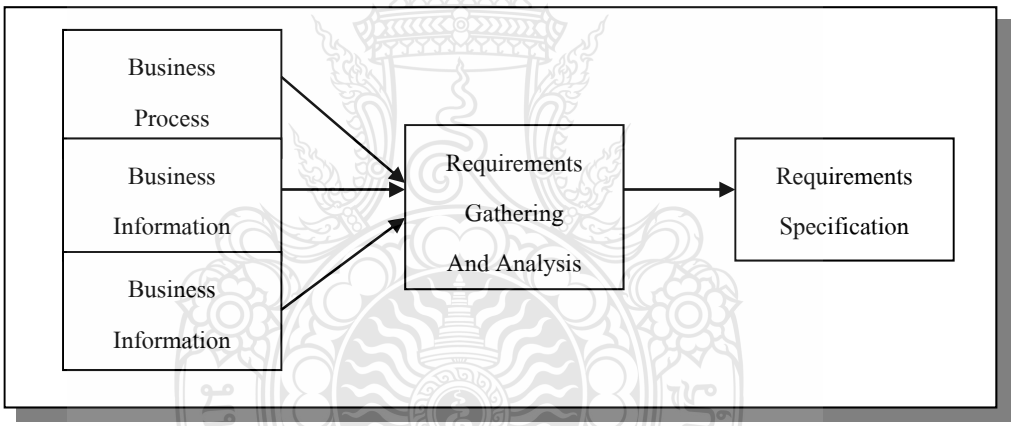


รูปที่ 2.2 วงจรการพัฒนาระบบ

1) ความต้องการ (Requirements)

การกำหนดความต้องการ เป็นขั้นตอนของการกำหนดขอบเขตของปัญหาสาเหตุของปัญหาจากดำเนินงานในปัจจุบัน ความเป็นไปได้กับการสร้างระบบใหม่ การกำหนดความต้องการระหว่างนักวิเคราะห์ระบบกับผู้ใช้งานโดยข้อมูลเหล่านี้ได้จากการสัมภาษณ์การรวบรวมข้อมูลจากการดำเนินงานต่างๆ เพื่อทำการสรุปเป็นข้อกำหนด (Requirements Specification) ที่ชัดเจนในขั้นตอนนี้หากเป็นโครงการที่ขนาดใหญ่ อาจเรียกว่า ขั้นตอนของการศึกษาความเป็นไปได้ (Feasibility Study) เช่น

- ความเป็นไปได้ทางด้านเทคนิค (Technical Feasibility) เป็นการประเมินว่าฮาร์ดแวร์และซอฟต์แวร์ที่มีอยู่ในปัจจุบันสามารถนำไปใช้กับระบบที่กำลังพัฒนาหรือไม่
- ความเป็นไปได้ทางด้านเศรษฐกิจ (Economic Feasibility) เป็นการประเมินว่าประโยชน์ที่ได้รับจากการพัฒนาระบบใหม่คุ้มค่ากับค่าใช้จ่ายที่คาดว่าจะเกิดขึ้นมากน้อยเพียงไรหรือถ้าไม่พัฒนาจะเกิดผลเสียหายอย่างไร
- ความเป็นไปได้เชิงปฏิบัติการ (Operational Feasibility) เป็นการประเมินถึงผลที่อาจจะเกิดขึ้นในทางปฏิบัติ อาทิเช่น ความตั้งใจที่จะนำระบบที่พัฒนาไปใช้จะมีการต่อต้านจากผู้ใช้ที่ไม่เห็นถึงประโยชน์จากการใช้ระบบที่พัฒนาขึ้นใหม่หรือไม่ หรือปัญหาทางด้านความพร้อมของบุคลากรที่จะนำระบบใหม่ไปใช้งาน



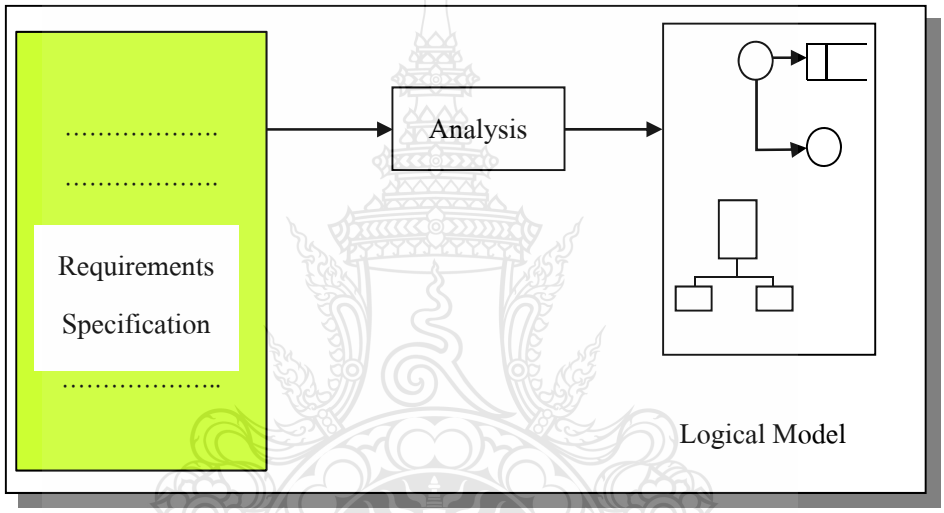
รูปที่ 2.3 Requirements Gathering

สรุปขั้นตอนการกำหนดความต้องการ คือ

- รับรู้สภาพของปัญหาที่เกิดจากการดำเนินงาน
- สรุปหาสาเหตุของปัญหา และสรุปผลยื่นแก่ผู้บริหารเพื่อพิจารณา
- ทำการศึกษาความเป็นไปได้ในแง่มุมต่างๆ เช่นด้านต้นทุน และด้านทรัพยากร
- รวบรวมความต้องการจากผู้ที่เกี่ยวข้องด้วยวิธีการต่างๆ เช่น การรวบรวมเอกสาร การสัมภาษณ์ การสังเกต แบบสอบถาม
- สรุปข้อกำหนดต่างๆ ให้มีความชัดเจน ถูกต้อง และเป็นที่ยอมรับทั้งสองฝ่าย

2) วิเคราะห์ (Analysis)

การวิเคราะห์เป็นขั้นตอนของการดำเนินการวิเคราะห์ของระบบปัจจุบัน โดยการนำ Requirement Specification ที่ได้มาจากขั้นตอนแรกมาวิเคราะห์ในรายละเอียด เพื่อทำการพัฒนาเป็นรูปแบบจำลองลอจิคัล (Logical Model) ซึ่งประกอบด้วย แผนภาพกระแสข้อมูล (Data Flow Diagram) คำอธิบายการประมวลผลข้อมูล (Process Description) และแบบจำลองข้อมูล (Data Model) ในรูปแบบของ ER-Diagram ทำให้ทราบถึงรายละเอียดขั้นตอนการดำเนินงานในระบบว่าประกอบด้วยอะไรบ้าง มีความเกี่ยวข้องมีความสัมพันธ์กับสิ่งใด



รูปที่ 2.4 Application Analysis

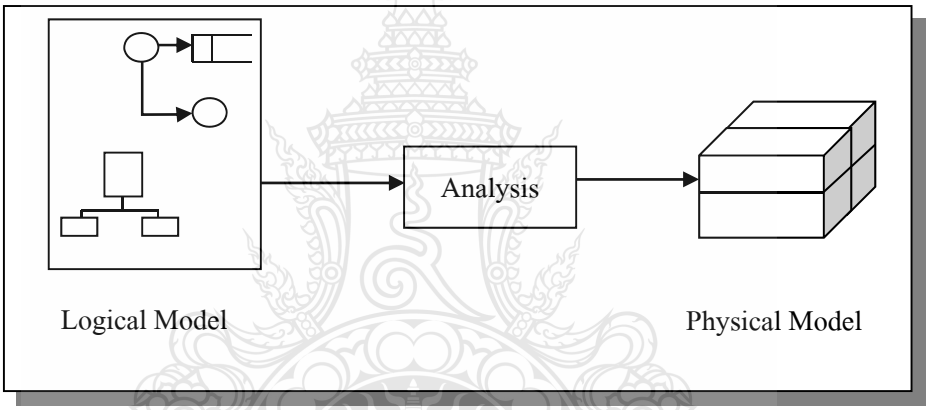
สรุปขั้นตอนวิเคราะห์ คือ

- วิเคราะห์ระบบงานปัจจุบัน
- รวบรวมความต้องการในด้านต่างๆ และนำมาวิเคราะห์เพื่อสรุปเป็นข้อกำหนดที่ชัดเจน
- นำข้อกำหนดมาพัฒนาออกมาเป็นความต้องการของระบบใหม่
- สร้างแบบจำลองกระบวนการของระบบใหม่ด้วยการวาดแผนภาพกระแสข้อมูล
- สร้างแบบจำลองข้อมูล ด้วยการวาดอีอาร์ไดอะแกรม (Entity Relationship Diagram : ERD)

3) ออกแบบ (Design)

การออกแบบเป็นขั้นตอนของการนำผลลัพธ์ที่ได้จากการวิเคราะห์ทาง ลอจิคัล มาพัฒนาเป็นฟิสิคัลโมเดล (Physical Model) ให้สอดคล้องกัน โดยการออกแบบจะเริ่มจากส่วนของอุปกรณ์เทคโนโลยีต่างๆ และโปรแกรมคอมพิวเตอร์ที่นำมาพัฒนาการออกแบบจำลองข้อมูล (Data Design) และการออกแบบจอภาพในการติดต่อกับผู้ใช้งาน (User Interface) การจัดทำพจนานุกรมข้อมูล (Data Dictionary) ซึ่งขั้นตอนของการวิเคราะห์และออกแบบจะมุ่งเน้นถึงสิ่งต่อไปนี้

- วิเคราะห์ มุ่งเน้นการแก้ปัญหาอะไร (What)
- การออกแบบ มุ่งเน้นการแก้ปัญหาอย่างไร (How)



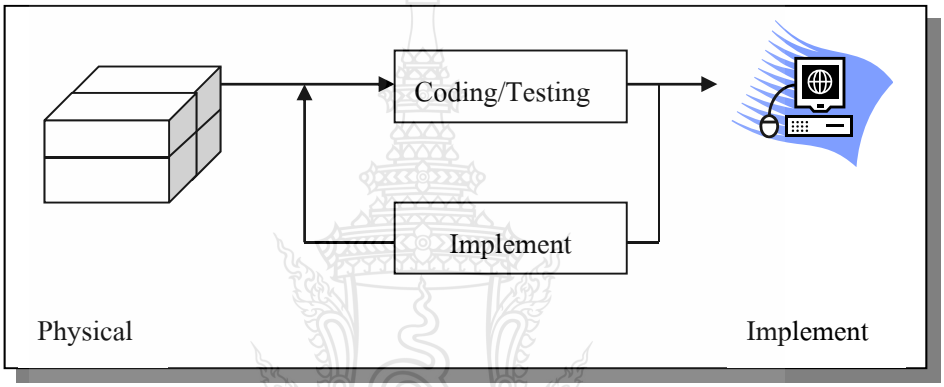
รูปที่ 2.5 การออกแบบโมเดล

สรุปขั้นตอนการออกแบบ คือ

- พิจารณาแนวทางในการพัฒนาระบบ
- ออกแบบสถาปัตยกรรมระบบ (Architecture Design)
- ออกแบบฐานข้อมูล (Database Design)
- ออกแบบเอาต์พุต (Output Design)
- ออกแบบอินพุต (Input Design)
- ออกแบบยูสเซอร์อินเตอร์เฟซ (User Interface Design)
- จัดทำต้นแบบ (Prototype)
- ออกแบบโปรแกรม (Structure Chart)

4) การพัฒนาและนำไปใช้ (Implementation)

การพัฒนาและนำไปใช้ จะทำให้ระบบเกิดผลขึ้นมาด้วยการสร้างระบบ ทดสอบระบบ และการติดตั้งระบบโดยวัตถุประสงค์หลักไม่ใช่เพียงแค่ความน่าเชื่อถือของระบบ หรือระบบต้องสามารถทำงานได้ดีเพียงเท่านั้น แต่ต้องมั่นใจว่าผู้ใช้ระบบต้องได้รับการฝึกอบรมเพื่อใช้งานระบบ และความคาดหวังในองค์กรที่ต้องการผลตอบแทนในด้านดีกับการใช้ระบบใหม่ ลำดับกิจกรรมต่างๆ ทุกกิจกรรมจะต้องเข้ามาดำเนินการร่วมกันในระยะนี้เพื่อให้ระบบการปฏิบัติงานลงเอยถึงที่สุด



รูปที่ 2.6 Coding / Test and Implement

สรุปขั้นตอนพัฒนา คือ

- สร้างระบบขึ้นมาด้วยการเขียนโปรแกรม
- ตรวจสอบความถูกต้องทั้งด้าน Verificationและ Validationและดำเนินการทดสอบระบบ
- แปลงข้อมูล (Convert Data)
- ติดตั้งระบบ (System Installation) และจัดทำเอกสารคู่มือ
- ฝึกอบรมผู้ใช้ และประเมินผลระบบใหม่

5) บำรุงรักษา (Maintenance)

เป็นขั้นตอนของการปรับปรุงแก้ไขระบบหลังจากที่ได้มีการติดตั้งระบบและใช้งานแล้ว ในขั้นตอนนี้อาจเกิดปัญหาของโปรแกรม (Bug) ซึ่งโปรแกรมเมอร์จะต้องแก้ไขให้ถูกต้อง หรือเกิดจากความต้องการของผู้ใช้งานที่ต้องการเพิ่มโมดูลในการทำงานอื่นๆ “ซึ่งทั้งนี้ก็จะ

เกี่ยวกับ Requirement Specification” ที่เคยตกลงกันไว้ก่อนหน้าด้วย ดังนั้นในส่วนงานนี้จะคิดค่าใช้จ่ายเพิ่มหรืออย่างไร เป็นเรื่องของรายละเอียดที่ผู้พัฒนาหรือนักวิเคราะห์ระบบจะต้องดำเนินการกับผู้ว่าจ้างต่อไป

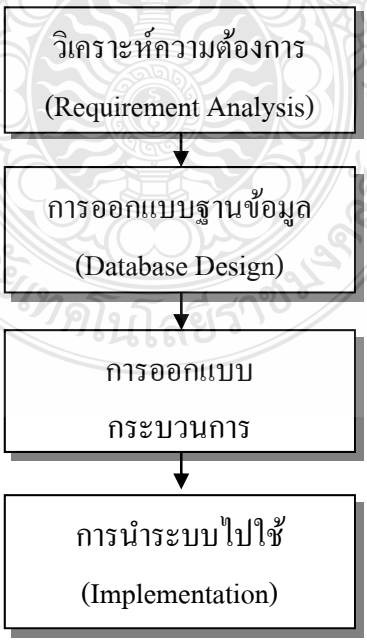
สรุปขั้นตอนการบำรุงรักษา คือ

- อาจมีข้อผิดพลาดบางอย่างต้องรีบแก้ไขโปรแกรมให้ถูกต้องโดยด่วน
- ในบางครั้งอาจมีการเพิ่มโมดูล หรืออุปกรณ์บางอย่างเข้าไปในระบบ
- การสนับสนุนงานของผู้ใช้
- การบำรุงรักษาจะกระทำทั้งสองด้าน คือ การบำรุงรักษาทางด้านซอฟต์แวร์และทางด้านฮาร์ดแวร์ (System Maintenance and Software Maintenance)

2.2.12 แนวทางในการออกแบบฐานข้อมูล

1) แนวทางที่เป็น Data Driven หรือ Data Oriented

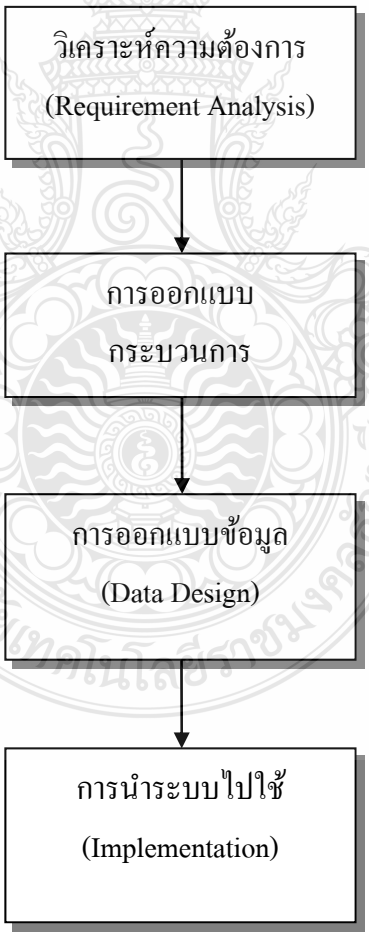
เป็นลักษณะการออกแบบระบบโดยการพิจารณาจากข้อมูลที่หน่วยงานต่างๆ ขององค์กร ต้องการเพื่อกำหนดรายละเอียดของฐานข้อมูล การออกแบบด้วยแนวทางนี้จะเริ่มต้นโดยพิจารณาจากข้อมูลที่ผู้ใช้ต้องการซึ่งอาจจะวิเคราะห์จากหน้าจอหรือรายงานของแต่ละงานแล้วรวบรวมข้อมูลทั้งหมดเพื่อกำหนดข้อมูลที่ต้องการใช้และความสัมพันธ์ของข้อมูลในฐานข้อมูลขึ้นมา ดังรูปที่ 2.7



รูปที่ 2.7 การออกแบบตามแนวทาง Data Oriented

2) แนวทางที่เป็น Process Driven หรือ Process Oriented

แนวทางการออกแบบฐานข้อมูลโดยเริ่มการวิเคราะห์จากกระบวนการที่เกิดขึ้นในระบบ แนวเส้นทางนี้สามารถใช้แผนภูมิแสดงกระแสข้อมูล ซึ่งเป็นเครื่องมือที่ใช้ในการออกแบบระบบประยุกต์ใช้งานแผนภูมินี้แสดงถึงการเคลื่อนไหวของข้อมูลที่ต้องจัดเก็บ เพื่อพิจารณาในภาพรวมว่าข้อมูลที่เกี่ยวข้องมีอะไรซึ่งสามารถใช้ประกอบกับการออกแบบฐานข้อมูลได้ โดยทั่วไปแนวทางแบบ Process Oriented นิยมใช้ในการออกแบบระบบประยุกต์งานในขณะที่แนวทางแบบ Data Oriented นิยมใช้การออกแบบฐานข้อมูล ในการออกแบบฐานข้อมูลที่ครบถ้วนควรจะมีแนวทางทั้งสองประกอบกัน ดังนั้นการออกแบบฐานข้อมูลที่ต้องจัดเก็บจากแผนภูมิแสดงกระแสข้อมูลและการออกแบบฐานข้อมูลด้วย E-RModel ประกอบกันสามารถช่วยให้การวิเคราะห์ข้อมูลและการออกแบบฐานข้อมูลที่สมบูรณ์ดังรูปที่ 2.8



รูปที่ 2.8 การออกแบบตามแนวทาง Process Oriented

3) แนวทางที่เป็น Product Driven หรือ Product Oriented

ในกรณีที่ผู้ออกแบบใช้ผลิตภัณฑ์ของบริษัทผู้จำหน่ายซอฟต์แวร์หรือเครื่องมือที่ช่วยในการออกแบบอาทิเช่น Case Tool เป็นเครื่องมือที่ช่วยในการออกแบบฐานข้อมูลโดยเฉพาะ อาจมีขั้นตอนที่จะต้องดำเนินการตามขั้นตอนที่ผลิตภัณฑ์เหล่านั้นกำหนดไว้ เช่น อาจจะเป็น Data Oriented โดยมีเครื่องมือที่ช่วยในการออกแบบ E-RModel และสามารถเชื่อมโยงกับเครื่องมืออื่นๆ ที่ช่วยในการกำหนดโครงสร้างฐานข้อมูลให้โดยอัตโนมัติรวมถึงการออกแบบระบบงานต่างๆเช่น Case หรือ Designer ของ Oracle

2.3 ทฤษฎีการพัฒนาระบบเชิงวัตถุด้วย UML (Unified Modeling Language)

UML ย่อมาจาก The Unified Modeling Language เป็นอีกภาษาเพื่อใช้อธิบายโมเดลต่างๆ ถ้าพูดถึงภาษาเราจะนึกถึงเท็กซ์ (Text) ที่มีไวยากรณ์ต่างๆ แต่ภาษาอีกรูปแบบหนึ่งที่เราอาจจะไม่ค่อยได้คุ้นเคยกันก็คือภาษาที่มีลักษณะของ Map Language กล่าวคือ UML เป็นภาษาที่ใช้กราฟฟิคเป็นสัญลักษณ์ โดยภาษาในลักษณะนี้จะกับคนเฉพาะบางกลุ่ม เช่น นักออกแบบ (Designer) หรือนักพัฒนาระบบคอมพิวเตอร์ (Developer) เป็นต้น

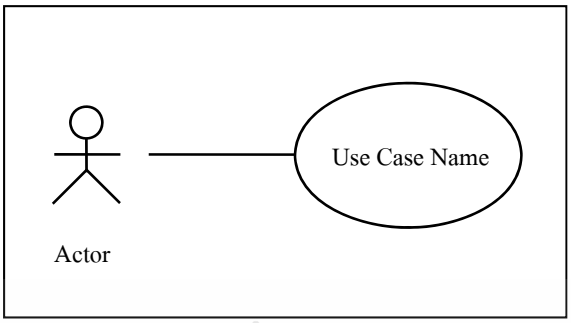
เนื่องจาก UML เป็นภาษาที่มีการใช้กราฟฟิคเป็นสัญลักษณ์จึงอาจมีผู้เข้าในสับสนว่า UML เป็นการสร้างไดอะแกรม เป็นเพียงการใช้สัญลักษณ์สร้างไดอะแกรมเพื่ออธิบายระบบงานเท่านั้น แต่แท้จริงแล้ว UML มีลักษณะของเมต้าโมเดล (Metamodel) คือเป็นโมเดลที่เอาไว้อธิบายโมเดลอื่นๆ อีกที

UML เป็นภาษามาตรฐานสำหรับสร้างแบบพิมพ์เขียว (Blueprint) ให้แก่ระบบงาน เราสามารถใช้ UML ในการสร้างมุมมอง กำหนดรายละเอียด สร้างระบบงานและจัดทำเอกสารอ้างอิงให้แก่ระบบงานได้

UML เป็นภาษาที่เหมาะสำหรับระบบงานระดับกิจการ ระบบแอปพลิเคชั่นบนเว็บ (Web Based Application) ไปจนถึงระบบงานแบบเรียลไทม์ (Real Time System)

2.3.1 ยูส เคส ไดอะแกรม (Use Case Diagrams)

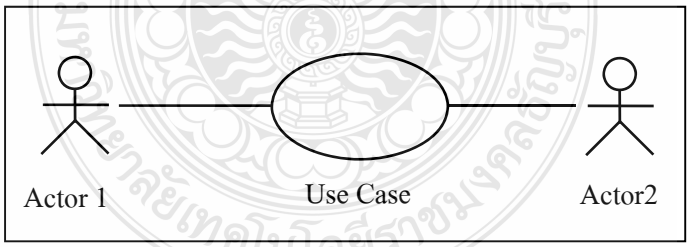
Use Case Diagrams จะแสดงถึงการใช้งานระบบ โดยมีองค์ประกอบ 2 ส่วน คือ Actor และ Use Case โดยที่ Use Case จะแสดงถึงขอบเขตงานที่กำลังสนใจและ Actor คือ สิ่งที่อยู่นอกระบบแต่เป็นผู้ให้อะไรบางอย่างแก่ระบบ อีกทั้งเป็นผู้ที่รับผลลัพธ์จากระบบด้วย สัญลักษณ์ ที่ใช้แทน Actor และ Use Case ใน Use Case Diagrams มีลักษณะดังรูปที่ 2.9



รูปที่ 2.9 แสดงสัญลักษณ์ที่ใช้แทนActor และ Use Case

ในภาพรวมแล้ว Use Case Diagrams จะใช้เพื่อแสดงความสัมพันธ์ระหว่าง Actor ที่ใช้ระบบแสดงความสัมพันธ์ของ Use Case ที่ Actorใช้ และ แสดงความสัมพันธ์ระหว่าง Use Case

1) การสร้าง Use Case Diagrams จะพิจารณาถึงรูปแบบการใช้งานระบบที่สามารถเกิดขึ้นได้โดยอธิบายเป็นลำดับของเหตุการณ์ เช่น ถ้าระบบเป็นตู้ขายน้ำอัดลมกระป๋องแล้วสามารถใช้งานระบบนั้นได้อย่างไรบ้าง (หยอดเหรียญ เลือกชนิดน้ำอัดลม กดปุ่ม ฯลฯ) เหตุการณ์ต่างๆที่จะเกิดขึ้นมี “ผู้กระทำ” ซึ่งผู้ที่กระทำดังกล่าวอาจเป็นคน ระบบ ฮาร์ดแวร์ หรืออะไรก็ตาม โดยจะเรียกผู้ที่กระทำให้เกิดเหตุการณ์ว่า “Actor” และผลลัพธ์ที่ Actor กระทำเหตุการณ์อย่างใดอย่างหนึ่งขึ้นมาก็จะถูก Actor เดิมหรือ Actor อื่นๆ นำไปใช้ต่อ ลักษณะของการเขียน Use Case Diagrams เพื่ออธิบายเหตุการณ์จะมีลักษณะดังรูปที่ 2.10

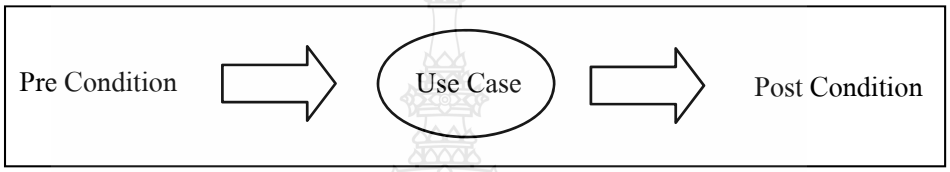


รูปที่ 2.10 แสดงลักษณะการเขียน Use Case Diagrams

นอกจากนี้ การกำหนดบทบาทของ Actor ก็เป็นเรื่องสำคัญอย่างยิ่ง คนๆ หนึ่งสามารถเป็น Actor ได้หลายอย่างและในขณะเดียวกัน Actor หนึ่งๆ ก็อาจมีคนเข้ามามีบทบาทได้หลายคน ดังนั้นการกำหนด Actor จึงต้องทำอย่างระมัดระวังและชัดเจนที่สุด

ข้อดีอย่างหนึ่งของการใช้ Use Case Diagrams คือ จะเห็นได้อย่างชัดเจนว่าขอบเขตของระบบที่กำลังสนใจอยู่ว่ามีอยู่แค่ไหน โดยที่ส่วนของ Use Case คือตัวระบบที่กำลังสนใจอยู่ ส่วน Actor จะเป็นส่วนที่อยู่นอกระบบและเมื่อมีการใช้ Actor ร่วมกับ Use Case ก็จะกลายเป็น Use Case Model หรือ Use Case Diagrams

2) การใช้งาน Use Case Diagrams นั้น ผู้เขียนจำเป็นต้องทำการกำหนดก่อนว่าใครเป็นผู้ใช้ระบบ โดยลักษณะของการใช้ Use Case Diagrams จะมีเงื่อนไขหรือสภาพที่ต้องคำนึงถึงอยู่ 2 อย่างดังรูปที่ 2.11 ได้แก่ Pre Condition และ Post Condition



รูปที่ 2.11 แสดงขั้นตอนการทำงานของ Use Case Diagram

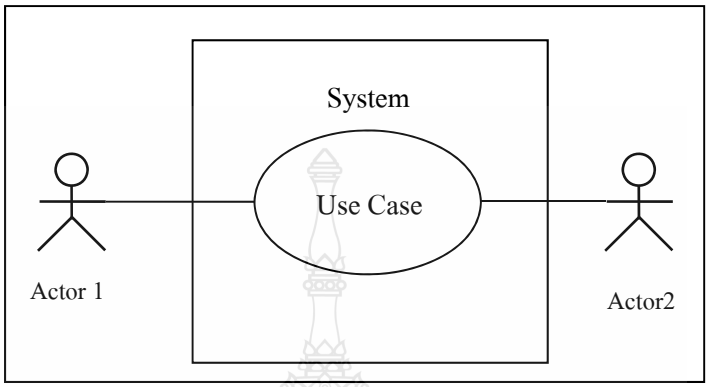
PreCondition คือ สภาพก่อนระบบที่กำลังสนใจ (Use Case) จะทำงานส่วน “Post Condition” คือสภาพหลังจากที่ระบบที่กำลังสนใจได้ทำงานไปแล้ว

ใน Use Case Diagrams หนึ่งๆ จะมี Actor เป็นตัวทำให้เกิด Use Case ขึ้น โดย Actor อาจจะเป็นบุคคลเดียวกันหรือหลายคนอาจจะเป็นระบบงานอะไรอย่างหนึ่งก็ได้ส่วน Use Case ก็คือระบบงานที่กำลังให้ความสนใจและระบบงานที่กำลังสนใจอยู่นั้นจะให้ผลลัพธ์บางอย่างออกมาเพื่อให้ Actor รับงานไปใช้ต่อไป ซึ่ง Actor ที่ได้รับผลลัพธ์จาก Use Case อาจจะเป็น Actor เดียวกับ Actor ที่เป็นผู้ทำให้เกิด Use Case หรือจะเป็น Actor คนละตัวก็ได้ นั่นคือ ขั้นตอนของการเกิด Use Case Diagrams หนึ่งๆ นั้นมีอยู่ 5 ขั้นตอนหลักๆ ได้แก่

- Actor หนึ่งทำให้เกิด Use Case หนึ่งขึ้น
- เกิด PreCondition สำหรับ Use Case
- Use Case มีการทำงานบางอย่าง
- เกิด Post Condition เมื่อ Use Case ทำงานเสร็จสิ้น
- มี Actor หนึ่งได้รับผลลัพธ์จากการทำงานของ Use Case

การวาด Use Case Diagrams จะให้ Actor ที่เป็นผู้ทำให้เกิดการทำงานบางอย่างอยู่ทางด้านซ้ายมือมี Use Case อยู่ถัดมาและมี Actor ที่เป็นผู้รับผลลัพธ์จาก Use Case อยู่ทางด้านขวา

ของ Use Case โดยที่ทั้ง 3 ส่วนใน Use Case Diagrams จะมีเส้นเชื่อมโยงถึงกันอยู่ ดังรูปที่ 2.12

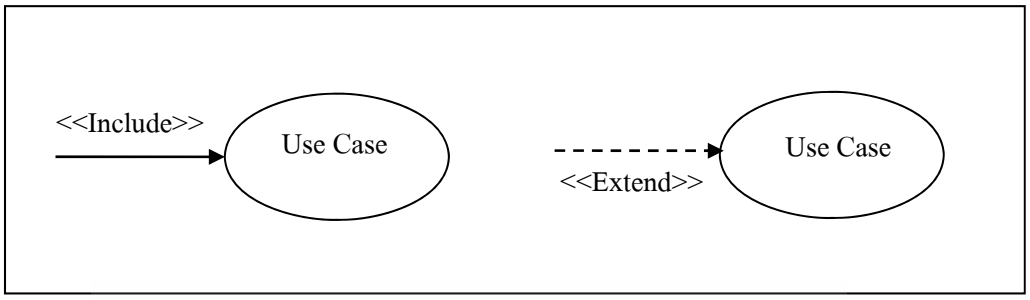


รูปที่ 2.12 แสดงการเกิด Use Case Diagrams

3) การนำ Use Case กลับมาใช้ใหม่ (Reuse)

วิธีการนำ Use Case กลับมาใช้ใหม่ มีอยู่ 2 วิธี คือ

- Inclusion คือ การนำขั้นตอนการทำงานที่ซับซ้อนกันมาสร้างเป็น Use Case แยกต่างหากเพื่อให้ Use Case อื่นได้เรียกใช้ เป็นลักษณะเดียวกันกับเวลาที่เขียนโปรแกรมเป็น Module ย่อยๆ เพื่อให้ Module อื่นๆ เรียกใช้ กล่าวคือ ในการใช้ Use Case Diagrams สำหรับระบบซอฟต์แวร์หนึ่งๆ จะพบว่าจำนวนของ Use Case นั้นมีมากมาย ทั้งนี้เพราะต้องนำเสนอการปฏิสัมพันธ์กันของสิ่งที่อยู่ภายในระบบกับสิ่งที่อยู่ภายนอกระบบในหลายมุมมองและเมื่อจำนวนของ Use Case เพิ่มขึ้นเรื่อยๆ จะพบว่ามีบาง Use Case ที่มีขั้นตอนการทำงานเหมือนๆ กันหรือซ้ำกัน ดังนั้น จึงอาจสร้าง Use Case สำหรับการเรียกใช้ในขั้นตอนที่เหมือนกัน
- Extension คือ การที่นำเอา Use Case เดิมที่มีอยู่แล้วมาเพิ่มการทำงานบางอย่าง



รูปที่ 2.13 สัญลักษณ์การใช้ Inclusion และ Extension

ความสัมพันธ์ 2 แบบแรกนั้นได้กล่าวไปแล้วคือ Inclusion เป็นการนำเอาขั้นตอนการทำงานที่ซ้ำๆ กันมาสร้างเป็น Use Case เพื่อให้ Use Case อื่นเรียกใช้และ Extension คือการนำเอา Use Case เดิมที่มีอยู่แล้วมาเพิ่มเติมการทำงานบางอย่างเข้าไป

Generalization เป็นการถ่ายทอดคุณสมบัติหรือพฤติกรรมบางอย่างจาก Use Case หนึ่งไปยังอีก Use Case หนึ่งหรือจาก Actor หนึ่งไปยังอีก Actor หนึ่ง โดย Use Case ที่เป็นผู้ถ่ายทอดพฤติกรรมจะเรียกว่า “Parent Use Case” ซึ่ง Use Case ที่รับการถ่ายทอดพฤติกรรมมาจะเรียกว่า “Child Use Case” ซึ่ง Child Use Case จะมีการเพิ่มเติมพฤติกรรมบางอย่างของตนเองเข้าไปด้วย

Grouping คือ ในบางครั้ง Use Case Diagrams ที่สร้างขึ้นมาอาจมีจำนวนของ Use Case มากมาย ทำให้ไม่สะดวกต่อการอ่านหรือการนำไปใช้งาน ดังนั้นจึงอาจมีการจับกลุ่มหรือหมวดหมู่ให้แก่ Use Case เหล่านั้นโดยใช้ความสัมพันธ์แบบ Grouping โดยเมื่อปรกติอยู่ในขั้นตอนของการหาความต้องการของระบบ โดยการไปสัมภาษณ์เก็บรวบรวมจากผู้ใช้นั้น จะมีการสร้างเป็น Use Case ต่างๆ เป็นส่วนๆ แยกออกจากกัน จากนั้นจึงค่อยมาทำ Grouping กับ Use Case เหล่านั้นอีกที

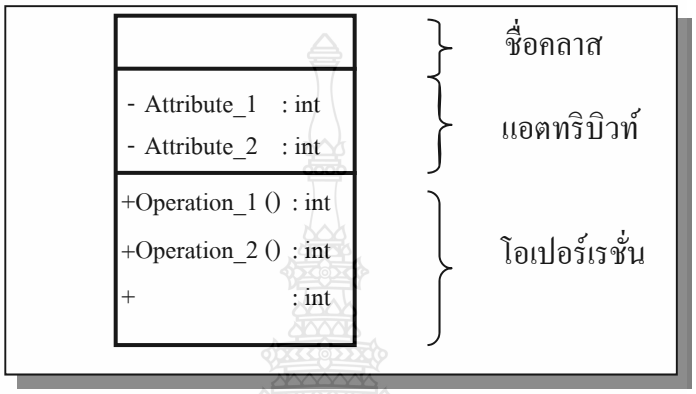
2.3.2 คลาสไดอะแกรม (Class Diagrams)

ลักษณะทั่วไปของ Class Diagrams คลาส (Class) เป็นองค์ประกอบที่สำคัญอย่างยิ่งสำหรับระบบงานเชิงวัตถุ (Object-Oriented System) คลาสเป็นการนำเอากลุ่มของออบเจ็กต์มาอธิบายความหมาย ออบเจ็กต์ซึ่งถูกจัดให้อยู่ในคลาสเดียวกันจะมีแอตทริบิวต์ โอเปอร์เรชั่น ความสัมพันธ์และความหมายบางอย่างเหมือนกัน โดยการจัดกลุ่มกันนี้สามารถทำได้ทั้งออบเจ็กต์ที่เป็นซอฟต์แวร์และฮาร์ดแวร์

การสร้างโมเดลให้แก่ระบบๆ หนึ่งจะเป็นการอธิบายถึงสิ่งต่างๆ ในมุมมองที่สนใจ โดยสิ่งที่จะอธิบายนั้นจะมีการสร้าง Vocabulary ของระบบขึ้นมา ยกตัวอย่างเช่น “ถ้าจะสร้างตึกสัก

หลัง” สิ่งที่จะต้องอธิบายอาจจะเป็น กำแพง ประตู หน้าต่าง เป็นต้น ซึ่งสิ่งของดังกล่าวบางอย่างอาจอยู่รวมกันเป็นหมวดหมู่เดียวกันก็ได้และบางชิ้นก็อยู่แยกออกไปต่างหาก

การกำหนดคลาสจะแทนด้วยสัญลักษณ์สี่เหลี่ยมผืนผ้า โดยแบ่งเป็น 3 ส่วน คือ ส่วนชื่อของคลาส แอตทริบิวต์และโอเปอร์เรชั่น ดังรูปที่ 2.14



รูปที่ 2.14 แสดงสัญลักษณ์การกำหนด Class

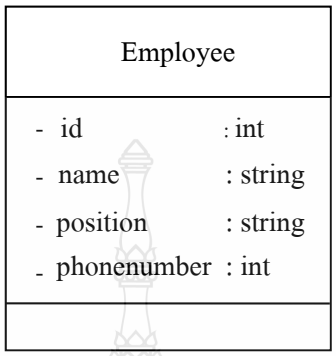
การจะจัดออบเจ็กต์ใดให้อยู่ในคลาสเดียวกันนั้น ขึ้นอยู่กับผู้ออกแบบระบบว่าจะมองที่คุณสมบัติ คือแอตทริบิวต์หรือพฤติกรรมใดเป็นหลัก เช่น ถ้ามองว่ากำแพงและประตูมีคุณสมบัติคือความกว้าง ความสูงและเป็นวัตถุที่แข็งเหมือนกัน ก็อาจจะจัดกำแพงและประตูไว้ในคลาสเดียวกัน แต่ถ้ามองถึงลักษณะของการใช้งานว่าประตูและหน้าต่างสามารถ เปิด - ปิดได้ ก็อาจจะจัดประตูและหน้าต่างเอาไว้ในคลาสเดียวกัน

เมื่อจัดออบเจ็กต์ต่างๆ ให้อยู่ในคลาสฯ หนึ่งแล้ว กลไกการใช้งานจริงๆ จะมีการสร้างอินสแตนซ์ (Instance) ขึ้นมาเป็นตัวแทนของคลาส ไม่ได้เรียกใช้ออบเจ็กต์หรือคลาสนั้นตรงๆ เช่น ถ้าต้องการจะจัดการอะไรบางอย่างกับกำแพง จะต้องมีการสร้างอินสแตนซ์ของกำแพงขึ้น อาทิ “กำแพงที่อยู่ทางทิศใต้ของบ้าน” จากนั้นจึงจัดการกับอินสแตนซ์นั้นต่อไปตามความต้องการ

1) แอตทริบิวต์ (Attributes) เป็นการบอกถึงคุณสมบัติของคลาส คลาสฯหนึ่งอาจจะมีแอตทริบิวต์ได้ตั้งแต่ 1 ค่าขึ้นไปหรืออาจเป็นคลาสที่ไม่มีแอตทริบิวต์เลยก็ได้ แอตทริบิวต์จะแสดงถึงคุณสมบัติที่ออบเจ็กต์ซึ่งอยู่ในคลาสเดียวกันมีร่วมกันหรือใช้งานร่วมกัน เช่น คลาสของพนักงาน (Employee) จะมีแอตทริบิวต์เป็น id, name, position, department, phoneNumber เป็นต้น

การตั้งชื่อให้แก่แอตทริบิวต์สามารถใช้คำๆ เดียวหรือวลีสั้นๆ ก็ได้ ในลักษณะคล้ายกับการตั้งชื่อของคลาส ถ้าในกรณีที่เป็นคำๆ เดียวจะใช้ตัวอักษรเป็นตัวพิมพ์เล็กทั้งหมด “แต่ถ้าเป็น

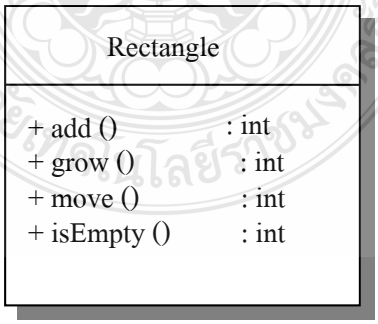
วลีที่มีมากกว่า 1 คำ” ตั้งแต่คำที่สองขึ้นไปจะให้ตัวอักษรตัวแรกของคำที่สองขึ้นไป ให้เป็นตัวพิมพ์ใหญ่ การแสดงแอตทริบิวต์ในคลาสจะเขียนในช่องที่ถัดจากชื่อของคลาส ดังรูปที่ 2.15



รูปที่ 2.15 การเขียน Attribute ใน Class

2) โอเปอร์เรชั่น (Operations) คือพฤติกรรมที่สามารถกระทำกับออบเจ็กต์ได้ โดยที่ออบเจ็กต์ทั้งหมดอยู่ในคลาสเดียวกันจะมีการใช้โอเปอร์เรชั่นของคลาสของตัวเองร่วมกัน คลาสหนึ่งๆ สามารถมีโอเปอร์เรชั่นได้มากกว่า 1 โอเปอร์เรชั่น โดยปกติแล้วการเรียกใช้โอเปอร์เรชั่นกับออบเจ็กต์หนึ่งๆ จะทำให้ข้อมูลเปลี่ยนแปลงไป

ในการวาดภาพคลาสจะเขียนส่วนของโอเปอร์เรชั่นถัดจากส่วนของแอตทริบิวต์ การตั้งชื่อให้แก่โอเปอร์เรชั่นจะทำในลักษณะเดียวกันกับชื่อของคลาส แต่ในทางปฏิบัตินิยมตั้งชื่อให้แก่โอเปอร์เรชั่นโดยใช้คำกิริยา ซึ่งอาจเป็นคำๆ อาจมีลักษณะดังรูปที่ 2.16

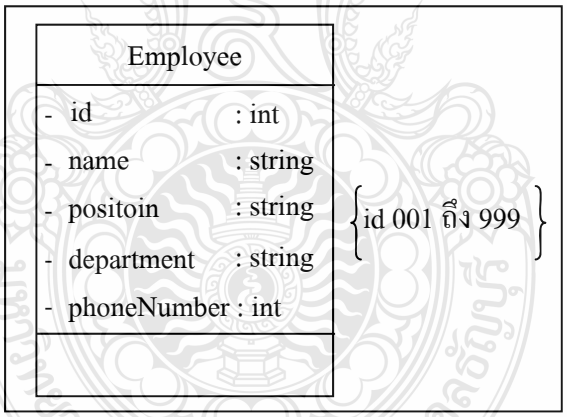


รูปที่ 2.16 การระบุพารามิเตอร์และประเภทของข้อมูลให้ Operation

ในการสร้างคลาส ไม่จำเป็นต้องมีแอตทริบิวต์และโอเปอร์เรชั่นทั้งหมดในคราวเดียวกันได้ เพราะว่าในทางปฏิบัติแล้ว คลาสหนึ่งๆ มักมีแอตทริบิวต์และโอเปอร์เรชั่นเป็นจำนวนมาก นั่นคือ สามารถเลือกได้ว่าจะแสดงแอตทริบิวต์และโอเปอร์เรชั่นเพียงบางตัวเท่านั้นหรือว่าเลือกที่จะไม่แสดงเลยก็ได้ ดังนั้น การที่รูปของคลาสใดไม่มีชื่อของแอตทริบิวต์หรือโอเปอร์เรชั่นแสดงอยู่ก็ไม่ได้หมายความว่าไม่มีชื่อดังกล่าวอยู่ในคลาสนั้น แต่อาจละไว้ไม่แสดงให้เห็นได้

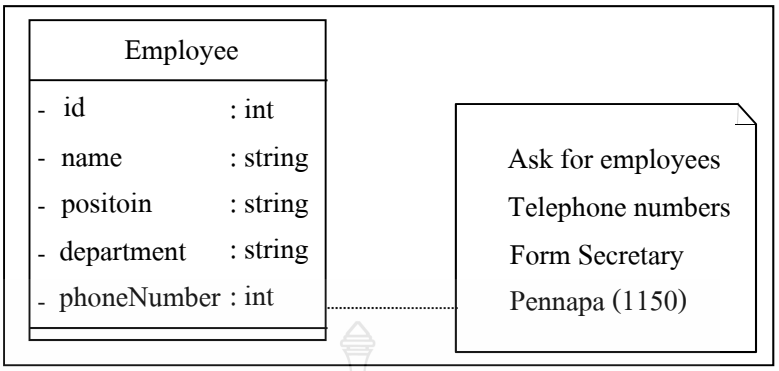
อย่างไรก็ตาม จำเป็นต้องมีการกำหนดสัญลักษณ์เพื่อให้ทราบว่ามีการละไม่แสดงชื่อของแอตทริบิวต์หรือโอเปอร์เรชั่นบางอย่างเอาไว้ ทั้งนี้เพื่อไม่ให้สับสนกับการที่คลาสไม่มีชื่อของแอตทริบิวต์หรือโอเปอร์เรชั่นบางอย่าง โดยจะใช้เครื่องหมาย “...” เป็นการบอกว่ามีการละไม่แสดงชื่อของแอตทริบิวต์หรือโอเปอร์เรชั่นที่ยาวมากๆ สามารถเลือกที่จะจัดกลุ่มให้แก่แอตทริบิวต์หรือโอเปอร์เรชั่นเหล่านั้นได้ โดยจะเรียกการจัดกลุ่มนั้นว่า “Stereotype”

3) การสร้างข้อบังคับให้แก่คลาส จะทำให้คลาสมีความชัดเจนมากขึ้น ซึ่งจะใช้กลไกที่ชื่อว่า Constrains มาอธิบายคลาสให้มีความชัดเจนยิ่งขึ้น การใช้ Constrains จะเขียนอยู่ภายในเครื่องหมายวงเล็บปีกกา ดังรูปที่ 2.17



รูปที่ 2.17 แสดงการสร้างข้อมูลระดับ Constrains เพิ่มเติมให้ Class

4) การเขียนหมายเหตุให้แก่คลาส Attached Notes หรือ Note เป็นกลไกของภาษา UML ที่ใช้ในการให้คำอธิบายแก่คลาสในลักษณะข้อมูลเพิ่มเติมหรือโน้ตย่อ เป็นเพียงข้อมูลเสริมไม่ใช่ข้อมูลที่จำเป็นจริงๆ จะใส่หรือไม่ใส่ก็ได้ ยกตัวอย่างเช่น จากคลาส Employee อาจมีการนำ Attached Notes มาใช้ดังรูป เป็นการอธิบายเพิ่มเติมว่าสามารถสอบถามหมายเลขโทรศัพท์ของพนักงานได้โดยสอบถามจากเลขาชื่อ “Pennapa” เบอร์ 1150 ดังแสดงในรูปที่ 2.18

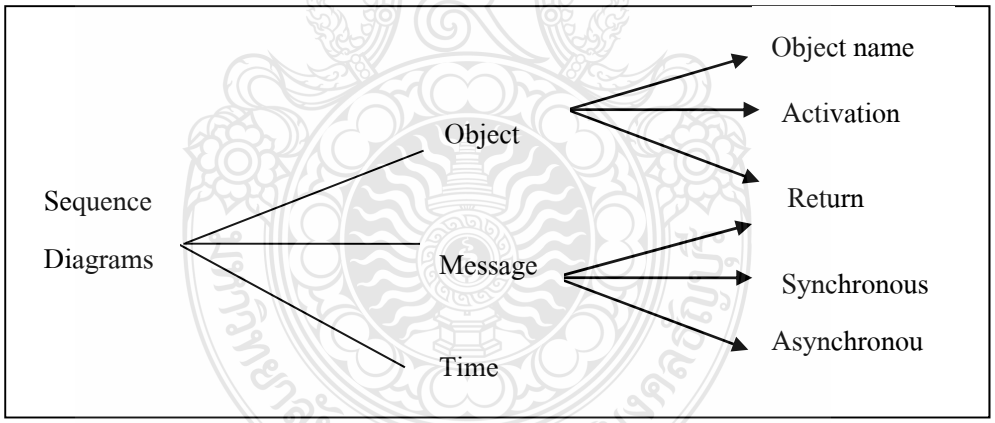


รูปที่ 2.18 ตัวอย่างการเขียนหมายเหตุให้แก่ Class

2.3.3 แผนภาพลำดับ (Sequence Diagrams)

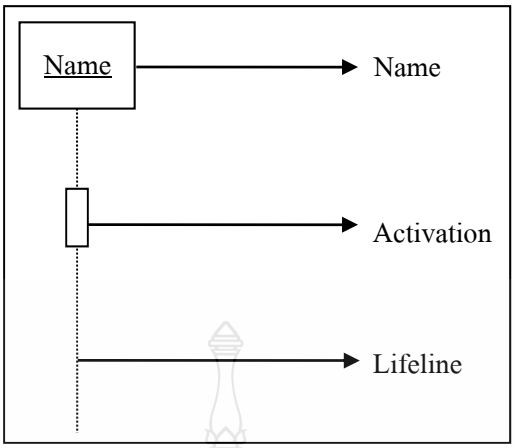
Sequence Diagrams จะแสดงให้เห็นว่าออบเจ็กต์ต่างๆในระบบงานหนึ่งมีการติดต่อสื่อสารกันอย่างไร ณ เวลาหนึ่ง

โครงสร้างของ Sequence Diagrams โดยรวม สามารถสรุปได้ดังนี้



รูปที่ 2.19 แสดงโครงสร้างของ Sequence Diagrams

1) ออบเจ็กต์ (Objects) จะประกอบด้วย 3 ส่วน คือ Object Name, Lifeline, Activationซึ่งองค์ประกอบทั้ง 3 สามารถแสดงเป็นสัญลักษณ์ได้ดังรูปที่ 2.20

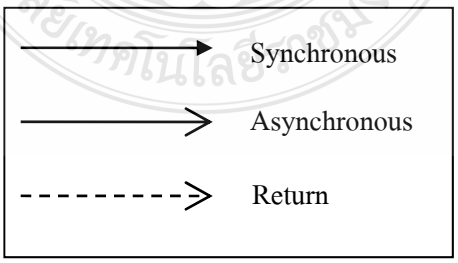


รูปที่ 2.20 แสดงองค์ประกอบทั้งหมดเป็นสัญลักษณ์

โดยที่ Name จะเป็นส่วนที่บอกถึงชื่อของออบเจ็กต์ว่าเป็นออบเจ็กต์อะไร โดยเรียงจากซ้ายไปขวาตามลำดับการทำงานของระบบ กล่าวคือ ออบเจ็กต์ที่อยู่ทางซ้ายมือจะทำงานก่อนออบเจ็กต์ที่อยู่ทางขวามือ ส่วนที่เป็นเส้นประที่ลากในแนวดิ่งจากออบเจ็กต์จะเรียกว่า Lifeline และสี่เหลี่ยมเล็กๆที่อยู่บนเส้น Lifeline จะเรียกว่า Activation ซึ่ง Activation จะแทนการทำงานต่างๆ ที่ออบเจ็กต์ของ Activation นั้นต้องกระทำนอกจากนี้ความยาวของรูปที่ใช้แทน Activation ยังบ่งบอกถึงระยะเวลาของการทำงานของออบเจ็กต์ด้วย

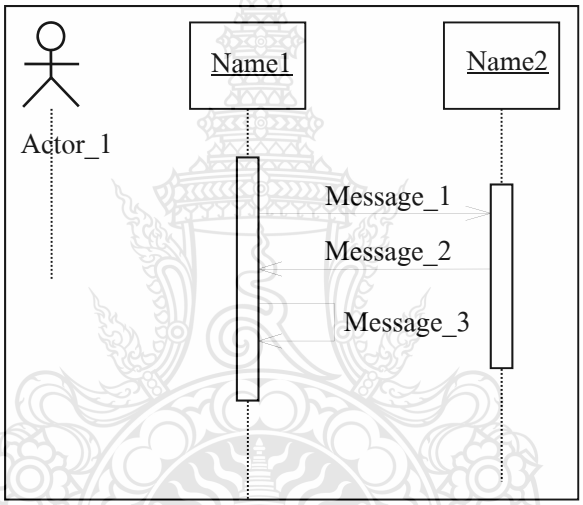
2) แมสเสจ (Message) เป็นการติดต่อที่ส่งจากออบเจ็กต์หนึ่งไปยังอีกออบเจ็กต์หนึ่ง หรืออาจจะส่งกลับมาหาตัวเองก็ได้ โดยที่จะแบ่งการติดต่อออกเป็น 3 แบบ คือ SynchronousAsynchronous และ Return

สัญลักษณ์ที่ใช้แทน Message ทั้ง 3 แบบ แสดงดังรูปที่ 2.21



รูปที่ 2.21 สัญลักษณ์แสดงรูปแบบการติดตั้งทั้ง 3 แบบของ Message

3) ไทม์หรือช่วงเวลา (Time) ลักษณะของไทม์หรือการแสดงเวลาของ Sequence Diagrams นั้นจะเป็นลักษณะแนวตั้ง คือจากบนลงล่าง Message ที่อยู่ด้านบนจะเป็นส่วนที่เกิดขึ้นก่อน Message ที่อยู่ด้านล่าง ลักษณะของการแสดงเวลาของ Sequence Diagrams จะมีลักษณะดังรูปจะเป็นการแสดงลำดับของเหตุการณ์ที่จะเกิดขึ้น เมื่อ Actor ทำงาน (Activation) กับ ออบเจ็กต์ Name1 ออบเจ็กต์ Name1 จะมีการทำงานโดยส่ง Message ไปยังออบเจ็กต์ Name2 และออบเจ็กต์ Name2 ส่ง Message กลับมายัง ออบเจ็กต์ Name1 หลังจากนั้นเมื่อ Message ที่ออบเจ็กต์ Name1 จะถูกส่งกลับมายังออบเจ็กต์ Name1 จะเป็นการสิ้นสุดการทำงานของไดอะแกรมตัวอย่างนี้แสดงดังรูปที่ 2.22



รูปที่ 2.22 แสดงลักษณะของการแสดงเวลาของ Sequence Diagrams

2.4 ทฤษฎีเว็บแอปพลิเคชั่น (Web Application)

ปัจจุบันไม่เพียงแค่ซอฟต์แวร์ของระบบงานทั่วไปที่ทำงานบนเครื่อง Standalone เท่านั้นที่ต้องการคุณภาพแต่ซอฟต์แวร์หรือระบบงานที่ทำงานบนเว็บ (Web-basedSystem/Application) ก็ต้องการคุณภาพเช่นกันเนื่องจากเว็บแอปพลิเคชั่นจะต้องโต้ตอบกับผู้ใช้ผ่านทางเว็บไซต์บนเครือข่ายอินเตอร์เน็ต ปัจจุบันความคาดหวังของผู้ใช้งานเว็บแอปพลิเคชั่นดังกล่าว ก็จะคล้ายกับความคาดหวังที่มีต่อซอฟต์แวร์บนเครื่อง Standalone กล่าวคือ เว็บแอปพลิเคชั่นจะต้องใช้งานง่าย น่าเชื่อถือ ทำงานเร็ว มีการรักษาความลับให้กับข้อมูลส่วนตัว และมีประสิทธิภาพเช่นเดียวกับซอฟต์แวร์ชนิดอื่นๆ คุณลักษณะเหล่านี้ก็คือ คุณลักษณะของ “คุณภาพ” นั่นเอง ดังนั้น จึงสามารถนำหลักการวิศวกรรมซอฟต์แวร์มาใช้กับการพัฒนาเว็บแอปพลิเคชั่นได้เช่นเดียวกัน

2.4.1 ทำความรู้จักกับวิศวกรรมเว็บ

วิศวกรรมเว็บ (WebEngineering) คือ การนำหลักการ ความรู้ และวิธีที่เป็นระบบทางด้านวิทยาศาสตร์ วิศวกรรมศาสตร์ และการบริหารงาน มาประยุกต์ใช้ในการพัฒนา การนำไปใช้งาน และการบำรุงรักษาระบบหรือแอปพลิเคชั่นที่ทำงานบนเว็บให้มีคุณภาพสูงสุด

จากอดีตจนถึงปัจจุบัน โครงสร้างพื้นฐานของเว็บแอปพลิเคชั่นมีความซับซ้อนขึ้นเรื่อยมา และปัจจุบันโครงสร้างดังกล่าวยังคงถูกพัฒนาศักยภาพมากขึ้นอย่างต่อเนื่อง เพื่อตอบสนองความต้องการของผู้ใช้และสร้างความพึงพอใจให้เกิดขึ้นแก่ผู้ใช้งานได้มากที่สุด สังเกตได้จากการเพิ่มจำนวนมากขึ้นของเทคโนโลยีเว็บ ที่ต่างก็มีวัตถุประสงค์ที่จะทำให้การใช้งานแอปพลิเคชั่นบนเว็บนั้น ให้ความรู้สึกเดียวกับการใช้งานแอปพลิเคชั่นบนเครื่องคอมพิวเตอร์แบบ Standalone ซึ่งก็คือ “การทำงานที่รวดเร็ว”แต่ในอีกมุมมองหนึ่ง เทคโนโลยีหรือโครงสร้างพื้นฐานเหล่านี้กลายเป็นปัจจัยสำคัญที่ทำให้โอกาสที่เว็บแอปพลิเคชั่นจะทำงานผิดพลาดหรือล้มเหลวเพิ่มสูงขึ้นเรื่อยๆ ดังนั้น เพื่อเป็นการหลีกเลี่ยงความเสี่ยงที่อาจจะเกิดขึ้นดังกล่าว และเพื่อให้การพัฒนาเว็บแอปพลิเคชั่นประสบความสำเร็จ จึงจำเป็นต้องมีการนำแนวทาง ระเบียบวิธีปฏิบัติและเครื่องมือใหม่ๆ เข้ามาใช้ในกระบวนการพัฒนา การนำไปใช้งาน และการบำรุงรักษาเว็บแอปพลิเคชั่นหรือระบบงานที่ทำงานบนเว็บให้มีคุณภาพขึ้นคือ“วิศวกรรมซอฟต์แวร์”

2.4.2 คุณลักษณะสำคัญของเว็บแอปพลิเคชั่น

ถึงแม้ว่าทีมงานพัฒนาเว็บแอปพลิเคชั่นจะสามารถนำหลักการวิศวกรรมซอฟต์แวร์มาใช้ได้เกือบทั้งหมดแต่ทีมงานต้องตระหนักถึงความแตกต่างของเว็บแอปพลิเคชั่นกับซอฟต์แวร์ที่ทำงานบนStandalone เพื่อให้การประยุกต์ใช้สมบูรณ์ยิ่งขึ้น คุณลักษณะที่สำคัญของเว็บแอปพลิเคชั่นที่แตกต่างจากแอปพลิเคชั่นทั่วไป มีดังนี้

1) เครือข่าย (Network) เว็บแอปพลิเคชั่นจะต้องทำงานอยู่บนเครือข่ายโดยเฉพาะเครือข่ายอินเตอร์เน็ต ซึ่งมีใช้บริการหลายกลุ่ม นอกจากเครือข่ายอินเตอร์เน็ตแล้ว เว็บแอปพลิเคชั่นยังสามารถทำงานอยู่ในเครือข่ายอินทราเน็ต ซึ่งเป็นเครือข่ายเชื่อมโยงภายในองค์กรอีกด้วย

2) การทำงานพร้อมกันของผู้ใช้หลายคน (Concurrency) เว็บแอปพลิเคชั่นต้องให้บริการผู้ใช้หลายคนพร้อมกันในเวลาเดียวกัน ซึ่งพฤติกรรมการใช้งานของผู้ใช้แต่ละคนย่อมแตกต่างกัน

3) ไม่สามารถคาดการณ์ปริมาณการใช้งานได้ (Unpredicted Load) ในแต่ละวันจำนวนผู้เข้ามาใช้งานเว็บแอปพลิเคชั่นมีจำนวนไม่เท่ากัน โดยทีมงานไม่สามารถคาดการณ์ได้

4) ประสิทธิภาพ (Performance) ในที่นี้คือความเร็วในการประมวลผลของเว็บแอปพลิเคชั่น โดยการที่แอปพลิเคชั่นทำงานช้านั้น ถือว่าเป็นปัจจัยสำคัญที่สุดที่จะทำให้ผู้ตัดสินใจละทิ้งเว็บไซต์นั้นไป

5) ความพร้อมในการใช้งาน (Availability) เว็บแอปพลิเคชั่นจะต้องพร้อมทำงานได้ตลอดเวลา โดยไม่จำกัดว่าจะเป็นช่วงเวลาใดหรืออยู่ในประเทศใด เนื่องจากการเข้าใช้เว็บไซต์นั้นสามารถทำได้ตลอดเวลาผ่านเครือข่ายอินเตอร์เน็ตที่เชื่อมโยงไปถึงทั่วทุกมุมโลก

6) ข้อมูล (Data) แม้ว่าหน้าที่พื้นฐานของเว็บแอปพลิเคชั่นคือ การนำเสนอข้อมูลในรูปแบบต่างๆ แต่หน้าที่นอกเหนือจากนั้น คือ การอนุญาตให้ผู้ใช้เข้าถึงฐานข้อมูลในฐานข้อมูลของเว็บซึ่งถูกจัดเก็บไว้บนเครือข่ายอินเตอร์เน็ตเช่นกัน

7) เนื้อหา (Content) เนื้อหาคือ ส่วนสำคัญของเว็บแอปพลิเคชั่นโดยความสวยงามและคุณภาพของเนื้อหาเป็นคุณลักษณะหนึ่งที่บ่งบอกถึงคุณภาพของเว็บแอปพลิเคชั่น

8) ความเร่งด่วน (Immediacy) เว็บแอปพลิเคชั่นต้องการการพัฒนาที่รวดเร็วเพื่อเปิดตัวใช้งานก่อนคู่แข่งขัน

9) ความปลอดภัย (Security) เนื่องจากเว็บแอปพลิเคชั่นสามารถเข้าถึงได้โดยการผ่านเครือข่ายอินเตอร์เน็ตที่ผู้ใช้ทุกคนสามารถเข้ามาใช้งานได้ เพื่อเป็นการป้องกันการขโมยข้อมูล เว็บแอปพลิเคชั่นจำเป็นต้องมีระบบรักษาความปลอดภัยให้กับข้อมูลของบริษัทและข้อมูลส่วนตัวของลูกค้า จะช่วยสร้างความน่าเชื่อถือและความไว้วางใจให้เกิดขึ้นกับผู้ใช้

10) ความสวยงาม (Aesthetic) เว็บแอปพลิเคชั่นจำเป็นต้องถูกออกแบบให้มีความสวยงามและน่าสนใจเพื่อดึงดูดลูกค้าให้เข้ามาใช้บริการ โดยเฉพาะอย่างยิ่งเว็บไซต์ขายสินค้า

11) การพัฒนาอย่างต่อเนื่อง (Continuous Evolution) เว็บแอปพลิเคชั่นจะต้องได้รับการปรับปรุงอย่างต่อเนื่องตลอดเวลา โดยเฉพาะการปรับปรุงเนื้อหาของเว็บ โปรโมชั่นใหม่ และสินค้าใหม่

2.4.3 การเริ่มต้นโครงการพัฒนาเว็บแอปพลิเคชั่น

ปัจจุบัน ไม่ว่าจะเป็นการพัฒนาแอปพลิเคชั่นขนาดเล็กหรือขนาดใหญ่เพียงใดก็ตามจำเป็นต้องมีการวิเคราะห์ถึงปัญหาที่เกิดขึ้นและความต้องการแอปพลิเคชั่นที่แท้จริงอย่างละเอียดรอบคอบ ทั้งนี้ ก็เพื่อไม่ให้เงินลงทุนนั้นสูญเปล่า และเพื่อเพิ่มผลกำไรให้กับบริษัทให้มากที่สุด การพัฒนาแอปพลิเคชั่นก็เช่นเดียวกัน จำเป็นต้องมีการวิเคราะห์ถึงปัญหาที่เกิดขึ้น วิธีแก้ไขปัญหาและความต้องการเว็บแอปพลิเคชั่นที่แท้จริงก่อนลงมือดำเนินการ เพื่อให้เว็บแอปพลิเคชั่นที่ผลิตขึ้นมานั้นสามารถสร้างผลกำไร และไม่ก่อให้เกิดปัญหาต่างๆ ตามมา ดังนั้น ในการเริ่มต้นดำเนินโครงการพัฒนาเว็บแอปพลิเคชั่นจึงควรเริ่มต้นด้วยกิจกรรมสำคัญ2กิจกรรมได้แก่ “กำหนดการ

เริ่มต้นพัฒนาเว็บแอปพลิเคชั่น” (Formulation) และการวางแผน (Planning)

2.4.4 การออกแบบเว็บแอปพลิเคชั่น

ไม่ว่าจะเป็นแอปพลิเคชั่นทั่วไปหรือเว็บแอปพลิเคชั่น ขั้นตอนสำคัญคือ ขั้นตอนการออกแบบ เนื่องจากในการออกแบบ ทีมงานจะต้องร่างแบบ ซึ่งก็คือ การสร้างแบบจำลองเพื่ออธิบายองค์ประกอบต่างๆ ขึ้นมาก่อน ทำให้สามารถตรวจสอบและประเมินคุณภาพของงานออกแบบได้ โดยเมื่อพบว่ามีข้อผิดพลาดในงานออกแบบส่วนใด ก็จะทำการแก้ไขให้ถูกต้องก่อนลงมือสร้าง นับว่าเป็นการลดข้อผิดพลาดให้น้อยลงได้อีกทางหนึ่ง

เพื่อให้การออกแบบเว็บแอปพลิเคชั่นมีคุณภาพตามลักษณะที่ได้กล่าวถึงในหัวข้อที่ผ่านๆ มาแล้ว ไม่ว่าจะเป็นเว็บแอปพลิเคชั่นประเภทใดก็ตาม สามารถยึดหลักการออกแบบ ที่ถูกแนะนำไว้ดังนี้

1) เรียบง่าย (Simplicity) เนื้อหาของเว็บเพจส่วนใหญ่ถูกนำเสนอด้วยสื่อในรูปแบบของภาพเคลื่อนไหวมากขึ้น อย่างไรก็ตาม ควรนำเสนอเนื้อหาด้วยสื่อในรูปแบบต่างๆ ที่เหมาะสมจะทำให้ดูเรียบง่าย ไม่รกจนเกินไป

2) สอดคล้อง (Consistency) ทุกองค์ประกอบบนเว็บแอปพลิเคชั่นจะต้องออกแบบให้สอดคล้องกัน เช่น การเลือกชนิดตัวอักษร ต้องเป็นไปในทิศทางเดียวกันทุกเพจ

3) เนื้อหาครบถ้วน (Robustness) เนื้อหาที่นำเสนอบนเว็บแอปพลิเคชั่นจะต้องเป็นสิ่งที่ผู้ใช้ต้องการอย่างแท้จริง

4) มีเส้นทางการเชื่อมโยงที่เข้าใจง่าย (Navigability) เส้นทางที่เชื่อมโยงควรทำให้ผู้ใช้เข้าใจได้ง่ายโดยไม่จำเป็นต้องมีคำแนะนำการเชื่อมโยง

5) สวยงาม (VisualAppeal) สิ่งที่ดึงดูดความสนใจจากผู้ใช้ได้มากที่สุดของเว็บแอปพลิเคชั่นคือ “ความสวยงาม” ของเว็บ ที่เกิดจากการจัดวางองค์ประกอบได้อย่างลงตัว อย่างไรก็ตาม ความสวยงามไม่ใช่ปัจจัยเดียวที่ ดึงดูดความสนใจของผู้ใช้ ประสิทธิภาพในการทำงานของเว็บก็เป็นอีกปัจจัยหนึ่งที่ไม่ควรมองข้าม

2.4.5 การทดสอบเว็บแอปพลิเคชั่น

การทดสอบเว็บแอปพลิเคชั่นจะใช้หลักการ เทคนิค และวิธีทดสอบเช่นเดียวกับแอปพลิเคชั่นหรือซอฟต์แวร์ทั่วไป ทั้งนี้ เพื่อเป็นการค้นหาข้อผิดพลาดต่างๆ ที่จะเกิดขึ้นในทุกส่วนของเว็บแอปพลิเคชั่นแล้วแก้ไขให้ถูกต้อง ก่อนที่จะกลายเป็นความล้มเหลวเมื่อนำไปติดตั้งใช้งานจริง

การทดสอบเว็บแอปพลิเคชั่นที่ดี คือ การที่ทีมงานจะต้องค้นหาข้อผิดพลาดให้มากที่สุด นั่นคือ ทีมงานต้องค้นหาข้อผิดพลาดจากทุกส่วนหรือทุกองค์ประกอบของเว็บแอปพลิเคชั่น

โดยเฉพาะการทดสอบกับสภาพแวดล้อมอื่นๆ เช่น เว็บบราวเซอร์ เซิร์ฟเวอร์ ระบบปฏิบัติการ และอุปกรณ์สื่อสาร เป็นต้น เนื่องจากทีมงานไม่สามารถควบคุมสภาพแวดล้อมดังกล่าวได้ จึงเป็นปัจจัยที่มีผลต่อการทำงานของเว็บแอปพลิเคชั่นอย่างมาก

คุณลักษณะสำคัญของเว็บแอปพลิเคชั่นที่ต้องทำการทดสอบ คือ ทดสอบเนื้อหา (Content Testing) การทดสอบเนื้อหาหรือการทบทวนเนื้อหาที่จะนำเสนอบนแอปพลิเคชั่น เป็นการค้นหาข้อผิดพลาดที่เกิดขึ้นในเนื้อหาทั้งหมด โดยทีมงานจะต้องตรวจสอบสิ่งต่างๆ ภายในเนื้อหา ดังนี้

1) พิสูจน์ตัวอักษร เช่น พิมพ์ผิด สะกดคำผิด

2) ตรวจสอบการใช้ไวยากรณ์

3) เนื้อหาที่อ้างอิงถึงไม่สอดคล้องกัน

4) ข้อผิดพลาดของการนำเสนอเนื้อหาในรูปแบบต่างๆ เช่น รูปภาพ เสียง วิดีโอคลิป ภาพเคลื่อนไหวเป็นต้น

5) ตรวจสอบเนื้อหาที่มาจากแหล่งข้อมูลอื่น เพื่อป้องกันไม่ให้เกิดปัญหาด้านลิขสิทธิ์

6) ตรวจสอบความถูกต้องข้อมูลที่อยู่ในฐานข้อมูล และข้อมูลที่มีการเปลี่ยนแปลง

2.4.6 การทดสอบโปรแกรม

การทดสอบแบบกล่องดำ (BlackBox Testing) บางครั้งเรียกว่า “การทดสอบเชิงพฤติกรรม” (Behavioral Testing) เนื่องจากเป็นการทดสอบผลการทำงานของซอฟต์แวร์ในแต่ละหน้าที่ตามข้อกำหนดความต้องการเท่านั้น เพื่อดูว่าซอฟต์แวร์ทำงานได้ถูกต้องตามที่กำหนดไว้หรือไม่โดยไม่คำนึงถึงคำสั่งภายใน

2.5 ทฤษฎีการใช้งาน Regular Expression

Regular Expression เป็นการกำหนดรูปแบบเพื่อการค้นหาข้อความหรือตัวอักษรว่ามีอยู่ในข้อความที่กำหนดหรือไม่ เช่น อยากรู้ว่าข้อความที่มีคนกรอกแบบฟอร์มเข้ามาบนเว็บของเรา มีคำหยาบหรือไม่ เราก็จะใช้ Regular Expression เป็นตัวตรวจสอบ นอกจากจะใช้ตรวจสอบแล้ว ยังสั่งแก้ได้อีกด้วยเช่น จะแก้คำว่า ประสิทธิ์ เป็นคำว่า ประสาท ก็ทำได้ โดยไม่ต้องไปค้นหาเองแต่สั่งให้โปรแกรมค้นหา โดยใช้ Regular Expression แล้วแทนที่คำคำนั้นด้วยคำที่เราต้องการ

ตารางที่ 2.1 สัญลักษณ์ของ Regular Expression

สัญลักษณ์	คำอธิบาย
^	คำหรืออักษรที่อยู่หน้าเครื่องหมายนี้ต้องเป็นคำขึ้นต้นของข้อความที่นำมาตรวจสอบ เช่น “^การ” เป็นการกำหนดว่าคำที่นำมาตรวจสอบต้องขึ้นต้นด้วยคำว่า การ เช่น “การทำดี” “การบ้าน” เป็นต้นคำที่ขึ้นต้นด้วย “การ” จะผ่านการทดสอบ
$	คำ/อักษรที่อยู่หน้าเครื่องหมายนี้ ต้องอยู่ตอนท้ายของข้อความที่นำมาตรวจสอบเช่น “มา$” จะถือว่าคำต่อไปนี้ถูกตามเงื่อนไข “ตามา” “ขอขมา” หรือแม้แต่คำว่า “หมา” แต่คำว่า “ทำดี” จะไม่ผ่าน เพราะไม่ได้ลงท้ายด้วยคำว่า “มา” ตามเงื่อนไข
+	คำ/อักษรที่อยู่หน้าเครื่องหมายนี้ ต้องมีปรากฏในคำที่นำมาตรวจสอบ อย่างน้อย 1 ตัว เช่น “ท+” จะถือว่าคำต่อไปนี้ผ่านการตรวจสอบ เช่น “ทองจุล” “วันทนา” “ถนนหนทางทุกแห่ง”
?, *	คำ/อักษรที่อยู่หน้าเครื่องหมายนี้ อาจจะมีปรากฏในคำที่นำมาตรวจสอบหรือไม่ก็ได้ ถ้ามีจะมีกี่ตัวก็ได้ “ก?ข+$” หมายถึง อาจจะมีด้วยตัว ก และอักษรตัวสุดท้ายต้องมีตัว ข อย่างน้อย 1 ตัว (เครื่องหมาย + แสดงว่ามีอย่างน้อย 1 และ เครื่องหมาย $ แสดงว่าเป็นตัวสุดท้าย)
s	ช่องว่าง หรือ Whitespace
[ ]	ใช้ระบุตำแหน่งในคำว่า ในตำแหน่งนี้จะมีตัวอักษรอะไรได้บ้าง เช่น 1) “[นร]” เป็นการกำหนดว่า คำที่นำมาตรวจสอบ ต้องเป็นตัว น หรือ ตัว รเท่านั้นจึงจะผ่าน มีความหมายเช่นเดียวกับ “น|ร” 2) “[ก-ค]” เป็นการบอกว่า คำที่นำมาจะต้องเป็น ตัว ก ข ค เท่านั้น เช่นในกรณีเลขประจำตัวที่ขึ้นต้นด้วย ก ข หรือ ค เท่านั้นถ้าพิมพ์ตัวแรกเป็นตัวอักษรตัวอื่นก็แสดงว่าพิมพ์ผิด เราจะเขียนได้ดังนี้ ^[ก-ค]ไม่ว่าตัวอักษร หรือสัญลักษณ์ใด ๆ ที่อยู่ภายในเครื่องหมาย [ ]จะกลายเป็นสัญลักษณ์

ตารางที่ 2.1 สัญลักษณ์ของ Regular Expression (ต่อ)

สัญลักษณ์	คำอธิบาย
[ ]	“กข{2,} หมายถึงให้มีตัว ข อย่างน้อย 2 ตัว เช่น “กขขขข” “กข{3,5} หมายถึงให้มีตัว ข จำนวน 3-5 ตัวเท่านั้น คือ “กขขข” “กขขขข” และ “กขขขขข”
( )	ใช้รวมกลุ่มเข้าด้วยกันเป็นส่วนเดียวกัน เช่น “ก(ขค)*” หมายถึง ตัว ก และอาจจะตามด้วยตัว ขค หรือไม่มีตัว ขค ก็ได้ เครื่องหมาย * แสดงว่าจะมีหรือไม่ก็ได้ “ก(ขค){1,5}” หมายถึง ตัว ก แล้วจะตามด้วย ขค จำนวน 1-5 ชุด เช่น “กขคขคขค” หรือ “กขคขค” ก็ได้
|	เสนอทางเลือกอย่างใดอย่างหนึ่ง เช่น “การ|ความ” เป็นการบอกว่า จะใช้คำว่า การ หรือ ความ ก็ได้ “(ก|ขค)งจ” เช่น กงจ หรือ ขคงจ ก็ได้

2.6 โครสร้างฐานข้อมูล (Database schema)

Database schema คือ รายละเอียดในภาพรวมของฐานข้อมูล โดยโครงสร้างเปรียบเสมือนพิมพ์เขียวทางเทคนิคฐานข้อมูล โดยมีโครงสร้างอยู่ 3 ระดับ

2.6.1 ระดับภายนอกหรือ วิว (External Level หรือ View) เป็นระดับของข้อมูลที่เป็นภาพที่ผู้ใช้งานมองเห็น (View) เค้าร่างของข้อมูลในระดับนี้เกิดจากภาพและความต้องการของผู้ใช้งาน

2.6.2 ระดับแนวคิด (Conceptual Level)

ประกอบด้วยเค้าร่างที่อธิบายถึง

1) บางครั้งเรียกว่า “ระดับตรรกะ”
2) ฐานข้อมูลมี Entity ใดบ้าง
3) โครงสร้างของข้อมูล
4) ข้อมูลเหล่านี้มีความสัมพันธ์กันอย่างไร
5) กฎเกณฑ์และข้อจำกัดต่างๆ
6) โดยจะผ่านการวิเคราะห์จาก นักวิเคราะห์และออกแบบระบบ และผู้บริหารฐานข้อมูล (DBA)

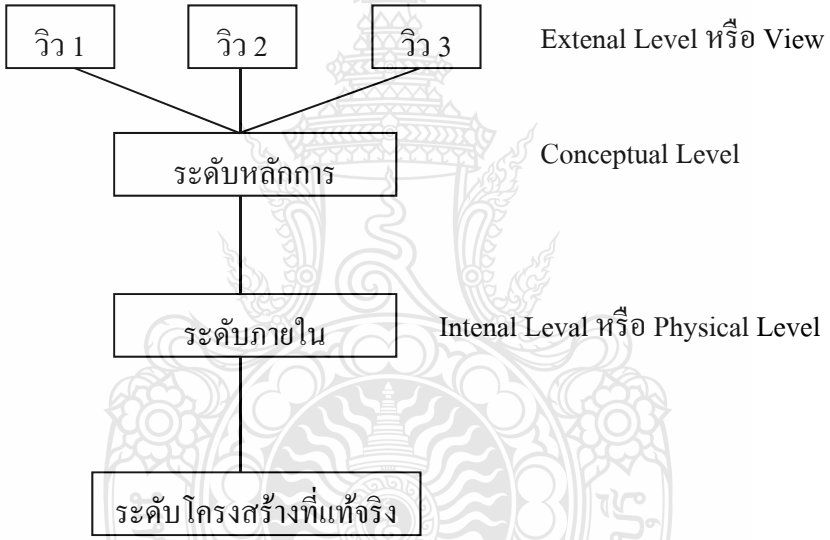
2.6.3 ระดับภายใน

บางครั้งเรียกว่า “ระดับกายภาพ”ประกอบด้วยเค้าร่างที่เกี่ยวกับการจัดเก็บข้อมูลจริงๆ ว่ามีโครงสร้างในการจัดเก็บอย่างไร รวมถึงวิธีการเข้าถึงข้อมูล

ความสัมพันธ์ระหว่างข้อมูลระดับต่างๆ ซึ่งเป็นการแปลความหมายจากระดับหนึ่งไปยังอีกระดับหนึ่ง เรียกว่า การแปลงส่ง (Mapping) โดยแบ่งเป็น 2 ลักษณะ

1) การแปลงส่งระหว่างระดับแนวคิดและระดับภายใน (Conceptual/Internal Mapping)

2) การแปลงส่งระหว่างระดับภายนอกและระดับแนวคิด (External/Conceptual Mapping)

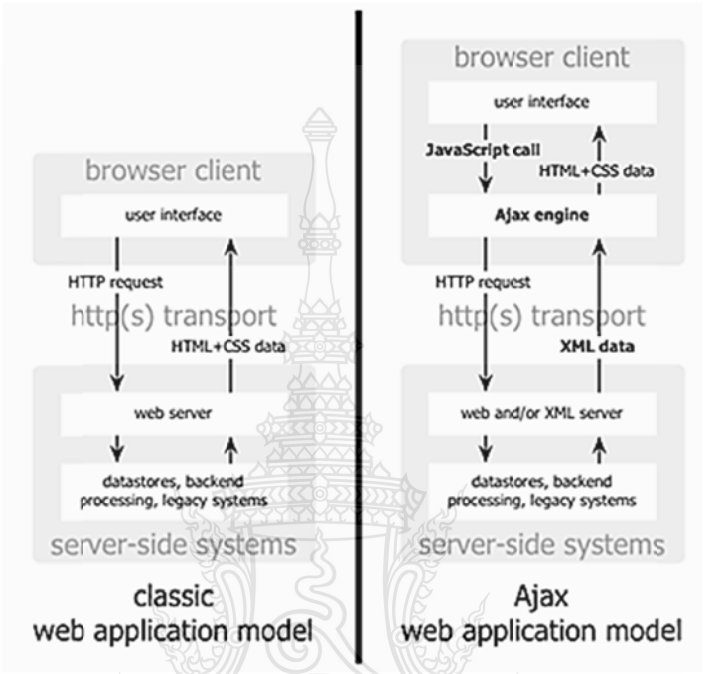


รูปที่ 2.23 โครงสร้างฐานข้อมูล

2.7 เอแจ็กซ์ (AJAX)

เอแจ็กซ์ (AJAX: Asynchronous JavaScript and XML)หมายถึงการทำงานร่วมกันของ JavaScript และ XML แบบ Asynchronous มีหลักการทำงาน 2 ประเด็น คือ การ Update หน้าจอแบบบางส่วน และการติดต่อสื่อสารกับ Server โดยใช้หลักการ Asynchronous ทำให้ผู้ใช้ไม่ต้องหยุดการทำงานเพื่อรอการประมวลผลจาก Server รวมถึงการโหลดและการรีเฟรชหน้าจอ ของบราวเซอร์ทางฝั่ง Client มีการใช้ AJAX โดยการเพิ่มเลเยอร์ระหว่าง User Browser กับ Server ทำให้ผู้ใช้สามารถทำงานได้โดยไม่ต้องรอให้ Client ติดต่อไปยัง Server รวมถึงการโหลดและการรีเฟรชหน้าจอทั้งหมดด้วย ดังนั้นผู้ใช้สามารถใช้งาน Application ได้อย่างมีประสิทธิภาพมากขึ้น

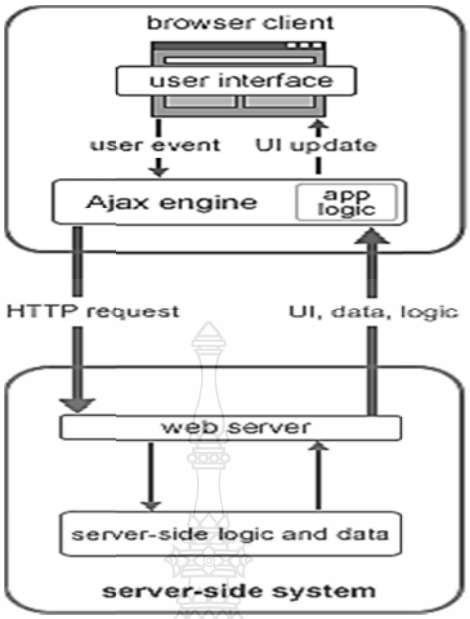
AJAX จึงไม่ใช่เทคโนโลยีในตัวของมันเอง แต่ว่าเป็นการนำเทคโนโลยีหลายๆตัวมารวมกันเช่น JavaScript, DHTML, XML, CSS, Dom และ XMLHTTPRequest



รูปที่ 2.24 เปรียบเทียบการทำงานแบบเดิม กับ AJAX

2.7.1 โครงสร้างของ AJAX

มุมมองของโครงสร้างทาง Software ของ AJAX ต่างจากเว็บแอพลิเคชันในทุกวันนี้ เนื่องจากมีการเพิ่ม Engine ทางฝั่ง Client



รูปที่ 2.25 แสดงโครงสร้างของ AJAX

2.7.2 ข้อดีของ AJAX

1) ผู้ใช้ไม่ต้องหยุดรอคอยการประมวลของ Server เนื่องจากการติดต่อแบบ Asyncronous

2) ตอบสนองต่อผู้ใช้ได้อย่างรวดเร็วเนื่องจากการ Update แบบบางส่วน

3) รองรับกับบราวเซอร์หลักๆที่สามารถใช้ JavaScript ได้

4) ทำให้การประมวลผลที่ Server มีความรวดเร็วขึ้นเนื่องจากการประมวลผลที่ Server ลดลง

5) ไม่ต้องทำการติดตั้ง หรือใช้ Plugs-in

6) ไม่ยึดติดกับ Platform หรือภาษาที่ใช้ในการเขียนโปรแกรม

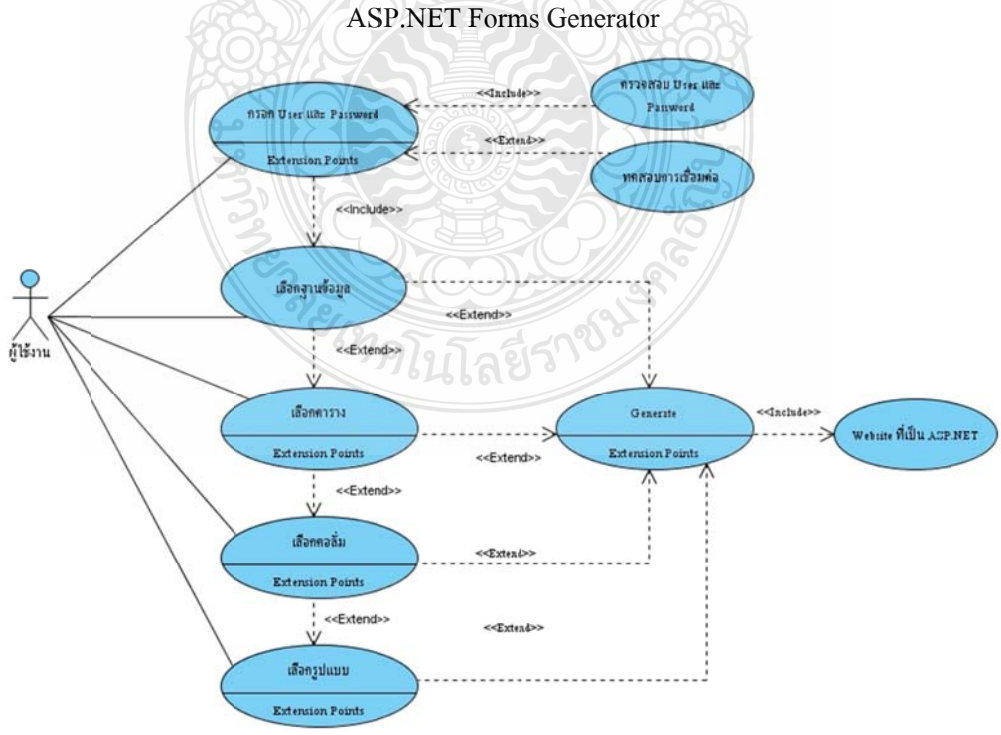
7) เป็นเทคโนโลยีใหม่ที่ไม่ได้เป็นของนักพัฒนาเว็บแอพพลิเคชั่นคนใดนั่นคือทุกคนมีสิทธิ์เข้ามาพัฒนาแอพพลิเคชั่นตัวนี้

บทที่ 3
วิธีการดำเนินงาน

โครงงานนี้ได้เพิ่มความสามารถของโปรแกรมในโครงงานเดิม อยู่สามส่วนดังนี้ ส่วนที่หนึ่งการตรวจสอบการ Update Database Schema เมื่อโครงสร้างฐานข้อมูลมีการเปลี่ยนแปลง ส่วนที่สองสามารถจัดรูปแบบฟอร์มข้อความและคอนโทรล จะมีผลในส่วนการเพิ่มข้อมูล และการแก้ไขข้อมูลเท่านั้น และส่วนที่สามได้นำเอาความสามารถของ AJAX (Asynchronous JavaScript and XML) มาช่วยทำให้ Web Application มีประสิทธิภาพมากขึ้นและงานต่อการใช้งานของผู้ใช้ และในโครงงานเดิมได้ทำการออกแบบไว้โดยใช้หลักการของการวิเคราะห์ และออกแบบระบบเชิงวัตถุ (Object Oriented Analysis) เป็นการมองจากภาพรวมของระบบว่ามีการทำงานอย่างไร และทำการแบ่งระบบออกเป็น Subsystem ย่อยๆ เพื่อทำการวิเคราะห์และออกแบบระบบ ซึ่งทำให้ง่ายต่อการนำมาพัฒนาต่อใน ASP.NET FORMS GENERATER VERSION 2 เป็นอย่างมาก

3.1 วิเคราะห์การทำงานของ ASP.NET Form Generator Version 1

3.1.1 Use Case Diagrams



รูปที่ 3.1 Use Case Diagrams ของระบบ ASP.NET Forms Generator

จากรูปที่ 3.1 แสดง Use Case Diagrams ของระบบ ซึ่งเป็น Use Case รวมทั้งระบบที่มีความสัมพันธ์กับผู้ใช้เพียงคนเดียว โดยผู้ใช้ในที่นี้หมายถึงผู้ใช้งานระบบ ในการทำงานของระบบเริ่มต้นจากผู้ใช้ทำการกรอก User และ Password แล้วทำการเชื่อมต่อฐานข้อมูล โดยจะมีกระบวนการตรวจสอบความถูกต้อง โดยจะตรวจทั้ง User และ Password และ ระบบยังมีกระบวนการทดสอบการเชื่อมต่อด้วย เมื่อทำการเชื่อมต่อกับฐานข้อมูลแล้วระบบจะทำการแสดงรายชื่อของฐานข้อมูล (Database) จากเซิร์ฟเวอร์ (Server) ที่เลือก เมื่อทำการเลือกฐานข้อมูล ระบบจะแสดงตาราง (Table) ที่มีอยู่ในฐานข้อมูลนั้นๆออกมา จากนั้นผู้ใช้จะต้องทำการเลือกตาราง โดยผู้ใช้งานจะเลือกทั้งหมด หรือเลือกกี่ตารางก็ได้ เมื่อเลือกตารางได้แล้ว ระบบก็จะแสดงคอลัมน์ที่มีในตารางที่เราทำการเลือกมา ผู้ใช้สามารถที่จะเลือกคอลัมน์ (Column) หรือไม่เลือกก็ได้ ซึ่งระบบทำการเลือกไว้ทั้งหมดอยู่แล้ว (Default) ต่อมาผู้ใช้ก็สามารถเลือกรูปแบบที่จะใช้ และยังสามารถแก้ไขในส่วนของ ธีม (Theme) ซึ่งแบ่งการทำงานออกเป็น 6 Use Case ย่อยได้ดังนี้

1) Use Case การรับรายละเอียดของการเชื่อมต่อกับฐานข้อมูลโดย Use Case นี้จะทำหน้าที่รับรายละเอียดการเชื่อมต่อฐานข้อมูล เพื่อนำไปตรวจเช็คสิทธิ์ในการเข้าใช้งานฐานข้อมูล

2) Use Case การตรวจสอบการเชื่อมต่อฐานข้อมูล โดยจะทำหน้าที่ตรวจสอบข้อมูลการเชื่อมต่อและแสดงผลให้ผู้ใช้งานทราบซึ่งผู้ใช้ก็สามารถเลือกว่าจะทดสอบหรือไม่ทดสอบการเชื่อมต่อก็ได้

3) Use Case การเลือกตารางและคอลัมน์ ทำหน้าที่รับข้อมูลตาราง จากฐานข้อมูลออกมาแสดงให้ผู้ใช้ทำการเลือก โดยจะเลือกกี่ตารางก็ได้ แล้วทำการเลือกคอลัมน์ ภายในตารางที่ได้เลือกไว้ก่อนหน้า เพื่อนำข้อมูลภายในคอลัมน์ไปใช้งานขั้นตอนต่อไป

4) Use Case การเลือกรูปแบบ ซึ่ง Use Case นี้ทำหน้าที่แก้ไขรูปแบบ โดยใช้ ธีม ทำให้ผู้ใช้สามารถเลือกรูปแบบได้ตามต้องการ และสามารถเพิ่มส่วนหัว (Header) ส่วนท้าย (Footer) ได้

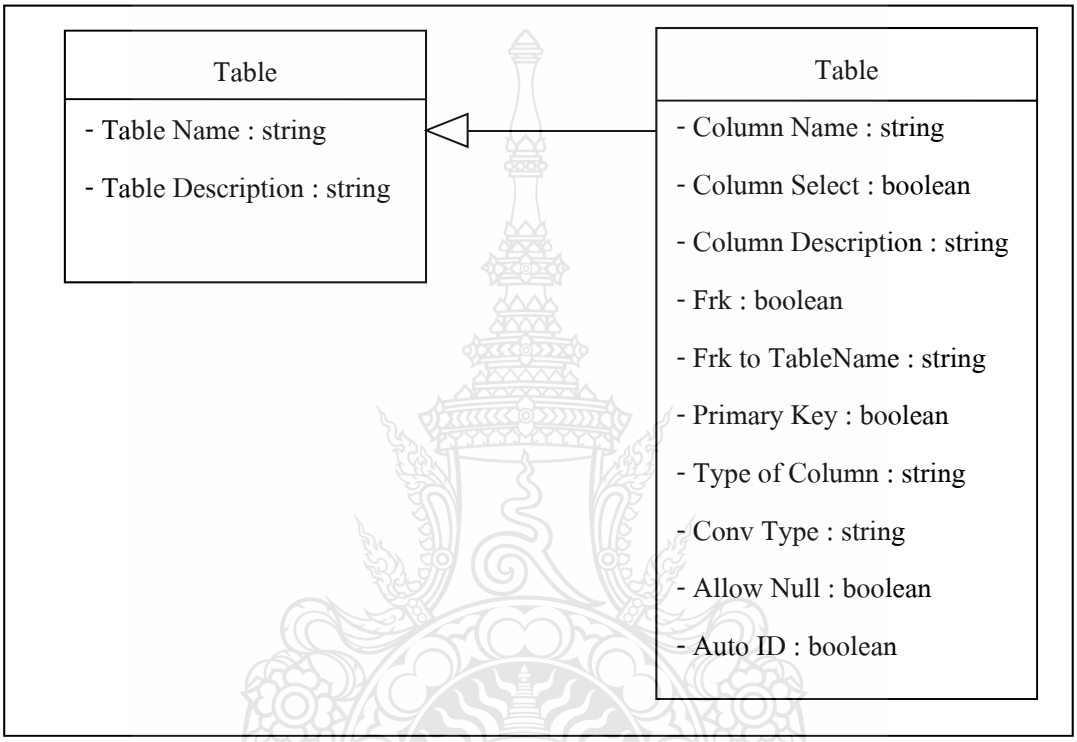
5) Use Case การแก้ไขรายละเอียดของคอลัมน์ ทำหน้าที่นำคอลัมน์ ที่ได้มาทำการแก้ไขรายละเอียดต่างๆก่อนนำไปใช้งาน

6) Use Case การ Generate ฟอร์ม ทำหน้าที่สร้างฟอร์มจากข้อมูลที่ได้ทำการบันทึกจาก Use Case การแก้ไขรายละเอียดคอลัมน์ออกมาเป็นโค้ดไฟล์ นามสกุล .aspx, .dbml, .config, .css และ .master บันทึกไว้ในไดเร็คทอรี่ที่ผู้ใช้งานต้องการ

3.1.2 ClassDiagrams

จากการวิเคราะห์ระบบมีการเก็บข้อมูลลงในภายใน Dataset และมี Class สำหรับเรียกใช้งานฟังก์ชันในการ Generate คือ GenClass

1) คลาสที่ใช้ในการเก็บข้อมูลของตารางที่ถูกเลือกนำไปใช้ในการ Generate



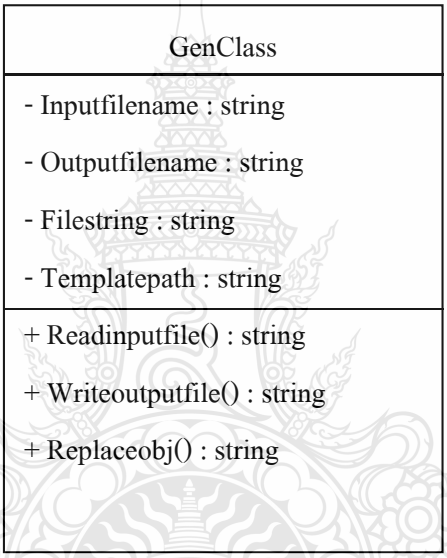
รูปที่ 3.2 แสดงคลาสที่ใช้ในการเก็บข้อมูลของระบบ ASP.NET Forms Generator

จากรูปที่ 3.2 เป็นคลาสที่ใช้ในการเก็บข้อมูลของตารางที่ถูกเลือกโดยจะเก็บข้อมูลต่างๆหลังจากการแก้ไขเสร็จเรียบร้อยมีค่าต่างๆ ดังนี้

- Table Name ชื่อตารางที่ถูกเลือกเพื่อทำการ Generate
- Table Description แสดงรายละเอียดของตาราง
- Column Name ชื่อคอลัมน์ทั้งหมดในตาราง
- ColumnSelect คอลัมน์ที่เลือก
- Column Description รายละเอียดของคอลัมน์
- Frk คอลัมน์ที่เป็น Forieng Key หรือคีย์สัมพันธ์
- Frk to Table ชื่อตารางที่มีความสัมพันธ์ไปยังตารางนั้นๆ

- Primary Key คอลัมน์ที่เป็น Primary Key หรือ คีย์หลัก
- Typ Of Column ชนิดของคอลัมน์จากฐานข้อมูล
- Conv Type แปลงชนิดของคอลัมน์เป็นชนิดตัวแปรในการเขียนโปรแกรม
- Allow Null แสดงว่าคอลัมน์ใดเป็น NULL
- Auto ID คอลัมน์ที่เป็น Auto ID

2) คลาสที่ใช้ในการ Generate



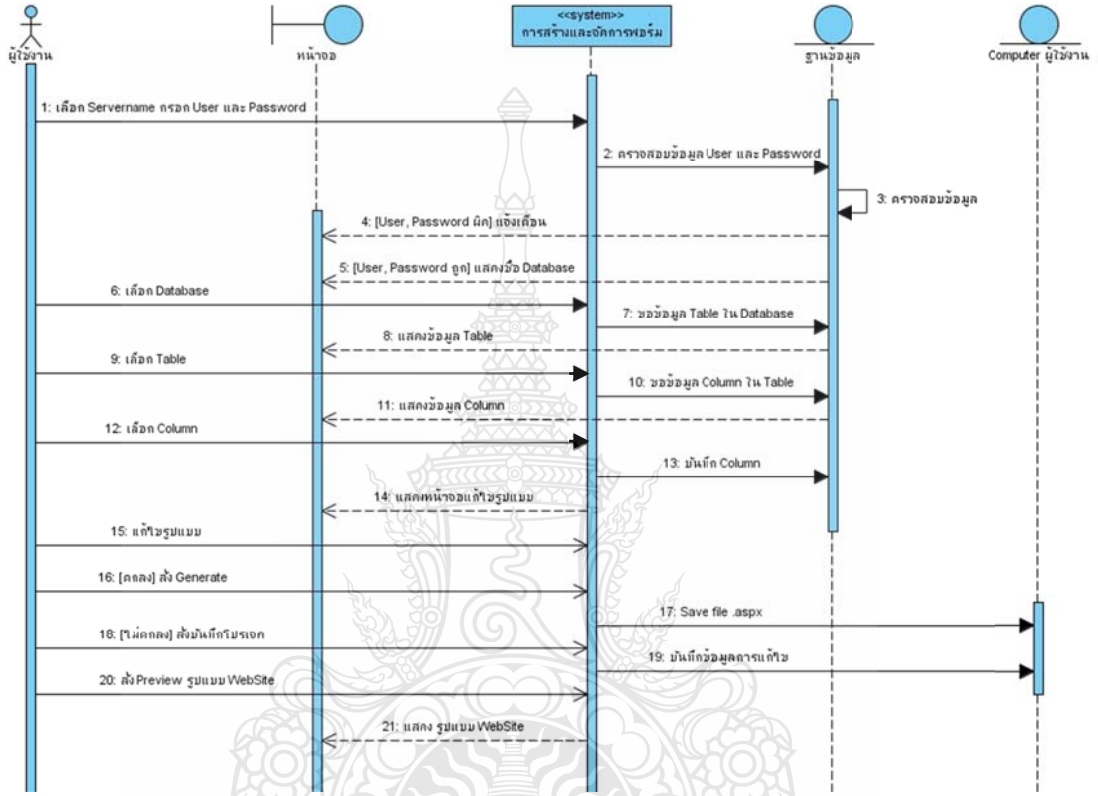
รูปที่ 3.3 แสดงคลาสที่ใช้ในการ Generate

จากรูปที่ 3.3 เป็นคลาสที่ใช้ในการ Generate โดยโปรแกรมจะเรียกใช้งานฟังก์ชันต่างๆ ภายในคลาส ซึ่งประกอบไปด้วย

- แอตทริบิวต์ Inputfilename ใช้เก็บชื่อไฟล์ของเทมเพลต
- แอตทริบิวต์ Outputfilename ใช้เก็บที่อยู่ของเอาต์พุตที่ต้องการสร้าง
- แอตทริบิวต์ Filestring ใช้เก็บชื่อไฟล์ของเอาต์พุตที่ต้องการสร้าง
- แอตทริบิวต์ Templatepath ใช้เก็บที่อยู่ของไฟล์เทมเพลต
- เมธอด Readinputfile() ใช้สำหรับอ่านข้อมูลภายในเทมเพลต
- เมธอด Writeoutputfile() ใช้สำหรับเขียนไฟล์เอาต์พุต
- เมธอด Replaceobj() ใช้สำหรับค้นหาและแก้ไขข้อมูล

3.1.3 Sequence Diagrams

ในแต่ละ Use Case จะมีกิจกรรมต่างๆที่เกิดขึ้นในระบบดังได้แสดงไว้ใน Sequence Diagrams ต่อไปนี้

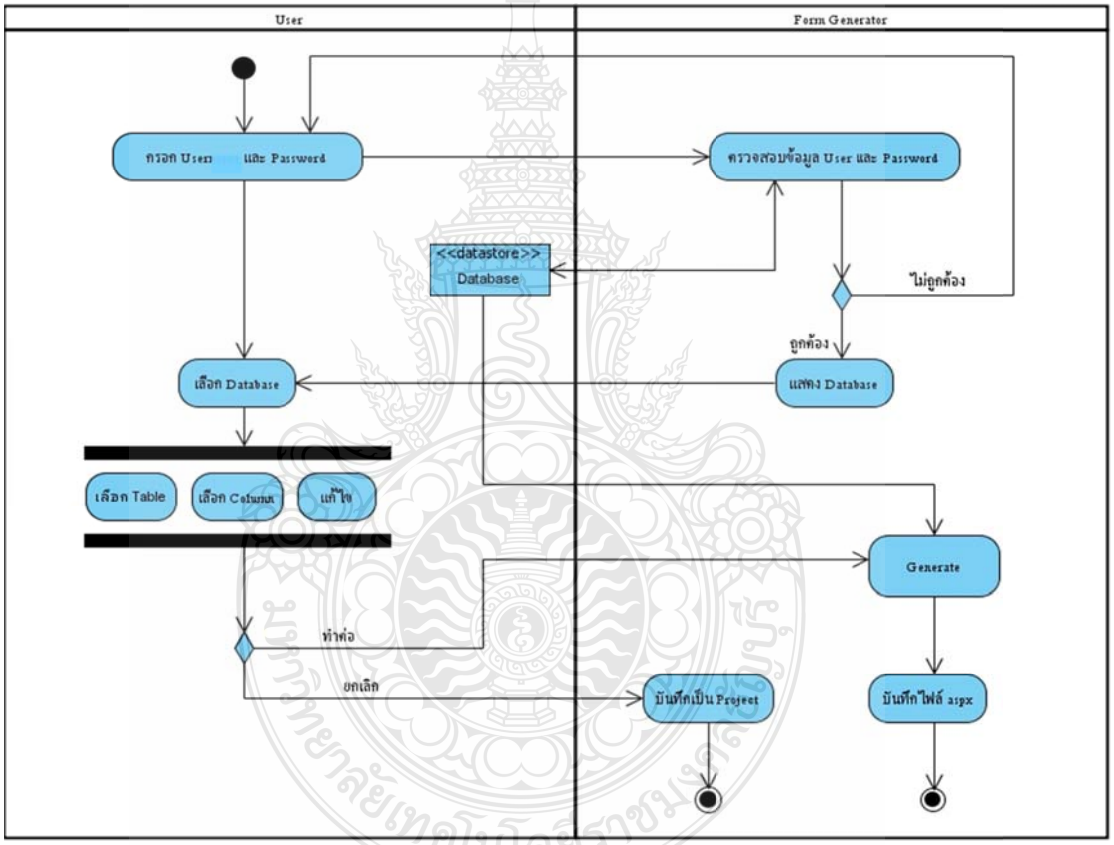


รูปที่ 3.4 Sequence Diagrams ของ Use Case รวมของระบบ ASP.NET Forms Generator

จากรูปที่ 3.4 เป็น Sequence Diagrams ของ Use Case ของระบบซึ่งจะเริ่มต้นจากผู้ใช้ทำการเลือกชื่อเซิร์ฟเวอร์ (Servername) และทำการกรอก User และ Password ระบบก็จะทำการตรวจสอบความถูกต้องของ User และ Password ในฐานข้อมูลก็จะมีการตรวจสอบ User และ Password ด้วยเช่นกัน ใน กรณี เลือกเป็น SQL Authentication หลังจากนั้น ถ้าเกิด User หรือ Password ผิดฐานข้อมูลก็จะทำการแจ้งเตือนแสดงที่หน้าจอแสดงผล ถ้าเกิด User และ Password ถูกต้อง ระบบก็จะทำการขอข้อมูลจากฐานข้อมูลเพื่อนำมาแสดงตาราง ทั้งหมด หลังจากนั้น ก็ให้ผู้ใช้ทำการเลือกตารางที่จะใช้ในการ Generate เมื่อทำการเลือกตารางเสร็จแล้วระบบก็จะทำการแสดงข้อมูลคอลัมน์จากฐานข้อมูล และ ให้ผู้ใช้ทำการเลือกคอลัมน์ ผู้ใช้ก็สามารถแก้ไขรายละเอียดของคอลัมน์ได้ ขั้นตอนต่อมาก็ทำการบันทึกคอลัมน์ที่แก้ไข เมื่อบันทึกเสร็จระบบก็จะแสดง

ตัวอย่างที่เราทำการแก้ไข แต่ไม่ใช่ตัวอย่างเว็บไซต์ ขั้นตอนต่อมาก็ให้ผู้ใช้เลือกรูปแบบซึ่งระบบจะตั้งค่าไว้แล้ว ถ้าหากผู้ใช้งานต้องการแก้ไขก็สามารถแก้ไขได้ เมื่อผู้ใช้งานทำการแก้ไขเสร็จแล้วก็ทำการ Generate ได้เลย หรือ ถ้ายังไม่ต้องการ Generate ก็สามารถบันทึกเป็น “โปรเจกต์ไฟล์” ไว้นำกลับมาแก้ไขและทำการ Generate ใหม่ได้ เมื่อทำการ Generate ผู้ใช้ก็สามารถเลือกได้ว่าต้องการที่จะนำไฟล์ที่ทำการ Generate ไปเก็บไว้ในโฟลเดอร์ใด หรือจะสร้างโฟลเดอร์ขึ้นใหม่ก็ได้

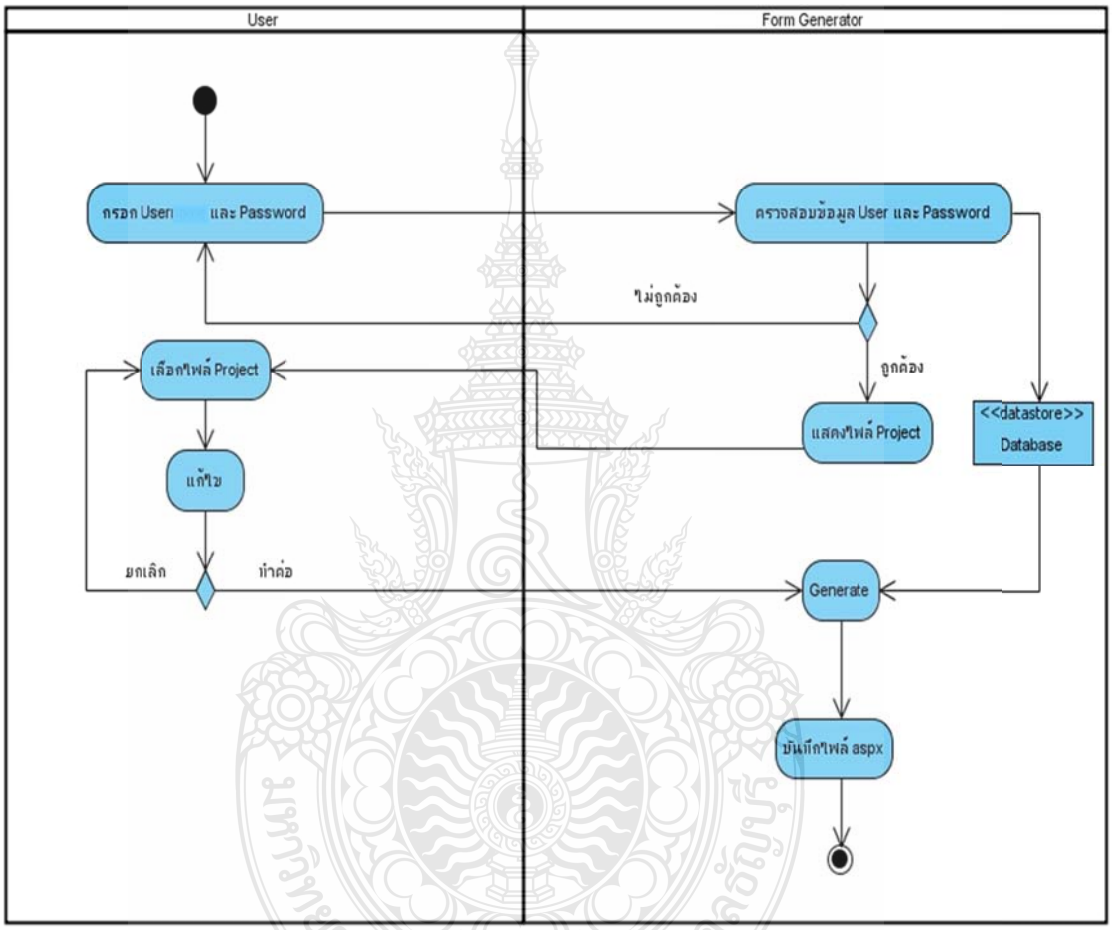
3.1.4 Activity Diagrams



รูปที่ 3.5 Activity Diagrams ของ Use Case รวมของระบบ ASP.NET Forms Generator

จากรูปที่ 3.5 จะแบ่งออกเป็น 2 อย่างคือ ผู้ใช้ กับ ระบบ Form Generator การทำงานจะเริ่มต้นจากผู้ใช้ทำการกรอกข้อมูล User และ Password ระบบ Form Generator ก็จะทำการตรวจสอบข้อมูล User และ Password เมื่อ User และ Password ผิด ระบบก็จะให้กรอกข้อมูลใหม่อีกครั้ง เมื่อ User และ Password ถูกต้องแล้วระบบก็จะทำการแสดงข้อมูลตารางจาก ฐานข้อมูล

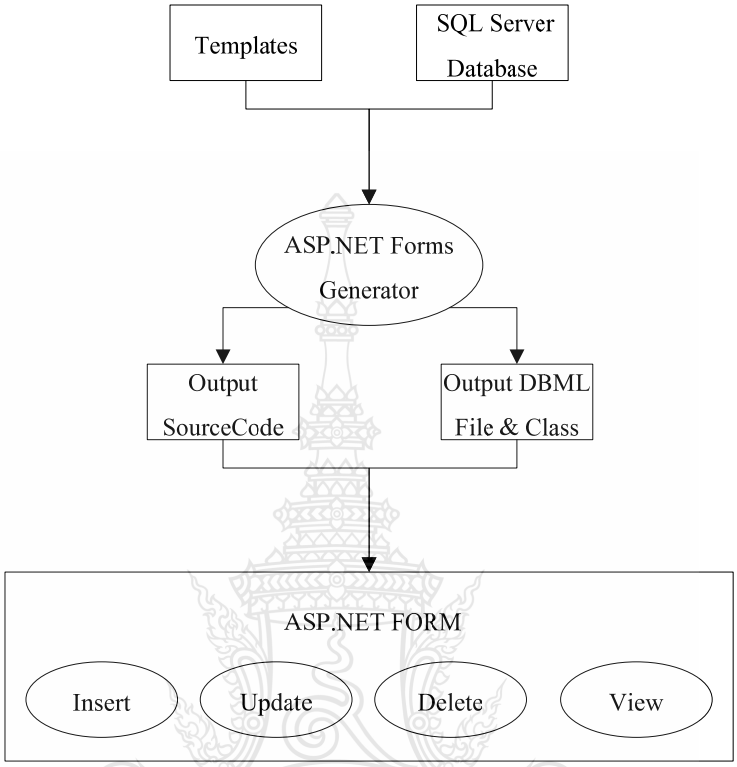
แสดงให้ผู้ใช้ทำการเลือกฐานข้อมูล ต่อมาผู้ใช้ก็จะทำการเลือกว่าจะใช้ตารางใดๆ ต่อมาทำการเลือกคอลัมน์ ใดบ้างในการ Generate อาจมีการแก้ไขธีม ต่อมาถ้าผู้ใช้ตกลง ก็ทำการ Generate ไฟล์ออกมาได้เป็นไฟล์เว็บไซต์ ASP.NET แต่ถ้าผู้ใช้ยังไม่ต้องการ Generate ก็สามารถบันทึกเป็นโปรเจกต์ไว้เพื่อสามารถนำกลับมาใช้แก้ไขและ Generate ในครั้งต่อไปได้



รูปที่ 3.6 Activity Diagrams ของการแก้ไขโปรเจกต์เดิม

ในกรณีที่บันทึกเป็นโปรเจกต์ไฟล์ไว้ ผู้ใช้ต้องการนำไฟล์โปรเจกต์ที่ได้บันทึกไว้มาทำการแก้ไขหรือทำการ Generate ใหม่ จะต้องเริ่มจากการเชื่อมต่อกับฐานข้อมูลใหม่อีกครั้ง โดยผู้ใช้จะต้องกรอก User และ Password ระบบก็จะทำการตรวจสอบข้อมูลการเชื่อมต่อ เมื่อถูกต้องแล้ว ผู้ใช้ก็จะทำการเลือกโปรเจกต์ไฟล์ที่ทำการบันทึกไว้ขึ้นมาทำการแก้ไขหรือ Generate เลยก็ได้ เมื่อทำการ Generate ก็จะได้ไฟล์ .aspx, .dbml, .config, .css และ .master

3.1.5 ผังการทำงานของระบบ ASP.NET Forms Generator



รูปที่ 3.7 ผังการทำงานของระบบโดยรวม

จากผังการทำงานโดยรวมของระบบจะเริ่มจากอ่านไฟล์เทมเพลตและเก็บรายละเอียดต่างๆ จากฐานข้อมูล แล้วทำการ Generate ออกมาจะได้ไฟล์ 2 ส่วนคือส่วนของ Source Code และส่วนของ DBML และ Class ไฟล์ที่ได้นั้นจะเป็น ASP.NET ซึ่งมีฟังก์ชั่นการทำงานคือ Insert, Update, Delete, View

3.1.6 ขั้นตอนการดำเนินงาน

1) แบ่งระบบออกเป็นโมดูลย่อยๆ ดังนี้
   - ขั้นตอนการเชื่อมต่อฐานข้อมูล
   - ขั้นตอนการเลือกตาราง
   - ขั้นตอนการเลือกและแก้ไขรายละเอียดคอลัมน์ (Column)
   - ขั้นตอนการเลือกและแก้ไขธีม (Theme)
   - ขั้นตอนการ Generate
2) ศึกษาโปรแกรมและภาษาที่ต้องในการจัดทำระบบดังนี้
   - โปรแกรม Microsoft Visual Studio ศึกษาพื้นฐานของการเขียนโปรแกรมด้วย Windows Form Application จากเครื่องมือต่างๆ และทดสอบการใช้เครื่องมือใน Visual Studio
   - ภาษา C# การสร้าง Windows Form Applications การสร้าง Class ในการเก็บข้อมูลด้วย Dataset
   - ภาษา ASP.NET 3.5 การสร้างเว็บไซต์ในการติดต่อกับฐานข้อมูล Insert Update Delete การ Validate Data Type การใช้งานเมนูใน Master Page
   - การติดต่อฐานข้อมูลด้วย SMO (SQL Server Management Object) การออกแบบการสร้างและจัดการฟอร์ม ASP.NET เมื่อเชื่อมต่อกับฐานข้อมูลแล้วจะต้องทำการ Query Schema ทั้งหมดของฐานข้อมูลเพื่อนำมาใช้ในการสร้างเว็บไซต์ ดังนั้นจึงต้องเชื่อมต่อกับฐานข้อมูลด้วย SMO
   - ระบบฐานข้อมูล Microsoft SQL Server เนื่องจากการสร้างและจัดการฟอร์ม ASP.NET สร้างเว็บไซต์จาก Microsoft SQL Server ดังนั้นต้องเข้าใจการทำงานและคุณสมบัติของ Microsoft SQL Serve
   - การใช้งาน SQLMetal ในการสร้าง LINQ to SQL Classes จะต้องมีการสร้าง Class Database เพื่อเชื่อมต่อกับฐานข้อมูลเป็นตัวการจัดการ Insert Update Delete และ View โดยใช้ SQLMetal.exe ไฟล์ที่ได้ออกมาจะเป็นไฟล์นามสกุล dbml และคลาส .designer.cs

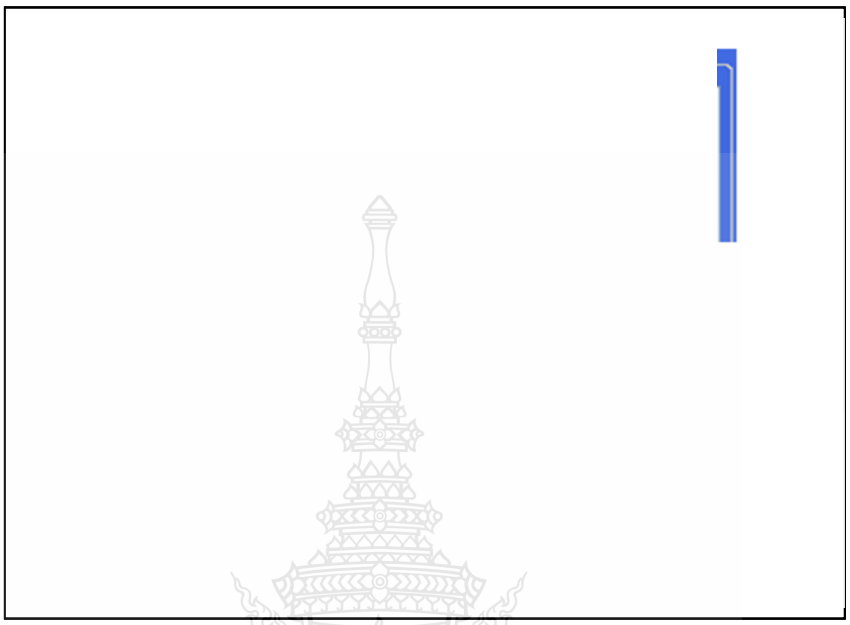
3) เขียนโปรแกรม

- เขียนโปรแกรมในส่วนการเชื่อมต่อกับฐานข้อมูลด้วย SMO ดังตัวอย่างการออกแบบและเขียนโปรแกรมดังรูปที่ 3.8

รูปที่ 3.8 วิธีการออกแบบหน้าจอการเชื่อมต่อฐานข้อมูลด้วย SMO

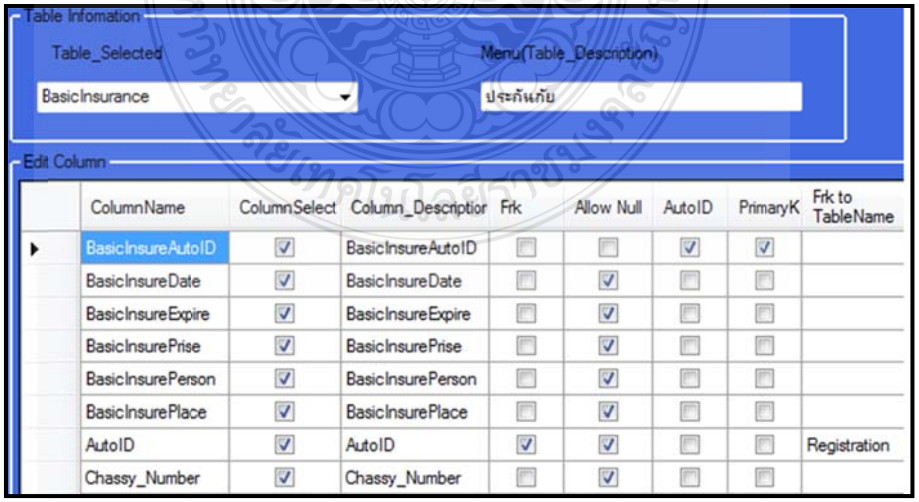
รูปที่ 3.9 Code Program ในการเชื่อมต่อกับฐานข้อมูลด้วย SMO

- เขียนโปรแกรมในส่วนของการเลือกตารางโดยใช้ ListBox ดังรูปที่ 3.10



รูปที่ 3.10 หน้าจอและตัวอย่าง Code Program ในการเลือกตาราง

- เขียนโปรแกรม Query ข้อมูลจากฐานข้อมูลลงใน Dataset แล้วทำการแสดงผลผ่านดาต้ากริดวิวโดย Query ข้อมูลจาก Class Dataset เป็นดังรูปที่ 3.11



ColumnName	ColumnSelect	Column_Description	Frk	Allow Null	AutoID	PrimaryK	Frk to TableName
BasicInsureAutoID	☑	BasicInsureAutoID	☐	☐	☑	☑	
BasicInsureDate	☑	BasicInsureDate	☐	☑	☐	☐	
BasicInsureExpire	☑	BasicInsureExpire	☐	☑	☐	☐	
BasicInsurePrise	☑	BasicInsurePrise	☐	☑	☐	☐	
BasicInsurePerson	☑	BasicInsurePerson	☐	☑	☐	☐	
BasicInsurePlace	☑	BasicInsurePlace	☐	☑	☐	☐	
AutoID	☑	AutoID	☑	☑	☐	☐	Registration
Chassy_Number	☑	Chassy_Number	☐	☑	☐	☐	

รูปที่ 3.11 การเพิ่มข้อมูลลง Dataset แล้วแสดงในดาต้ากริดวิว

- เขียนโปรแกรมในการเลือกธีมโดยสร้าง เทมเพลตไว้เป็น Default เมื่อผู้ใช้เลือกสีพื้นหลัง รูปภาพ และลักษณะของกริดวิว ก็ทำการ Replace ค่าต่างลงใน Skin File ตัวอย่างดังรูปที่ 3.12

```
Header                                          //file stytlesheet.txt
{position: relative;
background: #[:colour:];}
if (CBOBlackground.Text == "None")              //file Form1.cs
{ generate.ReplaceObj("colour", "ffffff");
generate.ReplaceObj("h1", "000000");
generate.ReplaceObj("header", "ffffff");
generate.ReplaceObj("tree", "000000");    }
```

รูปที่ 3.12 การ Replace สีพื้นหลังจากการเลือกของผู้ใช้งาน

- ออกแบบเทมเพลตโดยสร้างเทมเพลตจาก Code ASP.NET

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master"
AutoEventWireup="true" CodeFile="Login__View.aspx.cs" Title="ddd" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head"
Runat="Server"></asp:Content><asp:Content ID="Content2"
ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
<asp:Label ID="Label1" runat="server" Text="ใส่ข้อมูลที่ต้องการ
ค้นหา"BorderColor="Black" ForeColor="Black"></asp:Label>
 <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
 <asp:Button ID="Button1" runat="server" Text="Search"
onclick="Button1_Click" />
```

รูปที่ 3.13 Code ASP.NET ที่ใช้ในการออกแบบเทมเพลต

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master"
AutoEventWireup="true" CodeFile="[:Name:].aspx.cs" Inherits="[:Name:]"
Title="[:Title:]" %> <asp:Content ID="Content1" ContentPlaceHolderID="head"
Runat="Server"></asp:Content><asp:ContentID="Content2"ContentPlaceHolderID="Cont
entPlaceHolder1"Runat="Server"> <table style="width:100%;"><tr>  <td>
            [:gv:]
        <asp:Button ID="Button3" runat="server" PostBackUrl="~/[:back:]_Insert.aspx"
            Text="Insert" />  </table></asp:Content>
```

รูปที่ 3.14 การออกแบบเทมเพลตจาก Code ASP.NET

จากรูปที่ 3.14 การออกแบบเทมเพลต [:Name:] คือตัวแปรที่จะทำการแทนที่โดยใช้ Regular Expression ค้นหาคำและ แทนที่คำที่ต้องการเพื่อให้ถูกต้องตามรูปแบบของเว็บไซต์ ASP.NET โดยสร้างเทมเพลตไว้ดังนี้

- TemplateView.txt
- TemplateInsert.txt
- TemplateUpdate.txt
- TemplateDelete.txt
- TemplateMasterPage.txt
- TemplateGridviews.txt
- TemplateDetailViews.txt
- TemplateSkinfile.txt

- เขียนโปรแกรมในการ Generate โดยการอ่านไฟล์จาก Template ที่จัดทำไว้โดยใช้ Regular Expression ในการค้นหาตัวแปรที่เรากำหนดไว้ เช่น `[:label:]` เมื่อเจอตัวแปรก็ทำการแทนค่าใหม่เข้าไปเพื่อให้ถูกต้องตามรูปแบบของ ASP.NET โดยจะสร้างไฟล์เทมเพลตไว้ดังนี้

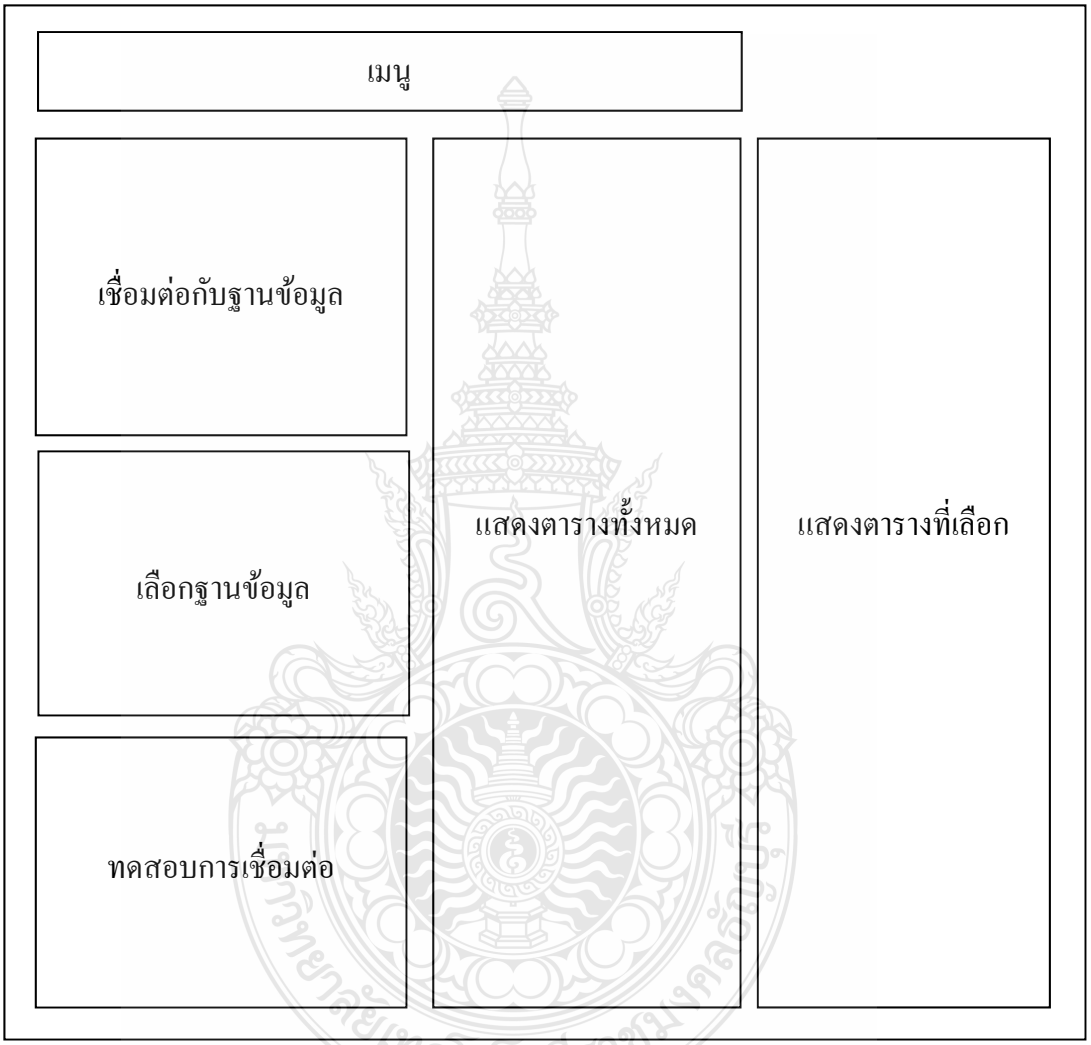
  เขียนโปรแกรม โมดูล Generate ดังรูปที่ 3.15

```
generate.OutputFilename = folderBrowserDialog1.SelectedPath + "\\" +
"MasterPage.master";                                    // เลือกที่เก็บไฟล์ output
generate.ReadInputFile("TemplateMasterPage.txt");       // อ่านไฟล์จากเทมเพลต
        if (x.PrimaryKey == true)
       { generate.ReplaceObj("keyid", x.ColumnName);    // ค้นหาและแทนที่คำ
         generate.ReplaceObj("primary", x.ColumnName);}
         generate.ReplaceObj("tbnme", table.TableName);
       if (x.Frk != true){  generate.ReplaceObj("nCL", nCL);}
         generate.ReplaceObj("nCLname", x.ColumnName);
         generate.ReplaceObj("nCLnamedis",x.Column_Description);}
        generate.WriteOutputFile();                     // เขียนไฟล์ทั้งหมด
```

รูปที่ 3.15 วิธีการ Generate ไฟล์ และเขียนไฟล์ไปยังไดเร็กทอรี่

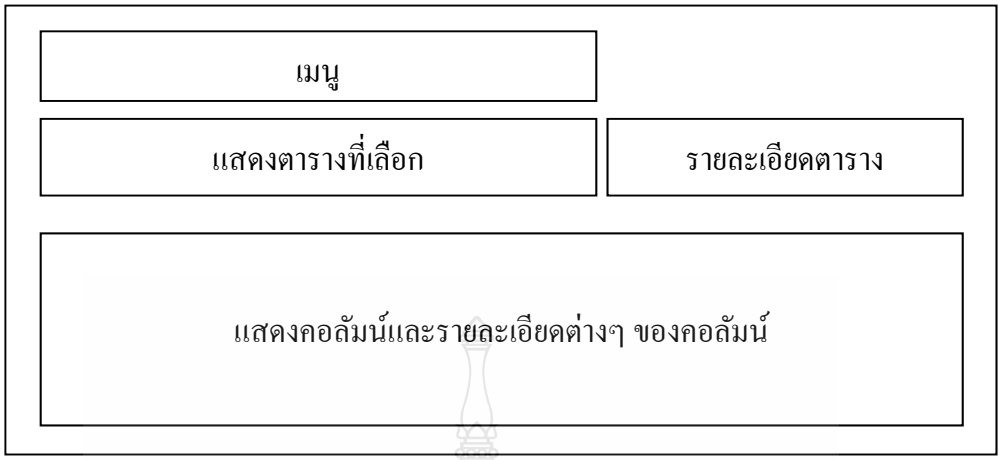
3.1.7 การออกแบบหน้าจอระบบ ASP.NET Forms Generator

การออกแบบโปรแกรมการใช้งานก็เป็นส่วนสำคัญ เพื่อที่ผู้ใช้สามารถใช้งานได้ง่าย สะดวก สบาย ดูแล้วเข้าใจการทำงานของโปรแกรม



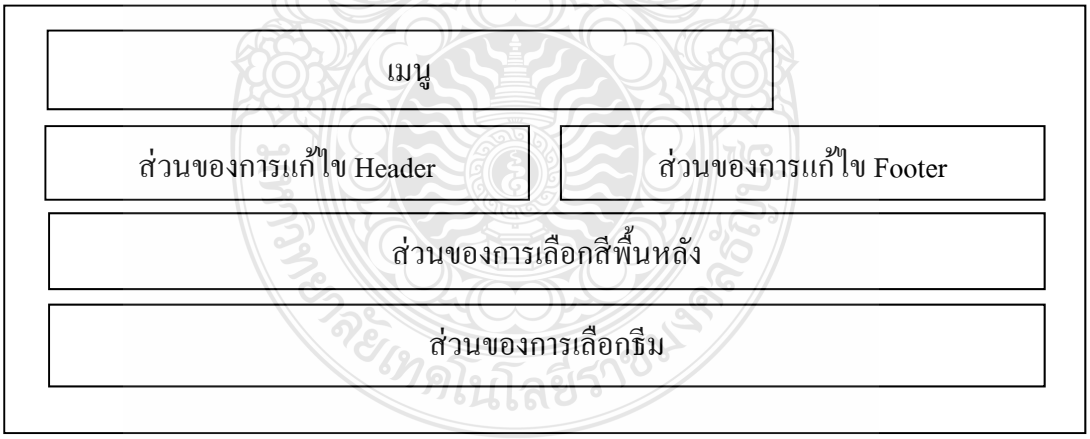
รูปที่ 3.16 แสดงการออกแบบการเชื่อมต่อฐานข้อมูล และเลือกตาราง

จากรูปที่ 3.16 ในส่วนของการเชื่อมต่อกับฐานข้อมูล จะประกอบด้วย การเลือก เซิร์ฟเวอร์ การเลือกรูปแบบการเชื่อมต่อว่าจะเป็นแบบ SQL Authentication หรือ Windows Authentication การกรอก User และ Password ในส่วนของการเลือกฐานข้อมูลจะแสดงชื่อของฐานข้อมูลให้ผู้ใช้เลือก ในส่วนของการแสดงตารางทั้งหมดจะแสดงรายชื่อของตารางทั้งหมดจากฐานข้อมูลที่เลือก ในส่วนของการแสดงตารางที่เลือก จะแสดงตารางที่ผู้ใช้เลือกไว้ทั้งหมด



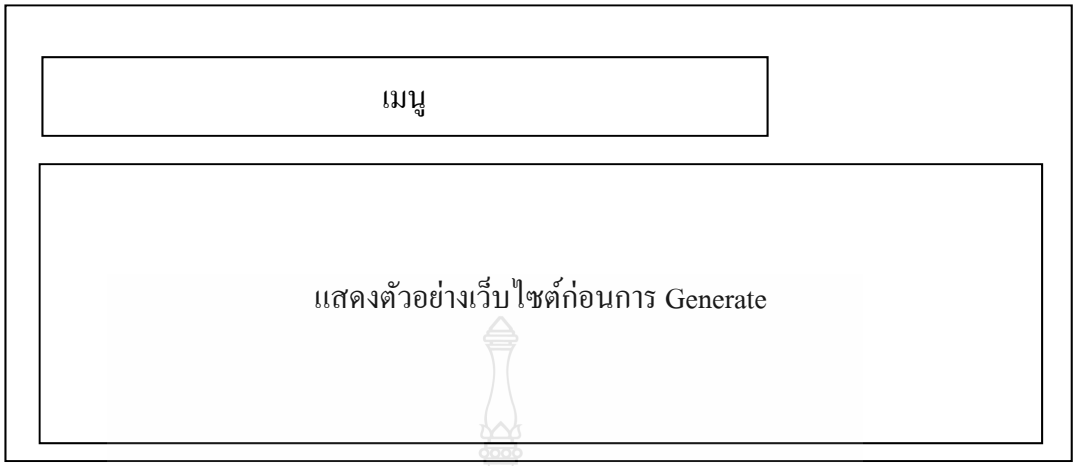
รูปที่ 3.17 การออกแบบหน้าจอแสดงคอลัมน์และการเลือกคอลัมน์ของตารางที่เลือกไว้

จากรูปที่ 3.17 ในส่วนของการแสดงตารางที่เลือก จะแสดงรายชื่อของตารางทั้งหมดที่ผู้ใช้ได้เลือกไว้และสามารถแก้ไขรายละเอียดของตารางได้ ในส่วนของการแสดงคอลัมน์และรายละเอียดต่างๆ ของคอลัมน์ จะแสดงรายชื่อคอลัมน์ภายในตารางที่เลือก และแสดงรายละเอียดต่างๆ สามารถเลือกได้ว่าจะใช้คอลัมน์ใดบ้าง



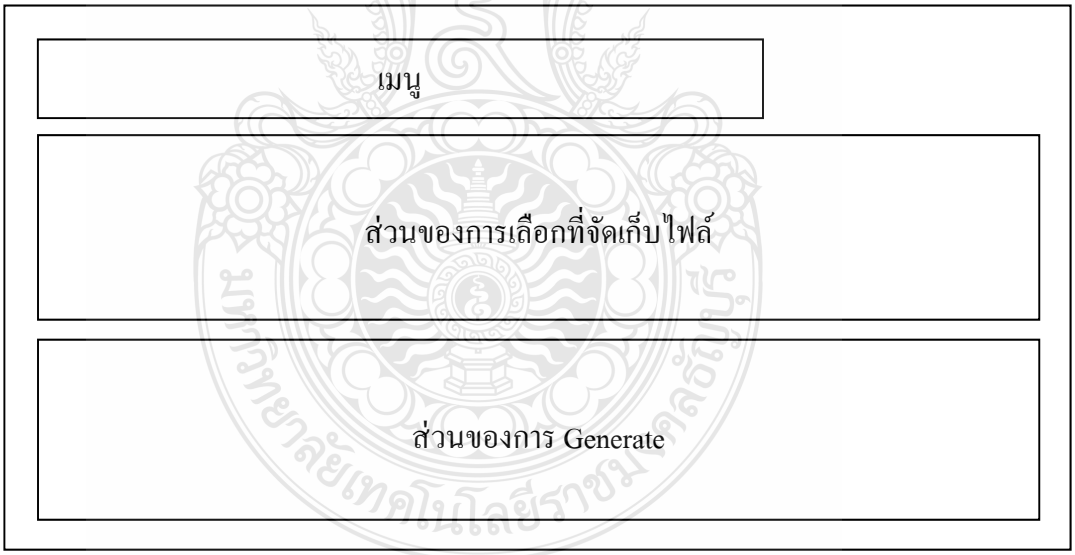
รูปที่ 3.18 การออกแบบหน้าจอการแก้ไข Header, Footer และเลือกพื้นหลังของ Master Page

จากรูปที่ 3.18 ในส่วนของการเลือกธีมผู้ใช้สามารถแก้ไข Header, Footer จะสามารถเลือกรูปภาพจากเครื่องผู้ใช้ได้ หรือใส่ข้อความในส่วน Header และ ส่วน Footer ได้ ในส่วนของการเลือกสีพื้นหลัง (Back Ground) จะมีให้ผู้ใช้เลือกคือ สีน้ำเงิน สีเทา สีขาว สีดำ และสีฟ้าคราม ในส่วนของการเลือกกริดวิวที่ใช้แสดงข้อมูล จะมีรูปภาพกริดวิวแสดงเป็นตัวอย่างด้วย



รูปที่ 3.19 การออกแบบหน้าตัวอย่างของเว็บไซต์ก่อนการ Generate

จากรูปที่ 3.19 ในส่วนของการแสดงตัวอย่างเว็บไซต์ก่อนการ Generate จะแสดงตัวอย่างหน้าเว็บไซต์แบบคร่าวๆ ให้ผู้ใช้งานดูก่อนการ Generate



รูปที่ 3.20 การออกแบบของการ Generate

จากรูปที่ 3.20 ส่วนของการเลือกที่จัดเก็บไฟล์ ผู้ใช้งานต้องทำการเลือก Directory ที่ต้องการจัดเก็บไฟล์ด้วย และในส่วนของการ Generate จะแสดงปุ่มสำหรับกดเพื่อทำการ Generate ไฟล์ทั้งหมดเมื่อ Generate เสร็จเรียบร้อยแล้วจะมีการรายงานผลการ Generate

3.2 แผนการดำเนินงาน

การดำเนินงานเริ่มจากการศึกษาข้อมูลจากระบบงานโครงงานเดิมแล้วนำข้อมูลที่ได้มารวบรวมแล้วนำไปแก้ไข ปรับปรุง และเพิ่มความสามารถของการทำงานโครงงานเดิม แต่ละส่วนการทำงานของระบบ

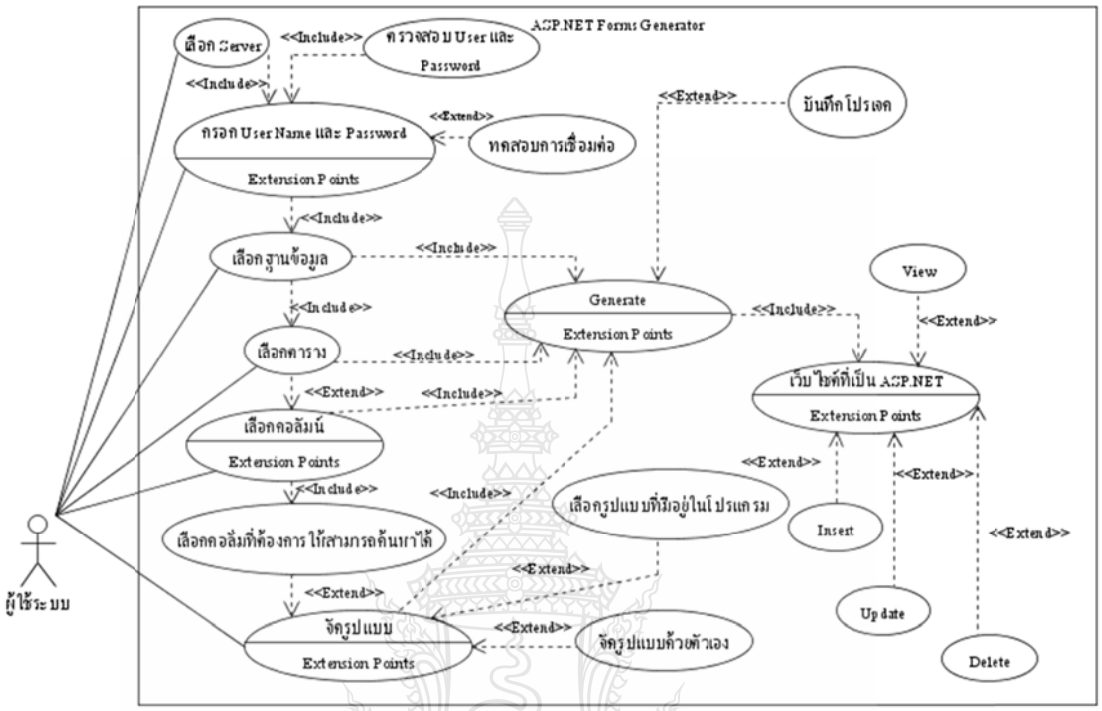
ตารางที่ 3.1 แผนการดำเนินงานของโครงงาน

กิจกรรม	ระยะเวลาการดำเนินงาน (เดือน)										
	ต.ค	พ.ย	ธ.ค	ม.ค	ก.พ	พ.ค	มิ.ย	ก.ค	ส.ค	ก.ย	ต.ค
ศึกษาเนื้อหาของโครงงานเดิม											
ศึกษาโปรแกรมและเครื่องมือเพิ่มเพื่อใช้ในการเขียนโปรแกรม											
ออกแบบการทำงานของระบบใหม่											
เขียนโปรแกรมการอัพเดท Schema และการจัดการฟอร์ม ASP.NET											
ปรับปรุงและแก้ไขข้อผิดพลาด											
ตรวจสอบการทำงานของระบบ											
จัดทำปริญญานิพนธ์และเข้าพบอาจารย์ที่ปรึกษา											

- - - - - แสดงแผนการดำเนินงาน

——— แสดงการดำเนินงานจริง

3.3 การวิเคราะห์ระบบงาน (Use Case Diagrams)



รูปที่ 3.21 Use Case Diagrams ของระบบ Forms Generator ASP.NET Version 2

จากรูปที่ 3.21 แสดง Use Case Diagrams ของระบบ ซึ่งเป็น Use Case รวมทั้งระบบที่มีความสัมพันธ์กับผู้ใช้เพียงคนเดียว โดยผู้ใช้ในที่นี้หมายถึงผู้ใช้งานระบบ ในการทำงานของระบบเริ่มต้นจากผู้ใช้ทำการเลือกเซิร์ฟเวอร์ (Server name) แล้วกรอก User และ Password แล้วทำการเชื่อมต่อฐานข้อมูล โดยจะมีกระบวนการตรวจสอบความถูกต้อง โดยจะตรวจทั้ง User และ Password และ ระบบยังมีการทดสอบการเชื่อมต่อกับ SQL Server แต่เมื่อทำการเชื่อมต่อกับฐานข้อมูลได้ ระบบจะทำการแสดงรายชื่อของฐานข้อมูล (Database) จากเซิร์ฟเวอร์ (Server) เมื่อทำการเลือกฐานข้อมูล ระบบจะแสดงตาราง (Table) ที่มีอยู่ในฐานข้อมูลนั้นๆออกมา จากนั้นผู้ใช้จะต้องทำการเลือกตาราง โดยผู้ใช้งานจะเลือกทั้งหมด หรือเลือกตารางตามต้องการ เมื่อเลือกตารางได้แล้ว ระบบก็จะแสดงคอลัมน์ที่มีในตารางที่ทำการเลือกมา ผู้ใช้สามารถที่จะเลือกคอลัมน์ (Column) หรือไม่เลือกคอลัมน์ ซึ่งระบบจะทำการเลือกไว้ทั้งหมด (Default) ต่อมาผู้ใช้ก็สามารถเลือกจัดรูปแบบฟอร์มข้อความและคอนโทรล หรือเลือกใช้ ธีม (Theme) ในระบบที่มีให้ 5 รูปแบบ ซึ่งแบ่งการทำงาน ออกเป็น 6 Use Case ย่อยได้ดังนี้

3.3.1 Use Case การรับรายละเอียดของการเชื่อมต่อกับ SQL Server โดย Use Case นี้จะทำหน้าที่รับรายละเอียดการเชื่อมต่อกับ SQL Server เพื่อนำไปตรวจเช็คสิทธิ์ในการเข้าใช้งานฐานข้อมูล

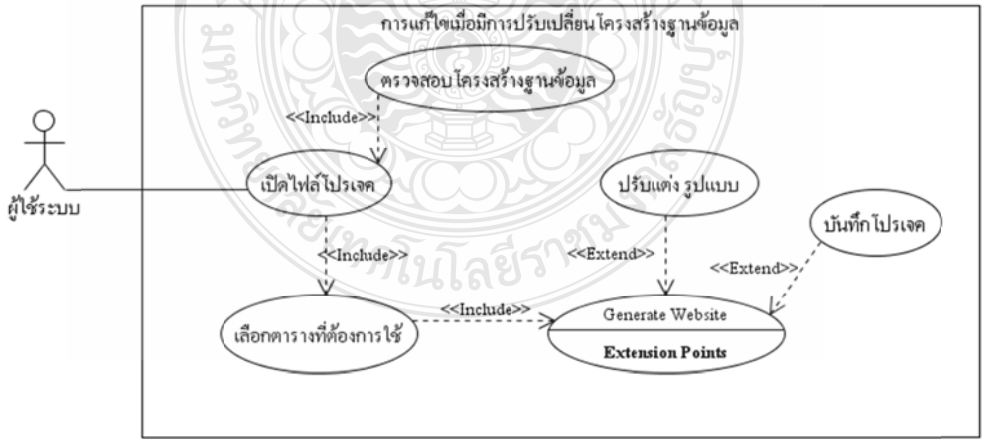
3.3.2 Use Case การตรวจสอบการเชื่อมต่อฐานข้อมูล โดยทำหน้าที่ตรวจสอบข้อมูลการเชื่อมต่อและแสดงผลให้ผู้ใช้งานทราบซึ่งผู้ใช้ก็สามารถเลือกว่าจะทดสอบหรือไม่ทดสอบการเชื่อมต่อก็ได้

3.3.3 Use Case การเลือกตารางและคอลัมน์ ทำหน้าที่รับข้อมูลตาราง จากฐานข้อมูลออกมาแสดงให้ผู้ใช้ทำการเลือกตาราง แล้วทำการเลือกคอลัมน์ ภายในตารางที่ได้เลือกไว้ก่อนหน้า และสามารถใส่ข้อความอธิบายตารางได้ เพื่อนำข้อมูลภายในคอลัมน์ไปใช้งานขั้นตอนต่อไป

3.3.4 Use Case การเลือกรูปแบบ ซึ่ง Use Case นี้ทำหน้าที่จัดรูปแบบฟอร์มข้อความและคอนโทรล หรือเลือกใช้ ธีม ที่ระบบมีให้ 5 รูปแบบ และสามารถเพิ่มส่วนหัว ส่วนท้ายได้ ใส่รูปเป็นเป็นพื้นหลัง ใส่รูปทำสไลด์รูปโชว์ได้

3.3.5 Use Case การแก้ไขรายละเอียดคอลัมน์ ทำหน้าที่นำคอลัมน์ ที่ได้มาทำการแก้ไขรายละเอียดต่างๆก่อนนำไปใช้งาน

3.3.6 Use Case การ Generate ฟอร์ม ทำหน้าที่สร้างฟอร์มจากข้อมูลที่ได้บันทึกจาก Use Case การแก้ไขรายละเอียดคอลัมน์ออกมาเป็นโค้ดไฟล์ นามสกุล .aspx, .dbml, .config, .css และ .master บันทึกไว้ในไดเร็คทอรี่ที่ผู้ใช้งานต้องการ



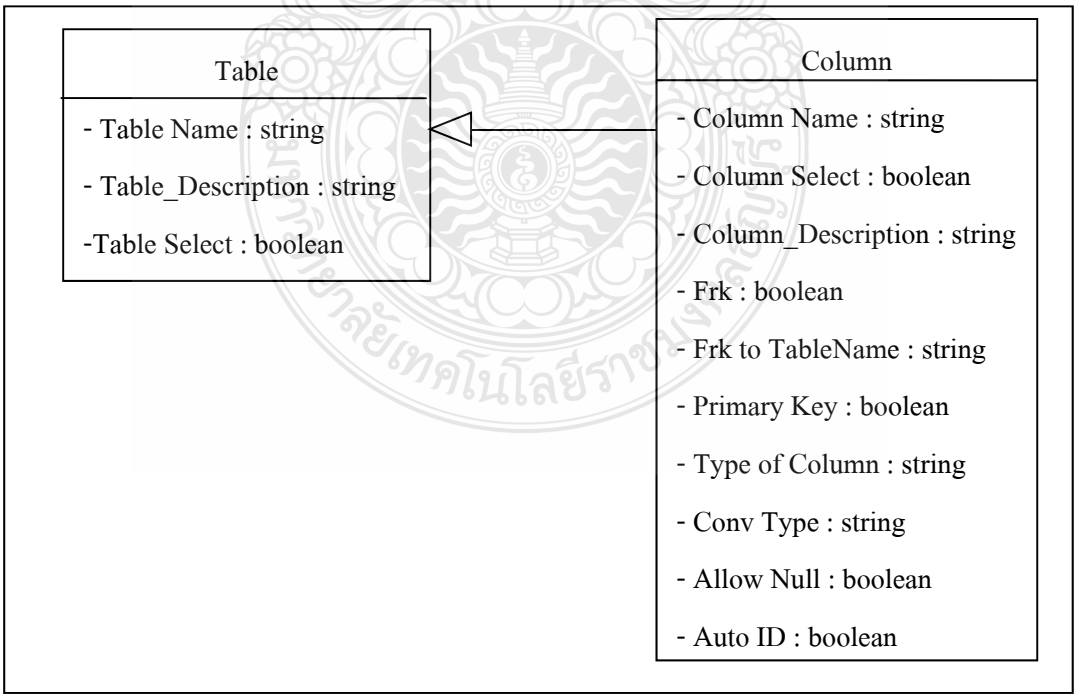
รูปที่ 3.22 Use Case Diagrams แสดงการตรวจสอบการ Update Database Schema

จากรูป 3.22 Use Case Diagrams แสดงการตรวจสอบการ Update Database Schema ซึ่งในโครงงานนี้ได้เพิ่มการตรวจสอบ เมื่อผู้ใช้เปิดไฟล์โปรเจกต์ขึ้นมาระบบจะตรวจสอบการ Update Database Schema ก่อน เมื่อตรวจสอบพบมีการ Update Database Schema ระบบจะแจ้งเตือนว่าจะให้เลือกใช้โครงสร้างฐานข้อมูลอันใหม่ แต่ถ้าไม่เลือกจะใช้โครงสร้างฐานข้อมูลอันเดิมในการ Generate แต่ในโครงงานเดิม นั้นไม่สามารถตรวจสอบการ Update Database Schema ได้ ถ้ามีการเลือกใช้โครงสร้างฐานข้อมูลอันใหม่ และผู้ใช้ทำการเลือกตารางอันใหม่เพิ่ม โดยผู้ใช้จะเลือกทั้งหมด หรือเลือกตารางตามต้องการ และหากผู้ใช้ต้องการการแก้ไขจัดรูปแบบฟอร์มข้อความและคอนโทรล หรือเลือกธีมใหม่ สามารถแก้ไขต่อจากเดิมที่มีการแก้ไขไว้ในไฟล์โปรเจกต์ที่เปิดขึ้นมา เมื่อแก้ไขเสร็จแล้วจะทำการ Generate ได้เลย หรือ ถ้ายังไม่ต้องการ Generate ก็สามารถบันทึกเป็นโปรเจกต์ไฟล์

3.4 แผนภาพที่ใช้แสดง Class และความสัมพันธ์ ระหว่าง Class (Class Diagrams)

จากการวิเคราะห์ระบบมีการเก็บข้อมูลลงในภายใน Dataset และมี Class สำหรับเรียกใช้งานฟังก์ชันในการ Generate คือ GenClass ซึ่งโครงงานนี้ได้ใช้ Class Diagrams จากโครงงานเดิม

3.4.1 คลาสที่ใช้ในการเก็บข้อมูลของตารางที่ถูกเลือกนำไปใช้ในการ Generate

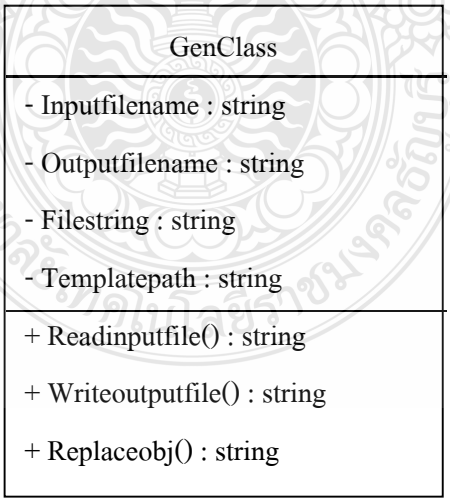


รูปที่ 3.23 คลาสที่ใช้ในการเก็บข้อมูลของระบบ ASP.NET Forms Generator Version 2

จากรูปที่ 3.23 เป็นคลาสที่ใช้ในการเก็บข้อมูลของตารางที่ถูกเลือกโดยจะเก็บข้อมูลต่างๆ หลังจากการแก้ไขเสร็จเรียบร้อยมีค่าต่างๆ ดังนี้

1) Table Name ชื่อตารางที่ถูกเลือกเพื่อทำการ Generate
2) Table_Description แสดงรายละเอียดของตาราง
3) Table Select ตารางที่เลือก
4) Column Name ชื่อคอลัมน์ทั้งหมดในตาราง
5) Column Select คอลัมน์ที่เลือก
6) Column_Description รายละเอียดของคอลัมน์
7) Frk คอลัมน์ที่เป็น Forieng Key หรือคีย์สัมพันธ์
8) Frk to Table ชื่อตารางที่มีความสัมพันธ์ไปยังตารางนั้นๆ
9) Primary Key คอลัมน์ที่เป็น Primary Key หรือ คีย์หลัก
10) Typ Of Column ชนิดของคอลัมน์จากฐานข้อมูล
11) Conv Type แปลงชนิดของคอลัมน์เป็นชนิดตัวแปรในการเขียนโปรแกรม
12) Allow Null แสดงว่าคอลัมน์ใดเป็น NULL
13) Auto ID คอลัมน์ที่เป็น Auto ID

3.4.2 คลาสที่ใช้ในการ Generate



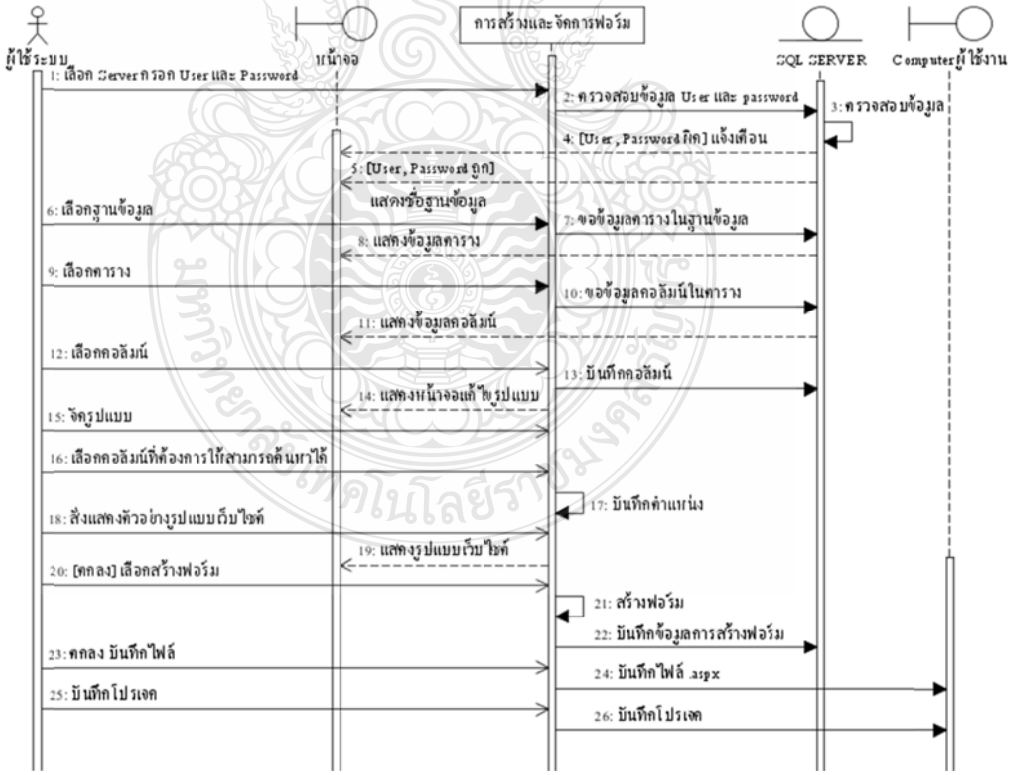
รูปที่ 3.24 คลาสที่ใช้ในการ Generate

จากรูปที่ 3.24 เป็นคลาสที่ใช้ในการ Generate โดยโปรแกรมจะเรียกใช้งานฟังก์ชันต่างๆ ภายในคลาส ซึ่งประกอบไปด้วย

1) แอตทริบิวต์ Inputfilename ใช้เก็บชื่อไฟล์ของเทมเพลต
2) แอตทริบิวต์ Outputfilename ใช้เก็บที่อยู่ของเอาต์พุตที่ต้องการสร้าง
3) แอตทริบิวต์ Filestring ใช้เก็บชื่อไฟล์ของเอาต์พุตที่ต้องการสร้าง
4) แอตทริบิวต์ Templatepath ใช้เก็บที่อยู่ของไฟล์เทมเพลต
5) เมธอด Readinputfile() ใช้สำหรับอ่านข้อมูลภายในเทมเพลต
6) เมธอด Writeoutputfile() ใช้สำหรับเขียนไฟล์เอาต์พุต
7) เมธอด Replaceobj() ใช้สำหรับค้นหาและแก้ไขข้อมูล

3.5 วิเคราะห์พฤติกรรมของระบบ Sequence Diagrams

ในแต่ละ Use Case จะมีกิจกรรมต่างๆที่เกิดขึ้นในระบบดังได้แสดงไว้ใน Sequence Diagrams ต่อไปนี้



รูปที่ 3.25 Sequence Diagrams ของ Use Case รวมของระบบ ASP.NET Forms Generator Version2

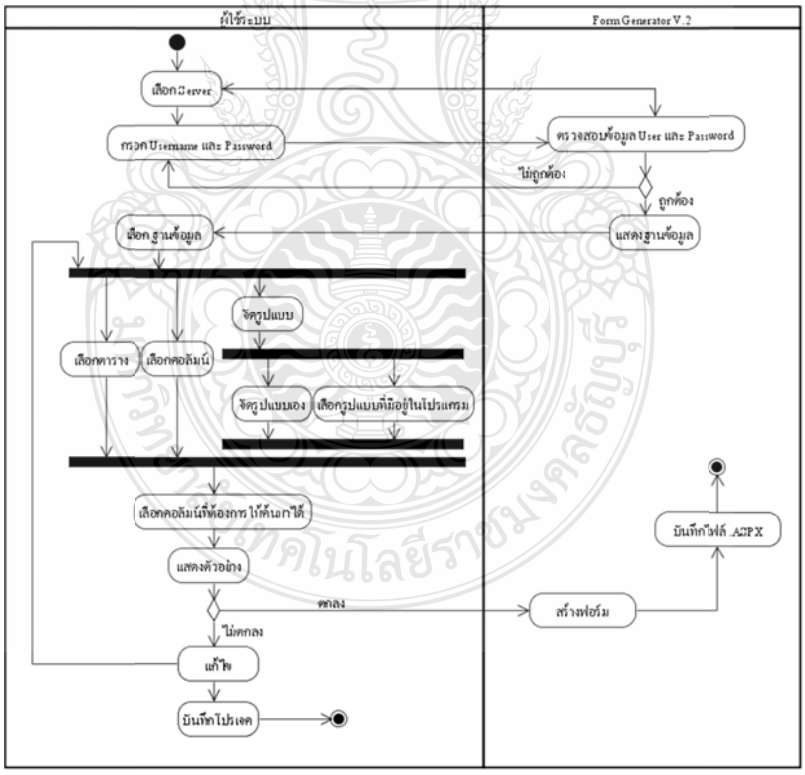
จากรูปที่ 3.25 Sequence Diagrams ของ Use Case รวมของระบบ จะเริ่มต้นจากผู้ใช้ทำการเลือกชื่อเซิร์ฟเวอร์ (Server name) และทำการกรอก User และ Password ระบบก็จะทำการตรวจสอบความถูกต้องของ User และ Password ในฐานข้อมูลจะมีการตรวจสอบ User และ Password ด้วยเช่นกัน ใน กรณี เลือกเป็น SQL Authentication หลังจากนั้น ถ้าเกิด User หรือ Password ผิดฐานข้อมูลจะแจ้งเตือนแสดงที่หน้าจอแสดงผล ถ้า User และ Password ถูกต้อง ระบบจะขอข้อมูลจากฐานข้อมูลเพื่อนำมาแสดงตาราง ทั้งหมด หลังจากนั้นผู้ใช้ทำการเลือกตารางที่จะใช้ในการ Generate เมื่อทำการเลือกตารางเสร็จ ระบบจะทำการแสดงข้อมูลคอลัมน์จากฐานข้อมูล และ ให้ผู้ใช้ทำการเลือกคอลัมน์ ผู้ใช้สามารถแก้ไขรายละเอียดของคอลัมน์ได้ ขั้นตอนต่อมาผู้ใช้สามารถจัดรูปแบบฟอร์มข้อความและคอนโทรล หรือเลือกใช้ ธีม (Theme) ในระบบที่มีให้ 5 รูปแบบ ถ้าหากผู้ใช้งานต้องการแก้ไขก็สามารถแก้ไขได้ เมื่อผู้ใช้งานทำการแก้ไขเสร็จและทำการแสดงตัวอย่างหน้าเว็บไซต์ จะทำการ Generate ได้เลย หรือ ถ้ายังไม่ต้องการ Generate ก็สามารถบันทึกเป็น “โปรเจกต์ไฟล์” เมื่อทำการ Generate ผู้ใช้สามารถเลือกได้ว่าต้องการที่จะนำไฟล์ที่ทำการ Generate ไปเก็บไว้ในโฟลเดอร์ใด หรือจะสร้างโฟลเดอร์ขึ้นมาใหม่

รูปที่ 3.26 Sequence Diagrams ของ Use Case การตรวจสอบการ Update Database Schema

จากรูปที่ 3.26 Sequence Diagrams ของ Use Case การตรวจสอบการ Update Database Schema ซึ่งในโครงงานนี้ได้เพิ่มการตรวจสอบ เมื่อผู้ใช้เปิดไฟล์โปรเจกต์ขึ้นมาระบบจะ

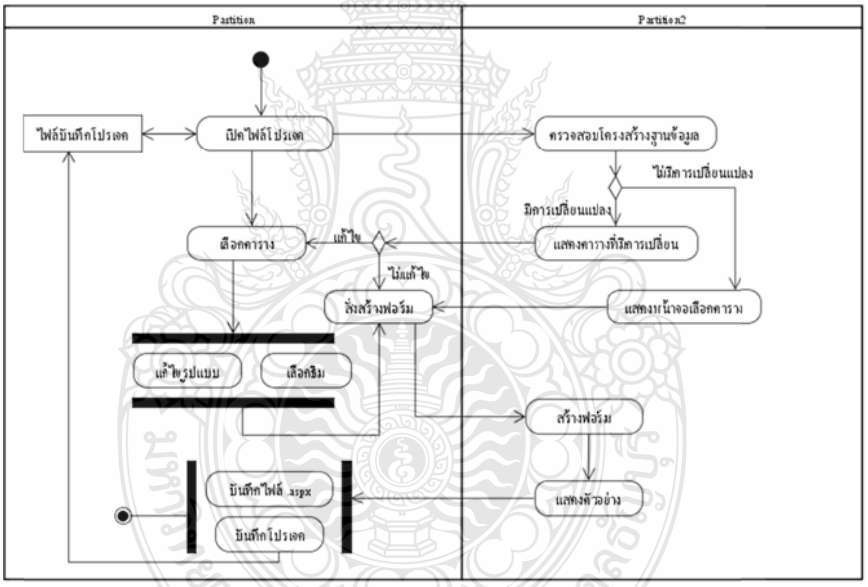
ตรวจสอบการ Update Database Schema ก่อน เมื่อตรวจสอบพบมีการ Update Database Schema ระบบจะแจ้งเตือนว่าจะให้เลือกใช้โครงสร้างฐานข้อมูลอันใหม่ แต่ถ้าไม่เลือกจะใช้โครงสร้างฐานข้อมูลอันเดิมในการ Generate แต่ในโครงงานเดิม นั้นไม่สามารถตรวจสอบการ Update Database Schema ได้ ถ้ามีการเลือกใช้โครงสร้างฐานข้อมูลอันใหม่ และผู้ใช้ทำการเลือกตารางอันใหม่เพิ่ม โดยผู้ใช้จะเลือกทั้งหมด หรือเลือกตารางตามต้องการ และหากผู้ใช้ต้องการการแก้ไขจัดรูปแบบฟอร์ม หรือเลือกธีมใหม่ สามารถแก้ไขต่อจากเดิมที่มีการแก้ไขไว้ในไฟล์โปรเจกต์ที่เปิดขึ้นมา เมื่อแก้ไขเสร็จแล้วทำการแสดงตัวอย่างหน้าเว็บไซต์ จะทำการ Generate ได้เลย หรือ ถ้ายังไม่ต้องการ Generate ก็สามารถบันทึกเป็นโปรเจกต์ไฟล์ เมื่อทำการ Generate ผู้ใช้สามารถเลือกได้ว่าต้องการที่จะนำไฟล์ที่ทำการ Generate ไปเก็บไว้ในโฟลเดอร์ใด หรือจะสร้างโฟลเดอร์ขึ้นมาใหม่

3.6 ลำดับกิจกรรมของการทำงาน (Activity Diagrams)



รูปที่ 3.27 Activity Diagram ของ Use Case รวมของระบบ Forms Generator ASP.NET Version 2

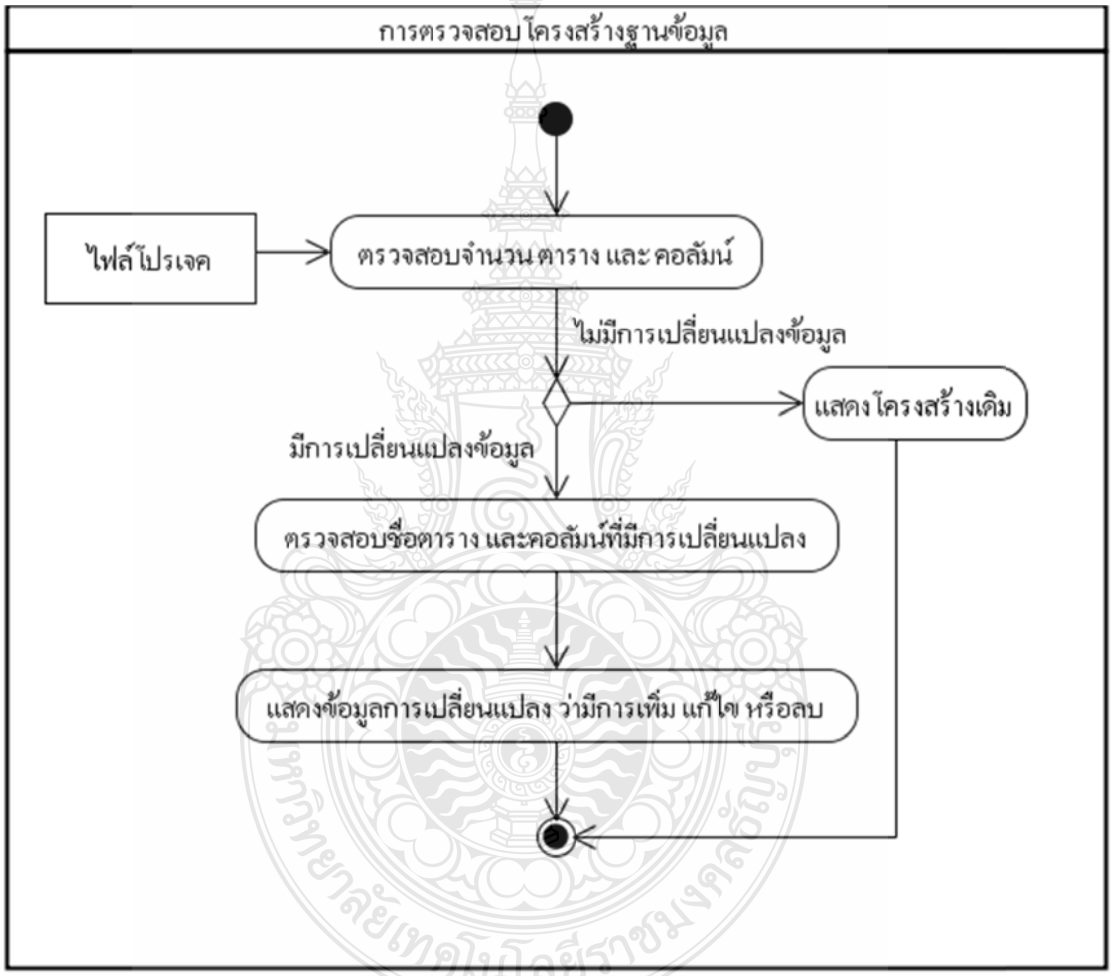
จากรูปที่ 3.27 จะแบ่งออกเป็น 2 อย่างคือ ผู้ใช้ และ ระบบ ASP.NET Forms Generator Version 2 การทำงานจะเริ่มต้นจากผู้ใช้ทำการกรอกข้อมูล User และ Password ระบบ Form Generator จะทำการตรวจสอบข้อมูล User และ Password เมื่อ User และ Password ผิด ระบบจะให้กรอกข้อมูลใหม่อีกครั้ง และ เมื่อ User และ Password ถูกต้องแล้วระบบจะทำการแสดงข้อมูลตารางจาก ฐานข้อมูลแสดงให้ผู้ใช้ทำการเลือกฐานข้อมูล จากนั้นผู้ใช้จะทำการเลือกว่าจะใช้ตารางใดๆ จากนั้นทำการเลือกคอลัมน์ ใดบ้างในการ Generate และทำการจัดรูปแบบฟอร์มข้อความและคอนโทรล หรือเลือกใช้ ธีม ในระบบที่มีให้ 5 รูปแบบ และทำการแสดงตัวอย่างหน้าเว็บไซต์ จากนั้นหากตกลงระบบจะทำการ Generate ออกมาได้เป็นไฟล์เว็บไซต์ .aspx, .dbml, .config, .css และ .master แต่ถ้าผู้ใช้ยังไม่ต้องการ Generate ก็สามารถบันทึกเป็น“โปรเจกต์ไฟล์” ไว้เพื่อสามารถนำกลับมาใช้แก้ไขและ Generate ในครั้งต่อไปได้



รูปที่ 3.28 Activity Diagrams ของ Use Case การตรวจสอบการ Update Database Schema

จากรูปที่ 3.28 Activity Diagrams ของ Use Case การตรวจสอบการ Update Database Schema ซึ่งในโครงงานนี้ได้เพิ่มการตรวจสอบ เมื่อผู้ใช้เปิดไฟล์โปรเจกต์ขึ้นมาระบบจะตรวจสอบการ Update Database Schema ก่อน เมื่อตรวจสอบพบมีการ Update Database Schema ระบบจะแจ้งเตือนว่าจะให้เลือกใช้โครงสร้างฐานข้อมูลอันใหม่ แต่ถ้าไม่เลือกจะใช้โครงสร้างฐานข้อมูลอันเดิมในการ Generate แต่ในโครงงานเดิม นั้นไม่สามารถตรวจสอบการ Update Database Schema ได้ ถ้ามีการเลือกใช้โครงสร้างฐานข้อมูลอันใหม่ และผู้ใช้ทำการเลือกตารางอันใหม่เพิ่ม

โดยผู้ใช้จะเลือกทั้งหมด หรือเลือกตารางตามต้องการ และหากผู้ใช้ต้องการการแก้ไขจัดรูปแบบฟอร์มข้อความและคอนโทรล หรือเลือกธีมใหม่ สามารถแก้ไขต่อจากเดิมที่มีการแก้ไขไว้ในไฟล์โปรเจกต์ที่เปิดขึ้นมา เมื่อแก้ไขเสร็จแล้วทำการแสดงตัวอย่างหน้าเว็บไซต์ จะทำการ Generate ได้เลย หรือ ถ้ายังไม่ต้องการ Generate ก็สามารถบันทึกเป็นโปรเจกต์ไฟล์ หรือเมื่อผู้ใช้งานทำการแก้ไขเสร็จจะทำการ Generate ได้เลย

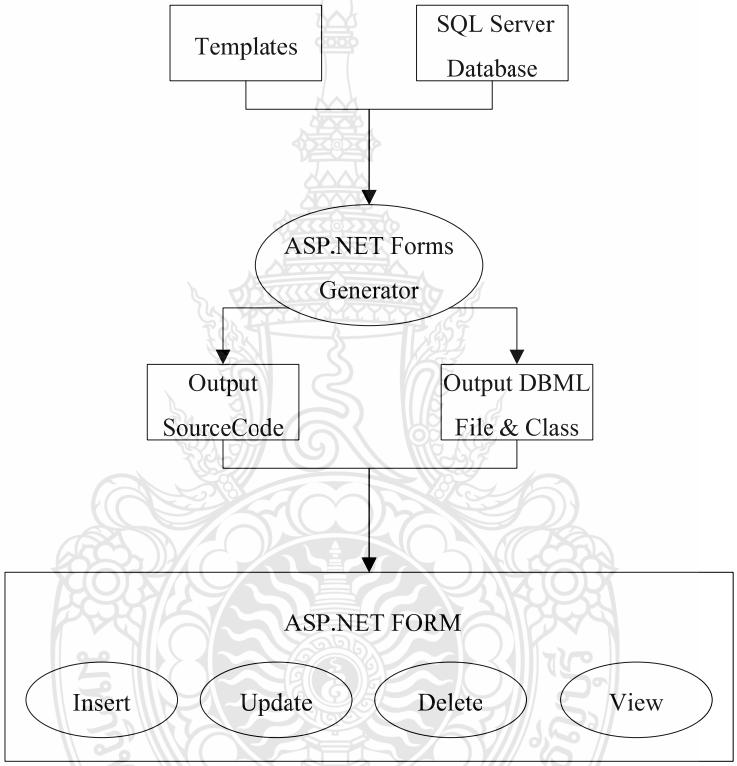


รูปที่ 3.29 Activity Diagrams การตรวจสอบโครงสร้างฐานข้อมูล

จากรูปที่ 3.29 Activity Diagrams การตรวจสอบโครงสร้างฐานข้อมูล เริ่มจากผู้ใช้เปิดไฟล์งานเดิมระบบจะทำการตรวจสอบโครงสร้างฐานข้อมูลทั้งหมด เมื่อตรวจสอบพบมีการเปลี่ยนแปลงโครงสร้างฐานข้อมูล ระบบจะแจ้งเตือนว่า Table หรือ Column ไหนมีการเปลี่ยนแปลงโครงสร้าง ถ้ามีการเพิ่มขึ้นของ Table หรือ Column ระบบจะแจ้งเตือนว่าชื่อ Table

หรือ Column ไหนที่เพิ่มขึ้นมา ถ้ามีการแก้ไขชื่อ Table หรือ Column ระบบจะแจ้งเตือนว่าชื่อ Table หรือ Column ไหนที่มีการแก้ไขชื่อ และถ้ามีการลบ Table หรือ Column ระบบจะแจ้งเตือนว่าชื่อ Table หรือ Column ไหนที่ถูกลบออกไป หากระบบแจ้งเตือนแล้วผู้ใช้ไม่เลือกใช้โครงสร้างฐานข้อมูลอันใหม่ ระบบจะแสดงโครงสร้างฐานข้อมูลจากไฟล์งานเดิม

3.7 ผังการทำงานของระบบ ASP.NET Forms Generator Version2



รูปที่ 3.30 ผังการทำงานของระบบโดยรวม

จากรูปที่ 3.30 แสดงผังการทำงานโดยรวมของระบบจะเริ่มจากอ่านไฟล์เทมเพลตและเก็บรายละเอียดต่างๆ จากฐานข้อมูล แล้วทำการ Generate ออกมาจะได้ไฟล์ 2 ส่วนคือส่วนของ Source Code และส่วนของ DBML และ Class ไฟล์ที่ได้นั้นจะเป็น ASP.NET ซึ่งมีฟังก์ชั่นการทำงานคือ Insert, Update, Delete, View

3.8 ขั้นตอนการดำเนินงาน

ในโครงงานนี้ จะมีส่วนที่แตกต่างจากโครงงานเดิม อยู่สามส่วนดังนี้ ส่วนที่หนึ่งการตรวจสอบการ Update Schema เมื่อโครงสร้างฐานข้อมูลมีการเปลี่ยนแปลง และหากไม่สามารถเชื่อมต่อกับฐานข้อมูลได้ระบบยังสามารถทำงานต่อได้ ส่วนที่สองสามารถจัดรูปแบบฟอร์มข้อความและคอนโทรลได้ และส่วนที่สามได้นำเอาความสามารถของ AJAX (Asynchronous JavaScript and XML) มาช่วยทำให้ Web Application มีประสิทธิภาพมากขึ้นและงานต่อการใช้งานของผู้ใช้

3.8.1 แบ่งระบบออกเป็นโมดูลย่อยๆ ดังนี้

1) ขั้นตอนการเชื่อมต่อฐานข้อมูล และการตรวจสอบการ Update Database Schema ส่วนในโครงงานเดิม ไม่สามารถตรวจสอบการ Update Database Schema ได้

2) ขั้นตอนการเลือกตาราง

3) ขั้นตอนการเลือกและแก้ไขรายละเอียดคอลัมน์ (Column)

4) ขั้นตอนการจัดรูปแบบฟอร์มข้อความและคอนโทรล และการเลือก ธีม (Theme) ส่วนในโครงงานเดิม ไม่สามารถจัดรูปแบบฟอร์มข้อความและคอนโทรล ได้

5) ขั้นตอนการ Generate

3.8.2 ศึกษาโปรแกรมและภาษาที่ต้องใช้ในการพัฒนาระบบต่อจาก โครงงานเดิม ดังนี้

1) โปรแกรม Microsoft Visual Studio 2010 ศึกษาพื้นฐานการเขียนโปรแกรม Windows Form Application ด้วยเครื่องมือต่างๆ และทดสอบการใช้เครื่องมือใน Visual Studio ส่วนในโครงงานเดิม ใช้โปรแกรม Microsoft Visual Studio 2008

2) ภาษา C# การสร้าง Windows Form Applications การสร้าง Class ในการเก็บข้อมูลด้วย Dataset และการตรวจสอบการ Update Schema

3) ภาษา ASP.NET 3.5 การสร้างเว็บไซต์ในการติดต่อกับฐานข้อมูล Insert Update Delete การ Validate Data Type การใช้งานเมนูใน Master Page Studio

4) การติดต่อฐานข้อมูลด้วย SMO (SQL Server Management Object) การออกแบบการสร้างและจัดการฟอร์ม ASP.NET เมื่อเชื่อมต่อกับฐานข้อมูลแล้วจะต้องทำการ Query Schema ทั้งหมดของฐานข้อมูลเพื่อนำมาใช้ในการสร้างเว็บไซต์ ดังนั้นจึงต้องเชื่อมต่อกับฐานข้อมูลด้วย SMO

5) ระบบฐานข้อมูล Microsoft SQL Server เนื่องจากการสร้างและจัดการฟอร์ม ASP.NET สร้างเว็บไซต์จาก Microsoft SQL Server ดังนั้นต้องเข้าใจการทำงานและคุณสมบัติของ Microsoft SQL Server

6) การใช้งาน SQLMetal ในการสร้าง LINQ to SQL Classes จะต้องมีการสร้าง Class Database เพื่อเชื่อมต่อกับฐานข้อมูลเป็นตัวการจัดการ Insert Update Delete และ View โดยใช้ SQLMetal.exe ไฟล์ที่ได้ออกมาจะเป็นไฟล์นามสกุล .dbml และคลาส .designer.cs

7) เอแจ็กซ์ (AJAX: Asynchronous JavaScript and XML) นำมาใช้เพื่อให้เว็บแอปพลิเคชันให้ความสามารถโต้ตอบกับผู้ใช้ได้ดีขึ้น โดยการรับส่งข้อมูลในฉากหลัง ทำให้ทั้งหน้าไม่ต้องโหลดใหม่ทุกครั้งที่มีการเปลี่ยนแปลง ซึ่งช่วยทำให้เพิ่มการตอบสนอง ความรวดเร็ว และการใช้งานโดยรวม

3.8.3 เขียนโปรแกรม

1) เขียนโปรแกรมในส่วนการเชื่อมต่อกับฐานข้อมูลด้วย SMO ดังตัวอย่างการออกแบบและเขียนโปรแกรมดังรูปที่ 3.31

รูปที่ 3.31 วิธีการออกแบบหน้าจอการเชื่อมต่อฐานข้อมูลด้วย SMO

รูปที่ 3.32 ตัวอย่าง Code Program ในการเชื่อมต่อกับฐานข้อมูลด้วย SMO

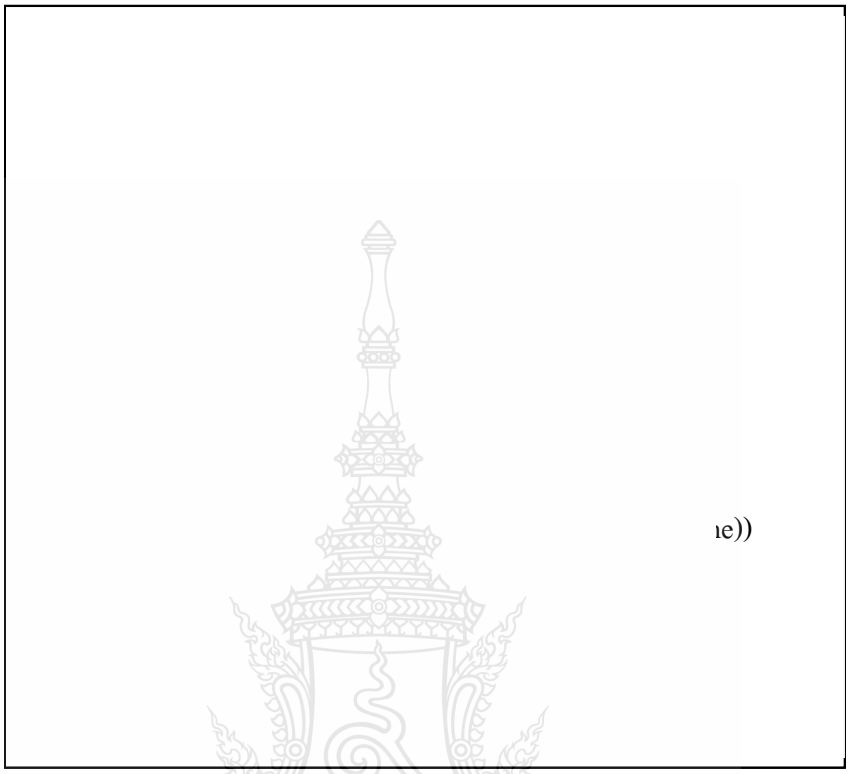
2) เขียนโปรแกรมในการตรวจสอบการ Update Schema เมื่อผู้ใช้เปิดไฟล์งานเดิมขึ้นมาใช้งาน

```
try{ SelectDB_();
 bool matchschema = true;
                matchschema = (tbl.Count == tbl_.Count);
                int countx = 0;
                if (tbl.Count > tbl_.Count)
                   countx = tbl.Count;
                else
                   countx = tbl_.Count;
                List<TableName> tempcompare = tbl;
                foreach (TableName item in tbl_)
 { Array.ForEach(tempcompare.Where(z=> z.x_TableName
                      == item.x_TableName).ToArray(), paratbl =>
 { paratbl.x_TableSelect = item.x_TableSelect;
   foreach (ForeignKeyName itemfor in item.ForeignKeys)
{ Array.ForEach(paratbl.ForeignKeys.Where(q=> q.ForeignKeyame
                      == itemfor.ForeignKeyame).ToArray(), parafor =>
               {if (!(parafor.ForeignKeyame == itemfor.ForeignKeyame))
                   matchschema = false;  });}
```

รูป 3.33 ตัวอย่าง Code Program การตรวจสอบการ Update Database Schema

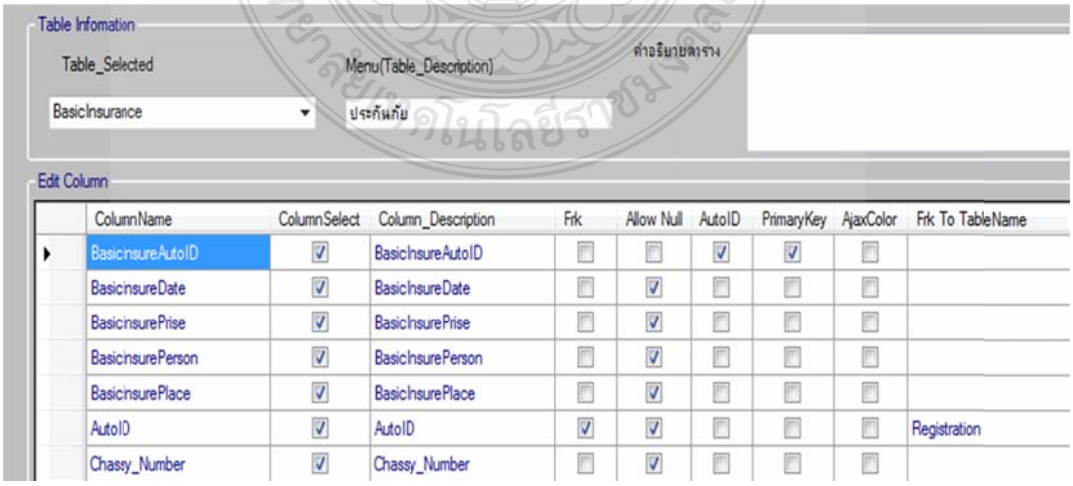
จากรูป 3.33 การตรวจสอบการ Update Database Schema เมื่อผู้ใช้โหลดไฟล์งานเดิมขึ้นมา ระบบจะตรวจสอบการ Update Database Schema ก่อน เมื่อระบบตรวจพบการเปลี่ยนแปลงโครงสร้างฐานข้อมูล ระบบจะแจ้งให้ผู้ใช้เลือกว่าจะเลือกโครงสร้างฐานข้อมูลอันใหม่ หรือเลือกโครงสร้างฐานข้อมูลจากไฟล์งานเดิม

3) เขียนโปรแกรมในส่วนของการเลือกตารางโดยใช้ ListBox ดังรูปที่ 3.14



รูปที่ 3.34 หน้าจอและตัวอย่าง Code Program ในการเลือกตาราง

4) เขียนโปรแกรม Query ข้อมูลจากฐานข้อมูลลงใน Dataset แล้วทำการแสดงผลผ่านดาต้ากริดวิวโดย Query ข้อมูลจาก Class Dataset เป็นดังรูปที่ 3.35



รูปที่ 3.35 การเพิ่มข้อมูลลง Dataset แล้วแสดงในดาต้ากริดวิว

5) เขียนโปรแกรมในการจัดการรูปแบบฟอร์ม โดยสร้างปุ่มกดสองปุ่มกดคือ ปุ่มเพิ่ม Column และปุ่มเพิ่ม Rowให้ผู้ใช้สามารถเพิ่มลดและจัดตำแหน่ง ข้อความและคอนโทรลได้ตามต้องการ หรือจะใช้รูปแบบที่ระบบมีให้ หากมีการปรับแต่งการจัดรูปแบบฟอร์ม ระบบจะเก็บตำแหน่งที่ผู้ใช้ได้ปรับตำแหน่งของ ข้อความหรือคอนโทรลเพื่อทำการ Generate ดังรูปที่ 3.36

```
List<ColumnName> Output = ColumnList;
Label tmpcontrol;
for (int i = 0; i < tableLayoutPanel1.ColumnCount; i++)
{  for (int j = 0; j < tableLayoutPanel1.RowCount; j++){
tmpcontrol = (Label)tableLayoutPanel1.GetControlFromPosition(i, j);
if (tmpcontrol != null)
{ if (tmpcontrol.Text.StartsWith("Ctr")){
Output.Single<ColumnName>(c => ("Ctr" + c.x_ColumnName ==
                                tmpcontrol.Text)).CtrFieldCol = i;
Output.Single<ColumnName>(c => ("Ctr" + c.x_ColumnName ==
                                tmpcontrol.Text)).CtrFieldRow = j;  }
if (tmpcontrol.Text.StartsWith("Lbl"))
{ Output.Single<ColumnName>(c => ("Lbl" + c.x_ColumnName ==
                                tmpcontrol.Text)).LblFieldCol = i;
Output.Single<ColumnName>(c => ("Lbl" + c.x_ColumnName ==
                                tmpcontrol.Text)).LblFieldRow = j; }}}
```

รูปที่ 3.36 ตัวอย่าง Code Program ในการจัดการรูปแบบฟอร์ม

6) เขียนโปรแกรมในการเลือกธีมโดยสร้าง เทมเพลตไว้เป็น Default เมื่อผู้ใช้เลือกสีพื้นหลัง, รูปภาพ และลักษณะของกริดวิว ก็ทำการ Replace ค่าต่างลงใน Skin File ตัวอย่างดังรูปที่ 3.37

```
Header                                          //file stytlesheet.txt
{position: relative;
background: #[:colour:];}
if (CBOBlackground.Text == "None")              //file Form1.cs
{ generate.ReplaceObj("colour", "ffffff");
generate.ReplaceObj("h1", "000000");
generate.ReplaceObj("header", "ffffff");
generate.ReplaceObj("tree", "000000");    }
```

รูปที่ 3.37 แสดงการ Replace สีพื้นหลังจากการเลือกของผู้ใช้งาน

7) ออกแบบเทมเพลตโดยสร้างเทมเพลตจาก Code ASP.NET

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master"
AutoEventWireup="true" CodeFile="Login__View.aspx.cs" Title="ddd" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head"
Runat="Server"></asp:Content><asp:Content ID="Content2"
ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
<asp:Label ID="Label1" runat="server" Text="ใส่ข้อมูลที่ต้องการ
ค้นหา"BorderColor="Black" ForeColor="Black"></asp:Label>
 <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
 <asp:Button ID="Button1" runat="server" Text="Search"
onclick="Button1_Click" />
```

รูปที่ 3.38 Code ASP.NET ที่ใช้ในการออกแบบเทมเพลต

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master"
AutoEventWireup="true" CodeFile="[:Name:].aspx.cs" Inherits="[:Name:]"
Title="[:Title:]" %> <asp:Content ID="Content1" ContentPlaceHolderID="head"
Runat="Server"></asp:Content><asp:ContentID="Content2"ContentPlaceHolderID="Cont
entPlaceHolder1"Runat="Server"> <table style="width:100%;"><tr>  <td>
            [:gv:]
        <asp:Button ID="Button3" runat="server" PostBackUrl="~/[:back:]_Insert.aspx"
            Text="Insert" />  </table></asp:Content>
```

รูปที่ 3.39 การออกแบบเทมเพลตจาก Code ASP.NET

จากรูปที่ 3.39 การออกแบบเทมเพลต [:Name:] คือตัวแปรที่จะทำการแทนที่โดยใช้ Regular Expression ค้นหาคำและ แทนที่คำที่ต้องการเพื่อให้ถูกต้องตามรูปแบบของเว็บไซต์ ASP.NET โดยสร้างเทมเพลตไว้ดังนี้

- TemplateView.txt
- TemplateInsert.txt
- TemplateUpdate.txt
- TemplateDelete.txt
- TemplateMasterPage.txt
- TemplateGridviews.txt
- TemplateDetailViews.txt
- TemplateSkinfile.txt

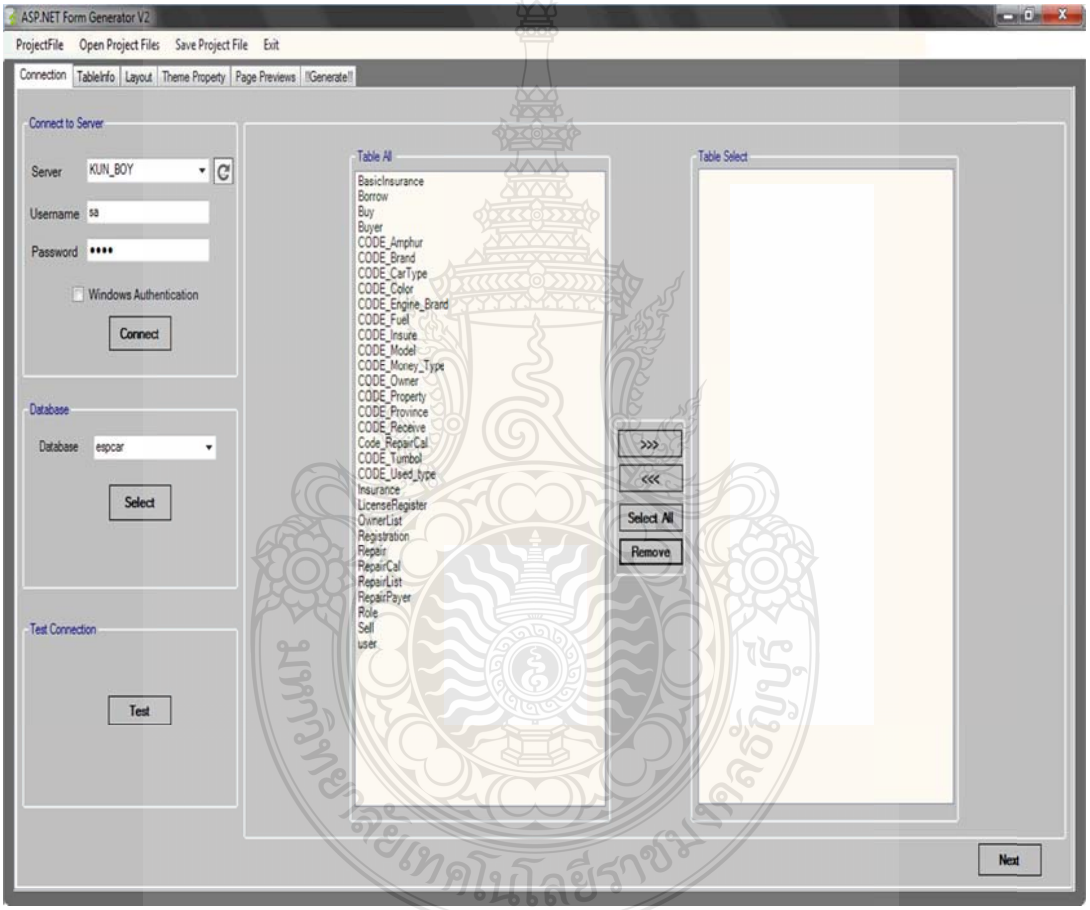
8) เขียนโปรแกรมในการ Generate โดยการอ่านไฟล์จากการจัดการรูปแบบฟอร์ม และ Template ที่จัดทำไว้โดยใช้ Regular Expression ในการค้นหาตัวแปรที่เรากำหนดไว้ เช่น [:label:] เมื่อเจอตัวแปรก็ทำการแทนค่าใหม่เข้าไปเพื่อให้ถูกต้องตามรูปแบบของ ASP.NET โดยจะสร้างไฟล์เทมเพลตไว้ดังนี้ เขียนโปรแกรม โมดูล Generate ดังรูปที่ 3.40

```
generate.OutputFilename = folderBrowserDialog1.SelectedPath + "\\" +
"MasterPage.master";                                    // เลือกที่เก็บไฟล์ output
rate.ReadInputFile("Template\\TemplateMasterPage.txt");  // อ่านไฟล์จากเทมเพลต
generate.ReplaceObj("Title", TBTitle.Text);              // ค้นหาและแทนที่คำ
            if (radioButton1.Checked == true)
             {generate.ReplaceObj("header", TBHeader.Text); }
             else{generate.ReplaceObj("header", "<asp:Image ID=" + "\"" + "Image1"
                                    + "\"" + " runat=" + "\"" + "server" + "\"" + "
                                    ImageUrl=" + "\"" + "~/Images/header.gif" + "\""
                                    + " />");}
            if (radioButton3.Checked == true)
             {generate.ReplaceObj("footer", TBFooter.Text); }
             else{generate.ReplaceObj("footer", "<asp:Image ID=" + "\"" + "Image2" + "\""
                                    + " runat=" + "\"" + "server" + "\"" + " ImageUrl="
                                    + "\"" + "~/Images/footer.gif" + "\"" + " />");}
           foreach (TableName table in tbl.Where(c=> c.x_TableSelect))
             {generate.ReplaceObj("Node", node);
              generate.ReplaceObj("NText", table.x_Table_Description);
              generate.ReplaceObj("LText", table.x_TableName + "_View");  }
            generate.ReplaceObj("Node", "");
            generate.WriteOutputFile();                  // เขียนไฟล์ทั้งหมด
```

รูปที่ 3.40 วิธีการ Generate ไฟล์ และเขียนไฟล์ไปยังไดเร็กทอรี่

3.9 การออกแบบหน้าจอระบบ ASP.NET FORMS GENERATOR VERSION 2

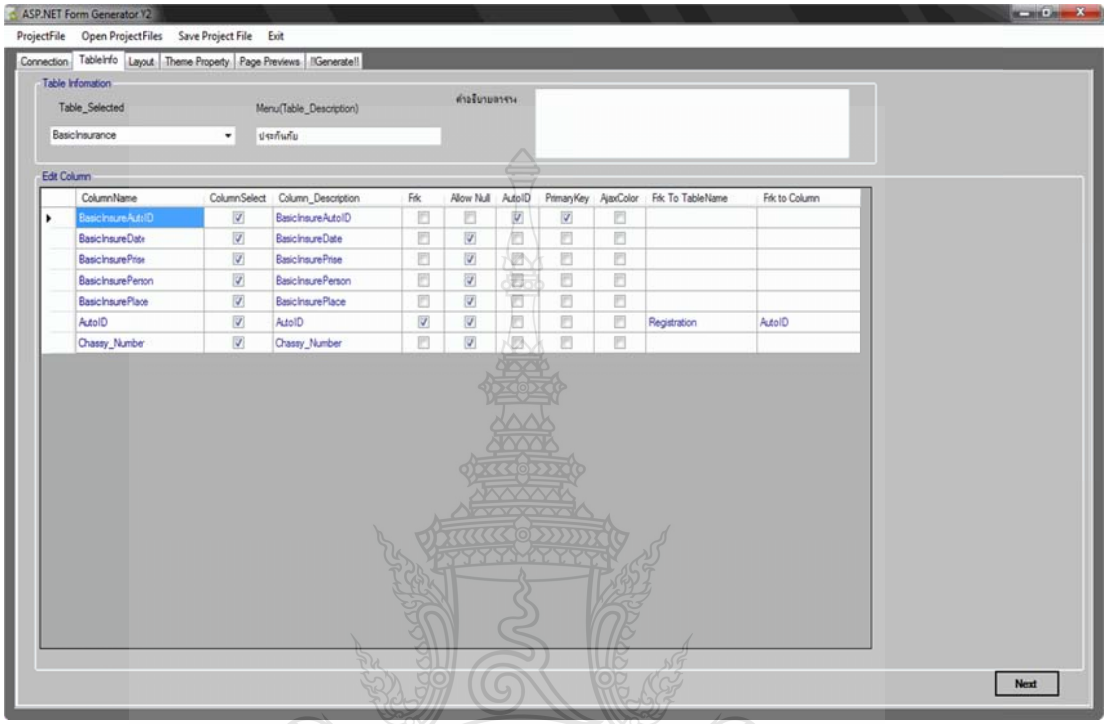
การออกแบบโปรแกรมการใช้งานก็เป็นส่วนสำคัญ เพื่อผู้ใช้สามารถใช้งานได้ง่าย สะดวกสบาย ดูแล้วเข้าใจการทำงานของโปรแกรม ซึ่งโครงงานนี้ได้ใช้การออกแบบหน้าจอระบบของ (Version1) เก็บทั้งหมดโดยจะออกแบบเพิ่มในส่วนของหน้าจอการจัดรูปแบบฟอร์ม ใส่คำอธิบายตาราง ใส่รูปเป็นพื้นหลัง ใส่รูปทำสไลด์โชว์รูปได้ และส่วนใหญ่จะเป็นการเพิ่ม และแก้ไข Code ให้แต่ละส่วนของระบบให้มีความสามารถในการทำงานมากกว่าเดิม



รูปที่ 3.41 การออกแบบการเชื่อมต่อฐานข้อมูล และเลือกตาราง

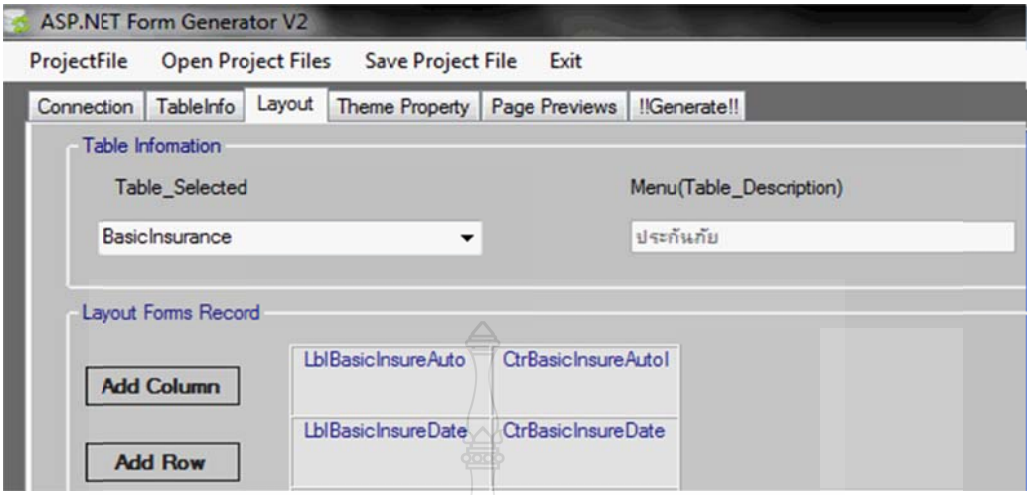
ในรูปที่ 3.41 ในส่วนของการเชื่อมต่อกับฐานข้อมูลจะประกอบด้วยการเลือกเซิร์ฟเวอร์ การเลือกรูปแบบการเชื่อมต่อว่าจะเป็นแบบ SQL Authentication หรือ Windows Authentication การกรอก User และ Password ในส่วนของการเลือกฐานข้อมูลจะแสดงชื่อของฐานข้อมูลให้ผู้ใช้

เลือก ในส่วนของการแสดงตารางทั้งหมดจะแสดงรายชื่อของตารางทั้งหมดจากฐานข้อมูลที่เลือก ในส่วนของการแสดงตารางที่เลือก จะแสดงตารางที่ผู้ใช้เลือกไว้ทั้งหมด



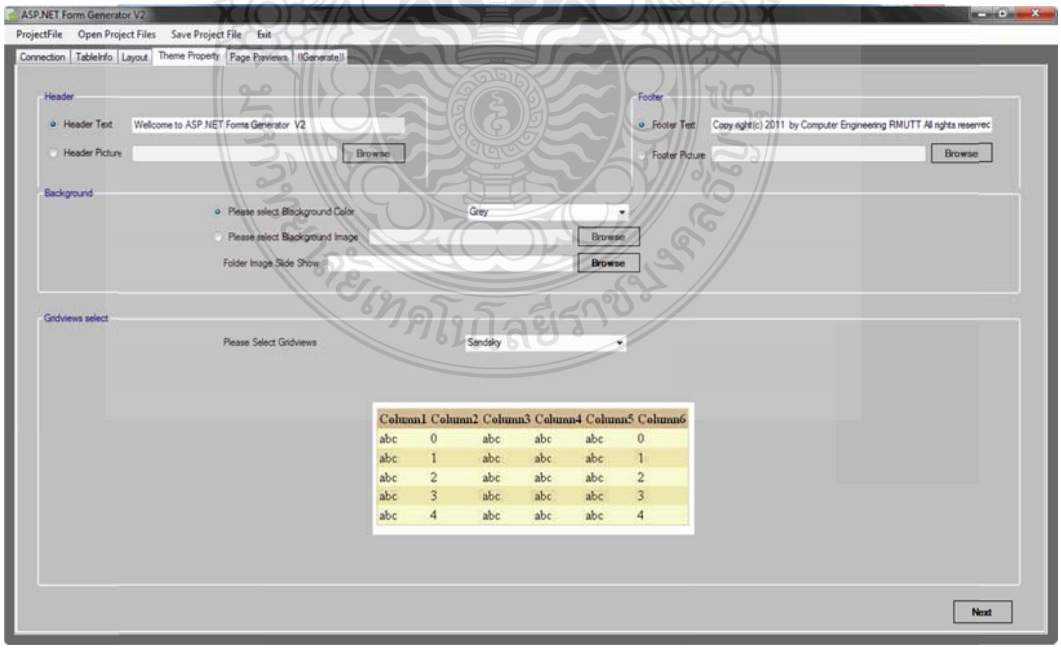
รูปที่ 3.42 การออกแบบหน้าจอแสดงคอลัมน์และการเลือกคอลัมน์ของตารางที่เลือกไว้

จากรูปที่ 3.42 ในส่วนของการแสดงตารางที่เลือก จะแสดงรายชื่อของตารางทั้งหมดที่ผู้ใช้ได้เลือกไว้และสามารถแก้ไขรายละเอียดของตารางได้ ในส่วนของการแสดงคอลัมน์และรายละเอียดต่างๆ ของคอลัมน์ จะแสดงรายชื่อคอลัมน์ภายในตารางที่เลือก และแสดงรายละเอียดต่างๆ สามารถเลือกได้ว่าจะใช้คอลัมน์ใดบ้าง และได้เพิ่มให้สามารถใส่คำอธิบายตารางได้ ข้อความจะไปแสดงที่หน้าเว็บเพท เมื่อผู้ใช้คลิกที่ปุ่มจะแสดงข้อความนั้น

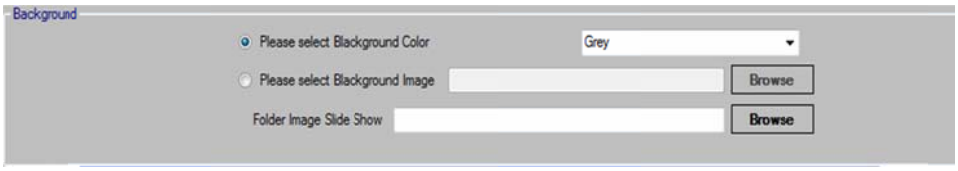


รูปที่ 3.43 การออกแบบหน้าจอการจัดรูปแบบฟอร์มของผู้ใช้

จากรูปที่ 3.43 การจัดรูปแบบฟอร์มโดยจะมีปุ่มกดสองปุ่มกดคือ ปุ่มเพิ่ม Column และปุ่มเพิ่ม Row ให้ผู้ใช้สามารถเพิ่มลดและจัดตำแหน่ง ข้อความและคอนโทรลได้ตามต้องการ หรือจะใช้รูปแบบที่ระบบมีให้ หากมีการปรับแต่งการจัดรูปแบบฟอร์มข้อความและคอนโทรล จะมีผลในส่วนการเพิ่มข้อมูล และแก้ไขข้อมูลเท่านั้น ระบบจะเก็บตำแหน่งที่ผู้ใช้ได้ปรับตำแหน่งของข้อความหรือคอนโทรล เพื่อทำการ Generate

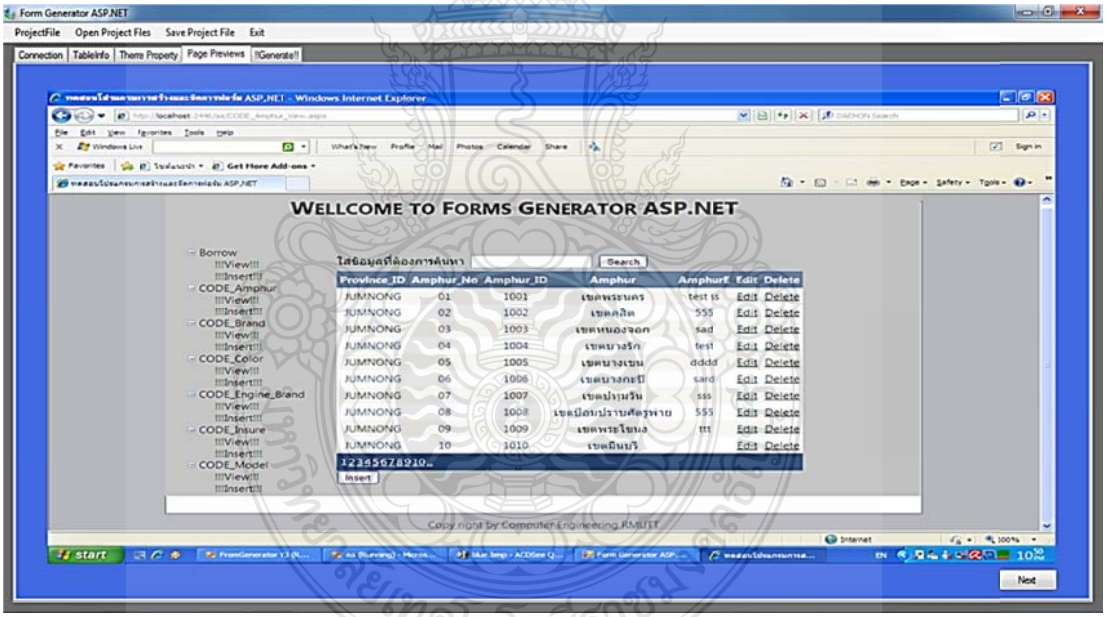


รูปที่ 3.44 การออกแบบหน้าจอการแก้ไข Header, Footer และเลือกพื้นหลังของ Master Page



รูปที่ 3.45 การออกแบบหน้าจอเพิ่มจากโครงงานเดิมสามารถใส่ Slide ,Background Image

จากรูปที่ 3.44 ในส่วนของการเลือกธีมผู้ใช้สามารถแก้ไข Header, Footer จะสามารถเลือกรูปภาพจากเครื่องผู้ใช้ได้ หรือใส่ข้อความในส่วน Header และ ส่วน Footer ได้ ในส่วนของการเลือกสีพื้นหลัง (Back Ground) จะมีให้ผู้ใช้เลือกคือ สีน้ำเงิน สีเทา สีขาว สีดำ และสีฟ้าคราม และจากรูป 3.45 ในโครงงานนี้ได้เพิ่มความสามารถในการใส่รูปเป็นพื้นหลังและ ใส่รูปทำสไลด์รูปโชว์ได้ โดยใช้ความสามารถของ AJAX



รูปที่ 3.46 การออกแบบหน้าตัวอย่างของเว็บไซต์ก่อนการ Generate

จากรูปที่ 3.46 ในส่วนของการแสดงตัวอย่างเว็บไซต์ก่อนการ Generate จะแสดงตัวอย่างหน้าเว็บไซต์แบบคร่าวๆ ให้ผู้ใช้งานดูก่อนการ Generate

รูปที่ 3.47 การออกแบบของการ Generate

จากรูปที่ 3.47 ส่วนของการเลือกที่จัดเก็บไฟล์ ผู้ใช้งานต้องทำการเลือก Directory ที่ต้องการจัดเก็บไฟล์ด้วย และในส่วนของการ Generate จะแสดงปุ่มสำหรับกดเพื่อทำการ Generate ไฟล์ทั้งหมดเมื่อ Generate เสร็จเรียบร้อยจะมีการรายงานผลการ Generate

บทที่ 4
การทดสอบการใช้งาน

การออกแบบแอพพลิเคชั่นการสร้างและการจัดการฟอร์ม นั้นเพื่อต้องการที่จะเพิ่มช่องทางในการสร้างเว็บไซต์เกี่ยวกับฐานข้อมูล นำไปช่วยในการกรองข้อมูล เพื่อเอาผลการทดลองมาใช้งานและหาความผิดพลาดที่จะเกิดขึ้น ส่วนในการทดสอบระบบนี้จะทดสอบโดยแยกการทดสอบออกไปตามประเภทการใช้งานแต่ละส่วน โดยจะใช้การทดสอบแบบ Black Box

4.1 การทดสอบการเชื่อมต่อกับฐานข้อมูลของผู้ใช้

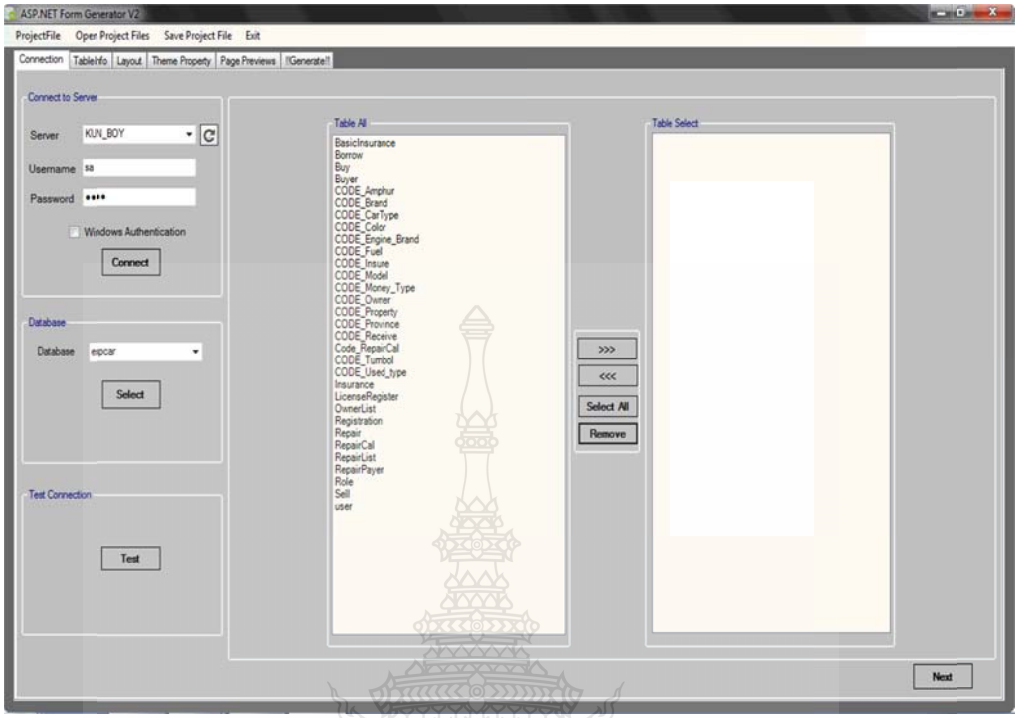
ในโครงงานนี้ใช้การทดสอบการเชื่อมต่อกับฐานข้อมูลเหมือนกับ ASP.NET FORMS GENERATOR โดยแบ่งการทดสอบออกเป็น 2 Mode คือการเชื่อมต่อกับฐานข้อมูลแบบ SQL Authentication Mode และ Windows Authentication Mode

ตารางที่ 4.1 แสดงรายละเอียดการเชื่อมต่อกับฐานข้อมูลด้วย SQL Authentication Mode

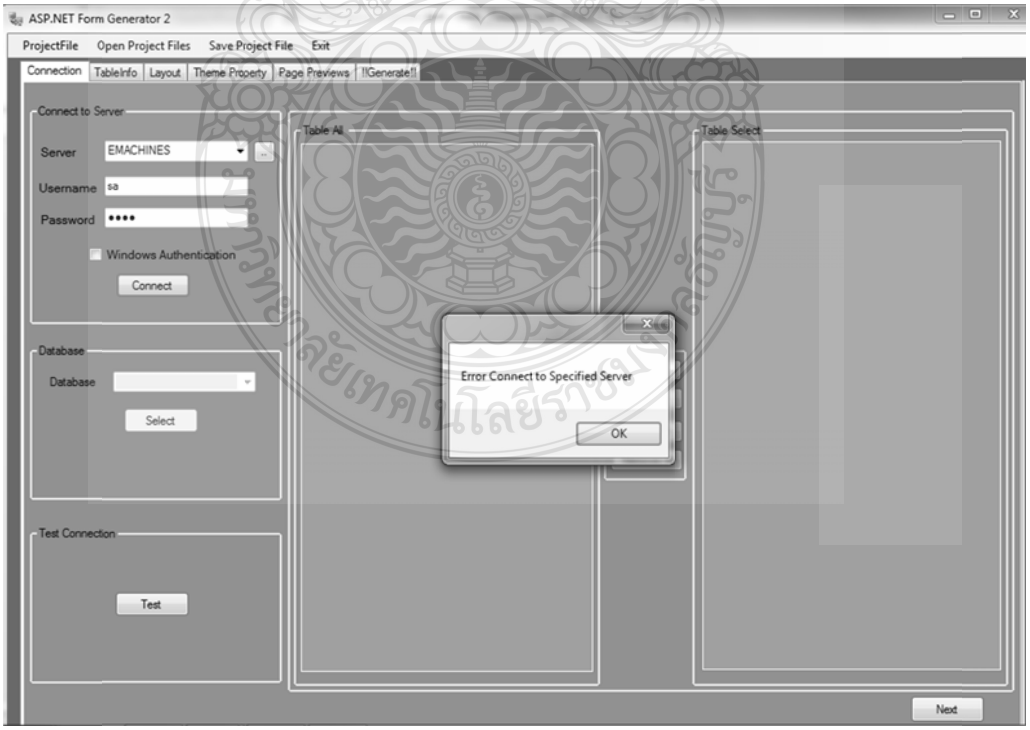
ชื่อฐานข้อมูล	รายละเอียดฐานข้อมูล	จำนวนตาราง	ผลการเชื่อมต่อ
1) espcar	ฐานข้อมูลทะเบียนรถ	31 ตาราง	สำเร็จ
2) esp2008	ฐานข้อมูลนักเรียน	118 ตาราง	สำเร็จ
3) Data_ST	ฐานข้อมูลเบิก-คืนอุปกรณ์	8 ตาราง	สำเร็จ
4) NorthWind	ฐานข้อมูลสั่งซื้อสินค้า	8 ตาราง	สำเร็จ
5) Special	ฐานข้อมูลทะเบียนรถ v.2	9 ตาราง	สำเร็จ

ตารางที่ 4.2 แสดงรายละเอียดการเชื่อมต่อกับฐานข้อมูลด้วย Windows Authentication Mode

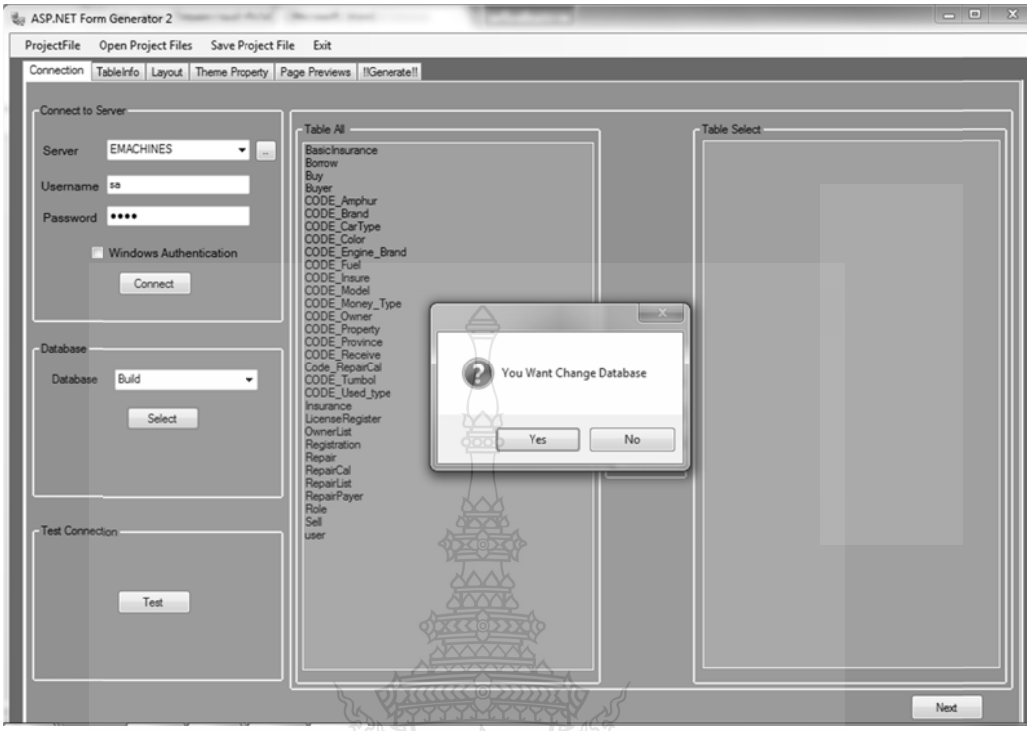
ชื่อฐานข้อมูล	รายละเอียดฐานข้อมูล	จำนวนตาราง	ผลการเชื่อมต่อ
1) espcar	ฐานข้อมูลทะเบียนรถ	31 ตาราง	สำเร็จ
2) esp2008	ฐานข้อมูลนักเรียน	118 ตาราง	สำเร็จ
3) Data_ST	ฐานข้อมูลเบิก-คืนอุปกรณ์	8 ตาราง	สำเร็จ
4) NorthWind	ฐานข้อมูลสั่งซื้อสินค้า	8 ตาราง	สำเร็จ
5) Special	ฐานข้อมูลทะเบียนรถ	9 ตาราง	สำเร็จ



รูปที่ 4.1 แสดงผลการเชื่อมต่อเมื่อเชื่อมต่อสำเร็จ



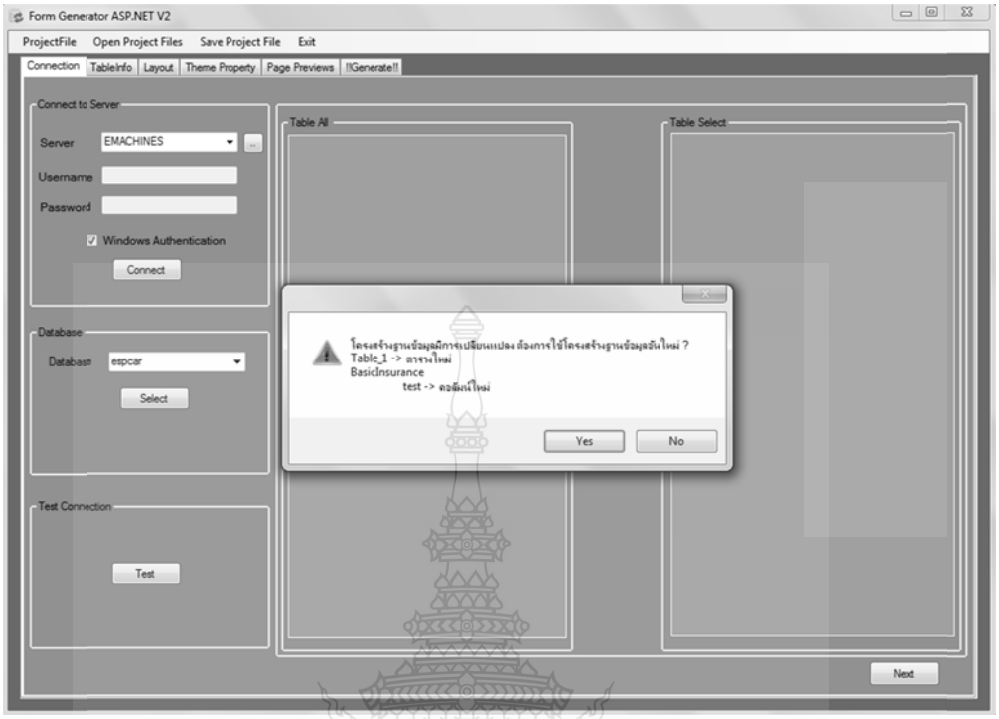
รูปที่ 4.2 แสดงผลการเชื่อมต่อ เมื่อเชื่อมต่อไม่สำเร็จ



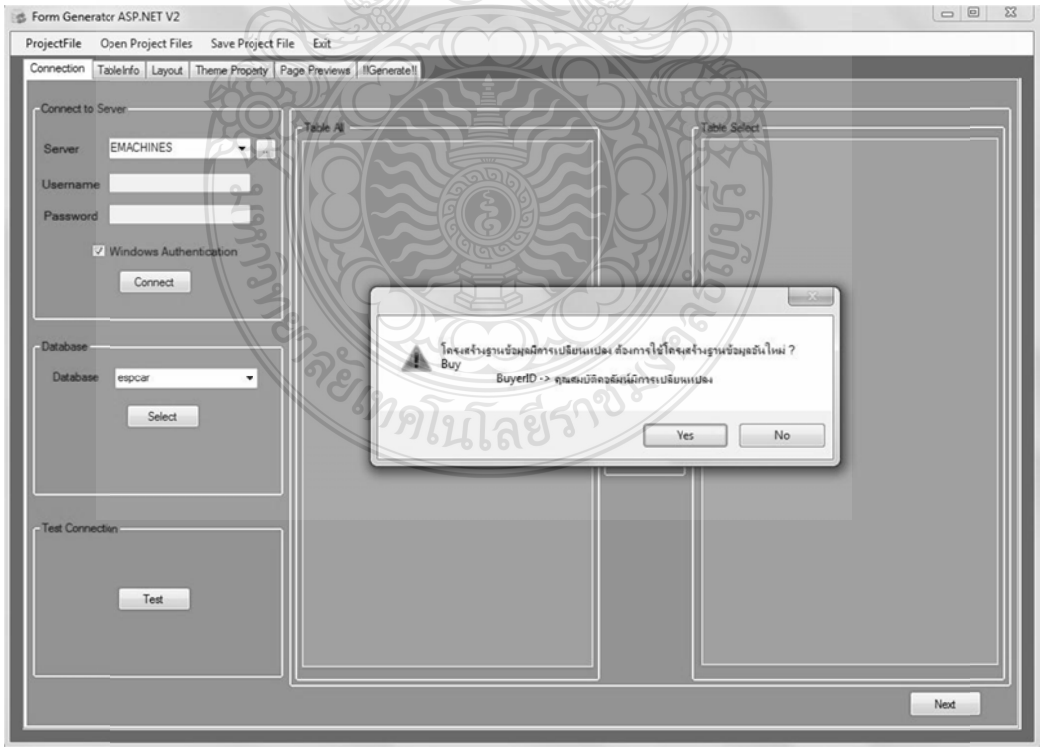
รูปที่ 4.3 แสดงข้อความแจ้งเตือน เมื่อมีการเปลี่ยนฐานข้อมูล

4.2 การทดสอบการตรวจสอบการ Update Database Schema

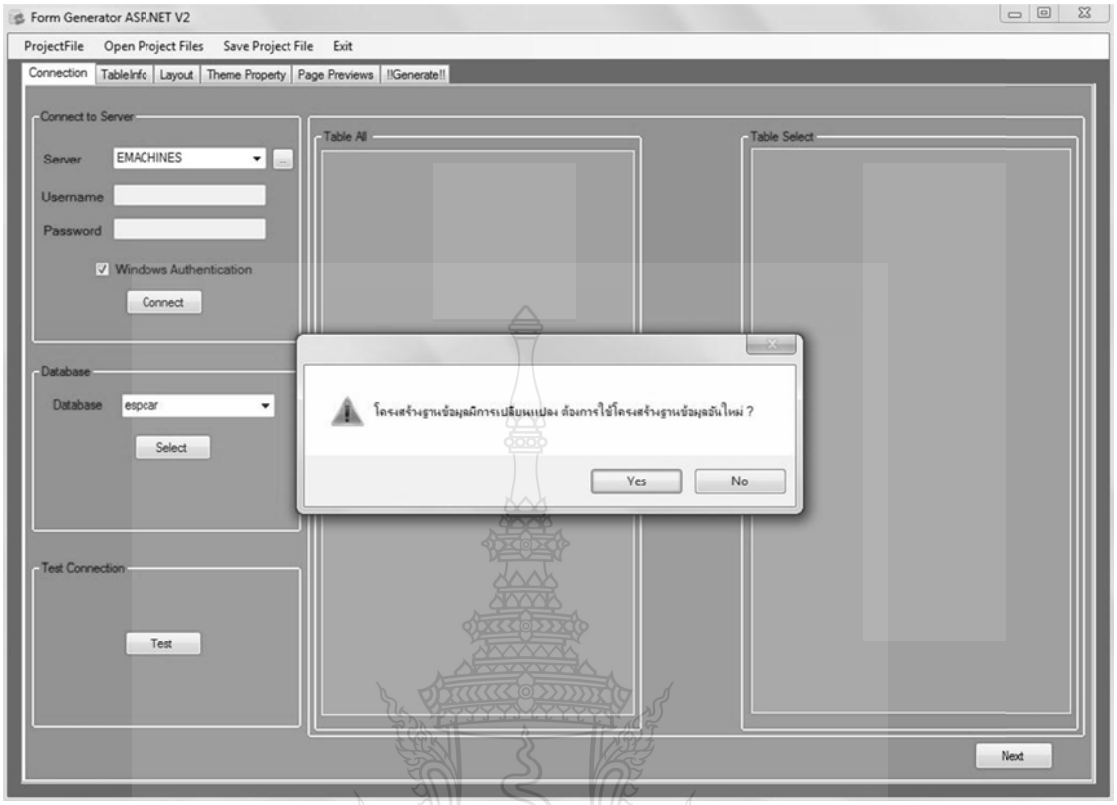
การทดการการตรวจสอบการ Update Database Schema นี้เป็นส่วนที่เพิ่มขึ้นมาจากโครงงานเดิม จะทำการตรวจสอบเมื่อผู้ใช้ทำการโหลดไฟล์งานเดิมขึ้นมา เมื่อระบบตรวจสอบพบมีการ Update Database Schema ระบบจะแจ้งเตือนให้เลือกว่าจะใช้โครงสร้างฐานข้อมูลอันใหม่ แต่ถ้าไม่เลือกจะใช้โครงสร้างฐานข้อมูลอันเดิมในการ Generate



รูปที่ 4.4 แสดงข้อความแจ้งเตือน เมื่อมีตารางและคอลัมน์ใหม่เพิ่มขึ้นมา



รูปที่ 4.5 แสดงข้อความแจ้งเตือน เมื่อมีคุณสมบัติของคอลัมน์มีการเปลี่ยนแปลง



รูปที่ 4.6 แสดงข้อความแจ้งเตือน เมื่อมีตารางหรือคอลัมน์ถูกลบออกจากฐานข้อมูล

4.3 การทดสอบการเลือกตาราง และการเลือกคอลัมน์

การทดสอบการเลือกตารางในการ Generate ผลที่ได้ดังตารางที่ 4.3

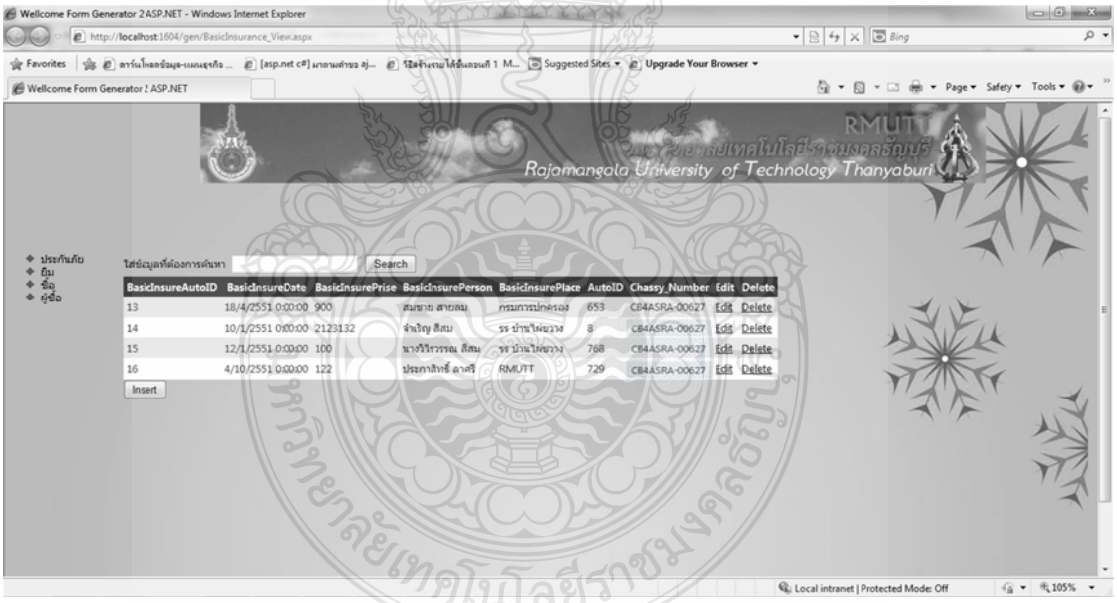
ตารางที่ 4.3 แสดงผลการทดสอบการเลือกตาราง

ชื่อฐานข้อมูล	รายละเอียดฐานข้อมูล	จำนวนตาราง	ตารางที่เลือกได้
1) espcar	ฐานข้อมูลทะเบียนรถ	31 ตาราง	31 ตาราง
2) esp2008	ฐานข้อมูลนักเรียน	118 ตาราง	118 ตาราง
3) Data_ST	ฐานข้อมูลเบิก-คืนอุปกรณ์	8 ตาราง	8 ตาราง
4) NorthWind	ฐานข้อมูลสั่งซื้อสินค้า	8 ตาราง	8 ตาราง
5) Special	ฐานข้อมูลทะเบียนรถ	9 ตาราง	9 ตาราง

ตารางที่ 4.4 แสดงผลการทดสอบการเลือกคอลัมน์

ชื่อฐานข้อมูล	รายละเอียดฐานข้อมูล	จำนวนคอลัมน์	คอลัมน์ที่เลือกได้
1) espcar	ฐานข้อมูลทะเบียนรถ	178 คอลัมน์	178 คอลัมน์
2) esp2008	ฐานข้อมูลนักเรียน	320 คอลัมน์	320 คอลัมน์
3) Data_ST	ฐานข้อมูลเบิก-คืนอุปกรณ์	35 คอลัมน์	35 คอลัมน์
4) NorthWind	ฐานข้อมูลสั่งซื้อสินค้า	76 คอลัมน์	76 คอลัมน์
5) Special	ฐานข้อมูลทะเบียนรถ	45 คอลัมน์	45 คอลัมน์

ทดสอบการ Generate จากฐานข้อมูล espcar โดยเลือก 4 ตารางในการ Generate แสดงผลดังรูปที่ 4.7

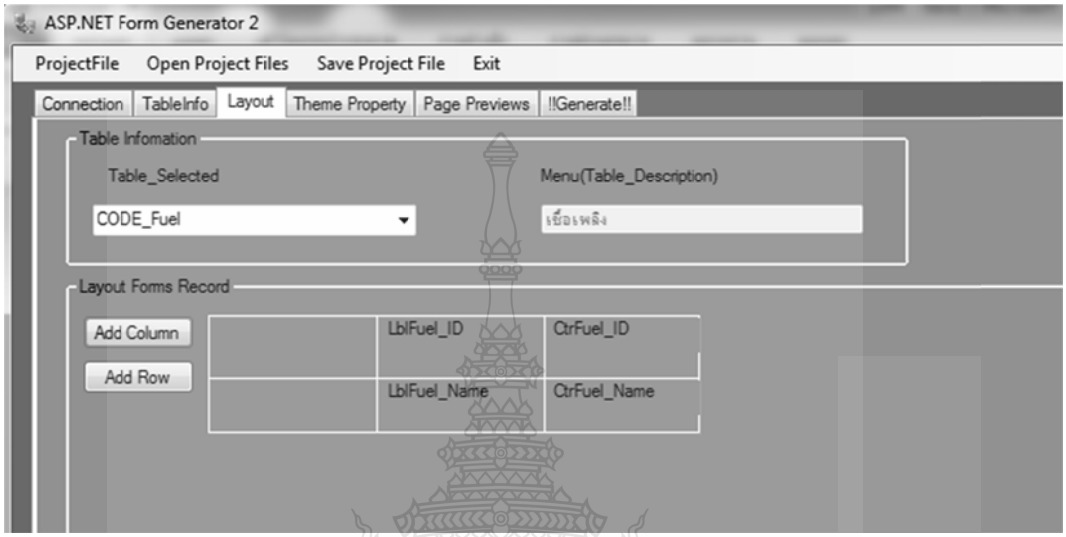


รูปที่ 4.7 แสดงเว็บไซต์เมื่อ Generate จากฐานข้อมูล espcar และเลือก 4 ตาราง

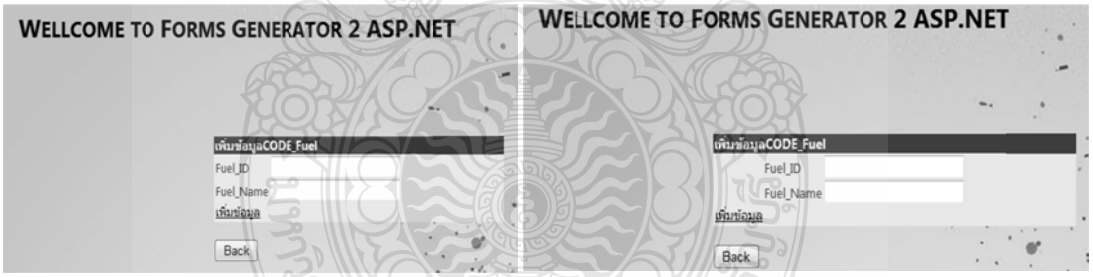
4.4 การทดสอบการจัดรูปแบบฟอร์มข้อมูล และการเลือกธีม (Theme)

ในโครงงานนี้ได้เพิ่มการจัดรูปแบบฟอร์มข้อมูล เพิ่มขึ้นจากโครงงานเดิม โดยในส่วนของการทดสอบการจัดรูปแบบฟอร์มข้อมูลนี้ ผู้ใช้สามารถจัดว่างตำแหน่งของ ข้อความและคอนโทรล

จะแสดงผลในส่วนของการ เพิ่มข้อมูลและการลบข้อมูล เท่านั้น ผลปรากฏว่าสามารถจัดรูปแบบฟอร์มข้อมูลได้ ดังรูปที่ 4.8



รูปที่ 4.8 แสดงการจัดรูปแบบฟอร์มข้อมูล



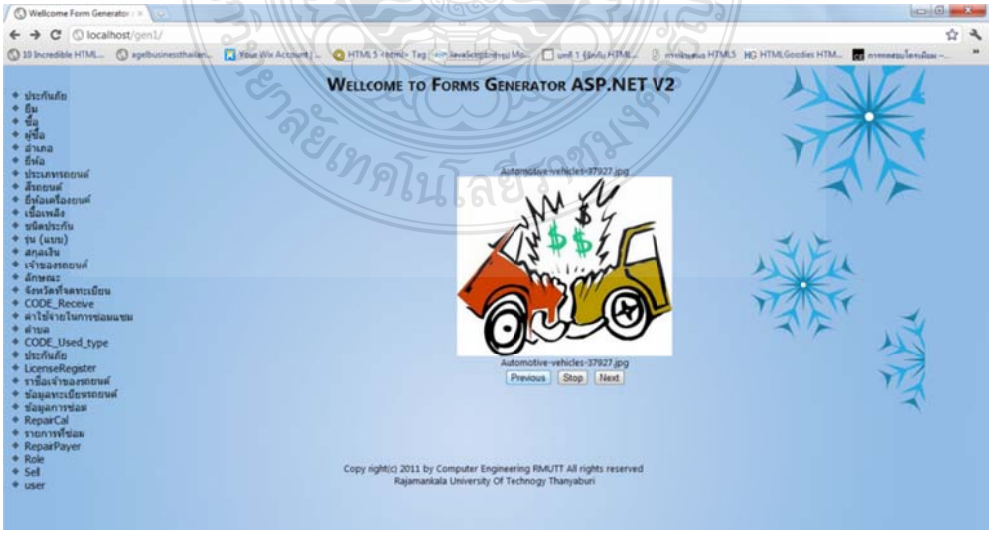
ก่อนจัดรูปแบบฟอร์มข้อมูล หลังจัดรูปแบบฟอร์มข้อมูล

รูปที่ 4.9 แสดงผลการจัดรูปแบบฟอร์มข้อมูล

ตารางที่ 4.5 แสดงรายละเอียดการทดสอบการเลือกธีม

การทดสอบการเลือกธีม	ครั้งที่1	ครั้งที่2	ครั้งที่3	ครั้งที่4	ครั้งที่5	ผลการทดสอบ
Header	ใช้รูปภาพ	ใช้รูปภาพ	ใช้ตัวอักษร	ใช้ตัวอักษร	ใช้ตัวอักษร	สำเร็จแสดงในรูป 4.10
Footer	ใช้ตัวอักษร	ใช้ตัวอักษร	ใช้ตัวอักษร	ใช้รูปภาพ	ใช้ตัวอักษร	
Background Color	ไม่ใช้	Blue	Grey	ไม่ใช้	ไม่ใช้	
Background Image	ใช้รูปภาพ	ไม่ใช้	ไม่ใช้	ใช้รูปภาพ	ใช้รูปภาพ	
Image SlideShow	ใช้	ใช้	ไม่ใช้	ไม่ใช้	ใช้	
Gridviews	Classic	Sandsky	Colorful	Professional	Simple	

ในการทดสอบการเลือกธีมทำการทดสอบโดยการเลือกทุกตารางและทุกคอลัมน์ในฐานข้อมูลแล้วทำการ Generate ไฟล์ออกมาแล้วรันบน IIS Server แสดงผลดังรูป 4.10



รูปที่ 4.10 เมื่อผู้ใช้เลือกพื้นหลังเป็นรูป ใช้ Image Slide Show และใช้รูปเป็น Header

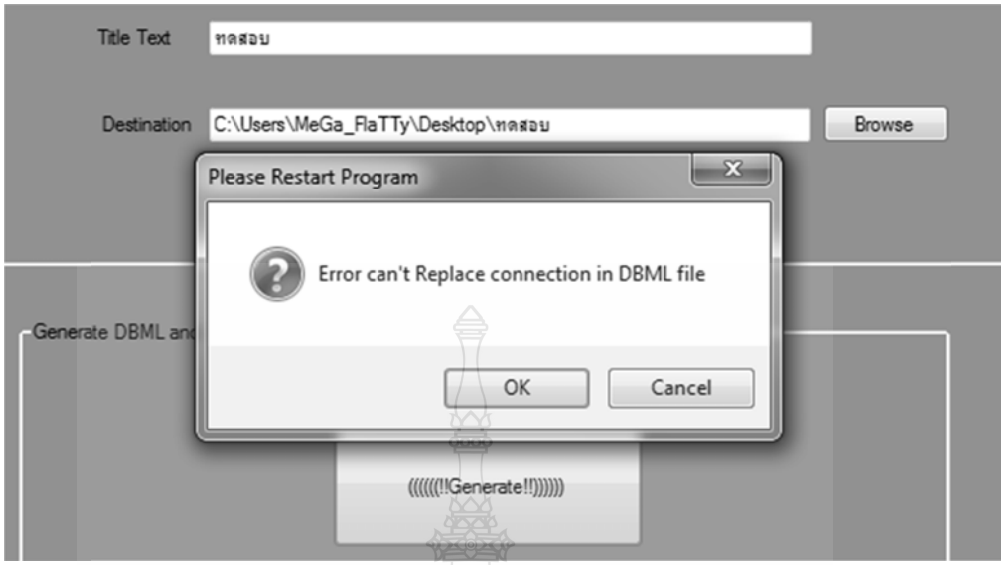
4.5 ทดสอบการ Generate ไฟล์ ASP.NET จากฐานข้อมูล

ตารางที่ 4.6 ผลการทดสอบ Generate ไฟล์

ชื่อฐานข้อมูล	รายละเอียดฐานข้อมูล	จำนวนตาราง	ผลการ Generate	
1) espcar	ฐานข้อมูลทะเบียนรถ	31 ตาราง	สำเร็จ	
2) esp2008	ฐานข้อมูลนักเรียน	118 ตาราง	สำเร็จ	
3) Data_ST	ฐานข้อมูลเบิก-คืนอุปกรณ์	8 ตาราง	สำเร็จ	ดังรูปที่ 4.9
4) NorthWind	ฐานข้อมูลสั่งซื้อสินค้า	8 ตาราง	สำเร็จ	
5) Special	ฐานข้อมูลทะเบียนรถ	9 ตาราง	สำเร็จ	

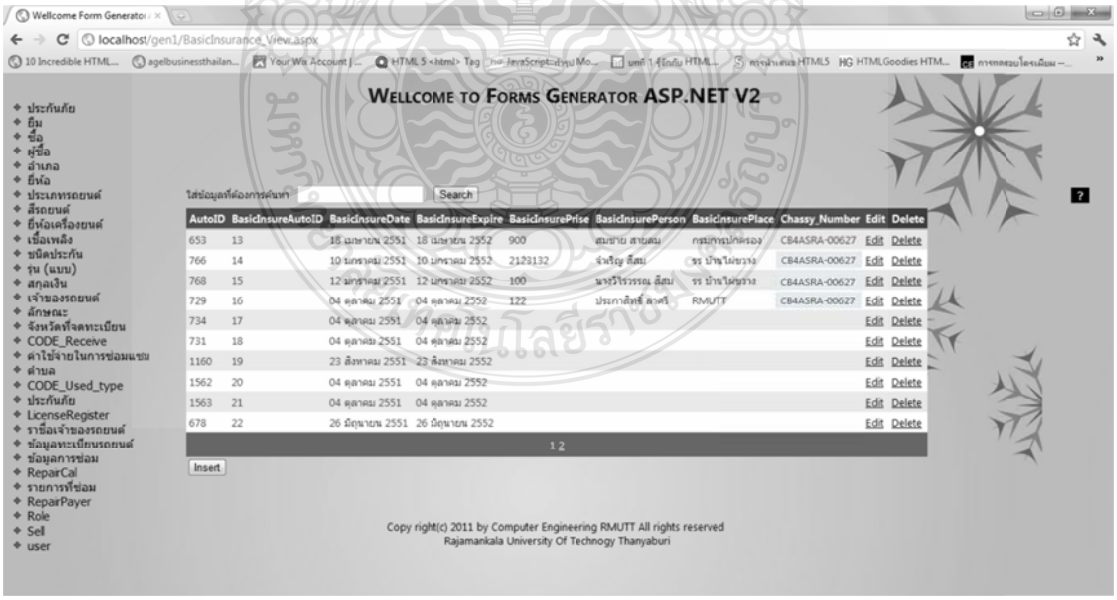
จากตารางสามารถสรุปได้ว่า การสร้างและจัดการฟอร์ม ASP.NET สามารถ Generate เว็บไซต์จากฐานข้อมูลต่างๆได้ และมีการรายงานผลการ Generate ดังรูปที่ 4.11

รูปที่ 4.11 แสดงข้อความเมื่อ Generate Source Code สำเร็จ



รูปที่ 4.12 แสดงข้อความเมื่อ Generate DBML ไม่สำเร็จ

ทดสอบการ Generate จากฐานข้อมูล espcar ซึ่งเป็นฐานข้อมูลทะเบียนรถยนต์ที่ไม่มีความซับซ้อนของระบบมากนัก ทดสอบการ Generate โดยการเลือกทุกตารางทั้งฐานข้อมูล เมื่อ Generate เสร็จแล้วนำไฟล์ที่ได้ทำการรันบน IIS Server แสดงผลดังรูปที่ 4.13

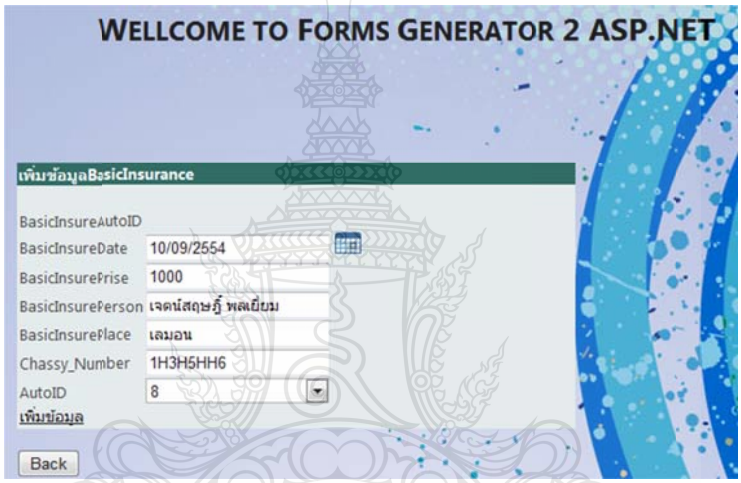


รูปที่ 4.13 แสดงเว็บไซต์เมื่อผู้ใช้เลือก Generate ฐานข้อมูล espcar

4.6 ทดสอบการ เพิ่ม ลบ แก้ไข และ ค้นหาข้อมูล

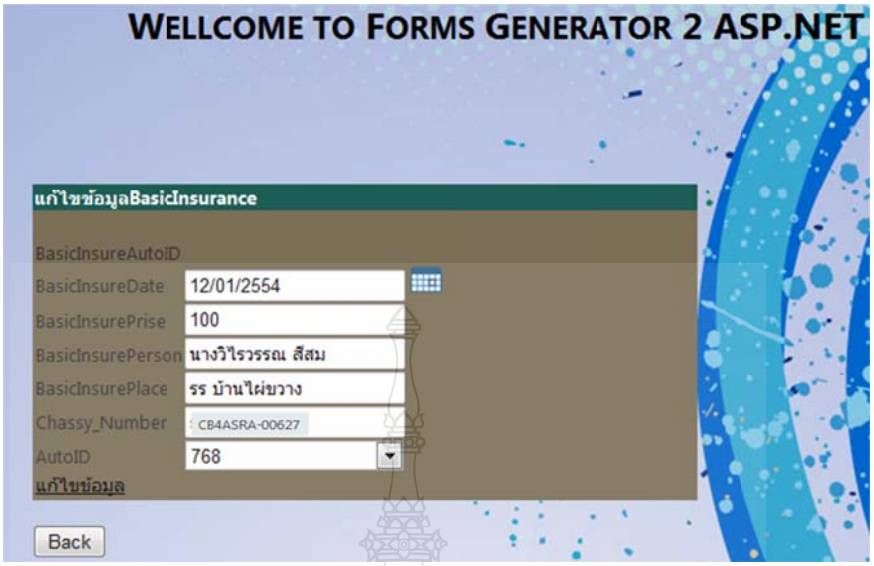
เมื่อ Generate เว็บไซต์เสร็จทำการทดสอบการ เพิ่ม ลบ แก้ไข และค้นหาข้อมูลจากเว็บไซต์ที่ทำการ Generate เสร็จสมบูรณ์ โดยโครงงานนี้ได้ใช้ฟังก์ชันของ AJAX มาใช้อำนวยความสะดวกแก่ผู้ใช้ โดยผู้ใช้คลิกที่รูป ปฏิทิน จะมีวันที่ให้เลือกใช้ ผู้ใช้จึงไม่ต้องพิมพ์ให้เสียเวลา และฟังก์ชันของ AJAX นั้นมีให้ใช้อยู่ส่วนของ Insert และ Delete

4.6.1 ทดสอบการเพิ่มข้อมูลจากเว็บไซต์ที่ทำการ Generate จากฐานข้อมูล espcar ผลปรากฏว่าสามารถเพิ่มข้อมูลได้ดังรูปที่ 4.14



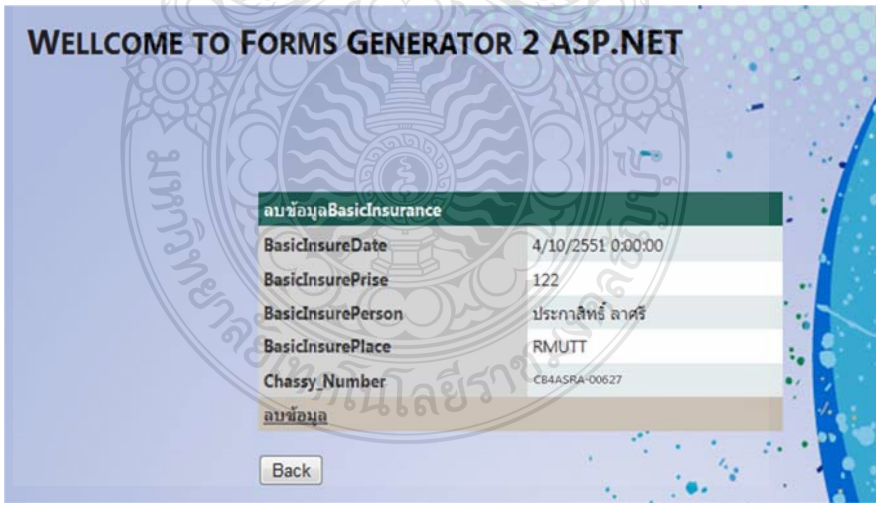
รูปที่ 4.14 แสดงการเพิ่มข้อมูล ลงฐานข้อมูล espcar

4.6.2 ทดสอบการแก้ไขข้อมูลจากเว็บไซต์ที่ทำการ Generate จากฐานข้อมูล Special แล้วผลปรากฏว่าสามารถแก้ไขข้อมูลได้ดังรูปที่ 4.15



รูปที่ 4.15 แสดงการแก้ไขข้อมูล ในฐานข้อมูล espcar

4.6.3 ทดสอบการลบข้อมูลจากเว็บไซต์ที่ทำการ Generate จากฐานข้อมูล espcar ผลปรากฏว่า สามารถลบข้อมูลได้ ดังรูปที่ 4.16

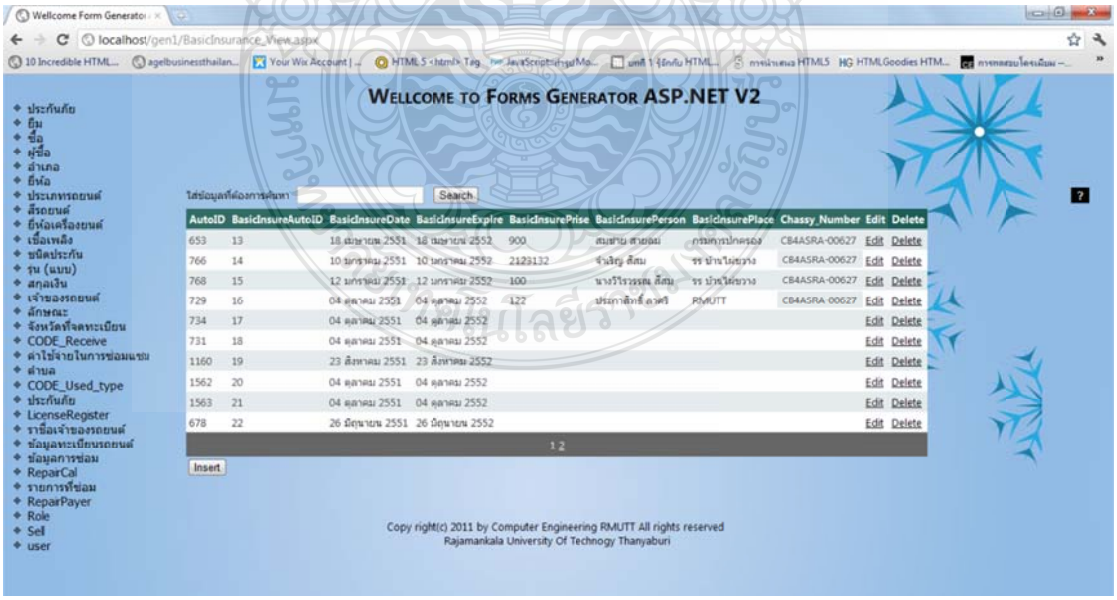


รูปที่ 4.16 แสดงการลบข้อมูลในฐานข้อมูล espcar

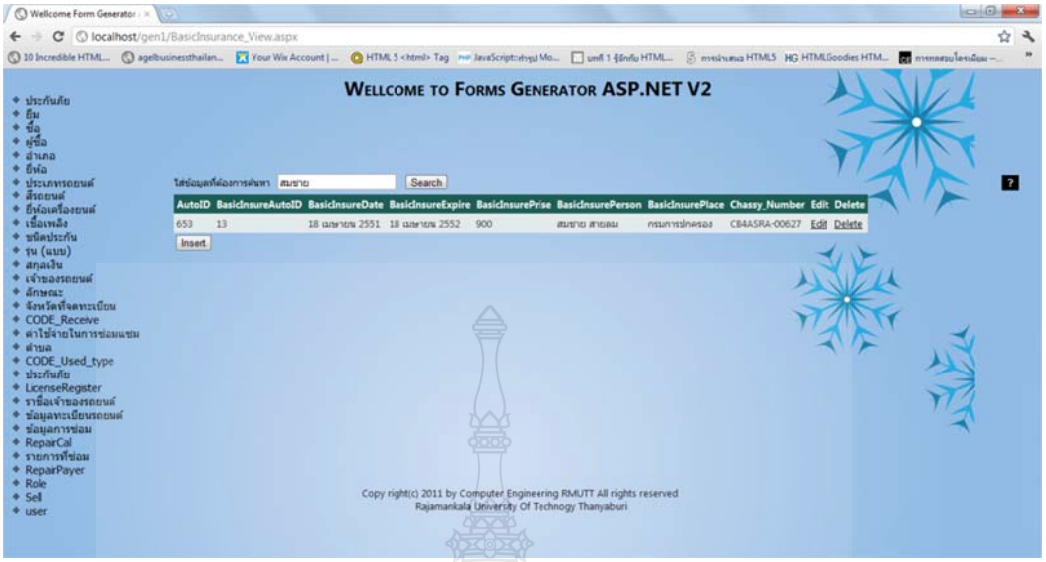
เมื่อลบข้อมูลเรียบร้อยมีข้อความแจ้งว่าลบข้อมูลเรียบร้อยดังรูปที่ 4.17

รูปที่ 4.17 แสดงข้อความแจ้งเตือนเมื่อลบข้อมูล เสร็จเรียบร้อย

4.6.4 ทดสอบการค้นหาข้อมูลจากเว็บไซต์ที่ทำการ Generate จากฐานข้อมูล espcar ผลปรากฏว่า สามารถค้นหาข้อมูลได้ ดังรูปที่ 4.18 และ เมื่อทำการค้นหาชื่อ สมชาย ระบบแสดงข้อมูลดังรูปที่ 4.19



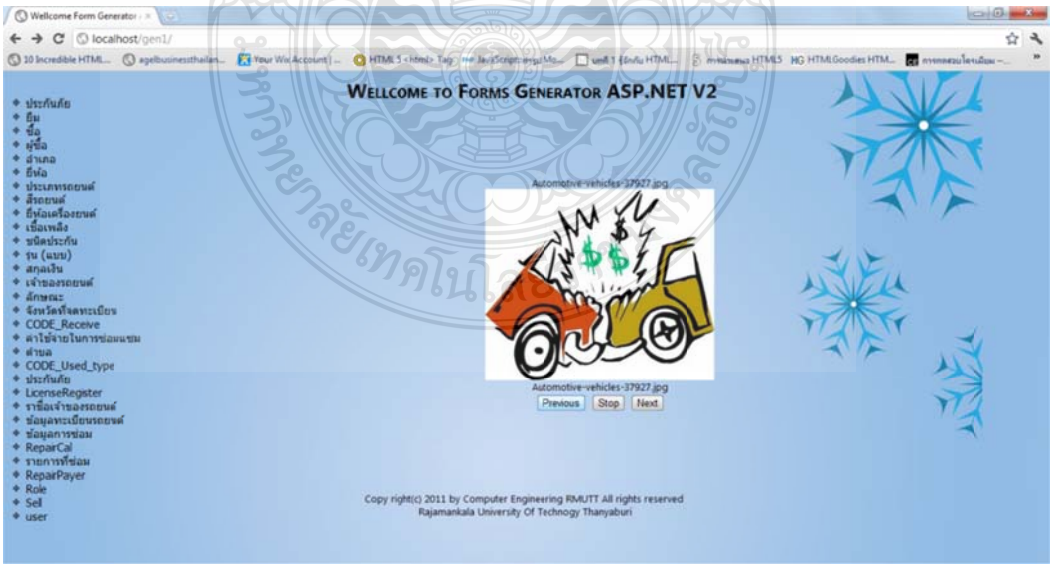
รูปที่ 4.18 แสดงข้อมูลทั้งหมดก่อนการค้นหา



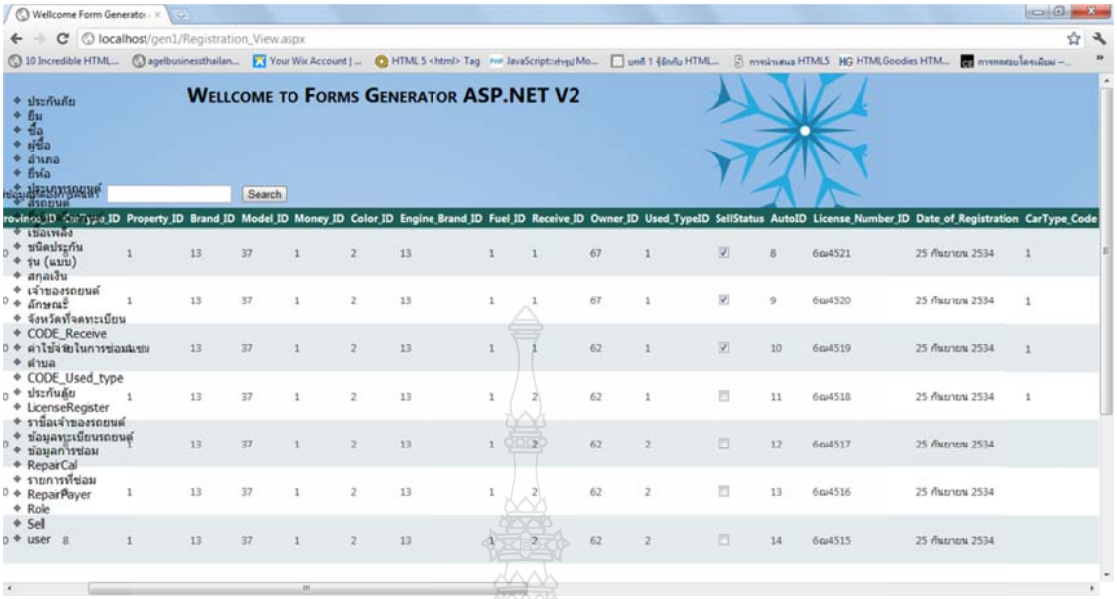
รูปที่ 4.19 แสดงข้อมูลเมื่อค้นพบข้อมูล

4.7 ทดสอบการทำงานของฟังก์ชั่น AJAX

การทดสอบการทำงานของฟังก์ชั่น AJAX ซึ่งได้เพิ่มขึ้นจากโครงงานเดิม โดยใช้ฟังก์ชั่นของ AJAX คือ Slide Show, Always Visible Control, Color Picker, Animation หลังจากทำการเลือกใช้งานผู้ใช้ผลปรากฏว่าสามารถทำงานได้

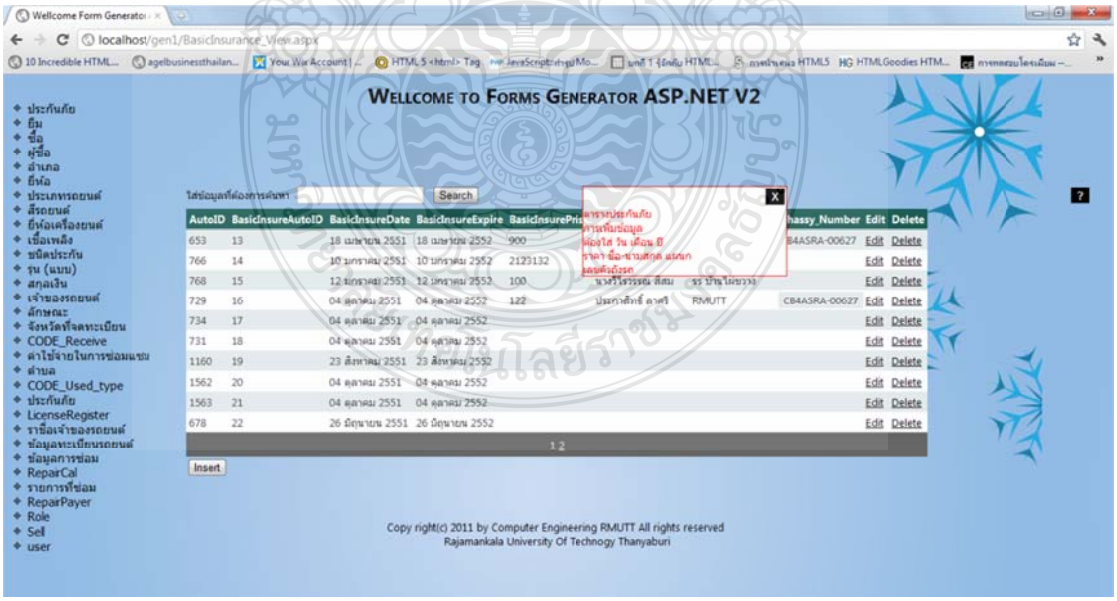


รูปที่ 4.20 แสดงผลการทำงานของฟังก์ชั่น AJAX SlideShow



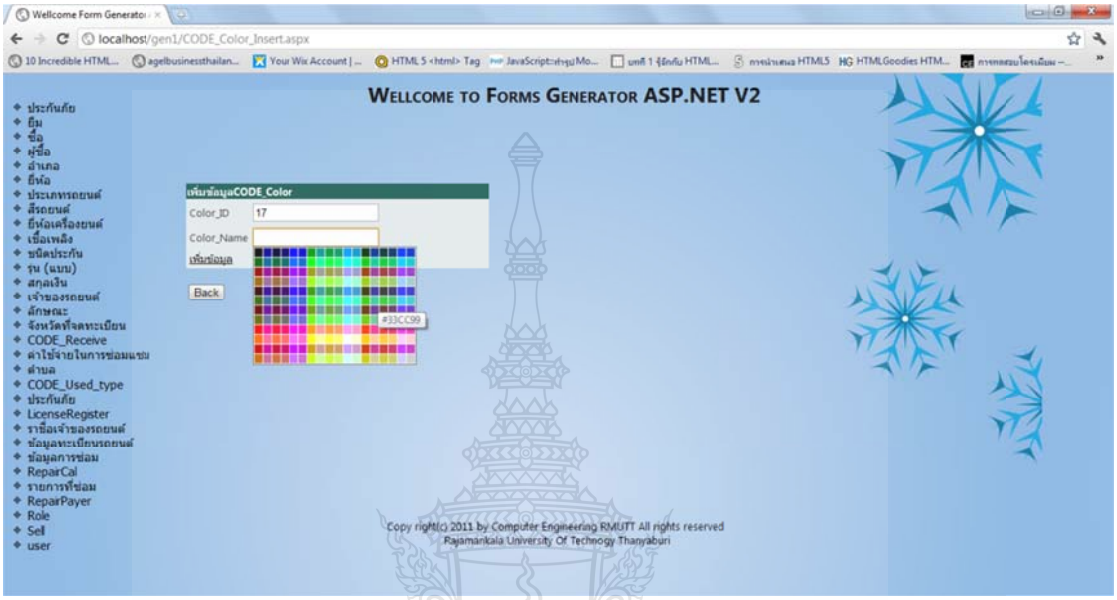
รูปที่ 4.21 แสดงผลการทำงานของฟังก์ชั่น AJAX Always Visible Control

จากรูปที่ 4.21 แสดงผลการทำงานของ Always Visible Control ที่ทำการฟิกตำแหน่งของเมนูไว้ด้านซ้ายจะมีแต่ส่วนหน้าเว็บเพจที่เลื่อนขึ้นลงและเลื่อนซ้ายขวาได้



รูปที่ 4.22 แสดงผลการทำงานของฟังก์ชั่น AJAX Animation

จากรูปที่ 4.22 จะแสดงผลการทำงานของ Animation เมื่อผู้ใช้คลิกที่ปุ่ม ? จะมีกล่องข้อความปรากฏขึ้นมา จากผู้ใช้ที่ได้ใส่ข้อมูลอธิบายรายละเอียดของตารางไว้ในตอนก่อน Generate



รูปที่ 4.23 แสดงผลการทำงานของฟังก์ชั่น AJAX Color Picker

บทที่ 5
สรุปผลของโครงงาน

จากการออกแบบการดำเนินงานและทดสอบระบบสร้างและการจัดการฟอร์ม ASP.NET FORMS GENERATER VERSION 2 ในส่วนนี้จะกล่าวถึงการสรุปตลอดจนปัญหาและอุปสรรคของการทำโครงงานนี้ รวมทั้งข้อเสนอแนะในการนำเอาโครงงานไปพัฒนาต่อเพื่อใช้เป็นแนวทางสำหรับผู้ที่สนใจ

5.1 สรุปผลที่ได้จากโครงงาน

จากโครงงานที่ได้จัดทำขึ้น เมื่อผู้ใช้มีฐานข้อมูลที่ถูกต้อง ระบบการสร้างและการจัดการฟอร์ม ASP.NET FORMS GENERATER VERSION 2 สามารถทำการสร้างและจัดการฟอร์มได้ดังนี้ สามารถสร้างเว็บไซต์ ASP.NET FORMS GENERATER VERSION 2 ติดต่อกับฐานข้อมูล Microsoft SQL Server 2005 – 2008 ได้ แต่ไม่สามารถสร้างฟอร์มจากฐานข้อมูลตัวอื่นได้เช่น Access, MySQL, Oracle ประกอบด้วยหน้า Insert, Update, Delete และ View ได้ สามารถค้นหาข้อมูลในหน้า View ได้ สามารถนำเอาไฟล์โปรเจคที่ได้ทำการบันทึกไว้มาให้โปรแกรมอ่านไฟล์ขึ้นมาพร้อมกับตรวจสอบว่าในฐานข้อมูลที่ใช้มีการ (Update Database Schema) หรือไม่ ถ้ามีก็จะแสดงโครงสร้างฐานข้อมูลอันใหม่ขึ้นมา และยังคงรูปแบบของรูปแบบฟอร์มเดิมไว้ ดังนั้นจึงสามารถแก้ไข้รูปแบบฟอร์มต่อจากเดิมได้ โดยไม่ต้องเริ่มต้นใหม่ นอกจากนั้น ผู้ใช้ยังสามารถจัดรูปแบบฟอร์มข้อมูลได้โดยจะจัดตำแหน่งของ ข้อความและคอนโทรล ที่จะแสดงตอน การเพิ่มข้อมูล และการแก้ไขข้อมูล และสามารถเลือกธีมโดยมีธีม อย่างน้อย 5 ธีม และสามารถเพิ่ม Header, Footer, Background ได้ และได้นำเอาฟังก์ชันของ AJAX เช่น SlideShow, Always VisibleControl, ColorPicker, Animation มาใส่ใน ธีม เพื่อความสวยงามและอำนวยความสะดวกแก่ผู้ใช้งาน ซึ่งเว็บไซต์ที่สร้างขึ้นมาสามารถใช้งานได้จริง

5.2 ข้อเสนอแนะในการพัฒนาโครงงาน

การสร้างและจัดการฟอร์มควรจะมีความสามารถในการสร้างเว็บไซต์ได้มากกว่านี้คือ

5.2.1 สร้างเว็บไซต์จากฐานข้อมูล MSSQL Server, Microsoft Access, Oracle และ MySQL

5.2.2 เลือกสร้างเว็บไซต์ได้ทั้ง ASP, PHP และ JAVA

5.2.3 สร้าง Report จากฐานข้อมูลได้

5.2.4 สร้าง Form จาก Store Procedure หรือ Function ของ Microsoft SQL Server ได้

5.2.5 มีการ Validate ข้อมูลจากฐานข้อมูล เช่น Compute Column เป็นต้น

5.2 อุปสรรคในการทำงาน

ในการสร้างระบบการสร้างและการจัดการฟอร์ม ASP.NET FORMS GENERATER VERSION 2 นั้นแม้จะประสบความสำเร็จในการพัฒนาแต่ก็มีอุปสรรคที่เกิดจากการพัฒนาเนื่องจากระบบมีความซับซ้อนของข้อมูลและการซับซ้อนของโครงสร้างทำให้สิ่งหนึ่งที่ขาดไม่ได้ก็คือการศึกษาวิธีการต่างๆที่ช่วยทำให้ระบบสามารถทำงานได้ซึ่งที่ได้ทำการศึกษาไปก็มีทั้งความรู้ใหม่และความรู้ที่มีอยู่แล้วทำให้อาจจะเสียเวลาไปบางส่วนเพื่อทำการศึกษาหาข้อมูลและที่สำคัญก็คือการขาดประสบการณ์ในการแก้ไขปัญหาต่างๆ ในการสร้างระบบการสร้างและการจัดการฟอร์ม ASP.NET FORMS GENERATER VERSION 2 ทำให้ขาดความต่อเนื่องในการพัฒนา

บรรณานุกรม

[1] ประกาสิทธิ์ ลาศรี, พิเชษฐ สีสม. 2553. “ASP.NET Forms Generator (Version1).” ปริญญานิพนธ์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีราชมงคล ธัญบุรี

[2] พิทยาโพธิ์ชะคุ้ม, ศิรินภา ระวิพานิช, ศุภรางศุ์ จินะใจ. 2551. “การสร้างและการจัดการฟอร์ม (FORMS GENERATOR).” ปริญญานิพนธ์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์มหาวิทยาลัยเทสโนโลยีราชมงคล ธัญบุรี

[3] พงษ์พันธ์ ศิริวิลัย. 2547. SQL Server 2005 ฉบับสมบูรณ์. กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

[4] พงษ์พันธ์ ศิริวิลัย. 2552. SQL Server 2008 ฉบับสมบูรณ์. กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

[5] ศุภชัย สมพานิช. 2551. Database Programming ด้วย VB2008 & VC#2008. นนทบุรี : ไอดีซีฯ.

[6] ศุภชัย สมพานิช. 2546. คู่มือการเขียนโปรแกรม Visual C# .NET ฉบับโปรแกรมเมอร์. นนทบุรี : ไอดีซีฯ.

[7] ทวีชัย หงษ์สุมาลย์. 2545. อินไซท์ ASP และ ASP.NET ฉบับสมบูรณ์. กรุงเทพฯ : โปรวิชั่น.

[8] กิตติ ภักดีวัฒนะกุล, จำลอง ครูอุตสาหะ. 2543. ASP ฉบับโปรแกรมเมอร์. กรุงเทพฯ : บริษัท เคทีพี คอมพ์ แอนด์ คอนซัลท์ จำกัด.

[9] ศุภชัย สมพานิช. 2552. เริ่มต้นอย่างมืออาชีพ ASP NET 3.5. นนทบุรี : ไอดีซี ฯ.

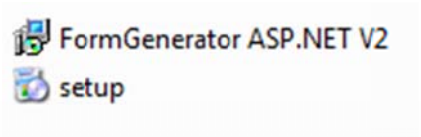
[10] ผู้แปล ศิรส สุภาวิตา. 2551. พัฒนาเว็บแอปพลิเคชันด้วย AJAX (ภาคปฏิบัติ). กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

ภาคผนวก ก.

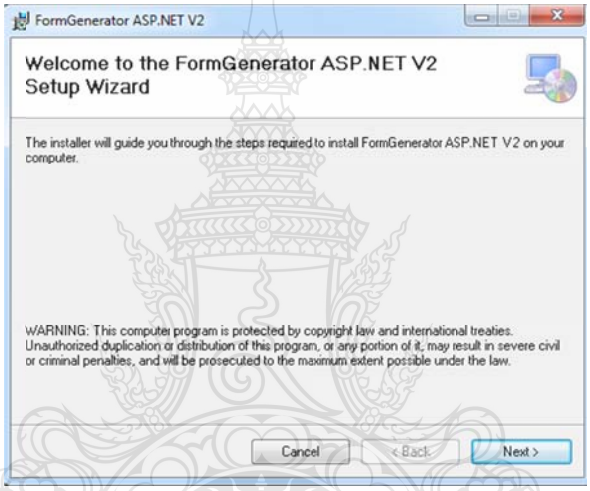
วิธีการใช้งาน

1. ขั้นตอนการติดตั้งโปรแกรม Form Generator ASP.NET V2

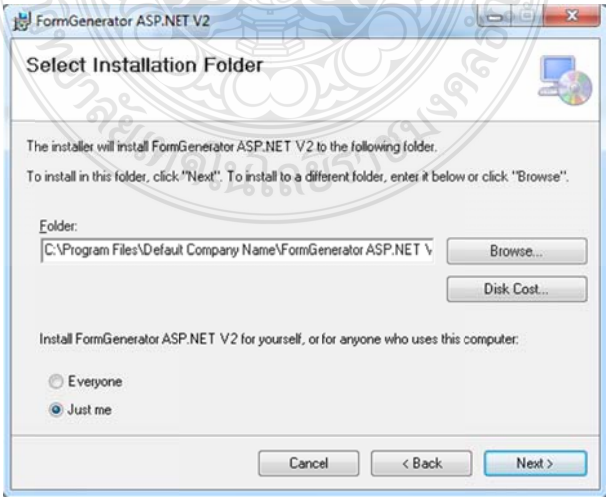
1) ดับเบิ้ลคลิกที่ไฟล์ชื่อว่า FormGenerator ASP.NET V2.msi



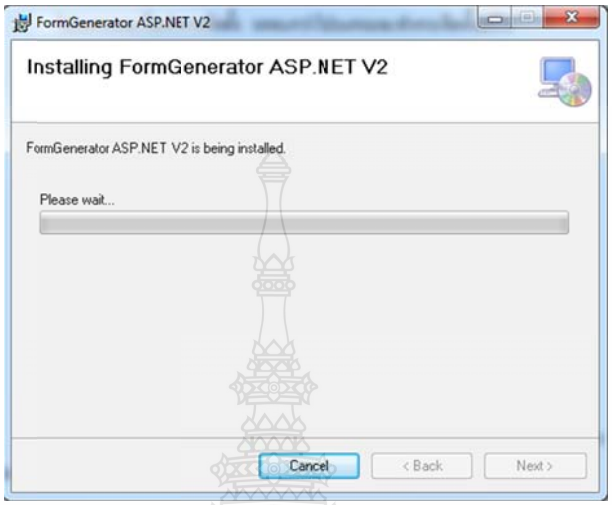
โปรแกรมจะทำการแสดงหน้าจอสำหรับติดตั้งโปรแกรมขึ้นมา



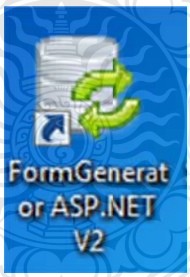
2) ทำการคลิกที่ปุ่ม Next และจะแสดงหน้าต่างให้เลือก ไดเร็กทรอรี่ที่ต้องการติดตั้งไฟล์



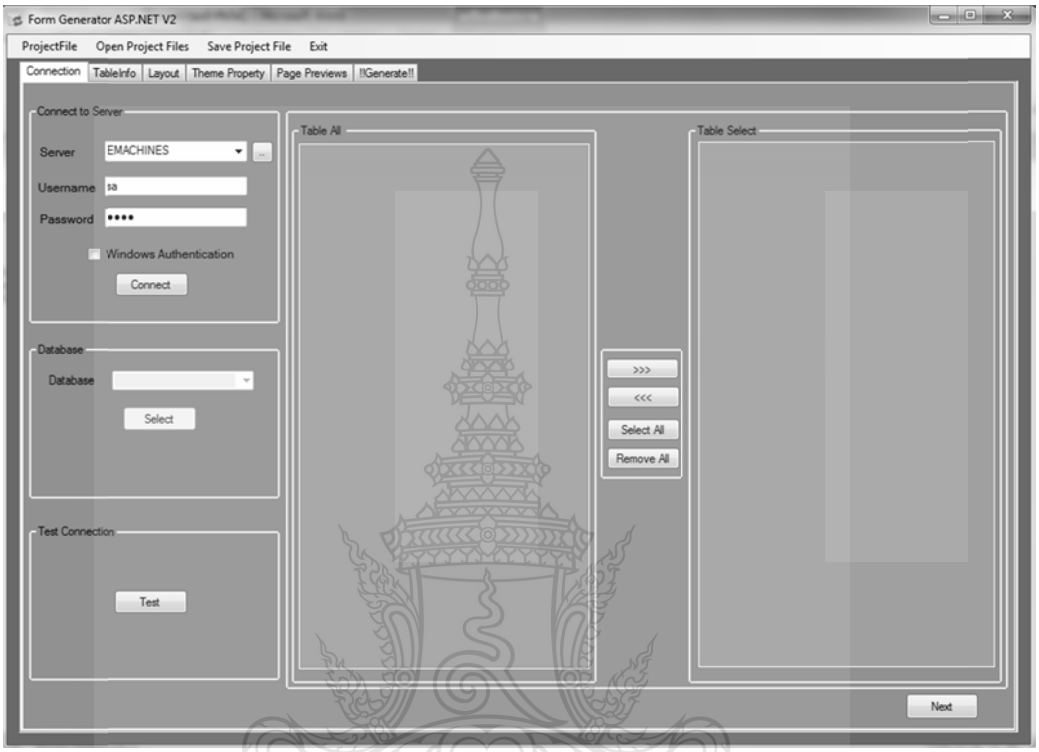
3) คลิกที่ปุ่ม Browse เพื่อเลือกไดเร็กทรอรี่ที่ต้องการติดตั้ง เมื่อเลือกเสร็จแล้วคลิกที่ปุ่ม Next โปรแกรมจะเริ่มทำการติดตั้ง รอจนกว่าโปรแกรมจะทำการติดตั้งเสร็จ



หลังจากการติดตั้งเสร็จสิ้น โปรแกรมจะสร้าง Short Cut ของโปรแกรมมายังหน้าจอ Desktop



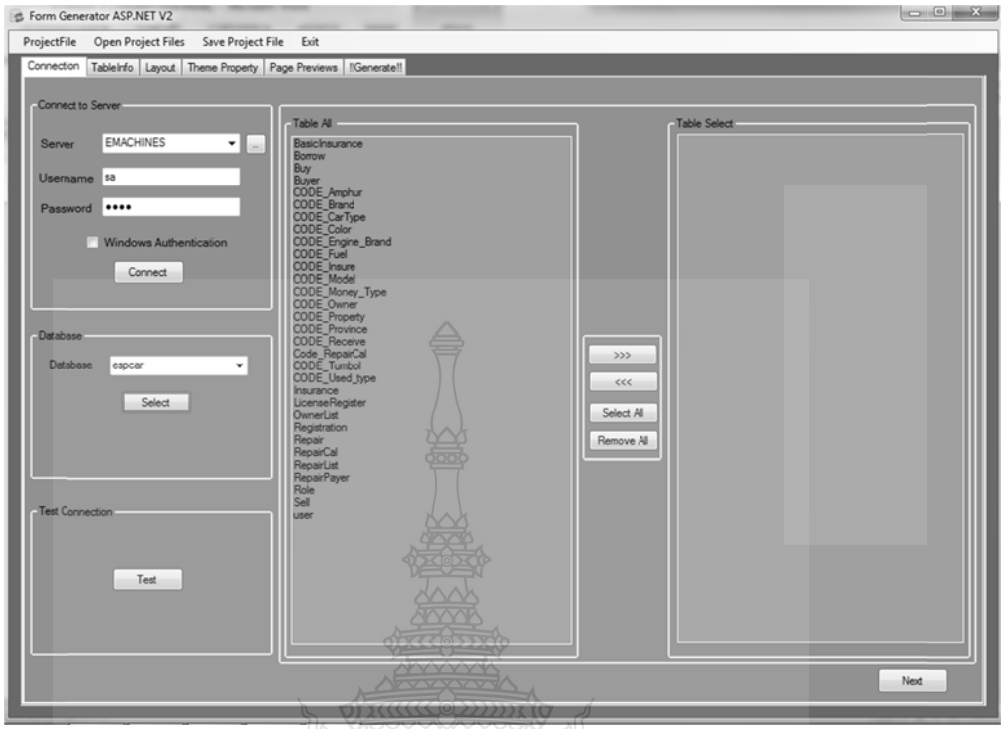
4) ทำการทดสอบไฟล์โดยการดับเบิ้ลคลิกที่ Short Cut บนหน้าจอแล้วจะทำการเข้าสู่โปรแกรม



2. คู่มือการใช้งาน การสร้างและจัดการฟอร์ม ASP.NET V2

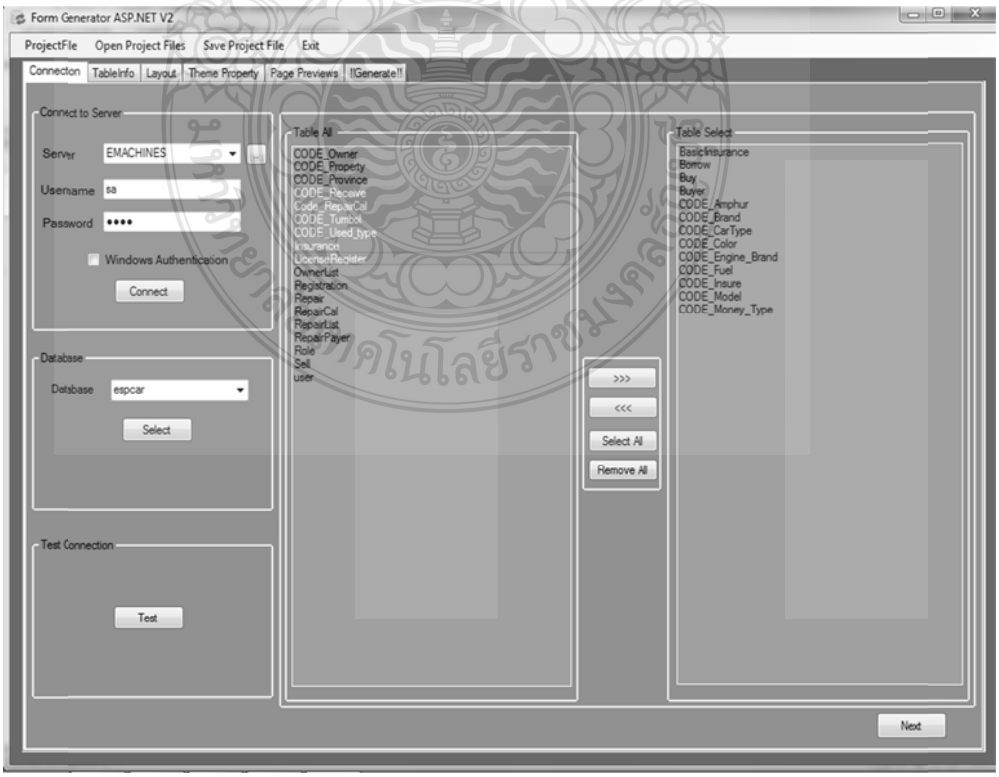
1) เมื่อผู้ใช้งานติดตั้งโปรแกรมเสร็จเรียบร้อยคลิกที่ Short Cut ที่หน้าจอแล้วระบบรันโปรแกรม

2) เลือก Authentication Mode ระหว่าง SQL Authentication กับ Windows Authentication Mode ในกรณีที่ผู้ใช้งานเลือก SQL Authentication ผู้ใช้งานจะต้องทำการกรอก Username และ Password ในการเชื่อมต่อ

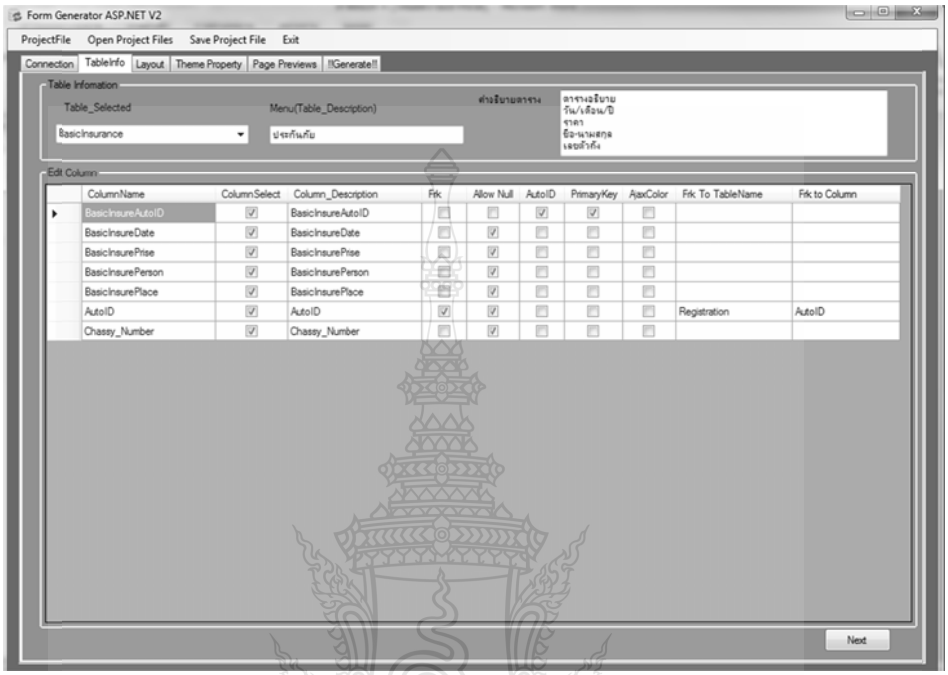


3) เมื่อเชื่อมต่อสำเร็จก็ทำการเลือกฐานข้อมูลที่จะทำการสร้างฟอร์ม

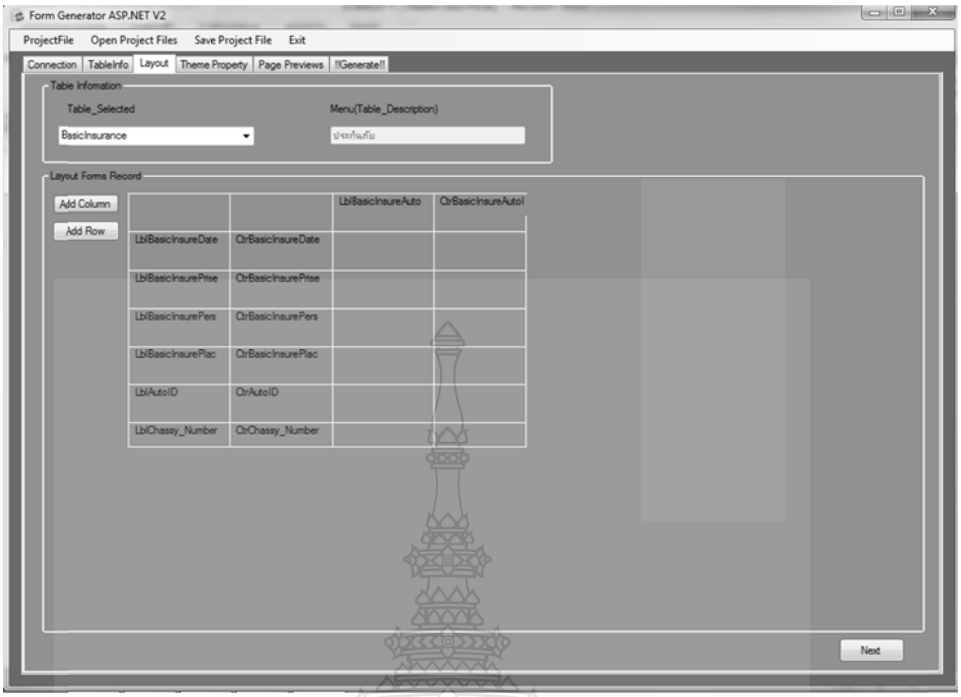
4) เลือกตารางที่ต้องการสร้างฟอร์ม โดยเลือกกี่ตารางก็ได้ แล้วกดปุ่ม Next



5) เลือกคอลัมน์ในการสร้างฟอร์ม ผู้ใช้อาจแก้ไข Description เพื่อเป็นหัวข้อที่แสดงในเว็บไซต์ และใส่ข้อความอธิบายรายละเอียดตาราง แล้วกดปุ่ม Next

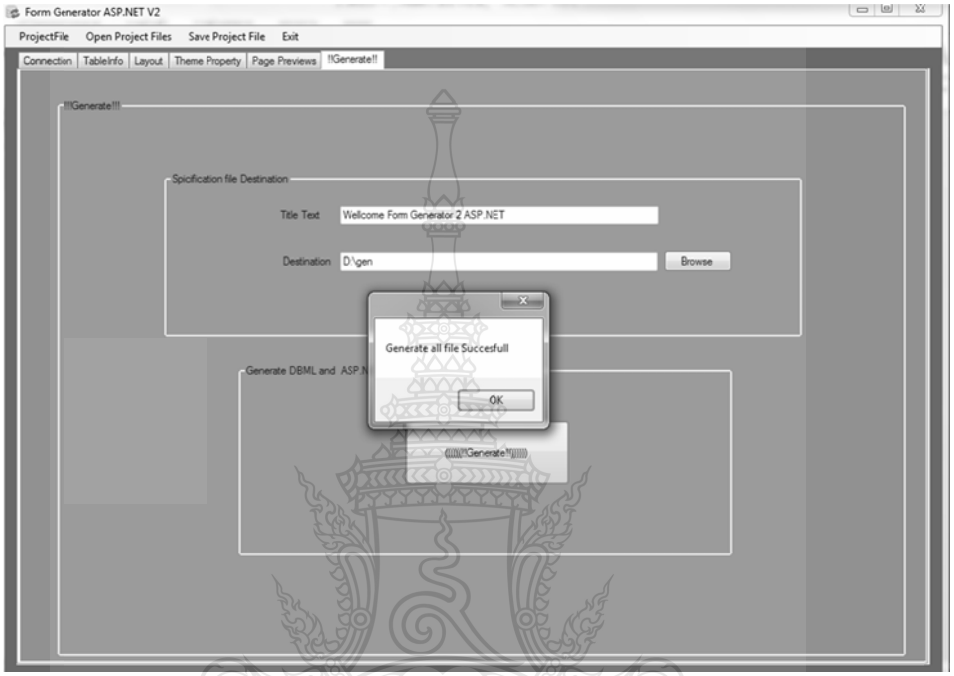


6) เลือกการจัดรูปแบบฟอร์มข้อมูล โดยจัดตำแหน่งข้อความและคอนโทรลเพื่อแสดงในส่วนการเพิ่มข้อมูล และการแก้ไขข้อมูลในเว็บไซต์ หากไม่จัดรูปแบบฟอร์มข้อมูล กดปุ่ม Next

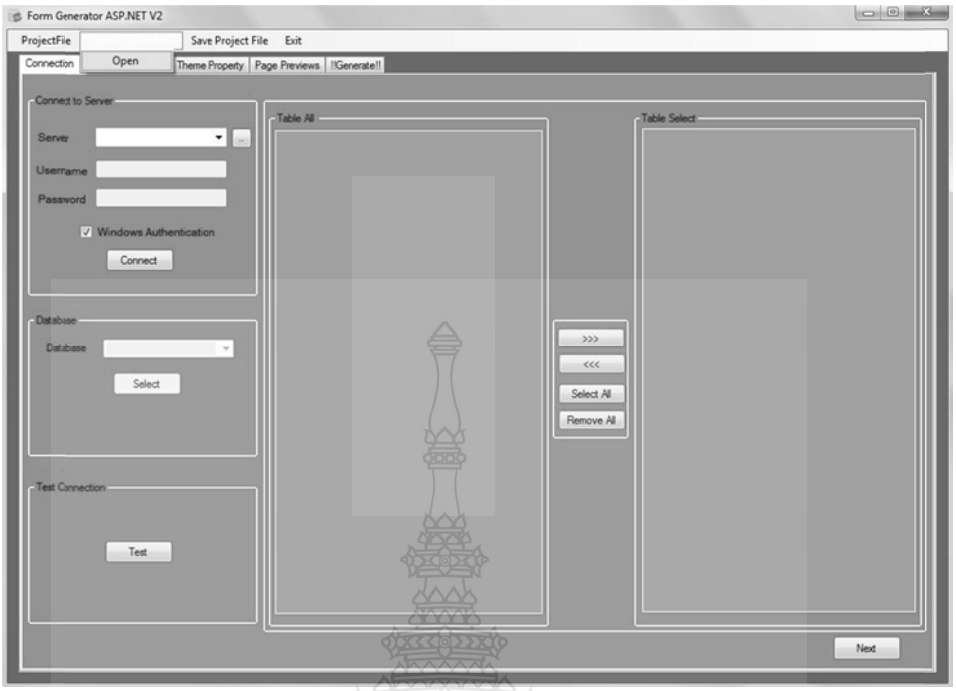


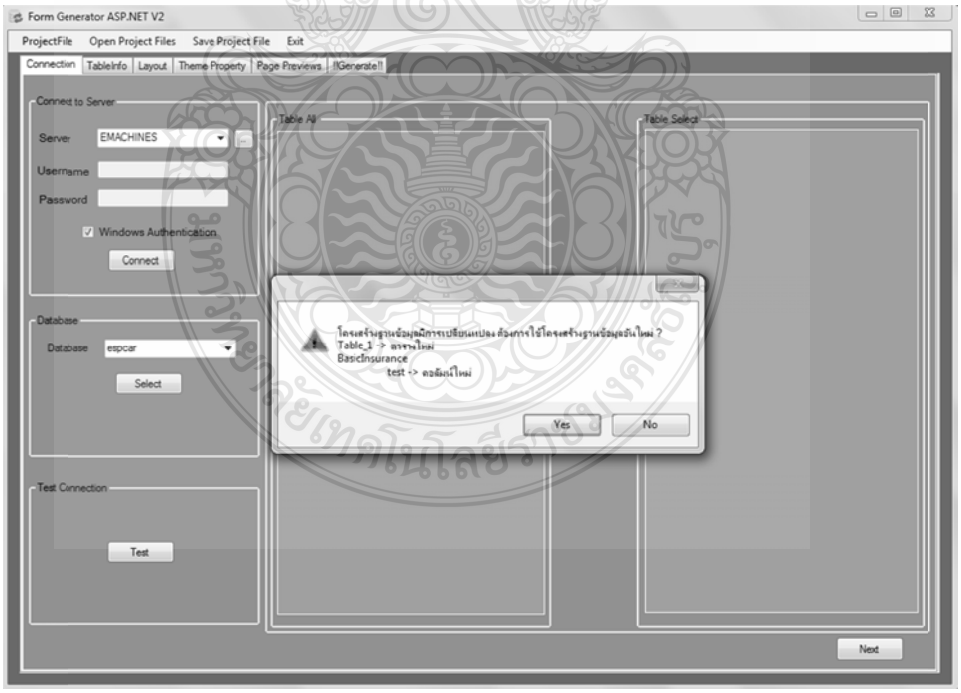
7) เลือก Header และ Footer ระหว่างใช้เป็นรูปภาพ หรือ ข้อความ ต่อมาทำการเลือกสี หรือรูปภาพพื้นหลังของเว็บไซต์ และเลือกใส่รูปทำสไลด์โชว์ แล้วเลือก กริดวิว ในการแสดงข้อมูล กดปุ่ม Next ระบบก็จะแสดงตัวอย่างเว็บไซต์ก่อนทำการสร้างไฟล์

8) เพิ่มข้อความ Title Text แล้วเลือกไดเร็กทอรี ที่ต้องการเก็บไฟล์ไว้
9) กดปุ่ม Generate ระบบก็จะรายงานผลการทำงาน

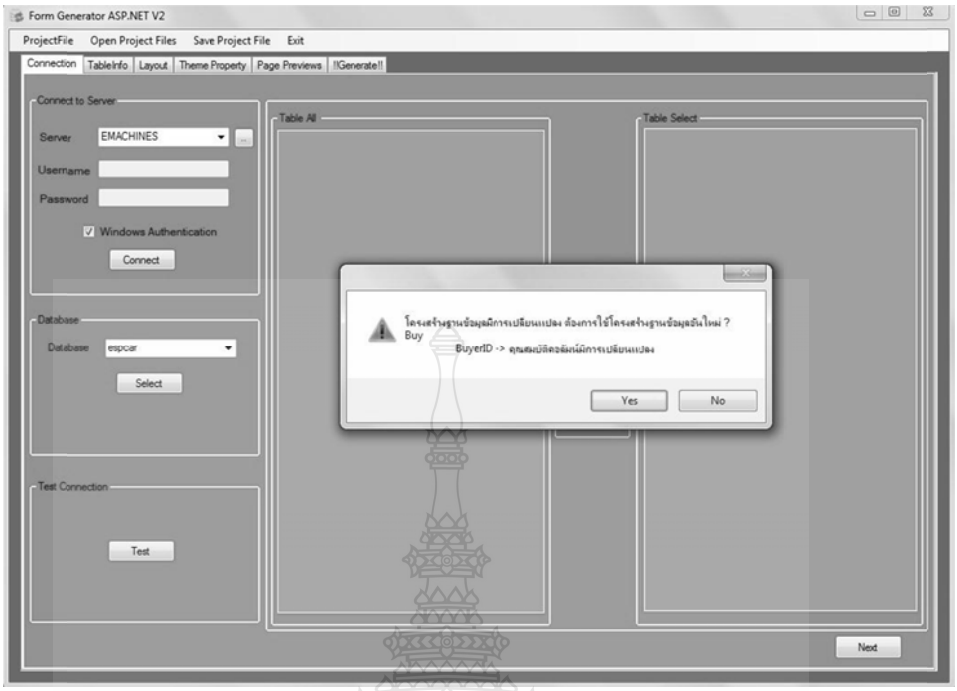


10) เมื่อผู้ใช้ทำการเปิดไฟล์งานเดิมขึ้นมา โปรแกรมจะทำการตรวจสอบโครงสร้างฐานข้อมูลว่าโครงสร้างฐานข้อมูลมีการเปลี่ยนแปลง ถ้ามีก็จะมีข้อความแจ้งเตือนขึ้นมา

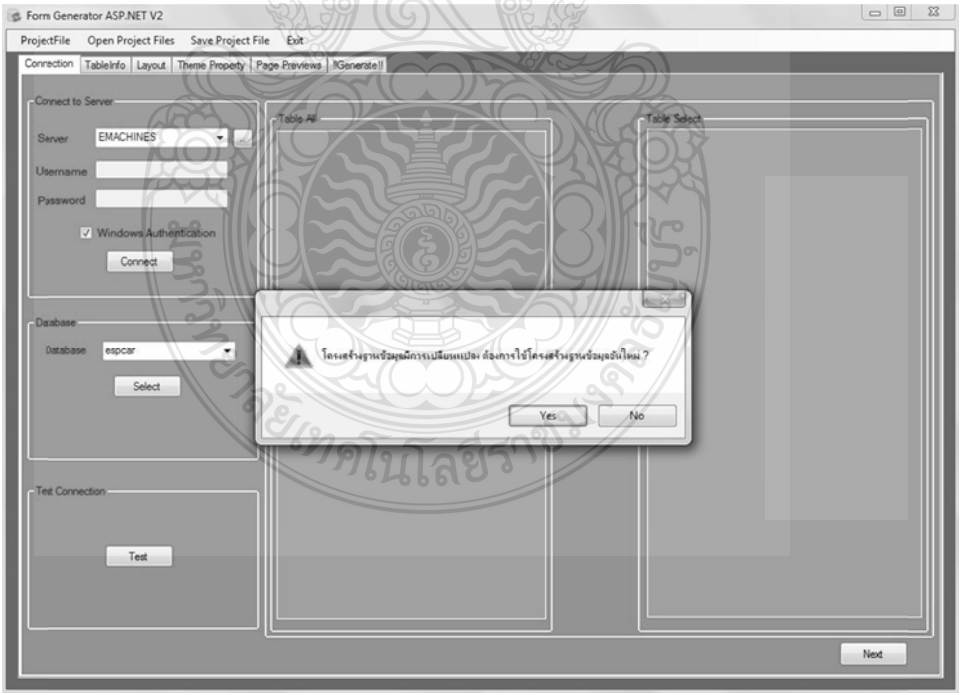




แสดงข้อความแจ้งเตือน เมื่อมีตารางและคอลัมน์ใหม่เพิ่มขึ้นมา

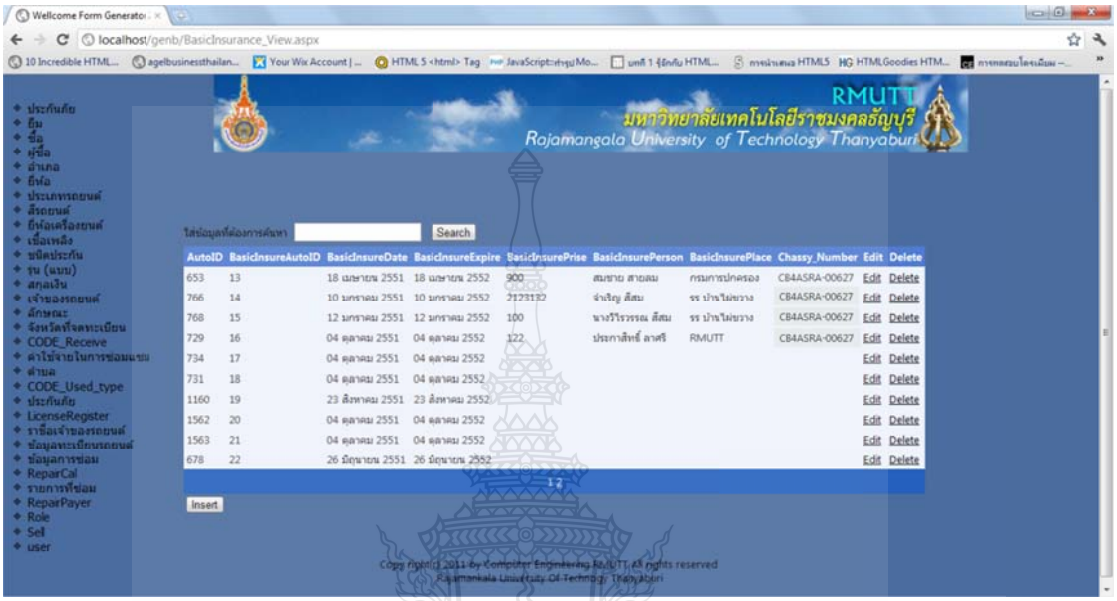


แสดงข้อความแจ้งเตือน เมื่อมีคุณสมบัติของคอลัมน์มีการเปลี่ยนแปลง

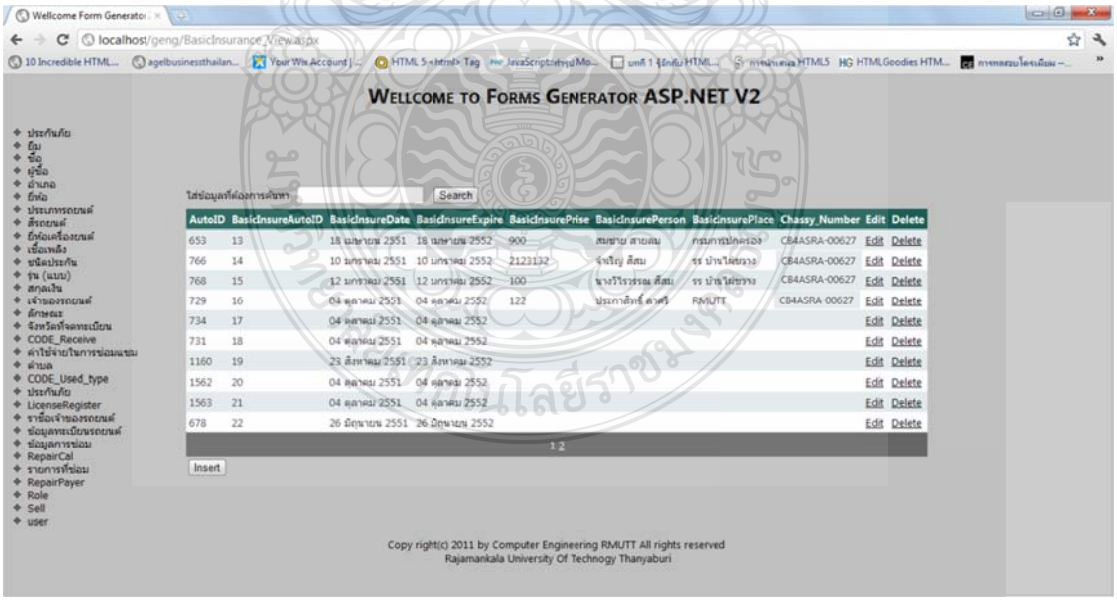


แสดงข้อความแจ้งเตือน เมื่อมีตารางหรือคอลัมน์ถูกลบออกจากฐานข้อมูล

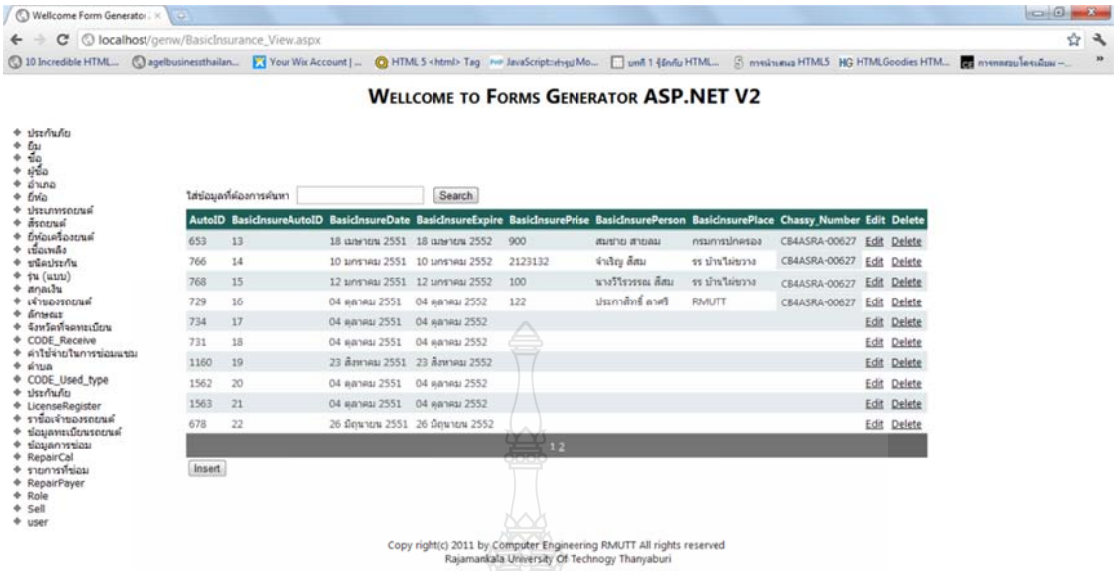
ในการทดสอบการเลือกธีมทำการทดสอบโดยการเลือกทุกตารางและทุกคอลัมน์ในฐานข้อมูลแล้วทำการ Generate ไฟล์ออกมาแล้วรันบน IIS Server แสดงผลดังรูป



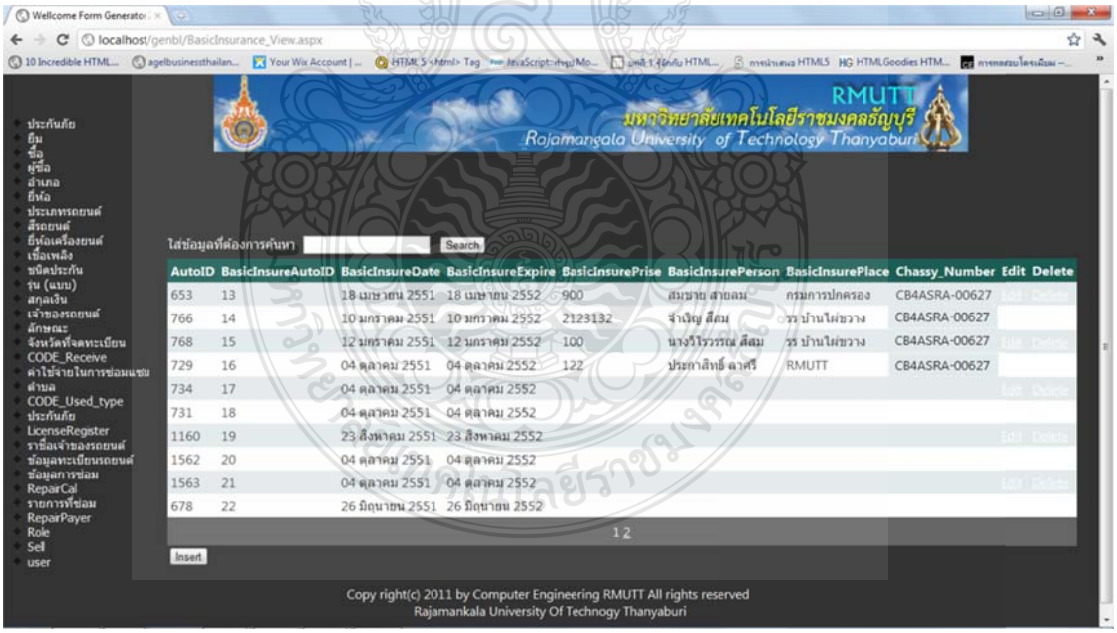
เมื่อผู้ใช้งานเลือกพื้นหลังสีน้ำเงิน และใช้รูปภาพเป็น Header



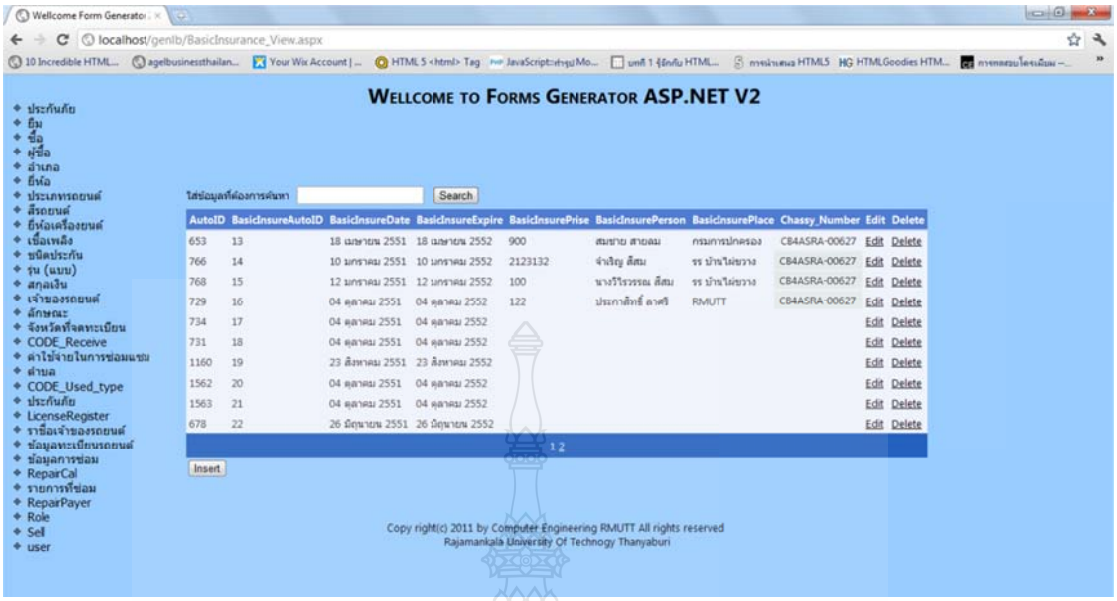
เมื่อผู้ใช้งานเลือกพื้นหลังสีเทา และใช้ตัวอักษรเป็น Header



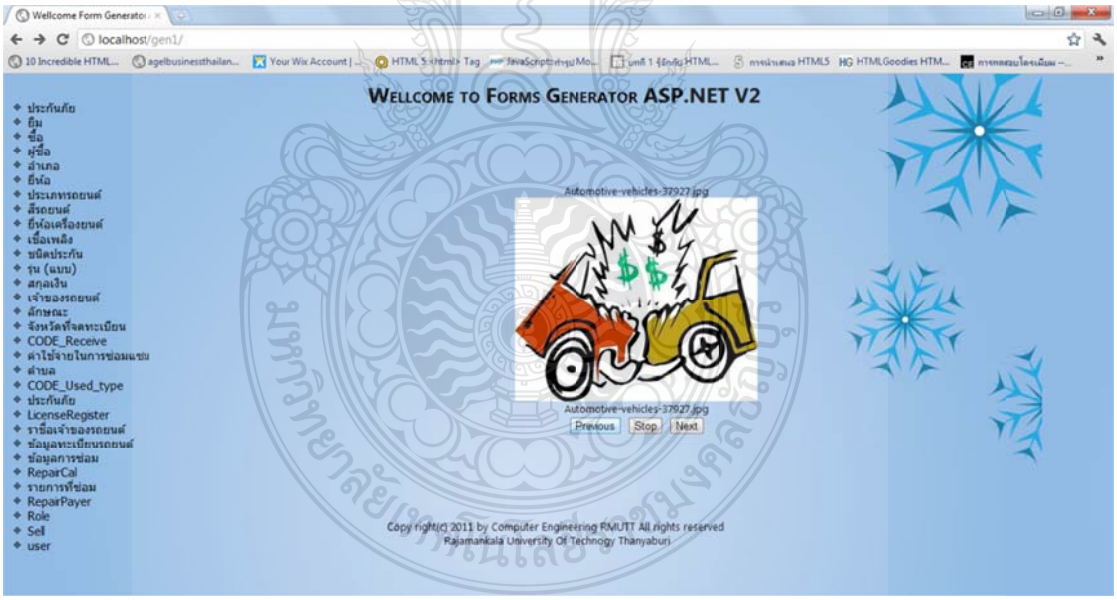
เมื่อผู้ใช้งานเลือกพื้นหลังสีขาว และใช้ตัวอักษรเป็น Header



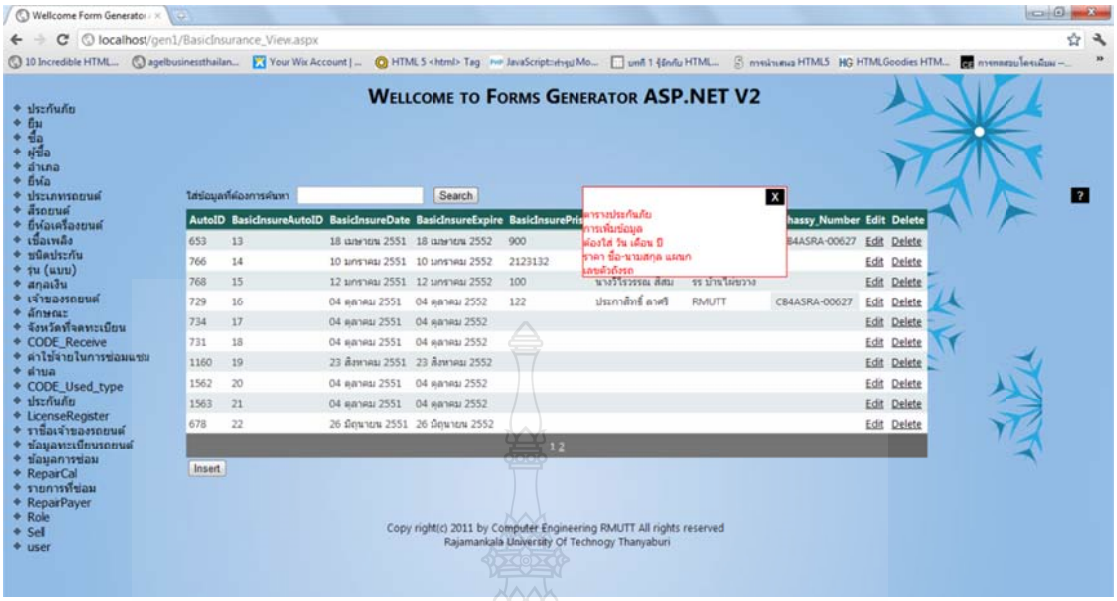
เมื่อผู้ใช้งานเลือกพื้นหลังสีดำ และใช้รูปภาพเป็น Header



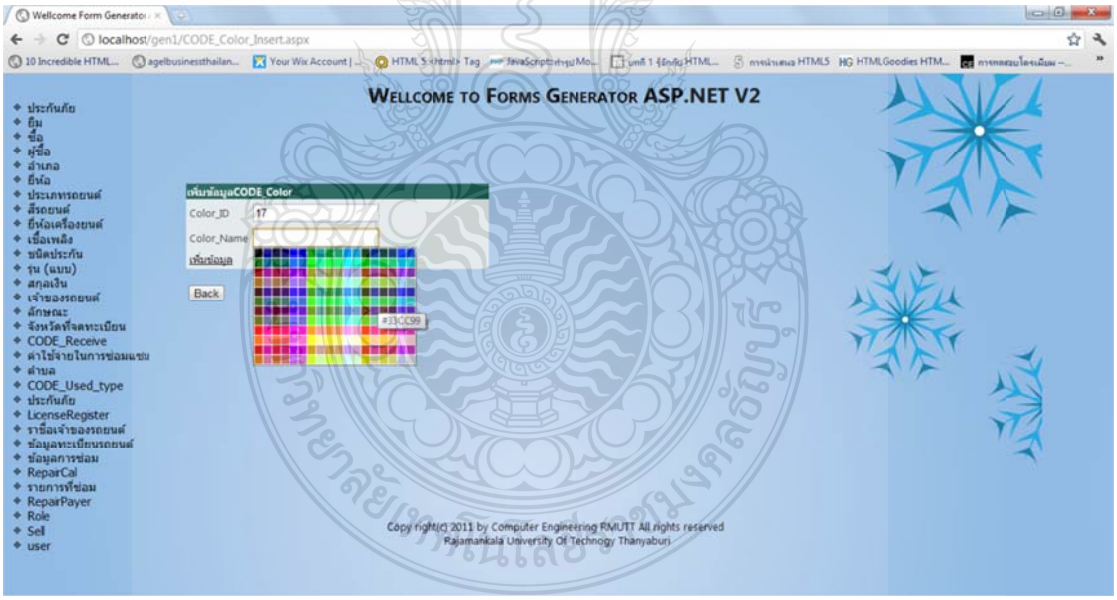
เมื่อผู้ใช้งานเลือกพื้นหลังสีฟ้าอ่อน และใช้ตัวอักษรเป็น Header



เมื่อผู้ใช้งานเลือกรูปเป็นพื้นหลัง เลือกรูปทำสไลด์โชว์ และใช้ตัวอักษรเป็น Header



เมื่อผู้ใช้งานคลิกที่ปุ่ม ? จะแสดงข้อความอธิบายตาราง



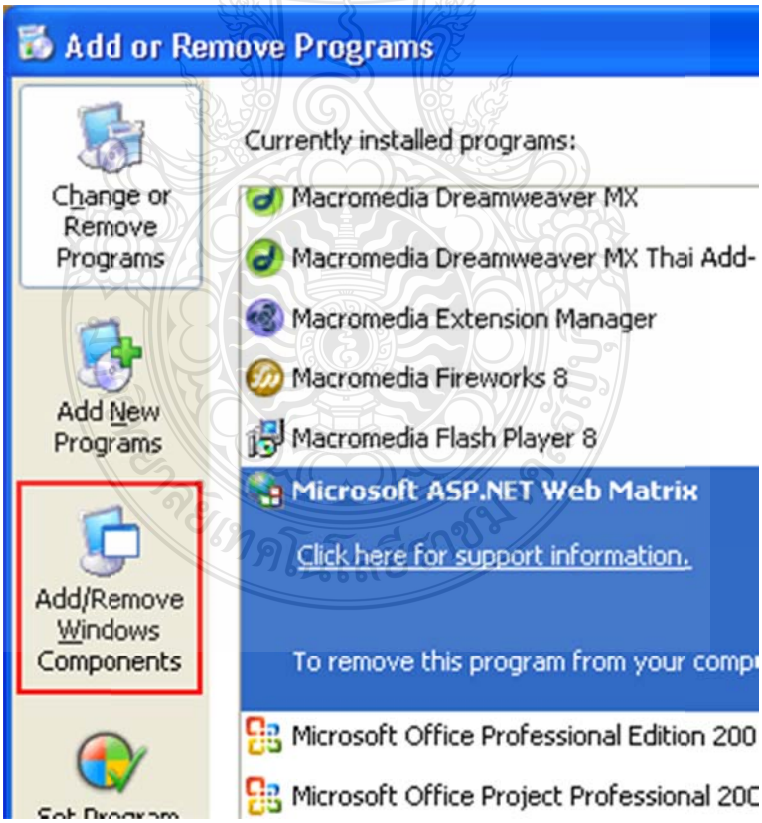
เมื่อผู้ใช้งานเพิ่มข้อมูลเลือกสี จะโชว์กล่องสีมาให้เลือกใช้งาน

3. ขั้นตอนการติดตั้ง IIS (Internet Information Services)

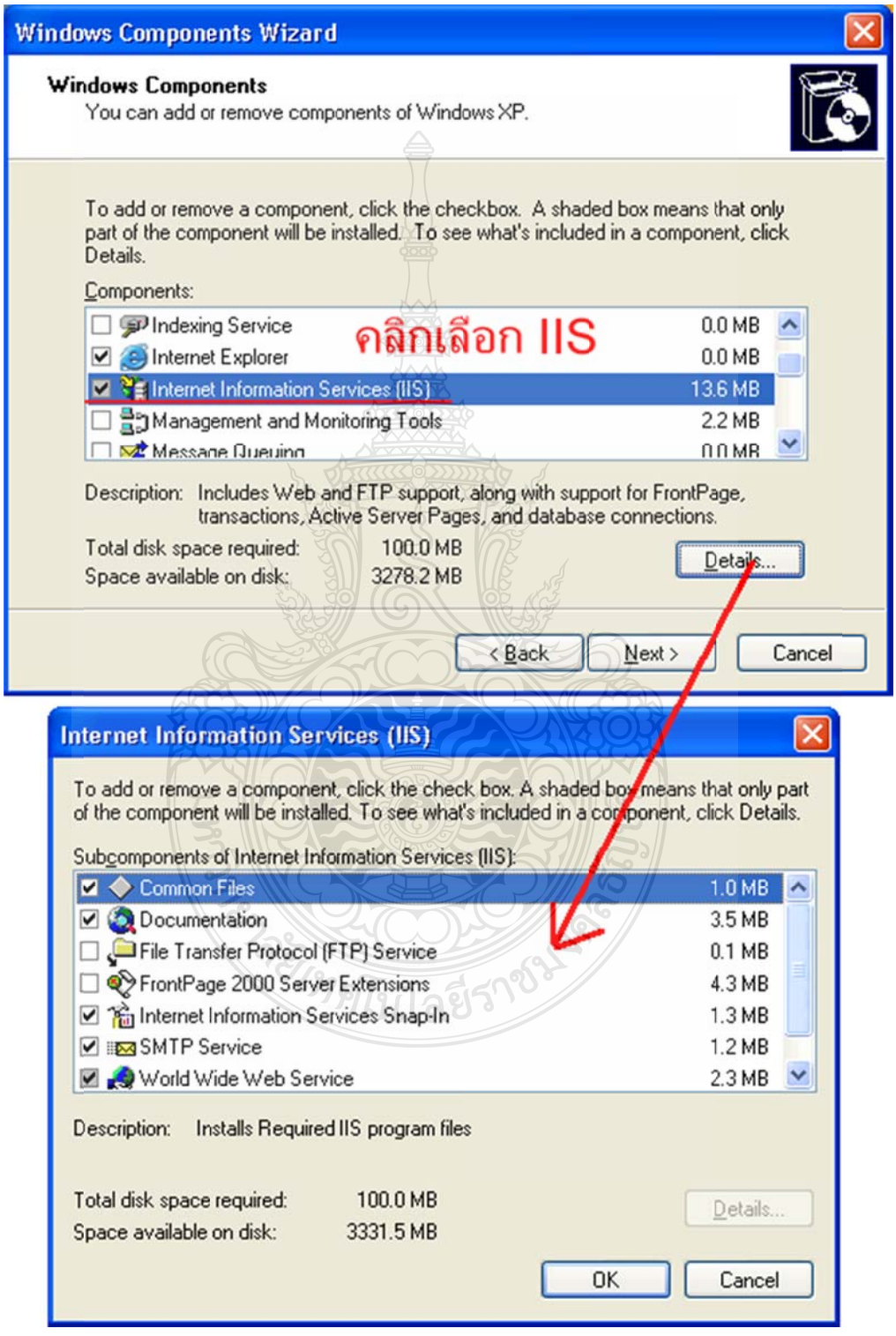
IIS (Internet Information Services) เป็นโปรแกรมที่ใช้สำหรับทำเว็บ Server ระบบจะต้องทำการติดตั้ง .NET Framework ก่อนจึงจะใช้งานได้ ความต้องการของระบบในการติดตั้ง IIS

- Microsoft Windows 2000, Windows 2003, Windows XP หรือสูงกว่า
- CPU Pentium III หรือมากกว่า
- ความเร็วของ CPU ตั้งแต่ 700 MHz ขึ้นไป
- แรม 512 MB หรือมากกว่า
- ความละเอียดหน้าจอ 800 x 600 ขึ้นไป
- พื้นที่ที่ใช้ในการติดตั้ง 100 MB หรือมากกว่า

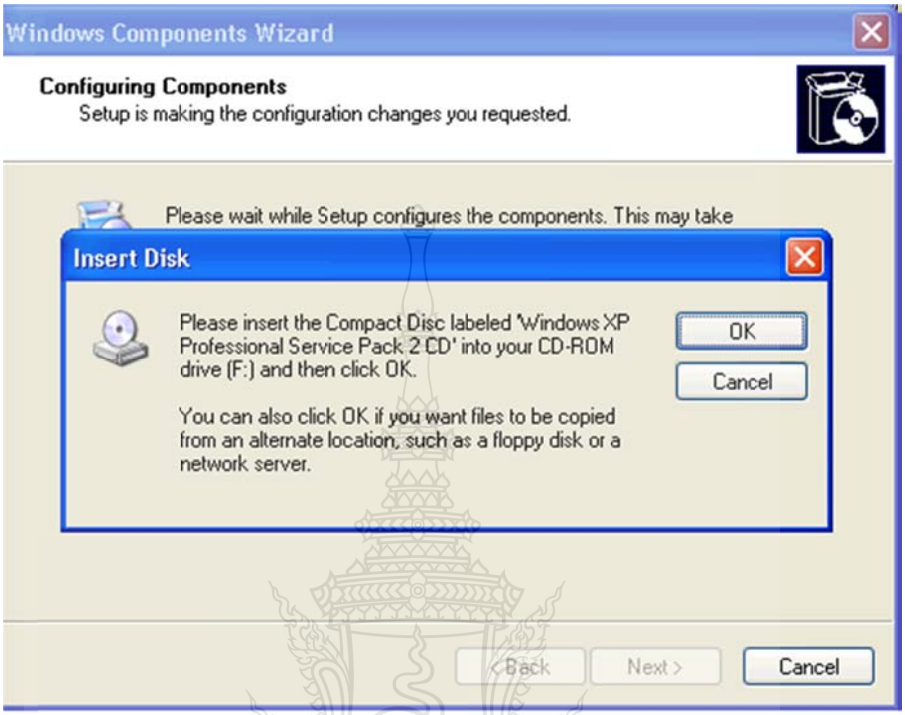
1) คลิกที่ Start -> Settings -> Control Panel แล้ว Double Click ที่ Add or Remove Programs แล้วคลิกที่ Add/Remove Windows Components



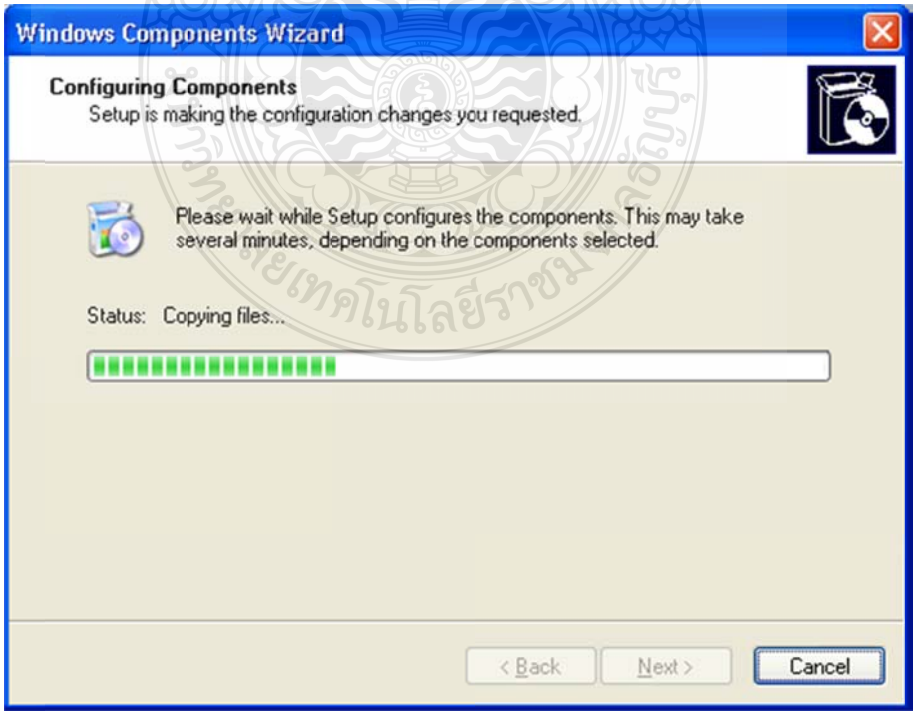
2) คลิกเลือกที่ Internet Information Services (IIS) กรณีต้องการดูรายละเอียดโปรแกรมที่ติดตั้งเพิ่มเติมให้คลิกเลือกที่ Details เสร็จแล้วกดปุ่ม Next



3) กรณีระบบถามหาแผ่นติดตั้ง Windows XP ให้แผ่นติดตั้ง Windows แล้วคลิกปุ่ม OK



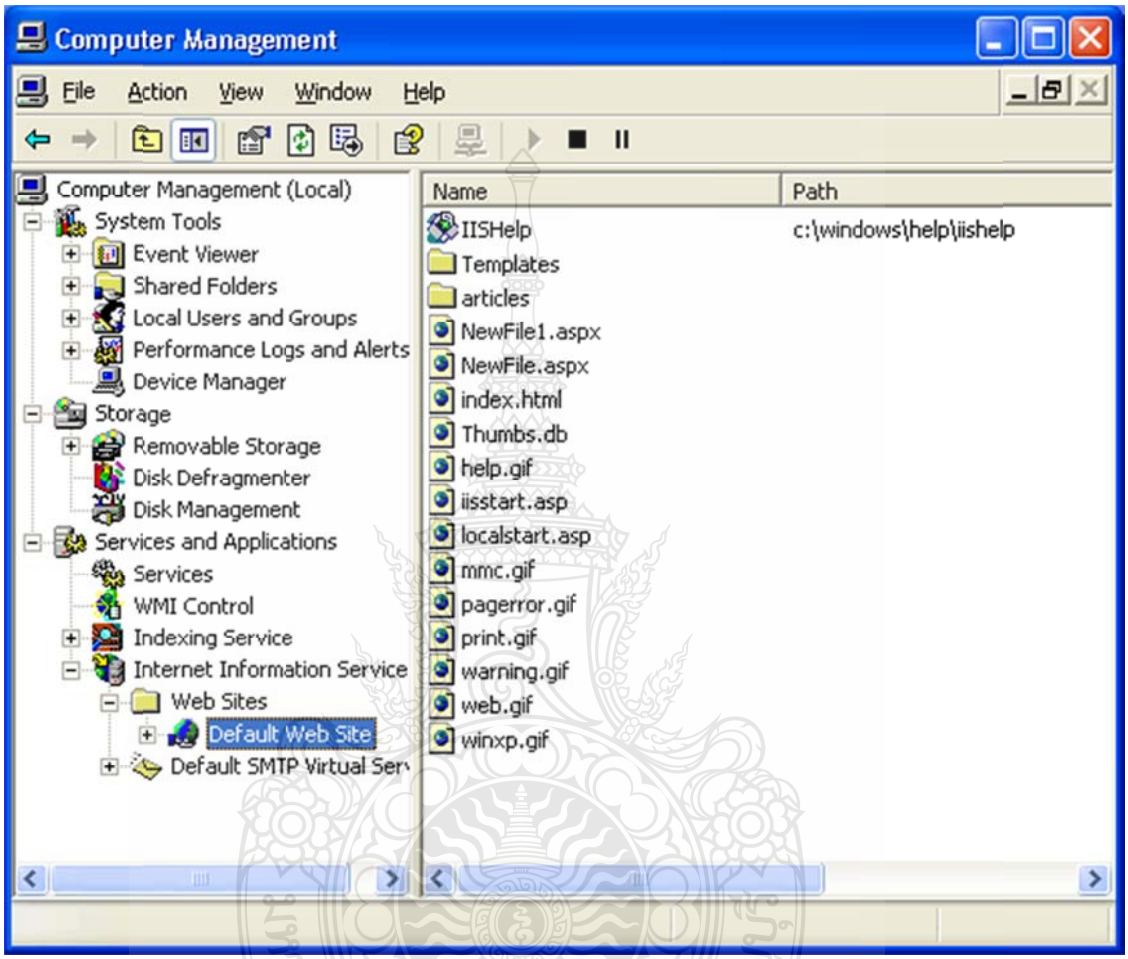
4) ระบบจะทำการติดตั้ง IIS ดังรูป



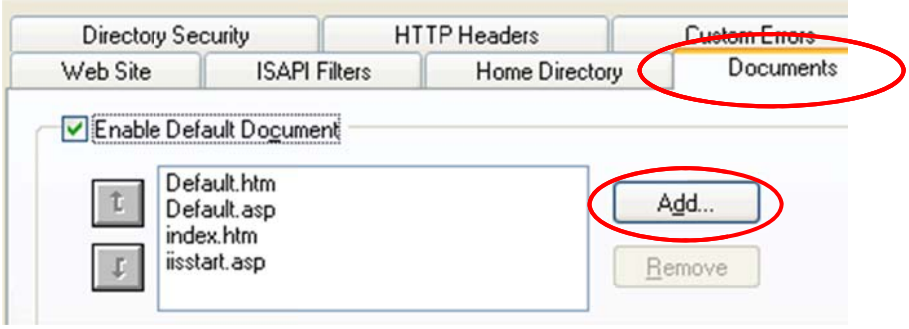
5) ระบบติดตั้งคลิกปุ่ม Finish

6) การติดตั้งไฟล์เว็บไซต์การสร้างและการจัดการฟอร์ม โดยคลิกขวา Icon ของ My Computer แล้วคลิกเลือกที่ Manage

7) คลิกที่ Services and Applications > Internet Information Services > Web Sites > Default Web Site



8) ทำการเพิ่มไฟล์ Home Page หรือไฟล์หน้าแรกที่ต้องการเรียกใช้งาน โดยการคลิกขวาที่ Default Web Site เลือกที่ Properties แล้วเลือกไปที่แท็บ Document แล้วกดปุ่ม Add เพื่อเพิ่มไฟล์เว็บไซต์การสร้างและการจัดการฟอร์ม



ภาคผนวก ข.

Source Code อยู่ในซีดีรอม

ประวัติผู้จัดทำปริญญานิพนธ์

ประวัติผู้จัดทำปริญญานิพนธ์

ชื่อ	นายวิสิฎฐ์ เพชรหนู รหัส 115140462014-8
ภาควิชา	ภาควิชาวิศวกรรมคอมพิวเตอร์
วัน-เดือน-ปี เกิด	วันที่ 16 มีนาคม 2529
สถานที่เกิด	จังหวัดตรัง
ที่อยู่	54 หมู่ 7 ต. โคกหล่อ อ.เมือง จ.ตรัง 92000
ประวัติการศึกษา	ประกาศนียบัตรวิชาชีพ (ปวช.) วิทยาลัยเทคนิคตรัง 2549
	ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.) วิทยาลัยเทคนิคหาดใหญ่ 2551

ประวัติผู้จัดทำปริญญานิพนธ์

ชื่อ	นายเจตน์สฤษฎ์ พลเยี่ยม รหัส 115140462028-8
ภาควิชา	ภาควิชาวิศวกรรมคอมพิวเตอร์
วัน-เดือน-ปี เกิด	วันที่ 5 กันยายน 2529
สถานที่เกิด	จังหวัดร้อยเอ็ด
ที่อยู่	204 หมู่ 8 ต.หนองพอก อ.หนองพอก จ.ร้อยเอ็ด 45210
ประวัติการศึกษา	ประกาศนียบัตรวิชาชีพ (ปวช.) วิทยาลัยการอาชีพโพนทอง 2548
	ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.) เทคโนโลยีสยาม 2550